

Universidade Federal de Ouro Preto
BCC 325 - Inteligência Artificial
Problemas de Satisfação de Restrições

Prof. Rodrigo Silva

1 Leitura

- Capítulo 3 do Livro *Artificial Intelligence: Foundations of Computational Agents, 2nd Edition* disponível em <https://artint.info/>
- Capítulo 4 do Livro *Artificial Intelligence: Foundations of Computational Agents, 2nd Edition* disponível em <https://artint.info/>
- Aquele que tudo sabe, tudo vê e nada teme.

2 Questões

2.1 Busca em espaços de estados

1. Preencha a seguinte tabela

Estratégia	Seleção da fronteira	Caminho Encontrado	Custo em Espaço
Busca em Largura			
Busca em Profundidade		(g)	
Guloso			
Menor Caminho Primeiro	(b)		
A^*			
Branch and Bound			

- (a) Menor $h(n)$
 - (b) Menor $c(S, n)$
 - (c) Menor $h(n) + c(S, n)$
 - (d) Primeiro caminho adicionado
 - (e) Último caminho adicionado
 - (f) Menor número de arcos
 - (g) Indefinido
 - (h) Menor custo
 - (i) Linear
 - (j) Exponencial
2. Considere o código a seguir e responda.
 - (a) Que tipo de agente este código implementa?
 - (b) Qual parte do código corresponde a cada componente de um agente como definido no capítulo 2?

```

1 class Agent:
2     def __init__(self, ambiente) -> None:
3         self.ambiente = ambiente
4         self.percepcoes = ambiente.percepcoes_iniciais()
5         self.F = [[self.percepcoes['posicao']]]
6         self.C = []
7         self.H = []
8         self.C_H = []
9
10    def act(self):
11
12        while self.F:
13            path = self.F.pop(0)
14
15            action = {'mover_para': path[-1]}
16
17            self.percepcoes = self.ambiente.muda_estado(action)
18
19            if (self.percepcoes['posicao'] == self.percepcoes['saida']).all():
20                return path
21            else:
22                for vi in self.percepcoes['vizinhos']:
23                    self.F.append(path + [vi])

```

Figure 1: agent.py

- (c) Este agente fica preso em ciclos?
- (d) Qual alteração devemos fazer para que o agente implemente uma poda de ciclos? Apresente o código da alteração.

3. O que é uma heurística admissível?

2.2 Problemas de satisfação de restrições

4. (Seção 4.1.3) Quais são os componentes de um problema de satisfação de restrições (Constraint Satisfaction Problem - CSP)? Apresente definições de cada componente.
5. Apresente aplicações reais que podem ser modeladas como problemas de satisfação de restrições?
6. Considere o problema a seguir:

$$\begin{aligned}
 X &= \{A, B, C\} \\
 D &= \{\{1, 2, 3, 4\}, \{1, 2, 3, 4\}, \{1, 2, 3, 4\}\} \\
 C &= \{A \geq B, B < C\}
 \end{aligned}$$

- (a) Qual o tamanho do espaço de busca? Ou seja, no pior caso, quantas soluções candidatas podem ser geradas?
 - (b) Apresente o pseudo-código de uma abordagem baseada em backtracking para resolver este problema.
 - (c) Apresente a árvore de chamadas do algoritmo backtracking para este problema.
 - (d) Represente este problema como uma rede de restrições?
7. (Seção 4.4) Considere o Generalized Arc Consistency (GAC) Algorithm apresentado abaixo e responda.

```

1 GAC(<X,D,C>, {<X,c> | c in C and X in scope(c)}):
2
3 def GAC(<X,D,C>, to_do):
4     while to_do:
5         select and remove <X, c> from to_do
6         let {Y1,...,Yk} = scope(c) \ {X}
7         new_domain = {x|x in D(X) and exists y1 in D(Y1),...,yk in D(Yk)
8                        such that c(X=x,Y1=y1,...Yk=yk)==True}
9         if new_domain not equal D(X):
10             to_do = to_do union {<Z,_c>| {X,Z} in scope(_c), _c!=c, Z!=X}
11             D(X) = new_domain
12 return D

```

Figure 2: Algoritmo GAC

- (a) Demonstre a execução do algoritmo de consistência de arcos, GAC (Generalized Arc Consistency Algorithm) para o problema da questão anterior.
- (b) Quais conclusões pode ser tiradas após a execução do GAC, no geral? O que podemos concluir após a execução do GAC para este problema?
- (c) Após a execução do GAC, qual o tamanho do espaço de busca?