Extending the Lifetime of Dynamic Underwater Acoustic Sensor Networks Using Multi-Population Harmony Search Algorithm

Chun-Cheng Lin, *Member, IEEE*, Der-Jiunn Deng, *Member, IEEE*, and Shang-Bin Wang

***Abstract*—Like wireless sensor networks, lifetime of sensors is the main constraint for performance of underwater acoustic sensor networks (UASNs). Most previous works on UASNs did not consider dynamics of networks, i.e., as time goes by, in practice, part of sensors may be malfunctioned, deplete their battery power, or get lost due to violent underwater environment changes. Therefore, this work considers a UASN in ocean and proposes a sleep scheduling scheme in which sensor nodes and autonomous underwater vehicles in this network can dynamically choose to sleep or work to adapt to the environmental change. The concerned problem is to dynamically determine a sufficient number of active nodes in the UASN at different times, such that the targets required to be detected are covered. A special static scenario of the problem has been shown to be NP-complete. Hence, this work proposes an improved multi-population harmony search algorithm to solve this dynamic problem. By simulation, the proposed algorithm shows high performance in terms of extending network lifetime, robustness, and computing time.**

***Index Terms*—Underwater acoustic sensor network, harmony search algorithm, multi-population, dynamic optimization**

I. INTRODUCTION

In underwater acoustic sensor networks (UASNs) [1], [2], sensor nodes are deployed under water to form a wireless network framework to discover resources, detect targets, and monitor pollution, and so on [3], [4]. Generally, UASNs consist of heterogeneous wireless sensor nodes with the function of transmitting acoustic waves, and autonomous underwater vehicles (AUVs) with pumps (e.g., underwater gliders).

As illustrated in Fig. 1, in UASNs, AUVs and sensors are used to monitor the situations of targets; and AUVs are taken as the delay nodes, such that underwater sensors can transmit messages to AUVs, and then AUVs relay the messages to the sensors or base stations in neritic zones or above sea level. Like wireless sensor networks (WSNs), the energy source of UASNs is supported by batteries with limited power. Since sensors are deployed under water or ocean, their power cannot be recharged by solar energy [5]. Hence, it is challenging to extend the lifetime of UASNs under limited battery power. To address this problem, it is common to develop a sleep scheduling scheme to control operations of sensors, in which at different times, sensors take turns to work or sleep. At the beginning of the schedule, only a part of sensors are active, while the others sleep (are turned off) and are rescheduled until some active sensor fails due to malfunction, battery power depletion, or lost. By developing a delicate sleep scheduling scheme, the total lifetime of UASNs could be extended. This concept has widely been applied to extending the lifetime of WSNs, e.g., [6], [7]. Therefore, this work aims to develop a sleep scheduling scheme specifically for extending the lifetime of UASNs.

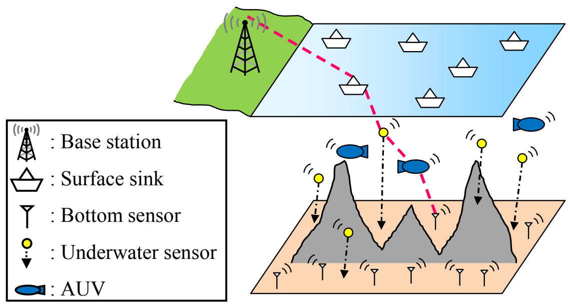

Fig. 1. Illustration of underwater acoustic sensor networks.

In some previous works on WSNs (e.g., [5], [8]), sensors are classified into multiple disjoint *covers*, each of which is a subset of sensors to cover all targets. If more covers are found, the lifetime of the WSN could be extended, as each cover can be a backup of an inactive cover. The work in [9] conducted similar research for UASNs, in which a sufficient number of sensors are deployed to cover the range to be detected such that sensors can communicate with each other. Different from [9] for the deployment problem for UASNs, this work considers the scheduling problem of dynamic heterogeneous UASNs in a 3D space, and further proposes a sleep scheduling scheme to control operations of sensors, such that sensors take turn to work or sleep at different times to avoid any redundant power consumption or malfunction. Aside from extending the network lifetime, more practical requirements are considered as

This work was supported in part by NSC 101-2628-E-009-025-MY3 and NSC 101-2221-E018-004-MY3, Taiwan. *(Corresponding author: Der-Jiunn Deng.)*

C.-C. Lin and S.-B. Wang are with Department of Industrial Engineering and Management, National Chiao Tung University, Hsinchu 3000, Taiwan. E-mails: cclin321@nctu.edu.tw, s31305911@hotmail.com

D.-J. Deng is with Department of Computer Science and Information Engineering, National Changhua University of Education, Changhua 500, Taiwan. E-mail: djdeng@cc.ncue.edu.tw

well, e.g., when some sensors are lost due to external influence, other survival sensors can be arranged to take charge of their tasks. To find the optimal sleep schedule, this work proposes a multi-population harmony search algorithm (MPHSA). The proposed MPHSA is implemented, and performance of the proposed algorithm is evaluated by simulation.

The contributions of this work are as follows. First, this work considers a dynamic problem, which meets the practice scenario. Second, in UASN, positions of some sensors are not fixed. The proposed algorithm can dynamically apply the updated positions to make a new sleep schedule. Third, the proposed metaheuristic algorithm can efficiently and effectively resolve the concerned NP-hard problem.

II. PRELIMINARIES

A. Assumption and Notation

Consider a UASN with n sensors (including AUVs, underwater and bottom sensors in Fig. 1) in a 3D cube space to monitor m targets. For example, in Fig. 2(a), there are 6 sensors s_1–s_6 and 4 targets π_1–π_4. Each sensor is associated with a sphere sensing range. Note that the sensing range size of each sensor differs due to its heterogeneous sensor type, and can be decided according to the specification of its type in practice. A base station (BS) is set up above the sea level or in a neritic zone to collect the transmitted messages. Note that since the functions of bottom sensors, underwater sensors, and AUVs are the same (i.e., to monitor targets), this problem does not specifically distinguish types of sensors.

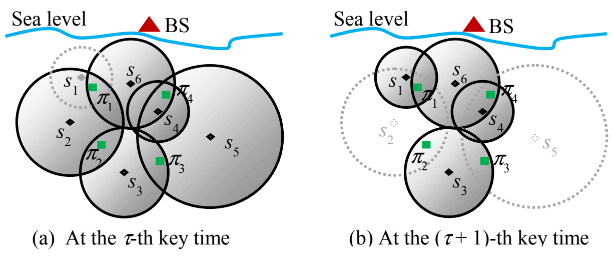

Fig. 2. An example of a dynamic UASN at two key times.

At a specific time, each sensor could be in one of four modes: *active*, *asleep*, *malfunctioned*, and *dead*. Only *active* sensors serve to detect targets and consume their battery power. To save power, sensors that are not active can be turned off, said to *sleep*. In practice, sensors may be temporarily *malfunctioned*, and could be recovered later. A sensor may be *dead* due to battery power depletion, or get lost due to external factors (e.g., flushed by ocean currents), and hence could not be recovered later. Sensors that are active or asleep are called to *survive*; and sensors that are malfunctioned or dead are called to *fail*.

Because of different sensor modes, the number and IDs of active sensors vary at different times. Hence, this work proposes a scheme to decide a *sleep schedule* at each *key time*, in which each sensor (except for failed sensors) is determined to be active or fall asleep such that all targets are covered. *Key time* is explained as follows. As illustrated in Fig. 3, the 1st key time is the initial time, at which each sensor is associated with a predefined initial battery power level. According to the UASN configuration, the proposed scheme decides a sleep schedule at the 1st key time. The sleep schedule keeps being applied until some target is not covered due to some failed sensors. The time when some target starts not to be covered is defined as the 2nd key time. At the 2nd key time, the information of survival sensors is updated; specifically, excluding failed sensors and including the sensors that are recovered at this key time. A new sleep schedule is decided for using the updated information of survival sensors to cover all targets. Similarly, the 3rd key time, 4th key time, …, and so on can be defined. And, a sleep schedule is decided at each key time until survival sensors cannot cover all targets.

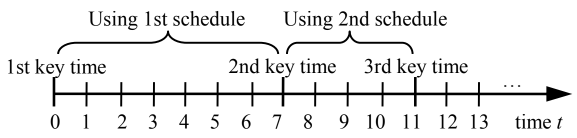

Fig. 3. The relationship between sleep schedules and key times.

With the system framework described above, the problem of concern in this work at each key time is to classify all active sensors into multiple disjoint *covers*, each of which is a subset of sensors that cover all targets. The greater number of covers is found, the longer the lifetime of the UASN could be extended. Notations used in this work are given as follows:

- S: Set of sensors, $S = \{s_1, s_2, \ldots, s_n\}$, where each s_i denotes a sensor for $i \in \{1, 2, \ldots, n\}$.
- T: Set of targets, $T = \{\pi_1, \pi_2, \ldots, \pi_m\}$, where each π_i denotes a target for $i \in \{1, 2, \ldots, m\}$.
- S_τ: Set of survival sensors at the τ-th key time.
- F_τ: Set of malfunctioned sensors at the τ-th key time, which could be recovered at later time.
- D_τ: Set of dead sensors (owing to battery power depletion or lost due to external influence, e.g., damaged or flushed by ocean currents) at the τ-th key time.
- R_τ: Set of the sensors that are recovered at the τ-th key time.
- P_f: Malfunction probability, i.e., the probability that a sensor is temporarily malfunctioned but could be recovered at later time.
- P_d: Dead probability, i.e., the probability that a sensor is dead due to battery power dissipation or external influence, so it cannot be recovered later.
- P_r: Recovery probability, i.e., the probability that a sensor is recovered to be used.
- C_τ: Set of covers at the τ-th key time, $C_\tau = \{C_1^\tau, \ldots, C_{K_\tau}^\tau\}$, where K_τ denotes number of covers at the τ-th key time.

Consider Fig. 2 for example. At the τ-th key time (Fig. 2(a)), sets of survival sensors and malfunctioned sensors are denoted by $S_\tau = \{s_2, s_3, s_4, s_5, s_6\}$ and $F_\tau = \{s_1\}$, respectively. Sensor s_1 is temporarily malfunctioned, so the contour of its sensing range is slashed-line in Fig. 2(a). Two disjoint covers are found at the τ-th key time, i.e., $C_1^\tau = \{s_2, s_5\}$ and $C_2^\tau = \{s_3, s_6\}$. Without loss of generality, suppose that C_1^τ is selected to serve to detect all targets during the period from the τ-th to (τ+ 1)-th key times. That is, s_2 and s_5 are active, while s_3 and s_6 fall asleep.

At the (τ+ 1)-th key time (Fig. 2(b)), suppose that sensor s_2 is dead due to power dissipation and sensor s_5 gets lost due to external factors (i.e., $D_{\tau+1} = \{s_2, s_5\}$), and sensor s_1 is recovered (i.e., $R_{\tau+1} = \{s_1\}$). That is, set of survival sensors is $S_{\tau+1} = \{s_1, s_3, s_4, s_6\}$. Then, the cover found at the (τ + 1)-th key time is $C_1^{\tau+1}=\{s_1,s_3,s_4\}$. Since the instance size in Fig. 2 is too small so that only one or two covers are found, multiple covers could be found at different key times in larger-scale problems.

This work considers the following assumptions. First, since the ocean environment is unpredictable, this work assumes that the monitored ocean area is relatively stable, for simplicity. Second, UASNs differs from WSNs, which allow to deploy a huge number of sensors to satisfy constraints of sensing and transmission. However, deploying UASNs is not easy and has higher cost, so it cannot satisfy those constraints.

B. Mathematical model

This work extends the two-dimensional model in [5] to a three-dimensional model that considers to transmit messages to a base station. As time goes by, some sensors may be malfunctioned or dead due to battery power depletion or external influence. Hence, the proposed model assumes fixed probabilities P_f and P_d to dynamically determine whether a sensor is temporarily malfunctioned and dead, respectively.

This work proposes a dynamic sleep scheduling scheme for sensors $S = \{s_1, s_2, \ldots, s_n\}$ in a UASN to monitor all targets in $T = \{\pi_1, \pi_2, \ldots, \pi_m\}$ at different key times. Consider the sleep schedule at the τ-th key time. First, it is required to compute the set S_τ of survival sensors at the τ-th key time as follows:

$$S_\tau = (S_{\tau\text{-}1} \cup R_\tau) \setminus (F_\tau \cup D_\tau) \qquad (1)$$

That is, set S_τ includes the survival sensors at the previous key time (i.e., $S_{\tau\text{-}1}$) and the sensors that are recovered currently (i.e., R_τ), excluding malfunctioned sensors (i.e., F_τ) and dead sensors currently (i.e., D_τ).

Let a cover C_τ be defined as a subset of S_τ whose sensing ranges cover all targets in T, expressed as follows:

$$\pi_j \in C_\tau \subseteq S_\tau \ \forall \ \pi_j \in T \qquad (2)$$

Suppose K_τ disjoint covers to be found at the τ-th key time, i.e.,

$$C_1^\tau, C_2^\tau, \ldots, C_{K_\tau}^\tau \subseteq S_\tau \qquad (3)$$

$$C_i^\tau \cap C_j^\tau \neq \varnothing \ \text{ for } i \neq j \qquad (4)$$

$$\pi_j \in C_i^\tau \ \forall \ \pi_j \in T \text{ for each } i = 1, 2, \ldots, K_\tau \qquad (5)$$

The objective of the concerned problem is as follows:

$$\text{Maximize} \quad K_\tau \qquad (6)$$

That is, this problem aims to maximize the number of disjoint covers found at each key time τ. During the period between two key times, an arbitrary cover chosen from the found covers is applied to monitoring all targets until some target cannot be monitored due to some failed sensors in this cover. By doing so, the lifetime of UASN has a higher probability to be extended.

Note that when a part of sensors are temporarily malfunctioned or dead, a centralized framework detects that the connectivity of the whole network is broken. At this key time, those failed sensors are excluded, and new disjoint covers for active sensors are found by the proposed algorithm efficiently.

The problem of computing number of covers to extend the lifetime of WSN can be reduced to the set covering problem [10], which is NP-complete [11], so it is NP-hard. Hence, this work will further propose a metaheuristic algorithm for the problem to efficiently find good-quality solutions.

C. Related Works

Most previous related works (e.g., [12]) focused on the two-dimensional UASN deployment problems with objective of minimizing the number of sensors such that all targets are covered and minimizing power consumption to extend the lifetime of the network. Aside from minimizing number of the sensors to be used, deploying positions of sensors plays an important role. A good deployment can make sensors and AUVs form a good-quality network topology and decrease power consumption to facilitate later tasks of discovering resources and monitoring objects [13], [14], [15]. The work in [16] was based on this concept to consider a linear programming model to search the optimal three-dimensional deployment of sensors. The work in [13] proposed a genetic algorithm to resolve the deployment problem with objective of minimizing the transmission delay time to decrease power consumption, and further extending the lifetime of UASNs. Consider that a part of sensors are called anchor nodes, and the others are called ordinary nodes. The work in [17] determines positions of anchor nodes to cover ordinary nodes as well as the detected ocean range and to satisfy the transmission conditions. They found that the most appropriate network topology structure consisting of anchor nodes is to deploy them as a regular tetrahedron.

III. The Proposed Method

This work proposes an improved multi-population harmony search algorithm (MPHSA) to resolve the concerned problem. Harmony search algorithm (HSA) [18] has been shown to perform better than genetic algorithm (GA), as compared with widely-used benchmark problem instances. Additionally, the work in [19] also showed that HSA performs better than GA when solving the network coverage problem in WSNs. To the best of our understanding, no previous works applied the MPHSA to solve the problems for dynamic UASNs. Hence, this work bases the HSA to design an algorithm suitable for the problem model of this work. The main merits are as follows: Multiple populations are considered for coping with dynamic problems in different environments, by keeping diversity and optimality of the solution population [20]; the key operations in the algorithm are designed specifically for the concerned problem, so that parameters and an adjustment scheme in the proposed algorithm are improved to increase the ability of searching solutions by referring the works in [21], [22].

The HSA is a metaheuristic that simulates the behavior of improvisation of multiple musicians. Each musician (decision variable) plays (generates) a musical note (a possible value for the variable) at different times. Hence, all the notes constitute a harmony (a solution for all decision variables). Given a problem, a fitness function is used to evaluate performance of harmonies. The HSA uses a harmony memory matrix *HM* to store a number of harmonies (solution population), in which each row in *HM* represents a harmony; each column corresponds to a musician, except that the last column records the fitness values of all harmonies. During each improvisation, each musician can either play one of the notes that were played ever (stored in *HM*), play a random note, or adjust her/his pitch to generate a different note. By doing so, the harmony (solution) constituted by all musicians might be improved by each improvisation. The musicians repetitively play improvisation until the most wonderful harmony (the global optimal solution for all decision variables) is found.

Since the concerned problem is dynamic, this work further extends the HSA to a multi-population version. If continuing using the original single-population HSA to solve a dynamic problem, the fitness value of each harmony is changed violently and even becomes infeasible at each key time when the environment is changed. By multiple populations, this issue could be alleviated, diversity of the solution space could be increased, and premature convergence could be avoided as well. Consider a solution population (i.e., *HM* matrix). The initial solution population is divided into multiple subpopulations (i.e., partial rows of the original *HM* matrix). Then, each subpopulation $subHM_i$ (for $i = 1, 2, \ldots, \delta$) takes charge of searching optimal solutions in a different environment. Finally, the optimal solutions found by all subpopulations are collected, and the best solution in the population is outputted as the final solution. By doing so, the ability of searching solutions space is increased to handle dynamic problems.

The proposed algorithm is given in Algorithm 1, which is explained as follows. Since the ocean environment has changed at the τ-th key time, Line 1 updates set of survival sensors S_τ by (1). Line 2 bases on S_τ to initialize harmony memory *HM* where *hms* harmonies are generated randomly and their fitness values are evaluated. Then, since the proposed algorithm is multi-population, Line 3 divides *HM* into δ equal-size subpopulations $subHM_1$, $subHM_2$, …, $subHM_\delta$. Then, the algorithm enters the main loop (Lines 4 – 19), in which η denotes the current iteration number. For each iteration, update each $subHM_i$ (Lines 6 – 17). For each $subHM_i$, if a random number from [0, 1] is smaller than the predefined *HMCR* value, then Line 8 uses tournament selection to choose two harmonies x^{new1} and x^{new2} from $subHM_i$, and Line 12 lets x^{new} be the one of x^{new1} and x^{new2} with a better fitness value; otherwise, Line 14 randomly generates a feasible harmony as x^{new}. When the above condition is true, Line 9 checks if a random number from [0, 1] is smaller than the $PAR(\eta)$ value (depending on the current iteration number η), Line 10 conducts a uniform crossover operator on x^{new1} and x^{new2}, and replaces x^{new1} and x^{new2} by the resultant two harmonies of the crossover operator. After the main loop, Line 20 decodes the best harmony in *HM* as a number of covers, and Line 21 randomly chooses one of the covers as the output, if existing. Then, the proposed sleep scheduling scheme lets sensors in the chosen cover be active, and the other survival sensors fall asleep during the period from the τ-th to (τ + 1)-th key times.

Algorithm 1 OUR_MPHSA(the τ-th key time)

```
 1: Update set of survival sensors at the $\tau$-th key time by (1)
 2: Initialize the parent harmony memory $HM$
 3: Divide $HM$ into $subHM_1, subHM_2, \ldots, subHM_\delta$
 4: Initialize the current iteration number $\eta$ to be 1
 5: while $\eta \le$ the maximal iteration number $NI$ do
 6:   for each $subHM_i$ do
 7:     if $rand(0, 1) < HMCR$ then
 8:       Use two times of tournament selection to choose two
          harmonies $x^{new1}$ and $x^{new2}$ from $subHM_i$
 9:       if $rand(0, 1) < PAR(\eta)$ then
10:         Conduct a uniform crossover operator on $x^{new1}$ and
            $x^{new2}$, and replace $x^{new1}$ and $x^{new2}$ by the resultant
            two harmonies of the crossover operator
11:       end if
12:       Let $x^{new}$ be the one of $x^{new1}$ and $x^{new2}$ with a better
          fitness value
13:     else
14:       Randomly generate a feasible harmony as $x^{new}$
15:     end if
16:     If $x^{new}$ is better than the worst harmony in $subHM_i$, $x^{new}$
        replaces it
17:   end for
18:   $\eta = \eta + 1$
19: end while
20: Decode the best harmony among all $subHM_i$'s as a
    solution of the concerned problem, i.e., number of covers
21: If number of covers is nonzero, randomly choose one of
    the covers as the output; otherwise, output "No solution
    after the $\tau$-th key time"
```

The main components of Algorithm 1 are detailed as follows.

A. Initialization of parameters

To explain the parameters used in the proposed algorithm, consider the following problem:

$$\text{Max} \quad f(x) \,/\, \text{Min} \quad c(x) \tag{8}$$

$$\text{s.t.} \quad x_i \in X_i \text{ for } i = 1, 2, \ldots, N \tag{9}$$

where (8) is the objective of the problem which could maximize $f(x)$ or minimize $c(x)$, depending on the requirement; (9) defines N decision variables, in which x_i is the value of the i-th decision variable; X_i is the feasible range for the i-th decision variable.

The parameters used in the proposed MPHSA are given as follows. *hms* denotes number of harmonies in *HM*, i.e., number of candidate solutions. *HMCR* denotes the harmony memory consideration rate, which is a ratio for choosing either one of the previous harmonies in *HM* or a random harmony as the new solution. *PAR* denotes the pitch adjusting rate, which is a probability of allowing a slight adjustment for the new solution, to increase the solution diversity and avoid premature convergence. *NI* denotes the maximal number of iterations of the main loop of the algorithm. η denotes the current iteration number. *ts* denotes the tournament size.

B. Solution representation and fitness calculation

The harmony memory matrix *HM* stores *hms* harmonies x^1, x^2, …, x^{hms} and their respective fitness values $f(x^1)$, $f(x^2)$, …, $f(x^{hms})$. For $i = 1, 2, \ldots, hms$, each musician $j \in \{1, 2, \ldots, n\}$ plays a note x_j^i impromptu (randomly), and hence, notes of all n musicians constitute a harmony $x^i = (x_1^i, x_2^i, \ldots, x_n^i)$. Then, the fitness value $f(x^i)$ of harmony x^i is computed for evaluating performance of this harmony. The *HM* matrix consists of all harmonies and their fitness values as follows:

$$HM = \begin{bmatrix} x_1^1 & x_2^1 & \ldots & x_n^1 & f(x^1) \\ x_1^2 & x_2^2 & \ldots & x_n^2 & f(x^2) \\ \vdots & \vdots & \ldots & \vdots & \vdots \\ x_1^{hms} & x_2^{hms} & \ldots & x_n^{hms} & f(x^{hms}) \end{bmatrix}.$$

Now consider how to encode a solution of the concerned problem as a harmony (i.e., a row in *HM*). At each key time τ, set S_τ of survival sensors is computed (Line 1 of Algorithm 1), say $\{s_{1'}, s_{2'}, \ldots, s_{|S_\tau|'}\}$. A harmony is encoded as a permutation of IDs of the sensors in S_τ, denoted by $\langle p_1, p_2, \ldots, p_{|S_\tau|}\rangle$ where each $p_i \in \{s_{1'}, s_{2'}, \ldots, s_{|S_\tau|'}\}$ and $p_i \neq p_j$ for $i \neq j$.

Next, consider how to decode a harmony as a solution of the concerned problem, and then evaluate the fitness of the solution. When considering a specific key time τ, position of each sensor determines what targets are covered by this sensor. Hence, the relationship between sensors and the covered targets can be represented as a bipartite graph $G_\tau = (S_\tau, T, E)$, in which S_τ and T are two sets of vertices; if a sensor s_i in S_τ covers a target π_j in T, then an edge (s_i, π_j) exists in E. For instance, the deployment in Fig. 2(a) corresponds to the bipartite graph in Fig. 4. Obviously, any matching in this bipartite graph is a cover in the concerned UASN. Decoding a harmony $\langle p_1, p_2, \ldots, p_{|S_\tau|}\rangle$, i.e., a permutation of IDs of sensors in S_τ, is given in Algorithm 2.

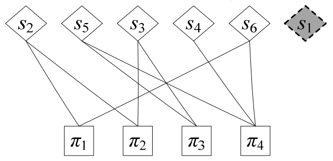

Fig. 4. The bipartite graph that represents the deployment in Fig. 2(a).

Algorithm 2 is explained as follows. The input of the algorithm is a harmony $\langle p_1, p_2, \ldots, p_{|S_\tau|}\rangle$. First, k is used to record the number of covers to be found, and hence, Line 1 lets $k = 1$ initially and Line 2 lets $C_k^\tau = \varnothing$ to initialize the 1st cover. Then, Lines 3 – 8 are a for loop that considers each ID in harmony $\langle p_1, p_2, \ldots, p_{|S_\tau|}\rangle$ from the leftmost to the rightmost. In the i-th iteration, p_i is included to the current cover set C_k^τ (Line 4). Line 5 checks if C_k^τ is a cover according to bipartite graph G_τ. If true, it means that the current cover set can monitor all targets, and hence we consider the next cover set by increasing the k counter (Line 6). Lines 9 – 13 calculate and output the fitness value according to the found cover sets. That is, the fitness value is defined as follows:

$$K_\tau \cdot |T| + \left|\bigcup_{s_i \in U} e(s_i)\right| \tag{10}$$

where U denotes set of sensors not in any found cover; $e(s_i)$ is an edge adjacent to vertex s_i in bipartite graph G_τ. The first term in (10) is the total times of targets monitored by the found covers, and the second term in (10) is the number of the targets monitored by the sensors not in any cover.

Algorithm 2 EVALUATE_FITNESS(harmony $\langle p_1, p_2, \ldots, p_{|S_\tau|}\rangle$)

```
 1: Let $k = 1$
 2: Let $C_k^\tau = \varnothing$
 3: **for** $i = 1, \ldots, |S_\tau|$ **do**
 4:     $C_k^\tau = C_k^\tau \cup \{p_i\}$
 5:     **if** $C_k^\tau$ is a cover according to bipartite graph $G_\tau$ **then**
 6:         $k = k + 1$
 7:     **end if**
 8: **end for**
 9: **if** $C_k^\tau$ is a cover **then**
10:     Output $k \cdot |T|$
11: **else**
12:     Output $(k - 1) \cdot |T|$ + number of the targets monitored by
        sensors in $C_k^\tau$
13: **end if**
```

C. Tournament selection

In each iteration, for each subpopulation $subHM_i$, a new harmony $x^{new} = \{x_1^{new}, x_2^{new}, \ldots, x_n^{new}\}$ is generated by applying tournament selection on the $subHM_i$ matrix (Lines 8 and 12 of Algorithm 1) or is a random harmony (Line 14 of Algorithm 1). Since each value in x^{new1} is discrete, a special design for tournament selection is required as follows.

The ratio of choosing one of the previous harmonies in $subHM_i$ as the new harmony is determined by the predefined *HMCR* (harmony memory considering rate) parameter. Algorithm 1 first generates a random value r in [0, 1], and then verifies if r is smaller than the *HMCR* value (Line 7). If true, Line 8 of Algorithm 1 conducts two times of tournament selection (by analogy from [21]), different from the classical HSA. Consider a parameter called tournament size (*ts*). Tournament selection randomly chooses *ts* harmonies from the *HM* matrix, and takes the best harmony among the *ts* harmonies for later operations. Since two times of tournament selection are conducted, two harmonies x^{new1} and x^{new2} are generated. Then, Line 12 of Algorithm 1 lets x^{new} be the one of x^{new1} and x^{new2} with a better fitness value.

D. Uniform crossover

Aside from adjustment by *HMCR*, the HSA adjusts the new generated harmony with the *PAR* (pitch adjusting rate) value. Different from the classical HSA, this work applies a dynamic *PAR* value by analogy from [22]. After lots of experimental trials on a fixed benchmark experimental dataset, the work in [22] found that when number of iterations of the HSA increases, the ability of searching optimality is affected by related parameter settings. Hence, a dynamic adjustment scheme for the *PAR* value is derived as follows:

$$PAR(\eta) = PAR_{\min} + (PAR_{\max} - PAR_{\min}) / NI \cdot \eta \quad (11)$$

where η is the current number of iterations; $PAR(\eta)$ is the *PAR* value used in the η-th iteration; $PAR_{\min}$ is the predefined minimum *PAR* value; $PAR_{\max}$ is the predefined maximum *PAR* value; *NI* is the maximal iteration number. By using a dynamic *PAR* value in the HSA, high performance has been shown.

In Algorithm 1, Line 9 generates a random value in [0, 1], and verifies if it is smaller than the $PAR(\eta)$ value. If true, Line 10 conducts a uniform crossover operator on x^{new1} and x^{new2}. Note that [23] has shown that uniform crossover operator performs better than other crossover operators to search optimal solutions in HSA. This work applies the crossover operator as follows. Consider the example in Fig. 5 to explain the uniform crossover operator. At first, two harmonies $x^{new1} = \langle s_2, s_5, s_3, s_4, s_6\rangle$ and $x^{new2} = \langle s_5, s_6, s_4, s_2, s_3\rangle$ are chosen from *HM* by two times of tournament selection (Fig. 5(a)). Based on the bipartite graph in Fig. 4, the number of covered targets of each sensor sequentially appearing in x^{new1} and x^{new2} is calculated (Fig. 5(a)). For fitness of x^{new1}, "$f(x^{new1}) = 2 + 2 + 2 + 1 + 1 = 8$" represents that sensors s_2 and s_5 contribute "2 + 2" as they constitute a cover (i.e., sensor s_2 covers two targets π_1 and π_2; sensors s_5 covers two targets π_3 and π_4); similarly, sensors s_3 and s_4 contribute another "2 + 2"; s_4 contributes 1 as it covers target π_4; sensor s_6 contributes 1 as it covers only target π_1.

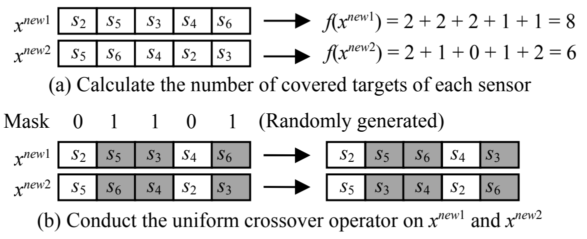
x^{new1} s_2 s_5 s_3 s_4 s_6 → $f(x^{new1}) = 2 + 2 + 2 + 1 + 1 = 8$
x^{new2} s_5 s_6 s_4 s_2 s_3 → $f(x^{new2}) = 2 + 1 + 0 + 1 + 2 = 6$
(a) Calculate the number of covered targets of each sensor
Mask 0 1 1 0 1 (Randomly generated)
x^{new1} s_2 s_5 s_3 s_4 s_6 → s_2 s_5 s_6 s_4 s_3
x^{new2} s_5 s_6 s_4 s_2 s_3 → s_5 s_3 s_4 s_2 s_6
(b) Conduct the uniform crossover operator on x^{new1} and x^{new2}

Fig. 5. An example of the uniform crossover operator.

Next, a uniform crossover operator is conducted on x^{new1} and x^{new2} as follows: First, a uniformly random binary number of length $|S_\tau|$ is generated as a mask that is used later. An entry in x^{new1} and x^{new2} is shaded if its corresponding position in the mask is 1. Without loss of generality, considering harmony x^{new1}, each shaded sensor in x^{new1} is swapped with the sensor at the same position in x^{new2} if no violation occurs – the same sensor does not appear more than twice in x^{new1}; otherwise, omit the sensors in x^{new2} that are not shaded in x^{new1}, and then take the remaining sensors in x^{new2} to replace the sensors with violation in x^{new1}. Specifically, for x^{new1} in Fig. 5(b), shaded sensors s_5, s_3, and s_6 in x^{new1} should be replaced by s_6, s_4, and s_3 in x^{new2}, respectively, but the replacement is infeasible. Hence, we first omit sensors s_2 and s_4 in x^{new2} and then observe that the remaining sensors in x^{new2} are s_5, s_6, and s_3 in sequel. Hence, after the shaded sensors in x^{new1} are replaced by the reaming sensors in x^{new2}, it turns out that the resultant harmony from x^{new1} is s_2 s_5 s_6 s_4 s_3, and by a similar way, the result harmony from x^{new2} is s_5 s_3 s_4 s_2 s_6 (Fig. 5(b)).

E. Updating the HM matrix

Line 16 in Algorithm 1 accepts or rejects x^{new} according to its fitness value. If harmony x^{new} has a better fitness value than the worst harmony in $subHM_i$, then the worst harmony in $subHM_i$ is replaced by harmony x^{new}; otherwise, x^{new} is discarded.

IV. IMPLEMENTATION AND SIMULATION RESULTS

This section implements the proposed MPHSA. Flowchart of the simulation design is given in Fig. 6. After lots of experimental trials, the parameter settings used in the proposed algorithm are as follows: number of subpopulations δ is 8; harmony memory size (*hms*) is 80; harmony memory considering rate (*HMCR*) is 0.95; minimum pitch adjusting probability ($PAR_{\min}$) is 0.35; maximum pitch adjusting probability ($PAR_{\max}$) is 0.99; number of iterations (*NI*) is 200; tournament size (*ts*) is 2.

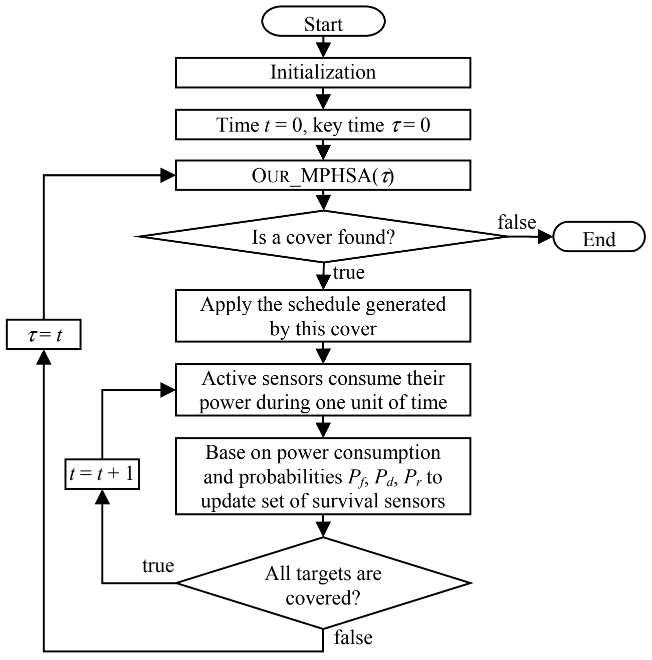
Start
Initialization
Time $t = 0$, key time $\tau = 0$
OUR_MPHSA(τ)
Is a cover found?
false
End
true
Apply the schedule generated by this cover
$\tau = t$
Active sensors consume their power during one unit of time
$t = t + 1$
Base on power consumption and probabilities P_f, P_d, P_r to update set of survival sensors
true
All targets are covered?
false

Fig. 6. Flowchart of the simulation design.

By referring many previous works, the experimental problem instances are created, and the default parameter setting in those instances is as follows: size of the 3D cube space to be monitored is $50 \times 50 \times 50$; number of sensors is 300; number of targets is 10; malfunction probability of each sensor (P_f) is 0.001; recovery probability of each sensor (P_r) is 0.001; dead probability of a sensor (P_d) is 0.0001; radii of sensing ranges of two types is 25 or 35; initial battery power of each sensor is 100; power consumption rate of an active sensor is 1 unit/unit time. Note that each sensor covers at least one target initially. The experiments are conducted on a PC with an Intel Core i7-4770

CPU and 3.40 GHz memory. Since the MPHSA converges after about 200 iterations, we run 200 iterations of the algorithm to find a cover at each key time. The average CPU time of running 200 iterations of MPHSA is about 0.139s. Note that it is hard to estimate the values of P_f, P_r, P_d in real world, and hence, for simplifying the simulation, this work assumes that P_f, P_r, P_d are fixed. It would be of interest to estimate them in the future. Additionally, since no pervious works studied the same problem of concern in this work, the proposed algorithm MPHSA is compared with the classical HSA.

A. Simulations with different numbers of sensors and targets

This section analyzes the results with different numbers of sensors and targets, to realize scalability of the algorithm. Fig. 7 (resp., Fig. 8) gives the plots of fitness values using HSA and MPHSA when number of sensors $|S|$ is 100 or 500 (resp., number of targets is 10 or 50), in which the fitness value has a sudden decrease at every 200 iterations, and later rises to a higher fitness level, i.e., both the two algorithms can adapt to the change. Each multiple of 200 iterations determines a number of covers at a key time; then, an arbitrary cover among them is chosen to serve to detect all targets until some target is not covered. The total lifetime is determined by the applied covers in sequel, but cannot be observed from the plots. Hence, even if the fitness using MPHSA falls to zero earlier than HSA, it does not imply that the MPHSA achieves shorter lifetime. On average, more sensors (Fig. 7) and more targets (Fig. 8) lead to more possible covers to be found, so achieve higher fitness.

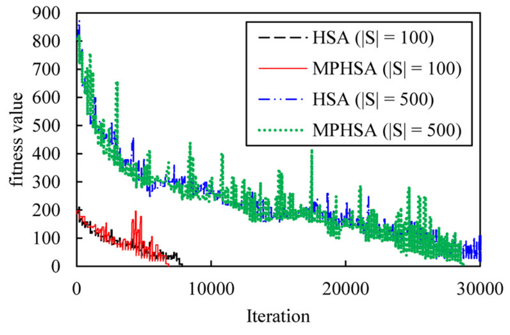

Fig. 7. Experimental comparison when number of sensors $|S|$ is 100 or 500.

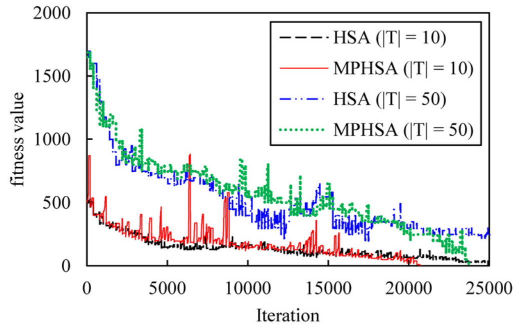

Fig. 8. Experimental comparison when number of targets $|T|$ is 10 or 50.

To analyze the UASN lifetime, Table I records the average (Avg) and standard deviation (StdDev) of the UASN lifetime for different numbers of sensors when 20 runs of the corresponding algorithm are executed. From Table I, the average lifetime of MPHSA is always longer than that of HSA in all cases. Although the standard deviation in Table I has no consistent conclusion, the MPHSA has smaller deviation when the number of sensors becomes larger.

TABLE I
STATISTICS OF THE LIFETIME UNDER DIFFERENT NUMBERS OF SENSORS

	HSA		MPHSA	
$\|S\|$	Avg	StdDev	Avg	StdDev
100	1176.05	154.69	1241.51	173.20
200	2952.10	250.63	2966.70	237.70
300	3693.65	350.92	3716.75	253.71
400	5461.80	576.91	5505.60	324.20
500	8210.15	788.53	8721.70	614.95

B. Simulations with different probability settings

This section analyzes the results with different probability settings, to realize the solution-search ability of the algorithm. Fig. 9, Fig. 10, and Fig. 11 give the plots of fitness values using HSA and MPHSA with different values of malfunction probability P_f, recovery probability P_r, and dead probability P_d, respectively. Note that the change scale of P_d in Fig. 11 is smaller than that of P_f in Fig. 9 because a dead sensor cannot be recovered so that the lifetime is too short. From those figures, the MPHSA performs better than the HSA on average.

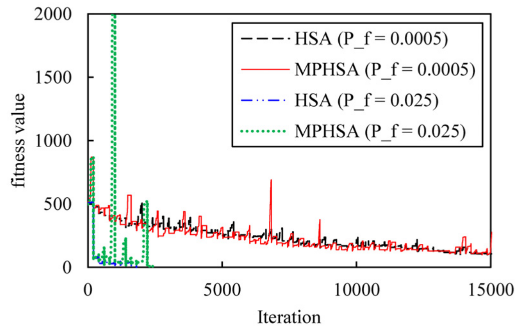

Fig. 9. Experimental comparison when $P_f = 0.0005$ or 0.025.

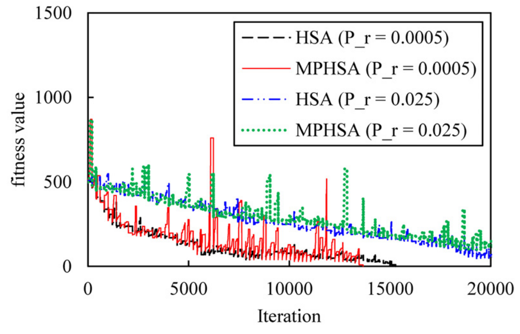

Fig. 10. Experimental comparison when $P_r = 0.0005$ or 0.025.

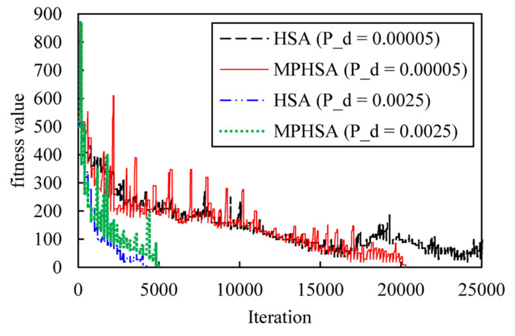

Fig. 11. Experimental comparison with $P_d = 0.00005$ or 0.0025.

From Fig. 9, when the malfunction probability increases, it is reasonable that the fitness decreases remarkably. Relatively, when the recovery probability in Fig. 10 increases with the same scale as Fig. 9, the fitness does not increase too much. This is reasonable because the effect of the recovery probability is restricted to the number of malfunctioned sensors. In Fig. 11, when the dead probability increases, the fitness decreases a lot.

V. CONCLUSION

This work has proposed a multi-population harmony search algorithm to resolve the problem of extending the lifetime of UASNs. We extends the two-dimensional model for WSNs to a 3D model with more practical concerns on UASNs. Different from WSNs, UASNs in this work not only consider that sensors may be inactive due to power consumption as time goes by, but also consider that sensors get lost due to change of the ocean environment. In the future, the UASN model for more practical requirements can be developed, because the ocean environment is unpredictable. It is also of interest to apply the proposed method for real-world UASN problems, to verify practicability and applicability of the proposed algorithm. This work can be extended with novel approaches, problem settings, and cross-layer design, e.g., using remotely powered UASNs [24].

REFERENCES

[1] D. Pompili, and I. F. Akyildiz, "Overview of networking protocols for underwater wireless communications," *IEEE Commun. Mag.*, vol. 47, pp. 97-102, 2009.

[2] M. Chitre, S. Shahabudeen, and M. Stojanovic, "Underwater acoustic communication and networks: Recent advances and future challenges," *Mar. Tech. Soc. J.*, vol. 42, pp. 103-116, 2008.

[3] P. Salvo Rossi, D. Ciuonzo, T. Ekman, and H. Dong, "Energy detection for MIMO decision fusion in underwater sensor networks," *IEEE Sensors J.*, vol. 15, no. 3, pp. 1630-1640, 2015.

[4] I. F. Akyildiz, D. Pompili, and T. Melodia, "Underwater acoustic sensor networks: Research challenges," *Ad Hoc Netw.*, vol. 3, pp. 257-279, 2005.

[5] C.-C. Liao and C.-K. Ting, "Extending the lifetime of dynamic wireless sensor networks by genetic algorithm," in *Proc. IEEE CEC*, 2012, pp. 1-8.

[6] F. Liu and C.-Y. Tsui, "Joint routing and sleep scheduling for lifetime maximization of wireless sensor networks," *IEEE Trans. Wireless Commun.*, vol. 9, pp. 2258-2267, 2010.

[7] Y. Zhao, J. Wu, F. Li, and S. Lu, "On maximizing the lifetime of wireless sensor networks using virtual backbone scheduling," *IEEE Trans. Parallel Distrib. Syst.*, vol. 23, pp. 1528-1535, 2012.

[8] C.-K. Ting and C.-C. Liao, "A memetic algorithm for extending wireless sensor network lifetime," *Inform. Sciences*, vol. 180, pp. 4818-4833, 2010.

[9] K. Akkaya and A. Newell, "Self-deployment of sensors for maximized coverage in underwater acoustic sensor networks," *Comput. Commun.*, vol. 32, pp. 1233-1244, 2009.

[10] S. Slijepcevic and M. Potkonjak, "Power efficient organization of wireless sensor networks," in *Proc. IEEE ICC*, vol. 2, 2001, pp. 472-476.

[11] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, NY: W. H. Freeman, 1979.

[12] Y.-Y Zhang, X. Li, and S.-L. Fang, "Deployment analysis in two-dimensional underwater acoustic wireless sensor networks," in *Proc. ICSPCC*, IEEE Press, 2011, pp. 1-5.

[13] J. Nie, D. Li, and Y. Han, "Optimization of multiple gateway deployment for underwater acoustic sensor networks," *Comput. Sci. Inf. Syst.*, vol. 8, pp. 1073-1095, 2011.

[14] G. Brataas, A. Lie, and T.A. Reinen, "Scalability analysis of underwater sensor networks," in *Proc. MTS / IEEE. Conf. Oceans*, 2013, pp. 1-9.

[15] S. Ibrahim, J. Liu, M. Al-Bzoor, J.-H. Cui, and R. Ammar, "Towards efficient dynamic surface gateway deployment for underwater network," *Ad Hoc Netw.*, vol. 11, pp. 2301-2312, 2013.

[16] S. Iyer and D. V. Rao, "Genetic algorithm based optimization technique for underwater sensor network positioning and deployment," in *Proc. IEEE UT*, 2015, pp. 1-6.

[17] G. Han, C. Zhang, L. Shu, and J.J.P.C. Rodrigues, "Impacts of deployment strategies on localization performances in underwater acoustic sensor networks," *IEEE Trans. Ind. Electron.*, vol. 62, pp. 1725-1733, 2014.

[18] Z.-W. Geem, J.-H. Kim, and G.V. Loganathan, "A new heuristic optimization algorithm: harmony search," *J. Simulat.*, vol. 76, pp. 60-68, 2001.

[19] S. E. Nezhad, H. J. Kamali, and M. E. Moghaddam, "Solving K-coverage problem in wireless sensor networks using improved harmony search," in *Proc. BWCCA*, IEEE Press, 2010, pp. 49-55.

[20] W. Jiang, J. Wang, W. Wang, L.-L. Cao, and Q. Jin, "A parallel harmony search algorithm with dynamic harmony-memory size," in *Proc. CCDC*, IEEE Press, 2013, pp. 2342-2347.

[21] M. Castelli, S. Silva, L. Manzoni, and L. Vanneschi, "Geometric selective harmony search," *Inform. Sciences*, vol. 279, pp. 468-482, 2014.

[22] M. Mahdavi, M. Fesanghary, and E. Damangir, "An improved harmony search algorithm for solving optimization problems," *Appl. Math. Comput.*, vol. 188, pp. 1567-1579, 2007.

[23] G. Syswerda, "Uniform crossover in genetic algorithms," in *Proc. ICGA*, 1989, pp. 2-9.

[24] A. Bereketli and S. Bilgen, "Remotely powered underwater acoustic sensor networks," *IEEE Sensors J.*, vol. 12, no. 12, pp. 3467-3472, 2012.