Agentes Autônomos – 2014.1

Equipe: máximo duas pessoas

Data de Entrega: 5 de maio de 2014

Questão 1: Considere uma pequena modificação do jogo da velha, envolvendo um fator de acaso. Um dado jogador inicia a primeira jogada e a partir daí uma moeda é lançada. Se o resultado for cara, o jogo procede normalmente. Se o resultado for coroa, o símbolo (# ou 0) inserido na última jogada é então invertido. Aplique o MINIMAX (com dois lances – two-ply) para escolher a melhor jogada do primeiro jogador.

Questão 2: Monkey-and-Bananas (Exercício adaptado do cap. 10 - AIMA)

O problema **Monkey-and-Bananas** consiste em ajudar um macaco a obter bananas que estão fora do seu alcance em um dado momento. Uma caixa é disponível para ajudar o macaco a alcançar as bananas caso ele suba nela. Inicialmente, o macaco está em A, as bananas em B, e a caixa em C. O macaco e a caixa têm altura BAIXA, mas se o macaco sobe na caixa ele terá altura ALTA, que é a mesma altura das bananas. As ações disponíveis para o macaco incluem MOVER de um lugar para outro, EMPURRAR um objeto de um lugar para outro, SUBIR e DESCER de um objeto, e SEGURAR e SOLTAR um objeto. SEGURAR resulta em estar com um objeto se o macaco e o objeto estão no mesmo lugar e na mesma altura.

a. Escreva as definições em PDDL das seis ações acima (**arquivo de domínio**).
b. Codifique o estado inicial do problema e objetivo do problema onde o macaco deve estar de volta na posição A com as bananas (**instância de problema**).
c. Execute o problema no FF, com os seus arquivos (http://fai.cs.uni-saarland.de/hoffmann/ff.html). Uma implementação correta deve rodar no FF.

Questão 3: Pintando Muros

Considere a seguinte definição (incompleta) de um domínio de planejamento. Um mural consiste em 3 painéis, numerados de 1 a 3. O agente pode ser posicionado diante de qualquer um dos painéis. O fato do agente estar posicionado diante do painel i é representado por $At(i)$. O agente pode se mover um painel para a esquerda ou um painel para direita, de cada vez. Quando ele está diante de um painel que não está pintado, ele pode pintá-lo. O domínio do problema é descrito pela seguinte HTN (onde os refinamentos para Go(x, 1) não foram fornecidos):

Left(x):
PRECOND: $At(x) \wedge x > 1$
EFFECT: $\neg At(x) \wedge At(x - 1)$

Right(x):
PRECOND: $At(x) \wedge x < 3$
EFFECT: $\neg At(x) \wedge At(x + 1)$

Paint(x):
PRECOND: $At(x) \wedge \neg Painted(x)$
EFFECT: $Painted(x)$

Refinement(PaintAll;
PRECOND: *At*(*x*)
STEPS: [*Go*(*x*; 1); *Paint*(1); *Right*(1); *Paint*(2); *Right*(2); *Paint*(3)])

Refinement(*Go*(x; 1);
??? ???)

a) Adicione refinamento(s) apropriado(s) para a HLA Go(x, 1).
b) Reescreva a representação do domínio usando representação com variáveis de estados, como adotado no Pyhop.
c) Implemente o problema no Pyhop. Assuma que o objetivo é PaintAll. Simule para o estado inicial "Agente diante do painel 1, mural não pintado", depois para "Agente diante do painel 2, mural não pintado" e finalmente como estado inicial "Agente diante do painel 3, mural não pintado"

OBS.1: A primeira questão não precisa ser implementada no computador. Anexe a resolução dela em PDF.

OBS 2: Enviar arquivos fonte das questões 2 e 3. Na questão 3, anexe também o trace do Pyhop, para os níveis de verbosidade 1, 2 e 3. Todos os arquivos fonte devem estar bem comentados, explicando a solução desenvolvida.

OBS.3: Incluir o termo **AA-Exercicio** no assunto da mensagem ao enviar os arquivos.