Bicriteria trajectory planning in demining operations

Michael Morin\textsuperscript{1}  
Armand Fruitet\textsuperscript{2}  
Irène Abi-Zeid\textsuperscript{3}  
Claude-Guy Quimper\textsuperscript{1}

\textsuperscript{1}Dep. of Computer Science and Software Engineering, Université Laval, Québec, Canada  
\textsuperscript{2}Dép. SES, Université Toulouse le Mirail, Toulouse, France  
\textsuperscript{3}Dep. of Operations and Decision Systems, Université Laval, Québec, Canada

✉ Michael.Morin.3@ulaval.ca

This work has been possible due to the collaboration with Yvan R. Petillot from the Ocean Systems Lab, Edinburgh, Scotland, UK

May 28, 2014
Introduction

Coverage?

- **Coverage path planning (CPP) problem** [Choset, 2001]
  - Plan an agent’s *path* to guarantee *complete coverage* of the environment [Mannadiar and Rekleitis, 2010]

- **Objectives**
  - Minimize the traveled distance
  - Minimize the number of turns

- **Off-line and sensor-based** approaches

- Usual order of priority of the objectives: distance then turns
Introduction

Context

- Underwater minesweeping operations using robots

A naval mine [Lippsett, 2004]

A robot underwater [Linder, 2006]
Introduction

Context

- The robots: REMUS\textsuperscript{1} [Nicholson and Healey, 2008, Fang and Anstee, 2010]
  - Swim long distances
  - Constant speed and altitude
  - Infrequent turns

A REMUS 100 swimming in Western Greenland [Kukulya, 2012]

\textsuperscript{1} Remote Environmental Monitoring UnitS
Introduction

Context

- Sensors
  - Sidescan sonar
  - “Gap filler” forward looking sonar

A side-scan sonar [Wikipedia, 2013]
Coverage path planning with imperfect extended detection

Specific formalism

- **Coverage path planning problem with imperfect extended detection (CPPIED)** [Drabovich, 2008]
  - *Imperfect* sensors: (conditional) detection probability [Gage, 1995]
  - *Minimal required coverage* instead of complete coverage
  - *Extended detection range*

- **Objectives**
  - Minimize the traveled distance
  - Minimize the number of turns

- **Off-line path planning**

- **Grids of square cells**
The different seabed type of a cell influences the sensor’s performance.
Coverage path planning with imperfect extended detection

Sensors scans

Scans with a range of $r = 3$ cells; darker cells have a higher scan frequency.
**Definition**

A *feasible path* achieves the *minimal required coverage* in each cell of matrix $D$.

![Matrix Table]

Minimal required coverage on a seabed grid
Definition

The *conditional detection probability* of a scan in a given cell is a function of

- the seabed type, and
- the distance (and the maximal lateral range $r$).
Coverage path planning with imperfect extended detection

Conditional detection probability

The distance between the robot’s position and the scanned cell influences the sensor’s performance.

\[ p^{\text{scan}} = \begin{bmatrix}
p^{\text{scan}}(1, c) & p^{\text{scan}}(2, c) & p^{\text{scan}}(3, c) \\
p^{\text{scan}}(1, r) & p^{\text{scan}}(2, r) & p^{\text{scan}}(3, r) \\
p^{\text{scan}}(1, f) & p^{\text{scan}}(2, f) & p^{\text{scan}}(3, f) \\
\end{bmatrix} = \begin{bmatrix}
0.7 & 0.3 & 0.1 \\
0.8 & 0.4 & 0.2 \\
0.9 & 0.6 & 0.3 \\
\end{bmatrix} \]
Let $C$ be the *current coverage* matrix.$^a$

$^a$The initial coverage is null.
Coverage path planning with imperfect extended detection

**Definition**

'C' is the *updated coverage* matrix after one or more scans, i.e., $C'_{ij}$ is the (bayesian) updated coverage in cell $(i, j)$ after a scan in $(i, j)$:

$$C'_{ij} := C_{ij} + (1 - C_{ij}) \times p^{\text{scan}}(d(x', y', i, j), O_{ij}).$$

$$p^{\text{scan}} = \begin{bmatrix} p^{\text{scan}}(1, c) \\ p^{\text{scan}}(1, r) \\ p^{\text{scan}}(1, f) \end{bmatrix} = \begin{bmatrix} 0.7 \\ 0.8 \\ 0.99 \end{bmatrix}$$

$$\approx 1$$
The following cases may occur in our bayesian update:

- a mine may have already been detected in \((i,j)\) with a probability \(C_{ij}\);
- or
- it has not been detected before with a probability \(1 - C_{ij}\), and
  - it will be detected now with a conditional probability 
    \[ p^{\text{scan}}(d(x,y,i,j),O_{ij}). \]

The updated coverage of a cell \((i,j)\), \(C'_{ij}\), is defined as:

\[
\begin{align*}
\text{Previously detected} & \quad + \quad \text{No mine detected yet} \\
C_{ij} & \quad \text{or} \quad (1 - C_{ij}) \\
\text{A mine is detected now} & \quad \times \quad p^{\text{scan}}(d(x',y',i,j),O_{ij}),
\end{align*}
\]

where cell \((x',y')\) is the current robot position.
The following cases may occur in our Bayesian update:
- a mine may have already been detected in \((i, j)\) with a probability \(C_{ij}\); or
- it has not been detected before with a probability \(1 - C_{ij}\), and
  - it will be detected now with a conditional probability \(p^{\text{scan}}(d(x, y, i, j), O_{ij})\).

The updated coverage of a cell \((i, j)\), \(C'_{ij}\), is defined as:

\[
\hat{C}_{ij} + (1 - C_{ij}) \times p^{\text{scan}}(d(x', y', i, j), O_{ij}),
\]

where cell \((x', y')\) is the current robot position.
The following cases may occur in our bayesian update:
- a mine may have already been detected in \((i,j)\) with a probability \(C_{ij}\); or
- it has not been detected before with a probability \(1 - C_{ij}\), and
  - it will be detected now with a conditional probability \(p_{\text{scan}}(d(x,y,i,j),O_{ij})\).

The updated coverage of a cell \((i,j)\), \(C'_{ij}\), is defined as:

\[
\begin{align*}
\text{Previously detected} & \quad \text{No mine detected yet} & \quad \text{A mine is detected now} \\
C_{ij} & + & (1 - C_{ij}) & \times & p_{\text{scan}}(d(x',y',i,j),O_{ij}),
\end{align*}
\]

where cell \((x',y')\) is the current robot position.
The following cases may occur in our bayesian update:
- a mine may have already been detected in \((i,j)\) with a probability \(C_{ij}\); or
- it has not been detected before with a probability \(1 - C_{ij}\), and it will be detected now with a conditional probability \(p_{\text{scan}}(d(x, y, i, j), O_{ij})\).

The updated coverage of a cell \((i,j)\), \(C'_{ij}\), is defined as:

\[
\widehat{C_{ij}} + (1 - C_{ij}) \times p_{\text{scan}}(d(x', y', i, j), O_{ij}),
\]

where cell \((x', y')\) is the current robot position.
The following cases may occur in our bayesian update:
- a mine may have already been detected in \((i, j)\) with a probability \(C_{ij}\); or
- it has not been detected before with a probability \(1 - C_{ij}\), and it will be detected now with a conditional probability \(p^{\text{scan}}(d(x, y, i, j), O_{ij})\).

The updated coverage of a cell \((i, j)\), \(C'_{ij}\), is defined as:

\[
C_{ij} + (1 - C_{ij}) \times p^{\text{scan}}(d(x', y', i, j), O_{ij}),
\]

where cell \((x', y')\) is the current robot position.
A CPPIED problem instance consists of:

- a seabed type set $\mathcal{T}$;
- a seabed matrix $\mathbf{O}$;
- a required coverage matrix $\mathbf{D}$;
- a conditional detection function $p_{\text{scan}}$; and
- an initial position $(i_{\text{init}}, j_{\text{init}})$ (optional).

$p_{\text{scan}} = \begin{bmatrix} p_{\text{scan}}(1, c) \\ p_{\text{scan}}(1, r) \\ p_{\text{scan}}(1, f) \end{bmatrix} = \begin{bmatrix} 0.7 \\ 0.8 \\ 0.99 \end{bmatrix}$
Coverage path planning with imperfect extended detection

Efficient coverage path

Definition

An efficient path achieves the required coverage and minimizes
- the length of the path (distance, robot’s moves, etc.); and
- the number of turns.

\[
p_{\text{scan}} = \begin{bmatrix}
    p_{\text{scan}}(1, c) \\
    p_{\text{scan}}(1, r) \\
    p_{\text{scan}}(1, f)
\end{bmatrix} = \begin{bmatrix}
    0.7 \\
    0.8 \\
    0.99
\end{bmatrix}
\]
More than 21,000 cells of a complex seabed topology

- Path planning on a cell by cell basis is not realistic due to the intractability of the problem...
More than 21,000 cells of a complex seabed topology

... clever algorithms are required.
The problem was first a lexicographic optimization problem:

Heterogeneous Coverage Path Planning algorithm [Drabovich, 2008]
The HCPP algorithm [Drabovich, 2008]

Parameterized Heuristic

- HCPP is a parameterized heuristic algorithm
- Construct the path segment by segment.
- The segment length is entirely determined by the algorithms parameters.
The HCPP algorithm [Drabovich, 2008]
Parameterized Heuristic

The HCPP algorithm does the following:

1. Configure the parameters on a given instance or set of instances.
   - The parameters that lead to the shortest path with the lowest number of turns are retained.

2. Using the parameters from the configuration phase:
   - Generate one path segment.
   - Link the segment to the current path.
   - If the required coverage is not achieved, iterate.
We started to explore the problem in its bicriteria form

Planification de chemins de couverture avec capteurs imparfaits en environnements hétérogènes

Armand FRUITET
The HCPP algorithm [Drabovich, 2008]
The HCPP algorithm [Drabovich, 2008]
The HCPP algorithm [Drabovich, 2008]

Parameterized Heuristic

The HCPP algorithm does the following:

1. First configure the parameters on a given instance or set of instances.
   - The parameters that lead to the shortest path with the lowest number of turns are retained.

2. Using the parameters from the configuration phase:
   - Generate one path segment.
   - Link the segment to the current path.
   - If the required coverage is not achieved, iterate.
The HCPP algorithm does the following:

1. First configure the parameters on a given instance or set of instances.
   - The parameters that lead to the shortest path with the lowest number of turns are retained.
   - *We may as well retain the parameters which minimize the number of turns and then the length.*

2. Using the parameters from the configuration phase:
   - Generate one path segment.
   - Link the segment to the current path.
   - If the required coverage is not achieved, iterate.
The HCPP algorithm [Drabovich, 2008]
Parameterized Heuristic

The HCPP algorithm does the following:

1. First configure the parameters on a given instance or set of instances.
   - The parameters that lead to the shortest path with the lowest number of turns are retained.
   - We may as well retain the parameters which minimize the number of turns and then the length.

2. Using the parameters from the configuration phase:
   - Generate one path segment.
   - Link the segment to the current path.
   - If the required coverage is not achieved, iterate.

Alternatively we may retain all the solutions and approximate the Pareto front.
The DpSweeper algorithm [Morin et al., 2013]

- Dynamic Programming Sweeper algorithm [Morin et al., 2013]

---

A Hybrid Algorithm for Coverage Path Planning With Imperfect Sensors

Michael Morin\textsuperscript{1, \textcopyright}, Irène Abi-Zeid\textsuperscript{2}, Yvan Petillot\textsuperscript{3}, Claude-Guy Quimper\textsuperscript{1}

Abstract—We are interested in the coverage path planning problem with imperfect sensors, within the context of robotics for mine countermeasures. In the studied problem, an autonomous underwater vehicle (AUV) equipped with sonar surveys the bottom of the ocean searching for mines. We use a cellular decomposition to represent the ocean floor by a grid of uniform square cells. The robot scans a fixed number of cells sideways with a varying probability of detection as a function of distance and of seabed type. The goal is to plan a path that achieves the minimal required coverage in each cell while given minimal coverage (minimal conditional probability of detection) must be achieved over the whole area of interest for the path to be feasible.

CPP problems are often solved in order to plan an agent’s (or multiple agents) path in such a way to guarantee complete coverage [6], or in the case of imperfect sensors, a minimal required coverage [7]. In complete CPP problems with perfect sensors, a cell is fully covered after a single scan and no further visits are needed. Complete CPP problems...
The DpSweeper algorithm [Morin et al., 2013]

Hybrid?

- Dynamic programming
- Traveling salesman problem reduction
- External solver: Concorde solver [Applegate et al., 2011]
- … The Dynamic Programming Sweeper (DpSweeper) algorithm…
DpSweeper does the following:

1. It greedily constructs a partial path made of disconnected segments in set $S$ to guarantee that the required coverage is achieved (dynamic programming).

2. It optimally links segments to create a path that is within the robot’s physical constraints ($TSP$ reduction).
The DpSweeper algorithm [Morin et al., 2013]

Part 1: Greedily Choosing a Set of Segments $S$

**Definition**

A *segment* is defined as a set of adjacent cells without any turn.

**Definition**

Two segments are *independent* if their scans do not overlap.
The DpSweeper algorithm [Morin et al., 2013]
Part 1: Greedily Choosing a Set of Segments $S$

- The **good**:
  - Computing the optimal segment length for independent segments is done in polynomial-time using dynamic programming.$^2$

---

$^2$Using Kadane’s algorithm [Bentley, 1984] for computing a segment’s length.
The DpSweeper algorithm [Morin et al., 2013]
Part 1: Greedily Choosing a Set of Segments $S$

- **The good:**
  - Computing the optimal segment length for independent segments is done in polynomial-time using dynamic programming.$^2$

- **The bad:**
  - Independent segments do not allow for minimal required coverage in many instances.
  - That is, overlapping segments are required to solve many instances.

---

$^2$Using Kadane’s algorithm [Bentley, 1984] for computing a segment’s length.
The DpSweeper algorithm [Morin et al., 2013]

Part 1: Greedily Choosing a Set of Segments $S$

- The good:
  - Computing the optimal segment length for independent segments is done in polynomial-time using dynamic programming.\(^2\)
- The bad:
  - Independent segments do not allow for minimal required coverage in many instances.
  - That is, overlapping segments are required to solve many instances.
- The pretty:
  - When two detections (scans) overlap their order is not important:

\[
C'_{ij} := C_{ij} + (1 - C_{ij}) \times p^{\text{scan}}(d(x', y', i, j), O_{ij}).
\]

\(^2\)Using Kadane’s algorithm [Bentley, 1984] for computing a segment’s length.
The DpSweeper algorithm [Morin et al., 2013]
Part 1: Greedily Choosing a Set of Segments $S$

Suppose that the lateral range is $r = 1$.

Each cell contains the number of scans left which depends on:

- the seabed types;
- the required coverage matrix; and
- the conditional probability of detection.
Suppose that the lateral range is $r = 1$.

Each cell contains the number of scans left which depends on:

- the seabed types;
- the required coverage matrix; and
- the conditional probability of detection.
Suppose that the lateral range is $r = 1$.

Each cell contains the number of scans left which depends on:

- the seabed types;
- the required coverage matrix; and
- the conditional probability of detection.
Suppose that the lateral range is $r = 1$.

Each cell contains the number of scans left which depends on:

- the seabed types;
- the required coverage matrix; and
- the conditional probability of detection.
Once $S$ has been determined, we need to compute an optimal path crossing each segment of $S$ once.
First, we forget about the map...

... to think about an optimal TSP tour.
Each segment is modeled as 3 vertices in the TSP.

The cost of the edges of a segment is *null*. 
We link the endpoints of each pair of segments.

The cost of the edges between the ends of two different segments is in number of moves (dotted edges).
We obtain the optimal TSP tour using Concorde solver\textsuperscript{3}.

\textsuperscript{3}[Applegate et al., 2011]
We reconstruct the path.
The DpSweeper algorithm [Morin et al., 2013]

Part 2: Reducing to a TSP and Reconstructing the Path $P$

DpSweeper plans the optimal tour and reconstructs the path

- Reduce to a TSP and obtain an optimal tour:
  - minimize the length then the number of turns;
  - or minimize the number of turns then the length.
- Reconstruct the path from the optimal tour.
The DpSweeper algorithm [Morin et al., 2013]

![Diagram showing a scatter plot with HCPP solutions and HCPP Pareto points.](image)
The DpSweeper algorithm [Morin et al., 2013]
Is the DpSweeper algorithm a flawless heuristic on all instances?
No, but it is especially effective in terms of path length.
Wrap Up

- HCPP:
  - Lengthy configuration (from hours to days)
  - Per instance (or instance set) configuration required
  - Fast for predetermined parameters 😊

- DpSweeper:
  - No parameter 😊
  - Fast even in presence of a TSP (running time under a minute)
  - High performance in terms of path length 😊
Conclusion

**Contributions:**
- Coverage path planning with imperfect extended detection formalism
- The extension of the HCPP algorithm to a bicriteria case
- The DpSweeper algorithm and its evaluation in a bicriteria context
Thank you for your attention.

Concorde TSP solver.
http://www.tsp.gatech.edu/concorde.
Accessed: 2013/03/15.

Programming pearls: algorithm design techniques.
Communications of the ACM, 27(9):865–873.

Coverage for robotics.

Automated mission trajectory planning for mine countermeasures operations.
Master’s thesis, Heriot Watt University, Edinburgh, Scotland, UK.

Coverage path planning for harbour seabed surveys using an autonomous underwater vehicle.
In Proc. of OCEANS 2010 IEEE-Sydney, pages 1–8, Sydney, Australia.


Side-scan sonar.