Algorítimos Genéticos e Lógica Difusa em Jogos de Tabuleiro

Relatório Final

AYKROYD, Christopher R B

EPUSP

1. Introdução e Objetivos

Othello é a evolução de um jogo de tabuleiro antigo. Apesar da aparentemente simplicidade em suas regras, o jogo leva anos para ser dominado, devido à possibilidade de mudanças abruptas no tabuleiro em uma única jogada. Por esse motivo foi por muitos anos um objeto de estudo na área de *Machine Learning*, com muitos programas de qualidade desenvolvidos ao longo dos anos - como IAGO, de Rosenbloom[2], ou BILL, feito por Lee e Mahajan. Em 1992 iniciou-se o desenvolvimento do Logistello[6], um programa que aprendeu ao jogar contra si próprio.

Baseado no trabalho de Jang, Sun e Mizutani[1], esse estudo usará ideias da Teoria de Conjuntos Difusos para implementar um programa autônomo jogador de Othello, e analisar a viabilidade de um método particular de auto-aprendizagem.

2. Métodos

O programa foi implementado em C++, com saída de caracteres no Console, e *softcoded* para um tabuleiro 6x6 para reduzir o tempo de treinamento. O seu conteúdo pode ser dividido em 3 grandes blocos, e está esquematizado na figura 1.

- Conjunto de regras e estruturas particulares ao jogo de Othello;
- Jogadores-máquinas e seus métodos de tomada de decisão dentro de uma partida;
- Estrutura de treinamento e seleção natural dos jogadores-máquinas;

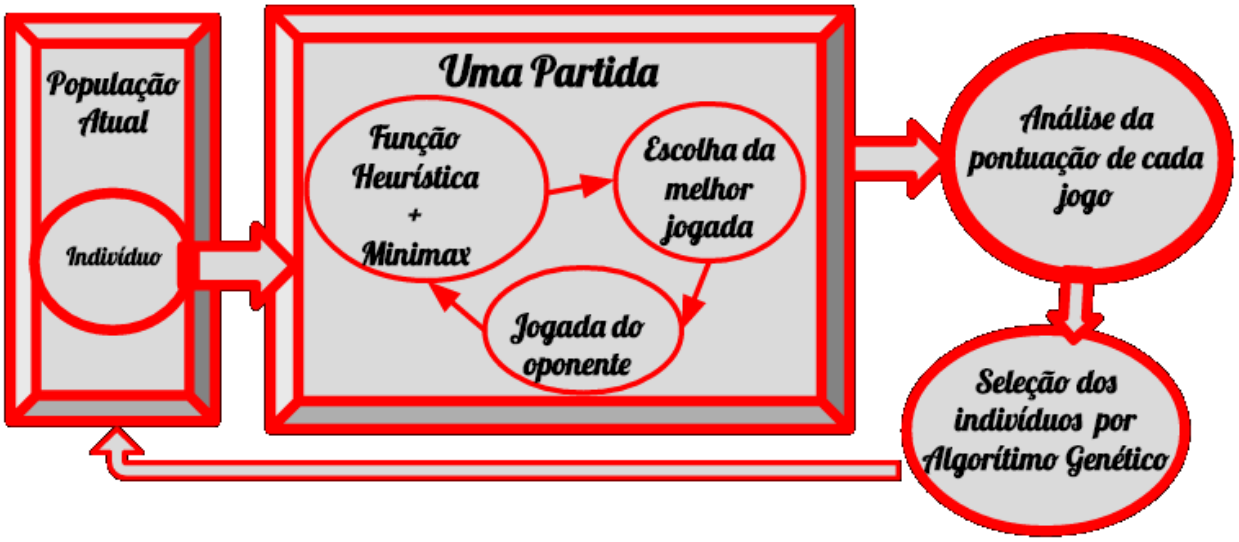


Figura 1. Esquematização da estrutura do programa final durante treinamento.

3. Implementação

Para a tomada de decisão em cada jogada foi usado o algorítimo de busca minimax simples, sem otimizações, acoplado à uma função de avaliação estática linear em relação aos atributos, mas difusa com relação ao turno do jogo. As funções de pertinência eram todas trapezoidais.

A função de avaliação estática é usada dentro do minimax para quantizar subjetivamente o quão vantajosa é uma configuração do tabuleiro, codificada em atributos. Esses elementos subjetivos estão embutidos nos pesos e funções de pertinência

$$h_n(t, \vec{f}) = \sum_{i=1}^{n} \mu_i(t) w_i f_i$$

armazenados pelos indivíduos, e de certa forma determinam a 'personalidade' de um jogador-programa.

Em um primeiro momento foram utilizados 11 atributos: vantagem em peças, mobilidade do jogador (número de jogadas possíveis), mobilidade do oponente, mobilidade potencial A e B (representando jogadas potenciais), e mais 6 atributos codificando a configuração do tabuleiro, fazendo uso de sua simetria.

Logo em seguida, notou-se a necessidade de implementar outros atributos. A estabilidade - número de peças que jamais poderão ser capturadas - é problemática na implementação, e foram ignorados múltiplos casos para obter um algorítimo rápido. Além da estabilidade, foram adicionados dois outros atributos: as mobilidades potenciais A e B do oponente.

O algorítimo implementado para calcular a estabilidade utilizou o seguinte princípio: peças serão totalmente estáveis se e somente se estáveis em todas as 4 direções (horizontal, vertical e diagonais). Para a estabilidade direcional, consideramos basicamente três diferentes situações – (1) a peça está adjacente à fronteira do tabuleiro, (2) está adjacente à outra peça estável do mesmo time, ou então (3) a linha de posições nessa direção está totalmente cheia de peças.

Perceba, no entanto, que esses não são os únicos casos de estabilidade direcional. Mais precisamente, uma linha quase cheia pode dar estabilidade direcional para algumas peças dessa linha e não para outras.

Note que na figura da direita, além das peças pretas riscadas em vermelho, a branca também tem estabilidade na direção vertical, já que existem somente duas casas vazias na linha analisada.

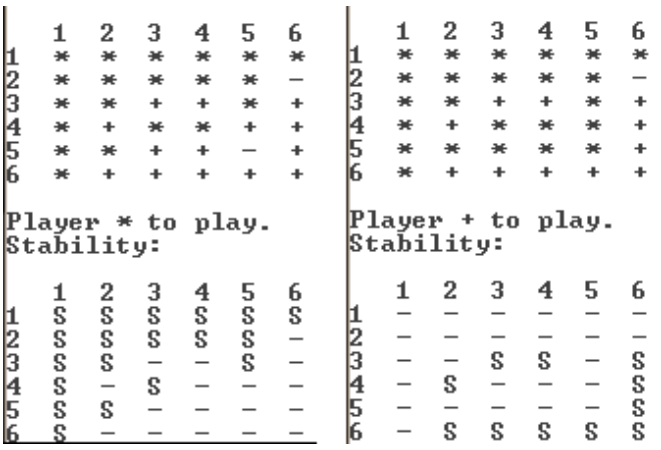


Figura 2. Posições estáveis

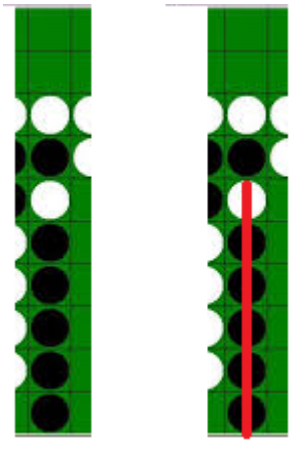


Figura 3. Casos não detectados

4. Treinamento

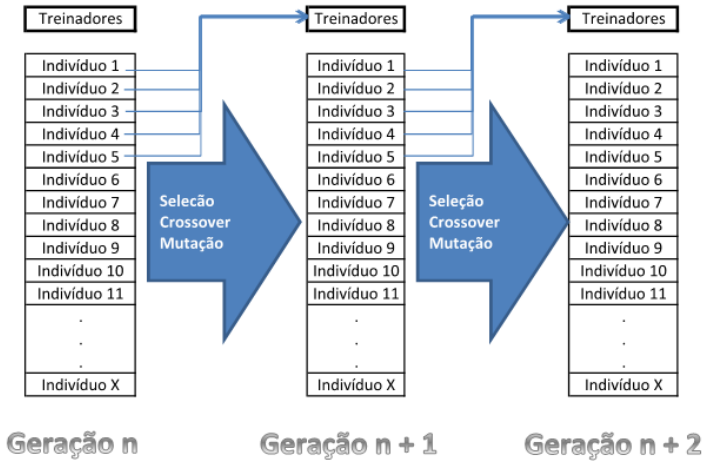
Nosso objetivo é obter uma máquina que (1) consiga vencer contra jogadores medianos, e (2) consiga se adaptar a jogadores diferentes. Para obter tais resultados devemos fazer uma seleção dos parâmetros da função de avaliação estática, usada para guiar as decisões de jogadas individuais. Fica claro, nesse caso, que o resultado (pontuação final) de um jogo é fortemente relacionada tanto a esses parâmetros quanto à habilidade do oponente, ambos de modo implícito.

O jogo de Othello é extremamente volátil, e por consêquencia fica difícil avaliar se uma jogada foi boa ou ruim, mesmo após a conclusão do jogo. Devido à dificuldade de relacionar o impacto de cada jogada com o resultado final do jogo, a solução mais obvia é o uso de métodos de otimização sem derivadas, como os Algorítimos Genéticos, explorados nesse estudo.

Os Algorítimos Genéticos, nesse caso, usariam como medida de aptidão a pontuação acumulada em vários jogos contra um grupo específico de treinadores, e por consêquencia seriam selecionados os parâmetros mais efetivos.

Para abordar a questão (2) decidimos usar um grupo de 5 treinadores, que são trocados ao final de cada geração. Em estudos anteriores, foi mostrado a eficácia de usar múltiplos treinadores diferentes, evitando que os jogadores se cometessem demais aos pontos fracos particulares do treinador[1].

Para a etapa de treinamento, foi gerada uma população de 100 indivíduos com parâmetros aleatórios. Realizamos um torneio local e os 5 melhores indivíduos selecionados como treinadores da próxima geração. A partir desse ponto, a cada geração, os indivíduos jogaram duas partida contra cada um dos treinadores, alternando entre pretas e brancas; no final dos jogos, através de cópias dos 5 jogadores de maior aptidão eram selecionados os treinadores da próxima geração.



Codificação

Os parâmetros foram codificados em um vetor (cromossomo) de 8 *bits* positivos, o *crossover* efetuado num único ponto com chance de 100%, e mutações múltiplas com média de 0.2 mutações por cromossomo. Cada parâmetro era decodificado novamente após essas etapas em seus respectivos espaços viáveis, para serem usados nas funções heurísticas. A persistência dos indivíduos também foi gerada com tal codificação.

5. Resultados

Durante o estudo, no início de cada geração cada individuo jogou duas partidas contra um jogador de referência, não usado no treinamento, como validação cruzada. Num primeiro momento, notamos que os valores demoravam muito para convergir e as oscilações das mutações acabavam impactando os jogadores bons. Assim, subtraímos da medida de aptidão de cada indivíduo o menor valor entre eles. Isso garantiu que os piores indivíduos fossem eliminados rapidamente, aumentando muito a velocidade e o limite de convêrgencia.

Nesse segundo momento, os três valores de pontuação (melhor, pior e média) convergiram em 10-30 épocas para 30 pontos de 36 [Fig 5]. Isto é, todos os indivíduos convergiram para um máximo. Nos restava analisar se tal máximo era era local ou não, e mais importante, se atendia as expectativas de vitória contra jogadores humanos.

Para testar os resultados do programa, um humano jogou 3 partidas contra a máquina treinada e mais 3 contra máquinas aleatórias, não treinadas.

	Jogo 1	Jogo 2	Jogo 3
Máquina Treinada	- 14	- 8	- 2
Máquina Aleatória	+ 16	+ 10	+26

Tabela. Pontuação do jogador Humano contra as máquinas.

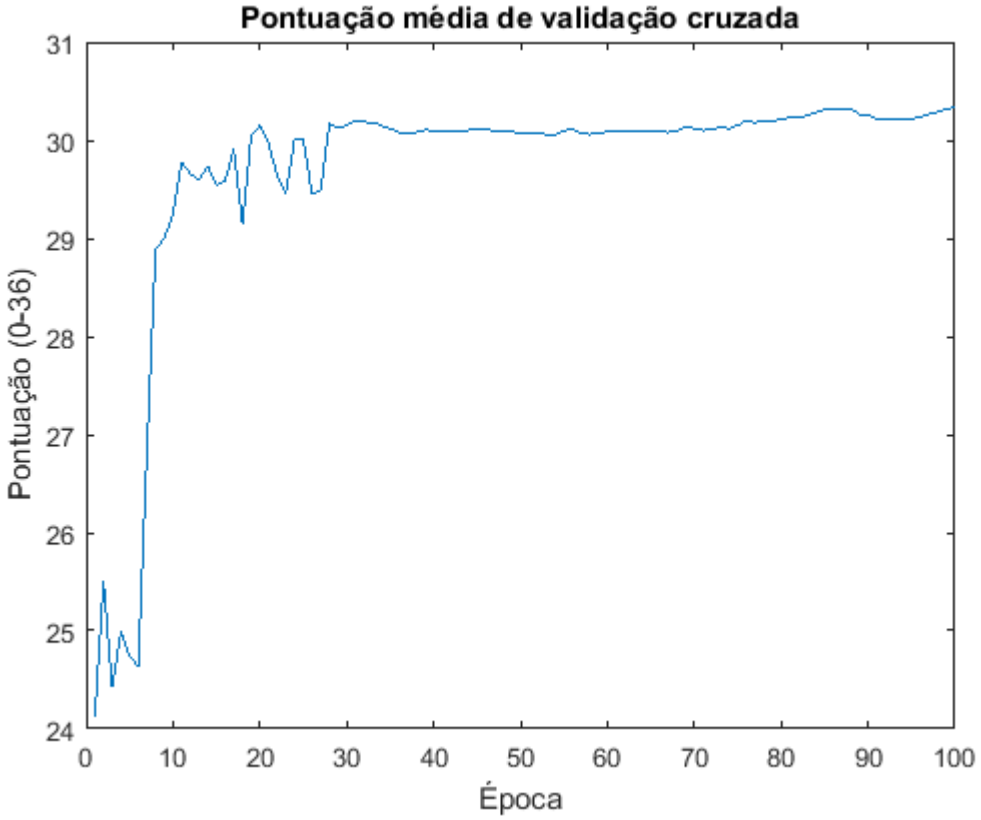


Figura 5.

6. Conclusões

Jang, Sun e Mizutani[1] demonstraram a fragilidade de programas aprendendo sobre a instrução de um único treinador, resultando em uma menor adaptação a oponentes novos. Apesar do método explorado de autoaprendizagem usar múltiplos treinadores, havíamos reconhecido que um dos seus riscos seria de prender-se a mínimos locais, já que os treinadores são os próprios jogadores – contrastando com os métodos usuais, utilizando como treinadores programas profissionais.

A partir do momento em que os cromossomos começaram a rapidamente convergir, a população foi se homogenizando, e como consequência os treinadores também foram perdendo sua personalidade, já que os mesmos eram sempre copias retiradas dos indivíduos normais. Assim, a noção de usar múltiplos treinadores na aprendizagem foi comprometida nesse final, e não podemos dizer com certeza que o máximo atingido era global.

Por outro lado, apesar das dúvidas, o método resultou em jogadores de certa competentencia. Nós acreditamos que esse método possa ser expandido ainda mais –

alterando a técnica de seleção dos treinadores, de modo a evitar sua homogenização, assim reduzindo a velocidade de convergência e aumentando a qualidade final do extremum.

7. Trabalhos Futuros

O aumento do tamanho do tabuleiro para 8x8 definitivamente acarretará em um aumento geométrico do tempo computacional de tomada de decisões. Nesse caso seria essencial otimizar o algorítimo minimax (que é quem mais leva tempo) com técnicas de poda alpha-beta e aprofundamento iterativo com ordenação de nós (aumentando a eficiência da poda).

Além disso, parece ser interessante introduzir uma certa aleatoriedade ponderada na escolha da jogada, para amenizar a característica determinística do método de tomada de decisões, e reduzir as frustrações da previsibilidade para os jogadores humanos.

Outro aprimoramento possível seria um cálculo mais completo do atributo estabilidade, através de uma tabela combinatória para determinar a estabilidade direcional de cada peça do tabuleiro, a qual incluiria os casos de linhas semi-cheias. Seria interessante analisar os benefícios de tal aprimoramento, e determinar sua viabilidade em relação à aumento de memória e tempo computacional.

Um futuro trabalho poderia ainda explorar a ideia de funções de avaliação estática não-lineares, através de redes neurais usuais ou de inferência difusa. Fica claro que a estrutura dessas redes deve ser relativamente simples ou então personalizada, devido à natureza exponencial do algorítimo minimax e a lentidão dos métodos de otimização sem derivadas para uma quantidade muito grande de parâmetros.

8. Referências

[1] J. S. R. Jang, C. T. Sun and E. Mizutani, *Neuro-Fuzzy and Soft Computing*

[2] Paul S. Rosenbloom, *A world-championship-level Othello program*, Carnegie Mellon University (1982) - http://repository.cmu.edu/cgi/viewcontent.cgi?article=3452&context=compsci

[3] https://en.wikipedia.org/wiki/Minimax

[4] Anton Leouski, *Learning of Position Evaluation in the Game of Othello* (1995) - http://people.ict.usc.edu/~leuski/publications/papers/UM-CS-1995-023.pdf

[5] A. CINCOTTI, V. CUTELLO, G. SORACE, Combining Fuzzy Sets and Genetic Algorithms for improving strategies in Game Playing http://www.wseas.us/e-library/conferences/tenerife2001/papers/548.pdf

[6] Código-fonte do Logistello, Michel Buro https://skatgame.net/mburo/log.html