UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIAS DA COMPUTAÇÃO – BACHARELADO

ANÁLISE DA APLICABILIDADE DE TÉCNICAS BASEADAS EM REGRAS E TÉCNICA DE MONTE-CARLO NO JOGO DE CARTAS RIKKEN

PAULO ROSA

BLUMENAU
2008

2008/2-17

PAULO ROSA

ANÁLISE DA APLICABILIDADE DE TÉCNICAS BASEADAS EM REGRAS E TÉCNICA DE MONTE-CARLO NO JOGO DE CARTAS RIKKEN

Trabalho de Conclusão de Curso submetido à Universidade Regional de Blumenau para a obtenção dos créditos na disciplina Trabalha de Conclusão de Curso II do curso de Ciências da Computação — Bacharelado.

Prof. Mauro Marcelo Mattos, Dr. - Orientador

BLUMENAU
2008

2008/2-17

ANÁLISE DA APLICABILIDADE DE TÉCNICAS BASEADAS EM REGRAS E TÉCNICA DE MONTE-CARLO NO JOGO DE CARTAS RIKKEN

Por

PAULO ROSA

Trabalho aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso II, pela banca examinadora formada por:

Presidente:	Prof. Mauro Marcelo Mattos, Dr. – FURB
Membro:	Prof. Adilson Vahldick, Mestre – FURB
Membro:	Prof. Dalton Solano dos Reis, Mestre – FURB

Blumenau, 17 de Dezembro de 2008

Dedico este trabalho a todos os professores, amigos e familiares que pacientemente me ensinaram a nunca desistir de minhas metas até alcançá-las.

AGRADECIMENTOS

Agradeço a Deus por nos dar saúde e paz para corrermos atrás de nossos objetivos.

Aos meus pais, eternos modelos de caráter e integridade que sempre depositaram confiança em mim e estiveram do meu lado quando precisei.

Aos meus irmãos pelos incentivos e convites para deixar um pouco de lado e sair um pouco de casa pra ver o sol.

À minha namorada Susan Braun que jamais negou ajuda e sempre esteve comigo nesta difícil caminhada dando incentivo, força, companheirismo e paciência.

Aos meus amigos entenderam minhas ausências nos compromissos para a elaboração deste trabalho.

À Wheb Sistemas pelo apoio e oportunidade para desenvolver meus conhecimentos.

Aos meus amigos Eduardo Paniz Mallmann e Moisés Baum que pacientemente se dispuseram a ajudar no presente trabalho.

Ao meu orientador Mauro Marcelo Mattos por estar sempre presente dando conselhos e auxiliando para a conclusão deste trabalho.

A sorte só favorece a mente preparada.

Isaac Asimov

RESUMO

A simulação de Monte-Carlo tem sido reconhecida como uma ferramenta de grande utilidade para tomadores de decisão tratarem situações como a melhor jogada a ser efetuada em um jogo de cartas. O presente trabalho tem como objetivo recriar o jogo Rikken e aplicar o método de Monte-Carlo, análise baseada em regras e as técnicas *Lose Trick Count* (LTC) e *Trick Procent Count* (TPC) objetivando comprovar os resultados obtidos em Vorsteveld (2007). Para tanto, o trabalho foi desenvolvido na IDE NetBeans,, utilizando a linguagem JAVA e o padrão de desenvolvimento *Model View Control* (MVC).

Palavras-chave: Rikken. Monte-Carlo. Inteligência artificial.

ABSTRACT

The Monte-Carlo simulation has been recognized as a very useful tool for decision-makers dealing with situations like the best move to be made in a card game. This paper aims to recreate the game Rikken and apply the Monte-Carlo simulation, analysis based on rules and techniques Lose Trick Count (LTC) and Trick Procent Count (TPC), aiming to demonstrate the results obtained in Vorsteveld (2007). Thus, the work was developed in the NetBeans IDE, using JAVA language and pattern of development Model View Control (MVC).

Key-words: Rikken. Monte-Carlo. Artificial inteligence.

LISTA DE ILUSTRAÇÕES

Figura 1 - Simulações de Monte-Carlo 19
Figura 2 - Jogo Rikken 20
Quadro 1- Regras para os tipos de jogos 22
Quadro 2 - Tipos de jogos 23
Quadro 3 - Fórmula para obter o valor de π 26
Figura 3 - Desenho geométrico para descobrir π 26
Figura 4 - Gráfico de freqüência por bloco 27
Figura 5 - Gráfico de freqüência acumulada 28
Figura 6 - Diagrama de casos de uso 30
Quadro 4 - Detalhamento do caso de uso UC01 30
Quadro 5 - Detalhamento do caso de uso UC02 31
Quadro 6 - Detalhamento do caso de uso UC03 31
Figura 7 - Diagrama de atividades 32
Figura 8 - Diagrama de classes 33
Figura 9 - Diagrama de atividades do jogador aleatório 34
Figura 10 - Diagrama de atividades do jogador baseado em regras 36
Quadro 7 - Fórmula de cálculo TPC 37
Quadro 8 - Um exemplo do cálculo usando TPC 37
Quadro 9 - Um exemplo do cálculo necessário para determinar o parceiro 38
Quadro 10 - Estratégias baseadas em regras usadas nos tipos de jogos Rik e Solo 8 39
Quadro 11 - Estratégia baseada em regras usadas no tipo de jogo Piek 39
Quadro 12 - Estratégia baseada em regras usadas no tipo de jogo Misere 40
Quadro 13 - Fórmula de cálculo para escolha do tipo de jogo 41
Quadro 14 - Exemplo de *Losing Trick Count* 42
Figura 11 - Jogador MC em ação 43
Figura 12 - Simulação jogador Monte-Carlo 43
Figura 13 - Jogo de cartas Rikken desenvolvido por Aernsbergen 45
Quadro 15 - Distribuição e ordenação das cartas 46
Quadro 16 - Margem para selecionar o tipo de jogo 47
Quadro 17 - Margem para obrigar escolha do tipo de jogo 47
Quadro 18 - Técnica para selecionar a melhor carta pelo jogador baseado em regras 48

Quadro 19 - Criação e distribuição das cartas para jogadores na simulação de Monte-Carlo . 49
Figura 14 - Tela das opções da aplicação 50
Figura 15 - Tela para configuração dos jogadores 51
Figura 16 - Aplicação Rikken 52
Figura 17 - Tela de análise de jogadas 53
Figura 18 - Resultado da análise de jogadas 54
Figura 19 - Tela para análise da licitação 55
Figura 20 - Resultado da análise de licitação 56
Figura 21 - Resultado da análise de licitação do jogador baseado em regras para os tipos de jogos Rik e Solo 8 (Vorsteveld) 57
Figura 22- Resultado da análise de licitação do jogador baseado em regras para os tipos de jogos Rik e Solo 8 57
Figura 23 - Resultado da análise de licitação do jogador baseado em regras para o tipo de jogo Piek (Vorsteveld) 58
Figura 24 - Resultado da análise de licitação do jogador baseado em regras para o tipo de jogo Piek 58
Figura 25 - Resultado da análise de licitação do jogador baseado em regras para o tipo de jogo Misere (Vorsteveld) 59
Figura 26 - Resultado da análise de licitação do jogador baseado em regras para o tipo de jogo Misere 59
Figura 27 - Resultado da análise de licitação do jogador Monte-Carlo para os tipos de jogos Rik e Solo 8 (Vorsteveld) 60
Figura 28 - Resultado da análise de licitação do jogador Monte-Carlo para os tipos de jogos Rik e Solo 8 60
Figura 29 - Resultado da análise de licitação do jogador Monte-Carlo para o tipo de jogo Piek (Vorsteveld) 61
Figura 30 - Resultado da análise de licitação do jogador Monte-Carlo para o tipo de jogo Piek 61
Figura 31 - Resultado da análise de licitação do jogador Monte-Carlo para o tipo de jogo Misere (Vorsteveld) 62
Figura 32 - Resultado da análise de licitação do jogador Monte-Carlo para o tipo de jogo Misere (Vorsteveld) 62
Figura 33 - Resultado da análise do jogador baseado em regras jogando Rik (Vorsteveld) ... 63
Figura 34 - Resultado da análise do jogador baseado em regras jogando Rik 63

Figura 35 - Resultado da análise do jogador baseado em regras jogando Solo 8 (Vorsteveld)64
Figura 36 - Resultado da análise do jogador baseado em regras jogando Solo 8 64
Figura 37 - Resultado da análise do jogador baseado em regras jogando Piek (Vorsteveld) .. 64
Figura 38 - Resultado da análise do jogador baseado em regras jogando Piek..... 65
Figura 39 - Resultado da análise do jogador baseado em regras jogando Misere (Vorsteveld) 65
Figura 40 - Resultado da análise do jogador baseado em regras jogando Misere..... 65
Figura 41 - Resultado da análise do jogador Monte-Carlo jogando Rik (Vorsteveld)..... 66
Figura 42 - Resultado da análise do jogador Monte-Carlo jogando Rik 66
Figura 43 - Resultado da análise do jogador Monte-Carlo jogando Solo 8 (Vorsteveld)..... 67
Figura 44 - Resultado da análise do jogador Monte-Carlo jogando Solo 8 67
Figura 45 - Resultado da análise do jogador Monte-Carlo jogando Piek (Vorsteveld)..... 67
Figura 46 - Resultado da análise do jogador Monte-Carlo jogando Piek 68
Figura 47 - Resultado da análise do jogador Monte-Carlo jogando Misere (Vorsteveld)..... 68
Figura 48 - Resultado da análise do jogador Monte-Carlo jogando Misere 68
Quadro 20 - Resultado da análise comparativa da licitação..... 73
Quadro 21 - Resultado da análise comparativa das jogadas 74

LISTA DE SIGLAS

IA – Inteligência Artificial

LTC – *Lose Trick Count*

MB – Margem Bruta

MC – Monte-Carlo

MMC – Método de Monte Carlo

MVC – *Model View Control*

NPC – *Non-Player Character*

TPC – *Trick Procent Count*

SUMÁRIO

1 INTRODUÇÃO 14
1.1 OBJETIVOS DO TRABALHO 15
1.2 ESTRUTURA DO TRABALHO 16
2 FUNDAMENTAÇÃO TEÓRICA 17
2.1 INTELIGÊNCIA ARTIFICIAL EM JOGOS 17
2.2 MÉTODO DE MONTE-CARLO 18
2.3 RIKKEN 19
2.3.1 Regras do jogo Rikken 20
2.3.2 Licitação 23
2.3.3 Jogadores 24
2.3.3.1 Jogador aleatório 24
2.3.3.2 Jogador baseado em regras 25
2.4 TRABALHOS CORRELATOS 25
2.4.1 Estimativa para Pi (π) 26
2.4.2 Análise de risco utilizando a simulação de Monte-Carlo 27
3 DESENVOLVIMENTO DO TRABALHO 29
3.1 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO 29
3.2 ESPECIFICAÇÃO 29
3.2.1 Casos de uso 30
3.2.2 Diagrama de atividades 31
3.2.3 Diagrama de classes 32
3.3 JOGADORES 34
3.3.1 Jogador aleatório 34
3.3.1.1 Licitação do jogador aleatório 35
3.3.1.2 Jogando com o jogador aleatório 35
3.4 JOGADOR BASEADO EM REGRAS 35
3.4.1 Licitação do jogador baseado em regras 36
3.4.2 Jogando com o jogador baseado em regras 38
3.5 JOGADOR MONTE-CARLO 40
3.5.1 Licitação do jogador Monte-Carlo 41
3.5.2 Jogando com o jogador Monte-Carlo 42

3.6 IMPLEMENTAÇÃO 44
3.6.1 Técnicas e ferramentas utilizadas 44
3.6.1.1 Contexto da aplicação 44
3.6.1.2 Classe `ControlaCartas` 45
3.6.1.3 Classe `Jogador` 46
3.6.1.4 Simulação de Monte-Carlo 48
3.6.1.5 Padrão Model View Controller (MVC) 49
3.6.2 Operacionalidade da implementação 50
3.7 RESULTADOS E DISCUSSÃO 56
3.7.1 Análise da licitação 56
3.7.2 Análise dos jogos 62
4 CONCLUSÕES 69
4.1 EXTENSÕES 69
REFERÊNCIAS BIBLIOGRÁFICAS 71
APÊNDICE A – Tabela comparativa do resultado da análise de licitação 73
APÊNDICE B – Tabela comparativa do resultado da análise das jogadas 74

1 INTRODUÇÃO

Com o desenvolvimento dos computadores na metade do século 20, uma nova era de jogos está nascendo. Todos os tipos de jogos que são jogados sobre tabuleiros podem ser visualizados na tela do computador. Mas, em vez de usar o computador apenas como uma interface para o jogo, o computador também pode participar como um jogador. Para agir como um jogador, o computador precisa de um comportamento, ou seja, é necessário utilizar técnicas para despertar a atenção das pessoas. A partir dessa necessidade é que se originou o domínio chamado jogador *Non-Player Character*[1] (NPC).

Um exemplo onde verifica-se este tipo de jogador é o jogo de cartas Rikken. Original da Holanda, este jogo é visto como uma variação de diversos jogos que seguem as mesmas regras, mas com objetivos diferentes. Normalmente o Rikken é jogado por quatro elementos, sendo que cada um pode escolher um tipo de jogo diferente. O objetivo de cada tipo de jogo é ganhar certo número de truques. Outras propriedades dos diferentes tipos de jogos incluem a participação de um parceiro e os prêmios para o vencedor dependendo do tipo de jogo.

Segundo Vorsteveld (2007, p. 2), nem todas as técnicas de Inteligência Artificial (IA) são úteis para todos os tipos de jogos, sendo estes caracterizados em:

a) jogos contendo informações perfeitas ou imperfeitas;

b) jogos estocásticos ou determinísticos.

Assim sendo, um jogador possui informações perfeitas se ele conhece todas as ações possíveis disponíveis para cada jogador (ou seja, consegue construir uma árvore completa do jogo) e todas as respostas potenciais de cada jogador. Se o jogador não possui todas estas informações, ele possui informação imperfeita. Jogos estocásticos são aqueles onde há certo grau de aleatoriedade (como jogar um dado ou sortear cartas). Xadrez é um exemplo de um jogo determinístico com informações perfeitas, *Gamão*[2] é um jogo com propriedades estocásticas e *Bridge*[3] é um exemplo de um jogo determinístico com informações imperfeitas (GIBBONS, 1992 apud VORSTEVELD, 2007, p. 2).

---

[1] *Non-player character* é um personagem que não é controlado por um jogador mas se envolve de alguma forma no jogo.

[2] *Gamão* é um jogo de tabuleiro com 24 casas (tradicionalmente decoradas com triângulos isósceles), agrupados 6 a 6, dividindo-o em quatro partes, duas para cada lado. Para a manipulação das jogadas são usadas 15 peças de duas cores distintas, todas iguais, cada jogador manipulando a sua respectiva cor.

[3] *Bridge* é um jogo inglês com cartas de baralho comum (sem curinga), de quatro jogadores, dois contra dois. O objetivo é fazer pontos. São quatro grupos de 13 cartas representando um naipe (espadas, copas, ouros e paus).

Para a elaboração de NPC, existem diversas técnicas que podem ser implementadas, entre elas, o método de Monte-Carlo. Segundo Minetto (2006, p. 1), este método explora as propriedades estatísticas dos números aleatórios para assegurar que o resultado correto é computado. Para resolver um problema através desta técnica, é usada uma série de tentativas aleatórias. A precisão do resultado final depende em geral do número de tentativas. Esse equilíbrio entre a precisão do resultado e o tempo de computação é uma característica extremamente útil do método de Monte-Carlo.

Segundo Vorsteveld (2007, p. 50) em uma situação de simulação, o jogador que utilizou o método de Monte Carlo venceu 8 de 12 jogos que participou. Utilizando a técnica baseada em regras o jogador venceu 10 dos 12 jogos efetuados, demonstrando também que método utilizado foi mais bem aplicado em 3 dos 4 jogos. Desta forma, concluiu-se que a técnica baseada em regras obteve maior sucesso comparada a técnica de Monte-Carlo.

Diante do exposto, o presente trabalho busca reproduzir os resultados da pesquisa desenvolvida em 2007 na Universidade de Maastricht, cujo objetivo foi identificar qual técnica de IA deveria ser utilizada para criar o melhor agente NPC para o jogo Rikken.

1.1 OBJETIVOS DO TRABALHO

O objetivo deste trabalho é desenvolver um jogo de computador que implemente o jogo de cartas Rikken utilizando NPCs baseados em diferentes técnicas de IA.

Os objetivos específicos do trabalho são:

a) implementar a funcionalidade do jogo e os modelos de NPC descritos em Vorsteveld (2007);
b) construir e executar a mesma bateria de testes objetivando validar a implementação;
c) comparar os resultados obtidos com aqueles relatados em Vorsteveld (2007).

Para que a análise comparativa tenha validade, devem ser considerados também como objetivos secundários:

a) a construção de três tipos de NPCs com suporte de IA utilizando as técnicas de (i) jogadas aleatórias, (ii) comportamento baseado em regras e (iii) comportamento baseado na técnica de Monte-Carlo;
b) o desenvolvimento de quatro variações do jogo Rikken (Rik, Solo 8, Piek, Misere).

1.2 ESTRUTURA DO TRABALHO

Esta monografia divide-se em fundamentação teórica, desenvolvimento do trabalho e conclusões. No capítulo de fundamentação teórica são mostradas as informações que o autor julga necessária para o bom entendimento do desenvolvimento deste trabalho e das técnicas em que ele foi baseado. Com tais conceitos apresentados são descritas as técnicas utilizadas no desenvolvimento da partida, na seleção do tipo de jogo e na escolha do melhor parceiro, juntamente com a apresentação de trabalhos correlatos. Logo após são apresentados os resultados obtidos com a aplicação e comparados com a bateria de testes descritos por Vorsteveld (2007). Por fim, é apresentada a conclusão, onde são destacados os objetivos alcançados e sugeridas algumas extensões a este projeto.

2 FUNDAMENTAÇÃO TEÓRICA

No presente capítulo são apresentados alguns aspectos teóricos referentes ao trabalho. A seção 2.1 apresenta uma visão geral do que representa a utilização de inteligência artificial em jogos. A seção 2.2 apresenta as definições da técnica de Monte-Carlo. A seção 2.3 aborda a definição do jogo Rikken. O capítulo é finalizado com a apresentação de trabalhos correlatos.

2.1 INTELIGÊNCIA ARTIFICIAL EM JOGOS

Atualmente, a Inteligência Artificial, ou IA, abrange uma enorme variedade de subcampos, desde áreas de uso geral, como aprendizado e percepção, até tarefas específicas como jogos de xadrez, demonstração de teoremas matemáticos e diagnóstico de doenças (RUSSEL; NORWIG, 2004, p. 157).

Segundo Kishimoto (2004), a principal diferença entre a IA acadêmica e a IA para jogos é o objetivo que cada uma busca. No primeiro caso, o objetivo é buscar solução para problemas extremamente difíceis, como imitar o reconhecimento que os seres humanos são capazes de realizar (reconhecimento facial e de imagens e objetos, por exemplo), entender e construir agentes inteligentes.

No segundo caso, o objetivo de usar inteligência artificial é a diversão. Sua importância é quanto aos resultados que o sistema irá gerar, e não como o sistema chega até os resultados; ou seja, o problema não é como o sistema pensa, mas sim como ele age. Isso se deve pelo fato que os jogos eletrônicos são negócios – os consumidores destes produtos compram em busca de diversão, e não lhes interessa como a inteligência de um personagem foi criada, desde que ela transforme o jogo divertido e desafiador, além, claro, de tomar decisões coerentes com o contexto do jogo (KISHIMOTO, 2004).

Segundo Russel e Norwig (2004, p. 157), para os pesquisadores de IA, a natureza abstrata de jogos os torna um assunto atraente para estudo. É fácil representar o estado de um jogo e, em geral, os agentes se restringem a um pequeno número de ações cujos resultados são definidos por regras precisas.

Um dos problemas encontrados sobre IA na indústria de jogos eletrônicos é a grande

variedade de gênero dos jogos existentes e os comportamentos dos personagens, resultando uma interpretação ampla do que é considerado IA para jogos. Há desenvolvedores que consideram a interface do jogo com o usuário parte da área de IA, assim como outros consideram algoritmos de movimento e colisão também como IA (KISHIMOTO, 2004).

2.2 MÉTODO DE MONTE-CARLO

O Método de Monte Carlo (MMC) é um método estatístico utilizado em simulações estocásticas com diversas aplicações em áreas como a física, matemática e biologia. Esta técnica tem sido utilizada há bastante tempo como forma de obter aproximações numéricas de funções complexas.

Segundo Peralta (2000, p. 1), o método tem como base a utilização de números aleatórios para a resolução de um conjunto grande de problemas.

A maior parte dos algoritmos de Monte Carlo manipula variáveis aleatórias uniformemente distribuídas e independentes, ou seja, assume-se que as variáveis independentes obedecem a uma Distribuição de Probabilidade (DP) ρ que satisfaz $\rho_u(x) = 1$ para x pertencente ao intervalo [0,1] e $\rho_u(x) = 0$, caso contrário. Destas variáveis aleatórias uniformes é possível obter outras distribuições não uniformes teóricas ou empíricas (SOUTO, 2006, p. 48).

Segundo Teknomo (2006), o MMC pode ser definido como um método para gerar dados aleatórios baseados em alguns conhecimentos distribuídos para experimentos numéricos, embora nem todos os métodos que envolvem números aleatórios podem ser categorizados como uma técnica de Monte Carlo.

Souto (2006, p. 51) descreve que para construir um algoritmo de Monte Carlo, primeiro cria-se um algoritmo que gera estados novos V aleatoriamente a partir dos estados U com um conjunto de probabilidades $g(U \rightarrow V)$, e depois aceita-se ou rejeita-se estes estados com grau de aceitação $A(U \rightarrow V)$. Isso satisfará todos os requisitos para as probabilidades de transição, e portanto produzir uma lista de estados para atingir a um equilíbrio.

Quando traduzido para o campo de jogos, isto se resume em simular movimentos aleatórios para obter uma indicação do resultado médio de uma determinada jogada, conforme ilustrado na Figura 2.

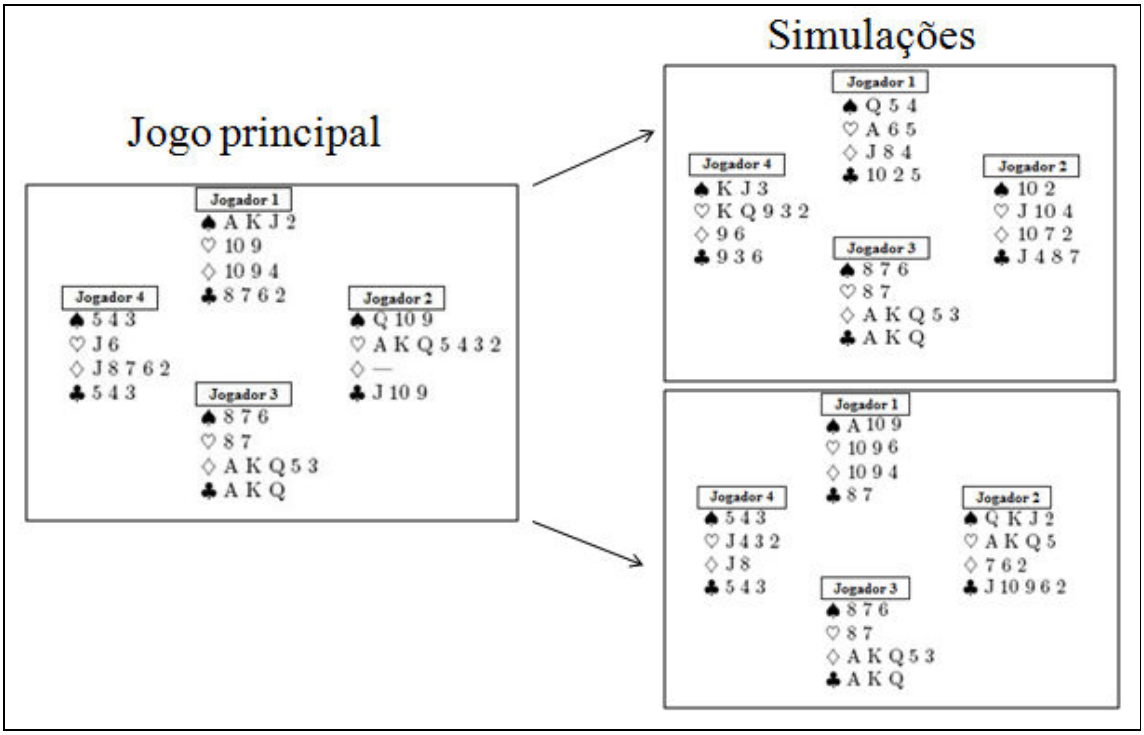


Figura 1 - Simulações de Monte-Carlo

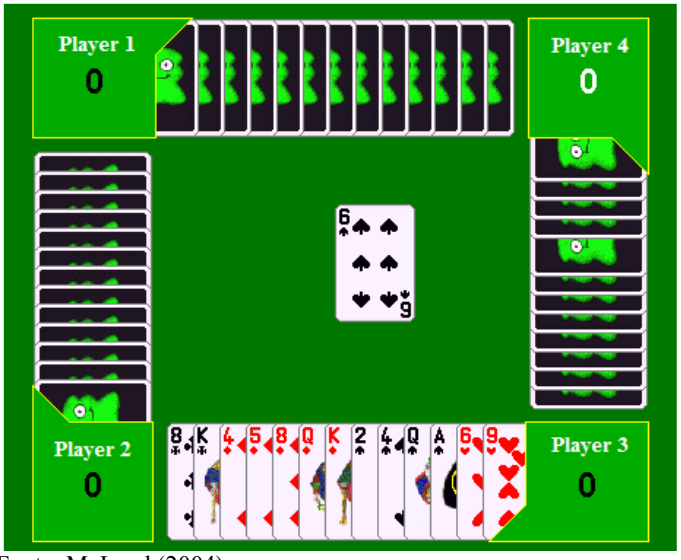
Nesta imagem nota-se que o jogador 3 está utilizando a técnica de Monte-Carlo, desta forma, este jogador realiza simulações de partidas com as mesmas cartas do jogo principal, enquanto os demais jogadores tem suas cartas re-distribuídas.

2.3 RIKKEN

De origem holandesa, o jogo de cartas Rikken é facilmente encontrado na cidade de Brabant. Este jogo normalmente é jogado por quatro jogadores conforme ilustrado na Figura 1. Uma partida é iniciada após a distribuição do mesmo número de cartas através de todos os participantes pelo *dealer*[4].

Depois de jogada a primeira carta da rodada, cartas com o mesmo naipe da primeira carta devem ser jogadas. Dentre todas as cartas de naipe igual, a maior ganha o truque. Caso o jogador não possua uma carta do mesmo naipe, ele deve escolher uma carta aleatória, mas, perde a chance de ganhar o truque.

[4] Jogador escolhido para efetuar a distribuição das cartas entre todos os participantes do jogo.



Fonte: McLeod (2004).

Figura 2 - Jogo Rikken

Este jogo é caracterizado por ser um *Plain Trick Game*, ou seja, o valor de truques não depende das cartas que o jogador possui. O objetivo basicamente é ganhar um número *x* de jogadas ou o maior número de truques possíveis, dependendo do tipo jogo escolhido. Existem também tipos de jogos simples onde, para vencer a partida, o jogador deve perder truques.

2.3.1 Regras do jogo Rikken

O jogador a esquerda do *dealer* sempre começa a partida, exceto nos tipos de jogos Open Piek, Open Misere e Solo, onde o *dealer* inicia o jogo.

Qualquer carta pode ser jogada e os demais participantes devem seguir o naipe da carta jogada. Os jogadores que não possuírem cartas do naipe jogado pelo primeiro jogador da rodada, devem jogar outra carta qualquer, mas só possuem a chance de ganhar o truque se jogarem uma carta do naipe *trump* escolhido pelo declarante.

Uma carta *trump* pode ser jogada a qualquer momento desde que o jogador não possua uma carta do naipe vigente da rodada. Se duas ou mais cartas do naipe *trump* forem jogadas na mesma rodada, a maior vence.

A sequência das cartas da maior para a menor é a seguinte: A – K – Q – J – 10 – 9 – 8 – 7 – 6 – 5 – 4 – 3 – 2.

O jogador que jogar a maior carta do mesmo naipe da carta jogada pelo primeiro jogador ganha o truque, exceto nos casos onde uma carta *trump* é jogada.

Um participante deve obrigatoriamente jogar uma carta do naipe definido na rodada

caso possuir esta carta. Se isto não acontecer, uma regra do jogo é infligida, causando a punição de uma moeda para cada adversário. Uma visão geral dos tipos de jogos é mostrada na Tabela 1.

Tipo de jogo	Meta
Pas (fold)	Quando as cartas não são boas o suficiente para realizar um lance maior que o atual.
Rik	Meta é ganhar pelo menos 8 truques. Declarante decide sobre o *trump* e solicita um Ás para definir a parceria.
Rik better	Mesma meta e opções que se aplicam a Rik, exceto que o *trump* é sempre copas.
Solo 8	Meta é ganhar pelo menos 8 truques jogando sozinho. Declarante decide sobre o *trump*.
Troela	Meta é ganhar pelo menos 8 truques. Este tipo de jogo só é possível quando o declarante possui 3 ases. O jogador que possui o quarto ás se torna seu parceiro. O parceiro decide o *trump*.
Piek	Meta é ganhar somente 1 truque sem parceiro nem *trump*.
Solo 9	Meta é ganhar pelo menos 9 truques jogando sozinho. Declarante decide sobre o *trump*.
Misere	Meta para vencer é se abster de vencer os truques.
Solo 10	Meta é ganhar pelo menos 10 truques jogando sozinho. Declarante decide sobre o *trump*.
Open Piek	Mesma meta e opções que se aplicam a Piek, mas após a primeira carta da segunda rodada o declarante deve revelar suas cartas.
Solo 11	Meta é ganhar pelo menos 11 truques jogando sozinho. Declarante decide sobre o *trump*.
Open Piek+	Mesma meta e opções que se aplicam a Open Piek, mas, os jogadores estão autorizados a falar sobre sua estratégia.
Open Misere	Mesma meta e opções que se aplicam a Misere, mas após a primeira carta da segunda rodada o declarante deve revelar suas cartas.
Solo 12	Meta é ganhar pelo menos 12 truques jogando sozinho. Declarante decide sobre o *trump*.
Open Misere+	Mesma meta e opções que se aplicam a Misere, mas os jogadores estão autorizados a falar sobre sua estratégia.
Solo	Meta é ganhar pelo menos 13 truques jogando sozinho. Declarante decide sobre o *trump*.

Quadro 1- Regras para os tipos de jogos

2.3.2 Licitação

A rodada de licitação é onde o tipo de jogo é escolhido e, se for necessário, o parceiro e um naipe *trump*.

Para decidir qual tipo de jogo será jogado durante a rodada, um leilão é realizado. Cada jogo representa um tipo de lance e o mais alto será jogado. A ordem dos jogos pode ser encontrada na Tabela 2.

Tipo de jogo	Nº de truques	Parceiro	*Trump*	Recompensa
Pas (fold)	-	-	-	-
Rik	8	Sim	Sim	1
Rik better	8	Sim	Sim (copas)	1
Solo 8	8	Não	Sim	1
Troela	8	Sim	Sim	2
Piek	1	Não	Não	3
Solo 9	9	Não	Sim	4
Misere	0	Não	Não	5
Solo 10	10	Não	Sim	6
Open Piek	1	Não	Não	6
Solo 11	11	Não	Sim	8
Open Piek+	1	Não	Não	10
Open Misere	0	Não	Não	10
Solo 12	12	Não	Sim	12
Open Misere+	0	Não	Não	15
Solo	13	Não	Sim	22

Quadro 2 - Tipos de jogos

Na Tabela 2 também é possível verificar o número de truques necessários para obter as recompensas dos oponentes. Por exemplo, quando estiver jogando Solo 8, o jogador precisa de 8 truques para ganhar, sem um parceiro. Se isto acontecer, o participante vitorioso ganhará 1 moeda de cada oponente perdedor.

O jogador à esquerda do *dealer* efetua o primeiro lance. Cada jogador pode passar (Pas) ou escolher o tipo de jogo. Após realizar um lance, os jogadores subseqüentes podem

passar ou realizar um lance maior que o selecionado pelo até o momento. Não é permitido para um jogador que já passou, dar o lance novamente.

A licitação continua em torno da mesa, quantas vezes forem necessárias para todos os jogadores que ainda não passaram. A licitação é terminada quando três jogadores passaram. Após isso, o jogador que realizou o lance mais alto é denominado declarante.

Após o leilão, dependendo do resultado do tipo de jogo, o declarante deve decidir quem será o seu parceiro.

2.3.3 Jogadores

Mesmo jogado entre quatro jogadores, estes podem formar parcerias, um contra três, dois contra dois ou cada um por si em última instância.

O jogador que colocou o lance mais alto durante o leilão é o vencedor da licitação da partida. Este jogador fica sendo chamado como o "declarante".

Para alguns tipos de jogos, o declarante deve obrigatoriamente escolher um parceiro. Esta escolha é feita através de um naipe selecionado pelo declarante. O jogador que possuir o Ás do naipe escolhido será o parceiro do declarante.

A posição dos jogadores numa parceria pode desempenhar um papel no resultado do jogo. Quando, por exemplo, uma dupla é composta pelo primeiro e pelo último jogador, esta poderia ser uma vantagem. Após o primeiro jogador jogar a carta, o último jogador será o seu *back-up*, pois pode analisar todas as cartas jogadas pelos demais adversários.

2.3.3.1 Jogador aleatório

Segundo Vignatti (2007), algoritmos aleatórios é uma das áreas que tem recebido grande atenção dos pesquisadores de otimização e teoria da computação nos últimos anos. Isto se deve às novas técnicas que tem surgido de caráter mais genérico no desenvolvimento de algoritmos para problemas em grafos. Estas técnicas têm sido usadas recentemente para problemas de projeto de sistemas computacionais, onde bons resultados têm sido desenvolvidos, tanto na teoria quanto na prática.

O objetivo da criação deste tipo de participante é a análise perante outros tipos de técnicas e utilizar este método para a simulação de jogadas no método de Monte-Carlo (seção

2.2).

2.3.3.2 Jogador baseado em regras

O jogador baseado em regras é considerado um sistema especialista, pois possui um domínio específico limitado(VORSTEVELD, 2007, p.9).

Um sistema baseado em regras é um tipo de modelo que utiliza regras explícitas para expressar o conhecimento do domínio de um problema e permite, através da confrontação do conhecimento existente com fatos conhecidos sobre um determinado problema, inferir regras relativas a esses fatos (SIMON, 2007, p.1).

A principal diferença entre um sistema especialista e um programa tradicional está na maneira como o conhecimento sobre o domínio do problema é codificado. Em aplicações tradicionais, o conhecimento sobre o domínio do problema é codificado tanto nas instruções propriamente ditas quanto nas estruturas de dados.

Já na abordagem de sistema especialista, todo conhecimento relativo ao domínio do problema é codificado exclusivamente nas estruturas de dados. Nenhum conhecimento é armazenado nas instruções ou nos programas propriamente ditos. Vários benefícios surgem imediatamente dessa estratégia.

Segundo Simon (2007), o sistema baseado em regras trata-se de um sistema de apoio à decisão que procura representar o modo de raciocínio e o conhecimento utilizado por especialistas na resolução de problemas no seu âmbito de especialidade.

Um exemplo de sistema baseado em regras é o Cyc[5]. A base de conhecimento Cyc é a representação de uma formalização do conhecimento humano fundamental: fatos, regras e raciocínios heurísticos sobre os objetos e acontecimentos da vida cotidiana.

2.4 TRABALHOS CORRELATOS

São descritos a seguir duas aplicações onde se utiliza a técnica de Monte-Carlo entre os quais são: Estimativa para Pi (π) (TRUNFIO;MCGRATH, 2000); análise de risco

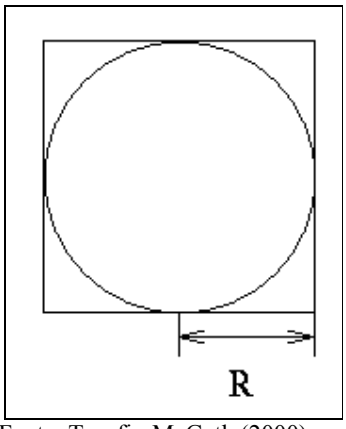
[5] Software distribuído pela empresa Cycorp, fornecedora de tecnologias semânticas de inteligência e senso comum para aplicações de software.

utilizando a simulação de Monte-Carlo (MOURA, 2004).

2.4.1 Estimativa para Pi (π)

Trunfio e McGath (2000) utilizam a técnica de Monte-Carlo no projeto para obter o valor de π de uma circunferência.

O experimento consiste simplesmente em atirar dardos aleatoriamente sobre um círculo inscrito em um quadrado, conforme mostrado na Figura 3. Dessa forma é realizada uma relação entre a geometria da figura e os dados estatísticos de jogar os dardos.



Fonte: Trunfio;McGath (2000).

Figura 3 - Desenho geométrico para descobrir π

Primeiramente é realizada a análise da região geométrica sobre o qual é realizada a experiência. Toma-se o raio da circunferência como R e a área do círculo como π x R². Desta forma, obtém-se a área do quadrado através da fórmula 4R². Assim, divide-se a área do círculo pela área do quadrado, chegando-se a fórmula π / 4.

No experimento, são simulados dardos sendo atirados no alvo. Todos os dardos caem dentro do quadrado, mas nem todos caem dentro do círculo. Este é o ponto principal da experiência. Atirando-se dardos completamente ao acaso, esta simulação estima a proporção da área do círculo para a área do quadrado pela contagem do número de dardos.

Na análise realizada sobre a figura geométrica, obtém-se a relação de π / 4. Desta forma pode-se estimar o valor de π através da fórmula conforme mostra o Quadro 1.

π = 4 x (N° de dardos no círculo) / (N° de dardos no quadrado)

Quadro 3 - Fórmula para obter o valor de π

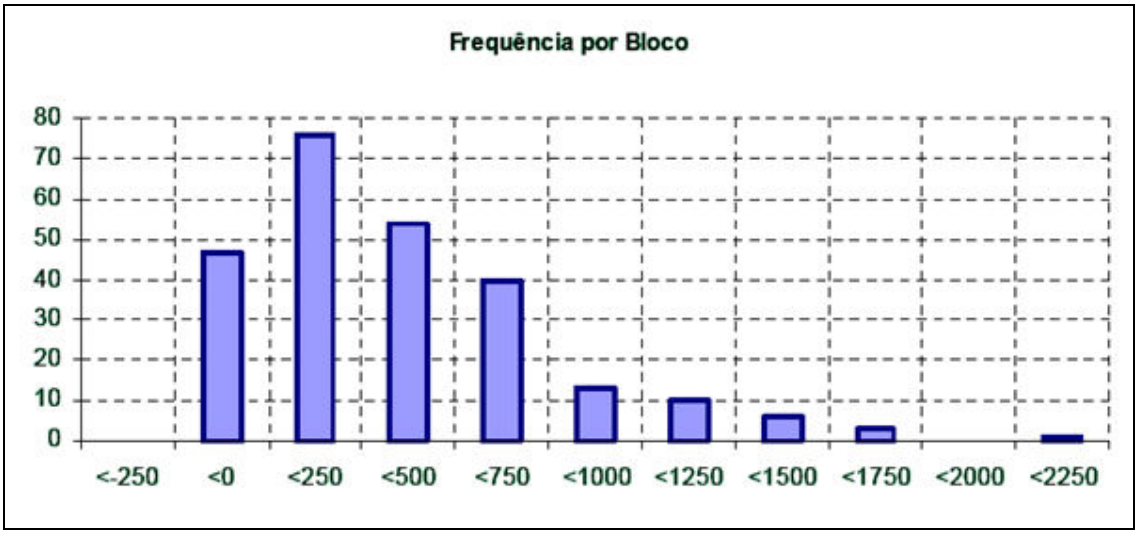
2.4.2 Análise de risco utilizando a simulação de Monte-Carlo

Segundo Moura (2004), a simulação de Monte-Carlo randomicamente gera inúmeros valores para variáveis consideradas incertas, simulando assim combinações de valores dessas variáveis que levam a resultados que são o foco da análise. No caso de análise de rentabilidade de projetos de investimento, cada simulação é como se o projeto tivesse sido executado e atingido uma determinada rentabilidade. As várias simulações então geram uma série de valores de possíveis rentabilidades que permitem ao tomador de decisão analisar o risco daquele projeto sob as condições que ele foi elaborado e modelado na planilha.

Para a elaboração do experimento de análise foram seguidas as seguintes etapas:

a) desenvolvimento do modelo;
b) identificar a incerteza e/ou risco;
c) identificar a(s) variável(eis) de análise (variáveis de saída);
d) gerar simulação;
e) análise do modelo simulado;
f) tomada de decisão.

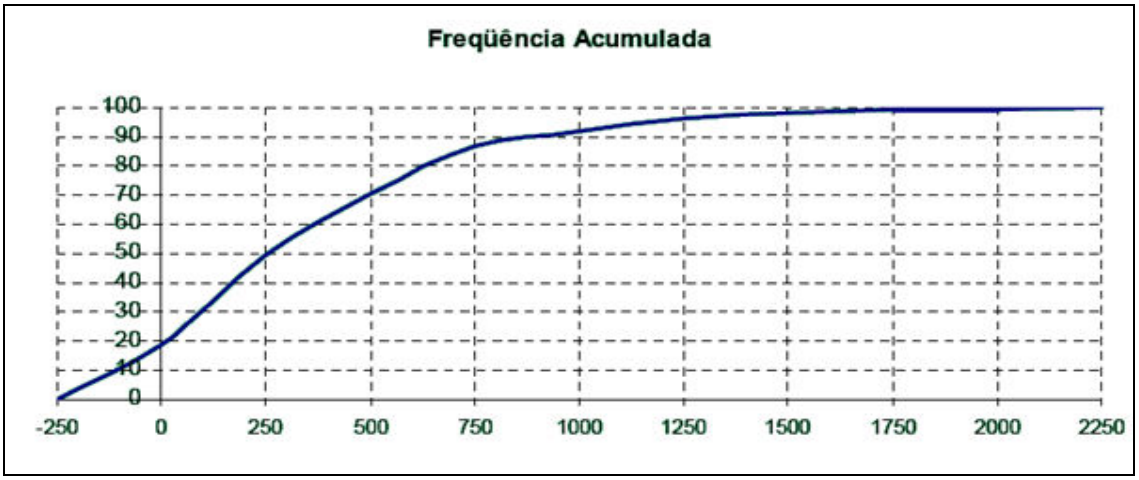
Considerando-se uma simulação realizada 250 vezes revelou o comportamento para a Margem Bruta (MB) de uma atividade (variável de saída) conforme ilustrado nas Figuras 4 e 5.



Fonte: Moura (2004).

Figura 4 - Gráfico de freqüência por bloco

O valor mais provável dentro dessas simulações é uma MB entre $0 e $250 (utilizando o gráfico de freqüência por bloco.



Fonte: Moura (2004).

Figura 5 - Gráfico de freqüência acumulada

Considerando-se um valor aceitável para a MB é $500, existe uma probabilidade de 70% de o valor da MB esteja abaixo de $500 (usando o gráfico de freqüência acumulada).

3 DESENVOLVIMENTO DO TRABALHO

Neste capítulo são descritos os requisitos, a especificação, a implementação e a operacionalidade da aplicação desenvolvida. Ao final deste capítulo, os resultados obtidos são comparados a conclusão do experimento de Vorsteveld (2007).

3.1 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO

O jogo deverá:

a) apresentar o cenário de jogo com os jogadores, disponibilizando os tipos de jogos, juntamente com o método de análise que será utilizado na partida (RF);
b) efetuar a análise da situação de jogo e calcular a melhor jogada (RF);
c) informar o resultado da análise e traduzir como uma jogada a ser efetuada pelo NPC (RF);
d) gerar um comparativo dos analisadores selecionados juntamente com os tipos de jogos escolhidos e montar uma tabela demonstrativa do resultado (RF);
e) ser implementado utilizando a linguagem de programação Java (Requisito Não-Funcional - RNF);
f) disponibilizar a técnica de Monte Carlo para o cálculo de análise das jogadas (RNF);
g) apresentar a técnica baseada em regras para cálculo de análise das jogadas (RNF).

3.2 ESPECIFICAÇÃO

Os tópicos subseqüentes descrevem a especificação do protótipo. A especificação do sistema foi realizada através da ferramenta Enterprise Architect 6.5 (SPARX SYSTEM, 2005).

3.2.1 Casos de uso

A aplicação possui 4 casos de uso, que correspondem às principais funcionalidades da aplicação:

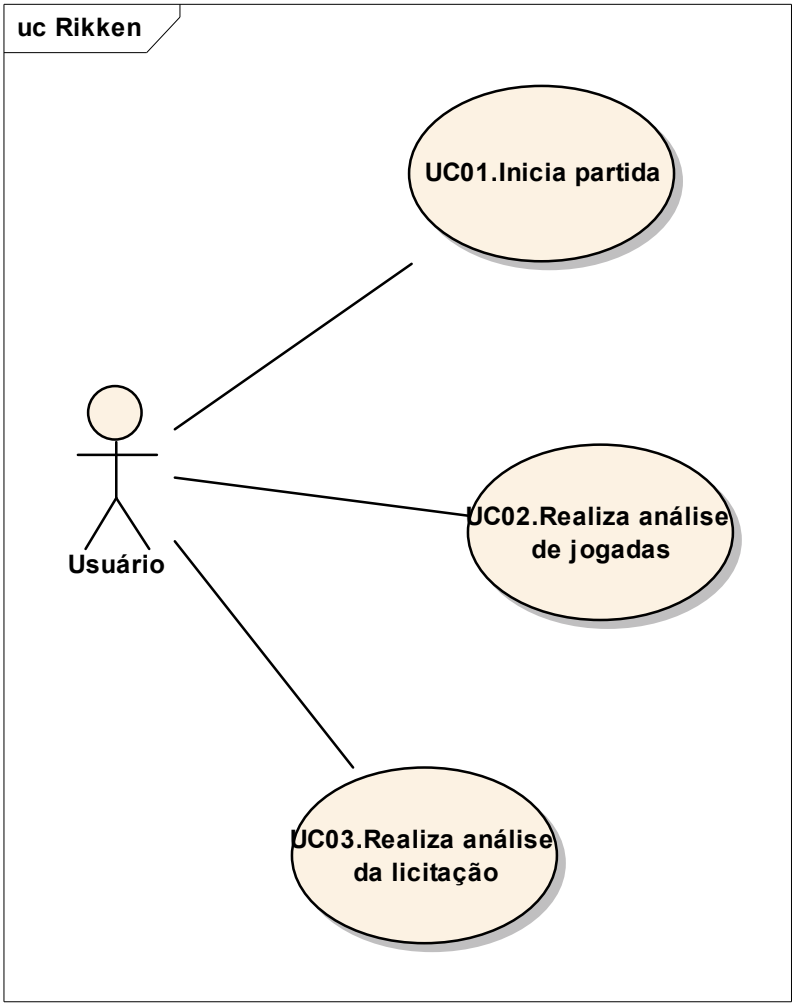


Figura 6 - Diagrama de casos de uso

A seguir são descritas as principais características dos casos de uso:

UC01: Inicia partida	
Resumo	Iniciar a partida de demonstração do jogo Rikken.
Sequência de ações	1- Usuário solicita início da partida. 2- O sistema apresenta tela de configuração do jogo. 3- Para cada jogador, o usuário informa a técnica e o nome do participante. 4- Usuário informa a quantidade de simulações que a técnica de Monte-Carlo irá realizar e quem irá iniciar a partida. 5- Usuário aciona o botão iniciar partida.

Quadro 4 - Detalhamento do caso de uso UC01

UC02: Realiza análise de jogadas	
Resumo	Realiza a análise das técnicas apresentadas quanto à jogadas.
Sequência de ações	1- Usuário solicita análise de jogadas. 2- Sistema apresenta tela para configurar a análise. 3- Usuário define a técnica a ser analisada, a técnica que será utilizada pelos demais jogadores e o tipo de jogo analisado. 4- Usuário escolhe a opção Iniciar análise. 5- Sistema apresenta o total de partidas vencidas por cada jogador e o percentual de sucesso das jogadas.

Quadro 5 - Detalhamento do caso de uso UC02

UC03: Realiza análise da licitação	
Resumo	Realiza a análise das técnicas apresentadas quanto à licitação.
Sequência de ações	1- Usuário solicita análise da licitação. 2- Sistema apresenta tela para configurar a análise. 3- Usuário informa a quantidade de vezes que será repetida a rotina de licitação de jogadas, a técnica a ser analisado, o tipo de jogo que será analisado, o limite inferior e o limite superior. 4- Sistema apresenta o valor que está sendo analisado e a quantidade de truques realizados.

Quadro 6 - Detalhamento do caso de uso UC03

3.2.2 Diagrama de atividades

A figura 7 apresenta o diagrama de atividades onde é demonstrado o fluxo das jogadas na visão do jogador.

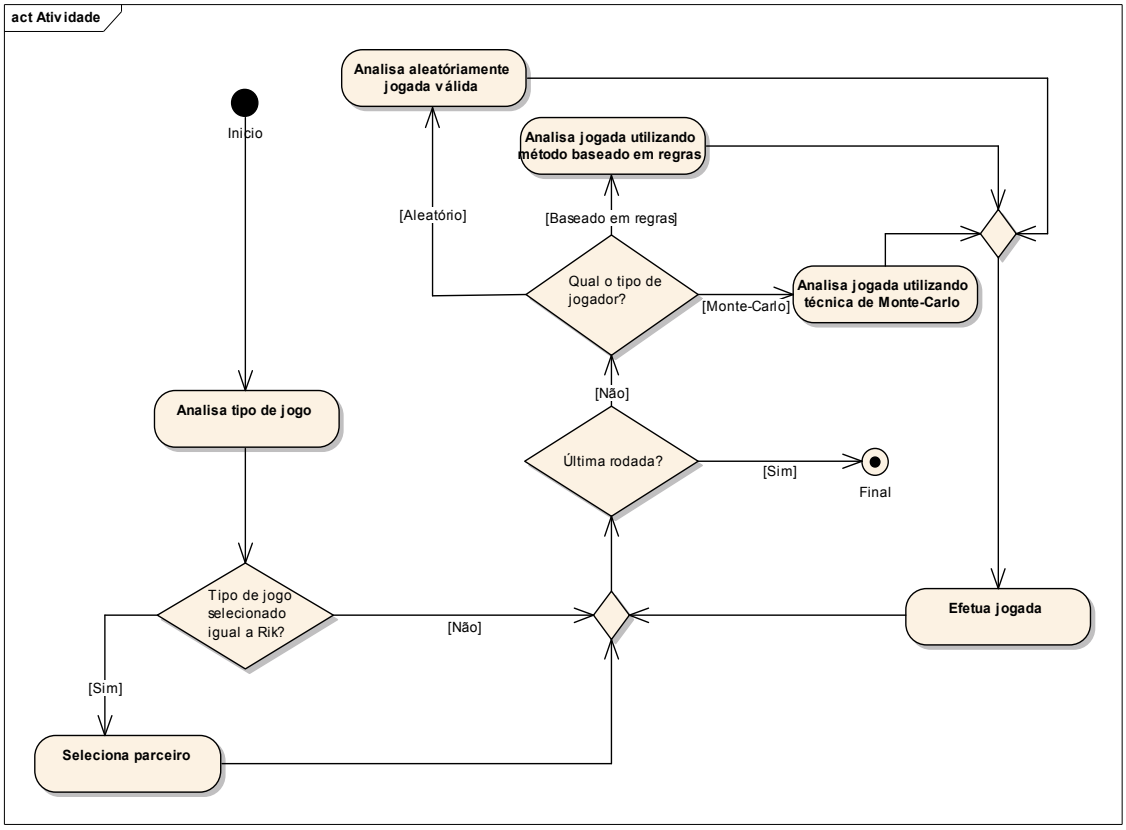


Figura 7 - Diagrama de atividades

Conforme apresentado na figura 7 ao iniciar uma partida, o jogador deve realizar uma análise para escolher o melhor tipo de jogo com as cartas que possui na mão. Dependendo do tipo de jogo escolhido é necessário escolher um parceiro. Após isso é realizada outra análise para escolher a melhor jogada a ser efetuada. Este processo é repetido até encerrar as cartas da mão do jogador.

3.2.3 Diagrama de classes

As classes implementadas no trabalho são representadas na figura 8. O pacote `View` foi abstraído do diagrama.

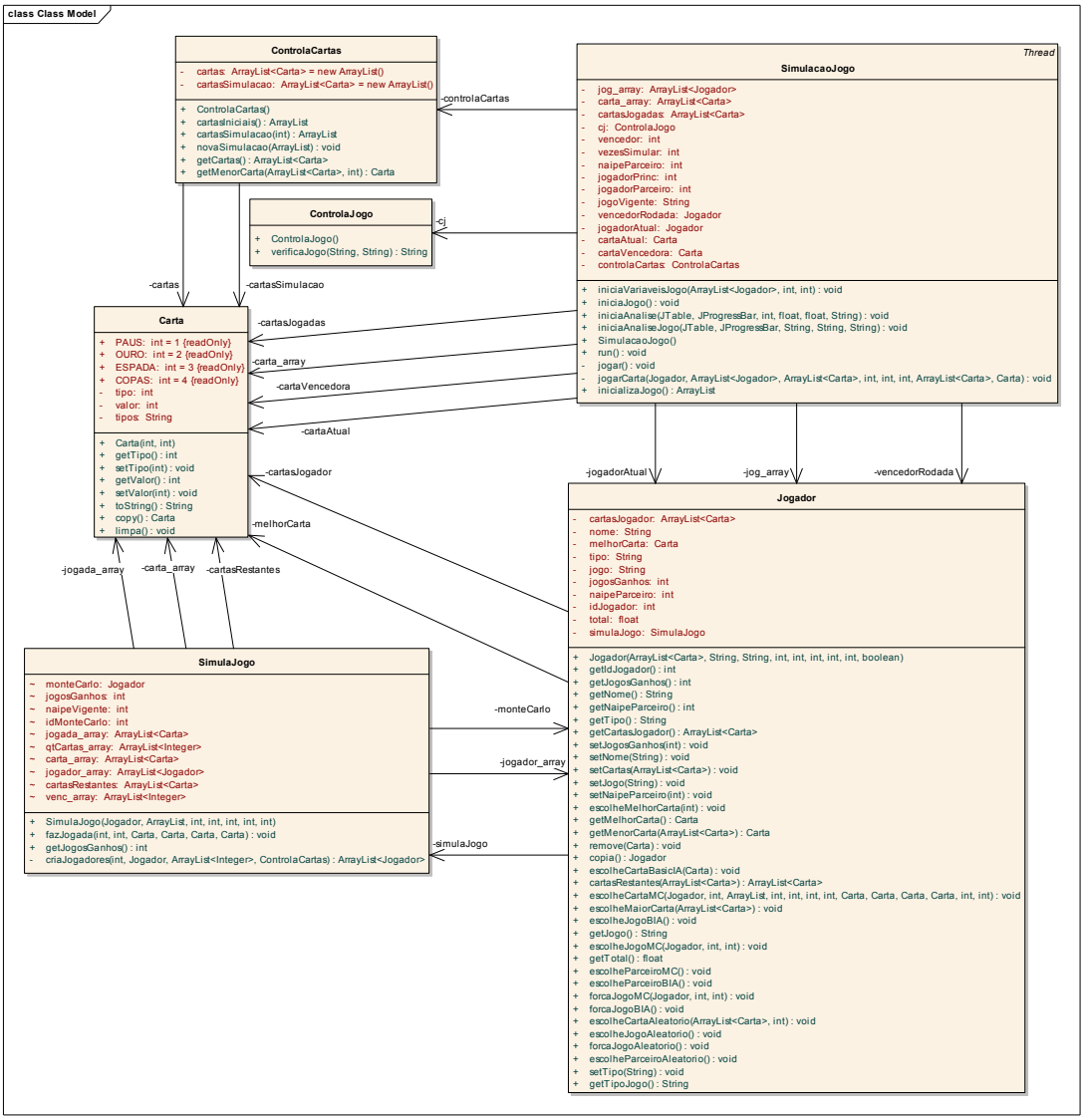


Figura 8 - Diagrama de classes

As classes representadas na Figura 8 possuem as seguintes funcionalidades:

a) `SimulacaoJogo`: responsável por controlar o fluxo das jogadas. Ela também auxilia o jogador a escolher o tipo de jogo de cada participante;

b) `ControlaJogo`: responsável pelo auxílio de decisões dos jogadores;

c) `Carta`: onde estão definidos os atributos de cada carta do jogo;

d) `ControlaCartas`: responsável pelo controle das cartas do jogo;

e) `Jogador`: simula a ação de cada jogador;

f) `SimulaJogo`: executa a simulação das partidas para o método de Monte-Carlo.

Na seção 3.6 são mostradas mais detalhadamente as classes mais importantes.

3.3 JOGADORES

Cada jogador possui uma técnica diferente para escolher o tipo de jogo, decidir o melhor parceiro e selecionar a melhor carta a ser jogada. Nesta seção são descritas as técnicas utilizadas na criação dos NPC's para os requisitos citados.

3.3.1 Jogador aleatório

A Figura 9 mostra o diagrama de atividades do fluxo executado pelo jogador aleatório.

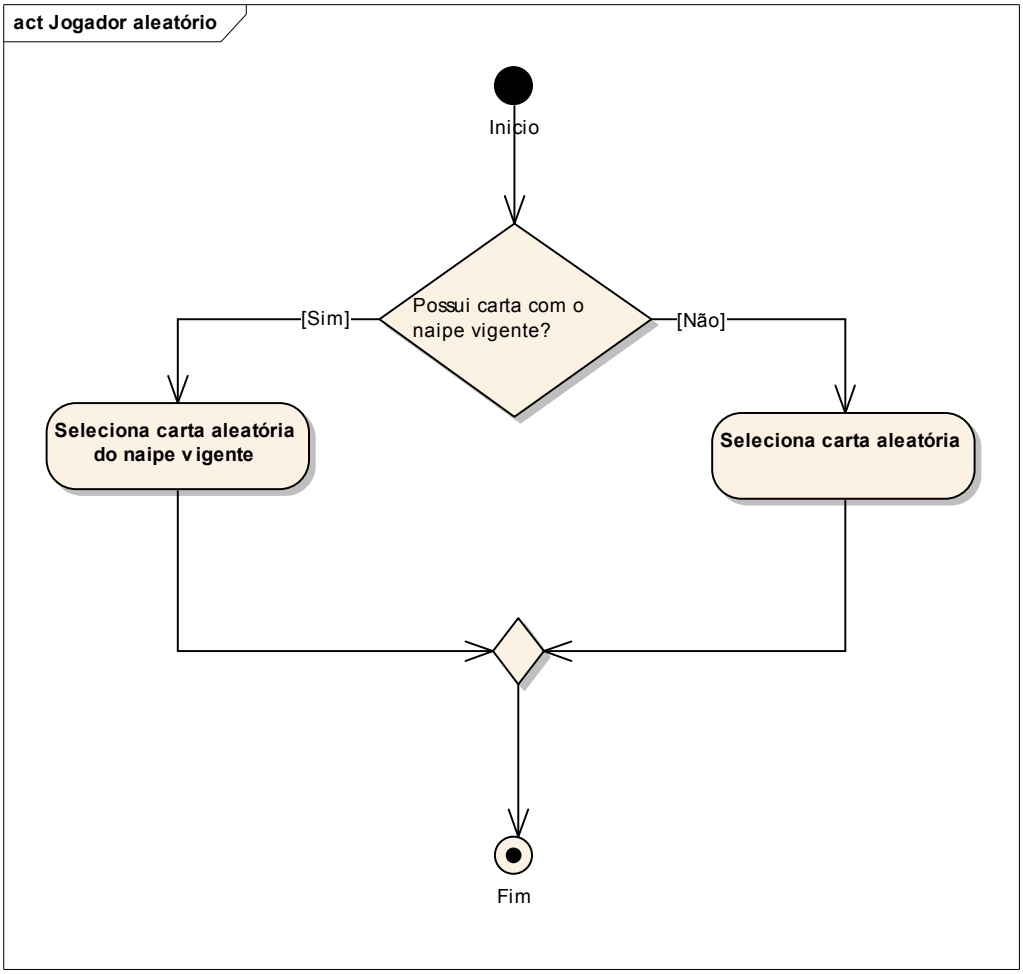


Figura 9 - Diagrama de atividades do jogador aleatório

O jogador aleatório utiliza a estratégia mais simples de todos os jogadores. Este tipo de técnica é capaz de obedecer todas as regras do jogo, mas apenas de forma aleatória.

3.3.1.1 Licitação do jogador aleatório

Na rodada de licitação, um leilão é realizado para decidir qual o tipo de jogo será escolhido. Este jogador escolhe aleatoriamente um tipo de jogo entre todos os possíveis. Se este tipo de jogo não é mais elevado que o jogo atual, o jogador passa.

Se vencer a rodada de licitação, é feita a escolha do parceiro e do naipe *trump* de forma aleatória dentro das possibilidades permitidas.

3.3.1.2 Jogando com o jogador aleatório

A estratégia utilizada pelo jogador aleatório é a mesma para todos os tipos de jogos. Primeiramente analisam-se todas as cartas que estão em posse do jogador para jogar de acordo com as regras.

Dentre todas as cartas possíveis, uma é escolhida e jogada aleatoriamente.

3.4 JOGADOR BASEADO EM REGRAS

Nesta seção é mostrada a estratégia utilizada pelo jogador baseado em regras na rodada de licitação e na análise da melhor carta a ser jogada. Na Figura 10 é mostrado o diagrama de atividades que ilustra o fluxo das informações do jogador para análise das jogadas.

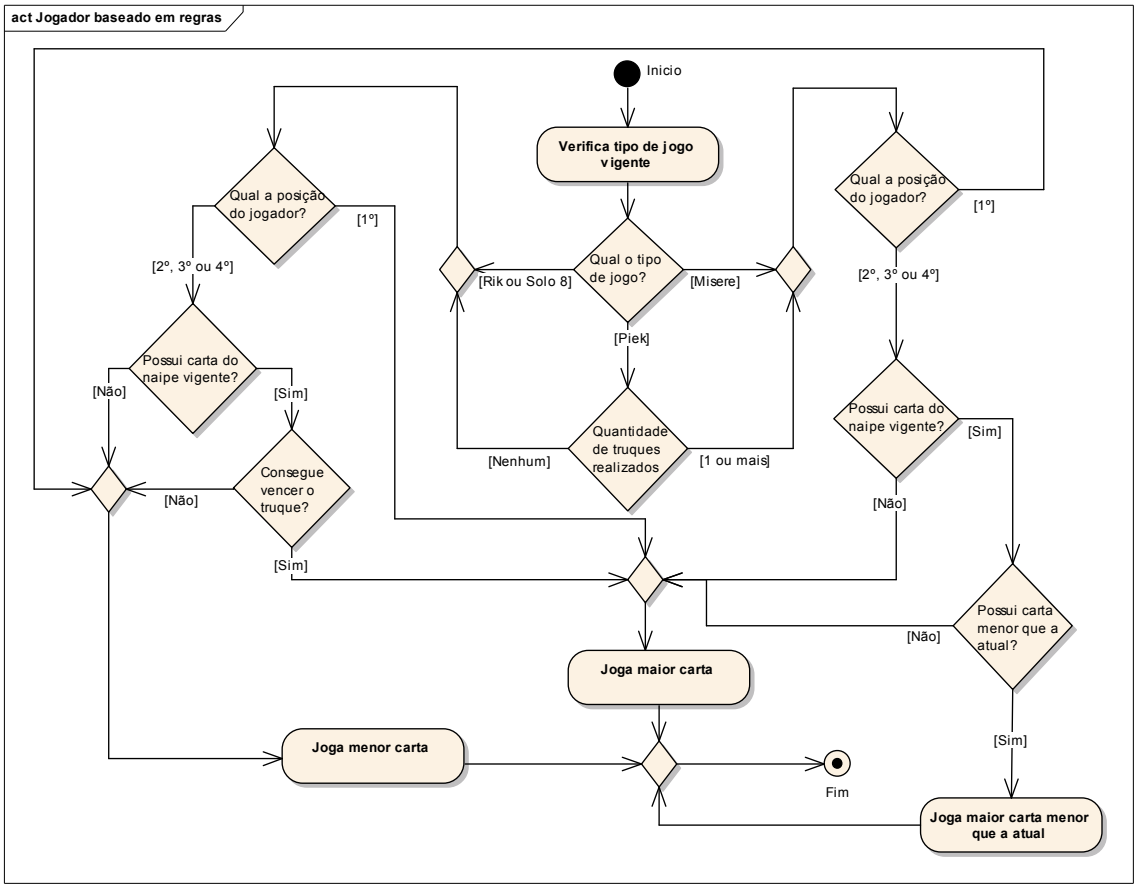


Figura 10 - Diagrama de atividades do jogador baseado em regras

3.4.1 Licitação do jogador baseado em regras

Para realizar uma boa oferta na rodada de licitação, o jogador efetua uma análise das cartas que estão sob sua posse. O método utilizado para tal análise é o *Trick Procent Count* (TPC), que determina a força das cartas no jogo.

A idéia por trás do método é definir a chance entre zero e um de cada carta da mão. Estas hipóteses representam a chance de ganhar um truque e quando somadas, dão uma previsão do número máximo de truques que as cartas do jogador podem realizar.

Quando possuir a maior carta de um naipe específico, o Ás, então o truque será ganho quando a carta for jogada. Caso o jogador possuir a segunda maior carta de um naipe, o rei, e não possuir o Ás do mesmo naipe, então existe uma possibilidade do rei não vencer o truque quando for jogado. Desta forma o Ás precisa ser eliminado antes para garantir que ao jogar o rei, o jogador ganhe o truque.

O mesmo acontece para as outras cartas do naipe. Ganhar um truque com uma carta

depende de 3 fatores especiais: o valor da carta (Ás, Rei, Damas...), o número total de cartas de um mesmo naipe em posse do jogador e o número de cartas que não esteja na mão do jogador.

A chance de ganhar um truque é dada pela fórmula conforme apresentado no Quadro 5.

$$\frac{\text{Total de cartas do mesmo naipe} - \text{Total de cartas maiores sob o não comando do jogador}}{\text{Total de cartas do mesmo naipe}}$$

Quadro 7 - Fórmula de cálculo TPC

As chances de cada carta são entre um e zero e, todos os cálculos negativos de chance são considerados como zero. Um exemplo do cálculo de uma mão de cartas é apresentado no Quadro 6.

Cartas	***%***
A (espadas)	(4 – 0) / 4 = 1.00
K (espadas)	(4 – 0) / 4 = 1.00
J (espadas)	(4 – 1) / 4 = 0.75
2 (espadas)	(4 – 9) / 4 = 0.00
Q (copas)	(1 – 2) / 1 = 0.00
K (ouros)	(3 – 1) / 3 = 0.66
J (ouros)	(3 – 2) / 3 = 0.33
4 (ouros)	(3 – 8) / 3 = 0.00
Q (paus)	(5 – 2) / 5 = 0.60
J (paus)	(5 – 2) / 5 = 0.60
10 (paus)	(5 – 2) / 5 = 0.60
8 (paus)	(5 – 3) / 5 = 0.40
6 (paus)	(5 – 4) / 5 = 0.20
Total # truques	6.14

Quadro 8 - Um exemplo do cálculo usando TPC

As condições básicas para o jogador efetuar um lance no leilão é baseada nas condições de vencer o tipo de jogo (seção 2.3.2). Por exemplo, a condição mínima para vencer o Solo 8 é a realização de 8 truques para ganhar.

Os resultados dos experimentos para determinar a precisão dos lances estão descritos na seção 3.7.1.

O procedimento para decidir o parceiro é baseado no sistema de pontos. A força de um naipe é obtida pela soma dos valores das cartas de naipes iguais.

O valor de cada carta é definido da seguinte forma:

a) 2 a 10 = 1 ponto;

b) J = 2 pontos;

c) Q = 3 pontos;

d) K = 4 pontos;

e) A = 5 pontos.

O naipe que possuir o número máximo de pontos será escolhido para ser o parceiro do jogador declarante.

Um exemplo do cálculo apresentado no Quadro 7.

Cartas	*Pontos*	*Cartas do naipe*	
A (espadas)	5	1	
K (espadas)	4	1	
J (espadas)	2	1	
2 (espadas)	1	1	
Total	**12 +**	**4 =**	***16***

Quadro 9 - Um exemplo do cálculo necessário para determinar o parceiro

3.4.2 Jogando com o jogador baseado em regras

A estratégia utilizada pelo jogador é definida por algumas regras. Estas regras dependem da posição atual na rodada e do tipo de jogo.

Para os tipos de jogos Rik e Solo 8, são utilizadas as estratégias conforme mostra o Quadro 8.

1- Se o jogador é o primeiro a jogar:

Se jogador possui uma ou mais das maiores cartas de um naipe **então**
Jogue aleatoriamente uma destas maiores cartas
Senão
Jogue uma carta aleatoriamente
Fim se

2- Se o jogador é o segundo ou terceiro a jogar:

Se possuir carta do naipe vigente **então**
Se possuir a maior carta do naipe vigente **então**
Jogue a maior carta do naipe vigente
Senão
Se possuir uma carta maior que a jogada anteriormente do naipe vigente **então**
Jogue aleatoriamente uma carta maior que a atual do naipe vigente
Senão
Jogue a menor carta do naipe vigente
Fim se
Fim se
Senão
Jogue a menor carta de outro naipe escolhido
Fim se

3- Se o jogador for o último a jogar:

Se possuir carta do naipe vigente **então**
Se conseguir vencer o truque com uma carta do naipe vigente **então**
Jogue a menor carta possível para vencer o truque
Senão
Jogue a menor carta do naipe vigente
Fim se
Senão
Jogue a menor carta de outro naipe escolhido
Fim se

Quadro 10 - Estratégias baseadas em regras usadas nos tipos de jogos Rik e Solo 8

As estratégias utilizadas nos jogos Piek e Misere são descritas nos Quadros 9 e 10 respectivamente.

1- Se o jogador é o primeiro a jogar

Se o jogador venceu zero truques **então**
Jogue a maior carta sob a sua posse
Senão
Jogue conforme a estratégia do jogo Misere
Fim se

2- Se o jogador é o primeiro a jogar

Se jogador venceu zero truques **então**
Jogue a maior carta dentro das permitidas nas regras do jogo
Senão
Jogue conforme a estratégia do jogo Misere
Fim se

Quadro 11 - Estratégia baseada em regras usadas no tipo de jogo Piek

1- Se o jogador é o primeiro a jogar

Jogue a menor carta aleatoriamente

2- Se o jogador não for o primeiro a jogar

```
Se o jogador é o declarante então
    Se o jogador possui cartas do naipe vigente então
        Se o jogador puder jogar a maior carta menor que a atual então
            Jogue a maior carta menor que a atual
        Senão
            Jogue a menor carta do naipe vigente
        Fim se
    Senão
        Jogue a maior carta de outro naipe
    Fim se
Senão
    Se o jogador possui cartas do naipe vigente então
        Se o declarante já jogou na rodada então
            Se o declarante está vencendo o truque no momento então
                Se o jogador puder jogar a maior carta menor que a atual então
                    Jogue a maior carta menor que a atual
                Senão
                    Jogue a menor carta do naipe vigente
                Fim se
            Senão
                Jogue a maior carta do naipe vigente
            Fim se
        Senão
            Se o jogador puder jogar a maior carta menor que a atual então
                Jogue a maior carta menor que a atual
            Senão
               Jogue a menor carta do naipe vigente
            Fim se
        Fim se
    Senão
        Selecione a maior carta de cada naipe e jogue aleatoriamente uma destas cartas
    Fim se
Fim se
```

Quadro 12 - Estratégia baseada em regras usadas no tipo de jogo Misere

3.5 JOGADOR MONTE-CARLO

O jogador Monte-Carlo (MC) utiliza a técnica de simulações de partidas para obter a melhor jogada a ser efetuada.

A implementação de uma simulação começa com o verdadeiro jogo sendo colocado em espera, ou seja, o jogo atual fica em espera enquanto são criadas partidas a parte.

Nessas partidas a parte, o jogador MC cria aleatoriamente quatro jogadores que vão representar os participantes verdadeiros da partida em espera. Todos os jogadores estão dispostos da mesma maneira, ou seja, se no jogo em espera o jogador 1 estiver na posição 1,

na partida criada ele também ficará na posição 1. O jogador MC recebe as mesmas cartas do que ele possui no jogo em espera. Os demais jogadores recebem a mesma quantidade de cartas que o participante ao qual estão representados no jogo em espera, sendo estas distribuídas aleatoriamente.

Desta forma é inicializada a simulação, sendo finalizada com os jogadores efetuando todas as jogadas.

3.5.1 Licitação do jogador Monte-Carlo

Para selecionar um tipo de jogo na rodada de licitação, o jogador utiliza a técnica de simular o jogo *n* vezes com as cartas que possui.

Dentre as simulações, é gravado o total de jogos ganhos. Deste valor é feito uma média pela quantidade de simulações realizadas. O resultado é obtido pela fórmula ilustrada no Quadro 11.

$$\frac{\text{Simulação do jogo x } N \text{ (Truques ganhos)}}{N}$$

Quadro 13 - Fórmula de cálculo para escolha do tipo de jogo

As condições mínimas são definidas pelas características de cada jogo, ou seja, para escolher o tipo de jogo Rik ou Solo 8, a condição mínima é vencer 8 truques. Para o jogo Piek, o resultado da fórmula deve ser 1 e para o jogo Misere deve ser 0.

Para escolher o parceiro, o jogador MC utiliza a técnica *Lose Trick Count* (LTC). Este método consiste na verificação de truques que possivelmente não será vencido pelo jogador, por exemplo, o jogador não irá vencer 10 dos 13 truques possíveis.

Para a análise, é realizada a contagem da quantidade de cartas de cada um dos quatro naipes existentes e gravados em uma variável *x*. Após isso, são verificadas as *x* maiores cartas dos naipes. A quantidade de cartas que não estiverem neste intervalo é gravada na variável *y*. O naipe que possuir o maior número em *y* é escolhido para ser o parceiro do jogador.

Por exemplo:

a) Nenhuma carta do naipe: 0 truques perdidos
b) 1 carta do naipe: A = 0 truques perdidos; x = 1 truque perdido
c) 2 cartas do naipe: AK = 0 truques perdidos; Ax, Kx = 1 truque perdido; xx = 2 truques perdidos
d) 3 cartas do naipe: AKQ = 0 truques perdidos; AKx, AQx, KQx = 1 truque

perdido; Axx, Kxx, Qxx = 2 truques perdidos; xxx = 3 truques perdidos.

O número máximo de derrotas possíveis é tomado como doze, três para cada naipe. Desta forma, o número de truques perdidos é subtraído de doze. A previsão do Quadro 12 será 12 – 6 = 6.

Cartas	**Perdidas**
(Espadas) A, K, J, 2	1
(Copas) Q, 10, 9	2
(Ouros) 10, 9, 4	3
(Paus) A, K, Q	0
Total	6

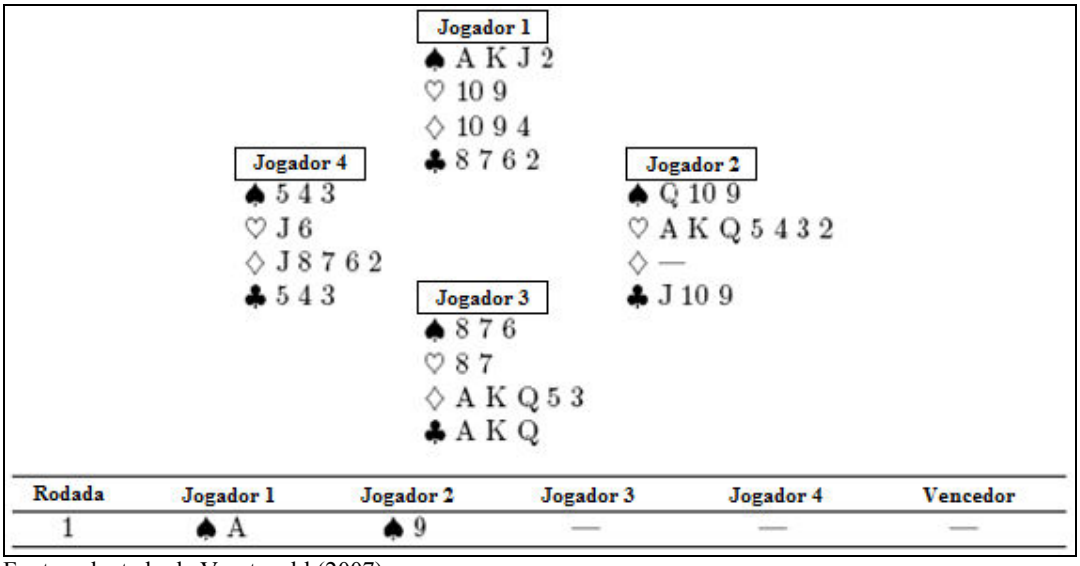
Quadro 14 - Exemplo de *Losing Trick Count*

3.5.2 Jogando com o jogador Monte-Carlo

Após finalizar a rodada de licitação, o jogo é iniciado. Quando é a vez do jogador MC efetuar a jogada, o jogo é atribuído ao estado de espera para a realização das simulações.

Na finalidade de escolher a melhor carta, são executados dois procedimentos. No primeiro, o jogo é simulado *n* vezes com as cartas atuais do jogador MC. O segundo procedimento consiste em escolher a melhor carta tendo em vista todas as simulações realizadas.

O primeiro procedimento é explicado através das Figuras 11 e 12, ilustrados em Vorsteveld (2007). Situação 1: O jogador 1 começa o jogo onde o objetivo é realizar a maior quantidade de truques.

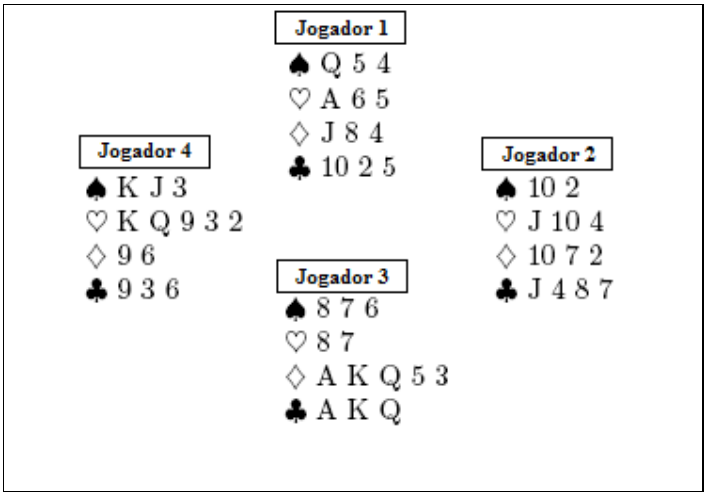


Fonte: adaptado de Vorsteveld (2007).

Figura 11 - Jogador MC em ação

Na Figura 11, o jogador MC é representado pelo jogador 3. O jogo neste momento é colocado em espera e simulações são geradas. O jogador MC mantém as cartas do jogo original e o restante das cartas é distribuído para os outros três participantes da simulação.

Para cada rotina de simulação, é gravada a primeira carta jogada e o número de truques realizados no jogo iniciando com a carta anteriormente gravada. Isto se repete para cada uma das cartas de posse do jogador MC.



Fonte: adaptação de Vorsteveld (2007).

Figura 12 - Simulação jogador Monte-Carlo

Neste caso apenas três cartas são possíveis de serem jogadas pelo jogador 3: 6, 7 e 8 de espadas. Esta simulação é repetida *n* vezes.

O próximo procedimento é determinar o valor de truques para cada uma das cartas possíveis de serem jogadas. Quando a simulação é encerrada, um vetor com as primeiras cartas jogadas e a quantidade de truques realizada para cada carta é guardado.

Para cada primeira carta, o número de truques correspondente de cada carta é somado e divido pela quantidade de simulações realizadas. Por exemplo, quando o 8 de espadas é selecionado em 5 dos 10 jogos, o total de truques somados é 25. Então divide-se o total de truques pela quantidade de vezes que a carta foi selecionada, ou seja, 25 / 5 = 5.

O próximo passo é escolher a melhor carta. Como o objetivo do jogo é maximizar a quantidade de truques, então a carta escolhida é a que possui a maior média.

As cartas com a menor média são escolhidas para os tipos de jogos Piek e Misere.

3.6 IMPLEMENTAÇÃO

Nesta seção são apresentadas de forma detalhada as técnicas e ferramentas utilizadas no desenvolvimento da aplicação, bem como questões relacionadas à sua operacionalidade.

3.6.1 Técnicas e ferramentas utilizadas

As técnicas empregadas no desenvolvimento deste trabalho buscaram sempre a facilidade de extensão e usabilidade em outras arquiteturas. Além disso, a utilização de alguns padrões de projeto e hierarquia das classes, através de heranças e interfaces, foram amplamente aplicados visando um código limpo de fácil manutenção e possível de reaproveitamento.

3.6.1.1 Contexto da aplicação

A aplicação e disposição dos jogadores e cartas é baseada no jogo desenvolvido na linguagem de programação Java por Aernsbergen (1999), conforme mostrado na Figura 13.

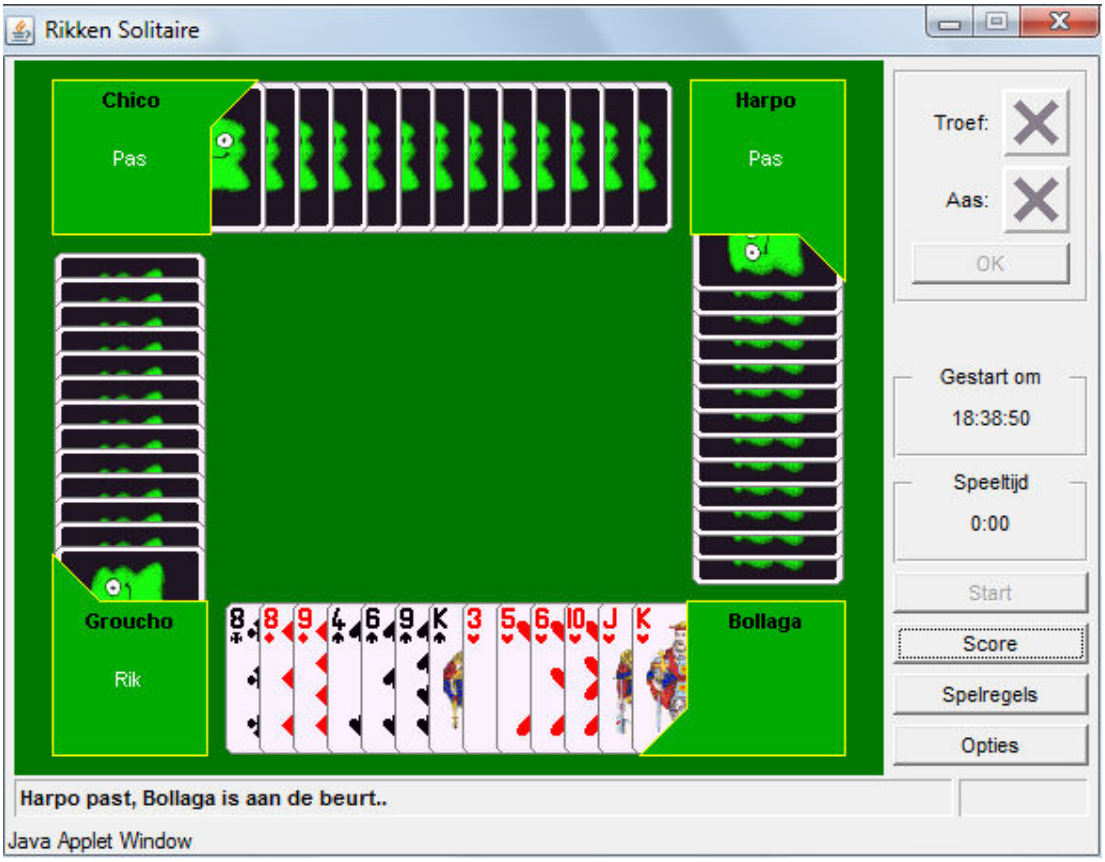


Figura 13 - Jogo de cartas Rikken desenvolvido por Aernsbergen

No presente trabalho serão analisados apenas 4 tipos destes jogos: Rik, Solo 8, Piek e Misere. Estes jogos citados são considerados os principais tipos de jogos, sendo os demais conhecidos como extensões destes.

Embora os outros tipos de jogos disponibilizem mais opções durante a licitação, eles exigem a mesma tática e estratégia dos tipos de jogos principais.

Não foram realizadas algumas implementações do jogo original como a presença do naipe *trump* da partida, sendo considerada de menor valia para a análise.

A interface gráfica foi desenvolvida de forma manual, sem o auxílio de geradores de interfaces ou bibliotecas especiais. Sendo assim, para a aplicação são criadas imagens que representam as cartas e componentes de manipulação como botões e caixas de texto.

3.6.1.2 Classe `ControlaCartas`

A distribuição das cartas iniciais é realizada de forma aleatória utilizando a função `Random`, presente na biblioteca `java.util.Random`. Esta rotina é encontrada na classe

`ControlaCartas` que atua tanto no jogo principal quanto na simulação de Monte-Carlo.

Além de realizar a distribuição das cartas, o método `cartasIniciais` retorna o *array* de cartas em ordem de naipe, conforme disposição apresentada no item 3.6.1.1. Este procedimento é apresentado no Quadro 13.

```
public ArrayList cartasIniciais() {
    ArrayList cartasSorteadas = new ArrayList();
    ArrayList<Carta> cartasAux = new ArrayList<Carta>();
    int achou = 0;
    Carta menorCarta = new Carta(0,0);

    Random r = new Random();
    // Realiza a distribuição das cartas
    for (int i = 0; i < 13; i++) {
        int num = r.nextInt(cartas.size());
        Carta cartaAux = (Carta) cartas.get(num);
        cartasAux.add(cartaAux);
        cartas.remove(cartaAux);
    }
    //Organiza as cartas de acordo com o naipe
    while (achou == 0) {
        achou = 1;
        for (int j = 0; j < cartasAux.size(); j++) {
            if (cartasAux.get(j).getTipo() == 1) {
                menorCarta = getMenorCarta(cartasAux, 1);
                cartasSorteadas.add(menorCarta);
                cartasAux.remove(menorCarta);
                achou = 0;
            }
        }
    }
```

Quadro 15 - Distribuição e ordenação das cartas

3.6.1.3 Classe `Jogador`

Os jogadores são representados pela classe `Jogador` e é considerada a mais importante da aplicação, pois é onde se localiza todos os métodos de escolha do tipo de jogo, de carta e de parceiro. A seguir serão mostrados como estão estruturados os jogadores.

Na rodada de licitação, o jogador utiliza a técnica TPC ou Monte-Carlo para a escolha do tipo de jogo. Do valor calculado é realizada uma análise e criada uma margem para cada tipo de jogo. Esta margem pode ser visualizada no Quadro 14.

```
if (total <= 0.5) {
    this.jogo = "M";
} else if (total < 1.5) {
    this.jogo = "P";
} else if (total >= 7.5) {
    this.jogo = "S";
} else if (total >= 4) {
    this.jogo = "R";
} else {
    this.jogo = "J";
}
```

Quadro 16 - Margem para selecionar o tipo de jogo

Para escolher o tipo de jogo Misere (M), o jogador deve estimar a realização de menos de 0,5 truques. Para o tipo de jogo Piek (P), o resultado da análise deve estar entre 0,6 e 1,4. Se este resultado for maior ou igual a 7,5, o tipo de jogo selecionado é o Solo 8 (S). A margem de análise dos truques estiver entre 4 e 7,4, o jogo escolhido é o Rik. Caso os valores não se encaixem nas margens citadas, o jogador escolhe a opção Jump (J), que significa que ele irá participar do jogo mas não irá escolher nenhum tipo de jogo.

Esta margem pode ser alterada caso todos os jogadores não selecionarem algum tipo de jogo. Neste caso, a rotina é repetida sem a opção Jump (J). Este tratamento encontra-se no método `forcaJogoMC` ou `forcaJogoBIA` conforme apresentado no Quadro 15, dependendo da técnica do jogador.

```
if (total <= 0.75) {
    this.jogo = "M";
} else if (total < 2) {
    this.jogo = "P";
} else if (total >= 7) {
    this.jogo = "S";
} else {
    this.jogo = "R";
}
```

Quadro 17 - Margem para obrigar escolha do tipo de jogo

Conforme citado no item 3.4.2, as estratégias utilizadas pelo jogador baseado em regras dependem do tipo de jogo escolhido. Essa diferença pode ser encontrada no Quadro 16, onde é mostrada a técnica utilizada na escolha da melhor carta pelo jogador.

```
public void escolheCartaBasicIA(ArrayList<Carta> maoJogador, Carta carta, String jogo) {
    int achou = 0; // 1 = melhor carta -- 2 = pior carta
    int naipe = carta.getTipo();
    int valorCarta = carta.getValor();
    int valorMelhor = 0, valorPior = 14, melhorPior = 0, menorCarta = 14;

    for (int i = 0; i < maoJogador.size(); i++) {
        if (maoJogador.get(i).getTipo() == naipe) { //Se naipe da carta atual do jogo for igual
            if (jogo.equals("R") ||
                    jogo.equals("S") ||
                    jogo.equals("J") ||// Estratégia dos jogos Rik, Solo 8 e Piek
                    (jogo.equals("P") && jogosGanhos == 0)) {
                // Se a carta da mão do jogador for maior que carta do jogo
                if (maoJogador.get(i).getValor() > valorCarta) {
                    if (maoJogador.get(i).getValor() > valorMelhor) {
                        valorMelhor = maoJogador.get(i).getValor();
                        this.melhorCarta = maoJogador.get(i);
                        achou = 1;
                    }
                } else if ((maoJogador.get(i).getValor() < valorCarta) &&
                        (achou != 1)) {
                    if (maoJogador.get(i).getValor() < valorPior) {
                        valorPior = maoJogador.get(i).getValor();
                        this.melhorCarta = maoJogador.get(i);
                        achou = 2;
                    }
                } // Estratégia dos jogos Misere e Piek
            } else if ((jogo.equals("M")) || (jogo.equals("P") && jogosGanhos > 0)) {
                if (maoJogador.get(i).getValor() < valorCarta) {
                    if (maoJogador.get(i).getValor() > melhorPior) {
                        melhorPior = maoJogador.get(i).getValor();
                        this.melhorCarta = maoJogador.get(i);
                        achou = 2;
                    }
                } else if (achou == 0) {
                    this.melhorCarta = getMelhorCarta();
                }
            }
        } else if (achou == 0) {
            if (maoJogador.get(i).getValor() < menorCarta) {
                menorCarta = maoJogador.get(i).getValor();
                this.melhorCarta = maoJogador.get(i);
            }
        }
    }
```

Quadro 18 - Técnica para selecionar a melhor carta pelo jogador baseado em regras

3.6.1.4 Simulação de Monte-Carlo

Para a reprodução do método de Monte-Carlo na aplicação, foi desenvolvida a classe `SimulaJogo`. Nela é feito uma cópia do jogo original onde são passadas as cartas que já foram jogadas, o jogador que utiliza a técnica, a quantidade de cartas que cada participante possui e o vencedor da última rodada.

Logo após é feita uma redistribuição das cartas entre os três jogadores fictícios criados conforme ilustrado no Quadro 17. Essas cartas representam as cartas que ainda estão vigentes

em jogo, ou seja, que não foram jogadas e nem estão sob posse do jogador Monte-Carlo. Este processo é realizado através do método `criaJogadores`.

```
private ArrayList<Jogador> criaJogadores(int idMonte,
        Jogador simular,
        ArrayList<Integer> cartas,
        ControlaCartas controle) {
    ArrayList<Jogador> jogs = new ArrayList<Jogador>();

    for (int i = 0; i < 4; i++) {
        Jogador temp;
        // Cria jogadores utilizados na simulação
        if (idMonte == (i + 1)) {
            temp = simular;
        } else {
            temp = new Jogador(null, "Jogador " + (i + 1), "IA", (i + 1));
            //Realiza a distribuição das cartas entre os jogadores restantes
            temp.setCartas(controle.cartasSimulacao(cartas.get(i)));
        }
        jogs.add(temp);
    }
    return jogs;
}
```

Quadro 19 - Criação e distribuição das cartas para jogadores na simulação de Monte-Carlo

O processo de criação dos jogadores fictícios e distribuição das cartas para os jogadores criados são repetidos a cada simulação realizada.

Após todo o cenário do jogo criado, simulações do jogo são realizadas exatamente como acontece no jogo principal através do método `fazJogada`. Após cada simulação é gravado o número de jogos ganhos e a primeira carta jogada da partida simulada.

Esta técnica é utilizada tanto na rodada de licitação quanto a cada jogada do participante. A diferença é o naipe vigente da rodada. Durante a partida um jogador pode ter poucas opções de cartas para jogar. Isto se dá ao fato de não possuir cartas do mesmo naipe que o jogador que efetuou a jogada anterior.

3.6.1.5 Padrão Model View Controller (MVC)

Uma das preocupações durante o desenvolvimento do protótipo era a de disponibilizar independência das classes do núcleo do sistema com a interface de manipulação. Seguindo esta abordagem seria possível portar facilmente a implementação para trabalhar num ambiente cliente servidor. A única dificuldade para realizar esta implementação seria o

desenvolvimento de uma nova interface que interagisse com o núcleo principal. Assim para a elaboração do protótipo foi adotado o padrão de projeto MVC para poder separar a interface do núcleo principal.

Segundo Stelting e Maassen (2002, p. 209) MVC é um padrão de arquitetura de aplicações que visa separar a lógica da aplicação, *Model*, da interface do usuário, *View*, e do fluxo da aplicação, *Controller*. Permite que a mesma lógica de negócios possa ser acessada e visualizada por várias interfaces.

Na camada *model* encontram-se as classes `Jogador` e `Cartas`, explicadas anteriormente. Já na *controller* é possível encontrar as classes que fazem a ligação das informações encontradas na *model* com a interface. Nela encontram-se as classes `ControlaCartas`, `ControlaJogo` e `SimulaJogo`. Já na camada *view* encontramos as classes visíveis para os usuários, ou seja, a *interface* da aplicação. Nela encontramos as classes `AnaliseJogo` e `AnaliseLicitação`, responsáveis pelas telas onde são realizadas as análises. Também encontra-se a classe `Mesa`, responsável por mostrar o comportamento do jogo Rikken e de seus jogadores.

3.6.2 Operacionalidade da implementação

Para demonstrar os resultados obtidos nas análises, foi criada a aplicação Rikken que, demonstra o comportamento de cada técnica citada no presente trabalho e também os resultados das análises realizadas.

Ao carregar o ambiente de simulação, a aplicação disponibiliza três opções para o usuário interagir com o sistema, o jogo e as análises, conforme ilustrado na Figura 14.

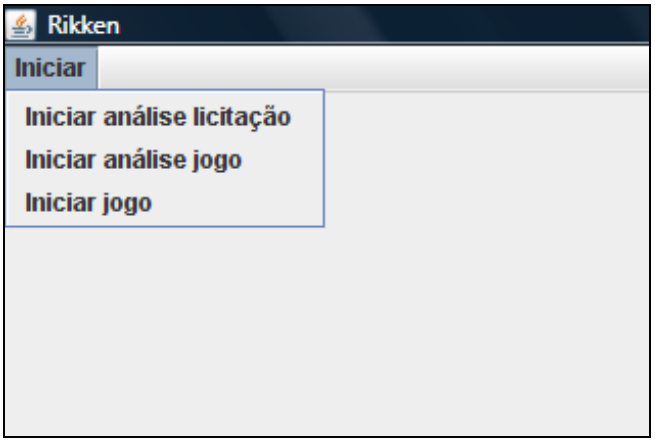


Figura 14 - Tela das opções da aplicação

Ao selecionar a opção [iniciar jogo], é disponibilizada uma tela com as configurações dos jogadores da simulação conforme mostrado na Figura 15.

Figura 15 - Tela para configuração dos jogadores

Nesta tela é possível dar nome a cada jogador, selecionar a técnica a ser utilizada, informar a quantidade de simulações que serão executadas pelo jogador Monte-Carlo na seleção de uma carta e definir quem irá iniciar a partida.

Após ajustar as configurações e selecionar o botão para iniciar a simulação, é iniciada a demonstração do jogo, conforme a Figura 16.

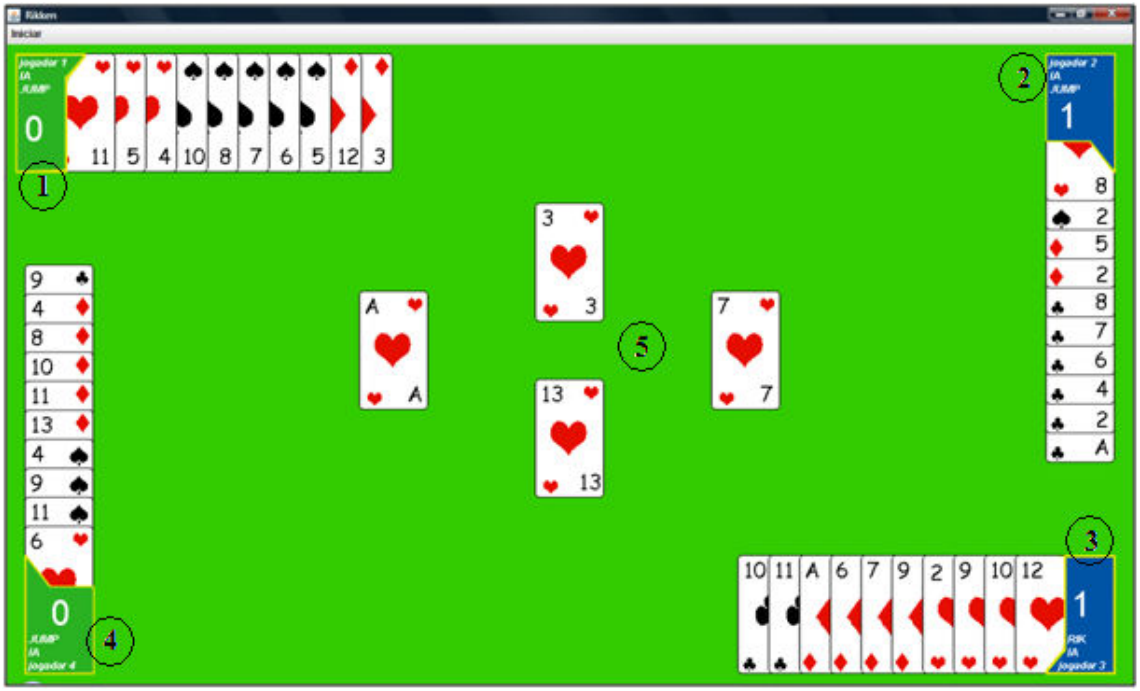


Figura 16 - Aplicação Rikken

Nesta tela são encontradas as cartas dos jogadores juntamente com seus identificadores, representados pelos números de 1 a 4. Nesses identificadores é onde se encontram informações de cada jogador como o nome, tipo de técnica utilizada, jogo que foi escolhido na rodada de licitação e pontuação. Esses itens podem mudar de cor durante o jogo. Isso ocorre quando uma dupla é formada no tipo de jogo Rik. No item 5 da figura, é apresentada uma rodada completa, que consiste no ato de todos os participantes jogarem uma carta.

Outras opções disponíveis no *menu* principal são as análises de jogo e de licitação. Ao selecionar a análise de jogo, é disponibilizada a opção para a configuração da análise conforme a Figura 17.

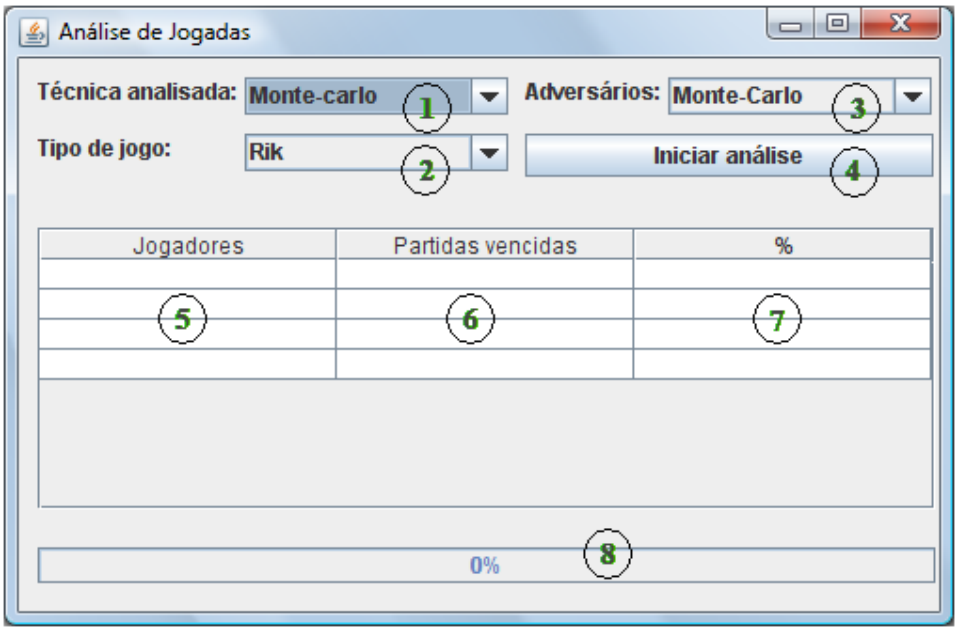


Figura 17 - Tela de análise de jogadas

Para a realização da análise, é necessário selecionar a técnica que será utilizada pelo jogador principal (item 1), representado pelo Jogador 1 e também a técnica utilizada pelos adversários (item 3). Também é necessário escolher o tipo de jogo que será analisado (item 2). Após todas as opções selecionadas, é iniciada a análise clicando na opção [iniciar análise] (item 4).

Nesta tela de análise também encontra-se uma barra de progressão (item 8) que indica o andamento da rotina. A cada 25%, é concluída a análise de um jogador. Na área representada pelo item 5 será listados os jogadores participantes da análise, juntamente com os truques realizados (item 6) e o percentual de sucesso no jogo (item 7).

Depois de concluída a rotina, são listados os resultados conforme mostrados na Figura 18. Para o número de partidas vencidas é tomado como base as 24 partidas realizadas na análise. O percentual é tomado pela divisão do número de partidas ganhas dividido pelo número de partidas realizadas.

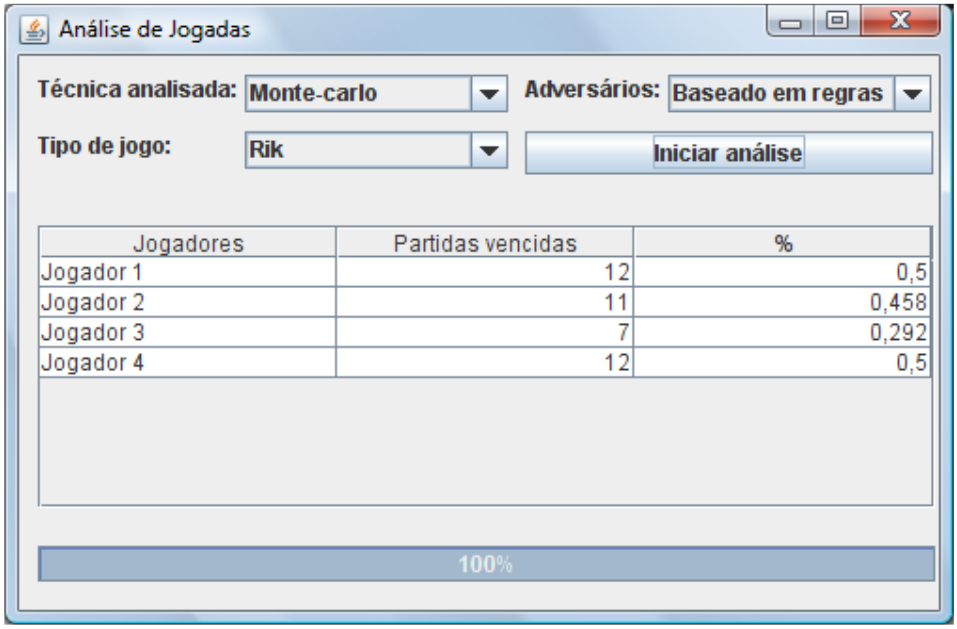


Figura 18 - Resultado da análise de jogadas

Nota-se que os valores percentuais variam entre 0% e 1%, ou seja, se o jogador vencer todas as 24 partidas, ele irá obter o total de 1%.

Nesta tela de resultados é possível perceber que o jogador 1, utilizando a técnica de Monte-Carlo, venceu 12 das 24 partidas realizadas, obtendo um percentual de 0,5% de sucesso nas partidas realizadas. Já o jogador 2 venceu 11 das 24 partidas, tendo um percentual de 0,458%. Os demais jogadores possuíram um percentual de 0,292% e 0,5%.

Finalmente, a outra opção da aplicação é a análise de licitação. Selecionando esta opção, o sistema apresenta uma tela de configuração da análise, conforme verificado na Figura 19.

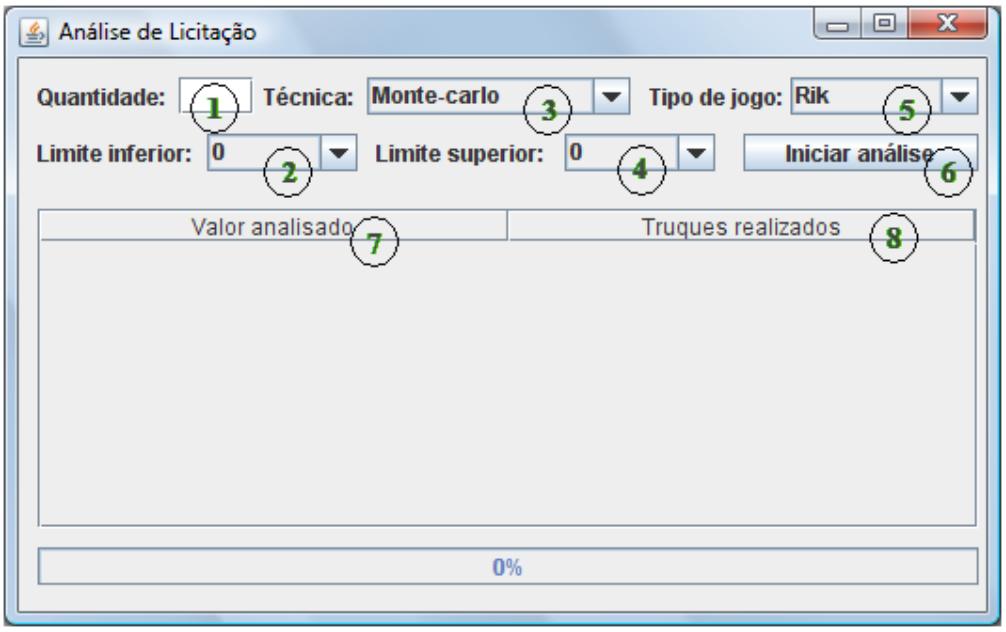


Figura 19 - Tela para análise da licitação

Nesta tela é informada a quantidade de vezes que a análise é repetida (item 1) para uma aproximação maior do resultado. Também deve ser informada a técnica (item 3) e o tipo de jogo a ser analisado (item 5). O limite inferior (item 2) e limite superior (item 3) referem-se a quantidade de truques estimados a serem verificados. Por exemplo, deseja-se que a análise seja realizada para os valores entre 2 (limite inferior) e 4 (limite superior). Desta forma, após iniciar a análise (item 6), a rotina é efetuada *x* vezes para quando o valor estimado de jogos é 2, outras *x* vezes para o valor de jogos estimados é igual a 3 jogos e finalizando mais *x* vezes para o valor de jogos igual a 4, sendo *x* o valor informado no campo do item 1.

Após finalizada a rotina, o resultado é apresentado de acordo com os limites selecionados. Na coluna valor analisado (item 7) é mostrado o número de truques estimados na rotina para cálculo da licitação. Já na coluna truques realizados (item 8) é listado o número de truques realizados após a partida onde o número estimado foi calculado, conforme mostra a Figura 20.

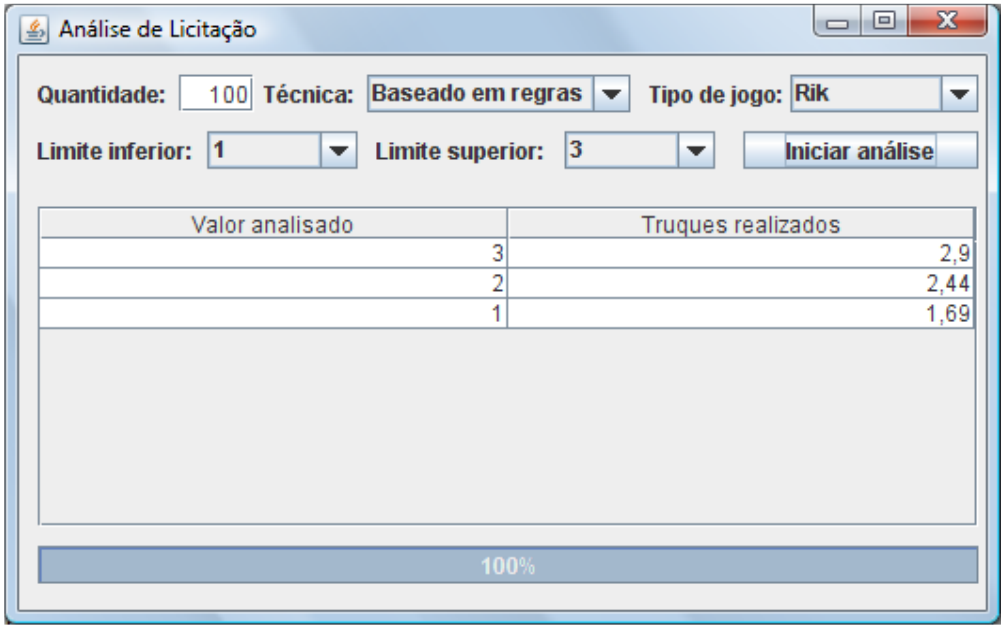


Figura 20 - Resultado da análise de licitação

3.7 RESULTADOS E DISCUSSÃO

Vorsteveld (2007) realizou um experimento buscando identificar a melhor técnica para o jogo Rikken. Esta seção descreve os resultados apresentados comparados às conclusões obtidas do presente trabalho.

3.7.1 Análise da licitação

Esta análise mostra qual método se aproxima mais do número de truques realizados na partida.

Cada ponto no gráfico representa a média de 500 execuções do experimento, exceto para ganhar 0, 11, 12 ou 13 truques nos tipos de jogos Rik e Solo 8. Estas médias baseiam-se em menos de 500 execuções, mas não são tão interessantes para esses tipos de jogos porque as condições mínimas para jogar os mesmos são entre 4 e 9 truques.

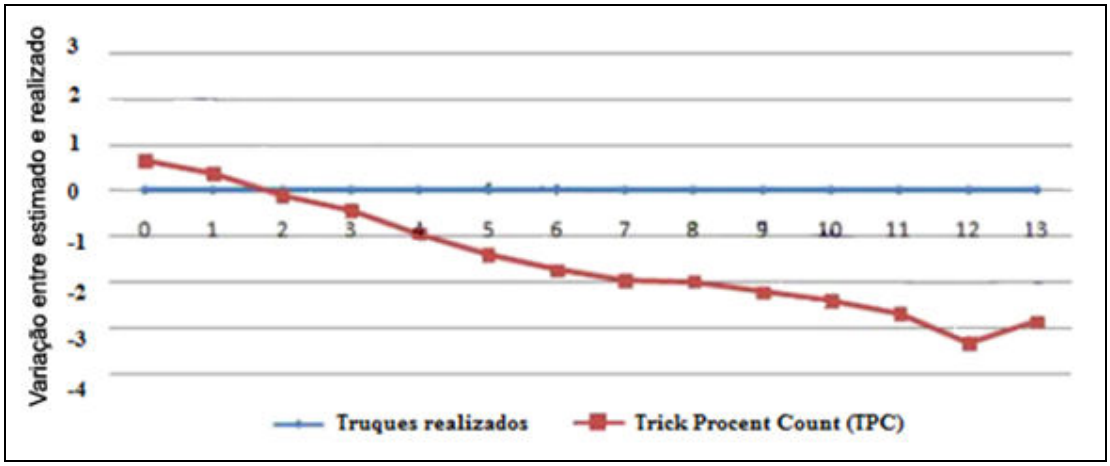
Para determinar qual o método de licitação é o mais exato, nós olhamos para o total de estimativas erradas.

A flutuação no início e no final é causada pelo baixo número de experiências, uma vez

que é pouco provável que se vença 0 ou 13 truques jogando o jogo Rik ou Solo 8.

O método TPC quase sempre subestima o número real de truques realizados na partida e a técnica Monte-Carlo superestima o número total de truques.

Nos tipos de jogos Rik e Solo 8 nota-se uma proximidade dos resultados da aplicação Rikken com os resultados obtidos em Vorsteveld (2007), como no caso onde a estimativa de jogos era de 5, o total de vitórias foi de aproximadamente 4,25 conforme mostrados nos Figuras 21 e 22.



Fonte: adaptação de Vorsteveld (2007).

Figura 21 - Resultado da análise de licitação do jogador baseado em regras para os tipos de jogos Rik e Solo 8 (Vorsteveld)

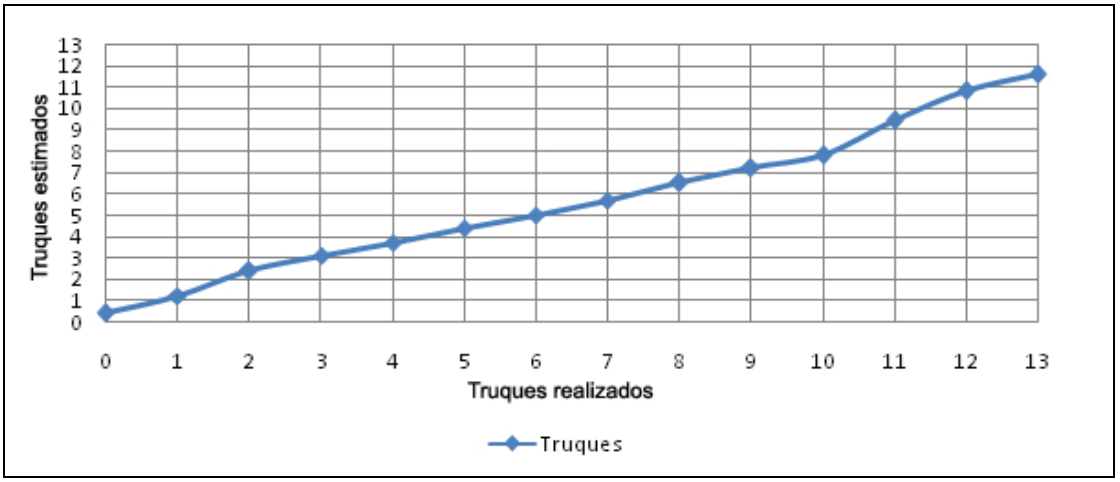
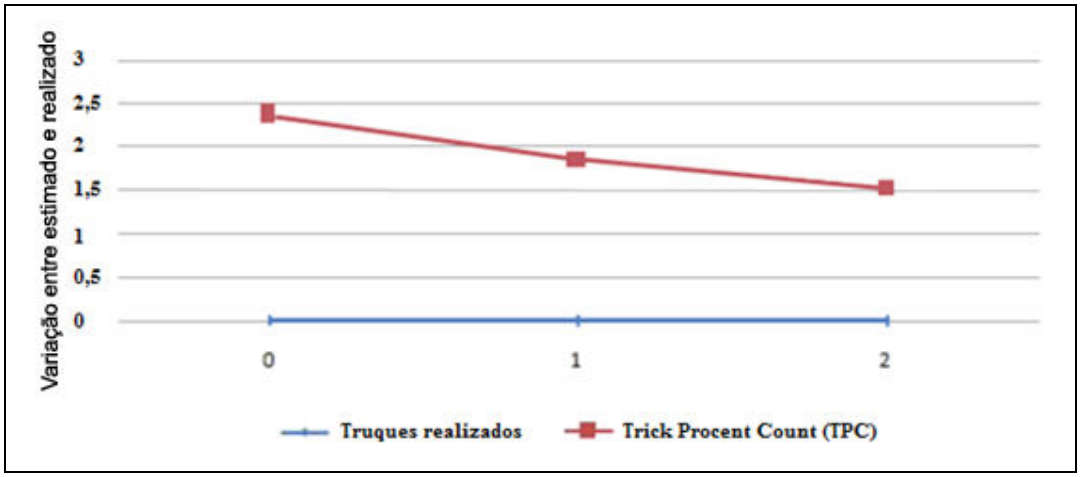


Figura 22- Resultado da análise de licitação do jogador baseado em regras para os tipos de jogos Rik e Solo 8

No tipo de jogo Piek nota-se um melhor desempenho na aplicação Rikken comparada à análise do artigo. Neste experimento utiliza-se um número baixo dos truques realizados pois o objetivo do jogo é a realização de 1 truque apenas. Os gráficos dos resultados podem ser verificados nas Figuras 23 e 24.



Fonte: adaptação de Vorsteveld (2007).

Figura 23 - Resultado da análise de licitação do jogador baseado em regras para o tipo de jogo Piek (Vorsteveld)

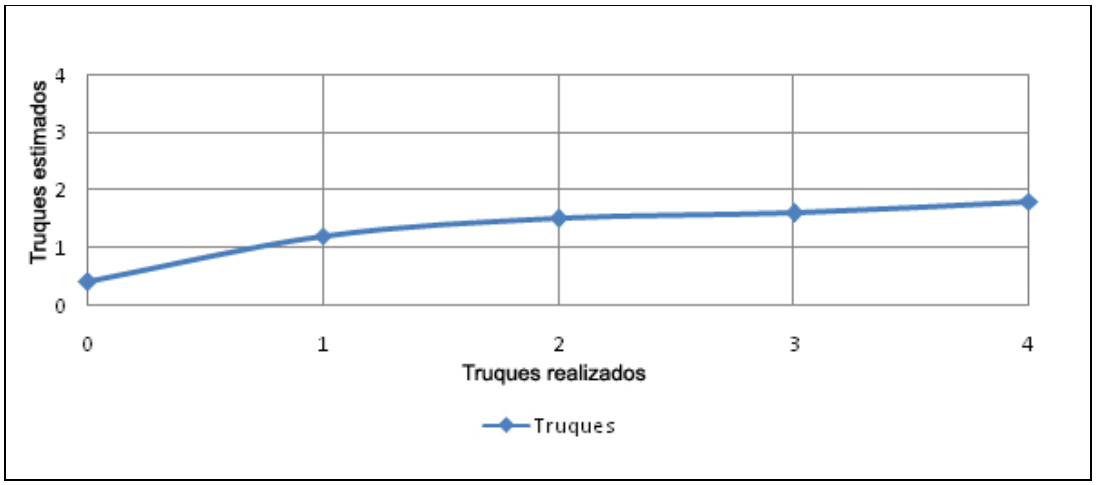
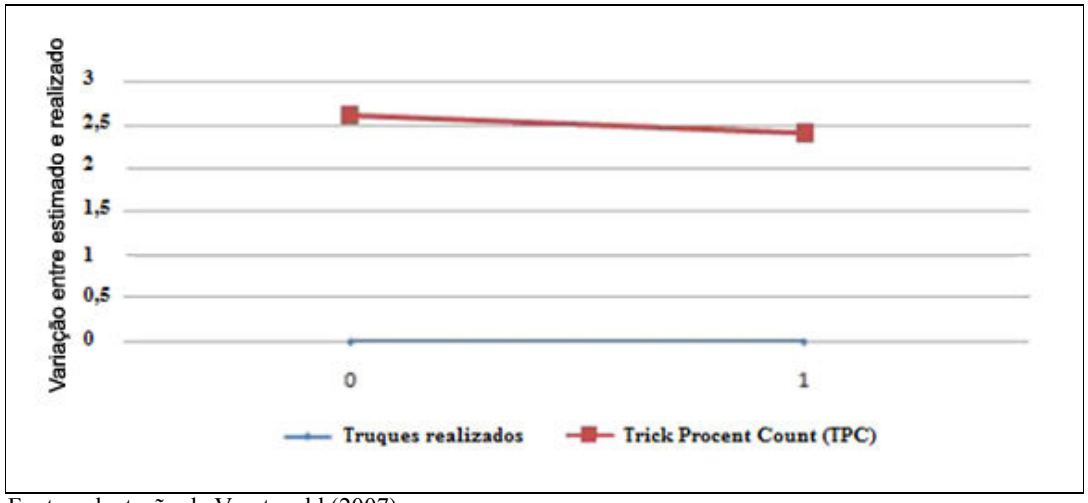


Figura 24 - Resultado da análise de licitação do jogador baseado em regras para o tipo de jogo Piek

Finalizando a análise da licitação do jogador baseado em regras, é possível identificar um melhor desempenho do participante desenvolvido no presente trabalho para o tipo de jogo Misere, conforme mostrado nos gráficos das Figuras 25 e 26.

Semelhante aos resultados obtidos para o tipo de jogo Piek, os resultados obtidos são baixos, pois o objetivo do jogo é a realização de 0 (zero) truques.



Fonte: adaptação de Vorsteveld (2007).

Figura 25 - Resultado da análise de licitação do jogador baseado em regras para o tipo de jogo Misere (Vorsteveld)

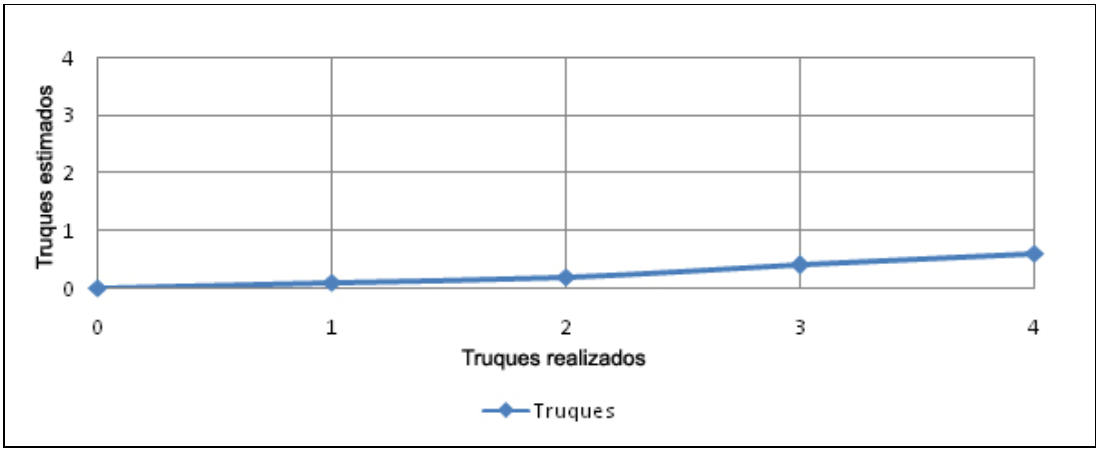
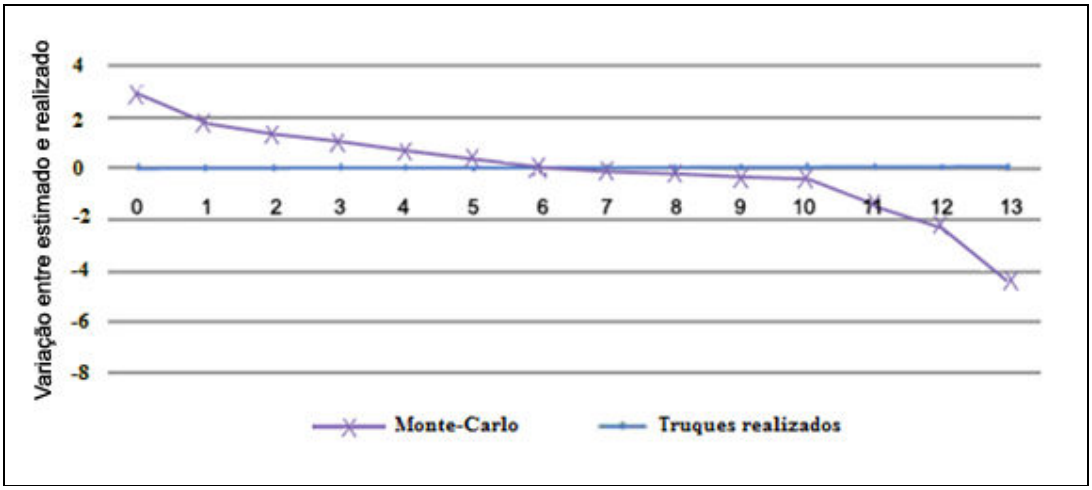


Figura 26 - Resultado da análise de licitação do jogador baseado em regras para o tipo de jogo Misere

Para as análises realizadas com o jogador Monte-Carlo, o número de repetições foi de 100 vezes para os valores entre 1 e 8. Para os demais valores o número de repetições alterado conforme necessidade. Para os valores entre 8 e 10 o número de repetições foi 50 e, para os demais foram realizadas apenas 10 repetições.

Utilizando a técnica de Monte-Carlo, o jogador obteve o resultado de truques realizados muito próximo ao que foi estimado na rodada de licitação para os tipos de jogos Rik e Solo 8, como pode ser observado na Figura 27. Comparado ao resultado obtido em Vorsteveld (2007), pode notar-se uma semelhança entre os gráficos. É possível também observar uma variação nos valores iniciais e finais da análise, devido ao fato de o número de repetições para estes valores serem menores, conforme mostrado na Figura 28.



Fonte: adaptação de Vorsteveld (2007).

Figura 27 - Resultado da análise de licitação do jogador Monte-Carlo para os tipos de jogos Rik e Solo 8 (Vorsteveld)

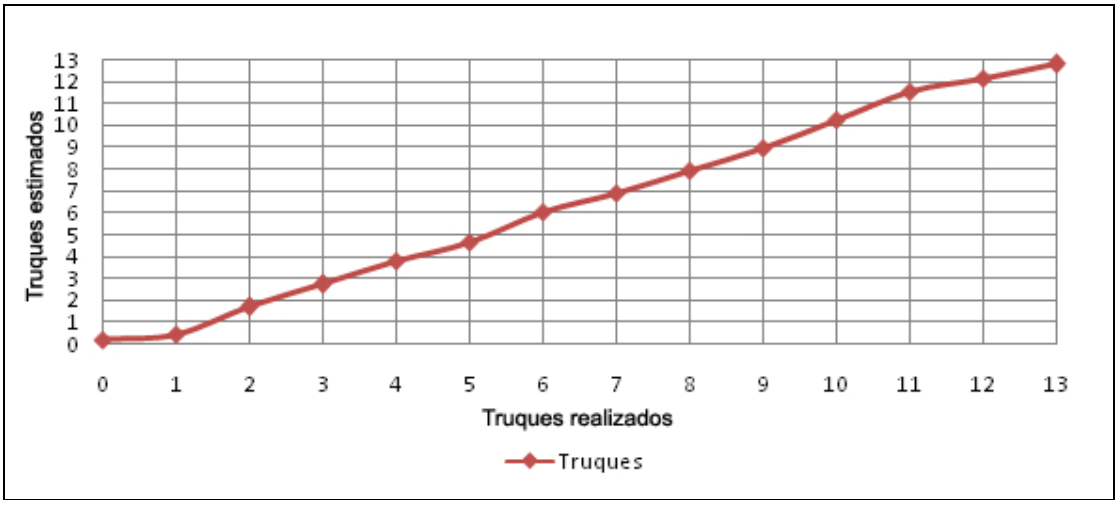
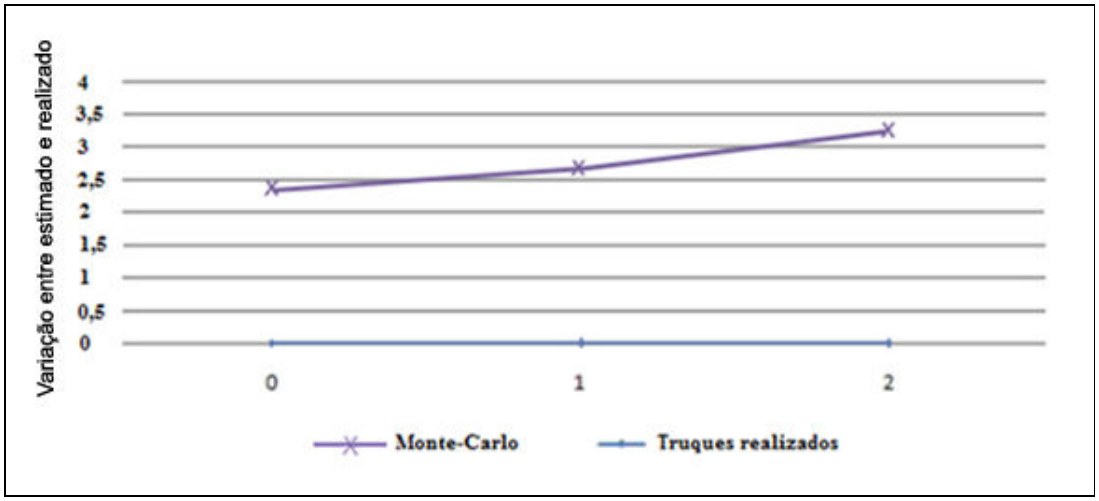


Figura 28 - Resultado da análise de licitação do jogador Monte-Carlo para os tipos de jogos Rik e Solo 8

Na análise para o tipo de jogo Piek, é possível verificar que o jogador da aplicação Rikken possui um resultado mais próximo ao esperado para este tipo de jogo, comparado ao gráfico apresentado por Vorsteveld (2007), conforme ilustrado nas Figuras 29 e 30.



Fonte: adaptação de Vorsteveld (2007).

Figura 29 - Resultado da análise de licitação do jogador Monte-Carlo para o tipo de jogo Piek (Vorsteveld)

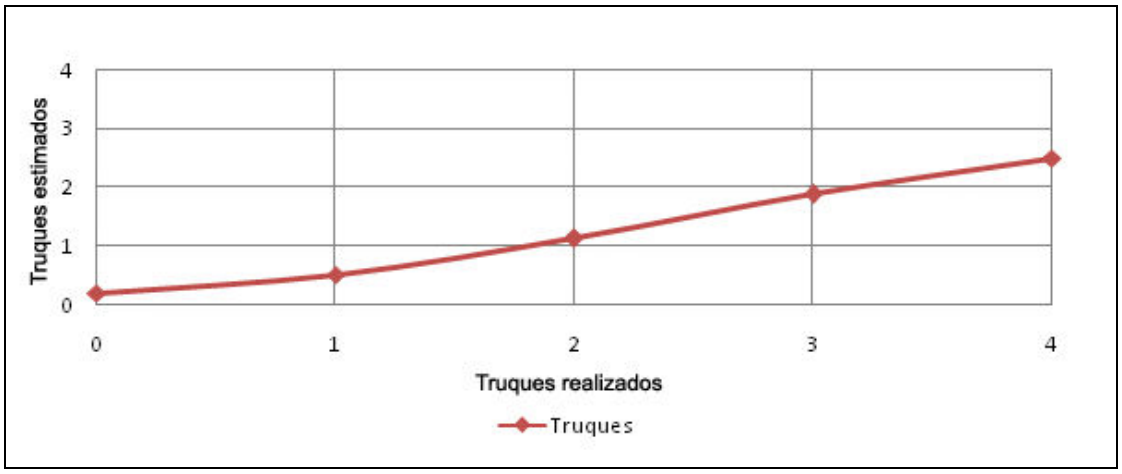
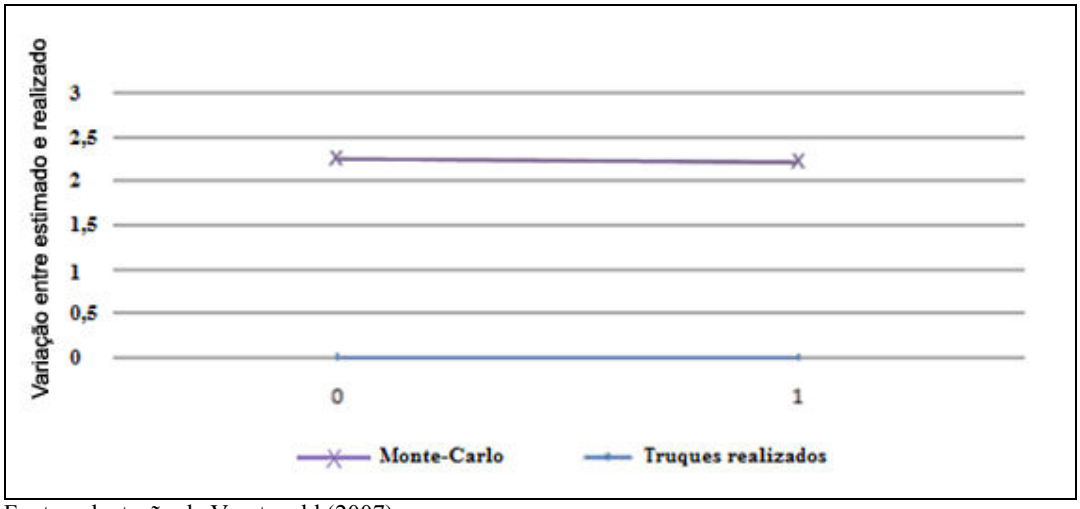


Figura 30 - Resultado da análise de licitação do jogador Monte-Carlo para o tipo de jogo Piek

Finalizando a análise da rodada de licitação, são apresentados os gráficos para o tipo de jogo Misere onde, semelhante ao visto no tipo de jogo Piek, o jogador da aplicação Rikken obteve um melhor desempenho comparado ao participante apresentado por Vorsteveld conforme apresentado nas Figuras 31 e 32.



Fonte: adaptação de Vorsteveld (2007).

Figura 31 - Resultado da análise de licitação do jogador Monte-Carlo para o tipo de jogo Misere (Vorsteveld)

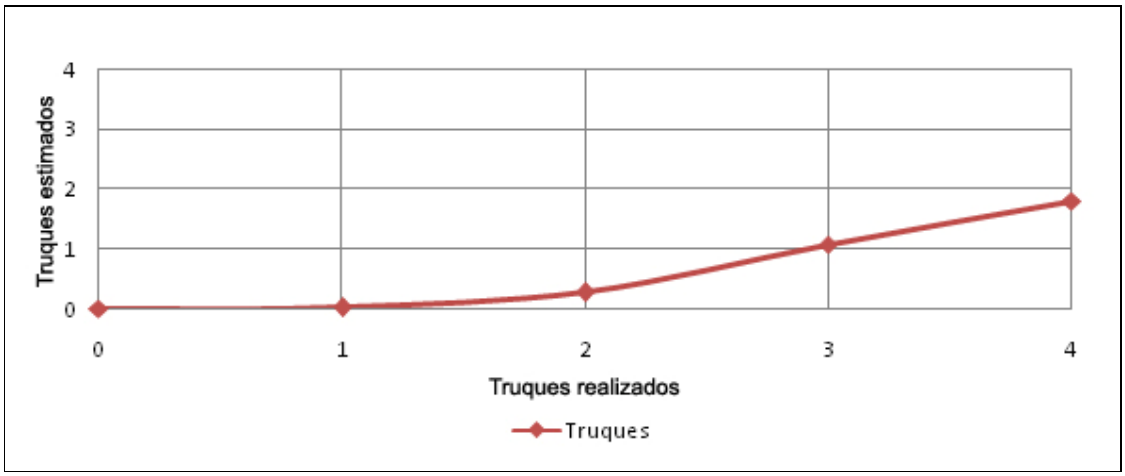


Figura 32 - Resultado da análise de licitação do jogador Monte-Carlo para o tipo de jogo Misere (Vorsteveld)

3.7.2 Análise dos jogos

Nas análises dos jogos, primeiramente são analisadas as cartas para verificar qual o tipo de jogo que o conjunto de cartas se enquadra. Após selecionar as treze cartas *x*, é dado início a um processo de quatro etapas.

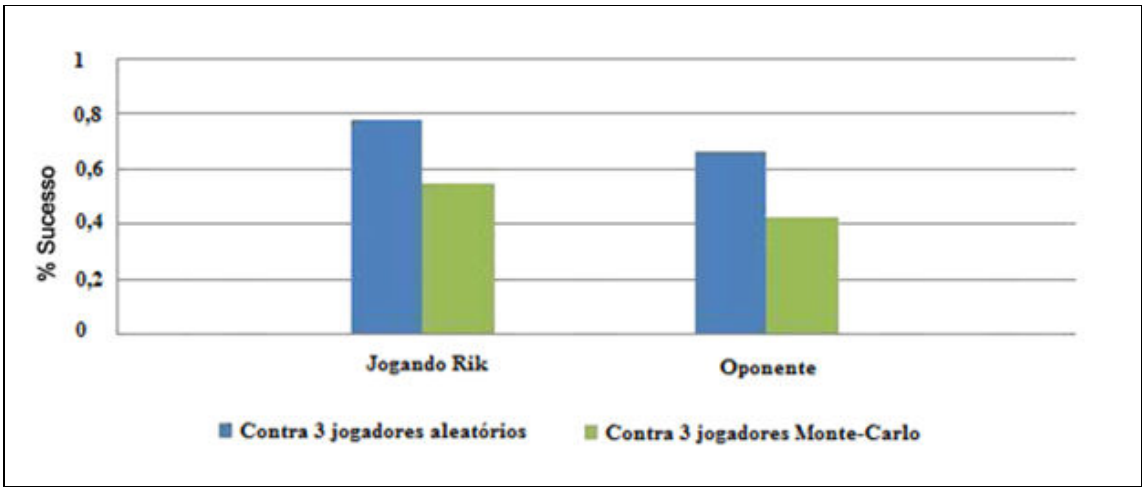
No início de cada etapa, um novo jogador recebe o conjunto de cartas *x*. Após isso cada participante inicia seis partidas, em um total 24 por etapa. A análise é concluída após a execução das quatro etapas, totalizando 96 partidas.

Este tratamento faz-se necessária para obtermos uma análise sobre o conjunto de cartas *x*, igualmente distribuídos entre todos os participantes. O resultado da análise de cada jogador

é obtido após a realização das 24 jogadas com o conjunto *x* de cartas.

Utilizando-se este método, a análise resulta em conclusões muito variadas, desta forma, foi realizada a repetição desta rotina dez vezes para uma aproximação do resultado satisfatório.

Para o jogador utilizando a técnica baseada em regras, verificou-se uma grande proximidade dos resultados da análise documentada em Vorsteveld (2007) como pode ser verificado nas Figuras 33 e 34. Jogando como oponente não é apresentada a mesma semelhança nos resultados.



Fonte: adaptação de Vorsteveld (2007).

Figura 33 - Resultado da análise do jogador baseado em regras jogando Rik (Vorsteveld)

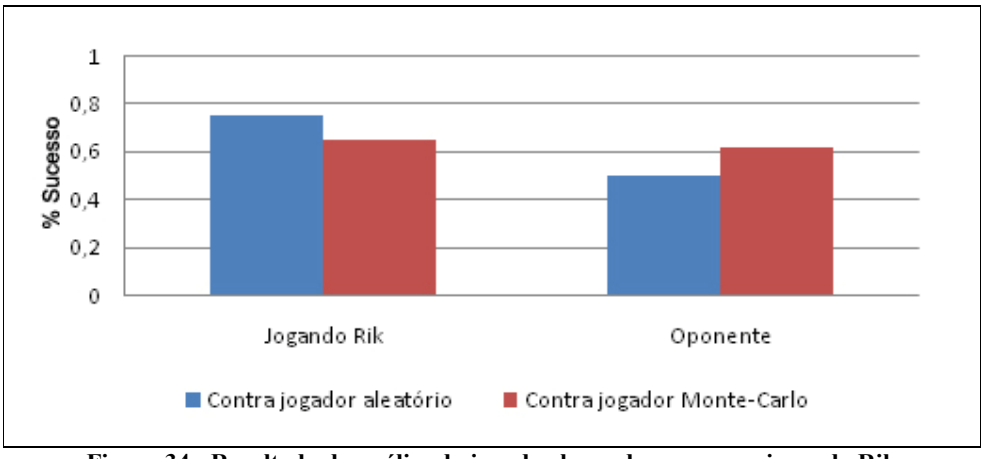
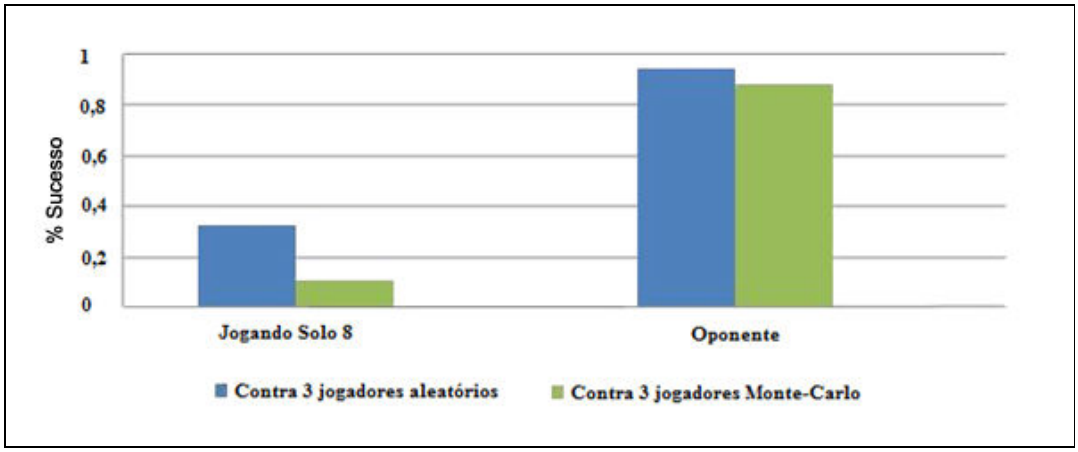


Figura 34 - Resultado da análise do jogador baseado em regras jogando Rik

Quando o tipo de jogo é o Solo 8, nota-se uma pequena variação dos resultados obtidos conforme os gráficos das Figuras 35 e 36.



Fonte: adaptação de Vorsteveld (2007).

Figura 35 - Resultado da análise do jogador baseado em regras jogando Solo 8 (Vorsteveld)

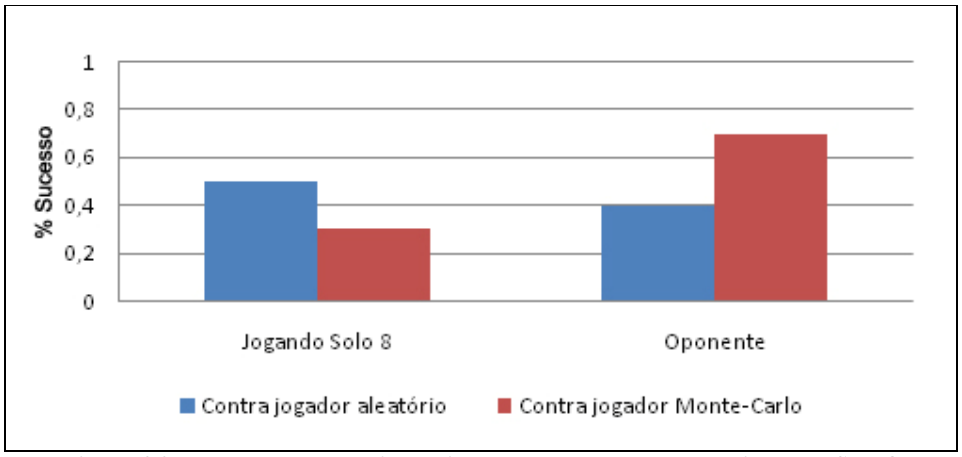
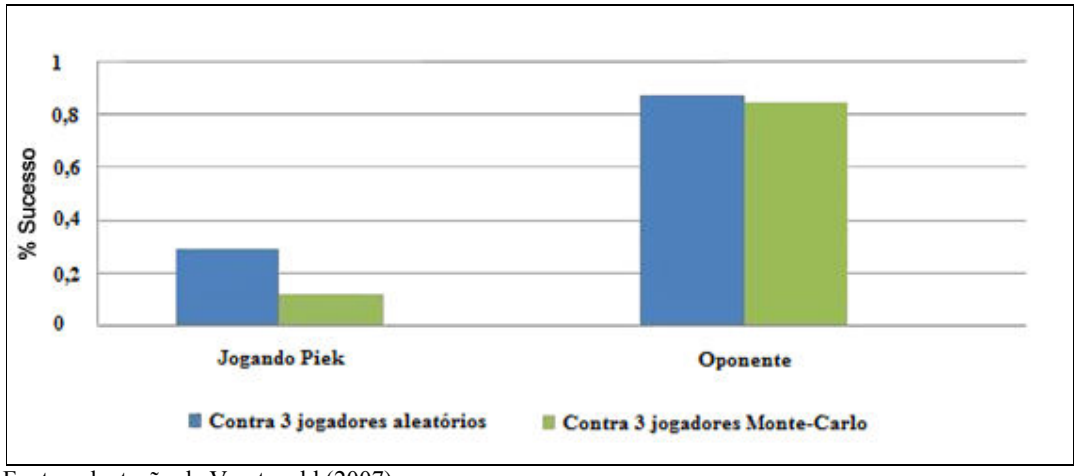


Figura 36 - Resultado da análise do jogador baseado em regras jogando Solo 8

No tipo de jogo Piek, é possível concluir que o jogador desenvolvido pela aplicação Rikken possui um melhor desempenho no jogo do que o documentado por Vorsteveld (2007) conforme mostrado nas Figuras 37 e 38.



Fonte: adaptação de Vorsteveld (2007).

Figura 37 - Resultado da análise do jogador baseado em regras jogando Piek (Vorsteveld)

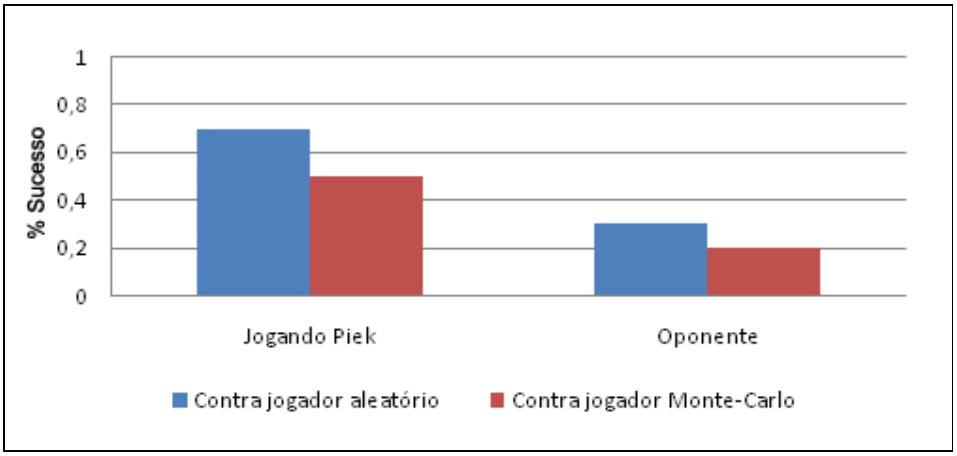
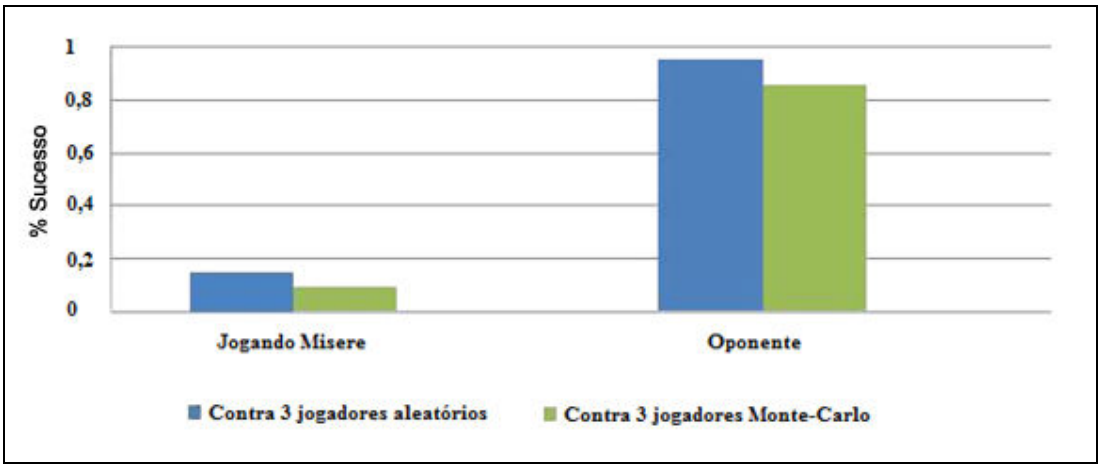


Figura 38 - Resultado da análise do jogador baseado em regras jogando Piek

Para o tipo de jogo Misere, observa-se que, semelhante a análise de licitação, o jogador da aplicação possui um resultado melhor que o apresentado por Vorsteveld (2007), conforme apresentado nas Figuras 39 e 40.



Fonte: adaptação de Vorsteveld (2007).

Figura 39 - Resultado da análise do jogador baseado em regras jogando Misere (Vorsteveld)

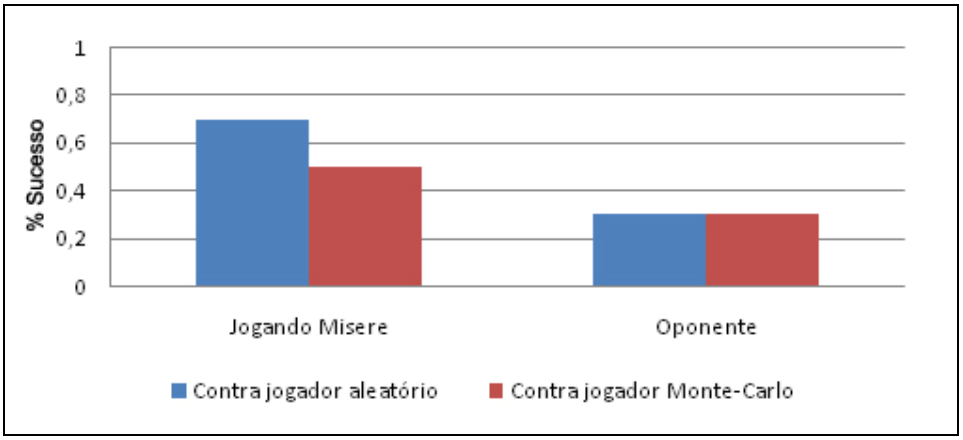
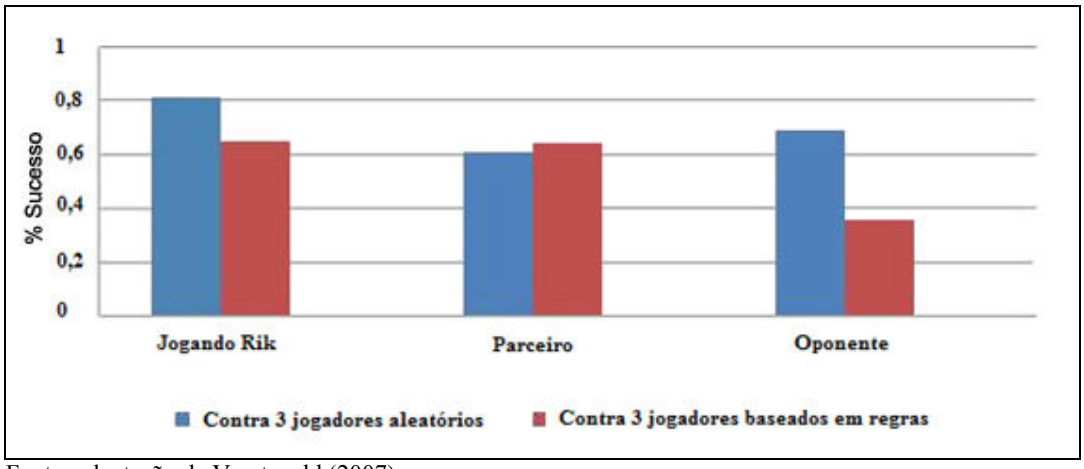


Figura 40 - Resultado da análise do jogador baseado em regras jogando Misere

Na análise de jogadas utilizando-se a técnica de Monte-Carlo, utiliza-se o número fixo de 100 simulações.

Realizando a análise comparativa, verifica-se que no tipo de jogo Rik existe uma grande proximidade dos gráficos apresentados por Vorsteveld (2007) conforme ilustrados nas Figuras 41 e 42.



Fonte: adaptação de Vorsteveld (2007).

Figura 41 - Resultado da análise do jogador Monte-Carlo jogando Rik (Vorsteveld)

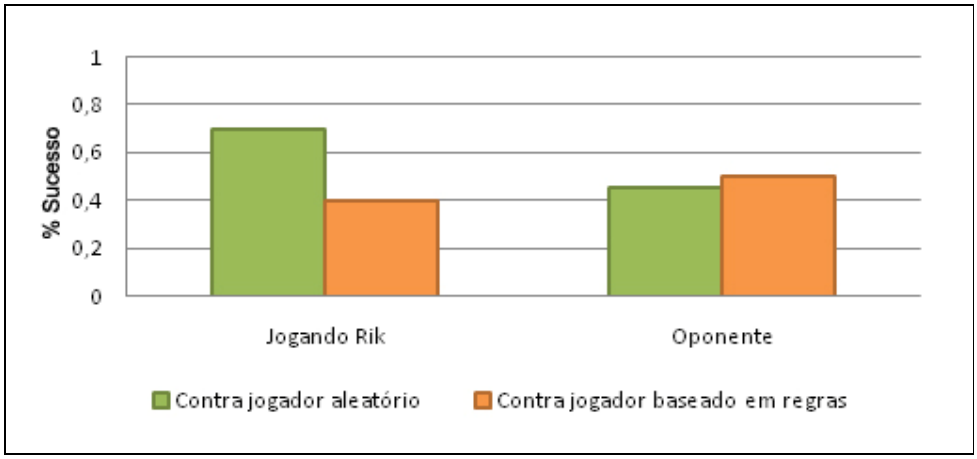
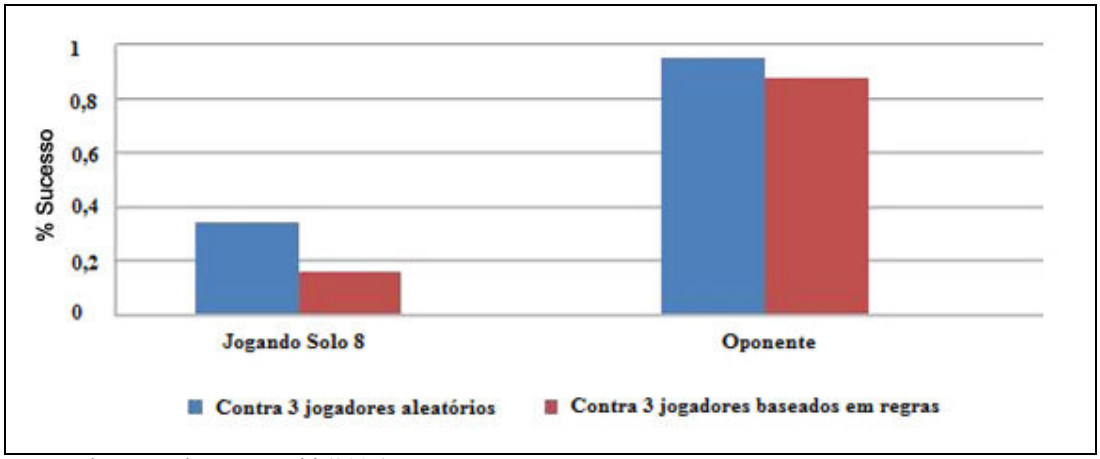


Figura 42 - Resultado da análise do jogador Monte-Carlo jogando Rik

Jogando o tipo de jogo Solo 8 é possível verificar uma variação entre os gráficos, principalmente quando trata-se do oponente, onde é realizado um grande número de truques conforme Vorsteveld (2007). Estes resultados estão apresentados nas Figuras 43 e 44.



Fonte: adaptação de Vorsteveld (2007).

Figura 43 - Resultado da análise do jogador Monte-Carlo jogando Solo 8 (Vorsteveld)

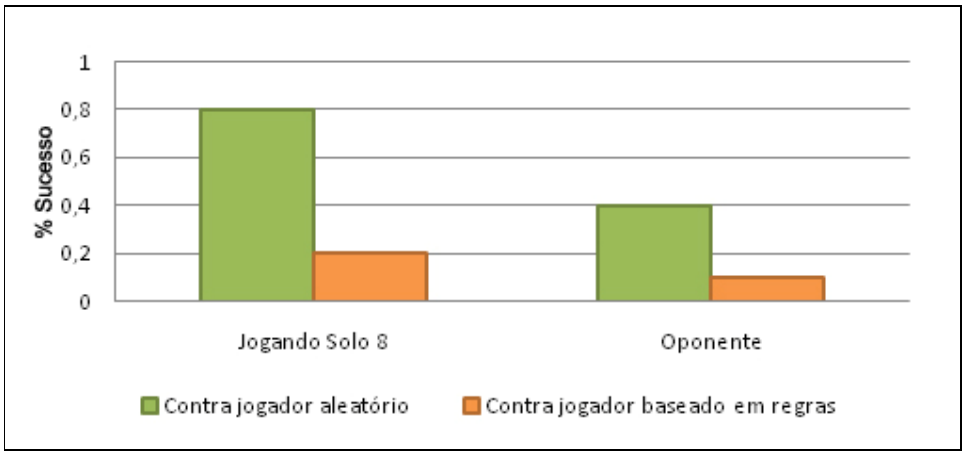
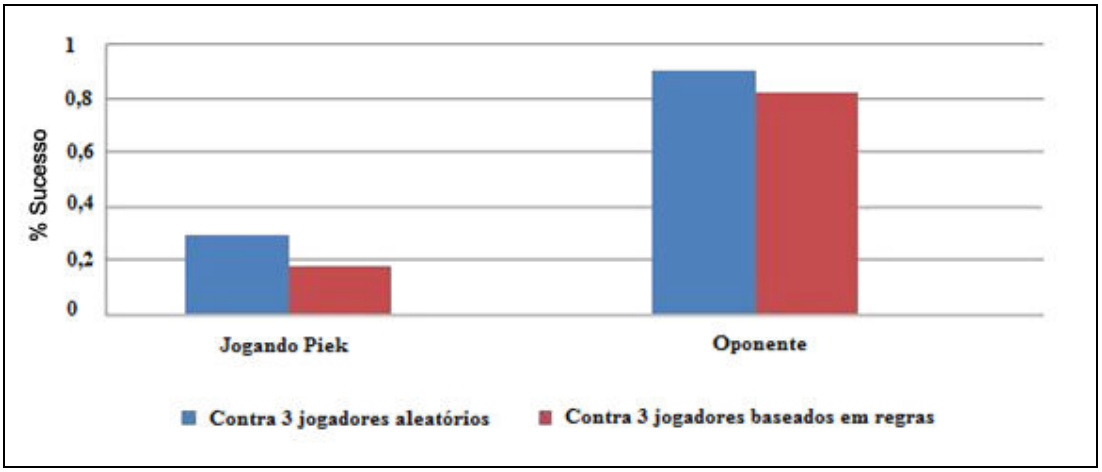


Figura 44 - Resultado da análise do jogador Monte-Carlo jogando Solo 8

No tipo de jogo Piek é possível verificar que os resultados obtidos são muito próximos com uma variação na análise de oponente, conforme ilustrado nas Figuras 45 e 46.



Fonte: adaptação de Vorsteveld (2007).

Figura 45 - Resultado da análise do jogador Monte-Carlo jogando Piek (Vorsteveld)

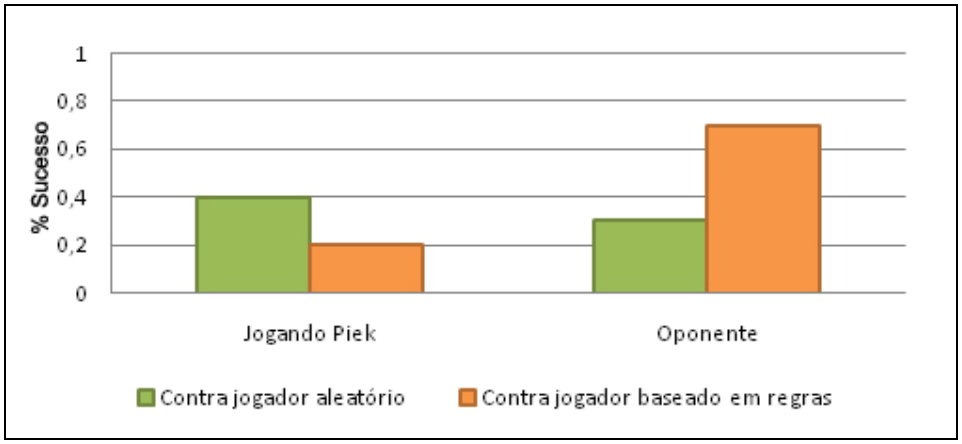
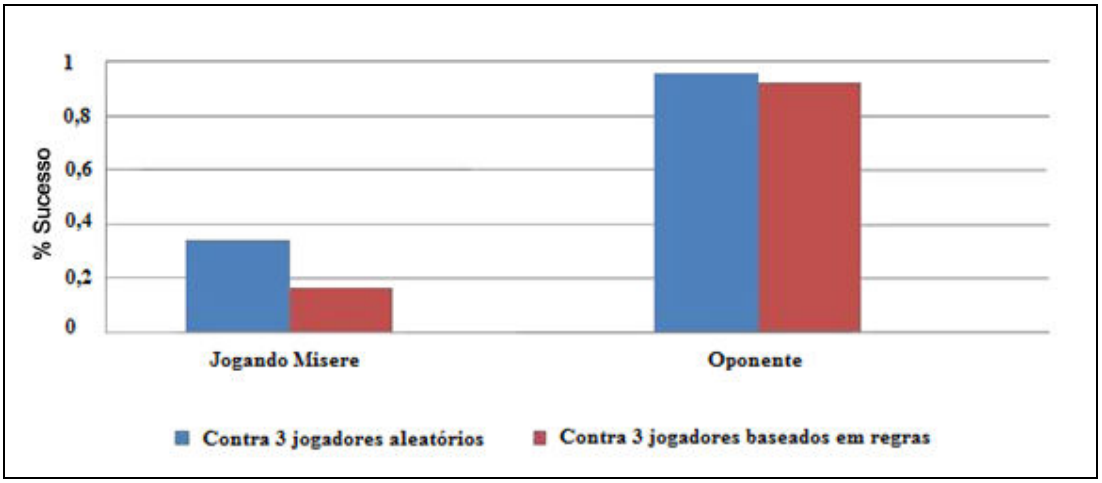


Figura 46 - Resultado da análise do jogador Monte-Carlo jogando Piek

Finalizando a análise, nota-se uma grande proximidade do jogador Monte-Carlo jogando Misere entre os gráficos apresentados nas Figuras 47 e 48.



Fonte: adaptação de Vorsteveld (2007).

Figura 47 - Resultado da análise do jogador Monte-Carlo jogando Misere (Vorsteveld)

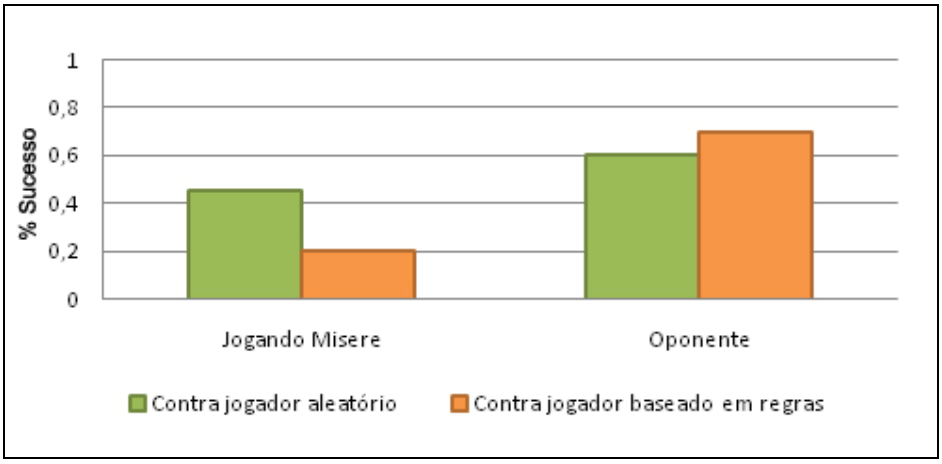


Figura 48 - Resultado da análise do jogador Monte-Carlo jogando Misere

4 CONCLUSÕES

O presente trabalho teve como proposta o desenvolvimento de uma aplicação para reproduzir e validar as técnicas relatadas no artigo descrito em Vorsteveld (2007). Durante a implementação deste protótipo foram encontradas várias dúvidas sobre como Vorsteveld realiza a distribuição das cartas e sobre como foram as condições da fase de testes.

Estas dúvidas podem ser uns dos fatores que prejudicam na análise dos resultados obtidos. Outra causa deve-se ao fato de, por ser um jogo de cartas, os resultados podem ser os mais variados, pois se trata de busca aleatória de informações imperfeitas. O baixo número de repetições nas análises de jogadas leva a resultados com uma grande variação de valores, impossibilitando uma conclusão precisa.

Uma das maiores preocupações, desde o princípio do experimento, foi quanto ao desempenho da aplicação, tendo em vista o número excessivo de repetições de jogos a cada carta que seria jogada pela técnica de Monte-Carlo. Dessa forma, foram utilizadas técnicas simples e de melhor desempenho na construção das simulações, não sendo necessário o uso de bibliotecas externas.

Os resultados do presente trabalho foram bastante satisfatórios, pois mesmo tratando-se de análise de técnicas, foram acopladas técnicas que enriqueceram o projeto como a LTC, utilizado na escolha do parceiro da técnica Monte-Carlo, e a TPC verificado na tomada de decisão do tipo de jogo pelo jogador baseado em regras.

Por fim, foi concluído que o desempenho ótimo em jogos de informações imperfeitas exige raciocínio sobre as informações correntes e futuras de cada jogador. O método de Monte-Carlo realiza uma aproximação simples calculando a média dos valores de uma ação sobre cada configuração possível de informações.

4.1 EXTENSÕES

Como extensão do presente trabalho propõe-se a implementação do naipe *trump* do jogo Rikken e novas técnicas de inteligência artificial para a realização de comparativos concluindo o melhor método para jogos determinísticos.

Outra proposta seria a possibilidade de interação de jogadores humanos de forma a

permitir a realização de campeonatos com cada autor aplicando sua técnica no seu jogador como uma forma de incentivo para a evolução das técnicas.

REFERÊNCIAS BIBLIOGRÁFICAS

AERNSBERGEN, Frans van. **Kaarten!** [S.l.], 1999, Disponível em: <http://www.toepen.eu/>. Acesso em: 17 nov. 2008.

KISHIMOTO, André. **Inteligência artificial em jogos eletrônicos.** [S.l.], 2004. Disponível em: <http://www.programadoresdejogos.com/trab_academicos/andre_kishimoto.pdf>. Acesso em: 15 abr. 2008.

MCLEOD, John. **Rikken.** [S.l.], 2004. Disponível em: <http://www.pagat.com/boston/rik.html>. Acesso em: 02 nov. 2008.

MINETTO, Elton. **Método de Monte-Carlo.** [S.l.], 2006. Disponível em: <http://www.eltonminetto.net/docs/monteCarloDistribuido.pdf>. Acesso em: 24 maio 2008.

MOURA, Altair. **Análise de projetos sob condições de risco**. [S.l], 2004. Disponível em: <http://www.gestaodoagronegocio.com.br/aziz/download/MonteCarlo.pdf>. Acesso em: 04 nov. 2008.

PERALTA, Luís. **Cálculos Monte Carlo:** um jogo de azar? [S.l.], 2000. Disponível em: <http://ltodi.est.ips.pt/dmat/seminarios/MonteCarlo.html>. Acesso em: 25 maio 2008.

RUSSEL, Stuart; NORVIG, Peter. **Inteligência artificial.** 2. ed. Tradução Vandenberg D. de Souza. Rio de Janeiro: Elseiver, 2004.

SIMON, Herbert. **Sistemas baseados em regras**. [S.l.], 2007. Disponível em: <http://mestradosiad.blogspot.com/2007/11/sistemas-baseados-em-regras.html>. Acesso em: 17 set. 2008.

SOUTO, Antonio C. S. **Uso de redes neurais artificiais na simulação Monte Carlo aplicado ao problema de dobramento de proteínas.** [S.l.], 2006. Disponível em: <http://bdtd.unisinos.br/tde_arquivos/1/TDE-2006-11-14T135841Z-114/Publico/Uso%20de%20redes%20neurais%20artificiais.pdf>. Acesso em: 04 jun. 2008.

SPARX SYSTEMS. **Enterprise Architect**. Victoria, Austrália. 2005. Disponível em: <http://www.sparxsystems.com.au/products/ea>. Acesso em: 04 nov. 2008.

STELTING, Stephen; MAASSEN, Olav. **Applied Java patterns**. Palo Alto: Sun Microsystems Press, 2002.

TRUNFIO, Paul; MCGATH, Gary. **Monte Carlo estimation for Pi**. [S.l], 2000. Disponível em: <http://argento.bu.edu/java/java/montepi>. Acesso em: 04 nov. 2008.

TEKNOMO, Kardi. **What is Monte Carlo simulation.** [S.l.], 2006. Disponível em: <http://people.revoledu.com/kardi/tutorial/Simulation/MonteCarlo.html>. Acesso em: 04 jun. 2008.

VIGNATTI, André L. **Aleatoriedade e suas aplicações em projetos de redes e sistemas distribuídos**. [S.l.], 2007. Disponível em: <http://www.ic.unicamp.br/~vignatti/downloads/random.pdf>. Acesso em: 27 out. 2008.

VORSTEVELD, Viktor. **Knowledge vs. power in the game Rikken.** 2007. 55 f. Master Thesis (Master of Science of Knowledge Engineering) – Faculty of Humanities an Sciences, Universiteit Maastricht, Maastricht. Disponível em: <http://www.cs.unimaas.nl/~uiterwyk/Theses/MSc/Vorsteveld_thesis.pdf >. Acesso em: 22 mar. 2008.

APÊNDICE A – Tabela comparativa do resultado da análise de licitação

No quadro 18 é apresentada a tabela com os resultados da análise comparativa da licitação.

Jogador baseado em regras

Rik/Solo8

Realizado	Rikken	Artigo*
13	11,6	10,2
12	10,8	9,8
11	9,45	10,3
10	7,8	7,8
9	7,2	6,9
8	6,5	6
7	5,7	5,2
6	5	4,7
5	4,4	3,8
4	3,7	3,2
3	3,1	2,7
2	2,4	1,9
1	1,2	1,2
0	0,4	0,7

Piek

Realizado	Rikken	Artigo*
2	1,5	3,5
1	1,2	2,8
0	0,4	2,3

Misere

Realizado	Rikken	Artigo*
1	0,1	3,2
0	0	2,6

Jogador Monte-Carlo

Rik/Solo8

Realizado	Rikken	Artigo*
13	12,78	9,8
12	12,12	11,01
11	11,56	10,8
10	10,23	9,8
9	8,97	8,2
8	7,94	7,5
7	6,92	6,8
6	6,06	6
5	4,64	5,1
4	3,81	4,6
3	2,74	4
2	1,7	3,2
1	0,43	2,8
0	0,22	1,8

Piek

Realizado	Rikken	Artigo*
2	1,12	5,5
1	0,51	4,1
0	0,19	3

Misere

Realizado	Rikken	Artigo*
1	0,02	3,5
0	0	2,6

* Resultados aproximados

Quadro 20 - Resultado da análise comparativa da licitação

APÊNDICE B – Tabela comparativa do resultado da análise das jogadas

No quadro 19 é apresentada a tabela com os resultados da análise comparativa das jogadas.

Jogador baseado em regras				Monte-Carlo			
		Jogando Rik	Oponente			Jogando Rik	Oponente
Rikken	Contra jogador aleatório	0,75	0,5	Rikken	Contra jogador aleatório	0,7	0,45
	Contra jogador Monte-Carlo	0,65	0,62		Contra jogador baseado em regras	0,4	0,5
Artigo	Contra jogador aleatório	0,78	0,62	Artigo	Contra jogador aleatório	0,8	0,7
	Contra jogador Monte-Carlo	0,58	0,42		Contra jogador baseado em regras	0,6	0,4
		Jogando Solo 8	Oponente			Jogando Solo 8	Oponente
Rikken	Contra jogador aleatório	0,5	0,4	Rikken	Contra jogador aleatório	0,8	0,4
	Contra jogador Monte-Carlo	0,3	0,7		Contra jogador baseado em regras	0,2	0,1
Artigo	Contra jogador aleatório	0,35	0,9	Artigo	Contra jogador aleatório	0,37	0,93
	Contra jogador Monte-Carlo	0,1	0,85		Contra jogador baseado em regras	0,1	0,85
		Jogando Piek	Oponente			Jogando Piek	Oponente
Rikken	Contra jogador aleatório	0,7	0,3	Rikken	Contra jogador aleatório	0,4	0,3
	Contra jogador Monte-Carlo	0,5	0,2		Contra jogador baseado em regras	0,2	0,7
Artigo	Contra jogador aleatório	0,3	0,83	Artigo	Contra jogador aleatório	0,3	0,85
	Contra jogador Monte-Carlo	0,18	0,82		Contra jogador baseado em regras	0,16	0,82
		Jogando Misere	Oponente			Jogando Misere	Oponente
Rikken	Contra jogador aleatório	0,7	0,3	Rikken	Contra jogador aleatório	0,45	0,6
	Contra jogador Monte-Carlo	0,5	0,3		Contra jogador baseado em regras	0,2	0,7
Artigo	Contra jogador aleatório	0,18	0,87	Artigo	Contra jogador aleatório	0,38	0,97
	Contra jogador Monte-Carlo	0,13	0,82		Contra jogador baseado em regras	0,15	0,96

Quadro 21 - Resultado da análise comparativa das jogadas