LINGI2261: Artificial Intelligence Assignement 1 : Solving Problems with Uninformed Search - Numberlink Project

Ndizera Eddy El Jilali Solaiman

October 4, 2015

Introduction

1 Python AIMA

Here are the answers to the questions regarding the AIMA library.

1. **In order to perform a search, what are the classes that you must define or extend ? Explain precisely why and where they are used inside a tree_search.**

   In order to perform a search, the **Problem** class must be defined. We'll have to implement the following 3 methods *init*, *goal_test* and *successor*. To explain why and where they are used inside a tree_search, i'll use the pseudocode for a tree_search (found in IA: a modern approach).

```
function TREE-SEARCH(problem) returns a solution, or failure
  initialize the frontier using the initial state of problem
  loop do
    if the frontier is empty then return failure
    choose a leaf node and remove it from the frontier
    if the node contains a goal state then return the corresponding solution
    expand the chosen node, adding the resulting nodes to the frontier
```

Figure 1: Tree-Search from *Artificial Intelligence a Modern Approach*

In the pseudocode at the line 1, we see that to initialize we need the initial state of the problem. This initial state is given by the *init* method found in the *Problem* class.
At line 5 of the same pseudocode, to know if we reach the goal, the *goal_test* method should be called to know that.
Finally, the method *successor* isn't called directly in the pseudocode given.

It is called in the **Node** class by the method*expand* to know the set of children of a given node.

All other methods and class in the AIMA library are already well defined.

2. **In the expand method of the class Node what is the advantage of using a yield instead of building a list and returning it afterwards?**

   The advantage of using a yield is memory usage. By using a yield rather than a list we save space. In the method expand, when we iterate over the list of children of node, we removed them also.
   Moreover, in python yield operator allows us to perform lazily. Indeed the call to the lazy operator do only one thing: returning a generator object that will be call later. It allows us to give out new values on the fly and to save memory space

3. **Both breadth_first_graph_search and depth_first_graph_search are matking a call to the same function. How is their fundamental difference implemented?**

   The fundamental difference between the two functions is the type of queue that they put as parameter when calling the function *graph_search*.
   The first one (breadth) put as queue a *FIFOQueue* whereas the second one (depth) use a *Stack LIFOQueue*.
   With a *LIFOQueue* the last most recently generated node is chosen for expansion. It implies that we explore first completely a branch before passing to another according to the definition of depth first tree search
   With a *FIFOQueue* the first generated node is chosen for expansion. It implies that we explore first all the nodes of a specific depth according to the definition of breadth first tree search

4. **What is the difference between the implementation of the graph_search and the tree_search methods and how does it impact the search methods?**

   The difference between the tree_search and the graph_search is the variable closed that is a data structures that stocks the nodes already visited. With that we can avoid the infinite loop problem (occuring in depth_first).

5. **What kind of structure is used to implement the closed list? What are the methods involved in the search of an element inside it? What properties must thus have the elements that you can put inside the closed list ?**

   The closed list is implemented using a dictionnary.
   It uses a key to search a value in it. So any element is stored as a key-value pair. The key is like an index in the dictionnary and the value, the element stored.

The elements must be comparable?, unique?, immutable?.

6. **How technically can you use the implementation of the closed list to deal with symmetrical states ?**

   By modifying the compare method of the type used as key as two symmetrical states are equal we can deal with it.

2 The Numberlink Problem

1. **Explain the advantages and weaknesses of the following search strategies on this problem (not in general): depth first, breadth first.**

   The depth first search in general takes less time than the breadth first one. Also in this particular problem, the depth first has the completness property like the breadth first because there cannot have infinite search (the square can be filled).

   For the space occupied by the two serach algorithm, this is not a issue because we use a yield to expand a node and not a list.

   The breadth first is advantageous with trees that have a high branching factor.

2. **How can we exploit the uniqueness of solution to reduce the search space? Why is the method pathExists useful?**

   The method pathExists is useful to know if a a current node don't give an answer.

3. **Is the order in which we choose the paths important? How can we use this to reduce the search space? When starting a new path, we can choose to start with any of its two endpoint. How should this choice be done?**

   Yes it's important. Depending of the order, we can reduce the search space. For that, let's use the example case in your assignement pdf, if we take the D path first, we have 4 choices in total. But if we take the A, we have 5 choices.

   The choice of endpoins is also important because if we take, in the same example, the e path and take the e point located at the top-right corner, we only have one choice.

4. **What are the advantages and disadvantages of using the tree and graph search for this problem. Which approach would you**

choose? Which approach allows you to avoid expending twice symmetrical states?

The tree search takes less time in case that the problem cannot have symmetrical states. But the graph search could save time in case symmetrical states are present in the search.

The tree search doesn't lose time in checking if the current node has already been visited. Note that this problem cannot have infinite loop (explain why).

We will choose the graph search approach.

5. **Implement this problem in Python 3. Extend the Problem class and implement the necessary methods and other class(es) if necessary. Your file must be named numberlink.py. You program must take as only input the path to the init file of the problem to solve. It must print to the standard output a solution to the problem satisfying the above format. Your file must be encoded in utf-8. Submit your program on INGInious.**

6. **Experiments must be realized with the 10 instances of the numberlink problem provided. Report in a table the results on the 10 instances for depth-first and breadth- first strategies on both tree and graph search (4 settings). You must report the time, the number of explored nodes and the number of steps from root to solution. When no solution can be found by a strategy in a reasonable time (3 min), explain the reason (time-out and/or swap of the memory). What do you conclude from those experiments?**

Conclusion