Agente para o jogo Colonizadores de Catan

Bruno Paz e Gabriel Rubin
Faculdade de Informática (FACIN) – Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS)
Porto Alegre – RS – Brasil
✉ gabriel.lima.002@acad.pucrs.br
✉ bruno.paz@acad.pucrs.br
✉ felipe.meneguzzi@pucrs.br

O Problema

- Jogos estratégicos modernos, também conhecidos como *eurogames*, são de grande interesse para a comunidade de IA devido as características que estes jogos dividem com jogos de tabuleiro clássicos e jogos de videogame [2];
- Técnicas tradicionais de IA, como Minimax, podem ser inviáveis para jogar jogos desta categoria, devido à maior complexidade destes jogos quando comparada a jogos de tabuleiro tradicionais [3];
- Colonizadores de Catan é um jogo que representa bem o arquétipo de um *eurogame* [3] e possui elementos e características de jogo que o tornam desafiador: aleatoriedade, mais de 2 jogadores, trocas e cartas oclusas entre os jogadores;
- Neste trabalho, desenvolvemos um agente capaz de jogar este jogo competitivamente sem a necessidade de estratégias pré-programadas, através de técnicas de IA modernas;

Nosso Agente

Nosso agente joga Colonizadores de Catan através de algoritmos de busca baseados em *Monte Carlo Simulation*.

Monte Carlo Tree Search (MCTS):

- Este algoritmo realiza uma amostragem na árvore de jogo, reduzindo o espaço de busca consideravelmente, fator essencial para jogos com *branching-factor* elevado, como Catan [2].
- O algoritmo constrói uma árvore de pesquisa a partir de um estado de jogo, e executa 4 etapas para realizar sua estimativa:

1. Seleção: O algoritmo percorre a árvore de pesquisa construída e seleciona um nodo através de uma política de seleção.
2. Expansão: Um novo nodo é adicionado como filho do nodo selecionado na fase anterior a partir de uma jogada disponível.
3. Simulação: Uma simulação é executada a partir do nodo adicionado na fase anterior até um estado de fim de jogo.
4. Propagação: A partir do resultado da simulação, o algoritmo calcula uma recompensa estimada para o nodo e atualiza as recompensas estimadas dos nodos visitados durante a fase de Seleção.

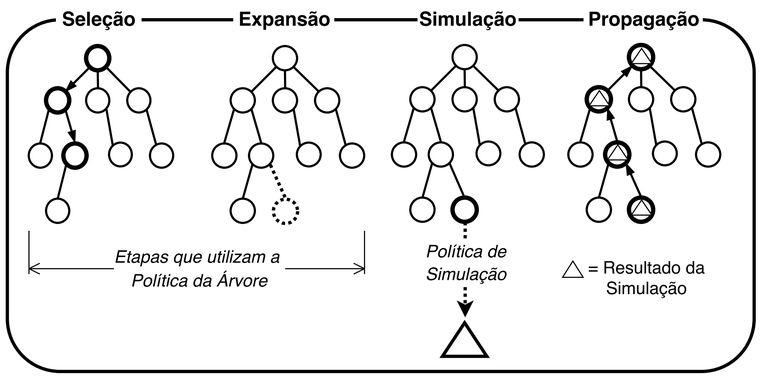

Figura 1: As 4 fases de uma iteração do algoritmo *Monte Carlo Tree Search*.

Upper Confidence Bounds for Trees (UCT):

- Especialização do algoritmo anterior que utiliza uma política de seleção de nodos na árvore de pesquisa que garante o equilíbrio ótimo entre *exploração* e *lucro* durante a seleção de nodos da árvore de busca [1].
- Este equilíbrio é obtido através da Equação 1, onde $\overline{X}_j$ é a recompensa média obtida escolhendo um nodo j, n_j é o número de vezes que a nodo j foi escolhido, n é o numero total de jogadas realizadas e C_p é uma constante de exploração.

$$UCT = \overline{X}_j + 2C_p\sqrt{\frac{2\ln n}{n_j}} \quad (1)$$

Experimentos e Resultados

- Para nossos experimentos, realizamos diversas partidas envolvendo nosso agente contra 3 agentes pseudo-aleatórios e, posteriormente, 3 agentes da implementação de Colonizadores de Catan chamada *JSettlers* [4];
- Para a validação dos resultados, medimos a quantidade de pontos de vitória obtidos por cada agente nas partidas executadas; Cada agente começa uma partida com 2 pontos e ganha a partida se obtiver 10 ou mais pontos, podendo obter até 12 pontos no total.
- Contra agentes aleatórios, nosso agente com MCTS vence cerca de 54% das partidas, já nosso agente UCT vence cerca de 88% destas partidas, com a distribuição de pontos demonstrada na Figura 2, onde cruzamos a quantidade de pontos pelo número de partidas de cada agente.

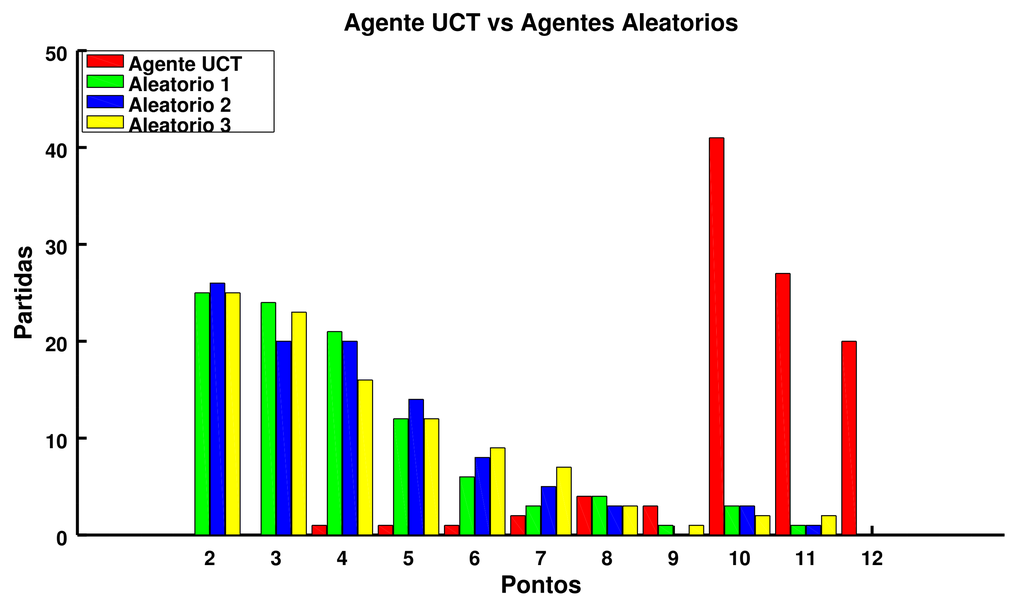

Figura 2: Performance do Agente UCT.

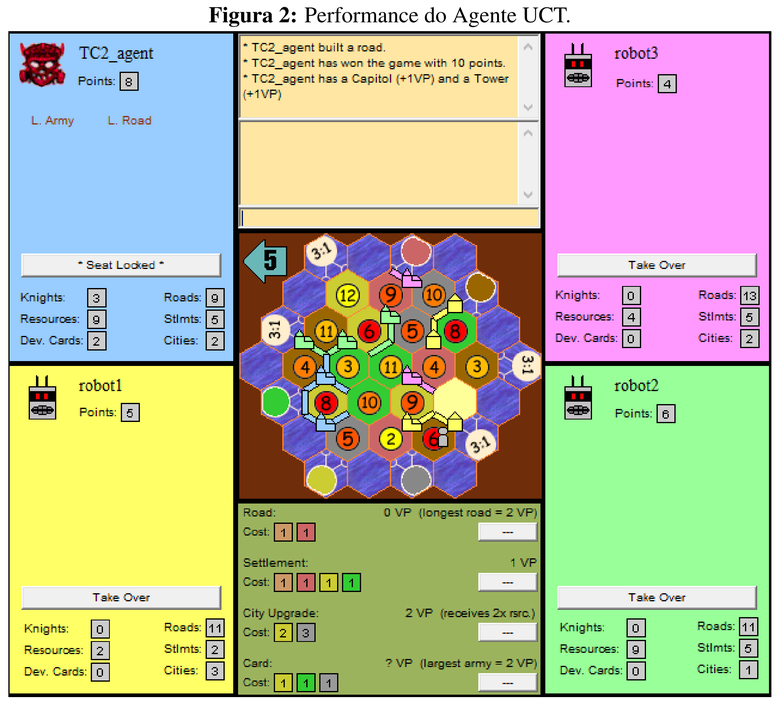

Figura 3: Vitória do agente UCT no cliente JSettlers.

Conclusão

Demonstramos empiricamente a capacidade do nosso agente de formular estratégias competitivas através do resultado das partidas realizadas em nossos experimentos. Podemos concluir com estes resultados que o algoritmo *Monte Carlo Tree Search* e suas variações são ideias para jogar Catan e tiveram performance superior a técnicas tradicionais, como Minimax, além de ter potencial competitivo contra outras implementações que utilizam estratégias *hard-coded*.

Referências

[1] C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(1):1–43, March 2012.

[2] Guillaume Chaslot, Er Bakkes, Istvan Szita, and Pieter Spronck. Monte-carlo tree search: A new framework for game ai. In *In Proceedings of AIIDEC-08*, pages 216–217, 2008.

[3] István Szita, Guillaume Chaslot, and Pieter Spronck. Monte-carlo tree search in settlers of catan. In *Proceedings of the 12th International Conference on Advances in Computer Games*, ACG'09, pages 21–32, Berlin, Heidelberg, 2010. Springer-Verlag.

[4] Robert Thomas and Kristian Hammond. Java settlers: A research environment for studying multi-agent negotiation. In *Proceedings of the 7th International Conference on Intelligent User Interfaces*, IUI '02, pages 240–240, New York, NY, USA, 2002. ACM.