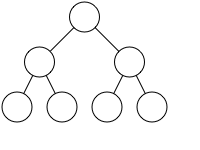
**Action visie: Pylos**



Hs

a2

a4

a3

a5

a6

a1

Action

-sphere

-location

-state

-color

1. Huidige state (wij zijn aan zet)

Doel: De volgende move doen, alle mogelijke acties hiervoor aanmaken

Mogelijkheden:

Er kan eender welke reserve bal geplaatst worden op alle mogelijke locaties

Er kan een reeds gezette bal verplaatst worden naar een hogere locatie op het veld

2. Niveau 1 (a1;a2)

Doel: Rekening houdend met de vorige zet, moeten alle mogelijke moves gedaan worden (ongeacht de huidige state van de board -> later nog verbeteren als er een betere uitkomst is in een reeds uitgewerkte tak dan dat we ooit in de niet uitgewerkte takken kunnen hebben). Hiervoor bijhorende acties aanmaken.

Mogelijkheden:

Tegenstander aan zet:

Er kan eender welke reserve bal geplaatst worden op alle mogelijke locaties

Er kan een reeds gezette bal verplaatst worden naar een hogere locatie op het veld

Wij aan zet (vierkant gemaakt) (na elke move checken of er een vierkant is gemaakt, enkel daarna extra kijken voor remove mogelijkheden):

Er kan 1 bal worden weggenomen (1 á 4 mogelijkheden)

Er kan gepast worden

3. Niveau 2 (a3;a4;a5;a6)

Doel : Door middel van een evaluatiefunctie bepalen welke actie het beste is en deze uitvoeren

Oplossing: Beste actie sequentie opslaan, evaluatiefunctie uitvinden die board states kan evalueren.

**Pseudocode:**

Static int Depth = 3

Sim = new simulator()

Stack<Action> currentActionSequence = new Stack

Stack<Action> bestActionSequence = new Stack

Static Int bestScore = 0

bestActionSequence = findBestMove(currentDepth, sim, currentActionSequence, bestActionSequence, board, game, this.color)

**findBestMove:**

ArrayList<Action> possibleActions = new ArrayList //used for each level (gets reset every time)

If(game.state == MOVE){ //Gaat altijd het geval zijn bij ons

For(allAvailableLocations : l){

If(player.hasReserves){ possibleActions.add(new Action(reserve, l, MOVE, color))}

If( l.Y>1 ){

For(allLocations : l2 < l.Y) {

If(l2.IsUsed && l2.ball == color)possibleActions.add(new Action(l2.ball, l, MOVE, color)

}

if(currentDepth < depth){

For(allActions : a){

currentActionSequence.add(a)

sim.moveSphere(a…)

if(vierkantAanwezig){

maak remove acties (1 a 4 \* 1 a 3 -> max 12) en werk ze uit zonder recursie

}

Else{ bestActionSequence = findBestMove(currentDepth +1, sim, currAckSeq, bestAck, board, game, this.color)}

Else{

Int score = evaluateBoardState(state)

If(score>bestScore){

bestScore = score

bestActionSequence = currentActionSequence.copy()

}

Sim.undo()???

Return bestActionSequence