applied sciences

MDPI

Article

Cooperative Computing Offloading Scheme via Artificial Neural Networks for Underwater Sensor Networks

Xin Liu [1], Xiujuan Du [2,*], Shuailiang Zhang [2] and Duoliang Han [2]

[1] Computer Department, Qinghai Minzu University, Xining 810007, China; 2004017@qhmu.edu.cn
[2] Computer Department, Qinghai Normal University, Xining 810008, China; qhnuslz@163.com (S.Z.); xihe0515@126.com (D.H.)
* Correspondence: dxj@qhnu.edu.cn; Tel.: +86-0971-6306211

Abstract: Aiming at the problem of being unable to meet some high computing power, high-precision applications due to the limited capacity of underwater sensor nodes, and the difficulty of low computation power, in this paper, we introduce the edge servers, known as base stations for underwater sensor nodes, and propose a scheme to process the computational tasks based on coalition game theory. This scheme provides functions such as cooperation among different base stations within the coalition, the smart division of tasks, and efficient computational offloading. In order to reduce the complexity of the algorithm, the artificial neural network model is introduced into the method. Each task is divided into sub-parts and fed to an artificial neural network for training, testing, and validation. In addition, the scheme delivers the computed task from base stations back to sink nodes via a shortened path to enhance the service reliability. Due to the mobility of the base station in the ocean, our proposed scheme takes into account the dynamic environment at the same time. The simulation results show that, compared with the existing state-of-the-art methods, the success rate of our proposed approach improves by 30% compared with the Greedy method. The total service time of our proposed approach decreases by 12.6% compared with the Greedy method and 31.2% compared with the Always-Migrate method.

Keywords: underwater sensors network; computational offloading; smart task division; coalition game theory; artificial neural network

check for updates

Citation: Liu, X.; Du, X.; Zhang, S.; Han, D. Cooperative Computing Offloading Scheme via Artificial Neural Networks for Underwater Sensor Networks. *Appl. Sci.* **2023**, *13*, 11886. https://doi.org/10.3390/app132111886

Academic Editor: Alessandro Lo Schiavo

Received: 25 August 2023
Revised: 10 October 2023
Accepted: 19 October 2023
Published: 30 October 2023

Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Recently, research on the Underwater Sensor Network (USNet) has attracted significant attention due to its potential application in marine data collection, pollution monitoring, offshore exploration, disaster prevention, navigation, and military and tactical surveillance. Although USNet is a type of wireless sensor network, it is different from the terrestrial wireless sensor networks in which the sensor nodes rely on radio waves to communicate with one another. The underwater sensor nodes use acoustic channels as the communication medium, with a speed of 1500 m/s, which is five orders of magnitude lower than that of radio signals [1]. USNet is characterized by a narrow bandwidth, long delay, high bit error, and limited energy resources and computing power [2,3], bringing about great challenges to the communication over underwater acoustic channels [4]. Furthermore, multipath transmission, signal power decline, and phase oscillation in underwater acoustic channels compound the challenges. It is necessary to design an effective transmission scheme over the acoustic channel [5]. Moreover, due to the low computational resource of Underwater Sensor Nodes (USNs), the computational problems need to be considered, especially in applications where a large amount of data needs to be processed. The data acquired by USNs need to be processed, which requires high computational power [6]. In the network model of this paper, we considered ships as the Base Stations (BSs). The USNs transmit the data to the Sink Node (SN), and the SN delivers the data to the BS for the computational process. To reduce the processing time of data, we introduce a scheme of

cooperative computation between BSs, in which tasks are divided into multiple components and transmitted between BSs [7,8]. This method can ensure a low processing time and high reliability. Therefore, the main difficulty in this scheme is how to select a proper task offloading and parallel computing algorithm. The problem becomes more severe when the communication ranges of BSs are overlapping. The service time will be further decreased by exploiting the methods of task partition and cooperative computing. The task division and cooperation among BSs are the techniques in which the task is divided into multiple components, and multiple BSs process the tasks in parallel.

Furthermore, there is also a service risk session due to the movement of BSs, the limited communication range [9]. Due to the movement of nodes, the nodes which are communicating with each other can be out of their communication range, which causes an ongoing session to be interrupted or a service failure [10]. For this purpose, we propose a scheme to process the computational tasks based on coalition game theory. In the coalition game, to avoid the service risk, the coalition is formed by the nodes of USNs, SNs, and BSs in the communication range. The coalition game is used to cooperate between the BSs for task divisions [11]. In this paper, we propose the cooperation policy based on the coalition game theory to provide better service reliability by dividing and allocating in parallel the tasks between BSs. Consequently, the service reliability can be improved through cooperation between BSs. After the coalition is created and the task is ready to transmit from SN to BSs, the next issue is to reduce the communication cost [12,13]. Hence, an efficient transmission mechanism of the computational task is proposed to minimize the communication time and improve the service reliability. In this scheme, each base station first finds the nearest neighbor base station within its transmission range. In the next phase, the BSs process the computational task sent by the SN. In addition, each BS chooses whether it partially or entirely executes computing tasks based on whether it establishes an alliance with other BSs. If the alliance has been established, the BS only processes part of the computing tasks and transmits the results of the computing process to another BS cooperating with it [14,15]. Furthermore, artificial neural networks (ANNs) can be utilized to divide tasks intelligently. The ANN model can attain efficient computational offloading, smart division of tasks and cooperation among BSs, and an efficient routing policy. The ANN model can also be used to acquire the computation results without interruption due to movement [16–18].

Therefore, aiming at the problem of computation and communication in USNet, the computational offloading, smart division of tasks, cooperation between BSs based on coalition game, and efficient transmission mechanism were investigated to reduce the total service time and improve the reliability of the communication.

The main contributions of our work are given as follows:

First, we present the mathematic formula and theoretical model of the coalition game among BSs. Based on the coalition game theory, the cooperation policy among USNs, SNs, and BSs is formulated. The coalition is established only if the tasks are divided and processed in parallel among BSs and better service reliability is provided.

Second, we study the strategy of smart task division and parallel processing to reduce the total service time. In addition, the cooperative computing method between BSs and an effective transmission mechanism is studied to reduce the communication time, enhance service reliability, and decrease the failure of service sessions due to the movement of BSs.

Third, because the computing task unloading, task division, task processing, and transmission process in this scheme are complex problems, the ANN model is introduced to reduce the complexity of the algorithm and the overall cost of service.

Finally, simulation results indicate that our proposed method outperforms the exiting techniques, i.e., Greedy, Always-Migrate, and Random Process, in terms of a total service time, success rate, and failure rate.

The remainder of this paper is organized as follows: In Section 2, we review the related works. In Section 3, the system model is analyzed. Section 4 gives a detailed description

of the scheme for cooperative computing offloading. Section 5 illustrates the performance evaluation of our proposed methods. Finally, the work is concluded in Section 6.

2. Related Work

In this section, the work related to the task offloading and cooperative computing offloading scheme is mainly discussed, and we analyze the dynamic characteristics in USNet.

2.1. Related Work of Task Offloading

Many scholars proposed some related research on the efficient computation and communication in other wireless networks (such as vehicle network, wireless network, wireless sensor network, etc.), based on methods of cloud computing, edge computing, collaborative computing offload, etc. Sensors are used in various fields to obtain data. When the data that are sensed are processed through Mobile Cloud Computing (MCC) or another computing method, they can be further turned into useful information [19]. For the communication problem in computation offloading, Haghighi V et al. proposed a task offloading scheme based on mathematical graph modelling. In this scheme, by considering the K-LARAC and M-LARAC heuristic algorithms, a new heuristic algorithm is introduced to find the optimal path that can evaluate the delay with the minimum service cost [20]. In the Cloud environment, due to the long distance between the Cloud and Mobile Devices (MDs), there is high latency and power consumption in resource allocation and task scheduling. In the application of the Internet of Things (IOT), Mobile Fog Computing (MFC) is introduced to improve speed and reduce energy consumption [21,22]. Aiming at solving the problem of computing delay and load balancing in edge computing, Meng et al. transformed the problem into a Markov decision process and designed a task offloading algorithm based on a depth deterministic policy gradient [23].

Because the Cloud Computing (CC) based on centralized remote infrastructure results in significant communication delays, it cannot meet the requirements of delay-sensitive applications. To solve this problem, the concept of Edge Computing (EC) is proposed; that is, the ability of CC is relocated to the terminal equipment closer to the edge of the network, in which the optimization methods of mathematics, artificial intelligence, and control theory can better process the computational task offloading [24]. For the distributed task offloading in Mobile Edge Computing (MEC), Su et al. proposed a distributed task offloading mechanism based on the Stackelberg game theory in [25].

2.2. Related Work of Cooperative Computing Offloading Scheme

For the application environment that requires intensive data exchange among nodes, Zhai Y et al. proposed a method called cooperative offload in [26], which considers the cooperation between mobile devices and the communication problems resulting from computing offload. Zhong S et al. formulated the problem of joint computation offloading, service caching, and resource allocation as a Mixed Integer Non-Linear Programming (MINLP) problem and proposed an efficient collaborative service caching and computation offloading algorithm to improve the Quality of Service (QoS) and reduce the computation complexity [27]. By making full use of the cooperative communication ability among mobile terminals, Peng et al. discussed the Device-to-Device (D2D) joint multi-user cooperative partial unloading, transmission scheduling, and computation allocating based on Mobile Edge Computing (MEC). An online resource coordinating and allocating scheme is proposed by considering random application requests, unpredictable Mobile Terminal (MTs) states, time-varying channel states, and computing resources [28]. Seid et al. proposed a Mobile Edge Computing Offload (MECO) architecture based on multiple Unmanned Aerial Vehicles (UAVs), in which the deep reinforcement learning method was used to deal with the complex problems of computing offload and resource allocation in a dynamic environment [29]. When multiple UAVs cooperate to complete a complex task, the task unloading between multiple UAVs must meet the interdependence of the task and realize parallel processing, so as to reduce the computing cost of terminals.

Considering the computing requirements of terminals, the delay constraints of computing tasks, and the safe distance of UAVs, Yu et al. constructed a UAV-assisted cooperative offloading system based on mobile edge computing in [30]. The system optimizes the number of local computing tasks, the number of computing unloading tasks, the trajectory of UAVs, and the unloading matching relationship between multiple UAVs and multi-user terminals.

2.3. Related Work of Dynamic Characteristics in USNet

The development of USNet has become increasingly mature, and users' expectations for the service quality provided by USNet are also increasing. There are also corresponding studies on the communication problems of USNet, but they mainly focus on the communication of the underwater acoustic channel, media access control mechanism, and routing strategy. In contrast, there are few research studies on efficient computation in USNet.

USNet plays an important role in marine environment monitoring. Some existing research is devoted to deployment optimization. For the dynamic characteristics of mobile sensors, Kuang et al. studied the scheduling problem in multi-sensor networks composed of a small number of mobile sensors and a large number of fixed sensors in [31]. The problem is expressed as a multi-objective dynamic scheduling model, in which mobile sensors are taken as cluster heads. Then, a dynamic multi-objective collaborative optimization algorithm is designed by decomposing decision variables, designing strategies to respond to environmental changes, and generating cooperation strategies that approximate Pareto optimal sets.

Due to the expensive system deployment cost, the existing USNet is usually deployed as a sparse network. Therefore, in a sparse USNet, AUVs are necessary to realize underwater surveillance and target detection [32]. With the increasing difficulty and complexity of underwater tasks, the cooperative positioning of Underwater Unmanned Aerial Vehicles (UUVs) in the form of clusters has become an inevitable trend. Therefore, due to the weak underwater communication conditions and the complexity of the meteorological environment, UUVs have problems such as observation limitations and time delays. Therefore, in [33], a distributed cooperative localization method for UUVs under the condition of weak underwater communication was proposed. In addition, Tu et al. proposed a location-free path routing scheme based on Q-learning, in which the Q value is obtained by considering the data rate, energy, and depth of underwater wireless sensor nodes. This scheme extends the life cycle of underwater wireless sensor nodes and reduces the communication time [34,35].

2.4. Related Work of Artificial Neural Networks

The artificial neural network (ANN) is a massively parallel distributed information processing system that has certain performance characteristics resembling biological neural networks of the human brain. The neural network is characterized by its architecture that represents the pattern of connection between nodes, its method of determining the connection weights, and the activation function [36]. In the past few decades, ANN methods have been extensively used in a wide range of applications, including hydrology, satellite telemetry, satellite images, etc. He et al. used an artificial neural network to forecast the river flow in the semiarid mountain region and the northwestern flow in the semiarid mountain region of Northwestern China [36]. This study suggests that ANN methods can be successfully applied to establish river flow with complicated topography forecasting models in semiarid mountain regions. Gozutok et al. estimated signal fading on telecommunication satellite telemetry signals with hybrid numerical weather prediction and an ANN approach under presence of an aerosol effect [37]. In this research, an implementation of artificial deep neural networks over outputs of 24-h multi-domain high-resolution nested real-case Weather Research and Forecasting (WRF) model runs was carried out over two high-resolution simulation domains, which are tested and compared for rainfall generation in order to assess the signal-fading event observed on geostationary

telecommunication spacecraft in orbit for a real multiscale storm case. In [38], Joshi et al. used satellite images to establish artificial neural network models to predict precipitation characteristics in the Western Himalayas region. In this research, the satellite images acquired in IR (10.5–12.5 μm) and WV (5.7–7.1 μm) bands were used for to develop an ANN model for both qualitative and quantitative precipitation forecasting. The model results were validated with ground observations, and skill scores were computed to check the potential of the model for operational purposes.

From the above research report, it can be seen that ANNs are more suitable for the related application fields compared with the traditional methods. Therefore, in this study, to reduce the service time and complexity, we introduce the ANN model.

3. System Model Analysis

In this section, we present the three models: network model, computing task processing model, and service time model.

3.1. Network Model

In the network model, we consider three types of nodes, namely Underwater Sensor Nodes (USNs), Sink Nodes (SNs), and Base Stations (BSs), as shown in Figure 1. The USNs and SNs are deployed randomly underwater, and BSs are deployed on the ships above the water's surface. The USNs, SNs, and BSs are represented as $\mathcal{U} = \{u_1, u_2, \ldots u_U\}$, $\mathcal{N} = \{n_1, n_2, \ldots, n_N\}$, and $\mathcal{B} = \{b_1, b_2, \ldots b_B\}$. A coalition is formed among the nodes of USNs, SNs, and BSs, which is depicted in Section 4. The computational task is offloaded by SNs to BSs. The communication time is calculated according to the attenuation model of the acoustic channel. Since the nodes of USNs and SNs have limited communication capability and computing power, while BSs have enough communication capability and computing power, BSs can be used to provide computational services to SNs and USNs in a process called ocean computing. The research objective of this paper is to provide computational services and reliable connectivity for SNs to guarantee high computing capability and low communication time. In our scheme, the task performed by the SNs is denoted by A_t, and we define a certain time span, T_s. It is assumed that, within a T_s, the task A_t is offloaded by n $\in \mathcal{N}$ to b $\in \mathcal{B}$. A_t must be offloaded to BS before the next T_s. Therefore, an offloading and computing policy is used for $\mathcal{N}$. The BS is considered to be the main controller that makes the offloading and computing decisions for USNs and SNs in a decentralized manner. Within a T_s, the offloading decision for $\mathcal{N}$ is denoted by a binary vector, i.e., $V_{n(T_s)}$, $\forall$n $\in \mathcal{N}$. For instance, if $V_{n(T_s)} = 1$, the computation task of n $\in \mathcal{N}$ is offloaded to b $\in \mathcal{B}$. Otherwise, if $V_{n(T_s)} = 0$, the computation task of the SN node n (n $\in \mathcal{N}$) is not offloaded to the BS node b (b $\in \mathcal{B}$). A computation task can be jointly processed by more than one BS, b $\in \mathcal{B}$, in which the results of computation task are subject to being transmitted to the SNs.

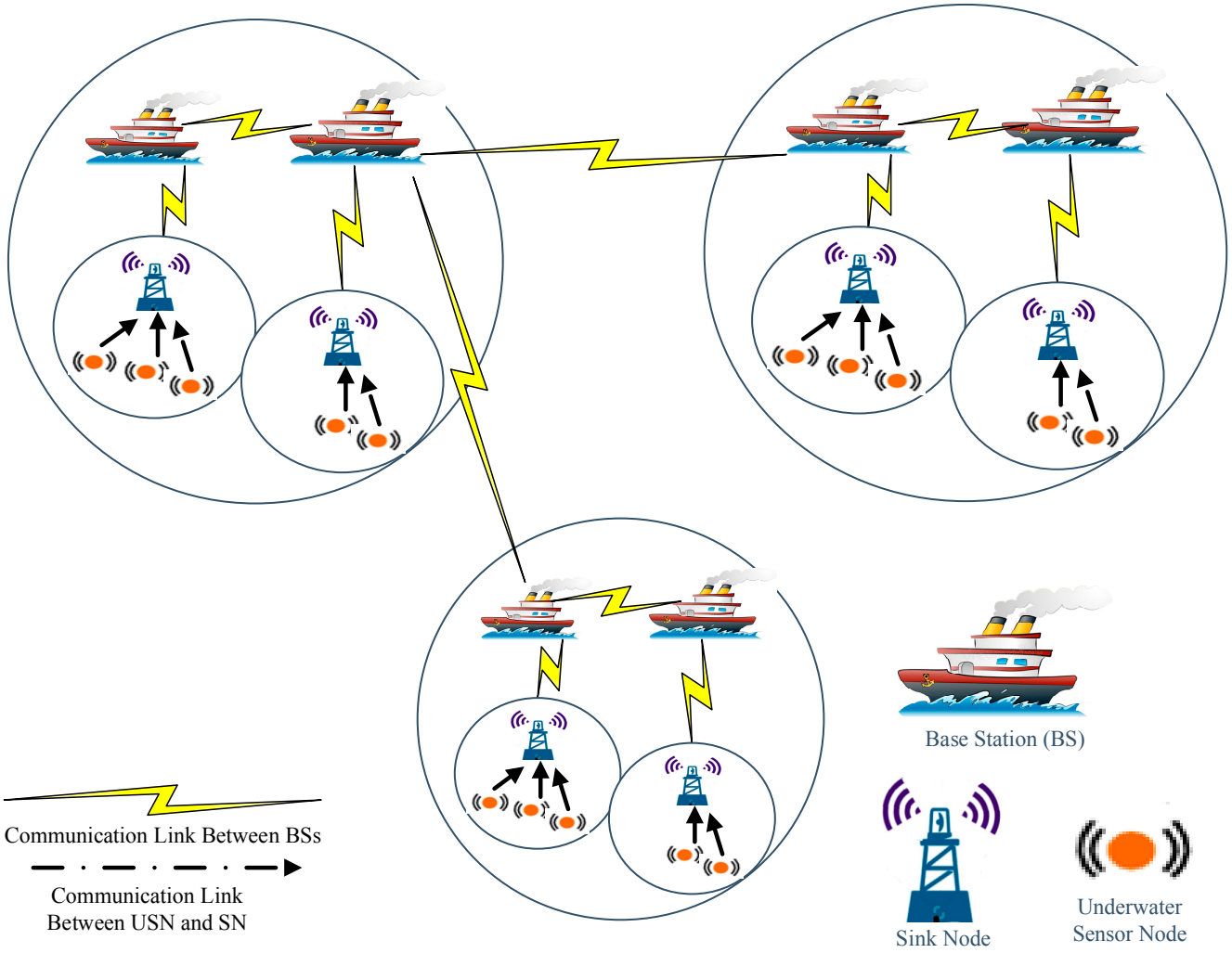


Figure 1. Network model for cooperative communication.

3.2. Computing Task Processing Model

To reduce the computational time, another BS, i.e., node $b_j, \forall j \in \mathcal{B}$, is required to complete the task in collaboration and in parallel. Thus, a task partition policy is needed to select proper BSs to complete the service. The task must be intelligently computed into sub-tasks in parallel. The BS, i.e., node $b_i, \forall i \in \mathcal{B}$, maintains a portion of the task and transmits the rest of the task to the nearest BS, i.e., node $b_j, \forall j \in \mathcal{B}$. Hence, node b_i and b_j compute the task in parallel. After completing the task, node b_j transmits the result to the previous BS, node b_i, and node b_i combines these results and transmits the combined result to the SN node, n_N. The cooperative computing decision is also considered between BSs and represented by $V_{b(T_s)}$. If $V_{b(T_s)} = 0$, the $b_j, \forall j \in \mathcal{B}$ will not cooperate with the desired $b_i, \forall i \in \mathcal{B}$. Otherwise, if $V_{b(T_s)} = 1$, the BSs will cooperate with each other. As the ship (considered BS node b_i) moves from one place to another, it might be out of the communication range of node SN. If the BS moves out of the transmission range of node SN, node SN broadcasts a message to another BS node, b_j, in the coalition, and then the node b_j restarts to seek an alliance by broadcasting its information to other BSs. The information contains the identity of the node b_j. When a BS node, $b_k, k \in \mathcal{B}$, and $k \neq i \neq j$ receive the identity and find that it is closest to the node b_j, then node b_k transmits the information with its own identity to node b_j, and the two nodes, b_j and b_k, form an alliance. The task is divided and allocated to the two BS nodes, and the computation result is transmitted to the SN node, then the SN node stops broadcasting the message to its BS node. Here, we assume a threshold time interval to reduce the service time in our proposed model. After the

threshold time expires, the SN node stops broadcasting without waiting for the feedback from node b_i. At this time, if the SN node receives the feedback from node b_k, k $\in \mathcal{B}$, and the SN rebroadcasts the offloading task to the BS node b_k, k $\in \mathcal{B}$ in its coalition.

The BS needs to divide the task intelligently into sub-components. The sub-components are divided in such a way that a minimum amount of time is needed to finish the total task. For instance, the computational tasks offloaded within a T_s are divided into two sub-components, one for node b_i, i $\in \mathcal{B}$ and one for node b_j, j $\in \mathcal{B}$, respectively. We denote node b_i, i $\in \mathcal{B}$ as the receiver BS and node b_j, j $\in \mathcal{B}$ as the cooperative BS. The divided portions of the computational task are denoted by T_{t_i} and T_{t_j}, respectively. Hence, $A_t = A_{t_i} + A_{t_j}$. Thus, we have the following cases if $V_{b(T_s)} = 1$ and $V_{n(T_s)} = 1$, respectively.

(1) $A_t = A_{t_i}$ and $A_{t_j} = 0$: In this case, the computational task is only offloaded to node b_i, i $\in \mathcal{B}$ in a T_s. Hence, node b_i, i $\in \mathcal{B}$ is both received and a cooperative BS.
(2) $A_{t_i} > 0$ and $A_{t_j} > 0$: In this case, the computational task is offloaded to both received and cooperative BSs. The node b_i, i $\in \mathcal{B}$ and node b_j, j $\in \mathcal{B}$ process the divided components of the task in parallel. The combined result is delivered from node b_i, i $\in \mathcal{B}$ to the SN node which offloaded the task.
(3) $A_t = A_{t_j}$ and $A_{t_i} = 0$: In this case, the computational task is offloaded to node b_i, i $\in \mathcal{B}$. But, if node b_i, i $\in \mathcal{B}$ moves outside the transmission range, its cooperative BS, i.e., b_j, j $\in \mathcal{B}$, processes the task. The result of the task is delivered to the SN by node b_j, j $\in \mathcal{B}$.

3.3. Service Time Model

The service time model includes the communication time and computational time.

3.3.1. Communication Time

We consider the communication time as the offloading time of the task from SN node n, n $\in$ N to BS node b, b $\in$ B; the forwarding time of the sub-task from node i $\in$ B to node j $\in$ B; and the return time of the final result from b $\in$ B to n $\in$ N. The pathloss, d, in the communication for underwater acoustic communication between the SN and BS can be calculated by using Urick's model [39], as shown in Equation (1):

$$d = K \times 10log(r) + \frac{r}{1000} \times 10\, loga(f) \tag{1}$$

where K is the spreading factor, and the value of K is taken as 1.5 for practical spreading; r is the distance (m) between the SN and BS (or from BS to BS); f is the frequency of the signal in kHz; and $a(f)$ is the absorption coefficient and is calculated by using Francois and Garrison's formula mentioned in [40]. The absorption coefficient depends on the frequency, pressure, temperature, salinity, PH, and acoustic propagation velocity, as given in [41]. The signal-to-noise ratio (SNR) is calculated at the receiver. It is assumed that losses occur due to the multi-path effect, and the Doppler effect can be neglected. Hence, the SNR S is calculated as shown in Equation (2) [42].

$$\mathrm{S} = \frac{\mathrm{T_{ASL}}}{\mathrm{d} \times \mu \times \mathrm{W}}, \tag{2}$$

where d is the path-loss factor, as mentioned in (1); W is the receiver bandwidth; and μ is the linear sum of the four ambient noise components, i.e., turbulence, shipping, waves, and thermal noise, which are calculated as shown in Equation (3) [39].

$$\begin{aligned} &10\log \mu_{\mathrm{Tur}} = 17 - 30\log(\mathrm{f}) \\ &10\log \mu_{\mathrm{Sp}} = 40 - 20(\mathrm{s} - 0.5 + 26\log(\mathrm{f}) - 60\log(\mathrm{f} + 0.003) \qquad 0 < s < 1 \\ &10\log \mu_{\mathrm{Wav}} = 50 + 7.5\mathrm{J}^{\frac{1}{2}} + 20\log(\mathrm{f}) - 40\log(\mathrm{f} + 0.4) \\ &10\log \mu_{\mathrm{Ther}} = -15 + 20\log(\mathrm{f}). \end{aligned} \tag{3}$$

The T_{ASL} is used to calculate the transmitter's acoustic signal level and is given by Equation (4):

$$T_{ASL} = 170.8 + 10logP + 10log\eta + I, \tag{4}$$

where P is the transmitted power, and η is the efficiency factor. The efficiency factor includes the losses associated with electro-acoustic conversion, and I is the projector directivity index [43]. Hence, the data rate from SNs to BS is given by Equation (5):

$$\mathcal{R}_{n,b} = W^n log_2(1 + S), \forall n \in \mathcal{N}, b \in \mathcal{B}, \tag{5}$$

where W^n denotes the bandwidth reserved for SNs. In addition, the data rate from BS i to BS j for the division of the task is calculated as shown in Equation (6):

$$\mathcal{R}_{i,j} = W^{i,j} log_2(1 + S), \forall i, j \in \mathcal{B}, \tag{6}$$

Here, $W^{i,j}$ is the bandwidth reserved for transmitting the task offloaded within T_s. The data rate, $\mathcal{R}_{b,n}$, from the BS to the SNs is given by Equation (7):

$$\mathcal{R}_{b,n} = W^{b,n} log_2(1 + S), \forall n \in \mathcal{N}, \in \mathcal{B}, \tag{7}$$

Hence, the communication time for offloading the computational task within T_s to $i \in \mathcal{B}$ is calculated as shown in Equation (8):

$$\mathcal{T}_{\mathrm{n}}(\mathrm{t}) = \frac{V_{n(T_s)} \times A_t}{\mathcal{R}_{n,b}}. \tag{8}$$

The communication time for forwarding the computational task from i to j at T_s is calculated as shown in Equation (9):

$$\mathcal{T}_{\mathrm{i,j}}(\mathrm{t}) = \frac{V_{b(T_s)} \times A_t}{\mathcal{R}_{i,j}}. \tag{9}$$

The return time of the final computation result from the BS to the SNs is calculated as shown in Equation (10):

$$\mathcal{T}_{\mathrm{b,n}}(\mathrm{t}) = \frac{\Phi \times A_t}{\mathcal{R}_{b,n}}. \tag{10}$$

3.3.2. Computation Cost

After the SNs offload the computational task, the receiver BS processes the task-specific portion, A_{t_i}, and forwards the other portion, A_{t_j}, to the cooperative BS for processing. Hence, the processing time for computational task to be offloaded to BS i within T_s is calculated as shown in Equation (11):

$$\mathcal{T}_{\mathrm{i}}^{\mathrm{p}}(\mathrm{t}) = \frac{N[V_{n(T_s)} \times A_{t_i}]}{C_c} \tag{11}$$

where $V_{n(T_s)}$ represents the binary vector of the offloading decision from SNs to node b_i, i $\in \mathcal{B}$ within T_s. In case there is no offloading of the computational task from an SN, the $V_n(T_s)$ is equal to zero, and there will be no processing of the task at node b_i, i $\in \mathcal{B}$.

The processing time for a computational task to be offloaded to node b_j, j $\in \mathcal{B}$ is calculated as shown in Equation (12):

$$\mathcal{T}_{\mathrm{j}}^{\mathrm{p}}(\mathrm{t}) = \frac{N[V_{n(T_s)} \times V_{b(T_s)} \times A_{t_j}]}{C_c} \tag{12}$$

where C_c is the computing capability of the BS, and N represents the number of computation cycles required to operate the data. In case there is no offloading of the computational task from node $b_i, \mathrm{i} \in \mathcal{B}$ to the node $b_j, \mathrm{j} \in \mathcal{B}$, the $V_{b(T_s)}$ is equal to zero, and there will be no processing of the task at node $b_j, \mathrm{j} \in \mathcal{B}$. Remarkably, a new computation task offloaded within this T_s can be processed only if the computing tasks offloaded within the previous T_s are completed. To offload the task to node $b_i, \mathrm{i} \in \mathcal{B}$, the total computational time (queuing time and processing time), $\mathcal{T}_i(t)$, is calculated as shown in Equation (13):

$$\mathcal{T}_i(t) = \mathcal{T}_i^q(t) + \mathcal{T}_i^p(t) = max\{\mathcal{T}_q(t)(t-1) + 2Y, 0\}, \tag{13}$$

where the value of Y is the time length of T_s. To offload the task to node $b_j, \mathrm{j} \in \mathcal{B}$, the total computational time is calculated as shown in Equation (14):

$$\mathcal{T}_j(t) = \mathcal{T}_j^q(t) + \mathcal{T}_j^p(t) = max\{\mathcal{T}_q(t)(t-1) + Y, 0\}, \tag{14}$$

where $\mathcal{T}_i^q(t)$ and $\mathcal{T}_j^q(t)$ are the queuing time of the offloaded task at node $b_i, \mathrm{i} \in \mathcal{B}$ and the node $b_j, \mathrm{j} \in \mathcal{B}$ in the T_s. The division of the computational tasks to be offloaded by the SNs is taken into account in the following manners. The computational task is divided into two sub-components which are independent of each other.

3.3.3. Total Service Time

In this paper, we computed the total service time, $\mathcal{T}_{sd}(t)$, which is shown in Equation (15):

$$\mathcal{T}_{sd}(t) = \mathcal{T}_n(t) + max\Big\{V_{n(T_s)}\mathcal{T}_i(t), \mathcal{T}_{i,j}(t) + V_{b(T_s)}\mathcal{T}_j(t)\Big\}, \tag{15}$$

where $\mathcal{T}_\mathrm{n}(\mathrm{t})$ is the offloading time from the SNs to BS, and i.$\mathcal{T}_\mathrm{n}(\mathrm{t})$ is the forwarding time from BS i to BS j. To divide the tasks into multiple sub-components, the following three scenarios are considered:

(1) In the first case, BS i computes the total computational task, and BS j does not take part in computation. In such a case, $V_{b(T_s)} = 0$. Hence, $\mathcal{T}_j(t) = 0$. Therefore, the maximum function, i.e., $[V_{b(T_s)}\mathcal{T}_i(t)]$, is used.

(2) In the second case, BS j computes the complete computational task, and BS i does not take part in the computation. In such a case, $V_{n(T_s)} = 0$, and, consequently, $V_{n(T_s)}\mathcal{T}_i(t) = 0$. Thus, the maximum function, i.e., $[\mathcal{T}_{i,j}(t) + V_{b(T_s)}\mathcal{T}_j(t)]$, is considered.

(3) In the third scenario, both BSs, i.e., node $b_i, \mathrm{i} \in \mathcal{B}$ and node $b_j, \mathrm{j} \in \mathcal{B}$ take part in the computation by dividing the task between them. In this case, $[V_{n(T_s)}\mathcal{T}_i(t), \mathcal{T}_{i,j}(t)] \neq 0$, $[V_{b(T_s)}\mathcal{T}_j(t)] \neq 0$. Therefore, the greater values between $[V_{b(T_s)}\mathcal{T}_i(t), \mathcal{T}_{i,j}(t)]$ and $[V_{b(T_s)}\mathcal{T}_j(t)]$ are selected as the maximum function with which the BSs compute the task in parallel.

When the signal between the primary BS node and its coalition BS node is weak and the noise is strong, the division and offloading between two coalition BSs may fail, so we define a threshold value, which is denoted by Γ, for the signal-to-noise ratio (SNR). The primary BS node offloads and forwards the sub-component of the computational task to the coalition BS node only when its SNR satisfies the following premise given by Equation (16) or Equation (17):

$$\mathrm{S} \geq V_{n(T_s)}(\mathrm{t})\Gamma \tag{16}$$

and

$$\mathrm{S} \geq V_{b(T_s)}(\mathrm{t})\Gamma \tag{17}$$

where S is the SNR of a BS node.

During a service session, if one BS moves out of the transmission range of the SN node which has offloaded the computation task to the BS node, the service session for the

SN node fails. The success or failure of a service session for an SN node within a T_s is represented by a variable. Thus, $\mathcal{F}_s$ is shown in Equation (18).

$$\mathcal{F}_s = s.t. \begin{cases} 1. & if\ S\ \geq V_{b(T_s)}\ \Gamma \\ 0. & Otherwise \end{cases} \tag{18}$$

The distance between the service providing node BS and the request node SNs within a T_s is $r_{i,j}(t + \mathcal{T}_{sd})$, which is considered to be the distance when the task is completed and the results are returned to the SN.

To increase the service reliability for task division and results migrating, a shortest-distance search method is proposed to find the best coalition BS. In the next section, we formulate the problem of cooperative computational offloading using coalition game theory. In addition, we adopt an artificial neural network (ANN) to divide the task intelligently and obtain efficient computational offloading.

4. Cooperative Computational Offloading Scheme

This section first describes the cooperation method using the coalition game theory. Then, the computational offloading scheme based on ANNs is introduced.

4.1. Service Time Model

In this work, we propose a method of computing offload based on coalition game theory, in which the nodes of SN or USN only enter into alliance with those BSs who provide better service performance in terms of minimum service time, minimum computation cost, and minimum total service time.

The coalition among USNs, SNs, and BSs, as well as the cooperation between BSs, is formulated based on the coalition game theory [44]. The coalition is established only when the task can be divided and processed in parallel to achieve better service reliability. Once the coalition is established and the task is transmitted to the BS by the SN, computational offloading and intelligent task division are performed based on the ANN model, which is shown in Algorithm 1. In coalition game theory, the players are the nodes of USNs and SNs who try to make a coalition with BSs. The coalition game is formulated by $\mathcal{K} = (\mathcal{U}, \mathcal{N}, \mathcal{B}, \mathcal{O}_{\mathcal{U}}, \mathcal{X})$. The concept model of coalition game theory is shown in Figure 2. The parameters are discussed as follows:

(1) Players: The USNs and SNs are the game players who try to make a coalition with the BSs, $\mathcal{B}$.
(2) Cooperation strategy: $\mathcal{O}_{\mathcal{U},\mathcal{N}}$ is the space of feasible cooperation strategies among all the players in the coalition with the BSs, respectively.
(3) Characteristics function: The game players exploit the characteristic function, $\mathcal{X}$, to map every nonempty subset, $\mathcal{P} \subseteq \mathcal{B}$, to the subset of the feasible cooperation strategies, i.e., $\mathcal{O}(\mathcal{P}) \subseteq \mathcal{O}_{\mathcal{U},\mathcal{N}}$.
(4) Coalition partition: The coalition partition is the set $\mathcal{P} = \{\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_{\mathrm{p}}\}$, where $\forall \mathrm{P}, \mathcal{O}(\mathcal{P}_{\mathrm{p}}) \subseteq \mathcal{O}_{\mathcal{U},\mathcal{N}}$, and $\mathcal{P}_p \cap \mathcal{P}'_p = \varnothing$ for $\mathrm{P} \neq \mathrm{P}'$ and $\cup_{P=1}^{P} \mathcal{P}_P = \mathcal{B}$. Hence, a cooperative coalition is established between the game players and BS.

Algorithm 1: ANN-Based Collaborative Computing

Input: T_s is $r_{i,j}[2744] \in \{100-1000\}$m, $r_{n,b}[2744] \in \{25-318\}$m, $A_t[2744] \in \{500-1000\}$MB,
Output: $s_1[2744]$, $s_2[2744]$, Dataset, $d_s[2744]$, $F_s[2744]$, distance[2744][14]
Step # 1 Calculating time:
for $i = 0 : 2743$ **do**
set delay $[A_t[i] + 1]$;
for $j = 0 : A_t[i]$ **do**
Calculate: (4)
Calculate: (10)
Calculate: (5)
Calculate: (11)
delay$[j] = [(4) + \max((10), ((5) + (11))]$
end
Calculate: minimum from delay $[A_t[i] + 1]$ array and store the index in [a].
$s_1[i] =$ divs[a][0]
$s_2[i] =$ divs[a][1]
If delay[a] ≥ 900 **then**
$r_{i,j}[i] = 0$;
$F_s = [i] = 1$;
else
$F_s = [i] = 0$
end
end
Step # 2 Cooperative Computing:
for $i = 0 : 2743$ **do**
if $F_s = 0$ **then**
if $s_1[i] = 0$ **then**
Check if $r_{i,j}[i]$ follows the SNR constraint from (15) and store the value in f.
if $f = 1$ **then**
$d_s[i] = 0$; $F_s[i] = 1$; **else** $d_s[i] = 2$; $F_s[i] = 0$;
end
else if
$s_1[i] > 0$ and $s_2[i] > 0$ **then**
Check if $r_{i,j}[i]$ follows the SNR constraint from (15) and store the value in f.
end
if $f = 1$ **then**
$d_s[i] = 0$; $F_s[i] = 1$; **else** $d_s[i] = 2$; $F_s[i] = 0$;
end
else if $s_2[i] = 0$ **then**
$d =$ abs$[r_{i,j}[i]] - S_{a,b}$;
Check if $r_{i,j}[i]$ follows the SNR constraint from (15) and store the value in f_1.
end
If $f_1 = 0$ **then**
$d_s[i] = 1$; $F_s[i] = 0$; **else**
Check if $r_{i,j}[i]$ follows the SNR constraint from (15) and store the value in f_2
If $f_2 = 1$ **then**
$d_s[i] = 0$; $F_s[i] = 1$; else $d_s[i] = 2$; $F_s[i] = 0$;
end
end
end
end

Algorithm 1: *Cont.*

Step # 3 Shortest Distance Search Algorithm:
Check $r_{i,j}$ to find the most suitable BS for delivering result by keeping in view of the SNR constraint from (15)
If the constraint is not met for an index, then make $d_s = 0$ and $F_s = 1$ for that index.
for $i = 0 : 2743$ **do**
  for $j = 0 : 13$ **do**
    If $d_s[i] = j + 1$ **then**
      for $k = 0 : 13$ **do**
distance[i][k] = D[j][k]
      end
    end
  end
end
Step # 4 Coalition Game:
Repeat
Randomly select coalition, P, i.e., $(\mathcal{U}, \mathcal{N}$ and $\mathcal{B}) \in \mathcal{P}$;
Randomly select another coalition, $\mathcal{P}'_{\mathcal{P}}, \mathcal{P} \neq \mathcal{P}'$;
Check the preference order of $\mathcal{N}$ and $\mathcal{U}$ for $\overline{\mathcal{B}} \in \mathcal{P}$ and $\overline{\mathcal{B}'} \in \mathcal{P}'$
  If (20) is not satisfied, **then**
$\mathcal{U}, \mathcal{N}$ splits from its current coalition and forms a new coalition;
Update the current coalition;
  else
Go to Repeat;
  end
Until Convergence to the final stable partition.
Step # 5 Training:
Divide the dataset into training, validating, and testing sets.
Network ←ANN[10, 10, 10, 10, 11]
  while $i \leq$ *dataset* **do**
trained = train(network, input, labels)
end
Test performance on the test set.
Accuracy ← Number of correctly predicted policies/Total number of policies.

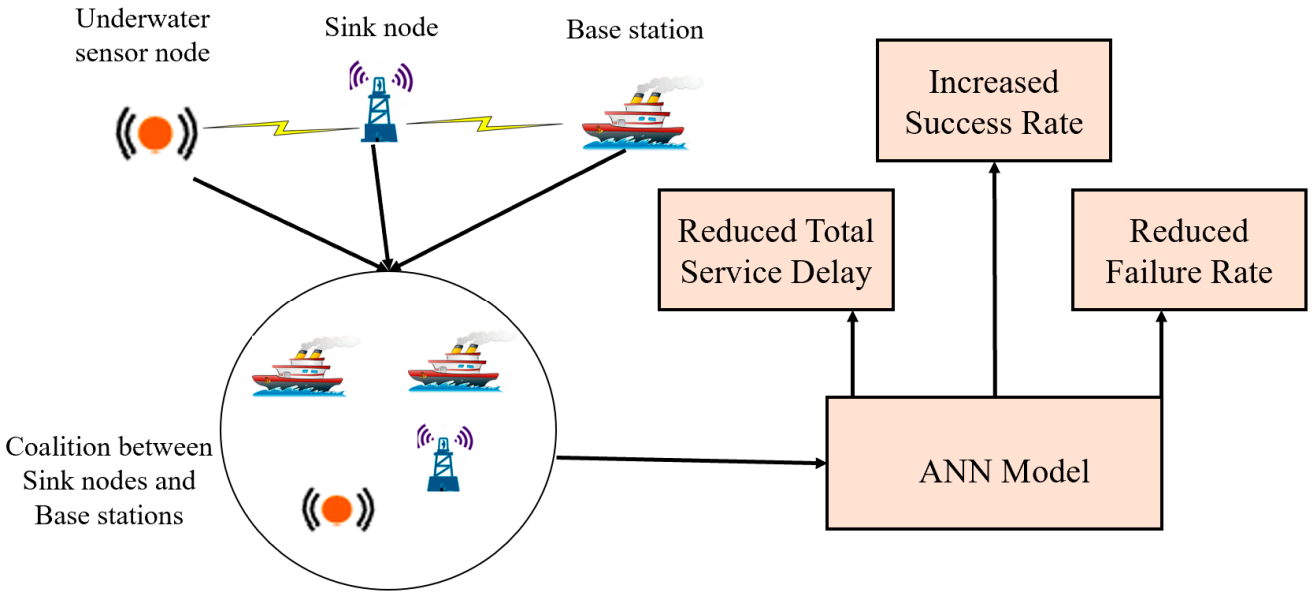


Figure 2. Concept model of coalition game theory.

The game players make a coalition according to the well-defined preferences and formulate a coalition partition. First, the USN players try to communicate to the SNs within its transmission range and make the SNs game players. Then, all the game players, i.e., USNs and SNs, try to seek the BSs which can provide minimum service time and computation cost to make coalition, as mentioned in Formulas (8)–(10), (13), (14), and (18). In other words, the USNs and SNs seek their preferred BSs as alliances by comparing the preference order of the potential alliances. Here, we need to define the concept of preference order. The preference order is defined as the reflexive, complete, and transitive binary relation over the set of possible BSs so that USNs and SNs can form the cooperative coalition, $\mathcal{P} \subseteq \mathcal{B}$, which satisfies the following:

$$\mathcal{P}_P \cap \mathcal{P}'_P = \varnothing, \forall \mathrm{P} \neq \mathrm{P}' \text{ and } \cup_{P=1}^{P} \mathcal{P}_P = \mathcal{B} \tag{19}$$

In the proposed coalition partition formation, the USNs and SNs prefer these BSs, which can provide minimum service time and computation cost as alliances. Therefore, the preference order is defined as shown in Equation (20).

$$\overline{\mathcal{B}_i} \succ \mathcal{U}_i, \mathcal{N}_i \, \overline{\mathcal{B}_j} \Leftrightarrow \overline{\mathcal{F}_{s_i}} > \overline{\mathcal{F}_{s_j}} \tag{20}$$

The procedure to select the preferences is given as follows. In the first step, SNs and USNs form the coalition $\mathcal{P}_p$ with a BS selected randomly from $\mathcal{B}$. Next, the selected BS is replaced by a BS which can provide better service performance (with less service time and computation cost). Then, the service performance (service time and computation cost) of the BSs continues to be compared till the BS with the minimum service time and computation cost is found out. Then, the BS node with the minimum service time and computation cost is selected as an alliance, and the SNs switch to a new coalition, p'_p.

After several rounds of switching operations based on the preference order, the coalition formation game converges to a stable and optimal coalition partition. It should be noted that only the BSs which are within the range of SNs can become the potential alliances, and the SNs can collaboratively find the BS which provides the minimum service time and computation cost.

The coalition partition is based on a decentralized rather than a centralized solution. Hence, the complexity of our proposed framework is lower than that of centralized systems. Once the coalition partitions are established, the tasks are divided intelligently. The BSs compute their sub-tasks in parallel, which can reduce the minimum service time, computation cost and total service time. Next, the process of computational offloading based on the ANN model is introduced.

4.2. ANN-Based Model

In the ANNs, we have an overfitting problem when the dataset is not well organized for training and testing. Therefore, we divide the dataset into training, validation and testing subsets. Hence, it overcomes the problem of overfitting. Our proposed ANN model detects overfitting by using the train–validate–test process. The training set can be used for learning, while the validation set is a part of the training set. At this stage, the learning process cannot be used, and the testing set contains input data to represent the unseen data having large variations in the output. The validation set is used to evaluate the model parameters based on the model's performance. Hence, overfitting will occur and then saturate the learning process. The model notices that the error on the training set decreases, while the error on the validation set remains constant or begins to decrease. Therefore, the model halts the training and moves forward towards the testing process; if it does not halt the training, this leads to overfitting. The validation set is not as difficult as the testing set. The validation set contains unseen data, but the output variations are insignificant. The dataset is divided into 70% for the training purpose and 15% for testing. Hence, the proposed model can intelligently divide the tasks and find the shortest path to deliver SN

results. The overall process is explained in Algorithm 1. In addition, the flowcharts in Figures 3 and 4 summarize the whole process.

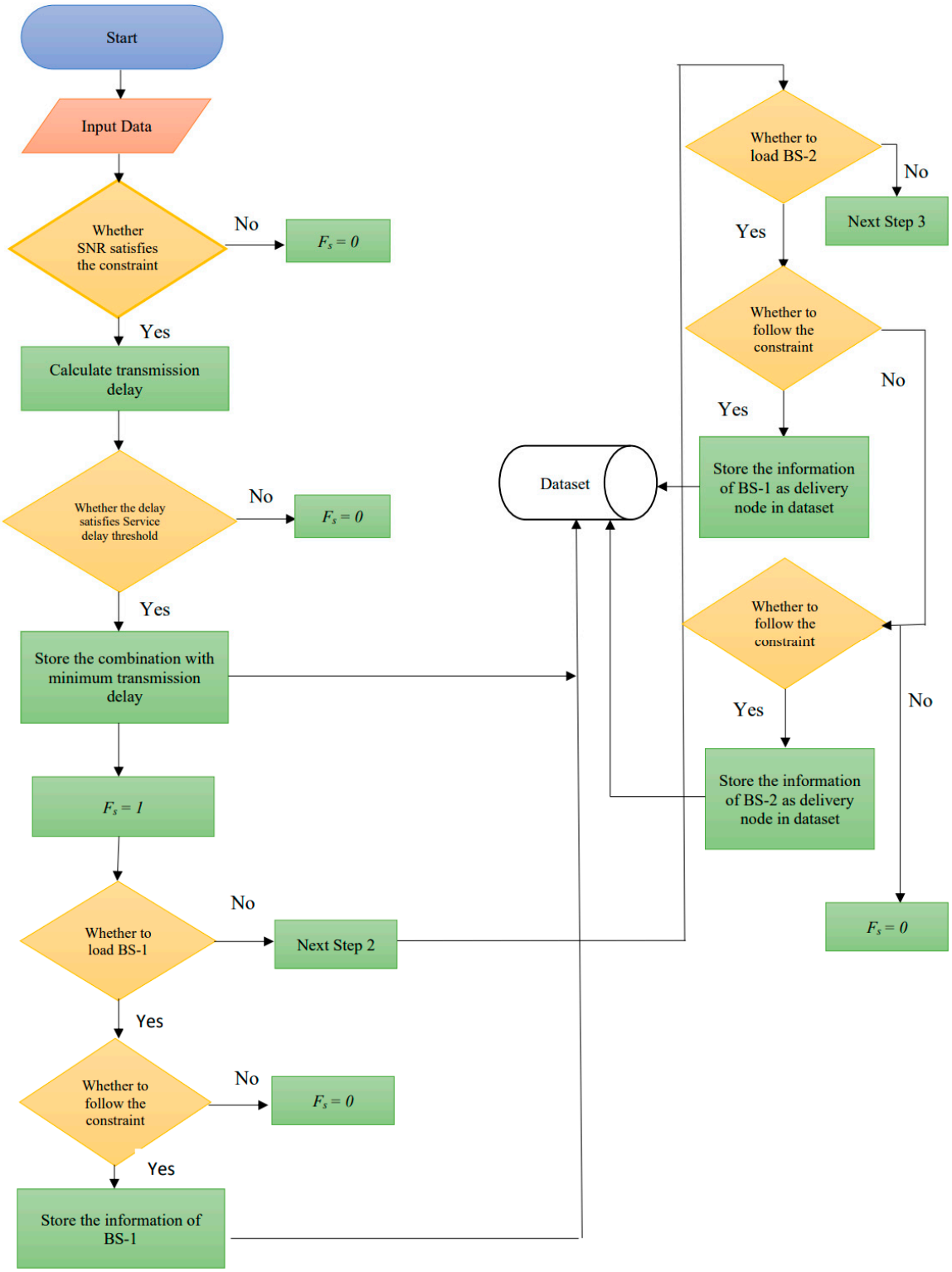


Figure 3. The flowchart of the collaboration process.

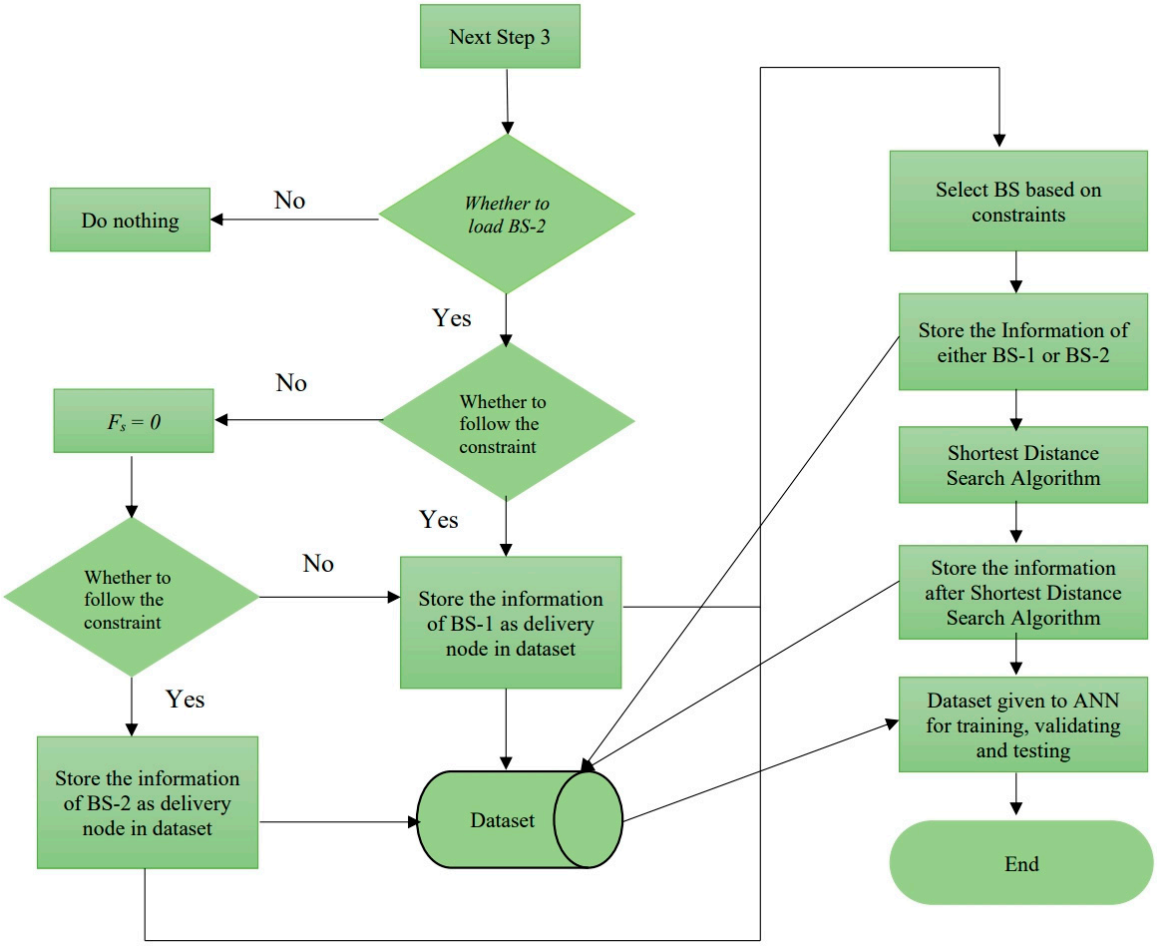


Figure 4. The flowchart of coalition and task offload.

4.3. Computational Offloading Scheme Based on the ANN Model

In the UANet, 14 BSs are deployed on the water. At first, one of the BS nodes is selected as an alliance by a SN node, and then the first BS as an alliance seeks another BS within its transmission range to cooperate and compute the task in parallel, as shown in Figure 1. There are at most two BSs in each coalition in the ANN model to divide the task and compute the sub-task in parallel. The dataset is generated with a size of 2744 values. The distance between an SN and a BS is considered to be a random variable from 25 m to 318 m. The distance between BSs is considered to be from 100 m to 1000 m. The size of the computational task A_t is from 500 MB to 1000 MB. Furthermore, we consider one input layer, five hidden layers, and one output layer, respectively. The input layer has 3 neurons, and the five hidden layers have 10, 10, 10, 10, and 11 neurons, respectively. The output layer has 18 neurons. In addition, the activation function is Hyperbolic Tangent (*Tanh)* for the first four hidden layers, and the activation function of the fifth layer is a linear function. The network is trained by the generated dataset, which is stored in a 2D array. The 2D array contains the number of rows equal to the size of the dataset, with 21 values in each row. Of the 21 values, the first value is the distance between the SN and BS. The second value is the distance between BSs, and the third value is the size of the task. The other three values are assigned as the input for the ANNs, and three neurons are assigned for the input layer. The values from 4 to 7 correspond to the sub-component task of the computational task for the cooperative BS, the ID of delivering BS, the communication time, and the compute time. The values from 8 to 21 are related to the distance between the receiver base station and the delivery base station. The value eight includes the ID of the receiver BS, and nine is for the ID of the nearest to the receiver BS in the shortest distance and so on until the ID of

the delivered BS for the rest of the values. If the ID of the delivered base station appears before 21, then the values after it will be 0. From values 4 to 21, there are 18 values in total, and these are considered to be the output values. Therefore, 18 neurons are used for the output layer.

Step1: The minimum delay is calculated according to the relevant formula and constraint conditions.

Step2: During a service session, if one BS moves out of the transmission range of the SN node which has offloaded the computation task to the BS node, the service session for the SN node failed. The success or failure of a service session for am SN node within a T_s is represented by a variable, $\mathcal{F}_s$. Thus,

$$\mathcal{F}_s = s.t. \begin{cases} \mathbf{1}. & \textit{if } S \geq V_{b(T_s)}\, \Gamma \\ \mathbf{0}. & \textit{Otherwise} \end{cases}$$

Step3: Find the most suitable BS for delivering result by keeping in view of the SNR constraint.

Step4: The nodes of SN or USN enter into alliance with those BSs that provide a better service performance in terms of minimum service time, minimum computation cost, and minimum total service time.

Step5: The dataset is divided into training, validation, and testing subsets, and the relevant data are trained in ANNs.

5. Performance Evaluation

In this section, the performance of our proposed method is evaluated and analyzed by comparing it with the Greedy algorithm [45], Always-Migrate [46] and Random Process [46].

The time complexity of our method is $O(n^5)$, where n is the size of the dataset. Hence, the computation time with our model is high. To evaluate the performance of our proposed algorithm, we compare it with three other algorithms, i.e., the Greedy method, Always-Migrate method, and Random method. The time complexities of the Greedy method, Always-Migrate method, and Random method are $O(n^2)$, $O(n^2)$, and $O(n^4)$, respectively, where n is the size of the dataset. The time complexity of our proposed method is $O(n^5)$. Consequently, the time complexity of the proposed method is higher than that of the other methods. Consequently, the proposed method increases the time significantly. Therefore, we can see from the simulation results that the proposed method outperforms the Greedy, Always-Migrate, and Random methods in terms of the success rate. To decrease the time complexity, an algorithm based on the above method and ANN technology is proposed, which reduces the service time significantly [47]. In the ANN algorithm, we train ANN from the dataset obtained from the previous proposed method. The time complexity of the new proposed algorithm based on ANNs is $O(nlog(n))$ [48]. Once the ANN is trained, the time complexity of our method becomes constant, i.e., $O(1)$; thus, the scheme based on ANNs reduces the service time to the minimum, and the time complexity becomes lower than that of Greedy, Always-Migrate, and Random methods.

6. Simulation Evaluation

In this section, we compare the performance of our proposed method with that of the Greedy algorithm, Always-Migrate, and Random Process through simulation experiments. First, we provide the simulation environment and parameters, and then we analyze the simulation results.

6.1. Simulation Settings

We implemented a CSSTU-MAC simulation with MATLAB platform. In the simulation setup, the value of T_{ASL} and C_c are considered to be 28 dBm and 8.2×10^4 c/s, respectively. In addition, the $W^{i,j}$ and $W^{b,n}$ are set to be 1 kbps and 9 kbps, respectively [49]. Moreover, Γ is set to be 8 dBm for $V_{n(T_s)}(t)$ and 6 dBm for $V_{b(T_s)}(t)$, respectively. The distances between

the SN and BSs are from 25 m to 318 m, whereas the distance between the BSs is 100 m to 1000 m, respectively. In the simulation scenario, nodes SNs and USNs are static, while BS nodes are mobile. The SNs are deployed on the water's surface. The transmission range of an SN node or a BS node is 318 m, whereas the transmission range of a BS node is 340 m in our simulation.

6.2. Simulation Results

This section evaluates the performance of our proposed method in terms of success rate, total service time, and computational time.

6.2.1. Success Rate

In our first experiment, the success rate and failure rate of the entire service sessions were evaluated and compared with the abovementioned existing methods, i.e., the Greedy, Always Migrate, and Random methods, which are summarized in Table 1. The results in Table 1 are generated by a large dataset which is offloaded by the SN and intelligently divided and processed in parallel between two BSs. As shown in Table 1, our proposed method outperforms the existing methods.

Table 1. Success rate and failure rate for four algorithms.

Algorithms	Success Rate	Failure Rate
Proposed method	90.12%	9.88%
Greedy method	60%	40%
Always-Migrate method	21.9%	78.1%
Random method	12.66%	82.34%

We compare our experiment results with those of the other three algorithms, namely the Greedy algorithm, Always-Migrate, and Random Process. In the Greedy method, the BSi sends the whole task to BSj with a higher SNR in the Greedy method, and the division and cooperation of the tasks between i and j are not considered. In addition, in the Always-Migrate algorithm, the entire computing task offloaded by the SN is fully migrated to a BS which the SN is approaching. Therefore, in this algorithm, when another BS is closer to this SN, the whole task is migrated to the BS closer to this SN. In the Random Process, the BSi divides the task into two random components. One component is processed by BSi, and another component is transmitted to a nearby BS, BSj, for cooperation. However, the choice mechanism of BSj is not considered in this method, thus resulting in some of the components being lost. For example, when one BS moves out of the SN's transmission range, the result of the task will not be delivered to the SN, and the service session can be terminated in failure.

6.2.2. Total Service Time

Figure 5 shows the effect of the size of the computational task offloaded by the SN on the service time. The total service time comprises the computation time, offloading time, and propagation latency of the sub-task and computing results. Here, the size of the computational task in the dataset is a variable parameter and considers 18 values. In order to calculate the propagation delay, the distance between the SN and BS is assumed to be 100 m when the task is offloaded. Considering the mobility of BS, the distance between the SN and BS is assumed to be 300 m when the BS sends back the result of the computational task. In our proposed method, the BS with the shortest distance to the primary coalition node is selected for the secondary coalition node, so the link between the cooperation nodes is more reliable with our method; meanwhile, with any other three methods, the service session can terminate in failure probably with the movement of BSs since the BS can move out of the transmission range.

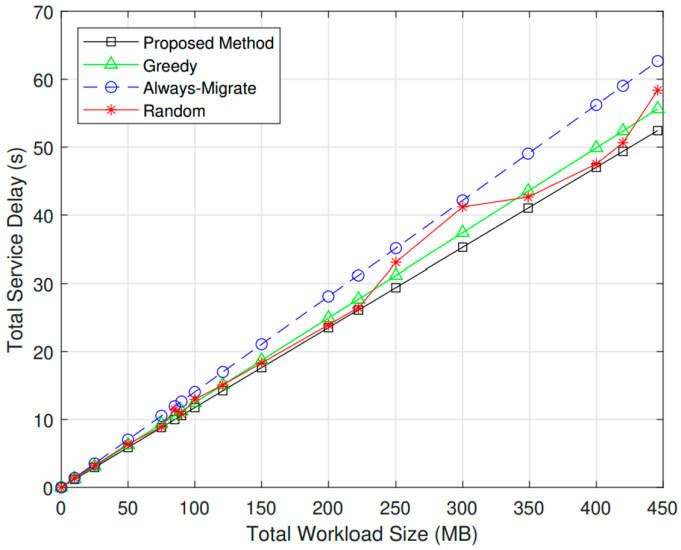


Figure 5. Overall service time vs. the total size of the computational task offloaded by the SN.

Figure 6 shows the effect of the number of SNs which offload different tasks to the same BSs on the total service time. Here, we consider three parameters, i.e., the size of computational task, the distance between SN and the primary BS, and the distance between the SN and the secondary BS. The computational size is 1000 MB, and the distance between the SN and the primary BS when offloading is 100 m. Alternatively, the distance is changed to be 300 m when the BS sends back the result to the SN after processing the task due to the mobility of BS. We assume that the number of SNs whose tasks are processed by the same BS is 14 in the experiment. It can be seen from Figure 6 that the total service time of the random scheme fluctuates between that of the Always-Migrate and our proposed scheme. Simultaneously, the overall service time of the Random Process is also near or equal to our proposed method. Nevertheless, it is never lower than that of our proposed method. Hence, our proposed method reduces the total service time. This is because, in our proposed method, the computational task is divided into two sub-tasks. Afterwards, the service time of each sub-task is calculated individually. From the combinations of service time of each sub-task, our proposed method chooses the divided sub-tasks with a lower service time. It should also be noted that the total service time can be further reduced by increasing the number of BSs for cooperative computing and parallel processing.

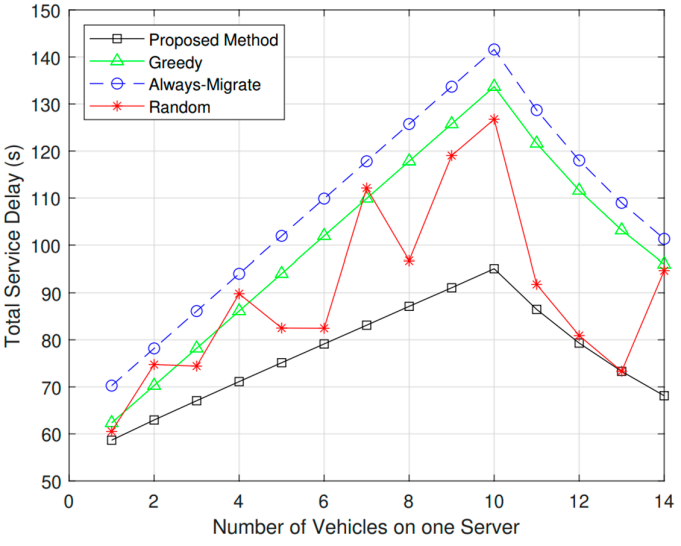


Figure 6. Overall service time vs. number of SNs on the same one BS.

6.2.3. Computational Time

Furthermore, we evaluated the computational time. The experiment results show that our proposed method outperforms the existing algorithms, i.e., Greedy, Always-Migrate, and Random, as in Figure 7. From Figure 7, we can see that the computational time is lower than that of the existing scheme due to the lower complexity of our proposed method. As we can see, when the total size of the workload is 300 MBs, the computational time is 15 s in our proposed method. At 300 MBs, the computational time of Always-Migrate is approximately 24 s, while that of the Random method has a computational time of 22 s. The computational time of the Greedy algorithm is 18 s. Hence, it is proved that, due to the coalition game theory, the cooperation of BSs, and the ANN method, the computational time of our proposed scheme is less than that of the existing schemes.

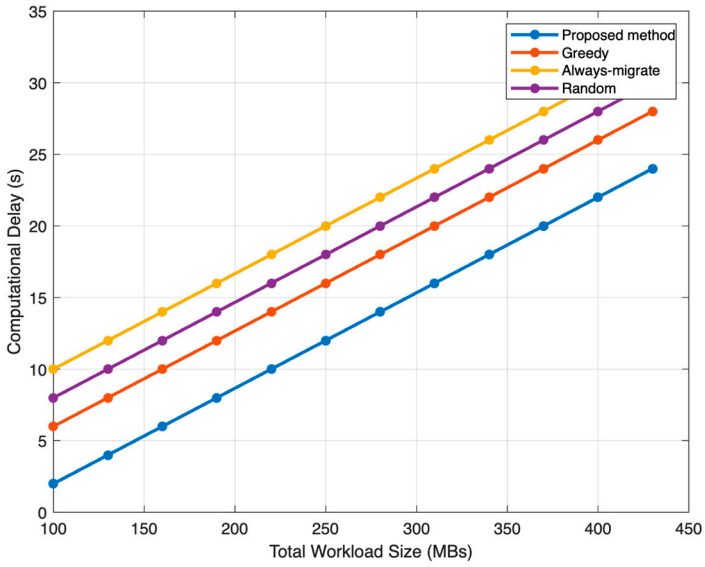


Figure 7. Computational time vs. the total workload size.

7. Conclusions

This work investigated the problem of computational task offloading based on alliance game cooperation in USNet and proposed an intelligent division scheme of task. To reduce the computational time and improve the service reliability, the divided task is offloaded to different BSs with the shortest distance. In order to implement cooperation among BSs, we propose a coalition game theory to process the divided computational task. Moreover, the component of the divided tasks was trained, tested, and validated by our proposed ANN model. The ANN was used because this method can reduce the complexity of our proposed algorithm. Consequently, we achieved an optimal solution from the trained ANN-based approach without calculating at every iteration. Finally, we compared our proposed scheme with the existing schemes, i.e., Greedy, Always-Migrate, and Random Process. It can be seen from the simulation results that our proposed approach outperforms these three schemes.

In the future, research will mainly focus on task offloading in underwater wireless network environments, where sensor nodes are passively moved by water flow and AUVs are actively moved.

Author Contributions: X.L. and X.D., methodology; S.Z., software; X.L. and D.H., validation; X.L., writing—original draft preparation; X.L. and X.D., writing—review and editing; X.D., supervision; X.D., funding acquisition. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China, Grant Number 61962052.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The authors declare that the data supporting the findings of this study are available within the article.

Acknowledgments: This work was supported in part by the National Natural Science Foundation of China, Grant No. 61962052; and in part by the Key Laboratory of IoT of Qinghai, under Grant 2022-ZJ-Y21.

Conflicts of Interest: The authors declare that they have no conflict of interest.

References

1. Liu, X.; Du, X.; Li, M.; Wang, L.; Li, C. A MAC Protocol of Concurrent Scheduling Based on Spatial-Temporal Uncertainty for Underwater Sensor Networks. *J. Sens.* **2021**, *2021*, 5558078. [CrossRef]
2. Li, S.; Qu, W.; Liu, C.; Qiu, T.; Zhao, Z. Sur vey on high-reliability wireless communication for underwater sensor networks. *J. Netw. Comput. Appl.* **2019**, *148*, 102446. [CrossRef]
3. Xiao, L.; Jiang, D.; Chen, Y.; Su, W.; Tang, Y. Reinforcement-Learning-Based Relay Mobility and Power Allocation for Underwater Sensor Networks Against Jamming. *IEEE J. Ocean. Eng.* **2020**, *45*, 1148–1156. [CrossRef]
4. Coutinho, R.W.L.; Boukerche, A.; Vieira, L.F.M.; Loureiro, A.A.F. Underwater wireless sensor networks: A new challenge for topology control based systems. *ACM Comput. Surv.* **2018**, *51*, 19. [CrossRef]
5. Jiang, S. On reliable data transfer in underwater acoustic networks: A survey from networking perspective. *IEEE Commun. Surv. Tutor.* **2018**, *20*, 1036–1055. [CrossRef]
6. Chen, X.; Wu, C.; Liu, Z.; Zhang, N.; Ji, Y. Computation Offloading in Beyond 5G Networks: A Distributed Learning Framework and Applications. *IEEE Wirel. Commun.* **2021**, *28*, 56–62. [CrossRef]
7. Qureshi, U.M.; Shaikh, F.K.; Aziz, Z.; Shah, S.M.Z.S.; Sheikh, A.A.; Felemban, E.; Qaisar, S.B. RF path and absorption loss estimation for underwater wireless sensor networks in different water environments. *Sensors* **2016**, *16*, 890. [CrossRef]
8. Cai, S.; Zhu, Y.; Wang, T.; Xu, G.; Liu, A.; Liu, X. Data Collection in Underwater Sensor Networks based on Mobile Edge Computing. *IEEE Access* **2019**, *7*, 65357–65367. [CrossRef]
9. Tran-Quang, V.; Miyoshi, T. A transmission range adjustment algorithm to avoid energy holes in wireless sensor networks. In Proceedings of the 8th Asia-Pacific Symposium on Information and Telecommunication Technologies, Kuching, Malaysia, 15–18 June 2010; pp. 1–6.
10. Liu, M.; Song, C. Ant-Based transmission range assignment scheme for energy hole problem in wireless sensor networks. *Int. J. Distrib. Sens. Netw.* **2012**, *8*, 290717. [CrossRef]
11. Afsar, M.M.; Crump, R.T.; Far, B.H. Energy Efficient Coalition Formation in Sensor Networks: A Game-Theoretic Approach. In Proceedings of the IEEE Canadian Conference of Electrical and Computer Engineering (CCECE), Edmonton, AB, Canada, 5–8 May 2019; pp. 1–6.
12. Zhao, J. Offloading Selection Based on Heterogeneous Utility in MEC Networks: A Coalition Formation Game-theoretic Approach. In Proceedings of the IEEE 6th International Conference on Computer and Communication Systems (ICCCS), Chengdu, China, 23–26 April 2021; pp. 469–472.
13. Chen, X.; Zhang, H.; Wu, C.; Mao, S.; Ji, Y.; Bennis, M. Optimized Computation Offloading Performance in Virtual Edge Computing Systems Via Deep Reinforcement Learning. *IEEE Internet Things J.* **2019**, *6*, 4005–4018. [CrossRef]
14. Zhang, M.; Cai, W. Energy-Efficient Depth Based Probabilistic Routing Within 2-Hop Neighborhood for Underwater Sensor Networks. *IEEE Sens. Lett.* **2020**, *4*, 7002304. [CrossRef]
15. Guan, Q.; Ji, F.; Liu, Y.; Yu, H.; Chen, W. Distance-Vector-Based Opportunistic Routing for Underwater Acoustic Sensor Networks. *IEEE Internet Things J.* **2019**, *6*, 3831–3839. [CrossRef]
16. Davari, N.; Aguiar, A.P. Real-Time Outlier Detection Applied to a Doppler Velocity Log Sensor Based on Hybrid Autoencoder and Recurrent Neural Network. *IEEE J. Ocean. Eng.* **2021**, *46*, 1288–1301. [CrossRef]
17. Waqas, M.; Tu, S.; Halim, Z.; Rehman, S.; Abbas, G.; Abbas, Z.H. The Role of Artificial Intelligence and Machine Learning in Wireless Networks Security: Principle, Practice and Challenges. *Artif. Intell. Rev.* **2022**, *55*, 5215–5261. [CrossRef]
18. Zhou, S.; Jadoon, W.; Shuja, J. Machine Learning Based Offloading Strategy for Lightweight User Mobile Edge Computing Tasks. *Complexity* **2021**, *2021*, 6455617. [CrossRef]
19. Panigrahi, C.R.; Sarkar, J.L.; Pati, B.; Buyya, R.; Mohapatra, P.; Majumder, A. Mobile Cloud Computing and Wireless Sensor Networks: A review, integration architecture, and future directions. *IET Netw.* **2021**, *10*, 141–161. [CrossRef]
20. Haghighi, V.; Moayedian, N.S. An Offloading Strategy in Mobile Cloud Computing Considering Energy and Delay Constraints. *IEEE Access* **2018**, *6*, 11849–11861. [CrossRef]
21. Rahbari, D.; Nickray, M. Computation Offloading and Scheduling in Edge-Fog Cloud Computing. *J. Electron. Inf. Syst.* **2019**, *1*, 26–36. [CrossRef]

22. Zaman, S.; Jehangiri, A.I.; Maqsood, T.; Haq, N.U.; Umar, A.I.; Shuja, J.; Ahmad, Z.; Dhaou, I.B.; Alsharekh, M.F. LiMPO: Lightweight mobility prediction and offloading framework using machine learning for mobile edge computing. *Clust. Comput.* **2023**, *26*, 99–117. [CrossRef]
23. Meng, H.; Wang, S.; Gao, F.; Lu, J.; Liu, Y.; Mei, Y. Edge Computing Task Offloading Method for Load Balancing and Delay Optimization. In *ACM TURC'21: Proceedings of the ACM Turing Award Celebration Conference, Hefei, China, 30 July–1 August 2021*; Association for Computing Machinery: New York, NY, USA, 2021; pp. 173–178.
24. Saeik, F.; Avgeris, M.; Spatharakis, D.; Santi, N.; Dechouniotis, D.; Violos, J.; Leivadeas, A.; Athanasopoulos, N.; Mitton, N.; Papavassiliou, S. Task offloading in Edge and Cloud Computing: A survey on mathematical, artificial intelligence and control theory solutions. *Comput. Netw.* **2021**, *195*, 108177. [CrossRef]
25. Su, Y.; Fan, W.; Liu, Y.; Wu, F. Game-based Pricing and Task Offloading in Mobile Edge Computing Enabled Edge-Cloud Systems. *arXiv* **2021**, arXiv:2101.05628.
26. Zhai, Y.; Ding, B.; Zhang, P.; Luo, J.; Wu, Q.; Shi, P.; Wang, H. Cooperative Offloading for Multiple Robot Applications. In Proceedings of the 2020 IEEE International Conference on Joint Cloud Computing (JCC), Oxford, UK, 3–6 August 2020.
27. Zhong, S.; Guo, S.; Yu, H.; Wang, Q. Cooperative service caching and computation offloading in multi-access edge computing. *Comput. Netw.* **2021**, *189*, 107916. [CrossRef]
28. Peng, J.; Qiu, H.; Cai, J.; Xu, W.; Wang, J. D2D-Assisted Multi-User Cooperative Partial Offloading, Transmission Scheduling and Computation Allocating for MEC. *IEEE Trans. Wirel. Commun.* **2021**, *20*, 4858–4873. [CrossRef]
29. Mohammed, A.; Nahom, H.; Tewodros, A.; Habtamu, Y.; Hayelom, G. Deep Reinforcement Learning for Computation Offloading and Resource Allocation in Blockchain-Based Multi-UAV-Enabled Mobile Edge Computing. In Proceedings of the 2020 17th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP), Chengdu, China, 18–20 December 2021.
30. Yu, X.-Y.; Niu, W.-J.; Zhu, Y.; Zhu, H.-B. UAV-assisted cooperative offloading energy efficiency system for mobile edge computing. *Digit. Commun. Netw.* **2022**, *in press*. [CrossRef]
31. Kuang, Y.; Sun, J.; Gan, X.; Gong, D.; Liu, Z.; Zha, M. Dynamic Multi-objective Cooperative Coevolutionary Scheduling for Mobile Underwater Wireless Sensor Networks. *Comput. Ind. Eng.* **2021**, *156*, 107229. [CrossRef]
32. Boukerche, A.; Sun, P. Design of Algorithms and Protocols for Underwater Acoustic Wireless Sensor Networks. *ACM Comput. Surv.* **2020**, *53*, 134. [CrossRef]
33. Du, Z.; Chai, H.; Xiang, M.; Zhang, F.; Hui, J.; Wang, Z. Cooperative Localization of UUVs with Decentralized Method Based on Underwater Weak Communication. In Proceedings of the China Satellite Navigation Conference, Beijing, China, 22–25 May 2022; Springer: Singapore, 2022.
34. Tu, S.; Waqas, M.; Ur Rehman, S.; Mir, T.; Abbas, G.; Abbas, Z.H.; Halim, Z.; Ahmad, I. Reinforcement Learning Assisted Impersonation Attack Detection in Device to-Device Communications. *IEEE Trans. Veh. Technol.* **2021**, *70*, 1474–1479. [CrossRef]
35. Mustafa, E.; Shuja, J.; Bilal, K.; Mustafa, S.; Maqsood, T.; Rehman, F.; Khan AU, R. Reinforcement learning for intelligent online computation offloading in wireless powered edge networks. *Clust. Comput.* **2022**, *26*, 1053–1062. [CrossRef]
36. He, Z.; Wen, X.; Liu, H.; Du, J. A comparative study of artificial neural network, adaptive neuro fuzzy inference system and support vector machine for forecasting river flow in the semiarid mountain region. *J. Hydrol.* **2014**, *509*, 379–386. [CrossRef]
37. Gozutok, A.A.; Yilmaz, U.C.; Demirel, S. Preliminary results on estimation of signal fading on telecommunication satellite telemetry signals with hybrid numerical weather prediction and artificial neural network approach under presence of aerosol effect. *Int. J. Satell. Commun. Netw.* **2022**, *40*, 305–316. [CrossRef]
38. Joshi, P.; Shekhar, M.S.; Kumar, A.; Quamara, J.K. Artificial neural network model for precipitation forecast over Western Himalaya using satellite images. *Mausam* **2022**, *73*, 83–90. [CrossRef]
39. Zanaj, E.; Gambi, E.; Zanaj, B.; Disha, D.; Kola, N. Underwater Wireless Sensor Networks: Estimation of Acoustic Channel in Shallow Water. *Appl. Sci.* **2020**, *10*, 6393. [CrossRef]
40. Zanaj, E.; Enesi, I.; Zanaj, B.; Lumi, J. Impact of Node Mobility at Underwater WSN. In Proceedings of the SoftCOM 2017, Split, Croatia, 21–23 September 2017.
41. Qiao, G.; Zhao, C.; Zhou, F.; Ahmed, N. Distributed Localization Based on Signal, Propagation Loss for Underwater Sensor Networks. *IEEE Access* **2019**, *7*, 112985–112995. [CrossRef]
42. Sehgal, A.; Tumar, I.; Schonwalder, J. Vari ability of Available Capacity Due to the Effects of Depth and Temperature in the Underwater Acoustic Communication Channel. In Proceedings of the OCEANS, Bremen, Germany, 11–14 May 2009.
43. Abdullah, Z.; Naz, S.; Raja, M.A.Z.; Zameer, A. Design of Wideband Tonpilz Transducers for Underwater SONAR Applications with Finite Element Model. *Appl. Acoust.* **2021**, *183*, 108293. [CrossRef]
44. Waqas, M.; Ahmed, M.; Li, Y.; Jin, D.; Chen, S. Social-Aware Secret Key Generation for Secure Device-to-Device Communication via Trusted and Non-Trusted Relays. *IEEE Trans. Wirel. Commun.* **2018**, *17*, 3918–3930. [CrossRef]
45. Nabavi, S.R.; Eraghi, N.O.; Torkestani, J.A. WSN Routing Protocol Using a Multiobjective Greedy Approach. *Wirel. Commun. Mob. Comput.* **2021**, *2021*, 6664669. [CrossRef]
46. Muzzammil, M.; Ahmed, N.; Qiao, G.; Ullah, I.; Wan, L. Fundamentals and Advancements of Magnetic-Field Communication for Underwater Wireless Sensor Networks. *IEEE Trans. Antennas Propag.* **2020**, *68*, 7555–7570. [CrossRef]
47. Jain, S.; Ali, M.M. Estimation of Sound Speed Profiles Using Artificial Neural Networks. *IEEE Geosci. Remote Sens. Lett.* **2006**, *3*, 467–470. [CrossRef]

48. Liu, Z.; Waqas, M.; Yang, J.; Rashid, A.; Han, Z. A Multi-Task CNN for Maritime Target Detection. *IEEE Signal Process. Lett.* **2021**, *28*, 434–438. [CrossRef]
49. Mustafa, E.; Shuja, J.; Zaman, S.K.; Jehangiri, A.I.; Din, S.; Rehman, F.; Mustafa, S.; Maqsood, T.; Khan, A.N. Joint wireless power transfer and task offloading in mobile edge computing: A survey. *Clust. Comput.* **2021**, *25*, 2429–2448. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.