Motivation: Non Combatant Evacuation

- Robots are used for rescue missions which involve finding the safest path out of the environment.
- The environment here is **dynamic and uncertain**.
- The robots - numbered, escort the human along a path to the goal. Mobile adversaries are shown in purple. As adversaries are eliminated, the risk in the environment changes and the agents want to update the rescue path.

**Risks Aware Search in an Uncertain Environment: RAGS**

- **Uncertain environment**: The algorithm RAGS [1] (Risk aware graph search) finds low cost paths through a graph.
- **Edge costs are normal distributions**: The cost of traversing an edge is only revealed when the edge is reached.
- The RAGS algorithm works by computing a set of **Non Dominated paths**: candidate paths likelyest to yield the lowest travel cost. Path A dominates B if

$$ A \succ B \Leftrightarrow \mu_A < \mu_B + \sqrt{2(\sigma_A^2 + \sigma_B^2)} \cdot c_r \cdot t^{-1} \cdot (1 - 2d_{	ext{thresh}}) $$

- It selects the final path by including the dynamically revealed edge costs.

The safest path to the goal changes with changes in the environment.

<table>
<thead>
<tr>
<th>The safest path to the goal changes with changes in the environment</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>The best candidate paths i.e non dominated paths are shown in red and the path taken is shown in blue.</th>
</tr>
</thead>
</table>

Dynamic Replanning in a Dynamic Environment: d-RAGS

Due to similarities with A* [2], replanning with RAGS can be improved following memoization techniques **similar to D* lite** [3].

Optimizations to RAGS for dynamic RAGS (d-RAGS) include:

- **More frequent non dominance checks**: Paths are checked for non dominance property before expansion (as well as before heap insertion).
- **Update of candidate set and backwards search**: By searching backwards, the root of the path sets remains the same and the information can be updated rather than recomputed.

Much of the candidate set remains similar between replans. Planning backwards takes advantage of this for updates.

Performance: Comparing d-RAGS and RAGS

- Graphs were generated with a range of sizes (number of nodes) and varying number of edges that would change upon inspection (i.e a 2 step lookahead).
- For each edge variance = \( U(1, 10) \)
- edge mean = \text{distance between(nodes)} + \( U(0,10) \)
- Performance is measured by observing percentage of extra algorithm operations in RAGS over d-RAGS.
- Path expansion - a subpath is evaluated as non dominated and paths one edge longer are added to the heap.
- Heap operations - subpaths are added or removed from the heap.

The costs of paths produced by each algorithm are similar.

Conclusions

- d-RAGS is able to find a low cost path with fewer operations in this case of 2 step lookahead for edge updates.
- The benefit becomes more pronounced as the graph size increases.
- The majority of the speedup comes from the more frequent non dominance checks.

Future Work

- Formally prove the correctness of this approach.
- Further optimize so performance gains are seen in the case of edge updates from anywhere in the graph to include updates seen by all robots in a multi-agent scenario.
- Include information theoretic concepts in the edge weight and updation to adapt algorithm for planning paths for information gain.

Acknowledgements

Along with the other authors and Prof. Katia Sycara, I would like to thank the Advanced Agent-Robots Technology lab for its support.

References