COMP3411/9414/9814 Artificial Intelligence
Session 1, 2018 Assignment 2 – Heuristics and Search

Yichen Zhu z5098663

Question 1: Search Algorithms for the 15-Puzzle

(a)

algorithm	start10	start12	start20	start30	start40
UCS	2565	Mem	Mem	Mem	Mem
IDS	2407	13812	5297410	Time	Time
A∗	33	26	915	Mem	Mem
IDA∗	29	21	952	17297	112571

(b)

UCS needs much more space than other algorithms, IDS needs a bit less than UCS. A∗ needs much less space than UCS and IDS. IDA∗ has the advantages of A∗ and IDS, also with less needing in space, which makes it the most efficient algorithm in 4.

IDA∗ and A∗ algorithm has nearly same time efficiency while IDS and UCS algorithms are slower. According to the time efficiency, UCS and IDS should be slower with its time complexity $O(b^{\lceil C^*/\varepsilon \rceil})$ and $O(b^d)$, then its A∗ and IDA∗.

Therefore, in this scenario, IDA∗ is most efficient and then A∗, IDS and UCS.

Question 2: Heuristic Path Search for 15-Puzzle (2 marks)

(a)&(c)

	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	140278
1.6	100	33504	148	55626	162	235848
Greedy	164	5447	166	1617	184	2174

(b)

Under the depthlim function, I change the F1 formula.

```
F1 is 0.8*G1 + 1.2*H1,
```

```
% Otherwise, use Prolog backtracking to explore all successors
% of the current node, in the order returned by s.
% 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),

        % it was F1 is G1 + H1,
        F1 is 0.8*G1 + 1.2*H1,

        F1 =< F_limit,
        depthlim([Node|Path], Node1, G1, F_limit, Sol, G2).
```

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

According to the formula , $F(n)=G(n)+H(n)$. IDA* equals to w = 1 and greedy equals to w = 2. As we can see, as w value increases, which represents closer to greedy algorithm, G grows and N decrease, which means length of path grows and state number decrease.
Though faster Greedy provides, it can not guarantee optimal, that's why solution path becomes larger, quality is not as good as IDA*. IDA* is optimal but much slower.
Therefore, we have w increase, with faster speed but worse solution quality.

Question 3: Maze Search Heuristics (2 marks)

(a)

$$h(x, y, x_G, y_G) = |x - x_g| + |y - y_G|$$

There I use Manhattan Distance heuristic because it dominates Straight-Line-Distance heuristic. we start from very left top of and goal is very right bottom, if there's not obstacles which enforcing you to take more walk, this Manhattan distance is equal cost.
If there's obstacles, Manhattan distance will be smaller than cost.
Therefore Manhattan Distance heuristic never overestimates cost, this heuristic is admissible.
And under this scenario, Manhattan distance is always larger or equal to Euclidean Distance, which is Straight-Line-Distance.
Therefore it dominates.

(b)
(i)
Straight-Line-Distance heuristic is not admissible under this scenario.
Consider from (0,0) to (1,1). This heuristic would be sqrt(2), but if move diagonally, actual cost would be 1. Therefore it overestimate the cost.

(ii)
My heuristic is not admissible under this scenario.
Consider from (0,0) to (1,1). My heuristic would be 2, but if move digonally, actual cost would be 1. Therefore it overestimate the cost.

(iii)

$$h(x, y, x_G, y_G) = \max(|x - x_g| + |y - y_G|)$$

My new formula considers diagonal move and up and down move.
If we assume $|y - y_G|$ is larger than $|x - x_g|$.
This heuristic formula is equal to $|x - x_g|$ + $|y - y_G - (x - x_g)|$ .which can be represented as move diagonal first and then move straight up or down.
For example, we move from (0,0) to (3,5).
We first choose to move from (0,0) to (3,3) then (3,3) to (3,5).
This heuristic is admissible because the cost can’t be lower than cost of one direction(x or y).

Question 4: Graph Paper Grand Prix (4 marks)

(a)

1: + -
2: + 〇-
3: + 〇〇-
4: + + - -
5: + + - 〇-
6: + + 〇- -
7: + + 〇-〇-
8: + + 〇〇- -
9: + + + - - -
10:+ + + - - 〇-
11:+ + + - 〇- -
12:+ + + 〇- - -
13:+ + + 〇- - 〇-
14:+ + + 〇- 〇- -
15:+ + + 〇〇- - -
16:+ + + + - - - -
17:+ + + + - - -〇-
18:+ + + + - - 〇- -
19:+ + + + - 〇- - -
20:+ + + + 〇- - - -
21:+ + + + 〇- - - 〇-

(b)

considering symmetric number of "+" and "-" situation, which can be seen as optimal solution for n1 scenario, $n1 = s^2$. (4: + + - -), s = 2
"+" traveling distance = s(s + 1)/2
"-" traveling distance = s(s + 1)/2
overall is s(s+1). and the M(n,0) is 2 ⋆ s because accelerate is 1.

And then we consider another scenario n2: (example: 5: + + - 〇-, 6: + + 〇- -)
$s^2<n2<=s(s+1)$
And we can easily find M(n2,0) is 2 ⋆ s + 1, which means we need one "stable" point to keep speed.

Then third scenario: (example: 7:+ + 〇-〇-, 8: + + 〇〇- -)
$s(s+1)^2<n3<(s+1)^2$
M(n3,0) is 2 ⋆ s + 2, which means we need 2 "stable" points to keep speed.
The reason we can't keep more than 2 "stable" points is that, under max velocity as k, all steps with 3 more points, can be done within 2 points.

Finally last scenario: (example: 9: + + + - - -)

N4 = $(s+1)^2$ It's pretty obvious as it's same as scenario n1. M(n4,0) is 2*(s+1) = 2s+2 same as n3.

Now that combine previous n1, n2, n3 and n4 scenarios, we have

$$\begin{aligned} &2s+1, \quad \text{if} \quad s^2 < n \le s(s+1) \\ &2s+2, \quad \text{if } s(s+1) < n \le (s+1)^2 \end{aligned}$$

By using this identity:

$$\lceil 2\sqrt{n} \rceil = \begin{cases} 2s+1, & \text{if} \quad s^2 < n \le s(s+1) \\ 2s+2, & \text{if } s(s+1) < n \le (s+1)^2 \end{cases}$$

We have $M(n,0) = \lceil 2\sqrt{n} \rceil$.

(c)

$n \ge \frac{1}{2}k(k-1)$ means M(n,0) max velocity is equal or larger than k.

Example: n = 9 v = 3. + + + - - -

Firstly, 1/2k(k+1) means the distance of keep acclerating for k steps. (k-1+k-2+··· 0 = 1/2k(k+1))

We have $\left\lceil 2\sqrt{n+\frac{1}{2}k(k+1)} \right\rceil = M(n+\frac{1}{2}k(k+1),0)$

Then we consider n + 1/2 k(k+1) as the overall distance because it can guarantee we have max velocity, which minimize overall steps.

$$\left\lceil 2\sqrt{n+\frac{1}{2}k(k+1)} \right\rceil - k$$

The last -k means reduce k steps. And from the end, going backward, before last k steps, the speed must be k because accelerate is 1, which makes it optimal for the overall distance.

Over all, it's like moving n + 1/2 k(k+1) distance, and decrease steps for last 1/2 k(k+1) distance by the end.

Therefore, we have $M(n,k) = \left\lceil 2\sqrt{n+\frac{1}{2}k(k+1)} \right\rceil - k$.

(d)

$$M(n,k) = \left\lceil 2\sqrt{\frac{k(k-1)}{2} - n} \right\rceil + k$$

$n \le \frac{1}{2}k(k-1)$ means M(n,0) max velocity is equal or smaller than k.

In this scenario, it has to travel opposite direction to get more space for acclerating and then stop, start accelerating, reaching speed at k and distance at n.

1. Travel $\frac{1}{2}k(k-1) - n$ distance opposite direction towards target and stop.

$$\left\lceil 2\sqrt{\frac{k(k-1)}{2}-n} \right\rceil$$

2. Travel $\frac{1}{2}k(k+1)$ + $\frac{1}{2}k(k-1)$ distance towards target and than exceeds target.

$$\left\lceil 2\sqrt{\frac{k(k+1)}{2}+\frac{k(k-1)}{2}} \right\rceil$$

3. reduce last k steps.

   -k

Finally we have:

$$M(n,k)=\left\lceil 2\sqrt{\frac{k(k-1)}{2}-n} \right\rceil+\left\lceil 2\sqrt{\frac{k(k+1)}{2}+\frac{k(k-1)}{2}} \right\rceil-k=\left\lceil 2\sqrt{\frac{k(k-1)}{2}-n} \right\rceil+k$$

(e)

Because we have $M(n,0)\geq M(n,k)$, which is obvious because max velocity difference.

$h(r,c,u,v,r_g,c_g)=\max(M(r_g-r,u),M(c_g-c,v))$