Homework 6 : Review of Motion Planning for Mobile Robots
Nahush Gondhalekar

Write the summary of the paper in your own words

Authors start off well with explaining the need of a method for the selection of a combination of Motion Planning Algorithms which forms a good motivational foundation of the paper. The background work in corroborating the motivation and the need of this paper,though not exhaustive is covering the required angles. The paper is mainly based on hypotheses and assumptions and does not talk in detail about a certain things while trying to find out the best approaches to combine algorithm classes. "Obstacle Uncertainty" , "Cluttered Environment" etc. are some of the terms which are very vague in their definition and lack a proper mathematical foundation which has a significant effect on choosing the Representation classes. Similarly in the section of "Search Algorithms classes", "Significantly Changing Environment" is a term which changes in every scenario and is very unclear. Assuming the design criteria definitions to be clear (Which are not as mentioned above!), the authors try to develop a systematic approach in finding out the best possible combination of combining algorithm classes, representation classes and selecting a search algorithm. So on the basis of a vaguely defined premise, authors try to conclude the correctness of their algorithm. Also, the considerations about a homogeneous or a heterogeneous team of robots is not done while doing the analysis. This is one of the drawbacks of the paper since many systems these days rely on a team of robots mainly involving heterogeneity. Even though the authors claim that the paper gives a true "generic combination of selecting multiple algorithm and representation classes, the real world scenarios pose different challenges which won't always be handled in the given settings. Though the claims that authors have made look promising, only three use cases to verify the claims are not enough since the planning problems given in those situations don't consider all the possibilities of planning problems exhaustively. Overall, this paper gives a good idea on selecting a proper strategy to combine algorithm classes and select representation classes and algorithm classes.

Seven requirements of a motion planning algorithm.

- Completeness : In terms of completeness not all bug algorithms are complete. Bug0 fails to declare failure in finite time, whereas Bug2 is complete since it can terminate if a solution exists or return failure in finite time if the solution does not exist. Dynamic programming algorithms re designed such that we check the convergence condition for success. In case of no convergence in a sufficiently large amount of time, the algorithm may be designed to return failure in finite time. A* is complete for a given graph under certain conditions where the nodes on an optimal path exist on the given input graph. Otherwise A* may be incomplete in scenarios where there are very narrow corridors and the discretization of the environment is not fine enough. RRT and its variants can be designed to return failure in the code by limiting the number of total nodes explored etc. to satisfy completeness. Again, the class of MDPs rely on bellman update equation which is dynamic programming and hence can be designed for completeness under certain conditions.

- Optimality : The class of Bug algorithms may not provide optimal solution. Dynamic programming, A* algorithms can provide optimal solutions under proper conditions of completeness where the algorithm terminates. RRT family will only give "A path" from start to goal which is not optimal. The class of MDPs provide optimality in expectation.
- Correctness : Bug0 may not always lead the robot to reach the goal but Bug2 guarantees that, the robot will reach the goal if a solution exists. A* and dynamic programming and RRT approaches will provide correctness when the solution exists. In the class of MDP algorithms, the robot may not reach the goal, it'll reach the goal in expectation.
- Dealing with kinodynamic Constraints : Bug algorithms may be modified to take care of Kinodynamic constraints. RRTs can certainly handle these type of constraints too. In other class of algorithms, there is no inherent support to handle these constraints.
- Robustness against a dynamic environment : Mainly, modifying the A* algorithm to D*, anytime D*, the dynamic environments can be handled. Bug algorithms and RRT and its variants certainly don't have the robustness against changing environments. The MDP class of algorithms may need revaluation of transition dynamics in case of dynamic environments.
- Robustness against uncertainty : The MDP class of algorithms is mainly designed for uncertainty and hence provides robustness against uncertainty. All the other algorithms don't provide inherent support against uncertainty and assume deterministic motion and environment for them to work. Dynamic programming class of algorithms can accommodate the uncertainty, in that case they'd likely become the MDP class of algorithms.
- Computational Complexity : Bug algorithms are very easy to compute. Dynamic programming and MDP class of algorithms run recursions which leads to high computation cost. A* type of algorithms work on a known graph in a deterministic environment hence depending on the size of the graph resolution and are more computationally complex as the graph becomes larger. RRT class of algorithms randomly try to explore the environment and try to create a tree which certainly leads to a high computation cost.

Analysis of Flowcharts

The conditions given in the flowcharts are not very specific and no type of exception handling is observed. For example, "Is the environment cluttered" does not provide any quantitative information about the environment and the appropriate representation or algorithm to choose. The computational complexity of the algorithm or the parallel nature of computation in certain algorithms is not taken into account while designing the flowcharts. The flowcharts are comprehensive and give a good guideline to select an appropriate representation and algorithm class, but they lack explanations about corner cases and about more specific systems like heterogeneous teams of robots etc.