University of Macedonia

Master of Science in

Applied Informatics : Computer Science and Technology

Design and Analysis of Slot Machine games and RTP Optimization using Variable Neighborhood Search (VNS)

Master Thesis

of

Pantelis-Arsenios Kamanas

Thessaloniki , February 2021

Design and Analysis of Slot Machine Games and RTP Optimization using Variable Neighborhood Search (VNS)

Pantelis-Arsenios Kamanas

Bachelor of Science in Applied Informatics, University of Macedonia, 2019

Master of Science

submitted for the partial fulfillment of its requirements

Master of Science in Applied Informatics

Supervisor
Angelos Sifaleras

It was approved by the three-member examination committee on 24/02/2021

Angelos Sifaleras	Nikolaos Samaras	Dimitris-Hristu Varsakelis
.....	.....	.....

Pantelis-Arsenios Kamanas

.....

Abstract

Slot machines, also known as fruit machines, are the most popular gambling games in casinos. They are either electronic or electro-mechanical devices that consist of a number of reels that spin independently and a screen with a number of rows. Each reel contains several symbols in specific arrangements and quantities. This work presents the proccess of design and analysis of complex slot machine games. Additionally, a Variable Neighborhood Search (VNS) approach is presented for solving the Return-To-Player (RTP) optimization problem. A large number of software companies in the gaming industry seeks to solve the RTP optimization problem, in order to develop modern virtual casino gambling machines. By using a VNS framework which guides two local search operators we show how to control the distribution of the symbols in the reels in order to achieve the desired RTP. In this manuscript, optimization refers only to base game, the core of slot machine games, and not in bonus games, since a bonus game is triggered once two, three or more specific symbols occur in the gaming monitor. Although, other researchers have tried to solve the RTP problem in the past, this is the first time that a VNS methodology is proposed for this problem in the literature with good computational results.

Keywords: Variable Neighborhood Search, Metaheuristics, VND, Game Design, RTP, Optimization and Slot Machine

CONTENTS

List of Figures

List of Tables

List of Algorithms

CHAPTER 1

Introduction

1.1 The history of slot machines

Figure **1.1.1**: Evolution of slot machines from 19th century

Charles August Fey, a Bavarian immigrant in the United States, was the inventor of the first slot machine, which can be traced back to 19th Century. The first machine, it is believed, was created between 1887-1895, because there is no concise detail regarding the date he created it. Charles Fey invented a machine that would allow automatic payouts, so he created a simple automatic mechanism with 3 spinning reels machine with 5 different symbols.

The exact symbols were Diamonds, Hearts, Diamonds, Spades, Horseshoes and a Liberty Bell.The name of this game was 'Liberty Bell', because the highest payout was awarded for 3-bell symbols. This slot machine gained huge popularity and it was copied by many slot machine manufacturers. In 1902, slot machines have been officially banned but Liberty Bell continued to be manufactured. Instead of the symbols that have been written before, another slot machine called 'Trade simulator, replaced them with fruit (Cherry, Melon) and prizes were

paid out in chewing gum and sweets of the corresponding flavor. The payment of food prizes was a commonly used technique to avoid laws against gambling in a number of states, and for this reason a number of gumball and other vending machines were regarded with mistrust by the courts.In 1907, Herbert Mills produced a slot machine called the 'Operator Bell'. It could be found, by 1908, in most tobacconists, bowling alleys, shops and salons. BAR symbol was introduced for the first time, at that time.

After 60 years of purely-mechanical slot machines, a new electro-mechanical machine was created for the first time. In 1964, Bally developed the first fully electromechanical machine called 'Money Honey'.It was electromechanical, because the game was still started by pulling the lever.Money Honey was the first slot machine with a bottomless hopper, which let it has a payout of up to 500 coins. In 1976 was developed the first true video slot in Kearney Mesa, California by the Las Vegas based company Fortune Coin. This slot machine used a modified 19" Sony Trinitron color receiver for the display and logic boards for all slot machine functions.In 1978, Fortune Coin was acquired by IGT.The first video slot machine with a 'second screen' bonus round was 'Reel Em'.It was developed and released by WMS Industried Inc. The bonus game was triggered and a completely different screen was displayed. During bonus game, additionally payouts could be won. Slots became increasingly popular at casinos and at this time with the rise of computer cultural anyone can play either at a case-bound machine in a bar or a physical casino, or at home, virtually, through an online casino in front of a computer. The 19th century was the 'Age of Hardware' and from 20th and after is the 'Age of Software'.

1.2 General description of a simple 5-reel slot machine game

A simple slot machine game, basically, consists of a number of reels that spin independently. Every reel contains several symbols in specific arrangements and quantities. These variables are based on the machine producer's design.Slot machines which are mechanical or electro-mechanical have cylinders as spinning reels which hold the symbols that are printed on a strip.For the electronic machines or computer slot games, virtual reels are being used.Each reel has a number of positions at which it can stop and this number is called stops and each stop has a symbol of that reel associated. The stops are being showed on the display of slot machine while they are spinning and when they stop, an (m×n) slot window is being created. The dimensions of the display depends on creators design. The display of the slot machine shows, also, game rules, coin values, line bet, payout schedule and more options as we can see below at the **Figure 1.2.2**. The player inserts cash or a paper ticket with a barcode in 'ticket-in, ticket-out' machine. When the player presses the button, if slot machine has the minimum amount of bet, the reels start spinning. After that, the reels produce the outcome of the game, which consists of all

symbols that remained visible through the window on the display.

The mechanical slot machines ensures the randomness of the outcome of the spin but in a virtual slot machine, a PRNG(pseudorandom number generator) is used. The player wins if a certain pre-defined combinations of symbols occur on one or more marked paylines. These are group of lines that usually have a geometrical configuration. They are collinear or broken, of various shapes and size for any game in part. Each winning combination is clearly defined and has a certain payout, noted in the displayed payout schedule. When a winning combination occurs on paylines, the player is paid the unit credit inserted multiplied by the payout rate of the combinations that occured on each payline.In subsection, below, will be explained every word that is related to slot machines.

The gambling industry is one of the most profitable industries globally. With more than US$ 450 billion dollars estimated value in 2020 and with a projected value of more than US$ 640 billion by 2027, the gambling market is definetely one of the largest and most profitable in the world (1). It is also worth mentioning that these figures are only related to the regulated markets, as the grey and black markets contribute with hundreds of billions of dollars to the total market value. There are hundreds of companies of different types of services that compose the gambling industry: from casinos and software developers to casino game developers and payment solutions. The growth of remote gaming, with online and mobile gambling market size is expected to grow by more than 10% by 2027, ensuring that the gambling industry will become even more profitable in the years to come. Hundreds of thousands of dollars are spent every year from several companies in order to participate in exhibitions around the world, with the most famous and attractive being held in Las Vegas, Paris, London, South Africa and Singapore. A list of the most well-known conferences and trade shows with a Gaming, Sports Betting, and Affiliate theme may be found online (2), e.g., Paris Games Week, iGB Live!, ICE Totally Gaming, etc.

1.3 Slots and mathematics

Ever since, the histories of mathematics and gambling have been intertwined. In the last decade, mathematics has been taken more and more seriously into account in gambling, as being the essence that governs the games of chance and the only rigorous tool providing information on optimal play, where possible. The mathematical models that are applied on slot machines, theoretically, belong to mathematical domains such as Combinatorics, Topology, Probability Theory and Statistics. Clever gamblers use mathematics to look for the smallest advantages, and casinos use sophisticated mathematical tools to devise new ways of drawing in players. A patent granted to the Norwegian scientist Inge Telnaes in 1984 reworked the gambling business.

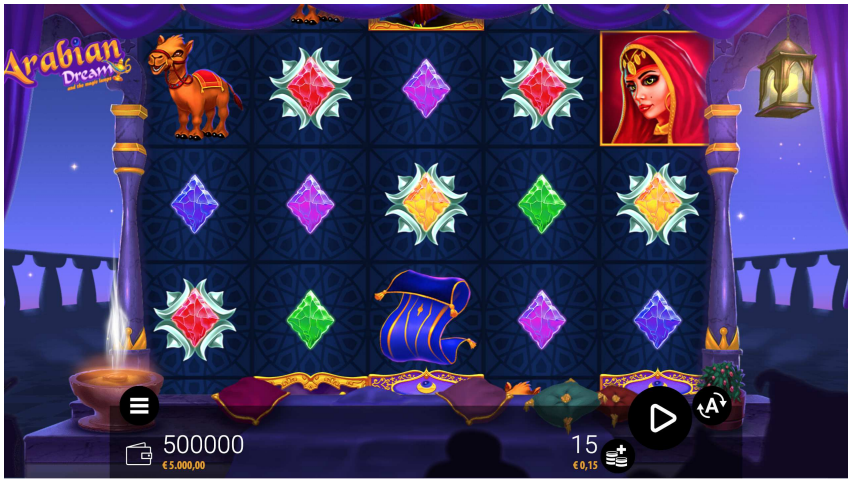


Figure **1.2.2**: A (5×3) slot machine called Arabian Dream

Prior to Telnaes' invention, slot machines were essentially mechanical devices.

Besides being difficult to tune and maintain, mechanical slot machines suffered from an essential problem. A machine with 3 reels, with 12 symbols on each, with one of those 12 symbols a cherry. The likelihood of getting three cherries, and winning the jackpot is 1 in 1728. If the casino wants to make money, the jackpot payout should be, say 1,700$ on a 1$ bet. If a fourth reel is added, a jackpot of about 20,000$ will occur. People do not like machines with more reels they intuitively, and rightfully, feel that extra reels diminish their chance of winning. Another possibility is to put more symbols on each reel. But the astronomical jackpots you see in casinos these days would then need actually monumental machines. Inge Telnaes proposed a simple solution: Let a random number generator a computer chip determine the combination of symbols that appear when the reels stop.In different words, use a chip to control where the reels stop on a spin, but create the illusion that the wheels stopped on their own. The history of gambling is also intertwined with that of a less reputable group tricksters and swindlers. In the long run, the only sure way to make money by gambling is to create the illusion that your opponent can win, while keeping the odds firmly on your side. That offers people who apprehend maths an awfully solid advantage.

1.3.1 The dictionary of slot machine games

We will explain below every key-word that is displayed on the game and every player should know. As an example game is presented 'Arabian Dream' a game designed and developed by ZeusPlay (3).

- **Paytable:** A table that will show what a payline can win you.

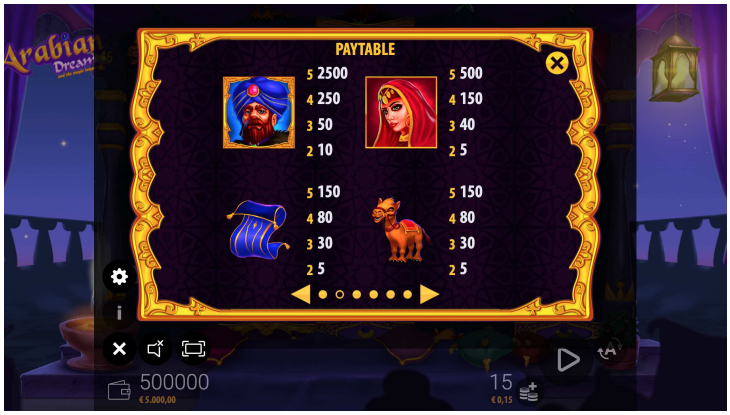


Figure **1.3.3**: The Paytable (Part 1)

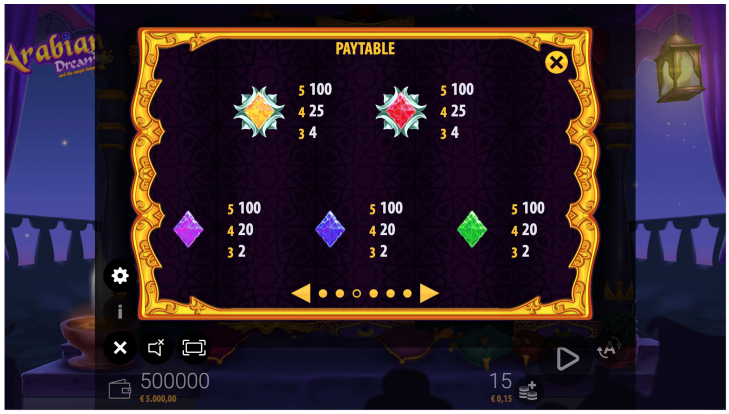


Figure **1.3.4**: The Paytable (Part 2)

- **Spin/Play:** Button or lever that spin the reels.
- **Reels:** A slot machine consists of them and they show the symbols and when the level or the button is pressed, they spin.
- **Coins:** The amount of money the player bets.
- **Coin Value/Bet:** The value of the coin that the player bets.
- **Line Bet:** The amount of money that you want to wager per payline.
- **Max Bet:** A game has 15 paylines and the maximum linebet is 0.01\$, so the max bet will be ($15 \times 0.01\$ = 0.15\$$).

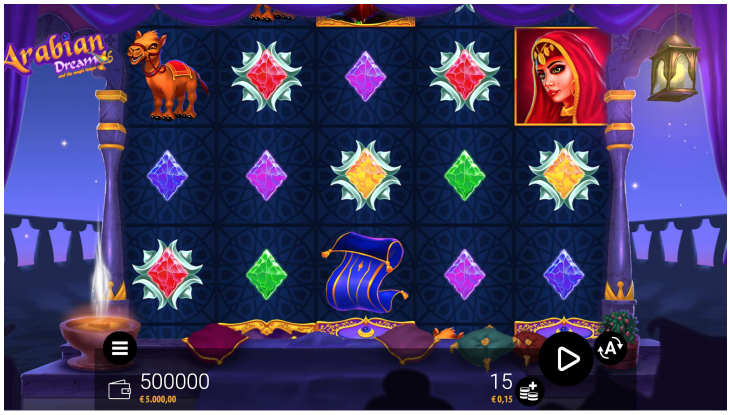


Figure **1.3.5**: The Paytable (Part 2)

- **Paylines:** The line in which a payout will be awarded based on the winning combination.A payline, also known as betting line or winning line, is a combination of symbols that results in a win, on a slot machine. Original slot machines only had one payline, and that would be won if three matching symbols created a horizontal line. When it comes to paylines, you can see how much your payline will win by looking at a paytable. Nowadays, paylines aren't just horizontal, and can be in a huge number of shapes, from zigzag to trapezium. At this time, in the majority of slot machine games, the common number of betting lines is 20-30 paylines per game. When the game developers want to build a slot machine, their first thought is to build a game with different paylines and different geometrical configurations, bonuses, free spins and multipliers. Some slot machines let the player choose how many paylines want to play and as the number of paylines grows, the bet grows too. In the majority of the slot machines, the player has the privilege to play with all paylines of the game and no extra bet is required.The paylines that don't require extra bet are known as fixed.

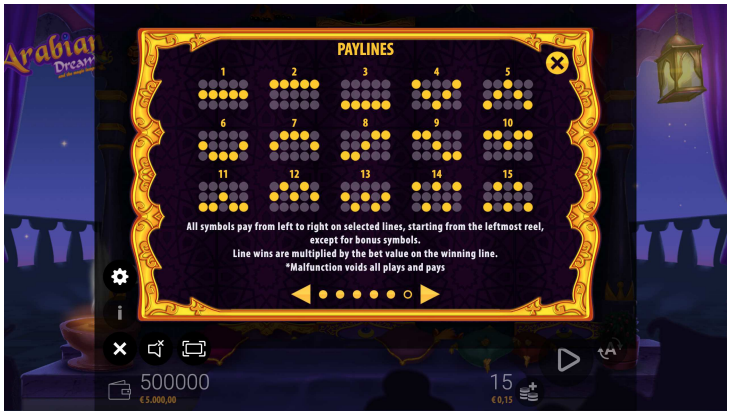


Figure **1.3.6**: The paylines of the game

- **Win/Display Box:** Box that shows how much a player wins.

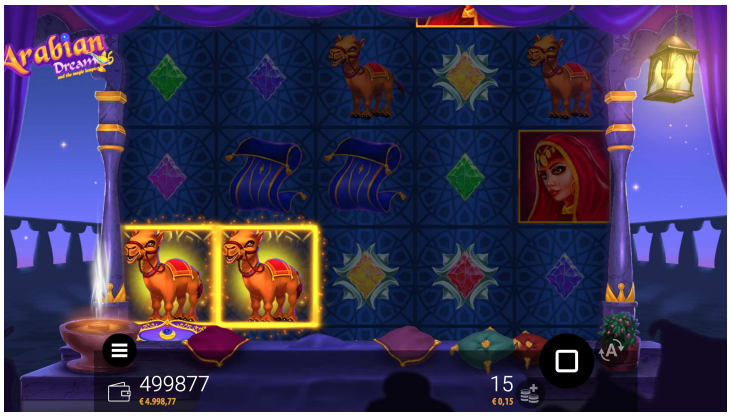


Figure **1.3.7**: Winning 2 Camels

- **Bonus Game:** Bonus games and features can make playing slots more enjoyable, and can give the chance to the player to win a whopping jackpot. The different types of bonus games and features in slot machines are(4):

1. **Free spins:** Player receives a number of free spins at your current stake where you keep any winnings and can't lose any money during the bonus.

Figure **1.3.8**: Bonus Game : Free Spins

2. **Re-spins:** The player is rewarded with an extra spin at his/her current stake for free, often with symbols held in place.

Figure **1.3.9**: Bonus Game : Re-Spins

3. **Pick X out of Y:** It is called a picking bonus, this let the player to play an interactive mini-game and win instant prizes by picking from random themed selections. The X of Y bonus in slots is commonly known as the bonus game on the second screen. The screen with the reels on it is replaced by the second screen, and the bonus game starts. Such a bonus game will follow the color scheme or theme with symbols of the main game and it usually have a name that is connected to the main theme. During the X of Y bonus game, the player has to do something specific, for instance, he may need to pick out one of many objects, some of which carry a prize and some which act as traps and cause the bonus game to end immediately. In some slots, the bonus game on the second screen has several levels, each of which must be completed before a player can advance to the next one. There are many good things about the bonus game on the second screen, it offers a player another chance to win rewards, apart from the main game. Also, it acts as a break from the main game and refreshes the player to enter the fray anew on the main screen.These bonuses come in a wide variety and it can be fun to explore the various interesting options offered in different slot machine games.Such bonus games are usually more technically advanced and exciting than the regular on reel bonus features and they are generally more rewarding and involve much more of a player's interest and attention.
4. **Find N Matching Items:** Player collects some items in this bonus game. When the player collects the necessary number of one of the items, he/she wins prizes like free-spins, respin, multipliers or a winning prize.

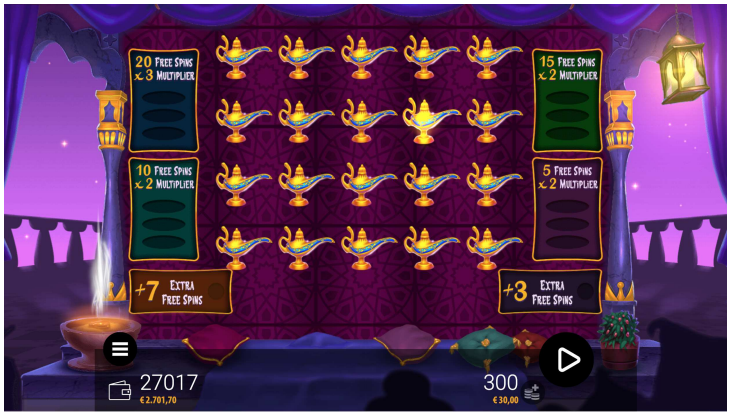


Figure **1.3.10**: Bonus Game:Find N Matching Items - The Game

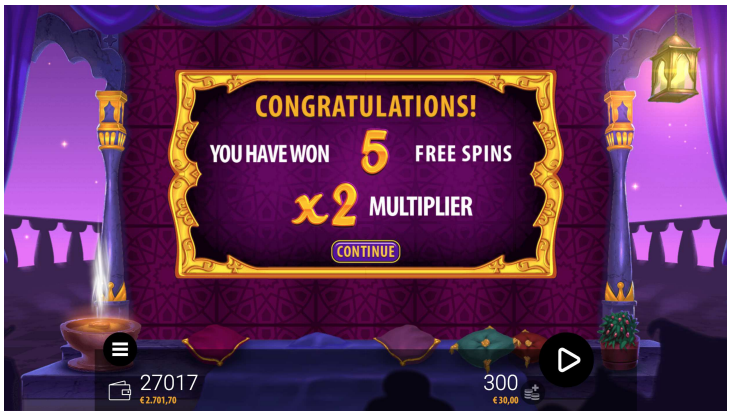


Figure **1.3.11**: Bonus Game:Find N Matching Items - Prizes

5. **Instant Win:** Player wins an instant cash prize.
6. **Added/Extra wilds:** New or extra wild symbols may be added to the reels to perform more winning combinations.
7. **Pathway game:** Progress through a series of mini-games and win bigger prizes as players goes along. It can often be part of a progressive jackpot game.
8. **Double Up:** The slot machine game 'FA-FA Twins' is made by BETSOFT(5). The player clicks the side of the coin that thinks will appear and flip. If he chooses correctly, his prize will be doubled.

Figure **1.3.12**: Fa-Fa Twins : Double UP

The probability of occuring the correct side and not, is not always equally possible. Sometimes the probability of occuring the correct side can be 35% and the wrong one 65%, but sometimes can be 50% each depends, it depends on developer's or company's preferences.

1.4 Symbol types

Except for the simple symbols, the reels of a slot machine game may consists of 2 other types. The first is 'Scatter' and the second one is 'Wild'.

1.4.1 Scatter

Scatter is a special symbol that the majority of slot machine games consists of and they do not have to appear on a payline to grant a win. These symbols , usually, appear anywhere on any of the reels for the player to win. In addition, the wild symbols (that will be discussed in next subsection) very rarely substitute scatter symbols. Rules governing scatters differ in different games, so it is important to check the paytable before playing. This way, the player will be able to know what is aiming for. It is common to search out 3 or more scatter symbols showing on the reels. The scatter symbols can multiply all the wins combined.

Figure **1.4.13**: Scatter symbol

In many games, if you get 3 scatters on the screen, you trigger a feature game of that slot. The features vary from free spins to bonus games. Some of the noted slots like Infinium, use a combination of scatter symbols to launch 15 free spins with win multiplier (×3). The scatter is an additional feature in the majority of the 5-reel slots however isn't thus common among the 3-reel slot machines. Scatter can also feature in progressive jackpot games, slots containing bonus games, 7-reel games and any other kind of slot that a player can think of.

1.4.2 Wild

In slots, the wild symbol it's the joker in deck of cards. It's a special symbol that can be used with any symbol on the reels in order to create winning combinations. That's why the wild symbol has the power to substitute for all other symbols on the reels and in doing so, complete winning paylines that otherwise wouldn't result in a win. Usually, the only exception to this rule is that the wild can't replace scatter symbols, usually, free spin or other bonus symbols. If there is a 'K' symbol on reels 1 and 2 and wild on reel 3 and 4, this will create the combination of 'K-K-K-K-Q' (from 'K-K-W-W-Q'). The wild symbols, in some slots, carry no individual value of its own, so they are not included in paytable. In this case, the wild symbol substitutes the symbol from previous reel in same line. The slot machine games that are presented in **??**, **1.4.15** and **1.4.16** are designed and developed by ZeusPlay (3).

Figure **1.4.14**: Wild symbol

Wild Not-In-Paytable

In case of having a scatter symbol in next reel of the same line and the reels before have only wild symbols like 'W-W-W-Scatter-A', they substitute the symbol that has the max payment in paytable of that n-combination of wilds(in this case if A is the most valuable symbol then, the combination becomes A-A-A-Scatter-A).If there are 5 wild symbols in the same line, the proccess that is being followed is as well as in previous case (e.g the 'W-W-W-W-W' becomes 'A-A-A-A-A').

Wild In-Paytable

The majority of slot games that have wild symbols, they come with prizes all of their own.In order to be understood how they work when they are included in paytable, the cases of what they substitute will be presented below.

- **Simple substitute:** The combination of 'A-W-A-K-Q' becomes 'A-A-A-K-Q'.
- **Substitute the most valued symbol:** If the combination W-W-W-A-K occurs, the game will compare the prizes of 3 wild and 4 A's, that they will be extracted from paytable. If 3-wild combination wins more than 4-A's, then they line will not change and three wild symbols will be payed. If 4-A's wins more that 3-wilds, the line will change to 'A-A-A-A-K' and four aces will be payed.

There are different types of slot wild symbols that they can be or not in paytable.Some of them are presented and explained below:

- **Regular Wild:** They are included in almost every slot, and they simply substitute for the regular symbols to increase the number of wins.They will be often found as the highest paying symbol.
- **Reel Specific Wild:** In some slots the wilds are only on certain reels.This means that is

not possible to create wins with wilds alone.Common configurations include wilds on reels 2, 3 and 4 or on reels 1, 4 and 5.

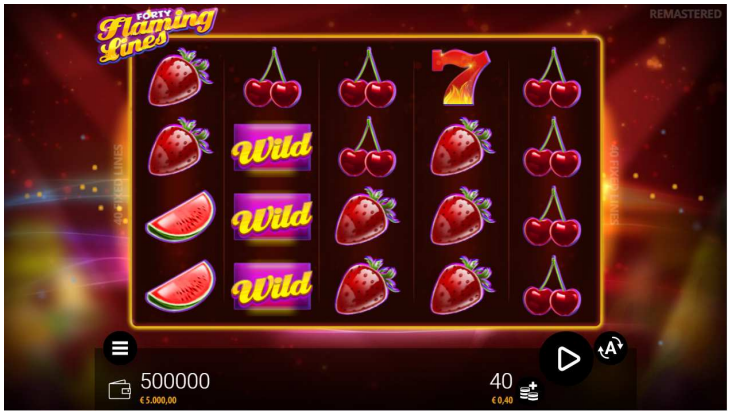


Figure **1.4.15**: 40 Flaming Lines

Figure **1.4.16**: Paytable of 40 Flaming Lines

- **Stacked Wild:** Many slots have symbols which come in long strips on the reels, this means the entire reel (usually 3 or 4 symbols) can be covered in the same symbol. When wilds are stacked, you can end up with 1 or more reels of them and create a lot of wins. Some games combine reel specific wilds with stacked symbols. If a game has stacked wilds on all reels, it is possible to have the entire grid wild at the same time (for maximum payouts on every line).

- **Expanding Wild:** The wild symnol expands to cover all the rows of a specific reel of the window game.Also, it may be possible to get wilds on all of the reels. More rarely, wilds will expand sideways.

Figure **1.4.17**: Exotic Cats

- **Shifting Wild:** The more complex the slots become, the more behaviours the wilds are take on. Moving wilds include bonus games where a different reel is wild each time and others where the wilds move on their own, usually sideways though sometimes from top to bottom.
- **Sticky Wilds:** During bonus games you will often get wilds which stick to the reels after they land. As you go through a free spins game, you will collect more and more wilds which means that your last few spins can win you a very large amounts of credits. Other games have sticky wilds whose reel is predetermined.For example the entire left reel can stay wild throughout the bonus.

Figure **1.4.18**: Wild Spartans

- **Random Wild:** In some slots, a PRNG determines where the wild symbols will take place in window.It can be used for the base game and bonus either.

1.5 Mathematics and Slots

In the last decade, mathematics has been taken more and more seriously into account in gambling, as being the essence that governs the games of chance and the only rigorous tool providing information on optimal play, where possible. The mathematical models that are applied on slot machines, theoretically, belong to mathematical domains such as Combinatorics, Topology, Probability Theory and Statistics. Clever gamblers use mathematics to look for the smallest advantages, and casinos use sophisticated mathematical tools to devise new ways of drawing in players. A patent granted to the Norwegian scientist Inge Telnaes in 1984 reworked the gambling business.Prior to Telnaes' invention, slot machines were essentially mechanical devices. Besides being difficult to tune and maintain, mechanical slot machines suffered from an essential problem.A machine with 3 reels, with 12 symbols on each, with one of those 12 symbols a cherry. The likelihood of getting three cherries, and winning the jackpot is 1 in 1728. If the casino wants to make money, the jackpot payout should be, say 1,700$ on a 1$ bet.If we add a fourth reel it will get us to a jackpot of about 20,000$.People do not like machines with more reels they intuitively, and rightfully, feel that extra reels diminish their chance of winning. Another possibility is to put more symbols on each reel.But the astronomical jackpots you see in casinos these days would then need actually monumental machines.Inge Telnaes proposed a simple solution: Let a random number generator a computer chip determine the combination of symbols that appear when the reels stop.In different words, use a chip to control where the reels stop on a spin, but create the illusion that the wheels stopped on their own.The history of gambling is also intertwined with that of a less reputable group tricksters and swindlers. In the long run, the only sure way to make money by gambling is to create the illusion that your opponent can win, while keeping the odds firmly on your side.That offers people who apprehend maths an awfully solid advantage.

CHAPTER 2

Mathematics and Slots

2.1 The Statistical Approach

The method that is presented below, is based on result from probability theory. If N is the number of spins of a reel, k the number of distinct symbols that is the recorded number of symbols in the N spins and (n_1=n_2= ..... =n_k) are the recorded numbers of occurrence of the symbols S_1,S_2,...,S_k respectively, according to the theory, the approximations for probability are shown P below:
for each i from 1 to k,

$$\frac{n_i}{N} \simeq P(S_i) = \frac{m_i}{s} \tag{1}$$

To understand for what are we talking about, some examples are being solved below in order to see how the probabilities are estimated.

Example. In a slot machine of k=13 symbols on a reel, we recorded the following frequencies after N=15,000 spins (n_i refers to symbol S_i) $n_1 = 2435, n_2 = 720, n_3 = 740, n_4 = 680, n_5 = 820, n_6 = 1200, n_7 = 1400, n_8 = 600, n_9 = 1405, n_{10} = 300, n_{11} = 1800, n_{12} = 2050, n_{13} = 850$ occurrences. The probability of each symbol occurence is calculated below. The sum of all probabilities usually are approximately 1 and not equal due to the totally random generated symbols:

$$\sum_{i=1}^{k} P(S_i) \approx 1 \tag{2}$$

$P(S_1) = \frac{n_1}{N} = \frac{2435}{15{,}000} = 0.1623$

$P(S_2) = \frac{n_2}{N} = \frac{720}{15{,}000} = 0.048$

$$P(S_3) = \frac{n_3}{N} = \frac{740}{15{,}000} = 0.0493$$

$$P(S_4) = \frac{n_4}{N} = \frac{680}{15{,}000} = 0.0453$$

$$P(S_5) = \frac{n_5}{N} = \frac{820}{15{,}000} = 0.0546$$

$$P(S_6) = \frac{n_6}{N} = \frac{1200}{15{,}000} = 0.08$$

$$P(S_7) = \frac{n_7}{N} = \frac{1400}{15{,}000} = 0.0933$$

$$P(S_8) = \frac{n_8}{N} = \frac{600}{15{,}000} = 0.04$$

$$P(S_9) = \frac{n_9}{N} = \frac{1405}{15{,}000} = 0.0936$$

$$P(S_{10}) = \frac{n_{10}}{N} = \frac{300}{15{,}000} = 0.02$$

$$P(S_{11}) = \frac{n_{11}}{N} = \frac{1800}{15{,}000} = 0.12$$

$$P(S_{12}) = \frac{n_{12}}{N} = \frac{2050}{15{,}000} = 0.1366$$

$$P(S_{13}) = \frac{n_{13}}{N} = \frac{850}{15{,}000} = 0.0566$$

In order to obtain good approximations of the ratios, a large enough number of spins N should be used. In example, a pseudo-random number generator(PRNG) has been used in order to generate the random occurences of the symbols. In order to obtain good and exact approximations via PRNG, the N must be greater(or equal) than 10 million, $\mathrm{N} \geq 10,000,000$.

2.2 The Combinatorial Approach

2.2.1 Hits, Hit Rate and Hit Frequency

The analysis of mathematics of a slot machine game starts by calculating all possible winning combinations of 2, 3, ..., n-pair of symbol$_i$ (where symbol$_i$ is not special), they called *hits*, where i=1, 2, 3, ..., k is the number of distinct symbols. All slot machine games, usually, hold a big number of symbols on each reel. Firstly, the sum of the number of symbols per reel-column (S_j) is calculated and then stored in a matrix S.

$$S_j = \sum_{i=1}^{n} t_{ij} \tag{3}$$

where j=1, 2, 3, ..., n

$$\mathrm{S} = \begin{bmatrix} s_1 & s_2 & s_3 & ... & s_n \end{bmatrix}$$

Figure **2.2.1**: The matrix S

Cycle (CL) represents the number of all possible combinations that can occur in game and the formula is:

$$CL = \prod_{j=1}^{n} S_j \tag{4}$$

- **i**: Current row in matrix
- **j**: Current column in matrix
- **m**: The number of rows in matrix
- **n**: The number of columns in matrix
- $\mathbf{t}_{ij}$: A symbol element in row i and column j

In order to create dynamic formulas that fit on every slot machine game without taking account the number of reels and no wild included in game, three general equations have been extracted (experimentally). All results are stored in an $m \times n$ matrix called H. In an array, called T, the number of each symbol per reel is stored. The number of symbol$_i$ in reel j is stored in T[i][j] etc.

$$\mathrm{T} = \begin{bmatrix} t_{11} & t_{12} & ... & t_{1n} \\ t_{21} & t_{12} & ... & t_{2n} \\ t_{31} & t_{32} & ... & t_{3n} \\ . & . & . & . \\ . & . & . & . \\ . & . & . & . \\ t_{m1} & t_{m2} & ... & t_{mn} \end{bmatrix}$$

Figure **2.2.2**: The 2D matrix T

If ($n = c$):

$$H_{ic} = \prod_{j=1}^{c} t_{ij} \tag{5}$$

If $(c < n < 2 + c)$:

$$H_{ic} = \prod_{j=1}^{c} t_{ij} \times (S_m - t_{im}) \tag{6}$$

If $(n \geq 2 + c)$:

$$H_{ic} = \prod_{j=1}^{c} t_{ij} \times (S_m - t_{im}) \times \prod_{p}^{n} S_p \tag{7}$$

If there is a wild symbol in game and it substitutes the current symbol that the formulas are being used, then the above formulas 5, 6 and 7 are going to be (only when wild symbols are on the middle reels and the rule is 'left to right' or 'right to left' e.g. if the game consists of 5 reels and the rule is 'left to right', then the first reel must not contains wild symbols):

If $(n = c)$:

$$H_{ic} = \prod_{j=1}^{c} (t_{ij} + t_{kj}) \tag{8}$$

If $(c < n < 2 + c)$:

$$H_{ic} = \prod_{j=1}^{c} (t_{ij} + t_{kj}) \times (S_m - t_{im} - t_{km}) \tag{9}$$

If $(n \geq 2 + c)$:

$$H_{ic} = \prod_{j=1}^{c} (t_{ij} + t_{kj}) \times (S_m - t_{im} - t_{km}) \times \prod_{p}^{n} S_p \tag{10}$$

The probability of occurring the winning combination of c-symbol is calculated with the equation below

$$P_{H_{ic}} = \frac{H_{ic}}{CL} \tag{11}$$

For the equations above:

- **i** : Index in row i
- **c** : Combination of doublet, triplet, quadruple, five, ..., last column e.g. if c=2 then we are looking for a doublet
- **n**: The number of columns in matrix

- **m** : c+1
- **p** : m+1
- $\mathbf{t}_{ic}$: A symbol element in row i and a combination of c in a row
- **k** : Index of wild

In order to calculate the hits of scatter symbols, all the possible combinations of k-scatter symbols on an n-reel slot machine game must be calculated with the formula below.

For each n= 1, 2, 3, ..., k pair of scatter symbols,

$$C_n^m = \binom{m}{n} = \frac{m!}{n!(m-n)!} \tag{12}$$

Hit rate shows how many games a player need to play on average to win a particular prize. Hit frequency is the odds that the machine will hit a payout on any given spin. It is calculated by the total winning combination divided by the total possible combinations. The higher the percentage the more frequently the machine will hit a winning combination. Hit rate of the game can be calculated with several different ways:

$$HR = \frac{CL}{Hits} \tag{13}$$

$$HF = \frac{100\%}{HR} \tag{14}$$

For specific combinations, the equations that can be used for a specific symbol-i and combination of c are given below:

$$HR_{ij} = \frac{CL}{Hits_{ij}} \tag{15}$$

$$HF_{ic} = \frac{100\%}{HR_{ic}} \tag{16}$$

In order to understand how the above formulas work, two examples with simple-symbol and scatter-symbol hits and hit frequency will be presented and analyzed.

Example 1. On a 5-reel slot machine game, will be counted the hits and the probabilities of occurring 3, 4 or 5 simple symbols on a (3×5) window. The reels do not need to have a specific arrangement of the symbols on each of them and there is no special symbol (e.g. wild). In the majority of games, the arrangement of the scatter symbols is much more essential and

major issue and this is because the behavior of scatter help to shape RTP (it will be analyzed in Chapter 3 in subsection 3.2.1).

$$\mathrm{R} = \begin{bmatrix} 10 & A & K & 10 & 10 \\ A & K & Q & K & J \\ Q & 10 & 10 & J & K \\ K & 10 & 10 & 10 & 10 \\ A & J & K & Q & 10 \\ J & Q & 10 & J & Q \\ K & 10 & J & 10 & 10 \\ 10 & Q & Q & Q & J \\ J & J & J & Q & J \\ 10 & J & K & J & J \\ J & 10 & J & 10 & 10 \\ 10 & 10 & Q & A & J \\ Q & J & A & 10 & A \\ 10 & J & & 10 & 10 \\ Q & & & Q & J \\ J & & & J & Q \end{bmatrix}$$

Figure **2.2.3**: The reels of a slot machine

In order to calculate the hits of the game, the T and the S matrix must be constructed and the cycle (CL) must be calculated. First row refers to '10' symbol, the second to 'J', the third one to 'Q', the fourth to 'K' and the last one to 'A'.

$$\mathrm{T} = \begin{bmatrix} 5 & 5 & 3 & 6 & 6 \\ 4 & 5 & 3 & 4 & 6 \\ 3 & 2 & 3 & 4 & 2 \\ 2 & 1 & 3 & 1 & 1 \\ 2 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Figure **2.2.4**: The matrix T

$$\mathrm{S} = \begin{bmatrix} 16 & 14 & 13 & 16 & 16 \end{bmatrix}$$

Figure **2.2.5**: The matrix S

$$CL = \prod_{j=1}^{n} S_j = \prod_{j=1}^{5} S_j = 16 \times 14 \times 13 \times 16 \times 16 = \mathbf{745{,}472}$$

The number of instances of occurring double or triplet of the symbol A can be extracted by using the equations 5, 6 and 7:

- **Double of A's:** Condition of equation 7 is satisfied $n \geq 2+c \Rightarrow 5 \geq 2+3 \Rightarrow 5 \geq 5$ which

is true and i=5 because 'A' symbol is on the last index (5th) of T matrix.

$$H_{ic} = \prod_{j=1}^{c} t_{ij} \times (S_m - t_{im}) \times \prod_{p}^{n} S_p \Rightarrow H_{52} = \prod_{j=1}^{2} t_{5j} \times (S_3 - t_{53}) \times \prod_{p=4}^{5} S_p$$

$$= 2 \times 1 \times (13 - 1) \times 16 \times 16 \Rightarrow H_{52} = \mathbf{6{,}144}$$

winning combinations that the player wins the instance of double A's (number of symbols A in Reel_1 $\times$ number of symbols 'A' in Reel_2 $\times$ X in Reel_3 *times* number of all symbols in Reel_4 $\times$ number of all symbols in Reel_5). Also, hit frequency and hit rate of this combination can be calculated.

$$HR_{ij} = \frac{CL}{H_{ij}} \Rightarrow HR_{52} = \frac{CL}{H_{52}} = \frac{745,472}{6,144} \Rightarrow HR_{52} = \mathbf{121.33}$$

This result informs that, statistically, every 121 spins a Double of A's occurs.

$$HF_{ic} = \frac{100\%}{HR_{ic}} \Rightarrow HF_{52} = \frac{100\%}{HR_{52}} = \frac{100\%}{121.33} \Rightarrow HF_{52} = \mathbf{0.82}\%$$

The probability of occurring a Double of A's is

$$P_{H_{ic}} = \frac{H_{ic}}{CL} \Rightarrow P_{H_{52}} = \frac{H_{52}}{CL} = \frac{6,144}{745,472} \Rightarrow P_{H_{52}} = 0.82\%$$

- **Triplet of A's:** Condition of equation 7 is satisfied $n \geq 2+c \Rightarrow 5 \geq 2+3 \Rightarrow 5 \geq 5$ which is true and i=5 because 'A' symbol is on the last index (5th) of T matrix. Hit frequency and hit rate of this combination can be calculated.

$$H_{ic} = \prod_{j=1}^{c} t_{ij} \times (S_m - t_{im}) \times \prod_{p}^{n} S_p \Rightarrow H_{53} = \prod_{j=1}^{3} t_{5j} \times (S_4 - t_{54}) \times \prod_{p=5}^{5} S_p$$

$$= 2 \times 1 \times 1 \times (16 - 1) \times 16 \Rightarrow H_{53} = \mathbf{480}$$

winning combinations that the player wins the instance of triplet A's (number of symbols 'A' in Reel_1 $\times$ number of symbols 'A' in Reel_2 $\times$ number of symbols 'A' in Reel_3 $\times$ X in Reel_4 $\times$ number of all symbols in Reel_5).

$$HR_{ij} = \frac{CL}{H_{ij}} \Rightarrow HR_{53} = \frac{CL}{H_{53}} = \frac{745,472}{480} \Rightarrow HR_{53} = \mathbf{1{,}553.067}$$

This result informs that, statistically, every 1,553 spins a Triplet of A's occurs.

$$HF_{ic} = \frac{100\%}{HR_{ic}} \Rightarrow HF_{53} = \frac{100\%}{HR_{53}} = \frac{100\%}{1,553.067} \Rightarrow HF_{53} = \mathbf{0.064}\%$$

The probability of occurring a Triplet of A's is

$$P_{H_{ic}} = \frac{H_{ic}}{CL} \Rightarrow P_{H_{53}} = \frac{H_{53}}{CL} = \frac{480}{745,472} \Rightarrow P_{H_{53}} = 0.06\%$$

- **Four of A's:** Condition of equation 7 is satisfied $c < n < 2{+}c \Rightarrow 4 < 5 < 2+4 \Rightarrow 4 < 5 < 6$ which is true and $i{=}5$ because 'A' symbol is on the last (5th) index of T matrix. Hit frequency and hit rate of this combination can be calculated.

$$H_{ic} = \prod_{j=1}^{c} t_{ij} \times (S_m - t_{im}) \Rightarrow H_{54} = \prod_{j=1}^{4} t_{5j} \times (S_5 - t_{55})$$

$$= 2 \times 1 \times 1 \times 1 \times (16-1) \Rightarrow H_{54} = \mathbf{30}$$

winning combinations that the player wins the instance of triplets A's (number of symbols 'A' in Reel_1 × number of symbols 'A' in Reel_2 × number of symbols 'A' in Reel_3 × number of all symbols in Reel_4 × X in Reel_5).

$$HR_{ij} = \frac{CL}{H_{ij}} \Rightarrow HR_{54} = \frac{CL}{H_{54}} = \frac{745,472}{30} \Rightarrow HR_{54} = \mathbf{24{,}849.07}$$

This result informs that, statistically, every 24,849 spins a Four of A's occurs.

$$HF_{ic} = \frac{100\%}{HR_{ic}} \Rightarrow HF_{54} = \frac{100\%}{HR_{54}} = \frac{100\%}{24,849.07} \Rightarrow HF_{54} = \mathbf{0.004}\%$$

The probability of occurring a Quadraple of A's is

$$P_{H_{ic}} = \frac{H_{ic}}{CL} \Rightarrow P_{H_{54}} = \frac{H_{54}}{CL} = \frac{30}{745,472} \Rightarrow P_{H_{54}} = 0.004\%$$

- **Five of A's:** Condition of equation 5 is satisfied, so $n{=}c{=}5$ and $i{=}5$ because 'A' symbol is on the last (5th) index of T matrix.

$$H_{ic} = \prod_{j=1}^{c} t_{ij} \Rightarrow H_{55} = \prod_{j=1}^{5} t_{5j} = 2 \times 1 \times 1 \times 1 \times 1 \Rightarrow H_{55} = \mathbf{2}$$

winning combinations that the player wins the instance of five A's (number of symbols 'A' in Reel_1 × number of symbols 'A' in Reel_2 × number of symbols 'A' in Reel_3 × number of symbols 'A' in Reel_4 × number of symbols 'A' in Reel_5), where **X** is the number of any of the symbols that are not A's. Hit frequency and hit rate of this combination can be

calculated.

$$HR_{ij} = \frac{CL}{H_{ij}} \Rightarrow HR_{55} = \frac{CL}{H_{55}} = \frac{745,472}{2} \Rightarrow HR_{55} = \mathbf{372{,}736}$$

This result informs that, statistically, every 121 spins a Double of A's occurs.

$$HF_{ic} = \frac{100\%}{HR_{ic}} \Rightarrow HF_{55} = \frac{100\%}{HR_{55}} = \frac{100\%}{372,736} \Rightarrow HF_{55} = \ \mathbf{0.00027}\%$$

The probability of occurring a Five of A's is

$$P_{H_{ic}} = \frac{H_{ic}}{CL} \Rightarrow P_{H_{55}} = \frac{H_{55}}{CL} = \frac{2}{745,472} \Rightarrow P_{H_{55}} = 0.0002\%$$

A random instance of a 3× 5 window is captured from the previous set of reels, which consists of simple symbols, left to right rule and 3 paylines

$$\mathrm{W} = \begin{bmatrix} 10 & 10 & 10 & 10 & 10 \\ A & J & J & 10 & J \\ Q & Q & Q & Q & K \end{bmatrix}$$

Figure **2.2.6**: A 3× 5 window from the slot machine game

$$\mathrm{P} = \begin{bmatrix} 0 & 3 & 12 & 24 & 60 \\ 0 & 0 & 15 & 30 & 90 \\ 0 & 0 & 15 & 30 & 90 \\ 0 & 0 & 30 & 90 & 300 \\ 0 & 0 & 60 & 180 & 600 \end{bmatrix}$$

Figure **2.2.7**: Paytable for the game

Each row represents a symbol and each column the combination of that symbol (one-double-triplet-quadruple-pentas). The first row has the payments of 10's, the second of J's symbol, the third one of Q's, the fourth one of K's and the last one of A's if a combination of 3's, 4's or 5's occur for each one. It can be noticed that there are two payments in specific instance, a **pentas** of 10's which awards the player with the prize of 60 coins and a **quadruple** of Q's with the prize of 30 coins.

Now, if a wild symbol is added in game then, the equations 8, 9 and 10will be used to calculate the hits instead of 5, 6 and 7. It must be noted that the wild symbols will be placed in reels 2, 3, 4 and 5, they will substitute all symbols in paytable and it will not be in it. All matrices will be renamed with their previous name plus a tone.

$$\text{R' } = \begin{bmatrix} 10 & A & K & 10 & 10 \\ A & K & Q & K & J \\ Q & 10 & 10 & Wild & K \\ K & 10 & 10 & 10 & 10 \\ A & J & K & Q & 10 \\ J & Q & 10 & J & Q \\ K & 10 & J & 10 & 10 \\ 10 & Wild & Q & Q & J \\ J & J & J & Q & J \\ 10 & J & K & J & J \\ J & 10 & J & 10 & 10 \\ 10 & 10 & Q & A & J \\ Q & J & A & 10 & A \\ 10 & J & Wild & 10 & 10 \\ Q & Q & & Q & J \\ J & & & J & Q \end{bmatrix}$$

Figure **2.2.8**: The new reels of the slot machine

$$\text{T' } = \begin{bmatrix} 5 & 5 & 3 & 6 & 6 \\ 4 & 5 & 3 & 3 & 6 \\ 3 & 2 & 3 & 4 & 2 \\ 2 & 1 & 3 & 1 & 1 \\ 2 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{bmatrix}$$

Figure **2.2.9**: The new matrix T'

$$\text{S' } = \begin{bmatrix} 16 & 15 & 14 & 16 & 16 \end{bmatrix}$$

Figure **2.2.10**: The new matrix S'

$$CL' = \prod_{j=1}^{n} S'_j = \prod_{j=1}^{5} S'_j = 16 \times 15 \times 14 \times 16 \times 16 = \mathbf{860{,}160}$$

The number of instances of occurring double or triplet of the symbol A can be extracted by using the equations 5, 6 and 7:

- **Double of A's:** Condition of equation 7 is satisfied $n \geq 2+c \Rightarrow 5 \geq 2+3 \Rightarrow 5 \geq 5$ which is true and $i=5$ and $k=6$ because 'A' symbol is on (5th) index and 'Wild' is on the last (6th) of T matrix.

$$H'_{ic} = \prod_{j=1}^{c} (t'_{ij}+t'_{kj}) \times (S'_m - t'_{im} - t'_{km}) \times \prod_{p}^{n} S'_p \Rightarrow H'_{52} = \prod_{j=1}^{2} (t'_{5j}+t'_{6j}) \times (S'_3 - t'_{53} - t'_{63}) \times \prod_{p=4}^{5} S'_p$$

$$= (2+0) \times (1+1) \times (13-1-1) \times 16 \times 16 = 2 \times 2 \times 11 \times 16 \times 16 \Rightarrow H'_{52} = \mathbf{11{,}264}$$

winning combinations that the player wins the instance of double A's ((number of symbols A +number of symbols 'Wild')in Reel_1 × (number of symbols 'A' in Reel_2 +number of symbols 'Wild') in Reel_2 × X in Reel_3 *times* number of all symbols in Reel_4 × number of all symbols in Reel_5). Also, hit frequency and hit rate of this combination can be calculated.

$$HR'_{ij} = \frac{CL'}{H'_{ij}} \Rightarrow HR'_{52} = \frac{CL'}{H'_{52}} = \frac{860,160}{11,264} \Rightarrow HR'_{52} = \mathbf{76.36}$$

This result informs that, statistically, every 76 spins a Double of A's occurs.

$$HF'_{ic} = \frac{100\%}{HR'_{ic}} \Rightarrow HF'_{52} = \frac{100\%}{HR'_{52}} = \frac{100\%}{76.36} \Rightarrow HF'_{52} = \mathbf{1.309}\%$$

- **Triplet of A's:** Condition of equation 7 is satisfied $n \geq 2{+}c \Rightarrow 5 \geq 2+3 \Rightarrow 5 \geq 5$ which is true and *i=5* because 'A' symbol is on (5th) index and WIld is on the last (6th) of T matrix. Hit frequency and hit rate of this combination can be calculated.

$$H'_{ic} = \prod_{j=1}^{c}(t'_{ij} + t'_{kj}) \times (S'_m - t'_{im} - t'_{km}) \times \prod_{p}^{n} S'_p \Rightarrow H'_{53} = \prod_{j=1}^{3}(t'_{5j} + t'_{6j})$$

$$\times (S'_4 - t'_{54} - t'_{64}) \times \prod_{p=5}^{5} S'_p = (2+0) \times (1+1) \times (1+1) \times (16-1-1) \times 16 =$$

$$= 2 \times 2 \times 2 \times 14 \times 16 \Rightarrow H'_{53} = \mathbf{1{,}792}$$

winning combinations that the player wins the instance of triplet A's ((number of symbols A +number of symbols WIld) in Reel_1 × (number of symbols A +number of symbols WIld) in Reel_2 × (number of symbols A +number of symbols WIld) in Reel_3 × X in Reel_4 × number of all symbols in Reel_5).

$$HR'_{ij} = \frac{CL'}{H'_{ij}} \Rightarrow HR'_{53} = \frac{CL'}{H'_{53}} = \frac{860,160}{1,792} \Rightarrow HR'_{53} = \mathbf{480}$$

This result informs that, statistically, every 480 spins a Triplet of A's occurs.

$$HF'_{ic} = \frac{100\%}{HR'_{ic}} \Rightarrow HF'_{53} = \frac{100\%}{HR'_{53}} = \frac{100\%}{480} \Rightarrow HF'_{53} = \mathbf{0.208}\%$$

- **Four of A's:** Condition of equation 6 is satisfied $c < n < 2{+}c \Rightarrow 4 < 5 < 2+4 \Rightarrow 4 < 5 < 6$ which is true and *i=5* because 'A' symbol is on (5th) index and WIld is on the

last (6th) of T matrix. Hit frequency and hit rate of this combination can be calculated.

$$H'_{ic} = \prod_{j=1}^{c}(t'_{ij} + t'_{kj}) \times (S'_m - t'_{im} - t'_{kj}) \Rightarrow H'_{54} = \prod_{j=1}^{4}(t'_{5j} + t'_{6j}) \times (S'_5 - t'_{55} - t'_{65}) =$$

$$= (2+0) \times (1+1) \times (1+1) \times (1+1) \times (16-1-0) = 2 \times 2 \times 2 \times 2 \times 15 \Rightarrow$$

$$\Rightarrow H'_{54} = \mathbf{240}$$

winning combinations that the player wins the instance of triplets A's ((number of symbols A +number of symbols WIld) in Reel_1 $\times$ (number of symbols A +number of symbols WIld) in Reel_2 $\times$ (number of symbols A +number of symbols WIld) in Reel_3 $\times$ (number of symbols A +number of symbols WIld) Reel_4 $\times$ X in Reel_5).

$$HR'_{ij} = \frac{CL'}{H'_{ij}} \Rightarrow HR'_{54} = \frac{CL'}{H_{54}} = \frac{860,160}{240} \Rightarrow HR'_{54} = \mathbf{3{,}584}$$

This result informs that, statistically, every 3,584 spins a Four of A's occurs.

$$HF'_{ic} = \frac{100\%}{HR'_{ic}} \Rightarrow HF'_{54} = \frac{100\%}{HR'_{54}} = \frac{100\%}{3,584} \Rightarrow HF'_{54} = \ \mathbf{0.028}\%$$

- **Five of A's:** Condition of equation 5 is satisfied, so n=c=5 and i=5 because 'A' symbol is on (5th) index and WIld is on the last (6th) of T matrix. Hit frequency and hit rate of this combination can be calculated.

$$H'_{ic} = \prod_{j=1}^{c}(t'_{ij}+t'_{kj}) \Rightarrow H'_{55} = \prod_{j=1}^{5}(t'_{5j}+t'_{6j}) = (2+0)\times(1+1)\times(1+1)\times(1+1)\times(1+0) =$$

$$= 2 \times 2 \times 2 \times 2 \times 1 \Rightarrow H'_{54} = \mathbf{16}$$

winning combinations that the player wins the instance of five A's ((number of symbols A +number of symbols WIld) in Reel_1 $\times$ (number of symbols A +number of symbols WIld) in Reel_2 $\times$ (number of symbols A +number of symbols WIld) in Reel_3 $\times$ (number of symbols A +number of symbols WIld) in Reel_4 $\times$ (number of symbols A +number of symbols WIld) in Reel_5), where **X** is the number of any of the symbols that are not A's. Hit frequency and hit rate of this combination can be calculated.

$$HR'_{ij} = \frac{CL'}{H'_{ij}} \Rightarrow HR'_{55} = \frac{CL'}{H'_{55}} = \frac{860,160}{16} \Rightarrow HR'_{55} = \mathbf{53{,}760}$$

This result informs that, statistically, every 121 spins a Double of A's occurs.

$$HF'_{ic} = \frac{100\%}{HR'_{ic}} \Rightarrow HF'_{55} = \frac{100\%}{HR'_{55}} = \frac{100\%}{53,760} \Rightarrow HF'_{55} = \ \mathbf{0.002}\%$$

A random instance of a 3× 5 window is captured from the new previous set of reels, which consists of simple symbols, left to right rule and 3 paylines

$$\text{W'} = \begin{bmatrix} K & 10 & K & 10 & 10 \\ 10 & Wild & 10 & K & 10 \\ J & J & J & Wild & Q \end{bmatrix}$$

Figure **2.2.11**: A 3× 5 window from the new slot machine game

There are two payments in specific instance, a **triplet** of 10's which awards the player with the prize of 12 coins and a **quadruple** of J's with the prize of 30 coins.

Example 2. On a 5-reel slot machine game, will be counted the hits and the probabilities of occurring 3, 4, 5 or 6 scatter symbols on a (3×5) window. The reels do not need to have a specific arrangement of the symbols on each of them and there is no wild symbol. The scatter symbol is symbolized with 'Scatter' in this example. Also, a random instance of a 3× 5 window is captured from the current set of reels, which consists of simple and scatter symbols, left to right rule and 3 paylines.

$$\mathrm{R} = \begin{bmatrix} K & 10 & 10 & 10 & 10 \\ S & A & K & 10 & 10 \\ A & 10 & Q & K & J \\ Q & K & 10 & J & K \\ A & J & K & Q & S \\ J & Q & S & J & Q \\ K & S & J & 10 & S \\ 10 & Q & Q & Q & J \\ S & J & J & Q & J \\ 10 & J & 10 & J & J \\ J & 10 & J & S & 10 \\ 10 & K & S & A & J \\ Q & J & A & 10 & A \\ 10 & J & J & J & J \\ Q & 10 & 10 & 10 & J \\ 10 & J & & 10 & 10 \\ Q & & & Q & J \\ J & & & J & Q \end{bmatrix}$$

Figure **2.2.12**: The reels of a slot machine

$$\mathrm{W} = \begin{bmatrix} S & Q & S & Q & S \\ A & S & J & Q & Q \\ Q & Q & Q & J & S \end{bmatrix}$$

Figure **2.2.13**: A 3× 5 window from the slot machine game

In order to calculate the hits of the game, the T and the S matrix must be constructed and the cycle (CL) must be calculated. First row refers to '10' symbol, the second to 'J', the third one to 'Q', the fourth to 'K', the fifth one to 'A' and the last to 'Scatter'.

$$\mathrm{T} = \begin{bmatrix} 5 & 4 & 4 & 6 & 4 \\ 3 & 6 & 4 & 5 & 8 \\ 4 & 2 & 2 & 4 & 2 \\ 2 & 2 & 2 & 1 & 1 \\ 2 & 1 & 1 & 1 & 1 \\ 2 & 1 & 2 & 1 & 2 \end{bmatrix}$$

Figure **2.2.14**: The matrix T

$$\mathrm{S} = \begin{bmatrix} 18 & 16 & 15 & 18 & 18 \end{bmatrix}$$

Figure **2.2.15**: The matrix S

$$\mathrm{P} = \begin{bmatrix} 0 & 3 & 12 & 24 & 60 \\ 0 & 0 & 15 & 30 & 90 \\ 0 & 0 & 15 & 30 & 90 \\ 0 & 0 & 30 & 90 & 300 \\ 0 & 0 & 60 & 180 & 600 \\ 0 & 0 & 100 & 300 & 2400 \end{bmatrix}$$

Figure **2.2.16**: Paytable for the game

$$CL = \prod_{j=1}^{n} S_j = \prod_{j=1}^{5} S_j = 18 \times 16 \times 15 \times 18 \times 18 = \mathbf{1{,}399{,}680}$$

All the possible winning combinations (hits) of n scatter symbols on an m reels slot machine game, is given from the formula below. In current example k=1, 2, 3, 4, 5, n=5 and the number of rows m=3, of the window instance that the player will play on it, and the hits (H_c) are calculated for each n that the prize of the paytable is greater than 0. The **X** in combinations below means any symbol except for scatter.

$$C_k^n = \binom{n}{k} = \frac{n!}{k!(n-k)!} \tag{17}$$

5 Scatter

$C_k^n = \frac{n!}{k!(n-k)!} \Rightarrow C_5^5 = \frac{5!}{5!(5-5)!}$ 1 possible combination(s)

- **SSSSS**

$$H_{51} = (2 \times 3) \times (1 \times 3) \times (2 \times 3) \times (1 \times 3) \times (2 \times 3) = 6 \times 3 \times 6 \times 6 \times 3 = 1{,}944$$

$$Prob_{51} = \frac{H_{51}}{CL} = \frac{1{,}944}{1{,}399{,}680} = 0.14\%$$

The probability (P(H_{5j})) where j=1, 2, ..., 5 of occurring a 5-scatter winning combination is approximately 0.1% and there are 1,944 winning combinations-hits of them (H_{5j}).

4 Scatter

$C_k^n = \frac{n!}{k!(n-k)!} \Rightarrow C_4^5 = \frac{5!}{4!(5-4)!} = 5$ combination(s)

- **SSSSX**

$$H_{41} = (2 \times 3) \times (1 \times 3) \times (2 \times 3) \times (1 \times 3) \times (18 - 2 \times 3) = 6 \times 3 \times 6 \times 3 \times 12 = 3,888$$

$$Prob_{41} = \frac{H_{41}}{CL} = \frac{3,888}{1,399,680} = 0.28\%$$

- **SSSXS**

$$H_{42} = (2 \times 3) \times (1 \times 3) \times (2 \times 3) \times (18 - 1 \times 3) \times (2 \times 3) = 6 \times 3 \times 6 \times 15 \times 6 = 9,720$$

$$Prob_{42} = \frac{H_{42}}{CL} = \frac{9,720}{1,399,680} = 0.69\%$$

- **SSXSS**

$$H_{43} = (2 \times 3) \times (1 \times 3) \times (15 - 2 \times 3) \times (1 \times 3) \times (2 \times 3) = 6 \times 3 \times 9 \times 3 \times 6 = 2,916$$

$$Prob_{43} = \frac{H_{43}}{CL} = \frac{2,916}{1,399,680} = 0.21\%$$

- **SXSSS**

$$H_{44} = (2 \times 3) \times (16 - 1 \times 3) \times (2 \times 3) \times (1 \times 3) \times (2 \times 3) = 6 \times 13 \times 6 \times 3 \times 6 = 8,424$$

$$Prob_{44} = \frac{H_{44}}{CL} = \frac{8,424}{1,399,680} = 0.6\%$$

- **XSSSS**

$$H_{45} = (18 - 2 \times 3) \times (1 \times 3) \times (2 \times 3) \times (1 \times 3) \times (2 \times 3) = 12 \times 3 \times 6 \times 3 \times 6 = 3,888$$

$$Prob_{45} = \frac{H_{44}}{CL} = \frac{3,888}{1,399,680} = 0.28\%$$

The probability (P(TH_{4j})) where j=1, 2, ..., 5 of occurring a 5-scatter winning combination is approximately 2% and there are 28,836 winning combinations-hits of them (TH_{4j}).

$$TH_{4j} = \sum_{j=1}^{5} H_{4j} = 2,916 + 3,888 + 3,888 + 8,424 + 9,720 = \mathbf{28,836}$$

$\mathrm{P(TH}_{4j}\mathrm{)}= \frac{TH_{4j}}{Cycle} = \frac{28{,}836}{1{,}399{,}680} = \mathbf{2.06}\%$

3 Scatter

$C_n^m = C_3^5 = \binom{5}{3} = \frac{5!}{3!(5-3)!} = 10$ combination(s)

- **SSSXX**

$$H_{31} = (2\times3)\times(1\times3)\times(2\times3)\times(18-1\times3)\times(18-2\times3) = 6\times3\times6\times15\times12 = 19{,}440$$

$$Prob_{31} = \frac{H_{31}}{CL} = \frac{19{,}440}{1{,}399{,}680} = 0.28\%$$

- **SSXSX**

$$H_{32} = (2\times3)\times(1\times3)\times(15-2\times3)\times(1\times3)\times(18-2\times3) = 6\times3\times9\times3\times12 = 5{,}832$$

$$Prob_{32} = \frac{H_{32}}{CL} = \frac{5{,}832}{1{,}399{,}680} = 0.417\%$$

- **SXSSX**

$$H_{33} = (2\times3)\times(16-1\times3)\times(2\times3)\times(1\times3)\times(18-2\times3) = 6\times13\times6\times3\times12 = 16{,}848$$

$$Prob_{33} = \frac{H_{33}}{CL} = \frac{16{,}848}{1{,}399{,}680} = 1.2\%$$

- **XSSSX**

$$H_{34} = (18-2\times3)\times(1\times3)\times(2\times3)\times(1\times3)\times(18-2\times3) = 12\times3\times6\times3\times12 = 7{,}776$$

$$Prob_{34} = \frac{H_{34}}{CL} = \frac{7{,}776}{1{,}399{,}680} = 0.55\%$$

- **SSXXS:**

$$H_{35} = (2\times3)\times(1\times3)\times(15-2\times3)\times(18-1\times3)\times(2\times3) = 6\times3\times9\times15\times6 = 14{,}580$$

$$Prob_{35} = \frac{H_{35}}{CL} = \frac{14{,}580}{1{,}399{,}680} = 1.04\%$$

- **SXSXS**

$$H_{36} = (2\times3)\times(16-1\times3)\times(2\times3)\times(18-1\times3)\times(2\times3) = 6\times13\times6\times15\times6 = 42{,}120$$

$$Prob_{36} = \frac{H_{36}}{CL} = \frac{42,120}{1,399,680} = 3\%$$

- **XSSXS**

$$H_{37} = (18-2\times3)\times(1\times3)\times(2\times3)\times(18-1\times3)\times(2\times3) = 12\times3\times6\times15\times6 = 19,440$$

$$Prob_{37} = \frac{H_{37}}{CL} = \frac{19,440}{1,399,680} = 1.39\%$$

- **SXXSS**

$$H_{38} = (2\times3)\times(16-1\times3)\times(15-2\times3)\times(1\times3)\times(2\times3) = 6\times13\times9\times3\times6 = 12,636$$

$$Prob_{38} = \frac{H_{38}}{CL} = \frac{12,636}{1,399,680} = 0.9\%$$

- **XSXSS**

$$H_{39} = (18-2\times3)\times(1\times3)\times(15-2\times3)\times(1\times3)\times(2\times3) = 12\times3\times9\times3\times6 = 5,832$$

$$Prob_{39} = \frac{H_{39}}{CL} = \frac{5,832}{1,399,680} = 0.42\%$$

- **XXSSS**

$$H_{3-10} = (18-2\times3)\times(16-1\times3)\times(2\times3)\times(1\times3)\times(2\times3) = 12\times13\times6\times3\times6 = 16,848$$

$$Prob_{3-10} = \frac{H_{3-10}}{CL} = \frac{16,848}{1,399,680} = 1.2\%$$

The probability (P(TH_{3j})) where j=1, 2, ..., 5 of occurring a 5-scatter winning combination is approximately 11.5% and there are 161,452 winning combinations-hits of them (TH_{3j}).

$$TH_{3j} = \sum_{j=1}^{5} H_{3j} = 19,440 + 5,832 + 16,848 + 7,776$$

$$+14,680 + 42,120 + 19,440 + 12,636 + 5,832 + 16,848 = \mathbf{161,452}$$

$$P(TH_{3j}) = \frac{TH_{3j}}{Cycle} = \frac{161,452}{1,399,680} = \mathbf{11.53}\%$$

Noticing the Figure 2.2.11, there are two payments, a triplet of Q‘s which awards the player with the prize of 15 coins and the second one is a

2.3 The Programming Approach

In order to understand how the algorithms of simple, scatter and wild symbols work and how the prize is calculated to be awarded to the player when the rules of the game are

- The rule is left to right
- Wild is **not** in paytable
- There are wild symbols only in 2, 3 and 4 reels (W3)
- The line bet is 1
- The number of paylines is 3, so the total bet = line bet×paylines = 1×3 = 3

two (2) algorithms and examples will be presented. The above matrices will be used in order to analyze the functionality of the algorithms. The **2.3.17**, **2.3.19** matrix will be used for Algorithm 1, and **2.3.18** for Algorithm 2. The P, the L matrices and the Map(K, V) are the same for all W matrices.

$$\mathrm{W1} = \begin{bmatrix} 10 & 10 & 10 & Q & K \\ A & J & Q & Q & Q \\ K & K & Q & J & S \end{bmatrix}$$

Figure **2.3.17**: A 3× 5 window from a random set of reels of a slot machine game

$$\mathrm{W2} = \begin{bmatrix} 10 & S & 10 & Q & K \\ S & J & Q & S & Q \\ K & S & Q & J & S \end{bmatrix}$$

Figure **2.3.18**: A 3× 5 window from a random set of reels of a slot machine game

$$\mathrm{W3} = \begin{bmatrix} 10 & Q & 10 & Q & K \\ A & Wild & A & Wild & Q \\ K & 10 & Q & J & A \end{bmatrix}$$

Figure **2.3.19**: A 3× 5 window from a random set of reels of a slot machine game

$$\mathrm{P} = \begin{bmatrix} 0 & 2 & 4 & 16 & 128 \\ 0 & 4 & 32 & 128 & 256 \\ 0 & 4 & 32 & 128 & 256 \\ 0 & 4 & 32 & 128 & 256 \\ 0 & 8 & 64 & 256 & 1024 \\ 0 & 2 & 32 & 256 & 2048 \end{bmatrix}$$

Figure **2.3.20**: The paytable of the random game

$$\mathrm{L} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 & 2 \end{bmatrix}$$

Figure **2.3.21**: The paylines of the random game

Each row i of P matrix refers to the payments of n symbols for '10', the second for 'J', the third one for 'Q', the fourth for 'K' and the last one for the 'A'. In scatter algorithm the C matrix holds the number of scatter symbols on each column. The n number refers to the j-index of the P matrix. In order to understand how that works, a data structure called 'map' which has as *key* the name of the symbol (type **char**) and as *value* the i-index (type **int**) of the symbol in P matrix.

Map(K, V)	
Key	**Value**
10	0
J	1
Q	2
K	3
A	4
S	5

Table **2.1**: Correspondence map between symbol and index M(K,V)

Algorithm 1 Payment for winning combinations (including wild symbol)

Data: $W, P, L, Map, linebet$
Result: $coins$
$aPair \leftarrow 0$
$index \leftarrow -1$
$coins \leftarrow 0$
$m \leftarrow length(W)$
$n \leftarrow length(W[0])$
$saveWildCoordinates()$
for *i=0 to length(L)* **do**
  if $W[L[i][j]][j] \neq$ '$Scatter$' **then**
    $aPair \leftarrow 0$
    for *j=0 to n* **do**
      if $W[L[i][j]][j] = W[L[i][j+1][j+1]$ *or* $W[L[i][j+1][j+1] =$ '$Wild$' **then**
        $W[L[i][j+1][j+1] \leftarrow W[L[i][j][j]$
        $aPair \leftarrow aPair + 1$
      else
        break
    end
  end
  $index \leftarrow Map.getKey(W[L[i][0]][0])$
  $coins \leftarrow coins + P[index][aPair] \times linebet$
  $retrieveWildSymbols()$
end

Algorithm execution for W1				
i	**j**	**aPair**	**index**	**coins**
-	-	0	-1	0
0	0	1	-1	0
0	1	2	-1	0
0	2	2	0	4
1	0	0	4	0
2	0	1	4	0
2	1	1	3	4

Table **2.2**: Trace table

The output (coins) of the algorithm for a triplet of 10's is **4 coins**.

Algorithm execution for W3					
i	**j**	**isPair**	**aPair**	**index**	**coins**
-	-	false	0	-1	0
0	0	false	0	-1	0
1	0	true	1	-1	0
1	1	true	2	-1	0
1	2	true	3	-1	0
1	3	false	3	4	256
2	0	false	0	3	4

Table **2.3**: Trace table

The output (coins) of the algorithm for a quadrauple of A‘s is **256 coins**.

Algorithm 2 Payment for winning combinations (scatter symbols)

Data: $W, P, L, Map, totalbet$
Result: $coins$
$m \leftarrow length(W)$
$n \leftarrow length(W[0])$
$times \leftarrow 1$
$coins \leftarrow 0$
$C[0...n] \leftarrow 0$
$aPair \leftarrow 0$
foreach i *to* n **do**
  foreach j *to* m **do**
    if $W[i][j] =$ '$Scatter$' **then**
      $C[i] \leftarrow C[i] + 1$
    end
  end
  if $C[i] > 0$ **then**
    $times \leftarrow times \times C[i]$
    $aPair \leftarrow aPair + 1$
  end
end
$index \leftarrow Map.getKey('Scatter')$
$coins \leftarrow P[index][aPair] \times totalbet \times times$

Algorithm execution					
i	**j**	**C[i]**	**times**	**aPair**	**coins**
-	-	-	1	0	0
0	0	0	1	0	0
0	1	1	1	0	0
0	2	1	1	1	0
1	0	1	1	1	0
1	1	1	1	1	0
1	2	2	2	2	0
2	0	0	2	2	0
2	1	0	2	2	0
2	2	0	2	2	0
3	0	0	2	2	0
3	1	1	2	2	0
3	2	1	2	3	0
4	0	0	2	3	0
4	1	0	2	3	0
4	2	1	2	4	1,536

Table **2.4**: Trace table

The output (coins) of the algorithm for 2-quadraple of S‘ is **12,288 coins**.

CHAPTER 3

Optimization

3.1 Variable Neighborhood Search (VNS)

Variable neighborhood search (VNS) is a metaheuristic, or framework for building heuristics, aimed at solving combinatorial and global optimization problems. Its basic idea is systematic change of a neighborhood combined with a local search. It is a metaheuristic that exploits systematically the idea of neighborhood change, both in descent to local minima and in escape from the valleys which contain them. VNS relies heavily upon the following facts (6):

- **Fact 1:** A local minimum with respect to one neighborhood structure is not necessarily so for another
- **Fact 2:** A global minimum is a local minimum with respect to all possible neighborhood structures
- **Fact 3:** For many problems local minima with respect to one or several neighborhoods are relatively close to each other.

Since its inception, VNS has undergone many developments and been applied in numerous fields (7). A deterministic optimization problem may be formulated as

$$min\{f(x)|x \in X, X \subseteq S\} \tag{1}$$

where S, X, x and f denote respectively the solution space and feasible set, a feasible solution and a real valued objective function, respectively. If S is a finite but large set, a combinatorial optimization problem is defined. If $S{=}R^n$, it's about continuous optimization. A solution $x^*{\in}$ S is optimal if an exact algorithm for problem 1, if one exists, finds an optimal solution x^*, together with the proof of its optimality, or shows that there is no feasible solution, i.e. $S{=}0$.

$$f(x^*) \leq f(x), \forall x \in S \tag{2}$$

In practice, the time to do should be finite. If one deals with a continuous function one must admit a degree of tolerance i.e. stop when a feasible solution x^* has been found such that

$$f(x^*) \leq f(x) + \varepsilon, \forall x \in Sor \tag{3}$$

$$\frac{f(x^*) - f(x)}{f(x^*)} \leq \varepsilon, \forall x \in S \tag{4}$$

for some small positive ε.

In operations research and some other fields, there are many practical instances of problems like 1, that cannot be found an exact solution in reasonable time, because they are too large. In complexity theory (8) there are thousands of problems that they are nondeterministic polynomial-time hard (NP-hard). That means there is no algorithm with a number of steps polynomial in the size of the instances, is known, for solving any of them and that finding one would entail obtaining one for each and all of them. If a problem admits a polynomial algorithm, the power of this polynomial may be so large that realistic size instances cannot be solved in reasonable time in the worst case and sometimes also in the average case or most of the time. The solution for this is the Heuristics, which find quickly an approximate solution or sometimes an optimal solution but without proof of its optimality. Some of the heuristics have a worst-case guarantee that is the solution x_h obtained satisfies

$$\frac{f(x_h) - f(x)}{f(x_h)} \leq \varepsilon, \forall x \in X \tag{5}$$

for some ε which is rarely small. The ε is usually much larger than the error observed in practice and may be a bad guide in selecting a heuristic. Also, heuristics address another problem called *local optima* in order to avoid excessive computing. A local optimum x_L of 1 is such that

$$f(x_L) \leq f(x), \forall x \in N(x_L) \cap X \tag{6}$$

where $N(x_L)$ denotes a neighborhood of x_L. If there are many local minima, the range of values they span may be large. The globally optimum value $f(x^*)$ may differ substantially from the average value of a local minimum, even from the best such value among many, obtained by some simple heuristic such as multistart. There are, many ways to get out of a local optima and more precisely, the valleys which contain them (6). A local search heuristic starts by choosing an initial solution x, finding a direction of descent from x, withing a neighborhood $N(x)$ and moving to the minimum of $f(x)$ with $N(x)$ along that direction. If there is no direction of descent, the heuristic stops, otherwise it is iterated. The steepest descent direction, usually, referred to as best improvement. If we assume that there is an initial solution x, all these set of rules can be summarized in Algorithm 3 above

Algorithm 3 BestImprovement

Function `BestImprovement(`x`):`
  repeat
    $x' \leftarrow x$;
    $x \leftarrow arg\ min_{y \in N(x)} f(y)$
  until $f(x) \geq f(x')$
return x

A neighborhood structure *N(x)* is defined for all $x \in X$. In discrete optimization problems, usually, consists of all vectors obtained from x with some simple modifications, like completing one or two components of a 0-1 vector. At each step, the neighborhood *N(x)* of x is explored completely. This may be time-consuming, so an alternative method is to use the *first descent* heuristic. The vectors $x_i \in$ *N(x)* are then enumerated systematically and a move is made as soon as a descent direction is found. All these are summarized in Algorithm 4.

Algorithm 4 FirstImprovement

Function `FirstImprovement(`x`):`
  repeat
    $x' \leftarrow x$;
    $i \leftarrow 0$;
    repeat
      $i \leftarrow i + 1$
      $x \leftarrow arg\ min\{f(x), f(x_i)\}, x_i \in N(x)$
    until $f(x) \leq f(x_i)$ *or* $i = |N(x)|$
  until $f(x) \geq f(x')$
return x

3.1.1 Basic Schemes

A finite set of preselected neighborhood structures can be denoted with $\mathcal{N}_k$, (k=1, ..., k_{max}) and with $\mathcal{N}_k$(x) the set of solutions in the k-th neighborhood of x. the notation $\mathcal{N}'_k$ will be used, k=1, ..., k'_{max}, when local descent is being described. The neighborhoods $\mathcal{N}_k$ or $\mathcal{N}'_k$ may be induced from one ore more metric functions introduced into as solution space S. An optimal solution $x_o pt$ (or global minimum) is a feasible solution where a minimum of 1is reached. The $x' \in X$ is called a local minimum of (1) with respect to $\mathcal{N}_k$, if there is no solution $x \in \mathcal{N}_k(x') \subseteq f(x) < f(x')$.

Algorithm 5 NeighborhoodChange

Function `NeighborhoodChange`(x, x', k)**:**

```
if f(x') ≥ f(x) then
    x' ← x;
    k ← 1                                   ▷ Make a move
end
else
    k ← k + 1                               ▷ Next neighborhood
end
```

In order to solve 1 by using several neighborhoods, facts 3.1 can be used in three different ways: (i) deterministic (ii) stochastic (iii) both deterministic and stochastic. In Algorithm 5 is given the steps of the neighborhood change function. The Function 5 compares the new value *f(x')* with the incumbent value *f(x)* obtained in the neighborhood k in line 1. The k is returned to its initial value and the new incumbent is updated if an improvement is obtained. Otherwise, the next neighborhood is considered.

3.1.2 Shaking procedure

In order to resolve local minima traps, the shaking procedure must be used withing a VNS heuristic. If $\mathcal{N} = \{\mathcal{N}_1, ..., \mathcal{N}_{k_{max}}\}$ is a set of operators such that each operator $\mathcal{N}_k, 1 \leq k \leq k_{max}$ maps a given solution x to a predefined neighborhood structure $\mathcal{N}_k(x)$.

Algorithm 6 Shake

Function `Shake`$(x, k, \mathcal{N})$**:**

```
    choosex' ∈ N_k(x) at random
return x'
```

The order of operators in set $\mathcal{N}$ defines the order of examining neighborhood structures of a given solution x. The shaking procedure consists in selecting a random solution from the k-th neighborhood structure $\mathcal{N}_k x$.

3.1.3 Variable Neighborhood Descent

The Variable neighborhood descent (VND) method is obtained if the change of neighborhoods is performed in a deterministic way. Its steps are presented in a Algortihm 7. It is assumed that an initial solution x is given.

Algorithm 7 VND

Function `VND`(x, k'_{max})**:**

```
repeat
    k ← 1;
    repeat
        x ← arg min_{y∈N_k(x)} f(x)        ▷ Find the best neighbor in N_k(x)
        NeighborhoodChange(x, x', k)        ▷ Change neighborhood
    until k = k'_max
until no improvement is obtained
```

Most of the local search heuristics use in their descents one or two neighborhoods (k'$_{max}\leq$2). The final solution should be a local minimum with respect to all k'$_{max}$ neighborhoods and the chances of reaching a global minimum solution are larger when using VND than with a single neighborhood structure.

3.1.4 Basic Variable Neighborhood Search

The basic VNS method (9) combines deterministic and stochastic changes of neighborhood. In Algorithm 8, the steps are given.

Algorithm 8 VNS

Function `VNS`(x, k_{max}, t_{max})**:**

```
repeat
    k ← 1;
    repeat
        x' ← Shake(x, k)                    ▷ Shaking
        x'' ← FirstImprovement(x')          ▷ Local search
        NeighborhoodChange(x, x'', k)       ▷ Change neighborhood
    until k = k_max
    t ← CpuTime()
until t > t_max
```

3.1.5 General Variable Neighborhood Search

The General Variable Neighborhood Search (GVNS) is formed if the local search step (VNS algorithm) is replaced by VND. By using the general VNS (VNS/VND), some of the most successful applications were (10), (11), (12), (13), (14), (15) and (16).

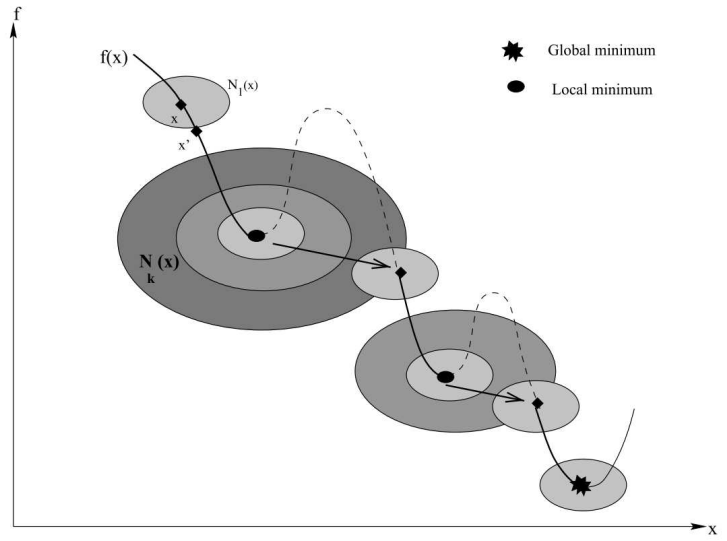


Figure **3.1.1**: General Variable Neighborhood Search (GVNS)

In Algorithm 9 are presented the steps of the General Variable Neighborhood Search (GVNS).

Algorithm 9 GVNS

Function `GVNS`(x, k'_{max}, t_{max})**:**
  repeat
    $k \leftarrow 1$;
    repeat
      $x' \leftarrow Shake(x, k)$
      $x'' \leftarrow FirstImprovement(x', k'_{max})$
      $NeighborhoodChange(x, x'', k)$
    until $k = k_{max}$
    $t \leftarrow CpuTime()$
  until $t > t_{max}$

3.2 RTP Optimization

3.2.1 Definition of Return To Player (RTP)

Return-To-Player (RTP) is one of the two most important parameters for a player to choose if he is going to play the game but it is also the main gambling game parameter which is under government regulation (17). It is the win as percentage of the bet and is the phrase that casinos and casino game makers use to describe the long term theoretical expected payback percentage from all wagers on a slot machine, video poker machine, or Video Lottery Terminal (VLT).

$$RTP = \frac{Coins\ Out}{Coins\ In} \times 100\% \tag{7}$$

or

$$RTP = \frac{Total\ Prize}{Cycle \times Paylines} \times 100\% \tag{8}$$

If a slot machine player is going to play a hundred rounds at 1$ on a slot game with a theoretical slot RTP of 93%, all other things being equal, he would expect to get a return of 93$ in wins from his staked 100$. A player wins back most of his wager. The remaining 7% of his bet, in this game, is the house edge, the money that casino earns off each bet. The RTP is calculated over millions spins.This process that RTP is calculated, called Monte Carlo simulation.Monte Carlo simulation is a computerized mathematical technique that help people account for risk in quantitative analysis and decision making.The technique is being used by professionals in such widely disparate fields as finance, project management, energy, manufacturing, engineering, research and development and transportation. Monte Carlo simulation furnishes alternative maker with a variety of attainable outcomes and therefore the possibilities they'll occur for any choice of action.The technique was initial utilized by scientists acting on the atom bomb.It was named for Monte Carlo, the Monaco resort city known for its casinos. Since its introduction in war II, Monte Carlo simulation has been accustomed model a variety of physical and conceptual systems.Monte Carlo is an approach to relate hypothetical slot machine gambling behavior to the statistical characteristics of the slot machines themselves. The formula for the computation of the RTP and also some necessary notations can be found with the Formula 7 or 8

- **Total Prize:** The amount of money that a window instance wins
- **Cycle:** How many times the reels spined in a Monte Carlo simulation
- **Paylines:** The number of lines that the game has enabled
- **Coins In:** Total amount of coins bet
- **Coins Out:** Total amount of coins won

In this thesis, Monte Carlo simulation with 100,000 and 1,000,000 separate runs will be used for the RTP calculations. In simulations, formula 8 is used, where the cycle represents the number of runs. In addition, there is another type of simulation called "full cycle". This latter type of simulation is used in the design phase (i.e., mathematical) of a slot machine game. It is an exhaustive cycle through all possible stop positions. The full cycle simulation allows an exact match to the calculations in the spreadsheet, but is not capable of calculating coinciding statistics across a series of feature games.These statistical characteristics-spins are the results of a pseudorandom number generator ((18)).The spins include your own as well as the spins by other players at online casinos.That means, that a player will not win back 96$ each time he

spins the reels a hundred times. You will win back your money, eventually, but the RTP makes no claim as to when exactly your winning spins will take place.

There are two types of simulation are Monte Carlo and Full Cycle. The Full Cycle simulation is used for the designing part (e.g. mathematical and developing) of a slot machine game. It is an exhaustive cycle through all possible stop positions.The full cycle simulation allows an exact match to the calculations in the spreadsheet, but is not capable of calculating coinciding statistics across a series of feature games.Monte Carlo simulation is typically using a random sampling of the possible outcomes. So it calculates coinciding wins across series of base/feature games, but does not provide a perfectly accurate result. Following, in Section 3.4, we discuss similar approaches for the RTP optimization problem, focusing on recent research work and the methods they utilize. Next, we introduce our methodology in Section 3.5. Section 3.6 presents the results of our methods on several games. Finally, we draw our conclusions and provide some research directns for future work in Section 4.2.

3.2.2 Volatility

Volatile in slot machine games, refers to the level of risk involved in the game. Volatility is used to describe how often and how much a player can expect to win during his playing sessions. Some games can present long dry spells with occasional big wins. Also, these games can often feature a large number of big wins in a short period of time, which are known as high-volatility slots. On the other hand, there are the low-volatility slots, where a player can score winning paylines frequently, but not large amount of credits. There are 5 main types of volatility:

- **Low:** The lower the volatility is, the lower risk takes the player in game. The payments are smaller but a lot more frequent. That's not to say that large wins are not possible on these games but it's not on the same scale as the wins on high volatility games. A game that has Low volatility with 94% RTP is Arabian Dream (19) developed by Zeusplay.

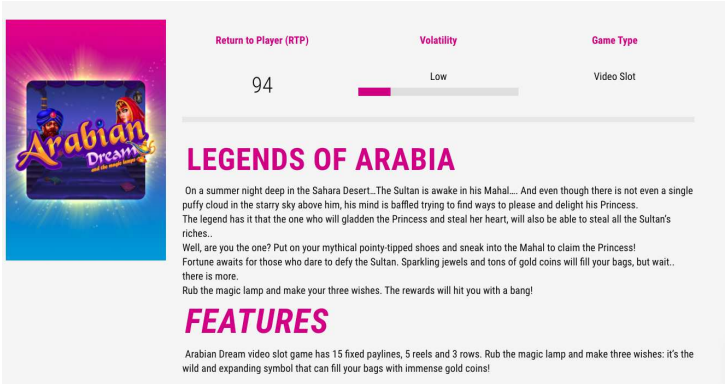


Figure **3.2.2**: Arabian Dream

- **Medium-Low:** A Medium-Low volatility game with 93% RTP is Super Hot (20) developed by Zeusplay.

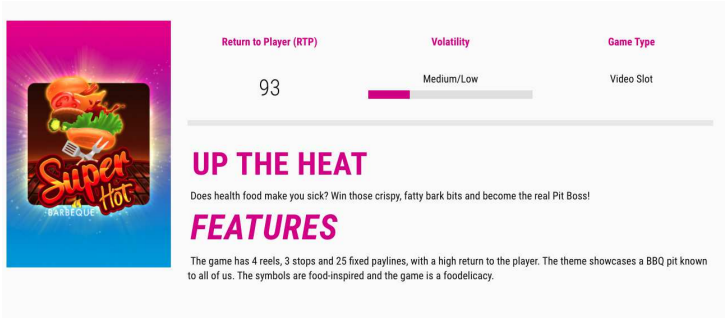


Figure **3.2.3**: Super Hot

- **Medium:** The games that are characterized by an average level of risk and they are popular to players-gamblers as it is a happy compromise between high and low volatility games. These games, with medium volatility, are perfect for those who do not want to risk too much of their bankroll but still are looking for big wins. A game that has Medium volatility with 94% RTP is Wild Charger (21) developed by Zeusplay.

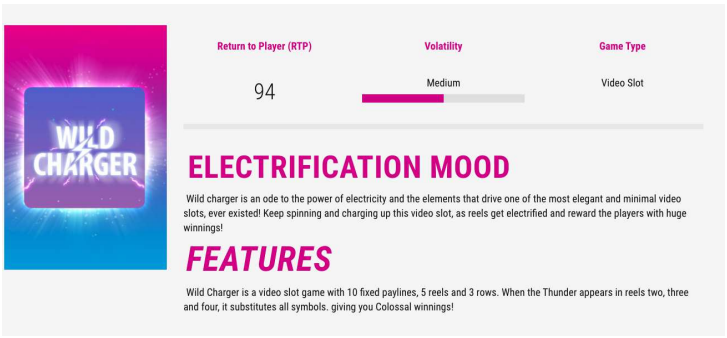


Figure **3.2.4**: Wild Charger

- **Medium-High:**A Medium-High volatility game with 94% RTP is Super Hot (22) developed by Zeusplay.

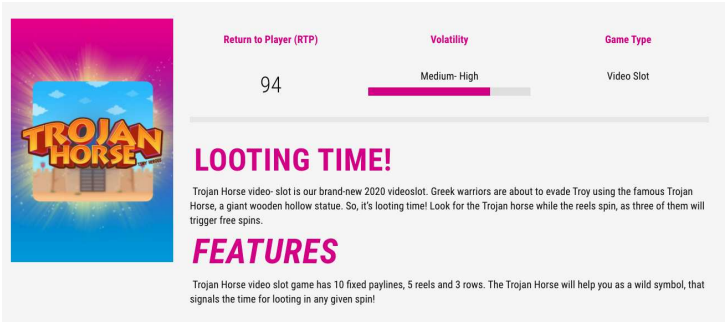


Figure **3.2.5**: Trojan Horse

- **High:** Higher volatility equates to higher risk.This means that slots rarely give the winning combinations, but massive payouts.These slots are preferred by people who have patience and a large enough bankroll to continue spinning without any big wins. A game that has High volatility with 94% RTP is Dice Tronic (21) developed by Zeusplay.

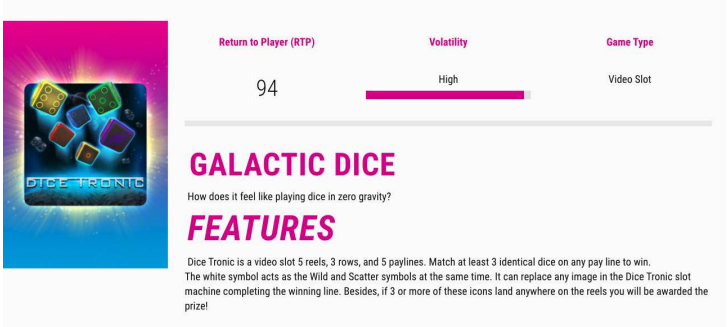


Figure **3.2.6**: Dice Tronic

In order to calculate the volatility, standard deviation must be found. Standard deviation (23) is a widely used measurement of variability or diversity used in statistics and probability theory.

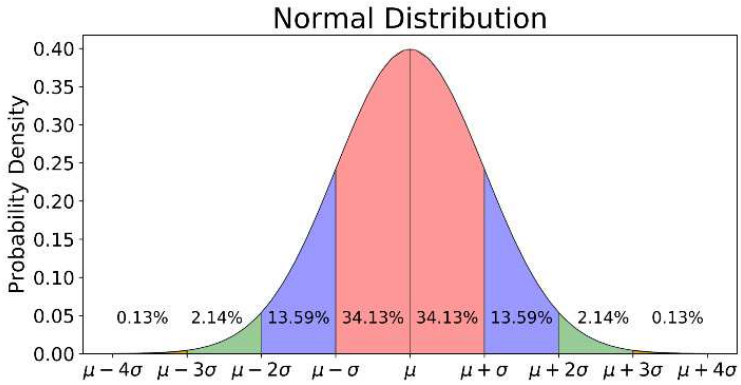


Figure **3.2.7**: Normal distribution

It shows how much variation or dispersion there is from the average (mean or expected value). A low standard deviation indicates that the data points tend to be very close to the mean, whereas high standard deviation indicates that the data is spread out over a large range of values.A normal distribution is a very important statistical data distribution pattern occurring in many natural phenomena. Random variation conforms to a particular probability distribution known as the normal distribution, which is the most commonly observed probability distribution. Fifty percent of the distribution lies to the left of the mean and fifty percent lies to the right of the mean ((24)).The equation of Standard Deviation is given below ((23)):

$$\sigma = \sqrt{\sum_{i=1}^{n} p_i \left(x_i - \bar{x}\right)^2} \tag{9}$$

p_i: the probability of each prize in base and feature game
$(x_i - \bar{x})$: difference between that prizes credit value and the mean credit win of the entire series of base/feature games.
The Equation 13 can be transformed in order to be applied on slot machine games to:

$$\sigma = \sqrt{\sum_{i=1}^{n} Prob_i \left(x_i - RTP\right)^2} \tag{10}$$

The formula of precise probability is given from:

$$Prob_i = \frac{Hits_i}{cycle} \tag{11}$$

The formula of precise payment is given from:

$$x_i = \frac{Payment_i}{NumberOfPaylines} \tag{12}$$

After calculating standard deviation, volatility index based on a 90%, 95% and 99% confidence interval each:

$$Volatility = 1.64 \times \sigma \tag{13}$$

$$Volatility = 1.96 \times \sigma \tag{14}$$

$$Volatility = 2.58 \times \sigma \tag{15}$$

There are three ways to determine the variation of slot machines:

- Study the payout rates of each particular slot. This information is publicly available as well as the game rules.
- Play the game!Developers of online gaming machines allow gamblers to play free demo games, so the user can see the frequency and amount of payments and decide whether it suits him or not.
- Look for various reviews and forums on the Internet.Gamblers usually share their experience together with the information on volatility on some forums.

3.3 Bonus Games

3.3.1 Free Games

A very common game feature is free spins, where the player plays them with no bet. There are two cases that can be considered. First, the case where free games do not retrigger additional free games and the second one that do. These free spins contribute in total RTP.If the AVG_B is the average win of the base game, AVG_F is the average win of the free game and the free games are being triggered m_1 times with probability of pr_1 per spin. The average number of free games with **no retrigger** is given from the formula:

$$AVG = m_1 pr_1 \tag{16}$$

The FINAL average win is then:

$$FAVG = AVG_B + AVG \times AVG_F = AVG_B + m_1 pr_1 \times AVG_F \tag{17}$$

The total RTP is:

$$RTP = \frac{FAVG}{bet} \times 100 = \frac{AVG_B + m_1 pr_1 \times AVG_F}{bet} \times 100 \tag{18}$$

The average number of free games that the free game can **retrigger** m_2 games with probability pr_2, is given from the formula:

$$AVG = m_1 pr_1 \sum_{i=1}^{\infty} (m_2 pr_2)^i = m_1 pr_1 (\frac{1}{1 - m_2 pr_2}) \tag{19}$$

So, the final RTP for the base and feature game with retrigger is:

$$RTP = \frac{AVG_B + m_1 pr_1 \times AVG_F \times (\frac{1}{1-m_2 pr_2})}{bet} \times 100 \tag{20}$$

As an example can be mentioned a game called Mystere du Prince whose feature game can retrigger when 3, 4 or 5 scatter symbols appear on the (3×5) window screen.

3.4 Related Work

Although the RTP optimization problem is one of the most important parts of the slot machine game design, there are not many scientific articles that present a heuristic or metaheuristic approach to solve this problem. Only three papers have been published that present a metaheuristic approach to solve the RTP optimization problem and are briefly presented in Table **3.1**.

Reference	Approach	Simulation	Special symbols	Runs	Games tested
This work	VNS	Monte Carlo	Yes	100,000 & 1,000,000	3 3
(17)	GA(s)	Monte Carlo	No	100,000 or 1,000,000	1
(18)	DE	Monte Carlo	No	1,000,000	1
(25)	DDE and GA(s)	Full Cycle	Yes	All possible	1

Table **3.1**: Key literature contributions on RTP optimization

Balabanov et al. (17) presented a solution for the slot machine RTP optimization problem, where the symbols distribution of the base game is being controlled and, also, some other

features like free spins frequency or bonus game frequency. The fitness function in this work, was the absolute difference between the desired RTP and the obtained RTP. For Monte-Carlo simulations 1,000,000 or 100,000 separate runs were used in order to calculate the obtained RTP. They used Genetic Algorithms (GAs), with a randomized initial population and appropriate crossover and mutation operators. The target RTP of the example that the authors presented was 99%, starting from 91%. So, the algorithm increase the RTP in a small range in case the game already exists (a quality starting RTP). However, in most cases, the mathematicians of slot games usually try to develop a new slot machine game (from scratch) that has initial reels with randomized symbol distribution and the RTP might be equal, e.g., to 630% (big random value). So, it is not an trivial task for such algorithms to handle the case of large reductions and the huge difference between the target RTP and the initial RTP. The decrease of the RTP in slot machine game is the most hard problem for the mathematicians, since they have to solve the RTP optimization problem with hand adjustments and this usually requires a long time. Furthermore, the algorithm's behavior in case where special symbols appear in reels (e.g., wild, scatter) is also unknown.

The same authors in (18) also proposed a Differential Evolution (DE) algorithm since the problem of optimal symbols distribution on the reels is discrete and its convergence was found to be faster than GAs. Additionally, they also used Monte-Carlo simulations with 1,000,000 separate runs over ten trials for better accuracy in base game. Their model of slot machine game contained only simple symbols and not scatter, wild or bonus games (scatter are symbols that pay everywhere on the screen and substitute all other symbols). The authors proposed a multicriteria optimization approach as future work, for the symbol frequencies, the free spins frequency, the game volatility, and the bonus game frequency.

Keremedchiev et al. (25) proposed a Genetic Algorithm methodology for the RTP optimization problem, in bonus and free games frequency as parallel computing evolutionary optimization. The fitness function of base game that was used, was computed with Full Cycle method instead of Monte-Carlo simulations. In current algorithm the model of slot machine game does contain as features bonus game, simple symbols that pay on each payline, scatter symbols, wild and bonus symbols that trigger the bonus game. This experiment, as it refers in it, showed that the use of Genetic Algorithm with exact numerical calculation of RTP as fitness function, was very efficient and improved the slot game parameters by better adjustment of RTP, bonus game, and free games frequency. Finally, the biggest drawback was that, the exact RTP computation was time consuming and slowed down the optimization process. The authors also proposed a Discrete Differential Evolution (DDE) implementation as a further research.

3.5 Research Methodology

Variable Neighborhood Search (VNS) constitutes a metaheuristic solution method which is systematically changing a neighborhood structure. VNS was initially proposed by Mladenović and Hansen (26), (6). Nowadays, a large number of successful VNS applications have been reported in a wide range of areas (27), (28), (29), (30),(31). In this work, a Variable Neighborhood Descent heuristic (VND) schema was applied in base game's RTP and takes all symbols into account. The VND metaheuristic features a strong local search phase (intensification) utilizing a number of different neighborhood structures but lacks a diversification phase (shaking method) as compared to other VNS variants. The VND variant has been applied in several research works in the literature (32), (33)

The proposed solution method is analytically described in the following subsections and an overview is depicted in the following Figure **3.5.8**:

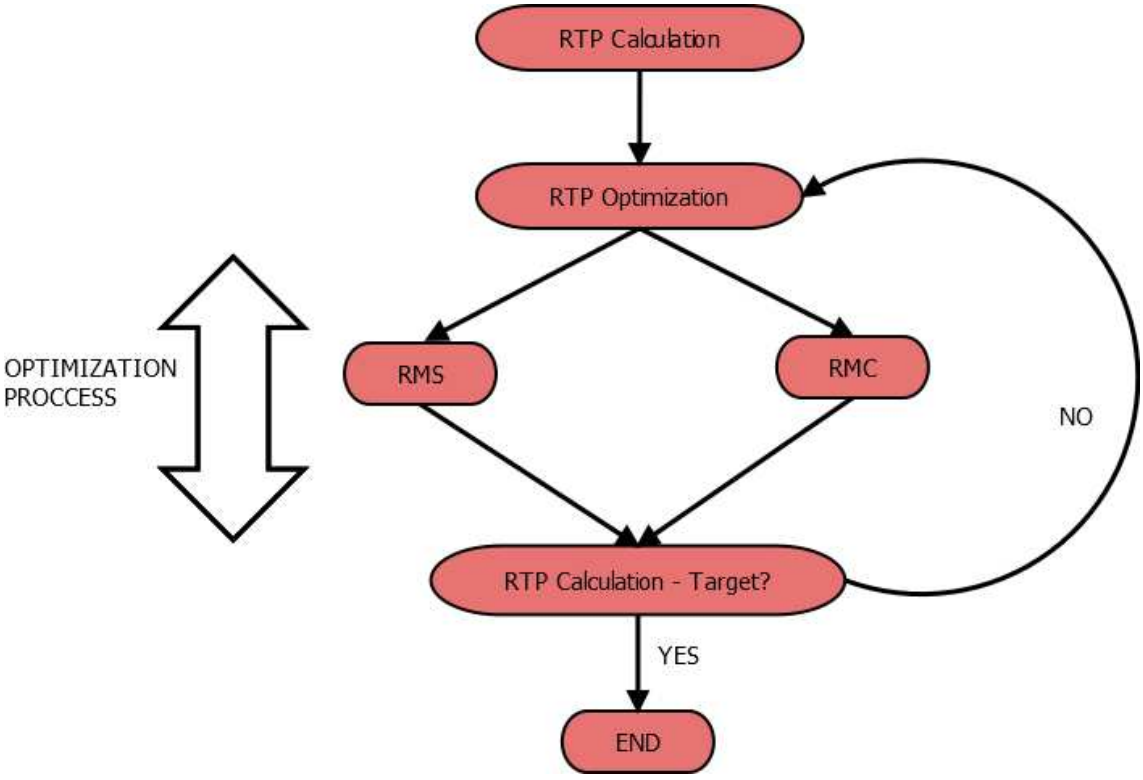


Figure **3.5.8**: Solution process

3.5.1 Solution Representation

A slot machine consists of n reels where the symbols are represented by a string and are equally distributed. The solution representation is a two-dimensional array with the distribution of the symbols on each reel. For example, in case of a five-reel (number of columns) slot machine

game with 48 symbols per reel (number of rows), the solution array (denoted as "Reels") would have dimension 48×5.

Three input of the proposed algorithm are: i) the maximum number of iterations as a termination criterion (also a maximum CPU solution time can be used), ii) the target (desired) RTP of the user (RTP_{des}), and iii) the allowed deviation denoted (R) from the target RTP. Usually, the computation of the precise RTP is quite difficult and due to this reason, game developers are satisfied if the RTP belongs in a specific range. Hence, if $RTP_{curr} \in \left[\text{ RTP}_{\text{des}} - R, \quad \text{RTP}_{\text{des}} + R \right]$ then, the algorithm stops. Assume for example that, if we are given $R = 0.03$, $RTP_{des} = 0.83$, and $RTP_{curr} = 0.80$, then the algorithm stops. An extra feature of the algorithm is that, it allows the user to declare a minimum number of each symbol per reel. Thus, a fourth input of the proposed algorithm is an additional two-dimensional array containing the minimum Number Of Symbols per Reel and is denoted as "minNSR". The number of columns of this array is equal to the number of reels while its number of rows is equal to the number of distinct symbols. Thus, $minNSR(i, j)$ denotes the minimum required number of occurrences of the i symbol in the j-reel.

3.5.2 Neighborhood Structures

The proposed VND methodology utilizes two neighborhood structures denoted as "RMS" and "RMC". Both these local search operators aim to reduce the RTP as follows:

- **RMS :** Initially, the symbols are sorted in ascending order based on their corresponding payments. The Replace Maximum Symbol (RMS) local search operator replaces the symbol_i with the biggest payment by the symbol_{i-1} that has the next biggest payment in paytable. The players of a slot machine game may win a monetary award in case a combination of at least two identical symbols occur in the paylines. The symbol that has the biggest payment in these combinations is considered the most expensive symbol. However, the replacement by the RMS operator is permitted only after checking the input array $minNSR$ (minimum number of symbols per reel requested by the game developer), otherwise the next most expensive symbol_{i-1} is considered for potential replacement.

- **RMC :** The RTP increases when the placement of the symbols in the reels leads to minimum winning combinations of symbols (i.e., combinations of two identical symbols only) with a high frequency. The Replace Minimum Combination (RMC) local search operator replaces a symbol_i which participates in a minimum winning combination with a symbol_j that does not participate in such winning combinations (e.g., it participates in winning combinations of three or more identical symbols). Such symbols may participate in combinations with a little larger payment but the frequency of their combinations is

much lower.

3.5.3 VND Metaheuristic Algorithm

The VND solution approach is depicted in Algorithm **??**. Also, Monte-Carlo simulation ($RunSimulation()$) is used for the calculation of the current RTP_{curr} with 100,000 or 1,000,000 separate runs. The two neighborhood structures RMS and RMC are also denoted as N_1 and N_2, respectively. Moreover, the number of different neighborhood structures is denoted as l_{max} and in this work it is equal to two.

Algorithm 10 VND

Input : $RTP_{des}, iteration_{max}, R, minNSR, Reels, l_{max}$
Output: $Reels$
$Reel_{curr} \leftarrow 1$
$RTP_{curr} \leftarrow RunSimulation()$
$Difference \leftarrow RTP_{curr} - RTP_{des}$
repeat
  $l \leftarrow 1$
  repeat
    Select $Reels' \in \mathcal{N}_l(x)$ such that $RTP(Reels') < RTP(Reels)$
    $RTP_{curr} \leftarrow$**RunSimulation()**
    NeighborhoodChange$(Reels, Reels', l)$
    $Reel_{curr} \leftarrow (Reel_{curr} + 1) mod(n)$
  until $l = l_{max}$
until *stopping criterion*

The proposed VND is initialized with a randomized initial symbol distribution per reel (the list of distinct symbols is given by the game developer) and terminates once a maximum number of iterations is performed rather than not finding an improvement. This is due to the fact that, in each iteration a Monte-Carlo simulation might return an approximate value of the RTP and not the exact RTP that would resulted after a (time consuming) full cycle RTP computation. Thus, in some cases although the two local search operators don't make any improvement, a better RTP value can be calculated by the Monte-Carlo simulations. Moreover, each time the neighborhood is changed the search is performed in a different (next) reel. Thus, the VND starts the optimization using the first reel ($Reel_{curr} = 1$) and afterward sequentially applies the local search operators in each one of the n reels.

3.5.4 Illustrative examples

Without loss of generality, let's assume a hypothetical slot machine game with n reels, each one of them containing 12 equally distributed symbols. The symbols are represented as string variables in an array (R) and the paytable is stored as integer values in another array (P).

$$P = \begin{bmatrix} 0 & 2 & 4 & 16 & 64 \\ 0 & 0 & 8 & 32 & 128 \\ 0 & 0 & 8 & 32 & 128 \\ 0 & 0 & 16 & 64 & 512 \end{bmatrix}$$

Following, an example of the $minNSR$ array is presented, where for simplicity we show only one of the columns corresponding only to the first reel. Each index represents how many of $sym0i$ symbols should remain in current reel and not get replaced by other, e.g., $minNSR[2] = 3$ means that $sym02$ cannot occur less than three times in the first reel. This way, it will not be possible to replace $sym02$ with $sym03$.

$$\min NSR^T = \begin{bmatrix} 2 & 3 & 1 & 1 \end{bmatrix}$$

The RMC operator changes the symbols of the first reel as follows:

$$\begin{bmatrix} sym01 \\ sym02 \\ sym02 \\ sym02 \\ sym01 \\ sym01 \\ sym03 \\ sym02 \\ sym01 \\ sym03 \\ sym01 \\ sym04 \\ sym04 \end{bmatrix} \Rightarrow \begin{bmatrix} sym02 \\ sym02 \\ sym02 \\ sym02 \\ sym01 \\ sym01 \\ sym03 \\ sym02 \\ sym01 \\ sym03 \\ sym01 \\ sym04 \\ sym04 \end{bmatrix}$$

The i-row of the pay-table contains the payments of each symbol $sym0i$ (arrays start from one) for combinations of one, two, three, four, and five respectively, e.g., $sym03$'s payment takes place in third row of paytable $P[3]$. The RMC local search operator replaces $sym01$ with $sym02$.

If the RMS operator was applied then $sym04$, i.e., the most expensive symbol according to paytable (combination of five pays 512), will be replaced by $sym03$ which is the next most

expensive, but cheaper than this symbol. So, in the first reel $sym04$ will be replaced by $sym03$ as follows:

$$\begin{bmatrix} sym01 \\ sym02 \\ sym02 \\ sym02 \\ sym01 \\ sym01 \\ sym03 \\ sym02 \\ sym01 \\ sym03 \\ sym01 \\ sym04 \\ sym04 \end{bmatrix} \Rightarrow \begin{bmatrix} sym01 \\ sym02 \\ sym02 \\ sym02 \\ sym01 \\ sym01 \\ sym03 \\ sym02 \\ sym01 \\ sym03 \\ sym01 \\ sym04 \\ sym03 \end{bmatrix}$$

3.6 Experimental Results

All the computational experiments have been done using a Dell Inspiron 5558 laptop with an Intel Core i5-5200U at 2.20 GHz, 8 GB DDR3 RAM at 1600 MHz, and running Microsoft Windows 10 Professional 64-bit. The experiments have been done using three commercial slot machine games developed by the Zeusplay (`https://zeusplay.com`) casino game developer company.

- **Le Mystere Du Prince** (34), a five-reel slot machine game with wild and scatter as special symbols,
- **Amun's Book** (35), a five-reel slot machine game that holds one special symbol called book and acts as a wild and scatter,
- **Wild Charger** (21), a five-reel slot machine game that holds one wild and acts as expanding and sticky at the same time (symbol that developed by Zeusplay).

All the above games have ten paylines (winning lines) and they are all paid from left to right. The smaller winning is two monetary units for a combination of two symbols for each game. The details (e.g., paytable, symbols) of the first slot machine game are presented below:

$$\mathrm{P} = \begin{bmatrix} 0 & 2 & 5 & 25 & 100 \\ 0 & 0 & 5 & 25 & 100 \\ 0 & 0 & 5 & 25 & 100 \\ 0 & 0 & 5 & 25 & 100 \\ 0 & 0 & 10 & 50 & 125 \\ 0 & 0 & 15 & 75 & 250 \\ 0 & 0 & 15 & 75 & 250 \\ 0 & 0 & 20 & 100 & 400 \\ 0 & 2 & 25 & 125 & 750 \\ 0 & 2 & 25 & 125 & 750 \\ 0 & 20 & 50 & 200 & 5000 \\ 0 & 10 & 250 & 2500 & 9000 \end{bmatrix}$$

In this game special symbols wild and scatter are included. The distinct symbols of the game are:

$$S = \begin{bmatrix} sym01 \\ sym02 \\ sym03 \\ sym04 \\ sym05 \\ sym06 \\ sym07 \\ sym08 \\ sym09 \\ sym10 \\ sym11 \\ scatter \\ wild \end{bmatrix}$$

Each $S[i]$ refers to payments of $P[i]$ element for one, two, three, four, and five combinations of this symbol. The only differences with the other two games, in base game, are the paytable and the number of distinct symbols. Amun's Book and Wild Charger consists of ten and eight symbols, respectively. The minNSR array for these two games has been set the same like Le Mystere Du Prince's.

$$minNSR = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ .. & .. & .. & .. & .. \\ .. & .. & .. & .. & .. \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

According to the above $minNSR$ table , at least one symbol on each reel must exist. In the tables below, all the necessary info about the optimization are presented, such as the number of separate runs with Monte-Carlo simulation, the target RTP, the initial RTP and the RTP that was finally obtained. The allowed deviation for the RTP (R variable) was set equal to 3% for the first game, meaning that if $77\% \leq \text{RTP}_{curr} \leq 83\%$, then the algorithm terminates since a good solution was found. The output is a set of reels with the modified symbols and the best RTP that has been found. For the second game and the third game, the R value was set equal to 2% and 1%, respectively. These three games are presented in order to show that the algorithm can indeed optimize games with either large or small difference between the initial RTP and the target RTP. The RTP of the first game was decreased from 715% to 80%, while the RTP of the second game was decreased from 57% to 40%, and the RTP of the third game was decreased from 107% to 53%, approximately. The results per game are presented in Tables **3.2**, **3.3**, and **3.4**, respectively.

Runs	RTP Initial	RTP Target	RTP Obtained
100,000	718.374%	77-83%	82.025%
1,000,000	712.661%	77-83%	78,996%

Table **3.2**: Le Mystere Du Prince

Runs	RTP Initial	RTP Target	RTP Obtained
100,000	56.105%	38-42%	41.277%
1,000,000	57.226%	38-42%	38.758%

Table **3.3**: Wild Charger

Runs	RTP Initial	RTP Target	RTP Obtained
100,000	111.005%	52-54%	52.015%
1,000,000	104.852%	52-54%	52.803%

Table **3.4**: Amun's Book

The required solution time and number of iterations are presented in Table **3.5**.

Game	Simulation Runs	Time (secs)	Iterations
Le Mystere Du Prince	100,000	353	2456
Le Mystere Du Prince	1,000,000	786	501
Wild Charger	100,000	2	2062
Wild Charger	1,000,000	15	350
Amun's Book	100,000	35	110
Amun's Book	1,000,000	223	149

Table **3.5**: Results for the three games

The RTP reduction through the whole process of optimization for each iteration of each game, is presented in the following Figures 2-7. It can be noticed that, Monte-Carlo simulation with 1,000,000 separate runs is more precise and need less iterations to find one acceptable solution rather than the version with 100,000 separate runs. That happens since the larger the sample (runs) is the more exact the results will be. A larger sample always leads to more reliable results.

Also, it can be noticed that Amun's Book RTP needs 110 iterations to finish for 100,000 runs and 149 iterations for 1,000,000. The difference with the other two games that makes these big changes in RTP is due to the fact that, the wild symbol is also a scatter, so once the wild symbol is modified to another simple symbol, the RTP rapidly decreases.

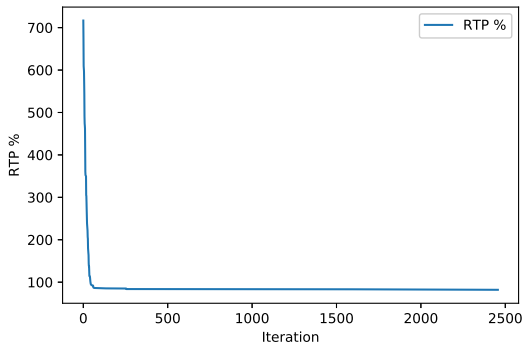


Figure **3.6.9**: Le Mystere Du Prince: 100,000 runs

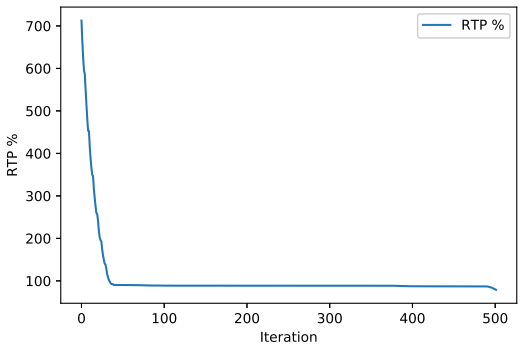


Figure **3.6.10**: Le Mystere Du Prince: 1,000,000 runs

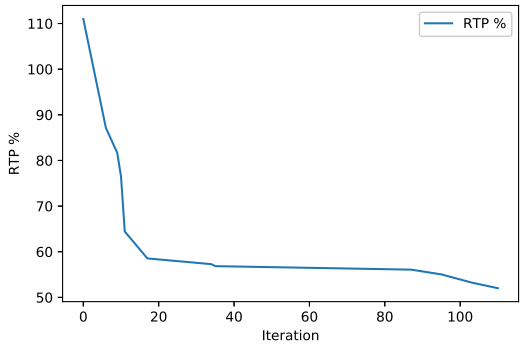


Figure **3.6.11**: Amun's Book: 100,000 runs

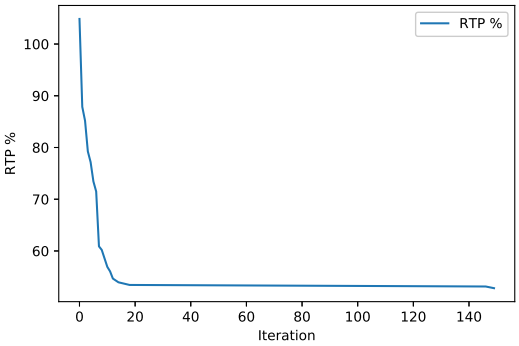


Figure **3.6.12**: Amun's Book: 1,000,000 runs

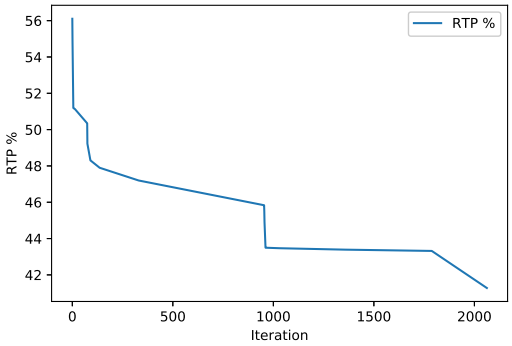


Figure **3.6.13**: Wild Charger: 100,000 runs

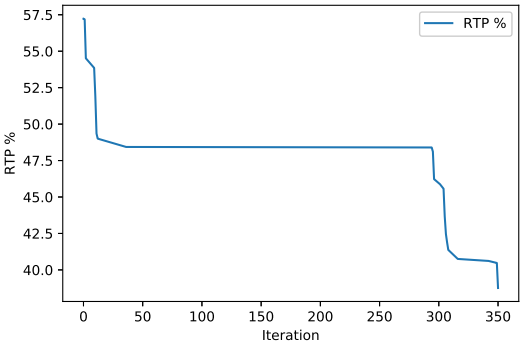


Figure **3.6.14**: Wild Charger: 1,000,000 runs

Creating an exciting, as well as attractive slot game, consists of a variety of elements, depending on the target group. There are several types of players, some of which prefer to risk on a game that might not return much when in base game, but they have a bonus feature that is more likely to return big wins. In order to create such games, a combination of acceptable paytables, hitrates, and bonus feature triggering frequency must be taken in consideration. The most common logic is to have a risky base game, with a low RTP but with recurring small wins, so that the player does not immediately lose his/her money, which triggers a bonus feature more frequently and gives the player the confidence that he is able to win big prizes.

CHAPTER 4

Design, Analysis and Optimization of a slot machine game

4.1 The mathematical approach

The table below will help to calculate the hits of the symbols and probabilities, in order to find the RTP and Volatility of the game.

Wild Sevens						
-	**A**	**B**	**C**	**D**	**E**	**F**
1	**Symbol**	**Reel 1**	**Reel 2**	**Reel 3**	**Reel 4**	**Reel 5**
2	sym01	20	15	20	15	20
3	sym02	15	18	18	18	20
4	sym03	12	12	12	12	12
5	sym04	10	10	10	10	10
6	sym05	9	9	9	9	9
7	sym06	6	6	6	8	6
8	scatter	2	1	2	1	2
9	wild	0	1	1	1	1
10	**Total**	**74**	**72**	**78**	**74**	**80**

Table **4.1**: The reelstrips of the game

The calculations have been done in Microsoft Office Excel and the summary will be presented after the calculations.

	A	B	C	D	E	F
1	**Symbol**	**Reel 1**	**Reel 2**	**Reel 3**	**Reel 4**	**Reel 5**
2	sym01	20	15	20	15	20
3	sym02	15	18	18	18	20
4	sym03	12	12	12	12	12
5	sym04	10	10	10	10	10
6	sym05	9	9	9	9	9
7	sym06	6	6	6	8	6
8	scatter	2	1	2	1	2
9	wild	0	1	1	1	1
10	Total	74	72	78	74	80

Figure **4.1.1**: The Table **4.1** in Microsoft Office Excel

The hits of sym01

- **2 sym01:** (B2+B9)× (C2+C9)×(D10-D2-D9)×E10×F10 = 107,980,800
- **3 sym01:** (B2+B9)× (C2+C9)×(D2+D9)×(E10-E2-E9)×F10 = 31,180,800
- **4 sym01:** (B2+B9)× (C2+C9)×(D2+D9)×(E2+E9)×(F10-F2-F9) = 6,343,680
- **5 sym01:** (B2+B9)× (C2+C9)×(D2+D9)×(E2+E9)×(F2+F9) = 2,257,920

The hits of sym02

- **2 sym02:** (B3+B9)×(C3+C9)×(D10-D3-D9)×E10×F10 = 99,544,800
- **3 sym02:** (B3+B9)×(C3+C9)×(D3+D9)×(E10-E3-E9)×F10 = 23,826,000
- **4 sym02:** (B3+B9)×(C3+C9)×(D3+D9)×(E3+E9)×(F10-F3-F9) = 6,070,215
- **5 sym02:** (B3+B9)×(C3+C9)×(D3+D9)×(E3+E9)×(F3+F9) = 2,160,585

The hits of sym02

- **2 sym02:** (B3+B9)×(C3+C9)×(D10-D3-D9)×E10×F10 = 99,544,800
- **3 sym02:** (B3+B9)×(C3+C9)×(D3+D9)×(E10-E3-E9)×F10 = 23,826,000
- **4 sym02:** (B3+B9)×(C3+C9)×(D3+D9)×(E3+E9)×(F10-F3-F9) = 6,070,215
- **5 sym02:** (B3+B9)×(C3+C9)×(D3+D9)×(E3+E9)×(F3+F9) = 2,160,585

The hits of sym03

- **3 sym03:** (B4+B9)×(C4+C9)×(D4+D9)×(E10-E4-E9)×F10 = 9,896,640
- **4 sym03:** (B4+B9)×(C4+C9)×(D4+D9)×(E4+E9)×(F10-F4-F9) = 1,766,388

- **5 sym03:** (B4+B9)×(C4+C9)×(D4+D9)×(E4+E9)×(F4+F9) = 342,732

The hits of sym04

- **3 sym04:** (B5+B9)×(C5+C9)×(D5+D9)×(E10-E5-E9)×F10 = 6,098,400
- **4 sym04:** (B5+B9)×(C5+C9)×(D5+D9)×(E5+E9)×(F10-F5-F9) = 918,390
- **5 sym04:** (B5+B9)×(C5+C9)×(D5+D9)×(E5+E9)×(F5+F9) = 146,410

The hits of sym05

- **3 sym05:** (B6+B9)×(C6+C9)×(D6+D9)×(E10-E6-E9)×F10 = 4,608,000
- **4 sym05:** (B6+B9)×(C6+C9)×(D6+D9)×(E6+E9)×(F10-F6-F9) = 630,000
- **5 sym05:** (B6+B9)×(C6+C9)×(D6+D9)×(E6+E9)×(F6+F9) = 90,000

The hits of sym06

- **3 sym06:** (B7+B9)×(C7+C9)×(D7+D9)×(E10-E7-E9)×F10 = 1,528,800
- **4 sym06:** (B7+B9)×(C7+C9)×(D7+D9)×(E7+E9)×(F10-F7-F9) = 193,158
- **5 sym06:** (B7+B9)×(C7+C9)×(D7+D9)×(E7+E9)×(F7+F9) = 18,522

The hits of scatter The number 3 is the number of rows. In screen rule, the scatter symbols are multiplied with the number of rows to count the hits (as it refers in 2.2.1) and that is because they are not counted in line rule (with paylines).

- **2 scatter: 79,447,752**
  - B8×3×C8×3×(D10-D8×3)×(E10-E8×3)×(F10-F8×3) = 6,809,184
  - B8×3×(C10-C8×3)×D8×3×(E10-E8×3)×(F10-F8×3) = 13,050,936
  - B8×3×(C10-C8×3)×(D10-D8×3)×E8×3×(F10-F8×3) = 6,617,376
  - B8×3×(C10-C8×3)×(D10-D8×3)×(E10-E8×3)×F8×3 = 12,698,208
  - (B10-B8×3)×C8×3×D8×3×(E10-E8×3)×(F10-F8×3) = 6,430,896
  - (B10-B8×3)×C8×3×(D10-D8×3)×E8×3×(F10-F8×3) = 3,260,736
  - (B10-B8×3)×C8×3×(D10-D8×3)×(E10-E8×3)×F8×3 = 6,257,088
  - (B10-B8×3)×(C10-C8×3)×D8×3×E8×3×(F10-F8×3) = 6,249,744
  - (B10-B8×3)×(C10-C8×3)×D8×3×(E10-E8×3)×F8×3 = 11,992,752
  - (B10-B8×3)×(C10-C8×3)×(D10-D8×3)×E8×3×F8×3 = 6,080,832

- **3 scatter: 5,117,688**
  - B8×3×C8×3×D8×3×(E10-E8×3)×(F10-F8×3) = 567,432
  - B8×3×C8×3×(D10-D8×3)×E8×3×(F10-F8×3) = 287,712
  - B8×3×(C10-C8×3)×D8×3×E8×3×(F10-F8×3) = 551,448
  - (B10-B8×3)×C8×3×D8×3×E8×3×(F10-F8×3) = 271,728
  - B8×3×C8×3×(D10-D8×3)×(E10-E8×3)×F8×3 = 552,096
  - B8×3×(C10-C8×3)×D8×3×(E10-E8×3)×F8×3 = 1,058,184
  - (B10-B8×3)×C8×3×D8×3×(E10-E8×3)×F8×3 = 521,424
  - B8×3×(C10-C8×3)×(D10-D8×3)×E8×3×F8×3 = 536,544
  - (B10-B8×3)×C8×3×(D10-D8×3)×E8×3×F8×3 = 264,384
  - (B10-B8×3)×(C10-C8×3)×D8×3×E8×3×F8×3 = 506,736
- **4 scatter: 160,056**
  - B8×3×C8×3×D8×3×E8×3×(F10-F8×3) = 23,976
  - B8×3×C8×3×D8×3×(E10-E8×3)×F8×3 = 46,008
  - B8×3×C8×3×(D10-D8×3)×E8×3×F8×3 = 23,328
  - B8×3×(C10-C8×3)×D8×3×E8×3×F8×3 = 44,712
  - (B10-B8×3)×C8×3×D8×3×E8×3×F8×3 = 22,032
- **5 scatter: 1,944**
  - B8×3×C8×3×D8×3×E8×3×F8×3 = 1,944

After the calculation of hits, probabilities and other parameters can, easily, be calculated. In above screenshots, the summary of the game is presented.

Symbol	Reel 1	Reel 2	Reel 3	Reel 4	Reel 5
sym01	20	15	20	15	20
sym02	15	18	18	18	20
sym03	12	12	12	12	12
sym04	10	10	10	10	10
sym05	9	9	9	9	9
sym06	6	6	6	8	6
scatter	2	1	2	1	2
wild	0	1	1	1	1
Total	74	72	78	74	80

Combination	Hits	Payment	Probability	P(Win-RTP)^2	Total Payments
2 sym01	107.980.800	2	0,0438900	0,0481	215.961.600
3 sym01	31.180.800	4	0,0126738	0,1177	124.723.200
4 sym01	6.343.680	20	0,0025785	0,9355	126.873.600
5 sym01	2.257.920	40	0,0009178	1,3993	90.316.800
2 sym02	99.544.800	2	0,0404611	0,0444	199.089.600
3 sym02	23.826.000	4	0,0096844	0,0899	95.304.000
4 sym02	6.070.215	20	0,0024673	0,8951	121.404.300
5 sym02	2.160.585	40	0,0008782	1,3390	86.423.400
3 sym03	9.896.640	5	0,0040226	0,0659	49.483.200
4 sym03	1.766.388	75	0,0007180	3,9366	132.479.100
5 sym03	342.732	150	0,0001393	3,0947	51.409.800
3 sym04	6.098.400	5	0,0024788	0,0406	30.492.000
4 sym04	918.390	75	0,0003733	2,0467	68.879.250
5 sym04	146.410	150	0,0000595	1,3220	21.961.500
3 sym05	4.608.000	12	0,0018730	0,2286	55.296.000
4 sym05	630.000	120	0,0002561	3,6291	75.600.000
5 sym05	90.000	200	0,0000366	1,4494	18.000.000
3 sym06	1.528.800	12	0,0006214	0,0758	18.345.600
4 sym06	193.158	120	0,0000785	1,1127	23.178.960
5 sym06	18.522	200	0,0000075	0,2983	3.704.400
2 scatter	79.447.752	5	0,0322925	0,5290	397.238.760
3 scatter	5.117.688	50	0,0020801	5,0041	255.884.400
4 scatter	160.056	500	0,0000651	16,2022	80.028.000
5 scatter	1.944	1000	0,0000008	0,7887	1.944.000
Total	390.329.680	2.811	0,1587	44,6933	2.344.021.470

Summary

Statistic	Data
RTP	95,28%
Hitrate	6,3030
Variance	44,6933
Standard Deviation	6,6853
Volatility	13,1032
Cycle	2.460.257.280

*The payments of sym_scatt is x10 (totalbet) because we have 10 paylines

**Hitrate shows how many times you get paid e.g. 9 means that if you play 9 times, statistically, you get a payment

***Volatility shows how hard is to get a payment

****Use commas (,) to declare the decimals places

Figure **4.1.2**: The summary of the game in Microsoft Office Excel (1)

2/5 sym_scatt					Hits
scatter	scatter	x	x	x	6.809.184
scatter	x	scatter	x	x	13.050.936
scatter	x	x	scatter	x	6.617.376
scatter	x	x	x	scatter	12.698.208
x	scatter	scatter	x	x	6.430.896
x	scatter	x	scatter	x	3.260.736
x	scatter	x	x	scatter	6.257.088
x	x	scatter	scatter	x	6.249.744
x	x	scatter	x	scatter	11.992.752
x	x	x	scatter	scatter	6.080.832
				Total	79.447.752
3/5 sym_scatt					**Hits**
scatter	scatter	scatter	x	x	567.432
scatter	scatter	x	scatter	x	287.712
scatter	x	scatter	scatter	x	551.448
x	scatter	scatter	scatter	x	271.728
scatter	scatter	x	x	scatter	552.096
scatter	x	scatter	x	scatter	1.058.184
x	scatter	scatter	x	scatter	521.424
scatter	x	x	scatter	scatter	536.544
x	scatter	x	scatter	scatter	264.384
x	x	scatter	scatter	scatter	506.736
				Total	5.117.688
4/5 sym_scatt					**Hits**
scatter	scatter	scatter	scatter	x	23.976
scatter	scatter	scatter	x	scatter	46.008
scatter	scatter	x	scatter	scatter	23.328
scatter	x	scatter	scatter	scatter	44.712
x	scatter	scatter	scatter	scatter	22.032
				Total	160.056
5/5 sym_scatt					**Hits**
scatter	scatter	scatter	scatter	scatter	1.944
				Total	1.944

Figure **4.1.3**: The summary of the game in Microsoft Office Excel (2)

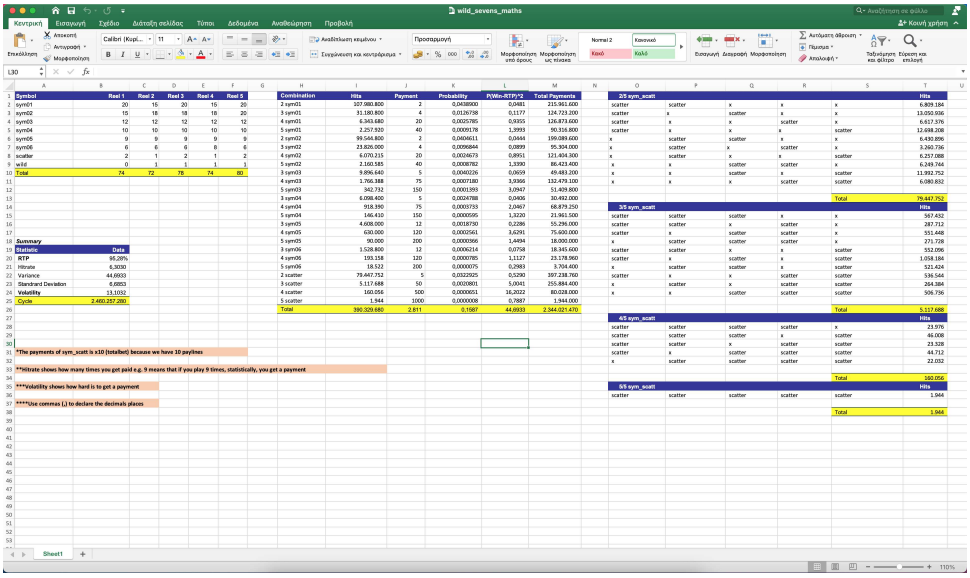


Figure **4.1.4**: The summary of the game in Microsoft Office Excel

It can be noticed that RTP = 95.28% or 0.9528, so that means, statistically, if a player bets 1 coin and plays for 10000 spins, that means 10000 credits in total. The game will return to him 9528 credits. In addition, volatility = 13.1 means that there is a medium volatility game. The hitrate = 6.3 variable means that every 6 spins, statistically, a payment is occured.

4.1.1 The programming approach : Monte Carlo simulation

The simulation that is about to be used is Monte Carlo, which that the game will run 10 million random simulations. In previous subsection, the Full Cycle (mathematical approach) method has been used in order to extract the exact RTP, Volatility and Hitrate. Monte Carlo is an approach to relate hypothetical slot machine gambling behavior to the statistical characteristics of the slot machines themselves. The source code of the game that has been analyzed (developed in Java) is open and the Java implementation can be found in Github (36). Also, another implementation in Python can be found in Github (37) of a slot machine game which name is 'Rolling Bars' and the source code can be found in 4.2 as screenshots. It, also, contains only simple symbols and a scatter symbol as special. Its mathematical analysis in Microsoft Excel can be found in this repository. The behavior of the game 'Wild Sevens' will be examined for 10 million runs for 10 times and the results and diagrams will be written and presented below. All the computational experiments for Monte Carlo simulations have been done using a MacBook Pro (Retina, 15-inch, Late 2013) with a 2.0GHz quad-core Intel Core i7 processor (Turbo Boost up to 3.2GHz) with 6MB shared L3 cache, 8GB DDR3L RAM at 1600MHz and running macOS Big Sur (11.0.1 version).

Wild Sevens : 10,000,000 Monte Carlo simulations				
Trial	**RTP**	**Volatility**	**Hitrate**	**Time**
1st	0.9531	3.302	1.77	5 seconds
2nd	0.9506	3.292	1.77	6 seconds
3rd	0.9504	3.293	1.77	6 seconds
4th	0.9512	3.297	1.77	6 seconds
5th	0.9561	3.308	1.77	5 seconds
6th	0.9506	3.292	1.77	5 seconds
7th	0.9522	3.302	1.77	6 seconds
8th	0.9531	3.301	1.77	6 seconds
9th	0.9514	3.300	1.77	5 seconds
10th	0.9539	3.304	1.77	5 seconds
Average	**0.9522**	**3.299**	**1.77**	**5.5 sec(s)**

Table **4.2**: Monte Carlo : 10 simulations

The difference is very small because in Monte Carlo has been chosen as number of spins 10 million. If 100,00 or 1 million spins were chosen, the RTP would had not been so precise.The more paylines are added in the game, the less Volatility and Hitrate occurs and that's because more paylines means more prizes.Conclusion for Volatility and Hitrate for each case: The details for each simulation (e.g. hits, hitrate) are in **Appendices**. In below screenshots is presented the source code that is uploaded in Github repository (36) for 'Wild Sevens'. The IDE that has been used is Eclipse IDE and the programming language is Java.

Main.java | Symbol.java | SlotMachine.java | Reel.java | Parser.java | Coordinates.java | WindowPrinter.java | SlotWindow.java

```java
package com.arseniumn.slots.model;
import java.util.ArrayList;

public class SlotWindow {

    private int m, n, totalBet, lineBet;
    private Symbol[][] W;
    private int[] RS;
    private int[][] P = {
                          {0,2,4,20,40},
                          {0,2,4,20,40},
                          {0,0,5,75,150},
                          {0,0,5,75,150},
                          {0,0,12,120,200},
                          {0,0,12,120,200},
                          {0,5,50,500,1000}
                        };
    private int[][] L = {
                          {0,0,0,0,0},
                          {1,1,1,1,1},
                          {2,2,2,2,2},
                          {0,1,2,1,0},
                          {2,1,0,1,2},
                          {0,0,1,0,0},
                          {2,2,1,2,2},
                          {1,0,0,0,1},
                          {1,2,2,2,1},
                          {2,1,1,1,2}
                        };
    private ArrayList<Coordinates> coords = new ArrayList<Coordinates>();
    private Random random = new Random();
    private Symbol wild;

    public SlotWindow(int nReels, int nRows, int totalBet, int aLineBet){

        // The m rows
        this.m = nRows;

        // The n columns-reels
        this.n = nReels;

        // The totalbet
        this.totalBet = totalBet;

        // The linebet
        this.lineBet = aLineBet;

        // The 3x5 random window instance
        W = new Symbol[3][5];

        // The Reels' Stops
        RS = new int[]{0,0,0,0,0};
    }
```

Figure **4.1.5**: The declarations of the slot window

In above source code, the variables that have been mentioned in Section 2.3 are declared and the constructor of the object 'Slot Window' (*SlotWindow.java*) is initialized by creating this in *Main.java* class file and calling it (as it can be noticed in figure below). The 2D-array W is initialized which holds the symbols of the random stops, the RS which holds all the random stops that have extracted from the reels that the slot machine (*Slotmachine.java*) object contains. The 2D-matrix P, also, is initialized which represents the paytable of the game and the L array that holds all the payline of the game. In addition, the necessary variables are passed through the constructor of SlotWindow object to initialize the matrices and the variables that have been declared. The arraylist *coords* holds all the coordinates of the wild symbols on each window instance that is randomly generated in order to use the methods *saveCoordinates()* and *retrieveWildSymbols()* that have been already mentioned in 1 and 2 algorithms.

Figure **4.1.6**: The saveWildCoordinates() method

Figure **4.1.7**: The retriveWildsymbols() method

In *saveWildCoordinates()* method, the program checks if there is a wild symbol on each pair of indices (i, j) in slot window instance and then it creates a new Coordinates object (*Coordinates.java*) that holds the indices that the wil symbol takes hold. After that, the program stores the reference of the wild symbol object to *wild* variable that has been declared in the beginning of the code dec. After the prize occured on a payline, the program should replace the symbols that the wild substituted in the window instance and that's what the *retrieveWildSymbols()* method does. It iterates through all the objects that are type of Coordinates and are stored in an arraylist type of Coordinates and it sets in the specific (i, j) element of the window instance (W) with the reference of wild symbol.

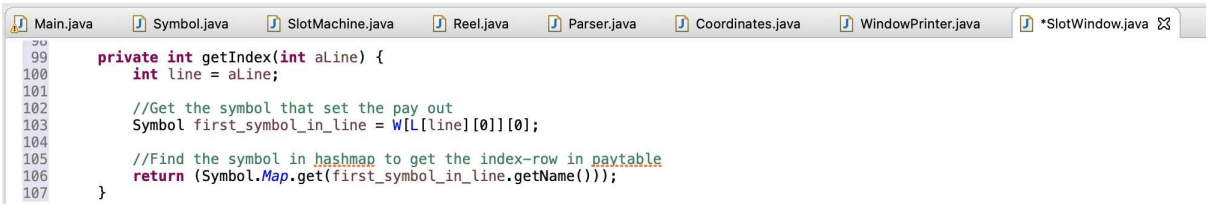


Figure **4.1.8**: The retriveWildsymbols() method

The method *getIndex()* refers to the lines of code, in Algorithm 1, where the P matrix gets the index of the symbol on the 1st reel from Map and it becomes the i index of it. The j index is the *aPair* variable.

Main.java | Symbol.java | SlotMachine.java | Reel.java | Parser.java | Coordinates.java | WindowPrinter.java | SlotWindow.java

```java
package com.arseniumn.slots;
import java.util.Scanner;

public class Main {

    // If the user changes the NUMBER_OF_PAYLINES variable, he/she must change the matrix P[][] in SlotWindow Object and
    // add more or remove paylines
    private static final int NUMBER_OF_PAYLINES = 10;
    private static final int LINEBET = 1;
    private static final int TOTAL_BET = LINEBET * NUMBER_OF_PAYLINES;

    public static void main(String[] args) {
        //Slot machine model creation
        SlotMachine slot_machine = new SlotMachine();

        //Create parser and insert the data
        Parser parser = new Parser();
        parser.parse(slot_machine);

        //Create the slot window where the user plays (reels,rows)
        SlotWindow slot_window = new SlotWindow(5, 3, TOTAL_BET, LINEBET);

        long sum = 0;
        double temp_reward=0;
        int COINS = 100000;
        boolean choice = true;
        Scanner in = new Scanner(System.in);

        while(choice) {
            System.out.println("==================================");
            System.out.println("|| Menu || Welcome to Wild Sevens ||");
            System.out.println("==================================");
            System.out.println("What would you like to do?");
            System.out.println("1.Spin & Win!");
            System.out.println("2.Monte Carlo Simulation");
            System.out.println("3.Exit");
            int mode = in.nextInt();
            //Spin and Win
            if(mode==1) {
                    COINS -= TOTAL_BET;

                    // Play the game
                    slot_window.generateStops(slot_machine);
                    temp_reward = (long) slot_window.runSimulation(COINS);

                    // Add the temp_reward
                    COINS += temp_reward;
            }
            // Monte Carlo simulation
            else if(mode==2) {
                    System.out.print("Give the number of steps:");
                    long NUMBER_OF_STEPS = in.nextLong();
                    for(int i=0; i<NUMBER_OF_STEPS; i++) {
                        slot_window.generateStops(slot_machine);
                        temp_reward = (long) slot_window.runSimulation(-1);
                        sum += temp_reward;
                    }
                    System.out.println("RTP = "+(double) sum/(NUMBER_OF_STEPS*NUMBER_OF_PAYLINES) * 100+"%");
                    sum =0;
            }
            else if(mode==3) {
                    choice = false;
                    System.out.println("See ya!");
            }
            else
                System.out.println("Not a valid choice, please try again!");
        }
    }
}
```

Figure **4.1.9**: The main class

In *Main* class the user has 3 options, the first one is to play 1 game (spin), the second is to give as input the number of simulations for Monte Carlo that he wants and extract the RTP of the game. The third one is to exit the program. The three (3) constants that have been declared are used in order to extract the right prize for the player or to substract the bet from the total available coins of the player. In *main* method the slot machine object is created (*SlotMachine.java*), after this the parser object (*Parser.java*) is called in order to get the input from a *.csv* file which holds all the distinct symbols in the first line and the reels of the slot machine game in next lines. After the slot machine, parser, symbol (*Symbol.java*) and

reel (*Reel.java*) objects creation, the program asks from user (**input**) the mode that he wants in order to do the right **proccess** and extract the desired **output**.

```
110    private double getOnPaylinePrize() {
111        double total_payline_prize = 0;
112        for(int i=0; i<L.length; i++) {
113            if(!W[L[i][0]][0].getType().equalsIgnoreCase("Scatter")) {
114                int aPair = 0;
115                for(int j=0; j<this.n-1; j++) {
116                    Symbol current_symbol = W[L[i][j]][j];
117                    Symbol next_symbol = W[L[i][j+1]][j+1];
118                    if(current_symbol.getName().equalsIgnoreCase(next_symbol.getName()) || next_symbol.getType().equalsIgnoreCase("Wild")) {
119                        W[L[i][j+1]][j+1] = current_symbol;
120                        aPair++;
121                    }
122                    else
123                        break;
124                }
125
126                // Get the payment for the n-combination of specific symbol
127                if(aPair>0)
128                    total_payline_prize += (P[getIndex(i)][aPair]*this.lineBet);
129
130                // Retrieve wild symbols on window instance
131                retrieveWildSymbols();
132            }
133        }
134        return total_payline_prize;
135    }
```

Figure **4.1.10**: The method for line rule prize

In Figure 4.1.1 is the Java implementation of the Algorithm 1 in 2.3 Section.

```
152    private double getScreenRuleCombinationPrize() {
153        int times = 1, aPair = 0;
154        Symbol temp_scatter = null;
155        int[] C = new int[] {0,0,0,0,0};
156        for(int i=0; i<this.n; i++) {
157            for(int j=0; j<this.m; j++) {
158                if((W[j][i].getType().equalsIgnoreCase("Scatter"))) {
159                    temp_scatter = W[j][i];
160                    C[i]++;
161                }
162            }
163            if(C[i]>0) {
164                times *= C[i];
165                aPair++;
166            }
167        }
168        if(aPair>0) return (P[Symbol.Map.get(temp_scatter.getName())][aPair-1]*times*totalBet);
169        else return 0;
170    }
```

Figure **4.1.11**: The method for screen rule prize

In Figure 4.1.1 is the Java implementation of the Algorithm 2 in 2.3 Section. In order to achieve the main goal of the process, which is the RTP optimization, a commercial software for slot machine game development will be used, called 'SlotWizard' (38). This work has been funded by the company ZeusPlay ((3)) which is a subsidiary company of ELICON SMPC ((38))(Research Committee of the University of Macedonia). Also, based on our signed contract, the source code of our implemented methodology, belongs to ZeusPlay. Thus, the nature of the data is commercial and ZeusPlay has their rights. This is the sole reason why source code of RTP Optimization cannot be released. The input that the software accepts is a **.sw** file and the form is

WildSevens-Opt.sw

```
BEGIN base_game
        instance 5,3;
        cycle 10000000;
        bet 1;

        Paylines {
                lines [0,0,0,0,0],
                      [1,1,1,1,1],
                      [2,2,2,2,2],
                      [0,1,2,1,0],
                      [2,1,0,1,2],
                      [0,0,1,0,0],
                      [2,2,1,2,2],
                      [1,0,0,0,1],
                      [1,2,2,2,1],
                      [2,1,1,1,2]
        };

        Symbol Name, Type, Rule, Substitutes {
                sym01, Simple, left, Yes
                sym02, Simple, left, Yes
                sym03, Simple, left, Yes
                sym04, Simple, left, Yes
                sym05, Simple, left, Yes
                sym06, Simple, left, Yes
                scatter, Scatter, any, No
                wild, Wild, left, Yes
        };

        Paytable {
                0,2,4,20,40
                0,2,4,20,40
                0,0,5,75,150
                0,0,5,75,150
                0,0,12,120,200
                0,0,12,120,200
                0,5,50,500,1000
                0,0,0,0,0
        };

        Reelstrips {
                Reelstripe1:
[sym04,scatter,sym01,sym03,sym04,sym05,sym05,sym04,sym03,sym04,sym02,sym02,sym01,sym01,sym06,sym05,sym03,sym02,sym06,sym02,sym06,sym06,sym03,sym01,sym01,sym05,sym03,sym04,sym01,sym01,sym04,sym01,sym01,sym02,sym02,sym01,sym04,sym03,sym01,sym02,sym05,sym02,scatter,sym01,sym02,sym03,sym01,sym02,sym04,sym05,sym01,sym03,sym06,sym01,sym03,sym02,sym05,sym01,sym03,sym01,sym01,sym06,sym05,sym01,sym04,sym04,sym02,sym03,sym02,sym02,sym02,sym03,sym01,sym05]

                Reelstripe2:
[sym04,sym06,sym02,sym04,sym02,sym03,sym04,sym02,sym04,sym01,sym01,sym05,sym02,sym02,sym02,sym01,sym01,sym03,sym03,sym02,sym05,sym06,sym03,scatter,sym02,sym05,sym02,sym04,sym05,sym01,sym01,sym03,sym05,sym02,sym02,sym04,sym04,sym03,wild,sym01,sym03,sym01,sym06,sym05,sym01,sym02,sym02,sym04,sym02,sym02,sym04,sym02,sym01,sym03,sym01,sym03,sym01,sym06,sym01,sym01,sym05,sym03,sym02,sym02,sym05,sym04,sym06,sym03,sym05,sym03,sym01,sym06]

                Reelstripe3:
[scatter,sym01,sym01,sym06,sym04,sym02,sym03,sym02,sym04,sym05,scatter,sym06,sym04,sym04,sym03,sym01,sym02,sym02,sym01,sym03,sym01,sym05,sym03,wild,sym01,sym06,sym02,sym05,sym02,sym01,sym02,sym06,sym01,sym02,sym03,sym05,sym03,sym05,sym06,sym05,sym05,sym04,sym04,sym01,sym01,sym02,sym04,sym03,sym01,sym04,sym02,sym01,sym06,sym02,sym03,sym05,sym03,sym02,sym01,sym03,sym02,sym05,sym02,sym01,sym01,sym02,sym01,sym01,sym02,sym03,sym04,sym02,sym01,sym03,sym01,sym02,sym04,sym01]

                Reelstripe4:
[sym06,sym02,sym02,sym02,sym04,sym02,sym01,sym04,sym05,sym05,sym04,sym01,sym02,sym02,sym05,sym01,sym06,sym06,sym02,sym03,sym03,sym04,sym02,sym06,sym05,sym03,sym01,sym06,sym01,sym05,sym03,sym05,sym02,sym04,sym03,sym05,sym02,sym01,sym02,sym04,sym02,sym01,sym03,sym01,sym02,sym01,sym02,sym03,sym01,sym03,sym03,sym02,sym01,sym03,sym01,sym06,sym01,sym05,sym02,sym01,sym03,wild,sym01,sym02,sym06,sym04,sym04,sym02,sym04,scatter,sym04,sym03,sym05,sym06]

                Reelstripe5:
[sym03,sym01,sym05,sym02,sym03,sym03,sym01,sym03,sym01,sym03,sym05,sym02,sym04,sym04,sym02,sym01,sym02,sym02,sym02,sym06,sym04,scatter,sym01,sym04,sym01,sym06,scatter,sym01,sym02,sym02,sym06,sym02,sym02,sym01,sym01,sym02,sym01,sym03,sym01,sym03,sym02,sym03,sym05,sym04,sym02,sym05,sym01,sym04,sym02,sym04,sym02,wild,sym04,sym03,sym05,sym03,sym01,sym01,sym03,sym04,sym02,sym01,sym05,sym02,sym05,sym02,sym01,sym06,sym01,sym02,sym06,sym02,sym05,sym01,sym04,sym01,sym06,sym03,sym05,sym01]

        };

        Optimization {
                target_RTP 0.85
                acceptable_range 0.03
                iterations 10000
                export_range 0.1
                minNSR1 : [1,1,1,1,1,1,1,1,1]
                minNSR2 : [1,1,1,1,1,1,1,1,1]
                minNSR3 : [1,1,1,1,1,1,1,1,1]
                minNSR4 : [1,1,1,1,1,1,1,1,1]
                minNSR5 : [1,1,1,1,1,1,1,1,1]
        };
END base_game
```

Figure **4.1.12**: The slot wizard file form

The main goal of RTP Optimization was to minimize the RTP of *Wild Sevens* game from 95% to 85%, 72% and . The optimal solution was found by running 10,000,000, 1,000,000 and 100,000 Monte Carlo simulations for RTP Calculation, in 5 iterations in about 142 seconds (2 min(s) and 22 sec(s)). The computational experiments for RTP Calculation and RTP Optimization have been done using a MacBook Pro (Retina, 15-inch, Late 2013) with a 2.0GHz quad-core Intel Core i7 processor (Turbo Boost up to 3.2GHz) with 6MB shared L3 cache, 8GB DDR3L RAM at 1600MHz and running macOS Big Sur (11.0.1 version). The RTP decreament is presented in figure below

Runs	RTP Initial	RTP Target	Acceptable Range	RTP Obtained	Iterations	Time (s)
100,000	94.20%	80%	2%	81.4%	6	30
1,000,000	95.97%	80%	2%	81.35%	6	24
10,000,000	95.46%	80%	2%	81.70%	16	407

Table **4.3**: Le Mystere Du Prince

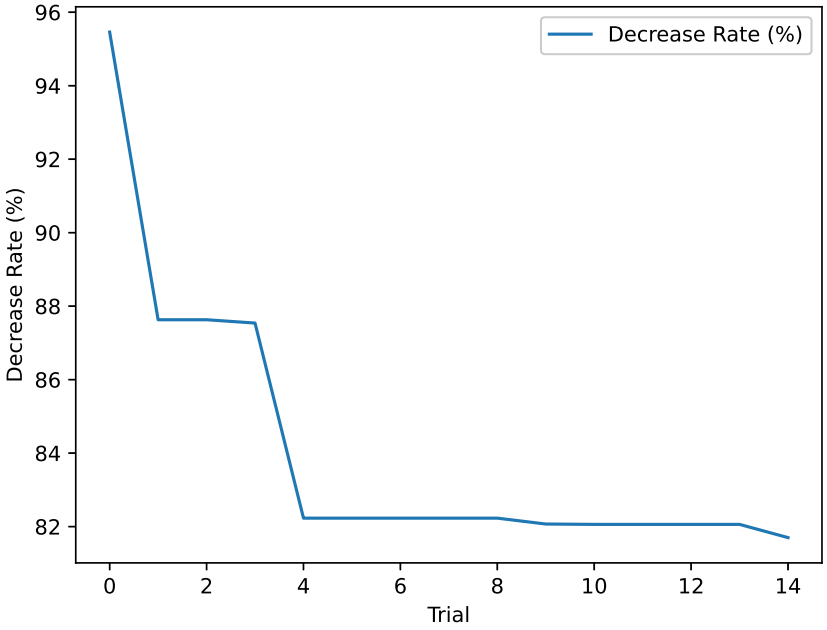


Figure **4.1.13**: RTP Optimization : 10,000,000 runs

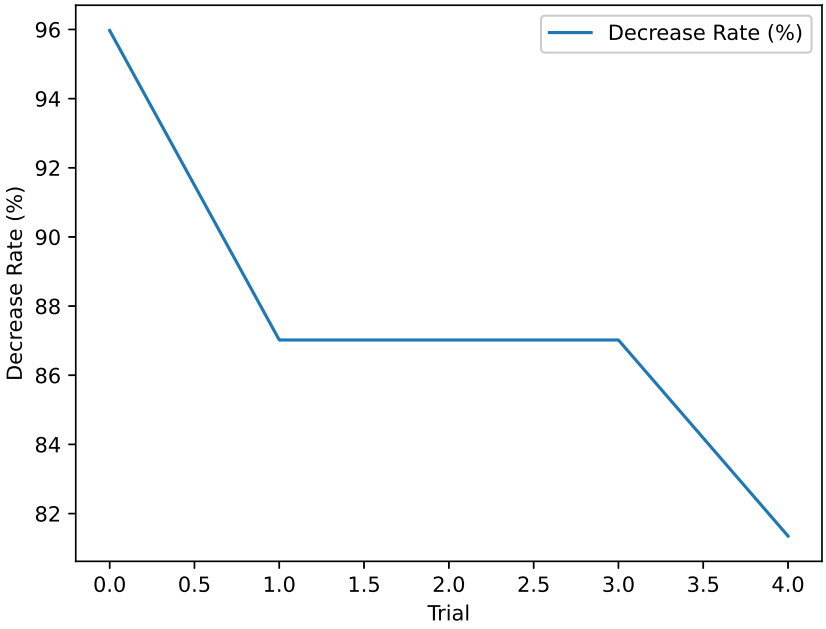


Figure **4.1.14**: RTP Optimization : 1,000,000 runs

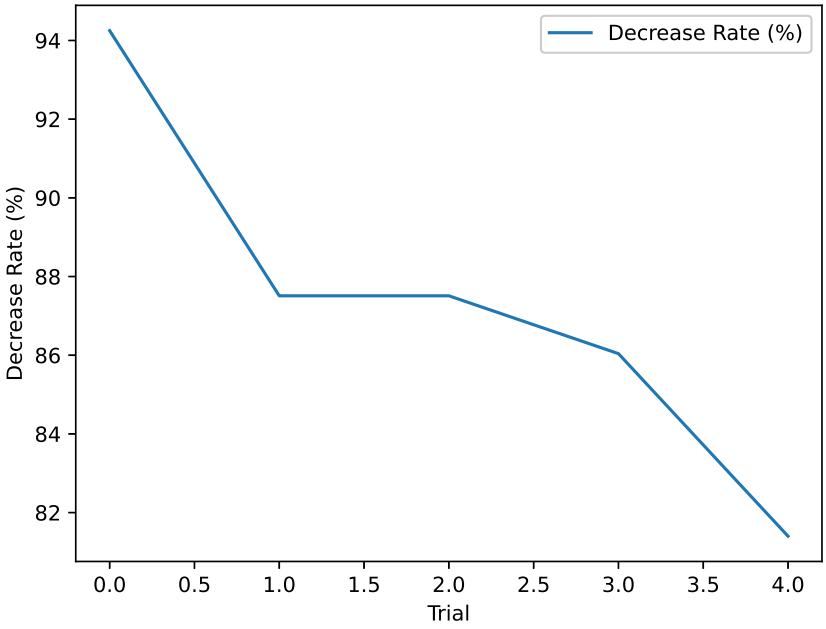


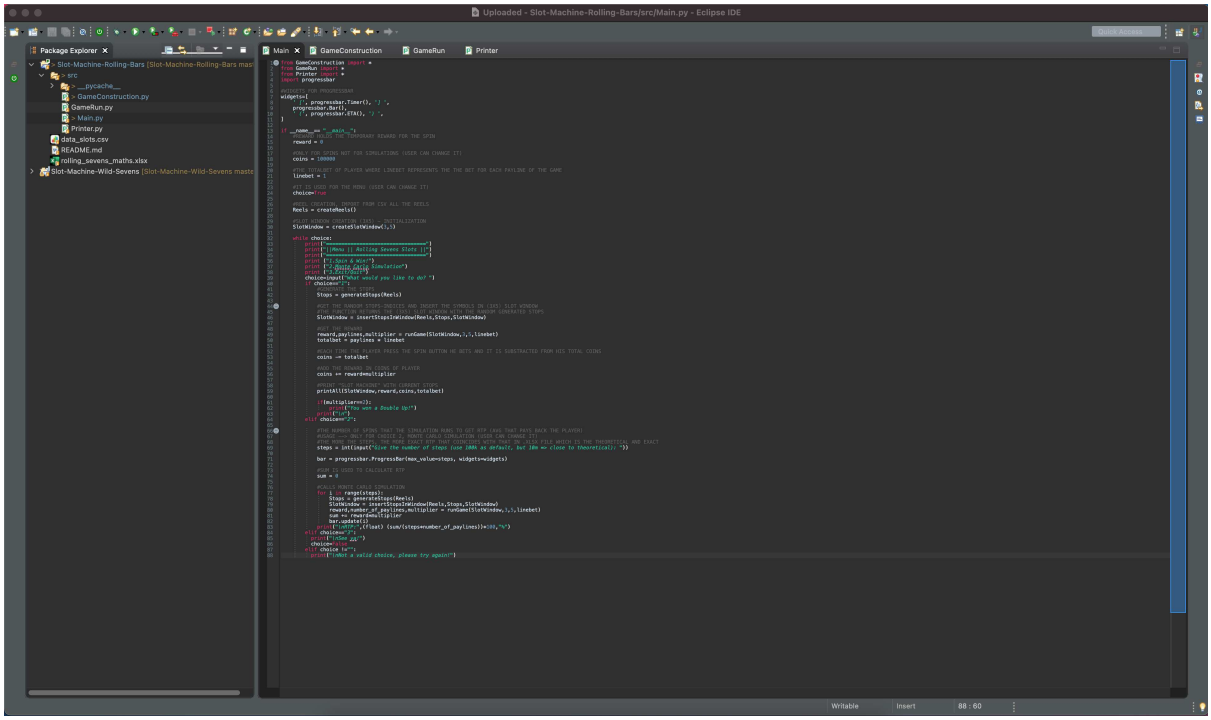
Figure **4.1.15**: RTP Optimization : 100,000 runs

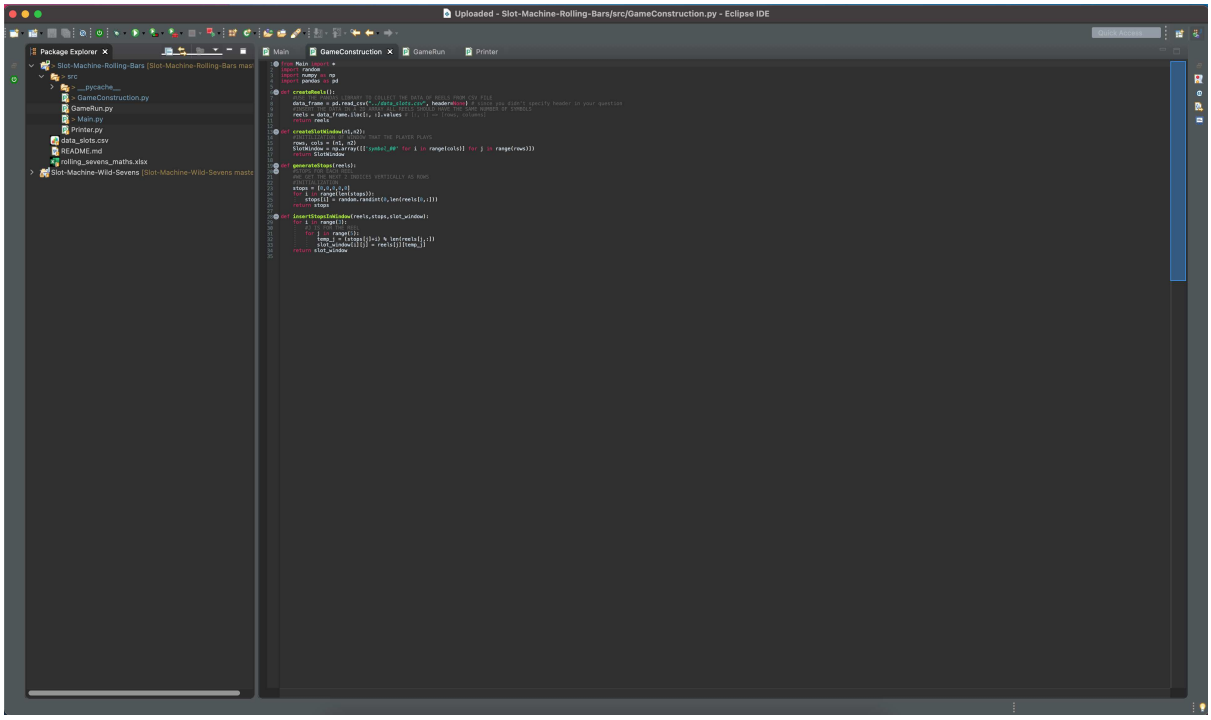
4.2 Conclusions and Future Work

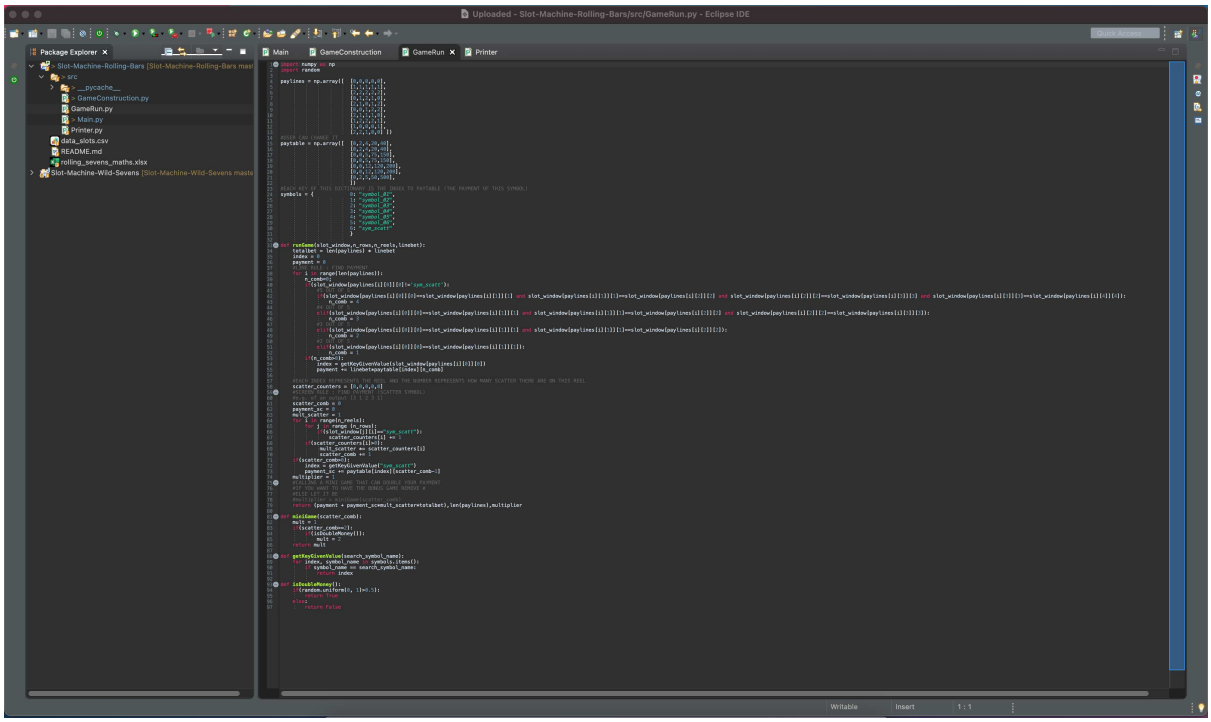
The experiments showed that, the use of VNS was very efficient in developing slot games from scratch, where the symbol distribution is not uniform. As shown in the Figures of the previous Section, the required number of iterations of the proposed method depends on the initial difference between the initial and target RTP. Also, in cases where a symbol contains two types of special symbols (e.g., such as in the second game where the "scatter" symbol is also a "wild" symbol) a large RTP decrease can be achieved by converting this symbol to another.

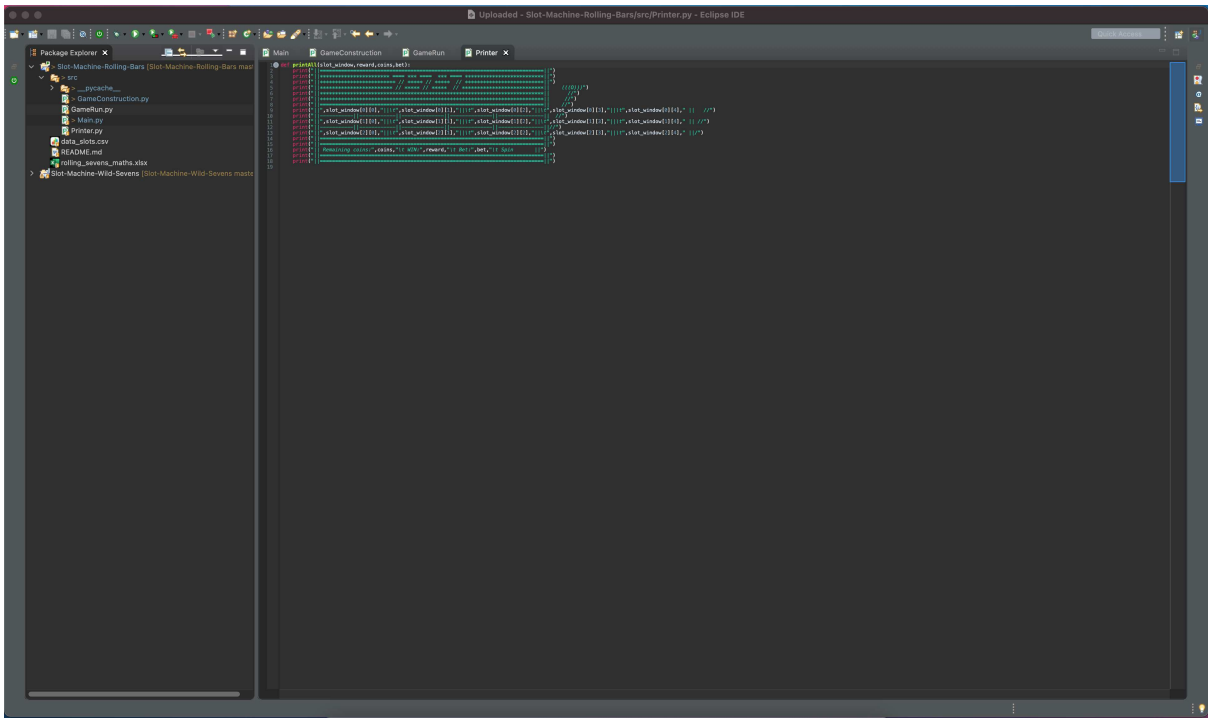
As the complexity of games increases, the required solution time increases too and this fact shows the need for speeding up the whole process. As future research, an implementation of much more complex games RTP can be proposed using parallel CPU and GPU computing techniques to decrease the computational time for finding an acceptable (RTP) solution. Furthermore, the proposed algorithm is a single-criterion approach for the RTP optimization. Another research idea is to develop a multi-criteria method for optimizing RTP, volatility, and hit-rate (i.e., how frequently a player is expected to stop on a winning combination).

Appendices


Uploaded - Slot-Machine-Rolling-Bars/src/Main.py - Eclipse IDE
Package Explorer
Main
GameConstruction
GameRun
Printer
Writable
Insert
88 : 60


Uploaded - Slot-Machine-Rolling-Bars/src/GameConstruction.py - Eclipse IDE
Package Explorer
Main
GameConstruction
GameRun
Printer


Uploaded - Slot-Machine-Rolling-Bars/src/GameRun.py - Eclipse IDE
Package Explorer
Main
GameConstruction
GameRun
Printer
Slot-Machine-Rolling-Bars [Slot-Machine-Rolling-Bars master]
src
__pycache__
GameConstruction.py
GameRun.py
Main.py
Printer.py
data_slots.csv
README.md
rolling_sevens_maths.xlsx
Slot-Machine-Wild-Sevens [Slot-Machine-Wild-Sevens master]
Writable
Insert
1 : 1


Uploaded - Slot-Machine-Rolling-Bars/src/Printer.py - Eclipse IDE
Package Explorer
Main
GameConstruction
GameRun
Printer
Slot-Machine-Rolling-Bars [Slot-Machine-Rolling-Bars master]
src
__pycache__
GameConstruction.py
GameRun.py
Main.py
Printer.py
data_slots.csv
README.md
rolling_sevens_maths.xlsx
Slot-Machine-Wild-Sevens [Slot-Machine-Wild-Sevens master]

	A	B	C	D	E	F	G	H
1								
2	**Game**	**Base RTP**	**Final RTP**	**Hit Rate**	**Base Volatility**	**Max Win**	**Line Bet**	**Cycle**
3	base_game	95,319%		1,777	3,302	10002,000	0,000	10000000
4								
5	**Symbol**	**Payment**	**Hits**	**Probability**				
6	2-sym01	2	4387138	0,43871380				
7	3-sym01	4	1267918	0,12679180				
8	4-sym01	20	258396	0,02583960				
9	5-sym01	40	91963	0,00919630				
10	2-sym02	2	4049039	0,40490390				
11	3-sym02	4	968480	0,09684800				
12	4-sym02	20	246963	0,02469630				
13	5-sym02	40	88441	0,00884410				
14	3-sym03	5	402417	0,04024170				
15	4-sym03	75	72003	0,00720030				
16	5-sym03	150	13902	0,00139020				
17	3-sym04	5	247144	0,02471440				
18	4-sym04	75	37090	0,00370900				
19	5-sym04	150	5974	0,00059740				
20	3-sym05	12	187435	0,01874350				
21	4-sym05	120	25598	0,00255980				
22	5-sym05	200	3634	0,00036340				
23	3-sym06	12	61870	0,00618700				
24	4-sym06	120	7936	0,00079360				
25	5-sym06	200	729	0,00007290				
26	2-scatter	5	322185	0,03221850				
27	3-scatter	50	20917	0,00209170				
28	4-scatter	500	644	0,00006440				
29	5-scatter	1000	9	0,00000090				
30								
31	**Payment**	**Hits**	**Probability**					
32	10000	9	0,00000090					
33	5005	1	0,00000010					
34	5004	9	0,00000090					
35	5002	44	0,00000440					
36	5000	590	0,00005900					
37	575	2	0,00000020					
38	540	10	0,00000100					
39	520	28	0,00000280					
40	512	13	0,00000130					
41	505	46	0,00000460					
42	504	369	0,00003690					
43	502	1573	0,00015730					
44	500	18876	0,00188760					
45	200	448	0,00004480					
46	170	36	0,00000360					
47	150	1925	0,00019250					
48	125	200	0,00002000					
49	120	3208	0,00032080					
50	90	346	0,00003460					
51	75	10700	0,00107000					
52	70	862	0,00008620					
53	62	367	0,00003670					
54	55	1115	0,00011150					
55	54	6032	0,00060320					
56	52	25998	0,00259980					
57	50	287218	0,02872180					
58	40	17669	0,00176690					
59	20	49813	0,00498130					
60	12	24484	0,00244840					
61	5	63623	0,00636230					
62	4	216969	0,02169690					
63	2	815410	0,08154100					
64	0	8452007	0,84520070					

	A	B	C	D	E	F	G	H
1								
2	**Game**	**Base RTP**	**Final RTP**	**Hit Rate**	**Base Volatility**	**Max Win**	**Line Bet**	**Cycle**
3	base_game	95,061%		1,777	3,292	10004,000	0,000	10000000
4								
5	**Symbol**	**Payment**	**Hits**	**Probability**				
6	2-sym01	2	4390591	0,43905910				
7	3-sym01	4	1267313	0,12673130				
8	4-sym01	20	257232	0,02572320				
9	5-sym01	40	91553	0,00915530				
10	2-sym02	2	4046015	0,40460150				
11	3-sym02	4	968303	0,09683030				
12	4-sym02	20	247440	0,02474400				
13	5-sym02	40	87494	0,00874940				
14	3-sym03	5	401416	0,04014160				
15	4-sym03	75	71692	0,00716920				
16	5-sym03	150	13886	0,00138860				
17	3-sym04	5	248175	0,02481750				
18	4-sym04	75	37340	0,00373400				
19	5-sym04	150	5868	0,00058680				
20	3-sym05	12	186673	0,01866730				
21	4-sym05	120	25473	0,00254730				
22	5-sym05	200	3497	0,00034970				
23	3-sym06	12	62126	0,00621260				
24	4-sym06	120	7872	0,00078720				
25	5-sym06	200	751	0,00007510				
26	2-scatter	5	322966	0,03229660				
27	3-scatter	50	20723	0,00207230				
28	4-scatter	500	631	0,00006310				
29	5-scatter	1000	10	0,00000100				
30								
31	**Payment**	**Hits**	**Probability**					
32	10002	1	0,00000010					
33	10000	9	0,00000090					
34	5005	1	0,00000010					
35	5004	7	0,00000070					
36	5002	39	0,00000390					
37	5000	584	0,00005840					
38	575	4	0,00000040					
39	540	8	0,00000080					
40	520	19	0,00000190					
41	512	14	0,00000140					
42	505	65	0,00000650					
43	504	299	0,00002990					
44	502	1468	0,00014680					
45	500	18846	0,00188460					
46	200	446	0,00004460					
47	170	42	0,00000420					
48	150	1947	0,00019470					
49	125	201	0,00002010					
50	120	3312	0,00033120					
51	90	312	0,00003120					
52	75	10797	0,00107970					
53	70	836	0,00008360					
54	62	414	0,00004140					
55	55	1066	0,00010660					
56	54	6036	0,00060360					
57	52	25961	0,00259610					
58	50	288081	0,02880810					
59	40	17615	0,00176150					
60	20	49652	0,00496520					
61	12	24323	0,00243230					
62	5	63876	0,00638760					
63	4	217366	0,02173660					
64	2	815939	0,08159390					
65	0	8450414	0,84504140					

Game	Base RTP	Final RTP	Hit Rate	Base Volatility	Max Win	Line Bet	Cycle
base_game	95,041%		1,776	3,293	10004,000	0,000	10000000

Symbol	Payment	Hits	Probability
2-sym01	2	4389257	0,43892570
3-sym01	4	1266233	0,12662330
4-sym01	20	256845	0,02568450
5-sym01	40	91854	0,00918540
2-sym02	2	4049578	0,40495780
3-sym02	4	968814	0,09688140
4-sym02	20	246533	0,02465330
5-sym02	40	87966	0,00879660
3-sym03	5	402852	0,04028520
4-sym03	75	71712	0,00717120
5-sym03	150	14083	0,00140830
3-sym04	5	248252	0,02482520
4-sym04	75	36978	0,00369780
5-sym04	150	5987	0,00059870
3-sym05	12	186776	0,01867760
4-sym05	120	25325	0,00253250
5-sym05	200	3541	0,00035410
3-sym06	12	62172	0,00621720
4-sym06	120	8080	0,00080800
5-sym06	200	732	0,00007320
2-scatter	5	323526	0,03235260
3-scatter	50	20666	0,00206660
4-scatter	500	623	0,00006230
5-scatter	1000	7	0,00000070

Payment	Hits	Probability
10002	2	0,00000020
10000	5	0,00000050
5005	2	0,00000020
5004	6	0,00000060
5002	45	0,00000450
5000	570	0,00005700
575	9	0,00000090
540	5	0,00000050
520	20	0,00000200
512	10	0,00000100
505	66	0,00000660
504	308	0,00003080
502	1557	0,00015570
500	18691	0,00186910
200	408	0,00004080
170	45	0,00000450
150	1938	0,00019380
125	184	0,00001840
120	3297	0,00032970
90	324	0,00003240
75	10679	0,00106790
70	855	0,00008550
62	374	0,00003740
55	1099	0,00010990
54	6078	0,00060780
52	26035	0,00260350
50	288516	0,02885160
40	17563	0,00175630
20	49661	0,00496610
12	24370	0,00243700
5	64334	0,00643340
4	216976	0,02169760
2	816166	0,08161660
0	8449802	0,84498020

	A	B	C	D	E	F	G	H
1								
2	**Game**	**Base RTP**	**Final RTP**	**Hit Rate**	**Base Volatility**	**Max Win**	**Line Bet**	**Cycle**
3	base_game	95,120%		1,777	3,297	10002,000	0,000	10000000
4								
5	**Symbol**	**Payment**	**Hits**	**Probability**				
6	2-sym01	2	4390138	0,43901380				
7	3-sym01	4	1267441	0,12674410				
8	4-sym01	20	258097	0,02580970				
9	5-sym01	40	92094	0,00920940				
10	2-sym02	2	4039551	0,40395510				
11	3-sym02	4	968351	0,09683510				
12	4-sym02	20	246420	0,02464200				
13	5-sym02	40	87444	0,00874440				
14	3-sym03	5	402727	0,04027270				
15	4-sym03	75	72025	0,00720250				
16	5-sym03	150	13732	0,00137320				
17	3-sym04	5	248252	0,02482520				
18	4-sym04	75	37660	0,00376600				
19	5-sym04	150	5957	0,00059570				
20	3-sym05	12	187196	0,01871960				
21	4-sym05	120	25675	0,00256750				
22	5-sym05	200	3592	0,00035920				
23	3-sym06	12	62116	0,00621160				
24	4-sym06	120	7905	0,00079050				
25	5-sym06	200	812	0,00008120				
26	2-scatter	5	322460	0,03224600				
27	3-scatter	50	20732	0,00207320				
28	4-scatter	500	624	0,00006240				
29	5-scatter	1000	10	0,00000100				
30								
31	**Payment**	**Hits**	**Probability**					
32	10000	10	0,00000100					
33	5004	5	0,00000050					
34	5002	39	0,00000390					
35	5000	580	0,00005800					
36	575	3	0,00000030					
37	540	4	0,00000040					
38	520	22	0,00000220					
39	512	9	0,00000090					
40	505	47	0,00000470					
41	504	324	0,00003240					
42	502	1579	0,00015790					
43	500	18744	0,00187440					
44	200	467	0,00004670					
45	170	29	0,00000290					
46	150	1885	0,00018850					
47	125	183	0,00001830					
48	120	3327	0,00033270					
49	90	321	0,00003210					
50	75	10795	0,00107950					
51	70	839	0,00008390					
52	62	414	0,00004140					
53	55	1046	0,00010460					
54	54	5949	0,00059490					
55	52	26137	0,00261370					
56	50	287523	0,02875230					
57	40	17689	0,00176890					
58	20	49416	0,00494160					
59	12	24516	0,00245160					
60	5	64105	0,00641050					
61	4	217559	0,02175590					
62	2	814903	0,08149030					
63	0	8451531	0,84515310					

Game	Base RTP	Final RTP	Hit Rate	Base Volatility	Max Win	Line Bet	Cycle
base_game	95,608%		1,776	3,308	10004,000	0,000	10000000

Symbol	Payment	Hits	Probability
2-sym01	2	4388416	0,43884160
3-sym01	4	1269062	0,12690620
4-sym01	20	257623	0,02576230
5-sym01	40	91397	0,00913970
2-sym02	2	4046263	0,40462630
3-sym02	4	966669	0,09666690
4-sym02	20	247824	0,02478240
5-sym02	40	87736	0,00877360
3-sym03	5	402261	0,04022610
4-sym03	75	72300	0,00723000
5-sym03	150	13744	0,00137440
3-sym04	5	247968	0,02479680
4-sym04	75	36916	0,00369160
5-sym04	150	5986	0,00059860
3-sym05	12	186975	0,01869750
4-sym05	120	25526	0,00255260
5-sym05	200	3669	0,00036690
3-sym06	12	62656	0,00626560
4-sym06	120	7779	0,00077790
5-sym06	200	738	0,00007380
2-scatter	5	323639	0,03236390
3-scatter	50	21139	0,00211390
4-scatter	500	671	0,00006710
5-scatter	1000	14	0,00000140

Payment	Hits	Probability
10002	1	0,00000010
10000	13	0,00000130
5005	1	0,00000010
5004	3	0,00000030
5002	48	0,00000480
5000	619	0,00006190
575	3	0,00000030
540	10	0,00000100
520	21	0,00000210
512	18	0,00000180
505	59	0,00000590
504	355	0,00003550
502	1604	0,00016040
500	19069	0,00190690
200	420	0,00004200
170	24	0,00000240
150	1894	0,00018940
125	208	0,00002080
120	3299	0,00032990
90	306	0,00003060
75	10779	0,00107790
70	870	0,00008700
62	352	0,00003520
55	1070	0,00010700
54	6027	0,00060270
52	26070	0,00260700
50	288698	0,02886980
40	17450	0,00174500
20	49560	0,00495600
12	24528	0,00245280
5	63822	0,00638220
4	217421	0,02174210
2	815525	0,08155250
0	8449853	0,84498530

Game	Base RTP	Final RTP	Hit Rate	Base Volatility	Max Win	Line Bet	Cycle
base_game	95,069%		1,776	3,292	10002,000	0,000	10000000

Symbol	Payment	Hits	Probability
2-sym01	2	4391617	0,43916170
3-sym01	4	1268950	0,12689500
4-sym01	20	258534	0,02585340
5-sym01	40	91607	0,00916070
2-sym02	2	4046112	0,40461120
3-sym02	4	967702	0,09677020
4-sym02	20	247467	0,02474670
5-sym02	40	87582	0,00875820
3-sym03	5	402427	0,04024270
4-sym03	75	71810	0,00718100
5-sym03	150	13966	0,00139660
3-sym04	5	247291	0,02472910
4-sym04	75	37223	0,00372230
5-sym04	150	5906	0,00059060
3-sym05	12	187337	0,01873370
4-sym05	120	25433	0,00254330
5-sym05	200	3693	0,00036930
3-sym06	12	61996	0,00619960
4-sym06	120	7854	0,00078540
5-sym06	200	759	0,00007590
2-scatter	5	323519	0,03235190
3-scatter	50	20780	0,00207800
4-scatter	500	612	0,00006120
5-scatter	1000	5	0,00000050

Payment	Hits	Probability
10000	5	0,00000050
5004	4	0,00000040
5002	48	0,00000480
5000	560	0,00005600
575	4	0,00000040
540	13	0,00000130
520	24	0,00000240
512	16	0,00000160
505	56	0,00000560
504	310	0,00003100
502	1492	0,00014920
500	18865	0,00188650
200	463	0,00004630
170	29	0,00000290
150	2009	0,00020090
125	172	0,00001720
120	3328	0,00033280
90	323	0,00003230
75	10813	0,00108130
70	886	0,00008860
62	405	0,00004050
55	1036	0,00010360
54	6167	0,00061670
52	25769	0,00257690
50	288718	0,02887180
40	17411	0,00174110
20	49861	0,00498610
12	24651	0,00246510
5	63779	0,00637790
4	217946	0,02179460
2	816035	0,08160350
0	8448802	0,84488020

	A	B	C	D	E	F	G	H
1								
2	**Game**	**Base RTP**	**Final RTP**	**Hit Rate**	**Base Volatility**	**Max Win**	**Line Bet**	**Cycle**
3	base_game	95,224%		1,777	3,302	10004,000	0,000	10000000
4								
5	**Symbol**	**Payment**	**Hits**	**Probability**				
6	2-sym01	2	4389434	0,43894340				
7	3-sym01	4	1269125	0,12691250				
8	4-sym01	20	257981	0,02579810				
9	5-sym01	40	91652	0,00916520				
10	2-sym02	2	4047845	0,40478450				
11	3-sym02	4	968590	0,09685900				
12	4-sym02	20	245415	0,02454150				
13	5-sym02	40	87778	0,00877780				
14	3-sym03	5	403042	0,04030420				
15	4-sym03	75	72036	0,00720360				
16	5-sym03	150	14144	0,00141440				
17	3-sym04	5	247676	0,02476760				
18	4-sym04	75	37182	0,00371820				
19	5-sym04	150	5932	0,00059320				
20	3-sym05	12	187720	0,01877200				
21	4-sym05	120	25648	0,00256480				
22	5-sym05	200	3637	0,00036370				
23	3-sym06	12	62045	0,00620450				
24	4-sym06	120	7795	0,00077950				
25	5-sym06	200	735	0,00007350				
26	2-scatter	5	322719	0,03227190				
27	3-scatter	50	20682	0,00206820				
28	4-scatter	500	653	0,00006530				
29	5-scatter	1000	7	0,00000070				
30								
31	**Payment**	**Hits**	**Probability**					
32	10002	1	0,00000010					
33	10000	6	0,00000060					
34	5005	2	0,00000020					
35	5004	9	0,00000090					
36	5002	49	0,00000490					
37	5000	593	0,00005930					
38	575	1	0,00000010					
39	540	8	0,00000080					
40	520	23	0,00000230					
41	512	9	0,00000090					
42	505	49	0,00000490					
43	504	284	0,00002840					
44	502	1543	0,00015430					
45	500	18765	0,00187650					
46	200	417	0,00004170					
47	170	38	0,00000380					
48	150	1970	0,00019700					
49	125	180	0,00001800					
50	120	3334	0,00033340					
51	90	318	0,00003180					
52	75	10889	0,00108890					
53	70	841	0,00008410					
54	62	384	0,00003840					
55	55	1070	0,00010700					
56	54	5920	0,00059200					
57	52	25935	0,00259350					
58	50	288008	0,02880080					
59	40	17724	0,00177240					
60	20	49358	0,00493580					
61	12	24542	0,00245420					
62	5	64178	0,00641780					
63	4	217878	0,02178780					
64	2	816981	0,08169810					
65	0	8448693	0,84486930					

Game	Base RTP	Final RTP	Hit Rate	Base Volatility	Max Win	Line Bet	Cycle
base_game	95,312%		1,776	3,301	10000,000	0,000	10000000

Symbol	Payment	Hits	Probability
2-sym01	2	4388498	0,43884980
3-sym01	4	1268015	0,12680150
4-sym01	20	257479	0,02574790
5-sym01	40	92028	0,00920280
2-sym02	2	4046960	0,40469600
3-sym02	4	969322	0,09693220
4-sym02	20	247143	0,02471430
5-sym02	40	88135	0,00881350
3-sym03	5	400657	0,04006570
4-sym03	75	71980	0,00719800
5-sym03	150	14098	0,00140980
3-sym04	5	248059	0,02480590
4-sym04	75	37213	0,00372130
5-sym04	150	5868	0,00058680
3-sym05	12	186925	0,01869250
4-sym05	120	25712	0,00257120
5-sym05	200	3691	0,00036910
3-sym06	12	61920	0,00619200
4-sym06	120	7882	0,00078820
5-sym06	200	738	0,00007380
2-scatter	5	322515	0,03225150
3-scatter	50	20911	0,00209110
4-scatter	500	652	0,00006520
5-scatter	1000	2	0,00000020

Payment	Hits	Probability
10000	2	0,00000020
5004	7	0,00000070
5002	51	0,00000510
5000	594	0,00005940
575	2	0,00000020
540	9	0,00000090
520	24	0,00000240
512	9	0,00000090
505	63	0,00000630
504	284	0,00002840
502	1617	0,00016170
500	18903	0,00189030
200	433	0,00004330
170	24	0,00000240
150	1990	0,00019900
125	168	0,00001680
120	3391	0,00033910
90	322	0,00003220
75	10651	0,00106510
70	799	0,00007990
62	429	0,00004290
55	1112	0,00011120
54	6027	0,00060270
52	25869	0,00258690
50	287748	0,02877480
40	17626	0,00176260
20	49846	0,00498460
12	24571	0,00245710
5	63551	0,00635510
4	216807	0,02168070
2	815276	0,08152760
0	8451795	0,84517950

Game	Base RTP	Final RTP	Hit Rate	Base Volatility	Max Win	Line Bet	Cycle
base_game	95,142%		1,777	3,300	10004,000	0,000	10000000

Symbol	Payment	Hits	Probability
2-sym01	2	4392356	0,43923560
3-sym01	4	1266752	0,12667520
4-sym01	20	257854	0,02578540
5-sym01	40	92443	0,00924430
2-sym02	2	4045997	0,40459970
3-sym02	4	969811	0,09698110
4-sym02	20	246968	0,02469680
5-sym02	40	87558	0,00875580
3-sym03	5	402661	0,04026610
4-sym03	75	71380	0,00713800
5-sym03	150	14044	0,00140440
3-sym04	5	246902	0,02469020
4-sym04	75	37183	0,00371830
5-sym04	150	5912	0,00059120
3-sym05	12	186871	0,01868710
4-sym05	120	25632	0,00256320
5-sym05	200	3621	0,00036210
3-sym06	12	61893	0,00618930
4-sym06	120	7922	0,00079220
5-sym06	200	762	0,00007620
2-scatter	5	322782	0,03227820
3-scatter	50	20653	0,00206530
4-scatter	500	635	0,00006350
5-scatter	1000	11	0,00000110

Payment	Hits	Probability
10000	11	0,00000110
5005	2	0,00000020
5004	7	0,00000070
5002	43	0,00000430
5000	583	0,00005830
575	2	0,00000020
540	9	0,00000090
520	29	0,00000290
512	8	0,00000080
505	44	0,00000440
504	352	0,00003520
502	1534	0,00015340
500	18675	0,00186750
200	473	0,00004730
170	35	0,00000350
150	1967	0,00019670
125	167	0,00001670
120	3345	0,00033450
90	321	0,00003210
75	10593	0,00105930
70	818	0,00008180
62	376	0,00003760
55	1122	0,00011220
54	6107	0,00061070
52	25954	0,00259540
50	287859	0,02878590
40	17780	0,00177800
20	49461	0,00494610
12	24317	0,00243170
5	63865	0,00638650
4	217071	0,02170710
2	815727	0,08157270
0	8451343	0,84513430

Game	Base RTP	Final RTP	Hit Rate	Base Volatility	Max Win	Line Bet	Cycle
base_game	95,398%		1,777	3,304	10002,000	0,000	10000000

Symbol	Payment	Hits	Probability
2-sym01	2	4395596	0,43955960
3-sym01	4	1266314	0,12663140
4-sym01	20	257631	0,02576310
5-sym01	40	91818	0,00918180
2-sym02	2	4039773	0,40397730
3-sym02	4	965908	0,09659080
4-sym02	20	245927	0,02459270
5-sym02	40	87687	0,00876870
3-sym03	5	402071	0,04020710
4-sym03	75	71540	0,00715400
5-sym03	150	13587	0,00135870
3-sym04	5	248922	0,02489220
4-sym04	75	37183	0,00371830
5-sym04	150	5907	0,00059070
3-sym05	12	187472	0,01874720
4-sym05	120	25329	0,00253290
5-sym05	200	3671	0,00036710
3-sym06	12	62039	0,00620390
4-sym06	120	7862	0,00078620
5-sym06	200	747	0,00007470
2-scatter	5	322694	0,03226940
3-scatter	50	20897	0,00208970
4-scatter	500	688	0,00006880
5-scatter	1000	13	0,00000130

Payment	Hits	Probability
10000	13	0,00000130
5005	1	0,00000010
5004	10	0,00000100
5002	43	0,00000430
5000	634	0,00006340
575	2	0,00000020
540	7	0,00000070
520	26	0,00000260
512	7	0,00000070
505	45	0,00000450
504	330	0,00003300
502	1622	0,00016220
500	18858	0,00188580
200	452	0,00004520
170	41	0,00000410
150	1971	0,00019710
125	187	0,00001870
120	3304	0,00033040
90	288	0,00002880
75	10785	0,00107850
70	844	0,00008440
62	384	0,00003840
55	1117	0,00011170
54	6113	0,00061130
52	26067	0,00260670
50	287627	0,02876270
40	17610	0,00176100
20	49767	0,00497670
12	24471	0,00244710
5	63855	0,00638550
4	217909	0,02179090
2	815983	0,08159830
0	8449627	0,84496270

Bibliography

Research and Markets, "Global Gambling Market Analysis & Forecasts 2012-2019 & 2020-2027." https://www.researchandmarkets.com, 2020.

Tradefest, "The best gaming, sports betting and affiliate trade shows and conferences in 2020 & 2021." https://tradefest.io/en/tag/gambling-and-casinos, 2020.

Zeusplay, "Zeusplay games." https://zeusplay.com/, February 2020.

O. Slots, "The different types of bonus games." https://www.onlineslots.com/bonus-games/, 2020.

Betsoft, "Fa-fa twins." https://betsoft.com/new_games/fa-fa-twins/, 2016.

N. Mladenović and P. Hansen, "Variable neighborhood search," *Computers & Operations Research*, vol. 24, no. 11, pp. 1097–1100, 1997.

N. Mladenović and P. Hansen, "Variable neighborhood search," *Computers & operations research*, vol. 24, no. 11, pp. 1097–1100, 1997.

C. H. Papadimitriou and M. Yannakakis, "On bounded rationality and computational complexity," in *Indiana University*, Citeseer, 1994.

N. Mladenović and P. Hansen, "Variable neighborhood search," *Computers & operations research*, vol. 24, no. 11, pp. 1097–1100, 1997.

A. A. Andreatta and C. C. Ribeiro, "Heuristics for the phylogeny problem," *Journal of Heuristics*, vol. 8, no. 4, pp. 429–447, 2002.

J. Brimberg, P. Hansen, N. Mladenović, and E. D. Taillard, "Improvements and comparison of heuristics for solving the uncapacitated multisource weber problem," *Operations research*, vol. 48, no. 3, pp. 444–460, 2000.

S. Canuto, M. Resende, and C. Ribeiro, "Local search with perturbations for the prize collecting steiner tree problem in graphs," *Networks*, vol. 38, pp. 50 – 58, 08 2001.

G. Caporossi and P. Hansen, "Variable neighborhood search for extremal graphs: 1 the autographix system," *Discrete Mathematics*, vol. 212, no. 1, pp. 29 – 44, 2000.

G. Caporossi and P. Hansen, "Variable neighborhood search for extremal graphs. 5. three ways to automate finding conjectures," *Discrete Mathematics*, vol. 276, no. 1-3, pp. 81–94, 2004.

P. Hansen and N. Mladenović, "J-means: a new local search heuristic for minimum sum of squares clustering," *Pattern Recognition*, vol. 34, no. 2, pp. 405 – 413, 2001.

C. C. Ribeiro, E. Uchoa, and R. F. Werneck, "A hybrid grasp with perturbations for the steiner problem in graphs," *INFORMS Journal on Computing*, vol. 14, no. 3, pp. 228–246, 2002.

T. Balabanov, I. Zankinski, and B. Shumanov, "Slot machines RTP optimization with genetic algorithms," in *Numerical Methods and Applications (NMA 2014)* (I. Dimov, S. Fidanova, and I. Lirkov, eds.), vol. 8962 of *LNCS*, pp. 55–61, Springer, Cham, 2015.

T. Balabanov, I. Zankinski, and B. Shumanov, "Slot machine RTP optimization and symbols wins equalization with discrete differential evolution," in *Large-Scale Scientific Computing (LSSC 2015)* (I. Lirkov, S. D. Margenov, and J. Waśniewski, eds.), vol. 9374 of *LNCS*, pp. 210–217, Springer, Cham, 2015.

Zeusplay, "Arabian dream." `https://zeusplay.com/portfolio_page/arabian-dream/`.

Zeusplay, "Super hot." `https://zeusplay.com/portfolio_page/super-hot/`.

Zeusplay, "Wild charger." `https://zeusplay.com/portfolio_page/wild-charger/`.

Zeusplay, "Trojan horse." `https://zeusplay.com/portfolio_page/trojan-horse/`.

R. Muir, *Slot Designer Tools for professional mathematicians.* GameDesignAutomation.com, 2 ed., 2013.

Mathplanet, "Standard deviation and normal distribution." `https://www.mathplanet.com`, 2019.

D. Keremedchiev, P. Tomov, and M. Barova, "Slot machine base game evolutionary RTP optimization," in *Numerical Analysis and Its Applications (NAA 2016)* (I. Dimov, I. Faragó, and L. Vulkov, eds.), vol. 10187 of *LNCS*, pp. 406–413, Springer, Cham, 2017.

P. Hansen, N. Mladenović, J. Brimberg, and J. Moreno-Perez, "Variable Neighborhood Search," in *Handbook of Metaheuristics* (M. Gendreau and J.-Y. Potvin, eds.), International Series in Operations Research & Management Science, 2019.

I. Alharkan, K. Bamatraf, M. A. Noman, H. Kaid, E. S. Abouel Nasr, and A. M. El-Tamimi, "An order effect of neighborhood structures in variable neighborhood search algorithm for minimizing the makespan in an identical parallel machine scheduling," *Mathematical Problems in Engineering*, vol. 2018, 2018. Article ID 3586731.

C. Alves, P. Bras, J. M. Valerio de Carvalho, and T. Pinto, "A variable neighborhood search algorithm for the leather nesting problem," *Mathematical Problems in Engineering*, vol. 2012, 2012. Article ID 254346.

R. Benmansour, A. Sifaleras, and N. Mladenović, eds., *Variable Neighborhood Search. 7th International Conference, ICVNS 2019, Rabat, Morocco, October 3-5, 2019, Revised Selected Papers*, vol. 12010 of *LNCS*, Springer, Cham, 2020.

A.-f. Ling, "A VNS metaheuristic with stochastic steps for max 3-cut and max 3-section," *Mathematical Problems in Engineering*, vol. 2012, 2012. Article ID 475018.

A. Sifaleras, S. Salhi, and J. Brimberg, eds., *Variable Neighborhood Search. 6th International Conference, ICVNS 2018, Sithonia, Greece, October 4-7, 2018, Revised Selected Papers*, vol. 11328 of *LNCS*, Springer, Cham, 2019.

A. Duarte, J. Sánchez-Oro, N. Mladenović, and R. Todosijević, "Variable Neighborhood Descent," in *Handbook of Heuristics* (R. Martí, P. M. Pardalos, and M. G. C. Resende, eds.), pp. 341–367, Cham: Springer International Publishing, 2018.

A. Sifaleras and I. Konstantaras, "Variable neighborhood descent heuristic for solving reverse logistics multi-item dynamic lot-sizing problems," *Computers and Operations Research*, vol. 78, pp. 385–392, 2017.

Zeusplay, "Le mystere du prince." `https://zeusplay.com/portfolio_page/le-mystere-du-prince/`.

Zeusplay, "Amun's book." `https://zeusplay.com/portfolio_page/amuns-book/`.

Arseniumn, "Wild sevens." `https://github.com/arseniumn/Slot-Machine-Wild-Sevens`, 2020.

Arseniumn, "Rolling bars." `https://github.com/arseniumn/Slot-Machine-Rolling-Bars`, 2020.

ZeusPlay, "Slot wizard : A software for slot machine game development." `https://eliconsoft.com/products/`, 2020.