A New Model of NBA teams and Policy Comparison of Season Simulation: ORF418 Final Project

Russell Kim

Introduction

Sports analytics has taken off rapidly in the past few years, and special attention is being paid to the NBA. The fast paced game of basketball lends itself well to statistical analysis, and I propose a new model of NBA lineups. Typically speaking, the world of NBA analytics has generally focused on what the best combination of 5 players is. For example, the statistic "Offensive Rating" of a lineup attempts to judge how offensively successful a lineup of 5 players is. However, as we see with a team like the Cleveland Cavaliers, there could be a situation in which the starting 5 players are exceptional players, but then the bench consists of inconsistent players. Therefore, I propose a new model that measures the overall "goodness" of a 10 player subset out of 15. The top five bench players are a massive contribution to any team, so this may give a better idea of who the best 10 players on a team are. Since this is a subset selection online learning problem, I will simulate an entire season.

Statement of Belief Model

I will be using a lookup table with correlated beliefs, since with 15 players on a typical NBA roster, each player has an impact on each other. However, in order to design a project which maximizes the output of any 10 players, we must consider the possible subsets of 10 players out of 15. It also makes sense that the different covariances of the lineups are relatively high, since there are bound to be at least 5 players that are on both lineups.

State Variable

My state variable is a lookup table with correlated normal beliefs. Essentially, what the policy believes at the time is a vector with estimated means corresponding to each lineup, and a covariance matrix giving the covariance between each lineup. The mean of a lineup represents what the (+/-) box score of the lineup is when they are on the floor for an entire game - it represents the average point differential between that lineup and So my state variable is given as a tuple:

$$S_n = \{\bar{\mu}^n, \sum^n\}$$

where n is the number of iterations that have passed.

Decision Variables

The decision variable is given by

$$X_{n,i} = x$$

where $X_{n,i}$ is an index that indicates which lineup will be chosen for the next simulation.

New Information

New information is given by

$$W_{n+1,x}$$

and this represents a sample point differential when lineup x plays a game.

Transition Function

Here we update our belief state using observations from simulations based on our choices - the usual Bayesian updating equations suffice since our beliefs are normally distributed. So if we choose lineup x for the trial at time n, then our beliefs for subsets y will change according to the equations:

$$\bar{\mu}_y^{n+1} = \bar{\mu}_y^n - \frac{W_n^{n+1} - \bar{\mu}_x^n}{\sigma_W^2 + \sum_{xx}^n} \sum_{xy}^n$$

$$\sum_{y,y'}^{n+1} = \sum_{y,y'}^{n} - \frac{\sum_{x,y}^n \sum_{x,y'}^n}{\sigma_W^2 + \sum_{xx}^n}$$

Objective Function

Performance Metric

My performance metric for each policy is given by

$$R_n = \sum_{i=0}^{n-1} r_i$$

where r_i is the point differential achieved by the policy's decision on the ith iteration, and R_n is the cumulative point differential accumulated by the nth iteration.

Objective Policy Maximization Function

Since we want our simulation, and therefore, our objective function to capture the cumulative reward, we will state our objective function as:

$$\max_{\pi} \mathbb{E}_{\mu} \mathbb{E}_{[W^1,\ldots W^n|\mu]} \sum_{n=0}^{N-1} \mu_{X^{\pi}(S^n)}$$

Design of Truth and Priors

To design the actual truth, then the priors, I will use a third order model.
The first order, which I will denote by θ_1, gives the points contributed by a certain player when that player is in the lineup. These will be a vector of 15 normally distributed random variables, all with a mean of that player's points per game metric for this year so far.
The second order, denoted by θ_2, will represent the point differential when two specific players are on the same lineup. For example, $\theta_2^{(1,2)}$ represents the point differential when players 1 and 2 are on the floor together. Hence, a negative value of $\theta_2^{(1,2)}$ would indicate that the two players have a net negative effect on the lineup as a whole. There would be $\binom{15}{2} = 105$ different θ_2 values, and would be generated via sampling from a uniform random distribution between the values of -10 and 10.
Similarly, the third order, denoted by θ_2 will represent the point differential when three specific players are on the same lineup. This would be generated via sampling from a uniform random distribution between the values of -5 and 5. There would be $\binom{15}{3} = 455$ different θ_3 values.
The values of volatility decrease because the effects of an additional player to a lineup decreases with the addition of every player.
I will also denote the set of 1 player combinations by P_1, the set of 2 player combinations by P_2, and the set of 3 player combinations by P_3.
The **truth** of lineup $i, \bar{\mu}_i$, will consist of this expression:

$$\bar{\mu}_i = \sum_{a \in P_1} \theta_{1,i}^a \cdot x_{1,i}^a + \sum_{a \in P_1} \sum_{b \in P_2} \theta_{2,i} \cdot x_{2,i}^{a,b} + \sum_{a \in P_1} \sum_{b \in P_2} \sum_{c \in P_3} \theta_{3,i} \cdot x_{3,i}^{a,b,c}$$

where the first order vector $x_{1,i}$ is a vector of 15 indicator vectors and $x_{1,i}^a$ if the ath player is in the lineup, the second order vector $x_{2,i}$ is a 15 by 15 matrix where the entry in the ath column and bth row, $x_{1,i}^{a,b}$, is a 1 if the ath and bth player are in the lineup together, and the third order vector $x_{3,i}$ is a 15 by 15 by 15 matrix where the entry in the ath column, bth row, and cth "stack" (the third dimension), $x_{3,i}^{a,b,c} = 1$ if the combination of player a, b, and c is present in the lineup. We will generate $\bar{\mu}$ before doing any actual learning, in order to have a truth against which we will test our policies. However, since we recognize that

$$x_{2,i}^{a,b} = x_{1,i}^a \cdot x_{1,i}^b$$

, and that

$$x_{3,i}^{a,b,c} = x_{1,i}^a \cdot x_{1,i}^b \cdot x_{1,i}^c$$

, we can write

$$\bar{\mu}_i = \sum_{a \in P_1} \theta_{1,i}^a \cdot x_{1,i}^a + \sum_{a \in P_1} \sum_{b \in P_2} \theta_{2,i} \cdot x_{1,i}^a \cdot x_{1,i}^b + \sum_{a \in P_1} \sum_{b \in P_2} \sum_{c \in P_3} \theta_{3,i} \cdot x_{1,i}^a \cdot x_{1,i}^b \cdot x_{1,i}^c$$

$$= \sum_{p \in P} \theta_p X_p$$

Truth Simulation

Simulating the truth will not be difficult, all we have to do is treat the truth as we have it as a normal random variable. The parameter that we can tune, however, to check the aggressiveness of our policies is the variance. Thus, we treat $\bar{\mu}_{sim}$ as a vector of normal random variables with manually tunable variances.

Prior Generation

Our priors, therefore, will be a random variable. I will add a normally distributed random variable $\epsilon \sim N(0, 40)$ so our prior means are defined as

$$\bar{\mu}_0 = \bar{\mu} + \epsilon$$

In order to find the covariances between the different subsets, we treat the means as linear belief models and use the formula

$$Cov(\mu^i, \mu^j) = Cov\left(\sum_k \theta_k X_k^i, \sum_k \theta_k X_k^j\right)$$

Monte Carlo Simulation for subset reduction

However, using a Bayesian updating equation on all $\binom{15}{10} = 3003$ subsets is computationally expensive and not feasible for the purposes of this project, so I propose reducing the size of the number of subsets to 100 by using Monte Carlo simulation.

1. Generate 10 iterations of $\bar{\mu}$ and average them to form a new variable, $\bar{\mu}_{sim10}$
2. Identify a random subset of 100 lineups and use only those corresponding subsets in forming our new priors
3. Throwing out the some 2900 other subsets means that we can obtain the reduced prior $\mathbb{N}(\bar{\mu}^{MC,n}, \sum^{MC,n})$
4. Denoting the winners of this process as $\{\bar{x}_1^{MC,n}, ..., \bar{x}_{100}^{MC,n}\}$, we can then define a matrix A^n of size $M \times 100$ by

   $$A^n = \left[e_{\bar{x}_1^{MC,n}}, \quad \dots \quad , e_{\bar{x}_1 00^{MC,n}}\right]$$

   where e_x is a vector of zeroes with only the xth component set to 1.
5. Then, we can calculate the reduced prior using the equations

   $$\bar{\mu}^{MC,n} = (A^n)^T \bar{\mu}^n$$

   Denote $P = \{P_1, P_2, P_3\}$. Then we can write the covariance of the mean of different lineups as

   $$\sum_{i,j} = Cov(\mu^i, \mu^j) = Cov\left(\sum_{p \in P} \theta_p X_p^i, \sum_{p \in P} \theta_p X_p^j\right)$$

   $$= \sum_{p,p' \in P} X_p^i X_p^j \Sigma^\theta_{p,p' \in P}$$

where we treat $\Sigma^{\theta}_{p,p' \in P}$ as a diagonal matrix with uncorrelated beliefs about θ for simplicity.

Policies

There are several policies that we can consider, and certainly more that are outside the scope of this project, but I will list four policies that I have implemented in this policy.

Epsilon Greedy Exploration

In this policy, I will use a tunable parameter, $\epsilon^n = \frac{c}{n}$ as the probability that I will explore, and hence $(1-\epsilon^n)$ is the probability that I will exploit. Let α be a Bernoulli random variable, distributed by $\sim Ber(1-\epsilon^n)$. Hence, this policy will exploit if $\alpha = 1$ and explore otherwise. Essentially, the policy will be represented by

$$X^n = \begin{cases} \arg\max_{x \in X} \mu^n_x & \text{if } \alpha = 1 \\ x^n \in X \text{ with probability } \frac{1}{|X|} & \text{otherwise} \end{cases}$$

Boltzmann Policy

This policy ensures that we have a good mix between exploration and exploitation. In the Boltzmann policy, we run experiment x on the nth trial with probability

$$p^n_x = \frac{e^{\theta^B(\bar{\mu}^n_x - \bar{\mu}^n)}}{\sum_{x' \in X} e^{\theta^B(\bar{\mu}_x - \bar{\mu}^n)}}$$

where $\bar{\mu^n} = \max_{x \in X} \bar{\mu^n}_x$
Then we generate a uniform random variable U which is uniform on [0,1]. Then, letting the cumulative distributive function of the probabilities be $P^n_x(\theta_B) = \sum_{x'=1}^{x} p^n_{x'}(\theta^B)$, we can write our Boltzmann policy as follows:

$$X^{Boltz}(S^n|\theta^B) = \arg\max_x \{x | P^n_x(\theta^B) \leq U\}$$

Upper Confidence Bounding

This policy involves choosing an alternative which has the greatest potential reward(instead of the greatest estimated reward), called a *bonus*, which we compute as a confidence interval. Designing this bonus is a subject of further research, but I will go with a bonus of

$$B^{unc,n}_x = \theta^B \sqrt{\frac{\log n}{N^n_x}}$$

where θ^B is a tunable parameter. I chose this bonus because it starts off relatively large and shrinks with time, which is what we want. As we collect more information, the confidence intervals get smaller, and

therefore, the bonus that we design should reflect this. The policy is given by

$$X^{UCB}(S^n|\theta^{UCB}) = \arg\max_x(\bar{\mu}_x^n + \theta^{UCB} B_x^{unc,n})$$

Knowledge Gradient

In order to apply the knowledge gradient algorithm to this specific problem we define a few terms. We define ζ_x^n as the *normalized influence of decision x.* This variable essentially captures the distance between a choice and the next best alternative, since the value of information is in its ability to sway our decision.

$$\zeta_x^n = -|\frac{\bar{\mu}_x^n - \max_{x' \neq x} \bar{\mu}_x^n}{\tilde{\sigma}}|$$

Then we define

$$f(\zeta) = \zeta\Phi(\zeta) + \phi(\zeta)$$

where $\Phi(\zeta)$ is the CDF of a standard normal distribution

$$\Phi(\zeta) = \sum_{-\infty}^{\zeta} \phi(x)dx$$

and $\phi(\zeta)$ is the PDF of a standard normal distribution

$$\phi(\zeta) = \frac{1}{\sqrt{2\pi}} e^{-\frac{\zeta^2}{2}}$$

We also define the variance of the experiment to be

$$\tilde{\sigma}_x^{2,n} = (\beta_x^n)^{-1} - (\beta_x^n + \beta^W)^{-1}$$

Then, the value of the information of an experiment x is given as

$$v_x^{KG,n} = \tilde{\sigma}_x^n f(\zeta_x^n)$$

and the KG lookahead policy is given by

$$X^{KG,n} = \arg\max_x \{\bar{\mu}_x^n + (N-n)v_x^{KG,n}\}$$

where N is the end time and n is the current time.

Procedure for simulation

1. Create an averaged prior mean for each lineup.
2. Take a random subset of 100 out of the possible 3003 lineups.
3. Create priors for these 100 lineups.

4. Create the actual truth.
5. Test each policy and measure its performance.
6. Repeat steps 3 to 5 with different parameters as needed.

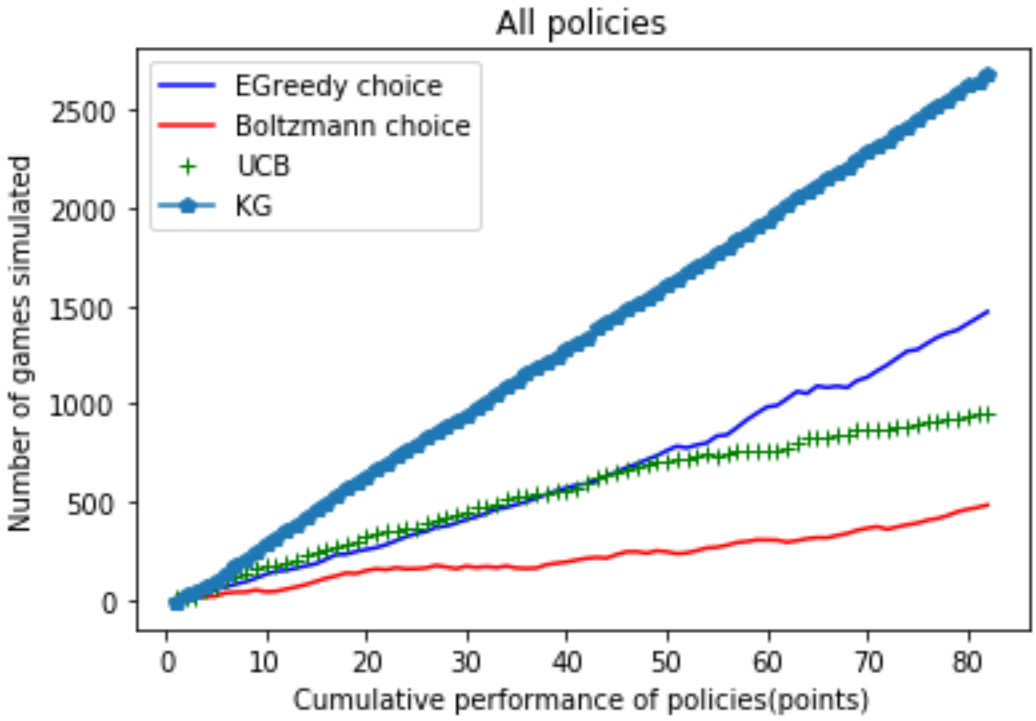
Simulation Results

At first, I used a vanilla model and simply ran the simulation using the following (random) parameters:

1. Epsilon Greedy: $\epsilon = \frac{0.4}{n}$ This represents the probability that the policy explores, and therefore, the policy runs pure exploitation with probability $= \frac{0.6}{n}$ where n is the current iteration of the simulation.
2. Boltzmann Policy: $\theta_B = 3$. This tunable parameter does not directly represent anything, but instead affects the probability of the policy choosing an experiment x.
3. Upper Confidence Bounding: $\theta_{UCB} = 5$. This value is directly tied to the *bonus*. Since the bonus gives the largest potential reward expressed as a confidence interval, a larger theta value would then assign a higher potential reward to that particular choice.
4. Knowledge Gradient: This policy does not have a tunable parameter.

Vanilla Run

Results of the vanilla run are given below as follows:

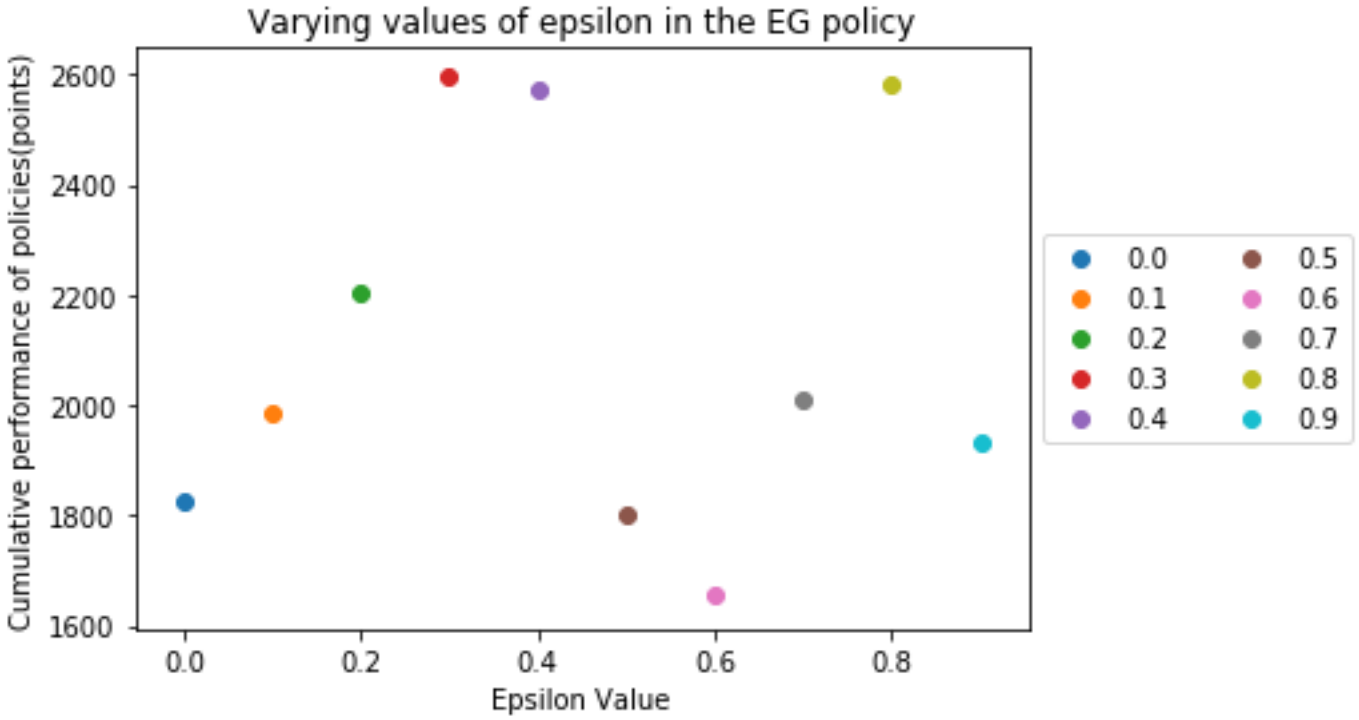


Clearly, the KG policy performed the best. However, one thing that is clear is that the Boltzmann policy

did not perform well at all, and the Epsilon Greedy policy performed unexpectedly well, despite it being the most mathematically basic policy out of the four. This suggests that some parameter tuning should give better results for the policies - or at least, some more accurate ones.

Parameter tuning for the Epsilon Greedy policy

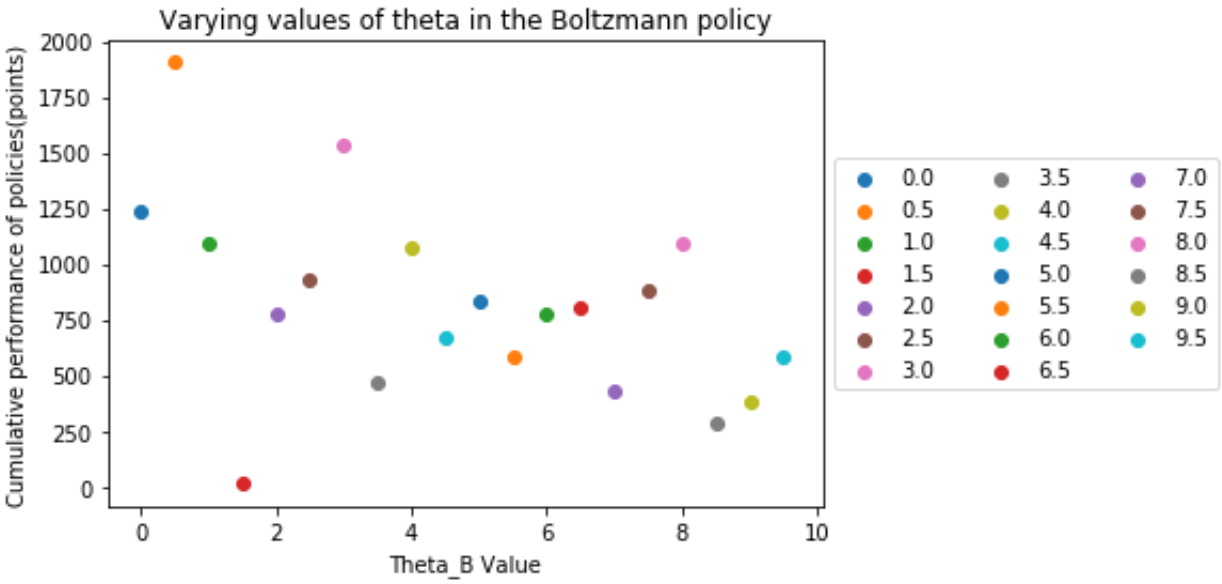
My strategy towards tuning c in $\epsilon = \frac{c}{n}$ in this policy was to test every value between 0 and 1 in increments of 0.1, and to average 100 simulations done for every value. The metric by which I would be measuring the value of the policy is by how much it maximizes the point differential over an entire season. The results are below:



The data is noisy but it is clear that some values of c are better than others. Also, $c = 0.3$ seems to be the best value for the epsilon greedy algorithm to work well. Although some values like 0.4 and 0.8 seem to work as well, I chose 0.3 to err on the side of caution, because I would want to encourage exploitation more than exploration, especially since we are analyzing a highly skill based sport like basketball.

Parameter tuning for the Boltzmann policy

For the parameter θ_B, I tested every value between 0 and 10 in increments of 0.5. Similarly to the testing of the epsilon greedy algorithm, I ran the algorithm five times and averaged out the values to get rid of outliers. The results are as follows:



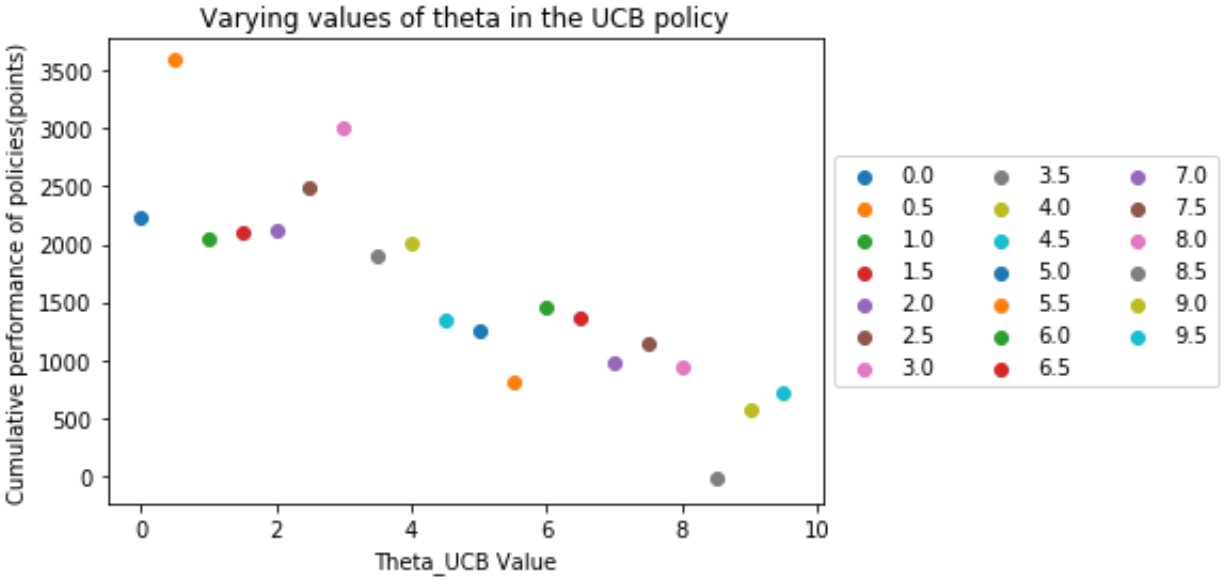
As we can see here, there is no clear relation between the value of θ_B and the performance of a policy except for maybe a very slight negative correlation with the theta values. However, it is hard to tell as the data is very noisy. Given more computation power, it would be possible to determine a more precise correlation, and perhaps even a line of best fit. It ultimately seems that the optimal value we should pick is $\theta_B = 0.5$.

Parameter tuning for the Upper Confidence Bounding policy

I tuned this parameter in the same way as the Boltzmann policy:

$$\theta_{UCB} \in [0, 10]$$

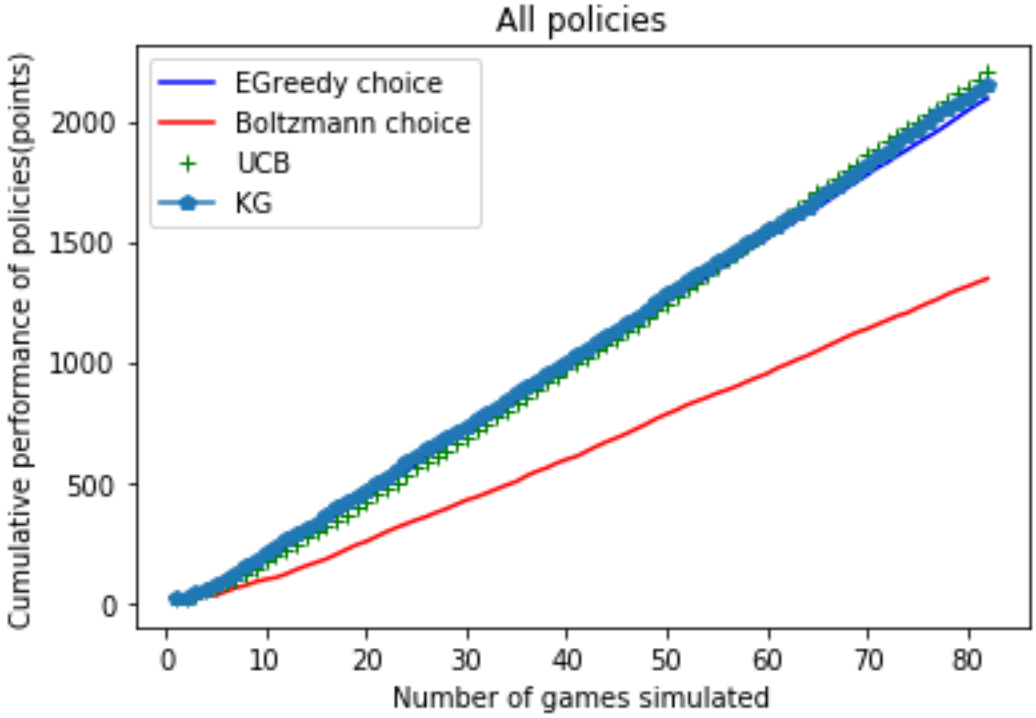
by increments of 0.5. The results are as follows:



As in the previous policies, this shows a negative correlation between the θ_{UCB} values and the performance of the policy. It seems that the optimal value we should pick is $\theta_{UCB} = 0.5$.

Comparison of tuned policies

After having tuned the previously mentioned parameters, I then simulated 30 times and averaged the performance of the different policies, as shown below:



Interestingly enough, the UCB policy seems to fare the best here. My speculation is due to the bonus that I designed in the policy itself being suited to this kind of problem. Surprisingly, the epsilon greedy policy also performed quite well compared to its mathematical simplicity. This can partially be attributed to the fact that the epsilon greedy policy that I used was not the vanilla epsilon greedy policy, but a modified version that takes into account the fact that we want more exploration at the beginning and more exploitation towards the end of the simulation. The Boltzmann algorithm also has done much better than the vanilla model, but still worse comparatively to the other models. The KG policy, as always, performs at a very high level - it does not have any tunable parameters, but it gives us a very high chance of a lower bound, or in other words, a minimum guaranteed performance.