University of Sheffield

AI Report

Testing searching methods

Junjin Chen
4/7/2016

Name: Junjin Chen Username: acc15jc Register No. 150144892

AI assignment report

Searching is a technique of problem solving in AI. This experiment was to test some searching methods and to find their features.

Experiment 1:

Initially, the LatticeSearch.java and LatticeState.java were created by copying and modifying from MapSearch.java and MapState.java. Then the goalP was set to return true if the hypothesis of End of Word is greater than 30. The sameState method was set to return true if this word compare to the word as parameter was 0. Here was the result:

Table 1: Word Lattice results of different searching methods.

getEnd>30	Branch and Bound	Breadth First	Depth First
Efficiency	0.5	0.571	1.0
No. of iterations	8	7	5
Result	Please lettuce know	Please lettuce know	Freeze let us know
Global cost	170	170	210

However, in the getSuccessors method, there was no transition cost, therefore the global cost was low. Besides, the result was not expected. The transition cost was to measure the confidence of link between 2 words. After adding the transition cost in getSuccessors, the result was:

Table 2: Word Lattice results of different searching methods when transition cost was added.

Transition cost added	Branch and Bound	Breadth First	Depth First
Efficiency	0.625	0.571	1.0
No. of iterations	8	7	5
Result	Please let us know	Freeze lettuce know	Freeze let us know
Global cost	290	370	360

Now the Branch and Bound search could find the expected result, the efficiency of breadth first search was low, depth first search had best efficiency but it was not admissible. This problem was concerned to be that the global cost was not limit. In goalP of SearchNode, after the global cost was limited to less than 300, here was the result:

Table 3: Word Lattice results of different searching methods when global cost was limited.

getGlobalCost<300	Branch and Bound	Breadth First	Depth First
Efficiency	0.625	Search fails	Search fails
No. of iterations	8	-	-
Result	Please let us know	-	-
Global cost	290	-	-

As result, the Breadth First search and Depth First search could not find a sentence whose global cost was less than 300. The problem was found that same word did not appear twice, which means that the word in a subtree would not be in any other subtree. This problem occurred in the sameState method, it should not return true if words were same. It was modified to always return false. The goalP in LatticeState, was previously returning true if the word ends at a number greater than 30, was modified to return true if no words can follow this word, the result was:

Table 4: Word Lattice results of different searching methods when sameState and goalP were modified.

sameState and goalP modified	Branch and Bound	Breadth First	Depth First
Efficiency	0.455	0.3125	0.357
No. of iterations	11	16	14
Result	Please let us know	Please let us know	Please let us know
Global cost	290	290	290

Now since the global cost should be less than 300, all 3 methods managed to find the expected result, though the efficiencies of all 3 methods were less than before. Comparing to their efficiencies, the Branch and Bound search was the most efficient method among others.

The TVLM.txt was tested. It did not work until the Word Lattice text file for this language model was created. The word lattice was set as following:

Table 5: Word Lattice of TV language model initiated.

Word	Start	End	Cost
TV	0	10	0
radio	0	20	0
channel	30	50	0
volume	20	40	0
up	40	50	0
down	40	50	0
on	10	20	0
off	10	20	0
record	10	30	0
playback	70	100	0
one	50	100	0
Two	50	100	0
Three	50	100	0
Four	50	100	0
Five	50	100	0
Six	50	100	0
Seven	50	100	0
Eight	50	100	0
nine	50	100	0

Table 6: Word Lattice results of different searching methods when sameState and goalP were modified.

Language Model switched	Branch and Bound	Breadth First	Depth First
Efficiency	0.455	0.294	1.0
No. of iterations	11	17	5
Result	Radio volume down six	TV record channel one	Radio volume down nine
Global cost	115	130	155

The result shows that Depth First most was most efficient, but it also had highest global cost, which indicates that it was not admissible. Breadth First search was least efficient due to the larger

language model, and the global cost was not the lowest, it was not admissible either. Branch and Bound search had the middle efficiency and least global cost though the difference of the result between Branch and Bound and Depth First was “six” and “one”, however 115 was not ensured to be the least cost of all possible sentences, another test was needed to confirm that the Branch and Bound was admissible:

Table 7: Word Lattice results of different searching methods when global cost was limited below 115.

getGlobalCost<115	Branch and Bound	Breadth First	Depth First
Efficiency	Search fails	Search fails	Search fails
No. of iterations	79	79	79
Result	-	-	-
Global cost	Increase along iterations	Varying with no order	Varying with no order

There was no result for the global cost less than 115, which means 115 was the least cost of all possible sentence, therefore, the branch and bound search was admissible. In addition, the trends of the global costs along the iterations were different. In Branch and Bound search, the global cost grew along the iterations but the others did not, which means Branch and Bound search follows the path of growing cost until it finds a solution, and the solution is the best solution, since other solution if continue to be discovered will cost more than this. However, depth first search always choose the last explored node, which means it would find the solution that is in the last of other solutions explored, and it is not necessarily the best solution. Breadth first search always choose the first explored node in open, which means it would find the solution that explored first than other solutions, and it is not necessarily the best solution. Thus, the Branch and Bound search is always admissible, while the others could be admissible when limited the global cost:

Table 8: Word Lattice results of different searching methods when global cost was limited below or equal to 115.

getGlobalCost<=115	Branch and Bound	Breadth First	Depth First
Efficiency	0.455	0.125	0.625
No. of iterations	11	40	8
Result	Radio volume down six	Radio volume down six	Radio volume down six
Global cost	115	115	115

They all found the best solution. However, in this case, the Depth First search was most efficient. The reason of that Breadth First search and Depth First search were admissible here was because there was only one solution, if there were more than 1 solution, they would find one of those solutions but it would be not necessarily the best.

In conclusion, Branch and Bound search is always admissible, Breadth First search and Depth First search are not always admissible.

Experiment 2:

A* search was imported to the Search.java. SearchState.java and SearchNode.java were modified to include the estimate remaining cost (ERC) and estimate total cost (ETC). A test was run with all the ERCs were set to 0, here shows the path of the test:

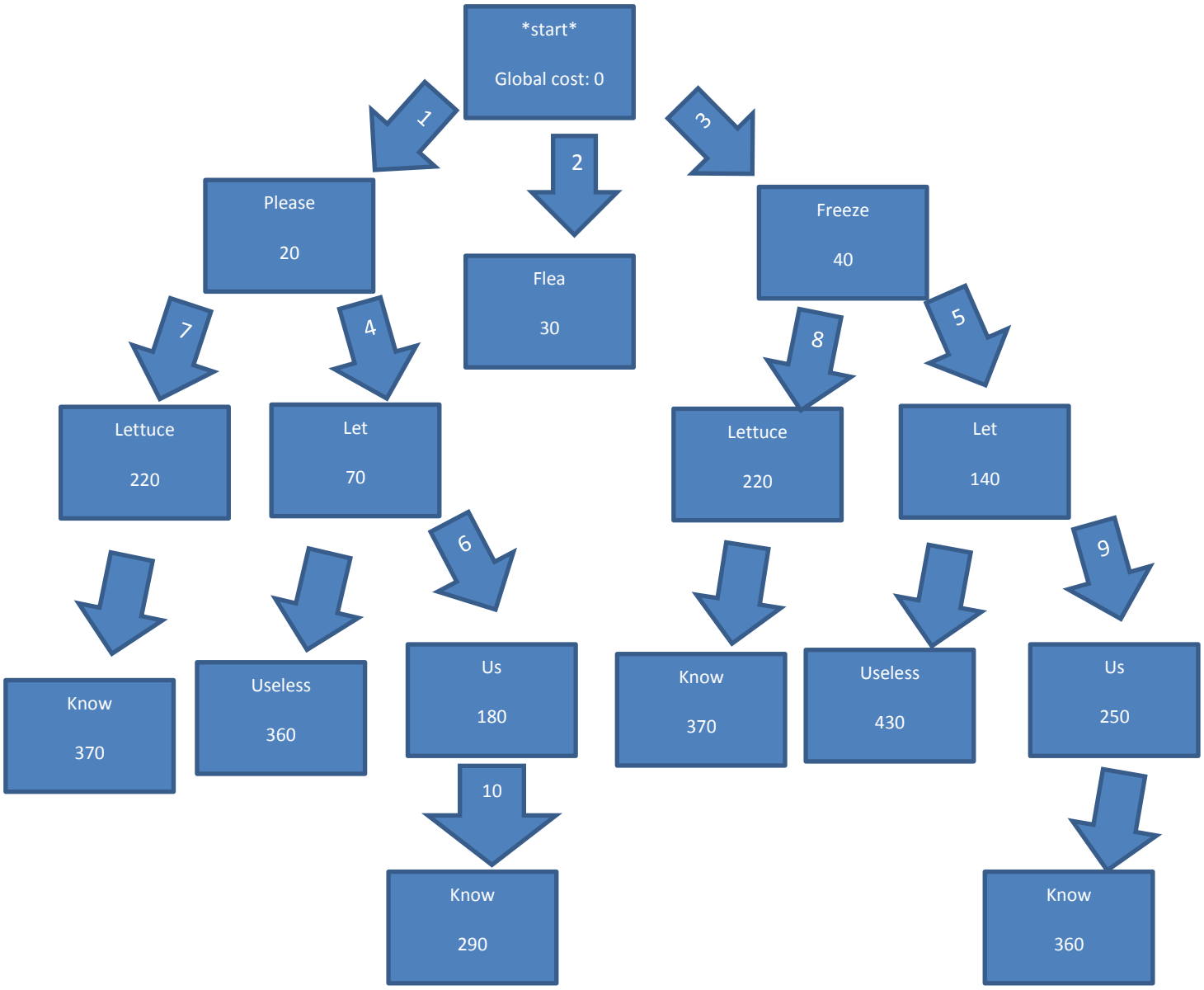

Figure 1: Tree of A* algorithm, with all 0 s.

Table 9: Word Lattice results of different searching methods when A* search was added, language model shifted and limitation of global cost removed.

A* algorithm added	A*	Branch and Bound	Breadth First	Depth First
Efficiency	0.455	0.455	0.444	1.0
No. of iterations	11	11	9	5
Result	Please let us know	Please let us know	Please lettuce know	Freeze let us know

Global cost	290	290	370	360

As result, A* algorithm acted exactly like Branch and Bound search. A* algorithm is the composition of Branch and Bound search and Best First search, however all the ERCs were 0, thus there was no Best First search involved. Here the A* algorithm was admissible.

A test was carried after the ERC was added in the latt1.txt.

Table 10: Word Lattice of latt1.txt with ERC added.

Word	Start	End	Cost	EstRemCost
please	0	10	20	270
Lettuce	10	30	100	150
Know	30	36	50	0
flea	0	9	30	0
use	16	29	120	70
throw	29	36	70	0
freeze	0	10	40	320
let	10	18	30	220
useless	18	31	200	80
throw	31	36	80	0
us	18	30	90	110

Table 11: Word Lattice results of different searching methods when ERC concerned.

Erc concerned	A*	Branch and Bound	Breadth First	Depth First
Efficiency	0.833	0.455	0.444	1.0
No. of iterations	6	11	9	5
Result	Please let us know	Please let us know	Please lettuce know	Freeze let us know
Global cost	290	290	370	360

Here the ERCs were calculated such that the estimate total cost could be the same as the final global cost. In this situation, the A* algorithm was about twice efficient as before, and it was admissible. The results of the others did not change.

The other language model with no change was involved:

Table 12: Word Lattice results of different searching methods when Language Model shifted.

Language Model shifted	A*	Branch and Bound	Breadth First	Depth First
Efficiency	0.455	0.455	0.294	1.0
No. of iterations	11	11	17	5
Result	Radio volume down six	Radio volume down six	TV record channel one	Radio volume down nine
Global cost	115	115	130	155

In order to test whether A* algorithm is always admissible, some values in the TV language model were changed. The end of word “one” was changed to 9, which means that the word “playback” would follow this word. Besides, the ERC of the number “one” was 0, while the costs of the other

"numbers" were 100, expecting that the A* algorithm would be affect by the inaccurate ERC and would go to "one playback" at the end.

Table 13: Word Lattice of TV language model with values modified.

Word	Start	End	Cost	EstRemCost
TV	0	10	20	20
radio	0	20	40	30
channel	30	50	30	20
volume	20	40	40	60
up	40	50	30	20
down	40	50	30	20
on	10	20	10	0
off	10	20	10	0
record	10	30	20	20
playback	70	100	90	0
one	50	70	0	0
Two	50	100	0	100
Three	50	100	0	100
Four	50	100	0	100
Five	50	100	0	100
Six	50	100	0	100
Seven	50	100	0	100
Eight	50	100	0	100
nine	50	100	0	100

Table 14: Word Lattice results of different searching methods when Word Lattice modified.

Word Lattice modified	A*	Branch and Bound	Breadth First	Depth First
Efficiency	0.3125	0.455	0.278	1.0
No. of iterations	16	11	18	5
Result	TV record channel four	TV record channel four	TV record channel two	Radio volume down nine
Global cost	190	190	200	265
estTotal Cost	290	290	300	365

The result of A* algorithm was not expected, it was admissible since it was the same as the result of Branch and Bound. However the efficiency of A* algorithm dropped down, A* search explored 5 more nodes than Branch and Bound search. Thus, the inaccurate ERCs do affect the A* algorithm.

Then the ERCs of the "numbers" increased to 200 except "one", which was kept 0. The test was run:

Table 15: Word Lattice of TV language model with ERC of “numbers” doubled.

Word	Start	End	Cost	EstRemCost
TV	0	10	20	20
radio	0	20	40	30
channel	30	50	30	20
volume	20	40	40	60
up	40	50	30	20
down	40	50	30	20
on	10	20	10	0
off	10	20	10	0
record	10	30	20	20
playback	70	100	90	0
one	50	70	0	0
Two	50	100	0	200
Three	50	100	0	200
Four	50	100	0	200
Five	50	100	0	200
Six	50	100	0	200
Seven	50	100	0	200
Eight	50	100	0	200
nine	50	100	0	200

Table 16: Word Lattice results of different searching methods when ERC of “numbers” doubled.

ERC of “numbers” doubled	A*	Branch and Bound	Breadth First	Depth First
Efficiency	0.286	0.455	0.278	1.0
No. of iterations	21	11	18	5
Result	TV record channel one playback	TV record channel four	TV record channel two	Radio volume down nine
Global cost	370	190	200	265
estTotal Cost	370	390	400	465

As result, A* algorithm was no longer admissible. In A* algorithm, though the estimate total cost was the least, the global cost was the most. Compare to the previous experiment, the admissible result was “TV record channel four” and the estimate total cost for it was 290, which was smaller than the estimate total cost of the sentence “TV record channel one playback”, 370. Here, the estimate total cost of the admissible sentence was 390, greater than the one of A* result. If the ERC of “TV” was changed to 400 and the one of “radio” was changed to 379, which means the ETC of “TV” was 420 and of “radio” was 419, then the result of A* algorithm would be “radio volume down one playback” and the global cost would be 405. Thus, A* algorithm would not be always admissible if the ERC is overestimate.

In conclusion, A* algorithm is not always admissible, unless all the ERCs are underestimate, but it could be efficient when the ERCs are accurate.