Eastern Michigan University
DigitalCommons@EMU

Senior Honors Theses | Honors College

2008

Optimal Strategy in the Game Show, "1 vs. 100"

Seth A. Hoxworth

Follow this and additional works at: http://commons.emich.edu/honors

Recommended Citation
Hoxworth, Seth A., "Optimal Strategy in the Game Show, "1 vs. 100"" (2008). *Senior Honors Theses*. 157.
http://commons.emich.edu/honors/157

This Open Access Senior Honors Thesis is brought to you for free and open access by the Honors College at DigitalCommons@EMU. It has been accepted for inclusion in Senior Honors Theses by an authorized administrator of DigitalCommons@EMU. For more information, please contact lib-ir@emich.edu.

Optimal Strategy in the Game Show, "1 vs. 100"

Abstract
In this paper, we will be using a simplified method (a geometric distribution statistical approach) and move through a more detailed approach (using dynamic programming) to analyze when contestants should quit versus when they should stay in the TV game show "1 vs. 100." We will observe optimal contestants' strategies for when they should statistically quit answering questions or stay for the highest probability of a contestant win and the greatest possible expected amount of money to walk away with.

Degree Type
Open Access Senior Honors Thesis

Department
Mathematics

Keywords
Games of strategy (Mathematics), Programming (Mathematics)

This open access senior honors thesis is available at DigitalCommons@EMU: http://commons.emich.edu/honors/157

Optimal Strategy in the Game Show, "1 vs. 100"

by

Seth A. Hoxworth

A Senior Thesis Submitted to the

Eastern Michigan University

Honors Program

In Partial Fulfillment of the Requirements for Graduation

With Honors in Mathematics

April 28, 2008

Ypsilanti Michigan

Abstract

In this paper, we will be using a simplified method (a geometric distribution statistical approach) and move through a more detailed approach (using dynamic programming) to analyze when contestants should quit versus when they should stay in the TV game show "1 vs. 100." We will observe optimal contestants' strategies for when they should statistically quit answering questions or stay for the highest probability of a contestant win and the greatest possible expected amount of money to walk away with.

1.1 Introduction

The game show "1 vs. 100" is a trivia-based game show that has become popular in many different countries. The objective in the game is for the contestant to win out against or beat the mob that contains 100 "mob members" in a battle of wits trivia game. For our analysis, we will consider only the U.S. version of the game.

Rules and Game Play

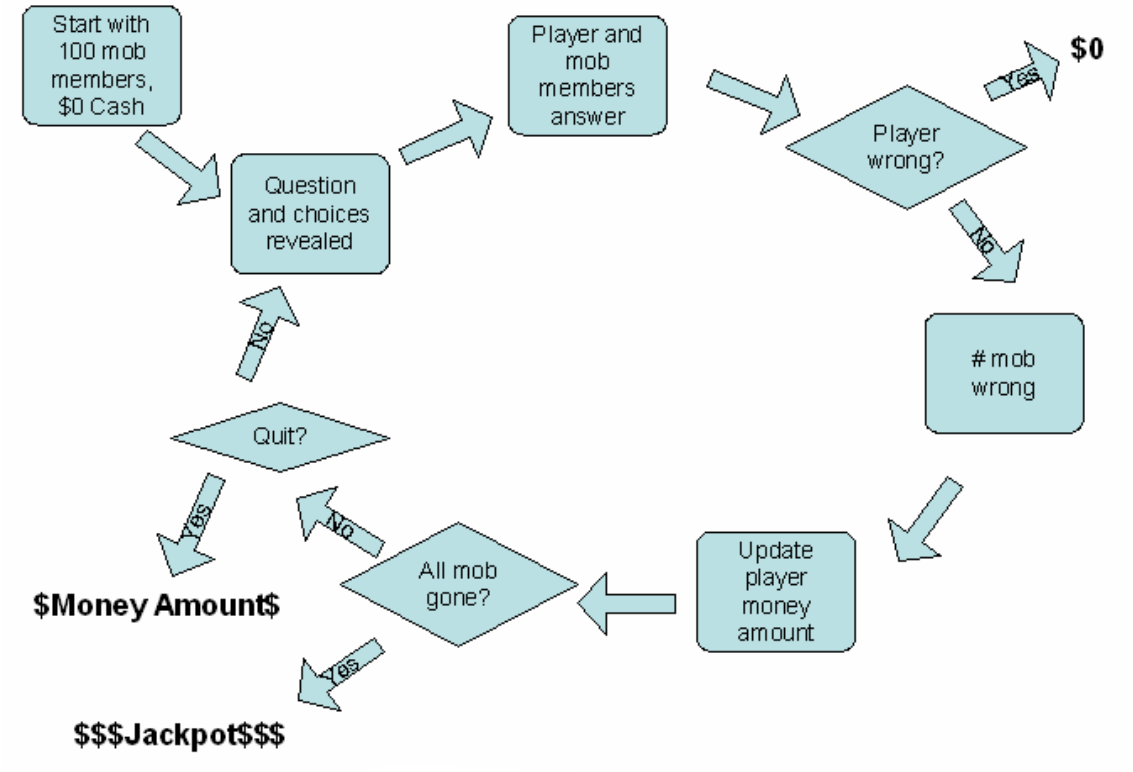


Figure 1

In this game, a single player goes up against the 100 “mob members” by answering trivia questions. Each question is given in multiple choice format with only three possible answers. If the player answers correctly, he will advance to the next round and will be rewarded an amount of money dependent on the number of mob members eliminated (by them answering the question incorrectly) and which round he is in. The value for mob members eliminated is as follows: $1,000 for each mob member eliminated in the first three rounds, in rounds 4-12 they are worth an extra $1,000 each per round advanced (i.e. $6,000 each for round 8), and for rounds 13 and above, the value of the mob members is capped out at $10,000 each. If the player is incorrect in his answer, he will automatically be sent home with nothing since there are no “safety nets” in this game. Similarly, a mob member that gets a question wrong is also eliminated. At each round, before the question is revealed, the player has a choice to either quit and take home the amount of money he has accrued up until this point or keep on playing. If the player outlasts all 100 mob members throughout the course of the game, he is awarded the “jackpot” of $1,000,000 US. In our model, we are ignoring the “helps” which are much like the “lifelines” of other game shows. There are some more complicated aspects to the game such as the fact that the player does not get a chance to decide whether to quit or stay until he makes it through the first three rounds. Another thing to consider is that as the player goes further and the round gets higher, the questions get more difficult and there is more on the line to lose (or gain). As the questions get more difficult, our p-value (probability that the player answers correctly) and q-values (probability that each of the mob members answers correctly) will decrease the higher the round is, but it will never drop below on third because that is the probability of just randomly guessing on the question. Lastly, there are three “helps” that the player may use at any point in the game to aid in him choosing the correct answer to the trivia question.

Existing Literature

There is a similar text that uses a dynamic programming model to analyze the game show “Who Wants to be a Millionaire.” This text goes through the step by step process that went into analyzing the millionaire game show and took into account lifelines and optimal strategies. The paper explains in detail the mathematical formulas and methods used to solve their model and analyze the data and construct game simulations. We have tried to do the very same thing in our project and analysis only using a different approach and many different methods of analyzing due to the uniqueness and dissimilarities between the two shows. One such dissimilarity is the size of the state space being used and analyzed in the two models; the millionaire model will have a much smaller state space (even though they took into account lifelines in their model) simply due to the enormity of the different possibilities in the 1 vs. 100 game and the more rigid state space for millionaire (only a few different possible rounds and not so many possibilities within those rounds).

2. Beginning Models

In this section of our analysis, we present two simplified. The first model, section 2.1 is the simplified model which uses a geometric distribution and also looks at a much smaller-scaled game. The results obtained from the first model will be covered in section 2.2. The second model, section 3.1, is our detailed dynamic programming model which presents an optimal strategy for the game show. Results from our second model are contained in section 3.2.

2.1-2.2 Simplified Models and Results

The first of the simplified models calculates the expected value of the entire game when the player has made it down to just one mob member left regardless of round; this value is dependent on the probability of our player answering the question correctly (the "p" value) and the remaining mob member answering the question correctly (the "q" value). For our example, we will be using probability values that start at 0.33 and go up to 1.0 since 0.33 is the "randomly guessing" value. Also in this example, the round isn't important because we are only calculating the value of the game once this point has been reached in the game and the only options are that the player either wins (value is jackpot) or loses (value is zero). The amount of money on hand is also not taken into account because we are only calculating the value of the game; the player would obviously quit if his amount of money on hand is higher than the expected value of the game. We then take the formula and apply it to a large array of different probabilities of our player getting the question right and the last remaining mob member getting the question right and compile it into a 68 x 68 matrix of expected values. The formula in question for the expected value for our player is as follows:

E[continue] = p*(1-q)*1,000,000 (*player wins*) + (1-p)*0 (*player loses*) + p*q*E[continue] (*both answer correctly*)
E[continue] – p*q*E[continue] = p*(1-q)*1,000,000 + 0
E[continue]*(1-p*q) = p*(1-q)*1,000,000
E[continue] = [p*(1-q)*1,000,000]/(1-p*q)

And the following is a selection from the table we obtained from this process:

p \ q	0.33	0.40	0.50	0.60	0.70	0.80	0.90	1.00
0.33	248120	228111	197605	164589	128739	89674	46942	0
0.40	308756	285714	250000	210526	166667	117647	62500	0
0.50	401198	375000	333333	285714	230769	166667	90909	0
0.60	501247	473684	428571	375000	310345	230769	130435	0
0.70	609883	583333	538462	482759	411765	318182	189189	0
0.80	728261	705882	666667	615385	545455	444444	285714	0
0.90	857752	843750	818182	782609	729730	642857	473684	0
1.00	1000000	1000000	1000000	1000000	1000000	1000000	1000000	500000

Table 1

As we thought, the expected value of the game rises as the player's probability of answering correctly increases and the expected value decreases as the mob member's probability of correctly answering increases. One should note that the probabilities in our matrix start at 33% and increase by 1% increments until they reach 100. Another side note should be that this computation is fairly easy because the only thing to take into consideration and compute for the value is the case where the player answers the question correctly since the expected payoff when he answers incorrectly is obviously zero.

The second model that we use to simplify the game is an approach that utilizes the geometric probability distribution. In this model, we compute the probability that the player will outlast the mob, supposing that the player never quits. In our model, we treat the number of questions in a row that the player answers correctly as its own independent geometric probability distribution dependent on an input "p" (the player's probability of answering the question correctly). Similarly, the time until each mob member answers incorrectly has a geometric distribution, with each mob member independent of the others and of the player. In using this model, we take advantage of the fact that if n represents the player's distribution, the fact that if n>max(100 IID geometric distributions), then n> 1 geometric distribution. Since this works, we were able to compile a table of probabilities where, once again, we utilize the values "p" and "q" and the data in the table represents the player's probability of winning the game given that he always chooses to continue without ever considering walking away with the money. In order to obtain the following table, some computational work need be done. The way to compute the data is to set up a program that calculates stochastic geometric functions of the following form: {[1-q^n]^100 * (1-p)*p^n} – Where n is just an index number (numbered from 1 to 1000) and p and q are defined as usual. This table is as follows:

		p				
		33%	50%	66%	83%	99%
q	33%	0.60%	3.52%	13.33%	39.46%	94.89%
	50%	0.06%	0.71%	4.94%	24.56%	92.30%
	66%	0.00%	0.04%	0.81%	10.18%	87.53%
	83%	0.00%	0.00%	0.01%	0.91%	74.93%
	99%	0.00%	0.00%	0.00%	0.00%	0.98%

Table 2: the Probability of the Player defeating the Mob Given He Does not Quit

This data can also be better understood with the help of a visual aid in the form of a three-dimensional graph of the player's probability of winning the game given he decides to always keep on playing, as a function of our "p" and "q" values which appears as follows:

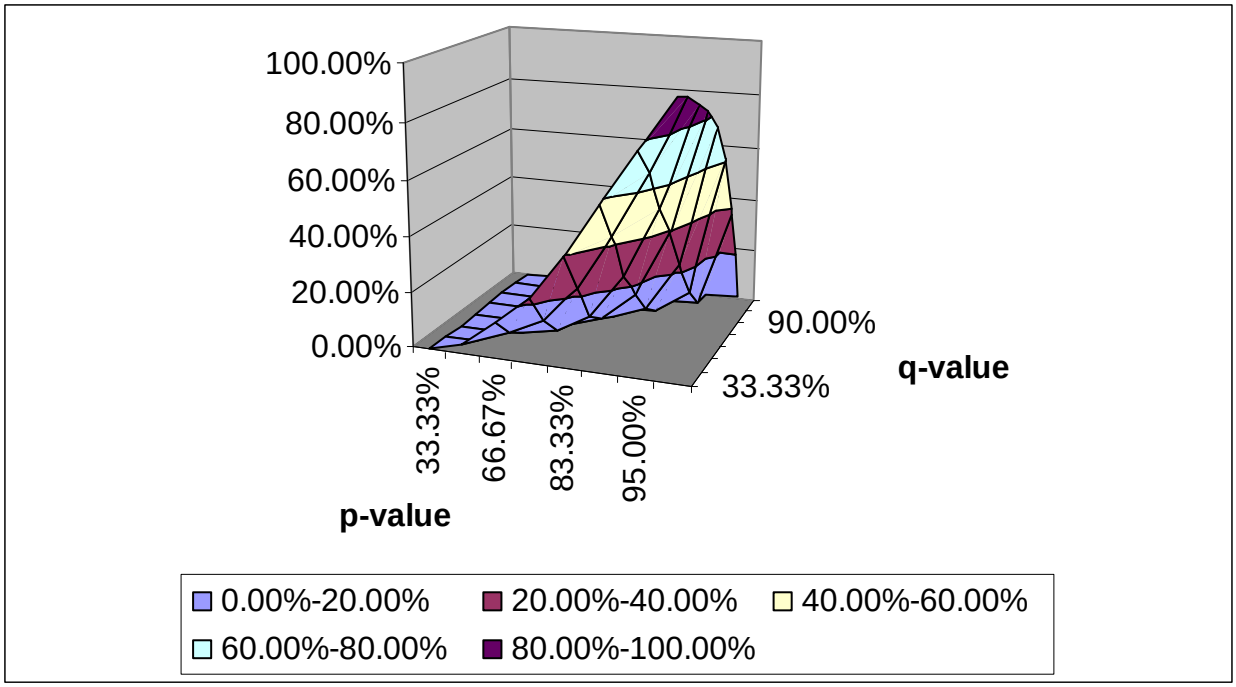


Figure 2: Probability of Player Beating the Mob (Geometric Example)

We can pull a few useful observations from this graph (and the data used to create it). First, we can see that it is concave down as a function of p; conversely, it is concave up as a function of q. And lastly, when considered as a function of both p and q, it is slightly concave up (although it is almost increasing at a constant rate) and it levels off at the very end.

We decided that the best way for us to get what we wanted out of our model was to take an approach that uses dynamic programming to devise and assess “optimal strategies” for this game show. In order to take on such a great task as that of considering all of the different possible states in a matrix of 15 rounds, 100 mob members, and 1000 different monetary amounts possible, we had to start off with something small. So, we worked through a much simpler model by hand. Since the computation gets out of hand quite quickly, our hand-worked model only considers a game that has a maximum of 3 rounds, 3 mob members and 9 possible monetary amounts.

In this model, we consider the possibility of the player eliminating all three mob members as a win and thus reward the jackpot at whichever point in the game the player achieves this. The first number under each round heading represents the probability of that particular state which is computed using a binomial distribution dependent on the number of mob members left, the number answering the question wrong and the probabilities of correctly answering for the mob and for the player. The second column underneath the row heading is the number of mob members eliminated in that particular round and the third column represents the expected value, then, of each state possible in every single round. The final number underneath the expected values for round one is the expected value of the game. All of these values were computed given a mob probability $q = 0.55$ and a player probability $p = 0.40$. In order for this to be a real dynamic program, we begin our analysis and computation for the last round first then compute the conditional expected values in each round prior until we came to the first round and were done with our computations overall. The importance of doing a hand-worked model far surpasses what is expected at first glance. The reason we did this hand-worked model first was so that we could have a guideline all set up for us, or a reference, to go by in writing our code for Scilab to take on the task of solving the full-sized game where massive amounts of calculations would need to be done on the computer rather than by hand.

3. Model – Two

This section contains our more complicated and more detailed model which makes use of dynamic programming in order to find the optimal policy for the player for the game show; it also contains the results that we obtained in researching our model.

3.1-3.2 Furthering the Model and Results

In order to take our model to the next level, we must turn to a computer program (Scilab) to do the numerous calculations necessary for figuring this all out. So we set out to program a model that uses dynamic programming to evaluate the model statistically and feeds us a possible optimal strategy for the game. The way dynamic programming works is by evaluating the last round first and calculating all of the values there where the computations are simpler and easier to do (since either the player loses or wins, or walks away and it doesn't hinge on future rounds). We chose to limit the number of rounds to 15 for the ease of computation and in our computations, the difference in changing the round higher is minimal enough to ignore since the probability of even getting to the 15^{th} round is quite low. After completing the computations for the final round, the dynamic program then moves on to the penultimate round and assesses the computational values there where now each possible state in this round is dependent on where the player could possibly end up in the next round. The computation then depends on a recursion based on binomial probability distributions and value functions for each possible state in the next round. After computing all of these values, the program moves on to the third-to-last round, and so on and so forth, until it finally reaches the first round and is able to give us an all-around expected value for the game. All of these computations are dependent upon the possible combinations of round, mob members remaining, and amount of money accrued (which is given in thousands of dollars so that "m" varies from 1 to 1,000), and each is a function of our "p" and "q" values. Our recursion function was completed as follows:

Definition: V(m,n,r) = Expected value of being in round #r with $m dollars in hand and n # of mob members remaining.

Computationally:

$$V(r,m,n) = \max\begin{cases} quit : m \\ stay : p * \begin{cases} binpmf(n;n,q) * jackpot \\ + \sum_{i=0}^{n-1} binpmf(i;n,q) * V(r+1, m + rv(i) * i, n-i) \end{cases} \end{cases}$$

In order to determine which states we should exclude and which states we should keep, we must take into consideration which states are attainable given the rules of the game. To do this, we set our program up so that it automatically discarded impossible states such as ones where the money amount is higher than the number of mob members eliminated up until that particular point multiplied by the value of each mob member in the preceding round. In our computation above, binpmf(i; n,q) is a binomial probability mass function of the number of mob members eliminated given the number of mob members remaining at this point and the probability that each mob member answers the question correctly. Also, in the above formula, rv(i) stands for round value of each mob member eliminated (this is dependent on which round we are in).

Confident in our success from the simplified model, we must test our program on a full-scale version of the game and run calculations with 100 mob members, 15 rounds and 1500 different possible money amounts (from $1,000 up to $1,500,000)in order to grasp the concepts we are utilizing. At first, this simulation seemed cumbersome on the Scilab program, so we observed the process and decided that by moving the reference index for the round in our value model to be the first index, we would increase the computation speed. This increase in speed is due to the fact that, in our first model, the program was referencing code in many different places for data for each round, but the change made it so that the program had all of the data for each round in the same place which made for quick and efficient referencing. This change made such a great difference in processing speed (from about 15 minutes per set of calculations down to around 90 seconds) that it enabled us to do much more analysis more quickly than if we had been running it through the way we originally had set it up.

As we do the computations, we record which decision (quit or stay) is optimal for each combination of round, money amount, and number of remaining mob members. Figure 3 shows, for round 13, when the player should quit or stay based on the number of remaining mob members and the accumulated money. States above or below the blue/black wedge are impossible.

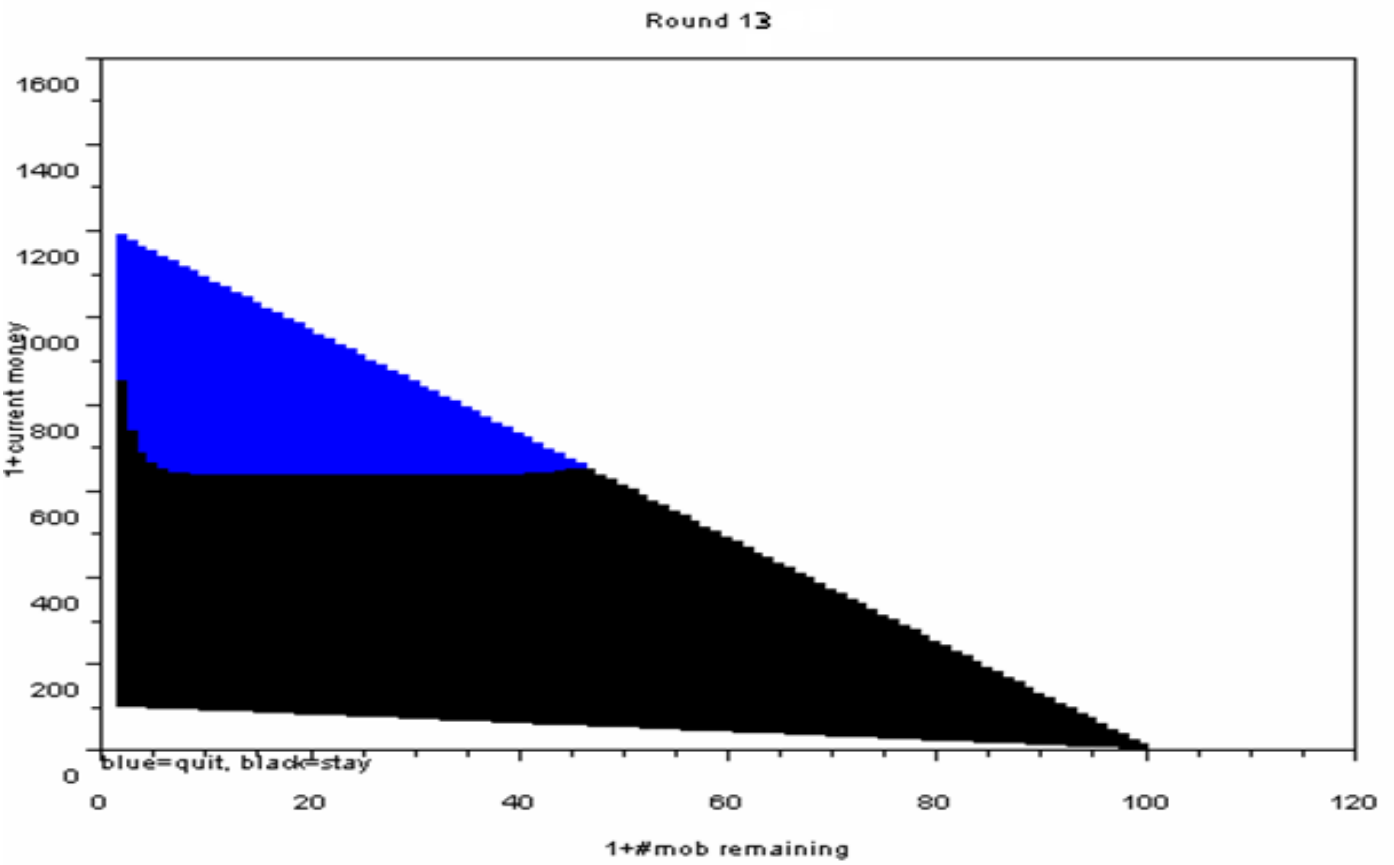


Figure 3

This matrix graph considers whether the player should stay or quit given the number of mob members remaining and the amount of money won already. In the above graph, the blue blocks represent a decision for the player to quit since it is statistically better to do so and the black blocks represent the player's decision to stay and play since the expected value of doing so is greater than the amount of money on hand for the player. All of these values are also dependent upon the p and q values within the model (for the above graph, p-value = 0.75 and q-value = 0.65).

A few interesting interpretations can be obtained through analysis of Figure 3. The first of these is that for any constant number of mob members remaining in any round, there is a distinct cutoff point (in the money value) above which the player should quit and below which a player should stay. This cutoff is clearly marked by the crossover between the black and blue colors in the graph.

A second, perhaps more peculiar interpretation of the above graph is that for a constant money amount there can either be no cutoff points, one cutoff point, or two cutoff points in terms of mob members remaining and the quit/stay decision. This occurrence happens because, first, there are money amounts that are so low in comparison to the expected money value of the game that they would never be above the cutoff in a given round; secondly, because there are states where the number of mob members remaining only matters to a certain point (very few left) after which the player is better off just quitting anyways; and finally, because the amount of money on hand is such that if there are a good number of mob members left, it is better to stay because your likelihood of eliminating enough of them to drive up your expected value is relatively high, and on the other end of the spectrum, there are so few remaining that you should stay in the game because the likelihood of receiving the jackpot and defeating the mob is such that it also drives up your expected winnings. This set of different possibilities leads to a U-shaped cutoff point in each of the rounds in our example.

Instead of plotting each round separately, Figure 4 shows for all rounds just the boundary between quit and stay decisions:

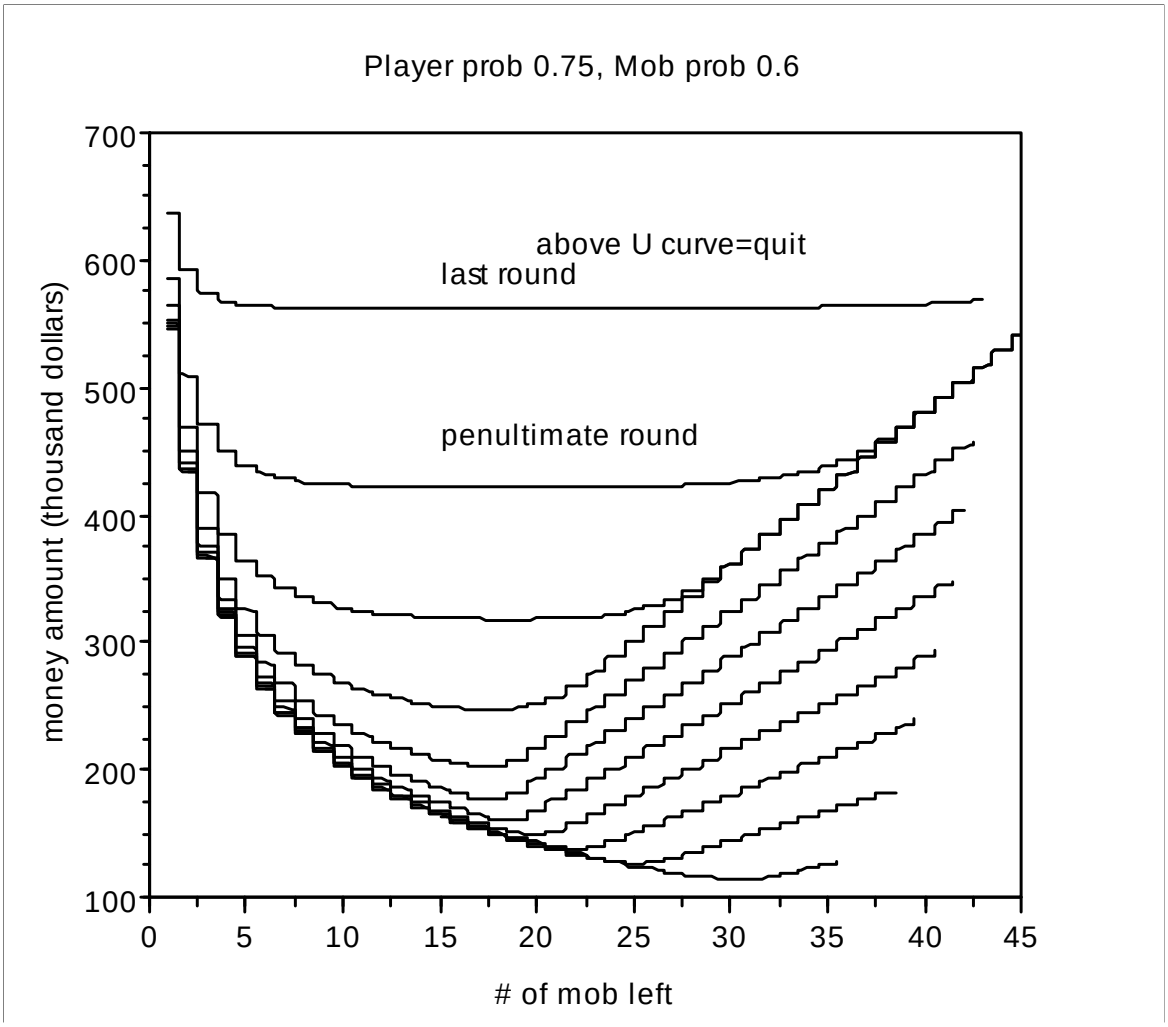


Figure 4

In this graph, anywhere above each of the u-curves represents an area where the player would quit instead of staying for each round respectively. The last round is the highest curve, with the second-to-last round as the second highest curve, all the way down to the lowest curve representing the second or third round decisions. As seen in the picture above, these calculations were made with a p-value of 0.75 and a q-value of 0.6 and these values are based on a maximum of $1,000,000.

When we had a good chance to go over all of the different statistics, print-outs, and models that we had considered so far, there were a couple more things that we discovered we could analyze from this game. First, we figured that we could write another program that actually ran random simulations of the game in Scilab given all the rules and restrictions we had placed on the game. So, in essence, it played out games according to our rules and captured the statistics in order to give us an idea of how well our model worked and where the players would end up. Our findings can be seen in the following histogram:

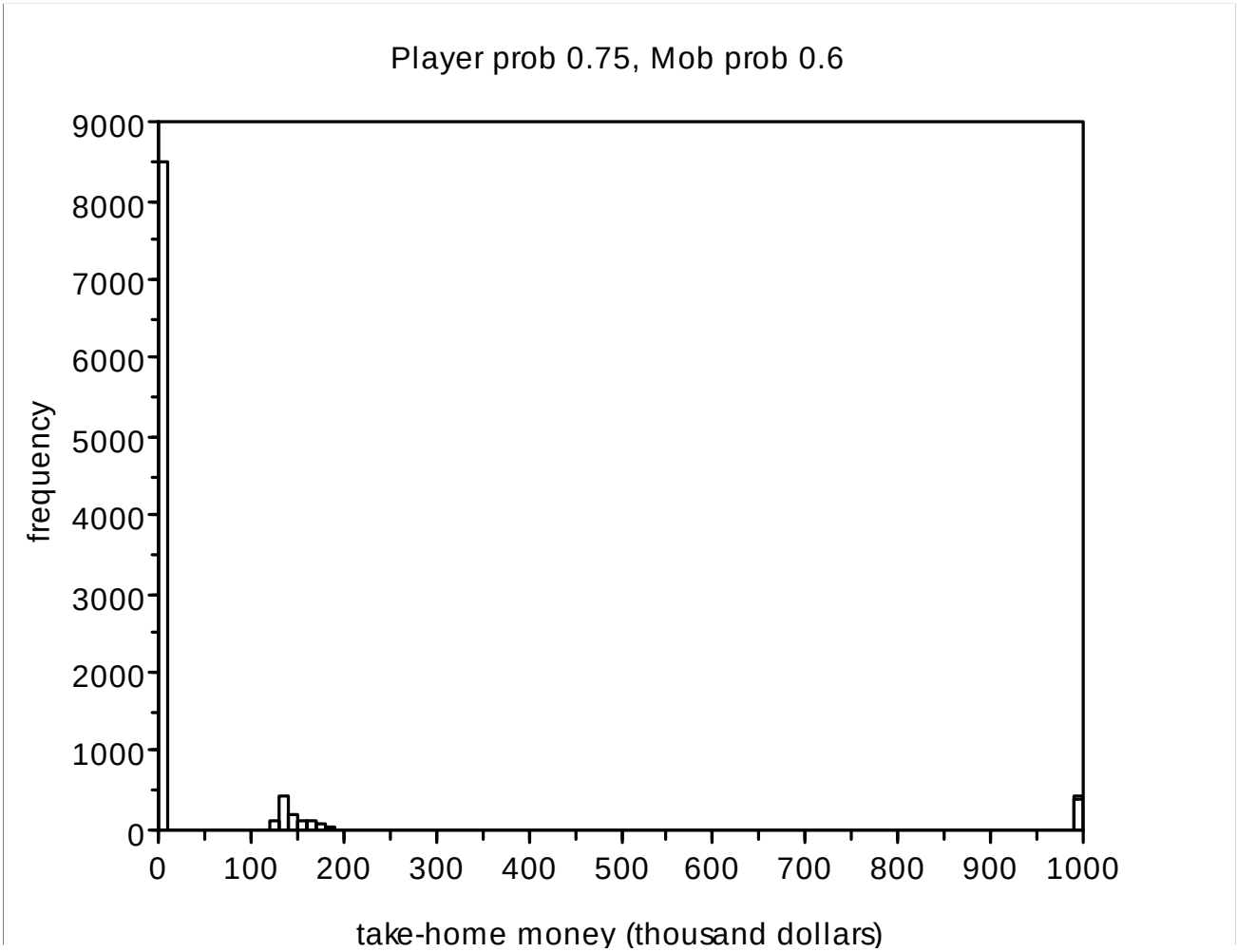


Figure 5

This histogram, which was computed using a p-value of 0.75 and a q-value of 0.60, shows the distribution of how much money our player went home with in the simulations of the game. As you can see, nearly 85% or 8,500 of our 10,000 simulations for these particular p and q values ended in the player failing or answering the question incorrectly since it was hard for the player to reach a state where the expected value of playing was lower than the amount of money on hand. We can also see that somewhere around 400 of our simulations ended in the player winning it all and beating the mob; this translates to about 4% of the time. Also, there are various spots throughout the histogram where the player walked away (portrayed between $100,000 in winnings and $200,000 in winnings in the histogram above). These data go along with the expected value of the game table that we calculated. This table follows:

Game value at start (in thousands of $)					
Player's probability of being right	0.85	285	193	121	103
	0.75	114	59	51	42
	0.65	43	38	32	25
	0.55	30	27	22	16
		0.5	0.6	0.7	0.8
		Mob member's probability of being right			

Table 3

An interesting observation of this table is that the first row is concave up as the q-value changes and rows three and four are concave down as the q-value changes; also, row two is neither concave up or concave down. This seems to imply that there is some cutoff point for the expected value of the game as the p-value increase above which the expected value is concave up as a function of the q-value and below which the expected value is concave down as a function of the q-value.

On a side note, here we see that as the player's probability of getting a question right goes up, the expected value of the entire game increases with it. Also, when the mob members' probability of answering correctly increases, the expected value of the game for the player then decreases. We came up with these same observations as we analyzed our earlier models, which shows consistency and gives us a good idea of where we are. Thus, the table turned out as we would have expected it to and now we can work on further analysis of our model.

4. Conclusions

4.1 Conclusions from Earlier Work

In our earlier work, we dealt with the more simplified models such as the “one mob member remaining” scenario and the geometric distribution approach. And from these models we were able to draw a few different conclusions. We found out that the geometric probability approach was not a bad way of going about analyzing since it did give us a good idea of how the probabilities in the game turn out. This also gave us a good idea of what kinds of things we should be looking for in the future as far as being to analyze the game and devise an optimal policy. And lastly, we figured out that we were going to need a completely different type of approach and model altogether if we were going to accurately come up with a way to devise an optimal policy for the game as a whole. This led us to taking up the dynamic programming approach that we used in our simplified (smaller scale) version of the game and our hand-worked model. This gave us the tools that we needed in order to construct the full-size game and be able to get things right without having to go through all of the pain of trying to start it from scratch. It also provided us a nice guide to go by in order to work out the kinks of the full-size model and get to analyze the real data. All of these things working together brought us to the conclusion that something more powerful was needed and thus we devised the full version model using a dynamic programming approach.

4.2 Conclusions from Present Work

From the full-size model and the most recent group of work that we did, we dealt with a dynamic programming model and used that to devise an optimal strategy or at least a list of statistics to aid in devising an optimal strategy for the game show “1 vs. 100.” We were able to construct, simulate, and obtain pertinent data from the models that we created using the software Scilab and running our dynamic program through that. From this work we were came up with some interesting conclusions.

We came up with a way to record the decisions that a player should choose and put them into a matrix that we called the decision matrix of our model. This matrix was represented by the blue and black graph in section 3 and it tells us the decision that the player should choose statistically in every single possible state within a single round. Each round has its own, unique decision matrix and set of possible states that could be attained in it. The black blocks represent a state where the player should choose to stay and continue to play because statistically, the money amount he has in hand is less than he is statistically expected to win given the specific number of mob members that are left in the single evaluation for that specific round. The blue blocks represent the opposite, a choice made by the player to simply walk away from the game because the amount of money he has accrued is greater than the expected amount he is to win given the specific set of parameters for that state.

From these graphs we were able to obtain a different graph or set of graphs that let us know about where the general cut-off ranges for our model was depending on the specific round that we were in. This gave us a good idea of what levels and cutoffs we can expect we would have to obtain in order to be better off in the coming rounds ahead. This lets us know a good general idea of when to quit and also allows for us to make a bit more of an observation based on when we should really look at quitting given the lower probability of making it through the higher rounds.

From all of this analysis, we can see that a good general rule for quitting in the game would be that if you make it above $200,000-$250,000 at any point in the first several rounds (perhaps first ten rounds), then you should statistically just “take the money and run,” because you are not expected to statistically get higher than that. Another reason you should quit at this amount even though the higher rounds have cut-off levels higher than $250,000 is because the probability that you get through the higher rounds is not very good and you will really only be able to go so far. This is a game show that is designed to make you take risks and lose the game; therefore, you shouldn’t get caught up in the greed that is built into the game if you would like to maximize your winnings. Just take it easy, relax, and play smart and in the end you could maximize your winnings by getting out with the money while you still can and not succumbing to the greed. You will definitely be better off if you adopt this approach to playing the game.

4.3 Possible Future Work

Having gone through our analysis and having come to the conclusions that we have come to, there is still a great deal that we have not considered in our model. Some possible things to think about for future analyses would be very helpful in learning to understand this game on an even greater level. A couple of very important things that we should definitely look at, first of all, would be to incorporate the "helps" into our model and devise a way of using these helps at certain levels in order to maximize our winnings and develop an optimal strategy that is more all-encompassing. Another important detail we should incorporate into future work would be the new rule changes that the show has made; these new rules were made regarding the money amounts the player wins and are given now as follows:

# of Mob Members Left	Player's Total Winnings
100 to 91	$0
90 to 81	$1,000
80 to 71	$5,000
70 to 61	$10,000
60 to 51	$25,000
50 to 41	$50,000
40 to 31	$75,000
30 to 21	$100,000
20 to 11	$250,000
10 to 1	$500,000
0	$1,000,000

Table 4

These new rule changes could have an effect on the expected value of the game overall and will definitely change all of the dynamics of the game and the analysis as we have done it into a completely different model that will still use a dynamic programming approach.

A few other things that we could look at would be those such as decreasing p and q values as the round number gets higher and higher to try to make up for the greater difficulty of the questions in the higher rounds; coming up with a formula or function for the probability values for both the player and the mob in order to accurately describe what is going on (this could be done by either evaluating and devising a function or taking the function from analysis of the game show itself to get an idea of the actual p and q values); and evaluating the game based on the risk-seeking behavior that is always present on game shows – this is contrary to human nature since we are naturally risk-neutral most of the time, but on game shows, this behavior tends to be risk-seeking due to nerves, adrenaline, the crowd, and a number of other things. So, as you can see, there is still quite a bit more work that could be done on our model and will hopefully be done in the near future.

5. References

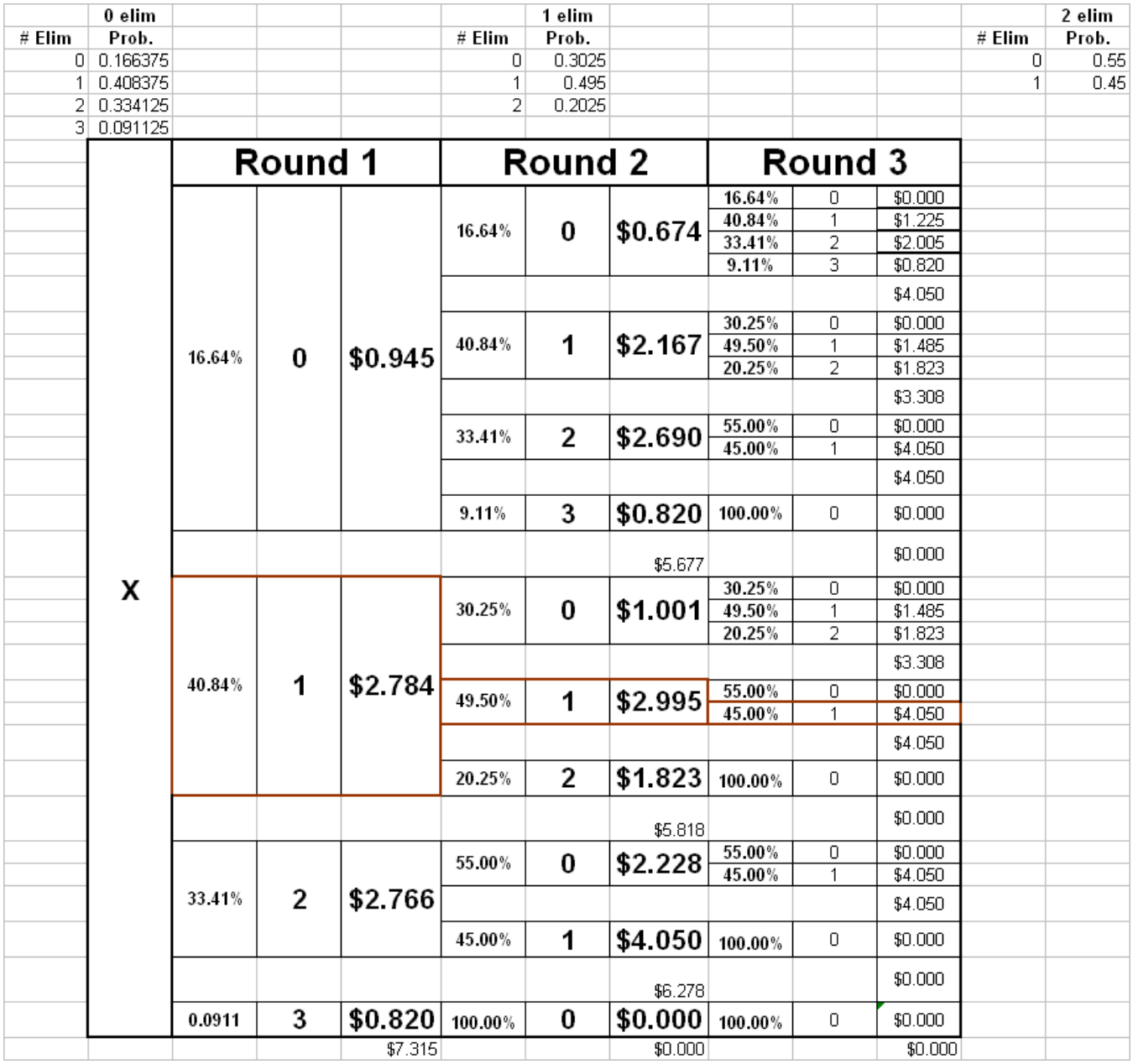
Perea, Federico and Justo Puerto. "Dynamic programming analysis of the game “Who Wants to be a Millionaire?"." European Journal of Operational Research 01 Dec 2007 805-811. 08 Jan 2008.

"1 vs. 100 official rules." 1 vs 100. 18 Feb 2007. NBC. 25 Apr 2008 <http://www.nbc.com/Casting/Applications/1v100_rules.pdf>.

Dreyfus, Stuart. The Art and Theory of Dynamic Programming. Berkeley: ASUC Custom Publishing Service, 1996.

6. Appendices

Carefully calculating and going through our own mini-dynamic programming scheme for this simplified version of the game, yielded the following figure:

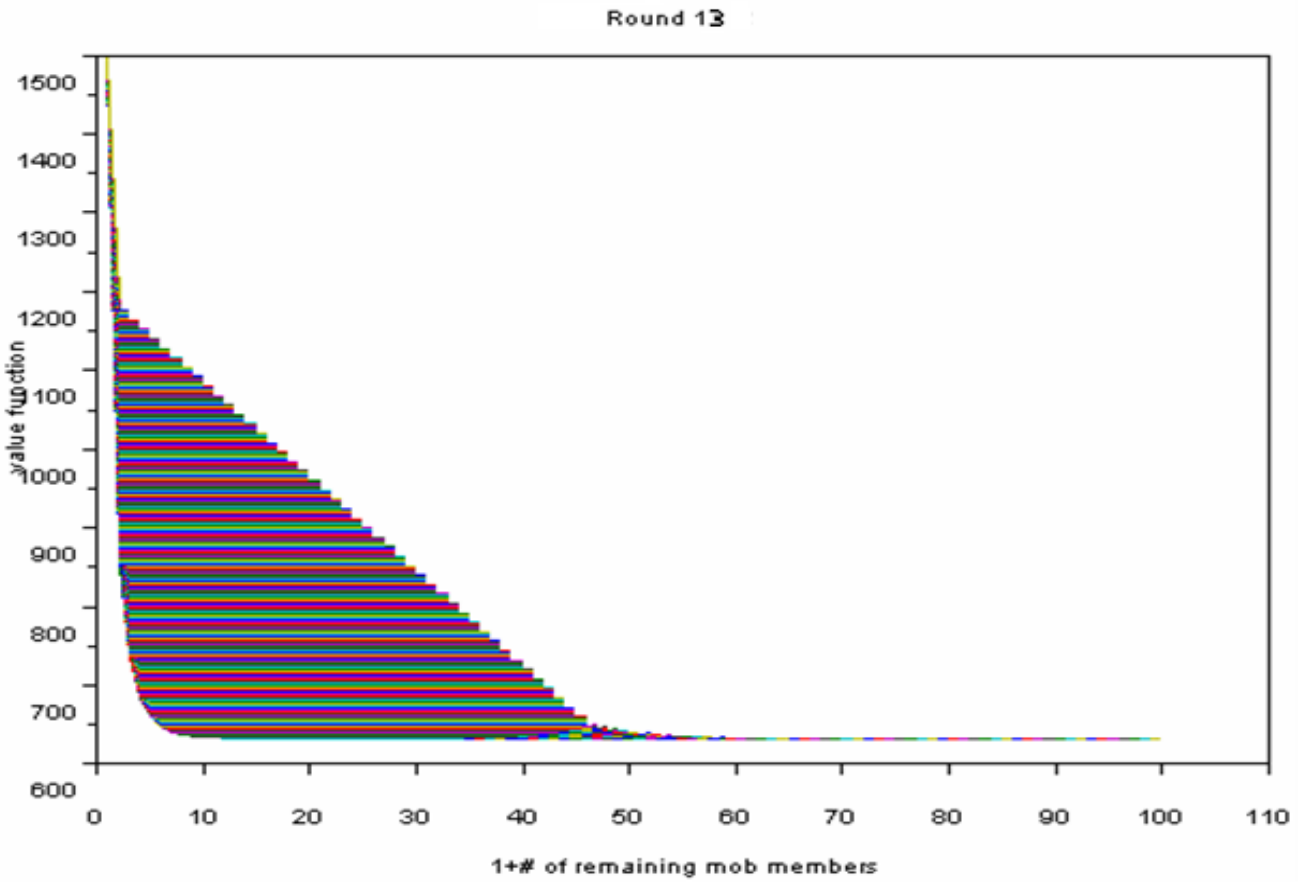


	0 elim					1 elim						2 elim
# Elim	Prob.				# Elim	Prob.					# Elim	Prob.
0	0.166375				0	0.3025					0	0.55
1	0.408375				1	0.495					1	0.45
2	0.334125				2	0.2025						
3	0.091125											

	Round 1			Round 2			Round 3		
	16.64%	0	$0.945	16.64%	0	$0.674	16.64%	0	$0.000
							40.84%	1	$1.225
							33.41%	2	$2.005
							9.11%	3	$0.820
						$4.050			
				40.84%	1	$2.167	30.25%	0	$0.000
							49.50%	1	$1.485
							20.25%	2	$1.823
						$3.308			
				33.41%	2	$2.690	55.00%	0	$0.000
							45.00%	1	$4.050
						$4.050			
				9.11%	3	$0.820	100.00%	0	$0.000
						$5.677			$0.000
X	40.84%	1	$2.784	30.25%	0	$1.001	30.25%	0	$0.000
							49.50%	1	$1.485
							20.25%	2	$1.823
						$3.308			
				49.50%	1	$2.995	55.00%	0	$0.000
							45.00%	1	$4.050
						$4.050			
				20.25%	2	$1.823	100.00%	0	$0.000
						$5.818			$0.000
	33.41%	2	$2.766	55.00%	0	$2.228	55.00%	0	$0.000
							45.00%	1	$4.050
						$4.050			
				45.00%	1	$4.050	100.00%	0	$0.000
						$6.278			$0.000
	0.0911	3	$0.820	100.00%	0	$0.000	100.00%	0	$0.000
			$7.315			$0.000			$0.000

In this model, we considered the possibility of the player eliminating all three mob members as a win and thus rewarded the jackpot at whichever point in the game the player achieved this. The first number under each round heading represents the probability of that particular state which was computed using a binomial distribution dependent on the number of mob members left, the number answering the question wrong and the probabilities of correctly answering for the mob and for the player. The second column underneath the row heading is the number of mob members eliminated in that particular round and the third column represents the expected value, then, of each state possible in every single round. The final number underneath the expected values for round one is the expected value of the game. All of these values were computed given a mob probability of $q = 0.55$ and a player probability of $p = 0.40$. We have highlighted the optimal path (based on expected value) in red in the figure above. In order for this to be a real dynamic program, we began our analysis and computation for the last round first then computed the conditional expected values in each round prior until we came to the first round and were done with our computations overall.

Another useful grouping of data that we were able to obtain from the data was a transpose matrix of the value function of the game for a given round vs. the number of mob members remaining, which appears as follows:



The analysis of this graph is that the lines represent the value of the game given a certain number of mob members left for round 13 which is given above; that is that the colorful lines are what the game is worth, all other things considered in round 13 with the differing numbers of mob members left in the game. This graph is actually a transposed three-dimensional graph projected on a two-dimensional plane where the curvature of the tail actually causes this graph to curve up on itself near the bottom right where the colors are more sporadic and dispersed.