Avaliação de Desempenho de Métodos Heurísticos e K-NN no Dominó

LEANDRO H. B. SILVA (IFPB, Campus João Pessoa), ARLEY W. N. SILVA (IFPB, Campus João Pessoa), THIAGO G. SILVA (IFPB, Campus João Pessoa)

Área de conhecimento:(Tabela CNPq): 1.03.03.04-9 Sistemas de Informação.
Palavras-Chave: dominó; heurística; aprendizagem de máquina, K-NN.

1 Introdução

Através da aprendizagem ou treinamento de uma Rede Neural Artificial (RNA), é possível realizar comportamentos complexos como dirigir um carro, auxiliar no controle de tráfego aéreo ou reconhecer padrões em imagens (SCHMIDHUBER, 2015). São cenários que exigem decisões em tempo real, com muitas variáveis constituindo problemas para solucionar. Para esse trabalho, foi escolhido o K-NN (k-nearest neighbors) que é um algoritmo de classificação supervisionado usado em machine learning. Consiste em utilizar os "K"vizinhos mais próximos em um banco de dados já classificado para classificar um novo dado.

A heurística é um conjunto de regras e métodos que conduzem à descoberta, à invenção e à resolução de problemas (BUENO, 2009). Para (BUENO, 2009), algoritmos exploratórios que buscam resolver problemas são conhecidos por métodos heurísticos, os quais se baseiam em sucessivas aproximações direcionadas a um ponto ótimo. Neste aspecto, a heurística (ISMAIL; AGWU, 2018) vem se destacando graças à ampla aplicação em outras áreas, como navegação de robôs, de helicópteros militares e até nos games.

A partir disso, optou-se pelo jogo de dominó para explorar os cenários das heurísticas e K-NN contra um algoritmo aleatório e entre si. Portanto, o objetivo deste estudo é avaliar o desempenho de ambos em disputas de jogos de dominó. Este trabalho está organizado da seguinte maneira: os trabalhos relacionados estão na Seção 2; os métodos empregados estão na Seção 3; resultados e discussões estão na Seção 4; conclusões estão presentes na Seção 5.

2 Trabalhos relacionados

Através das strings de busca "domino AND heuristic AND neural network", buscaram-se trabalhos relacionados nas bases IEEE e ACM, com 391 resultados encontrados (até 23/07/2021). Destes, 6 continham os termos "domino"e "heuristic".

(WIRTZ; HÄSELICH; PAULUS, 2010) aborda o reconhecimento de imagens de peças de dominó. Segundo os autores, as aplicações se estendem à identificação de semáforos e edifícios. (DIRIL et al., 2005) (PURI; BJORKSTEN; ROSSER, 1996) e (BUONANNO; SCIUTO; STEFANELLI, 1994) tratam da lógica domino, uma evolução baseada em CMOS das técnicas de lógica dinâmica baseadas em transistores PMOS ou NMOS. (COOK; ESPINOZA; GOYCOOLEA, 2005) estudam o conceito de domino-paridade envolvendo estrutura de subgrafo hereditário de distância. (CHEN; ZUKOWSKI, 1991) explora o dimensionamento de transistores. Nota-se, portanto, que não há, nas plataformas consultadas, esse tipo de pesquisa.

3 Materiais e Métodos

Implementou-se um jogo de dominó em ambiente gráfico a partir do framework Pygame (MCGUGAN, 2007). Para (MENINO; BARBOSA, 2006), um jogo de dominó pode ter pequenas modificações em suas regras. Dada a variabilidade das regras, este trabalho explorou a modalidade "burrinho", nas seguintes diretrizes: apenas dois jogadores; cada jogador começa com 3 peças; o primeiro a jogar é o que possuir a peça dupla mais alta que a colocará sobre a mesa; o jogador seguinte deverá colocar uma de suas peças, desde que ela possua indicação

numérica igual à inicial, conectando-as e assim sucessivamente. Na hipótese de o jogador não possuir alguma peça que permita a conexão, ele "compra" uma das peças restantes. Se elas não existirem é penalizado, passando a sua vez. Ganha o primeiro que ficar sem peças na mão e, caso o jogo seja travado (sem jogadas possíveis), é feitao o somatório do valor númerico de cada peça na mão dos jogadores e o jogador que obtiver o menor resultado é declarado vencedor. O perdedor começa com uma peça a mais na próxima rodada. O jogo segue até que a quantidade de peças na mão inicial de um dos jogadores seja catorze.

3.1 Heurísticas propostas

A primeira heurística (H1) tem um viés defensivo (manter-se jogando, evitar passar a vez ou buscar mais peças). Uma contagem nas peças disponíveis é feita para checar o número que mais se repete. No caso de empate, seria selecionada a peça de maior dígito para aumentar a chance de ganhar por fechamento.

A segunda heurística (H2) segue um viés ofensivo (impedir que o oponente jogue e que busque mais peças ou que passe a vez). Selecionaria-se o número que foi mais jogado e, em caso de empate, também seria selecionada a peça de maior dígito para aumentar a chance de ganhar por fechamento. Por exemplo, em um cenário em que as peças no campo e na mão são as referentes as figuras abaixo:

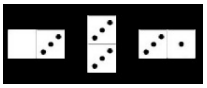

Figura 1: Peças no campo

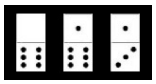

Figura 2: Peças na mão

H1 selecionaria a peça 6 | 1 (Fig. 2), pois há dois 6 e dois 1 na mão, e pelo critério de desempate, jogaria-se a maior peça e H2 selecionaria a peça 1 | 3, pois há 4 três somando a mão e o campo.

3.2 K-NN

A coleta de dados foi feita através do Pygame com a implementação do jogo e as regras especificadas, pessoas jogaram contra o código no modo aleatório (ou seja, a primeira peça na mão que fosse possível de ser jogada seria selecionada) e os dados e resultados, foram armazenados e totalizaram 20 jogos completos.

A estrutura proposta foi feita utilizando-se: quantidade de peças na mão (PM), quantidade de peças no campo (PC), quantidade de peças na mão do oponente (PO), peça que mais se repete dentre as peças na mão (qC) seguindo da mais comum para a menos comum (em caso de empate, seguindo da maior pra menor) e a quantidade de bordas disponíveis para jogar (qJ).

A partir dos dados coletados, organizou-se um vetor que representaria o estado do jogo em um plano com 11 dimensões com seu referido rótulo, ou seja, levou-se a uma vitória ou uma a derrota. No cenário das figuras 1 e 2 e, supondo que o oponente tivesse nesse momento 4 peças na mão, o vetor seria: [3 , 3 , 4 , 6 , 1 , 3 , 0 , 5 , 4 , 2 , 2].

Quanto à abordagem aplicada no jogo, ao iniciar, coletam-se as informações atuais do campo, mãos e bordas possíveis e se organizam todas em um vetor semelhante ao relatado na estrutura proposta. Com vetor preenchido com seus 11 campos, é feito o cálculo da distância euclidiana do estado atual com todos os estados no banco de dados, coletam-se as dez menores distâncias (estados de jogo mais semelhantes ao atual) e verifica-se a taxa de vitória. Isso é calculado para todas as possíveis jogadas e a jogada com maior probabilidade de vitória (em caso de empate, seleciona-se a jogada com maior peça).

4 Resultados e Discussão

No escopo de uma pesquisa em andamento, foram realizados 1000 jogos entre as heurísticas e o K-NN proposto para verificar a taxa de vitória entre eles e contra um algoritmo aleatório (a partir da biblioteca random

no Python). A Tabela 1 resume os resultados parciais.

Tabela 1: Resultado dos jogos realizados

	H1	H2	K	A
H1	-	496	717	683
H2	504	-	752	740
K	284	249	-	502
R	317	260	496	-

H1 = Heuristica 1, H2 = Heuristica 2, K = KNN, A = Aleatório

Observando os jogos entre as heurísticas e o algoritmo aleatório, observa-se que H1 e H2 obtém uma taxa de vitória de 68% e 74%, respectivamente. Entretanto, o K-NN, mesmo vencendo o random com 2 jogos de diferença, não obteve um bom resultado contra as heurísticas, vencendo menos de 300 jogos.

5 Considerações Finais

A partir dos resultados dos jogos, observa-se que as estratégias aplicadas nas heurísticas obtiveram um bom desempenho. Nota-se uma leve vantagem da H2 em relação a H1 (que fica mais notavél ao comparar os jogos contra o algoritmo aleatório, em que H2 vence 57 jogos a mais). Por outro lado, o K-NN não teve um desempenho tão bom, o que sugere a necessidade de alguns testes e mudanças como variação do K, uma coleta maior de dados ou mesmo uma remodelação da estrutura proposta.

Referências

BUENO, F. Métodos heurísticos. *Teoria e implementações. Araranguá: IFSC*, 2009.

BUONANNO, G.; SCIUTO, D.; STEFANELLI, R. Innovative structures for cmos combinational gates synthesis. *IEEE transactions on computers*, IEEE, v. 43, n. 4, p. 385–399, 1994.

CHEN, D.-P.; ZUKOWSKI, C. Cmos optimization including logic family mixing. In: IEEE. *1991., IEEE International Sympoisum on Circuits and Systems*. [S.l.], 1991. p. 2240–2243.

COOK, W.; ESPINOZA, D.; GOYCOOLEA, M. A study of domino-parity and k-parity constraints for the tsp. In: SPRINGER. *International Conference on Integer Programming and Combinatorial Optimization*. [S.l.], 2005. p. 452–467.

DIRIL, A. U. et al. Low-power domino circuits using nmos pull-up on off-critical paths. In: IEEE. *Proceedings of the ASP-DAC 2005. Asia and South Pacific Design Automation Conference, 2005.* [S.l.], 2005. v. 1, p. 533–538.

ISMAIL, I. M.; AGWU, N. N. Influence of heuristic functions on real-time heuristic search methods. In: IEEE. *2018 14th International Conference on Electronics Computer and Computation (ICECCO)*. [S.l.], 2018. p. 206–212.

MCGUGAN, W. *Beginning game development with Python and Pygame: from novice to professional*. [S.l.]: Apress, 2007.

MENINO, F. d. S.; BARBOSA, R. M. Dominós-um recurso lúdico na resolução de problemas para aprendizagem de sucessões. *Revista FAFIBE on line*, v. 2, n. 2, 2006.

PURI, R.; BJORKSTEN, A.; ROSSER, T. E. Logic optimization by output phase assignment in dynamic logic synthesis. In: IEEE. *Proceedings of International Conference on Computer Aided Design*. [S.l.], 1996. p. 2–8.

SCHMIDHUBER, J. Deep learning in neural networks: An overview. *Neural networks*, Elsevier, v. 61, p. 85–117, 2015.

WIRTZ, S.; HÄSELICH, M.; PAULUS, D. Model-based recognition of domino tiles using tgraphs. In: SPRINGER. *Joint Pattern Recognition Symposium*. [S.l.], 2010. p. 101–110.