Data Collection and Event Detection in the Deep Sea with Delay Minimization

Huanyang Zheng and Jie Wu
Department of Computer and Information Sciences, Temple University, USA
Email: {huanyang.zheng, jiewu}@temple.edu

***Abstract*—As special applications of delay tolerant networks (DTNs), efficient data collection and event detection in the deep sea pose some unique challenges, due to the need of timely data reporting and the delay of acoustic transmission in the ocean. Since underwater communications suffer from a significant signal attenuation, autonomous underwater vehicles (AUVs) deployed in the deep sea are used to surface frequently to transmit collected data and events to the surface stations. However, extra delay is introduced at each resurfacing, since AUVs are usually operated in the deep sea. In this paper, we want to minimize the average data and event reporting delay, through optimizing the number and locations of AUV resurfacing events. We also study the AUV trajectory planning using an extended Euler circuit, where the search space is a set of segments (e.g., oil pipes) in the deep sea. Finally, experiments in both the synthetic and real traces validate the efficiency and effectiveness of the proposed algorithms.**

***Keywords*—Deep sea searching, delay tolerant networks, autonomous underwater vehicles, Euler circuit, scheduling.**

I. Introduction

The deep sea is the largest habitat on earth and is largely unexplored. As shown in the recent search-and-rescue effort of Malaysia flight MH370 in the Pacific Ocean, it is extremely difficult to conduct an efficient search process in the deep sea for data collection and event detection. In addition to the huge area of the search space, the data (or events) reporting in the deep sea also pose a unique challenge, compared to those in regular land communications. Although several different types of media can be used under the sea, the acoustic transmission [1, 2] is most commonly used for underwater communications. However, it is well known that the acoustic transmission suffers from a very significant signal attenuation (and thus a low data rate). Therefore, to report data in a search-and-rescue effort, autonomous underwater vehicles (AUVs) deployed in the deep sea are used to surface frequently and transmit collected data (or events) to the surface station. A motivational example for the AUVs could be the detection of oil pipe leaks through robotic submarines in the Gulf of Mexico [3].

In this paper, we consider a special scheduling problem aiming to minimize the average data reporting delay. AUVs are used to search and collect data in a given 2-dimensional (2-D) search space, which is parallel to the water surface with a given depth. The data (or events) reporting should be done in a timely manner, however, extra delay is introduced at each AUV resurfacing. Fig. 1 shows such a scenario of data reporting from the deep sea. We consider the search space to be a set of segments (e.g., oil pipes), which is represented as a set of weighted edges in a graph. We propose an AUV trajectory planning using an extended Euler circuit, and then,

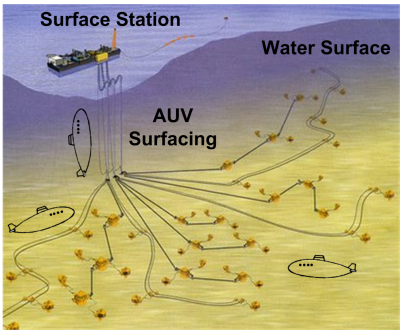
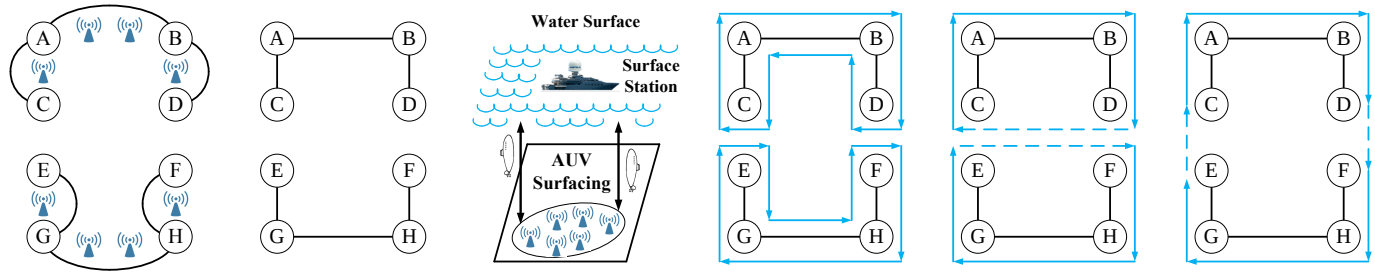


Fig. 1. Data reporting and event detection in the deep sea.

we determine the number and locations of resurfacing events on the circuit (or simply *cycle*). Specifically, we study the following problems in sequence. (1) Given the circumference of a cycle of a search space at a given depth, we determine the number and locations of AUV resurfacing events that minimize the average data reporting delay. (2) We study a more general case where the search space is a collection of edges, called *sensing edges*. AUVs can collect the data from the sensing edges. We then determine cycles that cover all sensing edges, where some edges may appear more than once. (3) Using the geometric property, we replace some multiple-visited sensing edges with geometrically-shortest-distance links that are not sensing edges in the graph (called *non-sensing edges*), as to shorten the cycle circumference. Note that no data is collected from the non-sensing edges. We also adjust the number and locations of AUV resurfacing events for cycles with non-sensing edges. (4) Given a search space that includes multiple cycles, we study a cooperative AUV trajectory planning, where the cycles are merged to further reduce the average data delay.

The key difference between our approach and the classic ferry approaches [4, 5] lies in the AUV resurfacing events that bring an extra delay. If the AUVs resurface frequently, then the uncollected data needs to wait a longer time to be reported, which leads to an increased average data reporting delay. On the other hand, if the AUVs resurface infrequently, then the collected data within the AUV needs to wait a longer time to be reported, which also leads to an increased average data reporting delay. This tradeoff poses some unique challenges on combining the design of AUV resurfacing events and trajectory planning in the deep sea, which have not been explored in existing works on underwater sensor networks [6–9] and corresponding protocols [10–15]. Although a preliminary work was proposed in [16], we conduct further extensions by considering AUV surfacing events in cycles with non-sensing edges, as well as the cooperative AUV trajectory planning.



(a) Search space layout. (b) The given graph. (c) AUV resurfacing. (d) Cycle construction. (e) Cycle enhancement. (f) Cycle merge.

Fig. 2. An illustration for the background and problem formulation.

The reminder of the paper is organized as follows. Section II surveys the related work. Section III states the background and the problem formulation. Section IV studies the number and locations of AUV resurfacing events, given a search space of a cycle with a depth under the water surface level. Section V focuses on finding a small cycle that includes all sensing edges in a given graph. Section VI discusses an extension of using geometrically-shortest-distance links to shorten the cycle. We also provide a refinement to the number and locations of AUV resurfacing events, given a cycle that includes both sensing and non-sensing edges. Section VII studies the cooperative AUV resurfacing, where we merge the cycles for the trajectory planning. Section VIII includes the experiments. The paper concludes in Section IX with a discussion of the future work.

II. Related Work

Recently, underwater sensor networks [6, 9] are becoming a hot topic, since our ocean remains largely under-explored. It is notoriously difficult to conduct an efficient search process in the deep sea for data collection and event detection, as shown in the recent search-and-rescue effort of Malaysia flight MH370 in the Pacific Ocean [17]. A significant amount of work has been reported on underwater sensor networks. For example, Chandrasekhar et al. [7] surveyed the localization problem in underwater networks. Pompili et al. [8] studied the routing algorithms for delay-insensitive and delay-sensitive applications, while more detailed surveys are reported in [10, 11]. The monitoring problem has also been studied. Eichhorn et al. [15] designed a modular AUV system for the sea water quality monitoring. Jawhar et al. [14] proposed an efficient framework in AUV-extended sensor networks for pipeline monitoring. We also consider the detection of oil pipe leaks through AUVs. However, we mainly focus on the AUV resurfacing decisions and the AUV trajectory planning (rather than a framework).

In traditional sensor networks, ferries are used to collect the data from different sensors [4, 18, 19]. In [4], a set of special mobile nodes called 'message ferries' are responsible for carrying data for nodes in the network. The design of ferry routes (i.e., trajectory planning) was focused. The key difference between our approach and the traditional ferry approach lies in the AUV resurfacing process that brings an extra delay. Traditional ferry approaches [20–24] are based on sensors distributed in a 2-dimensional space, where ferries are used to move among different sensors for data collection. They are usually formulated as traveling salesman problems (TSPs) or the extensions of TSPs. By comparison, this paper considers the AUV resurfacing as the third dimension for the AUV movement. We consider AUVs to go along pipes (i.e., edge traversal), which is formulated as Eulerian cycle problems.

III. Background and Problem Formulation

This paper studies the data collection and event detection in the deep sea with delay minimization. We are motivated by the detection of oil pipe leaks through robotic submarines in the Gulf of Mexico [3]. As shown in Fig. 2(a), we study a search space that is a set of oil pipes deployed in the seabed. Nodes are sources or destinations of oil pipes, which are not necessarily linear. Sensors are densely and uniformly deployed along pipes to detect the leakages. Another application scenario of our work could be the seabed settlement monitoring [25], where the sensors are deployed to monitor the seabed.

Since underwater communications suffer from a significant signal attenuation (and thus a low data rate), AUVs are used to go along the pipes to collect the data from the sensors, and then surface to report the data. The above data collection and AUV resurfacing are periodic. Our objective is to collect and report the data with minimized average delay. If the AUVs resurface frequently, then the uncollected data needs to wait a longer time to be reported, which leads to an increased average data reporting delay. On the other hand, if the AUVs resurface infrequently, then the collected data within the AUV needs to wait a longer time to be reported, which also leads to an increased average data reporting delay. For further processing, the search space is converted to a given graph with a certain depth in the sea, as shown in Fig. 2(b). The lengths of the pipes are the edge weights in the given graph. The edges in the given graph are also called sensing edges.

In Section IV, we will start with an ideal case, where the given graph is composed of only one cycle. As shown in Fig. 2(c), we would like to determine the number and locations of resurfacing events that minimize the average data delay. However, the assumption that the given graph is cyclic may not be very practical. Therefore, in Section V, we discuss how to construct cycles from the given graph, based on the extended Eulerian cycles. As shown in Fig. 2(d), each connected component in the given graph of Fig. 2(b) is converted to a cycle (i.e., cycles ABDBACA and EGHFHGE). The constructed cycles are only composed of sensing edges, where some edges may appear more than once, as the given graph is not necessarily Eulerian. Then, we could use the results in Section IV to schedule the AUVs for each constructed cycle.

In Section VI, we would improve the cycle construction, through replacing some multiple-visited sensing edges with geometrically-shortest-distance links that are not sensing edges in the graph, as to shorten the circumference of the resultant cycle. These geometrically-shortest-distance links are called non-sensing edges, since no data is collected from them. An example is shown in Fig. 2(e), where we use the non-sensing edges of DC and EF to shorten the circumferences of the cycles in Fig. 2(d). Smaller circumferences of the constructed cycles can result in smaller average data reporting delays. Furthermore, in Section VII, we observe that cycles can be merged with a cooperative AUV scheduling. As shown in Fig. 2(f), the two smaller cycles in Fig. 2(e) are merged, leading to a bigger cycle of ABDFHGECA. The cycle merge can also reduce the average data reporting delay [16].

IV. Resurfacing Frequency

We start with a cycle of search space in a given depth with several AUVs, as shown in Fig. 2(c). We determine the AUV resurfacing frequency, as to minimize the average data delay. In the search space, sensors or events are uniformly distributed along the cycle, while data or events have a constant generation rate. In subsequent discussions, we use data to represent both data in data collection and events in event detection.

Let us consider the case of only one AUV, which has a unit speed. Let C denote the circumference of the cycle. For simplicity, we consider that the depth from the search space to the water surface is fixed (denoted by L). Note that the cruising speed and the diving/surfacing speed of the AUV may not be the same. However, they can still be converted to the unit speed through distance scaling. An example is shown as follows. Suppose the cruising speed and the diving/surfacing speed of the AUV are 10m/s and 5m/s, respectively. When $C = 5,000$m and $L = 1,000$m, this case is equivalent to the scenario where the AUV has the unit speed with $C = 500$m and $L = 200$m (the average data reporting delay remains the same). For presentation simplicity, AUVs are assumed to have unit speeds in the following paper.

Let k denote the surfacing frequency per circulation of the cycle. The locations for surfacing are uniformly distributed along the cycle. We consider the data generation rate of the sensor to be larger than $\frac{1}{C}$, which implies that an AUV can always collect new data when it re-circulates the cycle. The objective is to minimize the average data reporting delay, from the time that the data is generated to the time that the data arrives at the water surface. It is assumed that the data can then be quickly transmitted in the air to a base station (and this part of delay is neglected). Therefore, the overall data reporting delay includes three parts as follows:

- For the AUV, its actual travel length is $C + 2kL$ per circulation of the cycle. Here, $2kL$ includes k times of surfacing of depth L, counting both AUV coming up and going down. On average, each data item needs to wait for a time of $\frac{C+2kL}{2}$ before being transmitted from the sensor to the AUV.
- The cycle is partitioned into k intervals by the surface points. The average delay, from the time that the data is received by the AUV to the time that the AUV arrives the surface point, is $\frac{C}{2k}$.
- Finally, the surfacing process takes a time of L.

In total, the average data reporting delay for one AUV (denoted by D_1) can be calculated as follows:

$$D_1 = \frac{C + 2kL}{2} + \frac{C}{2k} + L \tag{1}$$

Eq. 1 is minimized to $\frac{C}{2}+\sqrt{2LC}+L$, when $k=\sqrt{\frac{C}{2L}}$ (the surfacing frequency). This analysis gives the following theorem.

Theorem 1: Optimally, the AUV resurfaces after traveling a distance of $\frac{C}{k} = \sqrt{2LC}$ on the original cycle.

Here, we define the length of $\sqrt{2LC}$ as an *optimal interval*. When $L = 2C$, the traveling distance before resurfacing is $2C$, i.e., once every two circulations of the cycle. The insight behind optimal resurfacing is a trade-off: As k increases, waiting time for the AUV increases, but time spent on AUV before resurfacing also reduces. $k=\sqrt{\frac{C}{2L}}$ is the optimal value that balances the above tradeoff.

Now, suppose we have n AUVs for one cycle. Using a calculation that is analogous to Eq. 1, the average data reporting delay for n AUVs (denoted by D_n) is

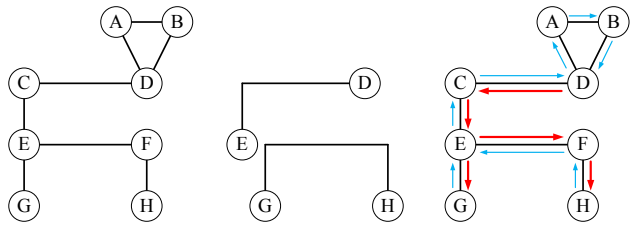
$$D_n = \frac{C + 2kL}{2n} + \frac{C}{2k} + L \tag{2}$$

Eq. 2 is minimized to $\frac{C}{2n} + \sqrt{\frac{2LC}{n}} + L$, when $k = \sqrt{\frac{nC}{2L}}$. As a corollary of Theorem 1, the optimal scheduling is that n AUVs start as being uniformly distributed on the cycle, and each AUV resurfaces after traveling a distance of $\frac{C}{k} = \sqrt{\frac{2LC}{n}}$.

V. Cycle Construction

In the previous section, it is assumed that the traveling cycle for AUVs is given. In this section, we focus on constructing such a cycle in a given search space, aiming to minimize the circumfluence of the cycle. We assume that the search space is a set of segments (oil pipes), represented by a weighted given graph G. The cost associated with each edge in G is the length of the corresponding segment (the length of the oil pipe). An example of the search space is shown in Fig. 2(a), while the corresponding given graph is shown in Fig. 2(b).

In graph theory, an Eulerian trail in a graph is a trail which visits every edge exactly once. Similarly, an Eulerian circuit or Eulerian cycle is an Eulerian trail which starts and ends on the same vertex. Eulerian cycle exists, if and only if each vertex in the given graph has an even degree. Given an Eulerian graph, we can construct such a cycle in a linear time proposed by Hierholzer [26]: Choose any starting vertex v in G, and follow a trail of edges from that vertex until it returns to v. It is not possible to get stuck at any vertex other than v. This is because the even degrees of all vertices ensure that, when the trail enters another vertex u, there must be an unvisited edge leaving u. The trial formed in this way may not visit all the edges of the given graph. As long as there exists a vertex v that belongs to the current trail and v has adjacent edges that are unvisited, we can start another trail from v, following unvisited edges until they return to v. This new tour starting at v can join the previous tour. If we repeat the above process, then all edges can be eventually visited by the tour.



(a) Given graph. (b) Odd-degree vertex matching. (c) Eulerian cycle.

Fig. 3. An example for Algorithm 1, where the extended Eulerian cycle is ABDCEFHFEGECDA.

Algorithm 1 Extended Eulerian cycle

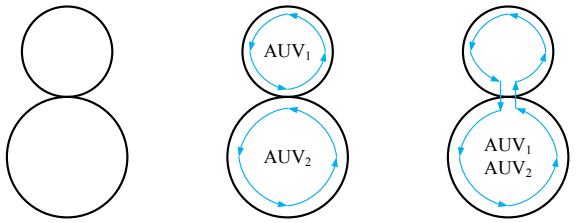
In: A given graph G;
Out: An extended Eulerian cycle;

1: Consider subset $V^{'}$ of all odd vertices in G;
2: Set the cost between pairs of vertices in $V^{'}$ as their shortest path distances in G;
3: Find a minimum weight perfect matching in $V^{'}$;
4: Construct a new weighted graph $G^{'}$ with vertex set $V^{'}$ and edge set of matching pairs;
5: Combine $G^{'}$ and G to obtain a new weighted graph $G^{''}$;
6: Return an Eulerian cycle in $G^{''}$ by applying Hierholzer's algorithm.

Let us consider a general graph G with odd-degree vertices (or simply odd vertices). Since the total degree of all vertices must be even (each edge is counted twice), there must exist an even number of odd vertices in G. We then pair odd vertices using *minimum weight perfect matching* [27] aiming to reduce added costs to paired odd vertices, where the cost of a pair (u, v) is the shortest path cost of u and v in G. Finally, we add a *virtual edge* between each matching pair to make all odd vertices even-degree vertices (or simply even vertices), leading to a new generated graph $G^{''}$. The linear Hierholzer's algorithm is then applied to derive the Eulerian cycle.

Note that the Eulerian cycle in $G^{''}$ is no longer an Eulerian cycle in G, as each virtual edge $G^{'}$ is mapped to a set of edges in G. Therefore, several edges will be visited more than once (i.e., it is no longer a tour, but a closed walk). Therefore, we call it an 'extended Eulerian cycle' for convenience. Then, the whole algorithm is described in Algorithm 1. An example is shown in Fig. 3. The given graph is shown in Fig. 3(a), while the corresponding odd-degree vertex matching is shown in Fig. 3(b). $G^{''}$ can be obtained through combining Figs. 3(a) and 3(b). The resultant Eulerian cycle is shown in Fig. 3(c).

To illustrate the reason for using only one large cycle instead of multiple small cycles to cover the search space, a motivational example is provided. Let us consider the scheduling of two AUVs for the search space of two neighboring cycles connected by one vertex, as shown in Fig. 4(a). Then, we have two scheduling policies as follows. Scheduling 1 assigns one AUV for each of the two neighboring cycles. The two AUVs operate independently, as shown in Fig. 4(b). Scheduling 2 considers the two neighboring cycles as one large cycle. The two AUVs operate cooperatively in the combined cycle, as shown in Fig. 4(c). Then, we have:



(a) Search space. (b) Scheduling 1. (c) Scheduling 2.

Fig. 4. Two scheduling policies for two neighboring cycles.

***Theorem** 2:* Scheduling 2 is no worse than Scheduling 1, in terms of the average data reporting delay.

Proof: Suppose the circumferences of the two neighboring cycles are C_1 and C_2, respectively. Then, their delays are $\frac{C_1}{2}+\sqrt{2LC_1}+L$ and $\frac{C_2}{2}+\sqrt{2LC_2}+L$, respectively. Their weighted average delay for Scheduling 1 is

$$\frac{C_1 \times (\frac{C_1}{2}+\sqrt{2LC_1}+L) + C_2 \times (\frac{C_2}{2}+\sqrt{2LC_2}+L)}{C_1+C_2} \quad (3)$$

For Scheduling 2, the circumference of the combined cycle is C_1+C_2. As shown in Eq. 2, its delay is

$$\frac{C_1+C_2}{4}+\sqrt{L(C_1+C_2)}+L \quad (4)$$

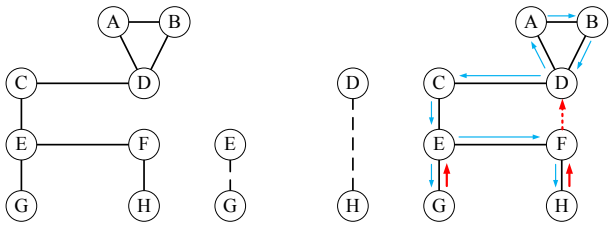
Note that we have $\frac{(C_1+C_2)^2}{4} \leq \frac{C_1^2+C_2^2}{2}$. It can also be proved that $C_1\sqrt{2LC_1}+C_2\sqrt{2LC_2} \geq (C_1+C_2)\sqrt{L(C_1+C_2)}$, or $\sqrt{2}C_1^{1.5}+\sqrt{2}C_2^{1.5} \geq (C_1+C_2)^{1.5}$. This is because derivations show that the function $\sqrt{2}+\sqrt{2}(\frac{C_2}{C_1})^{1.5}-(1+\frac{C_2}{C_1})^{1.5}$ is non-negative with respect to positive $\frac{C_2}{C_1}$. Therefore, the average data reporting delay in Eq. 3 is always no less than that in Eq. 4, meaning that Scheduling 2 is no worse than Scheduling 1. The key insight behind this theorem is that these two AUVs have balanced traversals in Scheduling 2, instead of unbalanced traversals in Scheduling 1. ■

Assuming that the given graph is connected, then Theorem 2 shows that independent schedules for several cycles with small circumferences are not better than a joint schedule that combines those small cycles to a larger one. Therefore, we favor the scheduling policy that constructs one extended Eulerian cycle for AUVs to traverse all the sensing edges, rather than scheduling policies that assign the AUVs to traverse small cycles independently. If the given graph is not connected, then Algorithm 1 would obtain multiple cycles, as shown in Fig. 2(d). This case will be further discussed in Section VII. In the next section, we will introduce non-sensing edges to further shorten the cycle circumference.

VI. Cycle Enhancement

A. Extended Cycles

In the previous section, we derive a small extended Eulerian cycle aiming at minimizing the circumference of the cycle. Such a cycle is a closed walk, in which each edge is visited at least once in the given graph. In this section, we will further shorten the cycle by visiting shorter non-sensing edges (edges not in G), instead of visiting redundant sensing edges (sensing edges that appear more than once in the cycle). As we recall



(a) Given graph. (b) Odd-degree vertex geometric matching. (c) Eulerian cycle.

Fig. 5. An example for Algorithm 2 with a cycle of ABDCEGEFHFDA.

Algorithm 2 Extended cycle with non-sensing edges

In: A given graph G;
Out: A cycle with all edges in G plus some links not in G;

Same as Algorithm 1, except the change of step 2: Set the cost between each pair of vertices in V' as their geometric distance.

that, a virtual edge is added between every two matching odd vertices. The cost of the virtual edge is the shortest path cost of these two vertices. Generally speaking, multiple appearances of an edge do not contribute to the reduction of the data reporting delay, since data is generated at a given rate. On the other hand, odd vertices can be connected via non-sensing edges with costs measured as geographic distances in straight lines.

Algorithm 2 describes such an extension of Algorithm 1, through replacing some multiple-visited sensing edges with geometrically-shortest-distance links that are not sensing edges in the graph. An example of Algorithm 2 is shown in Fig. 5. It further reduces the circumference of the cycle by using non-sensing edges. Moreover, we have the following theorem:

***Theorem** 3:* In the resultant cycle contructed by Algorithm 2, the total length of non-sensing edges is no larger than the total length of sensing edges.

Proof: We first show that no single edge (w, w') will appear in the shortest paths of two matching pairs $\{v, v'\}$ and $\{u, u'\}$. We prove this fact by contradiction. Suppose the shortest path from v to v' is $(v, ..., w, w', ..., v')$. Similarly, the shortest path from u to u' is $(u, ..., w, w', ..., v')$. Then, we will have two better matching pairs $\{v, u\}$ with paths $(v, ..., w, ..., u)$, and $\{v', u'\}$ with paths $(v', ..., w', ..., u')$. That is, edge (w, w') can be removed in the new pairings. This contradicts the goal of minimum cost perfect matching. Therefore, the total length of virtual edges generated from Algorithm 1 for G' is no larger than the total length of edges in G (i.e., the total length of sensing edges). Since Algorithm 2 is an enhancement of Algorithm 1 for matching in G', and not all virtual edges are non-sensing edges, Theorem 3 clearly holds. ■

In general, only a subset of virtual edges in Algorithm 1 are replaced by non-sensing edges in Algorithm 2. That is, some sensing edges still may appear twice in the resultant cycle as shown in Fig. 5 (e.g., sensing edges of GE and HF).

B. Shifting the Surface Point

In Section V, we have discussed the surfacing frequency for a cycle of sensing edges with a given number of AUVs.

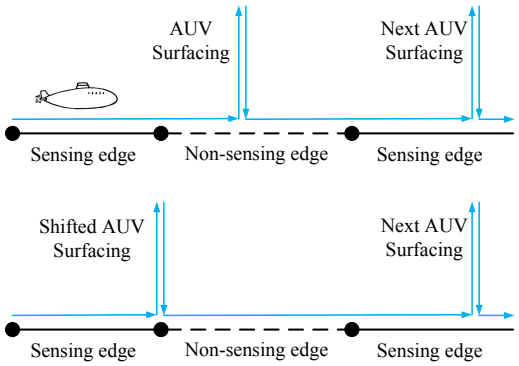


Fig. 6. An illustration of the shifting.

The cycles are constructed based on an extended Eulerian cycle through multiple visits of some sensing edges. Then, the methodology in Section IV can be used to determine the number and locations of resurfacing events in such cycles. However, in this section, we construct cycles with both sensing and non-sensing edges. It is meaningless for AUVs to resurface in the middle of non-sensing edges, since the data is only collected from sensing edges. If a schedule assigns an AUV to resurface at a non-sensing edge, then a better schedule can always be obtained through shifting that resurface time to an earlier time just when the AUV enters that non-sensing edge.

To better illustrate the idea of shifting, an example is shown in Fig. 6. If an AUV plans to resurface at a non-sensing edge, then this surface point is shifted to the end of the last sensing edge it traverses. Note that the current shifting will not result in a change of the next surface point. In Fig. 6, the first portion of the interval between AUV surfacing and next AUV surfacing belongs to a non-sensing edge, on which the AUV surfacing is shifted. This shifting scheme can always get a smaller delay, since it removes the unnecessary delay at a non-sensing edge, during which no data is collected. The shifting scheme can be viewed as a small adjustment to the surfacing location. However, can we totally remove the effect of non-sensing edges by adjusting both surfacing frequency and location? The next subsection gives a definite answer, but with a stringent constraint.

C. Exploring the Optimal Scheduling

In Section IV, we have explored the optimal AUV surfacing frequency for the search space of a cycle, which is composed of only sensing edges. Since Algorithm 2 constructs cycles with non-sensing edges, in this subsection, we re-explore the AUV surfacing frequency for such kinds of cycles. For simplicity, we only consider the scheduling with one AUV. Suppose the cycle is composed of $2m$ alternating sensing edges (denoted by S_i, with its length as C_i) and non-sensing edges (denoted by S'_i, with its length as C'_i). In other words, the cycle is C: S_1, S'_1, S_2, S'_2, ..., S_m, S'_m. Its circumference is $C = \sum_{i=1}^{m}(C_i + C'_i)$.

Here, we give out a new solution to determine the number and locations of resurfacing events for cycles with non-sensing edges. Let us remove all non-sensing edges from C to form a new cycle C^*: S_1, S_2, ..., S_m. Based on Theorem 1, we can calculate the optimal frequency and corresponding surface points within C^*, which can then be mapped back to the

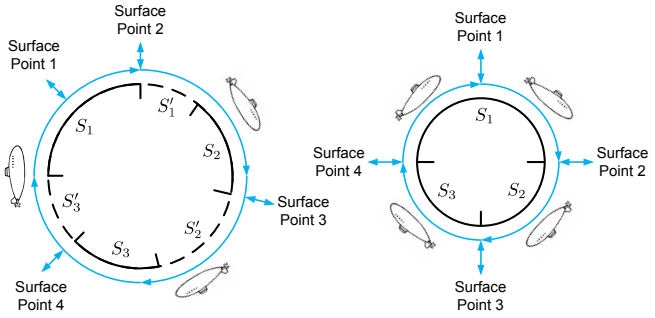
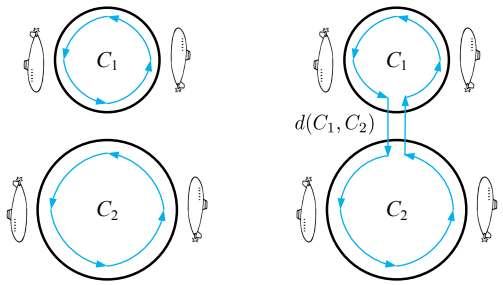


Fig. 7. An illustration for the algorithm optimality. The sensing edges are in the solid line, while the non-sensing edges are in the dotted line.



(a) Two cycles. (b) Merge result.

Fig. 8. An illustration for the cycle merge.

original cycle C as a solution. An example is shown in Fig. 7, where the original cycle C is in the left part and the new cycle C^* is in the right part. In C^*, we use the methodology stated in Section IV to calculate the surface points (the AUV resurfaces after traveling a distance of $\sqrt{2LC^*}$, based on Theorem 1). Four surface points are determined and then mapped back to the original cycle C as the final solution. In C^*, if the interval between adjacent surfacing points (with interval length of $\sqrt{2LC^*}$) never goes across two sensing edges in C^*, then this solution is optimal. In other words, the optimality prerequisite is that the length of each sensing edge should be an integer multiple of optimal interval length (i.e., $\sqrt{2LC^*}$). If an interval goes across two sensing edges in C^*, it will intersect a non-sensing edge in C, leading to a non-optimal result. This is because AUVs should not surface at a non-sensing edge (no data can be collected from the non-sensing edges).

Note that the optimality prerequisite for the above solution is very stringent and is not likely to be satisfied in real traces. If the length of S_i is not an integer multiple of optimal interval length, then the amount of resurfacing on S_i (calculated by $C_i/\sqrt{2LC^*}$) should be rounded off to the closest integer (except when it is less than one, then one should be used). For each sensing edge S_i, the surface points are equally distributed, so that all intervals within the sensing edge S_i have the same length and no interval goes across to a non-sensing edge. This scheme should work well, particularly when the length of each sensing edge is close to an integer multiple of $\sqrt{2LC^*}$ (near the optimality prerequisite).

VII. Cycle Merge

In Sections V and VI, we have discussed how to construct the cycles from the given graph of the search space. However, the given graph is not necessarily connected. Therefore, multiple cycles may be obtained, as shown in Figs. 2(d) and 2(e). In this section, we observe that cycles can be further merged with a cooperative AUV scheduling. As previously shown in Fig. 2(f), the two smaller cycles of ABDCA and EFHGE in Fig. 2(e) are merged, leading to a bigger cycle of ABDFHGECA. This kind of cycle merge can reduce the average data reporting delay [16], by balancing the AUV traversals in different cycles.

As shown in Fig. 8, suppose we have two cycles, C_1 and C_2. The distance between C_1 and C_2 is defined as the smallest distance between two points that are located in C_1 and C_2, respectively. Let $d(C_1, C_2)$ denote this distance. Suppose there are n_1 AUVs assigned to the cycle C_1, while there are

Algorithm 3 Cycle merge

In: The resulting cycles by Algorithm 2;
Out: The cycle merge result;

```
while there exists more than one cycles do
    for each pair of cycles do
        Calculate the merge benefit as the resulting value
        difference between Eqs. 5 and 6;
    if the largest merge benefit is positive then
        Merge that pair of cycles as a bigger cycle;
    else
        Break the while loop;
return The cycle merge result;
```

n_2 AUVs assigned to the cycle C_2. According to Eq. 2, the average data reporting delay for these two cycles is

$$\frac{C_1 \times (\frac{C_1}{2n_1}+\sqrt{\frac{2LC_1}{n_1}}+L) + C_2 \times (\frac{C_2}{2n_2}+\sqrt{\frac{2LC_2}{n_2}}+L)}{C_1 + C_2} \quad (5)$$

If C_1 and C_2 are merged, then we can obtain a bigger cycle with a circumference of $C_1 + C_2 + 2d(C_1, C_2)$. Meanwhile, $n_1 + n_2$ AUVs can be assigned to this merged cycle. The merged cycles include both sensing edges and non-sensing edges. If we use the shifting strategy in Section VI.B to schedule these AUVs, then the average data reporting delay for the merged cycle should be no larger than

$$\frac{C_1+C_2+2d(C_1,C_2)}{2(n_1+n_2)} + \sqrt{\frac{2L[C_1+C_2+2d(C_1,C_2)]}{(n_1+n_2)}} + L \quad (6)$$

If we compare the resulting values in Eqs. 5 and 6, then we can determine whether C_1 and C_2 should be merged or not. A more important insight behind the cycle merge is similar to that in Theorem 2. If the traversals of AUVs in C_1 and C_2 are more unbalanced, then we are more likely to merge C_1 and C_2, as to have more balanced AUV traversals. For example, if C_1 is large, C_2 is small, n_1 is small, and n_2 is large, then we should merge C_1 and C_2, if $d(C_1, C_2)$ is not too large. This is because C_2 has overmuch AUVs that can be re-balanced to collect the data from C_1, in which the AUVs are not sufficient. Following the above intuition, a greedy cycle merge method is proposed to further reduce the average data reporting delay, as shown in Algorithm 3. At each step, it greedily merges the pair of cycles that yields the largest merge benefit (the value difference between Eqs. 5 and 6). Note that the cycle merge only happens when the given graph is not connected.

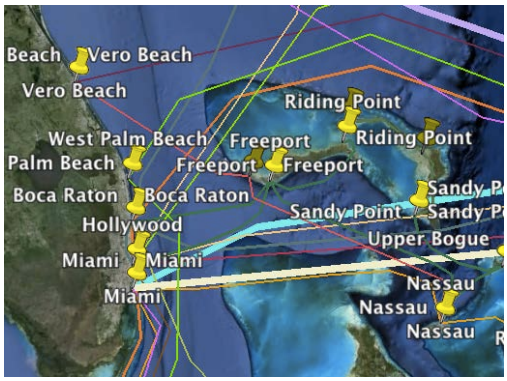


Fig. 9. A snapshot of the real trace (the oil pipe layout near Florida).

TABLE I. AVERAGE DATA REPORTING DELAY FOR THE SECOND KIND OF SYNTHETIC TRACE WITH ONE AUV.

The setting of the trace	The Algorithms	The given depth		
		L=10^1	L=10^2	L=10^3
The trace with an intersection angle of 10°	Algorithm 2	184	420	2304
	Algorithm 2s	178	403	2303
	Algorithm 2r	181	413	2311
The trace with an intersection angle of 30°	Algorithm 2	200	457	2369
	Algorithm 2s	188	406	2319
	Algorithm 2r	195	437	2371
The trace with an intersection angle of 50°	Algorithm 2	219	503	2357
	Algorithm 2s	204	420	2340
	Algorithm 2r	216	467	2434

VIII. EXPERIMENTS

In this section, experiments are conducted to evaluate the performances of the proposed algorithms. After presenting the datasets (two kinds of synthetic traces and one real trace) and basic settings, the evaluation results are shown from different perspectives to provide insightful conclusions.

A. Datasets and Basic Settings

In our experiments, several synthetic traces and one real trace are used to validate the efficiency and effectiveness of the proposed algorithms. The first type of synthetic trace is used to test the performance gap between Algorithms 1 and 2 (construct the cycle through shortest paths and that through geometrically-shortest-distance links). More specifically, we would like to verify the impact of the graph density, in terms of the average data reporting delay. Since we have adjusted the number and locations of surfacing events in Section VI for cycles with non-sensing edges, the second type of synthetic trace is introduced, as to validate the improvements of those adjustments. Finally, all the proposed algorithms are tested in a real trace (the oil pipe layout near Florida), as to verify their applicability in the real world.

The first type of synthetic trace is generated through a uniformly-random placement of 100 nodes on a 100×100 square unit. To guarantee the graph connectivity, a minimum spanning tree is constructed. Then, additional edges, with given total numbers of 20, 100, and 500, are introduced to uniform-randomly connect these nodes. Note that the given number of additional edges represents the graph density. Since this trace is randomly generated, experiments on this type of trace are repeated to determine the average, until the confidence interval of the average result is sufficiently small (1 percent for 90% probability). Then, the second type of synthetic trace includes 100 nodes and has a shape of V, which corresponds to the layout of the oil pipes in the search space. Each side of the V-trace has a length of 100 with 50 uniformly-distributed nodes. The intersection angle between the two sides of the V-trace is given as 10°, 30°, and 50°, respectively. Note that a smaller intersection angle brings a shorter geometrical distance between the two ends of the V-trace. For the synthetic traces, the speed of the AUV is one unit. The data generation speed is also set to be one unit, which is faster than the cycling time of the AUV. The given depth of the search space is set as 10, 100, and 1,000, respectively. The above parameter settings are used, since they can capture the properties (e.g., sensitivity to the graph density) of the proposed algorithms.

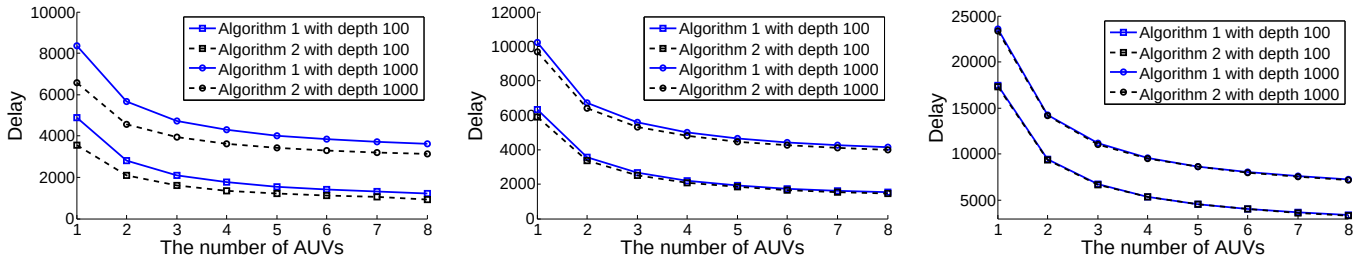
As for the real trace, we use the data published in [28]. In this real trace, we mainly focus on the oil pipe layout near Florida, including SAm-1, COLUMBUS I to III, Mid-Atlantic Crossing (MAC), BAHAMAS-1, BAHAMAS-2, GlobeNet, BDNSi, and so on. The total length of the oil pipes is 603km. A snapshot of this real trace is shown in Fig. 9. This area is selected for our experiments, since the corresponding oil pipe layout is very complex and representative. The sea depth is set to be 3,790 meters, which is the average sea depth in the real world [29]. Meanwhile, according to [30], the cruising speed of AUVs are set to be 37km/h, and the diving/surfacing speed of AUVs are set to be 26km/h. We assume that sensors are uniformly placed along each pipe, while the distance between two adjacent sensors on a pipe is 1km. Sensors are deployed to detect oil pipe leaks [14].

B. Comparison Algorithms and Metrics

In our experiments, we denote the shifting algorithm in Section VI.B as Algorithm 2s, and the approximated optimal algorithm with round-off in Section VI.C as Algorithm 2r. For Algorithms 1, 2, 2s, and 2r, if the given graph is not connected, then they would obtain multiple cycles. For this case, the number of AUVs distributed to each cycle is proportional to the cycle length. Algorithm 3 is also denoted as Cycle Merge. Finally, two additional baselines are used for comparison.

- Baseline 1 distributes AUVs evenly to the oil pipes. For each oil pipe, the corresponding AUVs uniformly go back and forth along that oil pipe. Baseline 1 can be regarded as an independent AUV scheduling method, where AUVs on different oil pipes do not cooperate with each other for the data collection.
- Baseline 2 distributes AUVs according to the lengths of the oil pipes. The number of AUVs assigned to a oil pipe is proportional to the length of that oil pipe. For each oil pipe, the corresponding AUVs also uniformly go back and forth along that pipe. Baseline 2 is an improvement of Baseline 1, since we should assign more AUVs to a longer oil pipe than a shorter one.

The data reporting delay serves as the performance metric in our experiments. We are interested in how the data reporting delay is impacted by the settings (e.g., the sea depth, the graph density, the percentage of non-sensing edges in the cycle, the number of AUVs, and so on).



(a) Given 20 additional edges. (b) Given 100 additional edges. (c) Given 500 additional edges.

Fig. 10. Average data reporting delay for the first kind of synthetic trace with multiple AUVs.

C. Experimental Results for the Synthetic Traces

The experimental results for the first kind of synthetic traces are shown in Fig. 10, where we study the relationship between the average data reporting delay and the number of AUVs. Different subfigures in Fig. 10 have different graph densities. The oil pipes in Fig. 10(a) are relatively sparse, while the oil pipes in Fig. 10(c) are relatively dense. The performance gap between Algorithm 1 and Algorithm 2 is significant in Fig. 10(a) that represents the results for the most sparse trace. However, the performance gap between these two algorithms decreases when the trace becomes denser, as shown in Figs. 10(b) and 10(c). The reason is because the gap of pairwising odd vertices through the shortest path and that through the geometrically-shortest-distance links becomes smaller as the trace gets denser. If the trace is sparse, pairwising odd vertices through the shortest path could be very costly, since the geometrical distances among these vertices could be much smaller. However, the real-world oil pipe layout is very sparse, as previously shown in Fig. 9. The experiments for the real trace (shown in the next subsection) demonstrate this point. Another observation is that a larger sea depth brings a larger delay. This is very intuitive, since AUVs need more time to resurface. Finally, the last observation is that the delay reduction brought by one more AUV decreases, with respect to the current number of AUVs (the effect of diminishing return). In Fig. 10(a), if the sea depth is $1,000$, the delay brought by Algorithm 1 with one AUV is about $8,000$. Meanwhile, if 8 AUVs are used, then the delay reduces to about $4,000$ (about 50% reduction). Generally speaking, a denser and larger trace needs more AUVs for a small average data reporting delay.

The experimental results for the second type of synthetic trace are shown in Table I, in terms of the average data reporting delay. Since we have adjusted the number and locations of surfacing events in Section VI for cycles with non-sensing edges, these traces are used to validate the improvements of those adjustments. As previously mentioned, the second type of synthetic trace has a shape of V. The intersection angles between the two sides of the V-trace are given as $10°$, $30°$, and $50°$, respectively. A smaller intersection angle means that the corresponding non-sensing edges are shorter (and thus the adjustment strategy should be less efficient). It can be seen that the shifting scheme is very effective, especially when the trace has a cycle of long non-sensing edges (i.e., the trace with an intersection angle of $50°$). On the other hand, if the total length of non-sensing edges is very small, then the performance improvement brought by the shifting scheme is limited. The delay reduction brought by the shifting scheme ranges from about 5% to 20%, compared to Algorithm 2. This is because a longer non-sensing edge means that AUVs are more likely to surface on that non-sensing edge, which should be adjusted by the shifting scheme. Meanwhile, Algorithm 2r has a limited delay reduction. Although Algorithm 2r could be optimal under a stringent constraint, that constraint is uncommon. Therefore, the shifting scheme is recommended for its simplicity and effectiveness.

D. Experimental Results for the Real Trace

The experimental results for the real trace are shown in Fig. 11. In this real trace, we use ten and twenty AUVs to collect the data, respectively. Baselines 1 and 2 have the worst performances. This is because they are independent AUV scheduling methods, where AUVs on different oil pipes do not cooperate with each other for the data collection. There is also a significant performance gap between Algorithms 1 and 2. As previously analyzed in the synthetic traces, this is because the real trace is sparse, leading to a large gap between pairwising odd vertices through the shortest path and that through the geometrically-shortest-distance links. Then, Algorithm 2s can further reduce the average data reporting delay of Algorithm 2 by about 5%. This is because AUVs should not resurface in the middle of non-sensing edges, since the data is only collected from sensing edges. Meanwhile, Algorithm 2r does not have a good performance. The optimality prerequisite of Algorithm 2r is very stringent and is not likely to be satisfied in real applications, leading to performance degradations. As for the Algorithm 3 (Cycle Merge), it brings a further reduction on the average data reporting delay. When we have 10 AUVs, Algorithm 3 has 5% less delay than Algorithm 2s. When we have 20 AUVs, Algorithm 3 has 10% less delay than Algorithm 2s. This is because more AUVs bring a better schedulability for the Cycle Merge. In summary, the proposed algorithms can obtain an acceptable data reporting delay in the real trace (the average data reporting delay is less than one hour).

IX. Conclusions

In this paper, we consider a data collection and event detection problem in the deep sea. The scenario is based on a search space that is a set of oil pipes deployed in the seabed. Sensors are deployed along the oil pipes for leak detection, while AUVs are used to collect the data from the sensors and then resurface to report the data. We focus on the scheduling of the AUV trajectory planning, as well as the

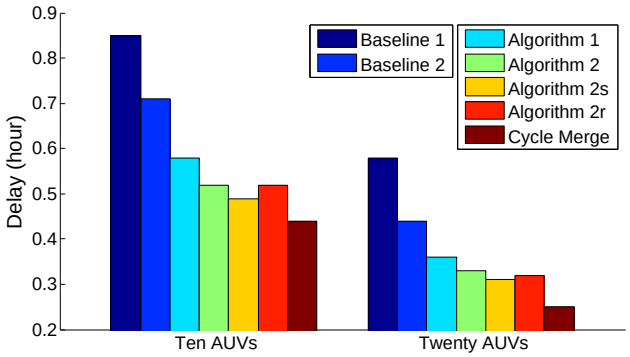


Fig. 11. The experimental results for the real trace.

AUV resurfacing frequencies and their locations. If the AUVs resurface frequently, then the uncollected data needs to wait a longer time to be reported, which leads to an increased average data reporting delay. On the other hand, if the AUVs resurface infrequently, then the collected data within the AUV needs to wait a longer time to be reported, which also leads to an increased average data reporting delay. According to the above tradeoff, an optimization problem is formulated by minimizing the average data reporting delay. Then, the AUV trajectory planning is simplified to an extended Euler cycle problem, where we construct cycles through both sensing edges and non-sensing edges. We have also discussed the cycle merge, where AUVs in different cycles can operate cooperatively. The cost-effectiveness of the proposed approach is validated in terms of both theoretical analysis and extensive experiments. As a part of future work, we will consider more general cycle merge algorithms instead of the simple greedy algorithm. The challenge lies in the decision of merging (or not merging) different cycles in the search space. The circumferences and locations of the cycles could be considered as their priorities for the cycle merge. We will also explore a better scheduling to determine the AUV resurfacing frequencies and locations, in cycles with both sensing and non-sensing edges.

REFERENCES

[1] J. Heidemann, M. Stojanovic, and M. Zorzi, "Underwater sensor networks: applications, advances and challenges," *Philosophical Transactions of the Royal Society A: Mathematical, Physical & Engineering Sciences*, vol. 370, no. 1958, pp. 158–175, 2012.

[2] N. Baccour, A. Koubâa, L. Mottola, M. A. Zúñiga, H. Youssef, C. A. Boano, and M. Alves, "Radio link quality estimation in wireless sensor networks: A survey," *ACM Transactions on Sensor Networks*, vol. 8, no. 4, pp. 34:1–34:33, 2012.

[3] http://www.washingtonpost.com/wp-dyn/content/article/2010/05/05/AR2010050505369.html.

[4] W. Zhao, M. Ammar, and E. Zegura, "A message ferrying approach for data delivery in sparse mobile ad hoc networks," in *Proceedings of ACM MobiHoc 2004*, pp. 187–198.

[5] ——, "Controlling the mobility of multiple data transport ferries in a delay-tolerant network," in *Proceedings of IEEE INFOCOM 2005*, pp. 1407–1418.

[6] I. F. Akyildiz, D. Pompili, and T. Melodia, "Underwater acoustic sensor networks: research challenges," *Ad hoc networks*, vol. 3, no. 3, pp. 257–279, 2005.

[7] V. Chandrasekhar, W. K. Seah, Y. S. Choo, and H. V. Ee, "Localization in underwater sensor networks: survey and challenges," in *Proceedings of ACM WUWNet 2006*, pp. 33–40.

[8] D. Pompili, T. Melodia, and I. F. Akyildiz, "Routing algorithms for delay-insensitive and delay-sensitive applications in underwater sensor networks," in *Proceedings of ACM MobiCom 2006*, pp. 298–309.

[9] J. Partan, J. Kurose, and B. N. Levine, "A survey of practical issues in underwater networks," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 11, no. 4, pp. 23–33, 2007.

[10] M. Ayaz, I. Baig, A. Abdullah, and I. Faye, "A survey on routing techniques in underwater wireless sensor networks," *Journal of Network and Computer Applications*, vol. 34, no. 6, pp. 1908–1927, 2011.

[11] H.-H. Cho, C.-Y. Chen, T. K. Shih, and H.-C. Chao, "Survey on underwater delay/disruption tolerant wireless sensor network routing," *IET Wireless Sensor Systems*, pp. 1–10, 2014.

[12] J.-H. Cui, J. Kong, M. Gerla, and S. Zhou, "The challenges of building mobile underwater wireless networks for aquatic applications," *IEEE Network*, vol. 20, no. 3, pp. 12–18, 2006.

[13] P. Xie, J.-H. Cui, and L. Lao, "VBF: Vector-based forwarding protocol for underwater sensor networks," in *Proceedings of IEEE NETWORKING 2006*, pp. 1216–1221.

[14] I. Jawhar, N. Mohamed, J. Al-Jaroodi, and S. Zhang, "An efficient framework for autonomous underwater vehicle extended sensor networks for pipeline monitoring," in *Proceedings of IEEE ROSE 2013*, pp. 124–129.

[15] M. Eichhorn, R. Taubert, C. Ament, M. Jacobi, and T. Pfuetzenreuter, "Modular auv system for sea water quality monitoring and management," in *Proceedings of MTS/IEEE OCEANS-Bergen 2013*, pp. 1–7.

[16] J. Wu and H. Zheng, "On efficient data collection and event detection with delay minimization in deep sea," in *Proceedings of ACM CHANTS 2014*, pp. 77–80.

[17] S. A. Kaiser, "Legal considerations about the missing malaysia airlines flight MH 370," *Air and Space Law*, vol. 39, no. 4, pp. 235–244, 2014.

[18] S. Jain, K. Fall, and R. Patra, "Routing in a delay tolerant network," in *Proceedings of ACM SIGCOMM 2004*, pp. 145–158.

[19] L. Tong, Q. Zhao, and S. Adireddy, "Sensor networks with mobile agents," in *Proceedings of IEEE MILCOM 2003*, pp. 688–693.

[20] M. M. Bin Tariq, M. Ammar, and E. Zegura, "Message ferry route design for sparse ad hoc networks with mobile nodes," in *Proceedings of ACM MobiHoc 2006*, pp. 37–48.

[21] R. Moazzez-Estanjini, J. Wang, and I. C. Paschalidis, "Scheduling mobile nodes for cooperative data transport in sensor networks," *IEEE/ACM Transactions on Networking*, vol. 21, no. 3, pp. 974–989, 2013.

[22] I. Jawhar, M. Ammar, S. Zhang, J. Wu, and N. Mohamed, "Ferry-based linear wireless sensor networks," in *Proceedings of IEEE GLOBECOM 2013*, pp. 304–309.

[23] M. Ma, Y. Yang, and M. Zhao, "Tour planning for mobile data-gathering mechanisms in wireless sensor networks," *IEEE Transactions on Vehicular Technology*, vol. 62, no. 4, pp. 1472–1483, 2013.

[24] X. Xue, X. Hou, B. Tang, and R. Bagai, "Data preservation in intermittently connected sensor networks with data priority," in *Proceedings of IEEE SECON 2013*, pp. 122–130.

[25] http://www.sonardyne.com/products/monitoring-a-control/autonomous-monitoring-system.html.

[26] H. Fleischner, "X.1 Algorithms for Eulerian Trails," *Eulerian Graphs and Related Topics: Part 1 (Annals of Discrete Mathematics)*, vol. 2, no. 50, pp. 1–13, 1991.

[27] V. Kolmogorov, "Blossom V: a new implementation of a minimum cost perfect matching algorithm," *Mathematical Programming Computation*, vol. 1, no. 1, pp. 43–67, 2009.

[28] http://www.cablemap.info/.

[29] http://en.wikipedia.org/wiki/Ocean.

[30] http://en.wikipedia.org/wiki/Ohio-class%5Fsubmarine.