A Novel Approach to Winning a Domino Game

New Mexico Supercomputing Challenge Final Report

April 8, 2020

School:Media Arts Collaborative Charter

Team Member:

Erich Mueller

Teacher:

Creighton Edington

Table of Contents:

Problem Summary ——————————————— page 3

Introduction——————————————————— page 3

Methods———————————————————— page 4

Code—————————————————————— page 5

Results————————————————————— page 18

Conclusion——————————————————— page 18

**Problem Summary:**

For my project I am looking into how to create a novel strategy to win a domino game. I will attempt to create a computer program that would be able to go against actual humans and other computers by choosing one of three strategies. For this I am having the computer play one of 3 strategies, the first one is where the computer plays a random legal move meaning a domino with the same amount of pips on one side as the last one played, the second one is where it will try to block the other player from playing a domino, the third will be where the computer tries to score as many points as possible, there will also be a fourth added in the future where it will play one of these strategies at random. From there I put the computer against actual human players and other computers and run tests. After the tests I will hopefully be able to determine what strategy works best for winning a domino game, I am planning on continuing this project.

**Introduction:**

In the “block” domino game you primarily have 2 to 4 people each with 7 dominoes, they try to match the “pips”, which are the studs on the sides of the dominoes, to one in a train. If the player can not play a domino, the turn goes to the person on their left and if no one can play the game ends with a “block” the players would then add up the number of pips total on all of their dominoes and the person with the lowest total would win. The model I am creating while used for the best way to win a domino game, could also be repurposed for something else that used game theory.

**Methods:**

My project focuses on determining the most successful way to play a domino game. This program is based on game theory, which addresses problems that are frequency dependent ( i.e. where the success of a strategy is determined by what strategy others employ), and is thusly able to be extrapolated to other game theoretical scenarios.

This program utilizes NetLogo for the actual code generation. The code was entirely self written, as opposed to copying existing code and modifying it. This code was designed to model only dyadic interactions/gameplay (i.e., those involving only two players). To setup game play, code was written to deal each player 7 random dominoes. One of the players represented the person whose decision was being modeled, while the other player represented the computer. The three strategies that were coded for this project included allowing players to play any random legal move, modeling both players attempting to score the most points, and allowing both players to try to block the other player from scoring. These strategies were then allowed to compete against each of the others.

**Code:**

For the start i made the domino pieces and import them and make sure they were functional, then I made sure the computer could play up down and left and right and count the pips in all directions, here is my code:

breed [ dominoes domino ] ;; call turtles dominoes

dominoes-own [ pipa pipb ] ;;makes pipa and pipb variable

globals

[

 counter

 pipa-counter

 pipb-counter ;;makes counter for pipa and pipb

 north-pip

 east-pip

 south-pip

 west-pip

 west-train

 east-train

 north-train

 south-train

 double-played?

 xcor-for-double

]

;

to setup

 clear-all ;;clears everything

 set counter 0 ;;creates counter

 set pipa-counter 0 ;;creates counter

 set pipb-counter 0 ;;creates counter

 set north-pip 7 ;;block dominoes from being played

 set east-pip 7 ;;block dominoes from being played

 set south-pip 7 ;;block dominoes from being played

 set west-pip 7 ;;block dominoes from being played

 set west-train 0 ;;keeps track of x coordinate of west train

 set east-train 0 ;;keeps track of x coordinate of west train

 set north-train 0 ;;keeps track of y coordinate of west train

 set south-train 0 ;;keeps track of y coordinate of west train

 set double-played? false

 set xcor-for-double 14 ;;keeps track of x value of double

 ;make background green

 ask patches

 [

 set pcolor green ;;makes background green

 ]

 ;create dominoes

 create-dominoes 28 ;;makes 28 dominoes

 [

 let bone-pile [] ;;create a list called bone pile

 set bone-pile ( shapes ) ;; makes a list of shapes

 set bone-pile but-first bone-pile ;; remove default shape

 set color white ;;make dominoes white

 while [ counter <= 27] ;; loop through all dominoes starting with 0 and ending with 27

 [

 ask domino counter ;; iteration asker

 [

 set pipa pipa-counter ;;sets counter

 set pipb pipb-counter ;;sets counter

 let temp-shape first bone-pile ;; creates temp-shape as variable (first variable named bone pile)

 ask domino who [ set shape temp-shape ] ;; change later

 set bone-pile but-first bone-pile ;;removes default shape out of shape library

 setxy (who) -1 ;; sets the who variable to -1

 set heading 0 ;;set the heading variable to 0

 set pipb-counter pipb + 1 ;;add to pipb counter

 if(pipb-counter > 6) ;;if the pipb counter is less than 6 then do next command

 [

 set pipa-counter pipa-counter + 1 ;;add to pipa counter

 set pipb-counter pipa-counter ;;makes pipa and pipb counter same

 ]

 set counter counter + 1 ;;add to counter

 ]

 ]

 ]

 ;; deal players hands

 ask patch -1 8 ;;asks the patch -1 8 to do next command

 [

 set plabel "player 1" ;;makes label player 1

 ]

 ask patch -1 7 ;;asks the patch -1 7 to do next command

 [

 set plabel "player 2" ;;makes label player 2

 ]

 ;; deal 7 dominoes

 let deal-counter 0 ;;set deal counter to 0

 while [ deal-counter <= 6 ] ;;if the deal counter is greater than or equal too 6 do next command

 [

 ask one-of dominoes with [ ycor = -1 ] ;; domino with ycor equal to -1 does next command

 [

 setxy deal-counter 8 ;;deal domino to coordinate 8

 ]

 ask one-of dominoes with [ ycor = -1 ];; domino with ycor equal to -1 does next command

 [

 setxy deal-counter 7 ;;deal domino to coordinate 8

 ]

 set deal-counter deal-counter + 1 ;;deal one more domino

 ]

 play-first-domino

end

to go

 second-player

end

to play-first-domino

 ask one-of turtles with [ pycor = 8 ]

 [

 setxy 12 4

 ifelse (pipa < pipb)

 [

 set west-pip pipa

 set east-pip pipb

 set heading 270

 ]

 [

 set north-pip pipa

 set south-pip pipb

 set heading 0

 ]

 ;;if first double played

 if (pipa = pipb and double-played? = false)

 [

 set heading 0

 set double-played? true

 set north-pip pipa

 set south-pip pipb

 set west-pip pipa

 set east-pip pipb

 set xcor-for-double xcor

 ]

 ]

end

to second-player

 let temp-counter 0

 let played? false

 let dominoes-in-hand-2 (count dominoes with [pycor = 7])

 while [ temp-counter < dominoes-in-hand-2 and played? = false ]

 [

 ask dominoes with [pycor = 7 and pxcor = temp-counter ]

 [

 ;;check pipa to play on west

 if (pipa = west-pip and played? = false)

 [

 setxy (11 - west-train) 4

 set heading 90

 set played? true

 set west-pip pipb

 set west-train west-train + 1

 ask dominoes with [pycor = 7 and pxcor = (dominoes-in-hand-2 - 1) ] ;; moving last domino in hand to replace domino that was just played

 [

 setxy temp-counter 7

 ]

 ;;if first double played

 if (pipa = pipb and double-played? = false)

 [

 set heading 0

 set double-played? true

 set north-pip pipa

 set south-pip pipb

 set xcor-for-double xcor

 ]

 ]

 ;;check pipb to play west

 if (pipb = west-pip and played? = false )

 [

 setxy (11 - west-train) 4

 set heading 270

 set played? true

 set west-pip pipa

 set west-train west-train + 1

 ask dominoes with [pycor = 7 and pxcor = (dominoes-in-hand-2 - 1) ] ;; mmoving last domino in hand to replace domino that was just played

 [

 setxy temp-counter 7

 ]

 ;;if first double played

 if (pipa = pipb and double-played? = false)

 [

 set heading 0

 set double-played? true

 set north-pip pipa

 set south-pip pipb

 set xcor-for-double xcor

 ]

 ]

 ;check pipa to play on east

 if (pipa = east-pip and played? = false)

 [

 setxy (13 + east-train) 4

 set heading 270

 set played? true

 set east-pip pipb

 set east-train east-train + 1

 ask dominoes with [pycor = 7 and pxcor = (dominoes-in-hand-2 - 1) ] ;; moving last domino in hand to replace domino that was just played

 [

 setxy temp-counter 7

 ]

 ;;if first double played

 if (pipa = pipb and double-played? = false)

 [

 set heading 0

 set double-played? true

 set north-pip pipa

 set south-pip pipb

 set xcor-for-double xcor

 ]

 ]

 ;;check pipb to play east

 if (pipb = east-pip and played? = false )

 [

 setxy (13 + east-train) 4

 set heading 90

 set played? true

 set east-pip pipa

 set east-train east-train + 1

 ask dominoes with [pycor = 7 and pxcor = (dominoes-in-hand-2 - 1) ] ;; mmoving last domino in hand to replace domino that was just played

 [

 setxy temp-counter 7

 ]

 ;;if first double played

 if (pipa = pipb and double-played? = false)

 [

 set heading 0

 set double-played? true

 set north-pip pipa

 set south-pip pipb

 set xcor-for-double xcor

 ]

 ]

 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

 ;;check pipa to play on north

 if (pipa = north-pip and played? = false)

 [

 setxy xcor-for-double ( 5 + north-train)

 set heading 180

 set played? true

 set north-pip pipb

 set north-train north-train + 1

 ask dominoes with [pycor = 7 and pxcor = (dominoes-in-hand-2 - 1) ] ;; mmoving last domino in hand to replace domino that was just played

 [

 setxy temp-counter 7

 ]

 ;;if first double played

 if (pipa = pipb and double-played? = false)

 [

 set heading 0

 set double-played? true

 set north-pip pipa

 set south-pip pipb

 set xcor-for-double xcor

 ]

 ]

 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

 ]

 set temp-counter temp-counter + 1

 ]

 if ( played? = false )

 [

 ask one-of dominoes with [pycor = -1 ]

 [

 setxy temp-counter 7

 ]

 ]

end

**Results:**

Currently I have one strategy fully working and I am working on adding the other 2 as well as a score counter and a way to choose the strategy, some of these things will be done next year. The pips are able to be counted in all 4 necessary directions, and play a legal move.

**Conclusion:**

In conclusion programming the strategies is possible even if I only got one fully complete and I am planning on continuing this project next year and running the necessary tests to determine the best strategy for winning a domino game and hopefully this program will be applicable and actually applied to real life scenarios