Grundlagen der Wissensverarbeitung
Computer Science
Universität Hamburg

Übung 1: Search Space Design

Rafael Ruz Gómez
Miguel Robles Urquiza

5 November 2017

1. Exercise 2.3

1.1. In the game Scotland Yard, Mister X has to evade several detectives using different means of transportation (and spending tickets). We use the board and the transportation rules but discard everything else from the game: Suppose (against the rules of the game) you as Mister X has a fixed amount of steps A before the detectives may move a fixed amount of steps B each (and that is all nor more steps afterwards!). Mister X and the detectives start at different but known positions on the board. How would you find a place to go where the detectives can't reach you? Formalize your answer!

- The tree root node would be a vector containing the inicial positions of Mister X and the detectives.
- A "valid" node is a node where the position of Mister X is different to the position of each detective.
- The goal node is a valid node with a deepness of Q steps (Q is the maximun between A nd B).
- Each arc going from one node to another represents one step made for every character in the game, just by Mister X (in case A >B) or just by the detectives(in case B >A).
- We could use breadth-first as the search strategy
- After that, we should use depth-first search with a variation: for every possible possition reachable by Mister X, we must expand all the nodes contemplating all the possible combination of movements for every detective, and if any of them is "invalid" then we must delete all the nodes in that level and come back to the parent node.
- If we reach the maximun between Q steps to a valid node, and no other node in the same deepness level (in which Mister X has the same position) is invalid, then we have found a path and the position we should go as Mister X so the detectives can't reach us.

2. Exercise 2.4

Define the search space, the goal, properties for the search space and an appropriate search strategy for the following problems:

2.1. Placing furniture in a flat. There are different kinds of furniture you can put in a set of places. Try to find an optimal placement, e.g. no door should be obstructed and the chairs should be near the table.

Let's define the search space:

- We have a set of states, each one is a matrix representing the positions (x,y) in the flat and a pair of values in the matrix representing the furniture (for example 0 if it's empty, 1 if it's a table on it, 2 if it's a chair, etc) and a value depending on how good placed is the furniture (0 if the furniture is good placed, 1 if not, for example a chair far away from any table, 2 if the furniture is blocking a door)
- The start state is a matrix with the pair <0,0>in every position where we will be able to place a furniture, and -1 otherwise.
- The actions that can be done is to put a furniture in an empty position or moving an already placed furniture to a better position.

The goal is to place the furnitures in the "optimal" placement. So if the value of the state (the sum of the values of the matrix) is 0 (not including the walls, etc), and we have placed the furnitures, we have reached the goal.

The properties for the search space are that it's infinite and it contains cycles, because we can move furnitures wherever we want to, and it's a tree where the child nodes are created moving or placing a furniture

An appropiate search strategy would be A* but with X minimun steps, where X is the number of furnitures we have to place

Another option would be to define the start state with the furnitures randomnly placed.

2.2. Construction Site planning. When building a house, you can e.g. only paint the walls after the walls have been built and so on. In addition, several people may work on site at the same time on different parts of the house. We need a plan on who is doing what when to build the house as fast as possible.

Let's define the search space:

- We have a set of states, each one is vector with all the elements that we have to build(Position 0 build walls, position 0 build doors, ... , 10 paint walls, etc) and a pair of values in the vector representing the priority (Less priority, more hurry) and a value counting the time that each worker have to wait to build each element
- The start state is a vector with the pair <0,0>in every element
- The actions that can be done is to put build a new element or wait if you can not build any element

The goal is to build in "optimal" time. So if the value of the time (the sum of the values of the time) is 0 (Thinking that we have build the first element), and we have build the house, we have reached the goal.

The properties for the search space are that it's finite and it doesn't contains cycles, because we can't some element aready built.It's a tree where the child nodes are created building elements

An appropiate search strategy breadth-first search, because we know the extension of the state space

2.3. An elevate has to transport people in a sensible way. Suppose you have an elevator and several people want to use it, standing in different doors. What should the elevator do?

Let's define the search space:

- The algorithm is based on the current floor, the destination floor and the time that each passenger has been waiting. It is assumed that an elevator can only answer calls if it is unoccupied or if a call made when the elevator is operating has its destination in the same direction as the route it is on. When there is a call that goes in the opposite direction of the current movement path, the elevator only saves it for the next pass that carries the direction. The space state is a vector with n elements (n is the number of floors) with a 3-tuple: One the current floor, other the destination floor and other with the time that each passenger has been waiting
- The start state is a vector with the 3-tuple <0,0,0>
- The actions that can be done is to go up or to go down

The goal is to build in "optimal" time. So if the value of the time (the sum of the values of the time) is 0 (Thinking that always each person travels alone), and we have build the house, we have reached the goal.

The properties for the search space are that it's infinite and it contains cycles, because we can to any floor we want to, and it's a tree where the child nodes are created going up or going down

An appropiate search strategy would be Depth-First Search