

# Modeling and Analyzing Board Games through Markov Decision Processes

DANIEL MOCANU, University of Twente, The Netherlands

## ABSTRACT

This research investigates the feasibility of using discrete-time Markov chains (DTMCs) and Markov Decision Processes (MDPs) to model the behavior of players in board games, which will allow for further analysis of the models. The analysis will be performed using the model-checking tool Prism, which supports a range of probabilistic models (including MDPs and DTMCs) but also the analysis of models (including automated metrics computation and strategy generation). This research paper will be focusing on analyzing the board game *Incan Gold* (in which players explore a temple trying to collect as many gems while avoiding hazards) as well as the *Combat Dice Roll* mechanic from the *Hobbit Adventure board game* (where the aim is to roll a high number using three dice in three rounds). The paper will focus on answering interesting questions regarding the 2 games, such as: "What is the probability of a roll resulting in a value  $\geq x$  where  $x$  is a possible roll value?" (*Combat Dice Roll*) or "At which point is the player encouraged to withdraw from the game?" (*Incan Gold*) but also try and answer more general questions about modeling board games as MDPs.

Ultimately, this paper aims to show how to model and analyze board games as MDPs to investigate interesting properties of the games (such as optimal strategies and important metrics) but also to conclude some limitations of the model in regard to more complex games.

Additional Key Words and Phrases: Probabilistic board game, Markov Chain, Markov Decision Process (MDP), Incan Gold, Combat Dice Roll, Prism, Prism-games, The Hobbit Adventure board game

## 1 INTRODUCTION

Board games have been a popular form of entertainment and social interaction among different cultures for centuries. However, in recent years, they have also become the focus of research. Besides being an excellent form of entertainment, board games can also be used as learning environments as they are an excellent way to improve one's cognitive abilities such as logical and mathematical thinking [2]. For example, board games can be used to model important daily life concepts or events, such as taking a risk [7].

From a scientific perspective, modeling and analyzing board games as mathematical frameworks is interesting for multiple reasons. First of all, some board games are very good examples of stochastic games which can be transitioned into stochastic models. Scientists who create model checkers often need good real-life examples to test their tools, and simplistic board games with predictable results are perfect examples that can be modeled to test these tools.

Another good reason to design board games as stochastic models is to ensure that a game is balanced. This is useful when trying to implement a new board game or trying to make sure that an existing aspect of a board game is balanced, which can be important in probabilistic gambling games such as Poker and Blackjack.

Lastly, modeling a board game into a probabilistic mathematical framework can provide useful insight into understanding player behavior. By modeling the game as a stochastic model we can not only predict player behavior in different scenarios but also derive good strategies that can be used in those scenarios. This can be particularly helpful when designing new game mechanics or trying to improve upon existing ones.

One of the possible ways in which a board game can be modeled and analyzed is through an *MDP*. An *MDP* is a mathematical framework that can be used to model decisions of a process in which the outcome of the decision is *random*, for example, the outcome of *rolling a die* or *flipping a coin*. The main advantage of MDPs which make them suitable for this research is that it is possible to evaluate and find an optimal solution, which makes them an excellent framework that can be used to analyze the *optimal strategy* of any board game that can be modeled as an MDP.

This research aims to discover ways in which probabilistic board games can be modeled as MDPs and analyze the resulting stochastic models using the probabilistic model checker tool Prism. While work regarding modeling and analyzing different boards games has already been done, we will specifically focus on the probabilistic game mechanic *Combat Dice Roll* used in the board game *The Hobbit Adventure board game*, as well as the probabilistic board game *Incan Gold*. Analysis concerning these games has yet to be performed in the research space. In the process of modeling these games, we will focus not only on answering game-specific questions such as "What is the expected value of a roll" (*Combat Dice Roll*) or "When is the player encouraged to leave the temple" (*Incan Gold*) but also more general and less researched questions such as:

- (1) How can the application of MDPs to board game analysis aid in identifying optimal strategies, evaluating game balance, or predicting the outcome of games?
- (2) How does the complexity of a game impact the resulting size of the created model?

## 2 RELATED WORK

With regard to modeling a board game into Markov Decision Processes or Markov Chains, work has been done for popular probabilistic games such as Blackjack [14], Monopoly [1], Game of Goose [4] and more. Research for the board game RISK has been done by Barış Tan [13] in which the author has answered 2 research questions related to the game, with the use of Markov Chains, mainly: "What is the probability that an attacked territory will be captured?" as well as "What is the expected loss of soldiers based on the number of troops that the enemy territory has?". Other interesting work has been done for the game of Monopoly [1], where Ash and Bishop have analyzed which are the most profitable provinces based on the assumption that a player stays in jail until they rolled double or 3 turns have passed. In [4] the authors describe the steps taken as well as the tools used in analyzing the Game of Goose for 2, 3,

TS&IT 39, July 7, 2023, Enschede, The Netherlands

© 2023 Association for Computing Machinery.

This is the author's version of the work. It is posted here for your personal use. Not for redistribution. The definitive Version of Record was published in <https://doi.org/10.1145/nnnnnnn.nnnnnnn>.

4 or 5 players. In [14] Michael B. Wakin and Christopher J. Rozell describes some steps that can be taken when modeling the game of BlackJack as a Markov Chain, where the author also implements well-known BlackJack strategies and compares them. Within the space of the University of Twente, research with regard to modeling and analyzing board games as probabilistic models has been done by Vishva Sundarapandian Raani [11] who modeled and analyzed the game of Snakes and Ladders as well as the Game of Goose where the student tried to answer interesting questions such as "Does the game end within 30 moves" (Snake and Ladders) or "What is the probability of the game ending in a draw" (2-player Game of Goose).

The above work also helps us get a better insight into what to expect out of the general research questions mentioned in the introduction. For example, the analysis of the Game of Goose presented in [4] mentions that the paper did not expand beyond 5 players due to the complexity of the resulting models and that, as the model was expanded upon, different tools and approaches were required to analyze the results. The Game of Risk analyzed in [13] presented results with regard to the possible outcomes of battles depending on the number of troops involved and in the analysis of BlackJack in [14], the paper specifies comparing different well-known BlackJack strategies.

Concerning the games that we will analyze in this research, no current study on how to model them as Markov Chains or Markov Decision Processes or any other probabilistic analysis has been found.

### 3 PRELIMINARIES

#### 3.1 Markov Decision Processes

*Markov chains* or *Discrete-time Markov chains* (DTMC for short) are one of the simplest probabilistic models [8]. A DTMC contains 2 key elements, *states* and *transitions*. Every single state has outgoing transitions to other possible states which indicate a change from a state to another. Every single transition also has a probability attached to it and every single state's outgoing transition probabilities sum up to 1. Formally, we can define a Markov Chain as:

*Definition 3.1 (DTMC).* The tuple  $(S, s_0, P)$  where:

- (1)  $S$  is the set of reachable Markov states from some initial state  $s_0 \in S$ .
- (2)  $P$  is the transition probability function,  $P: S \times S \rightarrow [0, 1]$  where  $P(i, j)$  represents the probability of reaching state  $j$  through state  $i$ , where  $i, j \in S$  and  $\sum_{j \in S} P(j|i) = 1$ .

$S$  is the set of reachable Markov states from some initial state  $s_0 \in S$ .  $P$  is the transition probability function,  $P: S \times S \rightarrow [0, 1]$  where  $P(i, j)$  represents the probability of reaching state  $j$  through state  $i$ , where  $i, j \in S$  and  $\sum_{j \in S} P(j|i) = 1$ .

A *Markov Decision Process* (MDP for short) extends Markov Chains with the introduction of *non-determinism*. In a DTMC the next state is chosen according to the probability distribution of the current state, however, in an MDP, a state may have more than 1 possible distribution, indicating that the resulting state is not only dependent on the current state but also on the choice of the distribution (known as *policy*) [8].

Formally, an MDP extends a Markov Chain [12] as:

*Definition 3.2.* The 4-tuple  $(S, s_0, A, P, R)$  where:

- (1) As before,  $S$  is the set of reachable Markov states from some initial state  $s_0 \in S$ .
- (2)  $A$  is the set of actions available to a state, where the set of actions for different states need not be the same.
- (3)  $P$  is the state transition probability where  $P(i, a, j)$  denotes the probability of reaching  $j$  from  $i$  given action  $a \in A$ ,  $i, j \in S$ .
- (4)  $R$  is the transition reward where  $R(i, a, j) \geq 0$  denotes the reward achieved in a transition from  $i$  to  $j$  given action  $a \in A$ ,  $i, j \in S$ .<sup>1</sup>

As noted, one of the unique characteristics of an MDP is a *policy* (also known as *strategy* or *adversary*). A policy determines the action which will be taken by some existing state  $s$ . Formally, a *policy* is a function  $\pi$  mapping a state to a resulting action  $\pi: S \rightarrow A$ . It should also be noted that a policy influences the reward system of a model. In general, a policy could be used to minimize or maximize a reward, however, the reward is determined based on the action taken at every single state, which in turn is determined by the policy.

#### 3.2 Prism and Prism-games

Since modeling an MDP based on a board game can quickly become very large, tools which can aid with the creation of such MDPs are necessary. To model a board game into an MDP, the probabilistic model checker *Prism* [9] version 4.7 as well as *Prism-games* [10] version 3.0 were used. Prism provides a simple state-based language that can be used to construct a variety of probabilistic models including Markov Decision Processes. The main advantage of using Prism is the broad support for constructing Markov Decision Processes, Prism helps keep track of the intended relevance of states and lets the user automatize the creation of states and relations. The tool also allows for probabilistic model checking which means that the user can analyze the behavior and outcome of an MDP. An addition to the tool which allows for a more in-depth analysis of a model is the tool Prism-games.

Prism-games is an extension of Prism that can incorporate competitive or collaborative behavior, modelled as stochastic multi-player games. Unlike Prism, Prism-games also supports *strategy generation* and *strategy analysis* which is crucial for some of our research questions. The tools will be helpful to us in analyzing the mentioned board games to find an answer to our research questions. Based on this information, we can then model and analyze Markov Decision Processes using the Prism modeling language. The analysis of the created models will be done using Prism's *property specifications* [9] support as well as Prism-games' strategy generation [10]. For property specification with regard to discrete-time models such as MDPs and DTMCs, a quantitative property specification that subsumes *Probabilistic Computational Tree Logic (PCTL)* is used. Applying Prism properties, the user can ask non-trivial questions about the model:

- (1)  $P = 1 [ F \text{ "reach G" } ]$  - Is the probability of reaching G (from the initial state) equal to 1?

<sup>1</sup>in the model description, we use both states and transition rewards, the only difference is that state rewards are collected at the state itself rather than on the transition.



Prism will return either True if the property holds or False (as well as a counterexample) if the property does not hold. Properties can also be used to determine the probability of reaching G ( $P=?$  [F "reach G"]) which will return the probability of reaching G from the initial state. We then can build strategies based on defined properties using Prism-games. The tool will generate the best strategy to be used to achieve the probability of reaching G, mainly the property  $Pmax=?$  [F "reach G"].<sup>2</sup>





## 4 COMBAT DICE ROLL

In this chapter, we will be looking at the *Combat Dice Roll* mechanic from the *Hobbit Adventure board game*. We will explain the game rules, the established research questions, the methodology for building the model and the results to the defined research questions.

### 4.1 Game Rules

Information about *The Hobbit Adventure board game* as well as the game rules of the *Combat Dice Roll* mechanic can be found in [5] (Last Access 02.07.2023).

The *Combat Dice Roll* mechanism in the *Hobbit Adventure game* is composed of 3 identical D6s (6-sided dice). Three of the six sides of the dice contain numbers 1 through 3: . The other three sides of the dice contain some pips 1 through 3: . The player first rolls all 3 Combat Dice. The player then separately sums up the value of the dice containing numbers and the dice containing pips. The value of the roll is the absolute difference between the sum of numbers and the sum of pips. If the roll results in 3 sides with only numbers or only pips, the player adds up the resulting number which will represent the value of the roll. Some example Combat rolls could be:

- (1) A roll of  results in a value of 1.
- (2) A roll of  results in a value of 0.
- (3) A roll of  results in a value of 4.
- (4) A roll of  results in a value of 6.

After the first roll, the player can decide to re-roll 1, 2, or 3 of the Combat Dice, the values of the dice which are not re-rolled are kept in the re-roll. After the first roll, the player can re-roll up to 2 times following the rolling procedure mentioned above.

The goal of the game is either to maximize the resulting combat roll value or to obtain a combat value higher or equal to a given value.

### 4.2 Research Questions

With regard to the *Combat Dice roll* mechanic present in the *Hobbit Adventure game* we will be answering the following research questions (with the usage of the *optimal strategy*):

- (1) What is the maximal probability (over up to 2 re-rolls) that the roll will result in a value  $\geq x$ , for  $x \in \{1, 2, \dots, 9\}$ ?
- (2) What is the maximal expected value which will result from the roll (for up to 2 re-rolls) and how does this value change based on the number of re-rolls?

<sup>2</sup>Note that Pmax is a property specification available only for models with non-determinism and cannot be used for Markov Chains. Conversely we cannot use  $P=?$  to ask questions about the probability of Markov Decision Processes since MDPs do not contain a fixed probability distribution for every state prior to fixing a strategy.

- (3) Which dice is the player encouraged to re-roll to achieve the optimal results?

### 4.3 Model

All further explanations are based on the Prism code used to model the mechanic that can be seen in Figure 1.

```

1 mdp
2
3  ## possible re-rolls of the game
4  const int k;
5
6  module Dice1
7    d1 : [-3..3] init 0;
8    ##The first roll is required
9    [firstRoll] r = 0 -> 1/6 : (d1' = -3) + ... + 1/6 : (d1' = 3);
10   ##Subsequent rolls are optional
11   [reRoll] r > 0 & r < k -> 1/6 : (d1' = -3) + ... + 1/6 : (d1' = 3);
12   [reRoll] r > 0 & r < k -> true;
13   ##After k rolls
14   [end] r = k -> (d1'=0);
15 endmodule
16
17 module Dice2 = Dice1 [d1 = d2] endmodule
18
19 module Dice3 = Dice1 [d1 = d3] endmodule
20
21 module Roll
22   r : [0..k+1] init 0;
23   [firstRoll] r = 0 -> (r' = 1);
24   [reRoll] r > 0 & r < k -> (r' = r + 1);
25   [end] r = k -> (r' = k+1);
26 endmodule
27
28 ## |d1+d2+d3|
29 rewards
30   r = k : pow(pow(d1+d2+d3, 2), 0.5);
31 endrewards

```

Fig. 1. Combat Dice roll code

To define the *model* of the game, we constructed 4 *different modules* in the Prism language, 3 of the modules represent the 3 different dice that need to be rolled with a variable indicating the value that was *rolled*. The values of the dice range contain values from the set  $\{-3, -2, -1, 1, 2, 3\}$  where the values 1, 2 or 3 are the values corresponding to a numerical roll from 1 to 3, and the values  $-1, -2$  or  $-3$  are the values corresponding to dots being rolled from 1 to 3. The 4<sup>th</sup> module represents the number of the *roll* (first, second or third roll). Therefore, every unique state in the model is defined by 4 values, the resulting rolls of the 3 different dice, and the number of the roll. Every single module contains several actions which each represent one of the possible rolls (one, two or three). The model also contains *non-deterministic choices* which represent a player's decision. The non-deterministic choice is only available for 2<sup>nd</sup> or 3<sup>rd</sup> roll and the 2 different choices (per dice) are to either *re-roll* the dice or *keep* its current value. To ensure that the rolls are done *simultaneously*, we also use synchronization actions on all the dice rolls and the roll number. This ensures that all the dice are rolled simultaneously, therefore a state in which one of the Dice has the value 0 (other than the initial state) does not exist. Last, but not least, another important aspect of our model is the use of *rewards*. This is useful when calculating the maximal expected value that the player can anticipate after 3 different rounds of rolls. Overall, the model is composed of 650 *unique states* (out of which 1 is the initial state and 1 final state) and a total of 148609 *transitions*.

To test our model and derive the answer to the research questions, we used Prism's property support. To answer the first 2 research

questions which ask about the probability of obtaining a desirable roll as well as the *expected value* of the roll, we defined 2 properties, mainly:

- (1) What is the maximal probability of obtaining a roll value  $\geq x$ ?
- (2) What is the maximal expected roll value which will result after  $k$  number of re-rolls?

This is equivalent to the following Prism properties:

- (1)  $P_{\max}=? [F \sqrt{(d1 + d2 + d3)^2} \geq x \& r = k]$ <sup>3</sup>
- (2)  $R_{\max}=? [F r = k + 1]$

where  $d1$ ,  $d2$  and  $d3$  are the resulting values rolled for Dice1, Dice2 and Dice3,  $r$  is the variable representing the current roll number and  $x$  and  $k$  are the constants representing the *desired value* and the *maximal number of rolls* respectively. In the properties, we are looking for the *maximal probability* as well as the *maximal reward*, this way, the tool will give us the highest possible result based on the *optimal choice* of non-determinism.

The third research question which focuses on the optimal strategy of obtaining a desirable result cannot be answered with the sole use of the Prism tool. For this research question, we need to change our model so that it can be used with Prism-games which extends to *strategy generation* (unlike Prism). For this reason, the model needs to be extended with the specification of a player as well as the module and the actions that the player controls. The model declaration also needs to be changed from MDP to SMG (*Turn-based stochastic multi-player game*). The SMG model can be viewed as a generalization of an MDP and acts the same way as an MDP [10].

To generate a *strategy* with regard to the newly declared player, we also need to change the first property to specify that we are trying to maximize the probability of rolling a desirable value  $x$ . The updated property: " $\langle 1 \rangle P_{\max}=? [F \sqrt{(d1 + d2 + d3)^2} \geq x \& r = k]$ " indicates that the generated strategy will maximize the roll of the first player.

#### 4.4 Results

The results for the first research question "What is the probability (over up to 2 re-rolls) that the roll will result in a value  $\geq x$ , for  $x \in \{1, 2, \dots, 9\}$ ?" can be viewed in Figure 2 where all the probabilities corresponding to roll values from 0 through 9 have been plotted.

From the results we can see that for the first 2 values the probability of rolling at least 1 or 2 is close to 1 (0.9996 and 0.9878 respectively). This probability decreases drastically for the last 2 values, where the probability of obtaining a roll value of 8 or 9 is 0.2959 and 0.1207 respectively.

Just like the first research question, the second question "What is the expected maximal value which will result from the roll (for up to 2 re-rolls) and how does this value change based on the number of re-rolls?" can also be answered with the use of experiments. The results can be viewed in Figure 3 where all the expected values corresponding to the number of rounds ranging from 1 to 20 have been plotted.

From the results we can see that one can expect that the *resulting value* from the first roll is approximately 3. The expected value

<sup>3</sup>Prism does not have a way of representing the absolute value of a mathematical expression, therefore we have to use  $\sqrt{a^2} = |a|$

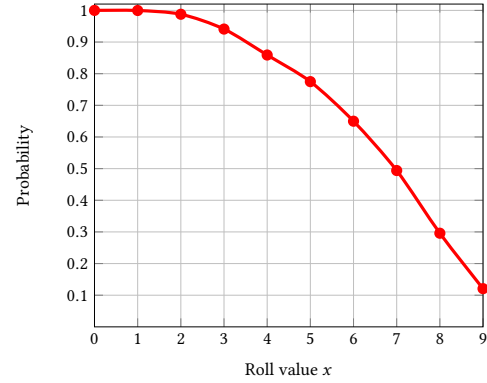


Fig. 2. Probability results for obtaining  $\geq x$  (under the strategy of maximizing getting at least  $x$ )

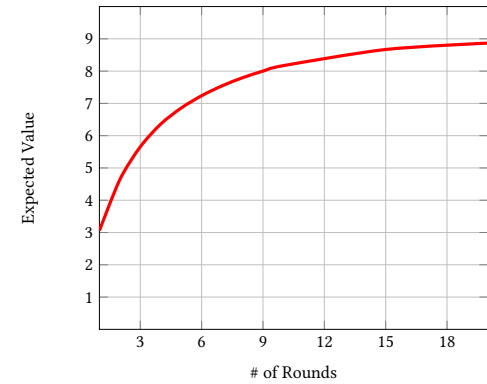


Fig. 3. Expected Value based on round #

however *increases* with the number of allowed re-rolls. Within the rules of the game, being allowed up to 2 re-rolls that is, the player can expect a roll value of 5.655. However, when the number of allowed re-rolls is increased, the expected outcome slowly reaches a value close to 9, the expected value of up to 19 re-rolls is approximately 8.868.

The third research question, mainly: "Which dice is the player encouraged to re-roll to achieve the optimal results?" was analyzed by creating a strategy for the aforementioned player1 property, the resulting strategy can then be analyzed using Prism's *model simulator* where the user can either choose to generate a random path or a path from an existing state.

Since it would be time-consuming to look at every single possible strategy for all values 0 through 9, we analyzed the tool's proposed strategy for *obtaining the value 9*. According to the generated strategy, the player should *keep* all 3 or 4 and *re-roll* all the other values unless a 3 and a 4 have both been rolled. If only a single 3 and a single 4 have been rolled, the player can re-roll either of the 2 dice as well as the third die. If there is a majority of rolls of 3 or 4, the remaining die should be *re-rolled*.

## 5 INCAN GOLD

In this chapter, we will be looking at the *Incan Gold* board game. We will explain the game rules, the established research questions, the methodology for building the model and the results to the defined research questions. However, due to some limitations with regard to memory issues caused by the size of the models but also because of some limitations of Prism, we will only be analyzing the game for only 1 player and a single round.

### 5.1 Game Rules

Information about *Incan Gold* as well as the game rules can be found in [3] (Last Access 02.07.2023).

*Incan Gold* is a board game for 3-8 players who explore a temple and whose goals are to maximize the treasure which the player takes out from the temple over the course of 5 rounds. Before the beginning of a new round, a card (picked out of a set of 5 Artifact cards) is added into the playable card deck. Only 1 new Artifact card may be added to the deck and the 5 artifact cards correspond to the 5 rounds that need be played. The deck of cards contains 30 cards (excluding the 5 Artifact cards) out of which 15 are Treasure cards each card indicating a value of treasures divided amongst the players and the other 15 are Hazard cards (3 copies each of 5 unique hazards). The total value obtainable by all the players combined is 124 gems (excluding Artifact cards).

At the start of each round, the Artifact card corresponding to the round is shown to all players and then shuffled into the deck of playable cards. If the Artifact card is not found by the end of the round, it is kept in the deck for the next round. Every round consists of a variable finite amount of turns. Before a new card is drawn, each player is faced with a decision. They can decide to either continue exploring the temple or bail out the temple, the decisions are made simultaneously. After all the decisions have been made, a card is drawn from the deck of cards and is revealed to all players.

Three possible cards can be drawn from the deck:

- (1) **A treasure card** - Indicates that the players have found a treasure of a given value. The treasure is split (rounded down) according to the number of players that are still exploring the temple.
- (2) **An artifact card** - If an artifact card is found, the artifact is left on the path back from the temple.
- (3) **A hazard card** - If the type of hazard card is drawn for the first time, nothing happens. If this is the second time that the same hazard type was drawn, however, the current round is over and all the players who are still exploring the temple have to give back all the treasure that they have found while exploring the temple (for the current round only). The last drawn hazard card is also removed from the deck for all further rounds.

If the round is not over, the players repeat the turn procedure until either the round ends because of a hazard card, or all the players have bailed out of the temple. When a player leaves the temple, they collect all the current treasure that they have obtained as well as the treasure left on the way out of the temple. If 2 or more people leave the temple at the same time, the treasure left on the way out is split according to the number of players leaving. Any remainder

as a result of division is left on the path. If there is also an artifact on the way out of the temple, the artifact can be collected if and only if a single player is leaving the temple. This means that if 2 players leave the temple at the same time, none of them can claim the artifact and it is left on the path. If an artifact left on the path is not collected by the end of the round, it is lost forever and removed from the game. The first 3 Artifacts to leave the temple are worth 5 treasure points, the latter 2 being worth 10 treasure points.

At the end of the game, all the treasures are counted and the player with the highest amount of treasure points wins. The tiebreaker between 2 players with the same amount of points is decided by the number of artifacts collected, if this is again tied, another round of the game should be played (but for the simplicity of this analysis, we will assume that the game ends in a draw).

Depending on the version of the game the cards in the deck might differ. For this research we will be using the following deck of cards:

- 5 different types of hazard cards - 3 of each
- Gem cards valued at 1,2,3,4,9,13,14,15,17 - 1 of each
- Gem cards valued at 5,7,11 - 2 of each

### 5.2 Research Questions

With regard to the *Incan Gold* board game, we will have the research questions split into two different categories. In this research, we will try to answer the following research questions:

- (1) *With regard to the Game Logic:*
  - (a) What is the expected number of cards that will be drawn in a single round?
  - (b) What is the total expected number of gems in a single round?
  - (c) What is the probability of drawing at least  $x$  amount of cards without finishing the game, where  $x \in \{1, 2, 3, \dots, 20, 21\}$ ?
- (2) *With regard to a Single-Player version of the game:*
  - (a) What is the maximal expected number of gems that a single player can collect under a generated optimal strategy?
  - (b) Given the strategy of maximizing the expected reward for a single player, what is the probability of achieving each possible reward value?
  - (c) What is a general strategy which can be used by a player to maximize the amount of gems they attain?

### 5.3 Model

The approach to the MDP model of the *Incan Gold* board game is very different from the one of the *Combat Dice roll* mechanic described above.

The general model is composed of multiple parts which each have been constructed as *distinct modules* of the model. Below we will show the different modules that we build and also discuss 2 *different models* that we created to answer the above research questions, a *game logic model* and a *single-player model*.

**5.3.1 Game Logic.** The Game Logic model is composed of a *single distinct module*, namely the Board module which contains all the *game logic* with regard to the board game itself. This includes variables corresponding to all the *unique types of cards* in the playable deck (where the value of the variable indicates the number of cards

of that type that has been drawn) as well as the logistics and constraints (in the forms of Prism commands) equivalent to the property of *drawing a card* or the condition of the *game ending* via player withdrawal or by drawing the same type of hazard card more than once. The model also contains 2 distinct reward types, "steps" and "gems" which are used to define important properties about the model. A snippet of the Prism code used to build this model can be seen in Figure 4.<sup>4</sup>

```

1 dtmc
2
3 formula totalCards = 31-h1-h2...-g17-d;
4 formula notGameOver = (h1<2)&...&(h5<2);
5 formula rewardFormula = g1+2*g2...+17*g17;
6
7 module Board
8   //Hazard Cards
9   h1 : [0..3] init 0;
10  ...
11  h5 : [0..3] init 0;
12
13  //Treasure Cards
14  g1 : [0..1] init 0;
15  ...
16  g17 : [0..1] init 0;
17
18  endGame : bool init false;
19
20  //Artifact
21  d : [0..1] init 0;
22
23  //CardDraw or GameOver
24  [cardDraw] notGameOver = true & endGame = false -> (3-h1)/totalCards :
    (h1'=h1+1) +...+ (1-g17)/totalCards : (g17'=g17+1)
25  [endGame] notGameOver = false -> 1 : (endGame'=true) & (h1'=0) &...&
    (g17'=0) & (d'=0);
26
27 endmodule
28
29 rewards "steps"
30 [cardDraw] true : 1;
31 endrewards
32
33 rewards "gems"
34 notGameOver = false : rewardFormula;
    endrewards

```

Fig. 4. Board Game Logic

To answer the above research questions with regard to the Game Logic, we also introduced several properties that will provide help with that matter. In the research questions sub-section, we ask 3 main questions which are of interest to us. To answer this, we define the following properties:

- (1) What is the expected number of steps after which a single round of the game will end?
- (2) What is the expected total number of gems obtainable in a single round of the game?
- (3) What is the probability of drawing at least  $x$  without losing the game?

Which are equivalent to the following properties defined in Prism:

- (1)  $R\{\text{"steps"}\}=? [ C ]$
- (2)  $R\{\text{"gems"}\}=? [ C ]$
- (3)  $P=? [ F \text{"drawnCards"} \& \text{notGameOver=true} ]$

where the *condition* "drawnCards" checks whether the number of drawn cards is  $\geq x$ . The 3 properties correspond to the research questions 1a, 1b and 1c stated above (in the corresponding order).

<sup>4</sup>In this module of the GameLogic, the player withdrawal condition is not added to the transition logistics. Also it should be noted the Artifact card is not counted towards the total reward formula since it might or might not be collected when there are multiple players involved. Both of these will be taken care of in the models discussed further

The results of these properties will be discussed in the next sub-section.

**5.3.2 Single-Player.** The Single-Player model includes both the Game Logic as well as a *Single-Player* added along. All the changes explained below can be seen in Figure 5.

Firstly, we have the same game logic module mentioned in the sub-section above. However, this module now takes into account the *player withdrawal condition* for the single player in the game. Next to this, we introduce 2 new modules, mainly the *Player1* module and the *Controller* module.

As per the rules of the game, before a card is drawn (except when the first card is drawn), each player is faced with a *decision* that needs to be done *simultaneously*, that of leaving or staying in the temple. The decision of the player is modeled as a *non-deterministic transition* where  $p1 = 0$  indicates that the player is still exploring the temple and  $p1 = 1$  that the player is leaving/left the temple. To ensure that the *sequence of moves* between a player making a choice and a card being drawn is respected, we also define a 3<sup>rd</sup> module named *Controller* which contains 2 distinct transitions that are each synchronized with the actions "cardDraw" and "playerChoice" of the Board and Player modules respectively. Another important note is the change in the *gems reward system*, which has now also been modified to only reward the player if they have managed to leave the temple in time<sup>5</sup>.

```

1 mdp
2
3 formula notGameOver...
4
5 module Board...
6   //CardDraw or GameOver
7   [cardDraw] notGameOver = true & endGame = false & p1 = 0 ->
    (3-h1)/totalCards : (h1'=h1+1) +...+ (1-g17)/totalCards : (g17'=g17+1)
8   [endGame] notGameOver=false | p1=1 -> 1 : (endGame'=true) & (h1'=0) &...&
    (g17'=0) & (d'=0);
9
10 endmodule
11
12 module Player1
13   //Player state (playing or not)
14   p1 : [0..1];
15   [playerChoice] p1=0 -> true;
16   [playerChoice] p1=0 -> 1 : (p1'=p1+1);
17
18 endmodule
19
20 module Controller
21   //Order of action
22   c : [0..1];
23   [cardDraw] c = 0 -> (c' = 1);
24   [playerChoice] c = 1 -> (c' = 0);
25
26 endmodule
27
28 rewards "gems"
29 [endGame] notGameOver = false : 0;
30 [endGame] notGameOver = true & p1 = 1 : rewardFormula;
31 endrewards

```

Fig. 5. Single-Player logic

To answer the research questions stated above we also define the following property to aid in finding solutions to those questions:

- (1) What is the expected number of steps after which a single round of the game will end (this time, the player withdrawal condition is taking into account)?

Which is the equivalent of the following Prism property:

- (1)  $R\{\text{"gems"}\}max=? [ F \text{endGame=true} ]$

<sup>5</sup>In this model, the reward formula was adjusted to contain the Artifact card as well, mainly  $+5 * d$

As with the *Combat Dice Roll* model, to generate *effective strategies*, we need to slightly modify the model in Figure 5. As before, the only addition required is that of changing the model type to an SMG as well as specifying the commands that the player has control of. Similarly, the *identifier «1»* needs to be added in front of the defined property to enable the user to generate a strategy. The results of the analysis is described in the next sub-section.

### 5.4 Results

**5.4.1 Game Logic.** The results to the first two research questions with relation to the game board logic, mainly "What is the expected number of cards that will be drawn in a single round?" and "What is the total expected number of gems in a single round?" can be answered simply by verifying the first and second property defined in the model section.

According to the model we have build, a *single round of Incan Gold* with the deck defined above *lasts an expected number of 7.656 turns*, which means that a game is expected to end via the drawing of 2 of the same type of hazard card between the 7<sup>th</sup> and 8<sup>th</sup> card drawn in the game. Next to this, in a single round of the game, a player can expect a *total of 29.668 gems*.

The results of the third research question "What is the probability of drawing at least  $x$  amount of cards without finishing the game, where  $x \in \{1, 2, 3, \dots, 20, 21\}$ ." can be seen in Figure 6.

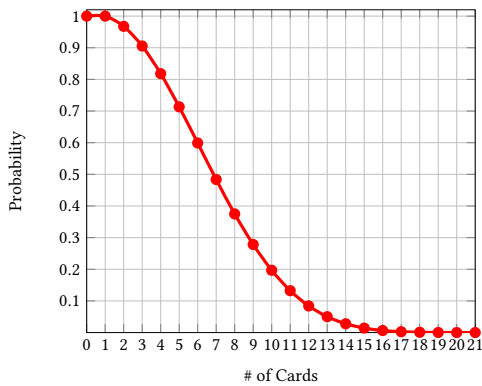


Fig. 6. Probability results for drawing at least x cards

As we can see from the results presented in the graph, between the 7<sup>th</sup> and the 8<sup>th</sup> card drawn, the represented probability has a value slightly below 0.5, confirming the results of the first research question stating that the round is expected to end on the 7<sup>th</sup> or 8<sup>th</sup> turn. We can also see that after the 15<sup>th</sup> card drawn, the chance that the game will not end approaches the value 0 indicating the *unlikelihood of the round lasting more than 15 turns*, where the likelihood of the event drops below 1%.

**5.4.2 Single-Player.** The first research question, mainly "What is the maximal expected number of gems that a single player can collect without losing the game?" can be easily answered by verifying the results of the defined property. According to the model, the *expected number of gems* a player single player can expect to obtain (using the best strategy) is *17.92 gems per round*, which differs from

the value showed in the previous section (that of 29.668) since the player is more likely to lose at that point in the game. This then helps us answer the second research question which asks about the probability of obtaining all the possible *reward values* and about their probabilities. The results to this research question can be seen in Figure 7

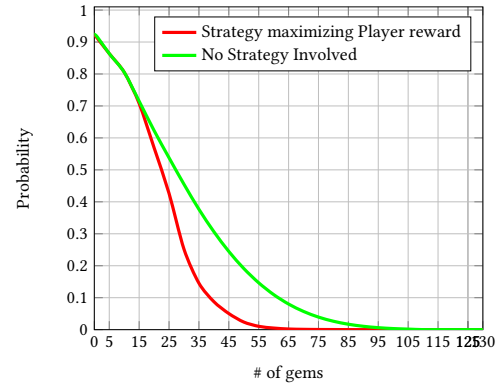


Fig. 7. Probability distribution of obtaining every possible amount of gems

As we can see from the graph, the probability of obtaining at least  $x$  amount of gems for the given strategy decreases in probability *quicker* than just the probability of obtaining at least  $x$  gems. Which indicates that the player *should withdraw sooner* and is *less likely* to achieve higher amounts of rewards while trying to minimize the risk of losing.

The results to the second research question can also be complemented with the results to the last research questions: "What is a general strategy which can be used by a player to maximize the amount of gems they attain?". The results were obtained by running several different scenarios that might affect the choices that an individual player might take. We have run different scenarios based on the probability of a player losing the game which is equivalent to the unique types of hazard cards currently on the board. According to the generated strategy, the following results can be observed based on the number of types of hazard cards on the board:

- If there is not a single hazard card on the board, the player can just keep on acquiring gems.
- For a single type of hazard card, the player should leave at the point they have achieved 43 gems
- for 2 different types, the player should leave when they achieved 26 gems
- for 3 different types, they should leave after 18 gems.
- for 4 different types, 15 is the lowest number of gems that the player should leave after
- If there are 5 different hazard types, the player should leave after achieving 12 gems.

This also explains the drop in probability indicated in Figure 7 but also the initial overlap in probabilities since the player is never encouraged to leave before achieving 12 gems (for 5 different types of hazards).

## 6 CONCLUSION

In this paper, we have investigated how board games such as *Incan Gold* or the *Hobbit Adventure game* can be modeled as MDPs and further analyzed using the model-checking tool *Prism*. This was done, to further try and answer the general research questions about modeling board games as *Markov Decision Processes* established in the Introduction. This section will focus on showing how the analysis of the aforementioned board games helps us answer the general research questions stated at the beginning of the paper.

*RQ1: How can the application of MDPs to board game analysis aid in identifying optimal strategies, evaluating game balance, or predicting the outcome of games?* As we saw from the *Combat Dice Roll* mechanic, as well as the single-player model of the *Incan Gold* board game, modeling a board game into an MDP resulted in the ability to define interesting properties of the model. Such properties included the generation of optimal strategies for maximizing a probability or a reward.

*RQ2: How does the complexity of a game impact the resulting size of the created model?* Although this research question cannot be directly answered based on the models shown in the paper, for the board game *Incan Gold*, creating a model and generating a strategy which included more than 2 players or more than 1 round proved to be expensive both in terms of memory and generation time required to build the model and generate the strategies. The complexity of the *Incan Gold* board game showed that the resulting sizes of the model can be in the order of billions of states and transitions.

To summarize, we have shown that board games can be modeled as Markov Decision Processes and then analyzed to answer important properties of the model, but the process of generating and analyzing the model depends on the complexity of the game. While some simplistic board games could be fully modeled as MDPs, other games are limited by the capacity of the tools used for the analysis.

## 7 LIMITATIONS AND FUTURE WORK

While building and analyzing the model, we encountered some limitations with regard to the board game *Incan Gold*. The main limitations were caused by the size of the model that the tool *Prism-Games* was able to support. When trying to extend the model to contain more than 1 player, the tool constantly run into memory issues due to the large size of the model which caused it to crash. This problem would not only appear during model generation but also when trying to generate or import a strategy for the model. As a result, we did not expand the model to contain more than a single player or more than a single round and were unable to generate a suitable strategy for more than a single player due to the limitations of *Prism* with regard to model generation and properties specification. In particular, while between 2 players the generation of a game and strategy is limited just by the size of the models, for more than 2 players another limitation that comes into play is the strategy generation. Some attempts at expanding the model beyond a single player were done using a simplified model of the game, however, even for 2 players, trying to expand the original model proved to be unfeasible because of constant memory issues which led to the app crashing when trying to load the model. However,

there are still possible improvements or adjustments which can be done to try and incorporate most of the aspects of the game:

First of all, the model could be simplified such that it only contains a single variable describing all the gem cards. This variable could represent the average value of all the gem cards combined.

Another possible solution is to use a different tool to build and analyze the model. One of the short-comings of the *Prism* tool is that the model did not support strategy generation, for which *Prism-games* had to be used, however, *Prism-games* has limitations in itself as it only supports a limited amount of models as well as some limitations in defining the properties with regard to more than 1 player in the game. For future research, other tools such as *Storm* [6] could be used to overcome the limitations mentioned above.

Another limitation of the project where the resources used to run *Prism*. Stronger hardware in terms of RAM and processing speed would greatly increase the model-building speed process, however, this does not ensure that memory issues would not arise while trying to handle larger models.

Finally, probabilistic board games could be analyzed to answer the general Research Questions this paper tried to address, however, because of the limitations of *Prism* and *Prism-games*, not every probabilistic board game is suitable to be analyzed with the tool. Simple board games such as *Snake and Ladders*, the *Game of Goose* or *Across the Board* whilst more complicated board games where large models are required, such as *Monopoly*, *7 Wonders* or *Catan* are more challenging to model and analyze which would be interesting to analyze in the future.

## 8 ACKNOWLEDGMENTS

Special thanks to Dr. M.Volk and Dr. M.A. Lopuhaä-Zwakenberg who were the supervisors of this paper and provided meaningful insights and a sturdy guideline for this research. Additional thanks to the track chair dr. P.Lammich and to the University of Twente for providing this opportunity.

## REFERENCES

- [1] Robert B Ash and Richard L Bishop. 1972. Monopoly as a Markov process. *Mathematics Magazine* 45, 1 (1972), 26–29.
- [2] Rebecca Yvonne Bayeck. 2020. Examining Board Gameplay and Learning: A Multidisciplinary Review of Recent Research. *Simulation & Gaming* 51, 4 (2020), 411–431.
- [3] Bruno Faidutti, Alan R. Moon, Jörg Asselborn, Matthias Catrein, Robert Islas, Prapach Lapamnuaysap, Paul Mafayon, Claus Stephan, and Christof Tisch. 2005. *Incan Gold*. <https://boardgamegeek.com/boardgame/15512/incan-gold>
- [4] Jan Friso Groote, Freek Wiedijk, and Hans Zantema. 2016. A Probabilistic Analysis of the Game of the Goose. *siam REVIEW* 58, 1 (2016), 143–155.
- [5] Jo Hartwig H. Jean Vanaise. 1994. *The Hobbit Adventure Boardgame*. <https://boardgamegeek.com/boardgame/1586/hobbit-adventure-boardgame>
- [6] Christian Hensel, Sebastian Junges, Joost-Pieter Katoen, Tim Quatmann, and Matthias Volk. 2021. The probabilistic model checker Storm. *International Journal on Software Tools for Technology Transfer* (2021), 1–22.
- [7] Joshua T Hertel. 2015. Understanding risk through board games. *The Mathematics Enthusiast* 12, 1 (2015), 38–54.
- [8] Joost-Pieter Katoen. 2016. The probabilistic model checking landscape. In *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science*. 31–45.
- [9] Marta Kwiatkowska, Gethin Norman, and David Parker. 2011. PRISM 4.0: Verification of probabilistic real-time systems. In *Computer Aided Verification: 23rd International Conference, CAV 2011, Snowbird, UT, USA, July 14–20, 2011. Proceedings* 23. Springer, 585–591.
- [10] Marta Kwiatkowska, Gethin Norman, David Parker, and Gabriel Santos. 2020. *PRISM-games 3.0: Stochastic game verification with concurrency, equilibria and*



- time. In *Computer Aided Verification: 32nd International Conference, CAV 2020, Los Angeles, CA, USA, July 21–24, 2020, Proceedings, Part II* 32. Springer, 475–487.
- [11] V Sundarapandian Raani. 2021. *Modeling and analysis of board games*. B.S. thesis. University of Twente.
- [12] Csaba Szepesvári. 2010. Algorithms for reinforcement learning. *Synthesis lectures on artificial intelligence and machine learning* 4, 1 (2010), 1–103.
- [13] Barış Tan. 1997. Markov chains and the RISK board game. *Mathematics Magazine* 70, 5 (1997), 349–357.
- [14] Michael Wakin and Chris Rozell. 2004. A Markov Chain Analysis of Blackjack Strategy. *Rice University* (2004).