# Last 30 Seconds: Exploring Optimal Strategies in the NBA

Doron Hazan, Francesca Macchiavello, Mae Dotan, Metin Say

May 09 2018

## 1 Introduction

This project placed us inside the shoes of an NBA team that was down by 1 point, with 30 seconds left in the game, and left us to decide what the optimal strategy was to maximize the team's probability of winning the game. We approached this problem as if we were the team with initial possession when there were 30 seconds left. For simplification purposes, we limited ourselves to only a couple of different possible scenarios - these included: taking a "quick" 2 point shot, a "slow" two point shot, a "quick" 3 point shot, a "slow" 3 point shot, or fouling. In our code, we specified "quick" to take approximately 4 seconds, but we also tinkered around with this to test other possibilities in our sensitivity analysis.

## 2 Parameters and States

All 30 NBA teams and their respective 29 opponents were put in a table. For each team name, we derived the 2-point percentage (2P%), 3-point percentage (3P%), free-throw percentage (FT%), 2-point defensive percentage (2PD%), and, lastly, the 3-point defensive percentage (3PD%). The defensive percentages were tracked for the opposing team. We also tracked different stages of the game, including: the time elapsed, the score difference, and current possession. The score difference was initially -1 and was consequently altered. For possession, it was 1 if the team we were analyzing had the ball and 0, otherwise. We used NBA.com for data collection and information gathering. We used this data as probabilities for the branches of our tree. We used the following algorithm to determine our chances for each possibility:

$$\frac{2P\% * 2PD\%}{2PL\%}$$

**(for two pointers)**

$$\frac{3P\% * 3PD\%}{3PL\%}$$

**(for three pointers)**

# 3 Method

The algorithm we use is a game theory based tree algorithm called minimax. *Minimax is a kind of backtracking algorithm that is used in decision making and game theory to find the optimal move for a player, assuming that your opponent also plays optimally.* It is commonly used for games like chess where both opponents can observe all the game states. The approach of the algorithm is to create a tree of all possible events and assigning a heuristic value to each leaf node of the tree. The algorithm then starts from the top and recurses to choose the branch that maximizes the heuristic outcome in our current opponent's turn and to choose the branch that minimizes the heuristic outcome in the opponent's turn. This branching leads to the sequence of plays that are the most optimal for both players. Since this is a backtracking based algorithm, it tries all possible moves, then backtracks and makes a decision.

Unfortunately, the minimax algorithm relies on determinism, where each action a player takes has a probability of 1. In deterministic models, the output of the model is fully determined by the parameter values, the shot percentages, and the initial conditions. Naturally, though, we were well aware that basketball is a game of chance and full of stochasticity. Stochastic models possess some inherent degree of randomness. Each shot has a chance of going in and a chance of not going in. Therefore, we used a variation of the algorithm called expectiminimax, which incorporates non-determinism, thus it was a stochastic model. The way it does this is by introducing chance nodes to the tree, where the heuristic value propagated up the tree is calculated by the total probability theorem. The law of total probability states that the total probability of an outcome can be realized via many distinct events.

# 4 Assumptions

1. The probabilities are **independent**, the chance that team A makes a shot (2 / 3 pts) does not depend on rather team B made the last shot, and vice versa.

2. The probability of team A to make a shot in the last 30 seconds is the **same** as in the beginning/middle of the game. These stats we have gathered are general and based on team's average in regular games in the season (which means 82 games per each team), thus, they are not specific for last 30 seconds of the game. We understand that in a real-game scenario factors such as fatigue and fouling-out can lower these percentages, but for the purpose of simplicity for this project, we kept percentages static.

3. Each shot a team takes is **independent** from the previous shots they took.

4. Whenever a team takes a shot, the other team is **guaranteed** to get the rebound.

5. Teams are **not** committing any turnovers or steals. This assumption was made because turnovers and steals are merely accidents of the game, not **decisions**.

# 5 Code

The code can be found in http://github.com/metinsay/Last-30-Seconds as a Jupyter notebook. The code follows the expectiminimax recipe that can be found in Wikipedia:

**Algorithm 1** Expectiminimax

```
1:  procedure EXPECTIMINIMAX(node, depth)
2:      if node is terminal or depth = 0 then return heuristic value
3:      if the adversary is to play at node then
4:          α ← +∞
5:          for each child of node do
6:              α ← min(α, EXPECTIMINIMAX(child, depth − 1))
7:      else if we are to play at node then
8:          α ← −∞
9:          for each child of node do
10:             α ← max(α, EXPECTIMINIMAX(child, depth − 1))
11:     else if random event at node then
12:         α ← 0
13:         for each child of node do
14:             α ← α + (Probability[child] ∗ EXPECTIMINIMAX(child, depth − 1))
        return α
```

We encode our basketball problem into this algorithm with three key components:

1. **Game State -** It contains all the deterministic information of the state of the game:

(a) **Team 1:** Your team we want to maximize the win probability for.

(b) **Team 2:** Opposing team that is going against your team.

(c) **Score difference:** (Team 2 Score) - (Team 1 Score)

(d) **Time:** Time left in the game (seconds).

(e) **Possession:** Indicates which team has the possession. 1 if Team 1, 2 if Team 2.

(f) **Possession type:** The possession type. Currently can be either offensive or defensive.

(g) **Last Move:** Last move that led to the current Game State.

2. **Chance State -** It enumerates all the non-deterministic outcomes of a move. For example, if your move is to take a free throw, the chance state would contain all the Game States where you (make first, miss second), (miss first, miss second), (miss first, make second), (make first, make second) and their corresponding probabilities. It stores:

   (a) **States:** A list of possible Game States when a moved is applied to the initial Game State.

   (b) **Probabilities:** A list of the probabilities of each Game States in States list.

3. **Move -** Move gets a Game State and outputs a Chance State. It contains numeric information that corresponds to each type of move. When apply is called, the Chance State is created according to these numeric descriptions:

   (a) **Time Consumption:** The time it takes to execute the move.

   (b) **Occurrence Count:** The number of occurrences. (For example, a free-throw occurs two times)

   (c) **Score Change:** The amount the score changes by when the move deterministically occurs.

   (d) **Type:** Move type, which can be an offensive (off) or defensive move (def) in our formulation.

   (e) **Position function:** A function that returns the probability of this move to occur, given the two teams in the Game State.

   (f) **Possession Change:** A boolean value that keeps track of possession changes after each move.

   (g) **Possession Type Change:** A boolean value that keeps track of the possession type after each move.

   You can see in the Figure 1 the types of moves that we have considered.

| | Quick 2 | Quick 3 | Slow 2 | Slow 3 | Foul | No Foul |
|---|---|---|---|---|---|---|
| **Time Consumption** | 7 | 7 | 17 | 17 | 2 | 0 |
| **Occurrence Count** | 1 | 1 | 1 | 1 | 2 | 1 |
| **Score Change** | 2 | 3 | 2 | 3 | -1 | 0 |
| **Type** | Off | Off | Off | Off | Def | Def |
| **Possession Change** | 0 | 0 | 0 | 0 | 1 | 1 |
| **Type Change** | 1 | 1 | 1 | 1 | 0 | 1 |

Figure 1: Table of possible moves

# 6 Results

We ran the algorithm on each combination of NBA games, thus all 30 teams played all 29 other opponents and the results are outlined in the Jupyter notebook. We observed that for all game combinations, playing league average against league average, taking a quick 3-point shot, in 7 seconds, is the best decision to maximize the team's winning probability. However, each team has a different chance of winning against each different team, as expected. For the Celtics in particular, taking a 3-point shot in 7 seconds will result in the 36 percent chance of winning.

## 6.1 Interactive Decision Tree

We created an interactive script that creates a visual representation of our decision tree and prints it out in HTML format. The visualization shows the optimal move after each step taken. Figure 2 is a sample picture of what this tree could look like. The bolded actions are the moves that are optimal for that team.

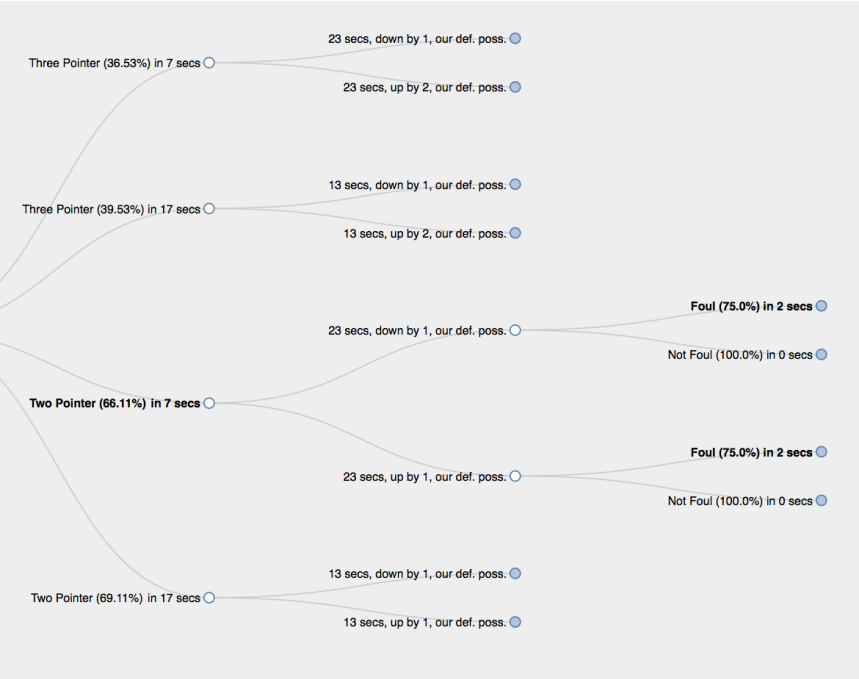


Figure 2: Interactive Optimal Tree

## 6.2 Sensitivity Analysis

For the sensitivity analysis portion of this project, we decided to analyze a variety of changing factors. These included, but are not limited to, when does the algorithm choose a 3-point shot over a 2-point shot...in other words, what is the tipping point in shooting percentages when this starts to happen? When does the algorithm switch from a quick two to a quick three...in other words, what is the ideal amount of time taken up by the possession?

Some other points of consideration included what were to happen if we were down by more than 1 point with 30 seconds left or if the other team had the possession at the beginning of those 30 seconds?

### 6.2.1 Time Left in Game

The original problem defines the scenario starting 30 seconds before the game ends. Examining the arbitrary league team when 1 point down, the results of a range of 3s to 1 minute before the end of the game (using 3 second intervals), are extremely interesting.

| Seconds left | Ideal Strategy | Chance of Winning [%] |
|---|---|---|
| 3 | 2-pt in 7s (quick) | 48 |
| 6 | 2-pt in 7s (quick) | 48 |
| 9 | 3-pt in 7s (quick) | 33 |
| 12 | 3-pt in 7s (quick) | 32 |
| 15 | 2-pt in 7s (quick) | 19 |
| 21 | 2pt in 24s (long) | 51 |
| 30 | 2-pt in 7s (quick) | 36 |
| 33 | 3-pt in 24s (long) | 31 |
| 39 | 2-pt in 24s (long) | 47 |
| 42 | 3-pt in 24s (long) | 31 |
| 45 | 2pt in 7s (quick) | 30 |

Figure 3: Sensitivity Table on Initial Time On the Clock

#### 6.2.2 Score Difference

The original question of this paper was what is the ideal strategy when a team is down 1 point, 30s before the end of the game. We wanted to investigate what is the optimal solution when the score difference is a larger range. This was important to us, since not many games have only 1 point different at 30s, so this analysis could be useful for more games.

Looking at the Boston Celtics against the league average opponent.

| Score Difference | Ideal Strategy | Chance of Winning [%] |
|---|---|---|
| -5 | 3-pt in 7s (quick) | 6.13 |
| -4 | 3-pt in 24s (quick) | 11.22 |
| -1 | 2-pt in 24s (quick) | 40.12 |
| 0 | 3-pt in 24s (quick) | 56.38 |
| 1 | 2-pt in 24s (quick) | 68.18 |
| 4 | 2pt in 24s (quick) | 93.31 |

Figure 4: Sensitivity Table on Initial Score Difference

For the Celtics, using this table, the strategy would be divided to the following point difference intervals:

1. Down 5 points or more: take a quick 3-pt
2. $-4 \leq point < -1$: take a slow 3-pt
3. Down 1 point: take a slow 2-pt
4. Tie: take a slow 3-pt

5. Leading (1 point and more): take a slow 2-pt

When leading, the Celtics want to waste time and make the safest shot- a 2-pt. However this changes when at a tie, because now there is a need to make a large enough difference from the other team in the chance of a foul. Being down 1 point is a very interesting case in basketball, because you can only score 2 or 3 points in shots, so the strategy needs to make up for this point difference in a safe way. When down more than 1 point, the team obviously will try to take the quickest shot it can take. An interesting observation, is that the optimal solution is to take a 3 point instead of a safer 2 point. In our opinion, this is due to the high (relatively) 3pt% of the Celtics.

This varies greatly depending on the team you are looking at, because their 3-pt and 2-pt averages are different.

#### 6.2.3 Three Pointer Shot Percentage

We wanted to consider the case that a team might have a 3 or 2 point average above or below the league average, and see how this affects their strategy. To do so, we set the team 2-point average constant at the league average, and varied the 3 point average by increments of 1%. The league average for a 3-point shot in 2018 was 36.2%. In a game where the team is down 1 point with 30 seconds left, here are samples of the outcome due to the analysis. There is the option of making a quick 2 or 3-pt throw, which is reduces your scoring chance by 3% (arbitrary chosen value), or taking a slow shot. Looking at this table, it is clear that any percentage above league average immediately changes the team's strategy to use a 3pt shot, and increases their chances of winning. Interestingly enough, the optimal choice is always to take a quick shot instead of a slow shot.

| **3-pt [%]** | **Optimal Winning Strategy** | **Chance of Winning [%]** |
|---|---|---|
| 30.2 | 2pt in 7s (quick) | 34.77 |
| 31.2 | 2pt in 7s (quick) | 35.08 |
| 36.2 | 2pt in 7s (quick) | 36.2 |
| 37.2 | 3pt in 7s (quick) | 36.45 |
| 40.2 | 3pt in 7s (quick) | 38.55 |

Figure 5: Sensitivity Table on Three-Point Percentage

#### 6.2.4 Two Pointer Shot Percentage

The 2-pt league average is 51%. Using our estimate, this lowers to 48% if taken as a quick shot. Figure 6 is the analysis of the effect of the 2-pt average variation (keeping the 3-pt average constant at league average).

| 2-pt [%] | Optimal Winning Strategy | Chance of Winning [%] |
|---|---|---|
| 46.0 | 3pt in 7s (quick) | 33.63 |
| 49.0 | 3pt in 7s (quick) | 34.97 |
| 50.0 | 2pt in 7s (quick) | 35.47 |
| 51.0 | 2pt in 7s (quick) | 36.2 |
| 56.0 | 3pt in 7s (quick) | 40.68 |

Figure 6: Sensitivity Table on Two-Point Percentage

#### 6.2.5 Quick Shot Percentage Reduction

Another interesting aspect that we wanted to examine is the percentage decrease in scoring when trying to take a quick shot. For a brief explanation: there are teams that specialize in taking quick shots as their strategy, as opposed to teams that don't do as well, and generally opt to taking long shots. Therefore, we changed the percentage reduction from 3% to 10%, to see what teams who do not do as well with quick shots would choose as their strategy.

For the Golden State Warriors playing against the Minnesota Timberwolves, the optimal strategy (when down 1 point 30s before end of the game) is taking a 2-point shot within 7 seconds (short). This will give a winning probability of 46.7%. If we reduce the Warriors scoring quick shot average by 10%, the optimal solution changes to a 2-point shot within 17s (long). The winning probability reduces to 37%. This is not surprising, but knowing that this change can affect the time interval of the shot is interesting.

We wanted to find where this transition happens, when looking at a team comparing to the league average as an opponent. We saw that for the Warriors, the transition between 2-point short shot to the 2-point long shot was between a 8 to 9 percent reduction. For an 8% reduction, the probability is 35.3% using a 7 second shot, while for the 9% reduction, the probability of winning goes up to 35.8% using a 17 second shot.

These conclusions vary greatly between teams. Looking at the Boston Celtics, who have a lower 2-point average, compared to the Warriors (49.1 vs 56%, respectively), the optimal strategy, regardless of whether the percent reduction is 3% or 10%, the ideal move is always to take a quick 3-point shot. Of course, the probability of winning decreases when your chance of making a quick shot reduces.

## 7 Conclusion

Our initial goal for this project was to find optimal move for Boston Celtics to take in a situation where they are down by 1 and 30 seconds left on the shot clock. We were thinking of using supervised methods to come up with solution.

However, after we shifted from supervised to non-supervised methods, we found out that we could create something more powerful that can be generalized. We can now optimize any team at any particular end game situation, which can also we visualized with an interactive tree.

The general conclusion was that for all game combinations, taking a quick 3-point shot, in 7 seconds, is the best decision to maximize the team's winning probability. However, each team has a different chance of winning against each different team, and taking into account the differences between them in chances of making a shot, the optimal solution can change widely. We examined the sensitivity of this model and optimal solution with regards to many variables: 2 and 3 point average compared to the league average, quick shot percent reduction, having a larger point difference, a different time before end of game and different average shot times.

Generally, when the team is down by 1 point, they should always take a quick shot. Whether it is 3 or 2 point, depends on the team average. When it comes to the quick shot percent reduction, this varies extremely depending on the team. When playing against a league average team, the Warriors should stick to a 2-pt (due to their extremely high 2 point percentage), while the Celtics should op for a 3-pt. The score difference changes the strategy greatly, because the decision on whether to take risk depends on the point difference. Lastly, the time left to the game and the average shot time do not give precise conclusions because out method is deterministic and depends on the team percentages and on what we define as average shot times.

# 8 Sources

http://bl.ocks.org/robschmuecker/7926762 - Javascript code for tree
http://nba.com - NBA Team percentages