Comparing Strategies to Play a 2-Sided Dominoes Game

André R. da Cruz*†, Frederico G. Guimarães‡ and Ricardo H. C. Takahashi§
*Instituto de Ciências Exatas e Tecnológicas, Universidade Federal de Viçosa, Rio Paranaíba, MG, Brazil
†Programa de Pós-Graduação em Engenharia Elétrica, Universidade Federal de Minas Gerais,
Av. Antônio Carlos 6627, 31270-901, Belo Horizonte, MG, Brazil
‡Departamento de Engenharia Elétrica, Universidade Federal de Minas Gerais, Belo Horizonte, MG, Brazil
§Departamento de Matemática, Universidade Federal de Minas Gerais, Belo Horizonte, MG, Brazil

***Abstract*—This work presents four agents with different strategies to play a version of the 2-sided dominoes game, usually played in Minas Gerais state, Brazil. This incomplete information game must be played with two players and the goal is to discard all tiles first according to the rules. Each pair of agents was tested in a computational experiment, for 1,000,000 matches, in order to evaluate the individual effectiveness. In the first strategy, the agent uses random rules to select an adequate tile; the second agent observes the tiles already on the table and on its hand and selects one using a simple probability information computed in an amateur way; the third strategy also observes the tiles on the table and on the hand, and computes a probability information using the two end tiles of the table and the candidates opposite values in order to decide which one must be thrown; in the last strategy, the agent uses the third strategy and the Boltzmann exploration with a roulette wheel to select the tile. The results showed that the last strategy is the best and that even the random strategy is capable to win a significant number of matches.**

***Keywords*-artificial intelligence; agents; incomplete information game; dominoes.**

I. Introduction and Related Work

Domino is a very popular game which consists of a set of rectangular pieces divided in two halves, called tiles or stones. Each half of a tile is marked with dots indicating a numeric value on the front face. If the tile back face is presented, where there is no mark, it is said that the tile is hidden. The dominoes game has different rules, depending on the geographical localization. There are versions which can be played with 2, 3 or 4 players [1]. This work considers a simple version, with 2 players, which is very common in Minas Gerais state, Brazil. Just to illustrate the importance of the game, some studies indicate that dominoes can help to improve human learning and cognition [2], [3], [4], [5], [6].

The 2 player version of dominoes game provides an environment in which it is possible to investigate the decision making abilities of individual agents. The players are in a game with incomplete information, in other words, there are hidden tiles on the opponent hand and on the table. In addition, they do not know previously which movement the opponent will execute. However, some strategies can be built at each turn in order to determine which moves the opponent can or cannot do. Probabilities about the printed number in hidden tiles can be computed based on what a player can observe. Consequently, at each turn, it is possible to determine a promising tile to play which will embarrass the opponent strategy and therefore improving the own chance of victory.

Some related work which developed agents to play dominoes with their particular rules can be found in [7], [8], [9], and [10]. These papers present different strategies to play 2-sided or 4-sided dominoes. In a 2-sided dominoes version, for example, two players dispute who throws the tiles first or obtains a specific number of points according to a particular rule. In a 4-sided dominoes version, a team of two players disputes against another double team according to a particular set of rules.

This work presents four agents, each one with a specific strategy to play 2-sided dominoes game according to a set of rules that are commonly played in Minas Gerais. Section II explains how to play 2-sided dominoes game using the rules adopted by *mineiros*[1]. Section III describes how the environment to play 2-sided dominoes was implemented and the four strategies adopted by each agent. Section IV presents the methodology to evaluate the strategies in a computational experiment and shows the corresponding results. And finally, Section V presents some conclusions about this work and points out possible future works.

II. How to Play 2-Sided Dominoes

This section explains the rules of the 2-sided dominoes adopted in this work. They are simple and easy to understand.

There is a set with 28 tiles in this game. As mentioned before, the front face of each tile is split in two halves, and each half has a number of dots between 0 and 6 painted on it. For each combination of dots, there is only one tile in the set. For instance, on the tile 2–5 or 5–2 there are 2 dots on one half and 5 on the other. The tiles which have the same number of dots on both halves are known as doubles, for example 2–2 and 5–5. Figure 1 shows the set with all tiles of the game.

In order to play, the two players are generally situated opposite to each other on a table. Before beginning a game, the entire set of tiles is placed on the table with the front face down (hidden). After this action, the tiles are randomly mixed by hand so that it is very difficult (impossibility is desirable) to guess the number of dots in any tile. Following, the players pick up randomly seven

[1]People who are resident or native from Minas Gerais

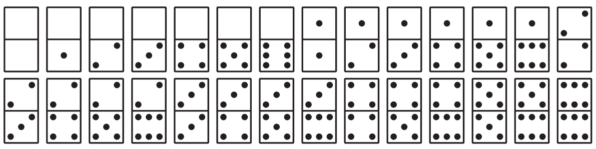

Figure 1. The 28 tiles of the dominoes game.

tiles from the stock to their hands and stand them up in front of, without allowing the opponent seeing the contents. The other fourteen tiles stay hidden on the stock to be caught (bought or eaten, in a popular language) later, when necessary. This initial operation can be done in $\binom{28}{7}\binom{21}{7} = 1.376801712 \cdot 10^{11}$ different ways. This illustrates only the initial difficulty to store all states in order to make a strategy to play dominoes based on sequential memory.

The player which will move first is the one who has the greater double tile (6–6 $>$ 5–5 $>$ 4–4 $>$ 3–3 $>$ 2–2 $>$ 1–1 $>$ 0–0). If nobody has one of these specific tiles, the player who has the tile with different halves with bigger sum must begin. For instance, if a player has 3–6 as the bigger tile and the other 2–5, then the first one begins because $9 > 7$. If a draw persists, a raffle will be done. This example shows the unique allowed communication about the tiles. In a real life, it is asked whether someone has the tile 6–6, else 5–5, and so on[2]. From this point, communication about the tiles is forbidden.

Suppose player A was the first to move and tile 6–6 was placed on the table. Then player A passes the turn to player B. Player B must move a tile with half equal to either end of the game on the table, in this case, a 6–something. If B has at least one, the same must select the adequate tile and place it on the table near the 6–6, and pass the turn. If player B does not have any tile with a 6 on it, he/she must eat a tile from the stock until finding one with a 6 on one half to move and pass the turn to player A. It is not allowed to eat two or more tiles with a 6. In a situation in which the stock is empty the player automatically passes the turn without moving (or placing) a tile. These operations are done until the end of the game is reached.

As moves are made, the tiles placed on the table must form a linear arrangement such that at most two other tiles end up being adjacent to any given tile. Doubles are usually placed vertically and the others horizontally, always with the same values positioned close to the adequate tile. Figure 2 presents an example of game state according to the tiles placed on the table after six moves. The player of the next turn can only move a tile which has on one half the number of the exposed or open ends. In this case, only tiles with 2 can be placed on the left and with 1 on the right. As mentioned before, if the player does not have the specific tile, then it is necessary to take a tile until a suitable one is found. If the stock is empty

[2]As it will be presented later, in the computational environment developed here, players are not asked about the initial tile. This task will be done by an oracle which will say which player must begin.

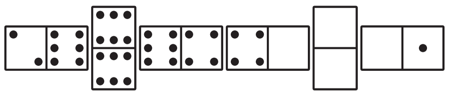

Figure 2. Tiles placed on the table after six moves.

and the player does not have the correct tile, then the turn is passed to the other player.

The objective of the game is to fall all tiles first, in other words, the first player who places correctly all tiles of the hand finishes and wins the match. Another possibility to finish the match is when the game is closed. This means that no player can make a move, despite the two players still have tiles remaining that were not moved. For example, when all 3 and 4 are already played on the table and the open ends have 3 and 4, and players have tiles on their hands, then the game is closed. In this situation, the player with fewer tiles on the hand wins and there is a draw when both players have the same number of tiles.

Most of the time there will be some candidate tiles which can be played at any given time during a game. This moment is when the inferential and decision making capabilities of a player become important. In order to select which of many appropriate candidates tiles to play, an analysis can be made of which movement may be interesting to the current player and detrimental to the opposing player. The way agents select the tiles to move will distinct each one. The objective of this work is to develop and analyze four different strategies according to the number of victories during a given number of matches.

The next section describes the developed environment to play the 2-sided dominoes game and explains how the four strategies for each agent were developed.

III. Environment and Agents

An agent is a computer system that is situated in some environment, and that is capable of autonomous action in this environment in order to meet its delegated design objectives [11]. It can be viewed as perceiving its environment through sensors and acting upon that environment through actuators. An agent can observe and percept some inputs at a given instant and choose and execute a specific set of actions according to state of the environment affecting it [12].

The environment of the 2-sided dominoes game is accessible, but with incomplete information, non-deterministic, dynamic and discrete. The task environment of the agent design describes the environment, the agent type, the performance measure, the actuators and the sensors [12]. The environment of the game is the one determined by the tiles and the rules which must be obeyed. The player which plays in an autonomous way is the agent type with its strategy. The performance measure is described by the number of moved tiles or the success of the tile movement. The three actuators are the capacity of movement of an adequate tile or buy a tile when the agent does not have an adequate tile in its turn, and pass the turn after doing one of the last actions. The sensors of the agent have the capacity to observe the tiles of game on the stock and on its hand.

The project of this work which developed the environment to play 2-sided dominoes with rules described in Section II and the four agents were developed in C++. The algorithm which simulates a match between two agent players is highlighted in Figure 3. It is self-explanatory and will be detailed on the next paragraph.

First, the 28 tiles are generated and stored in the stock list $Stock$ in a randomized order with the function GenerateTilesInStock. An empty double-ended queue (deque) $Table$ is created to store the tiles on the table with function EmptyTileList. It is removed from the stock $Stock$ and inserted in the tiles list 7 tiles to agent players $A1$ and $A2$ with the agent AddTiles method. The next step determines the player that begins the match with the boolean function which returns true if the first agent of the parameter must start the match. In other words, it is analyzed the highest tile to be moved, according to the rules. The opponent tile is not revealed. It is like an oracle which sees the game and says who must begin the match. In this way, there is no communication between players.

The agent method MoveInitialTile removes the adequate tile from the list to initiate the match and the function MoveFirstTile inserts the tile on the table. After the first tile is thrown, the match loop is started. Each loop iteration represents a turn. First, the reference of the next agent player which must move a tile is passed to A by the function NextToPlayReference, it means that it can reference the agent $A1$ or $A2$, depending on the turn. Then, a flag marking if the agent the agent bought a tile from the stock is set as false. Following, the tile movement loop begins at which the agent can observe the game state with incomplete information and try to throw an adequate tile.

Initially, as it did not buy any tile in the current turn, the agent searches for an appropriate tile on the hand and plays it if there is at least one and passes the turn to the other player, leaving the current loop. The agent method MoveOneTile selects and returns an adequate tile, removing it from the agent tiles list. The function AddOneTile inserts the tile on the table on the left side if IsOnLeft returns true, or on the right side otherwise. If the player does not have a suitable tile, it is necessary to get a tile from the stock if exists. If a tile is bought, this one is analyzed and, if adequate, played with MoveBoughtTile and AddOneTile, and the turn will be passed. Otherwise, the current player must buy another tile until it is found one which can be played on the table. If the stock is empty, then the turn is passed without moving any tile. Notice that a boolean flag is used to control if the current agent moved or did not move a tile on the turn.

Just before passing the turn, in the end of the main loop, it is verified if the match is over. In other words, it is checked if all tiles of an agent has ended up or if the match is closed. In the end, an answer which represents the final result is returned.

The characteristics which differ an agent to another one is the way it observes the game state, i.e., the tiles played on the table and the set of tiles on the hand, and how it chooses a suitable tile to be moved. Particularly, in the algorithm presented in Figure 3, the lines 15, 18, 19, 23 and 24 represents how agents execute their strategies.

The four developed agents, with their respective strategies, are detailed in the next subsections. Each one has a name that honors the brave players which drink cachaça and eat crackling in a game session which spends all night.

A. Juca: The Random Player

Juca is a very simple agent. When it is its turn, the strategy is to select the first adequate tile analyzing the set of stones on the hand in a randomized form. Juca only observes what values of the left and right side of the table are. In other words, values which indicate the next tile to be moved.

If it is possible to place the stone on the left or on the right, or if it is possible to spin the tile, then a raffle is performed. This agent simulates a novice player which only knows the rules of the game.

B. Pinduca: Only observes its tile

Pinduca is an agent which is capable to compute a simple probability value and select a promising tile. When it is the turn of Pinduca, it counts the number of values already on the table and on the hand.

Pinduca first selects a subset C of all suitable tiles in a possible configuration (on the left or right of the game on the table, spin or not spin). For each playable configuration, the tile opposite value i is used, through Equation 1, to compute the probability of the opponent not having this other side value. In this equation, n is the number of tile configurations which can be moved in the subset C, k represents each one of these configurations, and $v(i)$ is the number of contrary side value of the configuration i (candidate to be one of the two end values on the table) which Pinduca can see on the table and on its hand. The term is divided by six because there are six values in the game.

$$p(i) = \frac{v(i)/6}{\sum\limits_{k \in C}^{n} v(k)/6} = \frac{v(i)}{\sum\limits_{k \in C}^{n} v(k)} \quad (1)$$

Pinduca sorts the configurations by the probability values, selects and moves the first one with the highest $p(i)$. In this way, Pinduca thinks to move the tile which the opponent may not have at observing the contrary value.

C. Tião: Observes its tile and the two ends

Tião is an agent with a strategy which looks like the way Pinduca plays. However, Tião uses the two possible future end values on the table to select the tile. In other words, to decide the tile to be moved not only the opposite value of the candidate tile in a specific configuration (on the left or right of the game on the table, spin or not spin) is analyzed, but also the other side of the game end on the table is used.

```
Function DominoesMatch($A1, A2$)
 1) $Stock \leftarrow$ GenerateTilesInStock()
 2) $Table \leftarrow$ EmptyTileList()
 3) $A1$.AddTiles(RemoveSevenTiles($Stock$))
 4) $A2$.AddTiles(RemoveSevenTiles($Stock$))
 5) If A1MustStart($A1, A2$)
 6)    $tile \leftarrow A1$.MoveInitialTile()
 7)    MoveFirstTile($Table, tile$)
 8) Else
 9)    $tile \leftarrow A2$.MoveInitialTile()
10)    MoveFirstTile($Table, tile$)
11) Do
12)    $A \leftarrow$ NextToPlayReference($A1, A2$)
13)    $bought \leftarrow false$
14)    Do
15)       $A$.Observe($Table$)
16)       If Not $bought$
17)          If $A$.HasAnAdequateTile()
18)             $tile \leftarrow A$.MoveOneTile()
19)             AddOneTile($Table, tile, A$.IsOnLeft())
20)             $moved[A] \leftarrow true$
21)             Break
22)       Else If $A$.IsLastBoughtTileUsefull()
23)             $tile \leftarrow A$.MoveBoughtTile()
24)             AddOneTile($Table, tile, A$.IsOnLeft())
25)             $moved[A] \leftarrow true$
26)             Break
27)       If HasTiles($Stock$)
28)          $tile \leftarrow$ RemoveOneTile($Stock$)
29)          $A$.AddBoughtTile($tile$)
30)          $bought \leftarrow true$
31)       Else
32)          $moved[A] \leftarrow false$
33)          Break
34)    While $true$
35)    If $A1$.NumOfTiles() = 0 Or $A2$.NumOfTiles() =
       0
36)       Break
37)    If Not $moved[A1]$ And Not $moved[A2]$
38)       Break
39) While $true$
40) If $A1$.NumOfTiles() = 0
41)    Return $A1$ wins
42) Else If $A2$.NumOfTiles() = 0
43)    Return $A2$ wins
44) Else If $A1$.NumOfTiles() > $A2$.NumOfTiles()
45)    Return $A1$ wins
46) Else
47)    Return draw
```

Figure 3. Algorithm which simulates a match with two players.

For a candidate in a set C with a possible configuration i, the computed probability used by Tião, presented in Equation 2, uses the opposite side value $v(i)$, candidate to the new game end on the table, and the other game end on the table which remains on the next turn $t(i)$. If $v(i)$ will be placed on the left, $t(i)$ is the value which is on the right of the game table, and vice-versa. Notice that Tião analyses the two possibilities in a more general mode.

$$p(i) = \frac{v(i)t(i)/6}{\sum_{k \in C}^{n} v(k)t(k)/6} = \frac{v(i)t(i)}{\sum_{k \in C}^{n} v(k)t(k)} \quad (2)$$

In order to decide the tile in a specific configuration, Tião sorts the candidates by the probability values, selects and moves the one with the highest $p(i)$. In this way, Tião moves the tile which will embarrass the opponent player in a general way.

D. Nabucodonossor: Like Tião but with a roulette

Nabucodonossor also uses the two possible future end values on the table to select a candidate tile. The strategy is similar to the Tião, however it is used the Boltzmann exploration [13] in order to determine the probability of selecting a tile in a suitable configuration. The main idea is to raffle a candidate according to the chance of the opponent player not having the next two end side values on the table.

Given a set C of tile candidates with a possible configuration i, the probability of choosing this option is presented in Equation 3. In this equation $v(i)$, $t(i)$, k and n are the same as in Equation 2, and u represents the temperature which is a function of the number of tiles on the table, m, at the moment. The temperature function is computed by $u = 0.999^m$.

$$p(i) = \frac{e^{v(i)t(i)/u}}{\sum_{k \in C}^{n} e^{v(k)t(k)/u}} \quad (3)$$

Nabucodonossor sorts all configurations by the probability values $p(i)$. After, a roulette-wheel selection [14] is performed to determine the tile to be moved. In this way, all configurations have a chance to be chosen, but this chance is proportional to the probability value presented in Equation 3.

IV. Experiment and Results

The four agents were tested in a simple computational experiment in order to analyze their efficacy. It was simulated 1,000,000 matches between each pair of agents in random block experiment [15]. For each agent, it was computed the number of victories, loses and draws. The experiment results are presented in the next paragraphs.

The final result about the 1,000,000 matches between each pair of agents are, according to the victory proportion, represented in pizza charts in Figures 4, 5, 6, 7, 8 and 9. These graphics allow some observations:

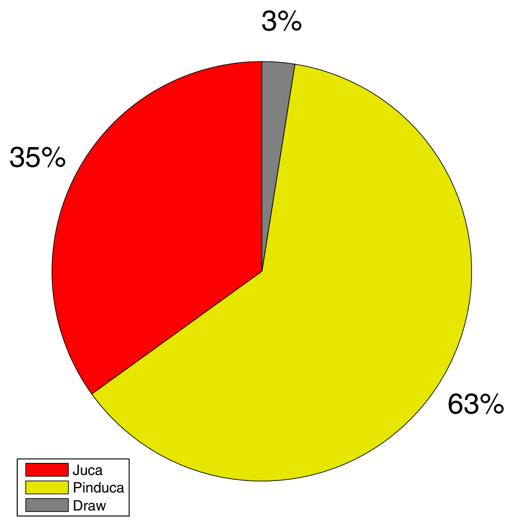

Figure 4. Victory proportion between Juca and Pinduca in 10^6 matches.

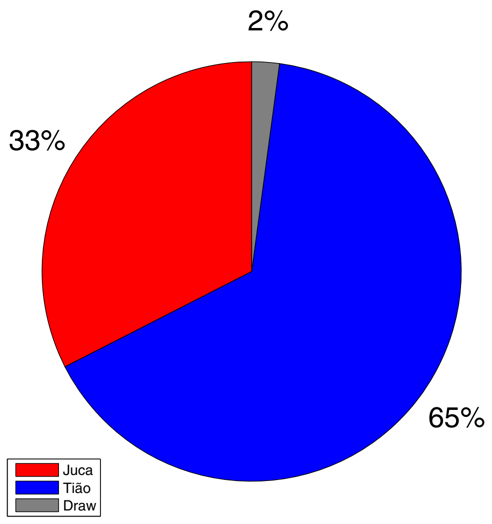

Figure 5. Victory proportion between Juca and Tião in 10^6 matches.

- Juca, the agent with random strategy, was the worst player. Juca has the smallest win proportion in all experiments. However, it obtained victory rates between 0.32 and 0.35, which means that random strategy can obtain a reasonable number of wins in this incomplete information game against other more sophisticated strategies.
- Pinduca, the greedy agent which only analyses its tile end, was the winner against Juca, but it did not obtained the same success against Tião and Nabucodonossor. This means that it is better to observe and analyse the state in a global way in order to decide which the promising tile to be moved is.
- Tião, the agent which has the strategy of analyzing the next two ends of the game on the table, won the agents Juca and Pinduca, in a rate point of view, but this situation did not repeat with Nabucodonossor. The fact of choosing the tile with highest probability of embarrassing the opponent on the next two ends showed to be efficacy against novice strategies.
- Nabucodonossor was the best agent. It won, in percentage, all other agent strategies. This was possible, in addition to the general analyses of tiles on the table and on its hand to embarrass the opponent strategy, because of the random selection of the tile configurations with different probabilities due to their computed importance at observing the end values.

The bar graph presented in Figure 10 shows the win sum for each agent player. This graphic emphasizes the same point of view in relation to the strategy efficacy of each agent. Therefore, the best agents are, in order, Nabucodonossor, Tião, Pinduca, and Juca, with 995427, $1.542e+6$, $1.601e+6$ and $1.620e{+}6$ wins, respectively.

V. Conclusions and Future Work

This work presented four agents with different strategies to play 2-sided dominoes with rules played in Minas

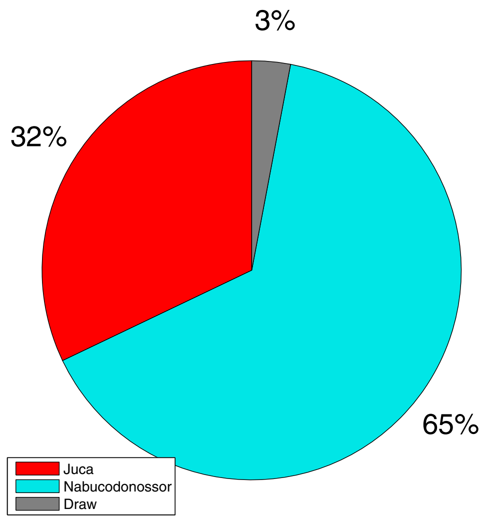

Figure 6. Victory proportion between Juca and Nabucodonossor in 10^6 matches.

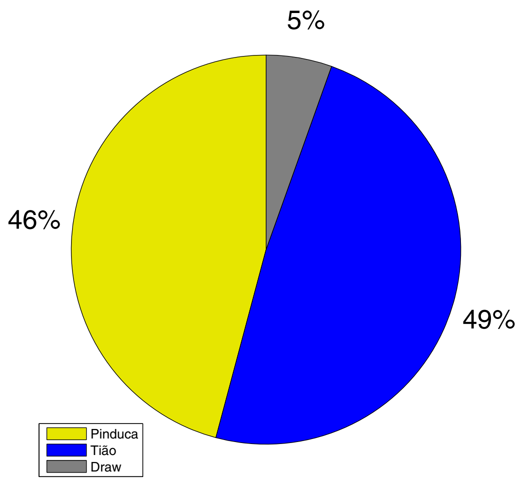

Figure 7. Victory proportion between Pinduca and Tião in 10^6 matches.

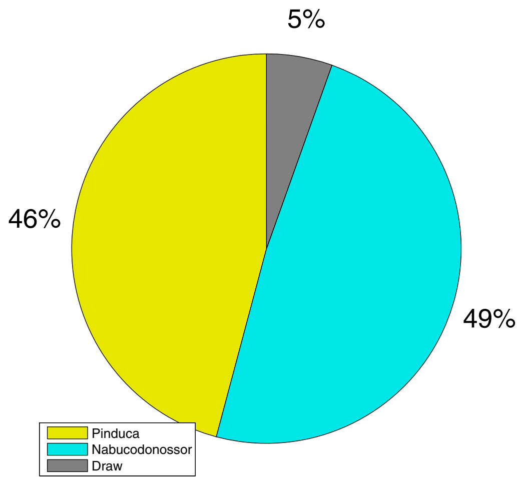

Figure 8. Victory proportion between Pinduca and Nabucodonossor in 10^6 matches.

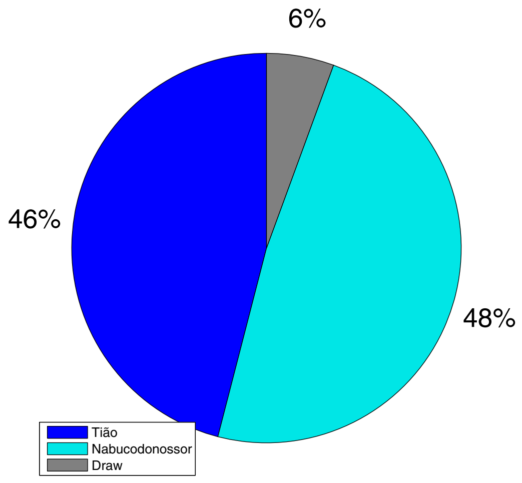

Figure 9. Victory proportion between Tião and Nabucodonossor in 10^6 matches.

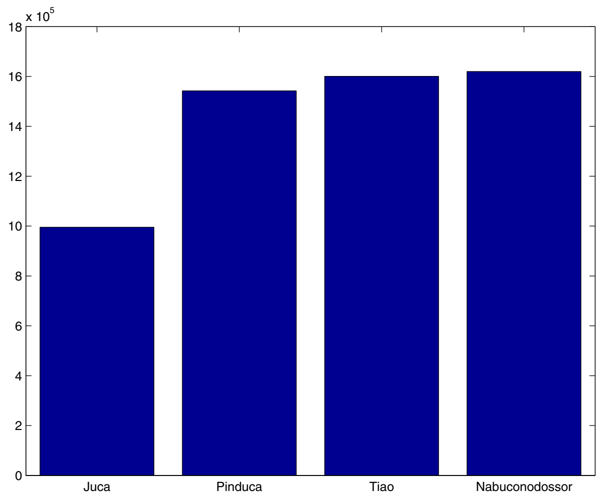

Figure 10. Number of wins in all matches for each agent player.

Gerais, a federation state of Brazil. The game is incomplete information and its goal is to discard all tiles first according to the rules. Each pair of agents was tested in 1,000,000 matches in a random blocks computational experiment in order to evaluate the strategies effectiveness. The first strategy, which used random rules to select an adequate tile, obtained the last place; the second strategy, which observes the tiles already on the table and on its hand and selects one using a simple probability information, obtained the third place; the third strategy, which also observes the tiles on the table and on the hand, computes a probability information using the two ends tile values of the table and the candidates in order to decide which one must be thrown, got the second place; and the fourth strategy, which uses the third strategy added with Boltzmann exploration and a roulette wheel to select the tile, was the best one.

It can also be concluded that all agents, even with random strategy, were able to win a respectable number of matches. This was possible because the game environment is stochastic and it is not possible to determine a strategy which will win always. In this case, it is necessary to capture and analyze all available information, trying to forecast the possible opponent moves and potentiating self-strategy.

It is intended to develop in future works more strategies, for instance, to prioritize the discard of the player tiles according to the values which are more common on the hand. It is also desired to mix the strategies during the match, alternating the way of playing the game.

ACKNOWLEDGMENT

The authors would thank to the Brazilian agencies Capes, CNPq and Fapemig for financial support.

REFERENCES

[1] http://www.dominorules.com, "Domino rules," Access in 01-12-2012.

[2] R. Ward, "Dominoes as fractions: Misconceptions and understandings," *Mathematics Teaching in the Middle School*, vol. 5, no. 3, pp. 162–65, 1999.

[3] J. Santos and J. Alves, "O jogo de dominó como contexto interativo para a construção de conhecimentos por pré-escolares," *Psicologia: Reflexão e Crítica*, vol. 13, no. 3, pp. 383–390, 2000.

[4] N. S. Nasir, "Individual cognitive structuring and the socio-cultural context: Strategy shifts in the game of dominoes," *The Journal of the Learning Sciences*, vol. 14, no. 1, pp. 5–34, 2005.

[5] E. Martins and H. Szymanski, "Brincadeira e práticas educativas familiares: um estudo com famílias de baixa renda," *Interações*, vol. 11, no. 21, pp. 143–164, 2006.

[6] A. Lima, "Fatores que modificam a função congnitiva e motora na doença de parkinson: um estudo sobre a influência do jogo de dominó," Master's thesis, Universidade Federal do Ceará, 2007.

[7] M. Smith, "A learning program which plays partnership dominoes," *Communications of the ACM*, vol. 16, no. 8, pp. 462–467, 1973.

[8] B. Chlebus, "Domino-tiling games," *Journal of Computer and System Sciences*, vol. 32, no. 3, pp. 374–392, 1986.

[9] A. de Silva Garza, "Evaluating individual player strategies in a collaborative incomplete-information agent-based game playing environment," in *Computational Intelligence and Games, 2006 IEEE Symposium on*. IEEE, 2006, pp. 211–216.

[10] N. Antonio, M. Costa, R. Padilla *et al.*, "Optimization of an evaluation function of the 4-sided dominoes game using a genetic algorithm," in *Computational Intelligence and Games (CIG), 2011 IEEE Conference on*. IEEE, 2011, pp. 24–30.

[11] M. Wooldridge, *An Introduction to Multiagent Systems*, 3rd ed. John Wiley and Sons, 2001.

[12] S. Russel and P. Norving, *Artificial Intelligence: A Modern Approch*, 3rd ed. Pearson, 2010.

[13] T. Sandholm and R. Crites, "Multiagent reinforcement learning in the iterated prisoner's dilemma," *Biosystems*, vol. 37, no. 1, pp. 147–166, 1996.

[14] D. Goldberg, "Genetic algorithms in search, optimization, and machine learning," 1989.

[15] D. Montgomery, *Design and analysis of experiments*. Wiley, 2008.