A Survey on Energy Aware Offloading Strategies in the Compute Continuum

Thimo Wuttge
Vrije Universiteit Amsterdam
t.r.wuttge@student.vu.nl

Matthijs Jansen
Vrije Universiteit Amsterdam
m.s.jansen@vu.nl

Tiziano De Matteis
Vrije Universiteit Amsterdam
t.de.matteis@vu.nl

Abstract

This survey investigates offloading strategies in the compute continuum. The compute continuum provides an infrastructure consisting of the endpoint, edge, and cloud layers. Offloading strategies refer to tactics that allow devices to execute tasks on other devices in this system if their resources are not capable of executing the tasks. Reasons for offloading tasks are usually driven by acquiring more compute power by utilizing more powerful or unused resources. We investigate different tactics on how endpoints can offload tasks to other devices. We focus thereby on energy-efficiency as a driver for more sustainable computing. We present different approaches and highlight how they contribute to energy optimization in the compute continuum. In addition, we highlight the advantages these tactics entail and present research gaps and future research possibilities.

1 Introduction

The compute continuum has emerged as a prominent infrastructure model, that offers compute and storage resources while reducing latency. It consists of the endpoint layer, including end user devices like computers, phones, or smart devices, the edge layer, and the cloud as visualized in Figure 1. [5] It is able to provide additional resources with low latency by leveraging the small network overhead from edge devices as well as nearly inexhaustible compute and storage resources from cloud services. Therefore, these two additions to the local execution systems grew more and more in popularity. [10] Edge computing, the usage of edge devices as an addition to local resources, is realized by deploying devices closer to endpoint devices. These edge devices come with additional compute and storage resources and can provide these with lower latency. If the deployed edge resources do not fulfill the requirements demanded by endpoint devices, workload-intensive tasks can be offload to the cloud where resource limitations are minimal.

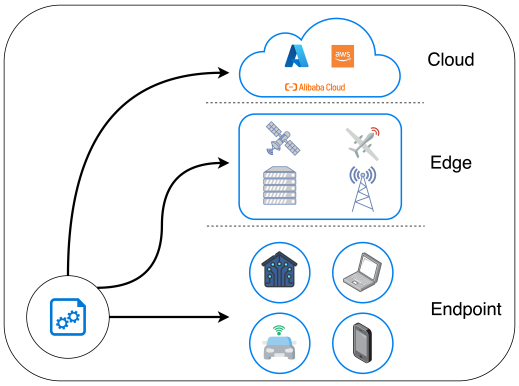

Figure 1: Offloading in the compute continuum environment

By deploying this infrastructure, two very pressing concerns in task execution can be addressed. It allows users to execute tasks that require additional resources by offloading them to a more suitable layer and simultaneously does not compromise execution deadlines by time-intensive offloading due to communication overhead. [27]

The advantages of the compute continuum can be especially noticeable when a variety of task types are issued by the endpoint devices. Many latency aware systems, like hospitals or highly efficient smart factories, deploy numerous endpoint devices, which issue various tasks. Some of these tasks may be especially time constraint, e.g. anomaly detection in patient monitoring, where low latency plays an important role. Other tasks are resource intensive but do not need to be executed within a defined time window, e.g. local database backups. Hereby, the compute continuum can provide an infrastructure capable of handling these cases.

But offloading tasks introduces increased energy consumption through additional networking and

Reference	Year	Individual layers	Offloading approach	Task estimation	Energy efficiency	Individual strategy	Sustainability
Boukerche et al. [9]	2019	✓	✓	–	✓	–	✓
Zabihi et al. [32]	2023	✓	✓	–	–	–	–
Dong et al. [11]	2024	–	✓	✓	✓	–	–
Zhang et al. [33]	2024	✓	✓	✓	–	✓	–
Kanupriya et al. [18]	2024	–	✓	✓	✓	✓	–
Nabi et al. [22]	2025	–	✓	✓	–	–	–
This survey	2025	✓	✓	✓	✓	✓	✓

Table 1: Related surveys and their focus

compute steps. The compute continuum can optimize this additional power consumption by assigning tasks to the most energy efficient layer or by exploiting idle resources to execute tasks. By implementing this collaborative computation approach, the users not only conserve environmentally costly materials, but also prevent unnecessary energy consumption from idle devices, contributing to a more sustainable overall system. [8] This infrastructure can contribute to energy efficiency through collaborative computing, where devices share their workload to achieve a more energy efficient execution and minimize the energy overhead introduced by additional networking and compute steps.

To highlight the advantages of the compute continuum, we investigate the state of offloading approaches in current research. We focus hereby on energy efficient techniques that contribute to a more environment friendly computing infrastructure. We aim to understand how this infrastructure is currently used, and how future research can help to make the compute continuum a more energy efficient platform which supports resource demanding endpoint systems. The continuous growth in the number of connected devices is creating additionally pressure on compute and storage resources, making it critical to investigate how resource utilization can be optimized. [1] Therefore, we contribute by focusing on following research questions:

RQ-1: What is the most suitable layer to offload certain task to and when should we offload these tasks?

RQ-2: Based on which strategy can we decide the most energy-efficient tasks offloading technique?

RQ-3: What are challenges and limitations in the current research of energy efficient task offloading and how can they be addressed in future research?

We provide answers to these research question in the form of following contributions:

C-1: We introduce a comprehensive system model that presents the infrastructure of the compute continuum, the identification of certain task complexity types, and a taxonomy of offloading approaches.

C-2: We outline advantages of each offloading layer and highlight how current offloading approaches exploit these.

C-3: We formulate observations throughout this survey that highlight research gaps and present promising research directions resulting from these observations.

The remainder of this paper is structure as follows: In Section 2 we present related surveys and how we contribute in a different aspect compared to these. In Section 3 we present our system model. Sections 5 and 6 present offloading approaches from the aspect of the target layer and the strategy. And Section 7 highlights current research gaps, challenges in the field and potential directions.

2 Related Work

Task offloading is a prominent research field and multiple previous surveys highlight the different categories of this research area. We distinguish our work by providing a comprehensive overview of the whole offloading landscape with a specific focus on energy efficient approaches. Additionally, we present a taxonomy on target layer specific approaches as well as the task offloading strategy. An overview of our research focus and a evaluation of related surveys is presented in Table 1.

Individual layers
By categorizing offloading approaches based on the target layer we can highlight different strategies specific to this certain offloading target layer. Dong et al. [11] and Zhang et al. [33] present surveys on individual layers, but both do not focus on energy efficiency. Zhang et al. [33] specify their survey on task type evaluation

Compute requirement	Storage requirement	Description
High	Intensive	Computation-intensive tasks that involve a large amount of data, such as training machine learning models.
High	Moderate	A representative compute-heavy task with moderate data needs is running a trained machine learning model.
Low	Intensive	Data-intensive tasks with lower computational needs, such as content delivery or video streaming.
Low	Moderate	Simple, lightweight tasks like sending control instructions to smart devices.

Table 2: Task types and task examples

which is one aspect we also highlight.

Individual strategies
As part of a second characterization, we want to investigate the theoretic strategy behind the proposed offloading approaches. Therefore, we categorize these techniques and present advantages for each strategy. A similar approach is presented by Zhang et al. [33] and Kanupriya et al. [18]. But both do not focus their surveys on energy efficiency and sustainability.

Energy efficiency and Sustainability
Our motivation for this survey is to study offloading approaches which optimize the energy consumption to implement a more sustainable compute system. Boukerche et al. [9] also focus their survey on sustainable offloading but only on the aspect of the offloading layer. They do not include task evaluation and a classification of the proposed strategies.

3 System Model

To provide a comprehensive overview of the landscape of the topic, we present a task type classification, describe the general structure of the compute continuum and introduce different strategies on which the offloading approaches are based on.

Task Types

We adopted a classification presented by Gorton et al. [14] to identify four different task types. We consider compute and storage resources as the fundamental capacity bottlenecks in our model. We differentiate task types by analyzing their requirements for these two resources. An overview of all task types with a representative example is introduced in Table 2. We adopted this classification because of the main advantages introduces by the edge and cloud layer which are the additional compute and storage resources. They built a fundamental contribution to the local device and are the most influencing factors in the investigated offloading approaches.

Compute Continuum Infrastructure

We use the three-layered compute continuum model, as introduced by Al-Dulaimy et al. [5]. It consists of the **cloud**, **edge**, and **endpoint** layers.

In this structure, the endpoints are on the lowest level. Their main purpose is to create data and computation tasks, which can be offloaded. Therefore, they act as the source for the offloading strategies presented in this survey. Additionally, they also provide resources that can be exploited by other endpoint devices.

The next layer includes edge devices. We include hereby all devices between the endpoint layer and cloud layer. These devices provide more compute and storage resources than the endpoint devices. The main characteristic of the edge layer is the reduced latency compared to the cloud layer that results from the local deployment location close to the endpoint devices.

The top layer of our infrastructure is the cloud. It provides the best compute resources and the most storage space. However, offloading to the cloud layer comes with higher latency and can, therefore, increase latency.

The three offloading strategies are presented in Figure 1.

Offloading Strategy

A third aspect we consider is the strategy behind the offloading approaches. We base our categories on a classification used by Raeisi-Varzaneh et al. [26] and distinguish three different categories on which the offloading technique can be based on:

Heuristics and policies encompass simple binary decisions that are taken throughout the offloading approach. These are based on comparisons and include different metrics to evaluate the task complexity.

Optimization algorithms include all algorithm-based approaches that aim to optimize the scheduling and distribution of the tasks.

Machine learning-based offloading approaches combine all offloading strategies that include a learning stage in their strategy. They also aim to optimize the distribution and scheduling of the tasks but have the advantage that they can use information gathered throughout the execution to tune their approach.

4 Methodology

To find a relevant set of literature we applied a combination of multiple research approaches. We got an initial set of publications by using a dataset extracted from multiple publication websites. Additionally, we looked at multiple conferences relevant to the examined subject. Since the extracted database and the conferences do not include the most recent publications, we decided to also explored multiple online platforms to find relevant literature from the last year. In the following sections we will present these three research steps in detail so future surveys can accurately reproduce our results.

4.1 AIP

To retrieve an initial set of literature we use a dataset created by the @Large Research team from VU Amsterdam[1]. This database combines literature sets from multiple publishing platforms. To use this dataset, we upload it into a PostgreSQL database running in Microsoft Azure. Our query is divided in three parts. The first should reflect the task we want to focus on. The second specifies the different environments, and the third specific constrictions. We use 2021 as our cut-off date to make sure we only look at the most recent publications. We use truncated versions of some search terms to capture all word forms, including verbs, nouns, and adjectives. These are indicated by the use of the word stem followed by an asterisk (*). The query in Figure 2 is a blueprint and reflects how the results are generated.

[1]The database can be retrieved from https://github.com/atlarge-research/AIP

A list of the different search terms is presented in Table 3. The search terms are organized based on the query structure and were collected during an initial exploration of the topic, using a set of seed publications.

```
SELECT *
FROM publications pub
WHERE pub.title ilike "<<task>>"
    AND pub.title ilike "<<environment>>"
    AND pub.title ilike "<<constraint>>"
    AND pub.year >= <<year>>;
```

Figure 2: Pseudo-SQL query for AIP

«task»	«environment»	«constraint»
offloading	cloud	energy
	edge	carbon
	fog	power
	compute continuum	sustainab*
	thing	
	iot	

Table 3: Search terms

This search results in a set consisting of 273 publications. To further analyze this result, we extract the *cited_by_count* number, which is the number of times this paper is referenced so far by other publications, using the crossref API for each entry. We calculate the average citations per year by dividing the total *cited_by_count* number by the years it is now published. This is necessary because more recent publications may are not as much cited as older ones.

In Figure 3 we present the amount of publications per year. As on can see, includes our result only one publication for *2024*. This is because the dataset was created in *2024* and does not include every resource from this year.

4.2 Conferences

Since the extracted literature from AIP does not include the most recent publications, we apply two additional research methodologies. Our next step is to look at different conferences from which we hope the retrieve fitting as well as high quality publications. Very prominent conferences are HotCarbon, which focuses on sustainable computer systems, and ICGCET, which focuses on green computing and engineering technologies. We included conferences related to either the environment or the constrain as mentioned in Section 4.1. A list of all examined conferences is presented in Table 4.

Name	Focus	Link
HotCarbon	Sustainable computer systems	https://hotcarbon.org/
ICGECT	Green computing and engineering technologies	https://icgcet.org/
USENIX ATC	Computer system research	https://www.usenix.org/
HiPEAC	Edge and cloud computing	https://www.hipeac.net/
SOCC	Cloud computing	https://acmsocc.org/
ICFEC	Fog and edge computing	https://icfec2024.ontariotechu.ca/
HotNets	Hot topics in networks	https://conferences.sigcomm.org/hotnets/
EuroSys	Systems software research and development	https://2024.eurosys.org/
CloudComm	Cloud computing technology and science	https://www.cloudcom2024.org/
ICDCS	Distributed computing systems	https://icdcs2024.icdcs.org/
GreenCom	Green computing and communications	https://ieee-cybermatics.org/2024/greencom/
IGSC	Green and sustainable computing	https://www.igscc.org/

Table 4: Conferences

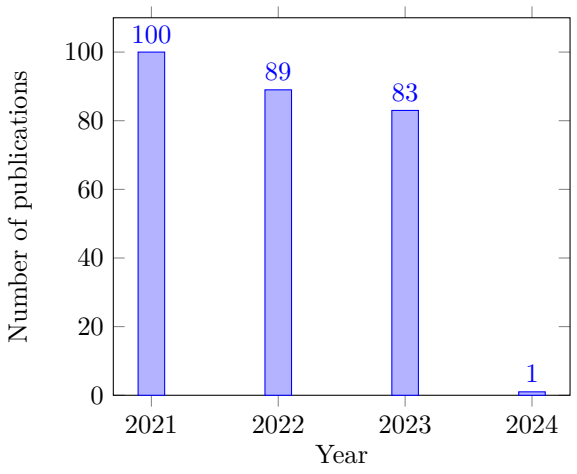

Figure 3: Publication count per year in AIP dataset

For each conference we analyze the accepted papers and use the search terms in Table 3 to determine wether we include the publication in our dataset.

4.3 Research Libraries

Our final step in procuring literature is to query several publishing websites. This allows us to find the most recent publications. We focused on three renowned services - IEEE Xplore, ACM Digital Library, and Elsevier. In this step we focus on the most recent literature, because we can assume that all relevant publications from 2021 till 2023 were already retriever through querying AIP or conferences. We adopt the same search terms as presented in Table 3 and include publications by reviewing the abstract to make sure it is suitable for our survey scope.

4.4 Inclusion Criteria

To further assess and evaluate the results, we defined requirements to decide whether we include the publication in further evaluation. Hereby, we differentiate between two types of requirements. The first set of requirements are hard rules. If one of these constraints is not fulfilled, the publications will not be further assessed.
These are:

- English language
- Full-text available

The second set of criteria were soft requirements. If a paper did not fulfill these, it can still be included if the content is relevant for the topic of energy efficient task offloading.
Our soft requirements are:

- Published between 2021 and 2025
- Research field is Computer Science
- Peer-reviewed

Our resulting set of publications, ordered by the normalized citation count introduced in Section 4.1, is further reviewed and categorized. Hereby, we differentiate following categories:

- Layer
  - Cloud
  - Edge
  - Endpoint
- Strategy
  - Heuristic/Policy
  - Optimization
  - Machine Learning

Each publication is added to the subcategory of the offloading target layer and the offloading strategy. This allows us to focus on the specific subset of publications which is relevant for each section.

During this categorization step, we can already observe a current research trend toward offloading to the edge layer, as most publications focus on this layer.

5 Offloading Layer

In this section, we investigate various offloading strategies from the endpoint layer towards an offloading target. We categorize the approaches in regard of the location of the target. First, we highlight the advantages and disadvantages offered by each destination infrastructure. Then, we present the inspected offloading approaches and, finally, analyze how they address energy efficiency.

5.1 Offloading to the Cloud

The cloud environment includes servers hosted by cloud providers as well as cloud infrastructure managed on-premise by the user. Cloud services are characterized by providing high computational power, large storage capacity, and advanced networking capabilities. These resources can be provisioned dynamically, which makes the cloud a popular environment to run tasks when local resources are not sufficient. Cloud services provide high availability by offering their services over the internet, allowing users to access them with just an internet connection. [12]

However, by constraining access through the internet, cloud services often introduce latencies that may not align with execution speed requirements of end-user applications. Additionally, the cloud layer supports multi-tenancy, making workload estimation a challenging task. To handle peak loads, providers run reserved hardware when exceptionally many user want to access their services. This overprovisioning of resources introduces higher total energy consumption. [27]

With the additional computation and storage resources, the cloud became an especially popular offloading destination for tasks from mobile devices. [33] But with the drawback of low latency, not all tasks types can run in the cloud. The endpoint must decide which tasks to offload in order to benefit from cloud resources, and which tasks to execute locally due to latency constraints. Accordingly, we present a range of strategies that can be employed to address this challenge. We focus on the strategy, the task type analysis, and how energy efficiency is realized within this approach and highlight advantages of each technique.

Task offloading strategies

One fundamental problem during the offloading evaluation is when the decision to offload tasks is taken and also when they are offloaded. Hereby, we consider two different approaches. Tasks can be offloaded either *sequential*, where each task is processed as soon as it is issued, or in *batch*. In *batch* processing, an endpoint device analyses a job consisting of multiple tasks.

Aishwaryna et al. [4] formulate a process which categorizes each single task into independent and dependent task. Independent task are fully offloaded or executed locally depending on their energy usage and execution time. For independent tasks, the system first estimates whether dividing them into subtasks would be beneficial. Based on this, the tasks or subtasks are then executed either locally or in the cloud. In both cases the tasks are offloaded immediately after the evaluation and every tasks is estimated individually, making this an *sequential* offloading approach.

Patel et al. [24] apply a strategy, where a local scheduling algorithm determines whether a task should be offloaded or not. When a task should be offloaded, a request is send to an agent running in the cloud. This agent can refuse the task, if not enough resources are available. Offloaded task are additionally scheduled in the cloud. Therefore, the approach is able to optimize the cloud resource utilization by including *batch* processing at least for the offloaded tasks.

Lu et al. [20] calculate the energy consumption from the view of the endpoint devices for local execution and offloading. Based on these values, the scheduling algorithm decides which action to take. The task is added to either the local of the cloud task queue. The queue is then reordered in regard of the deadline constraint. By utilizing these task queues, the offloading approach regularly reviews the workload. In this way, the system is able to comply with both energy and latency specifications by utilizing a *batch* processing approach.

Observation 1: *Batch* offloading enables a more optimized scheduling solution, but it requires the system to be aware of the existing tasks. Therefore,

it can only be performed at intervals when enough tasks have accumulated or when multiple devices offloaded simultaneously to a dedicated scheduler.

Task selection
A fundamental question that needs to be answered when implementing a good offloading strategy for complex tasks is how you define a complex tasks. The investigated approaches use multiple metrics gathered before the offloading decision is taken to estimate the task complexity. These can include task-specific information, such as the required data or the energy needed to execute the task, as well as contextual parameters including system information like available compute resources, current bandwidth, and more. Especially contextual parameters can very greatly depending on the current system and network utilization. Depending on which metrics the approach includes, we differentiate between *task-focused* and *context-aware* task evaluation.

Lu et al. [20] estimate the tasks based on the local energy consumption. Hereby, they include the power consumption and processing ability of the mobile device while executing a single instruction. For each task they analyze the amount of instructions needed and calculate the total energy consumption based on these values. To evaluate the offloading energy consumption, they include the energy consumed while sending the tasks and receiving the result. This approach can also be categorized as a *task-focused* complexity evaluation.

To estimate the task complexity, Patel et al. [24] focus on the energy consumption of a task. To calculate the energy consumption for local execution, they include the power consumption for the task, the CPU cycles needed to execute the task, and the execution speed of the endpoint device. For offloaded tasks, the formula includes the energy for sending and receiving data, as well as the energy consumed in the idle state of the endpoint device while the cloud executes the task. Additionally they include the battery level of the mobile device and offload tasks if it is either more energy efficient to run them in the cloud or if the local device has not enough energy left to execute the task. By including the battery level of the mobile device, this approach includes a variable factor in the task evaluation, which makes this approach *context-aware*.

The first two approaches focus on optimizing the energy consumption of the endpoint device. Hao et al. [16] also include the energy consumption of the cloud resources. This addition to the complexity estimation allows the authors to optimize the energy consumption of the whole system and represents a fully *context-aware* approach.

Observation 2: The more factors are included in the complexity estimation, the more accurately tasks can be offloaded. By including the current context of the system, the same task may be estimated differently. Therefore, an overview of the entire system is necessary to obtain a good task evaluation.

Energy efficiency
In the investigated offloading techniques, energy efficiency is addressed from two different perspectives: *device optimization* and *system optimization*. *Device optimization* focuses on lowering the energy consumption for the local device only, while the later

Lu et al. [20] and Patel et al. [24] focus on optimizing the energy consumption of the endpoint device. The motivation behind this is that endpoint devices often run on battery power. The offloading technique aims to maximize the device's lifespan until the next recharge.

Hao et al. [16] and Aishwaryna et al. [4] introduce offloading techniques that also analyze the energy consumption of the cloud resources. The energy consumption of cloud resources is estimated using simulated values.

Observation 3: A holistic energy model is difficult to develop because the energy consumption of cloud resources can only be estimated. The data provided by cloud providers is often limited and may exclude many relevant factors.

5.2 Offloading to the Edge

The edge layer pushes compute and storage resources towards the edge of the network. It provides additional resources while keeping a fairly limited latency downfall. The edge is a good candidate to offload tasks that demand additional resources but come with latency constraints. Often, edge servers are deployed to fit a certain purpose and, therefore, the amount of backup resources can be lowered. This does not only bring down hardware requirements but also the overall energy consumption. [3] The optimization strategies distinguish two different approaches. Edge resources may contribute in a similar manner as the cloud resources and act as the only addition to the endpoint devices. But they can also be deployed as an 'in-between' layer which provides additional resources to the endpoints with lower latency.

Task offloading strategies
We distinguish only *edge-based* offloading and *hybrid* offloading approaches. *Edge-based* offloading approaches only expand the system by including edge resources, while *hybrid* solution include edge and cloud resources. When offloading to edge devices without utilizing the

cloud, the approaches mirror strategies used in cloud-only offloading scenarios. The objective hereby remains the same and is set to find an optimal distribution between locally executed tasks and offloaded tasks.

Jiang et al. [17] introduce an offloading and resource allocation scheme. Their approach works in a time-sliced manner. In each time slice, a device can issue only one task to a manager on the edge device. Based on the task specifications and the resource utilization, the manager decides where the task should be executed. They present both a centralized and a distributed approach. The centralized algorithm exploits all available system information. Due to the fact that this may not be the case in a real scenario, they limit the system information in the decentralized algorithm. By doing so, the approach introduces an uncertainty factor which may exist in a real life scenario.

Bi et al. [7] on the other hand optimize a set of tasks through a combination of genetic algorithm, simulated annealing, and particle swarm algorithm. Their approach tries to find the perfect distribution for a set of particles at a certain point in time. Hereby the particles represent different attributes of the available tasks. The perfect distribution includes the offloading decision and, where applicable, the edge server that should executed the offloaded task. The endpoint devices and edge servers send their information to an external scheduler which runs the optimization algorithm. By doing so, an optimal offloading solution for the whole system can be found.

Observation 4: The availability of system information enhances the ability of the optimization approach to distribute tasks more effectively. Maintaining the decision-maker's awareness of the current system state is challenging and can only be achieved through continuous information updates.

Another prominent research direction in recent publications is the utilization of edge resources alongside the cloud, called *hybrid* offloading. This requires an additional decision-making step to determine whether the offloaded tasks should be executed on the edge device or in the cloud. The investigated approaches either consider the entire system, focus on a subpart of the system, or concentrate on one layer at a time to determine the offloading decision. Accordingly, they present three different offloading approaches: system-focused, subsystem-focused, or layer-focused offloading.

Wu et al. [30] consider a set of tasks belonging to an application. For each layer they calculate the response time, defined as the time from submitting a task until its completion. Additionally, they evaluate the energy consumption for the local device when offloading is done to each other layer. Based on these two metrics, they optimize the set of tasks using their proposed energy-efficient dynamic task offloading algorithm, which is based on the Lyapunov optimization. This approach aims to provide an optimal offloading solution for the set of tasks belonging to the application.

Aazam et al. [2] introduce a two-layered approach. The endpoint device sends tasks to a global gateway which analyses each task and either forwards it to the edge or to the cloud layer depending on whether the task includes complex data or not. Their approach only focuses on optimizing the offloaded tasks. The initial offloading decision is taken by the endpoint device. Additionally, a second scheduling decision is taken by the global gateway.

Teng et al. [29], similar to Jiang et al. [17], propose a time-slot-based offloading technique. An endpoint device issues a job consisting of multiple tasks, which are offloaded in each time slot. The presented break point strategy checks whether the endpoint possesses enough compute capacity as well as the required service. If the endpoint satisfies these conditions, the task is executed on the local device. Otherwise, the edge layer is examined. Hereby, only the requested service constraint is checked. If the edge layer cannot provide the requested service, the task is offloaded to the cloud layer.

Observation 5: *Hybrid* offloading solutions enable more fine-grained offloading decisions but introduce energy overhead through increased coordination, continuous context monitoring, and complex decision-making processes. Additionally, since they include all three layers, they require more system resources and communication effort across the entire architecture.

Task selection

If the offloading scheme only includes the edge layer as alternative execution handler, the task estimation strategy follows a similar approach like in cloud-only approaches.

Bi et al. [7] analyze several factors related to computation and offloading. These include the device's computation and offloading energy, the cloud's computation energy, CPU cycles, storage requirements, and the latency introduced by offloading. These measurements are combined to formulate an optimization problem. Using these factors, they evaluate the energy consumption of each task from the endpoint device's perspective. Additionally, they formulate a latency model which estimates the delay introduced by offloading the task to an edge device. This approach can be classified as *task-focused* with the addition of a latency model.

Jiang et al. [17] also consider the device's energy consumption for executing a task. Their main objective is to reduce the device's energy usage while meeting the task execution constraints. But task complexity is hereby only measured indirectly by finding the best

offloading decision for the task. As a result, even simple tasks may be treated as complex and offloaded when local resource availability is low. Therefore, this task evaluation approach can be classified as *context-aware*.

Since *hybrid* offloading approaches include edge devices alongside cloud resources in the offloading scheme, the task evaluation step must additionally decide whether the task is more suitable for the edge or the cloud.

Aazam et al. [2] utilize a global gateway that decides whether a task should be offloaded to the cloud or to an edge server based on the data complexity. In this *task-focused* offloading approach, tasks are only offloaded to the cloud if they include complex or bulky data. Otherwise they are executed on an edge device. Locally executed tasks are already excluded in this decision, therefore, the global gateway only needs to decide between the edge or the cloud as an offloading target.

Wu et al. [30] estimate task complexity using a response time model and an energy consumption model. Based on the outcomes of these two models, tasks are either offloaded to a dedicated layer or executed locally. This approach can be classified as a *model-based*, *task-focused* evaluation.

Teng et al. [29] consider tasks dynamically, depending on the current layer. In the endpoint layer, tasks are offloaded if they require excessive compute resources or if a needed service is unavailable on the endpoint device. In the edge layer, a task is considered complex only if it requires a service that is unavailable, in which case it is offloaded to the cloud. This offloading approach includes two *layer-specific*, *context-aware* estimation strategies.

Energy efficiency

By including the edge layer to the system architecture, the evaluation of energy efficiency becomes more complex and can now include either the whole system, a subset of the system, or only one layer.

Aazam et al. [2] use a simulation environment called SFogSim. This simulation is based on data from Amazon S3 services as well as Azure Cloud Service. By utilizing this simulation environment, energy efficiency of the edge layer is analyzed. This analysis shows that edge utilization and power consumption follow a linear distribution. Therefore, the edge resources can be utilized but only with additional energy consumption. Their evaluation focuses only on the edge layer and is, therefore, *layer-based*.

Bi et al. [7] use data provided by the Google cluster to estimate energy consumption on both local and edge devices. In their approach, they successfully reduce the combined energy usage of endpoint and edge devices. Their evaluation only focuses on a subsystem-edge and endpoint devices-while ignoring energy consumption in the cloud.

The offloading approaches investigated so far did not provide a comprehensive overview of energy usage across the entire system. Ahvar et al. [3] propose an estimation model for different system architectures in the cloud-edge infrastructure. Their approach consists of multiple calculations including the static energy consumptions of physical machines, the dynamic energy consumption of allocated machines, and the energy consumption of networking between the machines and from outside devices. Based on these and other metrics, an estimate of the system's total energy consumption is produced. Although relevant in the context of energy-aware computing, the paper by Ahvar et al. [3] is not primarily concerned with offloading optimization. Its focus lies on energy estimation across a cloud-edge infrastructure.

5.3 Offloading to other Endpoints

In addition to offloading tasks to another layer, they can also be split up and processed by multiple nearby endpoint devices. This is called *collaborative computing*. [31] The target endpoint devices typically come with very limited computational resources. However, their advantage lies in the very low latency, as they are often deployed in the same local network. We present different approaches for offloading tasks to other endpoint devices while ensuring correct execution.

Task offloading strategies

Through the limited resource capacity of the devices, often not more than one task can be considered to be executed by a collaborative endpoint device. Additionally, the set of available devices can change rapidly due to their non-fixed locations.

Tan et al. [28] propose an offloading scheme in which tasks are either executed locally, offloaded to a collaborative device, or offloaded to an edge server. They utilize an Ant Colony System to find the perfect distribution between the offloading strategies for each iteration. Hereby the availability of collaborative devices is always given. But this may not necessarily be the case in a real-life scenario. Since endpoint devices are often mobile, they may run out of range or a competitive task offloading source may try to offload to the same target endpoint device.

Qin et al. [25] introduce a conflict detection scheme in their offloading approach to solve this problem. They focus their work on collaborative computing in vehicles. Because of the mobility of the vehicles, continuous availability cannot be ensured. Therefore, Qin et al. [25] first analyze available collaborative vehicles by considering both the distance as well as their direction. If offloading is feasible, an offloading request is issued with the vehicle.

This request can be denied when not enough resources on the target vehicle are available. Additionally, they use a time slot-based approach to regularly verify the availability of the target endpoint devices and to continuously optimize the task distribution.

> **Observation 6:** Due to the mobility of target endpoint devices, offloading may not always be possible. Therefore, their availability must be verified to ensure that the endpoint device can receive and process offloaded tasks.

Task selection
Leveraging collaborative devices can only be effective if the offloaded task is manageable by the collaborative vehicle in appropriate time. Qin et al. [25] divide the tasks into subtasks which are small enough to be executed in one time slot. This ensures that time constraints can be met even with additional communication latency. By employing task slicing, the approach can offload tasks that would otherwise be too large.

Energy efficiency
In task offloading scenarios involving collaborative endpoints, the primary focus lies on reducing energy consumption at the source endpoint device. The investigated offloading strategies by Qin et al. [25] and Tan et al. [28] both use the local energy consumption as a key optimization constraint. Since the endpoint devices are often mobile and not connected to a stable power supply, energy usage must be carefully limited at the endpoint.

> **Observation 7:** In collaborative offloading scenarios, energy efficiency is predominantly addressed on the local endpoint device. Sharing information about energy consumption over the limited network would introduce an additional overhead. Therefore, estimating energy consumption of multiple devices with different consumption models becomes a significant challenge.

6 Offloading Strategies

Additionally to the environment where tasks can be offloaded to, we want to investigate the offloading strategy. We distinguish between three categories. Heuristics and policies, algorithms, and machine learning tactics are investigated. We highlight advantages and disadvantages of each approach and present how they are executed in the investigated offloading strategies.

6.1 Heuristics and Policies

Heuristics and policies include all decision-making approaches based on simple comparisons or constraints. Their simplicity is a significant advantage, as they require minimal resources and can be applied in every layer of the cloud-edge-endpoint infrastructure. Therefore, they are often used as an initial filtering step to identify tasks that are restricted to a specific layer.

Aishwaryna et al. [4] include multiple policies in their approach. This allows them to categorize the incoming tasks and decide whether they should be offloaded or not. Their policies analyze the dependencies between tasks, the energy consumption during local and offloaded execution, as well as the execution time during local and offloaded execution. Their policies compare these values and decide on an execution action.

> **Observation 8:** Heuristics and policies offer lightweight, easily deployable solutions that can be applied on endpoint devices. However, they are limited in optimizing total energy consumption and typically fail to account for the system as a whole.

6.2 Optimization Algorithms

Another commonly used approach in offloading strategies is the application of optimization algorithms. Optimization algorithms try to incrementally improve the design until it can no longer be improved or until the budgeted time or cost has been reached. [19] The objective in the investigated offloading approaches is to lower energy consumption while executing the tasks within certain service level constraints like execution time. Therefore, optimization algorithms can be particularly suitable for offloading techniques.

The range of optimization algorithms is broad and each one of them fits a certain objective or has a different approach. To illustrate the application of optimization approaches in offloading strategies, we present the most prominent techniques and explain how they were applied.

A population method-based metaheuristic algorithm is introduces by Bi et al. [7]. Population methods focus on optimizing a collection of design points. The proposed GSP algorithm is based on the particle swarm optimization. Each particle records its position, velocity, and optimal position it visited so far. The goal for this algorithm is to find a global minimum or maximum. The particles are then shifted towards this position. [19] In this optimization approach the particles represent different constraints like the speed of the local CPU, the power consumed to download data from the edge device, the available bandwidth, the progress of execution, and a penalty function based on the total energy consump-

tion. Through the particle swarm optimization the best distribution of these values is found.

Wu et al. [30] and Jiang et al. [17] base their approaches on Lyapunov optimization technique. Lyapunov optimization works by stabilizing network queues under a certain constraint. [23] In the investigated approaches the they try to stabilize the energy consumption under the constraint of execution delay.

In all proposed techniques, the optimization algorithm is executed on an additional scheduling node. This computation overhead introduces additional computation effort as well as network pressure when gathering information from the whole system. The tradeoff between optimization advantage and system pressure must be considered carefully.

Observation 9: Optimization algorithms can be particularly useful in offloading approaches. By defining objectives and constraints, they enable the optimization of energy consumption while satisfying certain service level agreements. However, these approaches often require considerable setup effort and a dedicated scheduling device. Moreover, they may introduce computation overhead and network pressure into the system.

6.3 Machine learning-based approaches

A third category of offloading approaches we highlight involves strategies based on machine learning algorithms. We define a machine learning algorithm as one that is able to learn from data. [13] Therefore, the models used in the approaches require data for a training phase beforehand. During the training phase, the models are fine-tuned to fit the designated purpose. During the execution the offloading algorithms can estimate their effectiveness and adapt to sudden system changes.

Tan et al. [28] utilize a Deep Q-Learning Network to find the optimal task offloading distribution between the local device and the edge devices. They include a decision matrix and try to find the perfect offloading decision to optimize the total energy consumption of the local device. The energy consumption only includes the local execution costs and the transmission costs and neglect the energy consumption of the edge layer. Their offloading approach requires a learning phase and the offloading algorithm is performed on a dedicated server. But by constantly evaluating the offloading decisions and feeding this information back into the algorithm, the approach is able to adapt to the current system state.

Observation 10: Machine learning algorithms can be fine tuned to different system infrastructures. But to achieve this, they require system data beforehand. The learning stage comes with additional effort and the model must run on a designated device, since endpoint devices are usually not powerful enough to run them.

7 Research Deficits, Challenges, and Opportunities

Based on our observations and the overall investigation of the topic, we identify the main challenges in offloading approaches, along with research gaps that present promising future research directions. Additionally, we discuss fundamental questions that emerged over the course of this survey.

System overview

As identified in Observations 1, 2, 4, and 6, a comprehensive system overview is a fundamental requirement for implementing an efficient offloading optimization technique. Two key challenges arise in this context:

First, gathering current utilization information is complex due to the amount and variety of devices within the computing system. The data collection process leads to additional pressure on the network, as all information must be send to a central collection point.

Second, the amount and complexity of tasks entering the system can very significantly over time, making it challenging to maintain consistent optimization. While including more tasks in the optimization approach can lead to more efficient resources utilization, waiting for tasks to accumulate may slow down the overall process.

Future research aimed at addressing these challenges should focus on the development of comprehensive information-gathering techniques capable of collecting data from all components of the system. Additionally, task estimation methods represent a promising direction for further investigation to effectively tackle the aforementioned challenges.

Energy modeling

Observations 3 and 7 highlight challenges in achieving a comprehensive energy estimation within offloading approaches. Modeling the energy consumption of cloud resources is difficult, as cloud providers limit the information about energy consumption. As a result, the energy consumption can only be approximated through estimation techniques.

In addition, obtaining accurate energy consumption data of endpoint devices is also difficult. The devices often run on batteries with limited power supply, making it essential to avoid unnecessary computational and networking overhead.

Future research should focus on developing models for comprehensive energy evaluation that includes the

most accurate data as possible from the cloud layer. Additionally, a lightweight energy data collection scheme for endpoint devices should be explored to collect data without applying excessive pressure on the devices or the network.

Offloading complexity
As detected in Observations 5, 8, 9, and 10, dedicated scheduling nodes and *hybrid* solutions can improve the overall performance and efficiency of the system but introduce additional hardware requirements and setup effort. This tradeoff must be considered and the advantages of a *hybrid* solution or advanced offloading approaches can only be exploited with an appropriate system size. A fundamental research question is how to integrate the energy-efficient offloading approaches. Additional focus hereby should be on use cases that may not necessarily align the proposed examples in the publications but may still benefit from deploying an offloading technique.

Energy-proportional computing
As witnessed in the examined offloading approaches is energy-efficiency either examined on the local device or a system wide level. But the approaches do not take the actual efficiency of the edge and cloud devices into account. Meisner et al. [21] and Barroso et al. [6] investigate approaches that introduce energy efficiency by analyzing server utilization. Their research shows that server utilization is usually between 10% and 50%. But the energy efficiency of the servers increases with higher utilization. This is due to the fact that an idle server already consumes around 50% of its maximum energy consumption. Their approach is to run some servers on a higher utilization level and in return shut down idle servers. This approach could be cooperated in the investigated offloading approaches.

Sustainability
Hanafy et. al. [15] compare energy-efficiency and carbon-efficiency and analyze their impact on sustainability. For this purpose they explore the trade-off of these two aspects with the help of four computing mechanisms. They present a fundamental question whether it is more sustainable to optimize for energy efficiency or carbon emission. Offloading strategies have a very strong influence on these two optimization strategies. Research in the area of sustainable offloading is still very sparse and the presented strategies do not necessarily represent the most sustainable solutions.

8 Conclusion

In this survey we investigated different offloading approaches which focus on energy efficiency. As local resources are often limited and tasks need to be offloaded to more powerful devices, additional network and compute steps can lead to increased energy consumption. Our goal was to explore approaches to minimize this energy overhead by implementing an energy-efficient offloading scheme. The three-layered compute continuum represents an efficient infrastructure to optimize energy consumption by offloading resource intensive task to the most suitable layer. We presented different offloading approaches based on the target execution layer as well as the offloading strategy they are based on. For each approach we analyzed the specific offloading technique, the task evaluation procedure, and the energy estimation method. We found that current research can be expanded in the direction of creating a comprehensive and continuously maintained system overview, evaluating the complete system energy consumption, and integrating the proposed techniques in current infrastructure. Additionally, we found that energy-proportional computing can be a promising approach to optimized research utilization and could be integrated into current offloading techniques. And, finally, we considered the question whether energy efficiency can be considered the only aspect in creating a more sustainable compute landscape.

References

[1] Cisco Annual Internet Report (2018–2023) White Paper. Tech. rep., Cisco Systems, 2023.

[2] Aazam, M., Islam, S. u., Lone, S. T., and Abbas, A. Cloud of Things (CoT): Cloud-Fog-IoT Task Offloading for Sustainable Internet of Things. *IEEE Transactions on Sustainable Computing 7*, 1 (Jan. 2022), 87–98. Conference Name: IEEE Transactions on Sustainable Computing.

[3] Ahvar, E., Orgerie, A.-C., and Lebre, A. Estimating Energy Consumption of Cloud, Fog and Edge Computing Infrastructures. *IEEE Transactions on Sustainable Computing 7*, 2 (Apr. 2022), 277–288. Publisher: IEEE.

[4] Aishwarya, R., and Mathivanan, G. COCAME: A Computational Offloading in Cloud Assisted Mobile Environments Structure to Enhance Performance and Energy. In *2024 14th International Conference on Cloud Computing, Data Science & Engineering (Confluence)* (Jan. 2024), pp. 264–271. ISSN: 2766-421X.

[5] Al-Dulaimy, A., Jansen, M., Johansson, B., Trivedi, A., Iosup, A., Ashjaei, M., Galletta, A., Kimovski, D., Prodan, R., Tserpes, K., Kousiouris, G., Giannakos, C., Brandic, I., Ali, N., Bondi, A. B., and Papadopoulos, A. V. The computing continuum: From IoT to the cloud. *Internet of Things 27* (Oct. 2024), 101272.

[6] Barroso, L. A., and Hölzle, U. The Case for Energy-Proportional Computing. *Computer 40*, 12 (Dec. 2007), 33–37.

[7] Bi, J., Yuan, H., Duanmu, S., Zhou, M., and Abusorrah, A. Energy-Optimized Partial Computation Offloading in

Mobile-Edge Computing With Genetic Simulated-Annealing-Based Particle Swarm Optimization. *IEEE Internet of Things Journal 8*, 5 (Mar. 2021), 3774–3785. Conference Name: IEEE Internet of Things Journal.

[8] Bocci, A., Forti, S., and Brogi, A. Sustainable Cloud-Edge Infrastructure as a Service. In *2023 12th Mediterranean Conference on Embedded Computing (MECO)* (June 2023), pp. 1–4. ISSN: 2637-9511.

[9] Boukerche, A., Guan, S., and Grande, R. E. D. Sustainable Offloading in Mobile Cloud Computing: Algorithmic Design and Implementation. *ACM Comput. Surv. 52*, 1 (Feb. 2019), 11:1–11:37.

[10] Casino, F., Lopez-Iturri, P., and Patsakis, C. Cloud continuum testbeds and next-generation ICTs: Trends, challenges, and perspectives. *Computer Science Review 56* (May 2025), 100696.

[11] Dong, S., Tang, J., Abbas, K., Hou, R., Kamruzzaman, J., Rutkowski, L., and Buyya, R. Task offloading strategies for mobile edge computing: A survey. *Computer Networks 254* (Dec. 2024), 110791.

[12] Gajbhiye, A., and Shrivastva, K. M. P. Cloud computing: Need, enabling technology, architecture, advantages and challenges. In *2014 5th International Conference - Confluence The Next Generation Information Technology Summit (Confluence)* (Sept. 2014), pp. 1–7.

[13] Goodfellow, I., Bengio, Y., and Courville, A. *Deep Learning*. MIT Press, 2016.

[14] Gorton, I., Greenfield, P., Szalay, A., and Williams, R. Data-Intensive Computing in the 21st Century. *Computer 41*, 4 (Apr. 2008), 30–32.

[15] Hanafy, W. A., Bostandoost, R., Bashir, N., Irwin, D., Hajiesmaili, M., and Shenoy, P. The War of the Efficiencies: Understanding the Tension between Carbon and Energy Optimization. *ACM SIGEnergy Energy Informatics Review 4*, 3 (July 2024), 87–93.

[16] Hao, Y., Cao, J., Wang, Q., and Ma, T. Energy-aware offloading based on priority in mobile cloud computing. *Sustainable Computing: Informatics and Systems 31* (Sept. 2021), 100563.

[17] Jiang, H., Dai, X., Xiao, Z., and Iyengar, A. Joint Task Offloading and Resource Allocation for Energy-Constrained Mobile Edge Computing. *IEEE Transactions on Mobile Computing 22*, 7 (July 2023), 4000–4015. Conference Name: IEEE Transactions on Mobile Computing.

[18] Kanupriya, Chana, I., and Goyal, R. K. Computation offloading techniques in edge computing: A systematic review based on energy, QoS and authentication. *Concurrency and Computation: Practice and Experience 36*, 13 (2024), e8050. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/cpe.8050.

[19] Kochenderfer, M. J., and Wheeler, T. A. *Algorithms for Optimization*. MIT Press, 2019.

[20] Lu, J., Hao, Y., Wu, K., Chen, Y., and Wang, Q. Dynamic offloading for energy-aware scheduling in a mobile cloud. *Journal of King Saud University - Computer and Information Sciences 34*, 6, Part B (June 2022), 3167–3177.

[21] Meisner, D., Gold, B. T., and Wenisch, T. F. PowerNap: eliminating server idle power. In *Proceedings of the 14th international conference on Architectural support for programming languages and operating systems* (New York, NY, USA, 2009), ASPLOS XIV, Association for Computing Machinery, pp. 205–216.

[22] Nabi, A., and Moh, S. Offloading decision and resource allocation in aerial computing: A comprehensive survey. *Computer Science Review 56* (May 2025), 100734.

[23] Neely, M. J. *Stochastic Network Optimization with Application to Communication and Queueing Systems*. Synthesis Lectures on Learning, Networks, and Algorithms. Springer International Publishing, Cham, 2010.

[24] Patel, Y. S., Reddy, M., and Misra, R. Energy and cost trade-off for computational tasks offloading in mobile multi-tenant clouds. *Cluster Computing 24*, 3 (Sept. 2021), 1793–1824.

[25] Qin, P., Fu, Y., Tang, G., Zhao, X., and Geng, S. Learning Based Energy Efficient Task Offloading for Vehicular Collaborative Edge Computing. *IEEE Transactions on Vehicular Technology 71*, 8 (Aug. 2022), 8398–8413. Conference Name: IEEE Transactions on Vehicular Technology.

[26] Raeisi-Varzaneh, M., Dakkak, O., Habbal, A., and Kim, B.-S. Resource Scheduling in Edge Computing: Architecture, Taxonomy, Open Issues and Future Research Directions. *IEEE Access 11* (2023), 25329–25350. Conference Name: IEEE Access.

[27] Shi, W., Cao, J., Zhang, Q., Li, Y., and Xu, L. Edge Computing: Vision and Challenges. *IEEE Internet of Things Journal 3*, 5 (Oct. 2016), 637–646.

[28] Tan, L., Kuang, Z., Zhao, L., and Liu, A. Energy-Efficient Joint Task Offloading and Resource Allocation in OFDMA-Based Collaborative Edge Computing. *IEEE Transactions on Wireless Communications 21*, 3 (Mar. 2022), 1960–1972. Conference Name: IEEE Transactions on Wireless Communications.

[29] Teng, M., Li, X., and Zhu, K. Joint Optimization of Sequential Task Offloading and Service Deployment in End-Edge-Cloud System for Energy Efficiency. *IEEE Transactions on Sustainable Computing 9*, 3 (May 2024), 283–298. Conference Name: IEEE Transactions on Sustainable Computing.

[30] Wu, H., Wolter, K., Jiao, P., Deng, Y., Zhao, Y., and Xu, M. EEDTO: An Energy-Efficient Dynamic Task Offloading Algorithm for Blockchain-Enabled IoT-Edge-Cloud Orchestrated Computing. *IEEE Internet of Things Journal 8*, 4 (Feb. 2021), 2163–2176. Conference Name: IEEE Internet of Things Journal.

[31] Yuan, H., and Zhou, M. Profit-Maximized Collaborative Computation Offloading and Resource Allocation in Distributed Cloud and Edge Computing Systems. *IEEE Transactions on Automation Science and Engineering 18*, 3 (July 2021), 1277–1287.

[32] Zabihi, Z., Eftekhari Moghadam, A. M., and Rezvani, M. H. Reinforcement Learning Methods for Computation Offloading: A Systematic Review. *ACM Comput. Surv. 56*, 1 (Aug. 2023), 17:1–17:41.

[33] Zhang, S., Yi, N., and Ma, Y. A Survey of Computation Offloading With Task Types. *IEEE Transactions on Intelligent Transportation Systems 25*, 8 (Aug. 2024), 8313–8333. Conference Name: IEEE Transactions on Intelligent Transportation Systems.