Universidade Federal do Rio Grande do Sul **Inteligência Artificial**
Instituto de Informática **Prof. Anderson Rocha Tavares**
Departamento de Informática Aplicada

Trabalho 1 – Parte 2
Poda alfa-beta em Othello/Reversi

Instruções preliminares

Para este trabalho, um kit com o servidor de partidas e um agente 'random' está disponível no moodle (arquivo `kit_othello.zip`).

As implementações podem ser feitas na sua linguagem de programação favorita. Porém, recomenda-se o uso de Python 3. Como incentivo, o kit fornece a "engine" do jogo em Python. Use-a e foque "apenas" na implementação da poda alfa-beta em si. De qualquer forma, seu agente deverá respeitar o protocolo de jogadas do torneio, definido neste enunciado. Além disso, certifique-se que seu código pode ser compilado (caso use uma linguagem compilada), e executado em uma máquina GNU/Linux.

Caso use uma linguagem de programação compilada, e/ou precise de instalar bibliotecas adicionais (assuma que os códigos serão executados em uma maquina virtual com uma instalação padrão do Ubuntu e interpretador Python 3.8 com Miniconda), escreva um script `prepara.sh` que faça essas etapas.

Introdução

O objetivo do trabalho é explorar a implementação de poda alfa-beta. Você deve implementar um agente capaz de jogar Othello (também conhecido como Reversi).

Basicamente, o jogo consiste em um tabuleiro onde dois jogadores (preto e branco) tentam capturar o maior número de posições com suas peças. O tabuleiro é formado por um arranjo 8x8 de posições cujas células centrais estão preenchidas por duas peças brancas e pretas, conforme a Figura 1.

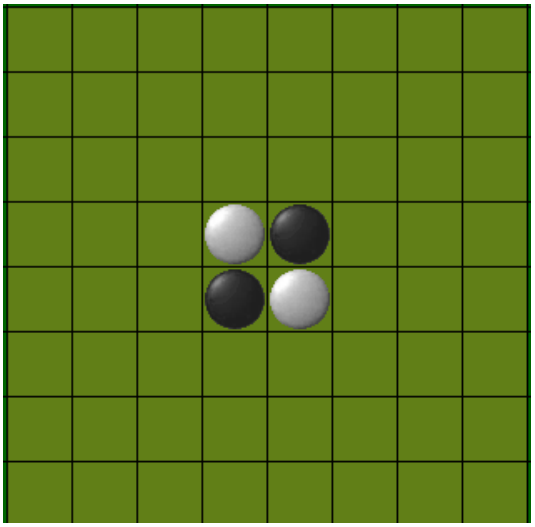

Figura 1 - Estado inicial do Othello

As pretas começam jogando e, em cada turno, um jogador deve colocar a próxima peça somente em posições onde uma peça adversária seja capturada. Uma ou mais peças do adversário são capturadas quando existe uma linha reta – horizontal, vertical ou diagonal – entre a peça colocada pelo jogador e uma das suas outras peças. Todas as peças capturadas mudam de cor e o jogo continua alternadamente. Caso um jogador não possua uma jogada válida, ele passa a vez. O jogo termina quando não houver jogadas válidas para ambos os jogadores. O vencedor é aquele com o maior número de peças no tabuleiro.

Acesse os seguintes links para entender mais sobre o jogo:

- http://en.wikipedia.org/wiki/Reversi (regras, história, etc)
- http://www.mah-jongg.ch/reversi (jogo online para praticar no navegador)
- Aplicativo Reversi Free (disponível para Android na Play Store)

Tarefa

- Implemente o algoritmo Minimax com poda alfa-beta. O algoritmo receberá o estado do jogo e a cor que deve fazer a jogada. Devido ao vasto número de estados no jogo, não será possível explorar completamente a árvore de busca. Portanto, você deve determinar uma estratégia que defina a profundidade máxima da busca, assim como a função de avaliação dos estados. Diversas características do tabuleiro podem ser utilizadas como função de avaliação (número de peças, quinas, posições estáveis, etc.) e várias estratégias de parada podem ser implementadas (profundidade máxima fixa, variada, quiescence search, singular extension, etc). Cabe ao grupo decidir qual é o melhor método a ser implementado.

- Haverá um torneio entre os programas dos estudantes. Para isso, você deve fazer com que seu programa siga o protocolo de jogadas definido a seguir.

Torneio

Os diferentes programas (agentes) irão competir entre si através de troca de arquivos mediada por um servidor. Quando for sua vez de jogar, seu programa será chamado, recebendo o caminho de um arquivo com uma representação padrão do tabuleiro e a cor com a qual a jogada deve ser feita (`black` ou `white`). Seu programa deve ler esse arquivo e escrever em arquivo uma jogada na representação padrão do servidor.

Você deverá escrever um script `launch.sh` que recebe os dois parâmetros (arquivo com estado e cor) e chama o seu programa. Observe a chamada de exemplo, que dará como entrada o `arquivo_estado_tabuleiro` e pedirá que seu agente jogue com as pretas:

```
./launch.sh arquivo_estado_tabuleiro black
```

No arquivo com o estado do tabuleiro, `W` representa uma peça branca (white), `B` uma peça preta (black) e `.` (ponto) representa um espaço livre. No exemplo abaixo, temos a representação do estado inicial da Figura 1. Observe que cada linha do arquivo possui exatamente 8 caracteres seguidos por uma quebra de linha.

```
........
........
........
...WB...
...BW...
........
........
........
```

Seu agente terá 5 segundos para ler este arquivo, e escrever a jogada em um arquivo chamado `move.txt`, que deverá ser criado no mesmo diretório do seu `launch.sh`. O arquivo `move.txt` conterá simplesmente a coordenada `x,y` correspondendo ao lugar onde sua peça será colocada. As coordenadas vão de 0 a 7. O eixo x cresce da esquerda para a direita e o eixo y cresce de cima para baixo. O exemplo abaixo mostra o sistema de coordenadas.

```
  01234567 --> eixo x
0 ........
1 ........
2 ........
3 ...WB...
4 ...BW...
5 ........
6 ........
7 ........
|
|
v
eixo y
```

Considerando o estado inicial, um dos movimentos válidos para as pretas é em `x = 5` e `y = 4`. O arquivo `move.txt` de um agente que decidiu por este movimento terá o seguinte conteúdo:

```
5,4
```

O arquivo deverá conter somente os dois números separados por vírgula, sem espaço entre eles. Caso não haja jogada válida para seu jogador, escreva `-1,-1` no arquivo.

Fluxo de jogo

Durante uma partida, seu `launch.sh` será chamado diversas vezes, uma para cada jogada a ser feita pelo seu jogador. Dessa forma, após a jogada, seu agente deverá encerrar sua execução, reiniciando quando o `launch.sh` for chamado novamente. De fato, o servidor encerrará seu script `launch.sh` após os 5 segundos que seu agente tem para fazer a jogada.

Basicamente, o servidor escreverá o estado do tabuleiro no seu diretório, chamará o seu `launch.sh`, aguardará sua jogada e então lerá a jogada feita através do `move.txt` escrito pelo seu programa. Em seguida o servidor escreverá o novo estado no diretório do adversário, chamará o `launch.sh` do adversário e lerá a jogada feita por ele. Isso se repetirá até o fim do jogo.

Entrega

Você deve entregar um arquivo `.zip` contendo:

- O código fonte completo.
- Um script `prepara.sh` que instale as bibliotecas necessárias e compile todo o seu código, caso programe em linguagem compilada.
- O script `launch.sh` que executará o programa que faz uma jogada, para o torneio.
- Um arquivo `relatorio.pdf` com um pequeno relatório da sua implementação. O relatório deve conter:
  - Nomes, cartões de matrícula e turma dos integrantes do grupo;
  - Bibliotecas que precisem ser instaladas para (compilar e) executar sua implementação.
  - Descrição da função de avaliação, da estratégia de parada; eventuais melhorias (quiescence search, singular extensions, etc); decisões de projeto e dificuldades encontradas; e bibliografia completa (incluindo sites).

ATENÇÃO: os shell scripts (`.sh`) e o `relatorio.pdf` devem se localizar na raiz do seu arquivo `.zip`! Isto é, o conteúdo do seu arquivo `.zip` deverá ser o seguinte:

```
prepara.sh
launch.sh
relatorio.pdf
[arquivos do código fonte] <<pode criar subdiretórios se necessário
```

Conteúdo do arquivo `.zip` a ser enviado.

Observações gerais

- O trabalho deve ser feito em trios.
- O tempo de 5 segundos é estipulado tendo como referência uma máquina linux com a seguinte configuração: processador Core i7 2.93 Ghz e 4Gb de memória RAM.
- Fiquem atentos à política de plágio!
- Haverá um torneio entre todos os agentes recebidos. Agentes que não conseguirem aderir ao protocolo serão desclassificados do torneio.
- A nota depende da correta implementação e de um bom relatório. Isto é, um mau desempenho no torneio não resultará em penalidade na nota (desde que seu agente consiga aderir ao protocolo). No entanto, como incentivo, os melhores colocados no torneio receberão pontuação extra.

Dicas

- Leia o `README` do `kit_othello.zip`, ele contém instruções para a execução do servidor e do jogador 'random'.
- Você pode usar as funções do `common/board.py` para gerar a lista de jogadas válidas e os estados resultantes das mesmas.
- No torneio, após realizar a jogada, seu agente deverá encerrar sua execução. Ele deverá se inicializar novamente suas estruturas e preparar outra busca quando seu `launch.sh` for

chamado novamente com o novo estado do tabuleiro, obtido após a jogada do oponente.
- Para facilitar a detecção de erros, é permitido imprimir o que for necessário no terminal, uma vez que as partidas são jogadas via arquivos. Você pode aproveitar o protocolo do torneio para iniciar o seu agente a partir de um estado que esteja causando erros (você escreve o arquivo com o estado problemático e executa seu `launch.sh` manualmente).
- Cuidado com a representação interna do tabuleiro adotada pelo seu agente. No protocolo do torneio, a jogada é `x,y` (coluna, linha) enquanto a representação via matriz endereça primeiramente a linha e depois a coluna.
- Use o jogador 'random' disponível no kit para testar a operação básica do seu agente, com relação ao torneio. O 'random' não é competitivo, portanto é encorajado que os grupos joguem partidas entre si antes do torneio (usem o servidor fornecido no kit), mas a troca de código é proibida.

Política de Plágio

Trios poderão apenas discutir questões de alto nível relativas a resolução do problema em questão. Poderão discutir, por exemplo, questões sobre as estruturas de dados utilizadas, as heurísticas de avaliação de estados, vantagens e desvantagens, etc. Não é permitido que os trios utilizem quaisquer códigos fonte provido por outros trios, ou encontrados na internet.

Pode-se fazer consultas na internet ou em livros apenas para estudar o modo de funcionamento das técnicas de IA, e para analisar o pseudo-código que as implementa. Não é permitida a análise ou cópia de implementações concretas (em quaisquer linguagens de programação) da técnica escolhida. O objetivo deste trabalho é justamente implementar as técnicas do zero e descobrir as dificuldades envolvidas na sua utilização para resolução do problema em questão. Toda e qualquer fonte consultada pelo trio (tanto para estudar os métodos a serem utilizadas, quanto para verificar a estruturação da técnica em termos de pseudo-código) precisa obrigatoriamente ser citada no relatório.

Usamos rotineiramente um sistema anti-plágio que compara o código-fonte desenvolvido pelos trios com soluções enviadas em edições passadas da disciplina, e também com implementações disponíveis online.

Qualquer nível de plágio (ou seja, utilização de implementações que não tenham sido 100% desenvolvidas pelo trio) poderá resultar em nota zero no trabalho. Caso a cópia tenha sido feita de outro trio da disciplina, todos os envolvidos (não apenas os que copiaram) serão penalizados. Esta política de avaliação não é aberta a debate. Se você tiver quaisquer dúvidas se uma determinada prática pode ou não, ser considerada plágio, não assuma nada: pergunte ao professor e aos monitores.

Note que, considerando-se os pesos das avaliações desta disciplina (especificados e descritos no Plano de Ensino) nota zero em qualquer um dos trabalhos de implementação abaixa muito a média final dos projetos práticos, e se ela for menor que 6, não é permitida prova de recuperação. Ou seja: caso seja detectado plágio, há o risco direto de reprovação. Os trios deverão desenvolver o trabalho sozinhos.