COMP3411 Assignment 1 Part 2 - Search

Jay Chen(z5261536)

Question 1: Search Algorithms for the 15-Puzzle

(a)

Algorithm	start10	start12	start20	start30	start40
USC	**2565**	**Mem**	**Mem**	**Mem**	**Mem**
IDS	**2407**	**13812**	**5297410**	**Time**	**Time**
A*	**33**	**26**	**915**	**Mem**	**Mem**
IDA*	**29**	**21**	**952**	**17297**	**112571**

(b)

USC: Worst and least efficient among the four algorithms, both time complexity and space(memory) usage are high. Since the algorithm needs to expand a significant number of nodes exponentially to find the optimal solution and designed to return only one answer, it runs out of memory from start12.

IDS: Its memory usage is much more efficient than USC since it would not need to store all the paths to find the most optimal one. However, it still requires a large nodes expansion and repeated work has been done, so the time usage is inefficient, and it runs out of time at start30 and start40.

A*: informed search, time-efficient, not very space-efficient. Compares to USC and IDS, it is much more time-efficient. A* algorithm has combined the advantage of time-efficient from greedy(heuristic) and disadvantage of space inefficient from USC. Although it's already much more space-efficient than USC and required a smaller number of nodes expansion, it still needs to store the previous result of g(n) and the estimated cost h(n), and it will run out of memory eventually if the search is too complex.

IDA*: Most efficient among the four algorithms, time usage efficient and memory usage efficient. Although it gives up some time efficiency compares to the A* for the memory efficiency, but its speed would not be affected too much. It combines the advantage of space-efficient from IDS and time-efficient from A*.

Question 2: Heuristic Path Search for 15-Puzzle

(a)

	start50		start60		start64	
IDA*	**50**	**14642512**	**60**	**321252368**	**64**	**1209086782**
1.2	**52**	**191438**	**62**	**230861**	**66**	**431033**
1.4	**66**	**116342**	**82**	**4432**	**94**	**190278**
1.6	**100**	**33504**	**148**	**55626**	**162**	**235848**
Greedy	**164**	**5447**	**166**	**1617**	**184**	**2174**

(b) $f(n) = (2 - w) * g(n) + w * h(n) \; where \; 0 \leq w \leq 2.$

When $w = 1.2 \rightarrow f(n) = 0.8 * g(n) + 1.2 * h(n).$

Section of code that would be changed.

```
nb_setval(counter, N1),
% write(Node),nl,   % print nodes as they are expanded
s(Node, Node1, C),
not(member(Node1, Path)),      % Prevent a cycle
G1 is G + C,
h(Node1, H1),
F1 is G1 + H1,
F1 =< F_limit,
depthlim([Node|Path], Node1, G1, F_limit, Sol, G2).
```

Replacement code

```
% Keep searching until goal is found, or F_limit is exceeded.
depthlim(Path, Node, G, F_limit, Sol, G2)  :-
    nb_getval(counter, N),
    N1 is N + 1,
    nb_setval(counter, N1),
    % write(Node),nl,   % print nodes as they are expanded
    s(Node, Node1, C),
    not(member(Node1, Path)),      % Prevent a cycle
    G1 is G + C,
    h(Node1, H1),
    F1 is 0.8 * G1 + 1.2 * H1,
    F1 =< F_limit,
    depthlim([Node|Path], Node1, G1, F_limit, Sol, G2).
```

(d)

All these five algorithms can be defined by $f(n) = (2-w) * g(n) + w * h(n)$. For the algorithm IDA*, w = 1 and for greedy w = 2, and for the rest three algorithms in the middle 1 < w < 2.

IDA* can be used to find the optimal path(high quality solution) but it takes a long time (lower speed). Greedy method is very time efficient, but it cannot guarantee an optimal solution, thus it has a lower quality of solution. And the algorithms in the middle run faster than IDA*, and get better quality of solutions than greedy.

From the table above we know that when w increase, the length of the path G increase but the total number of nodes expanded N decrease. Thus, we know that algorithms can trade off the quality of solutions (longer path) for their speed (run faster), and vice versa.