drones

Article

A Computation Offloading Scheme for UAV-Edge Cloud Computing Environments Considering Energy Consumption Fairness

Bongjae Kim [1,†], Joonhyouk Jang [2,†], Jinman Jung [3,*], Jungkyu Han [4], Junyoung Heo [5] and Hong Min [6,*]

[1] Department of Computer Engineering, Chungbuk National University, Cheongju 28644, Republic of Korea
[2] Department of Computer Engineering, Hannam University, Daejeon 34430, Republic of Korea
[3] Department of Computer Engineering, Inha University, Incheon 22212, Republic of Korea
[4] Division of Computer and AI, Dong-A University, Busan 49315, Republic of Korea
[5] Division of Computer Engineering, Hansung University, Seoul 04763, Republic of Korea
[6] School of Computing, Gachon University, Seongnam 13306, Republic of Korea
* Correspondence: jmjung@inha.ac.kr (J.J.); hmin@gachon.ac.kr (H.M.)
† These authors contributed equally to this work.

Abstract: A heterogeneous computing environment has been widely used with UAVs, edge servers, and cloud servers operating in tandem. Various applications can be allocated and linked to the computing nodes that constitute this heterogeneous computing environment. Efficiently offloading and allocating computational tasks is essential, especially in these heterogeneous computing environments with differentials in processing power, network bandwidth, and latency. In particular, UAVs, such as drones, operate using minimal battery power. Therefore, energy consumption must be considered when offloading and allocating computational tasks. This study proposed an energy consumption fairness-aware computational offloading scheme based on a genetic algorithm (GA). The proposed method minimized the differences in energy consumption by allocating and offloading tasks evenly among drones. Based on performance evaluations, our scheme improved the efficiency of energy consumption fairness, as compared to previous approaches, such as Liu et al.'s scheme. We showed that energy consumption fairness was improved by up to 120%.

Keywords: computational offloading; genetic algorithm; energy consumption fairness; drones; unmanned aerial vehicle

Citation: Kim, B.; Jang, J.; Jung, J.; Han, J.; Heo, J.; Min, H. A Computation Offloading Scheme for UAV-Edge Cloud Computing Environments Considering Energy Consumption Fairness. *Drones* **2023**, *7*, 139. https://doi.org/10.3390/drones7020139

Academic Editors: Hiroyuki Tomiyama, Ittetsu Taniguchi, Xiangbo Kong and Hiroki Nishikawa

Received: 12 January 2023
Revised: 11 February 2023
Accepted: 14 February 2023
Published: 16 February 2023

1. Introduction

Many services that operate based on UAVs (Unmanned Aerial Vehicles) have been developed and used. Drones are representative examples of UAVs. UAV devices can be combined and operated with high-performance embedded devices, such as NVIDIA Jetson AGX Orin and Xavier NX. Then, the UAV can perform the operations required for AI-based services by itself [1]. Another advantage of UAVs is that they are relatively free to move due to less influence from obstacles. Therefore, UAVs are suitable for the remote sensing and monitoring of large rivers, oceans, mountains, and urban areas [2]. However, most UAVs are operated with limited battery power, making energy efficiency important in drone-based applications and services.

Computational offloading schemes can be applied to reduce the energy consumption of UAVs and increase the energy efficiency for UAV-based applications and services. Computational offloading reduces energy consumption and improves the drone's processing power by delegating the drone's high-overhead tasks to the cloud. As shown in our previous study [2], tasks that included image- and artificial intelligence (AI)-related processing operations met their deadline with a computational offloading scheme. However, existing offloading schemes only consider direct communication between drones and cloud servers. Liu et al. [3] proposed an online computational offloading scheme that considered the

communication and routing-overhead of UAVs. However, this new approach did not consider the energy fairness and consumption issues that extend the operation duration of drone-based networks and guarantee the completion of a UAV's mission. During task allocation, tasks were not evenly allocated while taking into consideration the energy consumption of each drone. Therefore, the energy consumption of certain drones may be increased. As a result, the drone is unable to operate due to insufficient energy, and problems may occur, such as the drone-based network becoming partitioned.

Genetic algorithms (GA) are a method of optimization that can be applied to a wide range of problems, including those that are difficult or impossible to solve using traditional methods. GA can handle a large number of variables and constraints, and GA can find near-optimal solutions quickly. Previously, we had proposed a new concept in a preliminary study [4], in which a GA-based scheme considered energy consumption fairness. In this study, we proposed a computational offloading and workflow allocation scheme that considered energy consumption fairness based on our preliminary study. The computing environment of the proposed scheme involved a three-tier structure consisting of a UAV-based computing area, edge computing area, and cloud computing area, as shown in Figure 1. This three-tier structure was referred to as the UAV–edge–cloud computing environment. This environment was very similar to the modern mobile computing environment [3]. An application scenario in this three-tier environment could be used on a battlefield. Multiple drones could monitor the target field to detect various types of sensors. Simple location data could be locally processed in a drone while the image and video streaming data-processing could be offloaded to neighbor drones or edge servers. The entire target field analysis and the strategic movements of the drones could be processed in the cloud.

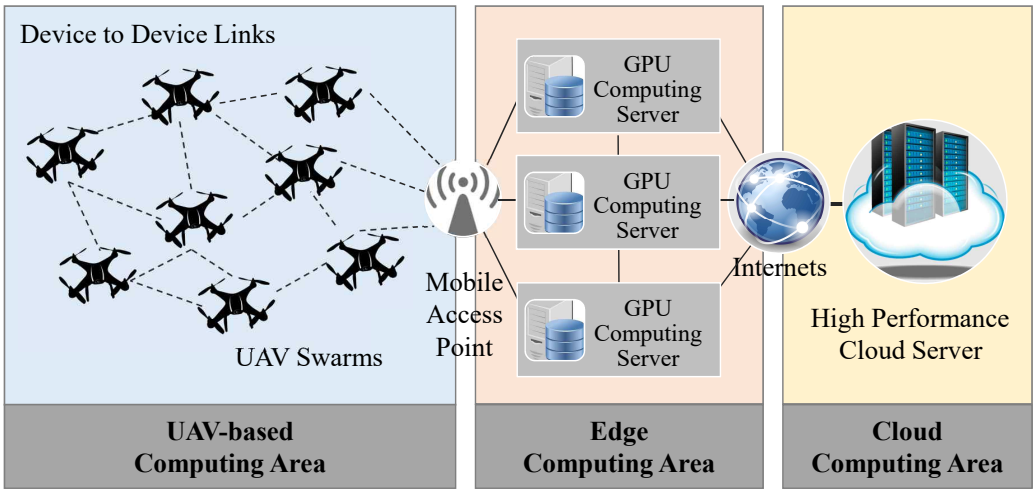


Figure 1. System model.

The proposed energy consumption fairness-aware computational offloading scheme allocated a workflow of multiple tasks to each node, constituting the UAV–edge–cloud computing environment, while considering the energy consumption in terms of fairness. A node could be one of the UAVs, edge servers, or cloud servers. If the tasks that needed to be allocated were not evenly distributed to each node, the energy consumption of each node to which the tasks were allocated would be unbalanced. In the case of a node that consumed a lot of energy, it could no longer operate due to energy depletion, shortening the lifetime of the entire network.

The proposed scheme was compared with Liu et al.'s Markov approximation-based scheme [3,5]. We evaluated the performance through simulations. The simulation results showed that the proposed scheme evenly consumed a drone's energy, as compared to Liu et al.'s scheme. It was shown that the energy consumption fairness was improved by

up to about 120.93%. Therefore, drone-based applications could be operated longer with better stability.The main contributions of this paper are the following:

- Our system model had three layers: UAV-based computing area, edge computing area, and cloud computing area. We considered the routing costs among the drones for each offloading task in the UAV-based computing area. We also considered the scheduling costs of requests of multiple users in the edge and the cloud computing areas.
- The energy efficiency is an important issue in drone-based systems, but the energy consumption fairness is also an important issue in order to sustain the network connection among the UAVs. The originality of our GA-based computational offloading and workflow allocation scheme considered the residual energy balance among the UAVs to extend the lifetime of the UAV-based computing area.
- We use a GA-based computational offloading decision scheme to determine a better solution by considering the energy consumption, the energy consumption fairness, and the resource constraints, such as processing power and network bandwidth. Our approach outperformed the previously reported Markov approximation-based schemes in terms of the energy consumption fairness.

The rest of this paper is organized as follows. In Section 2, we explain the existing offloading architectures and computational offloading schemes. Section 3 describes our computational offloading scheme while taking into consideration the energy consumption fairness. The performance evaluation results are presented and discussed in Section 4. In Section 5, we conclude this paper.

2. Related Works

Much research has been conducted on computational offloading architectures and computational offloading algorithms for UAVs. This section reviews and describes the related work on offloading architectures and algorithms.

2.1. Offloading Architectures

King explained a general architecture range from IoT devices to the cloud, as shown in Figure 2 [6]. In the edge tier, constrained devices handled sensing, actuation, and passing collected messages. The gateway devices focused on processing, interpreting, and integrating edge data and interacting with the cloud. In the cloud tier, end users used cloud services by connecting to the Internet, and the cloud stored collected data in data storage and provided various cloud services and analytical results to end users.

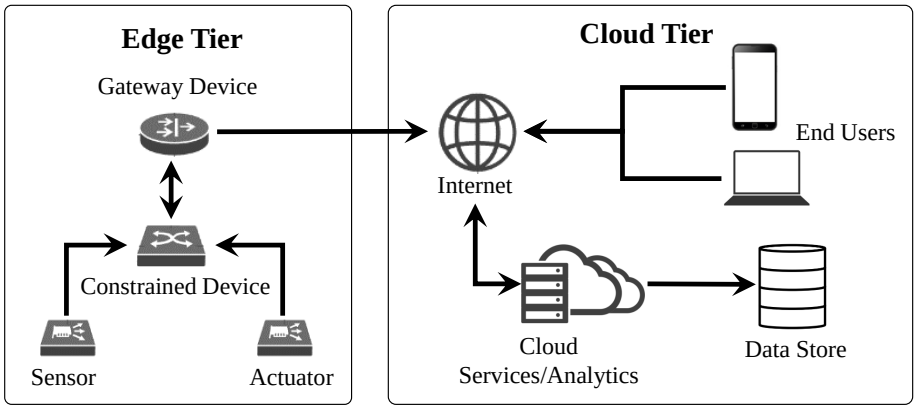


Figure 2. Cloud service architecture in IoT environments.

Serpanos et al. described the organization of an IoT system and the publish/subscribe model to support the heterogeneous and long lives of IoT systems [7]. The authors also described a typical architecture of the industrial Internet of things (IIoT). In the IIoT reference architecture model, there were six layers: management, security, device, network, service and application support, and application.

Cheng et al. proposed a UAV-assisted computational offloading architecture for the Internet-of-things (IoT) services [8]. The proposed architecture was based on virtual machines (VMs), and each VM operated offloading tasks from a low-powered ground device. UAVs had a gateway and edge-server role for the support of the Internet connection of ground devices and the execution of high-overhead tasks.

A UAV-enhanced intelligent edge-offloading network was also proposed [9]. A UAV passed an area with a low density of IoT mobile devices (IMDs) on the ground. However, a short time later, when the UAV flew over an area with a high density of IMDs, the IMDs had the option to offload their computationally intensive tasks or be execute them locally to minimize delays for critical tasks.

Yu et al. proposed a UAV-enabled mobile edge-computing system involving the interactions among IoT devices, a UAV, and edge clouds [10]. A set of ground mobile devices could not establish wireless communication due to shadowing and signal blockage. A UAV connected mobile devices to edge clouds by using ground-to-air up-link communication from mobile devices to the UAV and air-to-ground down-link communication from the UAV to the edge clouds.

An energy-efficient computational-offloading architecture was proposed [11]. Ground users uploaded data, including environment monitoring data, images, videos, and so on, to the UAV. After the UAV received the data from the ground users, the UAV processed the collected data and sent the results to the ground users. To improve the energy efficiency, transmitted bits of both the up- and down-links were allocated to time slots, and the trajectory of the UAV was jointly optimized.

Liu et al. designed a cooperative UAV-enabled edge-computing network with three layers: system, UAV, and ground device [12]. A base station provided the connection between an edge server and the UAVs, comprising the system layer. Each UAV was equipped with a processing unit and a communication module, and they hovered over assigned areas to provide communication and offloading services for ground devices, which comprised the UAV layer. Each ground device offloaded high-overhead tasks to a UAV, which defined the device layer.

In vehicle-assisted computational-offloading architecture for UAVs, an offloading task from a UAV was sent to a vehicle, and the result from the vehicle was then uploaded to UAV through air-to-ground links [13]. UAVs equipped with various sensors collected sensing data and performed lightweight tasks. Due to their limited onboard batteries, UAVs have limited computing power and operating time. UAVs conserved energy by offloading their heavy tasks to other, more powerful vehicles.

Alhelayly et al. proposed a mobile edge-cloud-computing system where UAVs played the role of mobile edge server to provide offloading services for multiple users [14]. The authors defined a communication model and a computational model to compare the energy consumption of local execution (operating a task on a mobile device) and a remote execution (offloading a task to the UAV or the cloud). A task assignment algorithm was also proposed to minimize the energy consumption of the mobile devices and the UAVs. This approach was similar to ours, as they also considered the energy consumption of the entire system. However, we based task offloading of UAVs according to the sequence of the workflow and the energy fairness among all the UAVS.

Huang et al. proposed a multi-UAV-based mobile edge-computing environment for offloaded tasks from ground users [15]. UAVs made a group and determined their roles as relay or computing nodes for each group. The relay node communicated with ground users and with the relay nodes of other groups. The commuting nodes executed offloaded tasks. This clustering was an efficient method of managing multiple UAVs, but in contrast, our approach considered the dynamic topology changes among the UAVs and their cooperation with edge and cloud servers.

2.2. Offloading Strategies

Kim et al. proposed a computational offloading scheme that supported remote sensing in real time using on AI-powered drones [2]. It consisted of a single drone and an edge-computing node. The proposed offloading scheme dynamically offloaded a deep-learning-based recognition and inference task efficiently while minimizing the energy consumption of the drone and meeting a given deadline constraint. However, the proposed scheme was not suitable for a swarm-based computing environment consisting of a large number of drones.

A computational offloading algorithm based on game theory was devised based on a non-cooperative theoretical game with N players according to three strategies: local computing, offloading to an edge server, and offloading to a more powerful cloud server rather than a local node and edge server [16].

Tang et al. proposed POSMU (Partial Offloading Strategy Maximizing the User task number) to obtain the optimal offloading ratio, transmission power, and local computing frequency [17]. The authors converted the optimization problem into multiple nonlinear programming problems and designed an efficient algorithm to solve the problem by applying block coordinate descents and convex optimization techniques. The authors also applied POSMU to UAV-enabled systems by analyzing a 3D communication model.

A hybrid offloading scheme was proposed by combining two offloading strategies: air-offloading and ground-offloading [18]. In the hybrid offloading scheme, each UAV could offload its tasks to nearby UAVs that had sufficient computational power and energy to perform the tasks. UAVs could also offload their tasks to edge cloud servers using the ground network maintained by ground stations.

Bai et al. proposed an algorithm to solve an energy-efficient computational offloading problem in the presence of both active and passive eavesdroppers [19]. The authors suggested the optimal solutions to the problems and then formulated and analyzed the conditions of the three offloading options: local execution, partial offloading, and full offloading. The solution also considered a computational overload event from a physical perspective.

An intelligent task offloading algorithm named iTOA for a UAV–edge-computing network was proposed [20]. iTOA could perceive a network's environment to determine the offloading process based on a deep Monte Carlo tree search, which was the core algorithm of Alpha Go. This approach estimated the trajectory of each UAV and identified the optimal server that could achieve the lowest power consumption and latency, as compared to other servers.

Zhao et al. proposed a software-defined networking (SDN)-based UAV-assisted vehicular computational offloading optimization algorithm to minimize the cost of vehicle computing tasks [21]. This system worked on behalf of vehicle users to execute computationally intensive and delay-sensitive tasks. The proposed game-theory-based offloading cost minimization algorithm used offloading parameters provided by an SDN controller.

These schemes assumed that UAVs directly access and communicate with a ground-based edge or cloud server. However, this assumption is not available in all applications, especially when monitoring large areas with UAVs.

Li et al. proposed a deep-learning-based offloading method between a single UAV, acting as an edge server, and users [22]. In this system, the UAV was able to serve all users as the mobile edge server and hovered over a user to receive offloaded tasks and to send back the results. This approach considered the energy consumption of the UAV, but in contrast, our approach considered multiple UAVs and the lifetime of the UAV network.

MADDPG (multi-agent deep deterministic policy gradient), based on reinforcement learning, was proposed to deploy edge servers near UAVs [23]. The proposed MADDPG attempted to balance the delay and energy consumption of a UAV and maximize the overall system utility. In the MADDPG system, UAVs directly communicated with edge servers, but in contrast, our approach had UAVs compose their own network, and an offload task was executed by the UAV's network or routed to the edge and cloud servers.

Liu et al. proposed a joint computational offloading and routing optimization algorithm for UAV–edge–cloud computing environments [3,5]. The proposed algorithms did not consider the energy consumption and energy fairness of each node when offloading and routing the task, however.

2.3. Genetic Algorithm-Based Offloading Schemes

Simon explained the various GA-related issues including concept, history, models, parameters, and examples [24]. GAs are useful tools to efficiently solve optimization problems. In terms of optimization problems, to implement mutation with a reasonable mutation probability is important because the mutation probability directly affects the quality of the derived solution. Wirsansky introduced key characteristics of GAs and distributed evolutionary algorithms in a Python (DEAP) framework [25]. The DEAP framework supported the rapid development of solutions using GAs as well as other evolutionary computation techniques. The author also explained the implementation of reinforcement learning with GAs.

Genetic-algorithm-based adaptive offloading (GA-OA) was proposed for effective traffic handling in an IoT–infrastructure–cloud environment [26]. In this system, the traffic handling was carried out by two methods: from IoT devices to the infrastructure and from the infrastructure to the cloud. The fitness function was designed by considering the capabilities of the two layers. This dual-layer-based approach was similar to our system model. However, we considered the routing costs among the drones in the first layer and extended the lifetime of the drone networks.

Liao et al. proposed an adaptive offloading scheme based on GA for mobile edge computing in ultra-dense cellular networks [27]. The authors proposed two phases for selecting a server and deciding to offload tasks. In the first phase, the mobile devices and servers form groups by preference, considering distance and workload. In the second phase, a binary-coded genetic algorithm was used to solve the 0–1 selection problems for identifying a near-optimal offloading decision when matching multiple mobile users to an edge server. This scheme focused on reducing the offloading latency and energy consumption of mobile devices in ultra-dense cellular networks. In our approach, however, we assumed that a drone could not directly communicate with an edge server because direct communication consumes more energy than multi-hop communications and a drone with a short communication range cannot always reach an edge server in sparse environments.

Wang proposed a drone-assisted offloading scheme using an improved GA for mobile edge computing [28]. Drones that received an offloading task from a mobile device and transmitted the offloading task to the base station acted as intermediaries in order to extend the base station's coverage for offloading. An improved GA was used to minimize the delay and energy consumption in this collaborative task-offloading model. Similarly, we also used a GA to minimize the entire system's delay and energy in our proposed scheme. However, our method more effectively distributed and offloaded the tasks the drones were required to handle. In addition, we considered the balanced energy consumption of the drones in our offloading system.

Zhu et al. used a joint optimization method based on a GA to minimize user task completion time [29]. The authors designed individual genes that considered the user's offloading proportion, bandwidth, and computing resources. This system model assumed that mobile devices could directly access the base station, but in contrast, our system consisted of three layers: UAV-based computing area, edge computing area, and cloud computing area. In our scheme, we considered the routing costs and the balance of residual energy among the drones as well as the factors mentioned in Zhu's scheme.

A particle swarm optimization genetic algorithm with a greedy algorithm (PSO-GA-G) was proposed to reduce the response time of offloading tasks [30]. In this system, randomly distributed mobile devices and a UAV were combined as a group, and the mobile devices offloaded tasks to the UAV. The GA was used to optimize the UAV deployment, and the greedy algorithm was used to minimize the offloading cost among mobile devices

in the group. In comparison, we focused on maximizing the lifetime of the networks in our approach.

Chakraborty et al. proposed a GA-based optimization technique for offloading decisions that considered task dependencies in a multi-user, multi-channel environment [31]. This technique reduced the energy consumption of mobile devices and the computational delay, despite handling multiple task dependencies. This is an important issue in the offloading decision process because the dependencies of the sub-tasks required by a user's job and the tasks among users affect the task completion time. In our scheme, we also considered the task dependency and fairness in terms of energy consumption for each drone.

Huda et al. investigated a GA-based optimization algorithm for a UAV-assisted offloading system [32]. In this system, UAVs provided a link between ground users and cloud servers. The authors devised a GA-based algorithm to minimize the energy consumption of ground users and UAVs. We also used UAVs as links to edge and cloud servers, but we considered more complex network topologies that affect the energy fairness of UAVs.

3. Computational Offloading Scheme That Considered Energy Consumption Fairness

3.1. Assumptions and Our System Model

Figure 1 shows an example of our system model. Table 1 shows the notations and their descriptions used in this paper to describe our energy consumption fairness-aware computational offloading scheme. As shown in Figure 1, we assumed a UAV–edge–cloud computing environment in our energy consumption fairness-aware computational offloading scheme. The UAV–edge–cloud computing environment comprised a UAV swarm and multiple edge and cloud servers that were designated as the UAV-based computing area, edge computing area, and cloud computing area, respectively. In such an environment, tasks that challenge the available computing capabilities or energy consumption, such as in UAVs, could be offloaded to edge or cloud servers for processing.

Table 1. Notations and their descriptions.

Notations	Descriptions
V	Set of all nodes that could be UAVs, edge servers, and cloud servers
E	Set of all topological network links between nodes
W	Set of all workflows to be allocated
w_i	The i-th workflow that needed to be allocated
T_{w_i}	Set of all tasks in a workflow $w_i \in W$
$RB_{v,v'}$	Remaining network bandwidth of link from v to v'
RC_v	Remaining computational processing rate of the node v
$NB_{t,t'}$	The amount of network bandwidth required between task t and t'
NC_t	The amount of computational processing rate required for task t
e_v	The summation of energy consumption of node v
e_v^{proc}	The amount of energy consumption of node v to process the computation of given tasks
e_v^{recv}	The amount of energy consumed to receive data required to perform tasks performed in node v
e_v^{trans}	The amount of energy consumed to transfer the results of given tasks in node v to the next node
e_{total}	Total energy consumption of all nodes in the system
$allocatility(w_i)$	The result of the allocability check for the w_i workflow
$\mathcal{F}$	Energy consumption fairness value
$\mathcal{F}_{threshold}$	A threshold value of energy consumption fairness
$Alloc_{w_i}$	Task allocation information of the workflow i on network topology

Our UAV–edge–cloud computing environment had powerful edge and cloud servers, as compared to the UAV swarm. We assumed that an edge server was equipped with

multiple GPUs. Therefore, an edge server would have a more powerful computational ability when compared with a UAV. It was assumed that a cloud server would have better computing performance than an edge server. For example, the Amazon ECS p4d.24xlarge instance supports 8 GPUs with 320 GiB of memory, 96 CPU cores, and 1152 GiB of main memory. Such a cloud server can provide powerful computing performance. In contrast, edge and cloud servers have more network delays than UAVs, but they provide more powerful computing capabilities, as compared to a single UAV. Each UAV connected through a wireless network connection, such as Wi-Fi, 4G, and 5G network technologies. Furthermore, modern UAVs support Wi-Fi and 5G mobile network technologies to communicate [33,34]. In addition, we assumed that our computing environment had a stable topological network structure, in which no additional UAVs would be joining or leaving. UAV devices, such as drones, that are used in UAV–edge–cloud computing environments typically operate on batteries. Hence, energy efficiency is a critical factor. Therefore, it was necessary to consider the computing capability, network delays, and energy efficiency of the UAVs, the edge server, and the cloud server, as assumed previously, and allocate each task accordingly.

We defined an application or service that could operate in a UAV–edge–cloud computing environment as a workflow. It was assumed that multiple workflows would be allocated and performed in the UAV–edge–cloud computing environment. W notates the set of all workflows to be allocated. We assumed that a workflow consisted of multiple tasks. We assumed that streaming data were continuously being transferred and processed between tasks. Each task had logical precedence, and each task could be allocated to one of the UAVs, edge servers, or cloud servers according to their logical precedence and the existing network topology. We modeled and abstracted our UAV–edge–cloud computing environment as a graph-based topological structure $G = (V, E)$. In our graph-based topological structural model, V denotes the set of all nodes that could be UAVs, edge servers, or cloud servers in our UAV–edge–cloud computing environment. E denotes the set of all topological network links between two nodes.

3.2. Proposed Computational Offloading Algorithm

Workloads could be allocated and executed on each node in the UAV–edge–cloud computing environment. Each node consumed a different amount of energy according to the given tasks in a workflow. As more tasks were allocated to a specific node, the energy consumption of that node increased, as compared to the other nodes. Therefore, it was necessary to evenly allocate the tasks to each node in order to balance the energy consumption. By allocating and offloading tasks to evenly balance the energy consumed by each node, we were able to increase the lifetime of the entire network.

The energy consumption of a certain node v was calculated by Equation (1). e_v^{proc} denotes the amount of energy consumed by the node v to process the computation of its given tasks. e_v^{recv} denotes the amount of energy consumed to receive the data required to perform tasks by node v. e_v^{trans} denotes the amount of energy consumed to transfer the results of the given tasks in node v to the next node. For the energy consumption model of each network communication technique, the energy consumption model proposed by Huang et al. [35] and Dusza et al. [13] were used. The total summation of each node's energy consumption were calculated by Equation (2), where n is the total number of nodes used for the task allocation. The energy consumption fairness ($\mathcal{F}$) was calculated using Equation (3). We modified the fairness index proposed by Gian et al. [36]. If the energy consumption of each node was equal, the network lifetime of the entire network could be extended by reducing problems, such as network partitioning. If the energy consumption of all nodes was the same, the energy consumption fairness value was 1. A value closer 1 for the energy consumption fairness indicated better performance. The goal of the proposed computational offloading scheme was to increase the fairness of the energy consumption. By increasing the energy consumption fairness, each node's energy consumption could be

increased, as well. As a result, the drone-based service could operate for a longer period of time with better stability.

$$e_v = e_v^{recv} + e_v^{proc} + e_v^{trans} \tag{1}$$

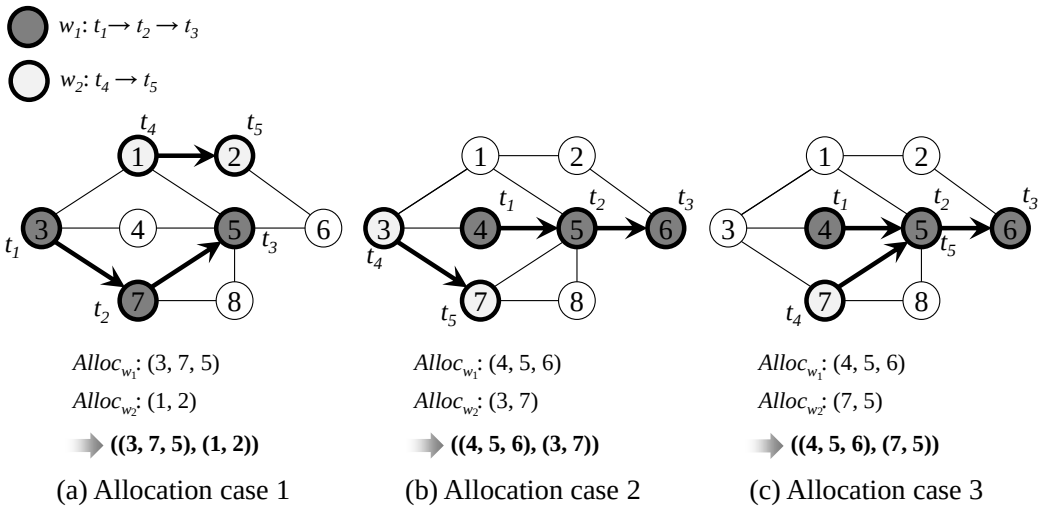
$$e_{total} = \sum_{i=0}^{n} e_i \tag{2}$$

$$\mathcal{F} = \frac{(e_1 + e_2 + \ldots + e_n)^2}{n \times (e_1^2 + e_2^2 + \ldots + e_n^2)} = \frac{{e_{total}}^2}{n \times (e_1^2 + e_2^2 + \ldots + e_n^2)} \tag{3}$$

Algorithm 1 shows our proposed computational offloading and workflow allocation algorithm based on a GA (genetic algorithm). The first line of the algorithm summarizes the initial population generation process. The initial population generation process is described in more detail, as follows. When generating the initial population, each workflow checked the allocability in a random manner. The allocability of w_i was calculated by Equation (4). As shown in Equation (4), if all task $t \in T_{w_i}$ satisfied the conditions $\exists\ NC_t \leq RC_v$ and $\exists\ NB_{t,t'} \leq RB_{v,v'}$, where v and $v' \in V$, then w_i was allocated to the network topology. Therefore, in this case, the value of $allocatility(w_i)$ was set to 1, as otherwise, it was set to 0. This review of allocability was repeated for all given workflows. A workflow that did not satisfy the conditions was not allocated to the network topology. Then, we acquired the $Alloc_{w_i}$ information for all workflows in W. Depending on the remaining processing rate of each node and the remaining bandwidth between nodes, which constituted the network topology, all workflows were not necessarily able to be allocated to the network topology.

$$allocatility(w_i) = \begin{cases} 1 & \text{If all task } t \in T_{w_i}, \exists NC_t \leq RC_v \text{ and } \exists NB_{t,t'} \leq RB_{v,v'} \\ & \text{where } vs. \text{ and } v' \in V \\ 0 & \text{Otherwise} \end{cases} \tag{4}$$

Each workflow allocation information was combined to create a chromosome. Each chromosome was considered as one of the candidates that could allocate given workflows onto the network topology. Figure 3 shows an example of three chromosomes for a given network topology and two workflows. In Figure 3, w_1 consists of three tasks, t_1, t_2, and t_3. w_2 consists of two tasks, t_4 and t_5. $Alloc_{w_1}$ and $Alloc_{w_2}$ represent the allocated nodes of the tasks in w_1 and w_2, respectively. Furthermore, a chromosome was a concatenation of $Alloc_{w_1}$ and $Alloc_{w_2}$. For example, a chromosome is represented as ((3, 7, 5), (1, 2)) in Figure 3a.



(a) Allocation case 1 (b) Allocation case 2 (c) Allocation case 3

Figure 3. An example of chromosomes in the proposed algorithm.

Algorithm 1: Proposed computational offloading and workflow allocation algorithm while considering energy consumption fairness

Result: Best allocation result for given workflows while considering energy consumption fairness

Generate initial population ;
$n = 0$;
/* $Iter_{max}$ is the maximum iteration number */
while $i \leq Iter_{max}$ **do**
    Select two chromosomes from current population ;
    Do crossover operation by using the two selected chromosomes ;
    Calculate $\mathcal{F}_{new}$ of the generated new chromosome ;
    if $\mathcal{F}_{new} \geq \mathcal{F}_{parent1}$ *or* $\mathcal{F}_{new} \geq \mathcal{F}_{parent2}$ **then**
        Remove the one chromosome which has minimum $\mathcal{F}$ from the current population ;
        Add the generated chromosome to the population ;
        if $\mathcal{F}_{new} \geq \mathcal{F}_{threshold}$ **then**
            break ;
        end
    end
    Select a chromosome from the current population ;
    Do mutation operation to the selected chromosome ;
    $i = i + 1$;
end
return Best chromosome which has maximum $\mathcal{F}$;

As shown in Algorithm 1, the crossover operation was repeated to create a new chromosome after generating the initial population. $\mathcal{F}_{parent1}$ and $\mathcal{F}_{parent2}$ denote the energy consumption fairness values of two selected chromosomes to form a new chromosome. Figure 4 shows the concept of the crossover operation in the proposed algorithm. Figure 4 assumes that the total number of given workflows is n. As shown in Figure 4, two different chromosomes were selected from the current population. Then, a new chromosome was generated by using two parent chromosomes. As shown in Figure 4, the proposed method used the one-point crossover method. The crossover point was selected in a random manner. Our algorithm calculated $\mathcal{F}_{new}$, which denotes the energy consumption fairness value of the generated new chromosome. If $\mathcal{F}_{new} \geq \mathcal{F}_{parent1}$ or $\mathcal{F}_{new} \geq \mathcal{F}_{parent2}$, then it removed the one chromosome that had the minimum $\mathcal{F}$ value from the current population and added the newly generated chromosome to the population. If $\mathcal{F}_{new} \geq \mathcal{F}_{threshold}$, then the proposed algorithm terminated immediately and returned the best chromosome information that had the maximum $\mathcal{F}$ value. Finally, the acquired chromosomal information was used to allocate and perform a given workflow in the network topology.

Our final goal is expressed in Equation (5). As shown in Algorithm 1, our goal had been to find a solution that had a maximum $\mathcal{F}_{max}$ for computational offloading and workflow allocation that maximized energy consumption fairness $\mathcal{F}$ among candidate allocation solutions, as expressed in Equation (5). $\mathcal{F}_{max}$ represents the maximum value of the energy consumption fairness value among candidate solutions of computational offloading and workflow allocation. S_i denotes the i-th solution that can allocate the given workflows and their tasks. $\mathcal{F}(S_i)$ represents the energy consumption fairness value when the given workflows and their tasks are allocated using the solution of S_i. $\mathcal{F}(S_i)$ was calculated using Equation (3).

$$\mathcal{F}_{max} = max(\mathcal{F}(S_1), \mathcal{F}(S_2), \ldots, \mathcal{F}(S_n)), \text{where } \mathcal{F}_{max} \geq \mathcal{F}_{threshold} \tag{5}$$

We also used a mutation operation to avoid becoming trapped in a local minimum. Figure 5 shows an example of a mutation operation that was applied in our GA-based com-

putational offloading and workflow allocation scheme. The mutation operation described in lines 14 and 15 of Algorithm 1 selected a random chromosome from the population. It randomly transformed the workflow allocation information of the selected chromosome to initiate a mutation in the chromosome. This process was carried out by randomly changing the nodes to which a task had been allocated.

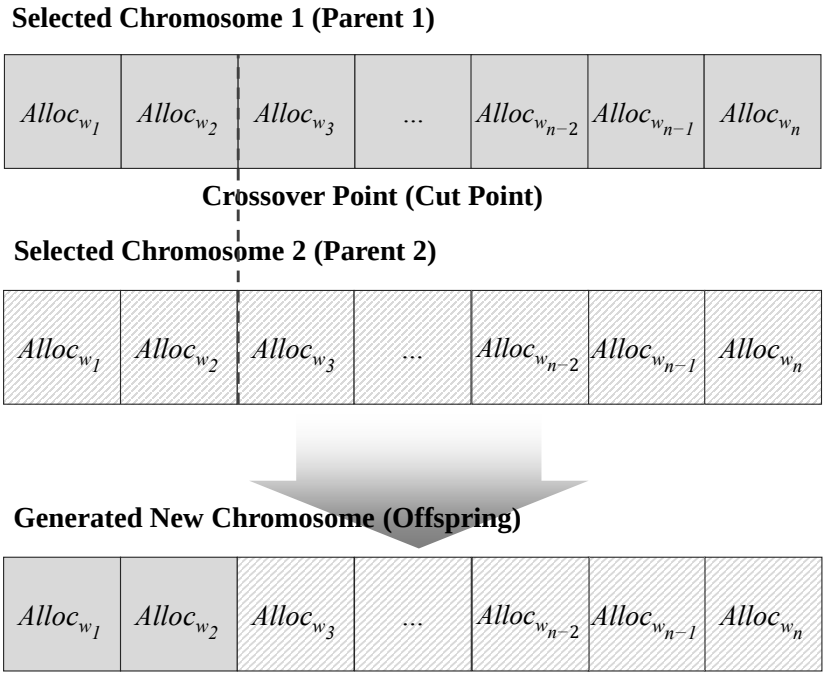


Figure 4. A concept of the crossover operation in the proposed algorithm.

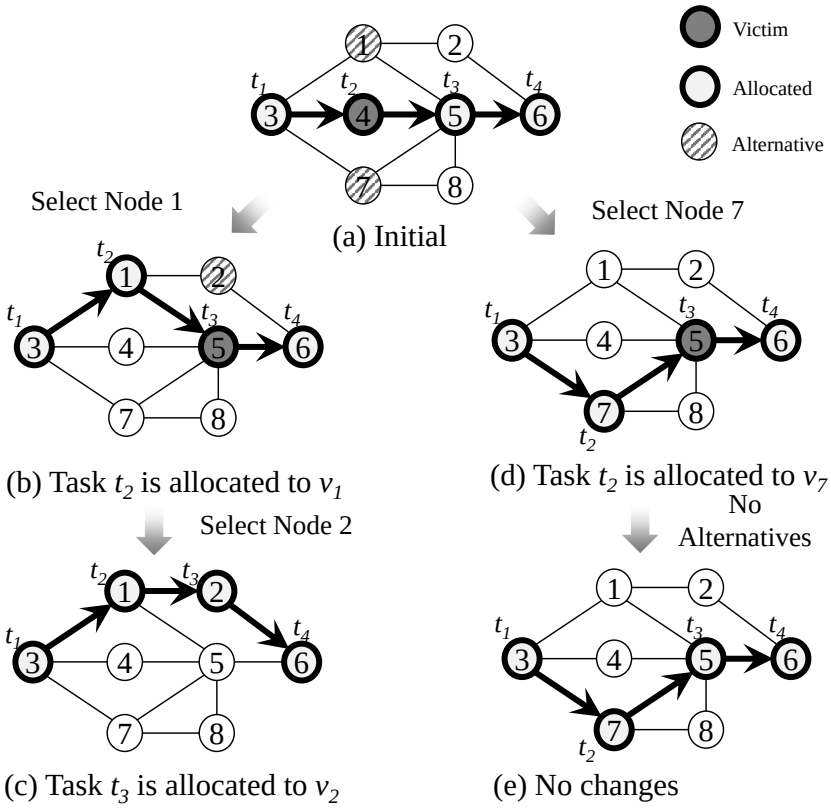


Figure 5. An example of mutation operation.

In Algorithm 1, the mutation operation selected a random chromosome from the current population and randomly changed the task-to-node allocation information of the chromosome. The mutation operation performed the process for each workflow $w \in W$, as described below:

For every 3-tuple of tasks t_i, t_j, $t_k \in w$ such that t_j depends on t_i and t_k depends on t_j, and nodes v_i, v_j, $v_k \in V$ are allocated to t_i, t_j, t_k, respectively, in the allocation information of the chromosome, a random node v_l is searched to perform t_j instead of v_j. The remaining resources of v_l must be enough to perform t_j. In addition, v_i and v_k must be in the transmission range of v_l. Then, the task-to-node allocation from (t_i, t_j, t_k) to (v_i, v_j, v_k) is changed from (t_i, t_j, t_k) to (v_i, v_l, v_k) in the allocation information of the chromosome. If no node can replace v_j because of the resource constraints (such as processing power, network bandwidth, remaining energy) or transmission range, the allocation information is not changed.

The mutation operation changed the overall allocation information in the selected chromosome by repeating the process above for all workflows. Figure 5 shows an example of the process for the workflow $w \in W$. To simplify the explanation, assume that all the nodes v_1, v_2, ..., v_8 in the figure have infinite resources. Figure 5a is the allocation information of w before a mutation operation. w contains four tasks t_1, t_2, t_3, and t_4. In w, t_2 depends on t_1; t_3 depends on t_2, and t_4 depends on t_3. The tasks, t_1, t_2, t_3, and t_4, are allocated to nodes v_3, v_4, v_5, and v_6, respectively.

First, alternative nodes are searched for (t_1, t_2, t_3). The alternatives that can perform t_2 instead of v_4 are v_1 and v_7, in Figure 5a. Figure 5b shows the case in which v_1 is selected to replace v_4 and Figure 5c shows the case in which v_7 is selected. In the case of Figure 5b, task t_2 is allocated to v_1, instead of v_4. Then, alternative nodes are searched to perform t_3, instead of v_5, for (t_2, t_3, t_4). In this case, t_3 is allocated to v_2, instead of v_5, because v_2 is the only alternative node. To continue, t_1, t_2, t_3, and t_4 are allocated to v_3, v_1, v_2, and v_6, respectively, instead of v_3, v_4, v_5, and v_6. This is shown in Figure 5c. Figure 5d shows the case where v_7 is selected as the alternative of v_4 from Figure 5a. In Figure 5d, t_2 is allocated to v_7, instead of v_4. Then, the allocation of t_3 is not changed because no alternative node of v_5 is found. Figure 5e shows the final state after the mutation to w in this case. By using this mutation operation, our GA-based computational offloading and workflow allocation algorithm could explore more diverse allocation solutions and avoid becoming trapped in the local minimum.

4. Performance Evaluations

4.1. Simulation Environments

We evaluated the performance of the proposed scheme by simulation. Table 2 presents the parameters and their values in the simulation. The parameters used in the simulation were divided into two types. The first type was the common parameters, and the other was the scheme-dependent parameters. For the common parameters, the computing performance and network bandwidth of the UAV, edge server, and cloud server were similar after taking into consideration the experimental environment values used in Liu et al.'s paper. For the scheme-dependent parameters, suitable hyperparameters were derived by considering the overall performance and execution time of each scheme from prior experiments, and the values were set accordingly.

As shown in Table 2, the performance in terms of the energy consumption fairness, total energy consumption, and average distance were analyzed by simulation for three network topology sizes: small, medium, and large.The proposed scheme was compared with Liu et al.'s scheme [3,5]. Liu et al.'s scheme had created a computational offloading policy based on Markov approximation. (Hereafter, Liu et al.'s Markov approximation-based scheme is referred to as MA-based, and our proposed scheme is referred to as the GA-based scheme.) In the simulation, each experiment was performed 100 times, and the results were averaged. In order to reduce the variability of the results, we conducted the experiment 100 times, taking into account the randomness of the small, medium, and large

network topologies. In the simulation, the energy consumption models were based on studies by Huang et al. [35] and Dusza et al. [37]. Because the energy consumption values for Wi-Fi and LTE communication have been well modeled, these were suitable for the proposed drone-based computing environment. The number of transferred bits was proportional to the network bandwidth requirements in Table 2, and the energy consumption per bit, according to transmission distances, was approximated from [35,37].

Table 2. Parameters and their values in the simulation.

Common Parameters	**Topology**		
	Small	**Medium**	**Large**
Field size (m)	50 × 50	100 × 100	200 × 200
Drone's transmission range (m)	10	30	50
Number of drones	10	30	100
Number of edge servers	4	4	4
Number of cloud servers	2	2	2
Processing rate of a drone		100	
Processing rate of an edge server		500	
Processing rate of a cloud server		10,000	
Network bandwidth of a drone (Mb/s)		200	
Network bandwidth of an edge server (Mb/s)		400	
Network bandwidth of a cloud server (Mb/s)		1000	
Number of workflows	4	20	30
Number of tasks per workflow	4	4	4–6
Processing rate required for each task	40–80	20–30	20–30
Network bandwidth required for each task (Mb/s)	20–30	20–30	20–30
Scheme-Dependent Parameters	Topology		
	Small	Medium	Large
Number of chromosomes in a population (GA-based)		10,000	
Number of iterations (GA-based)	100,000	1,000,000	1,000,000
Number of iterations (MA-based)		2000	

When measuring the energy fairness in the performance evaluations, the energy consumption fairness, total energy consumption, and average distance were only calculated for drones used to allocate tasks for each workflow. This was because it had been assumed that edge servers and cloud servers were always operated at full power. Therefore, the energy efficiency and consumption rates were not critical for edge and cloud servers.

The proposed scheme and the MA-based scheme were implemented in Python scripts. A prototype was initially written in C language, but it was later ported to Python due to extensibility concerns. The final Python scripts consisted of 17 files, with a total size of 56 KB and 1600 lines of code. The Matplotlib library was used for visualization. The simulations were conducted on a desktop computer, and the runtime environment is detailed in Table 3. Based on the environment shown in Table 3, the approximate runtimes for each trial of the proposed scheme were between 30 min and 1 h, while those of the MA-based scheme were between 3 and 5 h. The exact runtimes were not thoroughly investigated, but we assumed that the amount of memory usage had been the primary factor affecting runtime, rather than the processing load. Note that the execution time was not our concern. Our implementation was single-threaded, and the runtime could be significantly reduced if multi-threading techniques or GPU capabilities were utilized, or if the NumPy library was used.

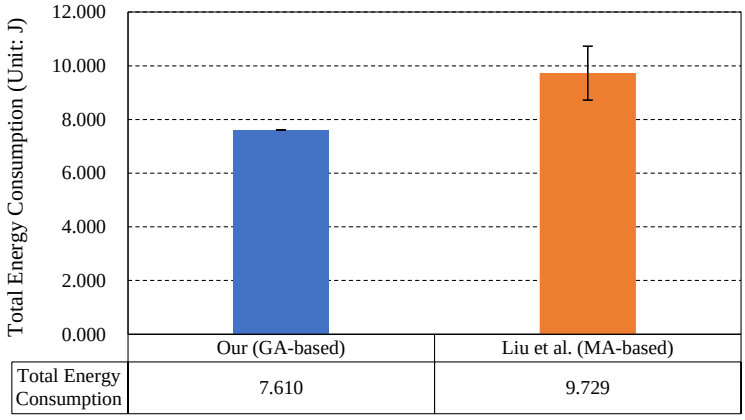
Table 3. Runtime environment of the simulations.

Component	Specification
CPU	Intel Core i7 13,700 K
Main memory	32 GB
Interpreter	Python 3.8
Library	Matplotlib 3.4.2

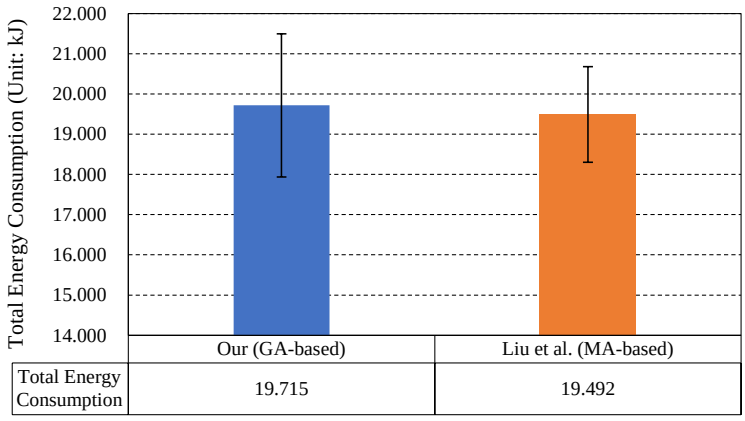
4.2. Simulation Results and Discussion

4.2.1. Comparison Results of the Total Energy Consumption

Figure 6 shows the comparison results of the total energy consumption of the proposed scheme and the MA-based scheme. As shown in Figure 6a, the total energy consumption of the proposed scheme was 7.610 *J*, and the total energy consumption of the MA-based scheme was 9.729 *J*. In the case of a small network topology, our scheme performed better in terms of total energy consumption. As shown in Figure 6b,c, our scheme consumed more energy, as compared to the MA-based scheme. However, the difference in energy consumption was not significant. Briefly, for the proposed scheme, the energy consumption increased by approximately 1.14% while the MA-based scheme increased by approximately 3.06%.

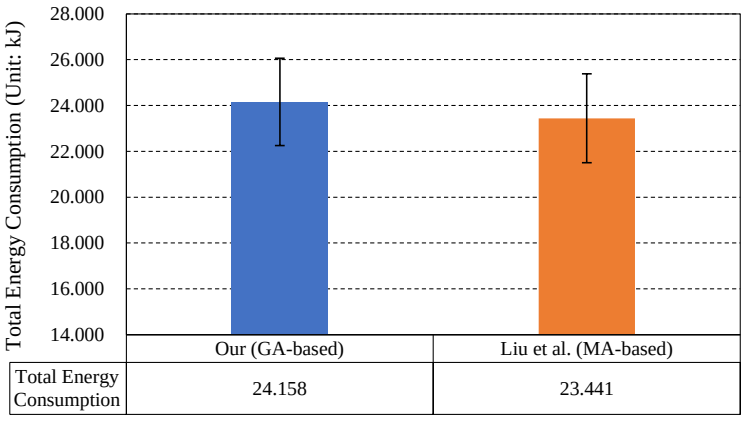


(**a**) Small Network Topology



(**b**) Medium Network Topology

Figure 6. *Cont.*



(**c**) Large Network Topology

Figure 6. Simulation results of the total energy consumption.

4.2.2. Comparison Results of the Energy Consumption Fairness

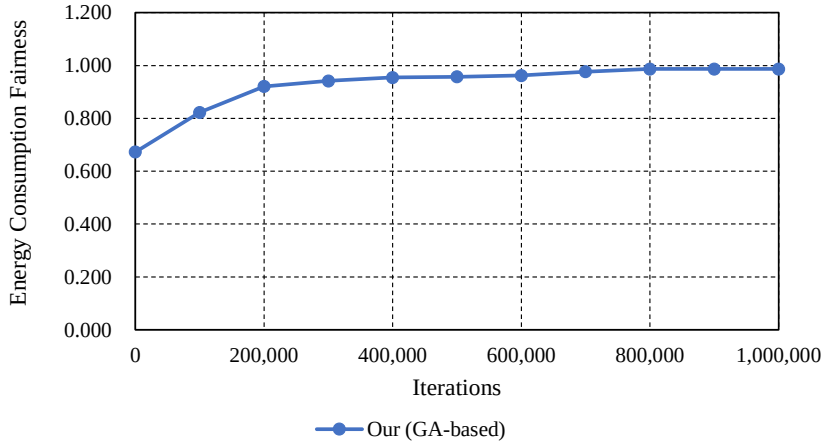
Figure 7a,b show the convergence of the energy consumption fairness in the large network topology using the proposed scheme and the MA-based scheme, respectively. In this example, the initial energy consumption fairness of the proposed scheme was 0.67, which reached 0.82 after 100,000 iterations and further increased to 0.95 after 1,000,000 iterations. In multiple simulations, it was observed that the proposed scheme achieved 95% of the energy consumption fairness after 400,000 iterations, as compared to 1,000,000 iterations. The improvement in the energy consumption fairness did not occur linearly after 40,000 iterations, but in every trial, several small improvements in a step-wise manner were observed as a result of the mutation operation. As a comparison, the MA-based scheme was not improved in terms of energy consumption fairness; instead, total energy consumption was improved. However, the MA-based scheme often converged to a local optimum, resulting in a sub-optimal solution for the total energy consumption.

Figure 8 shows the simulation results of the energy consumption fairness of the proposed scheme and the MA-based scheme. If the energy consumption of all nodes was the same, the energy consumption fairness value was 1. Therefore, as the energy consumption fairness value neared 1, the performance improved. If the energy consumption of all nodes was the same, the network lifetime could be increased without any problems, such as network partitioning.

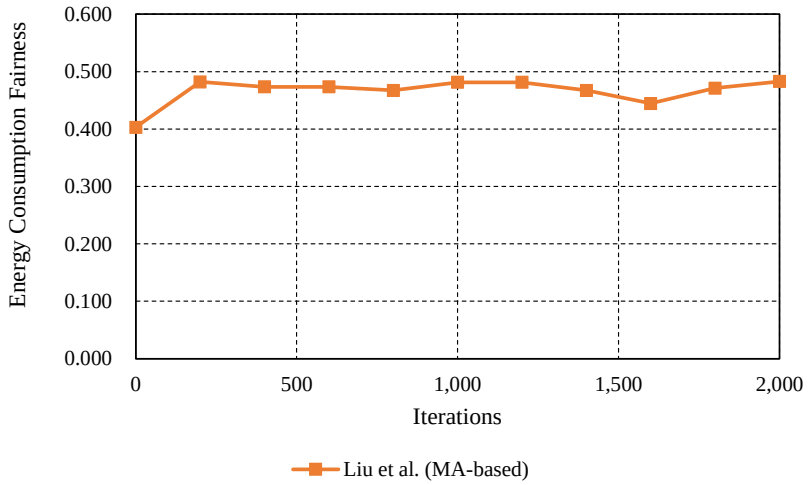
As shown in Figure 8, the proposed scheme outperformed the MA-based scheme in energy consumption fairness. As shown in Figure 8a, in the case of the small network topology, the energy consumption fairness of the proposed scheme was approximately 17.09% higher than the MA-based scheme. In the case of the medium network topology, as shown in Figure 8b, the energy consumption fairness significantly increased, from 0.415 to 0.917, with the proposed scheme. Similarly, even in the case of the large network topology, the proposed scheme outperformed the MA-based scheme in energy consumption fairness. As shown in Figure 8c, the energy consumption fairness significantly increased, from 0.506 to 0.809, with the proposed scheme.

Using the proposed GA-based scheme, the energy consumption fairness was reduced when applied to a large network topology, as compared to a small network topology. This result was due to the different values of simulation parameters that were applied according to each network topology. For a large network topology, the number of workflows to be allocated was larger than in a small network topology, as shown in Table 2. The average number of tasks allocated to each drone increased, and the degree of imbalance in the energy consumption of each drone was greater than in the small network topology. In addition, as the size of the network topology increased, so did the number of nodes in the network topology increased; thus, the size of the problem space explored by the proposed GA-based scheme also increased. Therefore, more various allocation solutions were possible, and it

was more difficult to define the global optimum for a large network topology. For these reasons, the energy consumption fairness was slightly reduced.



(**a**) The proposed scheme

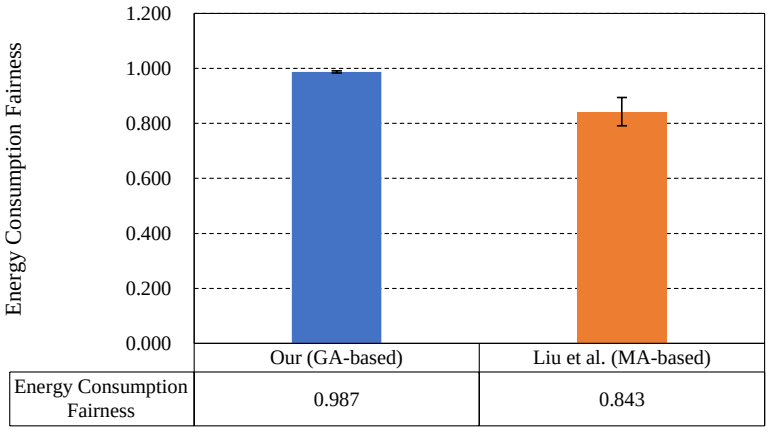


(**b**) Liu et al.'s scheme

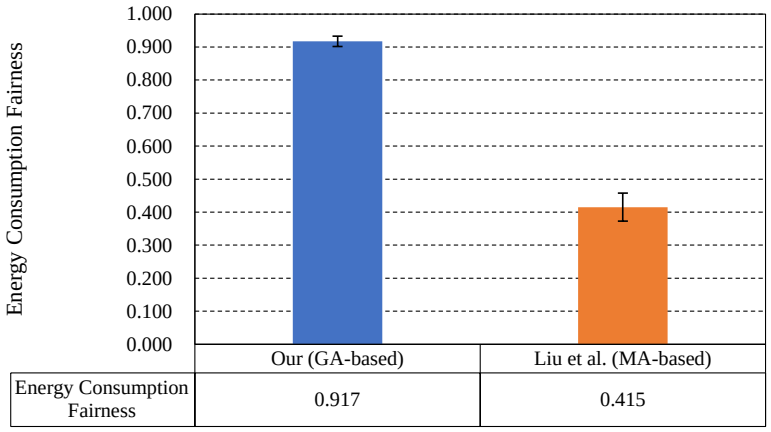
Figure 7. The convergence of the energy consumption fairness.

The proposed method also performed better than the MA-based scheme in terms of its standard deviation. Based on the simulation results, we confirmed that the proposed scheme could significantly increase the fairness of the energy consumption and the lifetime of the entire network by balancing the energy usage of the nodes allocated tasks.

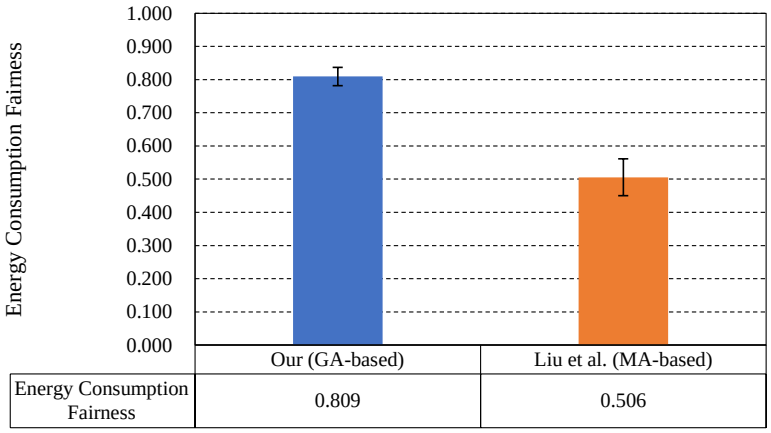
Overall, the proposed GA-based scheme showed better performance than the MA-based scheme in terms of the energy consumption fairness. These results were because the proposed GA-based scheme could explore more diverse allocation solutions in the problem space, as compared to the MA-based scheme. Therefore, the GA-based proposed scheme could identify an allocation solution that was closer to the optimal global solution. For a large network, this potential was larger, and the GA-based scheme showed better performance in balancing energy consumption.



(**a**) Small Network Topology



(**b**) Medium Network Topology



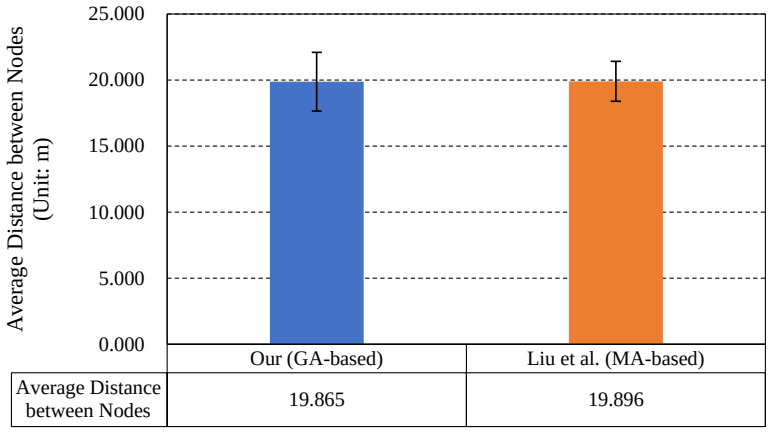
(**c**) Large Network Topology

Figure 8. Simulation results of the energy consumption fairness.

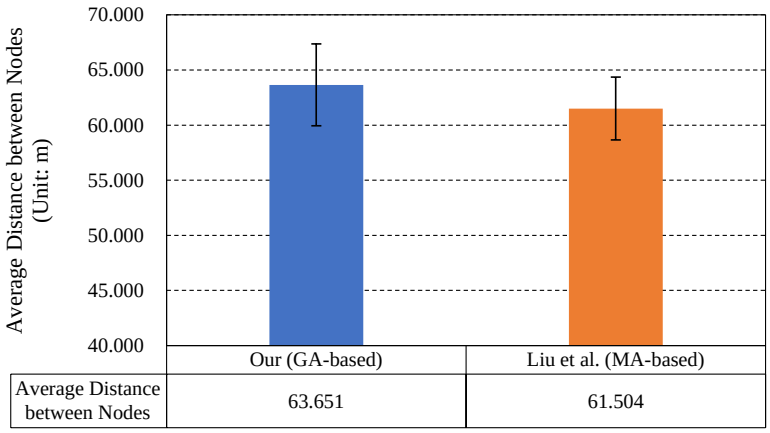
4.2.3. Comparison Results of the Average Distance between the Nodes

Figure 9 shows the simulation results of the average distance between the nodes that were allocated tasks. As the average distance between the nodes allocated to tasks increased, additional time could be required for transferring data between the nodes. Therefore, it could increase the overall latency or the energy consumption required to transmit data. As shown in Figure 9, in the case of the small network topology, the difference in the average distance between the nodes that had been allocated tasks was not significant. In the case of the medium network topology, the average distance of the proposed scheme only

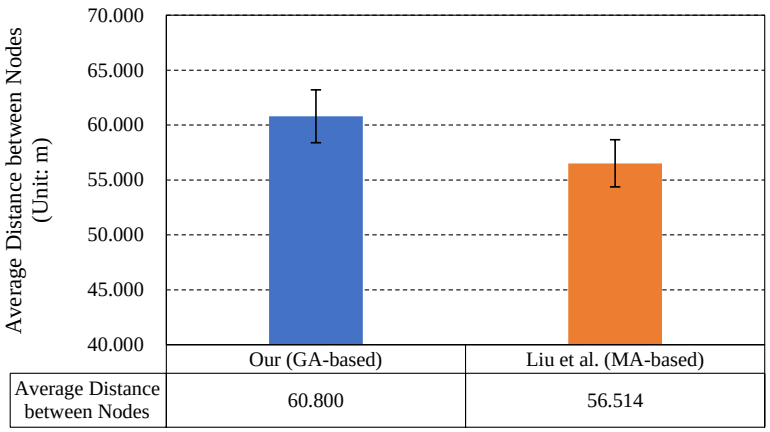
increased by approximately 3.50%, as compared to the MA-based scheme. In the case of the large network topology, the average distance of the proposed scheme increased by about 7.58%, as compared to the MA-based scheme. However, it showed that the increase in the average distance between the nodes was not large, and the energy consumption fairness was significantly improved with the proposed scheme. To summarize, the proposed GA-based scheme improved the average distance between nodes allocated to tasks and the energy consumption fairness without significantly increasing the energy consumption, as compared to the MA-based scheme.



(**a**) Small Network Topology



(**b**) Medium Network Topology

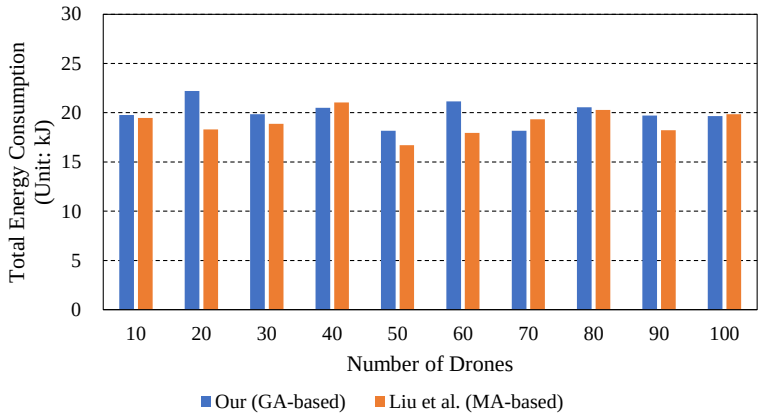


(**c**) Large Network Topology

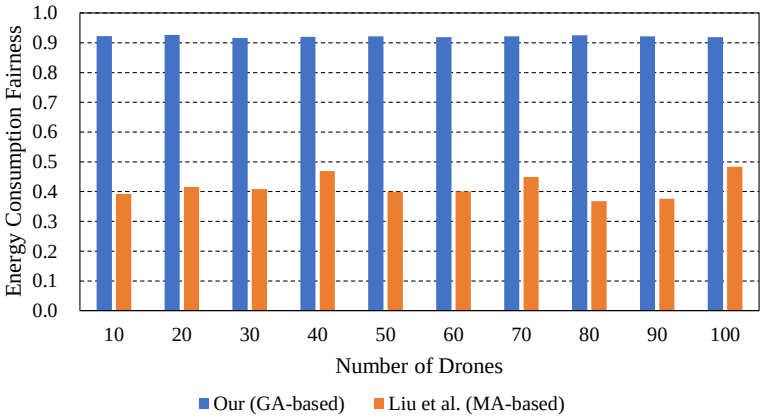
Figure 9. Simulation results of the average distance between nodes.

4.2.4. Comparison Results According to the Number of Drones

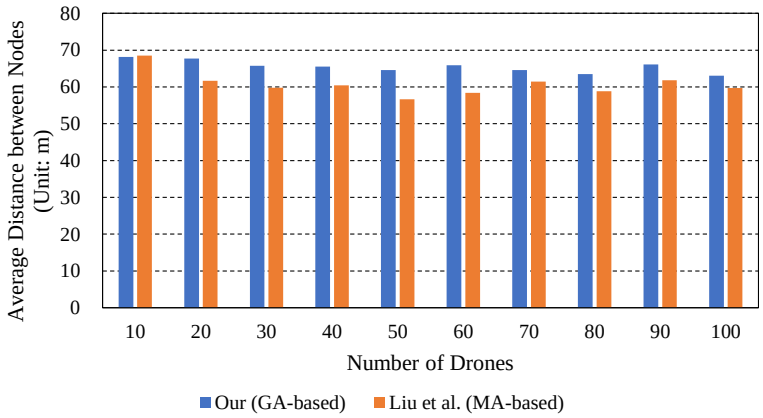
Figure 10 shows the simulation results according to the number of drones in the medium network topology. The three sub-figures, Figure 10a–c, show the total energy consumption, the energy consumption fairness, and the average distance between the nodes, respectively. The results indicated that the number of drones in the fixed-sized network did not have a significant impact on the total energy consumption and the energy consumption fairness. However, the average distance between nodes decreased, as the number of drones increased.



(**a**) Total Energy Consumption



(**b**) Energy Consumption Fairness



(**c**) Average Distance between Nodes

Figure 10. Simulation results according to the number of drones.

As the number of drones increased from 10 to 100, the average distance between the nodes in the proposed scheme decreased from 68.19 to to 63.02, while the the average distance between the nodes in the MA-based scheme decreased from 68.53 to 59.74. This was because both the cost functions of the proposed scheme and the MA-based scheme were attempting to reduce the energy consumption by reducing the transmission range between nodes. However, the decrement in the average distance between the nodes did not result in the decrement of the total energy consumption, as the energy consumption models of Huang et al. [35] and Dusza et al. [37] are not linear. In the energy consumption model, the energy consumption remained almost constant for short transmission ranges and only increased when the transmission range had exceeded a certain threshold.

5. Conclusions and Future Works

This paper proposed a computational offloading and workflow allocation scheme that considered the energy consumption fairness in UAV–edge–cloud computing environments. The proposed scheme used a genetic algorithm. The performance of the proposed scheme was verified through simulations, and it was shown that the performance of the proposed scheme was superior to that of Liu et al.'s Markov approximation-based scheme, in terms of energy consumption fairness. The energy consumption fairness improved by up to approximately 120%, as compared to Liu et al.'s Markov approximation-based scheme. In terms of the total energy consumption, the increase in energy consumption of the proposed method was not significant, as compared to Liu et al.'s scheme. The proposed scheme consumed up to approximately 3% more energy in total, as compared to Liu et al.'s scheme. Similarly, the average distance of the nodes allocated to tasks could increase slightly, but the increase was not significant. The average distance of nodes allocated to tasks in the proposed scheme increased by approximately 8%, as compared to Liu et al.'s scheme. In summary, the proposed scheme could significantly improve the fairness of energy consumption without significantly increasing the energy consumption or the average distance between nodes allocated to tasks.

In future work, we plan to improve the proposed scheme for efficiently allocating and offloading tasks by comparing and analyzing performance, particularly in cases where a single workflow is composed of multiple parallel tasks.

Author Contributions: Conceptualization, B.K., J.J. (Joonhyouk Jang) and H.M.; methodology, B.K. and J.H. (Jungkyu Han); software, B.K. and J.J. (Joonhyouk Jang); validation, J.J. (Jinman Jung), J.H. (Jungkyu Han) and J.H. (Junyoung Heo); formal analysis, B.K., J.J. (Joonhyouk Jang) and J.H. (Jungkyu Han); investigation, B.K. and J.J. (Joonhyouk Jang); resources, J.H. (Jungkyu Han), H.M. and J.J. (Jinman Jung); data curation, B.K. and J.J. (Joonhyouk Jang); writing—original draft preparation, B.K., J.J. (Joonhyouk Jang) and H.M.; writing—review and editing, B.K., H.M. and J.J. (Jinman Jung); visualization, B.K. and J.J. (Joonhyouk Jang); supervision, H.M. and J.J. (Jinman Jung); project administration, J.H. (Junyoung Heo); funding acquisition, J.H. (Junyoung Heo). All authors have read and agreed to the published version of the manuscript.

Funding: This research was financially supported by Hansung University.

Institutional Review Board Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Balamuralidhar, N.; Tilon, S.; Nex, F. MultEYE: Monitoring System for Real-Time Vehicle Detection, Tracking and Speed Estimation from UAV Imagery on Edge-Computing Platforms. *Remote Sens.* **2021**, *13*, 573. [CrossRef]
2. Kim, B.; Jung, J.; Min, H.; Heo, J. Energy Efficient and Real-Time Remote Sensing in AI-Powered Drone. *Mob. Inf. Syst.* **2021**, *2021*, 6650053. [CrossRef]
3. Liu, B.; Zhang, W.; Chen, W.; Huang, H.; Guo, S. Online Computation Offloading and Traffic Routing for UAV Swarms in Edge-Cloud Computing. *IEEE Trans. Veh. Technol.* **2020**, *69*, 8777–8791. [CrossRef]

4. Kim, B.; Min, H.; Jang, J.; Jung, J.; Han, J.; Heo, J. Computation Offloading Scheme Considering Energy Consumption Fairness for UAV–edge–cloud Computing Environments. In Proceedings of the the 10th International Conference on Smart Media and Applications, Gunsan-si, Republic of Korea, 9–11 September 2021; pp. 60–63.
5. Liu, B.; Huang, H.; Guo, S.; Chen, W.; Zheng, Z. Joint Computation Offloading and Routing Optimization for UAV–edge–cloud Computing Environments. In Proceedings of the 2018 IEEE SmartWorld, Ubiquitous Intelligence Computing, Advanced Trusted Computing, Scalable Computing Communications, Cloud Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI), Guangzhou, China, 8–12 October 2018; pp. 1745–1752. [CrossRef]
6. King, A. *Programming the Internet of Things: An Introduction to Building Integrated, Device-to-Cloud IoT Solutions*; O'Reilly Media: Sebastopol, CA, USA, 2021.
7. Serpanos, D. *Internet-of-Things (IoT) Systems Architectures, Algorithms, Methodologies*; Springer: Cham, Switzerland, 2018.
8. Cheng, N.; Lyu, F.; Quan, W.; Zhou, C.; He, H.; Shi, W.; Shen, X. Space/Aerial-Assisted Computing Offloading for IoT Applications: A Learning-Based Approach. *IEEE J. Sel. Areas Commun.* **2019**, *37*, 1117–1129. [CrossRef]
9. Guo, H.; Liu, J. UAV-Enhanced Intelligent Offloading for Internet of Things at the Edge. *IEEE Trans. Ind. Inform.* **2020**, *16*, 2737–2746. [CrossRef]
10. Yu, Z.; Gong, Y.; Gong, S.; Guo, Y. Joint Task Offloading and Resource Allocation in UAV-Enabled Mobile Edge Computing. *IEEE Internet Things J.* **2020**, *7*, 3147–3159. [CrossRef]
11. Li, L.; Wen, X.; Lu, Z.; Jing, W. An Energy Efficient Design of Computation Offloading Enabled by UAV. *Sensors* **2020**, *20*, 3363. [CrossRef]
12. Liu, Y.; Xie, S.; Zhang, Y. Cooperative Offloading and Resource Management for UAV-Enabled Mobile Edge Computing in Power IoT System. *IEEE Trans. Veh. Technol.* **2020**, *69*, 12229–12239. [CrossRef]
13. Dai, M.; Su, Z.; Xu, Q.; Zhang, N. Vehicle Assisted Computing Offloading for Unmanned Aerial Vehicles in Smart City. *IEEE Trans. Intell. Transp. Syst.* **2021**, *22*, 1932–1944. [CrossRef]
14. Alhelaly, S.; Muthanna, A.; Elgendy, I.A. Optimizing Task Offloading Energy in Multi-User Multi-UAV-Enabled Mobile Edge-Cloud Computing Systems. *Appl. Sci.* **2022**, *12*, 6566. [CrossRef]
15. Huang, W.; Guo, H.; Liu, J. Task Offloading in UAV Swarm-Based Edge Computing: Grouping and Role Division. In Proceedings of the 2021 IEEE Global Communications Conference (GLOBECOM), Madrid, Spain, 7–11 December 2021; pp. 1–6. [CrossRef]
16. Messous, M.A.; Senouci, S.M.; Sedjelmaci, H.; Cherkaoui, S. A Game Theory Based Efficient Computation Offloading in an UAV Network. *IEEE Trans. Veh. Technol.* **2019**, *68*, 4964–4974. [CrossRef]
17. Tang, Q.; Chang, L.; Yang, K.; Wang, K.; Wang, J.; Sharma, P.K. Task number maximization offloading strategy seamlessly adapted to UAV scenario. *Comput. Commun.* **2020**, *151*, 19–30. [CrossRef]
18. Ateya, A.A.A.; Muthanna, A.; Kirichek, R.; Hammoudeh, M.; Koucheryavy, A. Energy- and Latency-Aware Hybrid Offloading Algorithm for UAVs. *IEEE Access* **2019**, *7*, 37587–37600. [CrossRef]
19. Bai, T.; Wang, J.; Ren, Y.; Hanzo, L. Energy-Efficient Computation Offloading for Secure UAV-Edge-Computing Systems. *IEEE Trans. Veh. Technol.* **2019**, *68*, 6074–6087. [CrossRef]
20. Chen, J.; Chen, S.; Luo, S.; Wang, Q.; Cao, B.; Li, X. An intelligent task offloading algorithm (iTOA) for UAV edge computing network. *Digit. Commun. Netw.* **2020**, *6*, 433–443. [CrossRef]
21. Zhao, L.; Yang, K.; Tan, Z.; Li, X.; Sharma, S.; Liu, Z. A Novel Cost Optimization Strategy for SDN-Enabled UAV-Assisted Vehicular Computation Offloading. *IEEE Trans. Intell. Transp. Syst.* **2021**, *22*, 3664–3674. [CrossRef]
22. Li, J.; Liu, Q.; Wu, P.; Shu, F.; Jin, S. Task Offloading for UAV-based Mobile Edge Computing via Deep Reinforcement Learning. In Proceedings of the 2018 IEEE/CIC International Conference on Communications in China (ICCC), Beijing, China, 16–18 August 2018; pp. 798–802. [CrossRef]
23. Qi, W.; Sun, H.; Yu, L.; Xiao, S.; Jiang, H. Task Offloading Strategy Based on Mobile Edge Computing in UAV Network. *Entropy* **2022**, *24*, 736. [CrossRef]
24. Simon, D. *Evolutionary Optimization Algorithms*; Wiley: New York, NY, USA, 2013.
25. Wirsansky, E. *Hands-On Genetic Algorithms with Python: Applying Genetic Algorithms to Solve Real-World Deep Learning and Artificial Intelligence Problems*; Packt Publishing: Birmingham, UK, 2020.
26. Hussain, A.; Manikanthan, S.V.; Padmapriya, T.; Nagalingam, M. Genetic algorithm based adaptive offloading for improving IoT device communication efficiency. *Wirel. Netw.* **2019**, *26*, 2329–2338. [CrossRef]
27. Liao, Z.; Peng, J.; Xiong, B.; Huang, J. Adaptive offloading in mobile-edge computing for ultra-dense cellular networks based on genetic algorithm. *J. Cloud Comput.* **2021**, *10*, 1–16. [CrossRef]
28. Wang, H. Collaborative Task Offloading Strategy of UAV Cluster Using Improved Genetic Algorithm in Mobile Edge Computing. *J. Robot.* **2021**, *2021*, 3965689. [CrossRef]
29. Li, Z.; Zhu, Q. Genetic Algorithm-Based Optimization of Offloading and Resource Allocation in Mobile-Edge Computing. *Information* **2020**, *11*, 83. [CrossRef]
30. Chen, Z.; Zheng, H.; Zhang, J.; Zheng, X.; Rong, C. Joint computational offloading and deployment optimization in multi-UAV-enabled MEC systems. *Peer-Peer Netw. Appl.* **2022**, *15*, 194–205. [CrossRef]
31. Chakraborty, S.; Mazumdar, K. Sustainable task offloading decision using genetic algorithm in sensor mobile edge computing. *J. King Saud Univ.-Comput. Inf. Sci.* **2022**, *34*, 1552–1568. [CrossRef]

32. Huda, S.A.; Moh, S. Survey on computational offloading in UAV-Enabled mobile edge computing. *J. Netw. Comput. Appl.* **2022**, *201*, 103341. [CrossRef]
33. Abeywickrama, H.V.; Jayawickrama, B.A.; He, Y.; Dutkiewicz, E. Comprehensive Energy Consumption Model for Unmanned Aerial Vehicles, Based on Empirical Studies of Battery Performance. *IEEE Access* **2018**, *6*, 58383–58394. [CrossRef]
34. Muzaffar, R.; Raffelsberger, C.; Fakhreddine, A.; Luque, J.L.; Emini, D.; Bettstetter, C. First Experiments with a 5G-Connected Drone. In Proceedings of the 6th ACM Workshop on Micro Aerial Vehicle Networks, Systems, and Applications, DroNet '20, Toronto, ON, Canada, 19 June 2020; Association for Computing Machinery: New York, NY, USA, 2020. [CrossRef]
35. Huang, J.; Qian, F.; Gerber, A.; Mao, Z.; Sen, S.; Spatscheck, O. A close examination of performance and power characteristics of 4G LTE networks. In Proceedings of the MobiSys'12—10th International Conference on Mobile Systems, Applications, and Services, Lake District, UK, 25–29 June 2012; pp. 225–238. [CrossRef]
36. Van Giang, D.; Taleb, T.; Hashimoto, K.; Kato, N.; Nemoto, Y. A Fair and Lifetime-Maximum Routing Algorithm for Wireless Sensor Networks. In Proceedings of the IEEE GLOBECOM 2007—IEEE Global Telecommunications Conference, Washington, DC, USA, 26–30 November 2007; pp. 581–585. [CrossRef]
37. Dusza, B.; Ide, C.; Cheng, L.; Wietfeld, C. An accurate measurement-based power consumption model for LTE uplink transmissions. In Proceedings of the 2013 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), Turin, Italy, 14–19 April 2013; pp. 49–50. [CrossRef]