

Projeto Nº 1: Época Especial

Inteligência Artificial - Escola Superior de Tecnologia de Setúbal
2019/2020

Prof. Joaquim Filipe
Eng. Filipe Mariano

1. Jogo da Torre: Descrição Geral

O Jogo da Torre é uma variante do problema matemático conhecido como o Passeio do Cavalo, cujo objetivo é, através dos movimentos do Cavalo, visitar todas as casas possíveis de um tabuleiro similar ao de xadrez. Nesta versão a peça de xadrez utilizada é a **Torre** e decorrerá num tabuleiro de **7** linhas e **7** colunas (7x7), em que cada casa possui uma pontuação. Nesta secção pretende-se dar uma perspetiva geral do que é este jogo, deixando para a secção seguinte a explicação do que se pretende que seja desenvolvido no projeto de Inteligência Artificial relativo à resolução de um problema neste contexto por procura em Espaço de Estados.

1.1. Tabuleiro

O jogo possui as seguintes características:

- O tabuleiro tem as dimensões 7x7 em que os valores de cada casa são entre **11** e **77**, sem repetição e sem números que incluam os dígitos 0, 8 e 9. Por exemplo, 59 é um valor que não pode existir.
- Cada vez que é iniciado um jogo é construído um novo tabuleiro com o valor das casas distribuído aleatoriamente.
- O objectivo do jogo é acumular mais pontos que o adversário, usando uma Torre de xadrez. Cada jogador tem uma Torre da sua cor (branco ou preto).

1.2. Desenrolar do Jogo

- O jogo começa com a colocação da Torre branca numa casa da **1ª linha (A1-G1 do tabuleiro)**. Esta casa é escolhida pelo jogador com a Torre branca.
- Se a casa escolhida tiver um número com dois dígitos diferentes, por exemplo 57, então, em consequência, o número simétrico 75 é apagado do tabuleiro, tornando esta casa inacessível durante o resto do jogo. Ou seja, nenhuma Torre pode terminar outra jogada nessa casa.
- Se uma Torre for colocado numa casa com um número "duplo", por exemplo 66, então **deve ser removido o número duplo de maior valor**. Depois de um jogador deixar a casa para se movimentar para outra, a casa onde estava fica também inacessível para o jogo, ficando o número da casa apagado.
- Após a primeira jogada (colocar Torre branca) segue-se a jogada do adversário com colocação da Torre preta numa casa da **7ª linha (A7-G7)**.
- Depois de ambas as Torres serem colocadas, todas as jogadas seguintes são efectuadas através de um movimento da Torre (usando as regras tradicionais do Xadrez para essa peça). **Uma Torre não pode saltar para uma casa vazia (sem número) e também não pode fazê-lo para uma casa que esteja ameaçada pela Torre adversária.**
- A cada jogada de um jogador repete-se a regra do simétrico ou duplo.
- Um jogador ganha pontos por cada casa visitada pela sua Torre (igual ao valor da casa). Os pontos são contabilizados apenas pelas casas visitadas, não pelos números simétricos ou duplos removidos.

1.3. Determinar o Vencedor

O jogo termina quando não for possível movimentar qualquer uma das Torres no tabuleiro, sendo o vencedor o jogador que ganhou **mais** pontos.

2. Objetivo desta Fase do Projeto: Jogador Único

No âmbito deste projeto vamos considerar o Jogo da Torre como uma versão simplificada do que foi mencionado anteriormente, em que o principal objetivo consiste em atingir uma dada pontuação definida para o problema, no menor número possível de jogadas, i.e. de movimentos da Torre. Para isso, será necessário, a partir da casa inicial, deslocar a Torre branca ao longo do tabuleiro em jogadas sucessivas até não ser possível efetuar qualquer movimento ou até atingir o objetivo.

A transformação do jogo num problema, para esta fase do projeto, pressupõe as seguintes diferenças ao nível das regras:

- Existe apenas um jogador (Torre branca);
- O jogador começa por colocar a Torre numa casa da primeira linha do tabuleiro;
- O estado final é atingido quando a Torre chega a uma casa que lhe permite obter uma pontuação igual ou superior ao objetivo definido;
- Se não for possível atingir o objetivo, o programa deverá informar o utilizador de que o problema não tem solução;
- Os objetivos para os problemas A-F, fornecidos em anexo e que têm de ser resolvidos no âmbito deste projeto, são: A: 100, B: 200, C:300, D:425, E: 525, F:1000;
- A inicialização do processo de resolução do problema consiste na aplicação de um operador especial, de colocação da Torre nas casas disponíveis na primeira linha, que tenham uma pontuação numérica. Esse operador permite fazer a geração de todos os sucessores do nível 1 a partir do nó raiz do grafo que representa cada um dos problemas acima referidos. A partir daí, são aplicáveis os operadores de movimentação da Torre.

Pretende-se que os alunos escrevam e testem um programa, em **LISP**, para apresentar a sequência de estados ou de jogadas que conduzem da posição inicial do problema até à posição final, recorrendo aos algoritmos de procura lecionados nas aulas conforme se indica detalhadamente mais adiante. A solução óptima consiste no caminho com menor número de jogadas entre o estado inicial e o estado final.

3. Formulação do Problema

3.1. Tabuleiro

O tabuleiro é representado sob a forma de uma lista composta por **7** outras listas, cada uma delas com **7** átomos. Cada uma das listas representa uma linha do tabuleiro, enquanto que cada um dos átomos possui o valor da respetiva casa. Por outras palavras, o tabuleiro é representado por uma lista de listas em LISP, sendo que cada átomo representa uma casa em que a linha corresponde ao índice da sublista e a coluna ao índice do átomo dentro da linha. **As casas já visitadas terão o valor NIL, enquanto que a casa onde se encontra a Torre deverá ter o valor T.**

3.2. Solução Encontrada

A solução pode representar-se por uma sequência de estados, desde o estado inicial até ao estado final, ou então – por razões de legibilidade – pela lista de operações (i.e. jogadas) realizadas sobre as peças do tabuleiro. Cada jogada é identificada pela casa destino da Torre (uma letra representando a coluna A-G e um número representando a linha 1-7).

Pretende-se encontrar a solução ótima, ou seja a de menor custo, tendo em conta os custos dos operadores referidos na secção seguinte.

3.3. Operadores

Os operadores representam os movimentos possíveis num determinado estado. Os custos dos operadores são considerados unitários, qualquer que seja o algoritmo usado.

3.4. Estrutura do Programa

O programa deverá estar dividido em três partes, cada uma num ficheiro diferente:

1. Uma parte com a implementação dos métodos de procura e com a implementação das métricas de análise de eficiência, i.e. a parte do programa que é independente do domínio de aplicação;
2. Outra para implementar tudo o que envolve a resolução do problema concreto, incluindo a definição dos operadores e heurísticas, específicos do domínio de aplicação;
3. E a terceira parte para fazer a interação com o utilizador e para proceder à escrita e leitura de ficheiros.

O projeto deverá obrigatoriamente implementar e apresentar um estudo comparativo do comportamento dos três métodos: procura em largura primeiro (BFS), procura em profundidade primeiro (DFS) e A*. Além disso, deverá ainda implementar e apresentar um estudo comparativo do comportamento de um dos algoritmos de procura com limitação de memória, nomeadamente um dos seguintes (à escolha de cada grupo de alunos):

- Simplified Memory Bounded A* (SMA*),
- Interactive Deepening A* (IDA*), ou
- Recursive Best First Search (RBFS).

Tal como para os algoritmos obrigatórios, deverá apresentar-se a análise de desempenho de cada algoritmo opcional num ficheiro produzido automaticamente pelo programa, sendo descontado 0,1 valores por cada problema não resolvido.

No caso dos métodos informados, o programa deverá utilizar funções heurísticas modulares, ou seja, que possam ser colocadas ou retiradas do programa de procura como módulos.

As heurísticas não devem estar embutidas de forma rígida no programa de procura. Exige-se a utilização de duas heurísticas, uma fornecida no fim do presente documento e outra desenvolvida pelos alunos.

O projeto deverá incluir a implementação de cada um dos algoritmos, de forma modular, permitindo que o utilizador escolha qualquer um deles, conjuntamente com os respetivos parâmetros (heurística, profundidade, etc.) para a resolução de um dado problema.

4. Experiências

Pretende-se que o projeto estude, para cada problema fornecido em anexo, o desempenho de cada algoritmo e, no caso do A*, de cada uma das heurísticas propostas, apresentando, em relação a cada problema indicado em anexo, a solução encontrada e dados estatísticos sobre a sua eficiência, nomeadamente o número de nós gerados, o número de nós expandidos, a penetrância, o fator de ramificação média e o tempo de execução.

Os projetos deverão apresentar os dados acima referidos num ficheiro produzido automaticamente pelo programa, sendo descontado 0,5 valor por cada problema não resolvido. No caso de ser apresentada a solução, mas não o estudo de desempenho das heurísticas o desconto é de apenas 0,2 valor por cada caso.

5. Heurísticas

Sugere-se usar como heurística de base, uma heurística que privilegia visitar as casas com o maior número de pontos. Para um determinado tabuleiro x :

$$h(x) = o(x)/m(x)$$

em que:

$m(x)$ é a média por casa dos pontos que constam no tabuleiro x ,

$o(x)$ é o número de pontos que faltam para atingir o valor definido como objetivo.

Esta heurística pode ser melhorada para refletir de forma mais adequada o conhecimento acerca do domínio e assim contribuir para uma maior eficiência dos algoritmos de procura informados. Além da heurística acima sugerida, deve ser definida pelo menos uma segunda heurística que deverá melhorar o desempenho dos algoritmos de procura informados.

6. Grupos

Os projetos deverão ser realizados em grupos de, no máximo, duas pessoas sendo contudo sempre sujeitos a avaliação oral individual para confirmação da capacidade de compreensão dos algoritmos e de desenvolvimento de código em LISP.

7. Datas

Entrega do projeto: 07 de Setembro de 2020, até as 23:55.

8. Documentação a Entregar

A entrega do projeto e da respetiva documentação deverá ser feita através do Moodle, na zona do evento "Entrega do Projeto 1 de Época Especial". Todos os ficheiros a entregar deverão ser devidamente arquivados num ficheiro comprimido (ZIP com um tamanho máximo de 5Mb), até à data acima indicada. O nome do arquivo deve seguir a estrutura `nomeAluno1_numeroAluno1_nomeAluno2_numeroAluno2_E1`.

8.1. Código Fonte

Os ficheiros de código devem ser devidamente comentados e organizados da seguinte forma:

projeto.lisp Carrega os outros ficheiros de código, escreve e lê ficheiros, e trata da interação com o utilizador.

puzzle.lisp Código relacionado com o problema.

procura.lisp Código independente do problema. Deve conter a implementação de:

1. Algoritmo de Procura de Largura Primeiro (BFS)
2. Algoritmo de Procura de Profundidade Primeiro (DFS)
3. Algoritmo de Procura do Melhor Primeiro (A*)
4. Um dos algoritmos SMA*, IDA* ou RBFS
5. As funções de cálculo das métricas Penetrância e Fator de Ramificação.

8.2. Exercícios

Deverá haver um ficheiro de exercícios, com a designação **problemas.dat**, contendo todos os exemplos de tabuleiros que se quiser fornecer ao utilizador, organizados de forma sequencial, e que este deverá escolher mediante um número inserido no interface com o utilizador.

Esse número representa o número de ordem na sequência de exemplos. O ficheiro deverá ter várias listas, separadas umas das outras por um separador legal, e não uma lista de listas. Essas listas serão tantas quantos os problemas fornecidos.

Na discussão oral, os docentes irão solicitar que se adicione mais um exemplo, numa dada posição do ficheiro, que deverá imediatamente passar a ser selecionável através do interface com o utilizador e poder ser resolvido por qualquer dos algoritmos.

8.3. Manuais

No âmbito da Unidade Curricular de Inteligência Artificial pretende-se que os alunos pratiquem a escrita de documentos recorrendo à linguagem de marcação **Markdown**, que é amplamente utilizada para os ficheiros **ReadMe** no **GitHub**. Na Secção 4 do guia de Laboratório nº 2, encontrará toda a informação relativa à estrutura recomendada e sugestões de ferramentas de edição para **Markdown**.

Para além de entregar os ficheiros de código e de problemas, é necessário elaborar e entregar 2 manuais (o manual de utilizador e o manual técnico), em formato PDF juntamente com os sources em MD, incluídos no arquivo acima referido:

ManualTecnico.pdf O Manual Técnico deverá conter o algoritmo geral, por partes e devidamente comentado; descrição dos objetos que compõem o projeto, incluindo dados e procedimentos; identificação das limitações e opções técnicas. Deverá ser apresentada uma análise crítica dos resultados das execuções do programa, onde deverá transparecer a compreensão das limitações do projeto. Deverão usar uma análise comparativa do conjunto de execuções do programa para cada algoritmo e cada problema, permitindo verificar o desempenho de cada algoritmo e das heurísticas. Deverá, por fim, apresentar a lista dos requisitos do projeto (listados neste documento) que não foram implementados.

ManualUtilizador.pdf O Manual do Utilizador deverá conter a identificação dos objetivos do programa, juntamente com descrição geral do seu funcionamento; explicação da forma como se usa o programa (acompanhada de exemplos); descrição da informação necessária e da informação produzida (ecrã/teclado e ficheiros); limitações do programa (do ponto de vista do utilizador, de natureza não técnica).

9. Avaliação

Tabela 1: Grelha de classificação.

Funcionalidade	Valores
Representação do estado e operadores	2
Funções referentes ao problema	2
Procura em profundidade e largura	2
Procura com A* e heurística dada	2
Implementação de nova heurística	2
Procura com limitação de memória	1
Resolução dos problemas dados	3
Resolução do problema dado na avaliação oral	1
Qualidade do código	2
Interação com o utilizador	1
Manuais (utilizador e técnico)	2
Total	20

Os problemas a resolver estão descritos na Secção de Anexos. A avaliação do projeto levará em linha de conta os seguintes aspectos:

- Data de entrega final – Existe uma tolerância de 2 dias em relação ao prazo de entrega, com a penalização de 1 valor por cada dia de atraso. Findo este período a nota do projeto será 0.
- Correção processual da entrega do projeto – (Moodle; manuais no formato correto). Anomalias processuais darão origem a uma penalização que pode ir até 3 valores.
- Qualidade técnica – Objetivos atingidos; Código correto; Facilidade de leitura e manutenção do programa; Opções técnicas corretas.
- Qualidade da documentação – Estrutura e conteúdo dos manuais que acompanham o projeto.
- Avaliação oral – Eficácia e eficiência da exposição; Compreensão das limitações e possibilidades de desenvolvimento do programa. Nesta fase poderá haver lugar a uma revisão total da nota de projeto.

10. Recomendações Finais


Com este projeto pretende-se motivar o paradigma de programação funcional. A utilização de variáveis globais, de instruções de atribuição do tipo `set`, `setq`, `setf`, de ciclos, de funções destrutivas ou de quaisquer funções com efeitos laterais é fortemente desincentivada dado que denota normalmente uma baixa qualidade técnica.

As únicas exceções permitidas a estas regras poderão ser a utilização da instrução `loop` para implementar os ciclos principais dos algoritmos implementados (como alternativa a uma solução puramente recursiva), em conjugação com as variáveis globais `Abertos` e `Fechados` para manutenção das listas de nós.


ATENÇÃO: Suspeitas confirmadas de plágio serão penalizadas com a anulação de ambos os projetos envolvidos (fonte e destino), e os responsáveis ficam sujeitos à instauração de processo disciplinar.

Anexos – Problemas


Problema A: Objetivo 100 pontos

	A	B	C	D	E	F	G	
1	55						77	1
2								2
3								3
4								4
5								5
6								6
7	66						22	7
	A	B	C	D	E	F	G	


Problema B: Objetivo 200 pontos

	A	B	C	D	E	F	G	
1	55						77	1
2								2
3								3
4		74	75	76	57	47		4
5								5
6								6
7	66						22	7
	A	B	C	D	E	F	G	


Problema C: Objetivo 300 pontos

	A	B	C	D	E	F	G	
1								1
2								2
3			27	64	46			3
4			75	76	57			4
5			71	67	17			5
6								6
7								7
	A	B	C	D	E	F	G	

Problema D: Objetivo 425 pontos f

	A	B	C	D	E	F	G	
1								1
2			77		66			2
3		72	27	64	46	55		3
4			75	76	57			4
5		11	71	67	17	44		5
6			22		33			6
7								7
	A	B	C	D	E	F	G	

Problema E: Objetivo 525 pontos

	A	B	C	D	E	F	G	
1	55						77	1
2		72				73		2
3			65	64	63			3
4		74	75	76	57	47		4
5			66	67	69			5
6		71				70		6
7	66						22	7
	A	B	C	D	E	F	G	

Problema F:

Deverá ser estudado a partir de um tabuleiro completo gerado aleatoriamente.

Problema G:

Será disponibilizado durante a discussão do projeto.