COMP3411/9814 Artificial Intelligence
Term 1, 2022

Assignment 1 – Search and Constraint Solving

Due: Friday 11 March, 10:00 pm
Marks: 20% of final assessment for COMP3411/9814 Artificial Intelligence

Part 1 - Search

Question 1: Search Algorithms for the 15-Puzzle

In this question you will construct a table showing the number of states expanded when the 15-puzzle is solved, from various starting positions, using four different searches:

(i) Uniform Cost Search (with Dijkstra's Algorithm)

(ii) Iterative Deepening Search

(iii) A* Search (using the Manhattan Distance heuristic)

(iv) Iterative Deepening A* Search

Go to theWebCMS. Under "Assignments" you will find Prolog Search Code "`prolog_search.zip`". Unzip the file and change directory to prolog search, e.g.

```
unzip prolog_search.zip
cd prolog_search
```

Start prolog and load puzzle15.pl and ucsdijkstra.pl by typing

```
[puzzle15].
[ucsdijkstra].
```

Then invoke the search for the specified start10 position by typing

```
start10(Pos),solve(Pos,Sol,G,N),showsol(Sol).
```

When the answer comes back, just hit Enter/Return. This version of Uniform Cost Search (UCS) uses Dijkstra's algorithm which is memory efficient, but is designed to return only one answer. Note that the length of the path is returned as *G*, and the total number of states expanded during the search is returned as *N*.

a) Draw up a table with four rows and five columns. Label the rows as UCS, IDS, A* and IDA*, and the columns as **`start10, start12, start20, start30`**

and **`start40`**. Run each of the following algorithms on each of the 5 start states:

```
(I)   [ucsdijkstra]
(II)  [ideepsearch]
(III) [astar]
(IV)  [idastar]
```

In each case, record in your table the number of nodes generated during the search. ***If the algorithm runs out of memory, just write "Mem" in your table. If the code runs for five minutes without producing out- put, terminate the process by typing Control-C and then "a", and write "Time" in your table. Note that you will need to re-start prolog each time you switch to a different search.***

b) Briefly discuss the efficiency of these four algorithms (including both time and memory usage).

Question 2: Heuristic Path Search for 15-Puzzle

In this question you will be exploring an Iterative Deepening version of the Heuristic Path Search algorithm discussed in the Week 2 Tutorial. Draw up a table in the following format:

	start50		start60		start64	
IDA*	50	14642512	60	321252368	64	1209086782
1.2 1.4 1.6						
Greedy						

The top row of the table has been filled in for you (to save you from running some rather long computations).

(a) Run [**greedy**] for **start50**, **start60** and **start64**, and record the values returned for *G* and *N* in the last row of your table (using the Manhattan Distance heuristic defined in **puzzle15.pl**).

(b) Now copy **idastar.pl** to a new file **heuristic.pl** and modify the code of this new file so that it uses an Iterative Deepening version of the Heuristic Path Search algorithm discussed in the Weak 3 Tutorial Exercise, with $w = 1.2$.
In your submitted document, briefly show the section of code that was changed, and the replacement code.

(c) Run [**heuristic**] on **start50**, **start60** and **start64** and record the values of *G* and *N* in your table. Now modify your code so that the value of w is 1.4, 1.6 ; in each case, run the algorithm on the same three start states and record the values of *G* and *N* in your table.

(d) Briefly discuss the tradeoff between speed and quality of solution for these five algorithms.

Part 2 - Constraint Solving

Question 1: Arc Consistency

Consider a scheduling problem, similar to the one discussed in lectures, where there are five variables A, B, C, D, and E, each with domain $\{1, 2, 3, 4\}$. Suppose the constraints are: $E - A$ is even, $C /= D$, $C > E$, $C /= A$, $B > D$, $D > E$, $B > C$.

Show how arc consistency can be used to solve this problem. To do this you need to

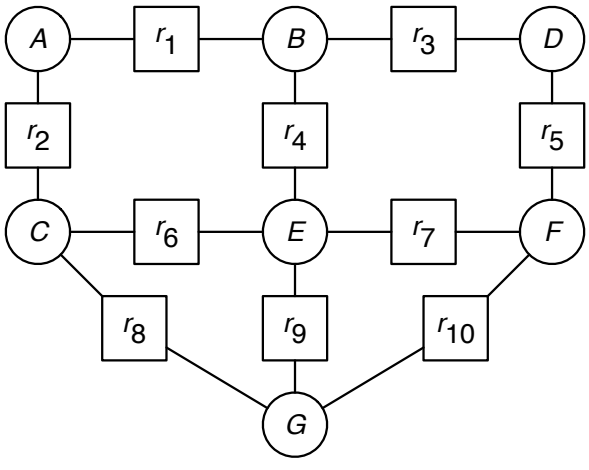
- draw the constraint graph,
- show which elements of a domain are deleted at each step, and which arc is responsible for removing the element,
- show explicitly the constraint graph after arc consistency has stopped.
- show how splitting domains can be used to solve this problem. Include all arc consistency steps.

Question 2: Variable Elimination

Consider the constraint graph, below, with named binary constraints. r_1 is a relation on A and B, which we can write as $r_1(A, B)$, and similarly for the other relations. Consider solving this network using VE.

(a) Suppose you were to eliminate variable A. Which constraints are removed? A constraint is created on which variables? (You can call this r_{11}).

(b) Suppose you were to subsequently eliminate B (i.e., after eliminating A). Which relations are removed? A constraint is created on which variables?



Submitting your assignment

Your submission will consist of a single PDF file *assign1.pdf* which should contain the results of your search experiments in part 1 and the answers to the questions in part 2.

To hand in, log in to a School of CSE Linux workstation or server, make sure that your files are in the current working directory, and use the Unix command:

```
% give cs3411 assign1 assign1.pdf
```

Please make sure your code works on CSE's Linux machines and generates no warnings. Remove all test code from your submission. Make sure you have named your predicates correctly.

You can submit as many times as you like - later submissions will overwrite earlier ones. Once `give` has been enabled, you can check that your submission has been received by using one of these commands:

The submission deadline is Friday 11 March, 10:00 pm.

10% penalty will be applied to the (maximum) mark for every 24 hours late after the deadline.

Questions relating to the project can be posted to the forums on the course Web site.

If you have a question that has not already been answered on the forum, you can email it to cs3411@unsw.edu.au

Plagiarism Policy

Group submissions are not allowed. Your program must be entirely your own work. Plagiarism detection software will be used to compare all submissions pairwise (including submissions for any similar projects from previous years) and serious penalties will be applied, particularly in the case of repeat offences.

DO NOT COPY FROM OTHERS. DO NOT ALLOW ANYONE TO SEE YOUR CODE

Please refer to the UNSW Policy on Academic Honesty and Plagiarism if you require further clarification on this matter.