Assignment 1

Maxime CHAMBREUIL
maxime.chambreuil@mail.mcgill.ca

Contents

1 Exercise 1 : Uninformed search

a) Considering a node n, if his successor are *2n* and *2n+1*, we can say that we traverse a binary tree in width. So the portion of the state space from state 1 to state 15 is :

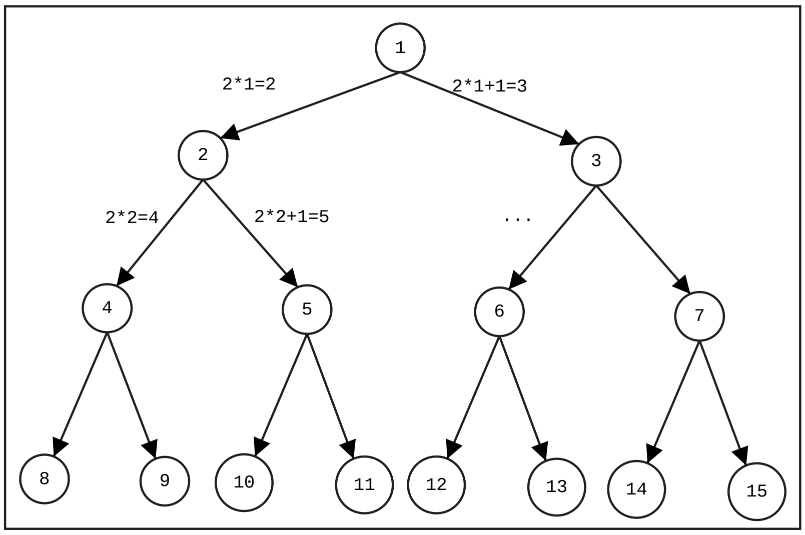


Figure 1: State tree from 1 to 15

b) We assume now, that the goal state is state 11. Here are the list of nodes that will be expanded during the process of differents algorithms :

- **Breadth-first search :** First, we enqueue 1 (Queue=1). Then we remove 1 from the queue, 1 is not a goal state. Therefore we had his children 2 and 3 to the end of the queue (Queue=2,3). After, we expand 2 and enqueue 4 and 5 (Queue=3,4,5). We go on this procedure until we remove 11 from the queue (Queue=12,13,14,15). 11 is the goal state, so we stop here to obtain the following list :

  1 - 2 - 3 - 4 - 5 - 6 - 7 - 8 - 9 - 10 - **11**

- **Depth-limited search (L=3) :** First, we enqueue 1 (Queue=1 and L=0). Then we remove 1 from the queue, 1 is not a goal state. Therefore we had his children 2 and 3 to the front of the queue (Queue=2,3 and L=1). After, we expand 2 and enqueue 4 and 5 (Queue=4,5,3 and L=2). Afterwards, we expand 4 and add 8 and 9 to the queue (Queue=8,9,5,3 and L=3). We expand 8 and 9, which are not the goal state, have no children and would make L greater than 3. So we then expand 5 (Queue=3 and L=2) and add 10 and 11 to the queue (Queue=10,11,3 and L=3). At this step, we can easily understand that the expanding list is :

  1 - 2 - 4 - 8 - 9 - 5 - 10 - **11**

- **Iterative deepening :** We have to repeat the previous procedure with :
  L = 0 : 1
  L = 1 : 1 - 2 - 3
  L = 2 : 1 - 2 - 4 - 5 - 3 - 6 - 7
  L = 3 : 1 - 2 - 4 - 8 - 9 - 5 - 10 - **11**

2 Exercise 2 : Uninformed/informed search

2.1 Optimality of iterative lengthening search (ILS)

Proof by contradiction : Assume that ILS returns G_{ret} so that $g(G_{ret}) > g(G_{opt})$. During an iteration i, $g(G_{opt})$ will be used as the limit and G_{opt} will be enqueued and found before G_{ret}.

Indeed, the algorithm use a limit function that keep increasing so G_{ret} will be discarded during the iteration i and $g(G_{ret})$ will be used as the limit after.

2.2 Uniform tree

During each iteration, one level will be discarded at the same time, because all nodes at this level has the same value. In the worst case, the goal state is at the level d and so, we will have to make d iterations to find it. The algorithm has a $O(d)$ complexity.

2.3 Step costs

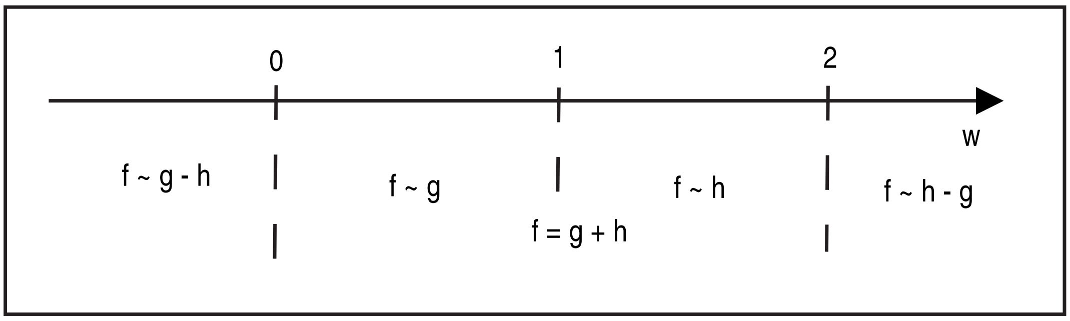
In the worst case, there is no 2 nodes, which have the same parent, with the same cost. So that the cost of each node will be used as the limit in the next iteration. As we have b^d nodes, the complexity is $O(b^d)$

3 Exercise 3 : Informed search

$$f(n) = (2 - w)g(n) + wh(n)$$

3.1 w

w is just regulating the weight of g(n) and h(n) in the f(n) function.



So f(n) is optimum for w = 1.

3.2 $w = 0$

$$f(n) = 2g(n)$$

So the algorithm is analog to an **uniform cost search**.

3.3 $w = 1$

$$f(n) = g(n) + h(n)$$

So the algorithm is analog to an **heuristic search**.

3.4 $w = 2$

$$f(n) = 2h(n)$$

So the algorithm is analog to a **best-first search**.

4 Exercise 4 : Heuristics

Let $h^*(n)$ be the shortest path from n to any goal state. We will consider the notation h^* as $h^*(n)$, whatever n. As h_1 and h_2 are admissible heuristics, we know that $h_1 < h^*$ and $h_2 < h^*$.

4.1 $h_1 + h_2$

So $h_1 + h_2 \leq 2 \times h^*$

Mathematically, we can conclude that $h_1 + h_2$ is admissible and is a better heuristic than h_1 or h_2, but only if $h_1 \leq h^* - h_2$. In the field of Artificial Intelligence, h^* is unknown (if it is, why not use it, as it is the best). Therefore we cannot test this condition and so, it is not an admissible heuristic whatever our initials heuristics.

4.2 $\frac{h_1+h_2}{2}$

We can pursue the previous calculus :

$$h_1 + h_2 \leq 2 \times h^*$$
$$\frac{h_1 + h_2}{2} \leq h^*$$

To conclude, we can say that $\frac{h_1+h_2}{2}$ is an admissible heuristic. We can easily understand geometrically that it is the center of h_1 and h_2. So it is just the mean of our 2 initials heuristics and does not provide a better heuristic.

4.3 $|h_1 - h_2|$

We know that :

$$0 \leq h_1 \leq h^*$$
$$0 \leq h_2 \leq h^*$$
$$-h^* \leq -h_2 \leq 0$$

So,

$$0 - h^* \leq h_1 - h_2 \leq h^* - 0$$
$$|h_1 - h_2| \leq h^*$$

As a conclusion, $|h_1 - h_2|$ is an admissible heuristic and represent the distance between our initials heuristics. This is not a useful heuristic, as we just have information about the relative position of h_1 and h_2. The maximum of these latters would be better.

4.4 $h_1 \times h_2$

$$h_1 \times h_2 \leq {h^*}^2$$

Then we can discuss the result depending on the value of each variable (admissible if $h_1 \leq \frac{{h^*}^2}{h_2}$). For the same reason as (a), we can conclude that it is not an admissible heuristic whatever h_1 and h_2.

4.5 $2 \times h_1$

$$2 \times h_1 \leq 2 \times h^*$$

So it is a better heuristic in the mathematical sens if $h_1 \leq \frac{h^*}{2}$, but it is not in the AI sens (Refer to (a) and (d)).

4.6 $\frac{h_1}{2}$

$$\frac{h_1}{2} \leq \frac{h^*}{2} \leq h^*$$

So $\frac{h_1}{2}$ is an admissible heuristic but is not useful as it takes us further from h^*

4.7 min(h_1,h_2)

min(h_1,h_2) is h_1 or h_2, so min(h_1,h_2) is an admissible heuristic but it is not useful as it does not create a new better heuristic and take the worst of the ones we already have.

4.8 max(h_1,h_2)

max(h_1,h_2) is h_1 or h_2, so max(h_1,h_2) is an admissible heuristic but it is not useful as it does not create a new better heuristic. Fortunately, it takes the best of the ones we already have.

4.9 $h_1 + 2h_2$

$$h_1 + 2h_2 \leq 3h^*$$

It is the same problem as in 4.1, 4.4 and 4.5 : The admissibility of h_1+2h_2 is up to some stronger conditions ($h_1 \leq h^* - h_2$).

4.10 min(h_1,$2h_2$)

$$min(h_1, 2h_2) = \begin{cases} h_1 & \rightarrow \text{admissible} \\ 2h_2 \leq h_1 \leq h^* & \rightarrow \text{admissible} \end{cases}$$

So min(h_1,$2h_2$) is an admissible heuristic which can create a new heuristic $2h_2$. Unfortunately, h_1 remains a better choice. This operation is useless in our research of a better heuristics by combining 2 known ones.

5 Exercise 5 : Genetic algorithms

5.1 The fitness function

The fitness for a chess player can be the number of won match.

5.2 Player representation

A simple representation can be a list of symbols : 0 for a defeat and 1 for a victory.

5.3 Operators

The crossover would be adding a symbol to the string (a new match is played between 2 individuals) and the mutation would be permuting 2 symbols.

6 Exercise 6 : Search/constraint satisfaction

6.1 General Search Problem

- **State space :** Grid with blanks, letters and shaded cases.
- **Operator :** Assign a letter to a blank case.
- **Goal state :** Test if all words in the grid are in the dictionnary.
- **Heuristic :** Inverse of number of possible words in the dictionnary, which can complete the grid.

6.2 Constraint Satisfaction Problem

- **Variables :** Set of cases for a word.
- **Domain :** Dictionnary.
- **Constraints :** Words belongs to the dictionnary.

6.3 Conclusion

To determine the best representation, let us take a look at the complexity. We have a grid of n cases, m words in the dictionnary, the alphabet has 26 letters and we have to place p words in the grid. Concerning the general search problem, we have a complexity of $(26^n \times m)$. As for the constraint satisfaction problem, we have $(p \times m)$. So we can say that the constraint satisfaction problem is better that the general search problem.

7 Exercise 7 : Constraint satisfaction

7.1 First representation : Gathering

- **Variables :** Houses.
- **Domain :** A set, where an item is composed with a color, a nationality, a cigarette, a drink and a pet.
- **Constraints :** Given by the exercise.

7.2 Second representation : Separating

We can consider the problem with 5 different domains : color, nationality, cigarette, drink and pet. For colors, the problem would be :

- **Variables :** Houses.
- **Domain :** red, yellow, blue, green, ivory.
- **Constraints :** Given by the exercise.

I think that the separating method is better, because as soon as we have assigned colors to houses, we have information for the other domains and so the problem is solved quicker than by testing everything in the same time.