Universiteit Leiden
The Netherlands

Opleiding Informatica

The Leprechaun Hatting Problem

Egon Janssen

Supervisors:
Walter Kosters
&
Jeannette de Graaf

BACHELOR THESIS

Leiden Institute of Advanced Computer Science (LIACS)
www.liacs.leidenuniv.nl

18/8/2017

Abstract

This paper investigates the Leprechaun Hatting Problem. This problem can be categorized as a mathematical puzzle problem. This category of problems has many ways to solve or finding a solution to the game. In this thesis the Leprechaun Hatting Problem will be solved to find the best solutions for this game. This problem is also in the category of Artificial Intelligence because there are agents(Leprechaun's) which need to guess their hat color based on the other agents.

The first similar appearance of this problem was by Martin Gardner 3 Prisoners problem [Gar06], in which prisoners needed to guess their chances of getting pardoned. The first occurrence described with the Leprechauns as the agents was by Todd Ebert [Ebe98]. He described for this problem that the key of solving this problem was to find the maximal properties for a individual strategy for the agents. To prove that to maximize the win chance of the players the strategy needs these properties, this paper will prove it by simulating all relevant deals. The thesis seeks to answer the question of how to find the strategies with the most desirable outcome. Furthermore this thesis will try to find a method to answer this question in the shortest possible time to calculate these deals.

This thesis will discuss and evaluate two methods, which make deals in two different ways. The first method is the Brute Force Method, which is a standard method for creating deals. The Brute Force Method creates each possible deal by making all combination of strategies possible for each agent. This is the exhaustive but complete method, and will be used as baseline to test the effectiveness of the second method. The second method is the Chain-Method, which is more robust and will not cover all strategies. The aim of this method is to use the properties of the maximal deals. The thesis defined so-called *Chains* to contain these properties.

Contents

Chapter 1

Introduction

This thesis will discuss the difficulties and considerations in finding solutions for the Leprechaun-Hatting Problem. The Leprechaun-Hatting problem is a problem within the fields of Game Theory, especially within the Mathematical and Puzzle field of Game Theory. The Leprechaun-Hatting Problem first appeared for public audience in the publication *Mathematical Puzzles: A Connoisseur's Collection* written by Peter Winkler [Wino3] published in 2003. Here Winkler described the principles of the Leprechaun-Hatting Problem in a mathematical format and explained some mathematical properties. Other researchers as Todd Ebert [Ebe98] and Martin Gardner [Garo6] discussed similar problems giving insight in different ways. Many of these researchers took a more mathematical approach to support their theories. We found originally the inspiration for this problem in the book [vDK15]. This thesis however focuses on a Computer Science approach by specifying algorithms to find solutions for this problem.

Leprechaun-Hatting Problem The Leprechaun-Hatting Problem is a cooperative game. Typically there are three players playing the game. The game starts with the players designing a tactic, which describes how and in which situation each player will act. After this tactic has been designed, each player will then get a Leprechaun Hat assigned; these can be either blue or red. In order for the players to win one single round, one of the players should guess their hat color. The other remaining players have two options in order to win, first they can also guess their hat color correctly or secondly they are allowed to say nothing and pass. Passing is neither good nor bad, because choosing to pass will never win or lose this round. The players will lose directly if one of the players makes a mistake in guessing their hat color.

Nontrivial Cases The Leprechaun-Hatting Problem described above is the standard case, however many of the rules can be altered or extended to create another version of the Leprechaun-Hatting Problem. To generalize the standard case the Leprechaun-Hatting Problem has N players playing this game. Furthermore the number of hat colors can be generalized to M colors. Another parameter within this problem is concerned with the rules, which specify the winning and losing conditions. These rules can also be altered and or completely rewritten. An example would be that at least half of the players needs to guess the correct hat color.

Methods This thesis will focus on two algorithms, the Brute Force Algorithm and the Chain Algorithm. *The Brute Force Algorithm* is the first algorithm this thesis will discuss and elaborate upon. This algorithm has the property to compare all possible solutions to each other, making it the base case for this research. The Brute Force Algorithm is an exhaustive and complete method, and will therefore have all details and specifications of each single solution and its relative performance. The Brute Force Algorithm can be used as the baseline to test the effectiveness of the second method.
The Second algorithm is the *Chain-Method*, this method will selectively pick solution based on desirable *Chain* properties. They consist of two properties, the *All for One* property and the *One is Enough* property:

All for One This property will ensure that no player makes an other player's answers incorrect by answering an incorrect hat color. This property is obvious when playing a cooperative game, however it gives much insight in the solution that we will be created.

One is enough This property will ensure that a player does not use their possibilities he/she has to make an already correct round again correct. This property is only applicable to the standard case, because the rules allow this kind of victory.

These properties will be explained further in Chapter 4.The Chain-Method is proposed by this thesis to provide fast and reasonable performing solutions. The desirable solutions for this method are the solutions which perform equivalent to or better than any other solution possible.

Results The Results show that the proposed Chain-Method indeed results in the deals found by [GKRT06]. The Results further more showed insights in how the Chain-Method behaves with standards with four agents. The results show also that the Chain-Method does not reduce the computational power to find the complete set of deals for the standard with five or more agents.

Thesis Overview This chapter contains an introduction to the thesis; Chapter 2 gives definitions for the theory supporting the methods proposed by this thesis; Chapter 3 discusses related work which contributed and formed concepts for this thesis and the methods proposed by the thesis; Chapter 4 explains how the chains work and how the method works step by step. Chapter 5 analyzes the results founded by testing the Brute Force Method and the Chain-Method. This chapter also analyzes the relation between the two methods by means of results found; Chapter 6 summarizes the current progress and insights in the Leprechaun-Hatting Problem made by this thesis.

This report is a bachelor thesis on the Leprechaun-Hatting Problem, written by Egon Janssen, as part of the Bachelor Computer Science at the University of Leiden, at the Leiden Institute of Advanced Computer Science (LIACS). This thesis was supervised by W.A. Kosters and J. de Graaf.

Chapter 2

Definitions

This chapter is dedicated to provide definitions and theorems to explain the Leprechaun-Hatting Problem game. The chapter is divided into two sections to elaborate the two parts of this thesis. The first part will explain and create some terminology to form a baseline of information for this research. The first Method, *The Brute Force Method*, will use this information. The second part focuses on introducing new definitions and theorems to support a new method, *the Chain-Method*, which generates only the best solutions.

2.1 The Standard Case

This Section will create a terminology for this thesis, to prove the correctness of the results and create a better understanding of the Leprechaun-Hatting Problem. This chapter will formulate and explain the basic rules and concepts of the game. In addition this Section will elaborate new theories suggested for the Leprechaun-Hatting Problem. These theories will be providing information which has contributed to the development of the first Method, the *Brute Force Method*.

To understand the characteristic aspects of the game, certain terms must be elaborated. Some of the terms are defined by other research papers, others are defined by the logic from the Leprechaun-Hatting rules. These terms will explain the game fundamentals and give insight into the overall complete problem situation.

2.1.1 The World

A World is the environment in which the Leprechaun-Hatting Problem is resolved in. Such a World contains rules and parameters specified by certain member values of the World. These members can be distinguished into two quantifications, rules and values. Each member value gives the World a different play style which will result in different behaviors when other values are used for these members values. The most common form of setting these member values is at the creation of the World.

If however these member values are changed afterwards this will cause inconsistency in the World and this will make the Deals before changing the member values incomparable with the Deals evaluated after changing the member values. Therefor we state that we do not alter a World during evaluations.

Values Values are members of the World, which like most of the members, are set at the creation of a World. Most of the values influence the size of the game, the calculation time and the options in evaluating a round. Each World can be generalized to have the following standard values assigned.

1. The number of *agents (Leprechauns)* N in this World. Here N is an integer with $N \geq 3$. It represents the number of agents (Leprechauns) there are available in the game. They will individually play the game and work together in order to win as many rounds as possible. The current round will also tell the agents (Leprechauns) how their views look like. This view is the information of the other agents that they need to base their strategy on. In the Standard World N will be 3, to keep the World as simple as possible.

2. The number of different colors M that can occur as the *agent's (Leprechaun's)* hats. Here M is an integer with $M \geq 2$. Representing a variety of colors, like Red (R), Blue (B), etc. In the Standard World there will be 2 colors, Red (R) and Blue (B) .

3. A probability set for the odds of each of the colors in the World. In the standard situation, the distribution for each color will be the same, so probability for a hat to be Blue (Red) is equal to a half.

Rules Rules are similar to values, however they do not assign a value to the game behavior. Rules instead form the basis of resolving the game. These rules have also an effect on the way in which the problem can be resolved. Within a Standard World the following rules are applied:

1. All the agents (Leprechauns) are allowed to formulate only one answer each round. This answer will contribute to the correctness of this round.

2. The agents (Leprechauns) can formulate an answer within the range of the variety of colors and are allowed to say nothing/passing in a round.

3. In order to have a round to be *Correct*, at least one of the agents (Leprechauns) needs to answer correctly. An agent answers correctly if he/she gives an answer, and the given answer is the same color as the color of the hat he/she is wearing. If some agent answers incorrectly, this will make the round incorrect.

4. Whenever an agent passes in a round, his/her answer is neither correct nor incorrect. Which gives the property that individual passing as an answer does not directly make a round incorrect. Only collective passing as answer makes a round incorrect.

5. All agents (Leprechauns) formulate an answer at the same time, meaning they formulate the answer and will then say their answer out loud all at the same time. After they all simultaneously tell their answers, the round will be evaluated. The round is evaluated by comparing the agent's hats and their answers.

Purpose of a World The purpose of a World is to test and evaluate different *Deals* to find the most desirable solutions for this World. These desirable solutions are solutions where the set of correct rounds is unique and maximal. These desirable solutions are obtained in many different Deals, depending on which criteria these solutions are based. A Deal defines how the agent will decide how to respond to possible game situations. Within the environment of one World multiple Deals can be tested. Each Deal can be evaluated and compared with the same environment. Determining the performance of a Deal is done in a *Play Round*. The information gathered in a Play Round is necessary to determine the performance of the Brute Force and Chain-Method. To define all Deals, each possible combination for all *Strategies* for all agents (Leprechauns) will be used. To rate each of these Deals and find the best Deals for a given World, each Deal is rated by a fitness function called *Fitness of a Deal*. There will be four kinds of World.

- The **Standard World** defined with the standard rules and the standard values. Here the number of agents N is equal to 3 and the number of colors M is equal to 2.
- The **Chance-Based World** is defined as the Standard World, however with one extra rule. This rule states that instead of creating an answer by what the agent can see, there is a probability to define the answer for the agent. This extension of the decision making for the agent can create each Deal with the standard rules, but also Deals which were not possible in the Standard World.

  For example if we have a World with three agents and two colors. Then a strategy for the Chance-Based World would be that agent one always answers the same way no matter what the agent sees. But instead of saying always Red, he wants to have it that 3 out of 4 games the agents answers red and the other game he answers blue. We would get a logic rule that looks like

  > ***If Agent 1 Sees {Red Red} he picks Random Between Red and Blue where Red has a 75% chance of being picked.***

- Another type of World would be a World where the rules are completely different. An example of such World would be a World where instead of the rule that answering can only occur once each round would be altered to a behavior where a round could exist of multiple sub-rounds. For this example it would be sufficient to replace the passing rule, by replacing this rule by a rule that allows the agents (Leprechauns) to enter a new round when all agents pass the first round. The Chance-Based World could be in this category, however, many consider this as a special case of World involving a non-deterministic case.
- The last type is a World where the values are different from the standard case. Examples of other values are:
  - The agents (Leprechauns) do not know their position and do not know if their view remains the same each time they need to guess. This can create ambiguity in seeing Red,Blue and Blue,Red.
  - The amount of agents is not equal to the number of agents described in the Standard World. Thus a World where $N > 3$ holds.

– The amount of colors is not equal to the number of colors described in the Standard World. Furthermore this could also be altered to represent other colors than given in the Standard World, this however does not influence the outcome or the approach. Thus a World where $M > 2$ holds.

2.1.2 Other World Standards

Chance-Based World In a World with Chance-Based rules the decision making of the agents requires more information than in the Standard World. The decision making will first compare the current view, the other hats the agent sees, to the agent's part of the Deal, the *Strategy*. This gives the agent in a Standard World the answer he/she should give. In a Chance-Based World the strategy instead describes a distribution of chances for the colors. The normal decision making looks like:

> ***If the colors I see are equal to the colors defined by my strategy then I think my hat color is Color X.***

Instead the decision making in a Chance-Based World will look like this

> ***If the colors I see are equal to the colors defined by my strategy then I think my hat color could be color X or color Y, but I do not know for sure.Let us flip a coin to decide whether it is color X or color Y***

With this new decision making the problem becomes non-deterministic, because of the chance-based property in the decision making. This requires for this type of World to be evaluated in a different method than the Standard World. To discard the irregulars and the imperfect true randomness each Deal should be tested multiple times to probe the reliability of a Deal. In order to combine the results into one performance function some sort of combined evaluation should be made. An easy method of giving a reliable performance function would be taking the median of all the performances over the individual Play Rounds.

Multiple Rounds World In this type of World an extra complexity is added through rules. In this type of World there are certain conditions defined in which a round is not ended after all agents answered once. Instead the agents are asked to formulate another answer. Each sub-round that is allowed to be made this way, increase the number of strategies and will therefore also make each strategy more difficult and having more conditions.

2.1.3 Leprechaun

The agents in this game are called the Leprechauns. The collective goal of these agents (Leprechaun) is to create a Deal together and have this Deal to perform as good as possible. Each agent has his own strategy which together with all agents forms a Deal. To properly execute this Deal each agent has been provided with the following information source in a World:

- The agent's hat color, with the range of $\{1, 2, \ldots, M\}$ representing a color.
- A unique identifier to distinguish each agent (Leprechaun) in the *World*, defined by a number within the range of $\{1, 2, \ldots, N\}$.
- The agent (Leprechaun) can execute his/her strategy and this will create an answer in the form of a color. The strategy can be executed without the knowledge of the agent's own real hat color.
- The agent (Leprechaun) is able to see the hat colors and the unique identifiers of all other agents (Leprechauns) in this *World*.
- Each agent knows the specifications of the Deal and will not alter his/her strategy during a Play Round.

Chances-Based Agents The Chance-Based Agents are agents (Leprechauns) and are specific agents to the Chance-Based World. They differ from standard agents (Leprechauns) in their usage of the logical rules they use to create an answer.

Multiple Rounds Agents The Multiple Rounds Agents are agents (Leprechauns) which are specific to the Multiple Rounds World. They differ from standard agents (Leprechauns) in the behavior that they execute when evaluating a single round. This has however also impact on the strategies the agents can create, beside the standard Deals. They have to calculate further than one round ahead. This makes the problem much more complex, because this allows infinite numbers of sequential rounds, and therefore infinite numbers of strategies and Deals. This makes the game mechanics much more complex and harder to evaluate all possible Deals. And now one will have to focus more on the maximum number of sequential rounds instead of optimizing the search for the best performing solutions.

2.1.4 Play round

A play round takes place in an already initialized *World*. A Play Round starts by defining a Deal for the agents (Leprechaun). When each agent (Leprechaun) has received his strategy, the Deal will be evaluated by testing in M^N *Rounds*, so that each possible color combination for all agent hats have been handled. The Play round will be evaluated based on the information retrieved from the rounds. The standard approach of evaluating the Play round is to count the number of Rounds that evaluated correctly. The standard form for the fitness of the Deal is represented as a percentage defined by the following equation

$$FitnessOfADeal = \frac{numberOfRoundsCorrect}{numberOfRounds} * 100\%$$

2.1.5 Rounds

A round is a sequential component of a *play round*. At the start of a round all agents (Leprechauns) get a hat color assigned depending on the round. Each round is a color combination that can be created for the agents. Examples for this are { Red Red Red } but also { Red Blue Blue }. The hat color is assigned as an integer in the range $\{1, 2, ..., M\}$. In each round all *agents (Leprechauns)* create a *Thought* based on their *Strategy*, which is part of the Deal. For the standard case this Thought is the same as the answers given. However for a non-deterministic World, like the Chance-Based World, the thought holds the Chance-Based Rules. This thought is generated and revealed at the end of the round, for all *agents (Leprechauns)*. The evaluation whether a round is correct is when the following rules are applicable:

- Each agent (Leprechaun) generates an answer executing his/her strategy. This will result in a hat color or in the special case of passing.
- At least one of the agents executes his/her strategy and creates an answer which is equal to his/her current hat color.
- There are no agents answering incorrectly their current hat color.

2.1.6 Deal

A Deal is a set of *N Strategies*. These Strategies are the agents individual Strategies in the current *World*. A characteristic is that a Deal can be evaluated in terms of the *Fitness of a Deal*. Another observation we can make is that we can test a Deal, but we cannot test a single strategy as this is for a single agent. A Deal is the agreement the agents made at the start of evaluation of the Deal. It will determine the behavior of all the agents during one evaluation.

2.1.7 Fitness of a Deal

The *fitness of a Deal* is a representation of how a Deal performs. The fitness of a Deal can be expressed in many ways and can held in account for many performance values. Often the fitness is represented as a percentage. Some standard fitness functions would be the Rounds Correct, the Strong Answering functions and Set Correct.

Rounds Correct The rounds correct function represents a percentage of K/M^N. Here K is the number of correct evaluated rounds. Which evaluates the Deals as the number of correct rounds. However in this Problem there are cases where one Deal results in the same amount and set of correct rounds.

Strong Answering The strong answering function is a function that looks at the Deal at a deeper level than the rounds correct. It evaluates based on the number of times the agent answered correctly or incorrectly. This however can be implemented in different ways, because unanswered or passed answer is neither correct or incorrect. Which leads to many variations of this function depending on the implementation of the passing value.

Set Correct The set correct function is a function that will not result in a number, it will instead result in a ranking between the different Deals. It will base this ranking on the rounds the Deals evaluates correctly. The ranking however includes the color combination represented by the round which gives more information than the Rounds correct function.

2.1.8 Strategy

An individual strategy is a behavior/set of rules that apply to an agent. This set contains behavior rules based on what this agent knows about the other agents. These rules contain a condition about the current state of the other agents, if this condition is true for the current situation, the agent will answer a color represented as an integer in $\{1, 2, ..., M\}$, described by their *Thought*. This color will be the representation of the *Thought*. Beside that the answer can be represented as a color it is possible the strategy defines that the agent should not answer any color. For this special case the thought can be represented by a special thought, the thought of passing.

Thought The thoughts are created as result of the *Strategy*, the thoughts represent what the *agent (Leprechaun)* thinks/calculated following it's own strategy rules about the color of the hat that he is wearing. The color that is generate as thought is in $\{1, 2, ..., M\} \cup \{0\}$, where $\{0\}$ is the special case of passing.

Logic Rule A Logic Rule is a rule for an agent. When an agent needs to make a decision, it will make a decision based on the rules the agent has. The agent compares in which situation of hat colors he is currently in, executing the corresponding rule. Then the rule states an answer for this situation. Each agent will therefore have logic rules for each possible situation it can occur in. In the general case that is M^{N-1}. All the logical rules are of the form:

> ***If the colors I see are equal to the colors defined by my strategy then I think my hat color is***.

-

2.1.9 Observations

With these definitions discussed in the previous paragraphs, some observations can be made. These observations reveal certain boundaries of the problem, when using the brute force method. Therefor they describe

the maximum size of the problem. Each of the observations below are set in a Standard World as described Section 2.1.1. Then for such a World W, the following properties are described, for the World M is the number of colors in the World W and N is the number of agents in the World W.

Each agent in the World W has the property that it can see a number of agents. This number effects how this particular agent decides it's answer. It is equal to the total number of agent except for himself. This can be summarized in the next equation:

$$OtherLeprechauns = N - 1$$

The number of logical rules each agent has. These rules together construct a strategy. An agent a rule for each possible combination of hat colors that can occur when seeing the other agents. This can be summarized in the next equation:

$$LogicRules = M^{OtherLeprechauns}$$

The number of resulting colors an agent can answer. An agent can answer with all colors defined by the World W, however they can also pass which is a special case of answer. This can be summarized in the next equation

$$ResultColors = M + 1$$

The number of rounds that take place in evaluating one Deal is equal to the number of combinations that can be made for the hat colors of the agents. This can be summarized in the next equation

$$Rounds = M^N$$

The number of strategies a single agent can have is independent of the other agents. This is all strategies that can be formulated and each of them is a different combination of all logical rules combined with a color that the agent can answer. This can be summarized in this equation:

$$Strategies = ResultColor^{LogicRules}$$

The number of Deals in this World, is the number of strategy combination. This number of Deals is equal to all possible combinations of all the strategies possible for each agent. This can be summarized in this equation

$$Deals = Strategies^N$$

The number of answers that have to be created in order to evaluate all Deals.This can be summarized in this equation:

$$Thoughts = Deals * Rounds * N$$

Theorem 1. Let the fitness of the Deal be of the function type Set Correct, then the fitness of the Deal will result in a set of color combinations. This set will represent the rounds that the Deal correctly answered. For this set of color combinations we introduce a property. A set of color combinations A can overrule another set

of color combinations B if A contains all elements from the set of B and has remaining color combinations left, we can say that

$$B \subset A$$

. Therefore if there is a color combination A which overrules B, then there is another set of color combination called C where

$$B \cup C = A$$

there is a way to extend B to A. For all sets of color combination ,which holds that there is no set which applies the above definition, we define these sets as maximal sets of Color Combinations. We call the Deals that have a maximal complete unique set of Color Combinations a optimal Deal.

Theorem 2. As shown in Section:2.1.9 the number of Deals in a World W for small N and M like $N = 3$ and $M = 2$ the number of Deals is already a huge calculation. For the W the number of Deals would be

$$Deals = (2+1)^{(2^{(3-1)})} = 531,441$$

And the real number of calculations is equal to the times an agent needs to answer, and therefor compare his answer with his hat color. Which is equal to the number of answers that should be created in order to evaluated all Deals, which is

$$Rounds = 2^3 = 8$$

$$Thoughts = 531,441 * 8 * 3 = 8,503,056$$

Corollary 1. As proved in Theorem 2 over a half million Deals are tested, and more than 8 and a half million calculations are made. But only 14 of these Deals are found to be maximal, proved by the Paper The Hat Problem And Some Variations [GKRT06] in Chapter 29.6. With this insight the need to create a more efficient method of finding the desirable solutions is formed.

2.2 Example of the Game Play

As example with the theorems discussed. Example 1 is defined as the following case below. To prove the operation of these theorems.

- Generating a World called World A described as a Case 1 World (Standard World) with the following properties. World A has the properties to be with in reasonable bounds of understanding.
  - $N = 3$, there for there are three agents.
  - $M = 2$, with the two colors "Red(R) and Blue(B)".

  furthermore with these properties set, the other values can be evaluated as well. (Section 2.1.9)
  - $OtherLeprechauns = 3 - 1 = 2$.
  - $LogicRules = 2^{OtherLeprechauns} = 2^2 = 4$.
  - $ResultColors = 2 + 1 = 3$
  - $Rounds = 2^3 = 8$
  - $Strategies = (ResultColor)^{LogicRules} = 3^4 = 81$.
  - $Deals = Strategies^N = 81^3 = 531,441$
  - $Thought = 531,441 * 8 * 3 = 8,503,056$

Testing a Deal To test a Deal within World A *the fitness of a Deal*.
As earlier described in World A can test multiple Deals. To demonstrate value of the fitness of a Deal. A Deal will be tested in World A down below. The Deal is also shown in Figure 2.1

- The Deal with the following properties, consisting of N Strategies
  - For agent (Leprechaun) 1, 2 and 3 : If this leprechaun sees two equal hat colors (for example "Red" "Red"), it will generate the thought of the other color (in the example "Blue"). Otherwise if it sees something else this leprechaun generates a passing thought.
  - The Logic Rule in these strategies are as followed described.
    When $Color, Color -> !Color$
    Else if $Color, !Color -> "Pass"$.
- Evaluating 8 = M^N rounds. Evaluated as:
  - 2 rounds, rounds 1 and 8 are evaluated as invalid.
  - 6 rounds, rounds 2 - 7 are evaluated as correct.

- The fitness of the Deal is equal to the number of correct(6) divide by the number of rounds(8) as percentage: $6/8 = 75\%$.

Other examples of Deals are:

- *Full passing*, this Deal will pass for each agent no matter which situation occurs. Each agent will have logic rules in the format of $Color1, Color2- > "Pass"$. Because each answer is "Pass" each round will have three "Passes" and doesn't meet the requirement that at least one Agent made a correct color guess. There for this Deal has a win rate of 0%.

- *Always Color*, this Deal will have all agents execute the *All Color* Strategy which gives the agent for each Logic rule a Color answer. Therefor each agent will have logic rules in the format of $Color1, Color2- > Color$.

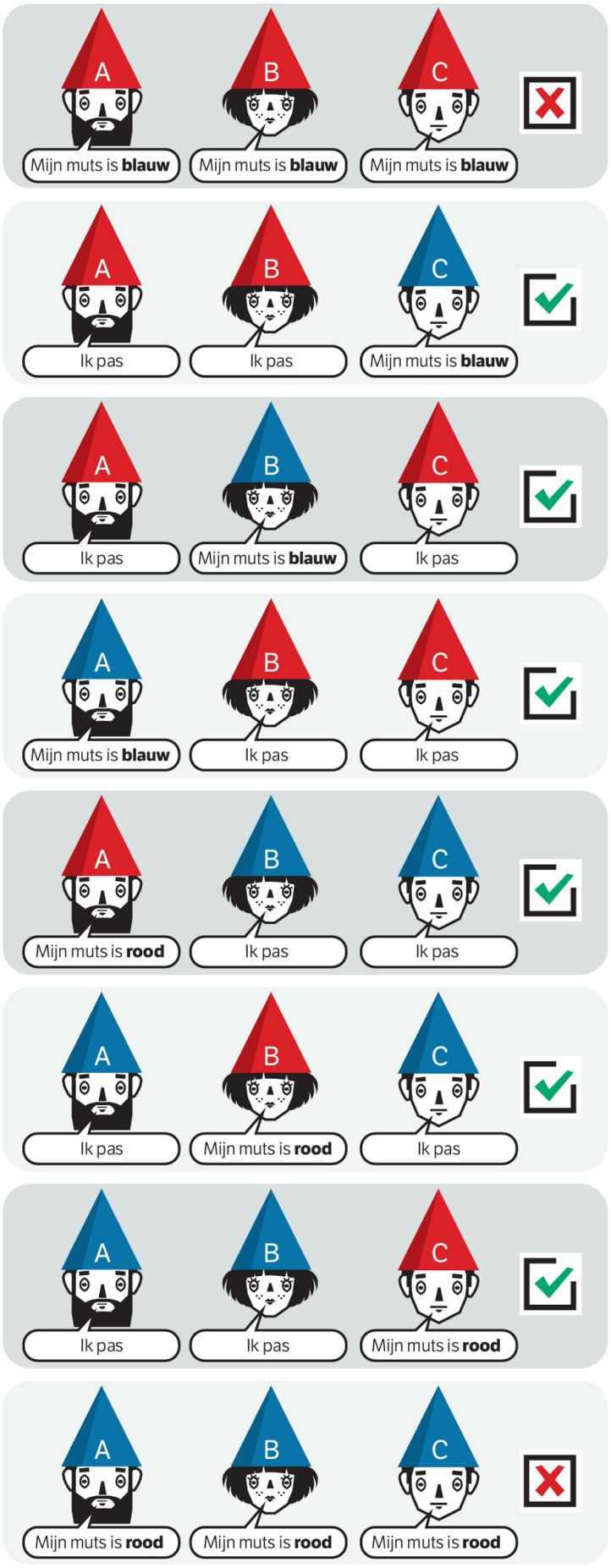

Figure 2.1: Alex van den Brandhof. Kabouters met een mutsprobleem, [vdB17]

Chapter 3

Related Work

The works down below contributed to the research of this thesis, in many different ways. Some of them helped to form the idea of this thesis. Some provide prove of reachable miles-stones. Other provide a way of thinking or a method of approach. In this section there will be an explanation of the information retrieved from other works.

Applications of Recursive Operators to Randomness and Complexity This is the thesis of Todd Ebert, he created the first step in describing this problem. Ebert's version is a simple version of the Leprechaun-Hatting problem. Where the players need to guess at the same time answer, but not required to guess. Just like the Leprechaun-Hatting the players can guess correctly, and the players win whenever one or more player guesses and all of those who guess do so correctly. Here Ebert wanted to research how to maximize their winning chance.

Ebert found a strategy for his version that uses Hamming codes. In normal context these are used for detecting an correcting errors in data transmission. He found out that with these hamming code similar strategies for a number of agents equal to 2^k1 a win rate of $k/k+1$ could be achieved. This contributed much to the development of the theorems in this thesis, where the prove of how this win rate is given(Theorem 4).

Mathematical Puzzles: A Connoisseurs Collection This is written by Winkler as part of a series of puzzle collection. He originally created this problem as the Leprechaun-Hatting problem. This is the original first occurrence of the problem. It is here that the rules and the game-flow is described. This is the foundation of this thesis, he describes the problem. Furthermore he describes some of the highest performing deals. Here he also describes the definition of a perfect deal, this definition is the foundation of the second method Chains.

Aha! A Two Volume Collection: Aha! Gotcha Aha! Insight. Here Martin Gardner wrote about a similar problem: *Three Prisoners Problem*. Which gives insight in the more probability side of the problem. Which states for the *Three Prisoners Problem* as follow. There are 3 prisoners, A, B and C, each in separate cells and condemned to death. One will at random be pardoned. The warden knows which one will be pardoned, however he will not tell the prisoners it. Prisoner A ask the warden to give him the other prisoners name who will be executed. The strategy sounds like: "When B is pardoned, give me C's name. And vice verse. However if I get the pardon, flip a coin."

This is another example of the Hatting problem where the solution was to be found in how the extra information is passed from one prisoners to another. This was key for the foundation of the chains. Because of the chains use the information and rules that restrict certain chains and strategies to work together.

The hat problem and some variations This is a paper about the Hatting Problem where they define how effective a strategy is, based on a set of winning rounds instead of a percentage. Furthermore they discussed more variations in the world parameter then this thesis will be exploring.

Chapter 4

Chains

In this chapter of the thesis we will explain the suggested Chain-Method. It will discuss the reason why the Chain Method is introduced, and will prove that the Chain Method generates the expected solutions. The Chain Method and the work flow of this method will be explained step by step. We will explain what a Chain is and why we should use it in order to get the maximal solutions. Finally this chapter gives a method of how to obtain the maximal solutions and proves that these are the maximal solutions.

4.1 Problem of the Brute Force Method

The current problem of the brute force method, as shown in Section 2.1.9, is that it uses an exhaustive search without any form of pruning. This makes it very expensive and time-consuming. However, for the Leprechaun Hatting problem, only the best and most covering solutions are of interest. As stated in Theorem 1 a solution with a set of color combinations is maximal if there is no solution with the same color combination and one or more other combinations represented. Therefore this thesis proposes a method that exploits the game rules to find these maximal solutions.

4.2 Definitions

To support the Chain Method step by step the maximal solutions are introduced. The first basic component in the research on minimizing the search for optimal solutions is the Chain. This component is maximal because it describes coherence between rounds. A Chain does however not form a solution, a set of chains does.

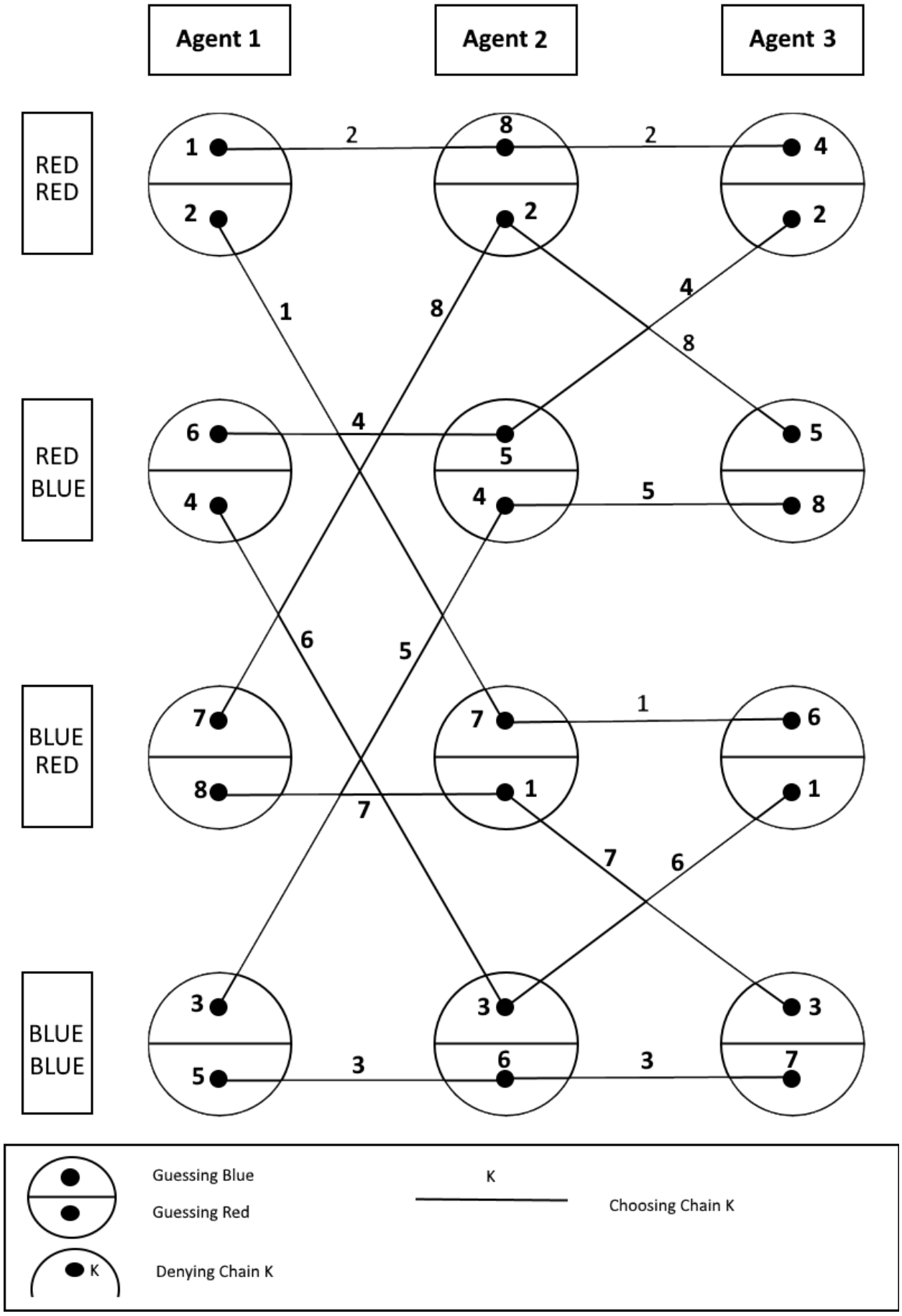

Figure 4.1: Bounding Graph for three agents (N) and two colors (M).

4.2.1 Chains

We start with an example, in Figure 4.1 the chains are represented as the lines. To interpret this figure we take the standard world where the number of agents is three and the number of colors is two. Suppose that the first agent will answer Blue when the agent sees (Red Red). Then this makes the round (Red Red Red) incorrect, and the round (Blue Red Red) correct. Then this Chain forces to do the same for the second agent. This means that the second agent will also answer Blue when the agent sees (Red Red). This however makes the round (Red Blue Red) correct, but the round (Red Red Red) will be twice incorrect. This repeats for the third agent and this agent will do the round (Red Red Blue) correct, while doing (Red Red Red) still incorrect. This represent the Chain two in Figure 4.1.

A Chain is defined in the standard world as a triple (X, Y, Z) with X,Y,Z $\in$ {Red, Blue}. This triple also represents one round, which the agents using this Chain will answer all three simultaneously incorrect. This ensures that in a chain no extra mistakes are introduced. When we investigate the single agents individually we can say that when agent 1 sees (Y, Z) then he will answer with $\bar{X}$. Here $\bar{X}$ is one of the possible colors, {Red, Blue} but is not equal to X. Therefore agent 1 will make the triple ($\bar{X}$, Y, Z) correct. The same applies to agent two and three, making the triples (X, $\bar{Y}$, Z) and (X, Y, $\bar{Z}$) correct.

For the standard world with N agents and $M = 2$ colors we can describe the Chain as defined by a N-Tuple (X_1, X_2, ... , X_N). Where X_1 , ... , $X_N \in$ { Red , Blue }. And for each agent applies that agent i answers with $\bar{X}_i$ when he sees (X_1, ..., $\hat{X}_i$, ... , X_N). Where $\hat{X}_i$ means that the i-th element of the all the agents will be removed, in order to create the set that the i-th agent sees.

We can extend the definition to a concatenation of other definitions defined in Chapter 2. A Chain is a combination of logical rules for the individual agents. The Chain is a representation of team behavior, where the team all guess incorrect for the same combination. This ensures that the maximum properties are used. These properties are:

- Whenever one agent makes a wrong guess all other agents make the same mistake. This also ensures that each agent makes one round correct and one round incorrect. And the agents share this incorrect round.
- A Chain has for each agent a guess for one combination it sees. For example in Figure 4.1 we see Chain 1. This does for agent 1 when it sees Red and Red it answers with Red, for agent 2 when it sees Blue and Red it answers with Blue and for agent 3 when it sees Blue and Red it answers with Blue. This also seen in 4.1 where it marks the line with a 1, the choices made by Chain 1 is visible.
- In the standard world where the number of colors M is defined as two. A Chain here has only two options it is either red for a specific logic rule or it answers blue for the rule. This also means that the passing rule will not occur as a possibility in a Chain.

With this definition we can find all the chains for the standard world, as displayed in Figure 4.1. The number of the chains is equal to the number of the rounds, because each Chain represents one of those rounds. In total there are eight rounds and therefore also eight chains. In the following Table 4.1 all the chains are represented by a number, the set of correct rounds and the incorrect round.

Chain	Correct	Incorrect
1	RRR BBR BRB	BRR
2	BRR RBR RRB	RRR
3	RBB BRB BBR	BBB
4	BRB RBB RRR	RRB
5	BBB RRB RBR	RBB
6	RRB BBB BRR	BRB
7	RBR BRR BBB	BBR
8	BBR RRR RBB	RBR

Table 4.1: Chain Correctness

This Table shows that there is an overlap between the eight chains. Some chains for example will not work together while others have no overlap. Many Chains have a common correct, for example Chain one and Chain four, they have both (Red Red Red) correct. It also occurs that one Chain makes a round correct and another Chain makes it incorrect. For example Chain one and Chain two, where Chain one makes (Blue Red Red) incorrect, while Chain two tries to make (Blue Red Red) correct.

This however doesn't necessarily mean that the combination of chains one and two does not make sense exist. The only consequence this has is, that on the point of Chain Crossing none of the two chains are correct, because the other two agents make those rounds already incorrect.

4.2.2 Chain Crossing

Chain one and Chain two are overlapping in agent one when agent one sees (Red Red). This means that Chain one does (Red Red Red) correct while Chain two tries to do (Red Red Red) incorrect. However this also works vice verse, because Chain one does (Blue Red Red) incorrect and Chain two is doing (Blue Red Red) correct. We call this property of two chains the Chain-crossing property. The definition of this property can be written as that you have a Chain X where the Chain is defined by the triple in the case of the standard world. Each chain then crosses at N locations, because there are N other chains that make the rounds this chains makes correct incorrectly.

4.3 Theorems

Now that we have created some definitions to support the Chain Method, we now give theorems which use the definition of the Chain to create insights. The theorems mainly focus on the relations between the chains for the general case.

Theorem 4 As stated earlier in a standard world where the number of agents N and the number of color M, the number of chains is equal to the number of rounds and therefore equal to the number of answers for each logical rule. Number of Chains = M * Logic Rules. Each of the chains has N color combinations correct. With the knowledge that each Chain is unique and there are exactly the same amount of rounds and also color combinations, there is a unique Chain which represents one incorrect color combination. Another property the Chain has is that each Chain does three good which are also represented by three other chains. These chains cannot be done correctly anymore or the first Chain picked will be incorrect, which is exactly the Chain Crossing property of Definition 4.2.2.

Theorem 5. We can visualize the dependencies and the property that each Chain has N other chains that make each other incorrect. In Figure 4.2 the dependencies are shown. When connected by a line it means that there is a Chain crossing at some point. As we can see there are many distinct chains, only a few combinations however create a maximum strategy.

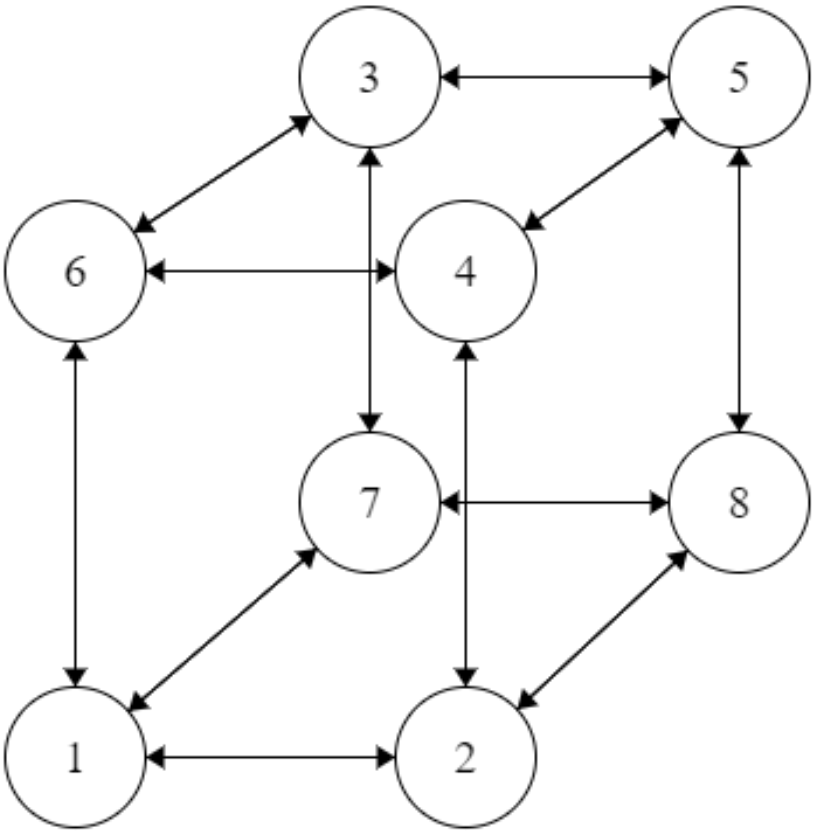

Figure 4.2: Visualization Chain Crossing Standard World

Theorem 6. Provide with Theorem 4 each Chain has N color combinations right and one wrong. We know that each game in this world has M^N rounds that a strategy can answer correct or incorrect. The maximal correct answers that can be given by selecting non crossing/depending chains is equal to $C * N$ where C is determined by $C = RoundedDown(M^N/(N+1))$. Where $N+1$ is the total amount of guesses for each Chain. C is rounded down because when $M^N/(N+1)$ is not a round number no other Chain can be chosen without doing it once more wrong.

Example A world with $N = 4$ and $M = 2$. Then C is at least three, because of $N = 4$ each Chain influences 5 answers meaning that the total chains that can be taken are the number of rounds (which is 16) divide by 5 is rounded down 3. And with 3 chains maximal 12 good answers and 3 wrong answers. Which leaves one answer not covered. And there is no Chain covering just one answer, without doing one wrong. And because there are no chains with the same wrong answer it has to be one that is already answered correctly or it answers the one remaining incorrectly. Which gives a situation where you cannot get gain anymore by adding another Chain.

Theorem 7. In the standard World the "Passing" rule is as follow, it is nor wrong nor right. Only when a other leprechaun has a correct answer the leprechaun have made a correct round. When a other leprechaun doesn't make a correct answer the round is considered as incorrect. Thus is passing within the standard world not limiting a strategy in creating. There is also no Chain where a passing answer is represented.

Theorem 8. Theorem 6 states that each Chain has influence on $N+1$ and C is the number of chains maximal can have influence. This is as well the minimum to create a maximal strategy. Because with more than C chains there will go at least one more round incorrect, as well as the current correct round may become corrupted and will result in a less containing set of correct answers. When using less than C chains, for example $C-1$. Then the strategy is compared to a strategy with C chains N less good answer are possible against one less fault.

Chapter 5

Chain-Method Procedure

In this chapter we will discuss the procedure we have developed with the support of the information that we have been gathering about this problem. With the help of the concepts and theories discussed in Chapter 2 en Chapter 4 we can prove that our method meets requirements to find the maximum Deals. In addition, in this chapter we will also discuss the results for different world standards, starting with the standard case. The standard case is for three agents and two colors, for this case we will first explain the method and show that it will find the maximum Deals as described by Guo et al [GKRT06]. This will provide a basis for solving of this problem on a larger scale, we will only Deal with different agent values.

5.1 Three Agents Standard

The first step towards proving the correctness of our method is to prove that the maximum Deals can be achieved in our theoretical way of the procedure. Later on we will also show that the Brute Force Method, in a less efficient way, will find the same Deals. To prove that we are actually going to find all Deals, we first have to determine the maximum Deals. We use the results found by Guo et al. Which are described in the paper [GKRT06]. This describes 12 Deals that are found to be maximum. These Deals are shown in Table 5.1.

5.1.1 Method

The method as mentioned earlier uses two properties. Here, in our figures, we use circles and crosses. Here, a circle means that this Chain's defining round is marked as correct and a cross marks the Chain's defining round as incorrect. The first property is that a circle is always connected to at least one cross, in order to have a valid Deal. This ensures that next to a good one is always a wrong one, because nothing can be done well without at least one mistake being made, as described in earlier chapters. In this case, if an agent formulates an answer, then it ensures that he immediately makes another combination wrong.

Strategy	Agent 1	Agent 2	Agent 3	Strategy	Agent 1	Agent 2	Agent 3
1	RR = B	RR = B	RR = B	7	RR = P	RR = R	RR = P
	RB = P	RB = P	RB = P		RB = P	RB = R	RB = P
	BR = P	BR = P	BR = P		BR = P	BR = R	BR = P
	BB = R	BB = R	BB = R		BB = P	BB = R	BB = P
2	RR = R	RR = P	RR = P	8	RR = P	RR = B	RR = P
	RB = P	RB = R	RB = R		RB = P	RB = B	RB = P
	BR = P	BR = B	BR = B		BR = P	BR = B	BR = P
	BB = B	BB = P	BB = P		BB = P	BB = B	BB = P
3	RR = P	RR = R	RR = P	9	RR = P	RR = P	RR = R
	RB = R	RB = P	RB = B		RB = P	RB = P	RB = R
	BR = B	BR = P	BR = R		BR = P	BR = P	BR = R
	BB = P	BB = B	BB = P		BB = P	BB = P	BB = R
4	RR = P	RR = P	RR = R	10	RR = P	RR = P	RR = R
	RB = B	RB = B	RB = P		RB = P	RB = P	RB = R
	BR = R	BR = R	BR = P		BR = P	BR = P	BR = R
	BB = P	BB = P	BB = B		BB = P	BB = P	BB = R
5	RR = R	RR = P	RR = P	11	RR = R	RR = R	RR = R
	RB = R	RB = P	RB = P		RB = B	RB = B	RB = B
	BR = R	BR = P	BR = P		BR = B	BR = B	BR = B
	BB = R	BB = P	BB = P		BB = R	BB = R	BB = R
6	RR = B	RR = P	RR = P	12	RR = B	RR = B	RR = B
	RB = B	RB = P	RB = P		RB = R	RB = R	RB = R
	BR = B	BR = P	BR = P		BR = R	BR = R	BR = R
	BB = B	BB = P	BB = P		BB = B	BB = B	BB = B

Figure 5.1: Maximal Deals

The second property for finding maximum Deals is that, there is always a circle next to each cross. This is with regards to the maximum Deals, namely if there is no circle next to a cross, it means that a cross only connects to other crosses, this means that the current situation is not maximal because a by chain defined round is answered incorrectly where there is no gain of this chain. Therefore in our search for the maximal Deals this cross should be changed into a circle. With these two properties, we can prove that the twelve Deals mentioned by Guo et al [GKRT06] are indeed exactly the maximum Deals.

To summarize the properties which we will use is that:

- At least one of the neighbours of a cross is a circle. To make this combination a maximal combination.
- At least one of the neighbours of a circle is a cross. To make this combination a valid combination.

The proof begins with assuming that our initial situation is as in Figure 5.2. In particular, we look at (1) as a candidate, as in Figure 5.2. From symmetry considerations it is sufficient to look at only this situation.

If we look at (1) as candidate, we see that in the case that (1) would be marked as incorrect, we can do all other points correct, without harming any of the two properties. This can be seen in Figure 5.3. This type of composition is representative of the Deals 1 to 4 described in [GKRT06].

Because of this we may now assume that (1) is done correct. However, this means that one of the adjacent points of (1) has to be done incorrect. That is why we now look at (2) and (3). The situation that follows then looks like in Figure 5.4.

Suppose that now (2) is chosen as good. That means that only one solution is possible, as shown in Figure 5.5. This is because in addition to (2), at the top left of the figure, a cross must be placed. This cross must be set in order to meet the two properties. This arrangement ensures that (3) also has to be marked as correct as well, otherwise the Deal that is created this way is already been treated by the Deals 1 to 4. With the same argument as for the top left, the right-bottom point must be marked as correct now to also satisfy the two properties. The situation shown in Figure 5.5 represented by this is representative of the Deals 5 through 10 as described in [GKRT06].

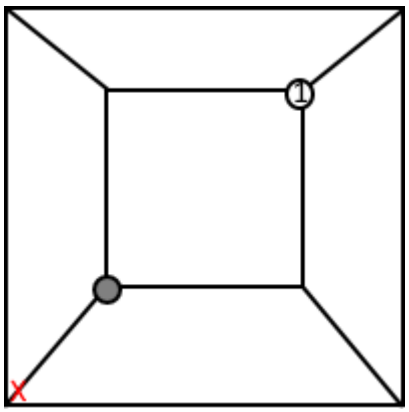

Figure 5.2: Random Chain selection

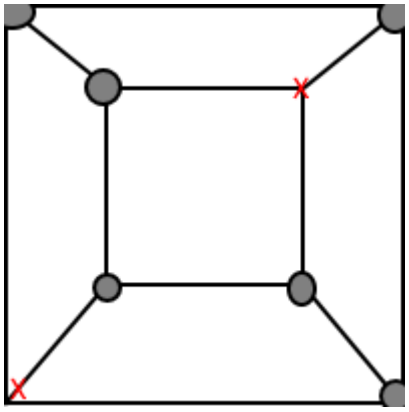

Figure 5.3: Deals 1 - 4

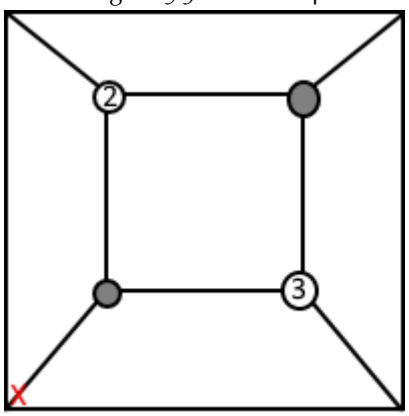

Figure 5.4

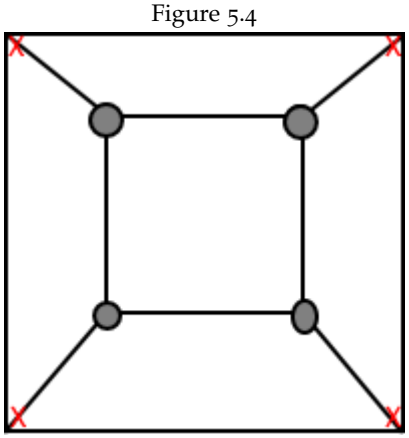

Figure 5.5: Deals 5 - 10

Now we have found that the only possibility left is that both (2) and (3) are marked as incorrect, as shown in Figure 5.6. This is because the above reasoning also applies if (3) is done correct first. In combination with the second property, to ensure that the Deals are maximal, both the top left corner and bottom right corner are marked as correct. This leads to the fact that top right corner must be marked as incorrect in order to satisfy the first property, namely that there must be a cross next to each circle.

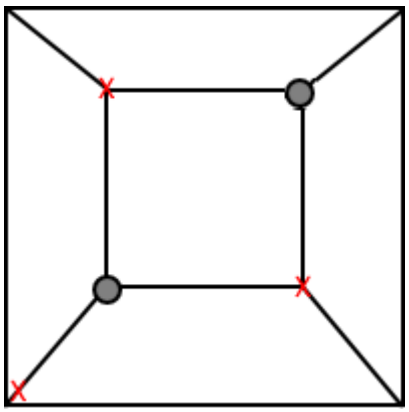

Figure 5.6

The situation that is then remaining is shown in Figure 5.7. This situation is representative for the Deals 11 and 12. Because we developed the situation individually and tried for each reasonable possibility we can conclude that there are no other options to investigate furthermore. We have also proven that the twelve Deals found in [GKRT06] are maximal and that this method can find these twelve Deals, so we can state that our method guarantees to find the maximum Deals for this problem.

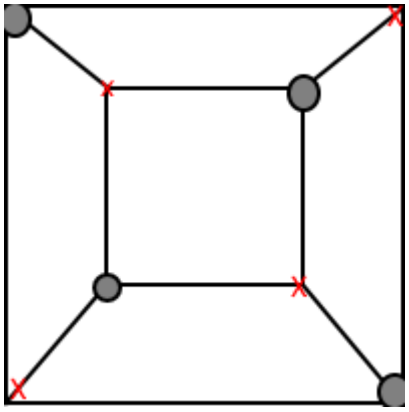

Figure 5.7: Deals 11 & 12

5.1.2 Program

The program develops the maximal Deals in a couple of steps. The first step the program takes is to generate the individual Chains. This is simply done by creating a Chain while looking at all rounds, because each Chain has a unique incorrect round. When defining each Chain with an incorrect round the program can immediately assign the correct rounds which define the complete Chain. After the Chains are initialized the program is ready to find the maximal Deals. These maximal Deals consist of combinations of Chains. To find these combinations the program uses each Chain as one of the first candidates of a begin for a combination of Chains.

The program assumes that these combination of Chains are an ordered set of Chains when developing the next Chains. This because a combination of Chains A and B is the same as the combination B and A. The next step to determine if the current combination satisfies to be a complete Chain. If this is not the case we define the current combination as an incomplete combination. We define a combination as complete whenever a combination says something about all possible rounds. The program handles this through computing the number of treated rounds, which is the sum of correct and incorrect rounds done by the combination. This number tells us how many of the rounds are treated and how many rounds are going incorrect because no agent is answering anything for that round.

If this number is not equal to all the rounds then it means that this combination is not maximal, when the program concludes that a combination is not maximal and is still viable then the program will try to add Chains with a higher rank one by to the combination to find new maximal combination. The program determines the viability by if the last added Chain changes the set of correct rounds. This ensures that we do not end up with a combination where we have a Chain which does not make a single round correct. We want this in our combination because this is one of the properties that ensures our combination could be maximal.

Next the program finds new combination that satisfies to be maximal combination after adding Chains to incomplete combinations, which is done in a breadth first way, is to filter out non maximal combinations which we cannot filter out by creating combinations. These combinations are not incomplete as there are still valid, however some combinations deliver a set of correct and incorrect rounds is when adding not a problem or making the combination invalid but there is a combination with a subset of Chains already giving a maximal combinations. For example when the combination of Chain 1 and Chain 8 gives a maximal combination, the combination of Chain 1 and Chain 2 is also valid but incomplete. Therefor we add Chain 8 to the combination making it a valid and possible maximal combination. However both the Chain 1 and 8 combination as the combination of Chain 1, Chain 2 and Chain 8 are chosen possible maximal combinations. The program removes these duplicates through that the program works in a breadth first way, to find these maximal combinations with a minimal number of Chains first. Subsequently thereto the program the checks when adding new possible maximal combination if that combination is not already found with fewer Chains.

Example of Incomplete Combination

Suppose we play with four agents, and we have already chosen two Chains, namely Chain A and Chain B. We now assume that Chain A does the round {Red Red Red Red} incorrect. Because of this we can say that A ensures that the rounds of {Blue Red Red Red}, {Red Blue Red Red }, {Red Red Blue Red} and {Red Red Red Blue} will go correctly. This means that if you convert this combination into a strategy for all agents, the strategy per agent will look like in Table 5.1. Suppose that Chain B does the round {Blue Blue Blue Blue} incorrect, then we can say that the rounds of {Red Blue Blue Blue}, {Blue Red Blue Blue}, {Blue Blue Red Blue} and {Blue Blue Blue Red} will go correctly. This means that if we have a combination of Chain A and Chain B that we say something about the rounds {Red Red Red Red}, {Blue Red Red Red}, {Red Blue Red Red}, {Red Red Blue Red}, {Red Red Red Blue}, {Blue Blue Blue Blue}, {Red Blue Blue Blue}, {Blue Red Blue Blue}, {Blue Blue Red Blue} and {Blue Blue Blue Red}, we have treated these. However, there are still a number of rounds that we have not mentioned yet with this combination: the rounds {Red Red Blue Blue}, {Red Blue Red Blue}, {Blue Red Red Blue}, {Blue Red Blue Red} and {Blue Blue Red Red}. These are all not treated and with the current combination are all answered incorrectly, because for these rounds all agents do not answer anything. As described in Table 5.1.

Because this error is done without anything opposed to it, this combination is not maximal. Therefore, the program will continue to develop by adding other Chains to this combination until all rounds have been dealt with.

	Agent 1	Agent 2	Agent 3	Agent 4
Chain A	BPPPPPPP	BPPPPPPP	BPPPPPPP	BPPPPPPP
Chain B	PPPPPPPR	PPPPPPPR	PPPPPPPR	PPPPPPPR
Chain C	RPPPPPPP	PPPPPPBP	PPPPBPPP	PPBPPPPP
Chain AB	BPPPPPPR	BPPPPPPR	BPPPPPPR	BPPPPPPR
Chain AC	PPPPPPPP	BPPPPPBP	BPPPBPPP	BPBPPPPPP

Table 5.1: Chain Conversion

Example of New Combination

Suppose we play with four agents and we have already found a combination that is maximum, namely a combination of Chains A, B, C and D. Now we are working on a new combination: here we have already chosen the Chains A, DEB and G in our combination. There is still no question of an incomplete combination, so the program will try to add another Chain. Now the program tries to add C. This yields a complete combination, however, this combination is not maximum and is not seen as maximum by the program because the first combination had already chosen the same Chains, namely A, B, C and D.

5.1.3 Results

To prove that the program works the same as the proof we run the program for both the brute force method and the Chain-Method implementations. For both the method data is gathered, for both the methods both the Deals tested are gathered. It reports also about a real world measure to see improvements, the time it took to to process the complete execution of the method. In Table 5.2 the results for both the brute force method and the Chain-Method, in both cases the same maximal Deals have been found.

Deal	Agent 1	Agent 2	Agent 3
1	{BPPR}	{BPPR}	{BPPR}
2	{RPPB}	{PRBP}	{PRBP}
3	{PRBP}	{RPPB}	{PBRP}
4	{PBRP}	{PBRP}	{RPPB}
5	{RBBR}	{RBBR}	{RBBR}
6	{BRRB}	{BRRB}	{BRRB}
7	{PPPP}	{RRRR}	{PPPP}
8	{PPPP}	{BBBB}	{PPPP}
9	{PPPP}	{PPPP}	{RRRR}
10	{PPPP}	{PPPP}	{BBBB}
11	{RRRR}	{PPPP}	{PPPP}
12	{BBBB}	{PPPP}	{PPPP}

Table 5.2: Maximum Deals

Figure 5.8 shows the calculation time of the brute force method. This only includes the time executing the Deals. For this method no pre-processing is needed. All the Deals are made on the fly while calculating the Deals. This results in a very expensive and time consuming process because of all the possibilities needed to be checked. On average the time to calculate a single Deal that is tested is 1250 / 531441 = 2,3 ms. This is acceptable for a single Deal if not too many Deals need to be checked.

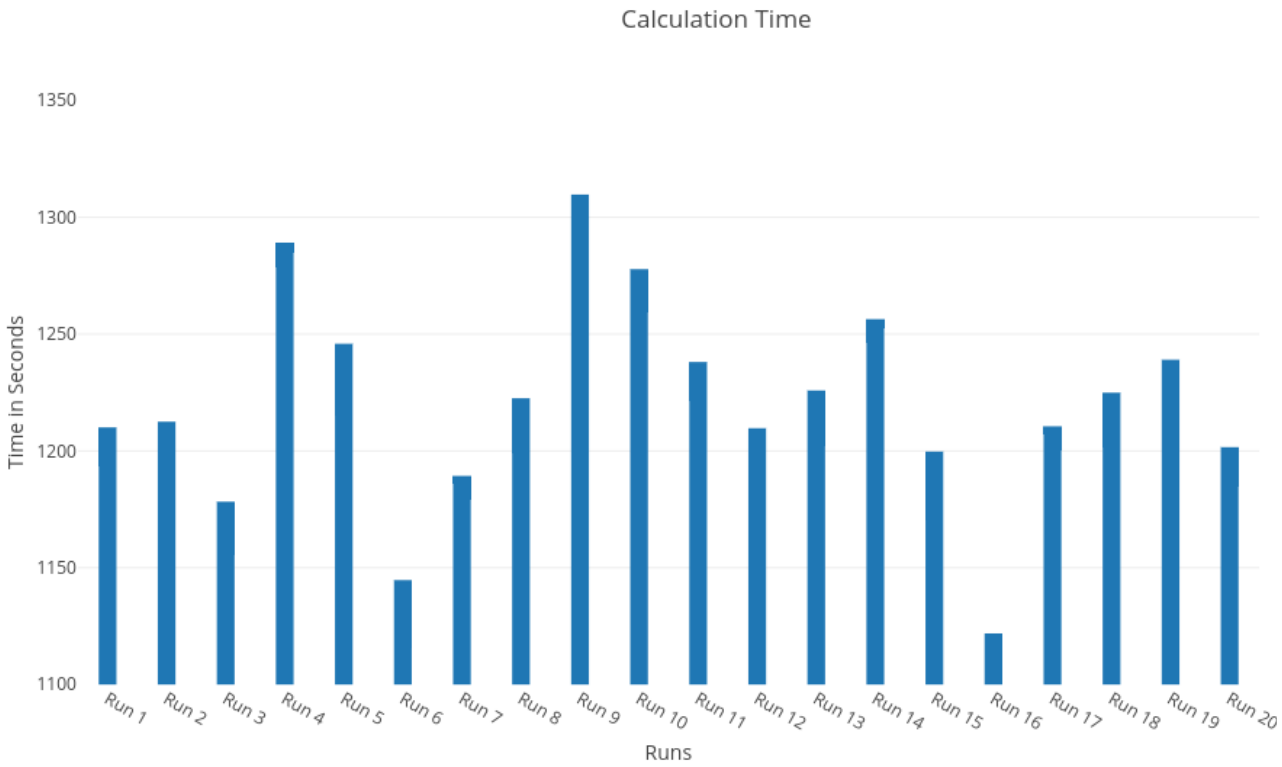

Figure 5.8: Calculation Time of Brute Force Method

The next Figure 5.9 shows the calculation time of the Chain-Method. This however also includes some pre-processing time to create the Chains, and to create the Chain combination creation. And it shows that because of the reduced number of Deals it is much faster to evaluate the necessary Deals. However the most time consuming part is the preparation which is on average 0.47 second. As shown in Table 5.2 there are only 12 Deals that needed to be tested. And with an average of 0.5 for the total number of seconds it makes the time for each Deal around 2.2ms which is similar to the brute force method.

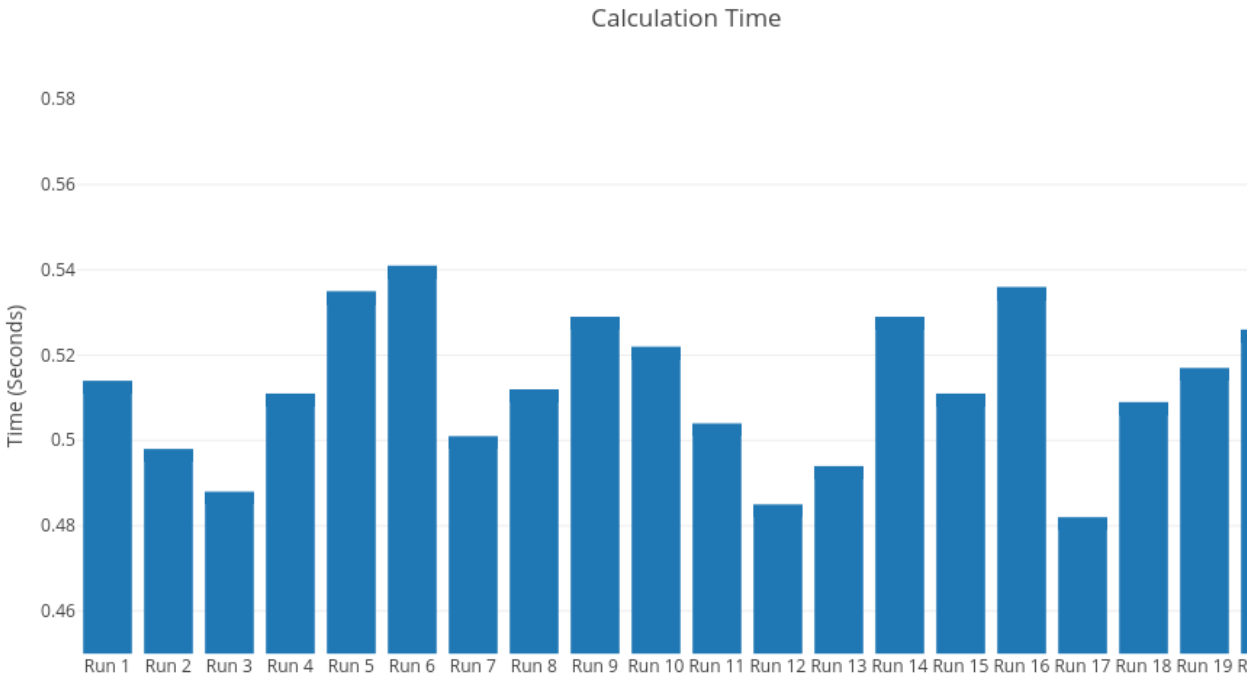

Figure 5.9: Calculation Time of Chain-Method

5.1.4 Conclusion

For this standard the results show that proven by hand, the Chain-Method works and gives the expected values. The computer program adds to this statement that it uses the Chain-Method and the brute force method and gives us the same results as the prove where the Chain-Method is fully explained and elaborated. However this only shows for and standard where the Chains align perfectly with each other. It requires a standard which is not with the number of agents a two power minus one. Other standard that could be used are four five and six. The results show that the Chain-Method is not only good and accurate for three agents, it is also faster. In the order of 2400 times faster, this because a specific pruning is applied which shows for this problem to be an improvement.

5.2 Four Agents Standard

In Chapter 5.1 we showed an algorithm for the Leprechaun problem with three agents to find the maximum Deals, through the Chain-Method. In addition, we have also shown that it is possible to have the Chain-Method implemented in a computer program using all the properties of the Chain-Method as described in Chapter 4. This chapter will further explain the results and special characteristics of the Chain-Method when applied to a World where there are four agents instead of three agents. We will use the implemented method to determine the maximum Deals for this standard with four agents. Here we look, among other things, at what sort of Deals are involved and how the method arrives at to these Deals. We also examine in depth how the method works for four agents. Further we discuss how this world is different and how this will result in the Deals.

5.2.1 Differences

The biggest difference is the difference in the number of agents N, where in the first standard only three agents play and in the second standard there are four agents. This is crucial because this affects the overall complexity of the game, but it will also show that the Chains Method reveals some imperfections of the Chains. With the information about the Chains and how they can be effectively applied in Theorem 4.3, we can already say a lot about how the maximum Deals are shaped for the four agents problem. As discussed in Theorem 4.3 we already proved that the number of Chains that must be chosen to make a valid maximum Deal depends on N, because it depends on if N can be divided by a power of two. This division is then rounded, because a Chain can not be taken in part. Because one of the characteristics of the Chains is that each Chain does one round incorrect, therefor choosing a Chains results always in doing one more error and perhaps one round better. As a result, the number of Chains will always be rounded to the next integer. We can even divide the equation in such a way that we always round up and overestimate the number of Chains unless we choose that $N+1$ is a power of the number of colors M. Then and only then do we have to Deal with an integer and do not have to round it. In our special case we always work with two colors, so $M = 2$, in order to come out well with the number of Chains $N+1$ must always be a power of two. Now if we look at the conclusions drawn in the paper by Guo et al. [GKRT06] we see that they also conclude here that for special cases where $N+1$ is a 2 power, improvements are found. This allows us to determine how the number of maximum correct rounds per number of agents is progressing, as shown in Table 5.3.

Agent Number	Maximal Number of correct rounds	Percentage
3	6	75%
4	12	75%
5	24	75%
6	48	75%
7	112	87.5%
15	30720	93.75%
31	2080374784	96.88%
63	9079256848778919936	98.44%

Table 5.3: Predicted winrate per number of Agents (N)

5.2.2 Results

Now that we have seen the expectations for the maximum correctness, we can expect that these percentages can also be seen in the results. Now we can make a statement about what kind of Deals we should expect to see, namely one of the conditions for maximum strategy was that an error in a chain is connected to a good state. This implies that a combination of Chains where these Chains do more than half wrong is not maximum strategy. So our expectation pattern for the Deals is that if they are maximized that their correct rounds percentage is between the minimum value of 50% and the maximum value shown in Table 5.3, then in the case of four agents, 75 %.

In the Tables Table 5.5, Table 5.6, Table 5.7 and Table 5.8 below we have shown the Deals for each correct rounds percentage. Table 5.5 shows all the Deals that have scored the highest, with a correct rounds percentage of 75%, this is for the standard with four agents, $N = 4$, this is the maximum achievable score. Table 5.6 and Table 5.7 shows the maximum Deals that score 62.5 % correct rounds and Table 5.8 shows the maximum Deals, with a correct rounds percentage of 50 %.

Each of these Deals are a representation of how the agents will behave for which combination of colors this agents sees. This is characterized by a set of colors that the agent does for all possible rounds, because the agent only has access to the hat colors of all other agents. This means that an agent can see $N - 1^M$ possible combinations. The possibilities that exist are shown in Table 5.4. For example, as shown in 5.5, Strategy three is that agent two is the next combination {BRPPPPRB }. This means that as agent two sees the hat colors {Red Red Red} it can be both the rounds {Red Red Red Red} and {Red Blue Red Red}. However in both cases agent will call two Blue, and will do one {Red Red Red Red} do wrong. This is however carefully selected and this round is done incorrectly by all agents. This is one of the properties of the Chains.

Seeing Combination	Color Combination
1	{ Red Red Red }
2	{ Red Red Blue }
3	{ Red Blue Red }
4	{ Red Blue Blue }
5	{ Blue Red Red }
6	{ Blue Red Blue }
7	{ Blue Blue Red }
8	{ Blue Blue Blue }

Table 5.4: Rounds seen by an agent

Deal Number	Agent 1	Agent 2	Agent 3	Agent 4
1	{PPPPPPPP}	{BBPPPPRR}	{BBPPPPRR}	{BBPPPPRR}
2	{BBPPPPRR}	{PPPPPPPP}	{BPBPPRPR}	{BPBPPRPR}
3	{BRPPPPRB}	{BRPPPPRB}	{BPPBPRRP}	{BPPBPRRP}
4	{BPBPPRPR}	{BPBPPRPR}	{PPPPPPPP}	{BPPRBPPR}
5	{BPRPPRPB}	{BPPBPRRP}	{BRPPPPRB}	{BPPRPBRP}
6	{BPPBPRRP}	{BPRPPRPB}	{BPRPPRPB}	{BPPRPRBP}
7	{BPPRBPPR}	{BPPRBPPR}	{BPPRBPPR}	{PPPPPPPP}
8	{BPPRRPPB}	{BPPRPBRP}	{BPPRPBRP}	{BRPPPPRB}
9	{BPPRPBRP}	{BPPRRPPB}	{BPPRPRBP}	{BPRPPRPB}
10	{BPPRPRBP}	{BPPRPRBP}	{BPPRRPPB}	{BPPRRPPB}
11	{RBPPPPBR}	{RBPPPPBR}	{PBBPRPPR}	{PBBPRPPR}
12	{RRPPPPBB}	{PPPPPPPP}	{PBPBRPRP}	{PBPBRPRP}
13	{RPBPPBPR}	{PBBPRPPR}	{RBPPPPBR}	{PBRPBPPR}
14	{RPRPPBPB}	{PBPBRPRP}	{PPPPPPPP}	{PBRPPBRP}
15	{RPPBBPPR}	{PBRPBPPR}	{PBRPBPPR}	{RBPPPPBR}
16	{RPPBRPPB}	{PBRPPBRP}	{PBRPPBRP}	{PPPPPPPP}
17	{RPPBPBRP}	{PBRPRPPB}	{PBRPPRBP}	{PBRPPRBP}
18	{RPPBPRBP}	{PBRPPRBP}	{PBRPRPPB}	{PBPRRPBP}
19	{RPPRPBBP}	{PBPRRPBP}	{PBPRRPBP}	{PBRPRPPB}
20	{PPPPPPPP}	{RRPPPPBB}	{PPBBRRPP}	{PPBBRRPP}
21	{PBBPRPPR}	{RPBPPBPR}	{RPBPPBPR}	{PRBPBPPR}
22	{PBRPBPPR}	{RPPBBPPR}	{PRBPBPPR}	{RPBPPBPR}
23	{PBRPRPPB}	{RPPBPBRP}	{PRBPPBRP}	{PRBPPBRP}
24	{PBRPPBRP}	{RPPBRPPB}	{PRBPPRBP}	{PPPPPPPP}
25	{PBRPPRBP}	{RPPBPRBP}	{PRBPRPPB}	{PPBRRBPP}
26	{PBPBRPRP}	{RPRPPBPB}	{PPPPPPPP}	{PRBPPRBP}
27	{PBPRRPBP}	{RPPRPBBP}	{PPBRRBPP}	{PRBPRPPB}
28	{PRBPBPPR}	{PRBPBPPR}	{RPPBBPPR}	{RPPBBPPR}
29	{PRBPRPPB}	{PRBPPBRP}	{RPPBPBRP}	{PRPBBPRP}
30	{PRBPPBRP}	{PRBPRPPB}	{RPPBPRBP}	{PPRBBRPP}
31	{PRBPPRBP}	{PRBPPRBP}	{RPPBRPPB}	{PPPPPPPP}
32	{PRRPBPPB}	{PRPBBPRP}	{PRPBBPRP}	{RPPBPBRP}
33	{PRPBBPRP}	{PRRPBPPB}	{PPRBBRPP}	{RPPBPRBP}
34	{PRPRBPBP}	{PRPRBPBP}	{PPPPPPPP}	{RPPBRPPB}
35	{PPPPPPPP}	{PPBBRRPP}	{RRPPPPBB}	{PPRRBBPP}
36	{PPBBRRPP}	{PPPPPPPP}	{RPRPPBPB}	{PRPRBPBP}
37	{PPBRRBPP}	{PPBRRBPP}	{RPPRPBBP}	{PRRPBPPB}
38	{PPRBBRPP}	{PPRBBRPP}	{PRRPBPPB}	{RPPRPBBP}
39	{PPRRBBPP}	{PPPPPPPP}	{PRPRBPBP}	{RPRPPBPB}
40	{PPPPPPPP}	{PPRRBBPP}	{PPRRBBPP}	{RRPPPPBB}

Table 5.5: Four agent, 75% correct Deals

Deal Number	Agent 1	Agent 2	Agent 3	Agent 4
41	{PBBPPPPR}	{PBBPPPPR}	{PBBPPPPR}	{BBBPBPPR}
42	{PBPPBPPR}	{PBPPBPPR}	{BBBPBPPR}	{PBBPPPPR}
43	{PRRPPPPB}	{BPPBPPRP}	{BPPBPPRP}	{BBPBPBRP}
44	{PRPPRPPB}	{BPPPPBRP}	{BBPBPBRP}	{BPPBPPRP}
45	{PPBPBPPR}	{BBBPBPPR}	{PBPPBPPR}	{PBPPBPPR}
46	{PPRPRPPB}	{BBPBPBRP}	{BPPPPBRP}	{BPPPPBRP}
47	{BPPBPPRP}	{PRRPPPPB}	{BPPBPRPP}	{BPBBPRBP}
48	{BPPPPBRP}	{PRPPRPPB}	{BPBBPRBP}	{BPPBPRPP}
49	{BBBPBPPR}	{PPBPBPPR}	{PPBPBPPR}	{PPBPBPPR}
50	{BBPBPBRP}	{PPRPRPPB}	{BPPPPRBP}	{BPPPPRBP}
51	{BRRPRPPB}	{BRPBPBRP}	{BRPBPBRP}	{BRPBPBRP}
52	{BRPBPBRP}	{BRRPRPPB}	{BPRBPRBP}	{BPRBPRBP}
53	{BPPBPRPP}	{BPPBPRPP}	{PRRPPPPB}	{BPPRBBBP}
54	{BPPPPRBP}	{BPBBPRBP}	{PRPPRPPB}	{BPPRPBPP}
55	{BPBBPRBP}	{BPPPPRBP}	{PPRPRPPB}	{BPPRPPBP}
56	{BPRBPRBP}	{BPRBPRBP}	{BRRPRPPB}	{BPPRRBBP}
57	{BPPRPBPP}	{BPPRPBPP}	{BPPRBBBP}	{PRRPPPPB}
58	{BPPRPPBP}	{BPPRBBBP}	{BPPRPBPP}	{PRPPRPPB}
59	{BPPRBBBP}	{BPPRPPBP}	{BPPRPPBP}	{PPRPRPPB}
60	{BPPRRBBP}	{BPPRRBBP}	{BPPRRBBP}	{BRRPRPPB}
61	{BPPRPRRB}	{BPPRPRRB}	{BPPRPRRB}	{BPPRPRRB}
62	{BPPRPRRR}	{BPPRPRPP}	{BPPRPRPP}	{BPPRPRPP}
63	{BPPRPRPP}	{BPPRPRRR}	{BPPRPPRP}	{BPPRPPRP}
64	{BPPRPPRP}	{BPPRPPRP}	{BPPRPRRR}	{BPPPPRRP}
65	{BPPPPRRP}	{BPPPPRRP}	{BPPPPRRP}	{BPPRPRRR}
66	{RPPRPPBP}	{RPPRPPBP}	{PBBPRPPP}	{PBBBRPPB}
67	{RPPPPRBP}	{RPPPPRBP}	{PBBBRPPB}	{PBBPRPPP}
68	{RBBPBPPR}	{RBBPBPPR}	{RBBPBPPR}	{RBBPBPPR}
69	{RBPRPRBP}	{RBPRPRBP}	{PBBRRPPB}	{PBBRRPPB}
70	{RRRPRPPB}	{PPPBPBRP}	{PPPBPBRP}	{PPPBPBRP}
71	{RRPRPRBP}	{PPPRPRBP}	{PBPPRPPB}	{PBPPRPPB}
72	{RPPRPBPP}	{PBBPRPPP}	{RPPRPPBP}	{PBRPBBPB}
73	{RPPPPBRP}	{PBBBRPPB}	{RPPPPRBP}	{PBRPBPPP}
74	{RPBRPBRP}	{PBBRRPPB}	{RBPRPRBP}	{PBRPBRPB}
75	{RPRRPBRP}	{PBPPRPPB}	{PPPRPRBP}	{PBRPPPPB}
76	{RPPBPRPP}	{PBRPBPPP}	{PBRPBBPB}	{RPPRPPBP}
77	{RPPBPPRP}	{PBRPBBPB}	{PBRPBPPP}	{RPPPPRBP}
78	{RPPBBRRP}	{PBRPBRPB}	{PBRPBRPB}	{RBPRPRBP}
79	{RPPBRRRP}	{PBRPPPPB}	{PBRPPPPB}	{PPPRPRBP}
80	{RPPBPBBB}	{PBRPRPPP}	{PBRPRPPP}	{PBRPRPPP}
81	{RPPBPBBR}	{PBRPRPBR}	{PBRPRPBR}	{PBRPRPBR}
82	{RPPBPBPP}	{PBRPRPRR}	{PBRPPPPR}	{PBRPPPPR}
83	{RPPBPPBP}	{PBRPPPPR}	{PBRPRPRR}	{PBPPRPPR}
84	{RPPPPBBP}	{PBPPRPPR}	{PBPPRPPR}	{PBRPRPRR}
85	{PPPBPBRP}	{RRRPRPPB}	{PPPBPRBP}	{PPPBPRBP}

Table 5.6: Four agent, 62,5% correct Deals

Deal Number	Agent 1	Agent 2	Agent 3	Agent 4
86	{PPPRPRBP}	{RRPRPRBP}	{PPBPRPPB}	{PPBPRPPB}
87	{PBBPRPPP}	{RPPRPBPP}	{RPPRPBPP}	{PRBPBPBB}
88	{PBBBRPPB}	{RPPPPBRP}	{RPPPPBRP}	{PRBPBPPP}
89	{PBBRRPPB}	{RPBRPBRP}	{RPBRPBRP}	{PRBPBPRB}
90	{PBRPBPPP}	{RPPBPRPP}	{PRBPBPBB}	{RPPRPBPP}
91	{PBRPBBPB}	{RPPBPPRP}	{PRBPBPPP}	{RPPPPBRP}
92	{PBRPBRPB}	{RPPBBRRP}	{PRBPBPRB}	{RPBRPBRP}
93	{PBRPRPPP}	{RPPBPBBB}	{PRBPRPPP}	{PRBPRPPP}
94	{PBRPRPBR}	{RPPBPBBR}	{PRBPRBPR}	{PRBPRBPR}
95	{PBRPRPRR}	{RPPBPBPP}	{PRBPPPPR}	{PRBPPPPR}
96	{PBRPPPPB}	{RPPBRRRP}	{PRBPPPPB}	{PPPRPBRP}
97	{PBRPPPPR}	{RPPBPPBP}	{PRBPRRPR}	{PPBPRPPR}
98	{PBPPRPPB}	{RPRRPBRP}	{PPPRPBRP}	{PRBPPPPB}
99	{PBPPRPPR}	{RPPPPBBP}	{PPBPRPPR}	{PRBPRRPR}
100	{PRBPBPPP}	{PRBPBPBB}	{RPPBPRPP}	{RPPBPRPP}
101	{PRBPBPBB}	{PRBPBPPP}	{RPPBPPRP}	{RPPBPPRP}
102	{PRBPBPRB}	{PRBPBPRB}	{RPPBBRRP}	{RPPBBRRP}
103	{PRBPRPPP}	{PRBPRPPP}	{RPPBPBBB}	{PRRPBPPP}
104	{PRBPRBPR}	{PRBPRBPR}	{RPPBPBBR}	{PRRBBPPR}
105	{PRBPRRPR}	{PRBPPPPR}	{RPPBPBPP}	{PRPPBPPR}
106	{PRBPPPPR}	{PRBPRRPR}	{RPPBPPBP}	{PPRPBPPR}
107	{PRBPPPPB}	{PRBPPPPB}	{RPPBRRRP}	{PPPBPRRP}
108	{PRRPBPPP}	{PRRPBPPP}	{PRRPBPPP}	{RPPBPBBB}
109	{PRRBBPPR}	{PRRBBPPR}	{PRRBBPPR}	{RPPBPBBR}
110	{PRRRBPPR}	{PRPPBPPR}	{PRPPBPPR}	{RPPBPBPP}
111	{PRPPBPPR}	{PRRRBPPR}	{PPRPBPPR}	{RPPBPPBP}
112	{PRPPBPPB}	{PRPPBPPB}	{PPPBPRRP}	{RPPBRRRP}
113	{PPPBPRBP}	{PPPBPRBP}	{RRRPRPPB}	{PPPRPBBP}
114	{PPPRPBRP}	{PPBPRPPB}	{RRPRPRBP}	{PPRPBPPB}
115	{PPBPRPPB}	{PPPRPBRP}	{RPRRPBRP}	{PRPPBPPB}
116	{PPBPRPPR}	{PPBPRPPR}	{RPPPPBBP}	{PRRRBPPR}
117	{PPRPBPPR}	{PPRPBPPR}	{PRRRBPPR}	{RPPPPBBP}
118	{PPRPBPPB}	{PPPBPRRP}	{PRPPBPPB}	{RPRRPBRP}
119	{PPPBPRRP}	{PPRPBPPB}	{PPRPBPPB}	{RRPRPRBP}
120	{PPPRPBBP}	{PPPRPBBP}	{PPPRPBBP}	{RRRPRPPB}

Table 5.7: Four agent, 62,5% correct Deals, (Continued)

Deal Number	Agent 1	Agent 2	Agent 3	Agent 4
121	{PPPPPPPP}	{PPPPPPPP}	{PPPPPPPP}	{BBBBBBBB}
122	{PPPPPPPP}	{PPPPPPPP}	{BBBBBBBB}	{PPPPPPPP}
123	{PPPPPPPP}	{BBBBBBBB}	{PPPPPPPP}	{PPPPPPPP}
124	{BBBBBBBB}	{PPPPPPPP}	{PPPPPPPP}	{PPPPPPPP}
125	{RRRRRRRR}	{PPPPPPPP}	{PPPPPPPP}	{PPPPPPPP}
126	{PPPPPPPP}	{RRRRRRRR}	{PPPPPPPP}	{PPPPPPPP}
127	{PPPPPPPP}	{PPPPPPPP}	{RRRRRRRR}	{PPPPPPPP}
128	{PPPPPPPP}	{PPPPPPPP}	{PPPPPPPP}	{RRRRRRRR}
129	{BRRBRBBR}	{BRRBRBBR}	{BRRBRBBR}	{BRRBRBBR}
130	{RBBRBRRB}	{RBBRBRRB}	{RBBRBRRB}	{RBBRBRRB}

Table 5.8: Four agent, 50% correct Deals

The results show the outcome of a C ++ program that we have written in order to see the results of the Chain-Method. This program returns all possible maximal combinations of Chains, with respect to the Chain-Method.

In the results we see a number of Deals that we expected to find, but we also find a number of unexpected Deals. For example, we find the Deals 121 to 128, for which it is in general clear that such a strategy is maximal because these Deals meet the minimum requirements of a set of combinations. Each of the chosen Chains in the Deals always do one round incorrect wrong and do in these combinations only one round correct. These Deals spans exactly a plane like in Figure 5.5. Each of these Deals can be defined as one of the agents always says one color, for example Blue. This means that B*** will go correct. This make it clear that in general such a strategy is maximal. The Deals 129 & 130 are also explainable by the three agents Deals, namely Figure 5.7 defines for the three agents another set of Deals with a 50% correct number of rounds. The same behavior happens in the four agents standard, this is represented by Deals 129 & 130.

We have reviewed the program 10 times for the case of four agents ($N = 4$) to determine how much faster the proposed method is. In addition, we also checked whether the results of the methods corresponded. This was the same for both methods and resulted in the combinations shown in Tables: Table 5.5, Table 5.6, Table 5.7 and Table 5.8.

	Shortest Run-time (millisec)	Longest Run-time (millisec)	Average Run-time (millisec)
Brute Force	1639236 (4 hour, 33 min)	1810419 (5 hour, 1 min)	1716388 (4 hour, 46 min)
Chain-Method	19622 (3 min)	38178 (6 min)	23816 (3 min)

Table 5.9: Calculation Time with Four Agents

5.2.3 Conclusion

In this chapter we have shown how the method works for the Leprechaun problem with four agents, and how the method Deals with this strange standard. We have also been able to demonstrate that there is no improvement in the number of maximum percentage rounds, and as discussed earlier, this is because in the case of four agents, there is not enough room between the chosen Chains. It is in fact the case that in the maximum Deals there can not be a set that contains Chains that are not overlapping at a point. We have also seen in this chapter that the Chain-Method is also faster for four agents than the brute force method. The Chain-Method is on average $\frac{1716388}{23816} \approx 72x$ faster than the brute force method, however, as discussed in the conclusion of the three agents, the method was 2400 times faster than the method here only a third of that acceleration.

5.3 Five Agents Standard

In the previous chapter we discussed the game situation with four agents. In this chapter we looked at whether the program can do the same for a game situation with five agents. However, this chapter will mostly discuss why it is not possible for us and even a computer to handle this problem. We will discuss in this chapter what the complexity of putting combinations of Chains together is, and why the program cannot compute it for this number of agents. This chapter also elaborates also on what we can expect to find in the results if the program would run.

5.3.1 Expectations

Our expectations for the outcome of the program are based on the sort of Deals found in the three and four agents standards. We see in both standards one kind of Deals returning. Because if we look at the method and the results that we have found in the four agents standard, we see that there are a Deals covering 50% of the rounds. These Deals are the strategy where a single agent is in control and always saying the same color regardless of what the agent sees. We see that this sort of Deals returns by both the three and four agents and when we look at the method in the three agents standard these Deals are all within the category of the the situation described in Figure 5.6. Futher more our expectation will be that all Deals will be just like within the four agent variant that the Deals will be between 50% of the rounds to a maximum of 75% correct of the rounds.

Examples of Deals that result in 50% of the rounds correct are:

Agent 1	Agent 2	Agent 3	Agent 4	Agent 5
{BBBBBBBBBBBBBBB}	{PPPPPPPPPPPPPPP}	{PPPPPPPPPPPPPPP}	{PPPPPPPPPPPPPPP}	{PPPPPPPPPPPPPPP}
{RRRRRRRRRRRRRRR}	{PPPPPPPPPPPPPPP}	{PPPPPPPPPPPPPPP}	{PPPPPPPPPPPPPPP}	{PPPPPPPPPPPPPPP}
{PPPPPPPPPPPPPPP}	{BBBBBBBBBBBBBBB}	{PPPPPPPPPPPPPPP}	{PPPPPPPPPPPPPPP}	{PPPPPPPPPPPPPPP}
{PPPPPPPPPPPPPPP}	{RRRRRRRRRRRRRRR}	{PPPPPPPPPPPPPPP}	{PPPPPPPPPPPPPPP}	{PPPPPPPPPPPPPPP}
{PPPPPPPPPPPPPPP}	{PPPPPPPPPPPPPPP}	{BBBBBBBBBBBBBBB}	{PPPPPPPPPPPPPPP}	{PPPPPPPPPPPPPPP}
{PPPPPPPPPPPPPPP}	{PPPPPPPPPPPPPPP}	{RRRRRRRRRRRRRRR}	{PPPPPPPPPPPPPPP}	{PPPPPPPPPPPPPPP}
{PPPPPPPPPPPPPPP}	{PPPPPPPPPPPPPPP}	{PPPPPPPPPPPPPPP}	{BBBBBBBBBBBBBBB}	{PPPPPPPPPPPPPPP}
{PPPPPPPPPPPPPPP}	{PPPPPPPPPPPPPPP}	{PPPPPPPPPPPPPPP}	{RRRRRRRRRRRRRRR}	{PPPPPPPPPPPPPPP}
{PPPPPPPPPPPPPPP}	{PPPPPPPPPPPPPPP}	{PPPPPPPPPPPPPPP}	{PPPPPPPPPPPPPPP}	{BBBBBBBBBBBBBBB}
{PPPPPPPPPPPPPPP}	{PPPPPPPPPPPPPPP}	{PPPPPPPPPPPPPPP}	{PPPPPPPPPPPPPPP}	{RRRRRRRRRRRRRRR}

Table 5.10: Some example of Deals

5.3.2 Problems

Unfortunately, it turned out that even with the reduction we have done by applying the method, the program can not calculate computers. We can explain this by looking at the method, because the method does in part the same thing as the brute force method does. The method also always develops all possible nodes, but there is a restriction on which nodes can still be used. With the brute force method this is unlimited, where the Chain-Method reduces this. However, in the case of $N = 5$, ie five agents, the number of nodes that must at least be developed is equal to eight. This is at least necessary if we assume that 75% is the maximum achievable score. Because of this, 25% of the 32 rounds must go wrong, which is equal to 8 rounds. And a

round goes wrong when one or more agents make a mistake. With the definition of a chain we can say that a chain equals an error, so we can say that unique Chains are needed to do 8 rounds. The minimum achievable score can be stated that 50% is the lower limit. This means that 16 rounds must go wrong and that 16 Chains have been developed. This ensures that our method with the knowledge of the maximum achievable score could start with the combinations with 8 Chains in order to find all maximum Deals. However, we do not apply this knowledge in advance to validate that no maximum Deals can be found before we have to combine all combinations into 16 Chains to find all maximum Deals. These are for the case of without the knowledge 6.0108039E + 8 number of possibilities to go.

5.4 Summary

In this chapter we have discussed various standards. We first looked at the standard with three agents. Here we looked at how the Chain-Method works for the three agents standard. In this we explained with examples which steps the method takes to find the maximum Deals. We then showed how a computer program obtained the same properties. We have this program run to get results, we have compared this with our expectations and a Brute Force Method. In this we saw a great acceleration of calculating the maximum Deals.

After we have been able to ensure that our method delivers the right maximum Deals for the standard with three agents, we have let the same program calculate the Deals for the standard with four agents. Here too we discussed what we expect and what similarities there are between the Deals found in the standard with three agents. Here too we have shown that the Chain-Method is faster than the Brute Force Method. However, the acceleration is less large in this standard than in the standard with three agents. This can be explained by the fact that the complexity of the Chain-Method exponentially increases depending on N.

Finally, in this chapter we discussed the results for the standard with five agents. We can conclude that even with the reduction we use through the Chain-Method it is still impossible to go through all possible Deals.

Chapter 6

Conclusions and Further Research

The main purpose of this thesis was to find the most efficient way to find the maximal deals. In order to establish a method finding these deals in the most efficient way we first discussed and made some ground theorems. We discussed the Chains and defined what the conditions are for a Chain to exist. We also discussed how a chain looks when formulated to a deal. We also discussed why we look for chains in order to find the maximal deals.

With these theorems we made some definitions about the combinations of chains that form the maximal deals. These theorems support the Chain-Method and the implementation of this method in the computer program.

To prove the correctness of the Chain-Method we looked at three different standard. The first standard was with three agents, we used this standard to prove that our method works. We did this by means of explaining the method step by step. Then we formed our definitions of this method to a program that helped us find the maximum deals. We compared those finding with findings of our Brute Force Method. When we compared these two methods we saw a great acceleration when calculating the maximum deals.

After we have been able to ensure that our method delivers the right maximum deals for the standard with three agents, we let the same program calculate the deals for the standard with four agents and five agents. Here we found for the four agent standard some similar results as in the three agents standard. However we also found that the program couldn't handle five agents yet as the reduction is not strong enough to compete with the exponential growth of deals. We made some predictions about which deals we expect to find in the set of maximal deals. We also made a predication about how the number of correct rounds will improve as the number of agents goes up.

6.1 Future Research

In this thesis we only elaborated on standards with different agents sizes. This only effected the number of total deals there where available for the Brute Force Method. It also increased the number of maximal deals. if we had deepened more on the subject of the number of colors, Chances-Based Standard or the Multiple Rounds Standard we would have seen that other factors would played a role in the effectiveness for our method. A good example for a future study would be to examine three colors ($M = 3$) instead of two colors. This would drastically change the chain method because choosing a chain would then no longer be one-sided, as is the case for standards with two colors ($M = 2$).

This shows that this thesis focused mostly on the direct effect of what the impact is on the performance of the Chain-Method when varying number of agents. We expect that the varying the number of colors and changing the core rules of the game have such an impact that the chain method must be altered tailor to the standard to find the maximum deals in an efficient way.

Another challenge would be to find out what the effect of the passing a round is in the game of Leprechaun-Hatting. This could be achieved to alter the rules and effect of passing. This would give some interesting insight in the effect of passing on the maximal deals that will be found.

Bibliography

[Ebe98] Todd Ebert. *Applications of Recursive Operators to Randomness and Complexity*. PhD thesis, University of California, Santa Barbara, 1998.

[Gar06] Martin Gardner. *Aha! A Two Volume Collection: Aha! Gotcha Aha! Insight*. Mathematical Association of America, 2006.

[GKRT06] Wenge Guo, Subramanyam Kasala, M Bhaskara Rao, and Brian Tucker. The hat problem and some variations. *Red*, 1:1–8, 2006.

[vdB17] Alex van den Brandhof. Kabouters met een mutsprobleem. 1 2017. NRC - 14.1.2017.

[vDK15] Hans van Ditmarsch and Barteld Kooi. *One hundred prisoners and a light bulb.* Cham: Springer/-Copernicus Books, 2015.

[Win03] Peter Winkler. *Mathematical Puzzles: A Connoisseur's Collection*. 2003.