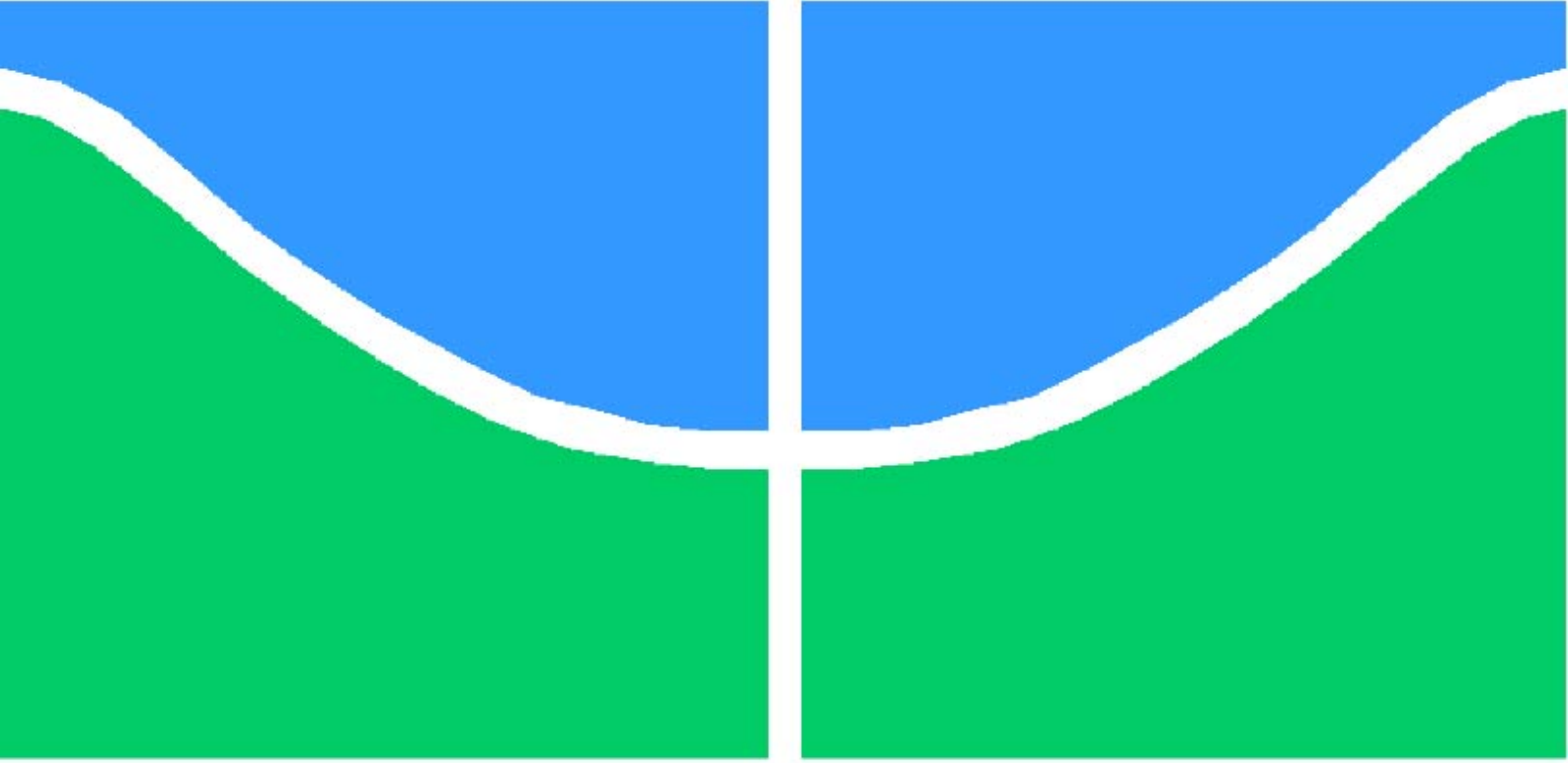

Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Engenharia de *Software*

Big Points: Uma Análise Baseada na Teoria dos Jogos

Autor: Mateus Medeiros Furquim Mendonça

Orientador: Prof. Dr. Edson Alves da Costa Júnior

Coorientador: Prof. Dr. Nilton Correia da Silva

Brasília, DF

2016

Mateus Medeiros Furquim Mendonça

Big Points: Uma Análise Baseada na Teoria dos Jogos

Monografia submetida ao curso de graduação em Engenharia de *Software* da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de *Software*.

Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Orientador: Prof. Dr. Edson Alves da Costa Júnior

Coorientador: Prof. Dr. Nilton Correia da Silva

Brasília, DF

2016

Mateus Medeiros Furquim Mendonça

Big Points: Uma Análise Baseada na Teoria dos Jogos

Monografia submetida ao curso de graduação em Engenharia de *Software* da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de *Software*.

Trabalho aprovado. Brasília, DF, 7 de julho de 2016:

Prof. Dr. Edson Alves da Costa Júnior
Orientador

Prof. Dr. Fábio Macedo Mendes
Convidado 1

Prof. Dra. Carla Silva Rocha Aguiar
Convidado 2

Brasília, DF
2016

Resumo

A Teoria dos Jogos estuda as melhores estratégias dos jogadores em um determinado jogo. Aplicando suas teorias em um jogo de tabuleiro eletrônico, este trabalho propõe analisar o jogo *Big Points* a partir de um determinado estado da partida e, como resultado, identificar as melhores heurísticas para os jogadores e uma possível inteligência artificial.

Palavras-chaves: Teoria dos Jogos, Análise Combinatória de Jogos.

Abstract

Key-words: Game Theory, Combinatorial Game Theory.

Lista de ilustrações

Lista de tabelas

Lista de símbolos

Símbolos para conjuntos e operações matemáticas

$\varnothing$	Um conjunto sem elementos, conjunto vazio
$\{\ \}$	Delimita conjunto, de forma que $S = \{\}$ é um conjunto vazio
$\forall$	Para cada elemento
$x \in S$	Elemento x pertence ao conjunto S
$x \notin S$	Elemento x não pertence ao conjunto S
$S \subseteq T$	Conjunto S é um subconjunto de T, significa que se $x \in S$ então $x \in T$
$S \cup T$	União entre dois conjuntos $\{x\ ; x \in S \text{ or } x \in T\}$
$S \cap T$	Inteseção entre dois conjuntos $\{x\ ; x \in S \text{ and } x \in T\}$
$S_1 \times \ldots \times S_n$	Produto cartesiano $\{(x_1, \ldots, x_n)\ ; x_i \in S_i (1 \leq i \leq n)\}$
$\sum_{i=1}^{n} x_i$	Somatório de x_1 até x_n de maneira que $\sum_{i=1}^{n} x_i = x_1 + x_2 + \ldots + x_n$
$\prod_{i=1}^{n} x_i$	Produto de x_1 até x_n de maneira que $\prod_{i=1}^{n} x_i = x_1 \cdot x_2 \cdot \ldots \cdot x_n$
$A_{p,q}$	Arranjo de p elementos tomados de q a q calculado $A_{p,q} = \frac{p!}{(p-q)!}$
$\binom{p}{q}$	Combinação de p elementos tomados de q a q calculado $\binom{p}{q} = \frac{p!}{q! \cdot (p-1)!}$

Para jogos de soma zero com dois jogadores

σ, τ	Estratégias puras
x	Estratégia mista para o jogador 1
X	Conjunto de todas as estratégias mistas para o jogador 1
y	Estratégia mista para o jogador 2
Y	Conjunto de todas as estratégias mistas para o jogador 2
$P(x, y)$	Ganho do jogador 1

Para jogos não cooperativos com n jogadores

σ_i	uma estratégia pura para o jogador i
S_i	Conjunto de todas as estratégias puras para o jogador i
x_i	uma estratégia mista para o jogador i
X_i	Conjunto de todas as estratégias mistas para o jogador i
$P_i(x_1, \ldots, x_n)$	Ganho do jogador i
$x\|\|x_i'$	Considerando $x = (x_1, \ldots, x_n)$ o conjunto com todas as estratégias dos n jogadores, jogador i substitui a estratégia x_i pela estratégia x_i'

Sumário

1 Introdução

1.1 Contextualização

A **Teoria dos Jogos** é uma área de estudos derivada da matemática que, por alguns anos vem estudando o comportamento de indivíduos sob uma situação de conflito, como em jogos, balança de poder, leilões, e até mesmo evolução genética (SARTINI et al., 2004). Esta área possui duas frentes de estudo: (a) *teoria econômica dos jogos*, o qual possui motivações predominante econômicas, e (b) *teoria combinatória dos jogos*, que faz uso dos aspectos combinatórios de jogos de mesa e não permite elementos imprevisíveis.

1.2 Objetivos Principais e Secundários

O objetivo deste trabalho é, fazendo uso da *teoria combinatória dos jogos*, encontrar um *winning move*[1]. Além do objetivo principal, este trabalho ainda possui dois objetivos secundários: (a) estabelecer uma heurística[2] para se maximizar o ganho (*payoff*), fazendo uso da *teoria econômica dos jogos* e; (b) criar uma inteligência artificial com três níveis de dificuldade para jogar contra um jogador.

1.3 Estrutura do Trabalho

O restante deste trabalho está organizado da seguinte maneira: Na seção 2 é narrado uma breve história da teoria dos jogos e seus conceitos fundamentais, além de conter explicação para os temas de análise de complexidade, análise combinatória e programação dinâmica, e explicação das regras do jogo *Big Points*. A seção seguinte (3) lista os equipamentos, *softwares* e metodologia utilizados para o desenvolvimento do trabalho e, também, a maneira que a foi analisado o jogo. Os resultados, até o momento, são descritos na seção 4, o cronograma de trabalho na seção 5, e as considerações finais na seção 6.

[1] *Winning move* é um movimento que, a partir de uma determinada jogada, garantirá a vitória independente do resto do jogo

[2] Heurística é uma abordagem para solucionar um problema sem garantias de que o restulado é a solução ótima.

2 Fundamentação Teórica

2.1 O que é Teoria dos Jogos?

Teoria dos jogos é o estudo do comportamento estratégico interdependente[1] (SPANIEL, 2011), não apenas o estudo de como vencer ou perder em um jogo, apesar de às vezes esses dois fatos coincidem. Isso faz com que o escopo seja mais abranjente, desde comportamentos no qual as duas pessoas devem cooperar para ganhar, ou as duas tentam se ajudar, ou, por fim, comportamento de duas pessoas que tentam vencer individualmente.

2.2 Histórico da Teoria dos Jogos

Pode-se dizer que a análise de jogos é praticada desde o séculco XVIII tendo como evidência uma carta escrita por James Waldegrave ao analisar uma versão curta de um jogo de baralho chamado *le Her* (PRAGUE, 2004, p. 2), explicado na Seção 2.3.11. No século seguinte, Augustin Cournot fez uso da teoria dos jogos para estudos relacionados à política. Mais recentemente, em 1913, Ernst Zermelo publica o primeiro teorema matemático da teoria dos jogos (SARTINI et al., 2004, p. 2).

Dois grandes matemáticos que se interessaram na teoria dos jogos foram Émile Borel e John von Neumann. Nas décadas de 1920 e 1930, Emile Borel publicou quatro artigos sobre jogos estratégicos (PRAGUE, 2004, p. 2), introduzindo uma noção abstrada sobre jogo estratégico e estratégia mista. Em 1928, John von Neumann demonstrou que todo jogo finito de soma zero[2] com duas pessoas possui uma solução em estratégias mistas. Em 1944, Neumann publicou um trabalho junto a Oscar Morgenstern introduzindo a teoria dos jogos na área da economia e matemática aplicada (SARTINI et al., 2004, p. 2–3).

Outro matemático que contribuiu para a área foi John Forbes Nash Júnior, que publicou quatro artigos importantes para teoria dos jogos não-cooperativos. Dois destes artigos provando a existência de um equilíbrio de **estratégias mistas** para jogos não-cooperativos, denominado **equilíbrio de Nash**. Os conceitos de estratégia mista será explicado na Seção 2.3.10.6 e equilíbrio de nash na Seção 2.3.10.5. Nash recebeu o prêmio Nobel em 1994, junto com John Harsanyi e Reinhard Selten, por suas contribuições para a teoria dos jogos (SARTINI et al., 2004, p. 3–4).

[1] Estratégia interdependente significa que as ações de uma pessoa interfere no resultado da outra, e vice-versa.

[2] Um jogo soma zero é um jogo no qual a vitória de um jogador implica na derrota do outro.

2.3 Conceitos Fundamentais da Teoria dos Jogos

Esta seção introduz os conceitos fundamentais da teoria dos jogos, tais como definição de um jogo não cooperativo, formas de representá-lo e teoremas para encontrar soluções. Para tanto, considere o exemplo do jogo *Renée v Peter* retirado do livro (JONES, 1980).

Exemplo 1. O jogo *Renée v Peter* é jogado com três cartas: Rei (K), Dez (T) e Dois (D). *Renée* escolhe uma carta dentre as três, coloca-a voltada para baixo e, então, *Peter* escolhe *Alta* ou *Baixa*. *Peter* ganha R$ 3 se acertar (*Alta* = K e *Baixa* = D), mas deve pagar R$ 2 a *Renée* se errar.

A terceira possibilidade de jogada de *Renée* é escolher o T. Neste caso, se *Peter* tiver escolhido *Baixa*, ele ganha R$ 2, mas se tiver escolhido *alta*, *Renée* deve escolher outra carta (dentre as duas restantes). Se *Peter* acertar a segunda carta, *Renée* deve pagar R$ 1, mas ganha R$ 3 se *Peter* errar.

2.3.1 Definição de Jogo Não Cooperativo

Tendo a definição de um jogo como sendo uma atividade interativa e competitiva no qual os jogadores devem obedecer a um determinado conjunto de regras, um **jogo não cooperativo** é um jogo que não permite nenhum tipo de acordo entre os jogadores e o ganho de cada jogador é determinado pelo conjunto de regras (JONES, 1980).

2.3.2 Jogo Simétrico e Assimétrico

Um **jogo simétrico** é um jogo no qual as regras são as mesmas para ambos os jogadores. Em termos mais matemáticos, tem-se a Definição 1.

Definição 1. Dado uma matriz G de um jogo, G é a representação de um **jogo simétrico** se for uma matriz quadrada e $g_{ij} = -g_{ji}$ para cada par i, j. Em particular, $g_{ii} = 0$ para todo i. Portanto, G é uma representação de jogo simétrica se for uma matriz simétrica simpléctica[3,4]. (JONES, 1980)

2.3.3 Função Utilidade (*payoff*)

Uma **função utilidade** para um jogador determina o ganho para aquele jogador ao executar sua estratégia. Uma definição mais matemática é dada por Sartini et al como:

[3] Uma matriz é simpléctica se sua diagonal principal for 0.
[4] Do inglês *if its matrix is skew-symmetric*.

Definição 2. Uma *função utilidade* para o jogador i é uma função real $u_i : \Psi_T \to \mathbb{R}$ definida no conjunto Ψ_T das jogadas completas. O valor de u_i em cada jogada determina o *payoff* do jogador i para esta jogada. (SARTINI et al., 2004)

2.3.4 Jogo de dois jogadores e soma zero

Um jogo de dois jogadores e de soma zero significa que a vitória de um jogador implica na derrota do outro. Por convenção, o ganho ao final do jogo é representado pelo ganho do primeiro jogador, normalmente seguindo a seguinte representação: 1 caso ele ganhe, −1 caso ele perca e 0 caso ele empate. Caso explicado nas regras do jogo, o ganho pode ser calculado pela **função utilidade**.

2.3.5 Forma Extensiva

Uma das formas de representar um jogo, de tal maneira que seja possível analisá-lo em seguida, é a **forma extensiva** faz uso de uma estrutura de árvore, onde para representar os estados do jogo se faz uso dos nós da árvore, as jogadas possíveis a partir daquele estado são as arestas, e o ganho para o primeiro jogador são as folhas. A Figura 1 representa o Exemplo 1 na forma extensiva.

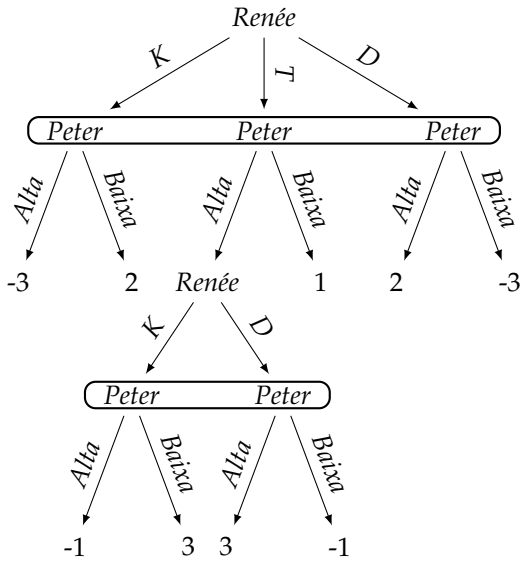

Figura 1: Árvore do jogo Renée v Peter

2.3.6 Conjunto Informação

Um **conjunto informação**, de acordo com Sartini et al, é um conjunto de estados do jogo que pertencem ao mesmo jogador e os movimentos disponíveis são os mesmos para cada estado (SARTINI et al., 2004). Jones diz, ainda, que um termo mais apropriado seria **conjunto de ignorância** (JONES, 1980), pois indica que ao fazer um movimento,

o jogador sabe em qual conjunto informação ele se encontra, mas não sabe dizer em qual estado está. Se o jogo está sendo representado por uma árvore, como na Figura 1, então o conjunto de informação é representado pelo

2.3.7 Estratégia Pura

Estratégia pura é um conjunto de escolhas a se fazer para cada momento de decisão possível. Voltando ao Exemplo 1, *Peter* pode escolher *Baixa* em sua primeira jogada, ganhando ganhando duas das vezes e perdendo uma, ou pode escolher *Alta* com uma possibilidade de ter que tomar outra decisão. Considerando essas possibilidades, tem-se que as estratégias puras de *Peter* são:

PI – Escolher *Alta*; Se *Renée* escolher *T*, escolher *Alta*.
PII – Escolher *Alta*; Se *Renée* escolher *T*, escolher *Baixa*.
PIII – Escolher *Baixa*.

Enquanto para *Renée*, suas opções são escolher entre as três cartas, mas se escolher *T* e tiver que escolher outra carta, deve escolher uma entre as duas restantes. Com isso, tem-se as seguintes estratégias puras de *Renée*:

RI – Escolher *K*.
RII – Escolher *T*; Se *Peter* escolher *Alta*, escolher *K*.
RIII – Escolher *T*; Se *Peter* escolher *Alta*, escolher *D*.
RIV – Escolher *D*.

2.3.8 Forma Normal

Essa propriedade **estratégias puras** permite uma outra representação de um jogo chamado de **forma normal**. Nesta forma o jogo é representado por uma matriz, onde na linha tem-se as estratégias puras do primeiro jogador e na coluna, as do segundo jogador. No caso do Exemplo 1, o jogo é representado pela Tabela 1.

		Peter		
		I	II	III
Renée	I	*P*	*P*	*R*
	II	*P*	*R*	*P*
	III	*R*	*P*	*P*
	IV	*R*	*R*	*P*

Tabela 1: Forma normal do jogo *Renée v Peter*

2.3.9 Matriz de *payoff*

Uma matriz de *payoff* nada mais é do que a representação em forma normal com o ganho indicado em cada situação. No caso do Exemplo 1 o ganho é e, com isso, tem-se a Tabela 2.

		Peter		
		I	II	III
Renée	I	-3	-3	2
	II	-1	3	-2
	III	3	-1	-2
	IV	2	2	-3

Tabela 2: Matriz de *payoff* do jogo *Renée v Peter*, fonte: (SPANIEL, 2011)

2.3.10 Soluções para jogos

Uma solução de um jogo é uma prescrição ou previsão sobre o resultado do jogo. Há soluções para os jogos que as escolhas dos jogadores são feitas simultaneamente e soluções para os jogos que possuem turnos. Para o primeiro caso, as soluções são encontradas fazendo uso de **dominância estrita iterada**, teorema **minimax** ou **equilíbrio de Nash**, podendo ser com **estratégias mistas** ou não, enquanto para o segundo caso a solução é encontrada utilizando-se de ***backward induction***.

2.3.10.1 Dominância estrita

É dito que uma estratégia é **estritamente dominada** para um jogador se esta estratégia gera um ganho maior do que qualquer outra estratégia independente do que o outro jogador fizer. Considera-se que jogadores racionais nunca fazem uso de estratégias estritamente dominadas, pois não há motivos para escolher uma estratégia que sempre será pior em todos os casos. O exemplo do **dilema do prisioneiro** demonstra tal conceito.

Formulado por Albert W. Tucker em 1950 (SARTINI et al., 2004), o dilema do prisioneiro é, provavelmente, um dos exemplos mais conhecidos na teoria dos jogos. O propósito de Tucker foi ilustrar a dificuldade de se analisar certos tipos de jogos por razões que ficarão óbvias após o exemplo.

Exemplo 2. Dois suspeitos são presos com suspeita de roubo mas os policiais podem apenas provar que os suspeitos invadiram o local. Precisando da confissão dos criminosos, o policial faz a seguinte proposta:

- Se nenhum confessar o roubo, o policial vai prendê-los por intrusão.

- Se um confessar e o outro não, o que confessou será liberto e o calado será preso por 12 meses.
- Se os dois confessarem, ambos serão presos por 8 meses.

Considerando que os dois suspeitos desejam minimizar seu tempo na cadeia, eles devem confessar à polícia?

Dado essas informações do Exemplo 2, é possível representá-las no formato **matriz de *payoffs*** de acordo com a Tabela 3, onde o ganho do *jogador linha* é representado pelo número à esquerda dentro de uma célula, e o ganho do *jogador coluna* é representado à direita.

		Coluna	
		Negar	Confessar
Linha	Negar	(-1,-1)	(-12,0)
	Confessar	(0,-12)	(-8,-8)

Tabela 3: Dilema do prisioneiro, fonte: (SPANIEL, 2011)

Para resolver este jogo é preciso raciocinar como um jogador responderia de acordo com a ação do outro. Supondo que o *jogador coluna* fique *Negar*, o *jogador linha* pode *Negar* e ir pra cadeia por 1 mês, sendo representado na Tabela 3 pela célula superior esquerda, ou aceitar a proposta do policial e *Confessar* o crime que iriam cometer, sendo representado pela célula inferior esquerda. Como os jogadores querem minimizar seu tempo na prisão, que é representado por um valor negativo, deve-se buscar o maior valor dentre essas duas escolhas, que neste caso é *Confessar* com um ganho de *0*. Observando a outra possibilidade do *jogador coluna*, que seria *Confessar*, o *jogador linha* teria um ganho de *-12* ao *Negar* e um ganho de *-8* ao *Confessar*. Em ambos os casos, o *jogador linha* terá um melhor ganho ao *Confessar*, e como o jogo é simétrico o mesmo raciocínio pode ser feito para o *jogador coluna*.

Essa preferência de *Confessar* a *Negar* para cada escolha do outro jogador (*Negar* ou *Confessar*) é dito que *Confessar* é estritamente dominante.

2.3.10.2 Eliminação iterada de estratégias estritamente dominadas

Eliminação por dominância estrita iterada, como o próprio nome já diz, é um processo iterado no qual estratégias são eliminadas por serem dominadas estritamente

(SPANIEL, 2011). Se uma estratégia for estritamente dominada, elimine-a imediatamente, não importando a ordem da eliminação. Se ao final do processo sobrar apenas uma única célula, este resultado será alcançado começando a eliminação por qualquer estratégia estritamente dominada. Considere a Tabela 4.

		Jogador Coluna		
		Esquerda	Centro	Direita
Jogador Linha	Cima	(13,3)	(1,4)	(7,3)
	Meio	(4,1)	(3,3)	(6,2)
	Baixo	(-1,9)	(2,8)	(8,-1)

Tabela 4: Exemplo de dominância estrita iterada, fonte: (SPANIEL, 2011)

Na Tabela 4, o *jogador linha* tem três estratégias *cima*, *meio* e *baixo*, enquanto o *jogador coluna* possui as estratégias *esquerda*, *centro* e *direita*, gerando um total de 9 resultados. Observando as estratégias do *jogador linha*, não é possível fazer nenhuma dominância estrita, pois o *jogador linha* possui uma preferência diferente de suas próprias estratégias para cada estratégia do *jogador coluna*. No caso da estratégia *esquerda*, a melhor estratégia para o *jogador linha* é *cima*, assim como *centro* e *meio*, e *direita* e *baixo*.

Porém, observando as estratégias *centro* e *direita*, é possível eliminar a segunda estratégia por dominância estrita, pois, para o *jogador coluna*, todos os ganhos da estratégia *centro* são melhores do que os ganhos da estratégia *direita*. Com a eliminação da estratégia *direita*, é possível eliminar a estratégia *baixo*, que é estritamente dominada pela estratégia *meio* para o *jogador linha*. Com essas duas eliminações, é possível eliminar as estratégias *esquerda*, pois, para o jogador *jogador coluna*, é estritamente dominada pela estratégia *centro*, e por fim, para o jogador *jogador linha*, a estratégia *cima* é estritamente dominada pela estratégia *meio*.

No final da eliminação iterada de estratégias estritamente dominadas. sobra apenas uma única célula da Tabela 4, como demonstrado na Tabela 5, e é chamado de **equilíbrio de estratégia dominante**.

	Centro
Meio	(3,3)

Tabela 5: Final do exemplo de dominância estrita iterada, fonte: (SPANIEL, 2011)

2.3.10.3 Estratégias mistas

Considere a situação a seguir: Dois caçadores devem decidir o que caçar no dia e levar o equipamento apropriado. Eles sabem que, no local de caça, existem duas lebres, que valem uma unidade de carne cada, e um veado, que vale seis unidades de carne. O veado vale mais carne dividindo para cada um do que a soma das duas lebres, mas é preciso o auxílio do outro caçador para caçar um veado, enquanto as lebres podem ser caçadas sem nenhuma ajuda (SPANIEL, 2011). Estas informações são condensadas na Tabela 6.

		Coluna	
		VEADO	LEBRE
Linha	VEADO	(3,3)	(0,2)
	LEBRE	(2,0)	(1,1)

Tabela 6: Caça ao Veado, fonte: (SPANIEL, 2011)

Pode parecer óbvio que o resultado desse jogo é os dois caçadores escolherem caçar *Veado*, pois o ganho para ambos é maior do que em qualquer outra situação, mas esse não é bem o caso.

2.3.10.4 Teorema Minimax

2.3.10.5 Equilíbrio de Nash

2.3.10.6 Estratégias Mistas

2.3.11 Análise primitiva do jogo *le Her*

O objetivo do jogo *le Her* é terminar o jogo com a carta mais alta, sendo que o baralho é contado de Ás (A) à Rei (K). Essa versão reduzida podia ser jogada apenas com dois jogadores, um deles chamado *dealer* e outro *receiver*. O *dealer* embaralha as cartas e distribui uma carta para o *receiver* e uma para si. O *receiver* tem a escolha de manter sua carta ou trocá-la com o *dealer*, e em seguida o *dealer* tem a mesma opção de manter ou de trocar sua carta com uma carta nova do baralho. A única regra que impede a troca é o caso da carta recebida ser um Rei (K), neste caso a troca deve ser desfeita e o jogador mantém sua carta original.

2.4 O Jogo *Big Points*

2.4.1 Conceito do Jogo

Big Points é um jogo abstrato e estratégico com uma mecânica de colecionar peças. São cinco peões de cores distintas, que podem ser usadas por qualquer jogador, para percorrer um caminho de discos coloridos até chegar ao pódio. Durante o percurso, os jogadores coletam alguns destes discos. A pontuação de cada jogador é determinada a partir da ordem de chegada dos peões ao pódio e a quantidade de discos adquiridos de cada cor. Ganha o jogador com a maior pontuação.

2.4.2 Regras do Jogo

O jogo *Big Points* pode ser jogado de dois a cinco jogadores. No seu turno, o jogador escolhe qualquer um dos cinco peões, que possuem as cores vermelha, verde, azul, amarela, violeta e preta, e o move para cima do próximo disco de sua mesma cor. Em seguida, o jogador deve pegar o próximo disco disponível[5] à frente ou atrás deste peão. Caso não haja discos atrás (à frente) do peão, o jogador deve pegar o disco que está à frente (atrás). Existem sete cores de discos, sendo elas as cores branca, preta e as cinco cores dos peões. Caso o jogador já possua um disco preto no começo do seu turno, tal jogador pode escolher descartá-lo após sua jogada para realizar um segundo movimento. Este movimento pode ser com qualquer cor de peão, como uma jogada normal, mas dessa vez o movimento pode ser feito para trás, se houver discos de sua cor para mover.

Se o jogador escolher um peão o qual não possua discos de sua cor à frente, então este peão deve ir para a posição mais alta da escada, posicionada ao final da trilha de discos, e o jogador pega um disco daquela mesma cor, reservado ao lado da escada. Uma vez em cima da escada, nenhum jogador pode escolher aquele peão para movimentá-lo. O jogo termina quando todos os peões estiverem em cima da escada e, consequente, nenhuma jogada possível restante.

A pontuação de cada jogador é contada de acordo com a posição dos peões na escada e a quantidade de discos de suas cores na mão do jogador. Os discos da cor do peão no topo da escada valem 4, os discos da cor do peão na posição anterior valem 3, e assim sucessivamente até os discos da cor do peão na última posição que não valem ponto algum. O disco branco vale um ponto para cada cor de disco diferente que o jogador possue em sua mão, com a exceção da cor branca[6].

[5] É dito indisponível aqueles discos que já foram pegos por algum jogador ou que possuem um peão em cima.

[6] A pontuação máxima de um disco branco é igual a 6. Um ponto para cada uma das seguintes cores de disco: Vermelho, Verde, Azul, Amarelo, Violeta e Preto.

3 Metodologia

Este capítulo descreve os passos para a realização deste trabalho, explicando os equipamentos e softwares utilizados.

3.1 Levantamento Bibliográfico

Após a definição do tema, foi realizado uma pesquisa a respeito dos conceitos básicos da Teoria dos Jogos, a existência de trabalhos semelhantes e materiais suficientes para a realização deste trabalho.

3.2 Lista de Equipamentos e Softwares

Para a realização deste trabalho foi utilizado um computador da linha *Inspiron 14Rx* fabricado pela *Dell*, no qual possui processador *Intel Core i7* de 2,2 GHz, GPU *NVIDIA GeForce GT 630M* de 1GB e 8GB de memória RAM. Quanto aos softwares utilizados, foram apenas os pacotes básicos de desenvolvimento em C/C++, incluindo *GCC* e *GNU Makefile*. Para serviço de versionamento foi utilizado o *GitHub* e para controle de tarefas e *issues* foi utilizado o *waffle.io*.

3.3 Análise combinatória

A possibilidade de encontrar a solução para este jogo depende da quantidade de estados existentes e da quantidade de espaço cada estado ocupará na memória. Se for possível calcular para uma partida, foi feito uma análise combinatória para descobrir a possibilidade de encontrar uma solução para várias partidas distintas.

3.3.1 Espaço de armazenamento

A quantidade de memória necessária para armazenar um *estado* do jogo depende das características que descrevem um *estado*. Como dito anteriormente, o jogo pode ter até cinco jogadores, possui um tabuleiro com 55 discos, uma escada com cinco degraus, e cinco peões no qual a posição varia entre 0 e 60. Considerando que o estado inicial do tabuleiro é aleatório mas conhecido desde o começo da partida e que os discos não mudam de posição, é possível representar o tabuleiro com uma máscara binária indicando se o disco está ou não disponível, $^{55\ D_{iscos}}/_{8\ B_{its}}$. A posição de cada peão pode ser representada por um *char*, pois varia apenas entre 0 e 60, sendo que 0 indica que o peão

não está no tabuleiro, 1 a 55 indica a posição que o peão está no tabuleiro e 56 a 60 indica a posição na escada. Por fim, para representar a mão de cada jogador, foi utilizado sete *chars*, cada um indicando a quantidade de discos de uma determinada cor, e como são cinco jogadores, tem-se $5 \cdot 7$. Todos esses *bytes* são somados como demonstrado na Equação e.q. Bytes na memória, totalizando 47 bytes na memória por estado da partida.

$$\begin{aligned} Bytes &= \frac{55}{8} + 5 + 5 \cdot 7 \\ Bytes &= 47\ bytes \end{aligned} \qquad \text{(e.q. Bytes na memória)}$$

3.3.2 Quantidade de partidas distintas

As características importantes que distinguem uma partida da outra no jogo *Big Points* são: (a) a quantidade de jogadores e; (b) a ordem dos discos no tabuleiro. O jogo pode ser jogado de dois a cinco jogadores (J) e o tabuleiro possui 55 discos no total (D_T), 5 das cores *branco* (D_W) e *preto* (D_K), e 9 das cores *Vermelho* (D_R), *Verde* (D_G), *Azul* (D_B), *Amarelo* (D_Y) e *Violeta* (D_V). Com isso, tem-se que a quantidade de jogos distintos é a combinação dos discos e dos jogadores. Considerando $\#D_{L1} = \#D_T - \#D_W$ como o restante dos discos após a primeira combinação, $\#D_{L2} = \#D_{L1} - \#D_K$ o restante da segunda combinação, e assim sucessivamente, temos a Equação e.q. Quantidades de Partidas Distintas.

$$\begin{aligned} Partidas &= (\#J - 1) \cdot \binom{\#D_T}{\#D_W} \cdot \binom{\#D_{L1}}{\#D_K} \cdot \binom{\#D_{L2}}{\#D_R} \cdot \binom{\#D_{L3}}{\#D_G} \cdot \binom{\#D_{L4}}{\#D_B} \cdot \binom{\#D_{L5}}{\#D_Y} \cdot \binom{\#D_{L6}}{\#D_V} \\ Partidas &= 4 \cdot \binom{55}{5} \cdot \binom{50}{5} \cdot \binom{45}{9} \cdot \binom{36}{9} \cdot \binom{27}{9} \cdot \binom{18}{9} \cdot \binom{9}{9} \\ Partidas &= 560'483'776'167'774'018'942'304'261'616'685'408'000'000 \\ Partidas &\approx 5 \times 10^{41} \end{aligned}$$

(e.q. Quantidades de Partidas Distintas)

Como demonstrado na Equação e.q. Quantidades de Partidas Distintas, a quantidade de partidas distintas é superior a 5×10^{41}. Considerando a possibilidade de solucionar uma partida do jogo por segundo, este resolução levaria mais do que 10^{34}

anos, como demonstrado na Equação e.q. Tempo de Computação das Partidas.

$$
\begin{aligned}
Anos &= \frac{N_{partidas\ distintas}}{partida/_{segundo} \cdot segundos/_{minuto} \cdot minutos/_{hora} \cdot horas/_{dia} \cdot dias/_{ano}} \\
Anos &= \frac{560'483'776'167'774'018'942'304'261'616'685'408'000'000}{1/_1 \cdot 60/_1 \cdot 60/_1 \cdot 24/_1 \cdot 365/_1} \\
Anos &= 96'526'964'154'064'571'465'728 \\
Anos &\approx 9 \times 10^{34}
\end{aligned}
$$

(e.q. Tempo de Computação das Partidas)

3.3.3 Quantidade de estados

Dado uma partida inicial $\gamma \in \Gamma$ de *Big Points*, sendo Γ o conjunto contendo todas as 5×10^{41} partidas distintas, foi calculado a quantidade de estados de γ. Para realizar este cálculo do número de estados, foi preciso determinar as características que definem um estado do jogo. De acordo com a Seção 2.4.2, tem-se que essas características são: (a) o estado do tabuleiro; (b) o estado dos peões; (c) o estado da escada; (d) o estado dos discos na mão dos jogadores e; (e) o jogador atual.

3.3.3.1 Quantidade de Estados do Tabuleiro

Já conhecendo a ordem de todos os discos no tabuleiro, o estado do tabuleiro foi descrito por uma máscara binária que representa a disponibilidade destes disco, para um peão mover-se para cima ou para ser pego por um jogador. O jogo possui 55 discos que podem ou não estar disponíveis, portando existem 55^2 estados diferentes para o tabuleiro.

3.3.3.2 Quantidade de Estados dos Peões

A próxima característica que define o estado de uma partida é o estado dos peões ou, mais precisamente, a posição deles. Um peão pode não estar no tabuleiro ainda (1), pode estar em cima de qualquer um dos 55 discos (1–56) e pode estar em cima de algum dos cinco degraus da escada (57–61). Como o jogo possue 5 peões, a quantidade de estados dos peões são $\text{Estado}_{[\text{peões}]} = 61^5$.

3.3.3.3 Quantidade de Estados da Escada

O terceiro estado importante é o estado da escada. Inicialmente a escada pode estar vazia, e então pode ter um peão de qualquer cor no topo. Para cada peão no topo da escada, é possível ter qualquer outra cor, que não seja aquela já escolhida, na posição

anterior à do topo. Seguindo essa lógica, tem-se que a quantidade de estados da escada é um somatório de arranjos como demonstrado na Equação e.q. Estado da escada.

$$
\begin{aligned}
Estado_{[escada]} &= \sum_{i=0}^{5} A_{5,i} \\
Estado_{[escada]} &= \frac{5!}{(5-0)!} + \frac{5!}{(5-1)!} + \frac{5!}{(5-2)!} + \frac{5!}{(5-3)!} + \frac{5!}{(5-4)!} + \frac{5!}{(5-5)!} \\
Estado_{[escada]} &= 326
\end{aligned}
$$

(e.q. Estado da escada)

3.3.3.4 Quantidade de Estados dos Discos

A outra característica é o estado dos discos na mão dos jogadores. Entendendo que a quantidade de discos é sempre a mesma ao longo do jogo e que a única coisa que altera é o local do disco (se está no tabuleiro ou na mão de algum jogador), então a seguinte equação sempre deve ser respeitada: $F + J_1 + J_2 + J_3 + J_4 + J_5 = 10$. Nessa simples representação, o F corresponde à quantidade de discos que estão fora da mão dos jogadores, J_1 à J_5 à quantidade de discos na mão dos jogadores de 1 à 5.

Uma outra abordagem para explicar tal estado é imaginar que cada disco corresponde ao caracter *o* e o símbolo + separa a mão dos jogadores e tabuleiro. Logo, é fácil de ver que no estado inicial, todos os discos estão no tabuleiro, sendo representado pelo estado $oooooooooo + + + + + = 10$. Caso o primeiro jogador consiga um disco daquela cor, o estado é alterado para $ooooooooo + o + + + + = 10$. Se, posteriormente, o jogador 4 colete 3 discos e o jogador 5 mais 5, o estado seria $o + o + + + ooo + ooooo = 10$. O cálculo para descobrir a quantidade de estados para cada disco comum é uma combinação de 15 elementos tomados de 5 a 5, como demonstrado na Equação e.q. Estado dos discos comuns, e para cada disco especial é a combinação de 10 elementos tomados de 5 a 5, como demonstrado na Equação e.q. Estado dos discos especiais.

$$
\begin{aligned}
Estado_{[discos_c]} &= \binom{15}{5} \\
Estado_{[discos_c]} &= \frac{15!}{5!(15-5)!} \\
Estado_{[discos_c]} &= 3003
\end{aligned}
$$

(e.q. Estado dos discos comuns)

$$
\begin{aligned}
Estado_{[discos_e]} &= \binom{10}{5} \\
Estado_{[discos_e]} &= \frac{10!}{5!(10-5)!} \\
Estado_{[discos_e]} &= 252
\end{aligned}
$$

(e.q. Estado dos discos especiais)

Ao final, tem-se que a quantidade de estados distintos dos discos é igual ao produto dos estados de cada disco individual, resultando na Equação e.q. Estado dos discos

$$\begin{aligned} Estado_{[discos]} &= \binom{15}{5} \cdot \binom{15}{5} \cdot \binom{15}{5} \cdot \binom{15}{5} \cdot \binom{15}{5} \cdot \binom{10}{5} \cdot \binom{10}{5} \\ Estado_{[discos]} &= 3003^5 \cdot 252^2 \\ Estado_{[discos]} &= 15'508'783'829'111'892'791'472 \\ Estado_{[discos]} &\approx 1 \times 10^{22} \end{aligned}$$

(e.q. Estado dos discos)

3.3.3.5 Quantidade de Estados do Jogador

A quantidade de estados distintos para o jogador atual é simplesmente a quantidade de jogadores, determinando assim a vez do jogador: $Estado_{[jogador]} = 5$.

3.3.3.6 Quantidade de Estados Total

Como demonstrado na Equação e.q. Caso Geral, a quantidade de estados existentes para uma partida é o produto de todos os estados. Considerando um caso geral, esse número excede 6×10^{37}, levando mais de 2×10^{30} anos para calcular todos os estados a uma velocidade de $1^{Estado}/_{segundo}$.

$$\begin{aligned} Estados &= 55^2 \cdot 61^5 \cdot 326 \cdot 3003^5 \cdot 252^2 \cdot 5 \\ Estados &= 64'586'224'969'618'554'770'829'159'998'863'764'000 \\ Estados &\approx 6 \times 10^{37} \end{aligned}$$

(e.q. Caso Geral)

$$\begin{aligned} Anos &= \frac{N_{estados\ distintos}}{{}^{estado}/_{segundo} \cdot {}^{segundos}/_{minuto} \cdot {}^{minutos}/_{hora} \cdot {}^{horas}/_{dia} \cdot {}^{dias}/_{ano}} \\ Anos &= \frac{64'586'224'969'618'554'770'829'159'998'863'764'000}{{}^{1}/_{1} \cdot {}^{60}/_{1} \cdot {}^{60}/_{1} \cdot {}^{24}/_{1} \cdot {}^{365}/_{1}} \\ Anos &= 2'048'015'758'803'226'720'862'434'492'416 \\ Anos &\approx 2 \times 10^{30} \end{aligned}$$

(e.q. Tempo de Computação dos Estados)

3.3.4 Podas

Podas são limites impostos à busca na árvore de possibilidades de jogadas. A seguir estão as possíveis podas descobertas para este problema.

3.3.4.1 Poda por Posição do Peão

No cálculo da quantidade de estados distintos para a posição dos peões foi levado em consideração que um peão poderia estar em qualquer uma de 61 posições. Na verdade, um peão só pode ficar em cima dos discos de sua própria cor, reduzindo esse número para 1 + 9 + 5 = 11, sendo que o peão pode estar fora do tabuleiro (1), em cima de algum disco de sua cor (9) e em algum degrau da escada (5). Porém, mesmo com essa poda, o número de estados distintos do jogo, de acordo com a Equação e.q. Poda por posição, é superior a 1×10^{34}, demorando mais do que 3×10^{26} anos para resolver um jogo (Equação e.q. Tempo de Computação da Poda 1).

$$
\begin{aligned}
Estados &= 55^2 \cdot 11^5 \cdot 326 \cdot 3003^5 \cdot 252^2 \cdot 5 \\
Estados &= 12'315'559'641'057'482'993'164'099'882'764'000 \qquad \text{(e.q. Poda por posição)} \\
Estados &\approx 1 \times 10^{34}
\end{aligned}
$$

$$
\begin{aligned}
Anos &= \frac{N_{estados\ distintos}}{estado/_{segundo} \cdot segundos/_{minuto} \cdot minutos/_{hora} \cdot horas/_{dia} \cdot dias/_{ano}} \\
Anos &= \frac{12'315'559'641'057'482'993'164'099'882'764'000}{1/_1 \cdot 60/_1 \cdot 60/_1 \cdot 24/_1 \cdot 365/_1} \\
Anos &= 390'523'834'381'579'220'798'144'512 \\
Anos &\approx 3 \times 10^{26}
\end{aligned}
$$

(e.q. Tempo de Computação da Poda 1)

3.4 Desenvolvimento do Jogo Eletrônico

Foi desenvolvido um programa para calcular a quantidade aproximada de estados distintos de uma partida reduzida do jogo *Big Points*. Dado d discos e p peões, o programa realiza todos os movimentos possíveis no primeiro turno, movendo uma vez com cada peão, além de capturar o disco atrás e à frente do peão. Então, recursivamente, a função de mover e coletar discos é chamada, cobrindo cada possibilidade de jogada.

O cálculo da quantidade de estados difere do proposto da Seção 3.3.3 pois não representa o tabuleiro como uma máscara binária, mas sim como um conjunto de caracteres. O caracter 0 representa que aquele disco naquela posição está indisponível, ou seja, já foi capturado por um jogador. Os caracteres 1,2,3,4 e 5 representam as cores Vermelha, Verde, Azul, Amarela e Violeta, respectivamente, e os caracteres R, G, B, Y, V representam os peões Vermelho, Verde, Azul, Amarelo e Violeta, respectivamente, percorrendo o tabuleiro até o fim. Não foi levado em consideração os discos na mão dos jogadores, e nem os estados da escada.

Código 3.1: *Quantidade de Estados Aproximados*

```
1
2  using namespace std;
3
4  static map<string,int> boards;
5  static map<int,string> masks;
6  static int counter = 0;
7
8  void add_board(string);
9  void move(string,int);
10 int move_pawn(string&, int);
11 void pick_right(string, int);
12 void pick_left(string, int);
13
14 static int num_discs = 2;
15 static int num_pawns = 2;
16
17 int main(int argc, char* argv[])
18 {
19         string board = "";
20
21         boards.clear();
22         masks.clear();
23         counter = 0;
24
25         int c;
26         while((c = getopt(argc, argv, "d:p:")) != -1) {
27                 switch (c) {
28                         case 'd':
29                         if(optarg) num_discs = atoi(optarg);
30                         break;
31
32                         case 'p':
33                         if(optarg) num_pawns = atoi(optarg);
34                         break;
35                 }
36         }
37
38         for (int disc = 0; disc < num_discs; disc++) {
39                 for (char pawn = 1; pawn <= (char) num_pawns; pawn++) {
40                         board.push_back(pawn+48);
41                 }
```

```
42          }
43          add_board(board);
44
45          for (int p = 1; p <= num_pawns; p++) {
46                  move(board,p);
47          }
48
49          fprintf(stderr,"P:␣%d\tD:␣%d\tTotal:␣%d\n", num_pawns, num_discs, boards.s
50          printf("%d\n", boards.size());
51
52          return 0;
53  }
54
55  int move_pawn(string& board, int pawn)
56  {
57          // Move Pawn
58          char c_pawn;
59          char n_char = '␣';
60          n_char = (char) pawn+48;
61
62          switch (pawn) {
63                  // Red
64                  case 1:
65                  c_pawn = 'R';
66                  break;
67
68                  // Green
69                  case 2:
70                  c_pawn = 'G';
71                  break;
72
73                  // Blue
74                  case 3:
75                  c_pawn = 'B';
76                  break;
77
78                  // Yellow
79                  case 4:
80                  c_pawn = 'Y';
81                  break;
82
83                  // Violet
```

```
84                  case 5:
85                  c_pawn = 'V';
86                  break;
87
88                  default:
89                  break;
90          }
91
92          int position = 0;
93          position = board.find_first_of(c_pawn);
94
95          if (position == (int) string::npos) {
96                  position = -1;
97          }
98          else {
99                  board.replace(position, 1, 1, n_char);
100         }
101
102         int pawn_pos = board.find_first_of(n_char, position+1);
103
104         if (pawn_pos == (int) string::npos) {
105                 return -1;
106         }
107
108         board.replace(pawn_pos, 1, 1, c_pawn);
109
110         return pawn_pos;
111 }
112
113 void pick_right(string board, int pawn_pos)
114 {
115         int position = board.find_first_not_of("RGBYV0", pawn_pos);
116         if (position != string::npos) {
117                 board.replace(position, 1, 1, '0');
118         }
119
120         for (int p = 1; p <= num_pawns; p++) {
121                 move(board,p);
122         }
123
124         return ;
125 }
```

```
126
127 void pick_left(string board, int pawn_pos)
128 {
129         int position = board.find_last_not_of("RGBYV0", pawn_pos);
130         if (position != string::npos) {
131                 board.replace(position, 1, 1, '0');
132         }
133
134         for (int p = 1; p <= num_pawns; p++) {
135                 move(board,p);
136         }
137
138         return ;
139 }
140
141 void move(string board, int pawn)
142 {
143         add_board(board);
144         int pawn_pos = move_pawn(board, pawn);
145         if (pawn_pos < 0) {
146                 return ;
147         }
148
149         // Picking discs
150         pick_right(board, pawn_pos);
151         pick_left(board, pawn_pos);
152
153         return ;
154 }
155
156 void add_board(string board)
157 {
158         if (boards.count(board) <= 0)
159         {
160                 boards[board.c_str()] = counter;
161                 masks[counter] = board.c_str();
162                 counter++;
163         }
164
165         return ;
166 }
```

4 Considerações Finais

Foi utilizado um *script bash* pra executar o programa e calcular o tempo de cada execução que, juntamente com a saída do programa de estados, permite estimar o tempo de execução para o cálculo dos estados.

4.1 Tempo de Execução

Dado $d \in [2,9]$ discos e $p \in [2,5]$ peões, a quantidade distintas de estados $[B]$ e o tempo (em segundos) do cálculo está representado nas Tabelas 7, 8, 9 e 10. Além das tabelas, um gráfico também foi desenhado com as informações (Figura 2) de crescimento da quantidade de estados de acordo com o número de peões e discos. Analisando as tabelas, o número de estados e, consequentemente, o tempo para calcular todos os estados, cresce rapidamente. Com o auxílio do gráfico, percebe-se que o crescimento é exponencial.

p	d	$[B]$	Seg
2	2	27	0
2	3	102	0
2	4	361	0
2	5	1251	0
2	6	4296	0
2	7	14746	1
2	8	50746	3
2	9	175230	19

Tabela 7: Quantidade de Estados e Tempo com P = 2

p	d	$[B]$	Seg
3	2	150	0
3	3	1219	1
3	4	9082	1
3	5	65195	19
3	6	457855	653
3	7	3173596	19929

Tabela 8: Quantidade de Estados e Tempo com P = 3

p	d	$[B]$	Seg
4	2	825	0
4	3	14907	2
4	4	243200	462

Tabela 9: Quantidade de Estados e Tempo com P = 4

p	d	$[B]$	Seg
5	2	4513	0
5	3	178898	505
5	4	6303528	526949

Tabela 10: Quantidade de Estados e Tempo com P = 5

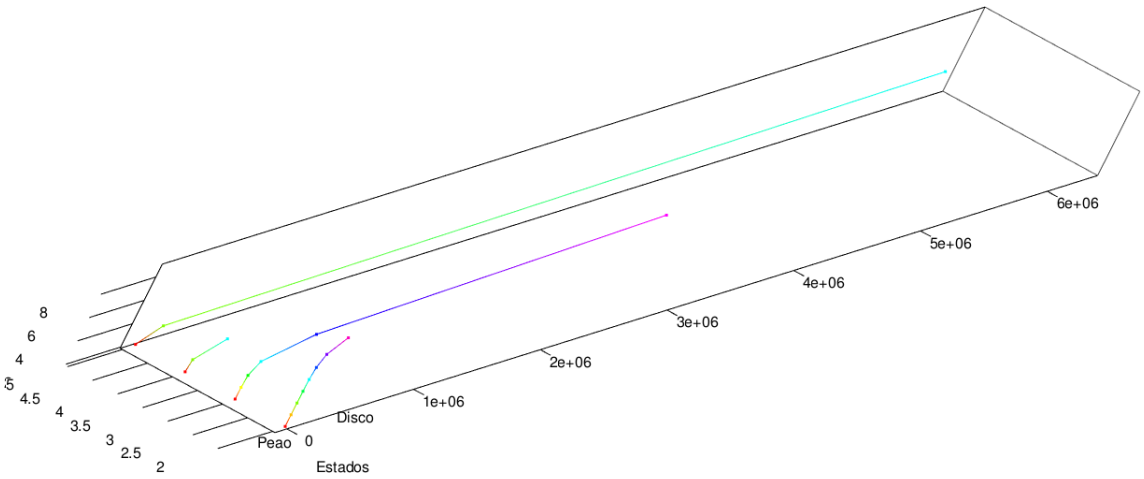

Figura 2: Crescimento de estados de acordo com número de *Peões* e *Discos*

5 Cronograma

6 Conclusão

Referências

JONES, A. J. *Game Theory*: Mathematical models of conflict. [S.l.: s.n.], 1980. Citado 2 vezes nas páginas 18 e 19.

PRAGUE, M. H. *Several Milestones in the History of Game Theory*. VII. Österreichisches Symposion zur Geschichte der Mathematik, Wien, 2004. 49–56 p. Disponível em: <http://euler.fd.cvut.cz/predmety/game_theory/games_materials.html>. Citado na página 17.

SARTINI, B. A. et al. *Uma Introdução a Teoria dos Jogos*. 2004. Citado 4 vezes nas páginas 15, 17, 19 e 21.

SPANIEL, W. *Game Theory 101*: The complete textbook. [S.l.: s.n.], 2011. Citado 6 vezes nas páginas 9, 17, 21, 22, 23 e 24.