Universidade Federal de Minas Gerais **Inteligência Artificial**
Instituto de Ciências Exatas **2º Semestre de 2015**
Departamento de Ciência da Computação **Prof. Luiz Chaimowicz**

Trabalho Prático 1
Trilha / 9 Men's Morris

Instruções preliminares

As implementações podem ser feitas na sua linguagem de programação favorita. Em troca, 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.

Para este trabalho, um kit com o servidor do torneio e um agente 'random' estará disponível no moodle em breve.

Introdução

O objetivo do trabalho é explorar a implementação das técnicas de busca com adversários. Você deve implementar um agente capaz de jogar Trilha (conhecido em inglês como 9 Men's Morris ou Mill, entre outros nomes).

O tabuleiro do jogo consiste em 24 pontos dispostos em três quadrados concêntricos, conectados por linhas horizontais e verticais. Cada jogador possui 9 peças e deve tentar formar “moinhos” - 3 peças sob a mesma linha na horizontal ou vertical. Ao formar um moinho, o jogador tira uma peça adversária do jogo. O jogador vence se reduzir o oponente a duas peças ou bloquear todos os seus movimentos.

O jogo é feito em 3 fases: colocação, movimentação e vôo.

- Fase 1 – colocando peças: o jogo começa com o tabuleiro vazio. Quem joga com as brancas começa. Alternadamente, cada jogador coloca uma peça em um ponto do tabuleiro na sua vez, até usar todas as 9 que possui.
- Fase 2 – movimentando peças: alternadamente, os jogadores podem mover peças para pontos adjacentes. Quando um jogador estiver reduzido a 3 peças, ele entra na Fase 3.
- Fase 3 – voando com as peças: o jogador que estiver com apenas 3 peças pode “voar”, colocando-as em qualquer ponto vazio do tabuleiro. Como o jogador com 3 peças está prestes a perder, esta é uma tentativa de melhorar suas chances.

Em qualquer fase do jogo, quando um jogador coloca 3 peças sob a mesma linha na horizontal ou na vertical, ele forma um moinho e retira uma peça do adversário do jogo. A peça a ser retirada não pode estar em um moinho, a menos que o adversário possua apenas moinhos. Uma peça retirada do jogo não retorna nunca mais.

Regra para o empate: neste trabalho, consideramos a regra descrita em [1]: o jogo termina em empate quando uma “jogada cíclica” é feita, isto é, ambos os jogadores fazem e desfazem um movimento e o estado do tabuleiro se repete. Note que, quando um moinho é desfeito e então feito novamente, uma peça adversária é removida e nesse caso o estado do tabuleiro não se repete.

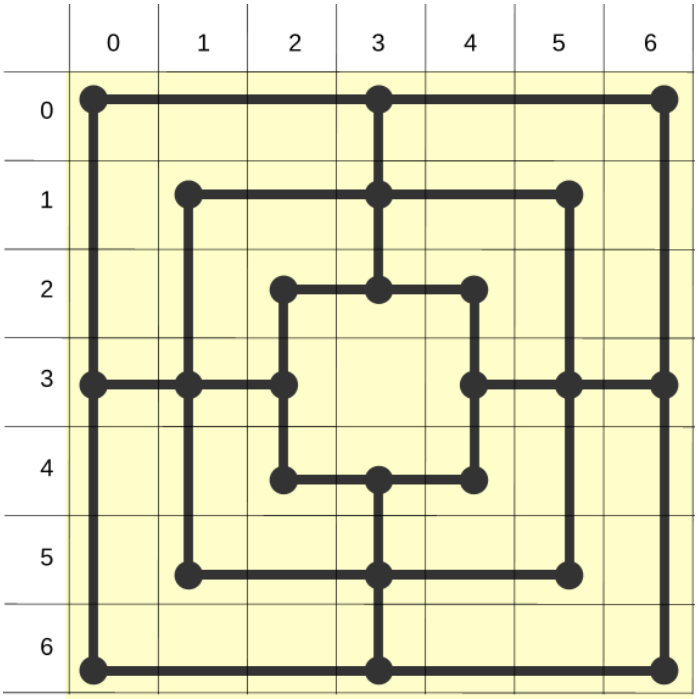

Figura 1: Estado inicial do jogo e o sistema de coordenadas

Links úteis:

- https://en.wikipedia.org/wiki/Nine_Men's_Morris - descrição em inglês
- https://pt.wikipedia.org/wiki/Trilha_(jogo) - descrição em português
- http://www.mathplayground.com/logic_nine_mens_morris.html - jogo online para praticar
- https://play.google.com/store/apps/details?id=org.doublemill.client - aplicativo Android

Tarefa

- Implemente o algoritmo Minimax com poda alfa-beta para jogar Trilha. 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 (dê uma olhada nos slides com a estratégia básica) e várias estratégias de parada podem ser implementadas (profundidade máxima fixa, variada, quiescence search, singular extension, etc). Cabe ao estudante decidir qual é o melhor método a ser implementado.

- Implemente um programa para um humano jogar Trilha contra a sua implementação da poda alfa-beta. Seu programa deve permitir a escolha da cor com a qual os jogadores vão jogar. Escreva um script `mill.sh` que execute este programa.

- Haverá um torneio entre os programas dos estudantes. Para isso, você deve respeitar o protocolo de jogadas definido na próxima seção.

Torneio

Os diferentes programas (agentes/bots) 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, a cor com a qual a jogada deve ser feita (`black` ou `white`), o tipo da jogada e as três últimas jogadas (para detectar iminência de empate). Seu programa deve ler o conteúdo do arquivo e escrever em outro arquivo a jogada em uma representação padrão do servidor.

Você deverá escrever um script `launch.sh` que recebe o arquivo com as informações e chama o seu programa. Observe a chamada de exemplo, que dará como entrada o `arquivo_informacoes.txt`:

```
./launch.sh arquivo_informacoes.txt
```

Arquivo de entrada

A primeira linha contém a cor com a qual seu bot deverá jogar (`black` ou `white`), um espaço em branco e o tipo de jogada que o bot deve fazer (`placement num` para a fase 1, onde `num` é a quantidade de peças que você ainda pode colocar, `movement` para as fases 2 e 3, e `mill` quando você formou um moinho). Na segunda linha, a jogada feita há 3 passos atrás. Na terceira linha, a jogada feita há 2 passos atrás e na quarta linha, a jogada feita há 1 passo atrás, para checagem de iminência de empate. Nessas 3 linhas de histórico, cada linha será da forma `cor tipo jogada`, onde o formato da jogada é o mesmo que você irá escrever quando fizer a sua. Da quinta à décima linha estará a representação do tabuleiro, onde `W` representa uma peça branca (white), `B` uma peça preta (black), `o` (letra 'o' minúscula) representa um espaço livre, `|` e `-` representam as linhas horizontais e verticais que conectam pontos adjacentes; e `+` representa o centro do tabuleiro, onde não se pode jogar. No exemplo abaixo, temos o arquivo com a representação do tabuleiro para as brancas posicionarem uma peça, faltando 8 peças a serem posicionadas. Duas jogadas foram feitas, mas como o histórico tem 3 entradas, a entrada mais antiga é preenchida com `none`. Observe que a representação do tabuleiro possui 6 linhas e 6 colunas.

```
white placement 8
none
white placement 2 3
black placement 4 3
o--o--o
|o-o-o|
||oWo||
ooo+oBo
||ooo||
|o-o-o|
o--o--o
```

Seu agente terá 5 segundos para ler este arquivo, calcular a jogada e escrevê-la em um arquivo chamado `move.txt`, que deverá ser criado no mesmo diretório do seu `launch.sh`.

Coordenadas do tabuleiro e formato da jogada

A orientação do tabuleiro segue a de uma matriz: a primeira coordenada é a linha e a segunda é a coluna onde será feita a jogada. A numeração começa com zero (ver Fig. 1 para uma ilustração do sistema de coordenadas).

Há 3 tipos de jogadas que seu agente pode escrever no `move.txt` e elas dependem da situação do jogo. Para a jogada `placement`, escreva apenas as coordenadas (linha,coluna) separadas por

vírgula onde sua peça será colocada. Para a jogada `movement`, escreva as coordenadas de origem (linha,coluna), um espaço em branco e as coordenadas de destino (linha,coluna). Para a jogada `mill`, escreva as coordenadas (linha,coluna), da peça do adversário que será removida. Exemplos são dados abaixo.

```
0,0
```

Conteúdo de um `move.txt` válido para uma jogada tipo `placement`, onde a posição 0,0 está vazia.

```
0,0 0,3
```

Conteúdo de um `move.txt` válido para uma jogada tipo `movement`. A posição de origem deve ter uma peça sua e a posição de destino deve ser adjacente a ela e estar vazia. Se você tiver apenas 3 peças, a posição de destino não precisa ser adjacente.

```
5,5
```

Conteúdo de um `move.txt` válido para uma jogada tipo `mill`, supondo que o adversário tenha uma peça em 5,5.

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 arquivo com as informações no seu diretório, chamará o seu `launch.sh`, aguardará sua jogada e lerá a jogada feita no `move.txt` escrito pelo seu programa. Em seguida o servidor escreverá o arquivo com as informações do novo estado no diretório do adversário, chamará o `launch.sh` dele 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.
- O script `mill.sh` que execute o programa que permite um humano jogar contra seu agente.
- O script `launch.sh` que executará o programa que faz uma jogada, para o torneio.
- Um script `compila.sh` que compile todo o seu código, caso programe em linguagem compilada.
- Documentação detalhada sobre a implementação (`.pdf`) contendo:
  - Descrição dos algoritmos, da função de avaliação, da estratégia de parada; eventuais melhorias (quiescence search, singular extensions, etc); eventuais decisões de projeto e dificuldades encontradas; testes do algoritmo (usando o servidor fornecido no `kit_tp1.zip` e eventuais outros testes); bibliografia completa (sites inclusos).

Observações gerais

- Para a graduação, o trabalho pode ser feito em duplas.
- O tempo de 5 segundos é estipulado tendo como referência uma máquina linux do laboratório de graduação, com a seguinte configuração: processador Core2Duo 2.93 Ghz e 4Gb de memória DDR2 800MHz (http://crc.dcc.ufmg.br/infraestrutura/laboratorios/labs_unix).
- Penalização por atraso: 2^dias – 1.
- Trabalhos copiados de colegas ou da internet serão penalizados com a nota zero.
- Haverá um torneio entre todos os agentes recebidos. Por isso, agentes que não conseguirem aderir ao protocolo serão desclassificados do torneio. Isso inclui a realização de jogadas inválidas.
- A nota depende da correta implementação e de uma boa documentação. 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.
- Trilha é um jogo já resolvido (veja [1]). Mesmo assim, você deve implementar a poda alfa-beta e não vale usar uma jogada 'perfeita' pré-computada.

Dicas

- Leia o `README` do `kit_tp1.zip` que será disponibilizado. Ele contém instruções para a execução do servidor e do jogador 'random'.
- No torneio, após realizar a jogada, seu agente deverá encerrar sua execução. Ele deverá 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 durante o torneio, uma vez que as partidas são jogadas via arquivos. Você pode aproveitar o protocolo do torneio para iniciar o seu agente a partir de uma situação que esteja causando erros (você escreve o arquivo com as informações sobre 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 `linha,coluna`.
- Use o jogador 'random' disponível no kit deste trabalho para testar a operação básica do seu agente, com relação ao torneio. O 'random' não é competitivo, portanto é encorajado que os estudantes joguem partidas entre si antes do torneio (usem o servidor fornecido no kit), mas a troca de código é proibida.

Referências

[1] GASSER, Ralph. Solving nine men's morris. Computational Intelligence, v. 12, n. 1, p. 24-41, 1996. Disponível em: http://library.msri.org/books/Book29/files/gasser.pdf