Homework2: Search - Written Component

Feng Gao (2015011208)

November 2018

Heuristics for n-puzzle

Here, we discuss some heuristics to be used with A^* graph search. Recall that heuristic $h_1(\cdot)$ returns the number of tiles that are in the wrong position and $h_2(\cdot)$ returns the sum of Manhattan distances of tiles from their goal positions. We introduce a third heuristic, $h_3(\cdot)$, that is the *minimum* number of moves necessary to get to the goal state if each action could move any tile to the blank slot. This is another *relaxed problem* heuristic.

Q1. Prove that h_3 is consistent (10')

Proof:

We assume that h_3 is not consistent, then there exist states A, C and G, where G is the goal state, s.t. $h_3(A) - h_3(C) > cost(A\ to\ C)$

In this case, we can get $h_3(A) > h_3(C) + cost(A\ to\ C)$, so the path, A through C to G, has less number of moves than $h_3(A)$ in contradiction with the definition of h_3.

Therefore, h_3 is consistent.

Q2. Prove or disprove: h_3 dominates h_1 (7')

Proof:

When we're proving that a dominates b, we're proving that for each x, $a(x) \geq b(x)$.

Sine h_1 returns the number of tiles that are in the wrong position, $h_3(\cdot)$ is the *minimum* number of moves necessary to get to the goal state if each action could move any tile to the blank slot, h_3 is equal to h_1 only if we can use one move for each tile in the wrong position to get to their goal states. Otherwise, we need use more than one moves to do it. Therefore, $h_3 \geq h_1$, that is, h_3 dominates h_1.

Q3. Prove or disprove: h_3 dominates h_2 (7')

Proof:

h_3 does not dominate h_2. To prove it, we need prove that there exists x, s.t. $h_3(x) < h_2(x)$.

As what we have proved in Q2, $h_3 \geq h_1$, and h_3 is equal to h_1 in some specific case. In such cases, we only need move tiles one step to get to their goal states, which are often not adjacent to themselves. Then we can get that $h_3(x) = h_1(x) < h_2(x)$ for some x.

Q4. Prove or disprove: the heuristic $h = max(h_2,\ h_3)$ is consistent (7')

Proof:

We have known that both h_2 and h_3 are consistent. Then,

$$
\begin{aligned}
h(n) &= max(h_2(n), h_3(n)) \\
&\leq max(h_2(n') + cost(n\ to\ n'), h_3(n') + cost(n\ to\ n')) \\
&\leq max(h_2(n'), h_3(n')) + cost(n\ to\ n') \\
&\leq h(n') + cost(n\ to\ n')
\end{aligned}
$$

Therefore, the heuristic $h = max(h_2,\ h_3)$ is also consistent.

Q5. An important feature of any heuristic is that it can be computed efficiently. Give a polynomial-time algorithm in the number of tiles to compute h_3 for a given state. Prove its correctness and running time (9')

Solution:

Without loss of generality, we can assume that the blank slot is not in the correct position. (Otherwise, we can move any tile in wrong position to the blank slot to make this condition true.)

We use remaining_tiles, a list, to note the tiles that are in the wrong position, whose element is number, and use goal(n) to note the goal position of the nth tile.

Algorithm 1 Polynomial-time algorithm in the number of tiles to compute h_3

```
h3 = 0
while length(remaining_tiles) > 0 do
    if the blank slot is in the correct position then
        Move remaining_tiles[0] to the blank slot.
    end if
    for n in remaining_tiles do
        if goal(n) is the blank slot then
            Move the nth tile to the blank slot.
            h3 = h3 + 1
            remaining_tiles.remove(n)
            break
        else
            continue
        end if
    end for
end while
return h3
```

Given $N = the\ number\ of\ tiles$, it's obvious that the running time of my algorithm is less than $O(n^2)$.