Universidade Federal de Uberlândia
Inteligência Artificial – PGC112 – Prof. Anderson Santos
Trabalho Final (20 pontos) – Primeiro Semestre de 2015

Enunciado

O *tic-tac-toe Rubik's Cube* é uma versão mais sofisticada do conhecido 'Jogo da Velha' na qual, em vez de 9 posições para marcar jogadas de dois oponentes, existem 64 posições. Estas posições estão distribuídas em quatro quadros (níveis) com cada quadro possuindo quatro linhas e quatro colunas.

O jogador que conseguir marcar uma sequência de quatro símbolos com as posições jogadas distribuídas em um intervalo constante, terá ganho o jogo.
Um exemplo de jogada com posições distribuídas em um intervalo constante é a jogada presente na Figura 1, onde o intervalo das posições de cada jogada sempre é de 19 posições (4+19 = 23 +19 = 42 +19 = 61):

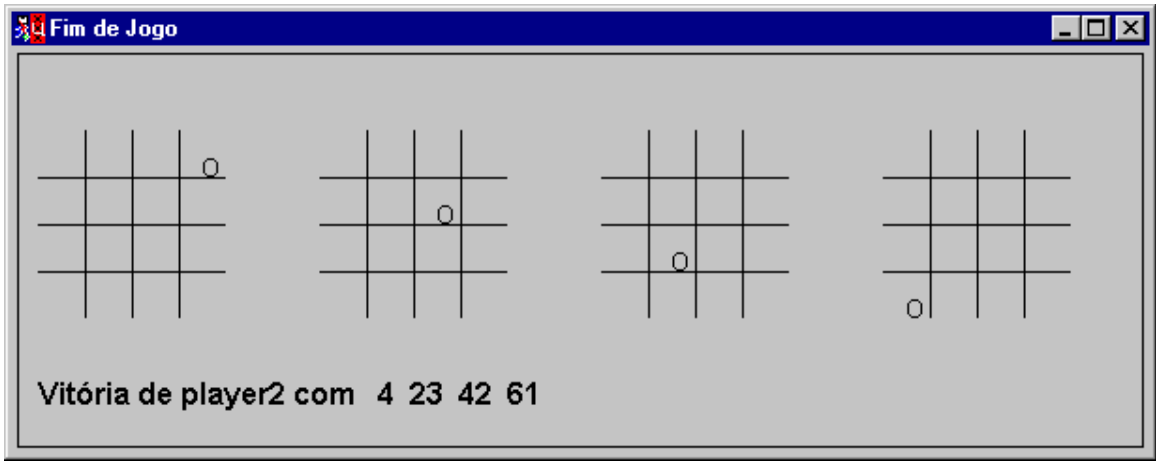

Figura 1: Exemplo de jogada vencedora no jogo Rubik Cube.

O trabalho final do nosso curso de Inteligência Artificial compreende o desenvolvimento de um agente jogador de Rubik's Cube desenvolvido na linguagem Prolog. O jogo deve ser feito para uma pessoa confrontar o agente (*default*) ou para o seu agente jogador confrontar outros agentes, desenvolvidos pelos demais alunos do curso. O usuário deve possuir opção de configurar qual destes modos de jogo será utilizado.

De modo a viabilizar o confronto entre agentes jogadores automáticos alguns padrões devem ser seguidos:

i. O estado do jogo será representado em um arquivo de texto denominado "game.txt". Este arquivo de texto simples terá uma única linha com 64 espaços em branco quando o jogo se inicia. À medida que o jogo prossegue, os espaços em branco devem ser substituídos por "X" (letra X maiúscula) ou "O" (letra O maiúscula). Um exemplo de arquivo "game.txt" é mostrado na Figura 2.

Figura 2: Exemplo do estado de um jogo.

ii. Terminada uma jogada cada agente jogador deve registrar seu movimento no arquivo “game.txt” e liberar o arquivo para escrita/leitura para o agente adversário.

iii. A escrita e leitura de arquivos deve acontecer no diretório onde o código dos dois agentes estiver presente (diretório local);

iv. Um agente pode tentar “perceber” que uma jogada foi feita pelo agente adversário por meio da comparação do estado anterior do arquivo “game.txt” ou pode receber um aviso (via teclado) para proceder com a sua vez de jogar.

v. Cada agente jogador deve implementar pelo menos o algoritmo *alfa-beta pruning* com capacidade para analisar a sua melhor jogada pelo menos após a jogada do adversário (três níveis). O usuário deve ser capaz de escolher qual heurística controlará a estratégia de defesa e ataque do agente e com quantas níveis.

vi. Ao menos uma representação visual do jogo deve ser fornecida para o usuário. A implementação de interface gráfica com janelas é opcional, mas se funcional valerá pontos adicionais.

Devem ser entregues o código em Prolog e documentação de todos os procedimentos implementados, bem como o relacionamento entre estes procedimentos no formato de um grafo direcionado com uma seta partindo do procedimento ‘‘chamador’' para o ‘‘chamado’'.

Ao final do curso (data ainda a ser selecionada), acontecerá um torneio entre os agentes jogadores de todos os alunos. O vencedor do torneio ganhará o direito de ficar com conceito A na disciplina, o segundo lugar ganhará o direito de ficar com conceito B e o terceiro lugar ganhará o direito de ficar com conceito C, ainda que as pontuações pela soma das demais atividades não proporcionem tais classificações. Caso o segundo e terceiro lugar possuam pontuação para o conceito prêmio, então o conceito prêmio será automaticamente incrementado para o próximo patamar, exemplo de C para B e de B para A. Em caso de empate generalizado, o conceito prêmio será revogado.