ISSN(Online) : 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 4, April 2016

Implementation of Board Games Using Monte Carlo Simulation

Swathi.Y[1], Manoj Challa[2]

Associate Professor, Dept. of CSE, CMR Institute of Technology, Bangalore, India[1].

Associate Professor, Dept. of ISE, CMR Institute of Technology, Bangalore, India [2]

ABSTRACT: Monte Carlo simulation is a problem solving technique that is used to approximate the probability of certain outcomes based on running simulations multiple times. A Monte Carlo simulation involves creating models of possible outcomes by using a range of values, or a probability distribution, for every result that has some factor of uncertainty. In order to compute the probability distribution, the simulation must be performed a large number of times. Each simulation is a separate and independent "future" for the system. Every time the simulation is repeated, it is called an iteration- this is essentially a set of samples from which the resulting outcome is recorded. Once the different results are calculated, they are normally put into a graph that can easily be analysed. Many people use Monte Carlo simulations to calculate the probabilities of gambling or winning a board game. In this paper, we will run through the applications of a Monte Carlo simulation to the well-known board game Snakes and Ladders.

KEYWORDS: Monte Carlo Simulation; iteration; Snakes and Ladders

I. INTRODUCTION

The board for Snakes and Ladders is a 10x10 board numbered 1-100. On some squares there are ladders, and if you land on the specific square then you can move your piece up to where the ladder ends (for example, landing on 28 brings you up to 84). To go up a ladder, you must land exactly on the bottom of the ladder. Other squares have snakes, causing you to slide down the board (the tile labelled 62 brings you all the way back down to tile 19). To fall down the snake, you must land exactly at the head of the snake. Players start off the board and roll a single die to determine how many squares to move. In the official version players must land exactly on the tile labelled 100 to win; in our simulation you can either land on 100 or go past it to win.

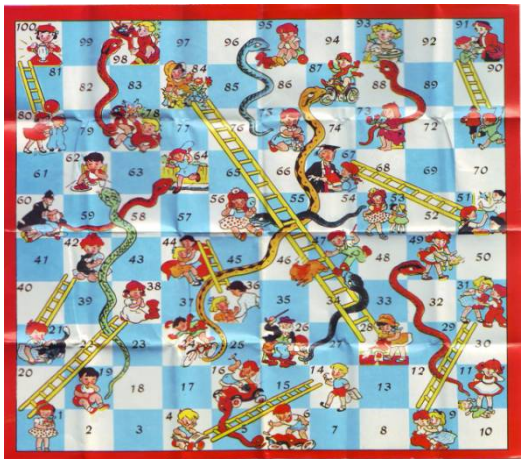

Figure 1 : A Typical Game Board

ISSN(Online) : 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 4, April 2016

II. RELATED WORK

Finite State Machine : This game can be described by finite state machine.We can look at the game as a sort of mechanical process. Any single player starts out o_ the board, then moves to some square on the board, then another and another. The only thing the player needs to know is the current position.In order to be consistent, we can say that, at the start of the game, the player is in square 0. That way, we can completely describe the progress of the game for one player by listing the position.A computational science would call the positions \states"; because there is a limited list of them, the game itself could be described as finite state machine

Inspired by the vending machine diagram, we could think of making a similar transition diagram for the game of Snakes and Ladders.It would consist of 101 circles, labeled \`0" through "100".
We need to draw arrows indicating what jumps or transitions are allowed from one circle to the next.Since we are rolling a die, roughly speaking our rule should be:At circle I, draw arrows to circles I+1, I+1, ..., I+6.The following problems exists as follows:

i)Since we can't go past circle 100, we need to modify our rule:At circle I, draw arrows to each circle I+1, I+1, ..., I+6 that is no greater than 100.

ii)Suppose we land on the foot of a ladder. Naturally, as part of the very same turn, we immediately move up the ladder. So if circle I is the foot of a ladder or the mouth of a snake, we never roll a die there, we just move.
If circle I is a regular circle, draw arrows to each circle I+1,I+1, ..., I+6 that is no greater than 100.
If circle I is a ladder foot or snake mouth, draw a single arrow to the ladder top or snake bottom.

Consider a 3*3 Example :It might help to sketch a small version of our problem:
7 -> 8 -> 9
^ |S|
6 <- 5 <- 4
|L| ^
0 -> 1 -> 2 -> 3
where we only have a 3x3 board, and one ladder, from square 1 to square 6, and one snake, from square 8 back to 5.To keep the game from ending too quickly, let's suppose our die only returns a 1 or 2.To record where the snakes and ladders are, we need:
0 1 2 3 4 5 6 7 8 9 Ladder at 1 goes to 6, connect |0|6|2|3|4|5|6|7|5|9| <-- Snake at 8 goes to 5

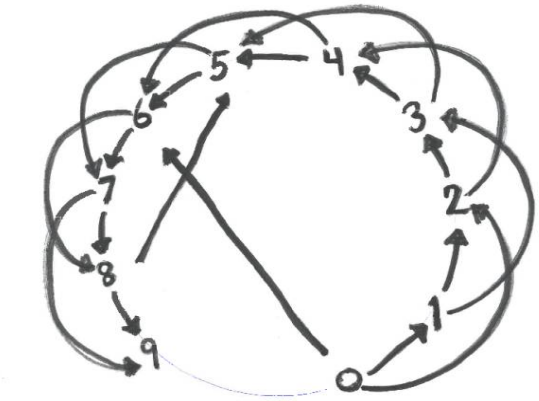

Figure 2:Transition diagram of 3*3 example

ISSN(Online) : 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 4, April 2016

In our diagram, there are two very special states, #0 and #9.State #0 is special because it only has arrows leading out of it.There is no way for the system to return to this state. It is an initial state.State #9 is special because no arrows leave it. If you reach this state, you are stuck there. It is a final state.

III. IMPLEMENTATION

Rolling the Die: For any player, the only important information is the current state,that is, the player's position on the board. This is a number between 0 (starting) and 100 (done).
The computer can remember this value if we give it a name.We'll call this value I, and it will start out with the value 0.
I = 0 <-- Initialize I to zero, almost any language

Making the Move: Having rolled the die, we should consider moving.Although we might step on a snake or ladder, or go past 100, it's perfectly correct to begin by taking the full step.Many computer languages use the equals sign as a way to update or change the value of a variable.
I = I + D <-- The VARIABLE on the left is set to the VALUE of the formula on the right

Changing the Value : In mathematics, a statement like I = I + D is an equality statement, saying the quantities on both sides are equal (and hence D must be 0!).In computing, this is an assignment statement, and has an operational meaning:
go get the value stored in I;
go get the value stored in D;
compute I + D
put this new value back into I.
Defining Snakes and Ladders: To implement the snakes and ladders, we start by making an array or list of the numbers 0 through 100:
final = range (0, 101) <-- 0, 1, 2, ..., 98, 99, 100.
and now, for each square that is a snake or ladder, we replace the square's index by the index of its final destination. The standard board has 19 snakes and ladders, but here is how to set the first five of them as in figure 3.
final[1] = 38;final[4] = 14;final[9] = 31;final[16] = 6;final[21] = 42

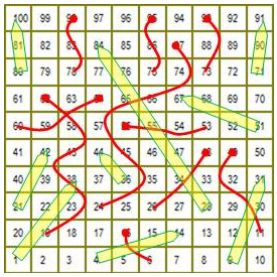

Figure 3: Defining Snakes and Ladders

final[1]=38 final[28]=__ final[56]=__ final[87]=__
final[4]=14 final[36]=__ final[62]=__ final[93]=__
final[9]=31 final[48]=__ final[64]=__ final[95]=__
final[16]= 6 final[49]=__ final[71]=__ final[98]=__
final[21]=42 final[51]=__ final[80]=__

First Move:Now that we know how to make one move, we can play the whole game. Ordinarily, a computer will implement a sequence of commands in the order in which they are given. Thus, the following program takes the first move.
(statements that de_ne _nal go here)
I = 0

ISSN(Online) : 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 4, April 2016

```
D = random_integers ( 1, 6 )
I = I + 6
if ( 100 < I ):
I = 100
I = final [ I ]
```

To take a second move, we could cut and paste another set of move commands. But repetition is an important part of computation. There is a while command that forces the computer to go back and repeat the commands until some condition is true.

```
(statements that de_ne _nal go here)
I = 0
while ( I < 100 ): <-- Until reaching square 100
D = random_integers ( 1, 6 )
I = I + 6
if ( 100 < I ):
I = 100
I = final [ I ]
```

IV. COMPUTER SIMULATION

Now we have implemented a sequence of computer statements that do the same thing as Snakes and Ladders. There is no board involved, no one rolls a die, but if we wanted to, we could keep track of the changes in the variable I, and show that they corresponded to a legal game.What we have done, therefore, is simulated the game of Snakes

We have seen that the computer can count the number of moves it takes in order for one player to play the game to completion.It's as easy for the computer to play the game thousands of times,and record the number of moves every time.By dividing the total number of moves by the number of games,we get an estimate for the average number of moves.Investigating a system with inherent randomness in this way is called the Monte Carlo method.This is as shown in the below table 1.

Trial	Average	Shortest	Longest
1	39.8	7	175
2	39.2	7	185
3	38.7	7	158
4	39.5	7	205
5	38.5	7	187
6	38.3	7	198
7	39.5	8	207
8	38.8	7	176
9	39.9	7	185
10	39.0	7	242

Table 1 : Monte Carlo method

It is popular for problems when there is no obvious mathematical way to compute an answer,it seems like 1000 Monte Carlo simulations should be enough for good estimates.Comparing ten Trials,the longest game data seems to vary.

V. CONCLUSION

Our investigation involved a number of steps:We first needed to make a mental model of the game, that identifed it as a system in which transitions were made from one square to certain other squares.We were then able to write out the

ISSN(Online) : 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 4, April 2016

rules for one move, and then for the whole game.We were able to ask the computer to report the game length.Then we were able to make the computer play many games, and to compute some statistics about game length

REFERENCES

1. Satinder-singh, honglak-lee, richard-l-lewis ”Deep Learning for Real-Time Atari Game Play Using Offline Monte-Carlo Tree Search Planning:” advances in Neural Information Processing Systems 2014.
2. Cameron Browne,Edward Powley,Daniel Whitehouse,Simon Lucas,Peter I. Cowling,Stephen Tavener, Diego Perez, Spyridon Samothrakis and Simon Colton,,”A Survey of Monte Carlo Tree Search Methods:” IEEE TRANSACTIONS ON COMPUTATIONAL INTELLIGENCE AND AI IN GAMES, VOL. 4, NO. 1, MARCH 2012
3. Baba Satomi, Yongjoon Joe, Atsushi Iwasaki, and Makoto Yokoo,”Real-Time Solving of Quantified CSPs Based on Monte-Carlo Game Tree Search: “Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence