Comparative Analysis of Evaluation Functions for Minimax with Alpha-Beta Pruning Applied to the Jaguar Game

Felipe G. L. do Nascimento, Elloá B. Guedes

[1]Grupo de Pesquisas em Sistemas Inteligentes
Laboratório de Sistemas Inteligentes
Escola Superior de Tecnologia
Universidade do Estado do Amazonas
Av. Darcy Vargas, 1200 – Manaus – Amazonas

{fgldn.eng, ebgcosta}@uea.edu.br

Abstract. *Jaguar Game is a board game from the Brazilian indigenous cultural tradition. It is played on an asymmetric board with 31 positions in which a jaguar duels against fourteen dogs and where the adversaries have different goals. In order to present preliminary results regarding the use of Artificial Intelligence techniques in this domain, the game tree complexity was estimated by a brute force approach with prunning and the resulting complexity was verified as being superior than the Checkers game. We also proposed and evaluated utility functions for the Minimax algorithm with Alpha-Beta pruning optimization. Results from simulations emphasize the challenges in designing utility functions for the dogs players.*

Resumo. *O Jogo da Onça é um jogo da tradição cultural indígena brasileira. É caracterizado por um tabuleiro assimétrico com 31 posições na qual uma onça duela contra catorze cachorros, em que há objetivos distintos para estes adversários. Com vistas a apresentar resultados preliminares do uso de técnicas da Inteligência Artificial neste domínio, estimou-se a complexidade da árvore de jogo a partir de uma simulação exaustiva com poda, em que verificou-se complexidade superior ao Jogo de Damas. Além disso, funções de utilidade foram propostas e avaliadas segundo o algoritmo Minimax sujeito à otimização Alfa-Beta. Os resultados obtidos de simulações ressaltam os desafios em projetar funções de utilidade para o jogador que representa os cachorros.*

1. Introdução

Jogos de tabuleiro fazem parte da ludicidade da civilização humana, representando uma aplicação do pensamento abstrato pelas massas, em que as habilidades nestes jogos são usualmente enaltecidas como sinal de inteligência e aprendizagem [Ghory 2004]. Os primeiros registros de estudos sistemáticos acerca de jogos de tabuleiro datam do Séc. XVII e, desde então, compreendem aspectos arqueológicos, linguísticos, filosóficos, artistísticos, psicológicos, dentre outros [Gobet et al. 2004]. Considerando o advento da Computação, o estudo dos jogos de tabuleiro foi de especial interesse da Inteligência Artificial (IA) em virtude do formalismo existente em tais jogos e do ambiente de tomada de decisão, que é complexo ainda que haja restrições relativas às regras [Yannakakis e Togelius 2018].

Do ponto de vista dos pesquisadores de IA, este domínio é interessante, pois embora os jogos de tabuleiro tenham um espaço finito de estados, as possíveis estratégias para um agente costumam ser vastas, o que resulta em um amplo espaço de busca com, normalmente, poucos exemplos viáveis, em que estes últimos representam tipicamente as soluções desejadas que levam à vitória. Em muitas ocasiões, há a impossibilidade de tomar decisões ótimas e ainda tem-se o ônus de que as más estratégias são punidas com a derrota [Yannakakis e Togelius 2018, Russell e Norvig 2016]. Apesar destas dificuldades, desenvolvimentos de soluções de IA aplicada ao jogos de tabuleiro alcançaram notoriedade mundial, a exemplo de Xadrez [Hsu 2002] e Go [Chen 2016].

Levando em conta este contexto, o presente trabalho visa apresentar os resultados preliminares do uso de técnicas de IA baseadas na exploração da árvore de busca em um jogo da tradição indígena brasileira denominado *Jogo da Onça*. Este jogo possui algumas particularidades muito específicas, tais como o fato de haver um número desigual de peças, sendo catorze cachorros versus uma onça, um tabuleiro assimétrico e tipos de jogadas e objetivos diferentes para os adversários. Considerando os desafios mencionados, este trabalho almeja primeiramente estimar a árvore do jogo e, em seguida, propor, avaliar e analisar funções de utilidade para os agentes, considerando algoritmo Minimax com otimização *Alpha-Beta Prunning*, uma estratégia tipicamente adotada para o desenvolvimento de jogadores automáticos em cenários de busca competitiva. Os resultados obtidos posicionam o referido jogo como tendo maior complexidade de árvore de jogo que Damas e ressaltam a dificuldade em propor funções de utilidade que capturem a noção de inteligência coletiva.

Para apresentar o que se propõe, este artigo está organizado como segue. Um breve contexto cultural, as características e regras do Jogo da Onça são apresentadas na Seção 2. A metodologia e os resultados para obtenção de estimativas para a árvore de jogo encontram-se na Seção 3. Em seguida, na Seção 4, são apresentadas a metodologia e os resultados obtidos da proposição e teste das funções de utilidade para os agentes do jogo. Por fim, as considerações finais e trabalhos futuros são detalhados na Seção 5.

2. Jogo da Onça

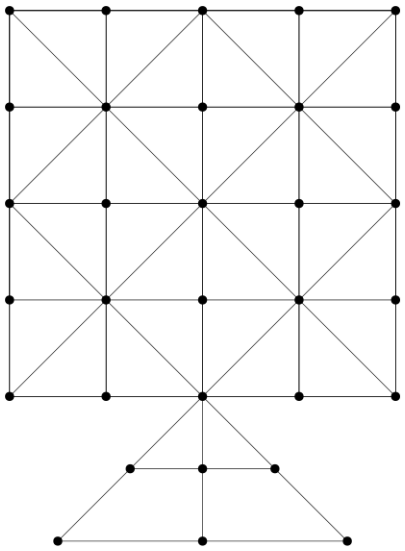
No Brasil, o Jogo da Onça faz parte da cultura de algumas tribos indígenas como os Bororos (Estado do Mato Grosso), os Manchineri (Acre), e os Guaranis (São Paulo), mas com testemunhas de sua prática também em tribos localizadas no Amazonas, Paraíba, Minas Gerais, Bahia e Paraná [Viegas et al. 2019]. Há indícios de que este jogo tenha sido introduzido pelos colonizadores e adotados por alguns grupos étnicos aqui presentes [Ferreira et al. 2008].

Na tradição indígena, o tabuleiro é desenhado na areia batida e pedras ou grãos atuam na representação das peças do jogo. O tabuleiro é composto por figuras geométricas simples, como quadrados e triângulos, dispostos conforme ilustrado na Figura 1a. As interseções, chamadas de vértices e representadas como círculos na Figura 1a, representam as casas do tabuleiro, que é composto por 31 destas. Cada casa possui duas ou mais casas adjacentes. Uma casa é dita ser adjacente a outra quando há uma aresta que as conecta. O triângulo que se externa do quadrado na parte inferior é chamado de "toca da onça".

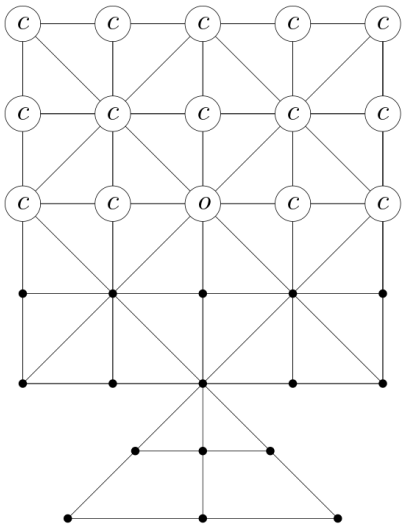
A disposição inicial das peças no tabuleiro é mostrada na Figura 1b. A partir

Figura 1: Representações do tabuleiro do Jogo da Onça. A letra 'c' denota o cachorro e a letra 'o' denota a onça.

(a) Disposição das casas no tabuleiro.



(b) Configuração inicial do jogo.



desta configuração, o jogador com a onça faz o primeiro movimento e então os jogadores se alternam a cada turno. As peças existentes, quer seja onça ou cachorro, podem se movimentar para quaisquer casas adjacentes do tabuleiro, desde que estas estejam vazias, uma casa por vez. A onça captura um cachorro de maneira análoga ao Jogo de Damas, fazendo um salto por cima do cachorro para uma casa vazia adjacente na direção do salto. Quando possível, também pode capturar vários cachorros de uma só vez com saltos duplos ou triplos,mas tal feito não é de natureza obrigatória. Este personagem vence quando consegue capturar cinco cachorros. O cachorro, por sua vez, não possui nenhum movimento de ataque, porém se faz em maior quantidade no tabuleiro. O jogador com os cachorros não pode capturar a onça, apenas mover as peças de forma a encurralá-la, deixando-a sem possibilidade de movimentos. Além das vitórias da onça ou do cachorro, é possível haver empates. Estes ocorrem quando, durante uma partida, for comprovado que uma posição se repetiu quatro vezes. É o chamado *empate pela repetição de lances*, que deve ser reclamado pelo jogador no momento em que ocorrer [Secretaria Municipal de Educação 2016].

Do ponto de vista formal da busca competitiva, o Jogo da Onça é tido como completamente observável, determinístico, multi-agente, com dinâmica baseada em turnos e de soma-zero. Por último, neste jogo, um agente conhece perfeitamente todas as ações que ocorreram anteriormente, assim como a posição de todas as peças em todos os momentos, ou seja, tem-se informação perfeita.

O Jogo da Onça tem por objetivo proporcionar, de maneira lúdica, o desenvolvimento do raciocínio lógico, dedutivo e a criação de estratégias. Cita-se também a colaboração no desenvolvimento de aspectos sociais, como a convivência em coletividade e a elaboração de inúmeras estratégias de sobrevivência, aprendizados essenciais para a vida adulta [Bettin e Pretto 2016]. Uma perspectiva também enfatizada por outros autores res-

salta a relevância deste jogo na Educação Infantil no tocante à *etnomatemática*, ou seja, ao aprendizado não-formal de conceitos matemáticos, a exemplo das figuras geométricas existentes no tabuleiro [de Oliveira Sardinha e Gaspar 2010].

3. Estimativas para a Árvore de Jogo

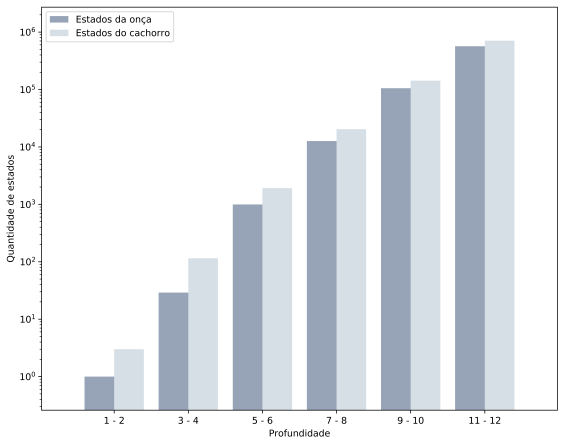
Uma *árvore de jogo* é uma grafo conexo e acíclico na qual os nós são estados do jogo e as arestas denotam as movimentações possíveis. A árvore do jogo considera todos os estados e movimentos possíveis a partir da configuração inicial e tem como nós-folha todos estados correspondentes a jogos concluídos. Embora permita uma visão completa de todas as possibilidades de jogo, a obtenção desta árvore é comumente intratável em cenários razoavelmente complexos, pois há um número arbitrariamente grande de estados de jogos a explorar. Para contornar esta dificuldade, portanto, é comum obter estimativas para esta árvore [Russell e Norvig 2016].

No cenário do Jogo da Onça aqui considerado, a obtenção de uma estimativa da árvore de jogo considerou primeiramente a implementação do jogo na linguagem Python, em interface orientada à caracteres, e adotou uma estratégia de poda com vistas a evitar produzir subárvores induzidas iguais. Para cada nó da árvore, considerou-se a exploração exaustiva de todos os seus nós-filhos distintos até o décimo segundo turno do jogo. O programa produzido foi executado em um computador *desktop* com processador Intel Core i7, 8 GB de memória principal e contou com o auxílio de um banco de dados relacional para armazenar os nós das árvores, representados segundo uma codificação em *strings* de tamanho fixo, e os seus respectivos níveis.

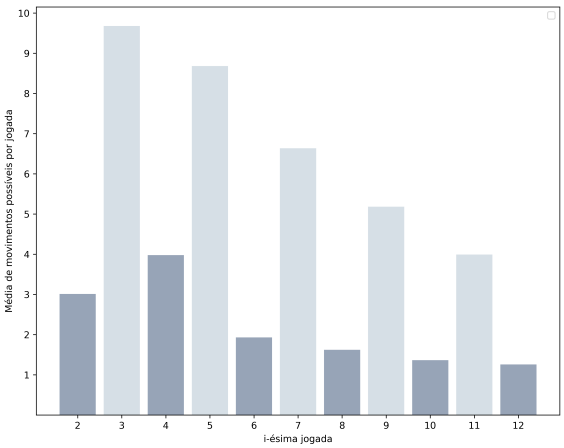
Como resultado deste processamento, obteve-se um quantitativo de $1.559.896$ estados distintos em uma árvore de jogo com 12 níveis (turnos). A ilustração do número de jogadas por turno pode ser vista na Figura 2a. Nota-se que, nos turnos iniciais o número de estados é pequeno, mas que, a partir do nono turno, o número de jogadas distintas é da ordem de 70.000, o qual segue de forma crescente no total de turnos explorados.

Figura 2: Histogramas produzidos a partir da exploração exaustiva, com poda de nós repetidos, da árvore de jogo com 12 níveis.

(a) Estados distintos por nível, em escala logarítmica com base 10.



(b) Razão do fator de ramificação por turno



A partir desses resultados, pôde-se então calcular o fator de ramificação, denotado por b (do inglês, *branch factor*), o qual indica quantos movimentos diferentes um jogador pode fazer em média. Para tanto, computou-se a razão entre o número de estados no i-ésimo nível e o número de estados no nível seguinte, conforme ilustrado na Figura 2b. Em seguida, considerando a média desta razão pelos número de níveis, obteve-se fator de ramificação $b = 4{,}667$. Isto significa que, em média, cada jogada enseja quase cinco jogadas distintas a partir dela.

A estimativa do comprimento do jogo, denotada por d é menos trivial de ser obtida. No caso de jogos de tabuleiro como Damas e Xadrez, por exemplo, tem-se que d é aproximadamente igual a 70 e 80, respectivamente [Chorus 2009]. Estes valores costumam ser obtidos por meios de análise combinatória ou então a partir de uma média de diversos jogos conduzidos por praticantes reais. Porém, uma vez que não existem bases de dados com partidas do Jogo da Onça para auxiliar nesta estimativa, optou-se por tomar como referência o valor de jogo de Damas, cujo tabuleiro possui 32 posições, apenas uma a mais que o tabuleiro do Jogo da Onça, e que possui que $d = 70$ [Schaeffer et al. 2007]. Tomando então os valores de b e d, tem-se um limitante inferior estimado para a complexidade da árvore de jogo (GTC, do inglês *Game Tree Complexity*), dado por:

$$GTC \geq b^d = 4{,}667^{70} \quad (1)$$

$$\gtrapprox 6{,}802 \times 10^{46}. \quad (2)$$

Se escalonado para um valor logarítmico na base 10, este valor resulta em cerca de 46. O jogo de Damas, por exemplo, possui complexidade da árvore de jogo em ordem logarítmica (base 10) igual a 31. Este é um resultado preliminar, mas que auxilia no dimensionamento dos esforços das etapas posteriores, como será mostrado a seguir.

4. Funções de Avaliação: Proposição, Metodologia e Resultados

Construir um jogador inteligente é análogo a resolver um problema de busca na árvore de jogo. Porém, no caso do Jogo da Onça, considerando o tamanho estimado desta árvore, efetuar buscas exaustivas em largura ou profundidade na mesma são tarefas intratáveis. Para lidar com esta dificuldade, considera-se a árvore de busca, uma árvore sobreposta à árvore de jogo na qual examinam-se os nós o suficiente para permitir que um jogador determine qual lance deve fazer.

No caso do Minimax, uma algoritmo clássico da IA para endereçar a questão, tem-se a obtenção de estimativas para um dos agentes a partir do estado corrente e da árvore de busca. Para tanto, calcula-se um *valor minimax* para cada estado, considerando jogadas futuras simuladas recursivamente, visando maximizar a pontuação do jogador de interesse e minimizar a pontuação do adversário. Assim, percorre-se o caminho descendente até os estados finais e propaga-se os melhores valores minimax de volta pela árvore de busca, segundo uma estratégia de *backtracking*. A estratégia de poda *Alpha-Beta* é aplicada neste algoritmo com vistas a ignorar partes da árvore que possivelmente não contenham o melhor movimento, sem afetar o resultado final, e com impactos significativos no desempenho do Minimax em árvores de busca grandes [Millington e Funge 2009].

No Minimax, a exploração da árvore de busca é feita até uma certa profundidade. Porém, se nesta exploração não forem encontrados estados finais, a princípio não há como

determinar o valor minimax a ser retornado. Para contornar esta dificuldade, as *funções de utilidade* exercem um papel estratégico: se ao chegar neste limiar o algoritmo não encontrar nenhum estado final, a função de utilidade de estado é chamada para retornar uma estimativa que aquele estado teria se a árvore de jogo completa fosse considerada [Russell e Norvig 2016]. O projeto de funções de utilidade deve levar em conta conhecimentos estratégicos do jogo em si, afinal deve refletir positivamente boas jogadas a serem maximizadas pelo Minimax.

No caso do Jogo da Onça, projetar tais funções de utilidade não é uma tarefa trivial, pois os adversários encontra-se em número distinto, isto é, tem-se um jogo entre desiguais, e há objetivos e movimentos diferentes. Porém, visando endereçar esta tarefa, a subseção a seguir considera a proposição de algumas funções de utilidade para este jogo e as estratégias adotadas para avaliação das mesmas.

4.1. Metodologia

Na proposição de funções de utilidade para este jogo, estabeleceu-se que os valores das mesmas deveriam estar no intervalo $[-1000, +1000]$, com pontuações intermediárias inteiras, visando aproveitar-se da eficiência da aritmética de inteiros no âmbito computacional [Millington e Funge 2009]. Como *baseline* comparativo, considerou-se que a escolha da ação a ser executada pelos agentes seria uma das opções disponíveis naquele estado sorteada de maneira aleatória, o que foi denominado $f_{o,0}$ e $f_{c,0}$ para onça e cachorro, respectivamente.

No caso da Onça, tem-se que esta alcança a vitória ao capturar 5 cachorros. Assim, uma função de utilidade intuitiva para a mesma, denotada por $f_{o,1}$, contabilizou $+200$ pontos a cada cachorro capturado, de modo que esta estimativa torna-se crescente à medida que a onça concretiza o seu objetivo, atingindo o valor máximo ao vencer o jogo. Nesta proposta de função de utilidade observa-se uma estratégia gulosa para a onça, em que a captura de um cachorro deve ser sempre priorizada.

No caso dos Cachorros, a proposição de tais funções foi menos trivial, pois deveria considerar uma estratégia coletiva das peças com vistas a encurralar a onça, ao passo que este cálculo não poderia ser custoso em virtude dos aspectos de desempenho. Neste caso, duas estratégias distintas foram traçadas. A primeira delas, estabelecida na função de utilidade $f_{c,1}$ e detalhada no Algoritmo 1, produz um valor a partir do número de cachorros vizinhos à onça e que não podem ser atacados pela mesma, visando estimar o quantitativo da restrição de movimentação da onça.

A função de utilidade $f_{c,2}$ visa quantificar a área de movimentação livre da onça a partir de uma busca em largura no tabuleiro, conforme apresentado no Algoritmo 2. A ideia desta função partiu de testes preliminares [Referência omitida, revisão double blind] e considera que, quanto menor a movimentação da onça, mais próximo os cachorros estão de seu objetivo.

Um dos aspectos subjetivos do Jogo da Onça consiste na reclamação dos empates, que dificulta a verificação de maneira automática e pode levar à situações de falta de combatividade, com os agentes movendo-se no tabuleiro, mas sem estarem orientados aos seus objetivos individuais. Para automatizar a detecção desta situação, foi definida uma estratégia de verificação automática da repetição de estados, a qual é descrita no

Algoritmo 1: Função de utilidade $f_{c,1}$ para o agente cachorro.

Entrada: Um estado de jogo, considerando o turno do cachorro.
Saída: Inteiro, correspondendo ao valor da função de utilidade $f_{c,1}$ para o cachorro.

```
1  início
2      t ← 0
3      p ← 0
4      para toda casa c adjancente à onça faça
5          se se há cachorro em c então
6              t ← t + 1
7              se se o cachorro em c não pode ser capturado pela onça então
8                  p ← p + 1
9      fim
10     retorna int(1000 · p/t)
11 fim
```

Algoritmo 2: Função de utilidade $f_{c,2}$ para o agente cachorro.

Entrada: Um estado de jogo, considerando o turno do cachorro.
Saída: Inteiro, correspondendo ao valor da função de utilidade $f_{c,2}$ para o cachorro.

```
1  início
2      s ← onça.posicao()
3      fila.enfileira(s)
4      area ← 0
5      enquanto fila não está vazia faça
6          area ← area + 1
7          casa ← fila.desenfileira()
8          para cada vizinha de casa faça
9              se vizinha.vazia() e vizinha.naoVisitada() então
10                 fila.enfileira(vizinha)
11                 vizinha.visitada ← True
12         fim
13     fim
14     retorna min(1000,max(−1000,1000 - (area·100)))
15 fim
```

Algoritmo 3. No critério estabelecido, pode-se determinar qual agente foi responsável pelo empate por falta de combatividade.

Para avaliar as funções de utilidade propostas, foram promovidos jogos ótimos entre os pares de funções apresentados e diferentes profundidades foram exploradas pelo Minimax na árvore de busca, a citar: 5, 6, 8 e 10 turnos à frente. As partidas consideradas tiveram até 25 turnos e foram repetidas 100 vezes por configuração, totalizando 2.100 partidas executadas e contabilizadas. Destas, foram coletados o número de partidas concluídas com vitória de cada agente, as partidas não concluídas e os cenários de falta

Algoritmo 3: Critério para definição de empate por falta de combatividade.

Entrada: Uma partida P com $T = \{t_i, t_{i+2}, \dots, \}$ turnos referentes a um jogador J

Saída: True, em caso de falta de combatividade; e, False, em caso contrário.

```
início
    C ← ∅
    p ← 0 // p denota a penalidade
    para t ∈ T faça
        Obtenha S que é um conjunto de posições do jogador J no turno t
        se S ⊆ C então
            p ← p + 1
        senão
            C ← C ∪ S
        fim
    fim
    se p ≥ |C|/2 então
        retorna True
    senão
        retorna False
    fim
fim
```

de combatividade, quando houve. O ambiente computacional utilizado para executar tais confrontos foi o mesmo reportado anteriormente, em que scripts na lingugagem Python auxiliaram na automatização dos testes e na coleta das métricas de interesse.

4.2. Resultados

Após a condução da metodologia descrita na seção anterior e da síntese das métricas de interesse, tem-se os resultados descritos na Tabela 1. Primeiramente, em relação ao *baseline* com ambos os jogadores movimentando-se aleatoriamente ($f_{o,0} \times f_{c,0}$), tem-se que há 28 vitórias da onça e que não foram registradas vitórias do cachorro. Para as partidas não concluídas, a falta de combatividade não foi analisada. A partir destes resultados, tem-se a percepção inicial de que vitórias da onça são mais prováveis que vitórias do cachorro, embora mais repetições precisem ser efetuadas para reforçar esta hipótese.

Ao analisar o desempenho da função de utilidade $f_{o,1}$ proposta para a onça versus o agente cachorro movendo-se aleatoriamente, percebe-se que a mesma é efetiva contra um adversário ingênuo, apesar do quantitativo superior deste último. Nas ocasiões de empate por falta de combatividade, todos estes foram ocasionados pelo adversário, sugerindo que a função proposta é eficiente na condução à vitória da onça neste cenário.

Quando considera-se a função de utilidade $f_{c,1}$ para o cachorro versus a onça movendo-se aleatoriamente, percebe-se que o número de vitórias do cachorro não foi expressivo. O número de partidas não concluídas é alto, há vitórias do adversário ainda que este não esteja munido de uma estratégia inteligente e os registros de falta de combati-

Tabela 1: Resultados obtidos do cenário de testes para as diferentes funções de utilidade.

Função de Utilidade		Profundidade	Vitórias		Partidas não concluídas	Falta de Combatividade (%)	
Onça	Cachorro		Onça	Cachorro		Onça	Cachorro
$f_{o,0}$	$f_{c,0}$	-	28	0	72	-	-
$f_{o,1}$	$f_{c,0}$	5	92	0	8	0	12.50
$f_{o,1}$	$f_{c,0}$	6	92	0	8	0	0
$f_{o,1}$	$f_{c,0}$	8	92	0	8	0	37.50
$f_{o,1}$	$f_{c,0}$	10	91	0	9	0	22.22
$f_{o,0}$	$f_{c,1}$	5	1	4	95	0	23.16
$f_{o,0}$	$f_{c,1}$	6	25	4	71	0	16.90
$f_{o,0}$	$f_{c,1}$	8	24	6	70	0	8.57
$f_{o,0}$	$f_{c,1}$	10	43	2	55	0	10.90
$f_{o,0}$	$f_{c,2}$	5	75	0	25	0	12.00
$f_{o,0}$	$f_{c,2}$	6	46	3	51	0	15.69
$f_{o,0}$	$f_{c,2}$	8	25	5	70	0	20.00
$f_{o,0}$	$f_{c,2}$	10	54	0	46	0	21.74
$f_{o,1}$	$f_{c,1}$	5	2	0	98	1.02	33.67
$f_{o,1}$	$f_{c,1}$	6	1	0	99	0	4.04
$f_{o,1}$	$f_{c,1}$	8	1	0	99	0	2.02
$f_{o,1}$	$f_{c,1}$	10	1	0	99	0	6.06
$f_{o,1}$	$f_{c,2}$	5	29	0	71	0	19.72
$f_{o,1}$	$f_{c,2}$	6	11	0	89	0	5.62
$f_{o,1}$	$f_{c,2}$	8	7	0	93	0	8.60
$f_{o,1}$	$f_{c,2}$	10	21	0	79	0	7.59

vidade são apenas relativos ao cachorro. O mesmo é verificado para a função de utilidade $f_{c,2}$, em que também observa-se um aumento no percentual das situações de falta de combatividade.

Considerando então o confronto entre as funções de utilidade propostas, a começar por $f_{o,1} \times f_{c,1}$, tem-se poucas vitórias da onça, um baixo percentual médio de falta de combatividade entre os adversários e muitas partidas não concluídas, o que sugere um bom equilíbrio entre os adversários, tornando o jogo acirrado para o número de turnos observado.

Por último, as simulações considerando os confrontos de agentes com funções $f_{o,1} \times f_{c,2}$ denotam para o agente onça uma predominância de vitórias. Além disso, não foram registradas vitórias por parte do agente cachorro e a falta de combatividade é ligeiramente menor que o cenário anterior – considerando as somas percentuais –, mas ainda comparável.

De maneira geral, tem-se um desempenho adequado para a função de utilidade $f_{o,1}$ proposta para a onça, com um bom quantitativo de vitórias contra um adversário aleatório e com um número razoável de ganhos sobre um adversário com a função $f_{c,2}$. Quando esta função é colocada contra um adversário com a função $f_{c,1}$, há um embate maior, com muitas partidas não concluídas para o número de turnos observado. Para o agente cachorro, entretanto, tem-se que as funções propostas $f_{c,1}$ e $f_{c,2}$ ainda são pouco efetivas para sintetizar uma estratégia de inteligência coletiva, o que pode ser constatado pelo baixo número de vitórias e pelo número de empates por falta de combatividade deste agente.

5. Considerações Finais

Este trabalho teve por objetivo apresentar os resultados preliminares do uso de técnicas de Inteligência Artificial baseadas na exploração da árvore de busca no Jogo da Onça, um jogo de tabuleiro de tradição indígena brasileira. Este é um jogo de informação perfeita, determinístico, completamente observável, com dinâmica baseada em turnos, soma-zero, multiagente, em que dois adversários sujeitos à quantidades, movimentações e objetivos distintos competem em um tabuleiro assimétrico.

Primeiramente, a partir de uma exploração exaustiva de estados da árvore de jogo até o décimo segundo nível, sujeita à poda para evitar subárvores induzidas iguais, estimou-se um fator de ramificação de $4{,}667$ e, considerando a ausência de bases de dados com partidas deste jogo, mas efetuando uma analogia com o jogo de Damas em virtude do número de casas, tomou-se o comprimento de jogo como aproximadamente igual a 70. Em decorrência, estimou-se um limiar inferior de $6{,}802 \times 10^{46}$ estados distintos para a complexidade da árvore de jogo, o que sugere maior complexidade deste jogo quando comparado ao jogo de Damas.

Em seguida, houve a utilização do algoritmo Minimax com estratégia de poda *Alfa-Beta* para explorar uma árvore sobreposta à árvore de jogo, em que o número de nós explorados nesta estratégia contorna os desafios de intratabilidade previamente estimados. Para viabilizar a utilização deste algoritmo, foram propostas três funções de utilidade, sendo uma para a onça e duas para os cachorros, além de um critério objetivo para determinação de empates. A realização de simulações permitiu avaliar as funções propostas, em que verificou-se que a estratégia proposta para onça mostrou-se efetiva contra um jogador aleatório, com bom desempenho contra uma das funções propostas para o cachorro e com pouca efetividade contra a outra função proposta para o jogador adversário. No geral, as funções de utilidade propostas para o cachorro mostram-se limitadas, o que revelou uma dificuldade em quantificar o aspecto de inteligência coletiva deste agente.

Há muitos aspectos da proposição de agentes inteligentes para o Jogo da Onça que podem ser explorados em trabalhos futuros. Além da proposição de outras funções de utilidade que possam contornar as dificuldades verificadas, tem-se a possibilidade de explorar outras técnicas de IA, como o *Monte Carlo Tree Search* e o Aprendizado por Reforço. A criação de meios para coletar exemplos de partidas com jogadores humanos com diferentes graus de *expertise* pode auxiliar também na criação de jogadores automáticos com diferentes graus de dificuldade, o que pode facilitar a disseminação do Jogo da Onça, um legado cultural indígena brasileiro.

Agradecimentos

Os autores agradecem o apoio financeiro e material provido pela Fundação de Amparo à Pesquisa do Estado do Amazonas (FAPEAM) e pelo Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) por meio do Programa PPP Edital 04/2017.

Referências

Bettin, A. D. H. e Pretto, V. (2016). A construção do jogo indígena a partir da matemática. In *Anais do XXI SIEDUCA – A escola já não é a mesma: novos tempos, novos paradigmas*, volume 21, pages 266–273.

Chen, J. X. (2016). The evolution of computing: Alphago. *Computing in Science & Engineering*, 18(4):4–7.

Chorus, P. (2009). Implementing a computer player for abalone using Alpha-Beta and Monte-Carlo search. Maastricht University, Dissertação.

de Oliveira Sardinha, A. G. e Gaspar, M. T. J. (2010). Jogos indígenas aplicados ao ensino de Matemática. In *Anais do X Encontro Nacional de Educação Matemática*, pages 1–10, Salvador, Bahia.

Ferreira, M. B. R., Vinha, M., e de Souza, A. F. (2008). Jogos de tabuleiro: um percurso em etnias indígenas. *Ciência em Movimento*, 16(1):47–55.

Ghory, I. (2004). Reinforcement learning in board games. Technical report, Department of Computer Science, University of Bristol, Inglaterra.

Gobet, F., de Voogt, A., e Retschitzki, J. (2004). *Moves in Mind – The psychology of board games*. Psychology Press, Taylor & Francis Group, Estados Unidos.

Hsu, F.-H. (2002). *Behind Deep Blue*. Princeton University Press, Estados Unidos, 1 edição.

Millington, I. e Funge, J. (2009). *Artificial Intelligence for Games*. Elsevier, Estados Unidos, 2 edição.

Russell, S. e Norvig, P. (2016). *Artificial Intelligence – A Modern Approach*. Pearson, Nova Jersey, 3 edição.

Schaeffer, J., Burch, N., Bjornsson, Y., Kishimoto, A., Muller, M., Lake, R., Lu, P., e Sutphen, S. (2007). Checkers is solved. *Science*, 317(5844):1518–1522.

Secretaria Municipal de Educação (2016). O jogo da onça - aprenda a jogar. Prefeitura de São Paulo. Disponível em:`https://bit.ly/2YAVX0P`. Acessado em 27 de agosto de 2019.

Viegas, O., Lindner, N., e Mallmann, C. S. (2019). Catálogo de jogos e desafios do mundo. Oficina do Aprendiz. Disponível em `http://bit.ly/2KIt6nz`. Acesso em 28 de junho de 2019.

Yannakakis, G. N. e Togelius, J. (2018). *Artificial Intelligence and Games*. Springer, Estados Unidos.