BRNO UNIVERSITY OF TECHNOLOGY FACULTY OF INFORMATION TECHNOLOGY

Excel@FIT 2018
http://excel.fit.vutbr.cz

Artificial Intelligence for a Board Game

Dominik Tureček*

Abstract
Dice Wars is a discrete stochastic board game. This work proposes an artificial intelligence able to play the game. Several strategies for AI players were implemented using rule-based approach and expectiminimax algorithm. A client-server implementation of the game was created to test the proposed AI players. Statistics from games played with these were collected. Data from the experiments are evaluated and discussed. Proposed single-turn expectiminimax algorithm can accurately estimate game state in a next turn and has a 34.9 % win rate in games against 5 opponents. Using the implementation and data collected in this work, more sophisticated AI players involving reinforcement learning could be created.

Keywords: Discrete Stochastic Games — Expectiminimax — Artificial Intelligence

Supplementary Material: Source Code

*xturec06@stud.fit.vutbr.cz, *Faculty of Information Technology, Brno University of Technology*

1. Introduction

Board games are popular target for developing artificial intelligence players [1], [2], [3]. This paper proposes an AI for Dice Wars[1]. As the original implementation's source code is not public, a client-server implementation was developed for this paper.

The games is turn-based, for which reason it is relatively uninteresting from the point of game theory, zero-sum, and stochastic. It is played on a stochastic board.

Because it would be difficult to use strategies by TD-Gammon[3], this work uses rule-based approach. Statistics where gathered during experiments with created AI players. These data will serve as a basis to create machine-learning algorithms.

2. Dice Wars

Dice Wars is a turn-based strategy game played with two to eight players on a board divided into *territories*. Each territory contains a number of dice. These dice determine a player's presence and strength in the territory and are rolled when attacking adjacent territories controlled by opponent players. The goal of the game is to conquer all territories. An example of a five-player game is shown in Figure 1.

[1]http://www.gamedesign.jp/flash/dice/dice.html

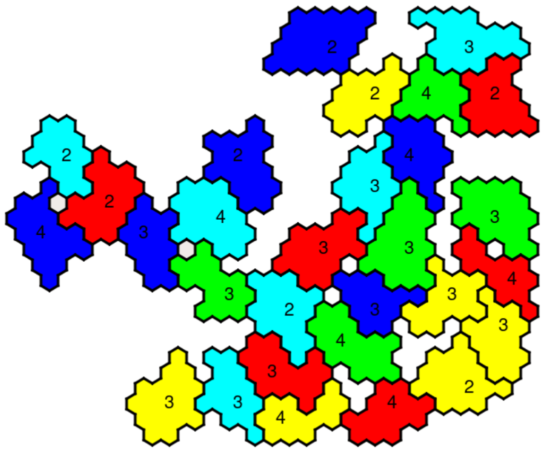

Figure 1. Example of a game board used in Dice Wars. Each colour represents territories controlled by one player. The number in each territory corresponds to its strength. If two territories share at least one edge, they are considered adjacent.

2.1 Setup and Terminology

The game board is randomly generated as follows: At first, 29–31 territories are generated using a seed-like algorithm. Territories are randomly assigned to indi-

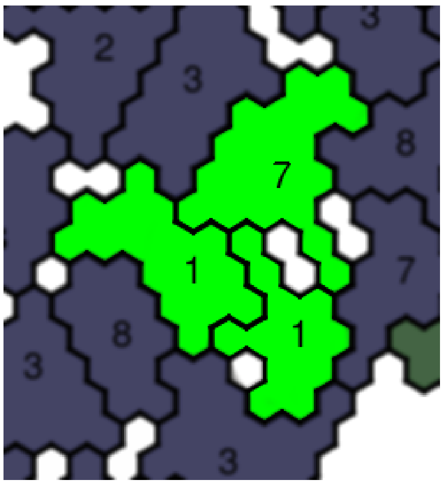

Figure 2. Picture showing a region controlled by a green player. The region has size of three and consists of one territory with strength of seven and two territories with strength of one. The green player has a score of three.

vidual players. Then, for each player, a predefined number of dice is distributed in territories owned by the player. Lastly, the player order is determined randomly.

The number of dice in a territory represents how much power a player holds over it and is referred to as the territory's *strength*. Strength of a territory can neither be lower than one nor higher than eight at any given time.

A *region* is a group of adjacent territories controlled by a single player. An example of a region is shown in Figure 2. *Score* of a player is the number of territories in his largest region.

2.2 Player's Turn

On each turn, a player has the option to attack as many territories held by his opponents as he wants, provided that the following two rules are satisfied:

- To attack an enemy territory, it must be adjacent to a player's territory from which the attack is initiated.
- The territory from which the attack is initiated has strength of two or more.

Whenever a player cannot or doesn't wish to make another attack, he or she declares that his turn has ended.

Then a number of dice equal to the player's score is distributed randomly across all territories under the player's control. Dice that cannot be placed due to the limit of eight dice per territory are kept in reserve up to the limit of sixty-four and can be distributed at later turns.

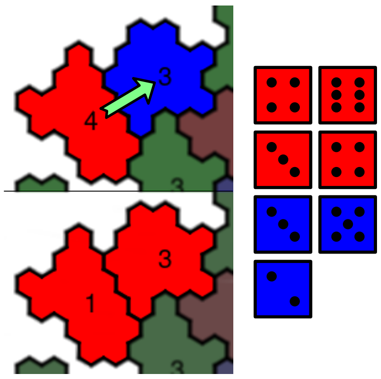

Figure 3. Red player chose to attack the blue terriory. Red rolled four dice with whereas blue rolled three. Red rolled higher, so he took control of blue's territory, placing there three dice and leaving one die in the territory from which the attack started.

2.3 Attack's Resolution

When a player starts an attack, a *battle* occurs. To determine the result of a battle, both players roll number of dice corresponding to the strength of their territory involved in the battle. The player with higher roll is the winner. In case of a tie, the defender wins.

In case the attacker wins the battle, he gains control of the attacked territory. Defender's dice from the territory are removed and the attacker moves there all dice except one from the territory from which he attacked.

If the attacker loses the battle, the number of dice in the territory from which he attacked is reduced to one and nothing happens to the attacked territory.

An example of a battle can be seen in Figure 3.

3. AI Player Agents

In this section, three AI player strategies that were proposed by this work are described, with each of these having higher complexity than the previous one.

3.1 Naïve Random Player

This strategy is the simplest possible. Each turn, the naïve player iterates over all his territories in random order. For each one that has strength higher than one, it checks all adjacent territories and if any of them belongs to another player, it is picked as a target for an attack. After the attack, the player starts over. The player ends its turn if and only if no further attack can be made.

This strategy will serve as a baseline to assess the performance of the other AIs.

3.2 Strength Difference Checking

This player attempts to make reasonable attacks. It judges the strength difference (SD) associated with every possible attack. Then he makes the one with the most favourable SD.

If there is no possible move with strength difference higher or equal to zero, the player ends its turn.

3.3 Single Turn Expectiminimax

This variant uses two-ply expectiminimax algorithm [4] to estimate game state after a single opponent's turn to choose optimal move. As a heuristic measure of success, it uses number of held territories at a start of player's turn. When looking for a possible move, the player searches for such that has the highest chance to increase this number. To achieve this, moves with highest probability of conquering a territory and holding it over a next player's turn are prioritized.

Let $P_{a \to d}$[2] be a probability that a player attacking a from territory a will conquer a defending territory d. Then the estimated probability of successfuly attacking a territory and then holding it through next player's turn can be formally described as follows:

$$P_{ad} = P_{a \to d} \times \prod_{i=1}^{k} (1 - P_{n_i \to d}), \qquad (1)$$

where n_i are neighbours of territory d that are controlled by an opossing player.

However, this estimate is limited because it treats individual territories as independent and omits any relation between them. For example, this doesn't take into account the possibility that a is taken over by an opponent and d is then attacked from a. Furthermore, in games played with more than two players, one of the neighbours of d might be conquered by an opponent with higher strength.

From all possible moves with calculated probability higher than 20 %[3], the highest one is chosen. If no such move is found, the agent then looks if there is a possible move from a territory containing eight dice to prevent a situation where no player is willing to make a move. Otherwise, the agent ends its turn.

4. Experiments

To test the AI agents described in previous section, a client-server implementation in Python was developed. Server handles the game logic and receives moves from the clients which can be controlled either by a human player or an artificial intelligence. Client controlled by a human player uses a GUI implemented using PyQt, a Python binding for Qt framework[4].

The most important metric that could be measured during experiments is win rate. However, the game gives an advantage for the player playing first, so win rate itself may be affected by the rate of which one is the first player, even though the advantage is lower in games played with larger number of players. To show this, a thousand games were played for each number of players with naïve player. Figure 4 shows win rate when starting as a first player.

Therefore, the rate at which a player wins despite the disadvantage of starting second is used as another metric. On top of that, in games played by more than two players, we use overall ranking of a player to get finer assessment of its performance.

All of the experiments described in the following sections were carried out with games that used randomly generated game board and random player order.

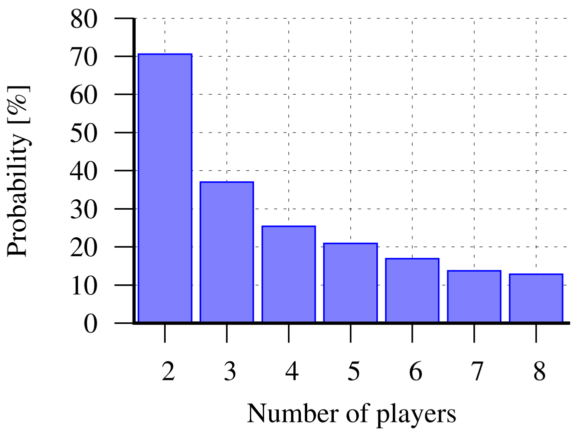

Figure 4. Win rate of players playing first in respect to number of players in a game. These data were gathered using only random naïve players.

4.1 Naïve Player vs Strength Difference

A series of ten thousand games were played between Naïve Player and Strength Difference Checking (SDC).

Table 1 shows data from these games. SDC has won 63.5 % of these games. In addition, it was able to win 40.4 % of games it started from the second position, whereas naïve player only won 12.5 % in such situation.

For each possible combination of the two AIs, a series of thousand multi-player games were played. From the collected data shown in Figure 5 can be seen that even with just the preference of highest strength difference, SDC is able to win more games than the naïve player.

[2]General formal description of this probability is non-trivial. However, as the number of dice rolled is limited, it is not needed for this work. For this reason, the probabilites were precalculated numerically for each possible combination of dice.

[3]This treshold was chosen ad hoc and if changed could probably slightly affect the performance.

[4]https://www.qt.io/

Table 1. Data from the two-player games between naïve player and strength difference checking (SDC). The table shows wins rate and a percentage of games starting as second that were won (Won 2nd).

	Won	Won 2nd
Naïve	36.5%	12.7%
SDC	63.5%	40.4%

Table 2. Data from two-player games between STE and naïve player or SDC. The table shows win rate and number of games won despite starting second (Won 2nd).

	Won	Won 2nd
STE	75.7%	60.9%
Naïve	24.3%	9.9%
STE	48.0%	24.3%
SDC	52.0%	29.1%

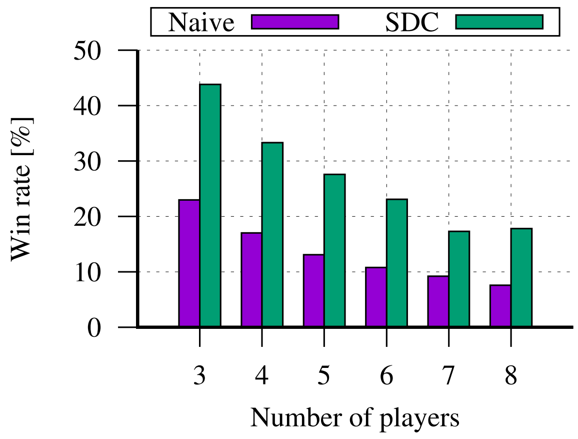

Figure 5. Win rate of naïve player and SDC and STE in games of three to eight players. These data were collected from every possible combination of the two players.

4.2 Single Turn Expectiminimax

A series of two-player games with random initialization were played against both naïve player and SDC. Table 2 shows data from these games. Single turn expectiminimax (STE) performs better against naïve player than SDC does, winning 75.7 % of games and 60.9 % started from second position.

However, in two-player games played against SDC, STE performs slightly worse, winning only 48.0 % of games. STE also wins only 24.3 % of games when starting second, whereas SDC wins 29.1 % of games in this situation.

However, as shown in Figure 6, STE was able to achieve higher win rates than SDC when playing games with three or more players.

For each move of the STE, the estimated probability of successfuly attacking and holding a territory

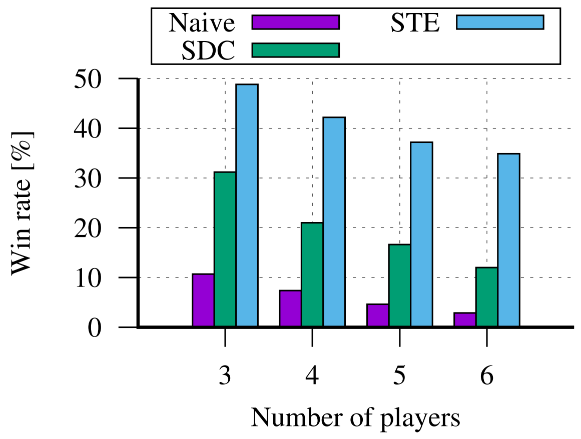

Figure 6. Win rate of naïve player, SDC and STE in games of three to six players. These data were collected from every possible combination of the three players.

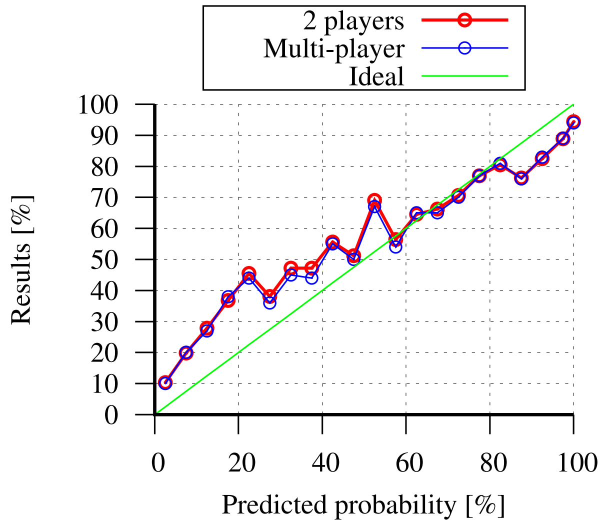

Figure 7. Success rate of attacking and holding a territory through opponents' turns in relation to calculated probabilities. This figure shows data collected from both two-player and multi-player games. The ideal line shows where the prediction would match the results.

(Eq. 1) were collected alongside the actual outcome. Figure 7 shows collected data. The probability estimation is quite accurate and very similar for both two-player and multi-player games.

Up to the probability of around 65 %, the actual results are better than the estimated probability. This is due to the fact, that the estimation is calculated with the assumption that all enemies will attack, which is not the case. On the other hand, for the probabilities higher than 80 %, the actual results are worse. This is because of the limits of used approximation as described in 3.3.

5. Conclusions

In this paper, the game of Dice Wars was presented. Three rule-based strategies that could be adopted by AI players were introduced. To test these strategies, a client-server implementation of the game was created. The experiments showed that even with a simple heuristic such as the probability to conquer and hold a territory to the next turn can increase chances for winning. The single-turn expectiminimax algorithm was able to successfuly predict this probability and has win-rate of 34.9 % in 6-player games.

In the future, the data gathered during these experiments can be used for implementing a learning algorithm for an AI player. In addition, thanks to the created implementation, it would be easy to create new AI clients without the need to provide the game logic.

Acknowledgements

I would like to thank my supervisor Karel Beneš for his help.

References

[1] Jan Černohub. Umělá inteligence pro deskovou hru carcassonne. Bakalářská práce, Vysoké učení technické v Brně, Fakulta informačních technologií, 2011.

[2] David Silver, Aja Huang, and et al. Chris J. Maddison. Mastering the game of go with deep neural networks and tree search. In *Nature*, pages 484–489, 2016.

[3] Gerald Tesauro. Programming backgammon using self-teaching neural nets. In *Artificial Intelligence*, pages 181–199, 2002.

[4] Stuart J. Russel and Peter Norvig. *Artificial Intelligence: A Modern Approach, 3rd ed.* Prentice Hall, 2010.