20CS6033 AI – I

Fall 2016

Instructor: Anca Ralescu

**Homework Assignment #2**

**Assigned on Tuesday, September 13, 2016**

**Due on Tuesday, September 20, 2016**

**40 points**

**Part I(20 pts):** **Some Direction First: Depth-Breadth First Search**

Combine the programs **depthF** and **breadthF** discussed in class into a hybrid search program which takes as arguments an *undirected graph*, an integer **(T)**, a starting node **(S)** and a goal node **(G)** in the graph and finds a path **PATH** without cycles from **S** to **G** in the order prescribed by the value of **T**:

1. if **T** < **0** the search should proceed as a depth first search of the whole graph **(depthF) ;**
2. Otherwise, if  **T > 5** the search should proceed as a breadth first search of the whole graph (**breadthF)**;
3. Otherwise, that is for values of **1 ≤ T ≤ 5,** the search should proceed as a *hybrid of the two searches*, as follows: At each node, **n**, the choice between the depth first and breadth first search is done according to the out-degree **d(n)** (recall that the out-degree of a node is the number of edges originating from that node):

3.1 if **d(n) < T** then the search proceeds as a depth first search;  
3.2 if **d(n) ≥ T**  the search  proceeds as a breadth first search.

Call your top predicate  **(hybridDB T S PATH)**. Assume that your graph is represented by clauses defining a symmetric **connect** predicate where **(connect X Y)** holds if the node **X** is connected to node **Y.**  The goal state **G** should be stated in the program knowledge base by the clause **((goal G))** (e.g. ((goal a)) or, for a blocks world problem as ((goal ((on a b) (on b table) (clear a)))) ).  
  
In addition to the well documented program please include a short discussion comparing this algorithm to the simple depth first and breadth first algorithms: your discussion should include the following points:  the hybrid search is better / worse / sometimes better than its simple search components.  You should either give example or a proof (informal) of your statements.  
   
Insert your discussion in a comment section at the end of your program.

**Part II(20 pts) :** **Depth-Breadth-Heuristic First Search**

Let us take this hybrid notion one step further.  Assume that edges now have a cost, so your connect predicate will be something like **(connect X Y C)** where **C>0** is the cost of traveling from **X** to **Y**.  Now your program should switch between **hybridDB** and the **A\*** algorithm according to some global information on the cost along the edges of the graph.  
  
First you need to implement the **A\*** algorithm. For this you must have a predicate which defines the heuristic function for each node given the start and goal nodes and one that computes the evaluation function. You can start by assuming very simple heuristics (e.g. constant).  
  
It is useful to notice that **A\*** is similar in spirit to either **depthF** or **breadtF** with the difference that the list of nodes to be expanded is maintained in non-decreasing order of the evaluation function (a priority queue).   
This means that  once a node, say **X**, has been expanded to  **Y**, the whole path passing through **X** and ending up in **Y** along with the corresponding value for the evaluation function for **Y**, **FY**,  must be  inserted in the list of paths to be expanded so that this list is always  ordered with respect to the values **FY**.  
  
Let **AVEcost** denote the average cost of an edge in your graph: this is calculated by finding the sum of all costs and divide by the number of edges (this number is actually the number of connect clauses in your program). Then find **STCost** the standard deviation of costs (average difference between costs and the average cost).  
  
Now **SCost** denote the number of edges with cost less than **AVEcost** and **BCost** the number of edges with cost greater than or equal that **AVEcost**.  
  
Let **AVED** denote the average degree in the graph (total number of edges divided by the number of nodes from which edges originate).  
  
Now, your new hybrid search, let us call it **hybridDBH,** takes as argument the start node (**S**) and a threshold **T**, and finds a **PATH**and its **PCost** as follows:

1. **When STcost  > T then the algorithm  acts as A\*;**
2. **Otherwise (that is, when STcost  ≤ T) :**

**2.1 When SCost > BCost the algorithm acts as hybridDB with the threshold AVED  
2.2  Otherwise it acts as A\***

Basically, this *hybridsearch* captures the following strategy: if there is a great variety of costs in the graph then use **A\***.  Else, if the number of edges with small cost (smaller than the average cost) is larger than those with large cost, switch to the hybridDB search (which will switch to **depthF** or **breadthF** depending on the average degree in the graph and will ignore the cost), else (more high cost edges than small cost edges) switch to **A\***.  
  
Once again, include a brief discussion comparing this algorithm to its components **(hybridDB** and **A\***).