BCC325 – Prova 2
Problemas de Satisfação de Restrições

Nome:

1. (1,5 pt) Considere um CSP com n variáveis, cada uma com domínio de tamanho d.

   a) Quantas atribuições totais existem neste caso?

   b) Qual é a complexidade assintótica do algoritmo *generate-and-test* em termos de número de variáveis n, tamanho do domínio d, e número de restrições e?

   c) Explique por que este algoritmo se torna inviável para valores grandes de n.

2. (1,5 pt) Considere as variáveis $A, B, C \in \{1, 2, 3, 4\}$ e as restrições $A < B$ e $B < C$.

   a) Liste todas as atribuições totais possíveis.

   b) Quantas dessas atribuições satisfazem todas as restrições?

   c) Mostre como a verificação de restrições em atribuições parciais pode evitar a geração de soluções inválidas.

3. (1,5 pt) Responda as questões a seguir com base no algoritmo DFS solve apresentado no Apêndice ao final desta prova.

   a) Descreva o funcionamento do algoritmo `DFS_solver` apresentado no livro

   b) Em que momento ocorre a verificação de restrições parciais?

   c) Justifique por que isso melhora a eficiência do algoritmo em comparação ao *generate-and-test*.

4. (1.5 pt) Classifique cada uma das seguintes restrições quanto à sua aridade (unária, binária ou de ordem superior) e quanto à forma de representação (intencional ou extensional):

   a) $A \neq B$

   b) Sala(Aula1) = Sala2

   c) Tabela que lista todas as combinações válidas de três variáveis

   d) $E < A \wedge E < B \wedge E < C \wedge E < D$

5. (2 pts) Considere a sua implementação do backtracking para o problema de alocação de salas?

   a) Quais são as variáveis, domínios e restrições envolvidas?

   b) O que o código faz para garantir que as restrições sejam satisfeitas?

   c) Apresente a primeira solução completa que o seu algoritmo irá gerar?

6. (2 pts) Resolva as questões a seguir nos labitintos abaixo:

   (a) Calcule o valor da heuística distância de Manhattan para cada casasa livre.

   (b) Numere os nós expandidos pelo algoritmo A*, assumindo custo unitário e heurística distância de Manhattan. Em caso de empate a ordem deve ser, cima, baixo, esquerda ou direita.

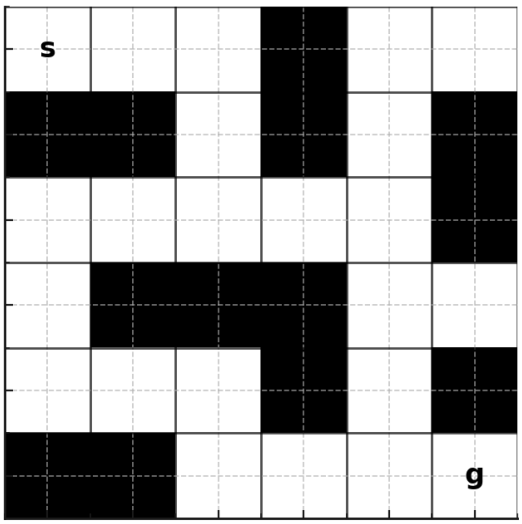

Figura 1: Labirinto para a heurística

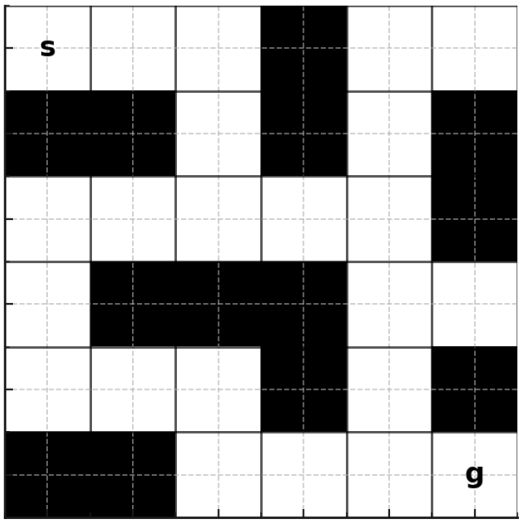

Figura 2: Labirinto para o A*

Apêndice – Algoritmo `DFS_solver`

Algorithm 1 DFS_SOLVER($V_s, C_s,$ context)

$c_e \leftarrow \{c \in C_s \mid c$ pode ser avaliado no context$\}$
if context viola alguma restrição em c_e **then**
    return $\{\}$
else if $V_s = \{\}$ **then**
    return {context}
else
    selecione variável $var \in V_s$
    $sols \leftarrow \{\}$
    for all $val \in$ domínio(var) **do**
        $sols \leftarrow sols \cup$ DFS_SOLVER($V_s \setminus \{var\}, C_s \setminus c_e, \{var = val\} \cup$ context)
    end for
    return $sols$
end if