Planning (16-350) HW 2 - Graduate

- Roshan Pradhan (roshanpr)

Planner details

The symbolic planner is implemented as an n-ary tree with each node of the tree containing - unordered_set of grounded conditions and a vector with corresponding grounded actions, vector containing pointers to the successor nodes, pointer to the parent node, and G and H values.

The n-ary tree is extended using a ‘calcSuccessor’ function, which is the major chunk of the algorithm. Using the STRIPS procedure, actions and symbols are iterated through to generate a set of grounded actions, validity of these actions is checked, and effects are computed to get a set of feasible successor states. These states, if not existing in the closed list, are added to the tree and the pointer is inserted into the open list. A standard A* / weighted A* search is used to search this graph and obtain the final list of actions.

In all 3 cases, the heuristic used is the number of goal literals missing from the current state. In the case of the fire extinguisher, the formulation is such that the heuristic counts the difference in the number of times fire has been extinguished at the current state versus the goal state (3). Refer to the env description files for more detail.

Results and Discussion

Comparison Table:

Case	**eps = 0 (Dijkstra)**	**eps = 1 (A*)**	**eps = 100 (Weighted A*)**
Blocks-world			
Time taken (ms)	34.62	6.90	5.63
States expanded	21	5	4
# of actions	3	3	3

Block-triangle			
Time taken (ms)	38.80	2.56	9.12
States expanded	1712	126	401
# of actions	6	6	7
Fire-extinguisher			
Time taken (ms)	19.14	10.87	4.99
States expanded	2242	1455	667
# of actions	21	21	21

Optimal task plans:

Blocks-world:
MoveToTable(A,B) -> Move(C,Table,A) -> Move(B,Table,C)

Block-triangle:
MoveToTable(T1,B3) -> MoveToTable(T0,B0) -> MoveToTable(B0,B1) -> MoveTo(B1,B4,B3) -> MoveTo(B0,Table,B1) -> MoveTo(T1,Table,B0)

Fire-extinguisher:
MoveMobile(A,B) -> LandQuad(B) -> MoveQuadMobile(B,W) -> FillTank(Q) -> MoveQuadMobile(W,F) -> TakeoffQuad(Q) -> ExtinguishFirst(Q) -> LandQuad(F) -> MoveQuadMobile(F,W) -> FillTank(Q) -> Charge(Q) -> MoveQuadMobile(W,F) -> TakeoffQuad(Q) -> ExtinguishSecond(Q) -> LandQuad(F) -> Charge(Q) -> MoveQuadMobile(F,W) -> FillTank(Q) -> MoveQuadMobile(W,F) -> TakeoffQuad(Q) -> ExtinguishThird(Q)

Discussion:

Firstly, we can observe that sometimes the weighted A* performs worse than A* in terms of number of expansions. This is because the heuristic used is not admissible, so the usual theoretical properties sometimes break down. Also, as expected we can observe weighted A* giving a suboptimal solution in the block-triangle case.

In the uninformed block-triangle case, we can see that even though # of expansions are lesser than the uninformed fire-extinguisher case, the time taken is more. This is because there are a lot more feasible successor actions per expansion that need to be checked for validity - which is an expensive operation.

As expected the number of expanded states are far fewer when the heuristics are used. This means the heuristic is doing its job of speeding up the search, however by using the inadmissible heuristic we lose theoretical guarantees of optimality. To maintain guarantees, we would have to implement the relaxed version of the problem for each state (without deletion), which itself is expected to be quite an expensive calculation- and not worth it for such a small scale problem.

Overall the heuristics seem to be guiding the search for the block-triangle case quite well, whereas not so much for fire-extinguisher.

Results and Discussion

The environment files are stored in ‘block_triangle.txt’ and ‘fire_extinguisher.txt’ respectively. To compile run with everything in the code folder the following command - ***g++ planner.cpp utilfunction.cpp -o planner.out***

In planner.cpp, add the environment to be tested in the int main() function and then compile.

To change the epsilon weight, go to ‘*utilfunction.h’* and change *eps* in struct CompareF{}.