McGill University

# Assignment 2

Nabil Chowdhury
ID: 260622155
COMP 424

## 1 Searching Under Uncertainty

### 1.1 Total number of possible belief states

The total number of belief states is $2^N - 1$ where $N$ is the number of physical states. Assuming we consider the following:

i) Only one individual
ii) We can be anywhere except G

We have a total of $N = 25$ physical states. Therefore the total number of belief states is $\mathbf{2^{25} - 1 = 33554431}$.

### 1.2 Number of distinct percepts

The different percepts, identified by the orientation of the walls, are as follows:

i) east-west-south
ii) east-west-north
iii) east-west
iv) north-south
v) east-north-south
vi) west-north-south
vii) no walls

Therefore we have a total of **7 distinct percepts**.

### 1.3 Unique percepts

The distinct percepts from part 1.2 can indicate, with the following probabilities where a player is on the map:

| Percept | Coordinates | Probability |
|---|---|---|
| east-west-south | (2, 2) (4, 1), (6, 1), (8, 1), (10, 2) | 1/5 |
| east-west-north | (2, 4) (4, 5), (6, 4), (8, 5), (10, 4) | 1/5 |
| east-west | (4, 2) (4, 4), (6, 2), (8, 2), (8, 4) | 1/5 |
| north-south | (3, 3) (5, 3), (7, 3), (9, 3) | 1/4 |
| east-north-south | (1, 3) | 1 |
| west-north-south | (11, 3) | 1 |
| no walls | (2, 3) (4, 3), (6, 3), (8, 3), (10, 3) | 1/5 |

East-north-south and west-north-south guarantee that the player is either on the leftmost or rightmost square on the map respectively. North-south has the second highest probability of where the player is. The other four percepts have equal probability.

### 1.4 Conformant plan to save Hopper

There is a conformant plan that can save Hopper. Assuming we want whichever players start at entrances 1 and 2 to reach Hopper, we can take the following steps:

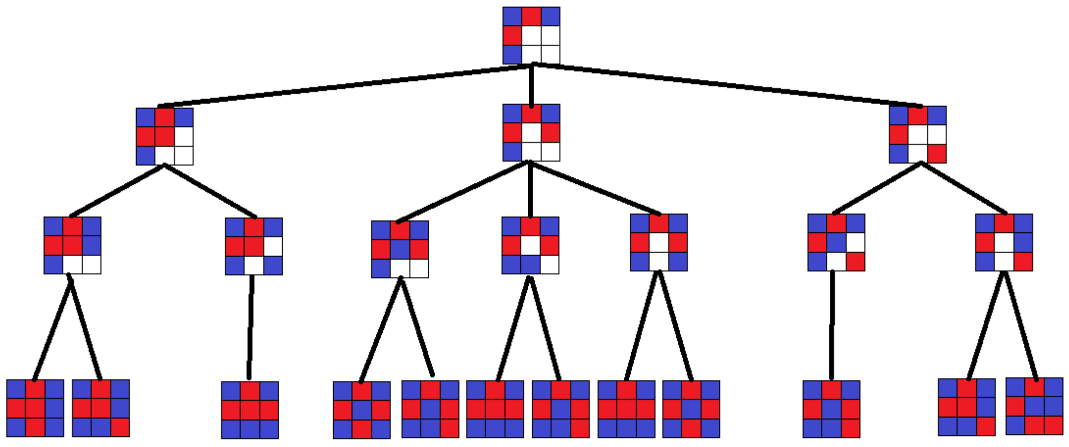
$$\Rightarrow N \Rightarrow E \Rightarrow N(\Rightarrow E)^2 \Rightarrow S \Rightarrow E(\Rightarrow S)^2$$

This takes 45 minutes for both players to reach Hopper.

## 2 Game Playing

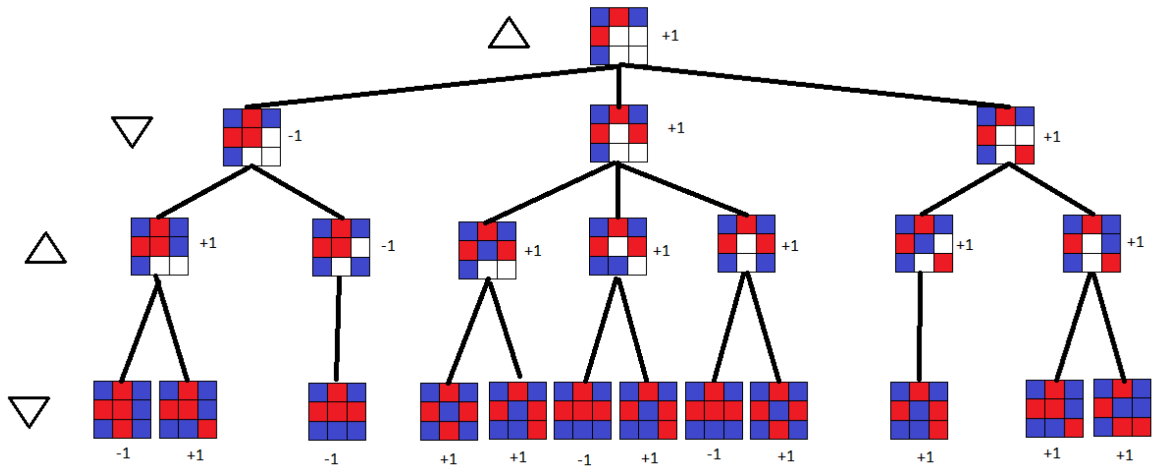
### 2.1 Set of game states

Here is the tree representing all states (note: sibling states that are symmetrically equivalent have been omitted):



### 2.2 Applying the Minimax Algorithm

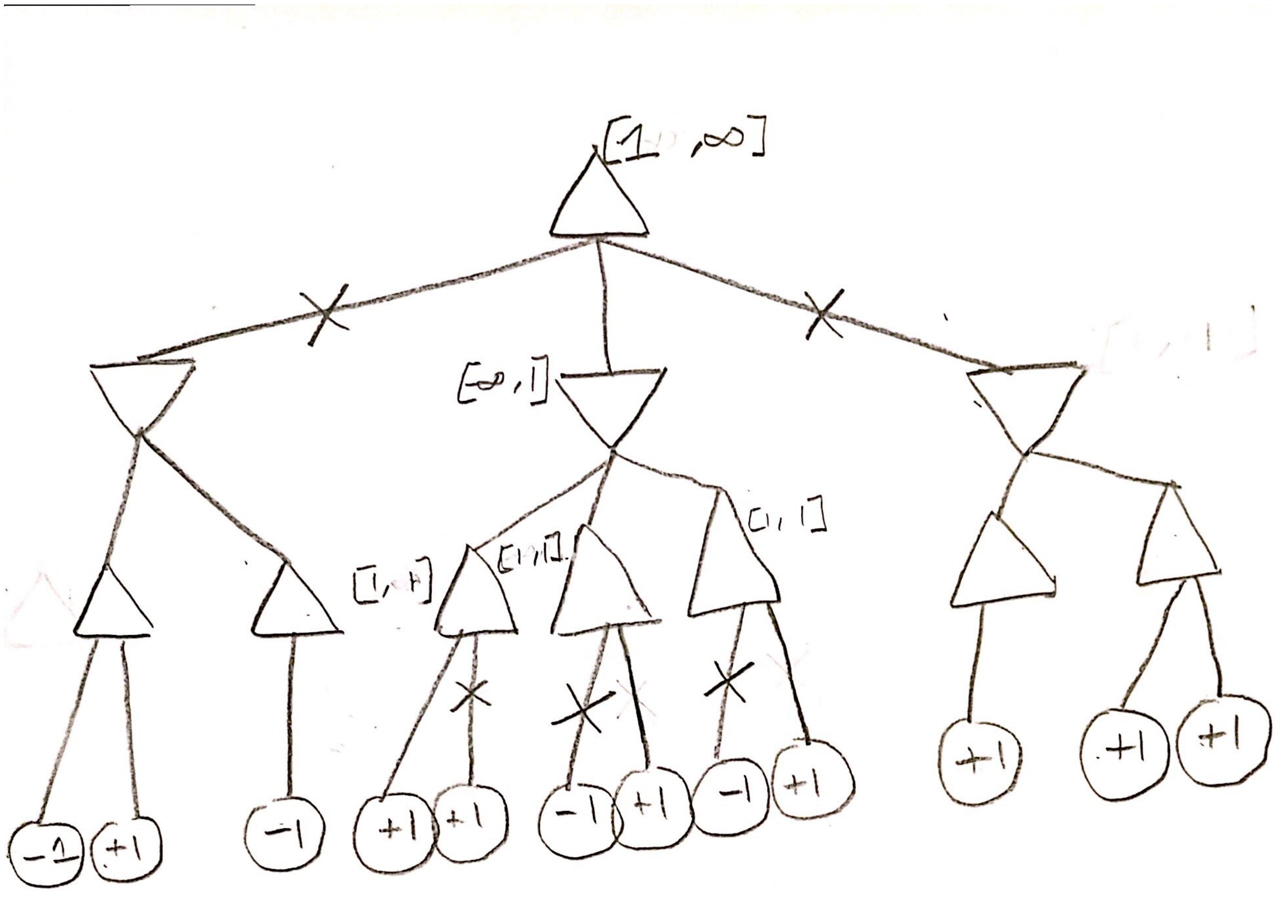
Here is the diagram of the minimax algorithm applied to the game:



The minimax diagram shows that Hopper can indeed be saved. +1 denotes winning states while -1 denotes losing states. One surefire way to always win is to simply place the red box in the bottom right corner, as all end states resulting from that move are winning states (see third branch of tree).

## 2.3 Alpha-beta pruning

### 2.3.1 Applying alpha-beta pruning



This drawing corresponds to the minimax diagram in 2.2. Assuming we go down the branch where the first move of red is 2nd row 3rd column, we can see that this branch lets the player win. We can immediately prune the other two branches as we have already found the winning path.

### 2.3.2 Time saved in terms of number of nodes pruned

For this game, we can use the fact that since the only evaluation values are +1 (win) and -1 (lose), if a max node's value is already +1, there is no need to explore its other children. For example in the diagram, we go from the root, down to the middle node, then the left child. The left child of this node is +1, and since it is a max node, this is the best value it can ever have, so we do not need to explore the right child. However, we must explore its other two siblings since the parent is a min node, and it could be -1. Assuming best case scenario, we have pruned 15 nodes.

### 2.3.3 Is this the best possible time achieved?

No. If we went down the branch where the first move of red is bottom right corner (3rd row, 3rd column), we would have pruned out 17 nodes (the first two branches of the tree, and one leaf node from the last branch).

# 3 Propositional Logic

## 3.1 Number of models that satisfy the following:

a) $2^2 - 1 = 3$. This is False only when A and B are both False.

b) $2^5 - 1 = 31$. This is False only when A, B, C, D, E are all False.

c) 5. If C is True, we have 4 combinations for A and B since they can be anything. If C is False, we have exactly one combination (A and B must both be True), so total of 5.

d) 0. This simplifies to $A \wedge B \wedge \neg B$, which is always false.

e) $2^6 = 64$. Using DeMorgan's law, this simplifies to $\neg(A \wedge D \wedge E \wedge F) \vee \neg(B \wedge C) \vee (B \wedge C)$, which is always true since $\neg(B \wedge C) \vee (B \wedge C)$ is always true.

## 3.2 State if Valid, Unsatisfiable or Satisfiable

a) Valid. If $A$ is True, then $\neg A$ is False, so the statement is True. If $A$ is False, then $\neg A$ is True, so the statement is still true.

b) Satisfiable. $A \wedge \neg A$ is always False, so B determines the outcome.

c) Valid. Here is the simplification:

$$(((A \Rightarrow B) \wedge A) \Rightarrow B) \Leftrightarrow (B \vee \neg B)$$

$$(((\neg A \vee B) \wedge A) \Rightarrow B) \Leftrightarrow T$$

$$((\neg A \wedge A \vee B \wedge A) \Rightarrow B) \Leftrightarrow T$$

$$((B \wedge A) \Rightarrow B) \Leftrightarrow T$$

$$(\neg(B \wedge A) \vee B) \Leftrightarrow T$$

$$(\neg B \vee \neg A \vee B) \Leftrightarrow T$$

$$T \Leftrightarrow T$$

d) Valid. We have False $|=$ True (since $A \wedge \neg A$ is always True). False $|=$ True is valid by definition.

e) Unsatisfiable. We have True $|=$ False, which is unsatisfiable by definition. The right hand side simplifies to the following:

$$(A \vee B) \wedge \neg A \wedge \neg B$$

$$(A \wedge \neg A \vee B \wedge \neg A) \wedge \neg B$$

$$B \wedge \neg A \wedge \neg B$$

$$False$$

## 4 First Order Logic

4.1

$C = \{$Dusty, Elody, Michael, William$\}$

$S = \{$Eggo, pudding, 3musketeers$\}$

i) $\forall x \in C \; \exists i \in S$ Bought$(x, i)$ (variables: $x, i$ )

ii) $\forall x \in C$ Bought$(x,$ pudding$) \rightarrow \neg$ Bought$(x,$ eggos$)$
(variables: $x$ , constants: pudding, eggos)

iii) $\forall x \in C$ Bought$(x,$ 3 musketeers$) \rightarrow$ Bought$(x,$ pudding$)$
(Variables: $x$ , constants: 3musk, pudding

~~iv)~~

v) $\forall i \in S$ Bought(Michael, $i$) $\rightarrow \neg$ Bought(Elody, $i$)
(var: $i$ , constants: Michael, Elody)

v) Bought(Michael, 3-musketeers) (constants: Michael, 3musk)

vi) Bought(Dusty, 3-musketeers) (constants: Dusty, 3musk)

4.2

i) Bought$(x, i)$

ii) $\neg$Bought$(x,$ pudding$) \vee \neg$ Bought$(x,$ eggos$)$

iii) $\neg$Bought$(x,$ 3musk$) \vee$ Bought$(x,$ pudding$)$

iv) $\neg$ Bought(Michael, $i$) $\vee \neg$ Bought (Elody, $i$)

v) Bought(Michael, 3musk)

vi) Bought(Dusty, 3musk)

4.3

→ statement $\exists_x$ Bought(x, Eggos) $\wedge \neg$ Bought(x, pudding) $\wedge \neg$ Bought (x, 3musk)

prove negation unsatisfiable

$\Rightarrow$ $\neg$ Bought(x, eggos) $\vee$ Bought(x, pudding) $\vee$ Bought(x, 3musk)

→ michael bought 3musk

→ since michael bought 3-musk he also bought pudding.

→ same with Dustin, so michael and dustin cannot disprove the statement.

→ Elody is <u>MAD</u> and does not buy what michael buys, so she can only buy eggos.

→ She buys exactly eggos due to constraint (iv) and (i)