UNIVERSIDADE FEDERAL DO AMAZONAS
FACULDADE DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

ENDREWS SZNYDER SOUZA DA SILVA

PROPOSTA DE UM AGENTE PARA O JOGO DE DOMINÓ DE 4 PONTAS UTILIZANDO O ALGORITMO *EXPECTIMINIMAX*

MANAUS
2015

UNIVERSIDADE FEDERAL DO AMAZONAS
FACULDADE DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

ENDREWS SZNYDER SOUZA DA SILVA

PROPOSTA DE UM AGENTE PARA O JOGO DE DOMINÓ DE 4 PONTAS UTILIZANDO O ALGORITMO *EXPECTIMINIMAX*

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal do Amazonas, como requisito parcial para a obtenção do título de Mestre em Engenharia Elétrica na área de concentração Controle e Automação de Sistemas.

Orientador: Prof. Dr. Cícero Ferreira Fernandes Costa Filho

Co-orientador: Profa. Dra. Marly Guimarães Fernandes Costa

MANAUS
2015

Ficha Catalográfica

Ficha catalográfica elaborada automaticamente de acordo com os dados fornecidos pelo(a) autor(a).

Silva, Endrews Sznyder Souza da
S586p Proposta de um agente para o jogo de dominó de 4 pontas utilizando o algoritmo expectiminimax / Endrews Sznyder Souza da Silva. 2015
93 f.: il. color; 31 cm.

Orientadora: Cícero Ferreira Fernandes Costa Filho
Coorientadora: Marly Guimarães Fernandes CostaMarly Guimarães Fer
Dissertação (Mestrado em Engenharia Elétrica) - Universidade Federal do Amazonas.

1. dominó. 2. expectiminimax. 3. algoritmos genéticos. 4. não determinístico. 5. busca por fases. I. Costa Filho, Cícero Ferreira Fernandes II. Universidade Federal do Amazonas III. Título

ENDREWS SNYDER SOUZA DA SILVA

PROPOSTA DE UM AGENTE PARA O JOGO DE DOMINÓ DE 4 PONTAS UTILIZANDO ALGORITMO EXPECTIMINIMAX.

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal do Amazonas, como requisito parcial para obtenção do título de Mestre em Engenharia Elétrica na área de concentração Controle e Automação de Sistemas.

Aprovado em 04 de setembro de 2015.

BANCA EXAMINADORA

Cícero Ferreira Fernandes Costa Filho, Presidente

Universidade Federal do Amazonas- UFAM

Prof. Dr. Lucas Carvalho Cordeiro, Membro

Universidade Federal do Amazonas- UFAM

Prof. Dr. José Francisco de Magalhães Netto, Membro

Universidade Federal do Amazonas- UFAM

AGRADECIMENTOS

Gostaria de agradecer primeiramente aos meus orientadores, Prof. Dr. Cícero Ferreira Fernandes Costa Filho e Profa. Dra. Marly Guimarães Fernandes Costa. Obtive muito conhecimento em suas disciplinas e orientação extraclasse;

Aos meus pais e familiares pelo suporte, que me permitiu ir até o fim nesta dissertação;

Aos meus colegas de classe que me acompanham desde a graduação: Jonilson, Larissa, Márcio e Robson. O seu apoio e companheirismo foram essências para concluir as disciplinas, e principalmente, para me manter focado em cada etapa do desenvolvimento deste trabalho;

Aos amigos e ex-colegas de faculdade Renata Suzan e Diego Ramon pelo incentivo ao ingresso no programa de pós-graduação;

Também gostaria de agradecer, ao corpo docente do programa de pós-graduação em engenharia elétrica. Grande parte das disciplinas lecionadas contribuíram para a implementação deste trabalho;

À Universidade Federal do Amazonas e aos profissionais do Centro de Tecnologia Eletrônica e da Informação – CETELI. Graças à estrutura oferecida como o espaço físico e materiais de pesquisa, foi possível o desenvolvimento deste trabalho;

Partes dos resultados apresentados neste trabalho foram obtidos através do Projeto de pesquisa e formação de recursos humanos, em nível de graduação e pós-graduação, nas áreas de automação industrial, *softwares* para dispositivos móveis e TV Digital, financiado pela Samsung Eletrônica da Amazônia Ltda., no âmbito da Lei no. 8.387 (art. 2º) /91.

RESUMO

O jogo de dominó é praticado por milhões de pessoas pelo Brasil e no mundo. As pessoas costumam praticar este jogo em reuniões familiares, círculos de amigos e nas universidades. O jogo de dominó não possui uma versão e regras universais, podendo variar de acordo com a região em que é praticado. Apesar de suas diversas variações, este jogo é simples de jogar. O dominó é um jogo de informação incompleta e não determinístico. As incertezas e a característica estocástica o fazem um jogo complexo de solucionar com as metodologias existentes na área de inteligência artificial. Nesse sentido, esta dissertação propõe uma agente para o jogo de dominó de 4 pontas. Esta versão do jogo de dominó é praticada no Amazonas. O agente proposto é baseado no algoritmo de busca *expectiminimax* com busca parcial. A busca parcial tem como consequência o efeito de horizonte, e para atenuar este efeito será realizada uma busca por fases dentro de uma rodada. Além disso, devido à presença dos eventos de chance nesse jogo será necessária a modelagem probabilística para os mesmos. Neste trabalho foram definidas oito estratégias com profundidades diferentes em cada fase das rodadas. Cada estratégia será testada contra a estratégia básica de jogo e contra o melhor resultado obtido pela estratégia que utiliza algoritmos genéticos. A melhor estratégia deste trabalho obteve 72,04% de vitórias em 5000 partidas, contra a estratégia básica de jogo. Contra a melhor estratégia que utiliza algoritmos genéticos o percentual de vitorias foi de 58,34 %.

Palavras-chave: dominó; informação incompleta; não determinístico; *expectiminimax;* efeito de horizonte; busca por fases; algoritmos genéticos.

ABSTRACT

Dominoes game is played by millions of people around Brazil and entire world. People usually play this game during family meetings, with friends and in universities. Dominoes game does not have universal version and rules, the way it is played can vary according its region. Although dominoes' version diversity, this game is simple. Dominoes game to is an incomplete information and non-deterministic type. Uncertainty and stochastic characteristic of this game make it difficult to solve by applying artificial intelligence techniques. Thus, this thesis proposes an agent for four-sided dominoes. This dominoes' version is played in Amazonas state. The proposed agent is based on partial *expectiminimax* algorithm. Partially searching the game tree generates horizon effect, and to reduce it, phase related search will be used in each round phase. Moreover, due to chance events it will be necessary probability modeling to obtain opponent's moves probability values. This thesis proposes eight strategies, which differ from each other in depth search through a round. Each strategy is evaluated against a pair, which uses the basic one and pair with strategy based on genetic algorithm. The best strategy is this work obtained 72,04% of victories in 5000 matches against the basic one, and 58,34% of victories against the strategy based on genetic algorithm.

Keywords: dominoes; incomplete information; non-deterministic; *expectiminimax*, horizon effect; phase related search; genetic algorithm.

LISTA DE ILUSTRAÇÕES

LISTA DE QUADROS

LISTA DE ALGORITMOS

SUMÁRIO

1 INTRODUÇÃO

O jogo de dominó é constituído por peças retangulares. Cada uma está dividida em duas metades iguais, onde são gravados pontos representando valores numéricos. Este jogo apresenta-se em diversas versões, sendo a duplo-seis a versão mais conhecida. Nessa versão, as peças que formam o conjunto geram 28 combinações de valores numéricos de 1 a 6 gravados em ambas as metades. A Figura 1 mostra o conjunto completo de 28 dominós.

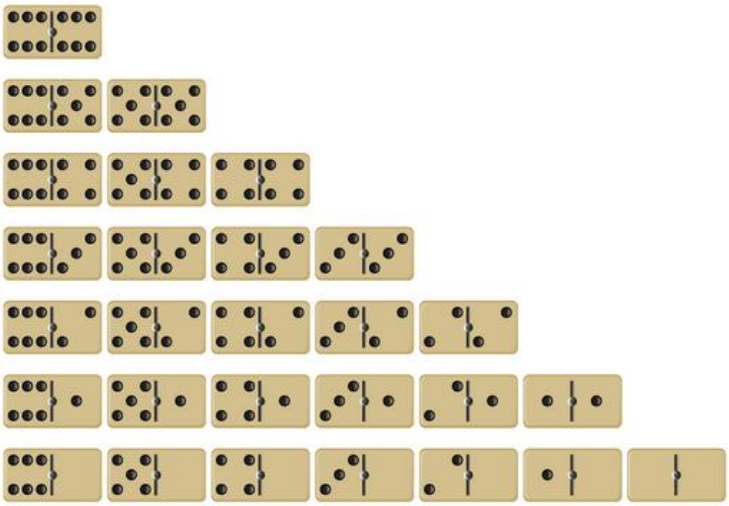


Figura 1 – Conjunto completo do dominó duplo-seis (28 peças).
FONTE – Retirado de:< http://howtoplaydominoes.org/tag/dominoes/>. Acesso em: 19 de novembro de 2014.

Existem algumas teorias sobre a origem do jogo de dominó. Imagina-se, porém, que tenha origem no oriente e que tenha sido inventado na China por um soldado chamado Hung Ming, (DUARTE, 2011). Os dominós são populares em um grande número de países. No Brasil, o jogo de dominó é bastante conhecido e a versão mais jogada é aquela com duas pontas e dois jogadores nas pontas. Para o agente proposto neste trabalho utiliza-se o jogo de dominó de 4 pontas e duas duplas, versão que é praticada no Estado do Amazonas.

Os jogos mais simples são de informação perfeita e de soma zero. Nesses jogos, os participantes tem conhecimento de todas as peças do tabuleiro e das possíveis jogadas de seus oponentes (informação perfeita). O resultado destes jogos pode ser um empate ou vitória de

um jogador e derrota do outro (soma zero). O jogo de damas ou de gamão, por exemplo, são exemplos de jogos de informação perfeita.

O dominó de 4 pontas é um jogo de informação incompleta, ou seja, um jogo onde os jogadores não têm informação das peças dos outros jogadores. Na sua forma mais usual é jogado por quatro jogadores divididos em duas duplas. No decorrer do jogo, através da observação das jogadas dos outros jogadores, um jogador mais experiente é capaz de conjecturar sobre as possíveis peças que os outros jogadores têm em mãos e sobre as jogadas mais prováveis que os mesmos irão executar.

A maioria dos esforços em inteligência artificial está concentrada em jogos determinísticos e de informação completa, de acordo com Whitehouse; Powley e Cowling (2011). Em um jogo determinístico não há elementos físicos envolvidos, o que torna as ações dos jogadores previsíveis. Os jogos mais conhecidos dessa categoria são o jogo de damas e o xadrez. Em contrapartida, os jogos não determinísticos são os que apresentam as imprevisibilidades do mundo real. Essa categoria de jogos possui um elemento aleatório, que pode ser proveniente de um lançamento de dados, como no gamão, ou o sorteio de cartas, como no pôquer.

Os trabalhos encontrados na literatura para a escolha da melhor jogada no jogo de dominó podem ser divididos em duas categorias: os que utilizam funções de avaliação estática como em, Garza (2006) e Antonio *et al*. (2013); e os que utilizam cálculos de probabilidade em torno das peças ocultas para um agente, onde se encontra apenas o trabalho de Cruz; Guimarães e Takahashi (2013). Entre os trabalhos citados apenas o de Antonio *et al*. (2013) foi voltado para o dominó de quatro pontas.

Os trabalhos publicados até então na literatura para o jogo de dominó, decidem sobre a melhor opção para a próxima jogada utilizando apenas informações do estado atual do jogo. Nesse trabalho, propõe-se a utilização de uma técnica para escolha da melhor opção para a

próxima jogada utilizando informações de como encontrar-se-á o estado do jogo várias jogadas à frente.

Para jogos determinísticos e de informação completa, como o xadrez, por exemplo, para escolha da melhor opção para a próxima jogada, em função de um estado futuro do jogo, o algoritmo mais utilizado é o algoritmo *minimax*, (VON NEUMANN e MORGENSTERN, 1944). Nesse algoritmo, para reduzir o número de caminhos avaliados na árvore de busca, a técnica de poda alfa-beta foi proposta por Hart e Edwards (1961).

Para jogos que incluem elementos de chance, como o dominó ou o gamão, o algoritmo *expectiminimax* (RUSSEL e NORVIG, 2004) é utilizado para o processo de busca. Este algoritmo é uma variação do algoritmo *minimax* para jogos não determinísticos com informações completa ou incompleta.

Neste trabalho, para escolha da melhor opção para a jogada seguinte no jogo de dominó de quatro pontas, será avaliado o potencial da técnica de busca em profundidade *expectiminimax*. Para o jogo de dominó, não se encontra na literatura a aplicação dessa técnica. Os trabalhos presentes são os de Garza (2006) e Cruz; Guimarães e Takahashi (2013), que realizam cálculos simples de probabilidade para a escolha da melhor jogada. Para a versão de quatro pontas, objeto dessa dissertação, encontra-se apenas o trabalho de Antonio *et al*. (2013). A aplicação mais comum do algoritmo *expectiminimax* é no jogo de Gamão. Nesse jogo, o desempenho de um jogador depende da combinação das habilidades do mesmo com elementos de chance, como a rolagem de dados.

Em adição aos nós de “min” e “max” da tradicional árvore *minimax*, o algoritmo *expectiminimax* inclui nós de chance, que assumem um valor numérico correspondente à probabilidade que um evento aleatório ocorra.

Para a adaptação do algoritmo *expectiminimax* ao jogo de dominó de quatro pontas, algumas hipóteses precisam preliminarmente ser feitas: a probabilidade de um evento ocorrer

não será um valor constante como no jogo de gamão. Este valor dependerá da quantidade de pedras na mão dos jogadores em qualquer instante do jogo; deve-se garantir que os jogadores da dupla que utilizam o algoritmo *expectiminimax* não conhecerão as pedras que estão na mão de seu parceiro e de seus adversários.

Na prática, segundo Smed e Hakonen (2006), as árvores de jogos são muito extensas para que se possa calcular a informação perfeitamente, dos nós folhas até a raiz. Nesse sentido, é necessário limitar a busca a uma árvore de jogo parcial. Isso significa parar a busca e salvar nós internos como se fossem folhas. Essa limitação no espaço de busca pode levar ao efeito de horizonte, situação em que um estado aparentemente promissor para o jogador virtual pode levar a estados indesejados no jogo. Com a finalidade de avaliar a ação do efeito de horizonte sobre o resultado do algoritmo *expectimimax*, utiliza-se uma abordagem para a limitação da profundidade chamada de antecipação *n-move*, onde *n* representa o número de camadas incluídas na busca. Utilizando-se este método, a profundidade da busca em antecipação *n-move* irá variar de acordo com as fases de jogo, que serão divididas em três: início, meio e fim.

Como expectativa em relação ao resultado desse estudo, espera-se que a dupla que utiliza o algoritmo *expectiminimax* para a escolha de suas jogadas obtenha um desempenho melhor comparado à dupla que utiliza a estratégia básica de jogo. Esta consiste na escolha de jogadas baseando-se apenas na pontuação da mesa. O desempenho do agente inteligente será avaliado em partidas contra duplas que utilizam a estratégia básica de jogo e contra duplas que utilizam o agente proposto por Antonio *et al*. (2013). No referido trabalho, foi criada uma função de avaliação do jogo de dominó de 4 pontas com coeficientes otimizados através de algoritmos genéticos.

1.1 OBJETIVO GERAL

O objetivo geral desta pesquisa é criar um agente inteligente, baseado no algoritmo de busca *expectiminimax*, para o jogo de dominó de quatro pontas. A busca deverá ser feita considerando-se todas as possibilidades de jogada em um dado instante do jogo e avaliando-se o efeito de diferentes níveis de profundidade do algoritmo de busca sobre o resultado do jogo.

1.2 OBJETIVO ESPECÍFICO

Os objetivos específicos consistem em:

- Definir oito estratégias cujas profundidades de busca serão baseadas na técnica *phase-related search* (busca por fases), em que as etapas do jogo serão dividias em: início, meio e fim;
- Explorar o efeito de diferentes níveis de profundidade do algoritmo de busca sobre o resultado do jogo;
- Criar a modelagem para eventos de chance no jogo de dominó de quatro pontas. Esse modelo probabilístico representa a possibilidade de um jogador ter pedras que se encaixem em qualquer uma das pontas que constituem o jogo de dominó de quatro pontas;
- Avaliar e comparar o desempenho das estratégias propostas, em partidas contra jogadores que utilizam a estratégia básica de jogo e contra a melhor estratégia obtida por Antonio *et al*. (2013).

1.3 ESTRUTURA DO TRABALHO

Este trabalho foi dividido em 6 Capítulos. O Capítulo 1 trata da caracterização do problema, o contexto histórico da área de estudo, a delimitação do trabalho e descreve o objetivo geral e os objetivos específicos da dissertação. No Capítulo 2 são apresentados alguns trabalhos relacionados à inteligência artificial com aplicação a jogos e que contribuíram para a riqueza deste trabalho, além de uma revisão bibliográfica de artigos referentes à busca heurística em jogos de dois jogadores de soma zero e jogos de informação imperfeita como o Pôquer. O Capítulo 3 aborda a fundamentação teórica, descrevendo o jogo de dominó de quatro pontas e os algoritmos de busca com aplicação a jogos determinísticos e não determinísticos.

O Capítulo 4 inclui o material utilizado e os procedimentos realizados no desenvolvimento desta pesquisa. O Capítulo 5 apresenta os resultados dos testes realizados com os comentários analíticos. O Capítulo 6 apresenta a conclusão acerca dos experimentos e indica trabalhos futuros.

2 REVISÃO BIBLIOGRÁFICA

De acordo com Watson (2013), a teoria dos jogos é uma metodologia que estuda formalmente situações de interdependência. Sobre o assunto, merece destaque o trabalho de Neumann e Morgenstern (1944), chamado “Teoria dos Jogos e Comportamento Econômico”, que apresenta o teorema *minimax* como solução para jogos de soma zero com dois jogadores, além de demonstrar também a teoria da utilidade.

O trabalho de Neumann e Morgenstern propôs de forma detalhada como representar jogos utilizando uma linguagem matemática precisa, e ofereceu um método geral de análise de comportamento. Esse método, porém apresentava limitações, podendo ser aplicado apenas a uma pequena classe de definições estratégicas.

Ao longo dos anos, outros pesquisadores deram sua contribuição, emprestando uma melhor fundamentação à teoria de jogos e preenchendo as “lacunas” da área: Flood (1952), Tucker (1950) e Nash, (1950). Com o advento dos computadores programáveis, os jogos foram estudados a fundo, estratégias de jogos foram criadas baseando-se em heurísticas, e, como consequência, máquinas derrotaram campeões humanos nos jogos de xadrez e gamão.

A maioria dos agentes inteligentes utilizados na área de jogos é baseada em mecanismos de busca. Um exemplo é o algoritmo *minimax*, (CAMPBELL; MARSLAND, 1983). Devido a problemas com o tempo gasto para gerar a árvore de busca, esse algoritmo é melhorado utilizando-se a poda alfa-beta, (KNUTH; MOORE, 1975). Novos algoritmos também foram propostos com o intuito de melhorar o processo de tomada de decisão na teoria de jogos. No entanto, são em sua maioria destinados a jogos de informação completa, como o xadrez e o jogo da velha (MORIARTY; MIIKKULAINEN, 1994), (HONG; HUANG; LIN, 1998), (SRIRAM *et al*., 2009) e (DÍEZ; LAFORGE; SAERENS, 2013).

Para jogos que utilizam estratégias baseadas em ambientes com informações incompletas, a maioria dos trabalhos é destinada a jogos de cartas como o bridge (XIAN *et al*., 2012) e pôquer (BILLINGS *et al*., 2004). Relacionados ao jogo de dominó de duas pontas, encontraram-se os trabalhos de Garza (2006) e Cruz; Guimarães e Takahashi (2013), e para a versão de quatro pontas, objeto dessa dissertação, encontrou-se apenas o trabalho de Antonio *et al*. (2013).

Para o desenvolvimento desta dissertação foram analisados os trabalhos relacionados às novas abordagens dos algoritmos *minimax* e *expectiminimax*, que revelam a preocupação dos pesquisadores em relação à complexidade da árvore dos jogos em estudo, bem como com o caráter estocástico dos mesmos.

Nas seções que seguem foram relacionados alguns trabalhos encontrados na literatura sobre melhorias nos algoritmos *minimax* e poda alfa-beta, com experimentos em jogos de informação completa e incompleta, além de aplicação de estratégias baseadas em heurísticas ao jogo de dominó.

2.1 PUBLICAÇÕES NA ÁREA DE INTELIGÊNCIA ARTIFICIAL USANDO ALGORITMOS *MINIMAX*, PODA ALFA-BETA E *EXPECTIMINIMAX* EM JOGOS DE INFORMAÇÃO COMPLETA E INCOMPLETA.

Para situações em que o adversário joga de forma não ótima, o algoritmo *minimax* apresenta falhas, tomando decisões não ótimas, conforme mostrado por Sriram *et al*. (2009). Para o jogo da velha, esses autores demonstraram que decisões não ótimas ocorreram, em média, em aproximadamente 40 casos. A partir deste fato, os autores propuseram um agente que atua de forma ótima, independentemente da estratégia adotado pelo adversário. O algoritmo desenvolvido foi baseado em heurísticas e árvore de decisão. Os autores elaboraram um experimento para comparar o desempenho de seu agente inteligente com o algoritmo *minimax* original, em partidas realizadas do jogo da velha. Para tanto, construiu-se a árvore de

busca manualmente para estados de jogos aleatórios. A partir de seus experimentos, pôde-se inferir sobre a eficiência nas decisões tomadas e ainda analisar a complexidade da árvore de jogo e no espaço de estados.

Ao final dos testes, comprovou-se que sua abordagem foi superior ao *minimax* convencional em termos de complexidade da árvore de jogo e em número de vitórias obtidas. Destaca-se, no entanto, que o conceito de árvore de decisão introduzido pelos autores poderia ser aplicado apenas ao jogo da velha, surgindo a necessidade de generalização dessa abordagem para outros jogos.

Em (HONG; HUANG; LIN, 1998) é aplicada uma nova abordagem baseada em algoritmos genéticos para a escolha da melhor jogada em uma árvore de busca em jogos. A estrutura dos cromossomos representa a codificação dos caminhos que vão do nó raiz até os nós folhas, variando de tamanho de acordo com a profundidade da busca.

Após as operações de cruzamento e mutação, os cromossomos de uma nova geração vão sendo inseridos em uma estrutura definida como árvore reserva, que será responsável pela comparação dos valores de aptidão de cada cromossomo para a escolha dos integrantes da próxima geração. O valor da função de aptidão do algoritmo genético de um dado nó representava o valor da propagação do valor de utilidade do algoritmo *minimax*, partindo de um nó folha até o nível em que o referido nó se encontrava.

Através de experimentos em sete níveis de busca, o agente inteligente baseado em algoritmos genéticos mostrou-se superior ao algoritmo *minimax* original, em termos de acurácia e tempo de execução, podendo ser aplicado a jogos de computadores reais.

Quanto à associação do algoritmo *minimax* com a poda alfa-beta, Knuth e Moore (1975) provam que a poda alfa-beta é mais efetiva quando o primeiro sucessor de cada nó é o melhor movimento, resultando em uma ordenação perfeita. A partir do exposto, Congpin e Jinling (2009) aplicaram esse princípio para verificar o espaço de busca que poderia ser

reduzido no jogo da velha. Para isso, os autores propuseram um método que faz com que os filhos dos nós que estão sendo expandidos sejam ordenados de acordo com seus valores estimados.

O desempenho dessa abordagem foi avaliado em termos do número de nós visitados em determinada profundidade, e posteriormente comparado com o algoritmo *minimax* associado à poda alfa-beta tradicional. O Quadro 1 ilustra os resultados obtidos em seu experimento. No Quadro 1, o método empregado pelos autores é denominado poda alfa-beta melhorado.

Quadro1 – Resultados obtidos por Congpin e Jinling para o jogo da velha com a abordagem de árvore perfeitamente ordenada (Congpin e Jinling, 2009).

Técnica de busca	Número de Nós (Profundidade da busca =2)	Número de Nós (Profundidade da busca=4)
Minimax	16	462
Poda α-β convencional	12	121
Poda α-β melhorada	7	50

A partir dos resultados obtidos, foi comprovado que para o jogo da velha, o algoritmo de poda alfa-beta melhorado é superior em relação ao número de nós percorridos, apesar de haver custo computacional extra na ordenação. Além disso, foi mostrado que, quanto mais profunda é a árvore de busca, mais nós serão cortados.

Em jogos não determinísticos, com a adição de nós de chance, a complexidade da árvore de jogo aumenta de forma exponencial. Com isso, surge o desafio de encontrar um método de poda que consiga manter o nível de acertos em seu processo de tomada de decisão. Schadd; Winands e Uiterwijk (2009) propuseram um método de poda para o algoritmo *expectiminimax* chamado *ChanceProbCut*. Esse método é baseado na correlação entre

avaliações obtidas em diferentes profundidades, realizando a poda em eventos de chance se o resultado de um nó de chance cair fora do intervalo de busca característico da poda alfa-beta.

Foram realizados experimentos em dois jogos: o *stratego* e o *dice*. Em ambos os experimentos o fator de redução de profundidade é definido através da técnica de regressão linear, sendo possível monitorar o percentual de efetividade do algoritmo de poda em função das reduções na árvore de jogo. Após os experimentos, os autores concluíram que o algoritmo *ChanceProbCut* era uma técnica eficaz na poda em jogos com o fator acaso, obtendo para o *stratego*, à profundidade de 11 níveis, um fator de redução de 31,3% e, para o *dice*, à profundidade de 13 níveis, uma redução de 57,9 %, sem perda de qualidade em suas jogadas.

Em (BILLINGS *et al.*, 2004) foi criado um agente inteligente chamado *Vexbot* para o jogo *Texas Hold'em*, uma versão do jogo de pôquer, utilizando uma árvore de busca para calcular o valor esperado em cada opção de jogada do oponente. Para aprimorar sua função de avaliação, realizaram a modelagem do oponente, registrando suas preferências e tendências, em tempo de execução do jogo. A modelagem do oponente foi fundamental para responder as imprevisibilidades do ambiente do jogo e das estratégias adotadas pelos adversários. O agente inteligente baseou-se no algoritmo de busca *expectiminimax*, com duas versões, chamadas de *Miximax* e *Miximix*. Para o jogo de pôquer, os nós que correspondem às ações do jogador e do oponente foram considerados dependentes, diferentemente do jogo de gamão.

O valor da avaliação de um nó foi utilizado para decidir que ação o jogador tomará: apostar/aumentar, checar/cobrir ou abandonar. Na versão *Miximax* o maior valor para as três ações é escolhido. A árvore de jogo consiste de nós misturados e de nós max, representando as ações do oponente e do jogador, respectivamente. Na versão *Miximix* os nós de ambos os jogadores encontram-se misturados, tornando-se um método de escolha randômica, a fim de evitar previsibilidade nas jogadas. Nos nós folhas que representam o *Showdown* (descida das cartas) *ou Fold* (abandono), foi utilizada a função de avaliação descrita pela Equação (1).

$$EV(L) = \left(P_{win} X\ L_{\$pot}\right) - L_{\$cost} \quad (1)$$

Em que:

EV(L) - Avaliação do nó folha;

P_{win} - Probabilidade de vitória;

$L_{\$pot}$ - Tamanho do pote de apostas;

$L_{\$cost}$- Custo do pote para o jogador até o nó folha.

Para o estado terminal de jogo em um *Fold*, o valor de P_{win} será 0 ou 1, e para um *Showdown* este valor é estimado através da modelagem do oponente. Para o seu agente foi criada uma função de densidade probabilística da força de sua mão em situações passadas. Nos experimentos foram realizadas partidas que consistiam em aproximadamente 40000 mãos de pôquer contra seis agentes inteligentes, entre eles o Sparbot e o Poki, desenvolvido pelos autores em trabalhos anteriores. Os resultados obtidos foram estatisticamente significantes com desvio padrão de aproximadamente ± 0.3 pequenas apostas por mão. Porém, o programa proposto ainda necessita de melhorias, definindo padrões a fim de diminuir o número de derrotas na fase de aprendizagem, que corresponde aos 100 primeiros jogos. Nesta etapa o agente realiza a modelagem do oponente, além da necessidade de generalização para versões com múltiplos jogadores.

2.2 PUBLICAÇÕES NA ÁREA DE JOGOS RELACIONADAS AO JOGO DE DOMINÓ

Em (GARZA, 2006), foi criado um agente inteligente para o jogo de dominó de duas pontas, comum em países de origem espanhola e que possui regras semelhantes as do jogo de dominó praticado em muitas regiões do Brasil.

Em seu trabalho, foi desenvolvido um sistema feito em Java denominado *Domage*. O objetivo do agente inteligente é determinar a melhor jogada. Os agentes de cada jogador

funcionam de forma independente. A única informação que possuem a respeito dos outros jogadores é a quantidade de pedras em suas mãos em qualquer instante do jogo.

A partir do exposto, por se tratar de um jogo de informação imperfeita, suposições e inferências são feitas no processo de tomada de decisão, tendo como base o tipo de pedra mais jogada pelos outros jogadores. Nesse trabalho foram implementadas e avaliadas as seguintes estratégias de jogo: Tradicional (T); Altruísta (A); Egoísta (S); Aleatória (R); Suicida (SU); Tradicional e Altruísta (TA) e Tradicional e Egoísta (TS). A principal diferença entre essas estratégias está na priorização dos objetivos de jogo no momento de decidir uma jogada. As duas últimas correspondem a uma combinação de algumas estratégias anteriores. O autor permitiu que as estratégias possuíssem mais de um objetivo de jogo, dentre os seguintes: jogar apenas com o objetivo de pontuação, realizar jogadas para benefício próprio, beneficiar o parceiro de jogo e atrapalhar as jogadas da dupla adversária.

Em cada estratégia o jogador define os seus objetivos de jogo, dentre os citados anteriormente. Como exemplo, pode-se citar a estratégia tradicional: a primeira prioridade para um jogador *P* que utiliza esta estratégia é beneficiar sua equipe. Se *P* é o líder do time ele decide uma jogada benéfica para ele. Se isso não é possível, o jogador *P* fará uma jogada benéfica ao seu parceiro, se possível. Caso *P* não seja o líder de seu time, então as prioridades relativas dessas duas opções são invertidas. Se nenhuma das opções é possível com as pedras jogáveis de *P*, então ele tenta bloquear as jogadas da dupla adversária. Caso nenhuma das ações mencionadas seja possível, *P* jogará a pedra mais alta em sua mão a fim de minimizar a pontuação de garagem em uma eventual perda.

Nos experimentos realizados pelo autor, um jogador *P* variou a estratégia de jogo, enquanto os outros três jogadores utilizavam a estratégia tradicional. A cada mudança na estratégia adotada por *P* eram realizadas 100 partidas. A maior quantidade de vitórias obtidas

pelo time de *P* foi 54% para a estratégia tradicional e egoísta. Garza não atingiu seu objetivo que era de 55% de vitórias para algumas das estratégias definidas.

Em (CRUZ; GUIMARÃES; TAKAHASHI, 2013), foram avaliadas quatro estratégias para o jogo de dominó de 2 pontas, comumente jogado no estado de Minas Gerais. Cada par de agentes foi testado em um milhão de partidas. A primeira estratégia consiste em selecionar a jogada apropriada, entre as disponíveis, de forma randômica. Na segunda estratégia, o agente avalia as pedras que estão na mesa e em suas mãos, e seleciona uma jogada através de um cálculo simples de probabilidade. O jogador seleciona um subconjunto *C* de possíveis pedras que poderá jogar, em qualquer posição da mesa, e utiliza o lado oposto *i* dessas possíveis pedras para calcular a probabilidade do oponente não possuir pedras com essa numeração. A Equação (2) mostra o cálculo de probabilidade utilizado pelo segundo agente.

$$p(i) = \frac{v(i)/6}{\sum_{k \in C}^{n} v(k)/6} = \frac{v(i)}{\sum_{k \in C}^{n} v(k)} \quad (2)$$

Em que:

n – Número de pedras que podem ser movidas no subconjunto *C*;

k – Cada uma das pedras de *C*;

$v(i)$ – Número de pedras com a ponta *i* que o jogador observa na mesa e em sua mão.

O jogador irá selecionar a pedra a ser jogada, através do cálculo de $p(i)$, escolhendo o maior valor, a fim de bloquear as jogadas do oponente. A terceira estratégia também avalia as pedras que estão na mesa e nas mãos do jogador. Entretanto, o cálculo de probabilidade para esta estratégia, Equação (3), incluirá o valor *v(i)*, que representa uma futura nova ponta criada a partir da pedra que será jogada e *t(i)*, representando a ponta remanescente depois que a jogada for realizada.

$$p(i) = \frac{v(i)t(i)/6}{\sum_{k\in C}^{n} v(k)t(k)/6} = \frac{v(i)t(i)}{\sum_{k\in C}^{n} v(k)t(k)} \quad (3)$$

Da mesma forma que a estratégia 2, o jogador escolherá a pedra com maior valor de $p(i)$, a fim de bloquear as jogadas do oponente, segundo os autores, de uma forma mais geral. A última estratégia utiliza a análise das duas possíveis pontas da mesa assim como a estratégia 3, e também utiliza a exploração de Boltzmann para determinar a probabilidade de selecionar uma pedra do subconjunto *C*. A Equação (4) apresenta o cálculo da probabilidade para cada umas das pedras do subconjunto *C*.

$$p(i) = \frac{e^{v(i)t(i)/u}}{\sum_{k\in C}^{n} e^{v(k)t(k)/u}} \quad (4)$$

Na Equação (4), u representa a temperatura que é função do número de pedras na mesa, m em cada instante. A temperatura é calculada por $u = 0.999^{m}$. O agente que utiliza essa estratégia irá ordenar as pedras possíveis de acordo com os valores de $p(i)$ e utilizará uma roleta, (GOLDBERG, 1989), para escolher a jogada. Nos experimentos, o melhor resultado foi obtido pela estratégia 4, vencendo em média 65% das partidas contra a estratégia 1.

Os trabalhos realizados até então para o jogo de dominó não empregaram a técnica de antecipação em mais de um nível. Essa foi a principal motivação desta pesquisa. Os trabalhos analisados, no entanto, provaram que qualquer estratégia adotada, mesmo que randômica, será capaz de obter um considerável número de vitórias. Isso se deve ao fato do ambiente de jogo ser estocástico e não ser possível determinar um agente que irá sempre ganhar.

Para o jogo de dominó de 4 pontas, Antonio *et al.* (2013) propuseram um método para otimização de uma função de avaliação do jogo de dominó de 4 pontas através de algoritmos genéticos. O jogo de dominó de 4 pontas, jogado no norte do Brasil, é normalmente jogado

com 4 jogadores, com duas duplas de 2 jogadores. Diferentemente do jogo de dominó de duas pontas, onde o objetivo do jogo é um jogador descartar todas as peças antes dos outros, no jogo de dominó de 4 pontas o objetivo do jogo é atingir uma maior pontuação.

A função de avaliação proposta pelos autores foi construída utilizando-se dois termos, conforme descrito na Equação (5):

$$f(n) = T_{n1} + T_{n2} \tag{5}$$

O termo T_{n1} corresponde aos pontos obtidos ao se fazer uma jogada, enquanto que o termo T_{n2} incorpora a estratégia do jogo. A estratégia do jogo incorpora informações que visam facilitar jogadas futuras do próprio jogador ou do companheiro de dupla, ou complicar as jogadas dos oponentes. O termo T_{n2} é formado por duas parcelas: E_1 e E_2, que embutem uma lógica para atrapalhar o jogo dos adversários e parar facilitar o jogo do parceiro da dupla, respectivamente. A Seção 4 irá mostrar com detalhes os termos dessa função de avaliação.

As parcelas E_1 e E_2 possuem termos multiplicados por sete coeficientes: α_1, α_2, α_3, α_4, α_5, α_6 e α_7, cujos valores iniciais foram definidos no intervalo de -10 a 10. Esses coeficientes definem a importância de um termo em relação aos demais, de acordo com as seguintes estratégias: a estratégia 1 inclui 3 possíveis ações que um jogador pode adotar na escolha de uma jogada: facilitar o próprio jogo, dificultar as jogadas dos adversários e facilitar o jogo de seu parceiro; a segunda estratégia, é equivalente a uma estratégia egoísta onde a jogada tem o objetivo de beneficiar o próprio jogador; a terceira estratégia tem como finalidade facilitar as jogadas do parceiro da dupla; a quarta estratégia prioriza dificultar a jogada dos adversários; a última estratégia representa um jogador iniciante, que escolhe as suas jogadas baseando-se apenas na pontuação da mesa.

Com o objetivo de obter o maior número de vitórias em partidas realizadas, os coeficientes da função de avaliação, determinada pela estratégia, foram otimizados pela

técnica de algoritmos genéticos. Em seu experimento os autores realizaram partidas entre duplas que utilizavam as 4 primeiras estratégias definidas anteriormente e duplas que utilizavam a estratégia básica de jogo. Para cada uma das funções de avaliação dessas estratégias, os cromossomos foram codificados pelos sete coeficientes. Para cada estratégia foram realizadas combinações dos parâmetros mostrados na Figura 2, resultando em 256 configurações do algoritmo genético.

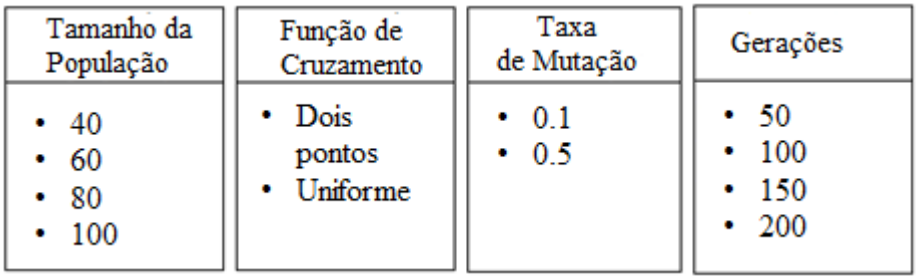


Figura 2 - Parâmetros utilizados para a otimização do algoritmo genético (ANTONIO *et al.*, 2013).

Cada configuração foi avaliada através de 5000 simulações de partidas. Nesses experimentos, o melhor resultado obtido foi 70,06% de vitórias para a estratégia 1. A função de avaliação para a estratégia 1, otimizada através do procedimento anteriormente descrito, foi utilizada para jogos contra jogadores humanos, com e sem experiência no jogo de dominó de 4 pontas. Em jogos com jogadores inexperientes, percentuais de vitória de até 62% foram obtidos. Em jogos com jogadores experientes, percentuais de vitória de 32% foram obtidos.

3 FUNDAMENTOS TEÓRICOS

Neste Capítulo será apresentada a fundamentação teórica que serviu de base para o desenvolvimento do agente que utiliza a busca em profundidade. Na primeira parte serão mostradas particularidades sobre a versão do jogo de dominó de quatro pontas, regras do jogo, e principais estratégias utilizadas pelos jogadores iniciantes e experientes. Na Seção seguinte, será abordado o algoritmo *minimax* e suas principais aplicações na área de jogos. A parte final deste trabalho está destinada ao método de busca *expectiminimax*, que será usado nesta pesquisa. Sobre o mesmo, será apresentado o contexto histórico, a estrutura da árvore de jogo, e serão discutidas algumas aplicações do algoritmo no processo de tomada de decisão em jogos não determinísticos.

3.1 O JOGO DE DOMINÓ

De acordo com o Capítulo 1 existem algumas versões para o jogo de dominó. Além da versão duplo 6, pode-se citar também a versão duplo 2, onde a peça com maior numeração nas duas metades é a peça com metades 2-2, ou peça 2-2. Em um conjunto, as peças que possuem uma numeração igual nas duas metades são ditas peças com apenas um *nipe*, ou carroças. Na versão duplo 6 tem-se as seguintes carroças: 1-1, 2-2, 3-3, 4-4, 5-5 e 6-6. Peças com numerações diferentes nas duas metades são ditas peças com dois *nipes*. Na versão duplo 6, por exemplo, as seguintes peças são de dois *nipes*: 1-0,1-4,1-6. Em cada versão, o número de peças *P* distribuídas inicialmente para os jogadores é dada pela Equação (6):

$$\mathrm{P} = \frac{n(n+1)}{2} \tag{6}$$

Em que:

$n - maior\ numeração\ + 1$.

Na versão duplo 6, $n = 6 + 1 = 7$ e $P = 28$. Na versão duplo 2, $n = 2 + 1 = 3$ e $P = 6$. Nas versões em que $P < 28$, as peças que não são distribuídas aos jogadores permanecem no "cemitério". Por definição, o cemitério é constituído por peças que sobram depois da distribuição e que podem ser resgatadas quando um jogador não possui pedras que encaixem nas pontas.

Para todas as versões do jogo de dominó, o número de combinações possíveis de peças que podem existir nas mãos dos jogadores é também conhecido como "número de mãos" ou "número de estados iniciais", *M*. Supondo que existam *L* peças no jogo, o número de mão é dada pela Equação (7). O Quadro 2 mostra o valor de M para algumas versões do jogo de dominó.

$$M = \left(\frac{L}{B}\right) X \left(\frac{L - B}{D}\right) \tag{7}$$

Em que:

$B -$ Número de peças do cemitério;
$D -$ Número de peças na mão de cada jogador.

Quadro 2 – Estados iniciais, *M*, para algumas versões do jogo de dominó (MYERS, 2014).

Versão	Peças em jogo	Jogadores	Distribuição	Peças do cemitério	Estados iniciais
Duplo 2	6	2	2-2	2	20
Duplo 3	10	2	3-3	4	4,200
Duplo 4	15	2	5-5	5	576,576
Duplo 5	21	2	7-7	7	271, 591,320
Duplo 6	28	4	7-7-7-7	0	$1{,}18 * 10^6$

No Brasil, a modalidade de dominó mais praticada é a versão duplo 6 com duas pontas, podendo ser jogado individualmente ou em duas duplas. O objetivo dessa versão de dominó é simplesmente alcançar primeiro a batida do jogo. Na região norte do Brasil, pratica-se com mais frequência a versão duplo 6 com 4 pontas, que possui objetivos e regras bem diferentes da versão duplo 6 de duas pontas.

Na versão duplo 6 de quatro pontas, antes do início de uma partida, o conjunto de peças é “embaralhado” sobre a mesa. Em seguida as pedras são distribuídas entre as duas duplas conforme a distribuição mostrada no Quadro 2. A primeira jogada de uma partida é iniciada pelo jogador que possuir a pedra 6-6, conhecida como “carroça de sena”. As jogadas são efetuadas no sentido horário, alternadas entre um jogador e um de seus oponentes (haverá mais de um oponente no caso de o jogo ser jogado por 2 duplas). Uma partida é constituída de várias rodadas. Uma rodada chega ao fim quando um dos jogadores não possui mais pedras em sua mão, ou quando o jogo fica “trancado”, situação em que nenhum dos jogadores possui mais pedras que possam ser encaixadas em qualquer uma das 4 pontas do jogo. Qualquer rodada, que não seja a primeira, pode ser iniciada com qualquer carroça, pelo jogador que finalizou a rodada anterior.

O objetivo do jogo de dominó é somar 200 ou mais pontos. O número de rodadas necessárias para atingir o objetivo vai depender de cada jogo. Se ao fim de uma rodada, uma dupla conseguir 200 ou mais pontos, ela ganha a partida. Conforme descrito a seguir, existem cinco formas de pontuar no jogo de dominó versão duplo 6 com 4 pontas. Cada uma dessas formas é denominada de $P_i, 1 < i < 5$:

- P_1: O resultado da soma das 4 extremidades das peças encaixadas na mesa é múltiplo de 5. Em uma jogada um jogador pode obter 5, 10, 15, 20 ou 25 pontos. A Figura 3 ilustra uma situação onde são obtidos 10 pontos;

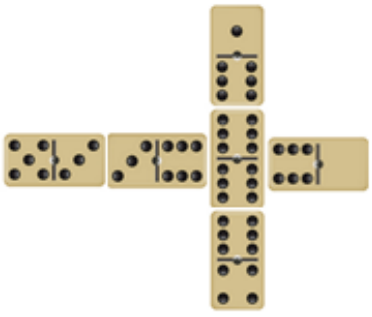


Figura 3 – Situação de jogo que gera pontuação 10.

- P_2: Um jogador não possui pedras que se encaixem nas pontas livres do jogo, a pontuação de 20 pontos é contabilizada para o adversário ou para a dupla adversária (no caso de ser jogado por duas duplas). A Figura 4 ilustra um estado de jogo onde o jogador da vez "passa" em sua vez;

Figura 4 – Situação de passe de um jogador.

- P_3: Quando um jogador realiza uma jogada e anuncia previamente que todos os outros jogadores irão passar na jogada. É anunciado "galo". A pontuação obtida é 50 pontos;

- P_4: No fim de uma rodada, quando o jogador finaliza um jogo, "batida", com uma carroça. A sua dupla receberá 20 pontos;

- P_5: Com o encerramento da rodada, a soma das pedras dos adversários é arredondada para o múltiplo de 5 abaixo e mais próximo de seu valor. Essa pontuação é contabilizada para a dupla a qual o jogador que bateu pertence. A Figura 5 ilustra um exemplo de garagem, em que a soma das pedras é 14, gerando 10 pontos.

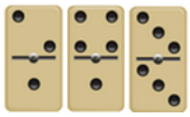


Figura 5 – "Garagem" exemplo de 10 pontos.

As estratégias adotadas pelos jogadores de dominó podem ser divididas em 3 categorias. Somente jogadores experientes são capazes de guardar informações cruciais ao longo do jogo

e, assim, executarem as estratégias mais complexas. As ações referentes às estratégias são listadas a seguir, em ordem crescente de dificuldade:

- **Jogar de forma "egoísta"**: Quando o jogador estiver em uma posição favorável no jogo, ou seja, possuir pedras altas e diversas, deve-se jogar pedras que somem 10, como 5-5 ou peças altas, como 4-6. Jogar peças altas favorece diminuir a contagem de pontos P_5, em caso da dupla adversária ganhe. Caso não seja possível, jogam-se carroças para manter o controle sobre a disposição das pontas. Em uma situação de desvantagem do jogador, ou seja, quando o mesmo possuir peças que não pontuem ou com baixo valor, deve-se tentar manter as pontas com *nipes* que permitam mais jogadas futuramente como, por exemplo, se as pontas da mesa são 2|6 e as pedras disponíveis 2-1, 3-5 e 6|3, a melhor escolha será 6|3, pois facilitará a jogada futura da peça 3-5;

- **Jogar em prol do parceiro**: O jogador deverá estar atento às pedras jogadas pelo parceiro, e através de estimativas, descartar as pedras cujos *nipes* o seu parceiro possa ter;

- **Bloquear a dupla adversária**: O jogador deverá estar atento às pedras jogadas pelo adversário seguinte, e tentar bloqueá-lo presumindo as peças que o mesmo não tem. A maneira de presumir quais *nipes* o adversário não possui é observar os seus passes durante uma rodada. A partir desta informação, o jogador deverá descartar na mesa pedras cujos *nipes* o seu adversário presumivelmente não possui. Deve-se também forçar o oponente a descartar suas peças mais baixas, jogando peças com valor 0 ou 1. Desta forma, em caso de uma batida favorável, aumenta-se a possibilidade de pontuação através da contagem de pontos P_5.

3.2 BUSCA EM JOGOS

Um jogo pode ser representado por uma "árvore", onde a posição atual é o nó raiz e as jogadas possíveis de um jogador e seus oponentes são as ramificações. As diversas

possibilidades de término do jogo, onde não há mais ramificações, são representadas por nós chamados terminais ou folhas. Geralmente, a estes nós é associado um valor que corresponde ao resultado do jogo, (por exemplo, vitória, derrota ou empate). No algoritmo *minimax* e suas variantes, valores prospectados para o término de um jogo são propagados pelo caminho da árvore de busca utilizada até chegar aos mesmos, para escolha da melhor jogada. Quando a prospecção não consegue atingir um nó folha, devido a complexidade no tempo envolvido na busca, uma heurística atribui um valor ao nó onde a busca foi interrompida. A Figura 6 mostra a estrutura básica de uma árvore de jogo genérica.

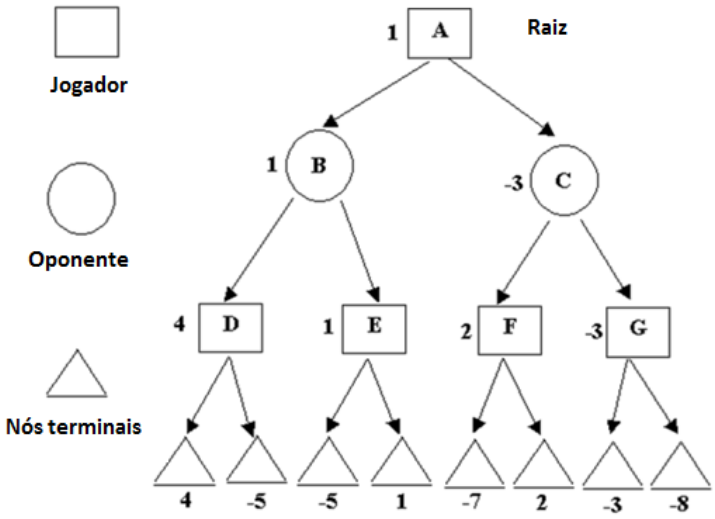


Figura 6 – Estrutura básica de uma árvore de jogo genérica.

Na Figura 6, as letras de A à G indicam os estados de jogo produzido pelas ações do jogadores. Neste exemplo o símbolo quadrado indica o jogador que utiliza a árvore de busca para determinar a melhor jogada. O símbolo círculo indica o seu oponente. Os números que acompanham esses nós representam a avaliação do estado de jogo do ponto de vista do primeiro jogador citado. O mesmo acontece para os nós representados por triângulos, chamados nós terminais. A próxima sessão irá apresentar o funcionamento da técnica de busca *minimax.*

3.3 ALGORITMO *MINIMAX*

Na teoria de decisão, o *minimax* é um método empregado para minimizar a perda máxima possível. Na área de jogos, para a criação de agentes inteligentes, muitos autores utilizam este algoritmo. No processo de busca, o primeiro jogador, chamado jogador *MAX*, tentará maximizar a sua pontuação, enquanto o segundo jogador, conhecido como jogador *MIN*, tenta minimizar a pontuação do oponente. Na prospecção de estados futuros do jogo, o algoritmo alternará recursivamente entre o jogador *MAX* e o jogador *MIN*, até atingir um nó terminal (ARTS, 2010).

A árvore de jogo é percorrida em uma busca em profundidade, que se constitui de uma expansão contínua explorando apenas uma ramificação, ou opção de jogada, em cada nível de profundidade, até que um nó terminal seja atingido, e depois disso os valores *minimax* são propagados de volta pela árvore à medida que a recursão retorna. Quando, a partir de um nó inicial, a busca visita todas possíveis ramificações, o algoritmo *minimax* pode, então, escolher a melhor jogada possível a partir daquele nó, avaliando o potencial de cada ramificação.

Segundo Russel e Norvig (2004), se a profundidade máxima da árvore é *m* e existem *b* movimentos válidos em cada ponto, a complexidade de tempo do algoritmo é O (b^m). A complexidade de espaço é O (*bm*) para um algoritmo que gera todos os sucessores de uma vez ou O(*m*) para um algoritmo que gera um sucessor de cada vez. A partir dessa constatação, há um problema na busca *minimax*, pois o número de estados de jogo que ela tem que examinar é exponencial em relação ao número de movimentos. Entre os estados de jogo examinados estão presentes alguns em que, dada a melhor jogada, não serão escolhidos pelo agente. Para eliminar esses nós da árvore de jogo e calcular a decisão ótima, os autores sugerem o algoritmo de "poda alfa-beta", que é capaz de reduzir o expoente de O (b^m) pela metade.

Outra maneira de reduzir o número de nós visitados é através de uma busca limitada *(depth-limited search)*, (ARTS, 2010). Nessa metodologia, ao invés de terminar a busca em

um nó terminal, para-se em determinado nível da árvore. Com isso, o valor a ser propagado para o nó raiz não será mais o equivalente a um valor de vitória, derrota ou empate e sim o resultado de uma função de avaliação para o estado do nó onde a busca foi interrompida. Um exemplo de árvore com busca *minimax* limitada em profundidade pode ser visto na Figura 7.

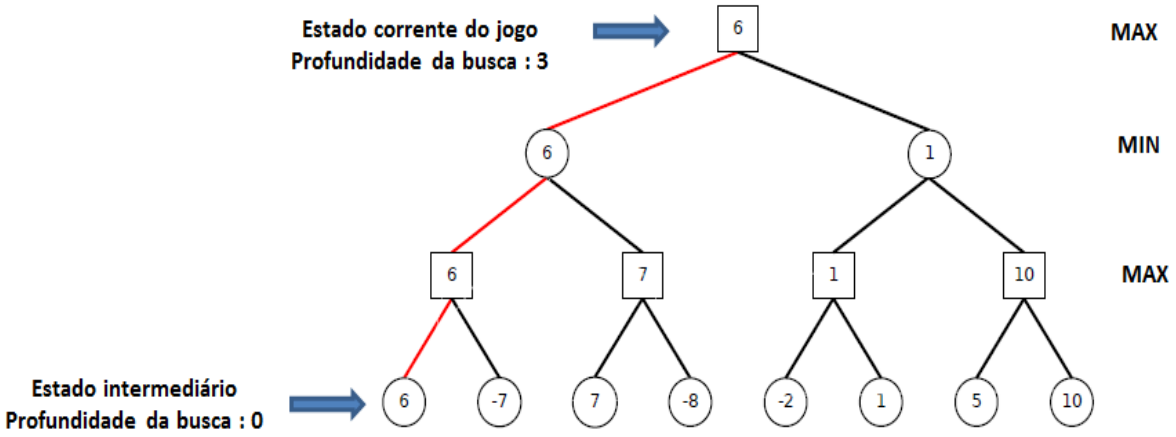


Figura 7 – Árvore de jogo em uma busca *minimax* limitada em profundidade.

Através da Figura 7, será mostrada a operação do algoritmo *minimax*. O primeiro nó da árvore representa o nó raiz e os últimos nós representam os nós folhas. O jogador que utiliza o algoritmo de busca, *MAX*, e o jogador oponente, *MIN* irão explorar, alternadamente, o ramo mais à esquerda em uma busca em profundidade. O jogador *MAX* encontra um nó terminal com valor 6 (o nó mais à esquerda da árvore de busca). Em seguida, é efetuada a exploração do segundo ramo, onde é encontrado o valor terminal -7, ao lado de 6. O algoritmo *minimax* escolherá a ramificação com maior valor, *MÁXIMO* (6,-7) = 6. O valor 6 então, é retornado para *MIN* que irá explorar o seu segundo ramo. Novamente, *MAX* escolherá o maior valor entre os dois nós terminais, *MÁXIMO* (7,-8) = 7. O jogador *MIN*, escolherá o nó com menor valor *MÍNIMO* (7,6) = 6. Esse valor é atribuído ao nó *MIN*. O processo descrito se repetirá até que todos os filhos do nó raiz sejam explorados. Nesse caso, a melhor ramificação de jogada a partir do nó raiz é aquela que resulta no nível seguinte no nó MIN com valor 6. O

pseudocódigo do algoritmo *MINIMAX*, (RUSSEL e NORVIG, 2004), pode ser visto no Algoritmo 1.

Algoritmo 1 *MINIMAX*

função DECISÃO-*MINIMAX* (estado) retorna uma ação
entradas: estado, estado corrente no jogo
υ ← VALOR-*MAX* (estado)
retornar a ação em SUCESSORES(estado) com valor v

função VALOR-*MAX* (estado) **retorna** um valor de utilidade
Se TESTE-TERMINAL(estado) **então retornar** UTILIDADE(estado)
υ ←- ∞
para a, s em SUCESSORES(estado) **faça**
υ ←*MAX* (v, VALOR-*MIN*(s))
retornar υ

função VALOR-*MIN*(estado) **retorna** um valor de utilidade
Se TESTE-TERMINAL(estado) **então retornar** UTILIDADE(estado)
υ ← ∞
para a, s em SUCESSORES(estado) **faça**
υ ←*MIN* (v, VALOR-*MAX*(s))
retornar υ

De acordo com o Algoritmo 1, a função DECISÃO-*MINIMAX* retorna a ação que corresponde ao melhor movimento para o jogador *MAX*, ou seja, o movimento que leva ao melhor estado terminal de jogo, supondo que o oponente joga de forma ótima. As funções VALOR-*MAX* e VALOR-*MIN* percorrem toda a árvore de jogo até os estados terminais a fim de propagar os valores de cada estado.

3.4 *MINIMAX* PARCIAL

Na prática, segundo Smed e Hakonen (2006), as árvores de jogos são grandes demais para permitirem a prospecção da melhor jogada utilizando todos os níveis da árvore de busca, da raiz até as folhas. Logo, tem-se que limitar a busca a uma árvore de jogo parcial, parando a busca em um nível intermediário *n*, e considerando os nós desse nível, *frontier nodes* (nós de fronteira), como se fossem nós folhas. Esta abordagem de limitar a profundidade da busca é chamada de estratégia *n-move look-ahead* (antecipação de *n* movimentos à frente), onde *n* é o número de níveis incluídos na busca.

Nessa abordagem não se utilizam os valores terminais de um jogo no final da busca, como, por exemplo, 1, 0, -1, para vitória, empate e derrota, respectivamente. Os nós de fronteiras devem ser avaliados através de uma função de avaliação heurística. Os valores dessa função, então, são propagados árvore acima segundo o princípio *minimax*. Infelizmente, existem casos onde, devido a problemas do efeito de horizonte, que serão explicados na Seção seguinte, os nós de fronteira contribuem para seleção de jogadas não ótimas.

3.5 CONTROLANDO A PROFUNDIDADE DA BUSCA

A profundidade utilizada na técnica de antecipação de *n* movimentos à frente não necessita ser a mesma para cada nó, podendo variar de acordo com a fase do jogo ou o fator de ramificação. Uma cadeia de sub-árvores relevantes e estreitas é mais fácil de percorrer até um nível mais profundo. Além disso, jogos podem ser divididos em fases (por exemplo, início, meio e fim) que relacionam o número de peças e suas posições no tabuleiro. As estratégias adotadas em cada fase podem diferir um pouco, e o método de busca deve adaptar-se a essas diferenças.

3.6 EXPECTIMINIMAX

Segundo Russel e Norvig (2004), na vida real, existem muitos eventos externos imprevisíveis, que nos colocam em situações inesperadas. Muitos jogos refletem essa imprevisibilidade incluindo um elemento aleatório, como o lançamento de dados. Para a análise desses jogos a metodologia *EXPECTIMINIMAX* deve ser utilizada. A árvore de jogo deverá incluir 3 tipos de nós: *max, min,* presentes também no *minimax*, e nós de chance, que tomam o valor esperado da ocorrência dos eventos aleatórios.

Um exemplo de aplicação desse algoritmo é no jogo de gamão, jogo não determinístico de informação completa, onde os dados são rolados no início da ação de cada jogador, para determinar o deslocamento de peças a ser executado. Um resumo do algoritmo com a inclusão do evento de chance pode ser visto no Algoritmo 2 proposto por Michie (1966).

Algorimo 2 *EXPECTIMINIMAX*

Função *expectiminimax (nó, profundidade)*

```
SE o nó é terminal ou a profundidade = 0
    retorne o valor heurístico do nó
SE o nó corrente representa o adversário (nó de min)
    α ← + ∞
    PARA CADA filho do nó
        α ← + min(α, expectiminimax(filho, profundidade-1))
SENÃO SE o nó corrente representa nossa jogada (nó de max)
    α ← - ∞
    PARA CADA filho do nó
        α ← + max(α, expectiminimax(filho, profundidade-1))
SENÃO SE o nó corrente representa evento de chance
    α ← 0
    PARA CADA filho do nó
        α ← α + (Probabilidade [filho] * expectiminimax (filho, profundidade-1))

retorne α
```

De acordo com o algoritmo 2, enquanto o jogador MAX escolhe o possível maior valor e o jogador MIN escolhe o possível menor valor em função dos seus objetivos no jogo,

nos nós de chance há um evento aleatório, onde o valor esperado (média ponderada) é calculado. A variável Probabilidade [filho] denota a probabilidade do nó filho (nó de MAX ou MIN) ser escolhido como sucessor do nó pai (nó de chance) atual. De acordo com Russel e Norvig (2004), para um nó de chance *n*, o seu valor *expectiminimax* é calculado de acordo com a Equação (8):

$$EXPECTIMINIMAX(n) = \sum_{s \in \text{Sucessores}(n)} P(s) \cdot EXPECTIMINIMAX(s) \qquad (8)$$

Na Equação (8), *P(s)* representa a probabilidade de cada sucessor *s*. O valor *expectiminimax* de cada nó da árvore de jogo é propagado de volta recursivamente por todo o caminho, até a raiz da árvore, da mesma maneira que no algoritmo *minimax*. A partir do conceito visto, pode-se afirmar que a estratégia do jogador MAX consiste em maximizar o ganho esperado. Enquanto que a estratégia do jogador MIN consiste em minimizar esse ganho. A Figura 8 ilustra um exemplo da aplicação do algoritmo *expectiminimax* em jogos não determinísticos.

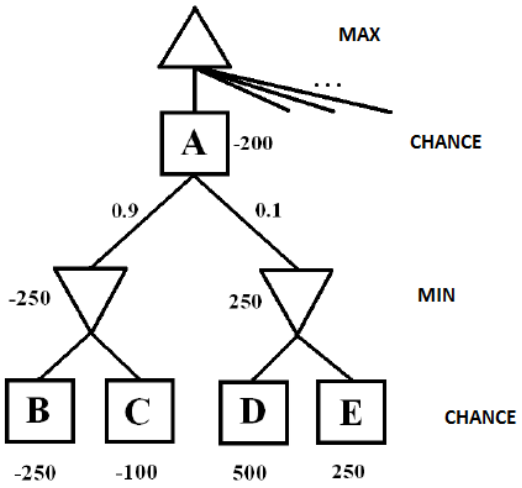


Figura 8– Exemplo de árvore para jogos não determinísticos.

Na Figura 8 o nó A corresponde a um nó de chance com dois eventos possíveis. O valor desse nó é obtido através do cálculo de pesos para cada um desses eventos de chance.

Esses pesos correspondem, via de regra, à probabilidade de ocorrência de cada uma das ramificações do jogo. No caso mostrado, esses pesos são 0,9 e 0,1. O valor *expectiminimax* para o nó A é dado então por: *Expectiminimax*(A) = $0.9 \times -250 + 0.1 \times 250 = -200$.

3.7 COMPLEXIDADE DO JOGO DE DOMINÓ DE 4 PONTAS

Existem alguns fatores que aumentam a dificuldade para a criação de um agente para o jogo de dominó de 4 pontas:

- O fator de ramificação. Quanto maior for o seu valor, maior será o tempo necessário para avaliação da melhor jogada;
- O número de estados iniciais do jogo. Ao contrário do jogo da velha e do xadrez, que são jogos de informação completa, o dominó de 4 pontas possui um enorme número de estados iniciais possíveis;
- O fato de ser um jogo não determinístico e de informação imperfeita, onde inferências devem ser feitas no processo de decisão, em função de valores probabilísticos para o cálculo dos quais não existe nenhuma modelagem anterior.

A seguir será mostrado algumas medidas de complexidade computacional do jogo de dominó de 4 pontas e comparar os resultados obtidos com alguns jogos existentes. Na Seção 3.7.1 será analisada a complexidade da árvore de jogo e na Seção 3.7.2 será analisada a complexidade do espaço de estados.

3.7.1 Complexidade da árvore de jogo

A complexidade de uma árvore de jogo é determinada pelo número de possibilidades de jogadas que podem ser realizados. Esse número é refletido na quantidade de nós folhas da árvore de jogo. Para o seu cálculo, são necessários os valores do fator de ramificação, movimentos válidos, e a profundidade da busca. O fator de ramificação dos nós de MIN e

MAX para o dominó de 4 pontas é calculado examinando-se o número de movimentos possíveis por jogada, nível a nível . O seu valor não apresenta comportamento linear como no jogo de xadrez ou no *stratego*. Isso se deve ao fato de depender das peças disponíveis na mão dos jogadores e das pontas livres na mesa. Partindo dessas considerações, pode-se calcular um limite superior para o fator de ramificação considerando um caso raro, porém possível, em que um jogador, na rodada inicial de um jogo, possui todas as pedras do *nipe* 6. Esse representa o pior caso, ou valor máximo, para o valor de ramificação.

As suas pedras podem ser posicionadas nas duas pontas livres iniciais. A sua escolha depende da estratégia adotada pelo jogador. A Figura 9 mostra como os dominós podem ser posicionados em um dos lados na carroça de sena (6-6).

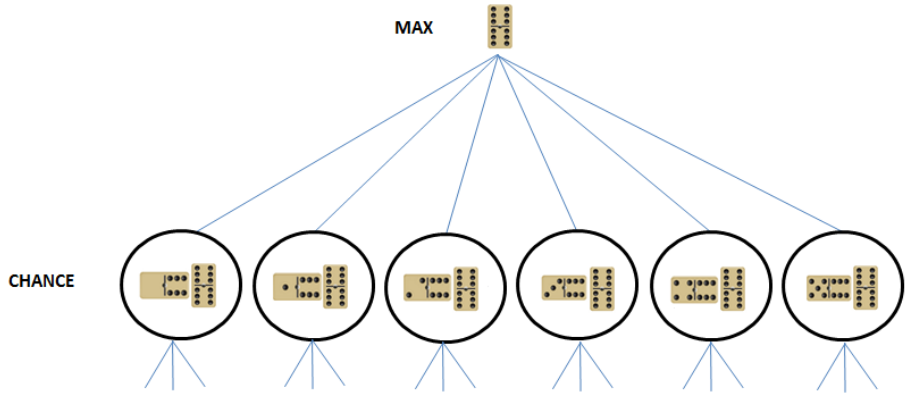


Figura 9 – Fator de ramificação máximo dos nós MAX e MIN para o jogo de dominó de 4 pontas.

A partir do exposto anteriormente, fixa-se um limite superior de 6 para o fator de ramificação dos nós MIN e MAX. Para os nós de chance, Figura 10, o número máximo de ramificações produzidas é 5, pois tem-se os eventos referentes às 4 pontas da mesa, quando estão habilitadas, e o evento de passe de um jogador. Na Figura 10 cada ramificação é identificada por $p(E)_i, 1 < i < 5$. Por último, define-se a profundidade máxima da busca.

Para o jogo de dominó o número máximo de jogadas possíveis é 25, segundo Hauk (2004), a complexidade da árvore de jogo para a busca em profundidade é dada pela Equação (9).

$$C = O\ ((fator\ de\ ramificação\ )^{profundidade} \times (eventos\ de\ chance)^{profundidade}) \quad (9)$$

Aplicando-se os valores definidos obtém-se:

$$C = O(6^{25} \times 5^{25}) \approx O(10^{36})$$

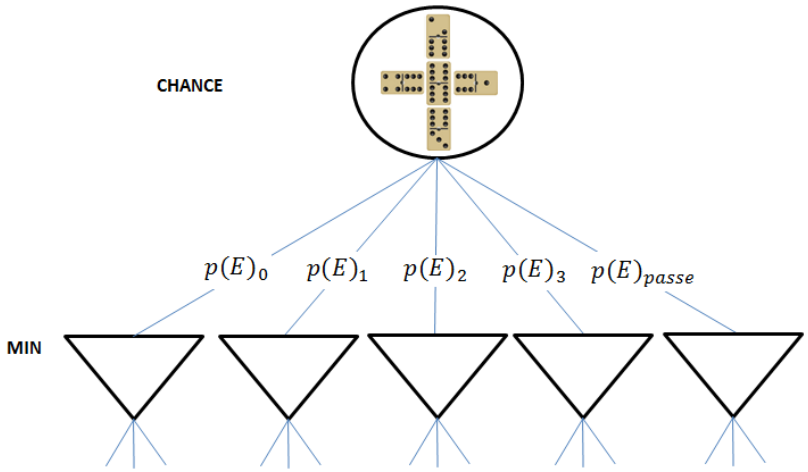


Figura 10 – Fator de ramificação máximo para nós de chance. $p(E)_0$, $p(E)_1$, $p(E)_2$, $p(E)_3$ e $p(E)_{passe}$ representam as probabilidades do jogador "MIN" possuir pedras que que encaixem na pontas 0, 1, 2, 3 e passar na vez, respectivamente.

O valor da complexidade da árvore de jogo encontrado a partir da Equação (9) representa a quantidade máxima de nós folhas que poderá ser encontrada no jogo de dominó. O Quadro 3 apresenta a complexidade da árvore de jogo de alguns jogos, considerando todos os níveis da árvore de busca. Através do Quadro 3 observa-se que bilhões de nós são gerados no processo de busca. Para a criação de agentes inteligentes os autores reduzem as peças do jogo ou realizam uma busca parcial. Conforme deduzido anteriormente, o processo de busca para o jogo de dominó de 4 pontas apresenta menor complexidade na árvore de busca do que alguns jogos analisados no Quadro 3.

Quadro 3 – Complexidade de alguns jogos (ARTS, 2010).

JOGO	Espaço de Estados	Árvore de Jogo
Trilha	10^{10}	10^{50}
Pentaminós	10^{12}	10^{18}
Connect-four	10^{14}	10^{21}
Damas	10^{21}	10^{31}
Othello	10^{28}	10^{58}
Xadrez	10^{46}	10^{123}
Xadres Chinês	10^{48}	10^{150}
Stratego	10^{115}	10^{535}
Go	10^{172}	10^{360}

3.7.2 Complexidade do espaço de estados

A complexidade de espaço de estados de um jogo é definida como o número de posições legais do jogo obtidas a partir do estado inicial, (HEULE; ROTHKRANTZ, 2007). O cálculo das configurações possíveis de alguns jogos é complexo, o que faz com que alguns autores obtenham um valor superestimado. Para o jogo OnToP, Briesemeister (2009) obteve um limite superior, que incluía configurações ilegais, para as possíveis configurações do tabuleiro em cada nível do jogo e somou os resultados obtidos. A complexidade do espaço de estados do jogo de dominó de 4 pontas foi obtida experimentalmente, pois para este jogo é difícil obter uma formulação matemática para o seu cálculo. Primeiramente, foi armazenada a quantidade de estados de jogo gerada em um processo de busca com profundidade 10, o que corresponde ao mapeamento até a fase 11 do jogo, esse critério de parada foi escolhido por se tratar do nível máximo de exploração da árvore de jogo realizada nesse trabalho. Por último, soma-se o resultado obtido em casa fase. O Quadro 4 apresenta a quantidade de configurações possíveis para o jogo de dominó de 4 pontas, iniciando-se com a carroça 6-6.

Quadro 4 – Configurações possíveis do jogo a partir da carroça 6-6.

Fase do jogo	Configurações
1	1
2	6
3	66
4	870
5	12510
6	192120
7	3019200
8	$4.70 \cdot 10^{7}$
9	$7.05 \cdot 10^{8}$
10	$1.02 \cdot 10^{10}$
11	$1.38 \cdot 10^{11}$
Total	$1.50 \cdot 10^{11}$

Em uma partida do jogo de dominó de 4 pontas, a rodada inicial deve ser iniciada com a carroça 6-6 e as rodadas seguintes com qualquer carroça, logo, tem-se 7 configurações iniciais possíveis. A partir do exposto, deve-se multiplicar o resultado obtido no Quadro 4 por 7, obtendo – se o seguinte valor de C_{EE} de acordo com a Equação (10).

$$C_{EE} = 7 \sum_{i=1}^{n} {C_{EE}}^{i} \tag{10}$$

Em que:

C_{EE} – complexidade do espaço de estados;

i – *i*-ésimo nível do jogo.

A variável *n* representa a fase do jogo. Aplicando-se o valor total obtido no Quadro 4 obtém-se:

$$C_{EE} = \quad 1.05 \cdot 10^{12}$$

O valor obtido acima representa a quantidade de posições legais do jogo que poderá ser atingida neste experimento, a complexidade de espaço de estados para o jogo de dominó de 4 pontas, até a fase 11 do jogo.

3.8 FUNÇÕES DE AVALIAÇÃO

Funções de avaliação (também conhecidas como avaliadores estáticos, porque são utilizados para avaliar um jogo em uma única posição estática) são vitais para a maioria dos programas de computador para jogos. Isso acontece porque a maioria dos jogos não permite a busca em toda a árvore de jogo, devido ao seu enorme tamanho. Por isso, a prospecção da melhor jogada raramente irá atingir os nós folhas da árvore, os quais representam uma vitória, derrota ou empate no jogo. Isso significa também que o programa necessita ser capaz de parar a busca em um determinado nível de profundidade e avaliar a posição do tabuleiro a partir do nó corrente. Por isso, uma função de avaliação é usada para examinar uma posição particular do tabuleiro de jogo e estimar a melhor jogada a ser realizada pelo jogador naquele instante do jogo. Devido à enorme quantidade de posições avaliadas de um jogo, a função de avaliação deverá ser eficiente para evitar decisões que levem à situações desfavoráveis.

3.8.1 Função de Avaliação para o jogo de dominó de 4 pontas proposta em (ANTONIO *et al.*, 2013)

A função de avaliação para o jogo de dominó de 4 pontas proposta por (ANTONIO *et al.*, 2013) consiste em dois termos, T_{n1} e T_{n2}, como definido na Equação (2). Nessa expressão, a variável *n* identifica a jogada que está sendo avaliada. O termo T_{n1} corresponde à pontuação que poderá ser obtida com a jogada, e o termo T_{n2} é um valor heurístico referente à estratégia adotada pelo jogador, que poderá optar por jogar em seu próprio benefício, ajudar o parceiro ou até mesmo tentar bloquear as jogadas da dupla adversária.

Para a escolha da melhor jogada, todas as informações relevantes do estado corrente do jogo foram armazenados em vetores, que serão incorporados aos termos T_{n1} e T_{n2}. Os vetores de estado serão atualizados a cada jogada. A definição de cada uma das componentes do vetor de estado é dada a seguir:

- V_0: quantidade de pedras em jogo para cada numeração;
- V_1: quantidade de pedras na mão do jogador para cada numeração;
- V_2: quantidade de pedras para cada numeração nas pontas da mesa;
- V_3: quantidade de pedras jogadas pela dupla para cada numeração;
- V_4: indica a numeração em que o oponente à esquerda passou;
- V_5: indica a numeração em que o oponente à direita passou;
- V_6: indica a numeração em que o parceiro passou.

O termo T_{n1}, dado pela Equação (11), incorpora a pontuação possível para as situações apresentadas na Seção 3.1.

$$T_{n1} = P_1 + P_2 + P_3 + P_4 \tag{11}$$

O termo T_{n2}, como pode ser visto na Equação (12), possui duas partes.

$$T_{n2} = -E_1 + E_2 \tag{12}$$

A parte E_1 do termo T_{n2} considera a possibilidade de bloquear as jogadas do oponente à esquerda do jogador, analisando-se a numeração das pedras que estão nas pontas da mesa, através de V_2. Nessa parte, são consideradas as pedras que já foram jogadas, incluindo-se o valor de V_0, e as pedras na mão do jogador na vez, indicadas por V_1, como pode ser visto na Equação (13). A variável L_1 representa a numeração da ponta em análise e os coeficientes α_1,

α_2 e α_3 controlam a importância de cada termo de E_1 em relação às outras parcelas da função de avaliação.

$$E_1 = \alpha_1 V_0(L_1) + \alpha_2 V_1(L_1) + \alpha_3 V_2(L_1) \quad (13)$$

A parte E_2 do termo T_{n2} tem o objetivo de facilitar as jogadas futuras do jogador, o seu cálculo é realizado de acordo com a Equação (14). Nesse caso, será analisada uma possível nova numeração para uma ponta, como resultado da jogada a ser realizada. Essa nova numeração é representada por L_2. O jogador também tentará facilitar as jogadas de seu parceiro, com a inclusão do vetor de estados V_3. Os coeficientes α_4, α_5, α_6 e α_7 controlam a importância de cada termo de E_2 em relação às outras parcelas da função de avaliação.

$$E_2 = \alpha_4 V_0(L_2) + \alpha_5 V_1(L_2) + \alpha_6 V_2(L_2) + \alpha_7 V_3(L_2) \quad (14)$$

A função de avaliação descrita através das Equações (5) e de 11-14 corresponde à estratégia que obteve melhor desempenho, estratégia 1, no trabalho de Antonio *et al.* (2013). Para as outras estratégias definidas pelos autores, os coeficientes das parcelas que não fazem parte do objetivo do jogador recebem valor zero. Os coeficientes $\alpha's$ foram otimizados através de simulações, utilizando-se a técnica de algoritmos genéticos, conforme descrito na Seção 2.2.

4 MATERIAIS E MÉTODOS

Para realizar as simulações do jogo de dominó de 4 pontas utilizando o processo de busca do algoritmo *expectiminimax* utiliza-se um microcomputador com processador Intel Core i5 @ 2.5 GHz e 4 GB de RAM, operando sob a plataforma Microsoft Windows 8.1. O programa que simula *n* partidas do jogo de dominó de 4 pontas foi desenvolvido em linguagem Java, assim como os *scripts* que descrevem o funcionamento do agente inteligente baseado no algoritmo *expectiminimax*. As simulações foram realizadas no Eclipse, que é uma IDE (ambiente integrado de desenvolvimento) que permite o desenvolvimento de software na linguagem Java. Para a implementação, definiram-se as seguintes características básicas do jogo:

- O jogo é constituído de 4 jogadores, divididos em 2 duplas (uma utilizando o agente inteligente e a outra utilizando a estratégia básica ou a melhor estratégia proposta por (ANTONIO *et al.*, 2013);
- O jogo é realizado com um conjunto de 28 pedras;
- A primeira pedra jogada na rodada inicial deverá ser a carroça 6-6;
- O jogo é finalizado quando uma dupla atingir 200 pontos;
- Não há cemitério.

Para a realização de partidas desenvolveu-se um aplicativo em Java. Este aplicativo possui interface gráfica que permite a integração entre jogadores humanos e jogadores virtuais em uma "mesa" de dominó de 4 pontas virtual. Cada jogador humano conecta-se ao jogo de dominó através da internet e o aplicativo cria dois agentes inteligentes para escolher as jogadas da dupla adversária. Também é possível realizar partidas em que todos os jogadores são virtuais. O algoritmo para a obtenção da melhor jogada utilizado por cada dupla será

apresentando no decorrer desta Seção. A Figura 11 mostra a interface gráfica do aplicativo desenvolvido para o jogo de dominó de 4 pontas.

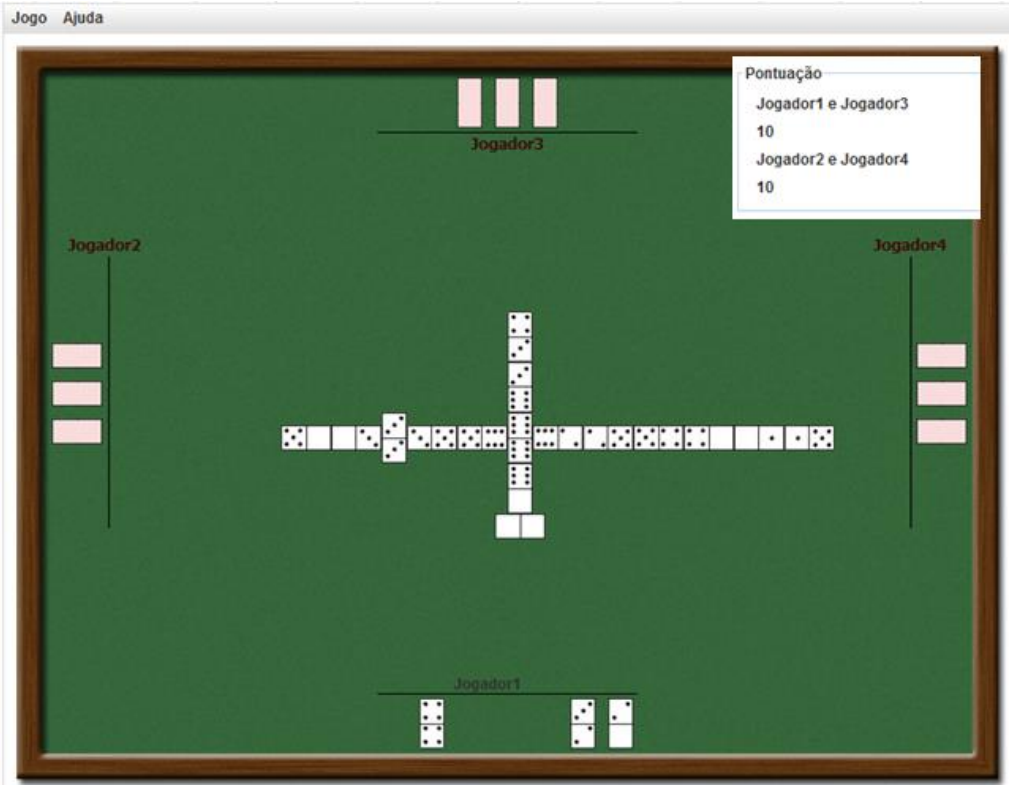


Figura 11 – Aplicativo desenvolvido para o jogo de dominó de 4 pontas.

A metodologia adotada para realizar o processo de busca *expectiminimax* para o jogo de dominó de 4 pontas segue as etapas descritas nas seções seguintes.

4.1 METODOLOGIA PARA A BUSCA *EXPECTIMINIMAX* PARCIAL

O diagrama apresentado na Figura 12 ilustra a divisão do programa jogado de dominó em dois módulos principais. A interface gráfica responsável pela iteração entre o humano e o computador possui dois modos de operação: no modo 1 permite-se que o usuário seja um dos jogadores da dupla formada por humanos e o agente *expectiminimax* seja utilizado pela dupla virtual. No modo 2, a interface gráfica permite que o usuário simule partidas entre a dupla *expectiminimax* e duplas que utilizam estratégias que serão mostradas na Seção 4.5.

O diagrama mostrado na Figura 12 mostra o programa do jogo de dominó de 4 pontas com a interface gráfica operando no modo 2, o programa é representado pelo módulo maior, e consiste de mais três sub-módulos: 1, 2 e 3. O usuário representa o humano, responsável pela configuração das duplas e início da partida.

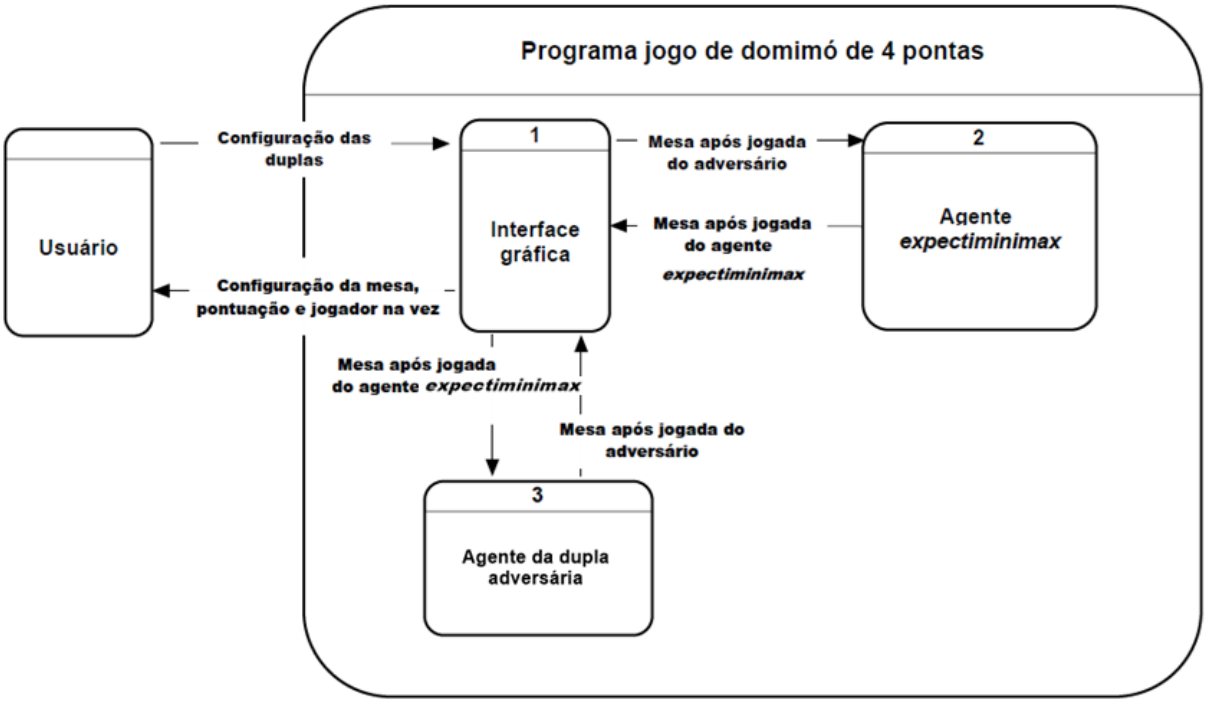


Figura 12 – Diagrama de fluxo de dados para o programa do jogo de dominó.

O módulo 1, que implementa a interface gráfica, permite ao usuário acompanhar o estado de uma rodada em tempo de execução. No diagrama acima, o módulo 3 representa o agente utilizado pela dupla adversária, que poderá adotar a estratégia básica de jogo ou a melhor estratégia obtida em (ANTONIO *et al.*,2013). Após cada jogada efetuada pelos jogadores, a interface gráfica também constrói uma descrição do estado corrente que será fornecida aos módulos dos agentes, que por sua vez, realizam suas jogadas alterando a configuração da mesa de jogo.

O diagrama apresentado na Figura 13 ilustra a constituição do módulo 2, agente *expectiminimax*. Este sub-módulo recebe a informação fornecida pela interface gráfica, utilizando-a como ponto de partida para a construção da árvore de jogo com busca limitada

em profundidade (*expectiminimax* parcial). O algoritmo *expectiminimax*, com adaptações ao jogo de dominó, pode ser considerado um algoritmo de pesquisa para jogos com múltiplos jogadores e de informação imperfeita, e que necessita recorrer a outros três sub-módulos para obter informações referentes ao estado de jogo.

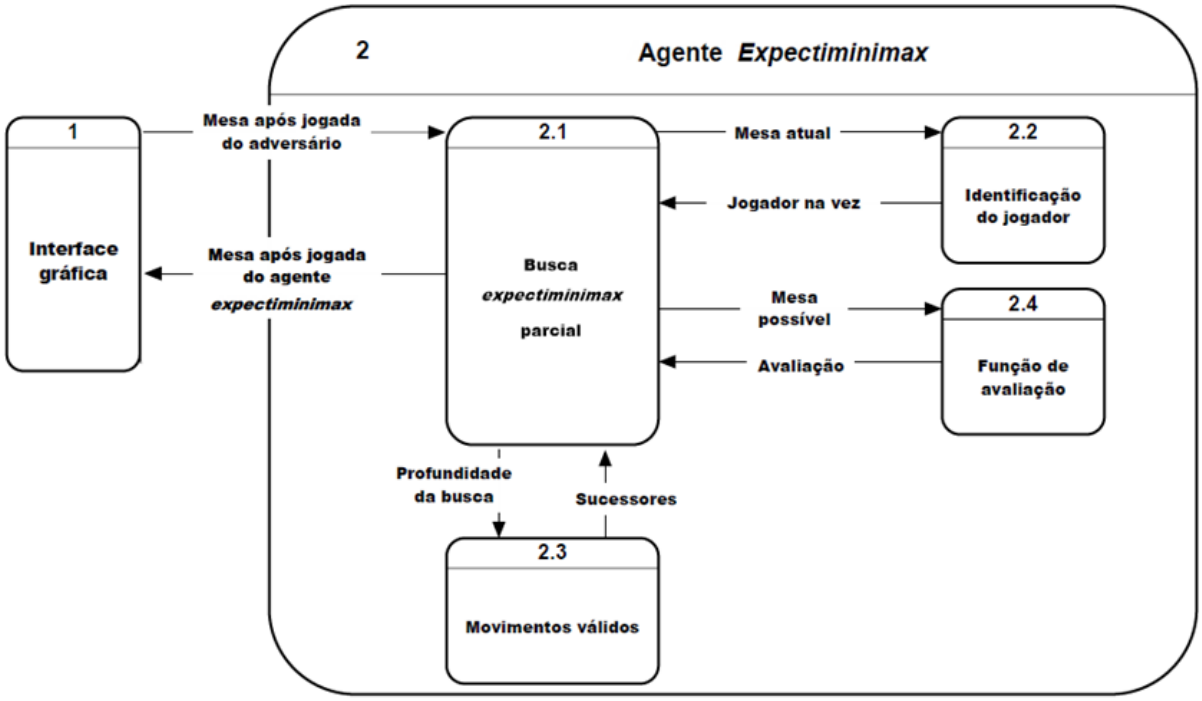


Figura 13 – Divisão em sub-módulos do módulo agente inteligente.

O sub-módulo 2.2, responsável pela identificação do jogador na vez, recebe uma descrição do estado da mesa, que inclui o *índice* do último jogador a realizar uma jogada e a pontuação das duplas, em que $índice \in \{0, 1, 2, 3\}$. Este sub-módulo retorna o índice do jogador que realizará a próxima jogada. Quando o sub-módulo 2.4 recebe uma possível configuração da mesa de jogo durante a pesquisa em profundidade, ele retornará um valor numérico proporcional às chances de vitória da dupla que utiliza o agente *expectiminimax* na posição de análise.

Para a construção da árvore de jogo, o sub-módulo 2.1 indicará a profundidade da busca ao sub-módulo 2.3, que retornará os possíveis estados sucessores (pedras conhecidas e

ocultas encaixadas em pontas válidas da mesa de jogo). É no sub-módulo 2.3 que se concentram os conhecimentos das regras especificas do jogo de dominó de 4 pontas que este agente possui.

4.2 ÁRVORE DE JOGO

A árvore de jogo elaborada para o dominó de 4 pontas possui 3 tipos de nós: MAX, MIN e CHANCE. O primeiro tipo de nó corresponde aos jogadores que utilizam o algoritmo *expectiminimax*, sendo chamados de dupla 1. O segundo, corresponde aos adversários, dupla 2. O último, aos eventos de chance referentes a cada uma das pontas válidas da mesa em um instante qualquer do jogo. Esses últimos informam sobre as chances do jogador em análise possuir pedras que se encaixem nas pontas. A Figura 14 ilustra a árvore de jogo parcial para o dominó de 4 pontas que será utilizada nessa dissertação.

Na situação apresentada na Figura 14, realiza-se uma busca limitada com profundidade 2. Na árvore de jogo elaborada para esta dissertação, a busca sempre deverá terminar em um nó de chance. Para este exemplo, o nó raiz corresponde a um estado intermediário de uma rodada, em que o ponto de partida da busca corresponde a mesa com as quatro pontas habilitadas para fazer jogadas.

O primeiro triângulo representa o jogador na vez da dupla que utiliza o agente *expectiminimax* (MAX). Os círculos indicam os eventos de chance, com seus valores de probabilidade associados. Os triângulos invertidos indicam a vez do oponente à esquerda do jogador da dupla 1 (MIN). Para essa dissertação o algoritmo 3 mostra a função utilizada para a busca *expectiminimax*.

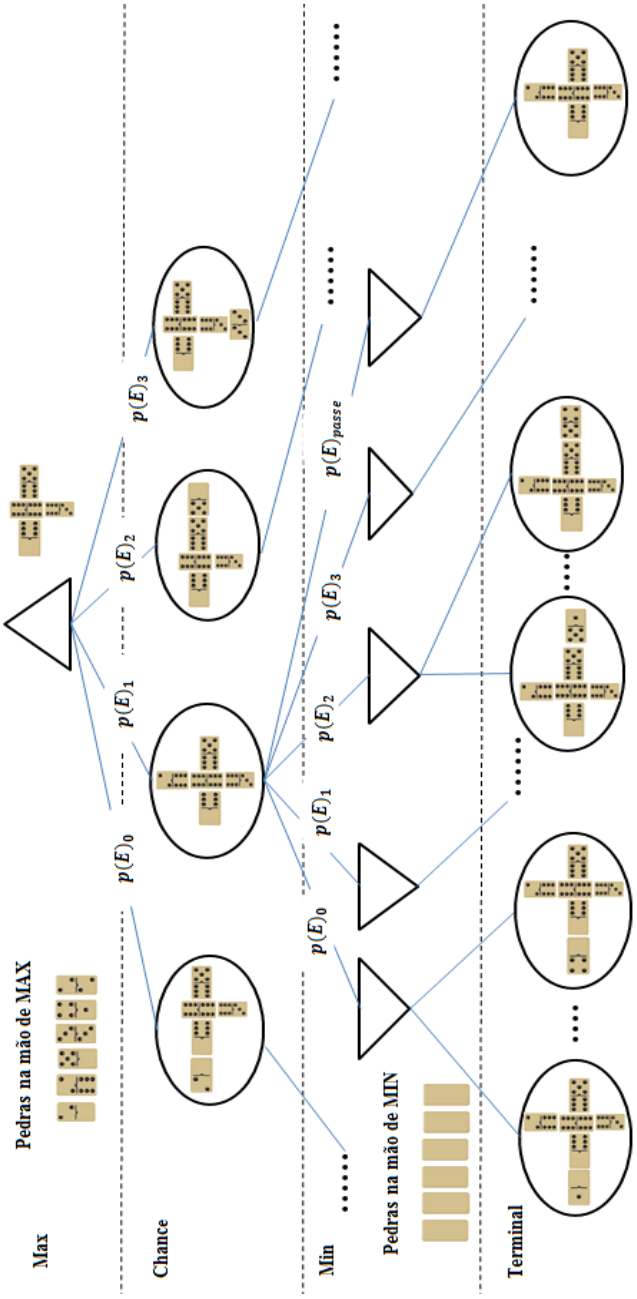


Figura 14 – Árvore de jogo parcial do dominó de 4 pontas, $p(E)_0$, $p(E)_1$, $p(E)_2$, $p(E)_3$ e $p(E)_{passe}$ representam as probabilidades do jogador "MAX" e "MIN" possuir pedras que se encaixem na pontas 0, 1, 2, 3 e passar na vez, respectivamente.

Algoritmo3 *EXPECTIMINIMAX para o jogo de dominó*

Função *EXPECTIMINIMAX (nó, profundidade, mesa)*

SE profundidade=0

retorne Pontuação Dupla*Expectiminimax* – Pontuação Dupla Adversária

SE o nó corrente é MIN

melhorPontuaçao ←+ ∞

PARA CADA movimento possível

pontuaçaoCorrente= *EXPECTIMINIMAX* (noChance,profundidade-1)
melhorPontuaçao←min(melhorPontuaçao, pontuaçaoCorrente)
desfaz jogada

SENÃO SE o nó corrente é MAX

melhorPontuaçao ← - ∞

PARA CADA movimento possível

pontuaçaoCorrente = *EXPECTIMINIMAX* (noChance,profundidade-1)
melhorPontuaçao←max(melhorPontuaçao, pontuaçaoCorrente)
desfaz jogada

SENÃO SE o nó corrente é CHANCE

melhorPontuaçao ←0

PARA CADA ponta válida da mesa para jogar

pontuaçaoCorrente = *EXPECTIMINIMAX* (noFilho , profundidade-1)
melhorPontuaçao←melhorPontuaçao + (Probabilidade[ponta]*pontuaçaoCorrente)

retorne melhorPontuaçao

Assim como o algoritmo *expectiminimax* original, o algoritmo de busca para o dominó de 4 pontas, elaborado nesta dissertação, é um procedimento recursivo direto, que toma por base dois procedimentos auxiliares, específicos do jogo, o gerador de movimentos e a avaliação estática. O procedimento *expectiminimax* para este jogo precisará de três parâmetros para ser iniciado, o nó representando o jogador que realizará a jogada, a profundidade inicial de busca, indicando o critério de parada e a configuração atual da mesa.

4.3 FUNÇÃO DE AVALIAÇÃO HEURÍSTICA

Quando o processo de busca termina, atingindo os nós de fronteira, que nesse trabalho substituem os nós folhas do algoritmo *minimax*, a função que estima o estado da dupla "MAX" nessa fase do jogo, será a diferença entre sua pontuação e a pontuação da dupla adversária. A dupla que utiliza o algoritmo *expectiminimax* tenta maximizar o valor dessa função, enquanto que a dupla adversária tenta minimizar essa diferença. No final da busca, esse valor será propagado para os nós acima na árvore, sofrendo modificações até chegar ao nó raiz, a fim de escolher-se a melhor jogada. O valor dessa diferença poderá ser de no mínimo -400 pontos e no máximo 400 pontos. A Equação (15) expressa essa função de avaliação de um nó de fronteira ou de um nó folha.

$$V = Pontuação_{dupla\_expectiminmax} - Pontuação_{dupla\_adversária} \tag{15}$$

Em que:

V – Valor da função de avaliação em um nó de fronteira ou em um nó folha.

4.4 MODELAGEM DOS EVENTOS DE CHANCE

De acordo com Walpole *et al*. (2011) se um espaço amostral contém *N* elementos, e cada um deles tem a mesma probabilidade de ocorrer, atribui-se a probabilidade de $1/N$ para cada um dos elementos. A probabilidade *p* de um evento qualquer *A*, contendo *n* elementos desse conjunto, será a relação entre o número de elementos em *A* e o espaço amostral *N*, como mostrado na Equação (16). Partindo disso, realiza-se a modelagem dos eventos de chance para o jogo de dominó de 4 pontas. Os possíveis casos são enumerados a seguir:

$$p = \frac{n}{N} \tag{16}$$

1. Passe do jogador na vez, da dupla *expectiminimax*. Como suas pedras são conhecidas, caso ele não possua pedras que encaixem nas pontas, a probabilidade deste evento ocorrer será $p(E) = 1$, caso contrário tem-se $p(E) = 0$;
2. Existe a possibilidade do adversário à direita, do parceiro ou do adversário à esquerda do jogador na vez, da dupla *expectiminimax*, possuir pedras que se encaixem em uma das quatro pontas. Seja o *nipe* dessa ponta expressa por $nipe_p$. O valor $p(E)$ é calculado de acordo com o procedimento a seguir:

    2.1. Primeiramente, calcula-se o numerador da expressão $p(E)$. Essa etapa corresponde a soma das possibilidades de um jogador em análise possuir de 1 a *t* pedras do $nipe_p$ correspondente a uma ponta. O valor de *t* é calculado de acordo com a Equação (17).

$$t = mínimo(P_i, P_{nipe_p}) \tag{17}$$

Em que:

P_i – Número de peças na mão de um jogador da mesa, neste caso representado por *i*, como existem quatro jogadores no jogo de dominó de 4 pontas, $i \in \{0, 1, 2, 3\}$;

P_{nipe_p} – Número de peças ocultas do $nipe_p$, com $nipe_p \in \{0, 1, 2, 3, 4, 5\ 6\}$.

Depois de calculado o número máximo de pedras para certo $nipe_p$ que o jogador poderá ter em mãos, calcula-se as distribuições possíveis na mão do jogador para essas *t* possibilidades. O cálculo das distribuições, conforme mostrado a seguir, envolve *t* termos $T_k, com\ 1 < k < t$.

Para descobrir a forma com que as pedras ocultas, representado por P_{oc}, poderão se combinar nas mãos dos jogadores da mesa, será apresentada uma situação de jogo, em que: P_0 (jogador virtual, que utiliza o algoritmo *expectiminimax*), P_1 (oponente à esquerda de P_0), P_2 (parceiro de P_0, que também utiliza o algoritmo *expectiminimax*) e P_3 (oponente à direita de P_0). A descrição das peças é dada por:

$P_{oc} - p_{11}, p_{12}, p_{13}, p_1, p_2, p_3, p_4$;

P_{nipe_p} – peças p_{11}, p_{12}, p_{13};

P_1 – 3 peças;

P_2 – 2 peças;

P_3 – 2 peças.

Em que p_{11}, p_{12} e p_{13} são pedras do *nipe* em análise, neste caso $nipe_1$, e as pedras p_1, p_2, p_3, p_4 são de *nipes* diferentes. Neste exemplo, os jogadores em análise, P_1, P_2 e P_3 possuem 3, 2 e 2 peças, respectivamente. De acordo com a definição, observa-se que *t*=3. A partir disto, pode-se mapear as distribuições possíveis de 7 pedras ocultas para P_0. A Figura 15 mostra essa distribuição.

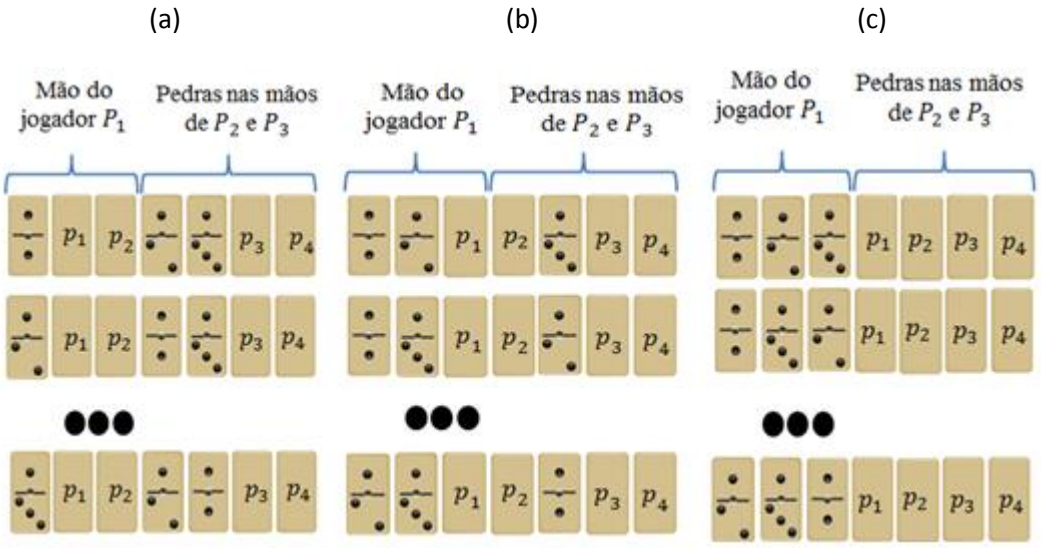


Figura 15 – Distribuição das pedras ocultas nas mãos de P_1, P_2 e P_3, para o exemplo de situação de jogo mostrado, com: (a) uma pedra (b) duas pedras e (c) três pedras no $nipe_1$ na mão de P_1.

Os Quadros 5, 6 e 7 mostram um resumo das distribuições para o caso em que o jogador P_1 possui 1, 2 e 3 pedras do $nipe_1$, respectivamente.

Quadro 5 – Distribuições possíveis das pedras ocultas com P_1 possuindo 1 pedra do $nipe_1$.

N° Pedras	Pedra	Pos1	Pos2	Pos3	Permutações				Distribuições
1	p_{11}	p_{11}	p_1	p_2	p_3	p_4	p_{12}	p_{13}	Arranjo de $Pnipe_p$, 1 a 1
		p_{11}	p_1	p_3	p_2	p_4	p_{12}	p_{13}	
		p_{11}	p_2	p_3	p_1	p_4	p_{12}	p_{13}	Combinação de P_1, 1 a 1
		....			....				
		p_1	p_{11}	p_2	p_3	p_4	p_{12}	p_{13}	Arranjo de $P_{oc} - Pnipe_p$, P_1 -1 a P_1 -1
		p_1	p_{11}	p_3	p_2	p_4	p_{12}	p_{13}	
		p_2	p_{11}	p_3	p_1	p_4	p_{12}	p_{13}	Permutação de $P_{oc} - P_1$
....	....	....			....				

Quadro 6 – Distribuições possíveis das pedras ocultas com P_1 possuindo 2 pedras do $nipe_1$.

N. Pedras	Pedra	Pos1	Pos2	Pos3	Permutações				Distribuições
2	p_{11},p_{12}	p_{11}	p_{12}	p_1	p_2	p_3	p_4	p_{13}	Arranjo de $Pnipe_p$, 2 a 2
		p_{11}	p_{13}	p_1	p_2	p_3	p_4	p_{12}	
		p_{12}	p_{13}	p_1	p_2	p_3	p_4	p_{11}	Combinação de P_1, 2 a 2
		....			....				
		p_1	p_{11}	p_{12}	p_2	p_3	p_4	p_{13}	Arranjo de $P_{oc} - Pnipe_p$, P_1 - 2 a P_1 - 2
		p_1	p_{11}	p_{13}	p_2	p_3	p_4	p_{12}	
		p_1	p_{12}	p_{13}	p_2	p_3	p_4	p_{11}	Permutação de $P_{oc} - P_1$
....	....	....			....				

Quadro 7 – Distribuições possíveis das pedras ocultas com P_1 possuindo 3 pedras do $nipe_1$.

N. Pedras	Pedra	Pos1	Pos2	Pos3	Permutações				Distribuições
3	p_{11},p_{12} e p_{13}	p_{11}	p_{12}	p_{13}	p_1	p_2	p_3	p_4	Arranjo de $Pnipe_p$, 3 a 3
		p_{11}	p_{13}	p_{12}	p_2	p_1	p_3	p_4	
		p_{12}	p_{11}	p_{13}	p_1	p_2	p_4	p_3	Combinação de P_1, 3 a 3
		....			....				
		p_{12}	p_{13}	p_{11}	p_1	p_4	p_3	p_2	Arranjo de $P_{oc} - Pnipe_p$, P_1 - 3 a P_1 - 3
		p_{13}	p_{12}	p_{11}	p_4	p_2	p_3	p_1	
		p_{13}	p_{11}	p_{12}	p_1	p_4	p_2	p_3	Permutação de $P_{oc} - P_1$
....	....	....			....				

As últimas colunas dos Quadros 5, 6 e 7 mostram os cálculos dos termos necessários para a obtenção da distribuição total T_k, de todas as pedras ocultas nessa situação de jogo. Para o exemplo acima serão necessários os cálculos de T_1, T_2 e T_3.

Para organizar os termos, as seguintes definições são necessárias:

P_{oc} – número de peças ocultas;

$$\alpha = P_{oc} - P_{nipe_p}. \tag{18}$$

Fazendo-se:

$\beta_1 = P_i - 1;$

$\beta_2 = P_i - 2;$

.....

$$\beta_n = P_i - n. \tag{19}$$

O cálculo de uma possibilidade T_k é dada, então por:

$$T_1 = A_{P_{nipe_p},1} \times C_{P_{j_i},1} \times A_{\alpha,\beta_1} \times (P_{oc} - P_i)!$$

$$T_2 = A_{P_{nipe_p},2} \times C_{P_{j_i},2} \times A_{\alpha,\beta_2} \times (P_{oc} - P_i)!$$

.....

$$T_k = A_{P_{nipe_p},t} \times C_{P_{j_i},k} \times A_{\alpha,\beta_k} \times (P_{oc} - P_i)! \quad (20)$$

Em que:

$A_{x,y}$ – Arranjo de x, y a y;

$C_{x,y}$ – Combinação de x, y a y.

O denominador da Equação (16), número de casos possíveis, é simplesmente dada por $P_{oc}!$. A probabilidade do evento correspondente a um indivíduo *i* possuir uma peça do $nipe_p$ de uma ponta é dada, então pela Equação (21).

$$p(E)_{nipe_p} = \frac{1}{P_{oc}!}\sum_{i=1}^{t} T_i \quad (21)$$

3. O adversário à direita, o parceiro ou o adversário à esquerda do jogador na vez da dupla *expectiminimax*, não possui pedras que encaixem em nenhuma das pontas da mesa. Neste caso as pedras são ocultas, e fazem parte do conjunto de informações que os jogadores da dupla *expectiminimax* utilizam. A probabilidade de esse evento ocorrer será dada por:

$$p(E)_{passe} = \frac{A_{P_{nipef},P_i} \times (P_{oc} - P_i)!}{P_{oc}!} \quad (22)$$

Em que:

P_{nipef} – Soma das pedras ocultas, em jogo, cujos *nipes* não estão nas pontas da mesa.

4.5 ESTRATÉGIAS PARA O AGENTE *EXPECTIMINIMAX*

Na prática não se pode explorar todos os níveis da árvore de jogo do dominó de 4 pontas, pois o número de estados a serem examinados pelo algoritmo *expectiminimax* é exponencial em relação em relação ao número de movimentos. Com o objetivo de explorar o máximo possível a memória disponível do computador antes do algoritmo ficar preso na recursão, utiliza-se o algoritmo *expectiminimax* com busca parcial. Limitando-se a profundidade da busca durante os experimentos. Antes de definir a profundidade para cada estratégia de busca do agente, realiza-se a divisão do jogo em 3 fases, em consonância com a técnica *phase-related search* (busca por fases): início, meio e fim. Para definir o intervalo das jogadas presentes em todas as fases definidas anteriormente, primeiramente foram realizadas 5000 partidas do jogo de dominó entre duplas formadas por jogadores com a estratégia básica de jogo, produzindo 10282 rodadas. Em cada rodada foi medido o total de movimentos de jogo necessários para o seu término, os resultados obtidos podem ser vistos na Figura 16.

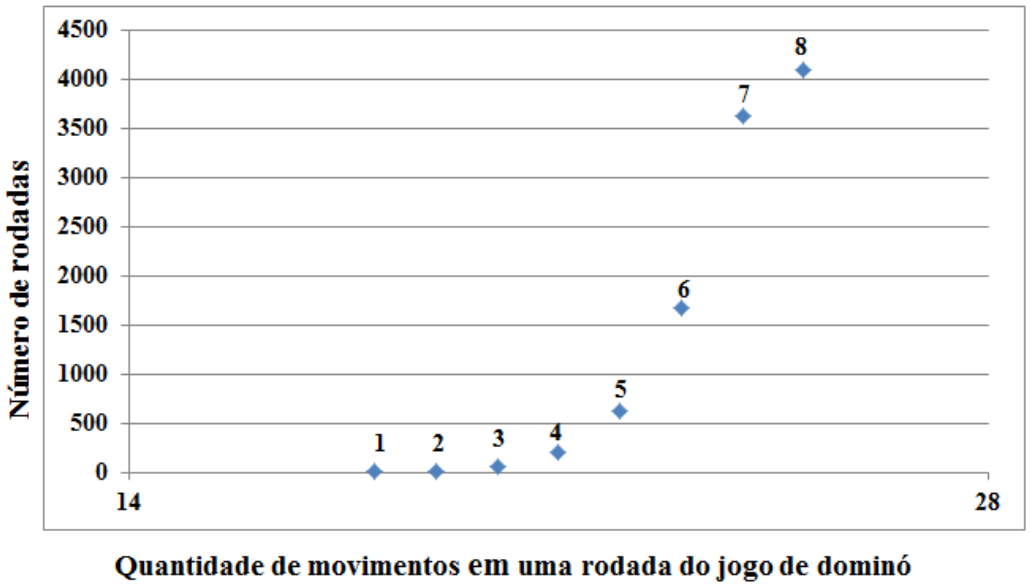


Figura 16 – Quantidade de movimentos máxima de 10282 rodadas.

No gráfico da Figura 16, os pontos indicam a quantidade de movimentos em uma rodada e o número de rodadas realizadas. Por exemplo, o ponto 1 informa que apenas 3 rodadas terminaram com 18 movimentos de jogo e o ponto 8 mostra que 4103 rodadas terminaram com 25 movimentos.

Briesemeister 2009 estimou o número de jogadas necessárias para finalizar o jogo OnTop através da média do número de jogadas em 1400 partidas, em seu experimento obteve o valor 31,36. A partir do exposto, pode-se calcular o número de jogadas em um rodada do jogo de dominó de 4 pontas através da média ponderada $\bar{p}$ da quantidade de movimentos de uma rodada obtidos a partir da Figura 16, de acordo com a Equação (23):

$$\bar{p} = \frac{p_1 x_1 + p_2 x_2 + \cdots + p_n x_n}{p_1 + p_2 + \cdots + p_n} = \frac{\sum_{i=1}^{n} p_i x_i}{\sum_{i=1}^{n} p_i} \tag{23}$$

Em que:

i – i-ésima quantidade de movimentos de uma rodada;

x_i – quantidade de movimentos de uma rodada;

p_i – Ocorrência de x_i em 10282 rodadas de jogo.

Aplicando-se os resultados apresentados na Figura 16 na Equação (23), obteve-se o valor de 24,03. Assim, aproximadamente, 24 movimentos são necessários para finalizar uma rodada de uma partida. Por fim, divide-se essa valor por 3 para determinar as fases do jogo. O Quadro 8 mostra o resultado da divisão de uma rodada.

Quadro 8 – Fases de uma rodada

Fase do Jogo	Número da Jogada
Início	1~8
Meio	9~16
Fim	>16

No Quadro 8, a fase início engloba as jogadas de 1 a 8. A fase meio engloba as jogadas de 9 a 16. A fase fim engloba as jogadas a partir da jogada 16. É importante que se faça a divisão do jogo em fases, pois como as informações disponíveis no jogo mudam com o decorrer do tempo, é possível a definição de diferentes estratégias em cada uma das fases em que o jogo foi dividido.

Nesta dissertação, em função da profundidade da busca em cada fase do jogo, foram definidas 8 estratégias distintas. O Quadro 9 mostra essas profundidades para cada estratégia e para cada fase do jogo.

Quadro 9 – Estratégias para o jogo de dominó de 4 pontas

Estratégias do Agente *Expectiminimax*	Profundidade em cada fase do jogo		
	Início	Meio	Fim
Estratégia 1	5	8	2
Estratégia 2	8	5	2
Estratégia 3	9	9	2
Estratégia 4	10	10	2
Estratégia 5	5	8	9
Estratégia 6	8	5	9
Estratégia 7	9	9	9
Estratégia 8	10	10	9

As profundidades de busca para as estratégias de 1 a 8 foram estabelecidas de acordo com a fase do jogo (*phase-related search).* As profundidades de busca nas fases início e fim variam de 5 a 10, este último valor corresponde ao maior potencial de exploração neste trabalho, antes do estouro computacional. Para avaliar a importância de explorações profundas na etapa final das rodadas, as estratégias 1 a 4, irão explorar esta fase em apenas dois níveis, enquanto as estratégias 5 a 8, irão explorar o seu maior potencial desta fase, que corresponde a 9 níveis.

A estratégia 8 corresponde a maior exploração da árvore de jogo nas fases início e meio. Como mencionado na Seção 3.7.1, a profundidade máxima da busca será 10, o que corresponde a 10 jogadas à frente do estado atual de jogo. Para profundidades maiores do que 10 a busca *expectiminimax* fica presa na recursão, retornando o resultado em tempos não aceitáveis para o nosso estudo. Para as estratégias de 1 a 4, na fase fim foi atribuída a profundidade de busca 2, com o objetivo de verificar se uma busca rasa na fase final de uma rodada propicia um maior número de vitórias da dupla *expectiminimax.*

Para as estratégias de 5 a 8, na fase fim foi atribuída a profundidade 9, pois, nessa fase de jogo, esse valor corresponde ao maior potencial de exploração, e nos momentos finais de uma rodada existem poucas pedras nas mãos dos jogadores, o que torna possível uma busca *expectiminimax* sem limitações de tempo.

5 RESULTADOS E DISCUSSÕES

Este capítulo apresenta os resultados obtidos para o agente proposto, para cada uma das estratégias definidas anteriormente. No processo de busca definido neste trabalho, será realizada a exploração parcial da árvore de jogo. Para determinar se as estratégias produziram resultados significantes, utilizam-se os testes de significância *t* de *Student* e Chi-Quadrado.

5.1 ANÁLISE DE DESEMPENHO DO ALGORITMO *EXPECTIMINIMAX* PARCIAL

De acordo com a metodologia apresentada no Capítulo 4, cada estratégia de busca consiste de três fases: início, meio e fim. O número de jogadas que define cada fase foi escolhido de acordo com as caraterísticas do jogo de dominó de 4 pontas. Com o critério de busca dividida em fases e com a árvore de jogo definida, pode-se avaliar o desempenho do algoritmo *expectimimax* no que diz respeito ao número de vitórias. A seguir será mostrado o resultado de quatro experimentos realizados com o intuito de avaliar a utilização do algoritmo *expectiminimax* com a modelagem dos eventos de chance proposta nesse trabalho.

Nos experimentos de 1 a 4 foram avaliados o desempenho médio em termos de vitórias, em 10 simulações de 5000 partidas. No primeiro experimento avalia-se o efeito de se variar a profundidade da busca nas fases de início, meio e fim sobre o número de vitórias, ao se jogar contra a estratégia básica. No segundo experimento avalia-se o desempenho das estratégias definidas no Quadro 9 contra a estratégia básica. No terceiro experimento será avaliado o desempenho das estratégias definidas no Quadro 9 contra a melhor estratégia obtida por Antonio *et al*. (2013). No experimento 4 avalia-se o desempenho das estratégias definidas no Quadro 9, quando as mesmas jogam entre si.

No experimento 5 foi avaliado o melhor desempenho de cada estratégia do Quadro 9 em 5000 partidas, contra a estratégia básica e contra a melhor estratégia definida por Antonio

et al. (2013). Por último, também foram avaliados os melhores resultados nos confrontos entre as estratégias do Quadro 9.

5.2 EXPERIMENTO 1: VITÓRIAS EM FUNÇÃO DA PROFUNDIDADE

O primeiro experimento consiste em avaliar o desempenho do algoritmo contra uma estratégia básica, variando-se a profundidade nas diferentes etapas do jogo. Os valores de profundidade foram definidos sem critério de investigação. Foram realizados três testes. No primeiro teste manteve-se a profundidade de busca das fases início e meio em 5 e 9, respectivamente, e variou-se o valor da profundidade da fase fim de 1 a 7. No segundo teste manteve-se a profundidade de busca das fases início e fim em 5 e 9, respectivamente, e variou-se o valor da profundidade da fase meio de 1 a 7. No terceiro teste manteve-se a profundidade de busca das fases meio e fim em 5 e 9, respectivamente, e variou-se a profundidade da fase início de 1 a 7. Os resultados obtidos são mostrados nas Figuras 17, 18 e 19.

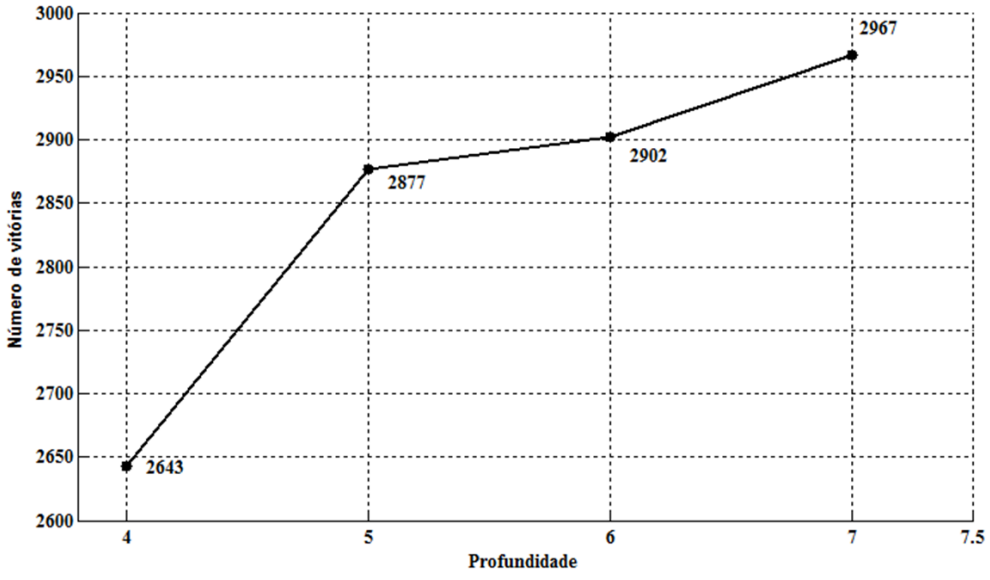


Figura 17 – Resultados obtidos com: início = 5; meio = 9; fim = 1, 2,..., 7.

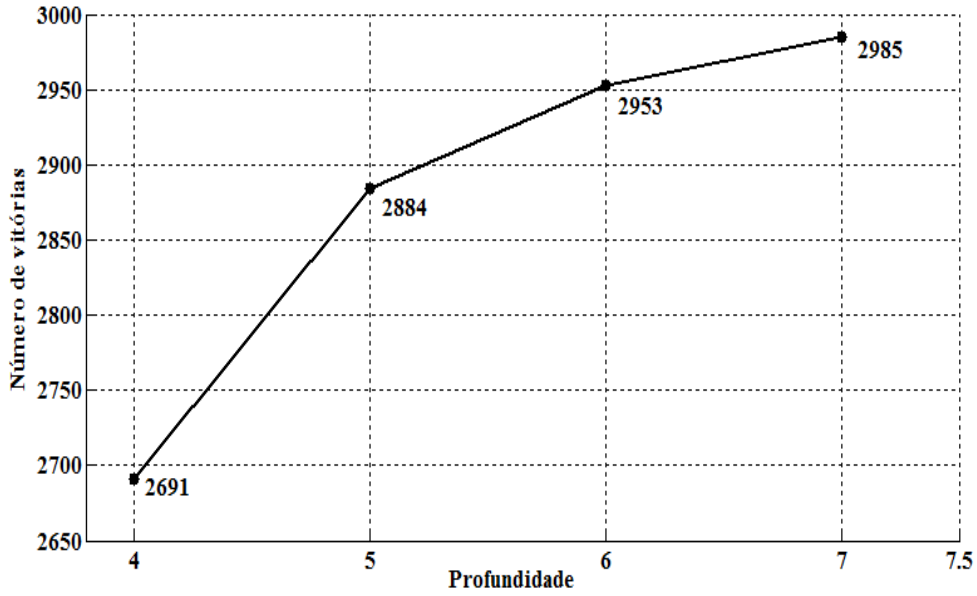


Figura 18 – Resultados obtidos com: início=5; meio = 1, 2,..., 7; fim =9.

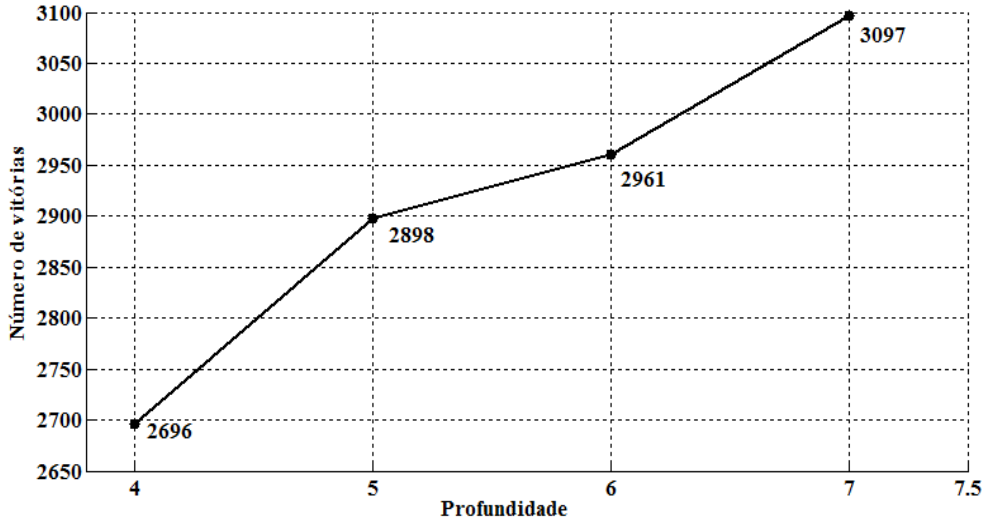


Figura 19 – Resultados obtidos com: início = 1,2,..., 7; meio = 5; fim = 9.

Esse experimento apresentou o desempenho para variações de profundidade nas fases início, meio e fim. Nos três testes pode-se observar que o número médio de vitórias aumenta com a profundidade da busca. Verifica-se também que o maior número médio de vitórias obtido contra uma estratégia básica, ocorreu no terceiro teste e foi de 3097, correspondendo a 130 vitórias a mais que o primeiro teste e 112 vitórias a mais do que no segundo teste. O terceiro teste corresponde a variações de profundidade em situações onde há maior quantidade de pedras em posse dos jogadores do que no primeiro teste e no segundo teste.

5.3 EXPERIMENTO 2: ESTRATÉGIAS DE 1 A 8 X ESTRATÉGIA BÁSICA

Neste experimento as estratégias descritas na Seção 4.5, serão testadas contra a estratégia básica de jogo. A avaliação de desempenho de cada uma foi feita através do teste de hipóteses conhecido como *t* de *Student*.

O teste de hipóteses *t* de *Student,* ou teste *t*, é um método estatístico utilizado para verificar se dois conjuntos de dados diferem significativamente. O método assume que os resultados seguem uma distribuição *t* de *student*. A hipótese nula geralmente assume que não há diferenças significantes entre as médias dos dois conjuntos de dados.

A hipótese nula definida anteriormente será H_0:$\bar{f} = \mu$, em que $\bar{f}$ é a média de vitórias obtida em 10 simulações com 5000 partidas cada. Neste trabalho cada simulação possui 5000 partidas pele fato de ser este o valor das simulações por Antonio *et al.* (2013), autores cujo trabalho que será comparado com o agente *expectiminimax* proposto, nas próximas seções. A variável μ é a média esperada para o experimento sob a hipótese de nulidade (2500). O valor de *t* é calculado por meio da Equação (24):

$$t = \frac{\bar{f} - \mu}{\sigma/\sqrt{N}} \tag{24}$$

Na Equação (24), σ e *N* representam, respectivamente, o desvio padrão e o tamanho do conjunto de dados e a variável *t* tem distribuição normal com (*N*-1) graus de liberdade. Nesta dissertação, o valor de *t* de *student* é calculado com o objetivo de comprovar ou rejeitar a superioridade estatística do agente inteligente proposto, e assim, rejeitar a hipótese nula (H_0). Isso será conseguidos se $t \geq tc$ ou $t \leq -tc$, onde *tc* corresponde ao valor crítico obtido na tabela de distribuição de valores de *t-Student*. Do ponto de vista estatístico, a média obtida experimentalmente pode ser: significativamente maior que $\mu = 2500$ (t $\geq$ t), significativamente menor do que $\mu = 2500$ ($t \leq -tc$), ou sem diferença significativa ($t_c \leq t$

≤ -*tc*). Neste trabalho, adota-se um nível de confiança de 99%, correspondendo a $tc = 2{,}821$. Os resultados podem ser vistos no Quadro 10.

Quadro 10 – Estratégias 1, 2, 3 e 4 contra a estratégia básica.

Testes	Adaptação f		H_0: $\bar{f} = 2500$
	$\bar{f}$	σ	Valor t
Estratégia 1 x Est. Básica	3095,6	8,64	218
Estratégia 2 x Est. Básica	3215,4	6,82	332
Estratégia 3 x Est. Básica	3489,4	6,20	504
Estratégia 4 x Est. Básica	3569,6	5,21	649
Estratégia 5 x Est. Básica	3116,1	8,53	228
Estratégia 6 x Est. Básica	3225,6	7,23	317
Estratégia 7 x Est. Básica	3535,6	7,12	460
Estratégia 8 x Est. Básica	3597,1	5,36	647

O Quadro 10 mostra a média de vitórias em 10 simulações de 5000 partidas e o desvio padrão calculado para cada estratégia. Todas as estratégias obtiveram resultados satisfatórios contra a estratégia básica de jogo, rejeitando assim, a hipótese nula ($tc = 2{,}821$). Nota-se que a estratégia 8 apresentou desempenho superior em relação às demais estratégias, com uma maior diferença em relação ao valor esperado e um menor valor para o desvio padrão. A propósito, a estratégia 8 é que realiza uma busca mais profunda nas fases início, meio e fim. Em contrapartida, a estratégia 1, que faz uma busca mais rasa na fase inicial das rodadas, teve pior desempenho entre as oito estratégias.

5.4 EXPERIMENTO 3: ESTRATÉGIAS DE 1 A 8 X ESTRATÉGIA DE ANTONIO *et al.* (2013)

Neste experimento será realizada uma comparação com o melhor resultado obtido em (ANTONIO *et al.*, 2013). No trabalho citado, também foram realizadas partidas contra a

estratégia básica de jogo. A função de avaliação, otimizada por algoritmo genético, que obteve melhor desempenho, ganhou em 70,06% das partidas contra a dupla que utilizava a estratégia básica de jogo, o que corresponde a 3503 vitórias em 5000 jogos. O Quadro 11 mostra os resultados obtidos quando as oito estratégias propostas nesse trabalho jogaram contra a melhor estratégia de ANTONIO *et al.* (2013).

Quadro 11 – Estratégias de 1 a 8 contra o melhor resultado em (ANTONIO *et al.*, 2013).

Testes	Adaptação f		$H_0: \bar{f} = 2500$
	$\bar{f}$	σ	Valor t
Estratégia 1 x Est. Antonio *et al.*(2013)	2314,7	7,21	-81,25
Estratégia 2 x Est. Antonio *et al.*(2013)	2363,2	6,34	-68,25
Estratégia 3 x Est. Antonio *et al.*(2013)	2783,2	6,20	144,52
Estratégia 4 x Est. Antonio *et al.*(2013)	2886,6	6,17	198,21
Estratégia 5 x Est. Antonio *et al.*(2013)	2331,6	5,21	-102,19
Estratégia 6 x Est. Antonio *et al.*(2013)	2375,8	5,35	-73,41
Estratégia 7 x Est. Antonio *et al.*(2013)	2795,1	5,92	157,76
Estratégia 8 x Est. Antonio *et al.*(2013)	2909,3	5,08	254,87

De acordo com o Quadro 11, as estratégias 3, 4, 7 e 8 obtiveram resultados superiores contra a melhor estratégia em (ANTONIO *et al.*, 2013). A propósito, dentre as estratégias propostas nesse estudo, essas estratégias representam os casos de busca mais profunda nas etapas início e meio. As estratégias 7 e 8 utilizam ao máximo o potencial de exploração na fase final das rodadas, como consequência, obtiveram melhor desempenho do que as estratégias 3 e 4, que realizam uma busca mais rasa na etapa final das rodadas.

5.5 EXPERIMENTO 4: CONFRONTOS ENTRE AS ESTRATÉGIAS DE 1 A 8

Nesse experimento, com o objetivo de validar os resultados obtidos contra as estratégias básica e a melhor estratégia em (ANTONIO *et al.*, 2013), cada estratégia foi

testada contra as outras sete estratégias definidas nesse estudo. Desta forma, a estratégia 1 realizou partidas contra as estratégias 2 a 8. O mesmo ocorreu para as demais estratégias. Os valores de *t* podem ser vistos no Quadro 12.

Quadro 12 – Resultados dos confrontos entre as estratégias de 1 a 8.

Testes		Adaptação f		$H_0: \bar{f} = 2500$
		$\bar{f}$	σ	Valor t
Estratégia 8	x Estratégia 1	3064,1	8,57	208,17
	x Estratégia 2	2788,9	7,45	122,70
	x Estratégia 3	2644,5	9,07	50,38
	x Estratégia 4	2578,5	7,60	32,64
	x Estratégia 5	3033,8	9,59	176,03
	x Estratégia 6	2878,7	5,46	219,42
	x Estratégia 7	2657,1	4,89	101,67
Estratégia 7	x Estratégia 1	2646,7	8,56	54,17
	x Estratégia 2	2614,4	7,55	47,94
	x Estratégia 3	2528,7	7,72	11,76
	x Estratégia 4	2485,8	7,54	-5,96
	x Estratégia 5	2650,9	6,08	78,46
	x Estratégia 6	2598,4	6,54	47,61
Estratégia 6	x Estratégia 1	2622,5	9,51	40,72
	x Estratégia 2	2545,6	10,71	13,46
	x Estratégia 3	2437,6	13,79	-14,31
	x Estratégia 4	2460,7	4,14	-30,03
	x Estratégia 5	2612,4	7,01	50,70
Estratégia 5	x Estratégia 1	2555,9	7,74	22,84
	x Estratégia 2	2440,6	11,35	-16,54
	x Estratégia 3	2411,6	9,89	-28,26
	x Estratégia 4	2404,4	15,72	-19,23
Estratégia 4	x Estratégia 1	3026,9	11,79	141,33
	x Estratégia 2	2865,8	6,78	170,64
	x Estratégia 3	2652,5	6,55	73,59
Estratégia 3	x Estratégia 1	2626,1	6,15	64,79
	x Estratégia 2	2596,5	11,67	26,14
Estratégia 2	x Estratégia 1	2606,9	11,52	29,34

*tc=2,821.

No Quadro 12 pode-se visualizar a média de vitórias e o desvio padrão para os testes realizados. Assim como nos outros experimentos, os resultados foram comparados com a hipótese de nulidade $\mu = 2500$. Em relação às estratégias 1 e 5, pode-se observar que as

mesmas não apresentaram desempenho aceitável perante todas as estratégias, pois todas as estratégias, quando confrontadas com as estratégias 1 e 5, obtiveram valores de média acima de 2500 vitórias e baixos valores de desvio padrão. O pior desempenho dessas estratégias foi observado contra a estratégia 8, onde obtiveram uma média de apenas 38,71% e 39,32% de vitórias em 10 simulações de 5000 partidas, para as estratégias 1 e 5, respectivamente.

Nas partidas realizadas pela estratégia 2 e 6, observa-se um desempenho um pouco melhor. As mesmas foram capazes de vencer as estratégias 1 e 5, alcançando valores médios de vitórias superiores a 50% em 10 simulações de 5000 partidas. Em contrapartida, perante as estratégias 3, 4, 7 e 8 as mesmas não obtiveram bons resultados, com valores médios de vitórias inferiores a 50%.

As estratégias 3 e 7 confirmaram sua superioridade perante as estratégias 1, 2, 5 e 6, com valores médios superiores a 50% de vitórias em 10 simulações de 5000 partidas. Porém, contra as estratégias com buscas mais profundas deste trabalho, estratégias 4 e 8, as mesmas não alcançaram a média de 50%.

As estratégias 4 e 8 confirmaram sua superioridade em relação a todas outras estratégias, em um confronto entre essas duas estratégias, a estratégia 8 foi superior, obtendo o percentual médio de vitórias de 51,57%. O melhor resultado obtido pela estratégia 8 foi o percentual médio de vitórias de 61,28% sobre a estratégia 1.

5.6 EXPERIMENTO 5: MELHORES RESULTADOS DAS ESTRATÉGIAS 1 À 8

Nos experimentos anteriores foram mostrados os resultados médios em experimentos que consistiam de 10 simulações de 5000 partidas. Nesta seção, será avaliado o melhor desempenho obtido em 5000 partidas, para cada estratégia contra as demais, incluindo a estratégia básica e a função de avaliação otimizada por algoritmo genético, apresentada por Antonio *et al.* (2013). No Quadro 13 estão presentes os resultados das simulações. A

significância estatística dos resultados foi avaliada através da aplicação do teste de significância Chi-quadrado (χ^2).

Quadro 13 – Melhores resultados obtidos para cada estratégia em 5000 partidas.

Jogos realizados			Vitórias		χ^2 (chi quadrado)
Dupla 1	x	Dupla 2	Dupla 1	Dupla 2	Dupla 1 x Dupla 2
Estratégia 1	x	Est. Básica	3105	1895	292,82
Estratégia 2	x	Est. Básica	3222	1778	417,03
Estratégia 3	x	Est. Básica	3521	1479	833,25
Estratégia 4	x	Est. Básica	3577	1423	927,94
Estratégia 5	x	Est. Básica	3127	1873	314,50
Estratégia 6	x	Est. Básica	3235	1765	432,18
Estratégia 7	x	Est. Básica	3543	1457	870,28
Estratégia 8	x	Est. Básica	3602	1398	971,52
Estratégia 1	x	Est. Antonio *et al*. (2013)	2325	2675	24,50
Estratégia 2	x	Est. Antonio *et al*. (2013)	2370	2630	13,52
Estratégia 3	x	Est. Antonio *et al*. (2013)	2791	2209	67,74
Estratégia 4	x	Est. Antonio *et al*. (2013)	2895	2105	124,82
Estratégia 5	x	Est. Antonio *et al*. (2013)	2342	2658	19,97
Estratégia 6	x	Est. Antonio *et al*. (2013)	2381	2619	11,33
Estratégia 7	x	Est. Antonio *et al*. (2013)	2805	2195	74,42
Estratégia 8	x	Est. Antonio *et al*. (2013)	2917	2083	139,11

O Chi-quadrado (χ^2), é um teste de significância estatística baseado numa comparação de proporções observadas com proporções esperadas. O Chi-quadrado compara uma frequência observada (número de ocorrência, distribuição de frequência) de certo evento com uma frequência esperada, obtida teoricamente. Adota-se a hipótese nula H_0: χ^2 (calculado) ≤ ${\chi_c}^2$ (tabelado), o valor de χ^2 (calculado) é dado por :

$$\boldsymbol{\chi^2 = \sum \frac{(f_0 - f_e)^2}{f_e}} \tag{25}$$

Na Equação (25) $\boldsymbol{f_0}$ é frequência observada, neste caso o número de vitórias de ambas as duplas, e $\boldsymbol{f_e}$ é a frequência esperada, no caso 2500 vitórias em 5000 partidas, para cada uma das duplas. Para a obtenção do valor de $\chi_c{}^2$ (tabelado) foi consultada a tabela de distribuição do Chi-quadrado (χ^2). Nesse experimento, adota-se um nível de confiança de 99%. O número de graus de liberdade é igual a 1. O valor de $\chi_c{}^2$ (tabelado) é de 10,827.

De acordo com o Quadro 13, todos os testes realizados apresentaram diferenças significativas em relação ao número de vitórias esperadas, pois os valores de χ^2(calculado) são maiores do que o valor de $\chi_c{}^2$(tabelado), 10,827 . Todas as estratégias mostram-se superiores em relação à estratégia básica de jogo. As estratégias 4 e 8, foram as que obtiveram melhor desempenho em relação à estratégia básica de jogo, com médias de 71,54% e 72,04% de vitórias, respectivamente. Todas as estratégias foram confrontadas com o melhor resultado obtido em (ANTONIO *et al.*, 2013). Apesar de todos os testes rejeitarem a hipótese nula, apenas as estratégias 3, 4, 7 e 8 foram melhores que o resultado obtido com a função de avaliação otimizada por algoritmos genéticos. No Quadro 14 mostram-se as simulações entre as estratégias definidas nesta dissertação, com a análise Chi-quadrado:

No Quadro 14, observa-se que alguns confrontos não apresentaram diferenças significativas em relação à quantidade de vitórias esperadas, apresentando valores abaixo do $\chi_c{}^2$ (tabelado), 10,827. Dentre esses confrontos pode-se citar o confronto entre as estratégias 1 e 2, 1 e 5 e 2 e 6. Isso deve-se ao fato de suas profundidades de busca, em cada fase de uma rodada, possuírem valores próximos.

A estratégia 8 apresentou uma diferença de vitórias estatisticamente significante em relação às demais estratégias. Percebe-se, também, valores significantes nos confrontos da estratégia 4 contra as estratégias 1, 2 e 3.

Nesta dissertação, o melhor resultado, para 5000 partidas, foi um número de vitórias igual a 72,04%. Esse resultado foi obtido pela estratégia 8, onde foi realizada a busca mais

profunda nas fases início, meio e fim de uma rodada. Esse aproveitamento foi superior ao obtido (ANTONIO *et al.*, 2013).

Quadro 14 – Simulações de jogos entre as estratégias.

Jogos realizados		Vitórias		χ^2 (chi quadrado)
		Dupla 1	Dupla 2	Dupla 1 x Dupla 2
Estratégia 8	x Estratégia 1	3056	1944	247,31
	x Estratégia 2	2776	2224	60,94
	x Estratégia 3	2634	2366	14,36
	x Estratégia 4	2619	2381	11,33
	x Estratégia 5	3026	1974	221,34
	x Estratégia 6	2871	2129	110,11
	x Estratégia 7	2652	2348	18,48
Estratégia 7	x Estratégia 1	2638	2362	15,24
	x Estratégia 2	2629	2371	13,31
	x Estratégia 3	2517	2483	0,23
	x Estratégia 4	2478	2522	0,39
	x Estratégia 5	2644	2356	16,59
	x Estratégia 6	2591	2409	6,62
Estratégia 6	x Estratégia 1	2799	2201	71,52
	x Estratégia 2	2535	2465	0,98
	x Estratégia 3	2527	2473	0,58
	x Estratégia 4	2455	2545	1,62
	x Estratégia 5	2619	2381	11,33
Estratégia 5	x Estratégia 1	2549	2451	1,92
	x Estratégia 2	2429	2571	4,03
	x Estratégia 3	2397	2603	8,49
	x Estratégia 4	2388	2612	10,04
Estratégia 4	x Estratégia 1	3015	1985	212,18
	x Estratégia 2	2855	2145	100,82
	x Estratégia 3	2641	2359	15,90
Estratégia 3	x Estratégia 1	2619	2381	11,33
	x Estratégia 2	2583	2417	5,51
Estratégia 2	x Estratégia 1	2592	2408	6,77

5.7 AVALIAÇÃO DO ALGORITMO DE BUSCA EM FUNÇÃO DO TEMPO COMPUTACIONAL (*CPU TIME*) E DO NÚMERO DE NÓS EXPANDIDOS.

Há uma forte relação entre a expansão dos nós e o tempo utilizado nos experimentos realizados. Neste trabalho os principais fatores que acentuam essa relação são: execução de

jogadas, armazenamento de estados de jogo e a avaliação estática dos nós de fronteiras. A partir disso, faz-se necessário verificar o tempo utilizado pelo computador para a execução da tarefa de busca e o número de nós expandidos do algoritmo *expectiminimax*. Apresenta-se nas Figuras 20 e 21 esses parâmetros em função da profundidade em escala logarítmica. Em ambos os gráficos, no eixo x, em escala linear, são mostradas as profundidades de busca para a fase inicial do jogo. O eixo y do gráfico mostrado na Figura 20 representa o número de nós expandidos, enquanto que o eixo y do gráfico mostrado na Figura 21 representa o tempo gasto para a execução da tarefa de busca. Os valores mostrados nesses eixos, para cada profundidade de busca, representam valores médios de todas as buscas da respectiva profundidade, realizadas em 5000 partidas.

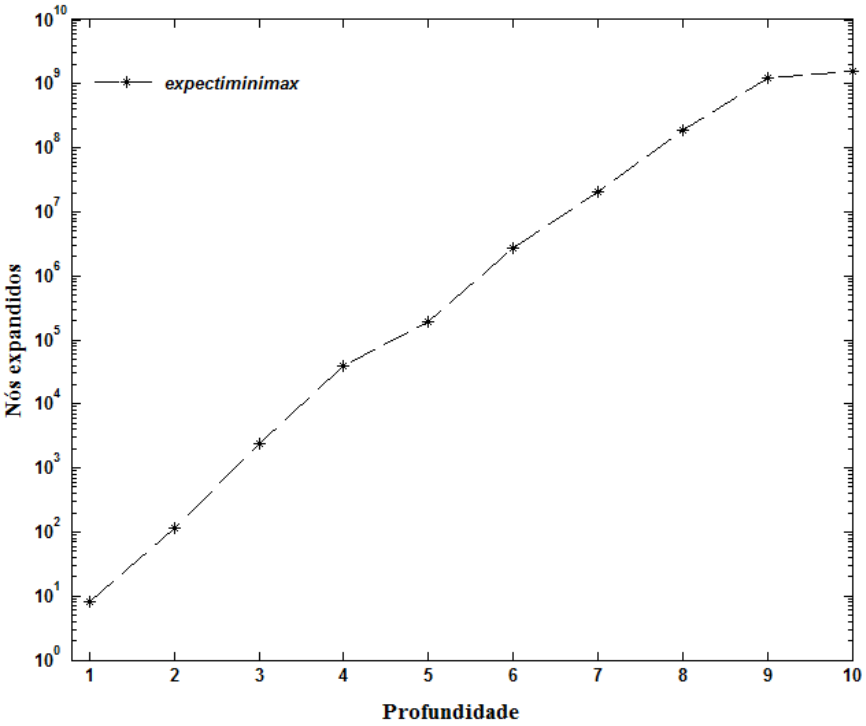


Figura 20 – Número de nós expandidos em função da profundidade da busca.

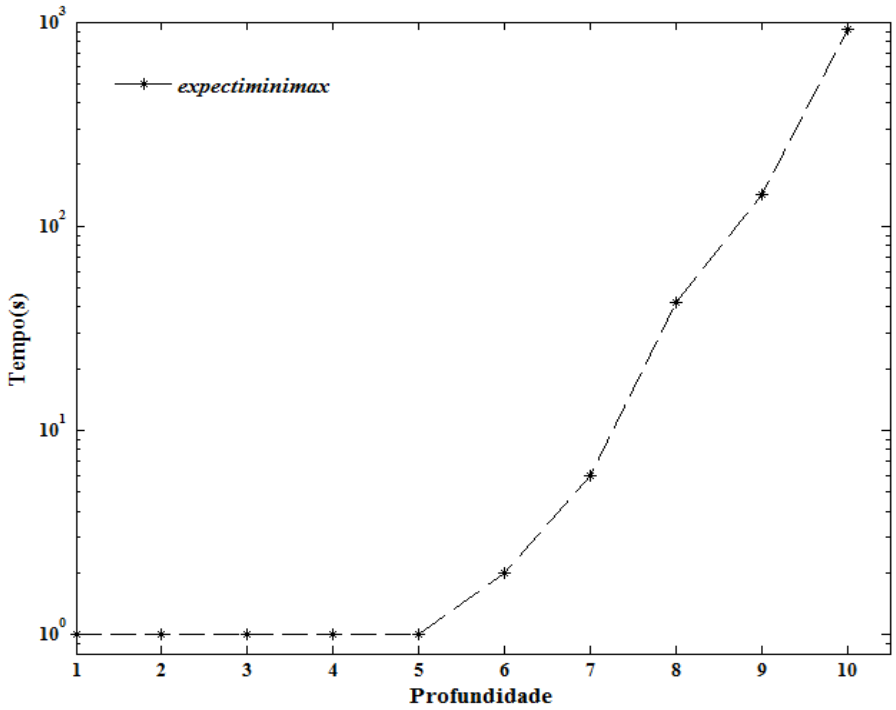


Figura 21– Gráfico mostrando o tempo médio gasto pelo computador para executar a tarefa de busca *expectiminimax* em uma simulação de 5000 partidas em função da profundidade da busca utilizada na fase inicial do jogo.

A partir da análise gráfica verificou-se que, para a profundidade 10 e para 5000 partidas, obteve-se, em média, 10^9 nós expandidos e foi consumido em torno de 10^3 segundos. Esse desempenho é satisfatório quando comparado com o trabalho de Hauk (2004), que, para o jogo de gamão, realizando um experimento à profundidade de busca 5, obteve em torno de 10^8 nós expandidos e tempo de execução acima de 1000 segundos em 500 posições diferentes do jogo.

6 CONCLUSÃO

Neste trabalho, foi apresentando um agente para o jogo de dominó de 4 pontas baseado no algoritmo *expectiminimax* com busca parcial. Para a escolha da melhor jogada, foram avaliadas oito estratégias, que se diferenciam pela profundidade da busca realizada em cada uma das fases de uma rodada. As quatro primeiras realizam uma busca rasa na fase final de cada rodada do jogo, enquanto que, as quatro últimas estratégias, utilizam todo o potencial de exploração da fase final de cada rodada do jogo. O maior potencial de exploração para as estratégias definidas foi de 10 níveis. Esse potencial de exploração é maior quando comparado com o trabalho de Hauk (2004), que conseguiu explorar, para o jogo de gamão, apenas 5 níveis de busca.

A partir dos experimentos da Seção 5.2, comprovou-se que a divisão de uma rodada em fases ajuda a atenuar o efeito de horizonte gerado no processo de busca em profundidade parcial. Os resultados obtidos mostraram que o número de vitórias aumenta com a profundidade da busca em cada fase.

Um ponto importante e crucial para o bom desempenho do algoritmo de busca foi a modelagem dos eventos de chance. No jogo de gamão, que também é um jogo que contém eventos aleatórios, as probabilidades associadas aos eventos de chance assumem um valor constante, sem a necessidade de sua modelagem. Para o jogo de dominó de 4 pontas, no entanto, essa probabilidade varia e o seu cálculo depende do armazenamento de informações da mesa de jogo e dos participantes em cada etapa do jogo. Essa modelagem foi possível graças à utilização de conhecimentos relativos à probabilidade condicional e a eventos independentes. Essa modelagem representou um grande desafio, devido ao fato do processo de busca envolvido ser mais complexo do que o algoritmo *expectiminimax* quando empregado em jogos de informação completa com dois jogadores, e por ser praticado por duas duplas, e cada jogador apenas conhecer as pedras que estão em suas mãos.

A árvore de jogo criada inclui nós MAX, que representaram os jogadores da dupla que utiliza o agente inteligente, os nós MIN referentes à dupla adversária, além dos nós CHANCE que recebem a soma da utilidade dos eventos de chance multiplicados pela probabilidade de sua ocorrência. Essa estrutura da árvore fez com que as complexidades da árvore de jogo e do espaço de estados fossem maiores do que as verificadas em um processo de busca *minimax* convencional. Apesar disso, a complexidade da árvore de jogo e a complexidade no espaço de estados para o jogo de dominó de 4 pontas são menores que alguns jogos como o xadrez, o xadrez chinês e o *stratego*.

Os resultados dos confrontos foram analisados através de testes de significância, a fim de verificar se o número de vitórias obtidas seria significante em relação à hipótese de nulidade de 2500 vitórias. Nas disputas contra uma dupla que utilizava a estratégia básica de jogo, todas as estratégias tiveram desempenho superior à média de vitórias esperadas de 2500 vitórias, porém apenas as estratégias 3, 4, 7 e 8 foram superiores ao melhor resultado obtido em (ANTONIO *et al.*, 2013).

6.1 TRABALHOS FUTUROS

Para a redução do número de nós visitados pelo algoritmo *expectiminimax* na árvore de jogo do dominó de 4 pontas, bem como a melhoria no processo de decisão, sugere-se a aplicação de algumas técnicas, tais como:

- Aplicação dos algoritmos *Star1* e *Star2*, (SCHADD; WINANDS; UITERWIJK, 2009), que são utilizados para realizar poda em árvore de jogos com elementos de chance;

- Utilizar funções de avaliações com a inclusão de termos referentes ao estado corrente do jogo, por exemplo, a inclusão de uma pontuação representando a importância da pedra para o jogador;
- Trocar a estratégia do adversário, que é baseada em soluções ótimas por uma modelagem da estratégia do oponente, a fim de melhorar as decisões do agente inteligente em situações inesperadas.

REFERÊNCIAS BIBLIOGRÁFICAS

ANTONIO, N.S.; FILHO, C.F.F.C.; COSTA, M.G.F.; PADILLA, R. **Optimization of an evaluation function of the 4-sided dominoes game using a genetic algorithm**. Transactions on Computational Intelligence and AI in Games, vol. 5, no.1, p.33-43, March 2013.

ARTS, A. F. C. **Competitive Play in Stratego**. 2010. 56 p. Dissertação (Mestrado) - Faculdade de humanidades e ciências, Universidade de Maastricht, Maastricht, Holanda. 2010.

BILINGS, D.; BOWLING, M.; BURCH, N.; DAVIDSON, A.; HOLTE, R.; SCHAEFFER, J.; SCHAUENBERG, T.; SZAFRON, D. **Game tree search with adaptation in stochastic imperfect information games**. Computers and Games. Springer – Verlag, 2004.

BRIESEMEISTER, R. **Analysis and Implementation of the Game OnTop**. 2009. 74 p. Dissertação (Mestrado) – Departamento de Engenharia do Conhecimento, Maastricht, Holanda.

CAMPBELL,M.S.; MARSLAND, T.A. **A comparison of *minimax* tree search algorithms**. Artificial Intelligence, vol. 20, no. 4, p. 347–367, 1983.

CONGPIN, Z.; JINLING, C. U. I. **Improved Alpha-Beta Pruning of Heuristic Search in Game-Playing Tree**. In: Congress on Computer Science and Information Engineering, 2009 WRI World, p.672-674.

CRUZ, A.R da ; GUIMARAES, F.G.; TAKAHASHI, R.H.C. **Comparing Strategies to Play a 2-Sided Dominoes Game**. In: Congress on Computational Intelligence and 11th Brazilian Congress on Computational Intelligence (BRICS-CCI & CBIC), 2013 BRICS, p.310-316, 8-11 Sept. 2013.

GARZA, A. G. de S. **Evaluating Individual Player Strategies in a Collaborative Incomplete-Information Agent-Based Game Playing Environment**. In: Symposium on Computational Intelligence and Games, 2006, p. 211-216.

DÍEZ, S.G.; LAFORGE, J.; SAERENS. M. **R*minimax*: An Optimally Randomized *minimax* Algorithm**. In: IEEE Transactions on Cybernetics, vol.43, no.1, p.385-393, Feb. 2013.

DUARTE, M. **Dominó**. In: Guia dos Curiosos, 2011. Disponível em: < http://guiadoscuriosos.com.br/blog/tag/domino//>. Acesso em: 31 de maio de 2015.

FLOOD, M. M. **Some Experimental Games**. Research memorandum RM-789.Santa Monica, Calif.: RAND Corporation, 1952.

GOLDBERG, D. E**. Genetic algorithms in search, optimization, and machine learning**. First edition. Boston: Addison-Wesley,1989.

HAUK, T., **Search in trees with chance nodes**. 2004. 85 p. Dissertação (Mestrado) - Departamento de Ciência da computação, Universidade de Alberta, Edmonton, Canada, 2004.

HART, T.; EDWARDS, D. **The Tree Prune (TP) algorithm**, no. 30, MIT.Artificial Intelligence Project, Cambridge MA. 449, 1961.

HEULE, M.J.H.; ROTHKRANTS, L.J.M. Solving Games - **Dependence of Applicable Solving Procedures**. Journal Science of Computer Programming, vol. 67, no. 1, p. 105–124, 2007.

HONG, T. P.; HUANG, K. Y.; LIN, W. Y. **A genetic *minimax* game-playing strategy**. In: IEEE World Congress on Computational Intelligence, 1998, Alaska, USA. Evolutionary Computation Proceedings, 1998, p.690- 694.

How to Play Dominoes. Disponível em: < http://howtoplaydominoes.org/tag/dominoes/>. Acesso em: 19 de novembro de 2014.

KNUTH, D.; MOORE, R. **An Analysis of Alpha-Beta Pruning**, Artificial Intelligence, vol. 6, no. 4, 1975, p. 293-326.

MICHIE, D. **Game-playing and game-learning automata**. In: L. Fox (ed.), Advances in Programming and Non-Numerical Computation, 1966, p. 183-200.

MYERS, M.M. **Outperforming Game Theoretic Play with Opponent Modeling in Two Player Dominoes**. 2014. 88 p. Dissertação (Mestrado) – Departamento de engenharia elétrica e computação, Instituto de tecnologia da força aérea, Ohio, United States, 2014.

MORIARTY, D.E.; MIKKULAINEN, R. **Improving game-tree search with evolutionary neural networks**. In: IEEE World Congress on Computational Intelligence. Proceedings of the First IEEE Conference on Evolutionary Computation, vol. 1, p. 496 – 501, 27-29 Jun 1994.

NASH, J. F. **The Bargaining Problem Econometrica**, Vol. 18, 1950, p. 155-162.

NEUMANN, J. V.; MORGENSTERN, O. **Theory of Games and Economic Behavior**. Princeton University Press, 1944.

RUSSEL, S.; NORVIG, P. **Inteligência Artificial**. Tradução da 2ª edição. Rio de Janeiro: Elsevier, 2004.

SCHADD, M.P.D.; WINANDS, M.H.M.; UITERWIJK, J.W.H.M**. CHANCEPROBCUT: Forward pruning in chance nodes**. In: IEEE Symposium on Computational Intelligence and Games, 2009. CIG 2009, pp.178,185, 7-10 Sept. 2009.

SMED, J.; HAKONEN, H. **Algorithms And Networking for Computer Games**, UK, Chichester: John Wiley & Sons, 2006.

SRIRAM, S.; VIJAYARANGAN, R.; RAGHURAMAN, S.; YAUN, X. **Implementing a no-loss state in the game of Tic-Tac-Toe using a customized Decision Tree Algorithm**. In: International Conference on Information and Automation, Zhuhai/Macau, China, 2009. ICIA '09. p. 1211-1216, 22-24 June 2009.

TUCKER, A. W. **A Two-Person Dilemma**: Discussion Paper, Stanford University, Stanford, CA, 1950.

WATSON, J. **Strategy: An Introduction to Game Theory**. Third Edition. San Diego: W. W. Norton, 2013.

WALPOLE, R. E.; MYERS, R. H,; MYERS, S. L.; YE, K E. **Probability and Statistics for Engineers and Scientists**. 9th Edition, Pearson, 2011.

WHITEHOUSE, D.; POWLEY, E.J.; COWLING, P.I. **Determinization and information set Monte Carlo Tree Search for the card game Dou Di Zhu**. In: *IEEE Conference on Computational Intelligence and Games (CIG)*, pp.87-94, Sept. 3 2011.

XIAN, M.; DIANLONG, Z.; QIJUN, L.; YANXIN, J.; ZHIGUANG, L. **The computer game model of contract bridge**. In: 24th Chinese Control and Decision Conference (CCDC), 2012, p.1498-1500, 23-25 May 2012.

APÊNDICE - 16ª CONFERÊNCIA ANUAL GAMEON

PROPOSING AN INTELLIGENT AGENT FOR THE FOUR-SIDED DOMINOES GAME USING THE EXPECTIMIMAX ALGORITHM

Endrews Silva
Marly Costa
Nirvana Antonio
Cicero Costa Filho
Technological and Information Center
Federal University of Amazonas
Av. General Rodrigo Otavio Jordão Ramos, 3000
Aleixo, Manaus, AM, Brazil. 69077-000
E-mail: cffcfilho@gmail.com

KEYWORDS

Four-sided Dominoes Game; Imperfect Information; Non-Deterministic; *Expectiminimax*.

ABSTRACT

This paper presents an intelligent agent to play the four-sided dominoes game, typically played in Amazonas state, Brazil. In this work, a version of the double-6 dominoes game is used, played by four players, forming two pairs. This game is of imperfect information and is non-deterministic, if we consider having pieces for any valid side as a chance event. The proposed agent is based on the *expectiminimax* technique. The methodology used for finding the best choice includes a depth-limited search with phase-related search, and a chance model based on the hidden pieces. We propose and test four strategies with different depths in the phases of the match, and tested each one against a pair playing with a basic strategy and a best strategy obtained by a genetic algorithm. In 10 simulations of 5,000 matches, the best strategy of this study obtained 72.04% of wins against a basic strategy and 58.34 % against the best strategy obtained by a genetic algorithm.

INTRODUCTION

The game of dominoes is comprised of rectangular "stones" divided into two halves. Each half has engraved dots, representing numbers, varying between 0 and N. Depending on *N* value, there are different versions of the game of dominoes. In Brazil, the most popular version of dominoes has two ends and *N=6*, resulting in 28 stones. Therefore, the stone with high numeration has six dots engraved on each half.

Dominoes is a non-deterministic game of imperfect information. This study uses probabilistic inferences to choose the best move. The need for probabilistic inference makes the task of developing an intelligent agent for dominoes games a complex one and suggests using the *expectiminimax* algorithm to determine the best move, (Michie, 1966) over the *minimax* (Neumann and Morgenstern 1944). Adding chance nodes that correspond to probabilistic events in the tree search implies an increase in its complexity by a factor multiplier O (b^d), where b is the number of chance events in each move and d is the search depth used to explore of the entire tree.

According to Russel and Norvig (2003), game trees are very complex, preventing the *minimax* and the *expectiminimax* algorithms from exploring the maximum depth of the game search tree (from the root node to the leaf nodes), when looking for the best move; therefore, some authors suggest dividing the game into phases and proceeding with a search into each phase. This technique is called a phase-related search (Smed and Hakonen 2006).

This study aims to develop an intelligent agent for the four-sided dominoes game and has the following secondary objectives:

- Proposing an intelligent agent for choosing the best move in a four-sided dominoes game using the expectiminimax algorithm;
- Using the phase-related search technique to explore the search tree of the four-sided dominoes game and evaluate the effect of changing the search depth in each phase over the number of wins of the *expectiminimax* algorithm;
- Proposing a probabilistic modeling of the chance events for the four-sided dominoes game.

RELATED WORKS

In the literature, some authors propose intelligent agents for the two-sided dominoes game. Developed for the 2-sided dominoes game, a study by Garza (2006) compares the performance of six game strategies against the basic strategy. The author proposes a static evaluation function with some terms. Each term incorporates the objectives of one strategy; nevertheless, the author provides few details about each term of the evaluation function. The strategies proposed in the study are not effective when tested against the basic strategy. The best result is 54% of wins of the selfish strategy against the basic strategy.

The study of Cruz et al. (2013) also addresses the two-sided dominoes game, obtaining better results than Garza (2006). The authors propose an evaluation function that

incorporates a probabilistic calculus, where a Boltzmann exploration is employed, with the temperature of the Boltzmann equation related to the number of stones in the table. Of the four strategies proposed by the authors, one that aims to block the adversary's game obtains the best result: 65% of wins against the basic strategy.

Concerning the four-sided dominoes game, the study of Antonio et al. (2013) describes a detailed evaluation function for choosing the best move. The evaluation function terms represent two types of information. One term refers to the number of points obtained by a player's move. The other terms refers to the game state, based on the stones already placed on the game table. A different coefficient precedes each term and determines its relevance. A genetic algorithm optimizes the value of these coefficients in order to obtain a maximum number of wins. The authors propose different evaluation functions differing from one another by the number of terms and obtaining the best results, 69.18% of wins against the basic strategy, with a maximum number of terms of the evaluation function.

DOMINOES

The dominoes game version of this study comprises 28 stones. People in Amazonas state, in northern Brazil, usually play this game. The rules of this game are listed below. These rules assume two pairs playing the game:

1. Initially, each player receives seven stones. Partners of the same pair place one in front of the other. Figure 1 illustrates the four ends of a dominoes game and the player's positions on the table. Player P_0 is partners with player P_2, and players P_1 and P_3 are opponents;
2. A game match is comprised of several rounds. The objective of each player pair is to place all the stones in their hand onto the table;
3. In the first round of the game, the player with the double-six stone, 6-6, places it on the table. The following moves occur counterclockwise. Each player tries to connect a stone to one of the ends of a stone on the table with the same number of dots.
4. From the second round onwards the player who ends the previous round starts the second round with any double-N stone;
5. Each pair scores a single count. A match ends when one of the pairs reaches a count of 200 points at the end of a round;
6. In a move, a player can add 5, 10, …50 points to the count of its pair. This sequence shows the ways of accumulating points:
   - P_1: After a player's turn, if the sum of the dots in all four game ends is a multiple of 5, the player adds this multiple to the count of its pair. Figure 1 illustrates a situation where a player adds 10 points to the count of their pair;
   - P_2: When a player does not have stones to add to any of the four game ends, the adversary pair adds 20 points to its count. Figure 2 illustrates a game situation where the player shown does not have stones to move in any game end;
   - P_3: When a player, before moving a stone, declares that no players will move any stone after their move, and this actually occurs, they add 50 points to the count of their pair. This move is traditionally called a "rooster";
   - P_4: In the end of a round, if a player finishes the round with a double-N stone, they add 20 points to the count of their pair;
   - P_5: If a player finishes a round, the sum of dots in the stones owned by the adversary pair is rounded down to the nearest multiple of 5 and added to his pair count. Traditionally players call these points "garage". Figure 3 shows, a hypothetical scenario with the stones held by the adversary pair, when a round finishes. As there are 14 dots, the garage is 10 (the result of rounding down 14 to 10, the nearest multiple of 5).

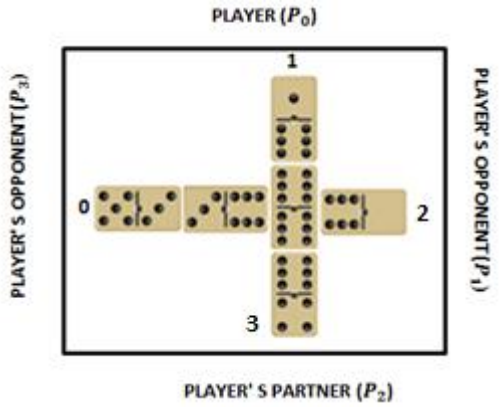


Figure 22: Example of a game situation where, after a player takes a turn, 10 points are added to the count of their pair.

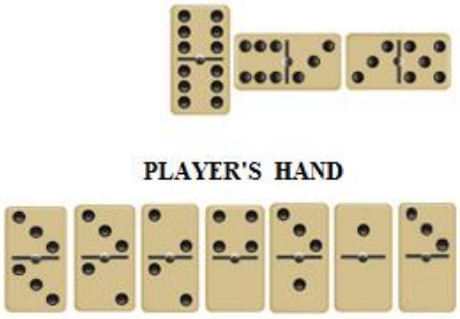


Figure 23: Example of a game situation showing stones in the hand of a player, when there are no stones to add to any of the ends of the game.

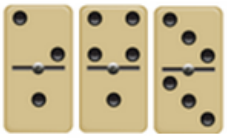


Figure 24: Example of a garage totaling 10 points.

METHODOLOGY

The depth search algorithm used in this study for the four-sided dominoes game uses a limited *expectiminimax* search. In this study, each search depth depends on the game phase. This study proposes dividing the dominoes game into three phases. In each phase, the search depth assumes a constant value.

The *expectiminimax* tree search for the four-sided dominoes game has three types of nodes: MAX, MIN and CHANCE. . Figure 4 shows an example of a search tree with a depth of 2. A MAX node corresponds to one of the pair of players who uses the *expectiminimax* algorithm.

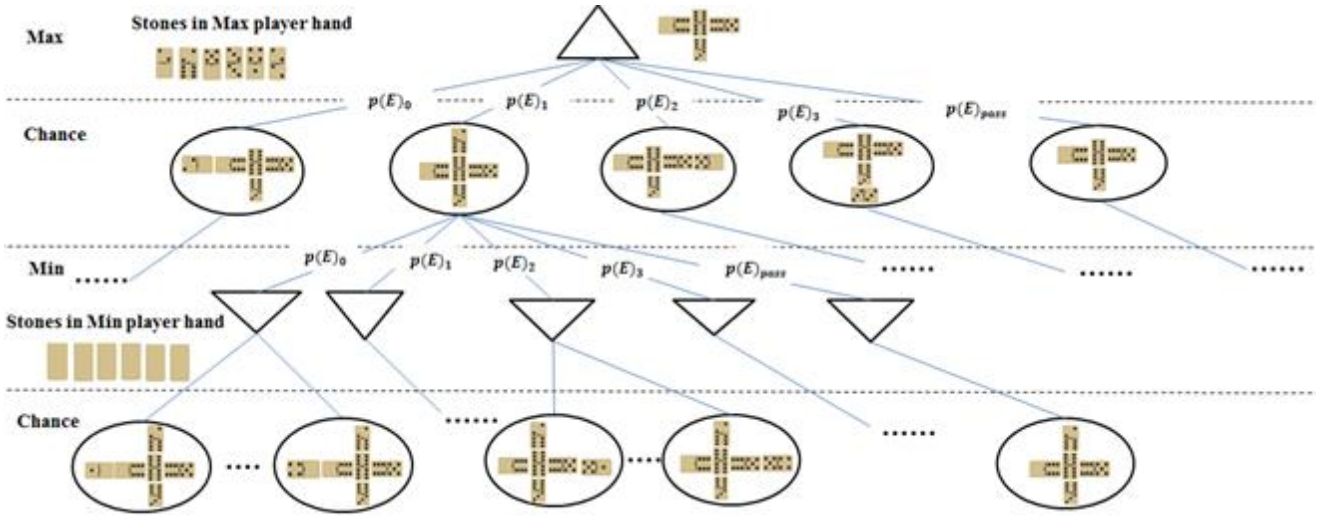


Figure 25: Search tree for the four-sided dominoes game with a depth of 2. $\boldsymbol{p(E)_0}$, $\boldsymbol{p(E)_1}$, $\boldsymbol{p(E)_2}$ and $\boldsymbol{p(E)_3}$ represent the probabilities of player "MAX" or "MIN" adding a stone to a game end 0, 1, 2 and 3. $\boldsymbol{p(E)_{no-stone}}$ represents the probability of players "MAX" or "MIN" having no stone to add to any end of the game.

In Figure 4, the upper triangle corresponds to a MAX player. A MIN node corresponds to one of the players of the adversary pair (the pair that does not use the *expectiminimax* algorithm). In Figure 4 the inverted triangles represent the MIN player. The circles represent a CHANCE node; A CHANCE node has several possible moves that are represented by probabilities: $p(E)_0$ – add a stone in to a game end 0; $p(E)_1$- add a stone in to a game end 1; $p(E)_2$- add a stone in to a game end 2; $p(E)_3$- move a stone in the game end 3; $p(E)_{no\ stone}$ – have no stone to add to any end of the game. Algorithm 1 shows the *expectiminimax* algorithm used for exploring this tree in the four-sided dominoes game .

Algorithm 1: *expectiminimax* algorithm for the four-sided dominoes game.

Function Expectiminimax (node, depth, State)

1) **IF** depth=0
2) **return** ExpectPairScore–OpponentPairScore
3) **ELSE IF** node is MIN node
4) BestScore ← + ∞
5) **FOR EACH** possible action
6) Score=Expectiminimax(Chance,depth-1)
7) BestScore←min**(**BestScore,Score)
8) Undo move
9) **ElSE IF** node is Max node
10) BestScore ← - ∞
11) **FOR EACH** possible action
12) Score=Expectiminimax(Chance,depth-1)
13) BestScore←max**(**BestScore,Score)
14) Undo move
15) **ElSE** node is a CHANCE event
16) BestScore ← 0
17) **FOR EACH** available end of the table
18) Score=Expectiminimax(Child,depth-1)
19) BestScore ← BestScore+(Prob(i)*Score
20) Return BestScore

As in the original *expectiminimax* algorithm, the one just shown for the four-sided dominoes game is a recursive procedure that uses two auxiliary procedures: the move generator and the static evaluation. The *expectiminimax* algorithm must be provided with three parameters: an identification of the player who will take a turn (node), the depth search (depth) and the current configuration of the game (state).

Complexity of four-sided dominoes game tree

The number of leaf nodes usually measures the complexity of a game tree. For an imperfect information game, this number depends on the ramification factor of the tree (number of possible moves available for each player), the search depth and the number of options of a CHANCE node. For the four-sided dominoes game, the ramification factor does not present a linear behavior, as in chess and Stratego games, because its calculation depends on the number of stones owned by the players. It is possible to fix an upper limit to the ramification factor, considering the following worst-case scenario: in the second move of the game, the player who performs this move owns all stones with numeration num_6 in one of the sides. This corresponds to a maximum value of the ramification factor, which is equal to six. Figure 5 shows these six possible moves.

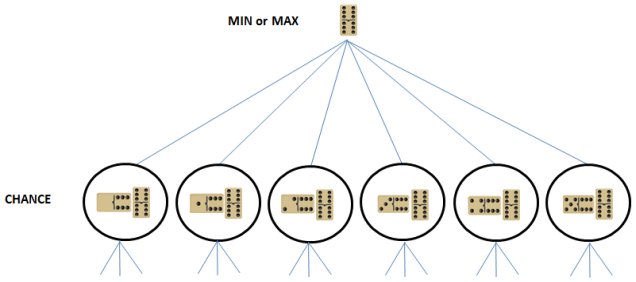


Figure 5: Maximum value for the ramification factor: a hypothetical situation where, in the second move of the game, the player who performs the move owns all stones with numeration num_6

As stated before, the number of options of CHANCE nodes is equal to 5. Figure 6 shows these options. The last parameter needed to calculate the four-sided dominoes game tree complexity is the search depth. This depth is equal to the maximum number of rounds of the game, which is 25.

According to Hauk (2004), equation (1) calculates the game tree complexity, C.

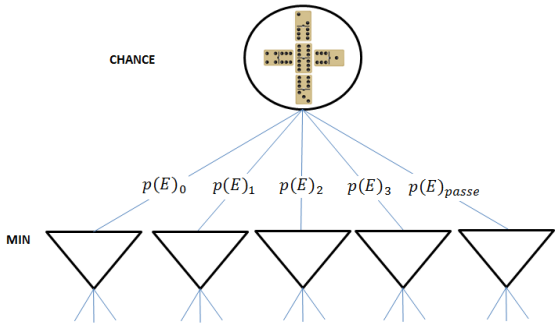


Figure 6 : Ramification factor for the CHANCE nodes.

$$C = O\left((RF)^p \times (CN)^p\right) \qquad (1)$$

Where:

RF – Ramification factor of a MAX or MIN node - 6;
CN – Number of options of a CHANCE node -5;
p – Search depth -25.

Using the values previously determined results for complexity C:

$$C = O(6^{25} \times 5^{25}) \approx O(10^{36})$$

This is the maximum number of leaves that can exist in the dominoes game tree. For comparison, Table 1 shows the game complexity of some other games (ARTS, 2010). As shown, the four-sided dominoes game complexity is only greater than the game Connect-Four.

Table 15: Game tree complexity of some games (ARTS, 2010).

Game	Game tree complexity
Trail	10^{50}
Connect-Four	10^{21}
Othello	10^{58}
Chess	10^{123}
Chinese Checkers	10^{150}
Stratego	10^{535}
Go	10^{360}

Probabilistic modeling of CHANCE nodes

According to Walpole et al. (2011), if a sample space has N elements, with each one with the same probability, they all have the same probability *1/N*, and the probability of an event *A*, with *p* elements, is given by equation (2).

$$p = \frac{n}{N} \qquad (2)$$

Calculating the probabilities $p(E)_i, 1 \leq i \leq 5$, associated with a CHANCE node requires a different calculation for the pair that uses the *expectiminimax* algorithm and for the adversary pair.

For the MAX node (a player of the pair who uses the expectiminimax algorithm), the calculus is very simple. As the stones are known by the algorithm, $p(E)_i = 1$ or $p(E)_i = 0$. For MIN node (a player of the adversary pair); nevertheless, the calculus is more complex and is explained in the sequence.

First, calculating the denominator of equation (2), N is explained. In this study, the stones not known by MAX are named hidden stones. Set_h comprises the set of hidden stones. The number of elements of Set_h is S_h. The sample space is formed by all hidden stone permutations: $S_h!$. Each permutation is an element of the sample space.

Second, calculating the numerator of equation (2), n is explained. The numeration of the game end i is num_i. The

maximum number of stones with num_i, t owned by MIN player is given by equation (3).

$$t = minimum(S_1, S_{num_i}) \quad (3)$$

Where:

S_1 - number of stones owned by player MIN.

S_{num_i}– number of stones with numeration num_i in one of the sides. .

The number n in this study is calculated as a sum of T_k terms, where $1 < k < t$. T_k represents the number of elements of the sample space where player MIN owns k stones with numeration num_i in one of the sides. As shown below, for exemplifying how T_k is determined, this study assumes that:

P_0 – MAX player;
P_1 – MIN player;
P_2 –P_0 partner;
P_3 - P_1 partner;
$Set_h = \{p_{11}, p_{12}, p_{13}, p_{AB}, p_{CD}, p_{EF}, p_{GH}\}$ (7 stones);
$num_i = 1$;
Stones with numeration $num_i - p_{11}, p_{12}, p_{13}$, ($S_{num_i}$ =3);
3 stones of Set_h held by P_1 (S_1=3);
2 stones of Set_h held by P_2 (S_2=2);
2 stones of Set_h held by P_3 (S_3=2)

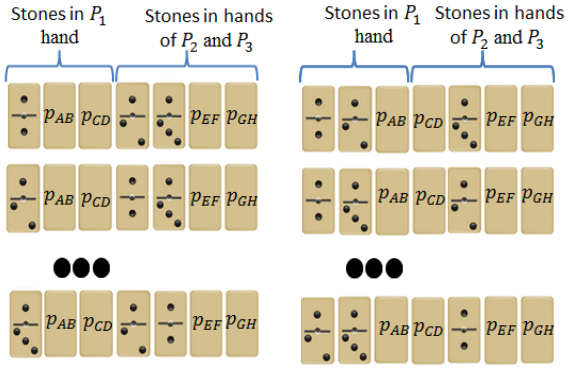
As S_{num_i} =3 and S_1=3, from equation (3), t=3. So, for calculating n we must add three terms: T_1, T_2 and T_3. Figure 7 shows some elements of the sample space with 1, 2 and 3 stones with numeration num_i owned by P_1. Before showing the expressions for T_1, T_2 and T_3, the following definitions are provided:

$$\alpha = S_h - S_{num_i} \quad (4)$$

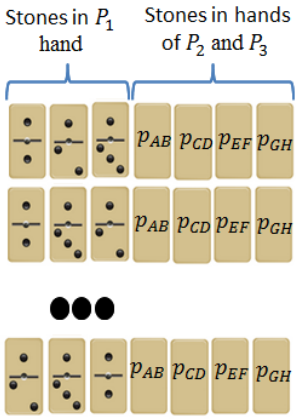
$$\beta_1 = S_1 - 1, \beta_2 = S_1 - 2 \ldots \beta_k = S_1 - k \quad (5)$$

Where:

$k \in \{1, 2, \ldots, t\}$.



(a) (b)



(c)

Figure 7: Elements of sample space: (a) examples of elements with 1 stone with numeration num_i owned by P_1; (b) examples of elements with 2 stones with numeration num_i owned by P_1; (c) examples of elements with 3 stones with numeration num_i owned by P_1.

This study calculates T_1, T_2 and T_3 using the following equations:

$$T_1 = A_{S_{num_i},1} \times C_{S_1,1} \times A_{\alpha,\beta_1} \times (S_h - S_1)! \quad (6)$$

$$T_2 = A_{S_{num_i},2} \times C_{S_1,2} \times A_{\alpha,\beta_1} \times (S_h - S_1)! \quad (7)$$

$$T_3 = A_{S_{num_i},3} \times C_{S_1,3} \times A_{\alpha,\beta_1} \times (S_h - S_1)! \quad (8)$$

Where:

$A_{j,k} - arragnements\ of\ j\ elements\ in\ groups\ of\ k$

$C_{j,k} - combinations\ of\ j\ elements\ in\ groups\ of\ k$

In a general way:

$$T_k = A_{S_{nipe_i},t} \times C_{S_1,k} \times A_{\alpha,\beta_k} \times (S_h - S_1)! \quad (10)$$

For the game situation assumed in this study, and using equations (6), (7) and (8), the calculus of T_1, T_2, T_3, n and $p(E)_i$ results: $T_1 = A_{3,1} \times C_{3,1} \times A_{4,2} \times 4! = 2592$, $T_2 = A_{3,2} \times C_{3,2} \times A_{4,1} \times 4! = 1728$ and $T_3 = A_{3,3} \times C_{3,3} \times A_{4,0} \times 4! = 144$. n= $T_1 + T_2 + T_3 = 4464$. The value of n=7!=5040 and $p(E)_i = \frac{4464}{5040} = 0{,}89$.

Game strategies

The depth search of the *expectiminimax* algorithm is limited according to the number of phases into which a game round is divided. In this study the four-sided dominoes round is divided into three phases: initial, middle, and end.

Aiming to define the size of each phase, this study conducts the following experiment: 5,000 matches between pairs using the basic strategy, resulting in 10,282 rounds. In each round, the number of moves was measured. Figure 8 shows the results of this experiment.

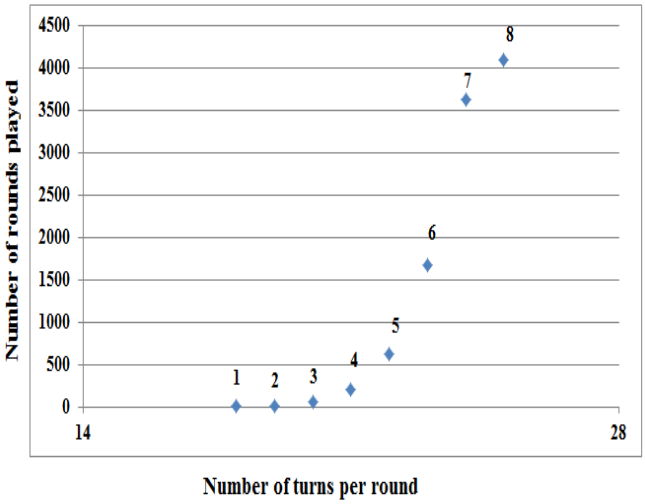


Figure 8: Number of rounds versus number of moves/round in an experiment with 5,000 matches (10282 rounds).

In Figure 8 the vertical axis represents the number of rounds, while the horizontal axis represents the number of moves/round. Point 1, for example, shows that only 3 rounds finished with 18 moves, while point 8 shows that 4,103 rounds finished with 25 moves.

Briesemeister (2009) estimated the number of moves/round of the game OnTop, simulating 1,400 matches. A mean value of 31.36 moves is reported.

The mean value of moves of a game can be calculated using equation (11).

$$\bar{p} = \frac{p_1 x_1 + p_2 x_2 + \cdots + p_n x_n}{p_1 + p_2 + \cdots + p_n} = \frac{\sum_{i=1}^{n} p_i x_i}{\sum_{i=1}^{n} p_i} \qquad (11)$$

Where:

x_i – quantity of moves/round;

p_i – number of rounds with x_i moves in the experiment.

Applying the point values of Figure 8 in equation (11) results in a mean value of moves of 24.03. Therefore, there are approximately 24 moves/round. This study calculates the number of moves in each phase of the four-sided dominoes game dividing this value by 3, as shown in Table 2.

This study tests different search depths in each phase of a round and proposes four strategies, depending on values of these depths. Table 3 shows these strategies.

Table 2: Phases for a match round of the four-sided dominoes game proposed in this study.

Match phase	Round
Initial	1~8
Middle	9~16
Final	>16

Table 3: Strategies defined for the four-sided dominoes game, varying the search depth in each round.

Strategies for *expectiminimax* agent	Depth search in round phase		
	Initial	Middle	Final
First	5	8	9
Second	8	5	9
Third	9	9	9
Fourth	10	10	9

The first and second strategies use low values for depth search in the first and middle phases, while the third and fourth strategies use higher values for depth searches in these two phases. In all strategies, the depth search of the last phase was fixed in a maximum value of 9. For depth searches above 10, this study verifies that the *expectiminimax* algorithm the algorithm gets stuck in recursion.

Results and discussion

All the experiments were done on a laptop with an Intel Core i5 @ 2.5 GHz processor, with a Windows operating system 8.1. The simulations used the Eclipe IDE. This study divides the results into four groups.

The first group of results, with the aim of evaluating the number of wins of the *expectiminimax* algorithm against the basic strategy, proposes two simple experiments. These experiments change the depth search of initial and middle phase, while fixing the depth search of final phase. Both experiments performed 5,000 matches.

In the first experiment, the depth search of the initial and final phases were fixed at 5 and 9, respectively, and the length of middle phase ranged from 1 to 7. Figure 9 shows the number of wins of the pair that uses the *expectiminimax* algorithm (vertical axis) based on depth search of the middle phase (horizontal axis). The best result for the pair that uses the *expectiminimax* algorithm is 2985 wins. This occurs for a depth search of 7 in the middle phase.

In the second experiment, the depth search of the middle and final phases were fixed at 5 and 9, respectively, and the depth search of the initial phase ranged from 1 to 7. Figure 10 shows the number of wins for the pair that used the *expectiminimax* algorithm (vertical axis) based on depth search of the initial phase (horizontal axis). The best result for the pair that used the expectiminimax algorithm was 2,985 wins. This occurs for a depth search of 7 in the initial phase.

These two experiments suggest that the number of wins of a pair that uses the *expectiminimax* algorithm against a basic strategy is proportional to the depth search of the initial and middle phases.

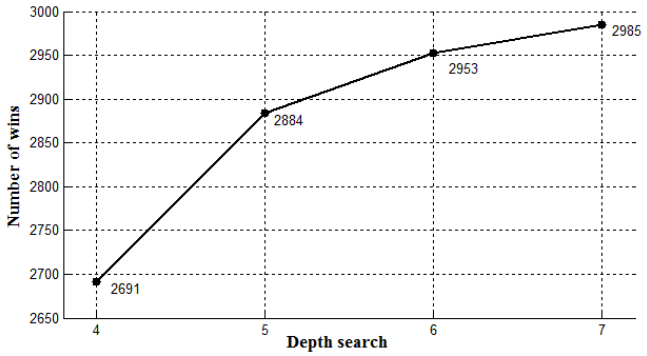


Figure 9: Number of wins for a pair using the *expectiminimax* algorithm against a basic strategy, with the following phase search depths: initial = 5; middle = 1, .., 7; final = 9.

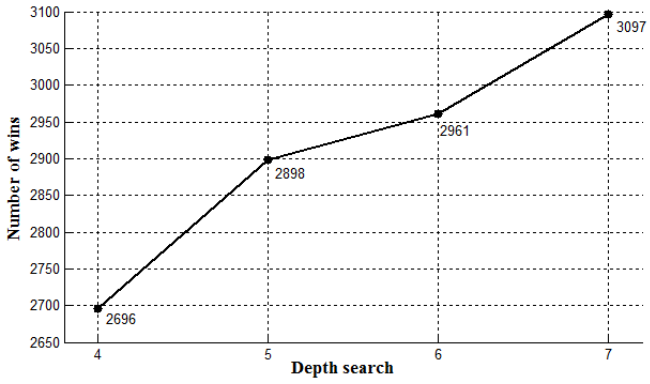


Figure 10: Number of wins for a pair using the *expectiminimax* algorithm against a basic strategy, with the following phase search depths: initial = 1…7; middle = 5; final = 9.

The second group of results evaluates the performance of the strategies defined in Table 3 against a basic strategy and against the best strategy defined by Antonio et al. (2013), hereafter-called strategy GA. Each test uses 10 simulations with 5,000 matches. The mean number of wins with the corresponding standard deviation are registered. This study proposes using the t-Student hypothesis test to evaluate the statistical significance of the results. The value of each t-Student test, *T*, is compared with a critical value t_c. The null hypothesis, $H_0 = 2{,}500$ wins, is rejected if $T \geq tc$ or $T \leq -tc$. A significance level of 99% is employed, with 9 freedom degrees. This result in a critical value $tc = 2.81$. Table 4 shows the results obtained in this step.

According to Table 4, all the strategies defined previously in this study obtained statistically significant results against the basic strategy, with mean victory values above 2,500. The fourth strategy obtained the largest number of wins and the lowest standard deviation, followed by strategy 3. Only strategies 3 and 4 obtained statistically significant results against the GA strategy. Again, the fourth strategy achieved the highest number of wins and the lowest standard deviation, followed by strategy 3.

In the third group of results, the aim is to evaluate the performance of each strategy playing against the other. Table 5 shows the results. Each test uses 10 simulations with 5000 matches, and the mean number of wins with the corresponding standard deviation are registered. Strategy 4 obtained statistically significant results against the other strategies. Perhaps, one possible reason is that this strategy uses deeper searches than other strategies in all four-sided dominoes game phases defined in Table 2.

Strategy 3 presents statistically significant results when playing against strategies 1 and 2. Strategy 2 provided a statistically significant result only against strategy 1. Strategy 1 did not obtain statistically significant results against other strategies. Perhaps, the possible reason is that this strategy uses shallower searches than other strategies in all four-sided dominoes game phases defined in Table 2.

Table 4: Performance of each strategy defined in this study against the basic strategy and against the GA strategy (Antonio et al., 2013): mean number of wins in 10 simulations with 5,000 matches.

Strategies	Number of wins f		H_0: $\bar{f} = 2500$
	$\bar{f}$	σ	T
First x Basic	3116.1	8.53	228
Second x Basic	3225.6	7.23	317
Third x Basic	3535.6	7.12	460
Fourth x Basic	3597.1	5.36	647
First x Ga	2331.6	5.21	-102.19
Second x Ga	2375.8	5.35	-73.41
Third x Ga	2795.1	5.92	157.76
Fourth x Ga	2909.3	5.08	254.87

Table 5: Performance of each strategy defined in this study playing against the other: mean number of wins in 10 simulations with 5,000 matches.

Tests		Fitness function f		H_0: $\bar{f} = 2500$
		$\bar{f}$	σ	T
First	Second	2378.6	5.58	-68.78
	Third	2324.8	6.39	-86.69
	Fourth	1952.2	7.24	-239.31
Second	First	2589.1	8.77	32.11
	Third	2383.8	8.22	-44.72
	Fourth	2101.8	8.93	-141.02
Third	First	2629.5	9.59	42.68
	Second	2591.8	7.18	40.45
	Fourth	2323.1	10.25	-54.60
Fourth	First	3017.7	8.45	193.82
	Second	2864.6	7.07	162.98
	Third	2645.6	5.87	78.40

In the fourth group of results, the aim is to evaluate the best performance (maximum number of wins) of each strategy defined in this study against the basic strategy and against the GA strategy. A total of 5,000 matches were conducted in each test and the maximum number of wins was registered. Table 6 presents the results. The statistical significance of results were evaluated using a Chi-square test, with a significance level of 99% and with 1 degree of freedom, resulting in a critical value of 10.83 ($\chi^2 = 10.83$).

Table 6**:** Performance of each strategy defined in this study against the basic strategy and against the GA strategy (Antonio et al., 2013): maximum number of wins in 5,000 matches.

Simulations			Number of wins		χ^2 (chi-square)
Pair 1	x	Pair 2	Pair 1	Pair 2	Pair 1 x Pair 2
First	x	Basic	3127	1873	314.50
Second	x	Basic	3235	1765	432.18
Third	x	Basic	3543	1457	870.28
Fourth	x	Basic	3602	1398	971.52
First	x	GA	2342	2658	19.97
Second	x	GA	2381	2619	11.33
Third	x	GA	2805	2195	74.42
Fourth	x	GA	2917	2083	139.11

Table 6 shows that all strategies achieved statistically significant results against the basic strategy. The best result is 72.04% of wins. This result is higher than the one obtained by Antonio et al. (2013), which was 69.18%. Only strategies 3 and 4 defined in this study achieved statistically significant values against the best results presented by Antonio et al. (2013).

CONCLUSION

This study presents an intelligent agent for the four-sided dominoes game based on the *expectiminimax* algorithm. The study evaluates four different strategies. These strategies differ from one another by the depth search in each phase of a round.

The aim of defining strategies with different depth searches in each phase of a round evaluated its effect on the number of wins of a playing pair. Two strategies are defined with a shallow search depth in initial and middle round phases, while the other two are defined with a high search depth in initial and middle round phases.

The best results in terms of the maximum number of wins in 5,000 matches is 72.04%. These values are higher than those obtained by Antonio et al. (2013), 69.18%, for the four-sided dominoes game and higher than those obtained by Cruz et al. (2013), 65%, for the 2- sided dominoes game.

An important contribution of this study is the probabilistic modeling of the chance events of the four-sided dominoes game. Differing from backgammon, the calculated probability for a chance event changes with each round, because its value depends on the stones placed on the table.

For future works, we propose reducing the number of visited nodes in the tree search of the *expectiminimax* algorithm and improving the decision process of choosing the best move, through the following actions:

1) Employ the *Star1* and *Star2* algorithms, (Schadd et al. 2009) for minimizing the number of nodes of the search tree;
2) Employ evaluation functions that incorporate the current game state to decide the best move;
3) Employ different strategies for the adversary pair, to evaluate the robustness of the expectiminimax method.

REFERENCES

Antonio, N.S.; Costa Filho, C.F.F.; Costa, M.G.F. 2013. "Optimization of an Evaluation Function of the Four-Sided Dominos Game Using a Genetic Algorithm," *Computational Intelligence and AI in Games, IEEE Transactions on* , vol.5, no.1, pp.33,43.

Arts, A. F. C. 2010. Competitive Play in Stratego. 56 p. M.Sc. thesis (Degree of Master of Science of Artificial Intelligence) - Faculty of Humanities and Sciences of Maastricht University, Maastricht, The Netherlands.

Briesemeister, R. 2009. Analysis and Implementation of the Game OnTop. 74 p. M.Sc. thesis (Degree of Master of Science of Artificial Intelligence) - Maastricht University, Dept. of Knowledge Engineering, Maastricht, Netherlands.

Cruz, A.R da .; Guimaraes, F.G.; Takahashi, R.H.C. 2013. Comparing Strategies to Play a 2-Sided Dominoes Game. In: *Computational Intelligence and 11th Brazilian Congress on Computational Intelligence* (BRICS-CCI & CBIC), 2013 BRICS Congress on , p.310-316, 8-11.

Garza, A. G. de S. 2006. Evaluating Individual Player Strategies in a Collaborative Incomplete-Information Agent-Based Game Playing Environment. In: *Symposium on Computational Intelligence and Games*, p. 211-216.

Hauk, T. 2004. Search in trees with chance nodes. 85 p. M.Sc. thesis (Degree of Master of Science) - Computing science department, university of Alberta, Edmonton, Canada.

Myers, M.M. 2014. Outperforming Game Theoretic Play with Opponent Modeling in Two Player Dominoes. 88 p. M.Sc. thesis (Degree of Master of Science in Electrical Engineering) - Air Force Institute of Technology University, Ohio, United states.

Michie, D. 1966. Game-playing and game-learning automata. In: L. Fox (ed.), Advances in: *Programming and Non-Numerical Computation*. p. 183-200.

Neumann, J. V.; Morgenstern, O. 1944. Theory of Games and Economic Behavior. Princeton University Press.

Ruseel, S. and Norvig, P. 2003. A*rtificial Intelligence: A Modern Approach 2/e*. Englewood Cliffs, NJ, USA: Prentice Hall. .

Schadd, M.P.D.; Winands, M.H.M.; Uiterwijk J.W.H.M. 2009. "CHANCEPROBCUT: Forward pruning in chance nodes". In : *Computational Intelligence and Games*, 2009. CIG 2009. IEEE Symposium, pp.178, 185.

Smed, J. and Hakonen, H. 2006. Algorithms And Networking for Computer Games, UK, Chichester: John Wiley & Sons.

Walpole, R. E.; Myers, R. H,; Myers, S. L.; YE, K E. 2011. Probability and Statistics for Engineers and Scientists. 9th Edition, Pearson.

AUTHOR BIOGRAPHY

ENDREWS SZNYDER SOUZA DA SILVA studied electrical engineering at University of Amazonas Manaus, Amazonas, Brazil, and obtained his degrees in 2013. After graduating, he moved in 2013 to Technological and information Center of Amazonas Federal University, Manaus, Brazil.