**Introduction to Artificial Intelligence**

**Exercise 3: 2048**

Amir Tuboul, Gal Getz

**Question 4 (2 points)**

**In the previous question, you implemented an alpha-beta agent that makes minimax search more efficient using pruning. In this question, we will deal with the option of pruning for an agent that also computes expectations.**

**This two-player game with chance is given in the following figure:**

**Question 4 .1**

תמונה שמכילה שרטוט, תרשים, ציור, קו

התיאור נוצר באופן אוטומטי

**Question 4.2**

**Given the values of the first six leaves, do we need to evaluate the seventh and eighth leaves? Given the values of the first seven leaves, do we need to evaluate the eighth leaf? Explain your answers**

Given the value of the first six leaves we still need to evaluate the last node. In this case we chose the left sub tree of this search tree. Suppose the min value of the last node is 100, the expected value in the right subtree would be 100.5 so the maximizing player will choose this direction over the left subtree. This means the value of the last node can always make an impact regarding the path choice thus we can’t skip evaluating it (meaning evaluating the seventh and eight leaf).

Given the values of the first seven leaves, we don’t to evaluate the eighth because we know that the minimum of the last node is smaller than -1 so the expected value of the right subtree is less the -0.5 so no matter what the value of the eighth leaf, we always choose the left subtree.

**Question 4.3**

**Suppose the leaf node values are known to lie between –2 and 2 inclusive. After the first two leaves are evaluated, what is the value range for the left-hand chance node?**

The value range is between 0 to 2.

**Question 4.4**

**Given the assumption of (3), and assuming we keep evaluating the leaves left to right until we know what is the best action for the max player, after which leave we can stop and return the answer?**

We can stop evaluating at the fifth node. The left-hand chance node expected value is 1.5. When we get to the fifth leaf we can calculate the value range of the right-hand chance node. Since the value of the fifth leaf is 0 the maximum value that the right-hand chance node can get is 1 (putting 2 in the remaining leaves and calculating the expected value accordingly), meaning the left-hand chance node will always be greater thus the maximizing player will always choose it.

**Question 6**

**In the previous questions you implemented minimax (with and without pruning) and expectimax agents. In this question, you will compare these adversarial search algorithms theoretically and empirically. You will do this with respect to the game 2048.**

**Success in the game (3 points)**

**Theoretically, which of the algorithms is more suitable for the game 2048? Explain what each of the algorithms assumes about the opponent and which of the assumptions better suits the game.**

**Minimax Algorithm**

* **Assumption:** The opponent plays optimally to minimize the player's score.
* **Suitability for 2048:** Not suitable because 2048 involves random tile placements, not a strategic opponent.

**Expectimax Algorithm**

* **Assumption:** Events are random, modeled with probabilities.
* **Suitability for 2048:** More suitable because 2048's tile generation is random, and Expectimax can handle this randomness effectively.

Expectimax is more suitable for 2048 because it models the game's random tile placement accurately, unlike Minimax which assumes an optimal adversarial opponent.

**Empirically, which of the algorithms is more successful in 2048? To answer the question, run each of the algorithms for 10 games and display the scores and the largest tiles that each of the algorithms achieves.**

|  |  |  |
| --- | --- | --- |
|  | **Highest Score** | **Largest Tile** |
| **Minimax** | 7304 | 512 |
| **Expectimax** | 15252 | 1024 |

Minimax performance:

minmax\_scores <- c(7184, 6924, 6524, 3212, 7032, 6928, 7048, 7068, 7305, 2668)

minmax\_highest\_tiles <- c(512, 512, 512, 256, 512, 512, 512, 512, 512, 256)

Expectimax performance:

expectimax\_scores <- c(12312, 5696, 14140, 7172, 7208, 7072, 7116, 6948, 12196, 15252)

expectimax\_highest\_tiles <- c(1024, 512, 1024, 512, 512, 512, 512, 512, 1024, 1024)

תמונה שמכילה טקסט, צילום מסך, תרשים, עלילה

התיאור נוצר באופן אוטומטי תמונה שמכילה טקסט, צילום מסך, תרשים, עלילה

התיאור נוצר באופן אוטומטי

**Are the theoretical and empirical results consistent with each other? If so, explain. If not, explain how this is possible.**

**Based on the data provided, the theoretical and empirical results are consistent with each other. Here's an explanation:**

The Expectimax method theoretically outperforms the MinMax method by better predicting the potential outcomes of moves in complex game scenarios. Empirically, this is reflected in the data where the Expectimax method achieves significantly higher mean scores (9891.2) compared to the MinMax method (6029.3). Additionally, the highest tiles reached by Expectimax (mean of 768.0) are also greater than those reached by MinMax (mean of 460.8). The consistency between the theoretical advantage of Expectimax and the observed empirical data demonstrates that Expectimax effectively leads to better performance in practice, corroborating the theoretical expectations.

**Compare the standard deviations in the score and the largest tile of the two algorithms. Give an intuitive explanation for the difference in results.**

The Expectimax method exhibits higher variability in both scores and highest tiles compared to the MinMax method. This variability is reflected in the larger standard deviations for Expectimax. While Expectimax can achieve higher performance (as indicated by higher mean values), it also shows more inconsistency. In contrast, the MinMax method, with lower standard deviations, indicates more consistent but lower performance.

The MinMax method is less inconsistent because it uses a deterministic approach, evaluating the worst-case scenarios for each possible move. This approach leads to more predictable and stable outcomes, but it may not always capitalize on opportunities for higher performance. The method focuses on minimizing potential losses rather than maximizing potential gains, resulting in more consistent but generally lower scores and tile values.

On the other hand, the Expectimax method incorporates probabilistic elements, considering both the expected rewards and risks associated with different moves. This approach can lead to higher scores and tile values by effectively navigating complex scenarios and capitalizing on opportunities. However, the probabilistic nature introduces greater variability in the outcomes, leading to higher standard deviations. The method's ability to take calculated risks results in a broader range of performance, from very high scores to occasional lower ones.

**Alternative games (2 points)**

**Imagine an alternative game "2048 Boom" where after every move of the player there is a one in a billion chance of a "boom" where the game stops and the player finishes with a score of 0. How will the change affect the performance of the algorithms? In this section only, assume an exact expectimax algorithm that takes into account the probabilities of the moves.**

Exact Expectimax Algorithm: The Expectimax algorithm, while acknowledging the "boom" event in its calculations, remains robust. Its probabilistic approach allows it to balance risks effectively and prioritize moves that maximize scores and highest tiles, ensuring competitive performance despite the negligible risk of the "boom" event. Since the probability of the boom event is so low it won’t have a significant effect on the calculation of the expected value for the chance nodes.

MinMax Algorithm: The MinMax algorithm's performance is severely impacted by the introduction of the "boom" event. As it evaluates each move under the assumption of minimizing the worst-case scenario without considering its probability, the algorithm consistently predicts a zero score outcome, mirroring the assumption that the minimizing player will always choose that option.

**In our implementation the expectimax player does not calculate the exact expectation but assumes that all outcomes have equal probability. Use this fact to create a new simple game where the minimax player outperforms this not-exactly-expectimax player.**

**New Game: Deterministic 2048**

**Grid Setup:** Play on a 4x4.

**Tile Placement:** Each turn, a new tile (2 or 4) appears based on a deterministic strategy.

Strategy places tiles to minimize score potential for the maximizing player. In simple words the board always act according to the worst-case scenario for the maximizing player.

**Minimax Advantage Over Expectimax:**

Minimax, being deterministic and assuming the worst-case scenario for tile placement (opponent's move), aligns precisely with the strategic demands of the new games proposed. In contrast, Expectimax's assumption of equal probability for each tile placement does not reflect the strategic constraints and deterministic nature of these game, thereby limiting its effectiveness in comparison.