

SIMULATION ANALYSIS OF THE GAME SYSTEM FOR PUBG

Project Final Report IE7215 Simulation Analysis - Spring 2020



APRIL 17, 2020

NORTHEASTERN UNIVERSITY
Boston, MA

Instructor: Wei Xie

Group 7

Jiacheng Ying	001058426
---------------	-----------

Yueqi Shen	001080838
------------	-----------

Kuangyu Jin	001347129
-------------	-----------

Hanzhang Xiong	001356252
----------------	-----------

Table of Contents

Problem Definition and Objectives	1
1.1 Introduction of PUBG	1
1.2 System Description	1
1.3 Objective	1
Simulation Modeling and Validation	3
2.1 Modeling Assumptions	3
2.2 Modeling	6
2.2.1 Input Process	6
2.2.2 Controllable Design Factors	6
2.2.3 Improvement for System	7
2.2.4 Define Entities, Variables in Sections	7
2.2.5 Model Description	7
2.3 Validation	8
2.3.1 Principles	8
2.3.2 Processes	9
Input Modeling and Uncertainty Quantification	12
3.1 Uncertainty Quantification	12
3.2 Input Modeling and Uncertainty Quantification	12
3.2.1 Model Sort	12
3.2.2 Ranking	13

3.2.3 Re-Play Desire	13
Output Analysis	14
Experimental Design and Simulation Optimization	17
5.1 Design	17
5.1.1 First version	17
5.1.2 Second version	17
5.1.3 Third version	17
5.1.4 Forth Version	17
5.2 Optimization	18
5.2.1 Number of Players in One Game	18
5.2.2 Time of One Game	19
Summary	21

Problem Definition and Objectives

1.1 Introduction of PUBG

The system we will model is the servicer system in an online game *PlayerUnknown's Battlegrounds (PUBG)*. PUBG is an online multiplayer battle royale game developed and published by PUBG Corporation, a subsidiary of South Korean video game company Bluehole. What is a battle royale game? For example, in PUBG, games are played between many individual players, pairs of two players or a number of small squads of 4 players. In each match, the goal is to be the last player or team standing by eliminating all other opponents.

In short, in one Battle Royale game, players aim to stand to the last. Battle Royale has become a type of game, and PUBG is the most representative one. PUBG is also one of the most popular computer games in the world. More than 300 thousand people play this game every day. In this game, 100 players will be set in one battle. They need to survive in the game and try to eliminate other players. The activity space in the game will shrink so every player has more probability to face other players. Until the last player wins the battle, game over.

1.2 System Description

When players log into the game servicer, they will choose a game mode. They can choose single-mode, two-person mode or 4-person mode. After choosing, they will be held on the waiting list. When any mode has 100 persons on the waiting list, the system will start a battle for them. Every battle in the game lasts 30 minutes and players will lose and quit the battle during the game until the last person or the last group wins the battle. After players quit the battle, they will choose to continue the next battle or just stop the game. If they want to continue, they can choose to change the game modes or not. And then they will be held on the waiting list again and start the next battle.

1.3 Objective

The capacity of the service system in PUBG game depends on how many players in the system and how long they will stay there. If the capacity of the servicer is large, game companies need to spend more and hire more engineers to maintain the servicer than the servicer capacity is small. Therefore, how to decide the capacity of the game servicer is an important question for game companies. There are some factors that may influence the player in the system. For example, the duration of each battle, players quitting time, the number of people in each team and the number of people in each battle. Adjusting these factors will cause some influence to players and change

the number of players in the system.

What we aimed do is to determine the maximum number of matches or games, which is also the number of servers, to help the game company know how much equipment they need and cut cost. They may also be able to minimize the peak number of servers without decreasing the number of players by changing some parameters of the game and save money, such as how long one game lasts or how many players are in one game, which would also be involved in the following contents.

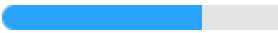
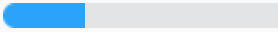
Simulation Modeling and Validation

2.1 Modeling Assumptions

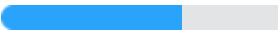
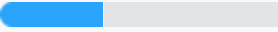
We started off with an NSPP (non-stationary Poisson Process) to generate the players, and then split them through three paths (Single row, double row and four row modes), then we calculated the time spent on each path, and calculate the service number. And there are many factors that will affect the final number of services we estimate, and we focused on some assumptions that might affect our data.

The first assumption is that the influence between player's position and the performance in each game, and the probability of entering the next game. And in order to apply this hypothesis to our system, we conducted a questionnaire survey to calculate the influence of rank on players' willingness to continue playing the game.


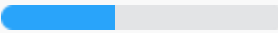
Do you prefer to continue the game after you get No.1?

Options	Sums	Ratios
Yes	238	 71.04%
No	97	 28.96%
Valid Participants	335	

Do you prefer to continue the game after you get top 50?


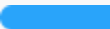
Options	Sums	Ratios
Yes	214	 63.88%
No	121	 36.12%
Valid Participants	335	

Do you prefer to continue the game after you get 50 ~ 90?

Options	Sums	Ratios
Yes	200	 59.7%
No	135	 40.3%



Valid Participants	335	
--------------------	-----	--

Do you prefer to continue the game if you lose the battle very soon?



Options	Sums	Ratios
Yes 会	205	 61.19%
No 不会	130	 38.81%
Valid Participants	335	

The second assumption is that the probability that the player moves to the next game in different modes. And In order to apply this hypothesis to our system, we conducted a questionnaire survey to calculate the influence of rank on players' willingness to change the model of the game.


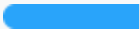
Do you prefer to change the model after you get the No.1?

Options	Sums	Ratios
Yes	118	 49.58%
No	120	 50.42%
Valid Participants	238	

Do you prefer to change the model after you get top 50?

Options	Sums	Ratios
Yes	106	 49.53%
No	108	 50.47%
Valid Participants	214	

Do you prefer to change the model after you get the top 50 ~ 90?

Options	Sums	Ratios
Yes	104	 52%
No	96	 48%
Valid Participants	200	

Do you prefer to change the model if you lose the battle very soon?

Options	Sums	Ratios
Yes	108	52.68%
No	97	47.32%
Valid Participants	205	

The third assumption is that the probability that there are different number of players entering the game varies from time to time. And In order to apply this hypothesis to our system, we conducted a questionnaire survey to calculate the different ratio of people who are intend to play the game in different time.

Which time do you usually play PUBG (PUBG Mobile)?

Options	Sums	Ratios
Before 7:00	26	7.76%
7: 00-9: 00	83	24.78%
9: 00-11: 00	72	21.49%
11: 00-13: 00	54	16.12%
13: 00-15: 00	42	12.54%
15: 00-17: 00	27	8.06%
17: 00-19: 00	36	10.75%
19: 00-21: 00	89	26.57%
21: 00-23: 00	89	26.57%
23: 00-1: 00	30	8.96%
After 1:00	15	4.48%
Valid Participants	335	

2.2 Modeling

2.2.1 Input Process

We use non-stationary passion distribution to generate the people who participate in the game. The code is showed in the following graph.

```
def Arrival():  
    if 0 < SimClasses.Clock < 720:  
        MeanTBA = 3.23  
    elif 720 < SimClasses.Clock < 2*720:  
        MeanTBA = 10.325  
    elif 2*720 < SimClasses.Clock < 3*720:  
        MeanTBA = 8.954  
    elif 3*720 < SimClasses.Clock < 4*720:  
        MeanTBA = 6.717  
    elif 4*720 < SimClasses.Clock < 5*720:  
        MeanTBA = 5.225  
    elif 5*720 < SimClasses.Clock < 6*720:  
        MeanTBA = 3.358  
    elif 6*720 < SimClasses.Clock < 7*720:  
        MeanTBA = 11.07  
    elif 7*720 < SimClasses.Clock < 8*720:  
        MeanTBA = 11.07  
    elif 8*720 < SimClasses.Clock < 9*720:  
        MeanTBA = 3.733  
    else:  
        MeanTBA = 1.867  
  
    interarrival = SimRNG.Expon(1/MeanTBA,1)
```

Due to survey of people who participate in the game, we predict each rate of the people in different time, and generate people by this way.

2.2.2 Controllable Design Factors

The controllable factor is the rate of each people generated and the number of people in a service and the time to finish one game.

2.2.3 Improvement for System

In order to improve the performance of the system and accuracy, we have done a lot of questionnaire survey, the data have been obtained, and the data was analyzed at the same time, we generate the number of cycles have been obtained, and the probability of players to choose different game modes and players under different circumstances will a game of probability.

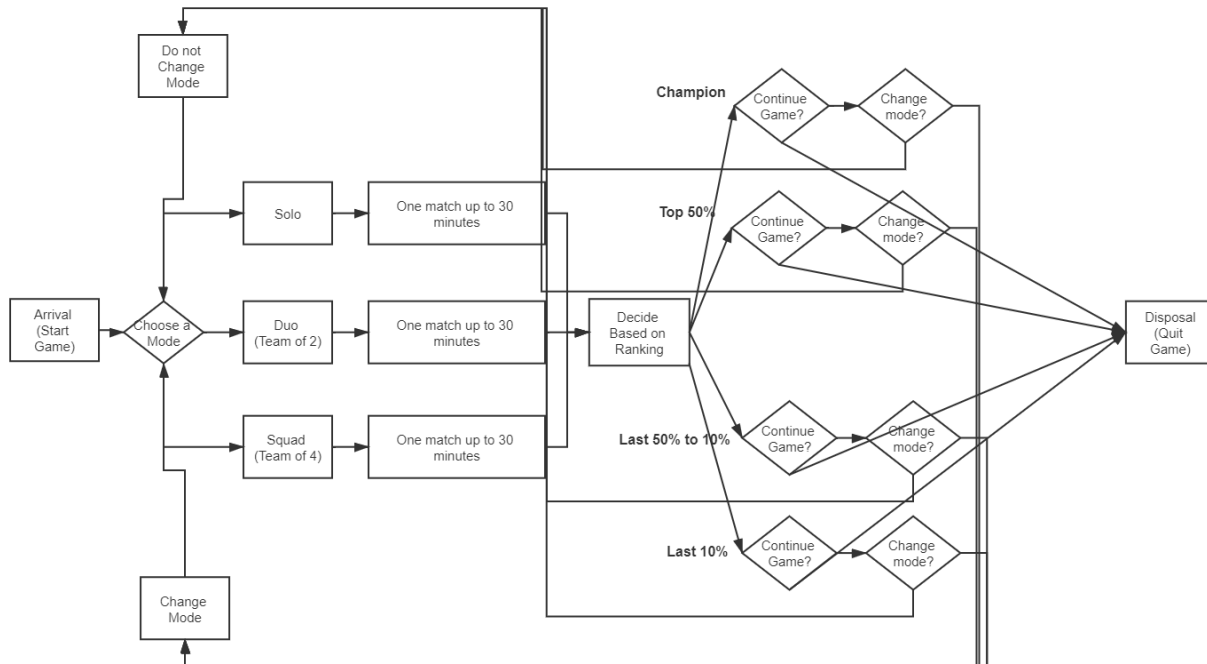
2.2.4 Define Entities, Variables in Sections

Definitions	Descriptions
Count	Count the number of times a player enters a column
servernumber	The number of servers
MeanTBA	mean time between arrival
MeanST	mean service time
Runlength	The length of run
BattleNum	The number of people in one server
AllWait1Mean	The number of people who wait to enroll in one people channel
AllQueue1Mean	The number of people in one people channel
AllWait2Mean	The number of people who wait to enroll in double people channel
AllQueue2Mean	The number of people in double people channel
AllWait4Mean	The number of people who wait to enroll in four people channels
AllQueue4Mea	The number of people in four people channels
AllQueueNum	The number of people in all channel
AllServerMean	The mean number of servers in whole system
AllServerMax	The most numbers of servers in whole system

2.2.5 Model Description

People in this system are 20 percent more likely to go into a single row, 40 percent more likely to go into a double row, 40 percent more likely to go into a fourth row, and then they assume that the average length of the game is 30 minutes, and then they leave the game to complete a loop. After leaving the game, people with different grades would have different probabilities to choose whether to play the game or not. We divided the players into the top 1, those with the top 50% grades, those with 50% ~ 90% grades and those die soon who would be the first 10% to quit the game and rank at last 10%. Then different people have different possibilities to choose whether to go to the next game. At the same time, the system will randomly generate a number of people to join the game, so that the last game to choose the continuation of the people and the newly

generated people queue up together into a new cycle. With this system, we need to calculate the number of people waiting in the channel and the waiting time every day, as well as the average number of servers and the maximum number of servers.



2.3 Validation

For the validation part, we would like to see if our program and simulation were modeled correctly. We used Arena to verify our simulation done by programming. Arena is a discrete event simulation and automation software developed by Systems Modeling and acquired by Rockwell Automation in 2000, which is also exactly suitable for our project.

2.3.1 Principles

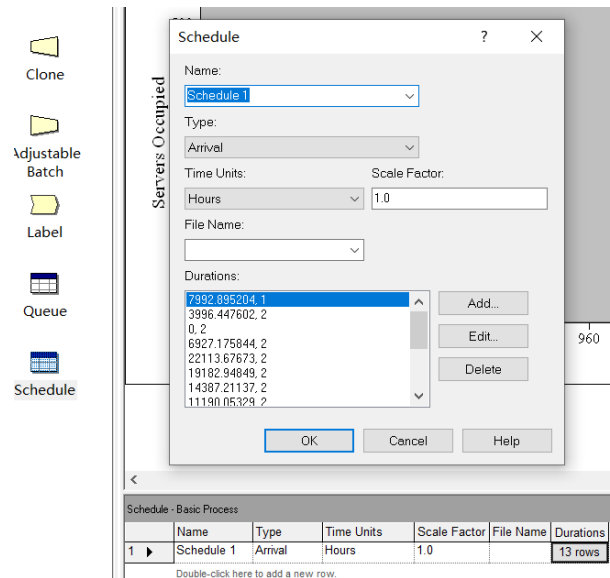
When built the Arena model, we followed some principles:

- (1) Principles Followed when built the model;
- (2) Satisfying the result of the questionnaire survey;
- (3) Using the same theory as the programming model;
- (4) Referring to some actual data provided by the game's company.

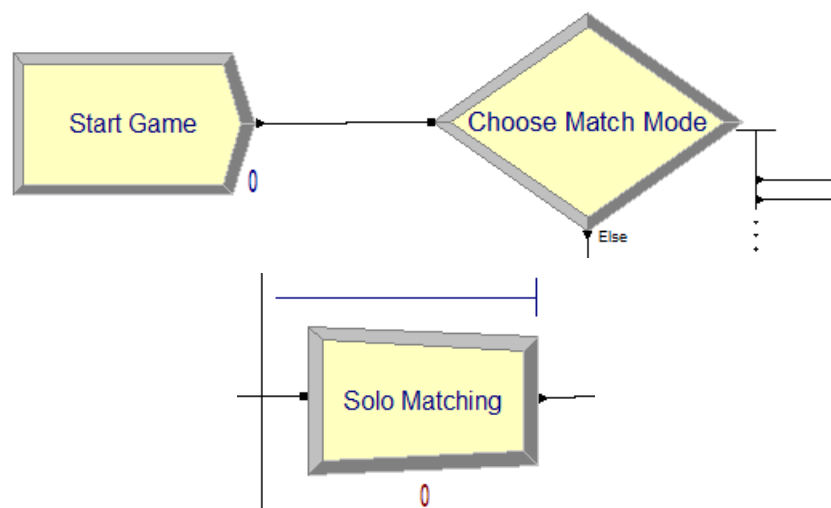
2.3.2 Processes

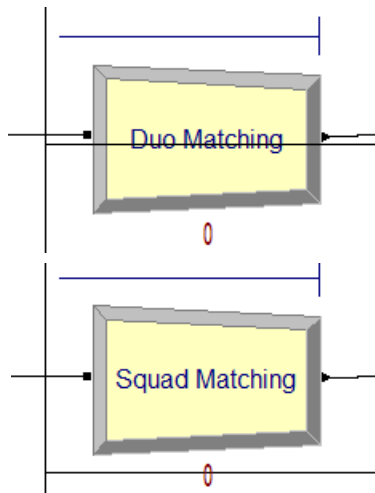
Let's see the model from the arrival. For the arrival of the game, we need to generate Non-Stationary Processes to make the arrivals follow Non-Homogeneous Poisson Process (NHPP). This could be done by schedule in Arena. First, we collected and organized our data into 24 hours from survey, then according to the official statistics of 300,000 players per day, we input the distribution into our model.

	A	B	C	D	E
1	15	0.026643	7992.895	1	7992.895204
2	15	0.026643	7992.895	2	3996.447602
3	0	0	0	2	0
4	26	0.046181	13854.35	2	6927.175844
5	83	0.147425	44227.35	2	22113.67673
6	72	0.127886	38365.9	2	19182.94849
7	54	0.095915	28774.42	2	14387.21137
8	42	0.0746	22380.11	2	11190.05329
9	27	0.047957	14387.21	2	7193.605684
.0	36	0.063943	19182.95	2	9591.474245
.1	89	0.158082	47424.51	2	23712.25577
.2	89	0.158082	47424.51	2	23712.25577
.3	15	0.026643	7992.895	1	7992.895204
.4	563	1	300000	24	

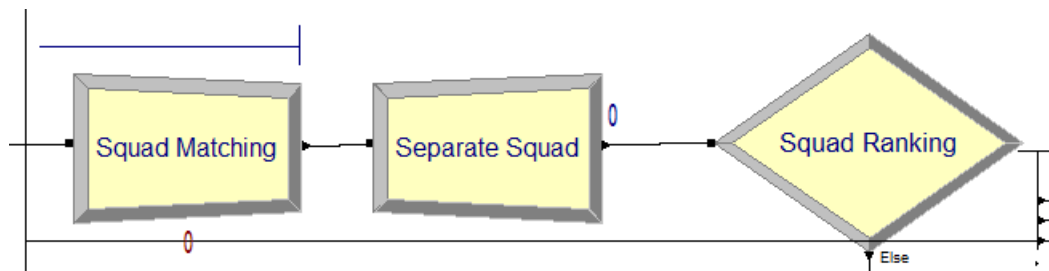


Next, the players would choose 1 of 3 game mode: Solo, Duo and Squad (to play individually or in a team of 2 or 4 members). The probabilities of their selection have been obtained by input analysis.





For every mode, the matching rule is to wait up to 60s or 100 players. Then one game would start and one server would turn into being busy. Each game lasts for 30 minutes. This is a $G/G/\infty$ model. So that resource capacity is infinite.



Adjustable Batch ? X

Name: Squad Matching Type: Temporary

Optimum Batch Size: 100 Save Criterion: Last

Representative Entity Type: Player Partial Batch Method: Maximum Wait Time

Maximum Wait Time: 1 Units: Minutes

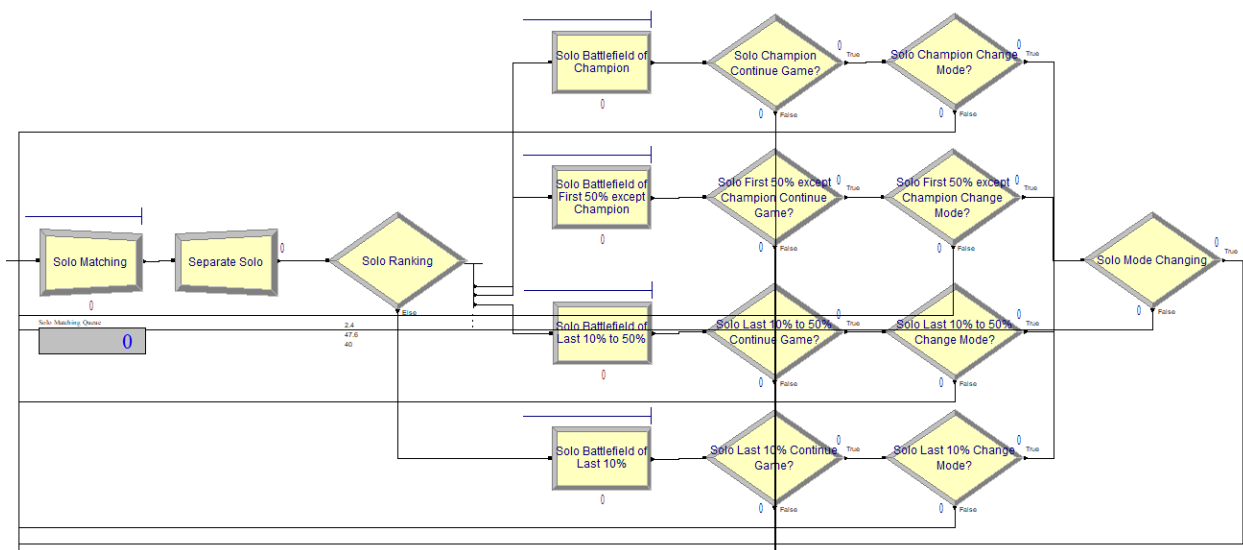
OK Cancel Help

Resource - Basic Process									
	Name	Type	Capacity	Busy / Hour	Idle / Hour	Per Use	StateSet Name	Failures	Report Statistics
1 ▶	Server	Fixed Capacity	Infinite	0.0	0.0	0.0		0 rows	<input checked="" type="checkbox"/>

Here, please let me explain our model again. In the questionnaire, we divided players into 4 groups, to know their preferences about whether to continue the game or change the game mode of the next match, according to their ranking in one game they just played. The 4 groups are:

- (1) Champion: Top 1
- (2) Above average: Top 50% except the champion
- (3) Mid: Ranked 50% to 90%
- (4) Losers: Last 10%

Therefore, in the model we also divided players into 4 groups, to predict their behavior after each game. On the basis of game rules, we used Uniform Distribution to determine the time players come out from one game.



By the way, there are normally thousands of entities in this model. To bypass the 150 entities limitation, we used the academic version Arena in the Virtual Lab of COE.

Input Modeling and Uncertainty Quantification

3.1 Uncertainty Quantification

To make the simulation smoothly. It's necessary to connect the simulation output with input models using some parameters. Denote the input models by $F = \{F_1, F_2, F_3, \dots, F_l\}$, where F_l with $l = 1, 2, \dots, L$ quantifies the uncertainty from the probability of modeling choosing, ranking level in the games and re-play desire. According to the potential function of the input modeling by $x = (X_1, X_2, X_3, \dots, X_k)$. Depending on the input models and parameters, the simulation output is $Y(x, F) = \{Y_{ji}(x, F), j = 1, \dots, J; i = 1, 2, \dots, n\}$, where $Y_j(x, F) = (Y_{j1}(x, F), \dots, Y_{jn}(x, F))$ is the output based on j -th replication. Obviously, $Y_j(x, F)$ is the i -th batch result. For example, it could be either gaming lasting time per person or the size of the whole PUBG system.

3.2 Input Modeling and Uncertainty Quantification

After analyzing the input modeling process between each step, we provide each dominant source of uncertainty in PUBG Games. According to the corresponding real-word data, we would put the detailed procedure below.

3.2.1 Model Sort

In this input part, there are three conditions. Firstly, one player with other 99 single players managed by one server. In double-player part, one double-player group with other 49 groups share one server. In the last part, 25 4-person group share one server. We donate the sever number $x_1 = \mu_1 x_0 + \mu_2 x_0 + u_3 x_0$. From the official website, we get the probability of users' modeling choosing willing in single queue, double queue and 4-player queue are 0.2, 0.4 and 0.4. Notice that the data in this part is similar to the multinomial distribution, we would use Chi-square test to verify our survey with the official data. Firstly, we design a PUBG survey to get the real-word data then compare them with official data. During the Chi-square test, we make a hypothesis that there are significant differences between our survey and official data in 95% CI. The survey data and official data are listed in the table below.

	One-person queue	Two-person queue	Four-person queue
Expected time	86	173	173

Observed time	96	164	154
---------------	----	-----	-----

After that we use formulation:

$$\chi^2 = \sum \frac{(\text{Observed time} - \text{expected time})^2}{\text{expected time}}$$

$$k = (r - 1) \times (c - 1)$$

We got $\chi^2=3.56$ and K (The degree of freedom) is 2. According to the χ^2 in percentage point χ^2 of Chi-square Distribution Table, $\chi^2=5.991$ when the freedom degree level is two and P-Value is 0.950. Obviously, the result $3.56 < 5.99$ rejects the hypothesis posted before. Therefore, we could use the parameters of multinomial distribution within this experiment. In this input part, the probability for each queue are corresponding to the relative parameters.

3.2.2 Ranking

The ranking process between the players constitutes the middle part of the whole system. We can assume that this step doesn't change the number of servers in the system. The reason is that it only changes the battle situation in one game but have no function on the servers' number So we could get

$$x_2 = x_1$$

.

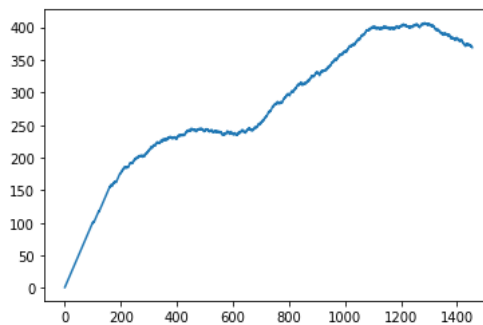
3.2.3 Re-Play Desire

After conducting the game, there will be different ranking levels between players. In the reality, ranking level influences whether players will continue another game or not. We assume that $x_3 = Qx_2$. The distribution of random Q could be estimated by using data $(x_{3,i}, x_{2,i})$ with $i= 1,2, \dots, m$ for four different ranking level. After using bootstrap, we are able to assume Q follows Uniform distribution, $Q_1 \sim (0.46, 0.48)$; $Q_2 \sim (0.47, 0.49)$; $Q_3 \sim (0.5, 0.51)$; $Q_4 \sim (0.5, 0.51)$.

Output Analysis

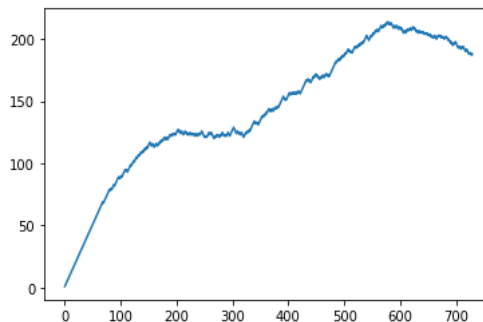
There is only one replication and 20 hours running length. Because first the procession of one replication is so long; second, the result of the server is only a little different by another replication; and last, we would have verified this system with Arena, so there is only one replication of this system.

The graph below shows our output.



The observation of our project is that when there are more numbers of people in one server (BattleNum), there would be less numbers of mean servers, and max servers will be output. And this finding is expected and controllable.

Because our variable is adjustable in the simulator, we can adjust different variables for the same customers in the simulator of a different cycle and it is concluded that the demand of the different server, through these requirements change, we can provide better suggestion to the present simulator to make the game company can save costs to operate the game. We try to change the factor named 'BattleNum' and 'MeanST' and analyze the change of the server number in the project. So, we change the BattleNum from 100 to 200 to see the output again.



Then We ran a 5 replication of 24 hours in Arena to verify the result. The peak number of servers in use is 536.

Resource

Usage

Number Busy	Average	Half Width	Minimum Average	Maximum Average	Minimum Value	Maximum Value
Server	247.07	1.50	245.99	249.16	0.00	536.00
Total Number Seized	Average	Half Width	Minimum Average	Maximum Average		
Server	11967.40	68.45	11928.00	12064.00		

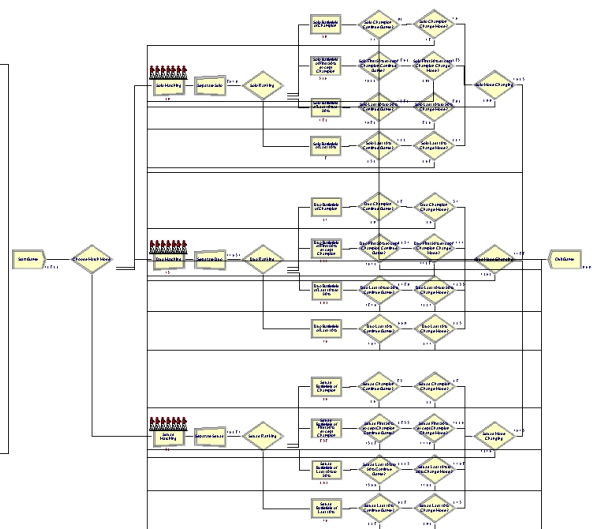
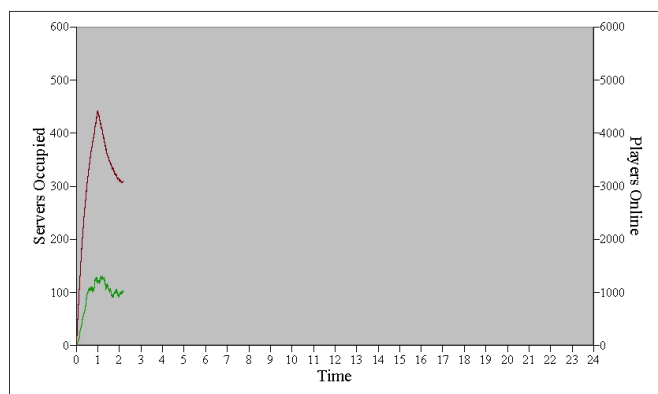
We also verified the result in the programming to change the number of players in one match from 100 to 200. The result is also similar to what we had from the programming model.

Resource

Usage

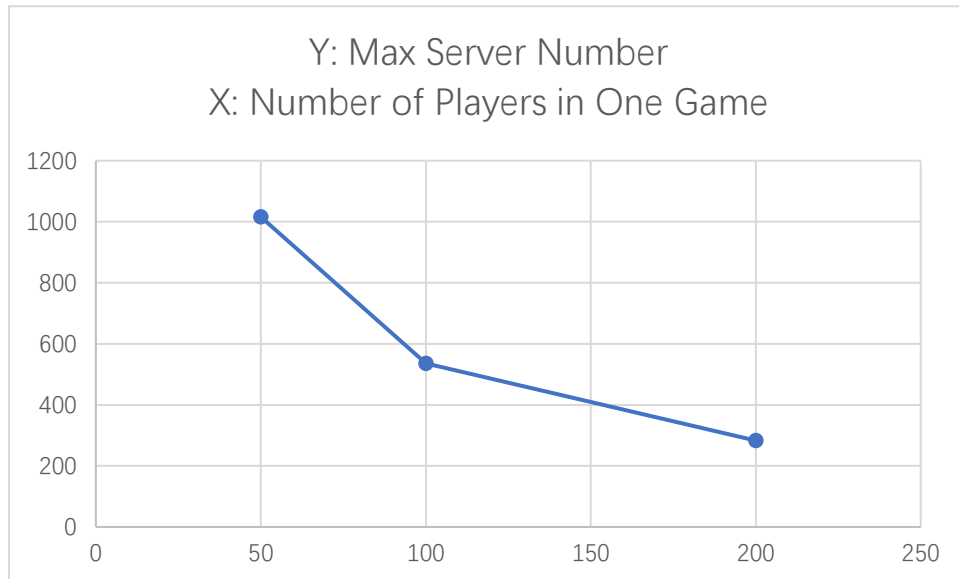
Number Busy	Average	Half Width	Minimum Average	Maximum Average	Minimum Value	Maximum Value
Server	125.94	2.84	123.18	127.92	0.00	283.00
Total Number Seized	Average	Half Width	Minimum Average	Maximum Average		
Server	6102.40	129.32	5978.00	6197.00		

The confidence interval estimation for the above two simulations are 1.50 and 2.84. And a line chart of it and the number of players online could be drawn.



To verify our conclusion again, we used Arena results to draw a graph of three Max server numbers, to see their trend when BattleNum Changes.

Number of Players	Time / min	Ave Serves Number	CI	Max Serves Number
50	30	495.68	2.77	1016
100	30	247.07	1.5	536
200	30	125.94	2.84	283



What we could conclude from the above part it that: To Minimize the capacity of the service system of PUBG, the game could consider increasing the number of players in one game.

Experimental Design and Simulation Optimization

5.1 Design

5.1.1 First version

The main program is PUBG simple.py. It's a simplified procedure. Realized the player according to a certain probability to generate probability to enter the team, the team full of a certain number of people or reach a certain time to grab the server into the game, the game in the player according to the probability to gradually quit, after the exit of the player according to the probability to choose whether to continue the game, after a game of all players are finished after a server.

5.1.2 Second version

Changed the single thread to 3 threads. Each thread has a separate team, but a Shared server.

5.1.3 Third version

When the game is over, the probability selection is added.

Die Soon will be the first 10% to quit the game, and 60% of the users will choose whether to continue the game or not.

The first 50% quit the game is Half, according to the probability to choose whether to continue the game.

The remaining players are Almost Win, who, with the exception of the chicken eaters, have a chance to continue playing.

Players who eat chicken choose whether to continue the game or not.

5.1.4 Forth Version

Probability of joining the change mode were added based on our survey.

After a game, if the player chooses to continue the game, the player will choose whether or not to

switch the game mode according to their ranking.

If you choose not to switch, enter the original mode queue.

If you choose to switch, enter the queue of the other two modes according to the probability.

Modified the arrival rate to conform to the actual situation of non-homogeneous distribution.

Modified the mean service time to be 18 seconds, that is, to complete 100 customers in an average of 30 minutes.

The modified mode probability is the actual probability.

Fixed an issue where the server could not be released.

About the problem that how the simulation system improves the real system, we can change the factor in the system which will lead the change in the result and help us to analyze and predict the real system change.

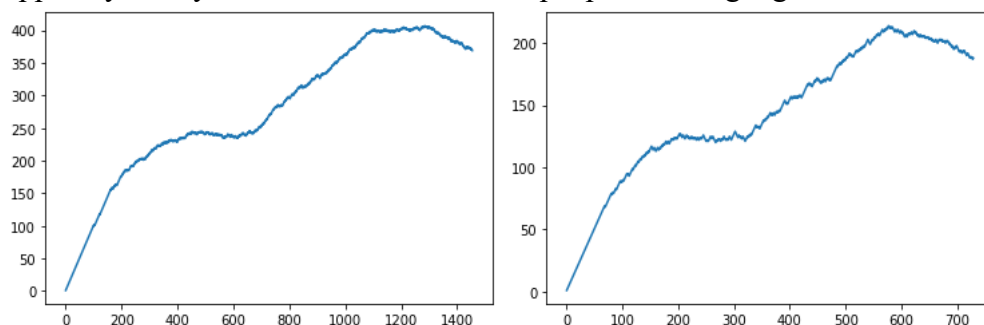
5.2 Optimization

5.2.1 Number of Players in One Game

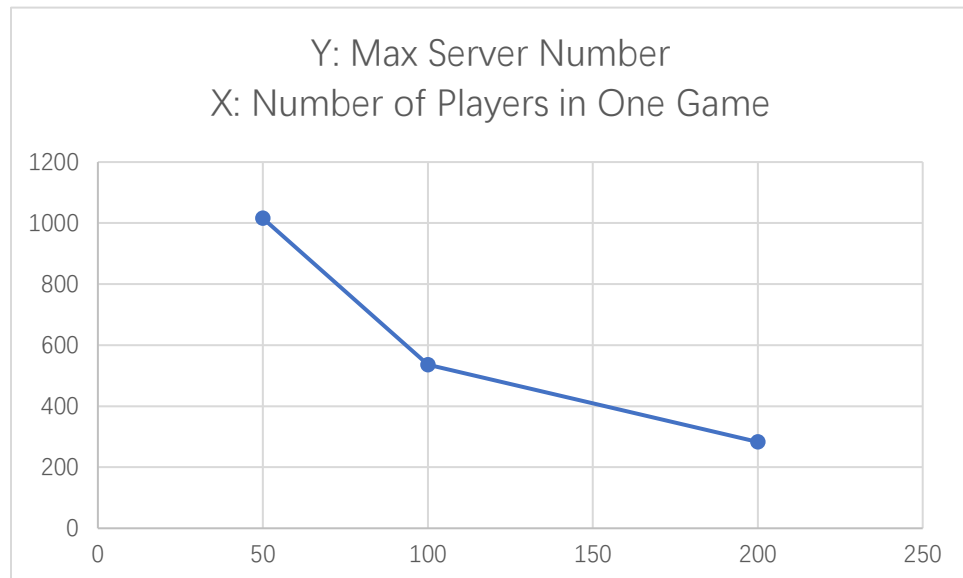
The factor named server number will be studied because we need to deduce the number of the server number to help the game company to save money to operate the game.

In order to lead the simulation system more like the real system, we optimize several times to make the simulation program more fit the game.

To reduce the number of servers, we changed the frequency of a single game and expanded the number of people playing the entire game. By running the program, we found that the number of servers dropped by nearly half when the number of people in a single game increased to 200.



We could also see the trend of the number of servers when we have 50, 100 and 200 players in one game.



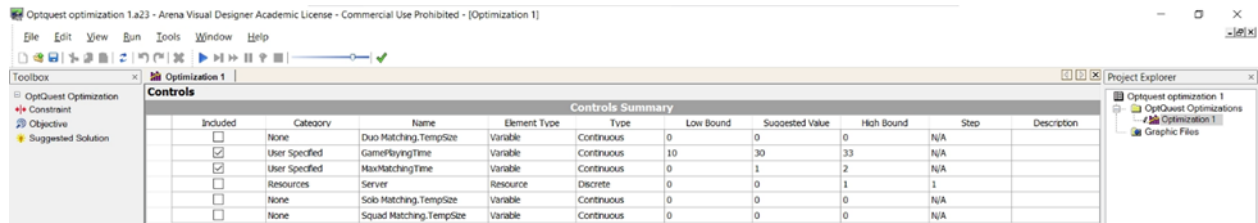
5.2.2 Time of One Game

Optimization could also be done by Arena. In this way, some of our parameters need to be set as decision variables.

Maximum Wait Time:	Units:
MaxMatchingTime	Minutes

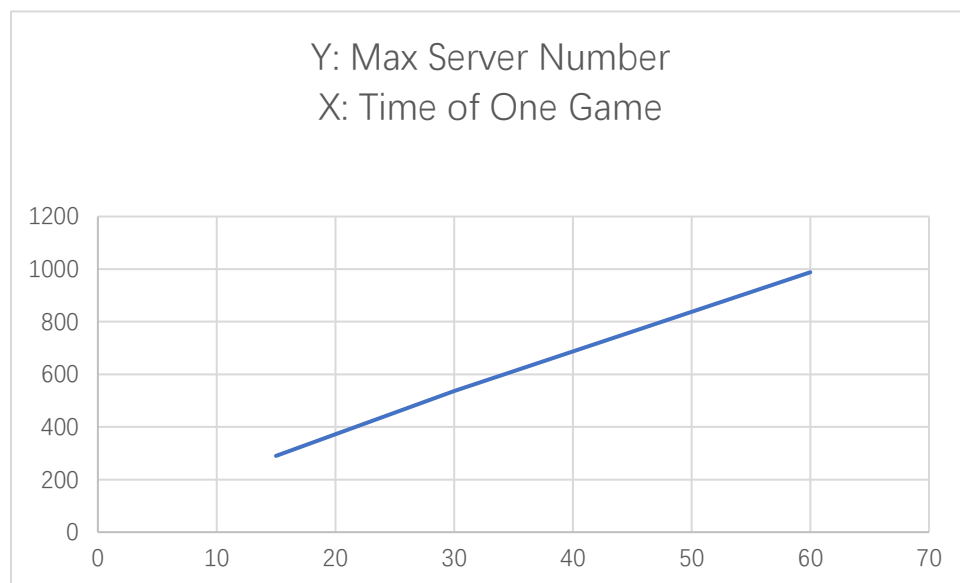
Variable - Basic Process									
	Name	Comment	Rows	Columns	Data Type	Clear Option	File Name	Initial Values	Report Statistics
1 ▶	MaxMatchingTime				Real	System		1 rows	<input type="checkbox"/>
2	GamePlayingTime				Real	System		1 rows	<input type="checkbox"/>

Of course, we could also use OptQuest to achieve optimization using the automatic search approach.



Yet we chose to do is to do the Optimization using a Trial-and-Error approach by changing some parameters and drawing the charts manually.

Number of Players	Time min	Ave Serves	CI	Max Serves
100	15	124.25	2.36	290
100	30	247.07	1.5	536
100	60	480.45	5.79	988



For the time of each game, in the default mode it's 30 minutes after Changing the time of one game (15min, 30min and 60min), we found that the number of servers is proportional to the time of one game.

Summary

Our Conclusions are:

- (1) For the PUBG game now, the Capacity of Servers number should be set according to our results of max server number, which is about 400-600 games at one time.
- (2) To Minimize the capacity of the service system of PUBG, the game company could consider increasing the number of players in one game.
- (3) The game company could consider decreasing the time of each game.