U.PORTO

FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Artificial Intelligence/ Inteligência Artificial

Lecture 3: Search Problems

Luís Paulo Reis

lpreis@fe.up.pt

Director of LIACC – Artificial Intelligence and Computer Science Lab.
Associate Professor at DEI/FEUP – Informatics Engineering Department,
Faculty of Engineering of the University of Porto, Portugal
President of APPIA – Portuguese Association for Artificial Intelligence

Resolução de Problemas por Pesquisa

Estrutura da Apresentação:

- **Métodos de Resolução de Problemas**
- **Formulação de Problemas**
- **Espaço de Estados**
- **Pesquisa Não Informada:**
  - Primeiro em Largura, Primeiro em Profundidade, Custo Uniforme, Aprofundamento Iterativo, Pesquisa Bidirecional.
- **Pesquisa Inteligente:**
  - Pesquisa Gulosa, Algoritmo A*
- **Exemplos Práticos de Aplicação**

Resolução de Problemas por Pesquisa

- **Como é que um agente pode agir, estabelecendo objetivos e considerando possíveis sequencias de ações para atingir esses objetivos!**
- **Resolução de Problemas:**
  - Formulação de um problema como um problema de pesquisa
  - Pesquisa não Informada (estratégias de pesquisa)
  - Pesquisa Informada (pesquisa gulosa, algoritmo A*)
  - Algoritmos de Melhoria Iterativa
  - Jogos (em que é incluído um agente hostil!)

Agente para Resolução de Problemas

- **"Problem Solving Agent": Procura encontrar a sequência de ações que leva a um estado desejável!**
- **Formulação do Problema:**
  - Quais as ações possíveis? (qual o seu efeito sobre o estado do mundo?)
  - Quais os estados possíveis? (como representá-los?)
  - Como avaliar os Estados
- **Problema de pesquisa**
  - Solução: sequência de ações
- **Fase final é a execução!**
- **Formular → Pesquisar → Executar**

Problemas de Pesquisa

- **Muitos do problemas em ciências da computação podem ser definidos como:**
  - Um conjunto S de ESTADOS (possivelmente infinito)
  - Um estado INICIAL s € S
  - Uma relação de TRANSIÇÃO T ao longo deste espaço de estados
  - Um conjunto de estados FINAIS (objetivos): O € S
- **Problema pode ser representado por um GRAFO, onde os nodos representam estados e os arcos (conexões) os pares da relação de transição**
- **Problema pode ser resolvido através de pesquisa e um caminho entre o estado inicial e um estado objetivo**

Agente para Resolver Problemas

- **Formulação do Problema:**
  - Representação do Estado
  - Estado Inicial (Atual)
  - Teste Objetivo (define os estados desejados)
  - Operadores (Nome, Pré-Condições, Efeitos, custo)
  - Custo da Solução

Agente de Resolução de Problemas Simples

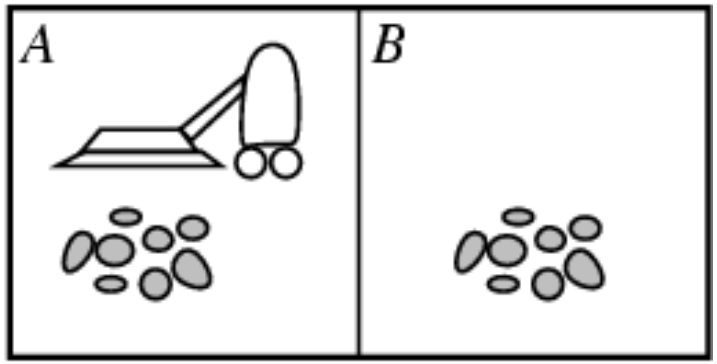
```
function SIMPLE-PROBLEM-SOLVING-AGENT(percept) returns an action
    static: seq, an action sequence, initially empty
            state, some description of the current world state
            goal, a goal, initially null
            problem, a problem formulation

    state ← UPDATE-STATE(state, percept)
    if seq is empty then do
        goal ← FORMULATE-GOAL(state)
        problem ← FORMULATE-PROBLEM(state, goal)
        seq ← SEARCH(problem)
    action ← FIRST(seq)
    seq ← REST(seq)
    return action
```

Formulação do Problema

- **Qual o conhecimento do agente sobre o estado do mundo e sobre as suas ações?**
- **Quatro tipos de problemas distintos:**
  - Problemas de estado único (ambiente determinístico e acessível)
  - Problemas de múltiplos estados (ambiente determinístico mas inacessível)
  - Problemas de contingência (ambiente não determinístico e inacessível, é necessário usar sensores durante a execução, solução é uma árvore ou política)
  - Problemas de exploração (espaço de estados desconhecido)

Exemplo: Problema do Aspirador

- **2 localizações, 3 Ações (left, right, suck), 8 Estados possíveis, Objetivo: limpar o lixo!**
- **Problema de:**
  - Estado Único se…
  - Múltiplos Estados se…
  - Contingência se…
  - Exploração se...



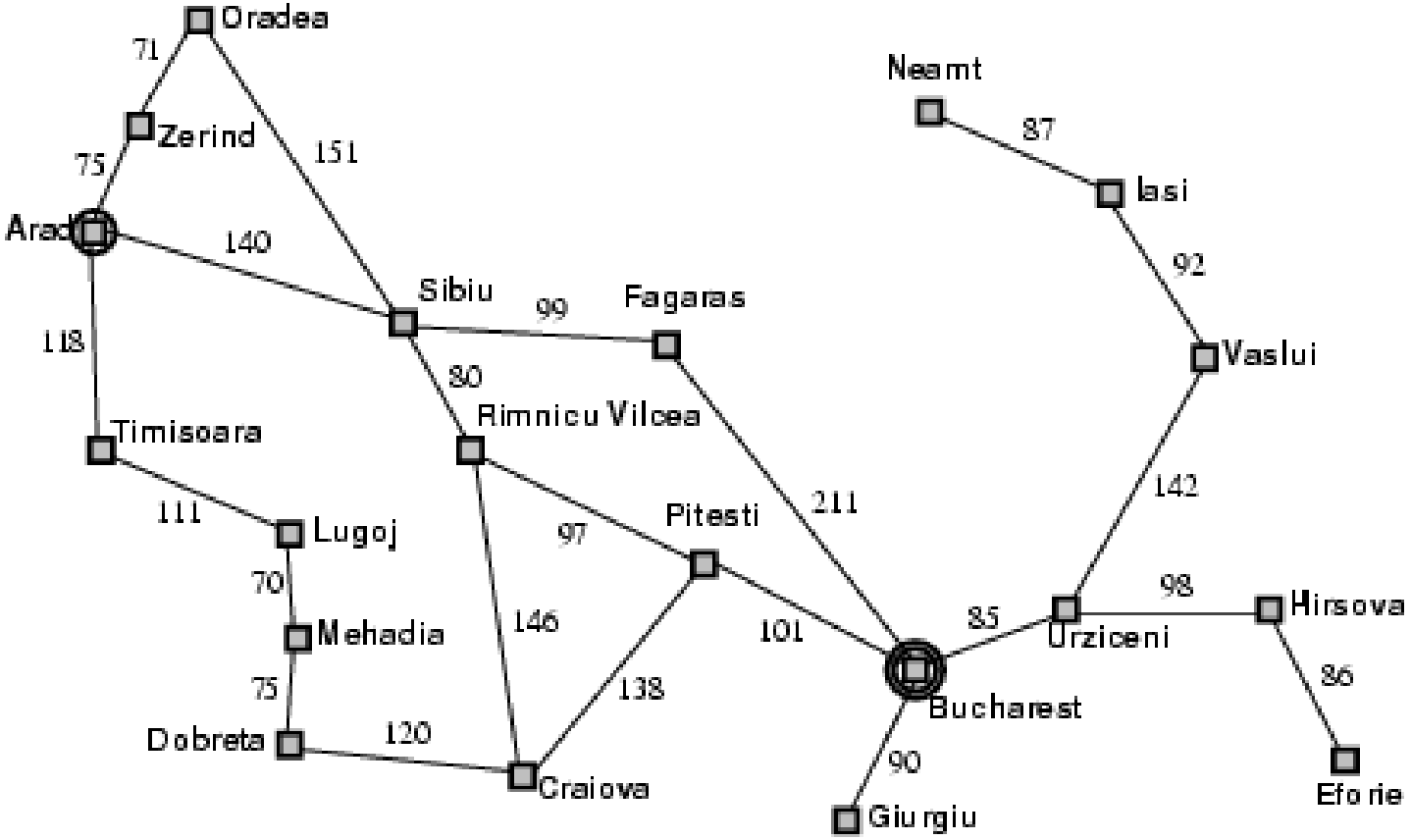
Problemas Bem Definidos (estado único)

- **Problema: Coleção de informação que o agente vai usar para decidir o que fazer!**
- **Formulação do Problema:**
  - Espaço de Estados:
    - Estado Inicial
    - Conjunto de Ações possíveis (operadores, função sucessores)
  - Teste do Objetivo
  - Função de Custo da Solução
- **datatype PROBLEM**

  components: INITIAL-STATE, OPERATORS, GOAL-TEST, PATH-COST-FUNCTION
- **Solução: Caminho do estado inicial até ao objetivo**
- **Custo Total = Custo da Solução + Custo da pesquisa**

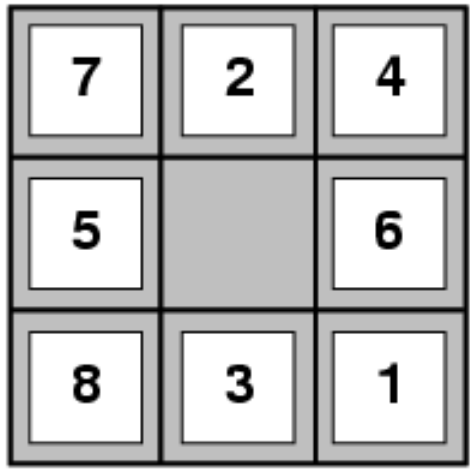
Exemplo: Mapa de Estradas da Roménia

- Estado Inicial: Arad; Estado Objetivo: Bucharest
- Operadores: Arcos com respetivos custos

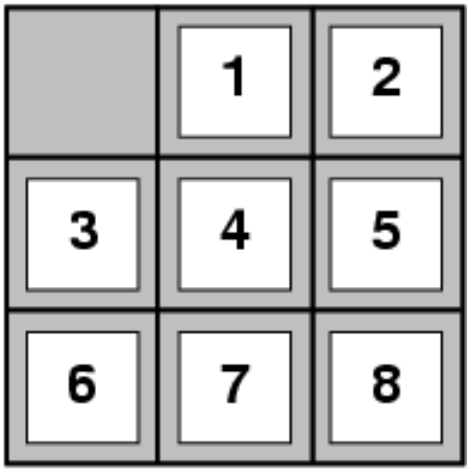


Problema do Puzzle-8

- **Estados, Operadores, Teste Objetivo, Custo da Solução**



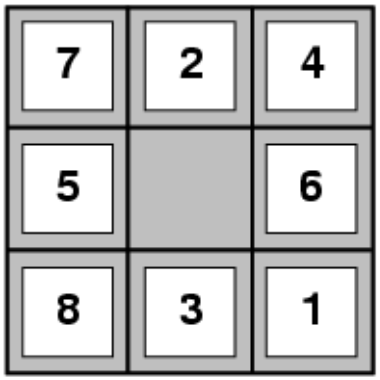
Start State



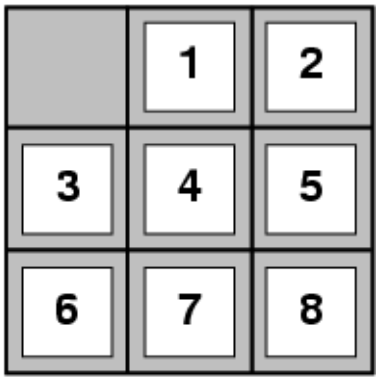
Goal State

Problema do Puzzle-8

- **Estados, Operadores, Teste Objetivo, Custo da Solução**
  - Estados: Especifica a posição de cada uma das peças e do espaço vazio (várias representações são possíveis)
  - Estado inicial: Representado na figura
  - Operadores - sucessores: gera os estados válidos que resultam da execução. São as quatro ações (mover espaço vazio para esquerda, direita, cima ou abaixo)
  - Teste de objetivo: Verifica se o estado corresponde à configuração objetivo (representado na figura)
  - Custo da solução: Cada passo custa 1, sendo o custo da solução o número de passos para resolver o problema



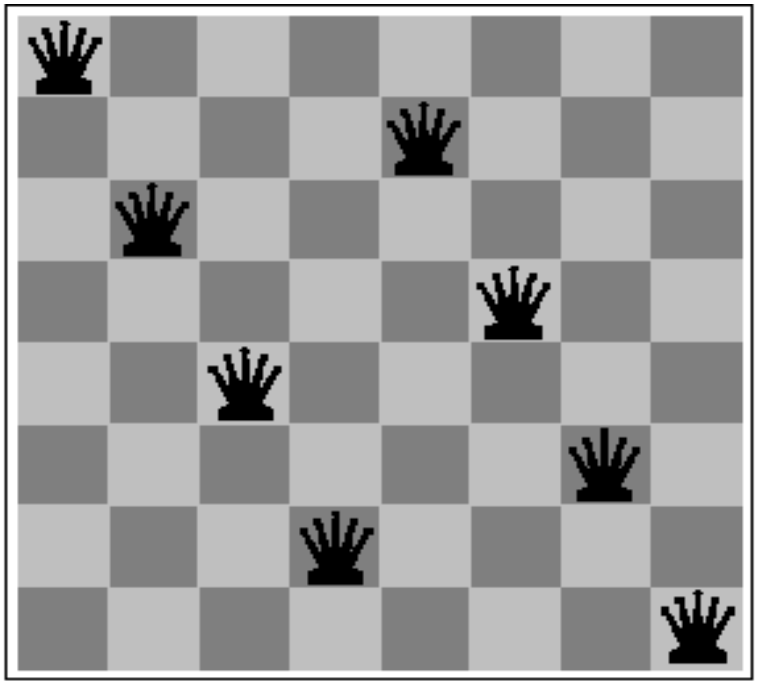
Start State



Goal State

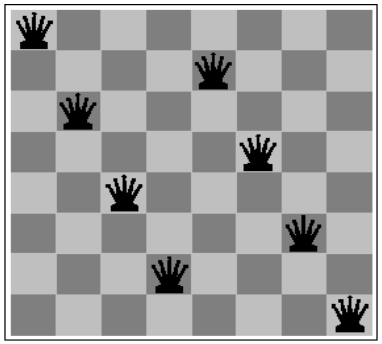
Problema das N-Rainhas

- **Estados, Operadores, Teste Objetivo, Custo da Solução**



Problema das N-Rainhas

- **Teste Objetivo: 8 Rainhas no tabuleiro sem nenhum ataque**
- **Custo da Solução: 0**
- **Formulação 1:**
  - Estado: Qualquer arranjo de 0 a 8 Rainhas no tabuleiro
  - Operador: Adicionar uma rainha em qualquer quadrado
    - Temos 64^8 sequências possíveis!



- **Formulação 2:**
  - Estado: Arranjos de 0 a 8 Rainhas, uma em cada coluna, sem ataques!
  - Operador: Adicionar uma rainha na coluna mais à esquerda que estiver vazia, sem atacar nenhuma outra
    - Temos 2057 sequências possíveis!
- **Formulação 3:**
  - Estado: Arranjos de 8 Rainhas no tabuleiro, uma em cada coluna!
  - Operador: Movimentar rainha atacada para casa da mesma coluna

Criptogramas

- **Encontrar dígitos (todos diferentes), um para cada letra de forma a que a soma seja correta!**
- **Estados: Puzzle com algumas letra substituídas por números**
- **Operadores: Substituir todas as ocorrências de uma letra por um dígito**
- **Teste Objetivo: Puzzle só contém dígitos e a soma está correta!**
- **Custo da Solução: 0 (todas as soluções são iguais)**

```
  F O R T Y
      T E N
+     T E N
----------------
  S I X T Y
```

```
C1 C2 C3 C4
   S E N D
+  M O R E
------------------
M  O N E Y
```

Criptogramas

- **Soluções dos Criptogramas:**
  - Criptograma 1: F=2, O=9, R=7, T=8, Y=6, E=5, N=0, S=3, I=1, X=4
    - Haverá mais soluções?
  - Criptograma 2: S=9, E=5, N=6, D=7, M=1, O=0, R=8, Y=2
- **Exercício: Inventar um criptograma!**

```
  F O R T Y
      T E N
+     T E N
----------------
  S I X T Y
```

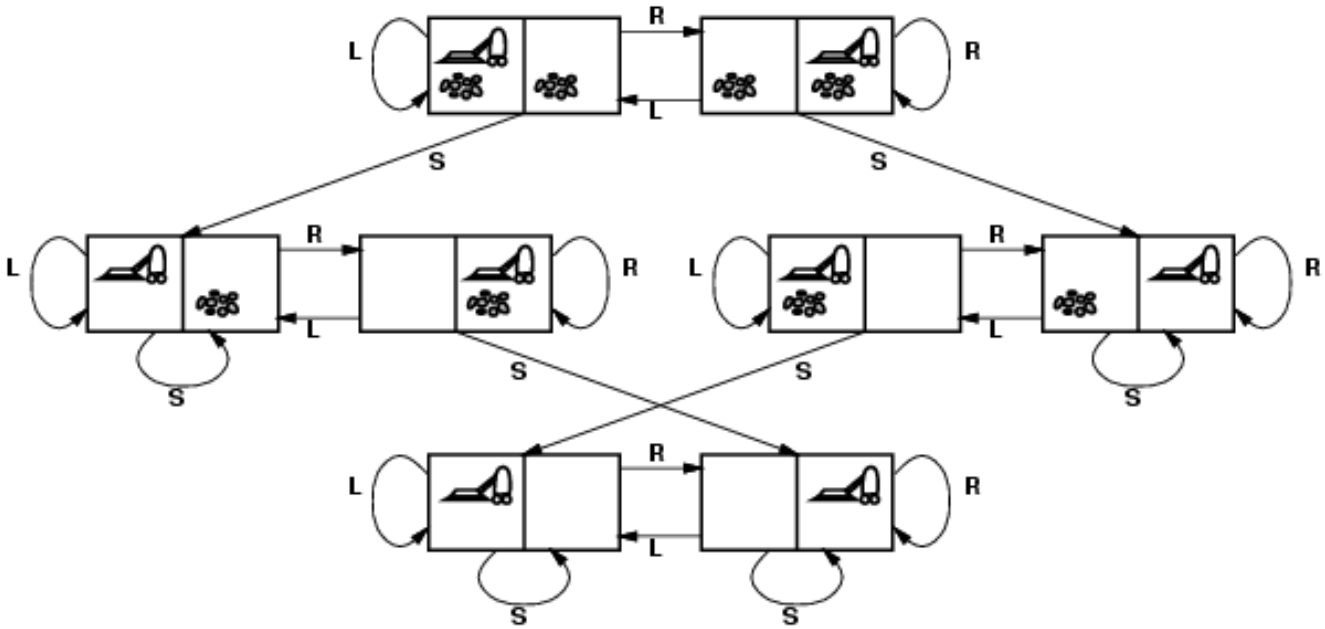
```
  2 9 7 8 6
      8 5 0
+     8 5 0
---------------
  3 1 4 8 6
```

```
C1 C2 C3 C4
   S E N D
+  M O R E
-----------------
M O N E Y
```

```
  1  0  1  1
   9 5 6 7
+  1 0 8 5
-----------------
 1 0 6 5 2
```

Exemplo: Mundo do Aspirador

- **Estado: 8 estados representados (definidos pela posição do robô e lixo)**
- **Estado inicial: Qualquer um**
- **Operadores: esquerda, direita, aspirar**
- **Teste Objetivo: Não há lixo em nenhum dos quadrados**
- **Custo da Solução: Cada ação custa 1 (custo total = número de passos da solução**



Exemplo: Mundo do Aspirador sem Sensores!

- **Problema de Múltiplos Estados**
  - Agora temos em cada instante um conjunto de estados possíveis!
- **Formulação do Problema**
  - Conjunto de Estados: Subconjunto dos estados representados
  - Operadores: esquerda, direita e aspirar
  - Teste Objetivo: Todos os estados do conjunto não podem ter lixo
  - Custo da Solução: Cada ação custa 1

Exemplo de Problemas do Mundo Real

- **Problema de Rotas/Caminhos**
  - Encontrar o melhor caminho de um ponto a outro (aplicações: google maps, redes de computadores, planeamento militar, viagens aéreas)
  - visitar cada ponto pelo menos uma vez num dado espaço (Ex: Caixeiro viajante visitar cada cidade exatamente uma vez, encontrar o caminho mais curto)

Exercício: Missionários e Canibais

- **Problema dos Missionários e Canibais**
- **Descrição:**
  - 3 missionários e 3 canibais estão numa das margens do rio com um barco que só leva 2 pessoas. Encontrar uma forma de levar os 6 para a outra margem do rio sem nunca deixar mais canibais do que missionários numa das margens durante o processo!
- **Formular este problema como um problema de pesquisa, definindo a representação do estado, estado inicial, os operadores (e respetivas pré-condições e efeitos), o teste objetivo e o custo da solução.**
- **Resolver o problema através de uma pesquisa em árvore**

Exercício: Problema dos Baldes

Dois baldes, de capacidades c1 (ex: 4 litros) e c2 (ex: 3 litros), respetivamente, estão inicialmente vazios. Os baldes não possuem qualquer marcação intermédia. As únicas operações que pode realizar são:

- esvaziar um balde
- encher (completamente) um balde
- despejar um balde para o outro até que o segundo fique cheio
- despejar um balde para o outro até que o primeiro fique vazio

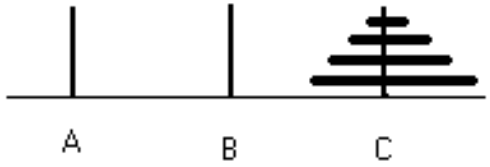
O objetivo consiste em determinar quais as operações a efetuar de modo a que o primeiro balde contenha n litros (exemplo: 2 litros)?

Formule o Problema como um problema de pesquisa

Exercício: Torres de Hanoi

a) Formule o problema das Torres de Hanoi como um problema de pesquisa.

Nesta versão do problema você tem 3 torres (A, B e C) e 4 discos (D1 a D4).



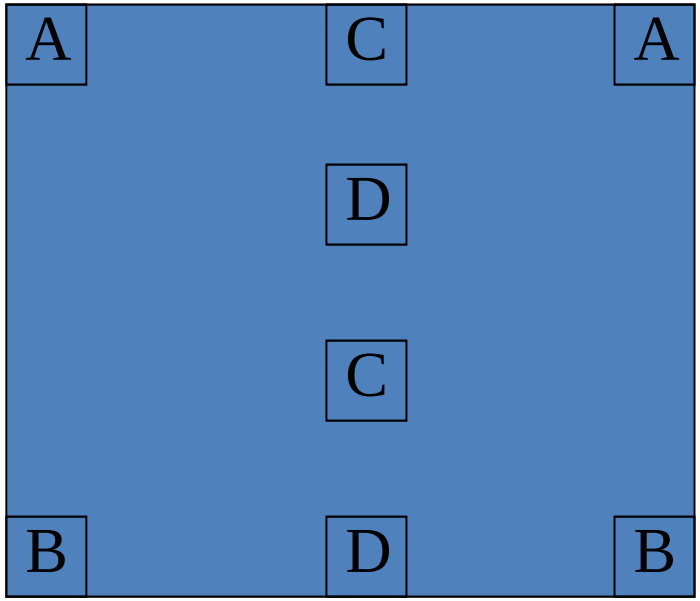
Inicialmente os discos encontram-se na torre C e o objetivo é transferi-los para a torre A.

Em cada jogada, o jogador pode deslocar um disco de uma torre para outra torre, desde que não coloque esse disco sobre um disco menor.

b) Suponha que o número de discos e o número de torres pode ser diferente (n discos e m torres) e formule esta versão genérica do problema como um problema de pesquisa.

Exercício: Quadrado Impossível

- **Problema do Quadrado Impossível**
  - Dado o quadrado apresentado, ligar o A com o A, o B com o B, o C com o C e o D com o D sem cruzar nenhuma linha!
- **Formular o problema do quadrado impossível como um problema de pesquisa e resolve-lo!**



U.PORTO

FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Artificial Intelligence/ Inteligência Artificial

Lecture 3: Search Problems

Luís Paulo Reis

lpreis@fe.up.pt

Director of LIACC – Artificial Intelligence and Computer Science Lab.
Associate Professor at DEI/FEUP – Informatics Engineering Department,
Faculty of Engineering of the University of Porto, Portugal
President of APPIA – Portuguese Association for Artificial Intelligence