Bounded Suboptimal  
N-Player Game Tree Search

Experiments analysis

Advisor: Dor Atzmon

Asi Zaks 302329693

Shachar Meretz 208240275

Peleg Biton 203842703

Omer Nagar 307937714

**Part One -** Random trees experiment

Examine the trend of the Expansion Rate in relation to Epsilon change:

* Compare our Bounded Suboptimal algorithm’s Expansion Rate with Shallow algorithm (Preforms optimal pruning) in relation to Epsilon change for different depths.
* To compare the different between the final scores of the algorithms to examine the distancing from optimal score in each simulation.

**Tasks –**

* Bounded Suboptimal algorithm implementation.
* Shallow algorithm implementation.
* Random Tree Generator Implementation.
* Game simulation on random tree Implementation.

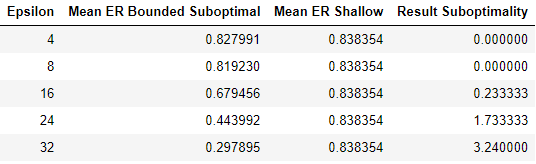
**The Experiment -**

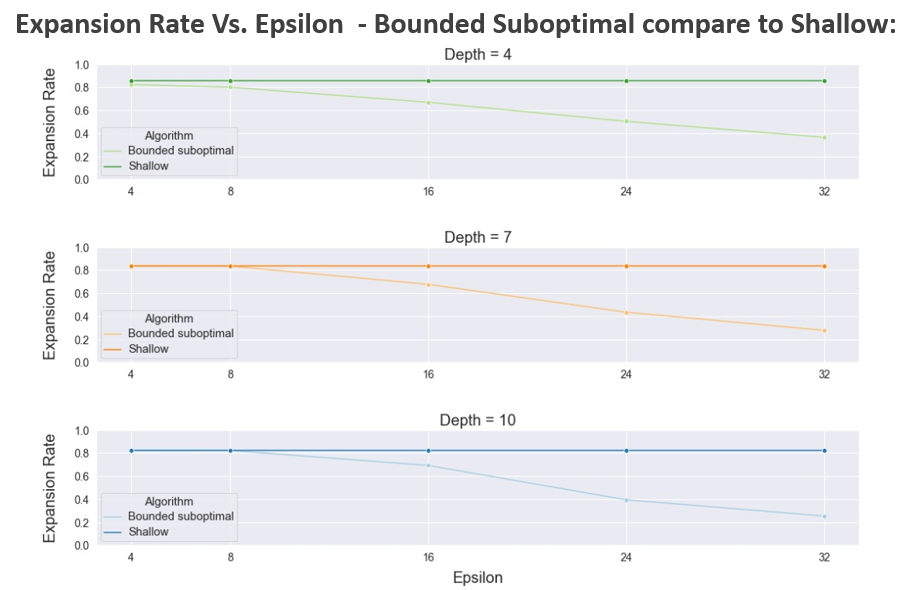
* we examine 3 different depths of random trees – 4, 7, 10.
* 50 different random trees were generated for each examined depth.
* we have created 2 groups of 3 players each:
  + Bounded Suboptimal, Shallow, Shallow.
  + Shallow, Shallow, Shallow.
* For each tree in each examined depth, we have simulated 2 games, 1 with each group to enable comparation between Bounded Suboptimal to Shallow.

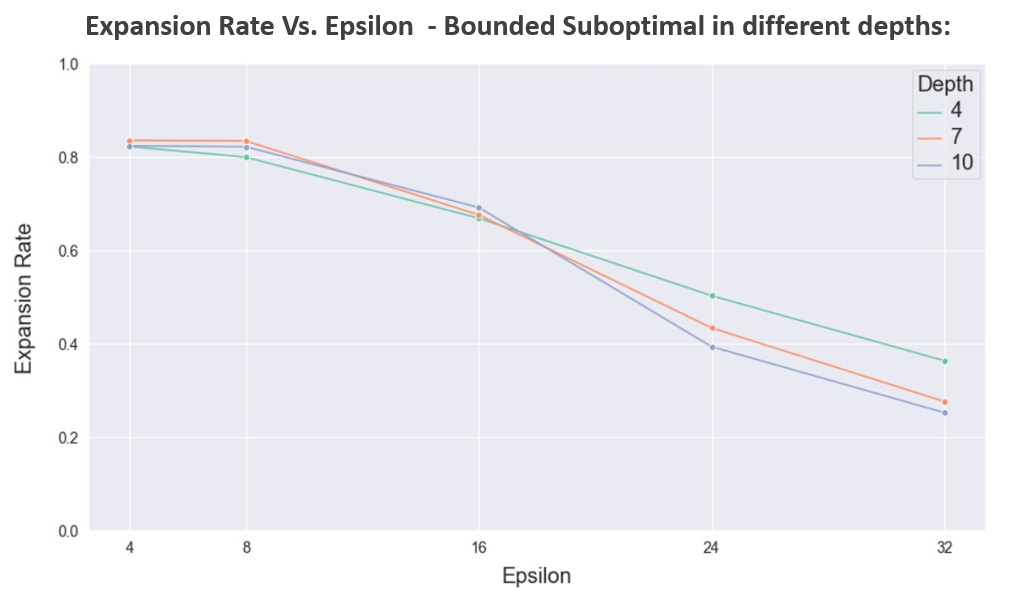
**Hypotheses –**

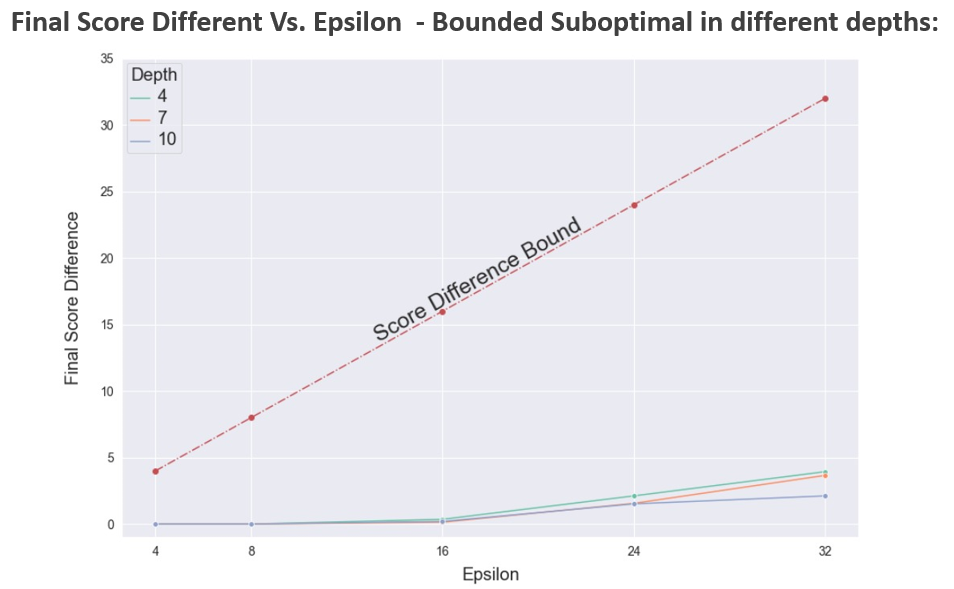
* Bounded Suboptimal algorithm final score high limit is the Shallow algorithm final score (Shallow is optimal).
* Find strong relation between Expansion Rate to Epsilon and depth :
  + As Epsilon grow up Expansion Rate become smaller.
  + As depth grow up Expansion Rate become smaller.
* Find weak relation between score different to Epsilon and depth :
  + As Epsilon grow up score different not grow up significantly.
  + As depth grow up score different not grow up significantly.

**Experiment’s results –**

****



****

****

**Conclusions –**

* Found strong relation between Expansion Rate to Epsilon and depth :
  + As Epsilon grow up Expansion Rate become smaller.
  + As depth grow up Expansion Rate become smaller.
* Found weak relation between score different to Epsilon and depth :
  + As Epsilon grow up score different not grow up significantly.
  + As depth grow up score different not grow up significantly.

**Part Two -** Epsilon Optimization experiment

Find the optimal Epsilon of Bounded Suboptimal & Bounded Paranoid in Rollit:

* Compare the ratio between the Average Searches Depth that bounded versions reached with optimal versions in relation to Epsilon change.
* Compare the ratio between the final scores of bounded versions of algorithms with optimal versions to examine which Epsilon returns the best final score.
* The optimal Epsilon will be used on the next experiment, then we want to test algorithm performance against different players.

**Tasks –**

* Rollit game implementation (N-players version of the game Othello / Reversi).
* Paranoid algorithm implementation.
* Bounded Paranoid algorithm Implementation.
* Game simulation with iterative deepening from initial state Implementation.
* Heuristic Function Implementation**.**

**The experiment –**

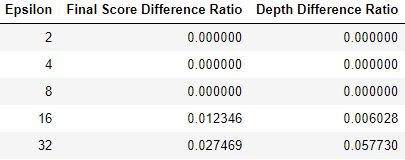
* we examine 5 different Epsilons – 2, 4, 8, 16, 32 (Score limit for Rollit game is 64).
* We defined 20,000 nodes as the high limit of nodes that we visit in single turn.
* We use iterative deepening search, when the algorithm reached the high limit of visited nodes – the search stops.
* 30 games with different initial state were generated (after 3 random moves by each player).
* We have created groups of 4 players each, when 3 players in the group are constants and the fourth player is variable:
  + Constants players - Paranoid, Paranoid, Shallow
  + Variable player Bounded Suboptimal/Bounded Paranoid (different Epsilons).
* We simulated the same game with different variable to enable comparation between the results that each Bounded algorithm reached with different Epsilons in relation to it optimalversion**.**

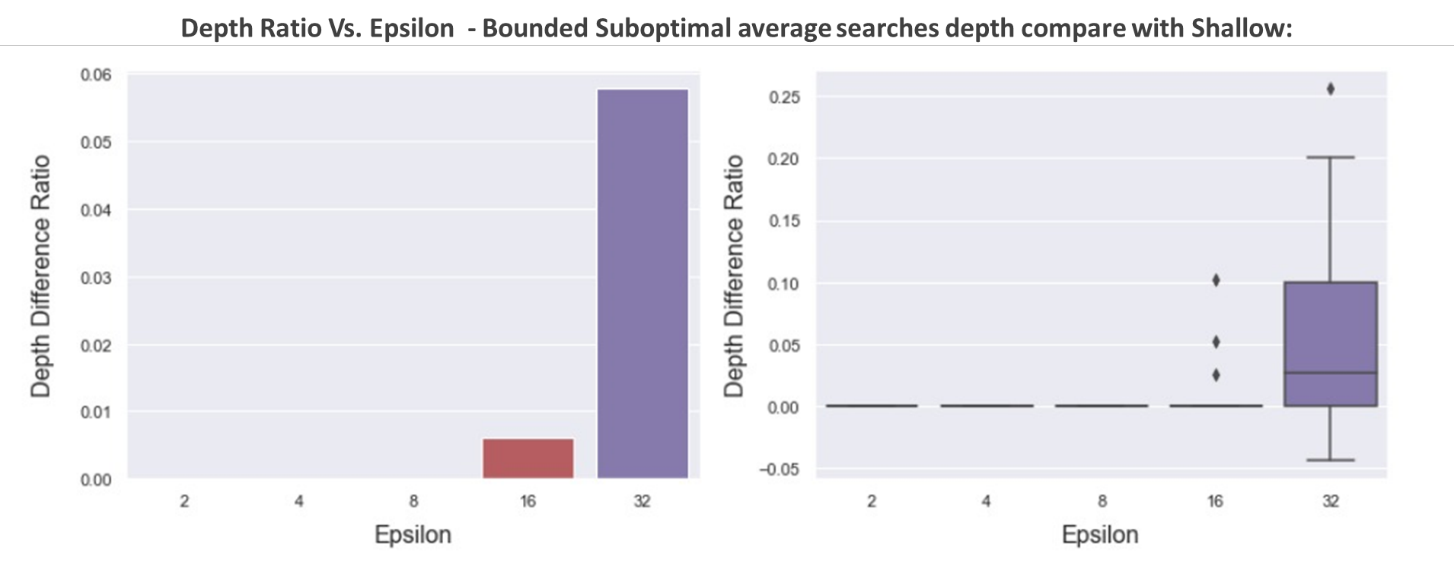
**Hypotheses –**

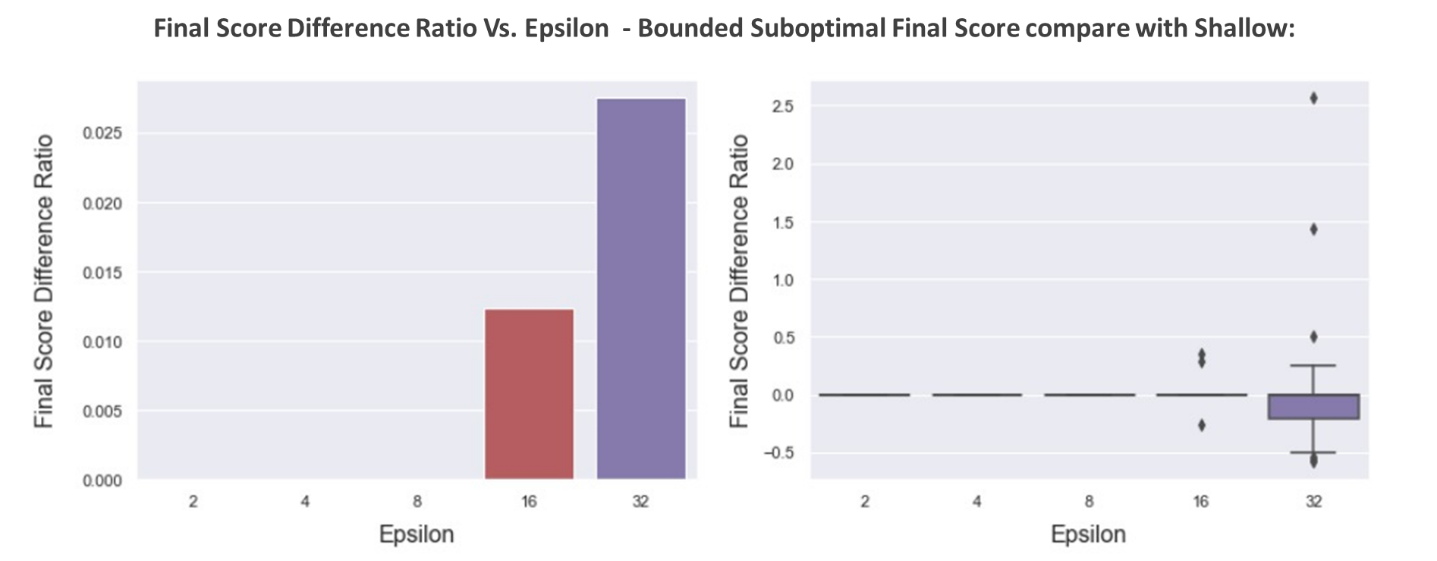
* We except to see that Suboptimal versions of optimal algorithms get better scores then optimal algorithms because they search deeper:
  + As Epsilon grow up Depth Ratio grow up.
  + Using too high Epsilon will result bad results as a result of over aggressive pruning.
* The Optimal Epsilon value is dependent on :
  + The variable player’s algorithm.
  + The constants opponent’s algorithms.
  + The game itself – rules, board size and num of players.

**Results –**

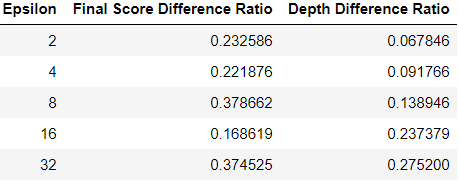
**Bounded Suboptimal vs. Shallow**

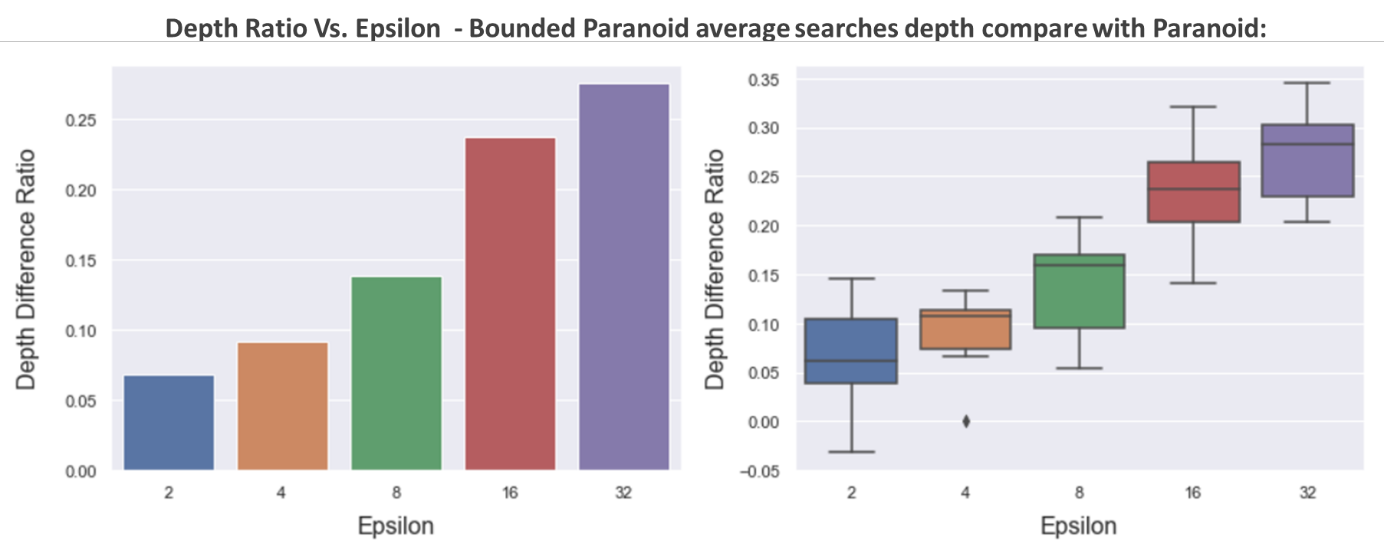


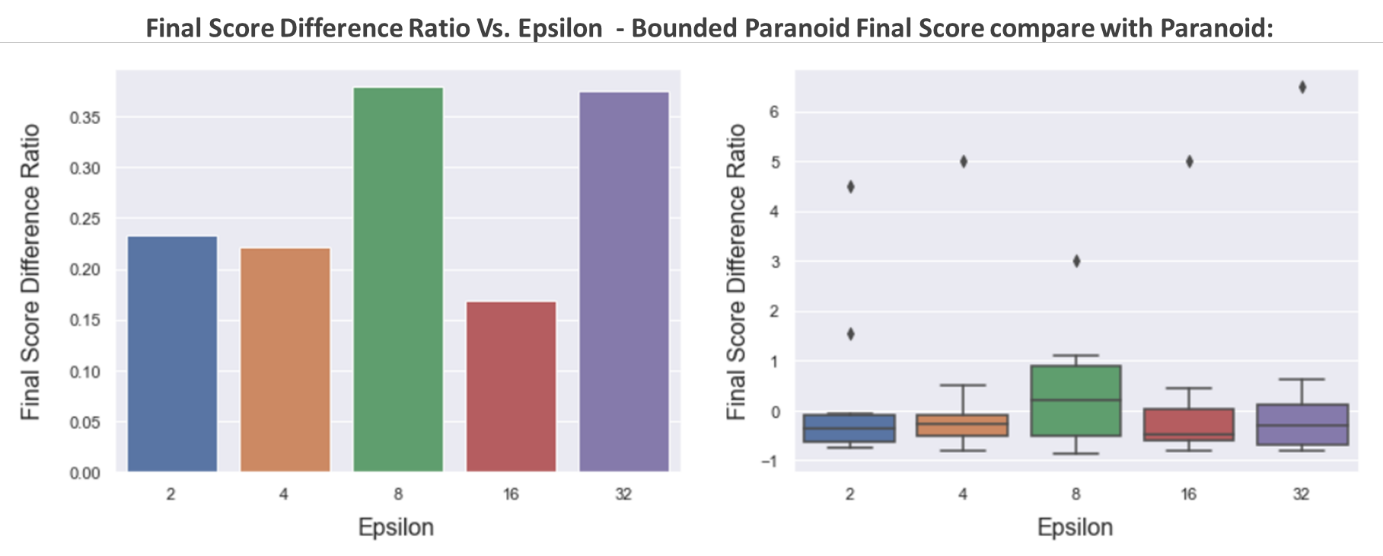




**Bounded Paranoid vs. Paranoid**

****

****

****

**Conclusions –**

**Bounded Suboptimal vs. Shallow**

* We looked for the optimal Epsilon, but the Epsilons that we chose didn’t display the Distribution well. hence, we will repeat this experiment with more Epsilons between 16 to 32 and Higher than 32.
* The improved in ‘Depth Ratio’ and ‘Final Score Difference Ratio’ is smaller and less significant than we expected (For Epsilons 16 & 32 less than 6% improved in ‘Depth Ratio’ and 3% improved in ‘Final Score Difference Ratio’, for other Epsilons its 0%). Hence, we will repeat this experiment with higher high limit of visited nodes for single run from 20,000 to 50,000.

**Conclusions –**

**Bounded Paranoid vs. Paranoid**

* It is hard to see a clear trend of ‘Final Score Difference Ratio’. hence, we will repeat this experiment with more Epsilons between 8 to 32 and Higher than 32.
* We can see a clear trend of ‘Depth Ratio’ as we expected.
* We improved the ‘Depth Ratio’ and the ‘Final Score Difference Ratio’ significantly as we expected in relation to paranoid.
* We believe that the reduction to 2-Players it’s the reason for the big difference.

**Next Step -** Test algorithms performance

Test Bounded Suboptimal & Bounded Paranoid performance against different players:

* Examine Bounded versions scores when the opponents are variables players.
* Compare Bounded versions scores with opponents (not with optimal version that played against same opponents).
* Compare the Won Ratio between all the players and to determine which algorithm reached the best scores.

**Tasks –**

* Weak Rational player algorithm implementation.
* Random player algorithm Implementation.
* An opponent’s combinations Generator Implementation.

**The experiment –**

* We examine 6 different players (algorithms).
* We define new limit for number of nodes to visit that we visit in single turn.
* We use iterative deepening search, when the algorithm reached the high limit of visited nodes – the search stops.
* 50 games with different initial state were generated (after 3 random moves by each player).
* Each game was played by 4 players that were chosen randomly.
* We test the performance of an algorithm by his games Won Ratio. The algorithm examine compare with his opponents.

**Hypotheses –**

* The bounded suboptimal algorithms will win more games than their optimal versions, due to future look-ahead resulted by searching deeper.