On Efficient Data Collection Using Autonomous Underwater Vehicles

Jie Wu
Center for Networked Computing
Temple University
Road Map

1. Introduction
2. Optimal AUV Resurfacing
3. Constructing A Cycle
4. Cycle Enhancement
5. Extensions
6. Experiments
7. Conclusions and Future Work
1. Introduction

- Earth is mostly sea
  - > 70% of the surface

- Signal propagation
  - Electromagnetic signal decays quickly in the water
  - Acoustic signal has limited bandwidth and long delay
    - Speed: 10 kbps
    - Distance: 100 m
1. Introduction

- Efficient search in deep sea is notoriously difficult
  - The detection of oil pipe leak in Mexico
1. Introduction

Malaysia Airlines MH 370

- **DigitalGlobe**
  - Crowdsourcing volunteers comb satellite photos for Malaysia Airlines jet

- **March 11, 2014 (from CSU prof. email)**

I just saw on our local Denver Fox news (KDVR.com) that a local company, DigitalGlobe, has reoriented their satellites to take high-res images in the area where the plane may have crashed. Crowdsourcing efforts are on to have people scan these images and find signs of debris. I was reminded of Jie Wu’s talk earlier this month.
1. Introduction

- **Multi-tiered networks**
  - In the air
    - Unmanned Aerial Vehicle (UAV)
  - On the ground
  - Under the sea
    - Autonomous Underwater Vehicle (AUV)

- **Communication**
  - A2A (Air-to-Air), A2G, and G2A
  - G2G (Ground-to-Ground)
  - U2U (Underwater-to-Underwater), U2G, and G2U

UAVs and AUVs: **swarm intelligence**

J. Wu, “A Multi-tiered Network with Aerial and Ground Coverage,” Computer Communications, 40-years special issue, 2018
1. Introduction

- Surfacing of multiple **AUVs** to transmit collected data
  - Parallel 2-D search space (a set of connected line segments) to the water surface

- Examples:
  - Undersea tunnel
    - Depth: 75m~300m
  - Sensors on oil pipes
    - Depth: 200m~5,000m
  - Submarine cable
    - Depth: up to 8,000m
How to Solve It

- If you can’t solve a problem, then there is an easier problem you can solve: find it.

- Four principles
  - Understand the problem
  - Devise a plan
  - Carry out the plan
  - Look back

Polya
2. Optimal AUV Resurfacing

- AUV trajectory planning: minimizing the average delay

- How can we schedule AUVs to resurface optimally in a circular search space (Eulerian cycle)?

- How can we schedule multiple AUVs to resurface in general search space?

- How can we convert a search space to a circular search space?

- How can we merge the cycles to reduce the average delay?
2. Optimal AUV Resurfacing

- Data are uniformly distributed with a fixed generation rate.

- **Objective**: minimize the long-term average delay to the water surface.

- The speed of an AUV is unit:
  - $C$: the cycle circumference
  - $L$: the depth of the search space
  - $k$: the frequency of resurfacing

Unit speed through distance scaling (cruising speed: 37 km/h, diving/surfacing: 26 km/h, current: 5 km/h)
2. Optimal AUV Resurfacing

- A larger AUV resurfacing frequency can bring node A's data to the water surface more quickly.
- However, node A's data needs to wait for the next AUV for a longer time, since resurfacing takes additional time.

\[
D = \frac{C + 2kL}{2} + \frac{C}{2k} + L
\]

\[
D_{min} = \frac{c}{2} + \sqrt{2LC} + L \quad \text{when} \quad k = \frac{c}{\sqrt{2L}}
\]

\[
\frac{C}{k} = \sqrt{2LC}
\]
2. Optimal AUV Resurfacings

**Theorem 1**: Optimally, the AUV resurfaces after traveling a distance of $\sqrt{2LC}$ on the original cycle (if only one AUV is used).

- If we have multiple AUVs ($n$ AUVs)
  - Evenly distribute these AUVs on the cycle
  - Each AUVs resurfaces after traveling a distance of $\sqrt{\frac{2LC}{n}}$
3. Constructing A Cycle

- Why do we use only one large cycle instead of multiple small cycles to cover the search space?

- **Theorem 2**: Scheme 2 is no worse than Scheme 1, due to more balanced cycling tasks among AUVs.
3. Constructing A Cycle

- General search space: a set of connected line segments (called sensing edges in the graph)

- Graph with an even degree for every vertex
  - An Eulerian cycle exists (i.e., a cycle that visits each edge once and only once)

- Graph has vertices with odd degrees
  - Add redundant edges to make odd degree even
  - We need to minimize pairwise odd degree nodes by adding one link (There is an even number of vertices with odd degrees)
3. **Constructing A Cycle**

- **Algorithm 1**: construct an Eulerian cycle by adding sensing edges

---

Given graph | Odd-degree vertex matching | Combined graph | Hierholzer's algorithm
---|---|---|---
![Given graph](image1.png) | ![Odd-degree vertex matching](image2.png) | ![Combined graph](image3.png) | ![Hierholzer's algorithm](image4.png)

- Some sensing edges are visited for multiple times
4. Cycle Enhancement

- **Geometric shortest non-sensing edges** (which may not be in the search space) can shorten the cycle circumference, although no data is collected from them.

**Algorithm 2**: construct the cycle by adding non-sensing edges.

**Given graph** + **Odd-degree vertex matching** = **Combined graph**

**Hierholzer’s algorithm**
**Theorem 3**: In the enhanced cycle construction, the total length of the non-sensing edges is no larger than the total length of sensing edges.

- No single edge will appear in the shortest paths of two matching pairs using sensing edges.
- In the worst case, all the edges in the given graph are used once in pair matching.
- Moreover, non-sensing edges provide “short-cuts” for all pairs using sensing edges.
Algorithm 2s (cycles with non-sensing edges): shift the surface point from each non-sensing edge to the end of the last sensing edge (i.e., change resurfacing locations)
4. Cycle Enhancement

- **Algorithm 2r** (cycles with non-sensing edges): by removing non-sensing edges (and change both resurfacing frequency and locations)

  - Optimal when $C^*$ is the total circumference of sensing edges and the length of each sensing edge is an integer multiple of $\sqrt{2LC^*}$
5. Extensions

- **Greedy Cycle Merge**
  - Initialize each connected component in the search space as a cycle
  - Merge two cycles in each greedy iteration
  - Average data delay before merge
    \[
    \frac{C_1 \times \left( \frac{c_1}{2n_1} + \sqrt{\frac{2LC_1}{n_1}} + L \right) + C_2 \times \left( \frac{c_2}{2n_2} + \sqrt{\frac{2LC_2}{n_2}} + L \right)}{C_1 + C_2}
    \]
  - Estimated average data delay after merge
    \[
    \frac{C_1 + C_2 + 2d(C_1, C_2)}{2(n_1 + n_2)} + \sqrt{\frac{2L[C_1 + C_2 + 2d(C_1, C_2)]}{n_1 + n_2}} + L
    \]
5. Extensions

- **Two-way Merge Criterion**
  - **Algorithm 3**: largest average delay reduction
  - **Algorithm 4**: largest cycle circumference difference (with delay reduction threshold)
  - **Algorithm 5**: closest geographical distance (with delay reduction threshold)

- **Merge Termination**
  - When no merges are available
5. Extensions

- Three-way merge

(a) Three cycles.
(b) Two-way merge.
(c) Three-way merge.

- Parallel Cycle Merge Implementation
  - Parallelism by dividing the scenario into small regions
5. Extensions

- **Parallelism performance tradeoff**
  - 500m by 500m with 10 cycles (sparse) and 25 cycles (dense)
  - Circle circumference is randomly chosen from 40m, 60m, 80m
5. Extensions

- 3-D search space

  - Use average sea depth to estimate
6. Experiments

- **Settings**
  - The test is based on a synthetic trace
  - A 100*100 square unit with a depth 100
  - To guarantee the graph connectivity, a spanning tree is constructed
  - Additional edges, with given total numbers of 20 and 100, are added
  - AUV has unit speed
6. Experiments

Simulation results:

Algorithms 1 and 2: cycles with sensing edges and non-sensing edges
6. Experiments

Summary:

- A sparser graph leads to a larger gap between Algorithms 1 and 2
- The gap between Algorithms 1 and 2 is becoming smaller, when the trace gets denser
- The delay reduction brought by one additional AUV decreases (i.e., the effect of diminishing return)
7. Experiments

- 2-D and 3-D pseudo search space
- 10 AUVs
- Depth randomly chosen from 50 to 150
6. Experiments

- Results for 2-D and 3-D search spaces

<table>
<thead>
<tr>
<th>Trace setting</th>
<th>Comparison algorithms</th>
<th>The scenario</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>2-D scenario</td>
</tr>
<tr>
<td>100</td>
<td>Algorithm 1</td>
<td>1853s</td>
</tr>
<tr>
<td>sensing edges</td>
<td>Algorithms 2, 2s, 2r</td>
<td>1805s, 1724s, 1796s</td>
</tr>
<tr>
<td></td>
<td>Algorithms 3, 4, 5</td>
<td>1643s, 1704s, 1694s</td>
</tr>
<tr>
<td>500</td>
<td>Algorithm 1</td>
<td>4175s</td>
</tr>
<tr>
<td>sensing edges</td>
<td>Algorithms 2, 2s, 2r</td>
<td>4104s, 4053s, 4089s</td>
</tr>
<tr>
<td></td>
<td>Algorithms 3, 4, 5</td>
<td>3861s, 3978s, 3942s</td>
</tr>
</tbody>
</table>

- **Algorithms 1 and 2**: cycles with sensing edges and non-sensing edges
- **Algorithms 2s and 2r**: adjust the surface point at the end of sensing edges, and round off the length of sensing edges in **Algorithm 2**
- **Algorithms 3, 4, and 5**: cycle merges with three merging criteria: largest delay reduction, largest cycle circumference difference, closest geographical distance
6. Experiments

- Real data-driven experiments
  - Oil pipes in Florida, Taiwan, and Japan
  - Sea depth 3,790m (average depth over the world)
  - AUV cruising speed 37km/h
  - AUV diving/surfacing speed 26km/h

(a) Florida oil pipes.  (b) Taiwan oil pipes.  (c) Japan oil pipes.
6. Experiments

- Results for Florida (in hours)

<table>
<thead>
<tr>
<th>Trace setting</th>
<th>Comparison algorithms</th>
<th>The number of AUVs</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>10 AUVs</td>
</tr>
<tr>
<td>Florida oil pipe trace</td>
<td>Algorithm 1</td>
<td>0.59h</td>
</tr>
<tr>
<td>Florida oil pipe trace</td>
<td>Algorithms 2, 2s, 2r</td>
<td>0.52h, 0.49h, 0.52h</td>
</tr>
<tr>
<td>Florida oil pipe trace</td>
<td>Algorithms 3, 4, 5</td>
<td>0.45h, 0.43h, 0.40h</td>
</tr>
</tbody>
</table>

- **Algorithm 1**: cycle with only sensing edges
- **Algorithms 2, 2s, and 2r**: cycle with non-sensing edges, adjust resurface points, and round off sensing edge schedules
- **Algorithms 3, 4, and 5**: cycle merges with three merging criteria: largest delay reduction, largest cycle circumference difference, closest geographical distance
6. Experiments

- Results for Taiwan (in hours)

<table>
<thead>
<tr>
<th>Trace setting</th>
<th>Comparison algorithms</th>
<th>The number of AUVs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Taiwan oil pipe trace</td>
<td>Algorithm 1</td>
<td>7.87h, 7.51h</td>
</tr>
<tr>
<td></td>
<td>Algorithms 2, 2s, 2r</td>
<td>7.49h, 7.26h, 7.29h</td>
</tr>
<tr>
<td></td>
<td>Algorithms 3, 4, 5</td>
<td>6.76h, 6.95h, 6.86h</td>
</tr>
</tbody>
</table>

- Results for Japan (in hours)

<table>
<thead>
<tr>
<th>Trace setting</th>
<th>Comparison algorithms</th>
<th>The number of AUVs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Japan oil pipe trace</td>
<td>Algorithm 1</td>
<td>9.84h, 8.92h</td>
</tr>
<tr>
<td></td>
<td>Algorithms 2, 2s, 2r</td>
<td>8.65h, 8.22h, 8.17h</td>
</tr>
<tr>
<td></td>
<td>Algorithms 3, 4, 5</td>
<td>7.75h, 7.56h, 7.81h</td>
</tr>
</tbody>
</table>
6. Experiments

Summary:

- There is a significant performance gap between Algorithms 1 and 2, since the real trace is sparse.

- **Algorithm 2s** can reduce the average data reporting delay of **Algorithm 2** by about 5% (AUVs should not resurface at non-sensing edges).

- **Algorithm 2r** may not outperform **Algorithm 2s**.

- **Algorithms 3, 4, and 5** have different performances, depending on the trace.
7. Conclusions and Future Work

- The **AUV trajectory planning** determines the AUV resurfacing frequencies and their locations.
- The deep sea trajectory planning is simplified to an extended Euler cycle problem.
- **Future Work**
  - More sophisticated AUV routing & resurfacing policies
  - More real data-driven experiments in 3-D search
  - Extension to the notion of age of information
  - Overall architectural design for multi-tired networks
Questions
