U.PORTO FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO	Mestrado Integrado em Eng. Informática e Computação **Inteligência Artificial**	2019/2020 (3º Ano) 2º Sem
DOCENTES: Luís Paulo Reis, Henrique Lopes Cardoso		

Trabalho Nº1

Métodos de Pesquisa Heurística para Resolução de Problemas
Métodos de Pesquisa Adversarial para Jogos
Métodos de Otimização/Meta-Heurísticas

Tema

O primeiro trabalho prático de IART consiste no desenvolvimento de um programa relacionado com uma de três temáticas possíveis.

Temática 1: Métodos de Pesquisa Heurística para Resolução de Jogo do Tipo Solitário

Um jogo do tipo solitário caracteriza-se pelo tipo de tabuleiro e de peças, pelas regras de movimentação das peças (operadores/jogadas possíveis) e pelas condições de terminação do jogo com derrota (impossibilidade de resolver, número máximo de movimentos foi atingido, tempo limite foi atingido) ou vitória (solitário resolvido) e com a respetiva pontuação. Tipicamente, no caso de vitória é atribuída uma pontuação dependendo do número de movimentos, recursos gastos, bónus recolhidos e/ou tempo despendido.

Para além de implementar um jogo do tipo solitário para um jogador humano, o programa deve ser capaz de resolver diferentes versões/quadros desse jogo, utilizando métodos de pesquisa adequados, focando na comparação entre métodos de pesquisa não informada (pesquisa primeiro em largura, primeiro em profundidade, aprofundamento progressivo, custo uniforme) e métodos de pesquisa heurística (pesquisa gulosa, A*), com diferentes funções heurísticas. Os métodos aplicados devem ser comparados a diversos níveis, com ênfase para a qualidade da solução obtida, número de operações analisadas e tempo despendido para obter a solução.

A aplicação deve ter uma visualização em modo de texto ou gráfico para mostrar a evolução do tabuleiro e realizar a comunicação com o utilizador/jogador. Deve permitir um modo de jogo em que o PC resolve o solitário sozinho utilizando o método e sua configuração selecionado pelo utilizador. Poderá também, opcionalmente, permitir um modo de jogo Humano em que o utilizador pode resolver o jogo, eventualmente pedindo "dicas" ao PC.

Temática 2: Jogo para dois jogadores

Um jogo de tabuleiro caracteriza-se pelo tipo de tabuleiro e de peças, pelas regras de movimentação das peças (operadores/jogadas possíveis) e pelas condições de terminação do jogo com a respetiva pontuação. Pretende-se neste trabalho implementar um jogo para dois jogadores e resolver diferentes versões desse jogo, utilizando o método de pesquisa Minimax com cortes $\alpha\beta$ e variantes.

Devem ser desenvolvidos modos de jogo humano-humano, humano-computador e computador-computador, apresentando o computador diferentes níveis de dificuldade. Deve ser comparado o desempenho do computador com diferentes níveis de dificuldade, ou seja, com diferentes funções de avaliação e diferentes níveis de profundidade e/ou variantes do algoritmo Minimax. Deve ser dada ênfase à análise da qualidade da solução obtida (vitórias, empates e derrotas e outros parâmetros de qualidade, como por exemplo o número de jogadas para obter a vitória/derrota) e tempo médio despendido para obter as soluções/jogadas.

A aplicação a desenvolver deve ter uma visualização adequada em modo de texto ou gráfico, para mostrar a evolução do tabuleiro e realizar a comunicação com o utilizador/jogador. Deve permitir os modos de jogo indicados acima, permitindo a seleção do modo de jogo, tipo de cada jogador e para o computador qual o nível e heurística de cada jogador. Deverá também permitir que os diversos níveis do computador joguem entre si). Poderá também permitir, para as jogadas do humano, a apresentação de "dicas".

Temática 3: Problema de otimização

Um problema de otimização é caracterizado pela existência de um conjunto (tipicamente grande) de soluções possíveis, comparáveis entre si, de entre as quais uma ou mais são consideradas soluções ótimas (globais). Dependendo do problema em concreto, uma função de avaliação permite estabelecer essa comparação entre soluções. Em muitos destes problemas é praticamente impossível encontrar a solução ótima, ou garantir que a solução encontrada é a ótima e, como tal, o objetivo é tentar maximizar a função de avaliação.

Pretende-se neste trabalho implementar um sistema para resolver um problema de otimização, utilizando diferentes algoritmos/ meta-heurísticas: subida da colina ("hill-climbing"), arrefecimento simulado ("simulated annealing"), pesquisa tabu e algoritmos genéticos (podendo incluir outros algoritmos ou variações destes). Devem ser resolvidas várias instâncias do problema escolhido e comparados os resultados obtidos em cada instância pelos diferentes algoritmos. Devem ser testadas e compradas diferentes parametrizações dos algoritmos e comparada a qualidade média da solução obtida e o tempo médio despendido para obter as soluções.

A aplicação a desenvolver deve ter uma visualização adequada em modo de texto ou gráfico, para mostrar a evolução da qualidade da solução obtida e a solução/soluções final(is) obtida(s) e realizar a comunicação com o utilizador. Deve permitir a seleção e parametrização dos algoritmos e a seleção da instância do problema a resolver.

Linguagem de Programação

Pode ser utilizada qualquer linguagem de programação e sistema de desenvolvimento, incluindo, a nível de linguagens, C++, Java, Python, C#, Prolog, entre outras. A escolha da linguagem e do ambiente de desenvolvimento a utilizar é da inteira responsabilidade dos estudantes.

Constituição dos Grupos

Os grupos devem ser compostos por 2 ou 3 estudantes. Não se aceitam grupos individuais ou compostos por 4 estudantes. Os grupos podem ser compostos por estudantes de turmas diferentes (no máximo de duas turmas diferentes) mas todos os estudantes têm de estar presentes nas sessões de acompanhamento e apresentação/demonstração do trabalho. A constituição de grupos compostos por estudantes de diferentes turmas não é aconselhada, dadas as dificuldades logísticas de realização de trabalho que tal pode provocar.

Checkpoint

Cada grupo deve submeter no Moodle uma breve apresentação (máx. 5 slides), em formato PDF, que será utilizada na aula para analisar, em conjunto com o docente, o andamento do trabalho. A apresentação deve conter: (1) especificação do trabalho a realizar (definição do jogo ou do problema de otimização a resolver) (2) trabalho relacionado com referências a trabalhos encontrados na pesquisa (artigos, páginas web e/ou código fonte), (3) formulação do problema como um problema de pesquisa (representação do estado, estado inicial, teste objetivo, operadores (nomes, pré-condições, efeitos e custos), heurísticas/função de avaliação) ou de otimização (representação da solução/indivíduo, funções de vizinhança/mutação e de cruzamento, restrições rígidas, funções de avaliação), e (4) trabalho de implementação já realizado (linguagem de programação, ambiente de desenvolvimento, estruturas de dados, estrutura de ficheiros, entre outros).

Entrega Final

Cada grupo deve submeter no Moodle dois ficheiros: uma apresentação (máx. 10 slides), em formato PDF, e o código implementado, devidamente comentado, incluindo um ficheiro "readme.txt" com instruções sobre como o compilar, executar e utilizar. Com base na apresentação submetida, os estudantes devem realizar uma demonstração (cerca de 10 minutos) do trabalho, na aula prática, ou em outro período a designar pelos docentes da disciplina.

O ficheiro com a apresentação final deve incluir, para além do já referido para a entrega intercalar (*checkpoint*), detalhes sobre: (5) a abordagem (heurísticas, funções de avaliação, operadores, …) e (6) algoritmos implementados (algoritmos de pesquisa, minimax, metaheurísticas), bem como (7) resultados experimentais, recorrendo a tabelas/gráficos apropriados e comparando os diversos métodos, heurísticas, algoritmos e respetivas parametrizações para diferentes cenários/problemas. A apresentação deve incluir um slide de conclusões e outro de referências consultadas e materiais utilizados (software, sítios web, artigos científicos, …).

Problemas Sugeridos

Temática 1: Jogo do Tipo solitário

1A) BoxWorld 2: http://hirudov.com/others/BoxWorld2.php

1B) Folding Blocks: https://play.google.com/store/apps/details?id=com.popcore.foldingblocks

1C) Bubble Blast: https://play.google.com/store/apps/details?id=com.magmamobile.game.BubbleBlast2&hl=en

1D) Pudding Monster: https://play.google.com/store/apps/details?id=com.zeptolab.monsters.free.google&hl=en

Temática 2: Jogo para Dois Jogadores

2A) Eximo: https://www.boardgamegeek.com/boardgame/137916/eximo

2B) Neutron (tabuleiro de dimensão variável - 5x5, 7x7): https://www.di.fc.ul.pt/~jpn/gv/neutron.htm

2C) Parquet (tabuleiro de dimensão 4x4 e/ou 6x6): https://www.boardgamegeek.com/boardgame/154092/parquet

2D) Pivit: https://www.boardgamegeek.com/boardgame/135473/pivit

Temática 3: Problema de Otimização

3A) Compiling Google:
https://storage.googleapis.com/coding-competitions.appspot.com/HC/2019/hashcode2019_final_task.pdf
https://storage.googleapis.com/coding-competitions.appspot.com/HC/2019/final_round_2019.in.zip

3B) Photo slideshow:
https://storage.googleapis.com/coding-competitions.appspot.com/HC/2019/hashcode2019_qualification_task.pdf
https://storage.googleapis.com/coding-competitions.appspot.com/HC/2019/qualification_round_2019.in.zip

3C) City Plan:
https://storage.googleapis.com/coding-competitions.appspot.com/HC/2018/hashcode2018_final_task.pdf
https://storage.googleapis.com/coding-competitions.appspot.com/HC/2018/final_round_2018.in.zip

3D) Self-driving rides:
https://storage.googleapis.com/coding-competitions.appspot.com/HC/2018/hashcode2018_qualification_task.pdf
https://storage.googleapis.com/coding-competitions.appspot.com/HC/2018/qualification_round_2018.in.zip