Computing the best coverage path in the presence of obstacles

Senjuti Basu Roy, Gautam Das, and Sajal Das

Outline

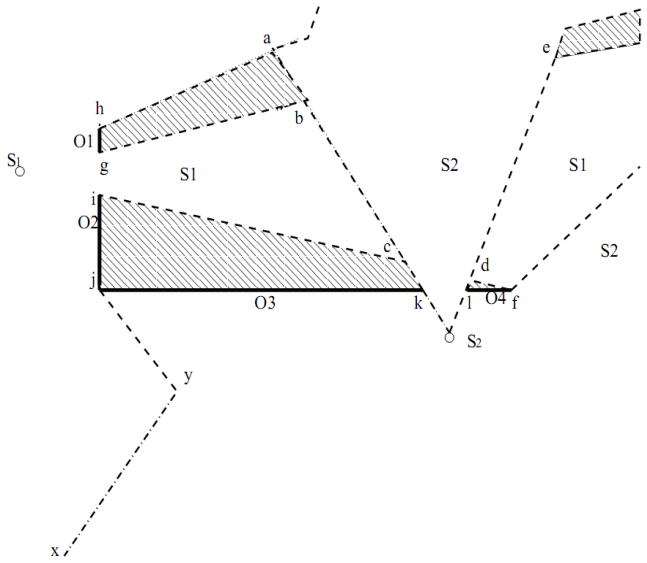
1. Problem formulation
2. Problem for opaque obstacles
3. Problem for transparent obstacles
4. Conclusion

Problem formulation

- The cover value of a path from s to t is the maximum distance from a point of the path the its closest sensor.
- Best coverage path from s to t, BCP(s, t), is the path that has minimum cover value.
- Model: Set S of n sensors and set O of m line segment obstacles.
- Two types of obstacles:
  - Opaque obstacles: obstruct paths and the line of sight of sensors
  - Transparent obstacles: obstruct paths but allow sensors to see through them.

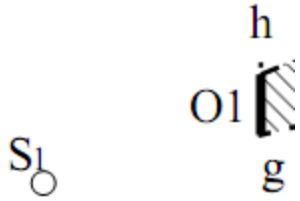
Problem for opaque obstacles

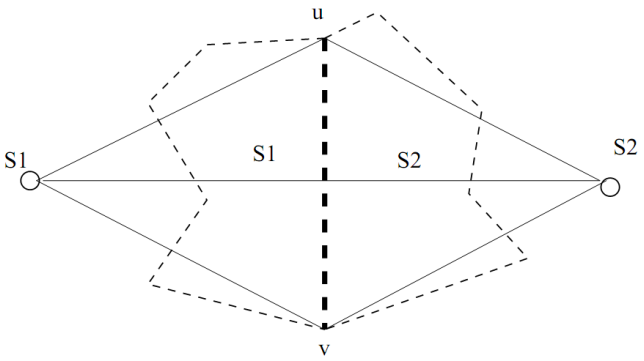
- Constrained weighted Voronoi diagram is a set of Voronoi cells $\{V(s_i) \mid s_i \in S\}$ such that $V(s_i) = \{x \in \mathbb{R}^2 \mid d_{wo}(x, s_i) \leqslant d_{wo}(x, s_j)$ and $d_{wo}(x, s_i) \neq \infty$ for all $s_j \in S, s_i \neq s_j\}$
- Observation: Each path go through a set of cells and intersects with them at the cell boundaries. The problem is solved if we can find the set of these intersections.

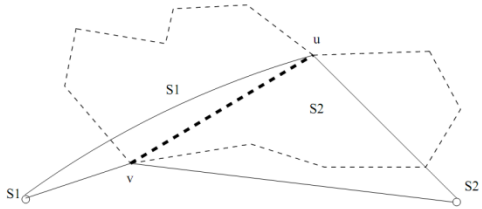


Problem for opaque obstacles

- There are three types of edges of CW-Voronoi Diagram:
  - (1) A part of obstacles
  - (2) A part of a perpendicular bisector between two sensors
  - (3) A part of an extension of a visible line
- Set of possible intersections:
  - Type 1, 2, 3: two ends of the edge.
  - Type 2: The intersection of the line formed by two sensors and the edge.







Dual graph of CW-Voronoi diagram

- Vertex set: set of sensors, the vertices of Voronoi diagram, s, t.
- Edge set:
  - For each edge (u, v) of Voronoi diagram that separate cells labeled by S_1 and S_2 , add four dual edges (u, S_1), (u, S_2), (v, S_1), and (v, S_2). If (u, v) intersect with S_1S_2 at T, add edges (S_1, T), (S_2 T).
  - For each edge (u, v) of type 1, which belongs to the cell labeled by sensor S, add two edges (u, S) and (v, S).
  - The weight is the Euclidian between two ends points.

Algorithm to compute BCP(s, t)

Algorithm 1 Calculate $BCP(S, O, s, t)$ for Opaque Obstacles

1: Use a known algorithm to construct the constrained Voronoi diagram of all n sensors and m obstacles (assign each sensor a weight of 1).

2: Construct the dual of this C-Voronoi diagram as described in Section 4.1.1.

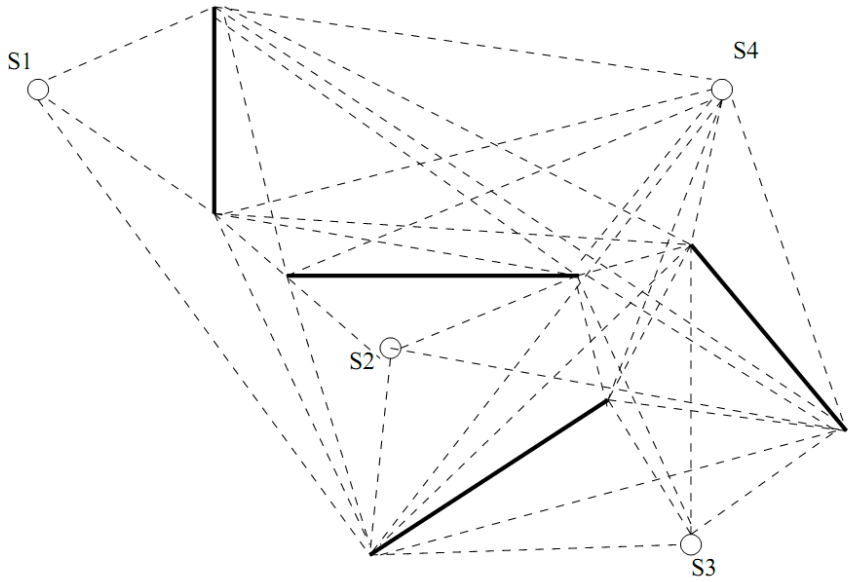
3: Run *Bellman-Ford* algorithm on this constructed dual graph starting at point s and ending at point t, which computes the *Best Coverage Path* between s and t.

4: The value of $Cover = \max(weight(e_1), weight(e_2), \ldots, weight(e_r))$ in the constructed path, where $e_1, e_2, \ldots, e_r$ are the edges in the best coverage path, $BCP(s, t)$.

- Time complexity: (1) takes $O(m^2n^2 + n^4)$ time and space to construct a CW-Voronoi diagram with $O(m^2n^2 + n^4)$ number of edges and vertices. Bellman-Ford algorithm at step 3 takes time $O((m^2n^2 + n^4)\log(mn + n^2)))$ .The total time is $O((m^2n^2 + n^4)\log(mn + n^2)))$.

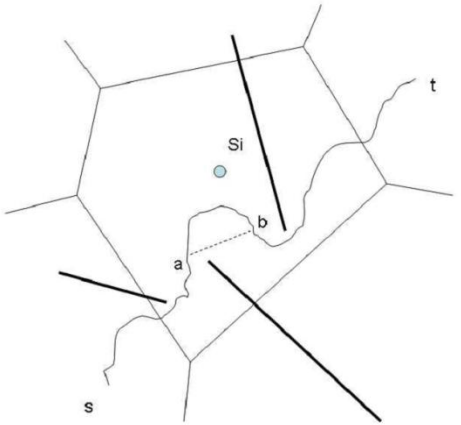
Problem for transparent obstacles

- Visibility graph o n locations is the graph of n vertices where there is an edge between a pair of vertices if they see each other.
- Observation:
  - At least a BCP is contained in visibility graph

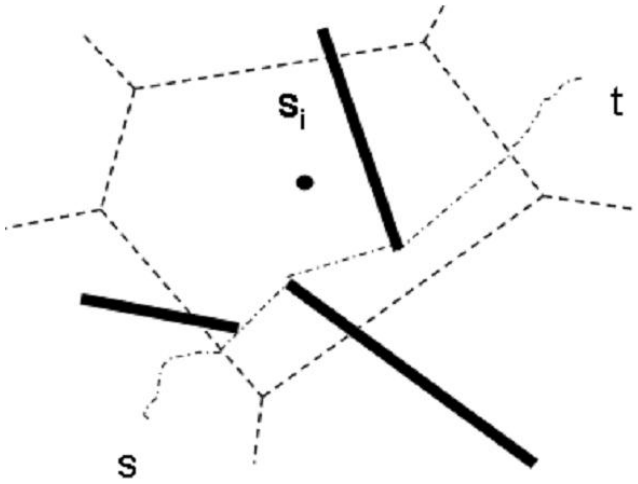


Problem for transparent obstacles

- A BCP which does not follow the visibility edges makes some bend either:
  - Type 1: Inside a Voronoi cell
  - Type 2: At a Voronoi bisector
  - Type 3: At a Voronoi vertex

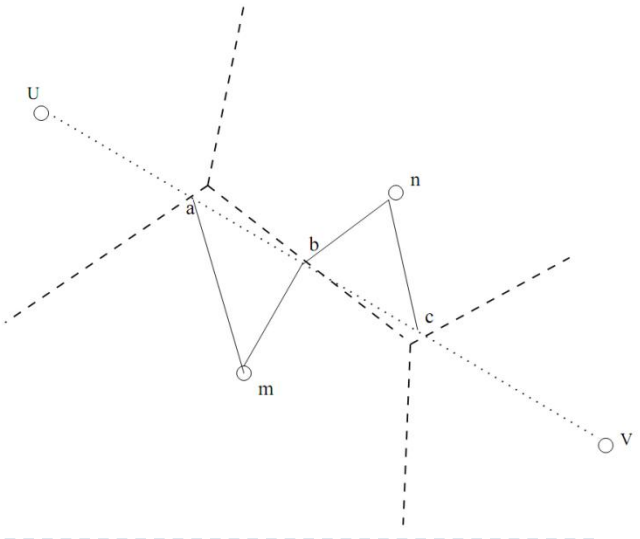


- Eliminate bends by replacing a arbitrary path from A to B by a line segment from A to B.



Problem for transparent obstacles

- Weight of visibility edge
  - Decompose an edge into separate line segments, each segment belong to a Voronoi cell.
  - Weight of each segment is Euclidian distance from the farther end to the corresponding sensor.
  - Weight of the edge is the max weight over all segments.



Algorithm computes BCP(s, t)

Algorithm 2. Calculate $BCP(S, O, s, t)$ for Transparent Obstacles

1: Construct the visibility graph of n sensor nodes, m line obstacles, point s and t.
2: Overlay a (normal) Voronoi diagram of the n sensor nodes on top of the created visibility graph.
3: Assign weight of each edge $e = (u, v) \in$ visibility graph
4: Run the Bellman-Ford algorithm on this weighted visibility graph starting at point s and ending at point t to compute a $BCP(s, t)$.
5: $cover = \max(weight(e_1), weight(e_2), \ldots, weight(e_r))$, where $e_1, e_2, \ldots, e_r$ are the edges of $BCP(s, t)$.

- Time complexity: Step 1 takes $O((m+n)\log(m+n)+x)$ to construct visibility graph where x is the number of visibility edges. Assigning weight to each edge takes O(n) time. Then step 3 takes $O(nm^2 + n^3)$. Step 4 finishes in $O((m+n)\log(m+n))$. Total time is $O(nm^2 + n^3)$.

Conclusion

- The algorithms requires to know exactly locations of all sensors, obstacles.
- Centralized algorithms.
- Does not solve the Maximum breach path problem.