Homework 01

Deepak Sharma
CSCI 630-01 Foundations of Intelligent Systems

September 9, 2016

1

If the 2D Space containing multiple convex obstacles have the total area more than the total area of convex obstacles then the cardinality of the set containing all real coordinates which belong to the free space is infinite. The set of free space coordinates contains all real numbered coordinates of the arena which does not belong to the convex obstacles. The presence of free space is a necessary condition for having a path between source and destination but not the only condition as the orientation of the obstacles also govern the possibility of a path. So the number of paths can be from 0 to infinity and each path can contain infinite number of states. However, the maximum number of optimal path between the source and destination is only one. The optimal path will contain segments of the straight line lines passing from some vertices of the convex obstacles.

In order to proof that shortest path between source and destination pass through some vertices of the convex obstacle polygons we will consider three below cases.

1. When source and destination points lie on any tangent of the same or different polygons, As shown in the figure 1 source point S and destination point D are placed on the polygon. Here by locating the point P on various locations we can generate multiple paths between source and destination which passes through the point P. If the path passing through P is optimal then by moving the point P on a specific neighboring coordinate we should not find a shorter path than the previous path. For example by moving the point P to the direction of the P' we successively approach to a shorter path than previous paths hence the previous path was not optimal. As shown in the figure 1 by iteratively moving the point P it converge at P'. The veracity of this process can also be verified by triangle properties as in triangle SPP' we can say SP' <SP + PP'. Again by repeating the process in triangle VP'V' a new point p' will converge at location V' as in triangle VP'V' we can say VP'+V'P' >VV' using triangle properties. SV'VD is the optimal path and it contains line segments which connect the vertices of the convex polygons. We have noticed that convergence of the point was achieved once one of the polygon vertices restricted the movement of point P' hence we can say optimal path will pass from the vertices of the convex obstacles.

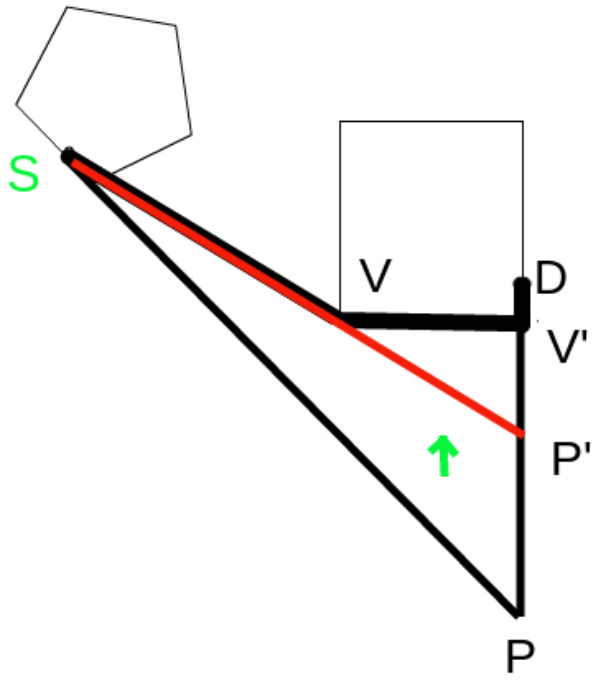

Figure 1: Source point S and Destination D on the tangent of the polygons

2. When both source and destination points don't lie on the tangent any polygon , as shown in the figure ....source point S and destination point D are placed at any arbitrary location in the space. We assume that optimal path between source and destination passes through the point P. If we find any neighboring point P' which offers more optimal path between source and destination then we can move the point P to P'. As we can say distance SP+PP'>SP' by triangle properties hence the point P' offer a shorter route than point P. By repeating the iterative process in triangle PED' and E'D'D we will find the optimal path between source and destination passes through the vertices E and E' of the polygon.

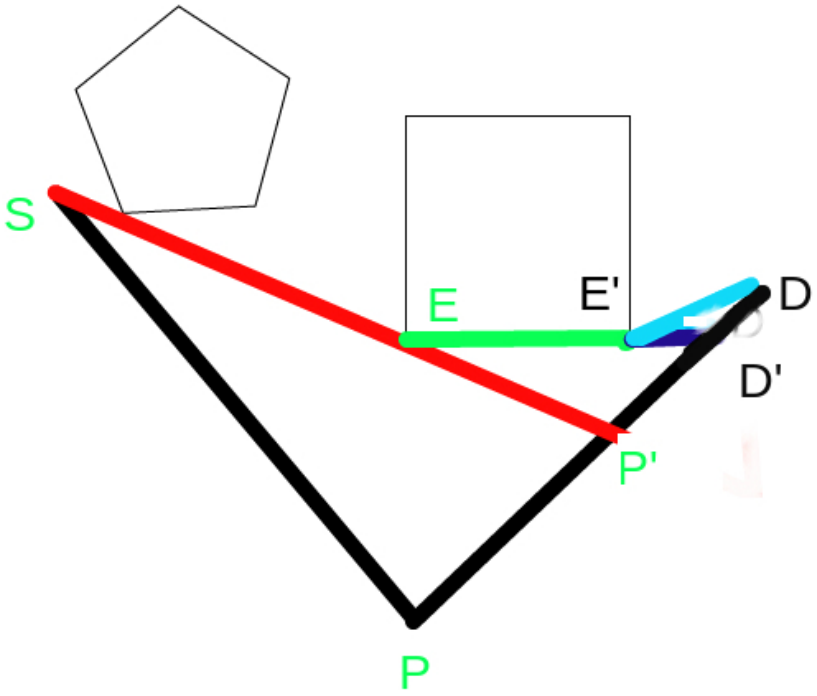

Figure 2: Source point S and Destination D is not situated on the tangent of the polygons

3. When one of the points will be on the tangent of a polygon and other on will not on the tangent then using above two cases we will again find that the optimal path between source and destination will pass from some of the vertices of the polygon obstacles.

Based on our above discussion, the state space for the given problem can be defined as the set of coordinates representing the vertices of the convex polygons. Using the source and destination coordinates along with the polygon vertices we can find the optimal path, if exist, between the sources and destination.

2

2.1 World State

World state define a particular real world configuration of the subject. It incorporates all information pertaining to a configuration of the subject while providing the description of its

state. For example, a world state description for a truck which is traveling from Rochester to buffalo provides information of current GPS coordinates, the personal state of the driver like drives mood and the information of music which driver is playing in the truck. All information captured by the world state may or may not be useful for solving the associated AI problem.

2.2 State Description

State Description is an abridged version of the world state which only induces information of the state which are relevant to the associated AI problem. Thus State description reduce the complexities of the world state which ultimately enable the designer of the AI agent to define formally state space for an AI agent. so that it can perceive and store and process efficiently. AI agent should be capable of distinguishing between two states using respective state description.

2.3 Search Node

A search node belongs to the search tree generated while executing any search algorithm for provided problem and state space. In a search tree, each search node can have multiple children and each child represents the states reached by taking associated action from the parent search node. These nodes represent the configuration obtain by the agent at various instances while executing the pertaining problem-solving algorithm. The path from root search node to the goal search node represents the history of actions taken by the agents in order to achieve the goal configuration. The root node of the search tree represents start state.

These distinctions facilities us to design an effective and efficient AI agent which does not require to store and process unnecessary information. AI agents only refer to the State description while inquiring the details of the Agent State. State Space provides successor function which facilitates the AI agent to know the set of action it can take while being in the state. Search tree graph facilitates the algorithm by providing references to the previously achieved states.

3

a

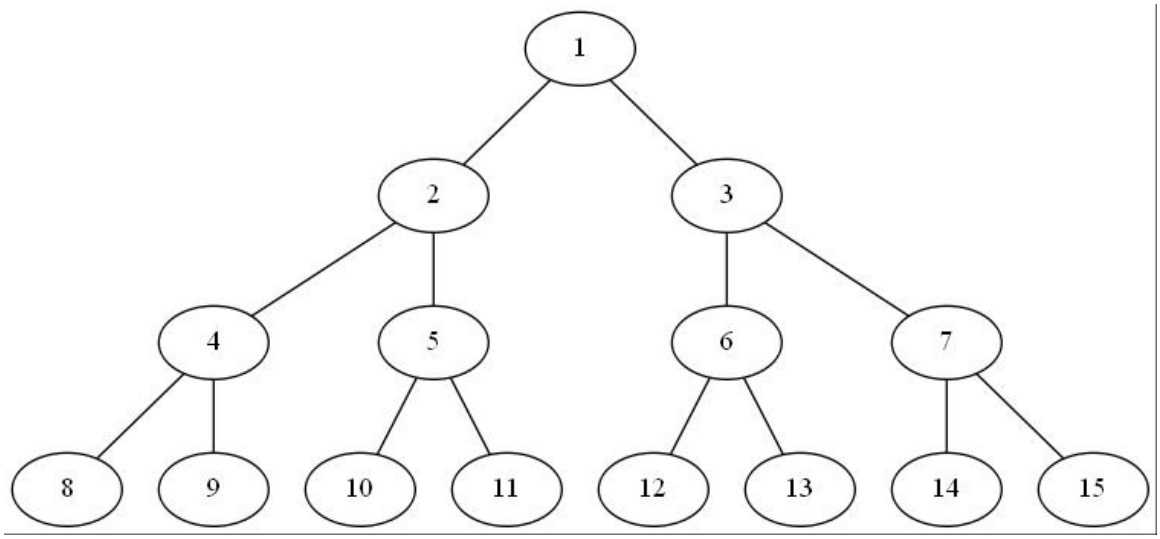

Figure 3: A portion of state space from 1st to 15th state

b

BFS Order

[1 2 3 4 5 6 7 8 9 10 11]

DFS Order

[1 2 4 8 9 5 10 11]

IDF Order

First Iteration [1]
Second Iteration [1 2 3]
Third Iteration [1 2 4 5 3 6 7]
Forth Iteration [1 2 4 8 9 5 10 11]

c

Yes, As the predecessor of the Nth node is $\lfloor N/2 \rfloor$, we can initiating a bi-directional search algorithm like BFS from both source and destination at the same time. Here forward branching factor will be two and backward branching factor will be one.

d

For any given goal state N, the predecessor node will be $\lfloor N/2 \rfloor$ and the branching factor while moving from Nth state to 1st state is only one. Hence we can without propagating in the search tree nodes we can generate actions sequence for reaching from source to destination.

e

```
def  search_solution (goal_state, source_state):
    """
    Generate sequence of actions for reaching to the goal state
    from the given source state if any valid path exists
   :param goal_state starting point
    :param source_state ending point
    return list of action or None if no path exist between the
    source and destination state
    pre:Assuming graph edges are not bidirectional
    """
    solution=[]
    while(goal_state>source_state):
        if goal_state%2>0:
            solution.insert(0,'Right')
        else:
            solution.insert(0,'left')
        goal_state=goal_state//2
    if goal_state==source_state:
        return solution
    else:
        return None
```