On Efficient Data Collection and Event Detection with Delay Minimization in Deep Sea

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

ABSTRACT

Efficient data collection and event detection in the deep sea, as special applications of delay tolerant networks, pose some unique challenges due to the need for timeliness of data and event reporting of coverage areas and the delay of acoustic transmission in the ocean. Usually, autonomous underwater vehicles (AUVs) deployed in searching need to surface frequently to transmit collected data and events, as communications in the air can be done more quickly than communications under the water. However, extra delay is introduced at each resurfacing as AUVs are usually operated in the deep sea. In this paper, we attempt to optimize the frequency of surfacing with the objective of minimizing the average data and event reporting delay. We also study trajectory planning using an extended Euler circuit, where the search space is a set of connected line segments in the deep sea.

Categories and Subject Descriptors

G.1.6 [**Approximation**]: Constrained optimization;
G.2.2 [**Graph algorithms**]: Network problems.

Keywords

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

1. INTRODUCTION

Our Earth is mostly covered by ocean, which is largely underexplored. It is notoriously difficult to conduct an efficient search process in 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. Another motivational example can be the detection of oil pipe leak through robotic submarines in Gulf of Mexico [1].

In addition to the coverage of a vast area of the search space, communications and reporting of collected data (or events) in 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, acoustic transmission [11, 4] is most commonly used for underwater communication. However, it is well known that underwater acoustic communication speed is much slower than that in the air. Therefore, to report data in a search-and-rescue effort, autonomous underwater vehicles (AUVs) are used to surface frequently to transmit collected data (or events), as communication in the air can be done more quickly than communication under the water.

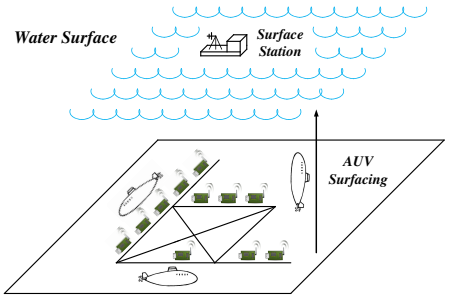


Figure 1: AUV resurfacing in deep sea.

In this paper, we consider a special scheduling problem aiming to minimize the average data delay. We assume that several 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. AUVs need to resurface to report collected data (or events) in a timely manner. Fig. 1 shows such a scenario. However, extra delay is introduced at each resurfacing. Given a search space that is a set of connected line segments (e.g., oil pipes), it is represented as a set of weighted edges in a graph. We propose a trajectory planning using an extended Euler circuit, and then, we determine the number and location of resurfacing on the circuit (or simply *cycle*). Specifically, we will consider the following problems in sequence: (1) Given the circumference of a cycle of search space at a given depth and the number of AUVs, we determine the number and locations of resurfacing that minimize the average data (or event detection) delay, assuming data (or events) are uniformly distributed. (2) We consider a more general case represented by a graph, where search space is the collection of all edges, called *sensing edges*. We then determine a cycle that covers all sensing edges, where some edges may appear more than once, as the given graph is not necessarily Eulerian. (3) Using the geometric property of 2-D space, we replace some multiple-visited edges with geometrically-shortest-distance links that are not edges in the graph, called *non-sensing edges*, to shorten the circum-

CHANTS'14, September 7, 2014, Maui, Hawaii, USA.

http://dx.doi.org/10.1145/2645672.2645686.

ference of the resultant cycle. (4) We conduct various simulations to critically compare the performances of various schemes using different traces.

The key difference between our approach and the traditional ferry approach [12, 8] is the AUV resurfacing process that brings an extra delay. Although a significant amount of work has been reported on underwater sensor networks [2] and their corresponding protocols [3, 5, 7, 10], to our best knowledge, this work is the first that combines the design of AUV resurfacing and AUV data collection (or event detection) in deep sea.

The reminder of the paper is organized as follows. Section 2 presents surfacing frequencies for AUVs. Section 3 focuses on finding a small cycle that covers the given search space. Section 4 discusses an extension of using non-sensing edges to shorten the cycle. Section 5 is the simulations. The paper concludes in Section 6 with our future work.

2. RESURFACING FREQUENCY

We start with a cycle of search space in a given depth with one AUV. We determine AUV resurfacing frequency that minimizes 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.

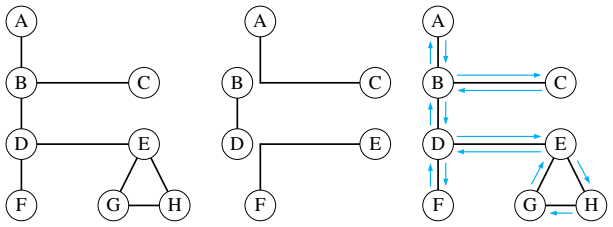
Let us consider the case of only one AUV, which has a unit speed. Let C denote the circumference of the cycle. L denotes the depth from the search space to the water surface. Let k denote the surfacing frequency per circulation of the cycle. The locations for surfacing are uniformly distributed along the cycle. We only consider a data generation rate that is 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 delay, from the time that data is generated to the time that 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 delay includes three parts as follows: (1) 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. Each data item needs to wait for a time of $\frac{C+2kL}{2}$ on average, before being transmitted to the AUV. (2) 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 $\frac{C}{2k}$. (3) Finally, the surfacing process takes a time of L. In total, the average data delay for one AUV, denoted by D_1, is

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

Eq. 1 is minimized to $\frac{C}{2} + \sqrt{2LC} + L$, when $k = \sqrt{\frac{C}{2L}}$. The above 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.



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

Figure 2: An example of Algrotithm 1 with an extended Eulerian cycle of ABCBDEHGEDFDBA.

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 [6].

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

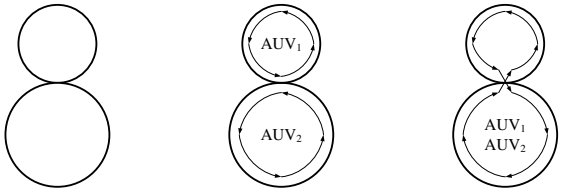
$$D_n = \frac{C + 2kL}{2n} + \frac{C}{2k} + L \tag{2}$$

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 an interval of $\frac{C}{k} = \sqrt{\frac{2LC}{n}}$.

3. CONSTRUCTION OF A CYCLE

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 connected line segments, represented by a weighted graph G, where each edge corresponds to a sensing edge. The cost associated with each edge is the length of the corresponding line segment.

In graph theory, an Eulerian trail is a trail in a graph 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. It is well known that Eulerian cycle exists if and only if the given graph has an even degree for every vertex. Given an Eulerian graph, we can construct such a cycle in a linear time proposed by Hierholzer [6]: 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, because the even degree of all vertices ensures that, when the trail enters another vertex



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

Figure 3: Two scheduling policies for two neighboring cycles with one common vertex.

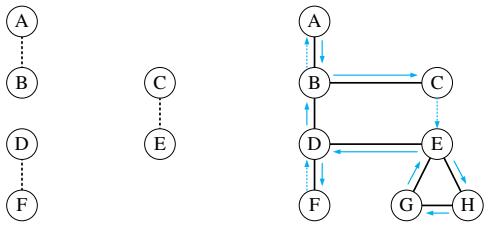
u there must be an unused edge leaving u. The tour (i.e., a closed trial) formed in this way is a closed tour, but may not cover all the vertices and edges of the initial graph. As long as there exists a vertex v that belongs to the current tour but that has adjacent edges not part of the tour, starting another trail from v, following unused edges until they return to v, and join the tour formed in this way to the previous tour.

Now, we consider a general graph G with odd-degree vertices (or simply odd vertices). It can be easily shown that there exists an even number of odd vertices in G. We then pair odd vertices using *minimum weight perfect matching* [9] 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 to even-degree vertices (or simply even vertices) to generate a new graph $G^{''}$. The linear Hierholzer's algorithm is then used to get 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 a closed walk rather than a tour). Therefore, we call it an extended Eulerian cycle.

The above algorithm is described in Algorithm 1. An example is shown in Fig. 2. In this example, there is only two even vertices G and H, which can be replaced by any subgraph consisting of even vertices only. To illustrate the reason of 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. 3(a). Then, we have two scheduling policies as follows: (1) Scheduling 1 assigns one AUV for each of the two neighboring cycles. The two AUVs operate independently, as shown in Fig. 3(b). (2) Scheduling 2 regards the two neighboring cycles as one large cycle. The two AUVs operate cooperatively in the combined cycle, as shown in Fig. 3(c). Then, we have the following theorem:

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

Assuming that the given graph is a connected graph, Theorem 2 shows that independent schedules for several cycles with small circumference are not better than a joint schedule that combines these small cycles to a larger one. This is because two AUVs should have balanced cycling tasks, instead of unbalanced cycling tasks. 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.



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

Figure 4: The Fig 2 example of Algorithm 2 with a cycle of ABCEHGEDFDBA.

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.

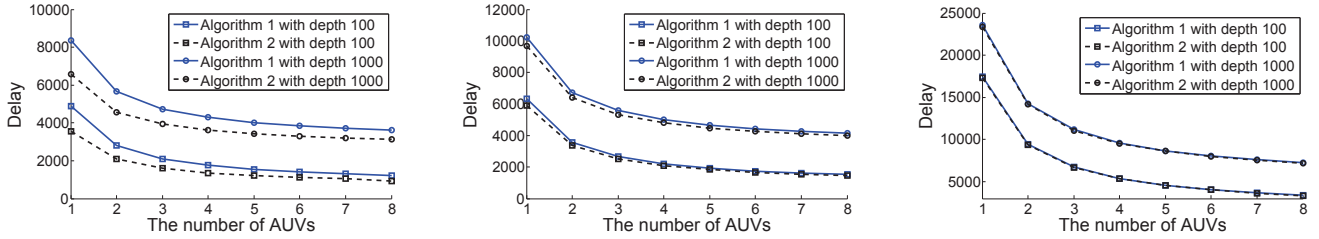
4. CYCLE ENHANCEMENT

In the previous section, we derive a small extended Eulerian cycle aiming to minimizing the circumference of the cycle. Basically, such a cycle is a closed walk in which each edge is visited at least once on any given weighted graph, In this section, we will further shorten the cycle by visiting shorter non-sensing edges, instead of visiting redundant edges. Redundant edges in a cycle are the ones that appear more than once. As we recall that, a virtual edge is added among every two matching odd vertices. The cost of the virtual edge is the shortest path cost of these two vertices. Usually, multiple appearances of an edge (path) does not contribute data collection delay reduction as data is generated at a given rate. On the other hand, odd vertices can be connected via non-sensing edges (i.e., edges not in G) with costs measured as geographic distance in straight lines.

Algorithm 2 describes such an extension of Algorithm 1, by replacing the virtual edge (i.e., the shortest path) of two vertices with geometric distance between them. An example of Algorithm 2 is shown in Fig. 4. Algorithm 2 further reduces the circumference of the cycle by using non-sensing edges. Moreover, we have the following theorem:

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

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 to 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). As Algorithm 2 is an enhancement of Algorithm 1 for matching in G' and not all virtual edges are non-sensing edges, Theorem 3



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

Figure 5: The simulation results for the first kind of synthetic traces with multiple AUVs.

clearly holds. Note that, 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. 4.

5. SIMULATIONS

In our simulations, synthetic traces are used to test the difference between Algorithms 1 and 2 (construct the cycle through shortest paths and through geometry paths). This trace is generated through uniform, 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 used to uniform-randomly connect these nodes. Note that the given number of edges represents the graph density. Since this trace is randomly generated, simulations 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). As previously mentioned, 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 period of the AUV. For both of the two traces, the given depth of the search space is set as 10, 100, and 1,000, respectively. Finally, the average data delay is used as the performance metric.

The simulation results on the average data delay reduction brought by using multiple AUVs are shown in Fig. 5. The performance gap (in terms of average data delay) between Algorithm 1 and Algorithm 2 is rather significant, especially in Fig. 5(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. 5(b) and 5(c). The reason is because the gap of pairwising odd vertices through the shortest path and that through the geometry link is becoming smaller, when the trace gets denser. Another important observation is that the delay reduction brought by one more AUV decreases, with respect to the current number of AUVs (i.e., the effect of diminishing return). Generally speaking, a denser and larger trace needs more AUVs.

6. CONCLUSIONS

We consider a data collection and event detection problem in deep sea: combining scheduling of AUV trajectory planning as well as AUV resurfacing frequencies and their locations. The deep sea trajectory planning is simplified to an extended Euler cycle problem. An optimization problem is formulated by minimizing the average data delay. The cost-effectiveness of the proposed approach is validated both in terms of theoretical analysis and extensive simulation. As a part of future work, we will consider a more general search space that is not a set of connected segments. The challenge lies in the decision of merging (or not merging) different connected components of the search space.

7. REFERENCES

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

[2] I. F. Akyildiz, D. Pompili, and T. Melodia. Underwater acoustic sensor networks: research challenges. *Ad hoc networks*, 3(3):257–279, 2005.

[3] 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*, 34(6):1908–1927, 2011.

[4] 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 Trans. Sen. Netw.*, 8(4):34:1–34:33, 2012.

[5] 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*, pages 1–10, 2014.

[6] H. Fleischner. X.1 Algorithms for Eulerian Trails. *Eulerian Graphs and Related Topics: Part 1 (Annals of Discrete Mathematics)*, 2(50):1–13, 1991.

[7] Z. Guo, Z. Peng, B. Wang, J.-H. Cui, and J. Wu. Adaptive routing in underwater delay tolerant sensor networks. In *Proc. of IEEE ChinaCom 2011*, pages 1044–1051.

[8] S. Jain, K. Fall, and R. Patra. Routing in a delay tolerant network. In *Proc. of ACM SIGCOMM 2004*, pages 145–158.

[9] V. Kolmogorov. Blossom V: a new implementation of a minimum cost perfect matching algorithm. *Math. Prog. Comp.*, 1(1):43–67, 2009.

[10] Y. Li and R. Bartos. A survey of protocols for intermittently connected delay-tolerant wireless sensor networks. *Journal of Network and Computer Applications*, 41(0):411–423, 2014.

[11] F. Yunus, S. H. S. Ariffin, and Y. Zahedi. A survey of existing medium access control (mac) for underwater wireless sensor network (uwsn). In *Proc. of AMS 2010*, pages 544–549.

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