sensors

MDPI

Article

A Distributed and Energy-Efficient Algorithm for Event K-Coverage in Underwater Sensor Networks

Peng Jiang *, Yiming Xu and Jun Liu

College of Automation, Hangzhou Dianzi University, Hangzhou 310018, China; xymhdu@163.com (Y.X.); liujunhdu@163.com (J.L.)

* Correspondence: pjiang@hdu.edu.cn; Tel.: +86-571-8691-9131 (ext. 512); Fax: +86-571-8687-8566

Academic Editor: Leonhard M. Reindl

Received: 7 October 2016; Accepted: 10 January 2017; Published: 19 January 2017

Abstract: For event dynamic K-coverage algorithms, each management node selects its assistant node by using a greedy algorithm without considering the residual energy and situations in which a node is selected by several events. This approach affects network energy consumption and balance. Therefore, this study proposes a distributed and energy-efficient event K-coverage algorithm (DEEKA). After the network achieves 1-coverage, the nodes that detect the same event compete for the event management node with the number of candidate nodes and the average residual energy, as well as the distance to the event. Second, each management node estimates the probability of its neighbor nodes' being selected by the event it manages with the distance level, the residual energy level, and the number of dynamic coverage event of these nodes. Third, each management node establishes an optimization model that uses expectation energy consumption and the residual energy variance of its neighbor nodes and detects the performance of the events it manages as targets. Finally, each management node uses a constrained non-dominated sorting genetic algorithm (NSGA-II) to obtain the Pareto set of the model and the best strategy via technique for order preference by similarity to an ideal solution (TOPSIS). The algorithm first considers the effect of harsh underwater environments on information collection and transmission. It also considers the residual energy of a node and a situation in which the node is selected by several other events. Simulation results show that, unlike the on-demand variable sensing K-coverage algorithm, DEEKA balances and reduces network energy consumption, thereby prolonging the network's best service quality and lifetime.

Keywords: event K-coverage; distributed algorithm; energy-efficient; sensing radius adjusting

1. Introduction and Related Works

Underwater wireless sensor networks (UWSNs) are network monitoring systems that consist of sensor nodes with self-organized perception, acoustic communication, and computation capabilities in an underwater environment. UWSNs can be applied to event detection in underwater environments, such as water pollution monitoring, underwater disaster warnings, and the detection of different types of underwater biology [1–3]. Such applications require UWSNs for event detection, as they have strong fault tolerance and robustness. Moreover, UWSNs have a network structure with a 3D distribution, acoustic signals as a communication medium, limited node energy, and a high cost for deployment [4,5]. Because of the limited node energy, energy-efficient performance must be considered when designing algorithms or protocols for wireless sensor networks [6–8]. For example, the authors of [9] proposed that the sensor networks should be clustered unevenly to implement energy consumption evenly among cluster head nodes, achieving effective use of the energy of the node; the authors of [10] organized sensors in cooperative groups to reduce the number of messages transmitted inside the network, thereby reducing the energy consumption of the network. Similarly, K-coverage in UWSNs should also consider energy-efficient performance.

K-coverage, in which each target monitoring area (or event) is required to at least be covered by K-sensor nodes (K $\geq$ 1), can achieve network redundancy detection for events. It is one of the technologies that are commonly used to improve network fault tolerance and robustness [11].

K-coverage has been studied extensively by some researchers [12–16]. Kumar et al. [17] studied how the fewest nodes can be used to ensure network K-coverage at high probability for a monitoring area, for deterministic and stochastic network deployment. Chen et al. [18] proposed a probability-based K-coverage control approach for 3D WSNs. This approach first models the 3D monitoring area by grid. Then, it uses a greedy iteration method to determine the position of each node in the grid until either the node number reaches the preset value or all grid points are covered by K-nodes at a certain probability. Compared with K-coverage methods with deterministic or stochastic network deployment, the algorithm with fewer nodes achieves the same coverage level of a monitoring area, but the overall network coverage could still be improved. Kim et al. [19] proposed a randomly ordered activation and layering protocol for ensuring network K-coverage. This protocol divides all network nodes into several disjoint sets (each disjoint set covers the monitoring area by 1-coverage), i.e., by layering and then randomly selecting K-layer nodes to work together for the network to achieve K-coverage within a certain period. This algorithm can achieve high-coverage level for a monitoring area. Habib M et al. [20] proposed a K-coverage algorithm for a 3D area based on geometric properties. The algorithm firstly analyzes the condition of the 3D area guaranteed to be K-coverage based on Helly's Theorem and the Reuleaux tetrahedron model, and, on this basis, the corresponding nodes are selected to be active to complete the K-coverage of the network. The algorithm can fully complete K-coverage in a 3D field and increase the lifetime of the network by scheduling the working mode of nodes. A similar idea is studied by Gupta et al. [21]. To prolong the network lifetime, they present a node scheduling strategy for K-coverage. It uses probability to schedule the node to minimize the number of active nodes, according to the number of neighbor nodes and the stage of the node (communication and sensing radius). Pal et al. [22] proposed a K-coverage algorithm based on the Sixsoid. The algorithm finds that using the Sixsoid to fill the 3D area has less overlapping volume than using the Reuleaux tetrahedron model and, according to the characteristic of the Sixsoid, deploys the node in the target area. The algorithm can reduce the number of active nodes, but it can run well also based on the assumption of high density deployment. These algorithms described here can achieve K-coverage for the monitoring area and not the event and will cause numerous idle nodes, thereby wasting energy when they are used in event monitoring. In addition, considering the UWSNs characteristic of high cost and sparse deployment [23], these algorithms, where large numbers of sensor nodes are needed, are unsuitable for event K-coverage for UWSNs.

Several works exist in the literature that covers the target or event on demand by the mobile nodes. Liu et al. [24] uses game theory to determine the optimal mobility strategy of each node to complete the dynamic coverage of the network. Chen [25] presents the energy-effective movement algorithm. The algorithm divides the target area into subareas and then selects nodes to one of the subareas to minimize the movement of nodes. Habib [26] proposed a mission-oriented K-coverage algorithm for mobile WSNs. This algorithm first calculates where the nodes move by using Helly's theorem and the geometric analysis of the Reuleaux triangle. Then, it adopts the distributed strategy to select a certain number of nodes to move to those positions to achieve K-coverage for the area. The approach reduces the number of nodes deployed in UWSNs and guarantees good event coverage. Xia et al. [27] proposed the fish swarm-inspired underwater sensor deployment algorithm (FSSDA), which calculates the position that nodes will move to by the method inspired by the fish feeding behavior, and combines the position with the crowded degree control to achieve coverage requirements for the events. The algorithm has low complexity and a quick convergence rate, but it considers only the required coverage for static events and ignores the dynamic and uncertainty of the events. Du et al. [28] proposed the particle swarm-inspired underwater sensor deployment algorithm, which is similar to the FSSDA, and considers event dynamic to make the algorithm more practical. Although the algorithm is the same as other event K-coverage algorithms by mobile nodes, it greatly reduces the number of nodes in network deployment. However, it also consumes considerable energy because of the nodes' mobility [29]. Alam et al. [30] proposed the on-demand variable sensing K-coverage algorithm (OVSKA), which implements event dynamic K-coverage in the network by adjusting the sensing radius of node and analyzes its own practical application. The OVSKA better reduces the number of nodes and redundant energy consumption by the dynamic event K-coverage method compared with the static method. It also effectively avoids energy consumption produced by node mobility by adjusting the sensing radius. However, for how to select the assistant nodes to achieve the K-coverage for each event, the paper only proposed a greedy algorithm, in which each management node selects its assistant nodes from its neighbor nodes to minimize the ratio of network energy consumption of its neighbor nodes to detection performance of the events it manages. The process of selecting adjusting nodes is simple. However, when the number of events is relatively large, the neighbor node of the management node can cover some events, which several management nodes manage by adjusting the sensing radius. At this time, in the method, a management node cannot consider a situation in which other management nodes may select its neighbor node to cover the event they manage. As a result, the number of nodes that broaden their sensing radius increases and network energy consumption to some extent increases. In addition, the method selects the assistant node, ignoring the residual energy of the node and causing some nodes to be scheduled frequently and die early. This approach makes the network energy consumption unbalanced, degrades the quality of network service, and shortens network lifetime.

To solve the aforementioned problems, this study proposes the distributed and energy-efficient event K-coverage algorithm (DEEKA). The monitoring area achieves 1-coverage or approximately 1-coverage. At the start of each round of the algorithm, nodes that detect the same event compete for the management node of the event by the indicator, including the number of candidate nodes, the average residual energy of neighbor nodes, and the distance to the event. Each management node then calculates the probability of each neighbor node's being selected by the corresponding event it manages. Then, each management node builds a multi-objective optimization model with regard to the energy consumption expectation and residual energy variance of its neighbor nodes, and to the detected performance for the events it manages as targets. Afterwards, each management node obtains Pareto solutions by using the constrained elitist non-dominated sorting genetic algorithm (NSGA-II) method and selects the best strategy by using the technique for order preference by similarity to an ideal solution (TOPSIS) method according to the target bias of practical application. The algorithm can make some nodes adjust the sensing radius to cover events that different management nodes manage synchronously, thereby reducing the number of nodes that increase their sensing radius and energy consumption over the whole network. In addition, determining management nodes by competition is beneficial for balancing residual energy of the neighbor nodes among different management nodes. When each management node selects its assistant nodes, it considers the residual energy of its neighbor nodes. This approach is beneficial for balancing energy consumption of its neighbor nodes. Simulation results show that, compared with the OVSKA, the DEEKA can better balance and reduce network energy consumption, thereby prolonging the network's best service quality and lifetime.

The rest of this paper is organized as follows: Section 2 describes the system model, assumptions, and definitions considered in this study. Section 3 presents details of the DEEKA. Section 4 analyzes the complexity of the DEEKA. Section 5 discusses the performance study and provides a detailed analysis of its result. Finally, Section 6 concludes the paper and presents future research directions.

2. Preliminaries: Models and Definitions

2.1. Models

2.1.1. Network Model

A monitoring area achieves 1-coverage or approximately 1-coverage by N nodes. Meanwhile, these nodes are anchored with weights to keep static in current positions. It is evenly divided into several sub-areas. Additionally, it has a mobile intermediate node (e.g., Autonomous Underwater Vehicle, AUV) that serves as the router that receives information from management nodes and then transmits that information to the sink node in every sub-area [31]. Every assistant node adjusts to the appropriate sensing radius to help the corresponding management node monitor the event together and sends the event sensing information to the management node. The management node obtains the result about the event by using the fusion algorithm and waits for the mobile intermediate node to gather this information. After the event leaves or disappears, the corresponding assistant nodes revert to the minimum sensing radius. The network model is depicted in Figure 1. Every AUV is responsible for handling the information of the management node with the same color as the AUV and for transmitting the information to the sink node.

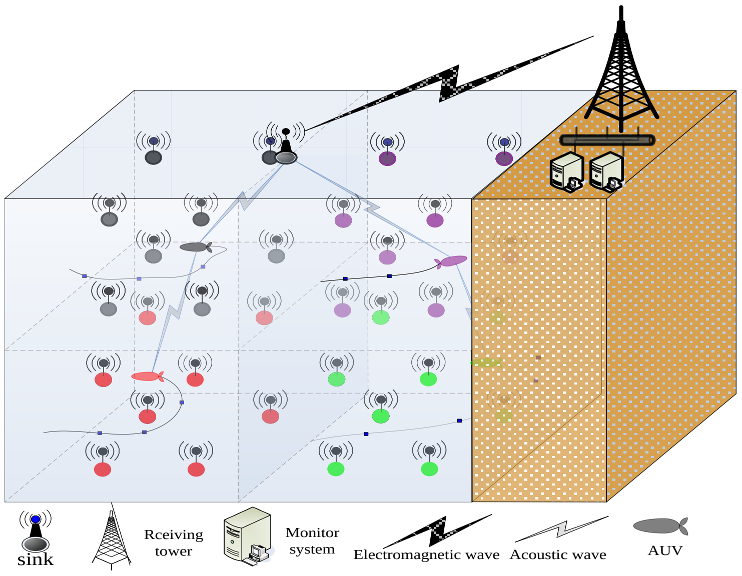

Figure 1. Underwater wireless sensor network (UWSN) system model.

This study denotes the i-th node by s_i, and the corresponding node set $S = \{s_1, s_2, \ldots s_n\}$, as well as the j-th event by e_j and the corresponding event set $E = \{e_1, e_2, \ldots e_z\}$. Moreover, the number of elements in the set S is N and that in the set E is Z. The following assumptions are considered:

1. Any node has the ability of communication and perception. Nodes communicate with one another through acoustic channels, and the sink node communicates with the ground monitoring station by radio.
2. All nodes are isomorphic before the algorithm runs. Then, the sensing radius of each node can be adjusted between the minimum $R_s^{\min}$ and maximum sensing radius $R_s^{\max}$ according to adjustment strategy.
3. The position of each event is randomly changed by the water current in the monitoring area but not beyond that. Time interval T is one round of network run. In every round, the algorithm readjusts the sensing radius of some corresponding nodes to achieve K-coverage for all events whose position is changed.

2.1.2. Network Energy Consumption Model

In sensor networks, each node is mainly responsible for sensing the surrounding environment and transmitting the sensing data. Therefore, in this study, the node's energy consumption includes sensing and communication [32,33].

Energy Consumption Model of Communication

Underwater sensor network nodes communicate with one another through acoustic signals. Accordingly, this study uses the energy consumption model of underwater sensor network data communication by utilizing sound wave as the medium [33]. Underwater acoustic signal attenuation model $SA(d)$ is given as follows:

$$SA(d) = d^{\lambda} \cdot \alpha^{d}. \tag{1}$$

Equation (1) describes energy attenuation when the transmission distance of data packet is d, where λ is the energy diffusion factor (cylindrical diffusion is 1, actual situation is 1.5, and spherical diffusion is 2). Parameter $\alpha = 10^{AC(f)/10}$, which is determined by absorption coefficient $AC(f)$, which is shown as follows:

$$AC(f) = 0.11\frac{10^{-3}f^2}{1+f^2} + 44\frac{10^{-3}f^2}{4100+f^2} + 2.75 \times 10^{-7}f^2 + 3 \times 10^{-6} \tag{2}$$

where f is the carrier frequency with the kHz unit. Absorption coefficient unit is dB/m.

The energy consumption of communication on the distance d ($ECC(d)$) is expressed as follows:

$$ECC(d) = P_r \times T_p \times SA(d) \tag{3}$$

where T_p is the data transmission time, and P_r is the minimum power packets that can be received.

Energy Consumption Model of Sensing

According to Reference [34], the ratio of sensing power to communication power is r_p ($0 < r_p < 1$). Thus, in this study, the energy consumption model of sensing on the distance d $ECS(d)$ is shown as follows:

$$ECS(d) = r_p \times P_r \times SA(d) \times T. \tag{4}$$

T is the time interval of the algorithms running.

2.1.3. Node Sensing Model

At present, the sensing model for the event can be divided into two categories: the Boolean sensing model and the probability sensing model. The former is unsuitable for practical applications because it is too idealistic. Therefore, this study uses the improved probability sensing model, which considers sensing ability characteristics of the sensor and interference characteristics of noise in the environment.

At first, the model considers sensor characteristics, i.e., the sensing performance of detecting the abnormal signal, including event and noise signals, which is good within a certain range but degrades with distance beyond the range [35]. Sensing probability of an event is formulated as follows:

$$p\left(d\left(s_i, e_j\right)\right) = \begin{cases} 1 & \text{if } d\left(s_i, e_j\right) \le R_s^{\min} \\ e^{-\gamma_1\left(d\left(s_i, e_j\right) - R_s^{\min}\right)^{\gamma_2}} & \text{if } R_s^{\min} < d\left(s_i, e_j\right) < R_s^{\max} \\ 0 & \text{if } d\left(s_i, e_j\right) \ge R_s^{\max}. \end{cases} \tag{5}$$

where γ_1 and γ_2 are specific parameters of the sensing device that are determined from physical properties of the sensor. They determine the speed of sensing probability decay beyond $R_s^{\min}$.

The judgment result of one sensor node to the event is vulnerable to environmental noise (i.e., noise signal). Thus, the influence of environmental noise on the current sensing probability of an event should be considered. Assuming that the noise signal can be expressed by Gaussian distribution $\Psi(0,1)$, whose average value is 0 and variance is 1, the abnormal signal threshold for event detection is η_t. When the signal strength is lower than η_t, the probability that the event is correctly sensed by the node reduces quickly. Therefore, the probability $P(s_i,e_j)$ that an event e_j is correctly sensed by node s_i is formulated as follows [30]:

$$P(s_i, e_j) = p(d(s_i, e_j)) \int_{\eta_t}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-(x-u(s_i))^2/2}\, dx \tag{6}$$

where $u(s_i)$ is the abnormal signal strength (including event and noise signals) node s_i receives, and $d(s_i, e_j)$ is the distance between s_i and e_j.

2.1.4. Event Mobility Model

The event randomly moves after it appears within the monitoring area through the water current. Assuming that the position coordinate of the event at t time is $(x(t), y(t), z(t))$, the event mobility model is

$$\begin{aligned} x(t) &= x(t-1) + d_x(t) v_{cx} \\ y(t) &= y(t-1) + d_y(t) v_{cy} \\ z(t) &= z(t-1) + d_z(t) v_{cz} \end{aligned} \tag{7}$$

where

$$d_x(t) = \begin{cases} 1 & t = 0 \\ -d_x(t-1) & t > 0 \end{cases}, \; d_y(t) = \begin{cases} 1 & t = 0 \\ -d_y(t-1) & t > 0 \end{cases}, \; d_z(t) = \begin{cases} 1 & t = 0 \\ -d_z(t-1) & t > 0 \end{cases}.$$

v_{cx}, v_{cy}, and v_{cz} are the flow speed of x-axial direction, y-axial direction, and z-axial direction, respectively. Their values are all randomly produced between 0 and maximum speed $V_{\max}$.

2.2. Definition

2.2.1. Nodes and Events

Neighbor Node: A node, whose distance to the node s_i is within its communication radius R_c, is s_i's neighbor node, and all of s_i's neighbor nodes form the set of the neighbor node, denoted by $S_N(s_i)$.

Management Node: The management node of one event e_j is produced as discussed in Section 3.2.1 and is responsible for scheduling its corresponding neighbor nodes to cover e_j on demand. In other words, e_j is an event that the management node manages. In addition, all of the management nodes form the set of the management node, denoted by M, where $M = \{m_1, m_2, \ldots m_m\}$, and m_m is the m-th element of M, namely, the m-th management node. Similarly, all of the events that m_m manages form the set denoted by $E_M(m_m)$. In Figure 2, s_2 is the management node of e_2; the management node of e_1 is generated from the competition between s_1 and s_2.

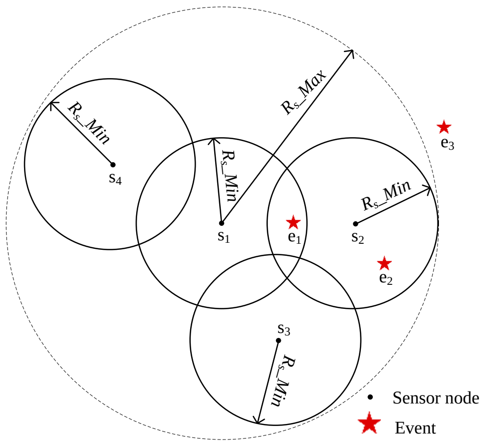

Figure 2. Sectional view of the relationship between nodes and events.

Candidate Node: The candidate node of one event e_j is defined as the node in the neighbor node set of e_j's management node $S_N(m_m)$, which can adjust its sensing radius to cover e_j. In other word, the candidate node of e_j is in the neighbor nodes of e_j's management node m_m, and its distance to e_j is less than $R_s^{\max}$. Therefore, when the management node of e_j is m_m, the set of the candidate node of e_j can be denoted by $m_m.C(e_j)$ and formulated as follows:

$$m_m.C(e_j) = \{s_i | d(s_i, e_j) < R_s^{\max}\, and\, s_i \in S_N(m_m)\}. \tag{8}$$

Because the management node of each event is determined in the study, except in Section 3.2.1, the study uses $C(e_j)$ to replace $m_m.C(e_j)$ for simplified representation. Needless to say, the number of e_j's candidate nodes is the number of elements in $C(e_j)$, denoted by $|C(e_j)|$. In Figure 2, candidate nodes of e_1 and e_2 for events e_1 and e_2 are both s_1 and s_2.

Dynamic Candidate Node: According to the description of the candidate node, the node whose distance to an event e_j is between $R_s^{\min}$ and $R_s^{\max}$ is defined as the dynamic candidate node, and the kind of the node forms the set of the dynamic candidate node, denoted by $C_d(e_j)$:

$$C_d(e_j) = \left\{ s_i \middle| R_s^{\min} < d(s_i, e_j) \le R_s^{\max},\, and\, s_i \in C(e_j) \right\} \tag{9}$$

where the i-th element in $C_d(e_j)$ is assigned as c_d^i. Clearly, the number of dynamic candidate nodes of e_j is denoted by $|C_d(e_j)|$. As shown in Figure 2, the dynamic candidate node for event e_1 is s_3, and, for event e_2, dynamic candidate nodes are s_1 and s_3.

Static Candidate Node: Similarly, the static candidate node of an event e_j is the node in e_j's candidate node set, whose distance to e_j is within $R_s^{\min}$. Correspondingly, the kind of the node forms the set of the static candidate node, denoted by $C_s(e_j)$ and formulated as follows:

$$C_s(e_j) = \left\{ s_i \middle| d(s_i, e_j) \le R_s^{\min},\, and\, s_i \in C(e_j) \right\} \tag{10}$$

where the i-th element in $C_s(e_j)$ is assigned as c_s^i. Clearly, the number of dynamic candidate nodes of e_j is denoted by $|C_s(e_j)|$. In Figure 2, the static candidate node for event e_1 is s_1, while event e_2 does not have a static candidate node.

The difference between the dynamic candidate node and the static candidate node is the distance between the node and the event. Because of the difference, the former needs to adjust its sensing radius to cover the event. However, the latter does not need to do this.

Assistant Node: The assistant node of one event e_j is defined as the node, which is scheduled by the management node and assists the management node to cover e_j. In this paper, the assistant node is also described as selected by e_j, and all of the nodes form the set of the assistant node, denoted by $A(e_j)$.

Whether or not the sensing radius is adjusted, the assistant node can be divided into two categories: dynamic and static. The dynamic assistant node assists the management node to detect the event by increasing its sensing radius. Conversely, the static assistant node does not need to increase its sensing radius. In Figure 2, s_1 and s_3 are the selected assistant nodes of event e_1. Between them, s_1 is the static assistant node of e_1, and s_3 is its dynamic assistant node.

Static Coverage Event: The static coverage event of one node s_i is defined as an event whose distance to s_i is within minimum sensing radius $R_s^{\min}$, and all of the events form the set of the static coverage event, denoted by $E_s(s_i)$ and formulated as follows:

$$E_s(s_i) = \left\{ e_j \middle| d(s_i, e_j) \le R_s^{\min} \right\}. \tag{11}$$

Clearly, the number of s_i's static coverage event is the number of elements in $E_s(s_i)$, denoted by $|E_s(s_i)|$. In Figure 2, the static coverage event for node s_1 is e_1.

Dynamic Coverage Event: The static coverage event of one node s_i is defined as an event whose distance to s_i is between $R_s^{\min}$ and $R_s^{\max}$, and all of the events form the set of the dynamic coverage event, denoted by $E_d(s_i)$ and formulated as follows:

$$E_d(s_i) = \left\{ e_j \middle| R_s^{\min} < d(s_i, e_j) \le R_s^{\max} \right\}. \tag{12}$$

Clearly, the number of s_i's dynamic coverage event is the number of elements in $E_d(s_i)$, denoted by $|E_d(s_i)|$. In Figure 2, e_1 is the static coverage event of node s_1. Both e_1 and e_2 are the dynamic coverage events of s_3.

The difference between the static coverage event and the dynamic coverage event can be summarized as follows. As shown in Equation (11), the event can be sensed by the node directly, because the distance between them is within the minimum sensing radius. Thus, the kind of the node is named after the static coverage event. However, the event described in Equation (12) wants to be covered by the node, which needs the node to adjust its sensing radius. Thus, the kind of the node is named after the dynamic coverage event.

Actual Dynamic Coverage Event: The actual dynamic coverage event is the event that s_i actually adjusts its sensing radius to cover after the algorithm runs, and the set is assigned as $E_{ad}(s_i)$. The element number of the set is assigned as $|E_{ad}(s_i)|$.

The difference between the dynamic coverage event and the actual dynamic coverage event is that an event is the dynamic coverage event of s_i, but it does not have to be the actual dynamic coverage event of s_i. Therefore, the number of actual dynamic coverage events is usually less than that of dynamic coverage events.

2.2.2. Event Detection Performance

Event detection performance reflects the quality of the network service and the robustness of the network to detect events. It is defined as the correct sensing probability of event e_j. Event detection performance is judged by information fusion center (i.e., the management node in this study) under the circumstance that information may be lost in the transmission process because of the influence of some interference in practice. The information fusion center uses the rule of k_1 out of K [11]. Thus, it considers event e_j to be detected correctly when at least k_1 nodes in K-nodes detect e_j correctly. Assuming that the failure probability of information transmission between two nodes is p_{lost}, the probability that the information of node s_i correctly detecting e_j is successful to be transmitted to the fusion center is expressed as follows and is denoted by $P_{reach}(s_i,e_j)$:

$$P_{reach}(s_i, e_j) = (1 - p_{lost}) \cdot P(s_i, e_j). \tag{13}$$

The probability of the fusion center correctly detecting event e_j is formulated and is denoted by $P_{result}(e_j)$.

$$P_{result}(e_j) = \sum_{b=k_1}^{K} \left[\sum_{C_{\mathrm{K},b}} \left(\prod_{i=1}^{b} P_{reach}(a_i, e_j) \prod_{i=b+1}^{K} (1 - P_{reach}(a_i, e_j)) \right) \right] \tag{14}$$

where $C_{\mathrm{K},b}$ is the value of combinations of b nodes in K-nodes correctly detecting events; a_i is i-th elements in e_j's assistant node set $A(e_j)$; the order of the element in $A(e_j)$ is changed by different combinations. In each combination, the nodes that detect the event correctly are in the front, while others are at the back.

2.2.3. Network Lifetime

Network lifetime is an important basis for evaluating the effectiveness of the energy algorithm [36] and is denoted by L_t. In this study, the network lifetime is defined as the value of the network running round until the ratio of the number of survival nodes to the total number of nodes achieves r_n

(0 evr_n < 1), where r_n is the proportional threshold of surviving nodes. When the ratio is less than r_n, the network cannot complete normal monitoring function, and network running ends.

3. Algorithm Description and Process

3.1. Problem Description

Some research has been conducted on the K-coverage problem of sensor networks [12]. Most existing algorithms for static K-coverage of an area need numerous nodes. The existing algorithms also cause great redundant energy consumption for event K-coverage in UWSNs. Others achieve event K-coverage by assisting mobile nodes, which consumes considerable energy because of nodes moving. A dynamic event K-coverage method, by adjusting the sensing radius of nodes, can reduce the required number of deployment nodes. This approach also considerably reduces the redundant energy consumption produced by the moving nodes. For assistant node selection, each management node selects the corresponding assistant nodes to achieve K-coverage and minimizes the ratio of network energy consumption to the detecting performance of the event it manages. The method can achieve a relatively good result when the neighbor node of the management node can only cover the event that the management node manages on its own. However, when the number of events is relatively large, the neighbor node can usually cover more than one event that different management nodes manage. At this condition, the method will also increase the number of nodes that adjust their sensing radius in the network and the energy consumption of the network. In addition, each management node in the algorithm ignores the residual energy of its neighbor nodes when it selects the assistant nodes. As a result, the sensing radius of some nodes increases frequently, thereby causing network energy consumption imbalances, degrading the quality of network service, and reducing network lifetime. Therefore, after the network achieves 1-coverage or approximately 1-coverage by N-nodes, an energy-efficient event K-coverage algorithm should be designed to achieve dynamic K-coverage for UWSNs, balance and reduce network energy consumption, prolong network lifetime, and increase network K-detecting performance.

This study proposes the DEEKA to solve this problem. The monitoring area achieves 1-coverage or approximately 1-coverage. First, at the start of each round of the algorithm, nodes that detect the same event compete for the management node of the event by the indicator, including the number of candidate nodes, the average residual energy of neighbor nodes, and the distance to the event. This process balances residual energy among neighbors with different management nodes and considers network detecting performance. Second, each management node calculates the probability of its neighbor nodes' being selected by the corresponding event it manages, according to levels of distance and residual energy of nodes in the set of the dynamic candidate node, as well as the number of dynamic coverage events of the node. This process is beneficial for each management node in that its neighbor nodes may dynamically cover several events when it selects assistant nodes. Moreover, it is beneficial for each management node to select synchronously and achieve distributed implementation of the algorithm. Third, each management node builds a multi-objective optimization model of the energy consumption expectation of its neighbor nodes, the residual energy variance of its neighbor nodes, and the detection performance of the events it manages as targets. Afterwards, the management node obtains Pareto solutions by using a constrained NSGA-II method. This process allows each management node to obtain all the best solutions (namely, node selection strategy) with three objectives constrained with one another. It is also beneficial for algorithm extension. Finally, m_m selects the best strategy using the TOPSIS method according to the target bias of practical application. Corresponding nodes adjust their sensing radius to achieve K-coverage for the event. A detailed description is presented below.

3.2. Algorithm Description

The DEEKA is divided into the following four steps: (1) management node formation; (2) calculation probability of an event-selected node; (3) a multi-objective optimization model; (4) a methodology of a constrained NSGA-II and optimization strategy.

3.2.1. Management Node Formation

At the beginning of each round of the algorithm, dynamical assistant nodes at the last round broadcast their residual energy. Each node updates the state information of its neighbor node, including residual energy and survival node number (the first round of the algorithm ignores this step). After the nodes detect the event (e.g., e_j), each node that detects e_j (e.g., s_i) calculates the distance between its neighbor nodes and e_j. Then, it compares the distance with the maximum sensing radius to produce the candidate node set of e_j within s_i ($s_i.C(e_j)$) and $s_i.|C(e_j)|$. S_i then broadcasts message I_c (node ID, ID of events it detects, and average residual energy of nodes in $s_i.C(e_j)$ and $s_i.|C(e_j)|$). After s_i receives I_c from other nodes (e.g., s_m), it saves the related information of other nodes who also detect e_j and then determines if it becomes the management node of e_j according to the cases as follows:

- If $s_i.|C(e_j)|$ is less than $\mathrm{K}-1$ and the candidate node number of the other nodes detecting e_j is also less than $\mathrm{K}-1$, s_i judges its $|C(e_j)|$ among nodes detecting e_j. If $s_i.|C(e_j)|$ is not the largest, it automatically gives up the competition of the management node; otherwise, it becomes the management node of e_j and joins the management node set M, as well as skips the rest of the process described in this section.
- If $s_i.|C(e_j)|$ is less than $\mathrm{K}-1$ and the candidate node number of the other nodes detecting e_j is not all less than $\mathrm{K}-1$, s_i gives up the competition of management node.
- If $s_i.|C(e_j)|$ is greater than $\mathrm{K}-1$, s_i then calculates its score $Score_1(s_i)$ and other nodes' score $Score_1(s_m)$ by Equation (15). It calculates according to the indicators of the average residual energy of and number of candidate nodes, and the distance to e_j. $Score_1(s_i)$ is formulated as follows:

$$Score_1(s_i) = w_1\frac{RE_{av}(s_i,e_j) - RE_{av}^{\min}}{RE_{av}^{\max} - RE_{av}^{\min}} + w_2\frac{d_{\max} - d(s_i,e_j)}{d_{\max} - d_{\min}} + w_3\frac{s_i.|C(e_j)| - |C(e_j)|_{\min}}{|C(e_j)|_{\max} - |C(e_j)|_{\min}} \quad (15)$$

  where $RE_{av}(s_i,e_j)$ is the average residual energy of s_i's candidate node for e_j, $RE_{av}^{\max}$ and $RE_{av}^{\min}$ are the maximum and minimum average residual energy of the candidate node of nodes detecting e_j, respectively; $d_{\max}$ and $d_{\min}$ are the maximum and minimum distances to e_j, respectively; $|C(e_j)|_{\max}$ and $|C(e_j)|_{\min}$ are the maximum and minimum numbers of the e_j candidate node, respectively.

  s_i then compares its score with others. If its score is not the greatest, then s_i gives up the competition for the management node of e_j. Otherwise, s_i becomes the management node of e_j and joins management node set M.
- If the information s_i receives does not include event e_j, which is to say, only s_i detects e_j, s_i becomes the management node of e_j and joins management node set M.

The flowchart of the management node of each round is shown in Figure 3, with event e_j as an example.

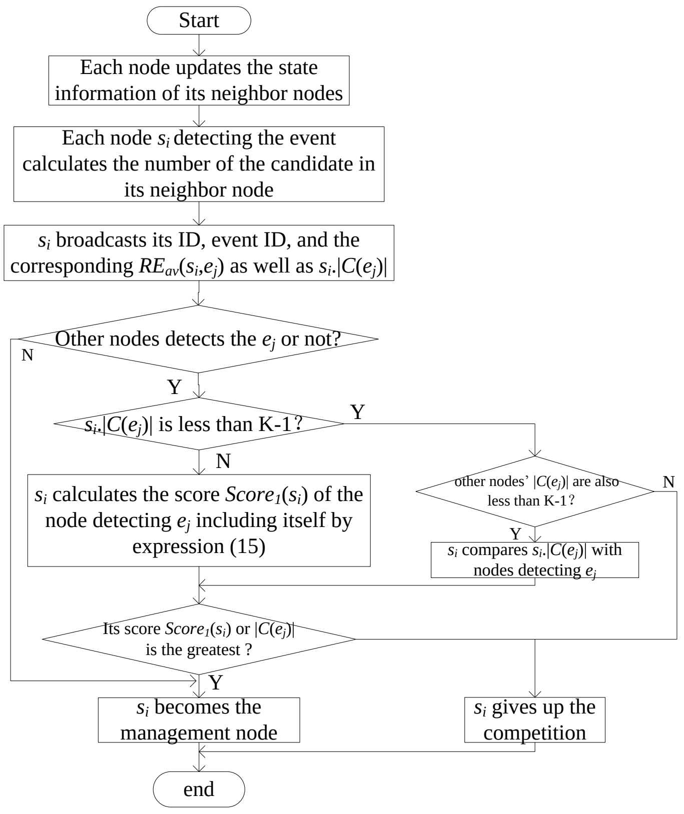
Start
Each node updates the state information of its neighbor nodes
Each node s_i detecting the event calculates the number of the candidate in its neighbor node
s_i broadcasts its ID, event ID, and the corresponding $RE_{av}(s_i,e_j)$ as well as $s_i.|C(e_j)|$
Other nodes detects the e_j or not?
N
Y
$s_i.|C(e_j)|$ is less than K-1?
Y
N
s_i calculates the score $Score_1(s_i)$ of the node detecting e_j including itself by expression (15)
other nodes' $|C(e_j)|$ are also less than K-1?
N
Y
s_i compares $s_i.|C(e_j)|$ with nodes detecting e_j
Its score $Score_1(s_i)$ or $|C(e_j)|$ is the greatest ?
Y
s_i becomes the management node
s_i gives up the competition
end

Figure 3. Flowchart of Section 3.2.1.

3.2.2. Calculation Probability of Event-Selected Node

After the management node is determined, each management node (e.g., m_m) broadcasts a helping message (node ID) with the communication radius R_c. Each node that receives the helping message sends its state message (node ID and number of dynamic coverage event $|E_d(s_i)|$) to the corresponding management node. After node m_m receives the state message, it calculates the distance between its neighbor nodes (e.g., s_i) and each event it manages (e.g., e_j). Then, it compares the distance with the minimum sensing radius $R_s^{\min}$. If $d(s_i,e_j) \le R_s^{\min}$, s_i joins into static candidate node set $C_s(e_j)$. Otherwise, s_i joins dynamic candidate node set $C_d(e_j)$. If $\left|C_s\left(e_j\right)\right|$ is greater than $K-1$, then the probability of the node in $C_d(e_j)$ being selected by e_j is 0. That is to say, to achieve K-coverage for e_j, none of the nodes need to adjust its sensing radius. Thus, m_m skips e_j, removes it out of $E_M(m_m)$, and calculates the probability of the node selected by other events it manages, namely, for e_j, it skips the rest of the process described in this section. Otherwise, m_m then calculates levels of residual energy $L_{RE}(c_d^i)$ and distance $L_d(c_d^i)$ of each dynamic candidate node (e.g., c_d^i) in $C_d(e_j)$. The process is shown as follows:

$$L_{RE}\left(c_d^i\right) = \frac{RE(c_d^i) - RE_{\min}}{RE_{\max} - RE_{\min}},\ L_d\left(c_d^i\right) = \frac{d_{\max} - d\left(c_d^i, e_j\right)}{d_{\max} - d_{\min}} \tag{16}$$

where $RE(c_d^i)$ is the residual energy of dynamic candidate node c_d^i; $RE_{\min}$ and $RE_{\max}$ are the minimum and maximum residual energy among nodes in $C_d(e_j)$, respectively; $d_{\min}$ and $d_{\max}$ are the minimum and maximum distances to e_j among nodes in $C_d(e_j)$, respectively.

According to the number of its dynamic coverage event $\left|E_d\left(c_d^i\right)\right|$, $L_{RE}(c_d^i)$, and $L_d(c_d^i)$, m_m then estimates the probability that c_d^i is selected to assist it to detect e_j, namely, e_j's assistant node. The number of c_d^i's actual dynamic coverage event $E_{ad}(c_d^i)$ is usually less than $\left|E_d\left(c_d^i\right)\right|$, and it is not a certain value before the algorithm run. Therefore, m_m considers a situation in which c_d^i is selected by e_j with different combinations of the actual coverage event number, which is produced by the dynamic coverage event number of nodes in $c_d(e_j)$, and m_m then estimates the probability of the node's being selected by e_j. In other words, m_m produces all the different combinations, and every combination consists of a number within the dynamic coverage event number of each node in $C_d(e_j)$. For example, if c_d^1, c_d^2, c_d^3 are the dynamic candidate nodes in $C_d(e_j)$, and $|E_d(c_d^1)|$, $|E_d(c_d^2)|$, and $|E_d(c_d^3)|$ are 1, 2, and 3, respectively, then all of the combinations of the actual dynamic coverage event are (0,0,0), (0,0,1), (0,0,2), (0,0,3), (0,1,0), (0,1,1), … , etc. Then, m_m determines c_d^i is selected or not according to Equation (17) for each combination and finally regards the frequency of c_d^i's being selected as the probability of c_d^i's being selected by e_j.

The total number of combination increases in factorial form with $\left|E_d\left(c_d^i\right)\right|$ and $|C_d(e_j)|$. Thus, this study estimates the probability of c_d^i's being selected by e_j by sampling the part among the total combinations. In order to improve the accuracy of the probability, repeat the above process one round and the average of the result of total rounds is regarded as the probability of c_d^i's being selected by e_j. According to Reference [37], when the sample size is greater than 10,000,000, the sample number is 400 with a 0.05 margin of error, and the confidence level is 95%. Thus, in this study, the sample number is 400 regardless of the sample size.

(1) For each node c_d^i, m_m randomly selects an integer in the interval [0, $|E_d(c_d^i)| - 1$] as the number of c_d^i's actual dynamic coverage events, and these integers that m_m selects form a combination of the number of actual coverage events, namely, one sampling.

(2) m_m calculates the score $Score_2(c_d^i)$ of each node c_d^i in the current combination by Equation (17), according to $L_{RE}(c_d^i)$, $L_d(c_d^i)$, and $\left|E_{ad}\left(c_d^i\right)\right|$:

$$Score_2\left(c_d^i\right) = w_1 L_{RE}\left(c_d^i\right) + w_2 L_d\left(c_d^i\right) + w_3 \frac{\left|E_{ad}\left(c_d^i\right)\right| - E_{ad}^{\min}}{E_{ad}^{\max} - E_{ad}^{\min}} \quad (17)$$

where w_1, w_2, and w_3 are indicator weights whose values are set according to the application focusing on the relevant indicators and $w_1 + w_2 + w_3 = 1$; $E_{ad}^{\min}$ and $E_{ad}^{\max}$ are the minimum and maximum numbers of the actual coverage events in the current combination, respectively.

(3) m_m selects $K - 1 - \left|C_s\left(e_j\right)\right|$ nodes as the dynamic assistant nodes of e_j, according to the scores in descending order (nodes in $C_s\left(e_j\right)$ must be the assistant nodes of e_j because they do not need to increase their sensing radius to cover e_j). Other nodes are not selected in the current combination.

(4) If the sample number is less than 400, then Step 1 is repeated. Otherwise, the next step is performed.

(5) For each node c_d^i, m_m records and calculates the frequency of c_d^i's being selected to cover e_j $Frequency(c_d^i, e_j)$ in the current round of sampling.

Steps 1–5 are repeated a times, and the average value of $Frequency(c_d^i, e_j)$ is calculated as the probability of c_d^i's being selected to cover e_j ($P_{select}(c_d^i, e_j)$). Then, each management node $\mathrm{m_m}$ sends probability $P_{select}(c_d^i, e_j)$ to the corresponding node c_d^i. c_d^i saves and broadcasts this message (its dynamic assistant event, the corresponding $P_{select}(c_d^i, e_j)$, and the number of its static assistant event $|E_s(s_i)|$). Each management node m_m receives the message and saves the information.

The flowchart for calculating the probability of node selected by event is shown in Figure 4, with management m_m as an example.

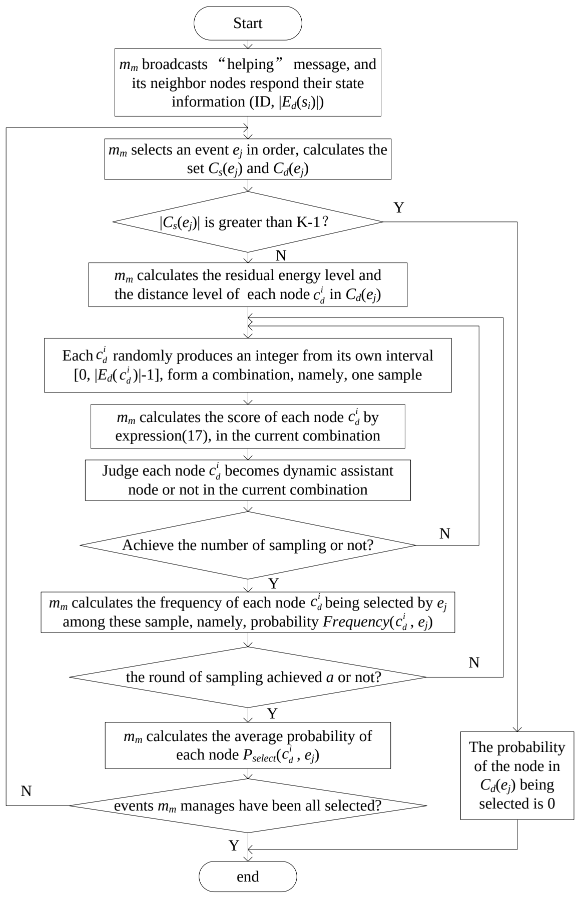
Start
m_m broadcasts "helping" message, and its neighbor nodes respond their state information (ID, $|E_d(s_i)|$)
m_m selects an event e_j in order, calculates the set $C_s(e_j)$ and $C_d(e_j)$
$|C_s(e_j)|$ is greater than K-1?
Y
N
m_m calculates the residual energy level and the distance level of each node c_d^i in $C_d(e_j)$
Each c_d^i randomly produces an integer from its own interval $[0, |E_d(c_d^i)|-1]$, form a combination, namely, one sample
m_m calculates the score of each node c_d^i by expression(17), in the current combination
Judge each node c_d^i becomes dynamic assistant node or not in the current combination
Achieve the number of sampling or not?
N
Y
m_m calculates the frequency of each node c_d^i being selected by e_j among these sample, namely, probability $Frequency(c_d^i, e_j)$
the round of sampling achieved a or not?
N
Y
m_m calculates the average probability of each node $P_{select}(c_d^i, e_j)$
The probability of the node in $C_d(e_j)$ being selected is 0
N
events m_m manages have been all selected?
Y
end

Figure 4. Flowchart of node selection.

3.2.3. Multi-Objective Optimization Model

After the process of Section 3.2.3, each management node knows the probability of its neighbor nodes' being selected by other corresponding events. Each management node (e.g., m_m) then calculates the expectation of the number of other events that its neighbor node (e.g., s_i) may cover $\overline{E}_{ad}(m_m, s_i)$:

$$\overline{E}_{ad}(m_m, s_i) = \sum_{e_j \in E_d(s_i) - E_M(m_m)} P_{select}(s_i, e_j). \tag{18}$$

In Equation (18), $P_{select}(s_i,e_j)$ is the probability of s_i's being selected by e_j; in other words, $P_{select}(s_i,e_j)$ is the probability that s_i assists the corresponding management node in covering e_j. Thus, the sum of the probability of events that s_i can cover, except what m_m manages, is the average number of events that s_i actually covers, namely, the expectation of the event s_i actually covers.

Meanwhile, m_m calculates the corresponding expectation of s_i's sensing radius $\overline{R}_s(m_m, s_i)$. This problem is equivalent to the question of calculating the expectation of s_i's sensing radius, with all known possible sensing radii of the node s_i and corresponding probabilities.

After m_m learns the number of events that its corresponding neighbor node covers and that the other management nodes manage, it builds optimization models according to the information, the residual energy of the node, and the distance to the corresponding event. For the event m_m manages and whose static assistant node number is greater than $\mathrm{K} - 1$, m_m need not consider it for the adjusting strategy because the events have been K-coverage without any nodes adjusting the sensing radius. Thus, the event is removed out of $E_M(m_m)$ before the optimization models are built.

Assuming that x_{ij} denotes whether s_i is selected to cover e_j or not, and that b_{ij} expresses whether e_j is the dynamic coverage event of s_i or not (for m_m, if $d(s_i,e_j)$ is known, then b_{ij} is also known),

$$x_{ij} = \begin{cases} 0 & s_i \text{ does not cover } e_j \\ 1 & s_i \text{ covers } e_j \end{cases}, \; b_{ij} = \begin{cases} 1 & R_s^{\min} < d(s_i, e_j) \le R_s^{\max} \\ 0 & else \end{cases}.$$

If $x_{ij}b_{ij}$ equals 1, then s_i is selected to adjust its sensing radius to cover e_j, namely, the dynamic assistant node of e_j, which is also the solution of the multi-objective optimization model.

Therefore, the energy consumption, the residual energy variance, and event detection performance are described after the strategy of the node adjusting sensing radius.

Energy Consumption

At first, the process whereby each management node (e.g., m_m) calculates the energy consumption of its neighbor nodes (e.g., s_i) with three cases is analyzed as follows:

(1) s_i dynamically covers one event or more, namely, s_i dynamically covers the event m_m manages, or its $\overline{E}_{ad}(m_m, s_i)$ is not equal to 0; then, m_m calculates the energy consumption of s_i produced by the cover of these events that m_m manages. For example, for the node s_2 in Figure 5, its detecting energy consumption is $ECS(R_2)$ in the network, but m_1 calculates its energy consumption produced by covering e_1; in other words, the $ECS(R_2)$ is divided into two parts by e_1 and e_4. Because m_1 does not know whether s_2 covers e_4 or not, in this study, the number of events that other nodes manage and that s_2 covers is estimated by the expectation of the actual coverage event number $\overline{E}_{ad}(m_1, s_2)$. Therefore, s_2's energy consumption that m_1 calculates is $ECS(R_2)/\left(1 + \overline{E}_{ad}(m_1, s_2)\right)$.

(2) s_i only covers some static events, while m_m calculates the energy consumption of s_i produced by the cover of these events that m_m manages. In Figure 5, the energy consumption of s_1 is $ECS(R_s^{\min})$ and is divided into two parts by e_2 and e_5. Because the static cover event number of s_1 is definite, the s_1'energy consumption m_1 calculates is $ECS\left(R_s^{\min}\right)/(1+1)$.

(3) s_i does not cover any event, and the total energy consumption of s_i is calculated. In Figure 5, s_3's energy consumption m_1 calculates $ECS(R_s^{\min})$.

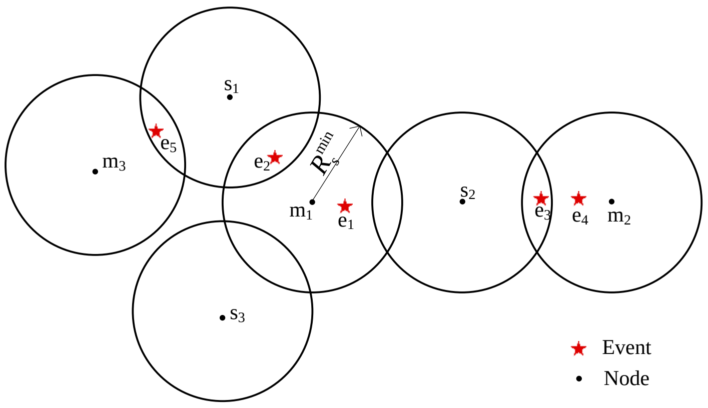

Figure 5. Schematic diagram.

(Assuming that m_1 manages events e_1 and e_2, m_2 manages events e_3 and e_4, and m_3 manages e_5, s_2 dynamically covers events e_1 and e_4, s_1 only covers events e_5 and e_2, s_3 does not cover any events, the sensing radius of s_1 and s_3 are both $R_s^{\min}$, and the sensing radius of s_3 is R_2).

The sensing energy consumption of its neighbor nodes is the sum of the energy consumption of each neighbor node and formulated as follows:

$$Energy(m_m) = \sum_{s_i \in S_N(m_m)} r_{energy}(s_i) \cdot ECS(R_i) \tag{19}$$

where R_i is the current sensing radius of s_i, whose value is the maximum among $\overline{R}_s(m_m, s_i)$ and is the distance to the event it covers in $E_M(m_m)$. When both of them are zero, R_i is the minimum sensing radius. The expression for R_i as follows:

$$R_i = \max\left(R_s^{\min}, x_{ij}b_{ij}d(s_i, e_j), \overline{R}_s(s_i)\right) \quad j = 1, 2, \ldots |E_M(m_m)|. \tag{20}$$

r_{energy} (s_i) is the proportionality coefficient, and its expression under the three case above is formulated as follows:

$$r_{energy}(s_i) = \begin{cases} \sum_{j=1}^{|E_M(m_m)|} x_{ij}b_{ij} / \left(\overline{E}_{ad}(m_m, s_i) + \sum_{j=1}^{|E_M(m_m)|} x_{ij}b_{ij}\right) & if\ \overline{E}_{ad}(m_m, s_i) + \sum_{j=1}^{|E_M(m_m)|} x_{ij}b_{ij} \neq 0 \\ k_2 / |E_s(s_i)| & if\ \overline{E}_{ad}(m_m, s_i) + \sum_{j=1}^{|E_M(m_m)|} x_{ij}b_{ij} = 0, \text{ and } |E_s(s_i)| \neq 0 \\ 1 & if\ \overline{E}_{ad}(m_m, s_i) + \sum_{j=1}^{|E_M(m_m)|} x_{ij}b_{ij} = 0, \text{ and } |E_s(s_i)| \neq 0 \end{cases} \tag{21}$$

where k_2 is the number of events that m_m manages; meanwhile, s_i can also cover statically. In the expression, the first item corresponds to Case (1), where the denominator expresses the number of events s_i dynamically covers in the whole network and the numerator expresses the number of events s_i dynamically covers in the event set $E_M(m_m)$. Similarly, the second item corresponds to Case (2), and the last one corresponds to Case (3).

Thus, the total sensing energy consumption of the network approximates to the sum of the energy consumption that each management node calculates, denoted by Equation (19).

Residual Energy Variance

The residual energy variance of neighbor nodes of m_m is *Variance*(m_m) after the adjusting strategy runs one round.

$$Variance(m_m) = \sum_{s_i \in S_N(m_m)} \left(RE_{after}(s_i) - \overline{RE}_{after}\right)^2 \tag{22}$$

where $\overline{RE}_{after}$ is the average residual energy of m_m's neighbor nodes after m_m uses the adjusting strategy, and $RE_{after}(s_i)$ is the residual energy of s_i, which can describe the residual energy before using the adjusting strategy $RE(s_i)$minus for the energy consumption of sensing and communication in the current round. The formulation as follows:

$$RE_{after}(s_i) = RE(s_i) - ECS(R_i) - bool \cdot ECC(R_c) \tag{23}$$

where *bool* is a Boolean parameter, which determines if s_i should transmit information to *mm*. Its formulation as follows:

$$bool = \begin{cases} 1 & x_{ij}b_{ij} = 1\, or\, d(s_i, e_j) < R_s^{\min} \\ 0 & else \end{cases}.$$

This equation indicates that, when the node is selected as the dynamic assistant node of e_j or the node statically cover e_j, the node should detect e_j and transmit information about e_j to mm.

Event Detecting Performance

The detecting performance of all events in $E_M(m_m)$ (*Performance*(m_m)) is equal to the sum of that individual event:

$$Performance(m_m) = \sum_{e_j \in E_M(m_m)} P_{result}(e_j). \tag{24}$$

In the process of calculating the detecting performance, the assistant node set of each event e_j ($A(e_j)$) is formulated as follows:

$$A(e_j) = \{a | a = x_{ij}b_{ij}s_i, i = 1, 2, \ldots |S_N(m_m)|\} \cup C_s(e_j) - \{0\}. \tag{25}$$

The element in $A(e_j)$ is the ID of the assistant node of e_j, consisting of the dynamic assistant node (the first item in the equation) and the static assistant node (the second item in the equation). Because the first item includes the element 0, which does not belong to the node ID, the set with the element 0 is removed (the last item in the equation).

The multi-objective optimization model that each management node m_m builds is discussed as follows:

Objective 1: The energy consumption of m_m neighbor nodes is minimized.

$$\min \ Energy(m_m).$$

Objective 2: The residual energy variance of m_m neighbor nodes is minimized.

$$\min \ Variance(m_m).$$

Objective 3: The detecting performance of the event $\mathrm{m_m}$ that manages is maximized.

$$\max \ Performance(m_m).$$

Some constraint conditions also exist, which are as follows:

Constraint condition 1: Each event must cover at least $\mathrm{K}-1$ assistant nodes.

$$\left|C_s(e_j)\right| + \sum_{i=1}^{|S_N(m_m)|} x_{ij} b_{ij} \geq K-1 \ \ j = 1,2,\ldots |E_M(m_m)|.$$

Constraint condition 2:

$$x_{ij} = 0, 1.$$

3.2.4. Methodology of the Constrained NSGA-II and Optimization Strategy

NSGA-II [38] is the improved version of the NSGA. It adopts a fast non-dominated sorting procedure, a crowded comparison operator, and a controlled elitism mechanism, and overcomes certain disadvantages in the NSGA, such as high computing complexity, premature convergence, and the requirement of an assigning sharing parameter. The progress of solving the multi-objective optimization model in Section 3.2.3 is shown as follows (if the space of the multi-objective optimization model is less than the number of population n_p, then the solution set is solved by an exhaustive method directly and ignores the following steps):

(1) Encoding the solution

To select the best node set in the neighbor nodes of each management node m_m, the solution can be expressed as bit string b_s (b_s [1] is the first element in b_s, and b_s [2] is the second element in b_s). Specific coding is shown in Figure 6, and the bit number of b_s is $|S_N(m_m)| \cdot |E_M(m_m)|$.

Then,

$$x_{ij} = b_s[(i-1) \cdot |E_M(m_m)| + j]. \tag{26}$$

(2) Fast Non-Dominated Sorting Procedure

After the value of each solution in the current generation is obtained, the rank of each solution is distributed by a fast, non-dominated sorting procedure. The rank of some non-feasible solutions may be greater than that of feasible solutions. Thus, the case may degrade the convergence speed of the algorithm. Therefore, during the rank distribution for each solution, this study adds the following two rules [39]:

1. Feasible solutions are greater than non-feasible solutions.
2. In non-feasible solutions, the rank of solution with a lower degree of restriction is greater.

(3) Selection, Evolution, and Recombination [39]

Selection involves choosing excellent individuals from one generation to implement evolution operation. This study uses the championship method to select parent individuals. Evolution includes crossover and mutation, and this study uses two-point crossover. Recombination involves eliminating inferior individuals to form new species groups of the same scale, according to a situation of parent-and-son generations. This study uses the recombination method in Reference [40].

The population in each generation produces a population of the next generation by using Steps (2) and (3). The method ends until the generation value achieves the maximum iteration number.

After NSGA-II obtains the Pareto solution, this study uses the TOPSIS method [41] to obtain the best strategy according to the energy consumption of detection, the residual energy variance, and the event detecting performance.

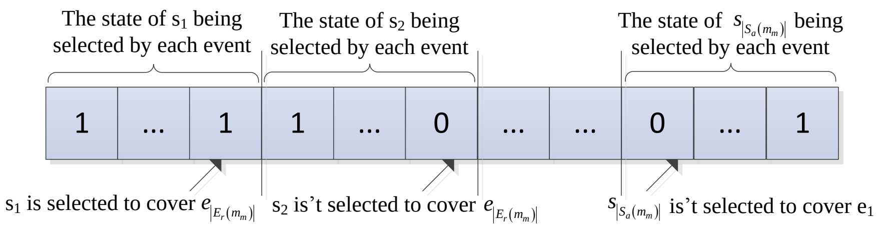
The state of s_1 being selected by each event
The state of s_2 being selected by each event
The state of $s_{|S_a(m_m)|}$ being selected by each event
1
...
1
1
...
0
...
...
0
...
1
s_1 is selected to cover $e_{|E_r(m_m)|}$
s_2 is't selected to cover $e_{|E_r(m_m)|}$
$s_{|S_a(m_m)|}$ is't selected to cover e_1

Figure 6. Solution encoding figure.

4. Algorithm Analysis

The message and time complexity of the DEEKA are evaluated in this section.

4.1. Message Complexity

The total number of sent and received messages determines message complexity. At the beginning of every round of the algorithm, each dynamic assistant node at the last round broadcasts its residual energy, and the nodes that receive the message save and update the state information of its neighbor nodes. The average number of dynamic assistant nodes in each round is assumed as n_d ($n_d \leq N$), and the average number of neighbor nodes is assumed as n_a. In this stage, the number of sent and received messages are n_d and $n_d \cdot n_a$, respectively. In the stage of management node formulation, the node broadcasts message M_c after it detects an event. The number of sent messages is n_c (n_c is the number of nodes detecting the event; $n_c \leq N$), and the number of received messages is $n_c \cdot n_a$. Each management node broadcasts a "help" message after its formulation. Other nodes receiving this message broadcast their state message. In the two processes, the number of sent messages is $n_m + n_r$ (n_m is the number of management nodes in every round, $n_m \leq N$; and n_r is the number of nodes receiving the "help" message, $n_r \leq N$), and the number of received messages is $n_a \cdot (n_m + n_r)$. Then, each management node sends the message about the probability of a node's being selected to the corresponding node after the probability has been calculated. Then, each node broadcasts its probability of being selected by the event. In the two processes, the number of sent and received messages is $n_m + n_r$ and $n_a \cdot (n_m + n_r)$, respectively. In the last stage, each management node broadcasts to direct the corresponding nodes to adjust their sensing radius when it has calculated the best strategy. In this process, the number of sent messages is n_m, and the number of received message is $n_a \cdot n_m$.

Therefore, the total number of sent messages for every round is expressed as follows:

$$n_d + n_c + 2(n_m + n_r) + n_m \leq 5N. \tag{27}$$

The complexity of sent messages is $O(N)$.

The total number of received messages for every round is expressed as follows:

$$n_d \cdot n_a + n_a \cdot n_c + 2n_a \cdot (n_m + n_r) + n_a \cdot n_m \leq 5N^2. \tag{28}$$

The complexity of received messages is $O(N^2)$.

4.2. Time Complexity

In management node formulation, all nodes that detect the same event compete to formulate the final management node of the event. The worst aspect of the process is that all of the nodes detect all of the events. In this situation, every node for each event calculates the score of all of the nodes about the event. Thus, the time complexity of the process is $Z{\cdot}N$. In the process of calculating the probability of the node's being selected by events, each management node synchronously calculates the probability of its neighbor nodes' being selected by each event by a sampling and statistics method. According to Reference [37], when the number of combinations is greater than 10,000,000, the number of samplings does not exceed 400. Thus, this study sets the sample number regardless of the total number of combinations. In addition, for a more accurate and reliable result, this study samples at round a and uses the average as the probability of the node's being selected. The worst aspect of the process is that all the events are managed by a management node, and other nodes are the neighbor nodes of the management node. In this situation, time complexity does not exceed $MN(400a\text{K} + 1)$. In the process of solving the multi-objective optimization model, the time complexity of NSGA-II is $\text{O}\left(3n_p^2\right)$, where n_p is the population number. Finally, each management node selects the best strategy from the Pareto solution set by the TOPSIS method. The TOPSIS method in this study does not need to sort for all the results and selects only the best one. Assuming that the number of Pareto solution sets is n_s, the time complexity of the process is n_s.

The time complexity of the whole process of algorithm T_c is shown as follows:

$$T_c = ZN + ZN(400aK + 1) + 3n_p^2 + n_s \leq (400aK + 3)ZN + 3n_p^2. \tag{29}$$

Because n_p is the constant relative to Z and N, the time complexity can be expressed as $O(ZN)$.

5. Simulation and Performance Analysis

In this study, the analysis of the DEEKA will be divided into two parts. At first, the validity of the DEEKA is verified for the four indicators, namely, event detecting performance, network energy consumption, network reliability, and energy consumption balance, compared to an algorithm of the same class, the OVSKA. Then, the DEEKA is compared to an algorithm of another class (DMNSA [26]) in terms of network energy consumption, event detecting performance, and the event detecting delay to verify the correctness of the analysis.

5.1. Simulation Scenario and Parameter Settings

A UWSN's event K-coverage process is simulated by maxtrix laboratory (MATLAB) based on the background of Zhejiang offshore breeding sites, in which the speed of the water current is relaxed, underwater terrain is relatively flat and has little impact on the capability of communication and sensing of the node, and there is enough deep to deploy the nodes. Therefore, during the simulation, the target water area (length $\times$ width $\times$ depth) is set to 100 m $\times$ 100 m $\times$ 100 m.

After the target water area is determined, the network achieves 1-coverage by using deterministic deployment of the truncated octahedron (the method needs the minimum number of nodes), in which the number of nodes is 560. The network then is simulated according to the description of Section 2.1.1. Because the paper focuses on the application effectiveness of the algorithm on event K-coverage, the paper assumes an ideal physical layer, a MAC layer, and error-free communication links for simplicity, namely, nodes can communicate with each other when they are in the communication range and all of the data bags each node sends or receives are right, but may be lost in a certain probability in the communication process, and the probability p_{lost} is set to 0.2. For the energy consumption during the network running, the paper use of the network energy model mentioned in Section 2.1.2 calculates the energy consumption of each node. Meanwhile, the paper uses the detection performance model to calculate the performance of the network by the algorithm running.

For the value of each indicator, it is produced by an average of 30 rounds of simulation data. The parameters are set in Table 1 as follows:

Table 1. Simulation parameter table.

Parameter	Value	Parameter	Value
Information transmission failure p_{lost}	0.2	Minimum sensing radius $R_s^{\min}$	10 m
Energy consumption of data reception P_r	5 mW	Maximum sensing radius $R_s^{\max}$	28 m
Data transmission speed underwater	1000 bit/s	Communication radius R_c	20 m
Interval of algorithm operation T	6 s	Length of data packet l	150 bit
Adjusting parameters γ_1, γ_2	0.23, 0.71	Energy diffusion factor λ	1.5
k_1, a	1, 10	Carrier frequency f	24 kHZ
Ratio of sensing to communication power r_p	43/80	Maximum iteration number	300
Values of weight w_1,w_2,w_3	1/3, 1/3, 1/3	Number of population n_p	60

5.2. Simulation Example

5.2.1. Comparison with the OVSKA

Figure 7 presents the average energy consumption of each node of both the OVSKA and the DEEKA in every round with the varying number of events under different coverage requirements. In Figure 7, under the same coverage requirements, the energy consumption of both the OVSKA and the DEEKA increases. However, the trend gradually slows down with the increasing number of events. When the number of events is also the same, the energy consumption of the DEEKA is lower than that of the OVSKA. In addition, under different coverage requirements, the energy consumption of the two algorithms also increases. In the DEEKA, each management node selects the assistant nodes considering the expectation number of actual dynamic coverage events of the node. When selecting the assistant node for one event, a situation in which the node is selected by other dynamic coverage events is considered. The process will reduce the number of dynamic assistant nodes and the energy consumption to a certain extent. It will also slow down the increasing speed of the number of dynamic assistant nodes with the increasing number of events or the increasing value of K. On the contrary, the OVSKA does not consider the above situation, which leads to the number of events or the value of K having a relatively great influence on energy consumption.

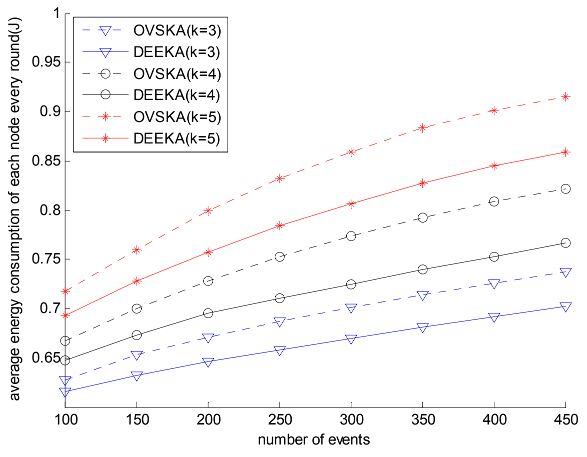

Figure 7. Average energy consumption of each node with event number and K-value.

Figure 8 presents the network lifetime of each node of both the OVSKA and the DEEKA with the varying number of events under different coverage requirements. As shown in Figure 8, the network lifetime of both algorithms decreases with the increasing number of events. With the same coverage requirements, the decreasing speed of the DEEKA is less than that of the OVSKA, and when the number of events is the same, the network lifetime of the DEEKA is greater than that of the OVSKA. In addition, the decreasing trend of the network lifetime of the DEEKA is less than that of the OVSKA. In the OVSKA, each management node ignores the residual energy of the node and situations in which a node is selected by several dynamic coverage events when it selects the assistant nodes. The process produces several dynamic assistant nodes in the network and frequently makes some nodes the dynamic assistant nodes, which is not beneficial to prolonging network lifetime. However, the DEEKA considers these situations. In the DEEKA, a node can cover several dynamic coverage events to a great extent, which can slow down the increasing speed of network energy consumption. In addition, both the management node formulations of each event and the selection of the dynamic assistant node in the DEEKA considers the residual energy of the node, which can balance network energy consumption and is beneficial to prolonging network lifetime.

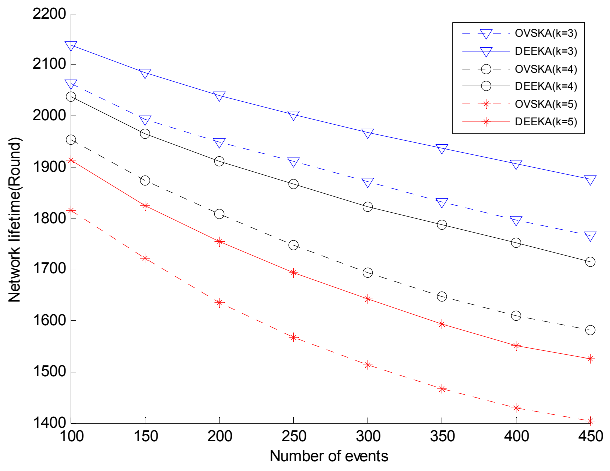

Figure 8. Network lifetime with event number and K-value.

Figures 7 and 8 shows the trend of each indicator with the varying number of events and the varying value of K. The trend of each indicator from the network starting to run to the network ending will be shown in this section. The number of events is set 250, and the value of K is set 4.

Figure 9 presents the number of survival nodes with the network running round. As shown in Figure 8, both the running time where none of the nodes died and the network lifetime are greater in the DEEKA. Moreover, the number of survival nodes reduces slowly at first, and the trend then increases gradually when the first dying node appears. In addition, the number of survival nodes of the DEEKA under the same round is greater than that of the OVSKA. In the OVSKA, each management node ignores the residual energy of the node when it selects the assistant nodes, which makes some nodes become the dynamic assistant nodes frequently and then accelerates them to die. However, in the DEEKA, each management node regards the residual energy variance of its neighbor nodes as one of the optimization objectives to select its assistant nodes. Specifically, it considers the residual energy of its neighbor node during the process, which can balance network energy consumption to slow down node death and prolong network lifetime. In addition, in the DEEKA, each management node calculates the energy consumption of its neighbor node, considering the expected actual dynamic coverage event of the node, which reduces the energy consumption of the network. Thus, the process is also beneficial to slowing down node death.

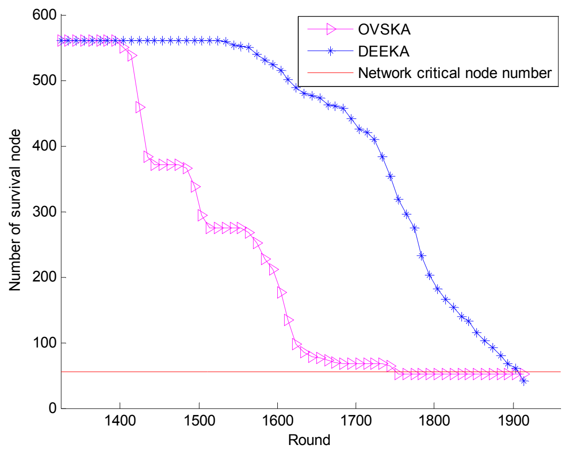

Figure 9. Number of survival nodes with the number of network running rounds.

Figure 10 presents the network detecting performance of both the OVSKA and the DEEKA with the network running round. When nodes did not die in the network, the network detecting performance of the OVSKA was better than that of the DEEKA with a slight difference. However, when the dying nodes appear, the network detecting performance of the OVSKA decreases quickly, and that of the DEEKA decreases slowly and maintains a relatively good performance in a certain round. In the DEEKA, each management node selects the assistant node considering the balance of the energy consumption of the node, which can reduce the speed of node death and maintain a relatively high number of survival nodes in certain rounds, as shown in Figure 9.

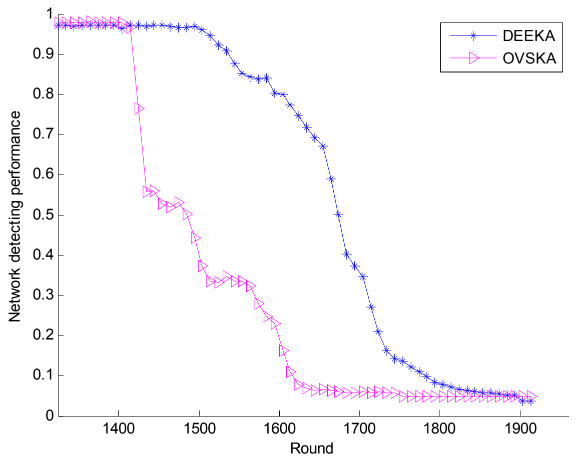

Figure 10. Network detecting performance with the number of network running rounds.

Figure 11 shows the residual energy variance of both the OVSKA and the DEEKA with the network running round. As shown in Figure 10, the variance of both algorithms increases, but the increasing trend of the OVSKA is greater than that of the DEEKA. Additionally, they decrease quickly with the decrease in the number of survival nodes. This phenomenon benefits from each management node with regard to residual energy variance of its neighbor nodes as one of the optimization objectives in the DEEKA, that is, considering the residual energy of its neighbor when it selects the assistant nodes and making network energy consumption more balanced.

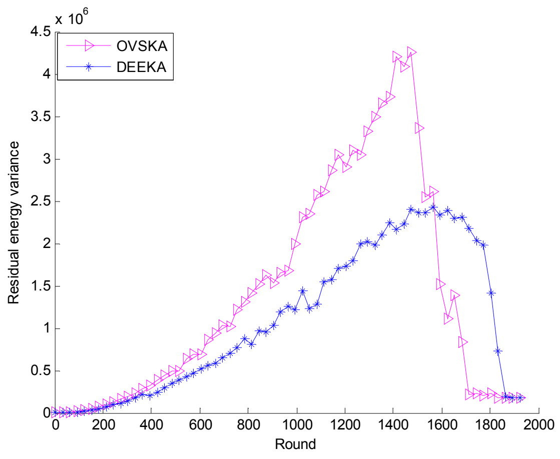

Figure 11. Residual energy variance with the number of network running rounds.

Figure 12 shows the average energy consumption of each node of the OVSKA and the DEEKA with the network running round. In Figure 12, the network energy consumption of the two algorithms in the network without node death moves down and up among certain values, but the energy consumption of the DEEKA is less than that of the OVSKA. Because in the DEEKA, each management node selects the assistant node considering a situation in which the node may be selected by several dynamic coverage events, which can reduce the number of dynamic assistant nodes to a certain extent and reduce network energy consumption. When the death node appears in the network, the average energy consumption of both algorithms increases at first and then decreases, but the increasing trend of the DEEKA is less than that of the OVSKA because the number of survival node decreases as round number increases, which makes the average sensing radius of the node increase in turn. In other words, network energy consumption increases. When the number of survival nodes decreases to a certain value, dynamic assistant nodes do not exist for some events. Thus, energy consumption decreases. The DEEKA considers the balance of network energy consumption, thereby slowing down the death rate of the node and maintaining the number of survival nodes at a high level for a period of time, thereby slowing down the increasing rate of the average sensing radius of the node, i.e., slowing down the increasing speed of the network energy consumption.

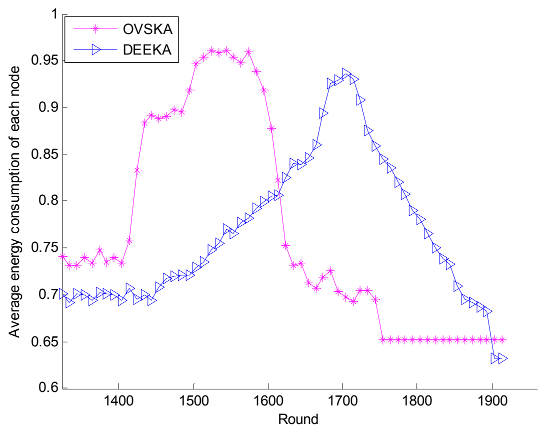

Figure 12. Average energy consumption of each node with the number of network running rounds.

5.2.2. Comparison with the DMNSA

The DEEKA is compared to another class of algorithm in this section. For the kind of algorithm used for static K-coverage for the area, the K-coverage for the area, regardless of whether a sleep strategy of the node is used, should be completed. Clearly, the number of nodes that the K-coverage network needs is K times as much as that 1-coverage needs. Thus, this study does not compare the DEEKA to this kind of algorithm because the result is obvious. For the kind of algorithm that assisted mobile nodes, we chose instead the distributed mobile node selection algorithm (DMNSA) proposed by reference [26] for event K-coverage using mobile nodes. The position of the mobile node is randomly distributed in the area, and the number of mobile nodes is equal to the number of nodes in the network for K-coverage. Meanwhile, the energy consumption of moving unit distance is 1.2 J [26].

Figure 13 presents the energy consumption of the DEEKA and the DMNSA, and the average moving distance of the DMNSA with the varying value of K. In Figure 13, energy consumption increases with increasing coverage level, but the trend of the DEEKA is less than that of the DMNSA. Meanwhile, the energy consumption of the DEEKA is less than the DMNSA with the same coverage level. The reason is that the DEEKA completes the event K-coverage by adjusting the sensing radius of the node, and the sensing energy consumption is much lower than the energy consumption of the moving node. In addition, the average moving distance of the node decreases as coverage level increases. However, the node still needs some time to move to the destination because of the low moving speed of the node in water [29]. Contrarily, the method of adjusting the sensing radius that the DEEKA uses only needs to adjust the transmit power of the sensor, which is completed instantaneously.

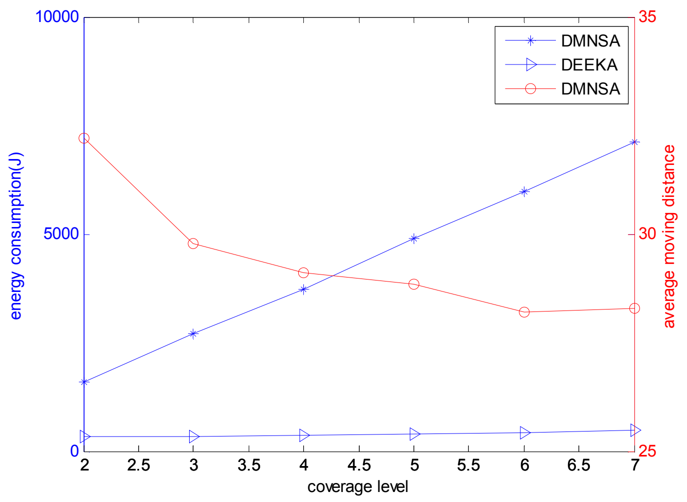

Figure 13. Energy consumption and average moving distance with varying K.

Figure 14 shows that the network detecting performance of the DEEKA and the DMNSA with the varying coverage level. In the figure, the network detecting performance of the DEEKA is less than that of the DMNSA, but the difference closes with increasing coverage level and the greatest difference between them is also no more than 5% because the DMNSA uses mobile nodes to complete K-coverage and the event is all within the minimum sensing radius of the node. However, the DEEKA adjusts the sensing radius to complete the event K-coverage, and, for each event, there are some nodes among the K nodes that need to increase their sensing radius to cover the event, which degrades the detecting quality of the event. When the coverage level increases, its impact on the network detecting performance, caused by the increasing the sensing radius, weakens gradually. That is to say, the robustness of the network increases.

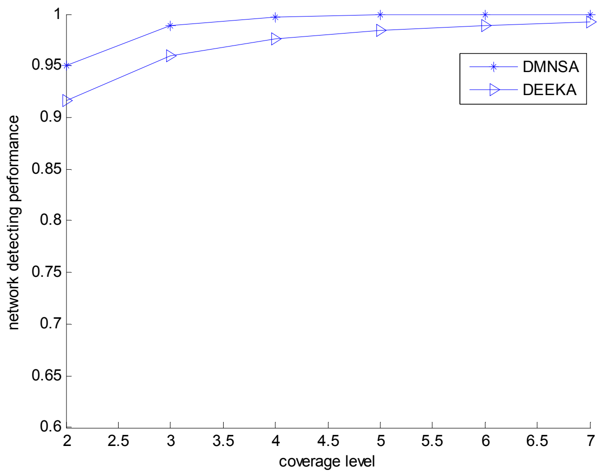

Figure 14. Network detecting performance with varying K.

6. Conclusions

This study presents the DEEKA and considers the effect of harsh underwater environments on information collection and transmission. It also considers the residual energy of the node and a situation in which the node is selected by several other events. At the beginning of each round of the algorithm, the nodes that detect the same event compete for the management node of the event with the number of neighbors and the average residual energy, as well as distance to the event. Then, each management node calculates the probability of each dynamic candidate node's being selected by the corresponding event it manages according to the levels of distance and the residual energy of nodes in the set of dynamic candidate nodes, as well as the number of dynamic coverage events of the node. Afterwards, each management node builds a multi-objective optimization model of the expected energy consumption of its neighbor nodes, the residual energy variance of its neighbor nodes, and the detecting performance of the events it manages as targets. Then, the management node obtains Pareto solutions by using the constrained NSGA-II method. Finally, m_m selects the best strategy using the TOPSIS method, according to target bias of practical application. Simulation results show that, unlike the OVSKA, the DEEKA balances and reduces network energy consumption, thereby prolonging the network's best service quality and lifetime.

In future work, we plan to find a more precise method to establish the multi-objective programming model combining the characteristics of UWSNs. In addition, we will try to find a special or simple method to solve the model to further optimize our algorithm.

Acknowledgments: This paper was supported by The National Key Research and Development Program of China (2016YFC0201400), the National Natural Science Foundation of China (NSFC61273072), the Provincial Key Research and Development Program of Science and Technology Department of Zhejiang Province (2016C03G2040707) and the National Natural Science Foundation of China and Zhejiang Joint Fund for Integrating of Informatization and Industrialization (U1509217).

Author Contributions: Peng Jiang and Yiming Xu conceived and designed the research; Peng Jiang and Yiming Xu performed the research; Peng Jiang and Yiming Xu wrote the paper.

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

References

1. Guo, Z.W.; Luo, H.J.; Hong, F.; Yang, M.; Lionel, M.N. Current Progress and Research Issues in Underwater Sensor Networks. *J. Comput. Res. Dev.* **2010**, *47*, 377–389.
2. Heidemann, J.; Stojanovic, M.; Zorzi, M. Underwater sensor networks: Applications, advances and challenges. *Philos. Trans. R. Soc. Lond. A Math. Phys. Eng. Sci.* **2012**, *370*, 158–175. [CrossRef] [PubMed]
3. Lloret, J. Underwater sensor nodes and networks. *Sensors* **2013**, *13*, 11782–11796. [CrossRef] [PubMed]

4. Yick, J.; Mukherjee, B.; Ghosal, D. Wireless sensor network survey. *Comput. Netw.* **2008**, *52*, 2292–2330. [CrossRef]
5. Han, G.; Zhang, C.; Shu, L.; Sun, N.; Li, Q. A survey on deployment algorithms in underwater acoustic sensor networks. *Int. J. Distrib. Sens. Netw.* **2013**, *1*, 1–11. [CrossRef]
6. Sendra, S.; Lloret, J.; García, M. Power saving and energy optimization techniques for wireless sensor neworks. *J. Commun.* **2011**, *6*, 439–459. [CrossRef]
7. Ye, W.; Heidemann, J.; Estrin, D. An energy-efficient MAC protocol for wireless sensor networks. In Proceedings of the Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies, INFOCOM, New York, NY, USA, 23–27 June 2002; pp. 1567–1576.
8. Ganesan, D.; Govindan, R.; Shenker, S.; Estrin, D. Highly-resilient, energy-efficient multipath routing in wireless sensor networks. *ACM SIGMOBILE Mob. Comput. Commun. Rev.* **2001**, *5*, 11–25. [CrossRef]
9. Chen, G.; Li, C.; Ye, M. An unequal cluster-based routing protocol in wireless sensor networks. *Wirel. Netw.* **2009**, *15*, 193–207. [CrossRef]
10. Garcia, M.; Sendra, S.; Lloret, J.; Murshed, M. Saving energy and improving communications using cooperative group-based wireless sensor networks. *Telecommun. Syst.* **2013**, *52*, 2489–2502. [CrossRef]
11. Alam, K.M.; Kamruzzaman, J.; Karmakar, G.; Murshed, M.; Azad, A.K.M. QoS support in event detection in WSN through optimal k-coverage. *Procedia Comput. Sci.* **2011**, *4*, 499–507. [CrossRef]
12. Zhou, Z.; Das, S.; Gupta, H. Connected k-coverage problem in sensor networks. In Proceedings of the 13th International Conference on Computer Communications and Networks, ICCCN 2004, Chicago, IL, USA, 11–13 October 2004; pp. 373–378.
13. Li, L.; Zhang, B.; Zheng, J. A study on one-dimensional k-coverage problem in wireless sensor networks. *Wirel. Commun. Mob. Comput.* **2013**, *13*, 1–11. [CrossRef]
14. Ammari, H.M.; Das, S.K. A study of k-coverage and measures of connectivity in 3D wireless sensor networks. *IEEE Trans. Comput.* **2010**, *59*, 243–257. [CrossRef]
15. Ammari, H.M. A unified framework for k-coverage and data collection in heterogeneous wireless sensor networks. *J. Parallel Distrib. Comput.* **2016**, *89*, 37–49. [CrossRef]
16. Ammari, H.M. 3D-k Cov-ComFor: An Energy-Efficient Framework for Composite Forwarding in Three-Dimensional Duty-Cycled k-Covered Wireless Sensor Networks. *ACM Trans. Sens. Netw.* **2016**, *12*, 35. [CrossRef]
17. Kumar, S.; Lai, T.H.; Balogh, J. On k-coverage in a mostly sleeping sensor network. In Proceedings of the 10th Annual International Conference on Mobile Computing and Networking, Philadelphia, PA, USA, 26 September–1 October 2004; ACM: New York, NY, USA, 2004; pp. 144–158.
18. Jiang, P.; Ruan, B.F. Cluster-Based Coverage-Preserving Routing Algorithm for Underwater Sensor Networks. *Acta Electron. Sin.* **2013**, *10*, 2067–2073.
19. Kim, H.; Kim, E.J.; Yum, K.H. ROAL: A randomly ordered activation and layering protocol for ensuring K-coverage in wireless sensor networks. In Proceedings of the Third International Conference on Wireless and Mobile Communications, Guadeloupe, French, 4–9 March 2007; p. 4.
20. Ammari, H.M. On the problem of k-coverage in 3D wireless sensor networks: A Reuleaux tetrahedron-based approach. In Proceedings of the 2011 Seventh International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), Adelaide, Australia, 6–9 December 2011; pp. 389–394.
21. Gupta, H.; Rao, S.; Venkatesh, T. Sleep Scheduling Protocol for k-Coverage of Three-Dimensional Heterogeneous WSNs. *IEEE Trans. Veh. Technol.* **2015**, *65*, 8423–8431. [CrossRef]
22. Pal, M.; Medhi, N. Sixsoid: A new paradigm for k-coverage in 3D wireless sensor networks. In Proceedings of the 2015 International Conference on Computing, Communication and Security (ICCCS), Pamplemousses, Mauritius, 4–5 December 2015; pp. 1–5.
23. Zeng, D.; Wu, X.; Wang, Y. *A Survey on Sensor Deployment in Underwater Sensor Networks. Advances in Wireless Sensor Networks*; Springer: Berlin/Heidelberg, Germany, 2014; pp. 133–143.
24. Liu, B.; Dousse, O.; Nain, P.; Towsley, D. Dynamic Coverage of Mobile Sensor Networks. *IEEE Trans. Parallel Distrib. Syst.* **2013**, *24*, 301–311. [CrossRef]
25. Chen, Z.; Gao, X.; Wu, F.; Chen, G. A PTAS to minimize mobile sensor movement for target coverage problem. In Proceedings of the INFOCOM 2016—The 35th Annual IEEE International Conference on Computer Communications, San Francisco, CA, USA, 10–14 April 2016; pp. 1–9.

26. Ammari, H.M. On the problem of k-coverage in mission-oriented mobile wireless sensor networks. *Comput. Netw.* **2012**, *56*, 1935–1950. [CrossRef]
27. Xia, N.; Wang, C.S.; Zheng, R. Fish swarm inspired underwater sensor deployment. *Acta Autom. Sin.* **2012**, *38*, 295–302. [CrossRef]
28. Du, H.; Xia, N.; Zheng, R. Particle Swarm Inspired Underwater Sensor Self-Deployment. *Sensors* **2014**, *14*, 15262–15281. [CrossRef] [PubMed]
29. Detweiler, C.; Doniec, M.; Vasilescu, I.; Rus, D. Autonomous Depth Adjustment for Underwater Sensor Networks: Design and Applications. *IEEE/ASME Trans. Mechatron.* **2012**, *17*, 16–24. [CrossRef]
30. Alam, K.M.; Kamruzzaman, J.; Karmakar, G.; Murshed, M. Dynamic adjustment of sensing range for event coverage in wireless sensor networks. *J. Netw. Comput. Appl.* **2014**, *46*, 139–153. [CrossRef]
31. Han, G.; Zhang, C.; Shu, L.; Rodrigues, J.J.P.C. Impacts of deployment strategies on localization performance in underwater acoustic sensor networks. *IEEE Trans. Ind. Electron.* **2015**, *62*, 1725–1733. [CrossRef]
32. Zhou, Z.; Das, S.R.; Gupta, H. Variable radii connected sensor cover in sensor networks. *ACM Trans. Sens. Netw.* **2009**, *5*, 8–19. [CrossRef]
33. Ahmed, S.; Javaid, N.; Khan, F.A.; Durrani, M.Y.; Ali, A.; Shaukat, A.; Sandhu, M.M.; Khan, Z.A.; Qasim, Q. Co-UWSN: Cooperative Energy-Efficient Protocol for Underwater WSNs. *Int. J. Distrib. Sens. Netw.* **2015**, *75*, 59–72. [CrossRef] [PubMed]
34. Dantu, K.; Rahimi, M.; Shah, H.; Babel, H.; Dhariwal, A. Robomote: Enabling mobility in sensor networks. In Proceedings of the 4th International Symposium on Information Processing in Sensor Networks, Los Angeles, CA, USA, 24–27 April 2005; IEEE Press: Piscataway, NJ, USA, 2005.
35. Elfes, A. Occupancy grids: A stochastic spatial representation for active robot perception. *arXiv* **2013**, arXiv:1304.1098.
36. Bahi, J.; Haddad, M.; Hakem, M.; Kheddouci, H. Efficient distributed lifetime optimization algorithm for sensor networks. *Ad Hoc Netw.* **2014**, *16*, 1–12. [CrossRef]
37. Liangqing, L.; Mengjue, L. *Statistics*, 2nd ed.; Hunan University Press: Changsha, China, 2014; pp. 132–149.
38. Deb, K.; Pratap, A.; Agarwal, S.; Meyarivan, T. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. Evol. Comput.* **2002**, *6*, 182–197. [CrossRef]
39. Jia, J.; Chen, J.; Chang, G.; Wen, Y.; Song, J. Multi-objective optimization for coverage control in wireless sensor network with adjustable sensing radius. *Comput. Math. Appl.* **2009**, *57*, 1767–1775. [CrossRef]
40. Huang, J.; Wang, Z.; Ma, R. A New Genetic Algorithm for Constrained Multi-objective Optimization Problems. *Comput. Eng. Appl.* **2006**, *42*, 47–51.
41. Zhou, J.; Li, J.; Zhang, Y. Efficient deployment scheme selection based on TOPSIS for clustering protocols of WSN. *J. Netw.* **2014**, *9*, 1838–1845. [CrossRef]