Inteligência Artificial – ACH2016

Aula 06 – Problemas de Satisfação de Restrições I

Norton Trevisan Roman
(norton@usp.br)

12 de março de 2019

CSP – Constraint Satisfaction Problems

Definidos por:

CSP – Constraint Satisfaction Problems

Definidos por:

- Conjunto de variáveis $\{X_i\}$

CSP – Constraint Satisfaction Problems

Definidos por:

- Conjunto de variáveis $\{X_i\}$
  - Cada variável tem um domínio de valores possíveis

CSP – Constraint Satisfaction Problems

Definidos por:

- Conjunto de variáveis $\{X_i\}$
  - Cada variável tem um domínio de valores possíveis
  - Cada domínio D_i consiste de um conjunto de valores permitidos $\{v_1, \ldots, v_k\}$ para a variável X_i

CSP – Constraint Satisfaction Problems

Definidos por:

- Conjunto de variáveis $\{X_i\}$
  - Cada variável tem um domínio de valores possíveis
  - Cada domínio D_i consiste de um conjunto de valores permitidos $\{v_1, \ldots, v_k\}$ para a variável X_i
- Conjunto de restrições

CSP – Constraint Satisfaction Problems

Definidos por:

- Conjunto de variáveis $\{X_i\}$
  - Cada variável tem um domínio de valores possíveis
  - Cada domínio D_i consiste de um conjunto de valores permitidos $\{v_1, \ldots, v_k\}$ para a variável X_i
- Conjunto de restrições
  - Envolvem um subconjunto de variáveis

CSP – Constraint Satisfaction Problems

Definidos por:

- Conjunto de variáveis $\{X_i\}$
  - Cada variável tem um domínio de valores possíveis
  - Cada domínio D_i consiste de um conjunto de valores permitidos $\{v_1, \ldots, v_k\}$ para a variável X_i
- Conjunto de restrições
  - Envolvem um subconjunto de variáveis
  - Especificam as combinações permitidas de valores para esse subconjunto

CSP – Constraint Satisfaction Problems

Estados do problema

CSP – Constraint Satisfaction Problems

Estados do problema

- Definidos por uma associação de valores a algumas ou todas as variáveis

CSP – Constraint Satisfaction Problems

Estados do problema

- Definidos por uma associação de valores a algumas ou todas as variáveis
  - Uma associação que não viola nenhuma restrição é chamada de **associação consistente** ou **legal**

CSP – Constraint Satisfaction Problems

Estados do problema

- Definidos por uma associação de valores a algumas ou todas as variáveis
  - Uma associação que não viola nenhuma restrição é chamada de **associação consistente** ou **legal**
  - Uma associação em que toda variável é associada a um valor é chamada de **associação completa**

CSP – Constraint Satisfaction Problems

Estados do problema

- Definidos por uma associação de valores a algumas ou todas as variáveis
  - Uma associação que não viola nenhuma restrição é chamada de **associação consistente** ou **legal**
  - Uma associação em que toda variável é associada a um valor é chamada de **associação completa**

Solução

CSP – Constraint Satisfaction Problems

Estados do problema

- Definidos por uma associação de valores a algumas ou todas as variáveis
  - Uma associação que não viola nenhuma restrição é chamada de **associação consistente** ou **legal**
  - Uma associação em que toda variável é associada a um valor é chamada de **associação completa**

Solução

- Trata-se de uma associação completa e consistente

CSP – Constraint Satisfaction Problems

Estados do problema

- Definidos por uma associação de valores a algumas ou todas as variáveis
  - Uma associação que não viola nenhuma restrição é chamada de **associação consistente** ou **legal**
  - Uma associação em que toda variável é associada a um valor é chamada de **associação completa**

Solução

- Trata-se de uma associação completa e consistente
  - O problema estará resolvido quando cada variável tiver um valor que satisfaz todas as restrições sobre ela

CSP – Exemplo

Agendamento de aulas

- Problema:

CSP – Exemplo

Agendamento de aulas

- Problema:
  - Há um número fixo de professores e salas, uma lista de aulas a serem oferecidas, e uma lista de horários possíveis para elas

CSP – Exemplo

Agendamento de aulas

- Problema:
  - Há um número fixo de professores e salas, uma lista de aulas a serem oferecidas, e uma lista de horários possíveis para elas
  - Cada professor tem um conjunto de aulas que pode dar

CSP – Exemplo

Agendamento de aulas

- Problema:
  - Há um número fixo de professores e salas, uma lista de aulas a serem oferecidas, e uma lista de horários possíveis para elas
  - Cada professor tem um conjunto de aulas que pode dar
- Variáveis:

CSP – Exemplo

Agendamento de aulas

- Problema:
  - Há um número fixo de professores e salas, uma lista de aulas a serem oferecidas, e uma lista de horários possíveis para elas
  - Cada professor tem um conjunto de aulas que pode dar
- Variáveis:
  - Uma para cada aula (queremos que aulas sejam dadas)

CSP – Exemplo

Agendamento de aulas

- Problema:
  - Há um número fixo de professores e salas, uma lista de aulas a serem oferecidas, e uma lista de horários possíveis para elas
  - Cada professor tem um conjunto de aulas que pode dar
- Variáveis:
  - Uma para cada aula (queremos que aulas sejam dadas)
- Valores:

CSP – Exemplo

Agendamento de aulas

- Problema:
  - Há um número fixo de professores e salas, uma lista de aulas a serem oferecidas, e uma lista de horários possíveis para elas
  - Cada professor tem um conjunto de aulas que pode dar
- Variáveis:
  - Uma para cada aula (queremos que aulas sejam dadas)
- Valores:
  - A tripla <sala, horário, professor>

CSP – Exemplo

Agendamento de aulas

- Domínio:

CSP – Exemplo

Agendamento de aulas

- Domínio:
  - Para cada variável C_i, seu domínio correspondente D_i conterá todas as possíveis triplas, após eliminarmos aquelas em que o professor não ensina C_i

CSP – Exemplo

Agendamento de aulas

- Domínio:
  - Para cada variável C_i, seu domínio correspondente D_i conterá todas as possíveis triplas, após eliminarmos aquelas em que o professor não ensina C_i
- Restrições:

CSP – Exemplo

Agendamento de aulas

- Domínio:
  - Para cada variável C_i, seu domínio correspondente D_i conterá todas as possíveis triplas, após eliminarmos aquelas em que o professor não ensina C_i
- Restrições:
  - Uma para cada par de variáveis $\{C_i, C_j\}$, dizendo:

CSP – Exemplo

Agendamento de aulas

- Domínio:
  - Para cada variável C_i, seu domínio correspondente D_i conterá todas as possíveis triplas, após eliminarmos aquelas em que o professor não ensina C_i
- Restrições:
  - Uma para cada par de variáveis $\{C_i, C_j\}$, dizendo:
    - Horário + sala de uma não podem bater com o da outra

CSP – Exemplo

Agendamento de aulas

- Domínio:
  - Para cada variável C_i, seu domínio correspondente D_i conterá todas as possíveis triplas, após eliminarmos aquelas em que o professor não ensina C_i
- Restrições:
  - Uma para cada par de variáveis $\{C_i, C_j\}$, dizendo:
    - Horário + sala de uma não podem bater com o da outra
    - Horário + professor de uma não pode bater com o da outra (a ligação entre professor e sala é feita via horário)

CSP – Exemplo

Coloração de mapas

- Tarefa:

CSP – Exemplo

Coloração de mapas

- Tarefa:
  - Colorir cada região de um mapa, de modo a que regiões vizinhas não tenham a mesma cor

Fonte: Slides de AIMA. Russell & Norvig.

CSP – Exemplo

Coloração de mapas

- Tarefa:
  - Colorir cada região de um mapa, de modo a que regiões vizinhas não tenham a mesma cor
- Variáveis:

Fonte: Slides de AIMA. Russell & Norvig.

CSP – Exemplo

Coloração de mapas

- Tarefa:
  - Colorir cada região de um mapa, de modo a que regiões vizinhas não tenham a mesma cor
- Variáveis:
  - As regiões

Fonte: Slides de AIMA. Russell & Norvig.

CSP – Exemplo

Coloração de mapas

- Tarefa:
  - Colorir cada região de um mapa, de modo a que regiões vizinhas não tenham a mesma cor
- Variáveis:
  - As regiões
- Domínio:

Fonte: Slides de AIMA. Russell & Norvig.

CSP – Exemplo

Coloração de mapas

- Tarefa:
  - Colorir cada região de um mapa, de modo a que regiões vizinhas não tenham a mesma cor
- Variáveis:
  - As regiões
- Domínio:
  - Cores: Vermelho, Verde e Azul

Fonte: Slides de AIMA. Russell & Norvig.

CSP – Exemplo

Coloração de mapas

- Tarefa:
  - Colorir cada região de um mapa, de modo a que regiões vizinhas não tenham a mesma cor
- Variáveis:
  - As regiões
- Domínio:
  - Cores: Vermelho, Verde e Azul
- Restrições:

Fonte: Slides de AIMA. Russell & Norvig.

CSP – Exemplo

Coloração de mapas

- Tarefa:
  - Colorir cada região de um mapa, de modo a que regiões vizinhas não tenham a mesma cor
- Variáveis:
  - As regiões
- Domínio:
  - Cores: Vermelho, Verde e Azul
- Restrições:
  - Regiões vizinhas devem ter cores diferentes

Fonte: Slides de AIMA. Russell & Norvig.

CSP – Exemplo

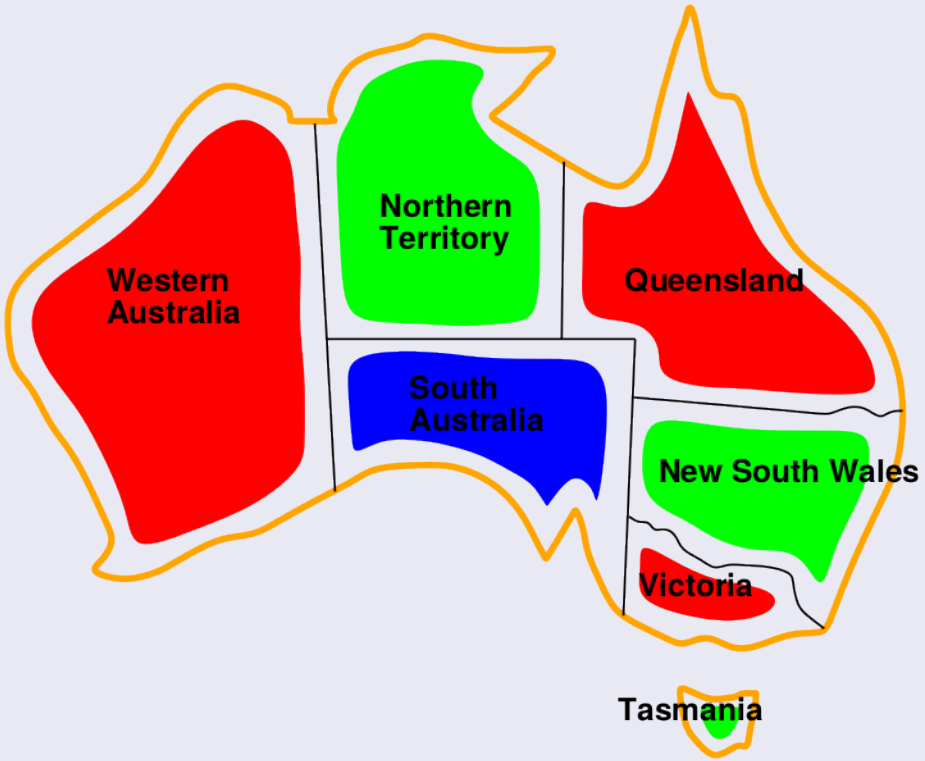
Coloração de mapas

Fonte: Slides de AIMA. Russell & Norvig.

CSP – Exemplo

Coloração de mapas

- Possível solução:
  - WA – vermelho
  - NT – verde
  - SA – azul
  - Q – vermelho
  - NSA – verde
  - V – vermelho
  - T – verde



Fonte: Slides de AIMA. Russell & Norvig.

CSP – Constraint Satisfaction Problems

Tipos de Restrição

CSP – Constraint Satisfaction Problems

Tipos de Restrição

- Unárias:

CSP – Constraint Satisfaction Problems

Tipos de Restrição

- Unárias:
  - Restringem os valores de uma única variável

CSP – Constraint Satisfaction Problems

Tipos de Restrição

- Unárias:
  - Restringem os valores de uma única variável
  - Ex: SA $\neq$ verde

CSP – Constraint Satisfaction Problems

Tipos de Restrição

- Unárias:
  - Restringem os valores de uma única variável
  - Ex: SA $\neq$ verde
  - Podem ser eliminadas pela redução do domínio da variável

CSP – Constraint Satisfaction Problems

Tipos de Restrição

- Unárias:
  - Restringem os valores de uma única variável
  - Ex: SA $\neq$ verde
  - Podem ser eliminadas pela redução do domínio da variável
- Binárias:

CSP – Constraint Satisfaction Problems

Tipos de Restrição

- Unárias:
  - Restringem os valores de uma única variável
  - Ex: SA $\neq$ verde
  - Podem ser eliminadas pela redução do domínio da variável
- Binárias:
  - Relacionam duas variáveis

CSP – Constraint Satisfaction Problems

Tipos de Restrição

- Unárias:
  - Restringem os valores de uma única variável
  - Ex: SA $\neq$ verde
  - Podem ser eliminadas pela redução do domínio da variável
- Binárias:
  - Relacionam duas variáveis
  - Ex: SA $\neq$ WA

CSP – Constraint Satisfaction Problems

Tipos de Restrição

- Unárias:
  - Restringem os valores de uma única variável
  - Ex: SA $\neq$ verde
  - Podem ser eliminadas pela redução do domínio da variável
- Binárias:
  - Relacionam duas variáveis
  - Ex: SA $\neq$ WA
  - Restrições são combinações de valores para cada par de variáveis

CSP – Constraint Satisfaction Problems

Tipos de Restrição

- Ordens maiores:

Tipos de Restrição

- Ordens maiores:
  - Envolvem 3 ou mais variáveis (ex: $Z = X + Y$)

Tipos de Restrição

- Ordens maiores:
  - Envolvem 3 ou mais variáveis (ex: $Z = X + Y$)
  - Podem ser reduzidas a binárias (se seus domínios forem finitos) com o uso de variáveis auxiliares

Tipos de Restrição

- Ordens maiores:
  - Envolvem 3 ou mais variáveis (ex: $Z = X + Y$)
  - Podem ser reduzidas a binárias (se seus domínios forem finitos) com o uso de variáveis auxiliares
  - Ex: $Z = X + Y$, com $X, Y, Z \in \mathbb{N}$

Tipos de Restrição

- Ordens maiores:
  - Envolvem 3 ou mais variáveis (ex: $Z = X + Y$)
  - Podem ser reduzidas a binárias (se seus domínios forem finitos) com o uso de variáveis auxiliares
  - Ex: $Z = X + Y$, com $X, Y, Z \in \mathbb{N}$
    - Seja $XY \in \mathbb{N} \times \mathbb{N}$ (variável auxiliar)

Tipos de Restrição

- Ordens maiores:
  - Envolvem 3 ou mais variáveis (ex: $Z = X + Y$)
  - Podem ser reduzidas a binárias (se seus domínios forem finitos) com o uso de variáveis auxiliares
  - Ex: $Z = X + Y$, com $X, Y, Z \in \mathbb{N}$
    - Seja $XY \in \mathbb{N} \times \mathbb{N}$ (variável auxiliar)
    - Restrição 1: $(X \leftrightarrow XY)$ O valor de X deve ser o 1º elemento de XY

Tipos de Restrição

- Ordens maiores:
  - Envolvem 3 ou mais variáveis (ex: $Z = X + Y$)
  - Podem ser reduzidas a binárias (se seus domínios forem finitos) com o uso de variáveis auxiliares
  - Ex: $Z = X + Y$, com $X, Y, Z \in \mathbb{N}$
    - Seja $XY \in \mathbb{N} \times \mathbb{N}$ (variável auxiliar)
    - Restrição 1: $(X \leftrightarrow XY)$ O valor de X deve ser o 1º elemento de XY
    - Restrição 2: $(Y \leftrightarrow XY)$ O valor de Y deve ser o 2º elemento de XY

Tipos de Restrição

- Ordens maiores:
  - Envolvem 3 ou mais variáveis (ex: $Z = X + Y$)
  - Podem ser reduzidas a binárias (se seus domínios forem finitos) com o uso de variáveis auxiliares
  - Ex: $Z = X + Y$, com $X, Y, Z \in \mathbb{N}$
    - Seja $XY \in \mathbb{N} \times \mathbb{N}$ (variável auxiliar)
    - Restrição 1: $(X \leftrightarrow XY)$ O valor de X deve ser o 1º elemento de XY
    - Restrição 2: $(Y \leftrightarrow XY)$ O valor de Y deve ser o 2º elemento de XY
    - Restrição 3: $(Z \leftrightarrow XY)$ O valor de Z deve ser a soma dos elementos de XY

CSP – Constraint Satisfaction Problems

Tipos de Restrição

- Restrições globais

CSP – Constraint Satisfaction Problems

Tipos de Restrição

- Restrições globais
  - São restrições de ordem maior que a binária envolvendo um número arbitrário de variáveis

CSP – Constraint Satisfaction Problems

Tipos de Restrição

- Restrições globais
  - São restrições de ordem maior que a binária envolvendo um número arbitrário de variáveis
- Preferências:

CSP – Constraint Satisfaction Problems

Tipos de Restrição

- Restrições globais
  - São restrições de ordem maior que a binária envolvendo um número arbitrário de variáveis
- Preferências:
  - São "restrições leves" $\rightarrow$ sua violação não exclui uma solução potencial

CSP – Constraint Satisfaction Problems

Tipos de Restrição

- Restrições globais
  - São restrições de ordem maior que a binária envolvendo um número arbitrário de variáveis
- Preferências:
  - São "restrições leves" $\rightarrow$ sua violação não exclui uma solução potencial
  - Ex: é melhor usar verde em vez de vermelho, quando possível

CSP – Constraint Satisfaction Problems

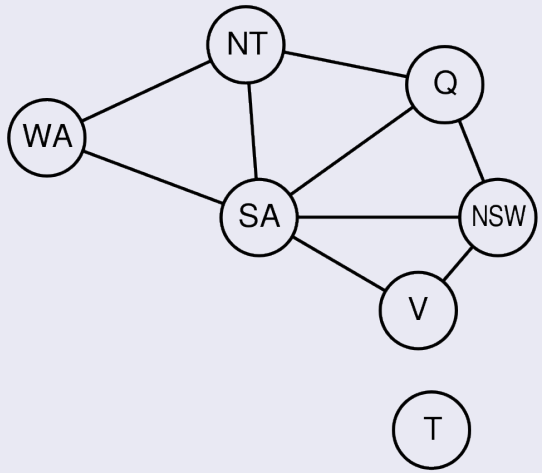
Tipos de Restrição

- Restrições globais
  - São restrições de ordem maior que a binária envolvendo um número arbitrário de variáveis
- Preferências:
  - São "restrições leves" $\rightarrow$ sua violação não exclui uma solução potencial
  - Ex: é melhor usar verde em vez de vermelho, quando possível
  - Representadas por um custo dado a cada associação de variáveis

CSP – Representação

Grafo de restrições

- Grafo em que

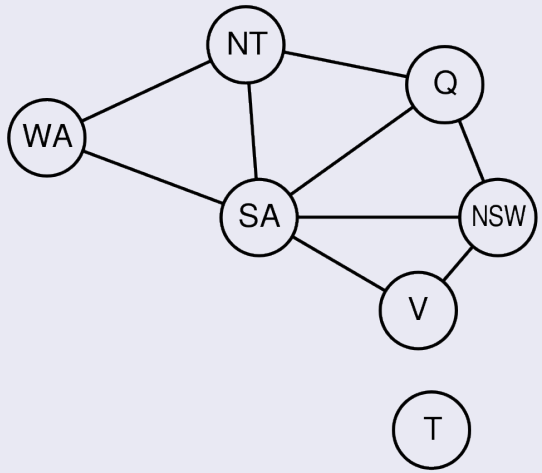


Fonte: Slides de AIMA. Russell & Norvig.

CSP – Representação

Grafo de restrições

- Grafo em que
  - Os nós correspondem às variáveis do problema

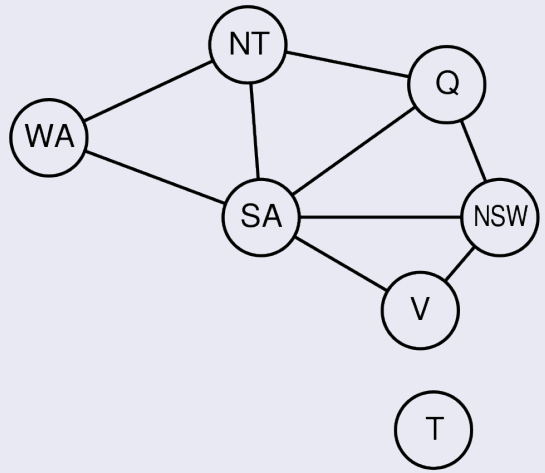


Fonte: Slides de AIMA. Russell & Norvig.

CSP – Representação

Grafo de restrições

- Grafo em que
  - Os nós correspondem às variáveis do problema
  - Uma aresta coneta quaisquer duas variáveis que participam em uma restrição

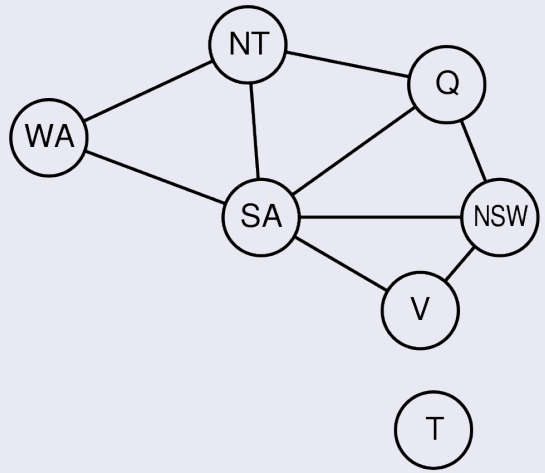


Fonte: Slides de AIMA. Russell & Norvig.

CSP – Representação

Grafo de restrições

- Grafo em que
  - Os nós correspondem às variáveis do problema
  - Uma aresta coneta quaisquer duas variáveis que participam em uma restrição
- Útil para CSPs binários

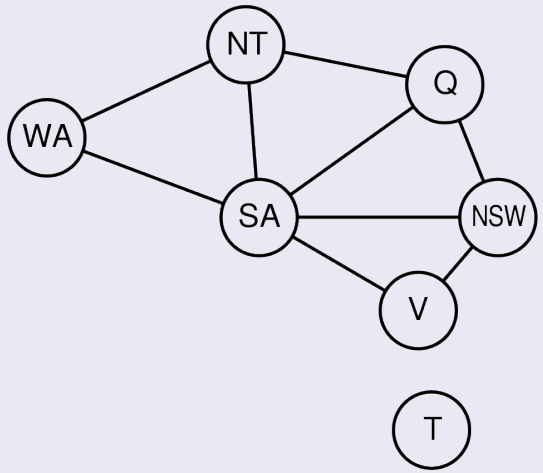


Fonte: Slides de AIMA. Russell & Norvig.

CSP – Representação

Grafo de restrições

- Grafo em que
  - Os nós correspondem às variáveis do problema
  - Uma aresta coneta quaisquer duas variáveis que participam em uma restrição
- Útil para CSPs binários
  - Aqueles com restrições binárias apenas

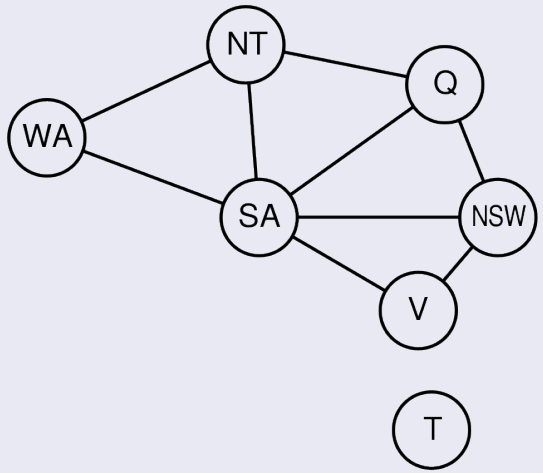


Fonte: Slides de AIMA. Russell & Norvig.

CSP – Representação

Grafo de restrições

- Contudo, toda restrição pode ser transformada em binária

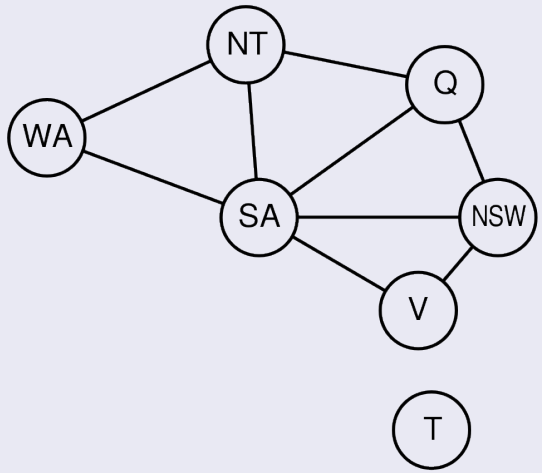


Fonte: Slides de AIMA. Russell & Norvig.

CSP – Representação

Grafo de restrições

- Contudo, toda restrição pode ser transformada em binária
  - Vimos redução de 3 para 2 variáveis

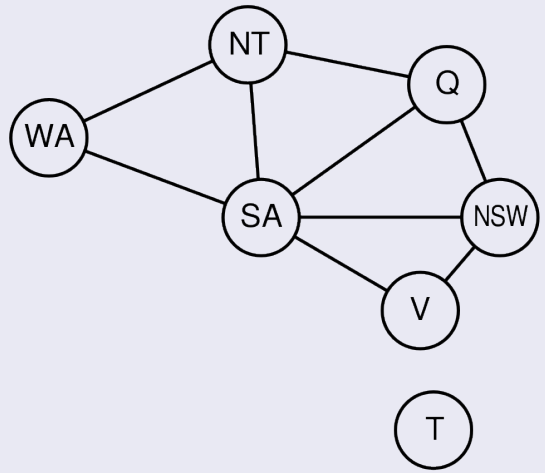


Fonte: Slides de AIMA. Russell & Norvig.

CSP – Representação

Grafo de restrições

- Contudo, toda restrição pode ser transformada em binária
  - Vimos redução de 3 para 2 variáveis
  - 4 serão reduzidas para 3, etc

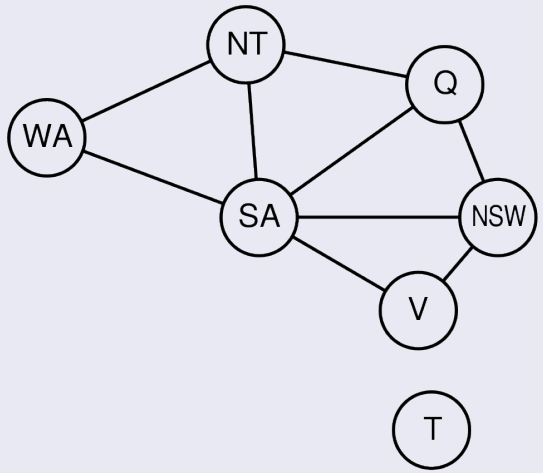


Fonte: Slides de AIMA. Russell & Norvig.

CSP – Representação

Grafo de restrições

- Contudo, toda restrição pode ser transformada em binária
  - Vimos redução de 3 para 2 variáveis
  - 4 serão reduzidas para 3, etc
  - Restrições unárias podem ser eliminadas alterando-se o domínio da variável



Fonte: Slides de AIMA. Russell & Norvig.

CSP – Constraint Satisfaction Problems

Característica de CSPs

CSP – Constraint Satisfaction Problems

Característica de CSPs

- Comutatividade:

CSP – Constraint Satisfaction Problems

Característica de CSPs

- Comutatividade:
  - A ordem de associação de valores a variáveis não influencia a resposta

CSP – Constraint Satisfaction Problems

Característica de CSPs

- Comutatividade:
  - A ordem de associação de valores a variáveis não influencia a resposta
  - $\{WA = vm, NT = vd\} = \{NT = vd, WA = vm\}$

CSP – Constraint Satisfaction Problems

Característica de CSPs

- Comutatividade:
  - A ordem de associação de valores a variáveis não influencia a resposta
  - $\{WA = vm, NT = vd\} = \{NT = vd, WA = vm\}$
- Só precisamos considerar associações a uma única variável, em cada nó

CSP – Constraint Satisfaction Problems

Característica de CSPs

- Comutatividade:
  - A ordem de associação de valores a variáveis não influencia a resposta
  - $\{WA = vm, NT = vd\} = \{NT = vd, WA = vm\}$
- Só precisamos considerar associações a uma única variável, em cada nó
- Removemos redundâncias, caso testássemos todas as combinações

CSP – Constraint Satisfaction Problems

Busca Retroativa (Backtracking Search)

- Estados: Definidos pelos valores associados até então $\rightarrow$ uma associação parcial

CSP – Constraint Satisfaction Problems

Busca Retroativa (Backtracking Search)

- Estados: Definidos pelos valores associados até então $\rightarrow$ uma associação parcial
- Estado inicial: Variáveis sem valor algum: $\{\}$

CSP – Constraint Satisfaction Problems

Busca Retroativa (Backtracking Search)

- Estados: Definidos pelos valores associados até então $\rightarrow$ uma associação parcial
- Estado inicial: Variáveis sem valor algum: $\{\}$
- Ações (geração do sucessor):

CSP – Constraint Satisfaction Problems

Busca Retroativa (Backtracking Search)

- Estados: Definidos pelos valores associados até então $\rightarrow$ uma associação parcial
- Estado inicial: Variáveis sem valor algum: $\{\}$
- Ações (geração do sucessor):
  - Incluir *var* = *valor* na associação, tomando cuidado para não conflitar com os valores já atribuídos

CSP – Constraint Satisfaction Problems

Busca Retroativa (Backtracking Search)

- Estados: Definidos pelos valores associados até então $\rightarrow$ uma associação parcial
- Estado inicial: Variáveis sem valor algum: $\{\}$
- Ações (geração do sucessor):
  - Incluir *var* = *valor* na associação, tomando cuidado para não conflitar com os valores já atribuídos
  - Se não houver tal valor, falhe

CSP – Constraint Satisfaction Problems

Busca Retroativa (Backtracking Search)

- Estados: Definidos pelos valores associados até então $\rightarrow$ uma associação parcial
- Estado inicial: Variáveis sem valor algum: $\{\}$
- Ações (geração do sucessor):
  - Incluir *var* = *valor* na associação, tomando cuidado para não conflitar com os valores já atribuídos
  - Se não houver tal valor, falhe
- Teste: Se todas as variáveis têm um valor associado

CSP – Backtracking Search

Algoritmo (Recursivo)

Função *BUSCA(csp)*: **associação**
    retorna *BACKTRACK({},csp)*

Função *BACKTRACK(associação,csp)*: **associação**
    se *associação for completa* **então**
        retorna *associação*
    var $\leftarrow$ selecione uma variável ainda sem valor
    para cada *valor no domínio de var* **faça**
        se *valor for consistente com a associação* **então**
            Inclua $\{var = valor\}$ em *associação*
            resultado $\leftarrow$ BACKTRACK(*associação,csp*)
            se *resultado* $\neq$ *falha* **então**
                retorna *resultado*
            Remova $\{var = valor\}$ de *associação*
    retorna *falha*

CSP – Backtracking Search

Algoritmo (Iterativo)

Função *BUSCA(csp)*: **associação**

    Empilhe a raiz {} em *prox*;
    enquanto *prox* $\neq$ *vazia* **faça**
        associação $\leftarrow$ desempilhe uma associação de *prox*;
        se *associação for completa* **então**
            retorna *associação*
        var $\leftarrow$ seleciona uma variável sem valor atribuído em *associação*;
        para cada *valor no domínio de var* **faça**
            se *valor for consistente com a associação* **então**
                Empilhe $\{var = valor\} \cup$ *associação* em *prox*;
    retorna *falha*

CSP – Backtracking Search

Algoritmo (Iterativo)

Função *BUSCA(csp)*: **associação**

- Empilhe a raiz {} em *prox*;
- **enquanto** *prox* $\neq$ *vazia* **faça**
  - *associação* $\leftarrow$ desempilhe uma associação de *prox*;
  - **se** *associação for completa* **então**
    - **retorna** *associação*
  - *var* $\leftarrow$ seleciona uma variável sem valor atribuído em *associação*;
  - **para cada** *valor no domínio de var* **faça**
    - **se** *valor for consistente com a associação* **então**
      - Empilhe $\{var = valor\} \cup$ *associação* em *prox*;
- **retorna** *falha*

Pilha de nós contendo os nós que ainda podem ser visitados, na ordem em que serão visitados

CSP – Backtracking Search

Algoritmo (Iterativo)

Função *BUSCA(csp)*: **associação**

- Empilhe a raiz {} em *prox*;
- **enquanto** *prox* $\neq$ *vazia* **faça**
  - *associação* $\leftarrow$ desempilhe uma associação de *prox*;
  - **se** *associação for completa* **então**
    - **retorna** *associação*
  - *var* $\leftarrow$ seleciona uma variável sem valor atribuído em *associação*;
  - **para cada** *valor no domínio de var* **faça**
    - **se** *valor for consistente com a associação* **então**
      - Empilhe $\{var = valor\} \cup$ *associação* em *prox*;
- **retorna** *falha*

Se não for possível gerar um valor consistente para `var`...

CSP – Backtracking Search

Algoritmo (Iterativo)

Função *BUSCA(csp)*: **associação**

    Empilhe a raiz {} em *prox*;
    enquanto *prox* ≠ *vazia* **faça**
        associação ← desempilhe uma associação de *prox*;
        se *associação for completa* **então**
            retorna *associação*
        var ← seleciona uma variável sem valor atribuído em *associação*;
        para cada *valor no domínio de var* **faça**
            se *valor for consistente com a associação* **então**
                Empilhe $\{var = valor\} \cup$ *associação* em *prox*;
    retorna *falha*

Na próxima iteração o algoritmo volta ao nó anterior (que está empilhado)

CSP – Backtracking Search

Algoritmo – Funcionamento

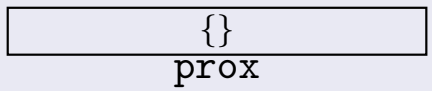
Função *BUSCA(csp)*: **associação**
    Empilhe a raiz {} em *prox*;
    enquanto *prox* $\neq$ *vazia* **faça**
        associação $\leftarrow$ desempilhe uma associação de *prox*;
        se *associação for completa* **então**
            retorna *associação*
        var $\leftarrow$ seleciona uma variável sem valor atribuído em *associação*;
        para cada *valor no domínio de var* **faça**
            se *valor for consistente com a associação* **então**
                Empilhe $\{var = valor\} \cup$ *associação* em *prox*;
    retorna *falha*

CSP – Backtracking Search

Algoritmo – Funcionamento

Função *BUSCA(csp)*: **associação**

- Empilhe a raiz {} em *prox*;
- **enquanto** *prox* $\neq$ *vazia* **faça**
  - *associação* $\leftarrow$ desempilhe uma associação de *prox*;
  - **se** *associação for completa* **então**
    - **retorna** *associação*
  - *var* $\leftarrow$ seleciona uma variável sem valor atribuído em *associação*;
  - **para cada** *valor no domínio de var* **faça**
    - **se** *valor for consistente com a associação* **então**
      - Empilhe $\{var = valor\} \cup$ *associação* em *prox*;
- **retorna** *falha*



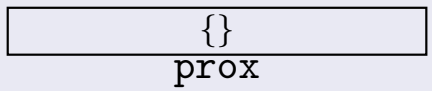
Fonte: Adaptado dos slides de AIMA. Russell & Norvig.

CSP – Backtracking Search

Algoritmo – Funcionamento

Função *BUSCA(csp)*: **associação**

- Empilhe a raiz {} em *prox*;
- **enquanto** *prox* $\neq$ *vazia* **faça**
  - *associação* $\leftarrow$ desempilhe uma associação de *prox*;
  - **se** *associação for completa* **então**
    - **retorna** *associação*
  - *var* $\leftarrow$ seleciona uma variável sem valor atribuído em *associação*;
  - **para cada** *valor no domínio de var* **faça**
    - **se** *valor for consistente com a associação* **então**
      - Empilhe $\{var = valor\} \cup$ *associação* em *prox*;
- **retorna** *falha*



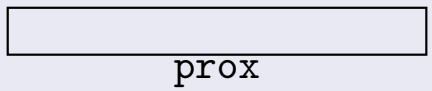
Fonte: Adaptado dos slides de AIMA. Russell & Norvig.

CSP – Backtracking Search

Algoritmo – Funcionamento

Função *BUSCA(csp)*: **associação**

- Empilhe a raiz {} em *prox*;
- **enquanto** *prox* $\neq$ *vazia* **faça**
  - *associação* $\leftarrow$ desempilhe uma associação de *prox*;
  - **se** *associação for completa* **então**
    - **retorna** *associação*
  - *var* $\leftarrow$ seleciona uma variável sem valor atribuído em *associação*;
  - **para cada** *valor no domínio de var* **faça**
    - **se** *valor for consistente com a associação* **então**
      - Empilhe $\{var = valor\} \cup$ *associação* em *prox*;
- **retorna** *falha*



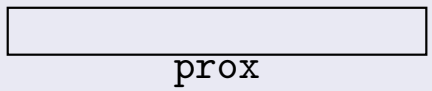
Fonte: Adaptado dos slides de AIMA. Russell & Norvig.

CSP – Backtracking Search

Algoritmo – Funcionamento

Função *BUSCA(csp)*: **associação**

- Empilhe a raiz {} em *prox*;
- **enquanto** *prox* $\neq$ *vazia* **faça**
  - *associação* $\leftarrow$ desempilhe uma associação de *prox*;
  - **se** *associação for completa* **então**
    - **retorna** *associação*
  - *var* $\leftarrow$ seleciona uma variável sem valor atribuído em *associação*;
  - **para cada** *valor no domínio de var* **faça**
    - **se** *valor for consistente com a associação* **então**
      - Empilhe $\{var = valor\} \cup$ *associação* em *prox*;
- **retorna** *falha*



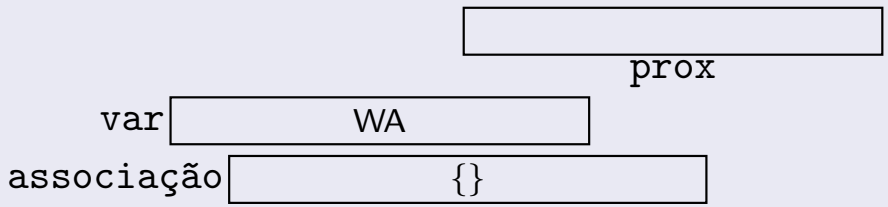
Fonte: Adaptado dos slides de AIMA. Russell & Norvig.

CSP – Backtracking Search

Algoritmo – Funcionamento

Função *BUSCA(csp)*: **associação**

- Empilhe a raiz {} em *prox*;
- **enquanto** *prox* $\neq$ *vazia* **faça**
  - *associação* $\leftarrow$ desempilhe uma associação de *prox*;
  - **se** *associação for completa* **então**
    - **retorna** *associação*
  - *var* $\leftarrow$ seleciona uma variável sem valor atribuído em *associação*;
  - **para cada** *valor no domínio de var* **faça**
    - **se** *valor for consistente com a associação* **então**
      - Empilhe $\{var = valor\} \cup$ *associação* em *prox*;
- **retorna** *falha*



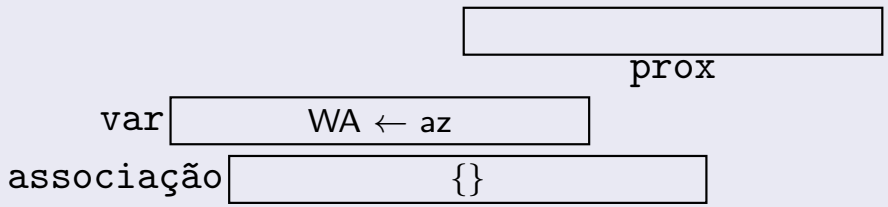
Fonte: Adaptado dos slides de AIMA. Russell & Norvig.

CSP – Backtracking Search

Algoritmo – Funcionamento

Função *BUSCA(csp)*: **associação**

- Empilhe a raiz {} em *prox*;
- **enquanto** *prox* $\neq$ *vazia* **faça**
  - *associação* $\leftarrow$ desempilhe uma associação de *prox*;
  - **se** *associação for completa* **então**
    - **retorna** *associação*
  - *var* $\leftarrow$ seleciona uma variável sem valor atribuído em *associação*;
  - **para cada** *valor no domínio de var* **faça**
    - **se** *valor for consistente com a associação* **então**
      - Empilhe $\{var = valor\} \cup$ *associação* em *prox*;
- **retorna** *falha*



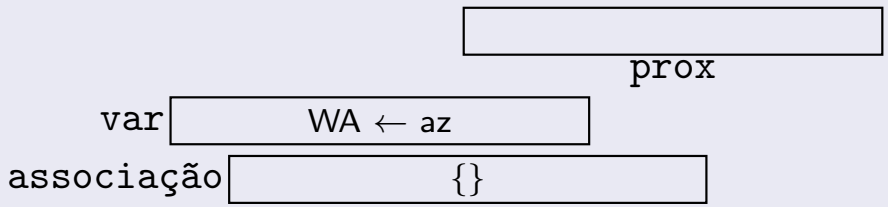
Fonte: Adaptado dos slides de AIMA. Russell & Norvig.

CSP – Backtracking Search

Algoritmo – Funcionamento

Função *BUSCA(csp)*: **associação**

- Empilhe a raiz {} em *prox*;
- **enquanto** *prox* $\neq$ *vazia* **faça**
  - *associação* $\leftarrow$ desempilhe uma associação de *prox*;
  - **se** *associação for completa* **então**
    - **retorna** *associação*
  - *var* $\leftarrow$ seleciona uma variável sem valor atribuído em *associação*;
  - **para cada** *valor no domínio de var* **faça**
    - **se** *valor for consistente com a associação* **então**
      - Empilhe $\{var = valor\} \cup$ *associação* em *prox*;
- **retorna** *falha*

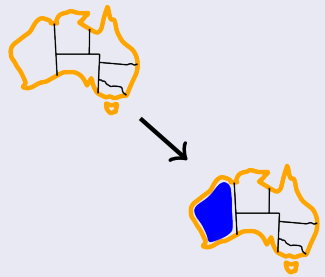


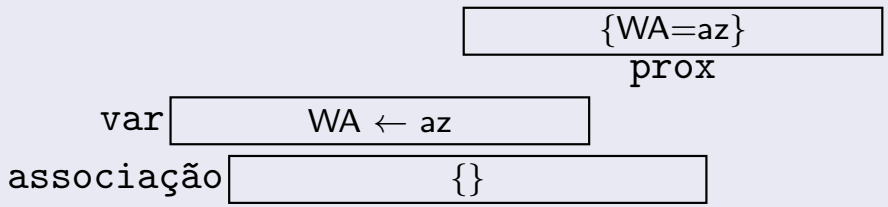
Fonte: Adaptado dos slides de AIMA. Russell & Norvig.

CSP – Backtracking Search

Algoritmo – Funcionamento

Função *BUSCA(csp)*: **associação**
- Empilhe a raiz {} em *prox*;
- **enquanto** *prox* $\neq$ *vazia* **faça**
  - *associação* $\leftarrow$ desempilhe uma associação de *prox*;
  - **se** *associação for completa* **então**
    - **retorna** *associação*
  - *var* $\leftarrow$ seleciona uma variável sem valor atribuído em *associação*;
  - **para cada** *valor no domínio de var* **faça**
    - **se** *valor for consistente com a associação* **então**
      - Empilhe $\{var = valor\} \cup$ *associação* em *prox*;
- **retorna** *falha*





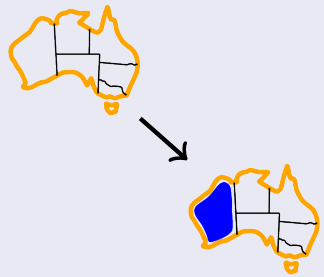
Fonte: Adaptado dos slides de AIMA. Russell & Norvig.

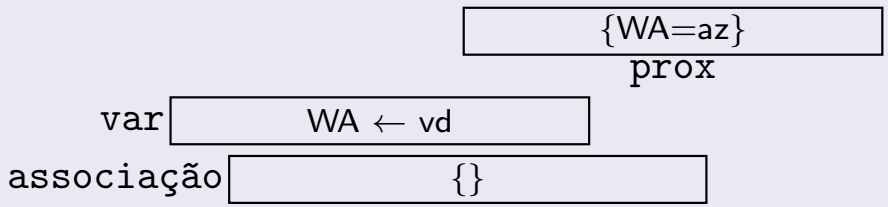
CSP – Backtracking Search

Algoritmo – Funcionamento

Função *BUSCA(csp)*: **associação**

- Empilhe a raiz {} em *prox*;
- **enquanto** *prox* $\neq$ *vazia* **faça**
  - *associação* $\leftarrow$ desempilhe uma associação de *prox*;
  - **se** *associação for completa* **então**
    - **retorna** *associação*
  - *var* $\leftarrow$ seleciona uma variável sem valor atribuído em *associação*;
  - **para cada** *valor no domínio de var* **faça**
    - **se** *valor for consistente com a associação* **então**
      - Empilhe $\{var = valor\} \cup$ *associação* em *prox*;
- **retorna** *falha*



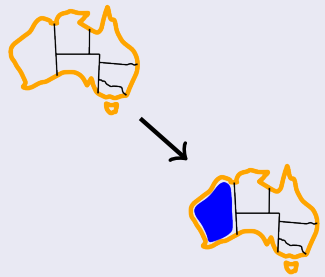


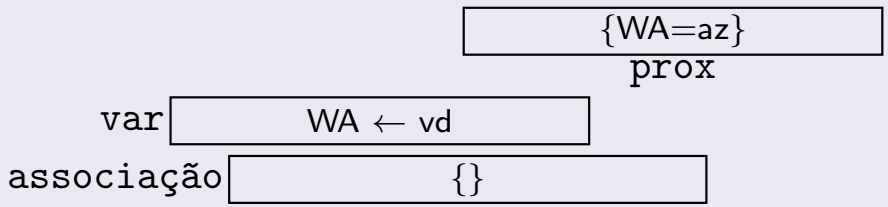
Fonte: Adaptado dos slides de AIMA. Russell & Norvig.

CSP – Backtracking Search

Algoritmo – Funcionamento

Função *BUSCA(csp)*: **associação**
- Empilhe a raiz {} em *prox*;
- **enquanto** *prox* $\neq$ *vazia* **faça**
  - *associação* $\leftarrow$ desempilhe uma associação de *prox*;
  - **se** *associação for completa* **então**
    - **retorna** *associação*
  - *var* $\leftarrow$ seleciona uma variável sem valor atribuído em *associação*;
  - **para cada** *valor no domínio de var* **faça**
    - **se** *valor for consistente com a associação* **então**
      - Empilhe $\{var = valor\} \cup$ *associação* em *prox*;
- **retorna** *falha*



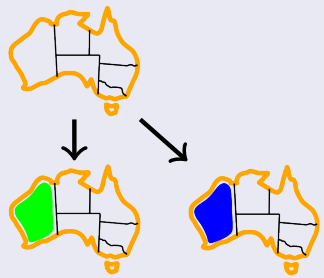


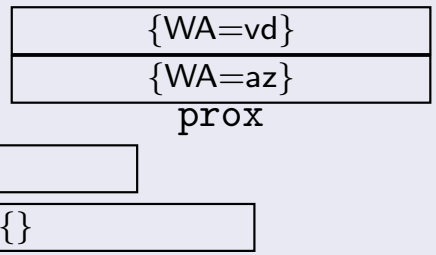
Fonte: Adaptado dos slides de AIMA. Russell & Norvig.

CSP – Backtracking Search

Algoritmo – Funcionamento

Função *BUSCA(csp)*: **associação**
- Empilhe a raiz {} em *prox*;
- **enquanto** *prox* $\neq$ *vazia* **faça**
  - *associação* $\leftarrow$ desempilhe uma associação de *prox*;
  - **se** *associação for completa* **então**
    - **retorna** *associação*
  - *var* $\leftarrow$ seleciona uma variável sem valor atribuído em *associação*;
  - **para cada** *valor no domínio de var* **faça**
    - **se** *valor for consistente com a associação* **então**
      - Empilhe $\{var = valor\} \cup$ *associação* em *prox*;
- **retorna** *falha*



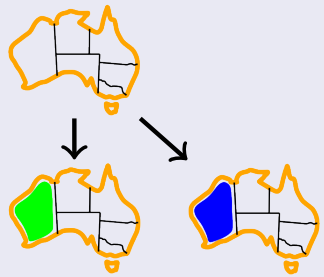


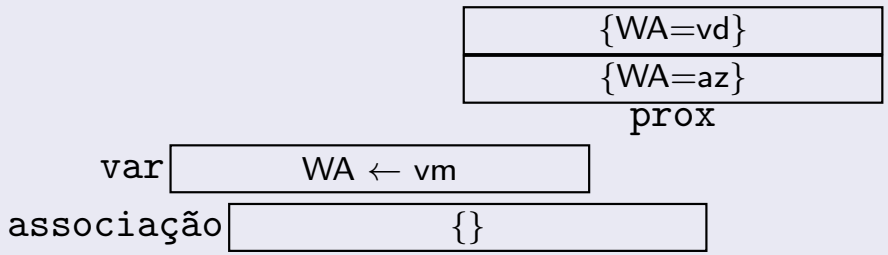
Fonte: Adaptado dos slides de AIMA. Russell & Norvig.

CSP – Backtracking Search

Algoritmo – Funcionamento

Função *BUSCA(csp)*: **associação**
- Empilhe a raiz {} em *prox*;
- **enquanto** *prox* $\neq$ *vazia* **faça**
  - *associação* $\leftarrow$ desempilhe uma associação de *prox*;
  - **se** *associação for completa* **então**
    - **retorna** *associação*
  - *var* $\leftarrow$ seleciona uma variável sem valor atribuído em *associação*;
  - **para cada** *valor no domínio de var* **faça**
    - **se** *valor for consistente com a associação* **então**
      - Empilhe $\{var = valor\} \cup$ *associação* em *prox*;
- **retorna** *falha*





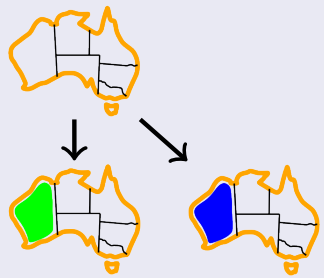
Fonte: Adaptado dos slides de AIMA. Russell & Norvig.

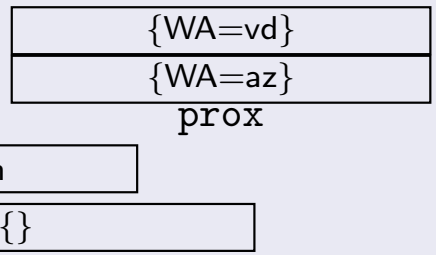
CSP – Backtracking Search

Algoritmo – Funcionamento

Função *BUSCA(csp)*: **associação**

- Empilhe a raiz {} em *prox*;
- **enquanto** *prox* $\neq$ *vazia* **faça**
  - *associação* $\leftarrow$ desempilhe uma associação de *prox*;
  - **se** *associação for completa* **então**
    - **retorna** *associação*
  - *var* $\leftarrow$ seleciona uma variável sem valor atribuído em *associação*;
  - **para cada** *valor no domínio de var* **faça**
    - **se** *valor for consistente com a associação* **então**
      - Empilhe $\{var = valor\} \cup$ *associação* em *prox*;
- **retorna** *falha*



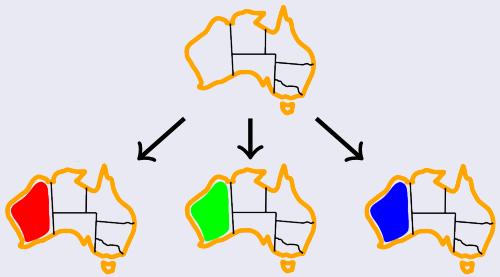


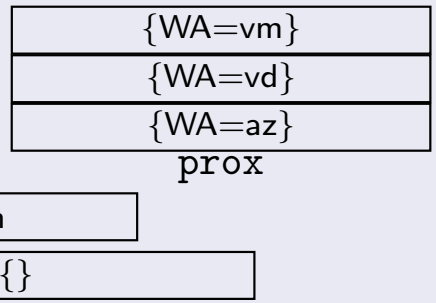
Fonte: Adaptado dos slides de AIMA. Russell & Norvig.

CSP – Backtracking Search

Algoritmo – Funcionamento

Função *BUSCA(csp)*: **associação**
- Empilhe a raiz {} em *prox*;
- **enquanto** *prox* $\neq$ *vazia* **faça**
  - *associação* $\leftarrow$ desempilhe uma associação de *prox*;
  - **se** *associação for completa* **então**
    - **retorna** *associação*
  - *var* $\leftarrow$ seleciona uma variável sem valor atribuído em *associação*;
  - **para cada** *valor no domínio de var* **faça**
    - **se** *valor for consistente com a associação* **então**
      - Empilhe $\{var = valor\} \cup$ *associação* em *prox*;
- **retorna** *falha*





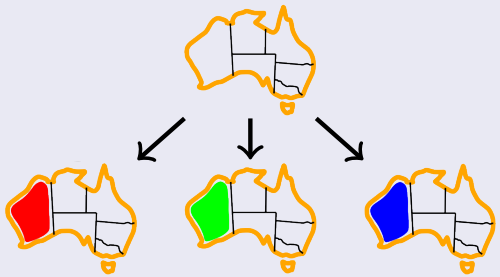
Fonte: Adaptado dos slides de AIMA. Russell & Norvig.

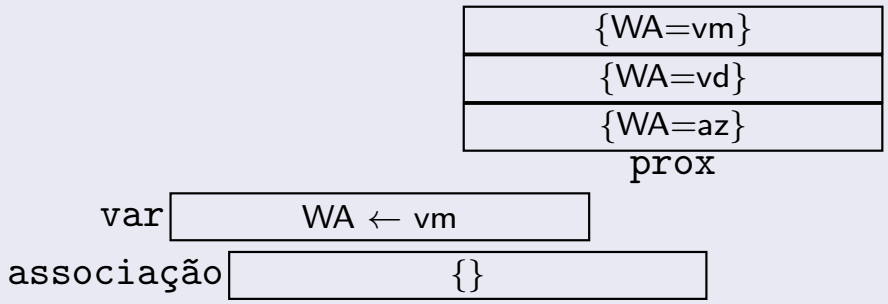
CSP – Backtracking Search

Algoritmo – Funcionamento

Função *BUSCA(csp)*: **associação**

- Empilhe a raiz {} em *prox*;
- **enquanto** *prox* $\neq$ *vazia* **faça**
  - *associação* $\leftarrow$ desempilhe uma associação de *prox*;
  - **se** *associação for completa* **então**
    - **retorna** *associação*
  - *var* $\leftarrow$ seleciona uma variável sem valor atribuído em *associação*;
  - **para cada** *valor no domínio de var* **faça**
    - **se** *valor for consistente com a associação* **então**
      - Empilhe $\{var = valor\} \cup$ *associação* em *prox*;
- **retorna** *falha*





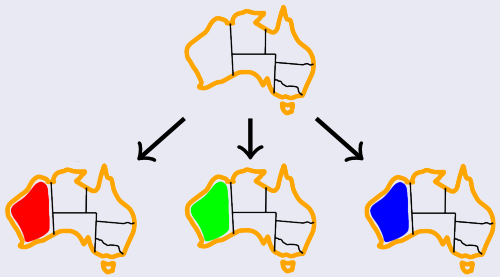
Fonte: Adaptado dos slides de AIMA. Russell & Norvig.

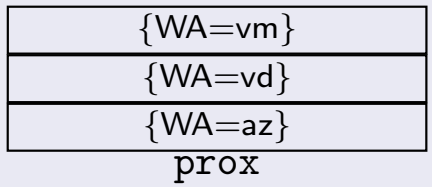
CSP – Backtracking Search

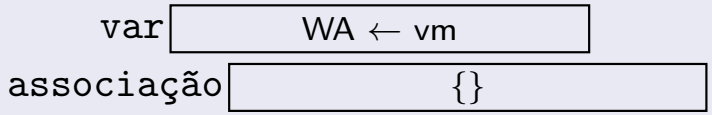
Algoritmo – Funcionamento

Função *BUSCA(csp)*: **associação**

- Empilhe a raiz {} em *prox*;
- **enquanto** *prox* $\neq$ *vazia* **faça**
  - *associação* $\leftarrow$ desempilhe uma associação de *prox*;
  - **se** *associação for completa* **então**
    - **retorna** *associação*
  - *var* $\leftarrow$ seleciona uma variável sem valor atribuído em *associação*;
  - **para cada** *valor no domínio de var* **faça**
    - **se** *valor for consistente com a associação* **então**
      - Empilhe $\{var = valor\} \cup$ *associação* em *prox*;
- **retorna** *falha*







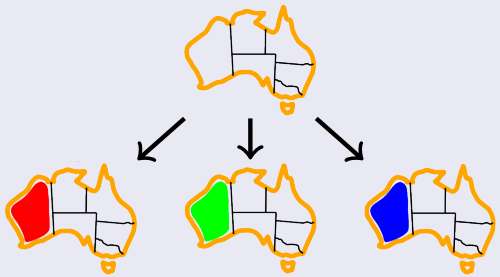
Fonte: Adaptado dos slides de AIMA. Russell & Norvig.

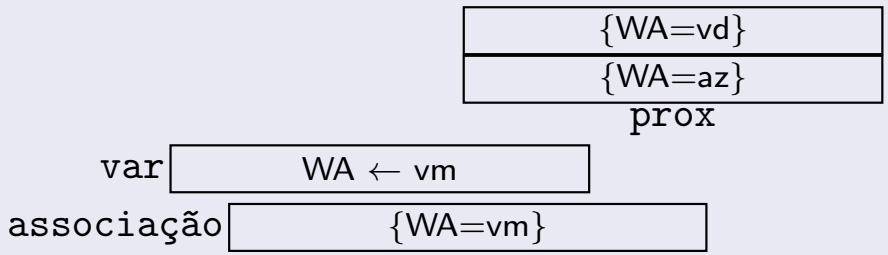
CSP – Backtracking Search

Algoritmo – Funcionamento

Função *BUSCA(csp)*: **associação**

- Empilhe a raiz {} em *prox*;
- **enquanto** *prox* $\neq$ *vazia* **faça**
  - *associação* $\leftarrow$ desempilhe uma associação de *prox*;
  - **se** *associação for completa* **então**
    - **retorna** *associação*
  - *var* $\leftarrow$ seleciona uma variável sem valor atribuído em *associação*;
  - **para cada** *valor no domínio de var* **faça**
    - **se** *valor for consistente com a associação* **então**
      - Empilhe $\{var = valor\} \cup$ *associação* em *prox*;
- **retorna** *falha*





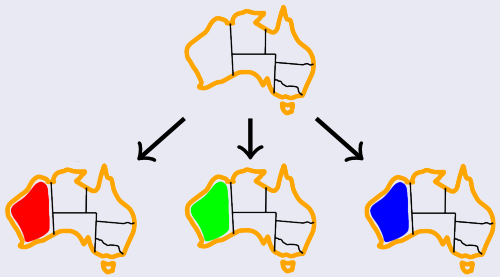
Fonte: Adaptado dos slides de AIMA. Russell & Norvig.

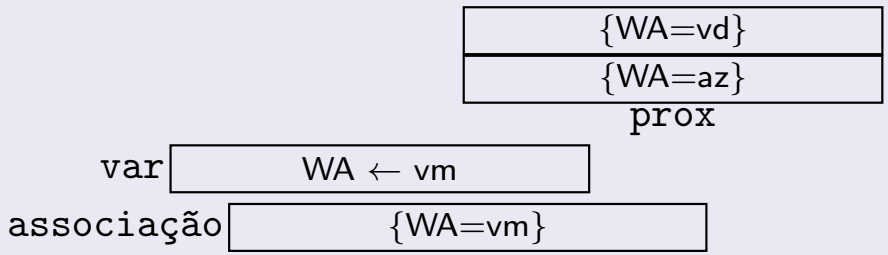
CSP – Backtracking Search

Algoritmo – Funcionamento

Função *BUSCA(csp)*: **associação**

- Empilhe a raiz {} em *prox*;
- **enquanto** *prox* $\neq$ *vazia* **faça**
  - *associação* $\leftarrow$ desempilhe uma associação de *prox*;
  - **se** *associação for completa* **então**
    - **retorna** *associação*
  - *var* $\leftarrow$ seleciona uma variável sem valor atribuído em *associação*;
  - **para cada** *valor no domínio de var* **faça**
    - **se** *valor for consistente com a associação* **então**
      - Empilhe $\{var = valor\} \cup$ *associação* em *prox*;
- **retorna** *falha*





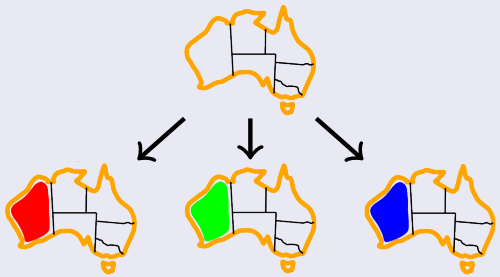
Fonte: Adaptado dos slides de AIMA. Russell & Norvig.

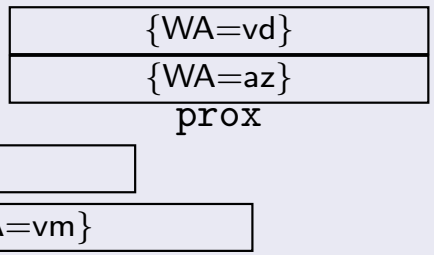
CSP – Backtracking Search

Algoritmo – Funcionamento

Função *BUSCA(csp)*: **associação**

- Empilhe a raiz {} em *prox*;
- **enquanto** *prox* $\neq$ *vazia* **faça**
  - *associação* $\leftarrow$ desempilhe uma associação de *prox*;
  - **se** *associação for completa* **então**
    - **retorna** *associação*
  - *var* $\leftarrow$ seleciona uma variável sem valor atribuído em *associação*;
  - **para cada** *valor no domínio de var* **faça**
    - **se** *valor for consistente com a associação* **então**
      - Empilhe $\{var = valor\} \cup$ *associação* em *prox*;
- **retorna** *falha*





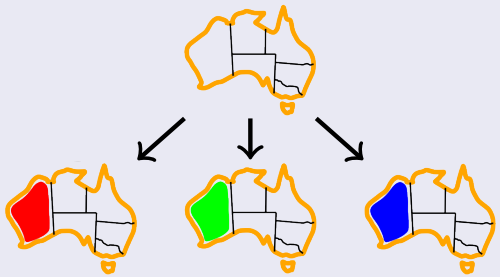
Fonte: Adaptado dos slides de AIMA. Russell & Norvig.

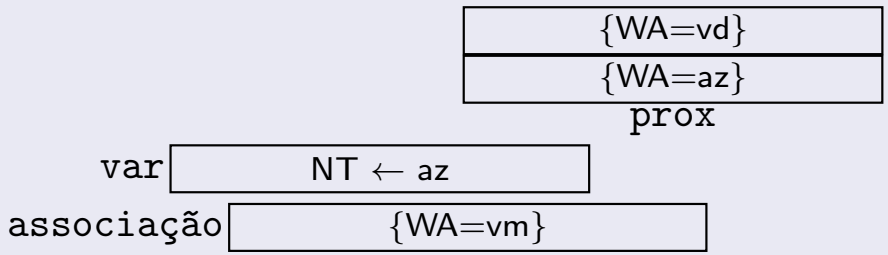
CSP – Backtracking Search

Algoritmo – Funcionamento

Função *BUSCA(csp)*: **associação**

- Empilhe a raiz {} em *prox*;
- **enquanto** *prox* $\neq$ *vazia* **faça**
  - *associação* $\leftarrow$ desempilhe uma associação de *prox*;
  - **se** *associação for completa* **então**
    - **retorna** *associação*
  - *var* $\leftarrow$ seleciona uma variável sem valor atribuído em *associação*;
  - **para cada** *valor no domínio de var* **faça**
    - **se** *valor for consistente com a associação* **então**
      - Empilhe $\{var = valor\} \cup$ *associação* em *prox*;
- **retorna** *falha*





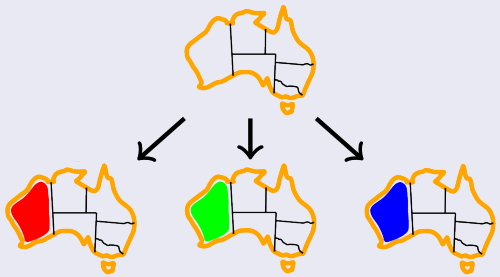
Fonte: Adaptado dos slides de AIMA. Russell & Norvig.

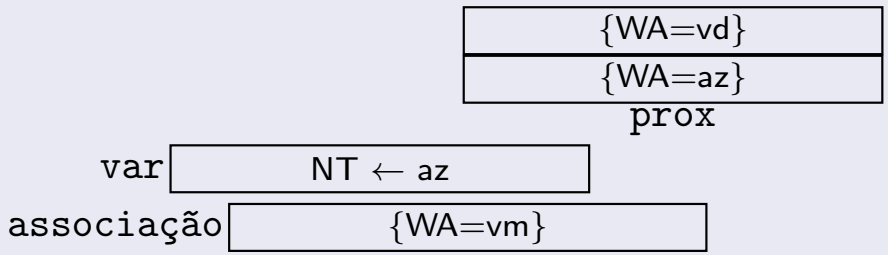
CSP – Backtracking Search

Algoritmo – Funcionamento

Função *BUSCA(csp)*: **associação**

- Empilhe a raiz {} em *prox*;
- **enquanto** *prox* $\neq$ *vazia* **faça**
  - *associação* $\leftarrow$ desempilhe uma associação de *prox*;
  - **se** *associação for completa* **então**
    - **retorna** *associação*
  - *var* $\leftarrow$ seleciona uma variável sem valor atribuído em *associação*;
  - **para cada** *valor no domínio de var* **faça**
    - **se** *valor for consistente com a associação* **então**
      - Empilhe $\{var = valor\} \cup$ *associação* em *prox*;
- **retorna** *falha*



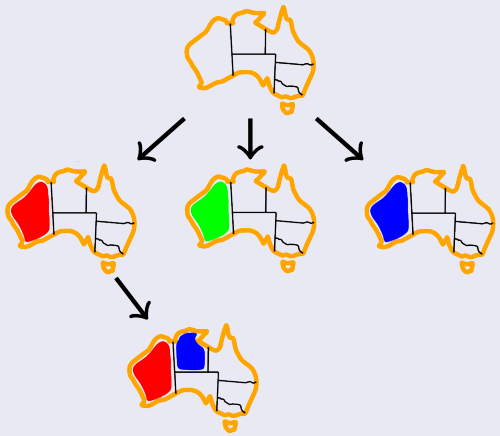


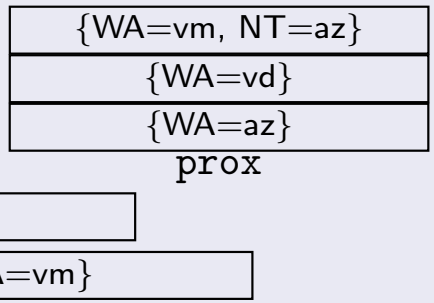
Fonte: Adaptado dos slides de AIMA. Russell & Norvig.

CSP – Backtracking Search

Algoritmo – Funcionamento

Função *BUSCA(csp)*: **associação**
- Empilhe a raiz {} em *prox*;
- **enquanto** *prox* $\neq$ *vazia* **faça**
  - *associação* $\leftarrow$ desempilhe uma associação de *prox*;
  - **se** *associação for completa* **então**
    - **retorna** *associação*
  - *var* $\leftarrow$ seleciona uma variável sem valor atribuído em *associação*;
  - **para cada** *valor no domínio de var* **faça**
    - **se** *valor for consistente com a associação* **então**
      - Empilhe $\{var = valor\} \cup$ *associação* em *prox*;
- **retorna** *falha*





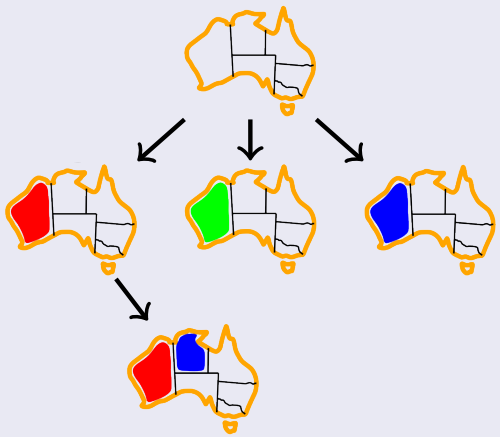
Fonte: Adaptado dos slides de AIMA. Russell & Norvig.

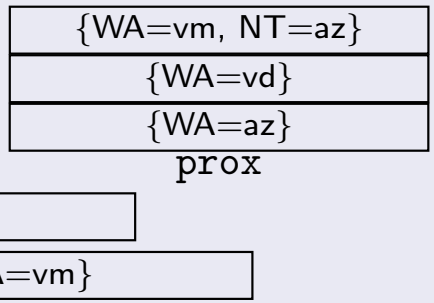
CSP – Backtracking Search

Algoritmo – Funcionamento

Função *BUSCA(csp)*: **associação**

- Empilhe a raiz {} em *prox*;
- **enquanto** *prox* $\neq$ *vazia* **faça**
  - *associação* $\leftarrow$ desempilhe uma associação de *prox*;
  - **se** *associação for completa* **então**
    - **retorna** *associação*
  - *var* $\leftarrow$ seleciona uma variável sem valor atribuído em *associação*;
  - **para cada** *valor no domínio de var* **faça**
    - **se** *valor for consistente com a associação* **então**
      - Empilhe $\{var = valor\} \cup$ *associação* em *prox*;
- **retorna** *falha*





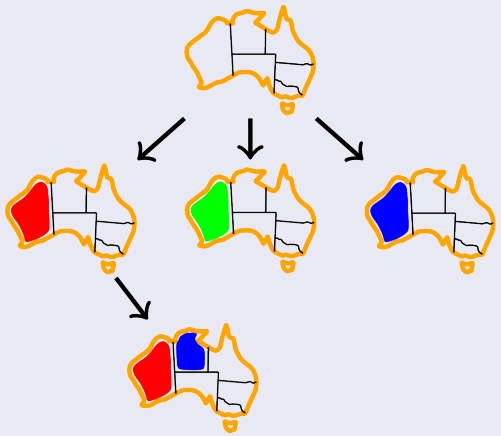
Fonte: Adaptado dos slides de AIMA. Russell & Norvig.

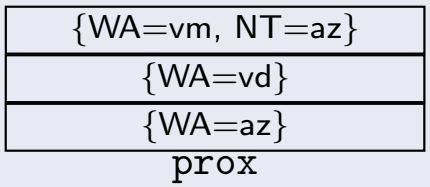
CSP – Backtracking Search

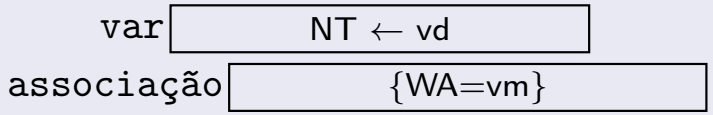
Algoritmo – Funcionamento

Função *BUSCA(csp)*: **associação**

- Empilhe a raiz {} em *prox*;
- **enquanto** *prox* $\neq$ *vazia* **faça**
  - *associação* $\leftarrow$ desempilhe uma associação de *prox*;
  - **se** *associação for completa* **então**
    - **retorna** *associação*
  - *var* $\leftarrow$ seleciona uma variável sem valor atribuído em *associação*;
  - **para cada** *valor no domínio de var* **faça**
    - **se** *valor for consistente com a associação* **então**
      - Empilhe $\{var = valor\} \cup$ *associação* em *prox*;
- **retorna** *falha*







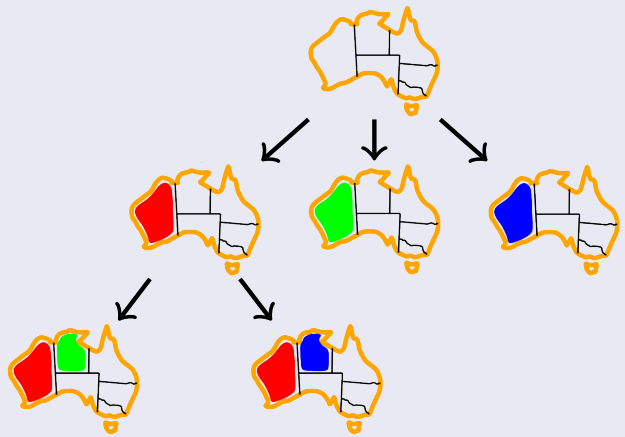
Fonte: Adaptado dos slides de AIMA. Russell & Norvig.

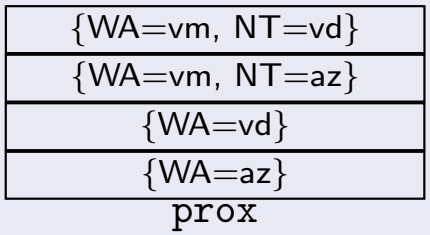
CSP – Backtracking Search

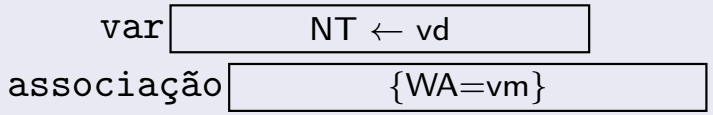
Algoritmo – Funcionamento

Função *BUSCA(csp)*: **associação**

- Empilhe a raiz {} em *prox*;
- **enquanto** *prox* $\neq$ *vazia* **faça**
  - *associação* $\leftarrow$ desempilhe uma associação de *prox*;
  - **se** *associação for completa* **então**
    - **retorna** *associação*
  - *var* $\leftarrow$ seleciona uma variável sem valor atribuído em *associação*;
  - **para cada** *valor no domínio de var* **faça**
    - **se** *valor for consistente com a associação* **então**
      - Empilhe $\{var = valor\} \cup$ *associação* em *prox*;
- **retorna** *falha*







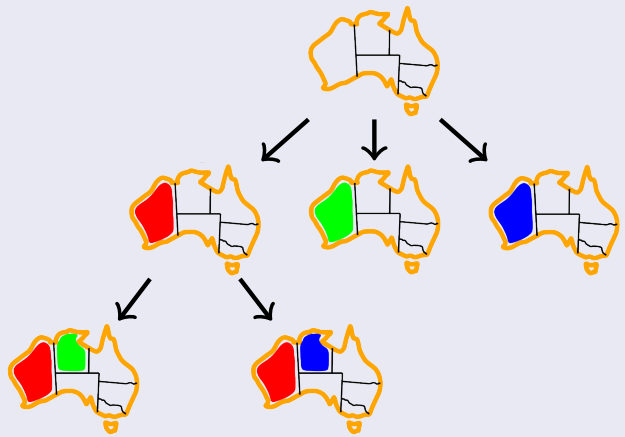
Fonte: Adaptado dos slides de AIMA. Russell & Norvig.

CSP – Backtracking Search

Algoritmo – Funcionamento

Função *BUSCA(csp)*: **associação**

- Empilhe a raiz {} em *prox*;
- **enquanto** *prox* $\neq$ *vazia* **faça**
  - *associação* $\leftarrow$ desempilhe uma associação de *prox*;
  - **se** *associação for completa* **então**
    - **retorna** *associação*
  - *var* $\leftarrow$ seleciona uma variável sem valor atribuído em *associação*;
  - **para cada** *valor no domínio de var* **faça**
    - **se** *valor for consistente com a associação* **então**
      - Empilhe $\{var = valor\} \cup$ *associação* em *prox*;
- **retorna** *falha*



{WA=vm, NT=vd}
{WA=vm, NT=az}
{WA=vd}
{WA=az}

prox

var | NT $\leftarrow$ vm

associação | {WA=vm}

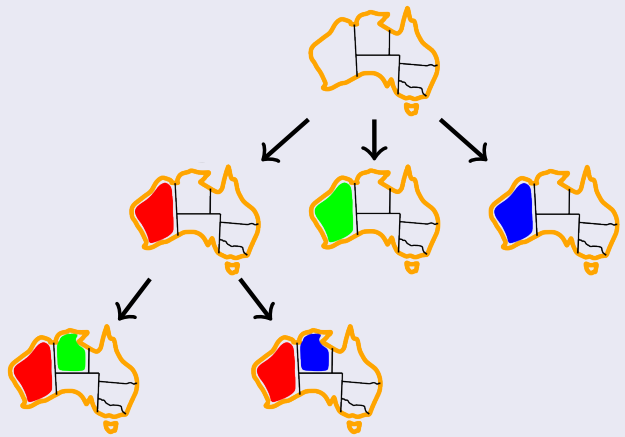
Fonte: Adaptado dos slides de AIMA. Russell & Norvig.

CSP – Backtracking Search

Algoritmo – Funcionamento

Função *BUSCA(csp)*: **associação**

- Empilhe a raiz {} em *prox*;
- **enquanto** *prox* $\neq$ *vazia* **faça**
  - *associação* $\leftarrow$ desempilhe uma associação de *prox*;
  - **se** *associação for completa* **então**
    - **retorna** *associação*
  - *var* $\leftarrow$ seleciona uma variável sem valor atribuído em *associação*;
  - **para cada** *valor no domínio de var* **faça**
    - **se** *valor for consistente com a associação* **então**
      - Empilhe $\{var = valor\} \cup$ *associação* em *prox*;
- **retorna** *falha*



{WA=vm, NT=vd}
{WA=vm, NT=az}
{WA=vd}
{WA=az}

prox

var | NT $\leftarrow$ vm

associação | {WA=vm}

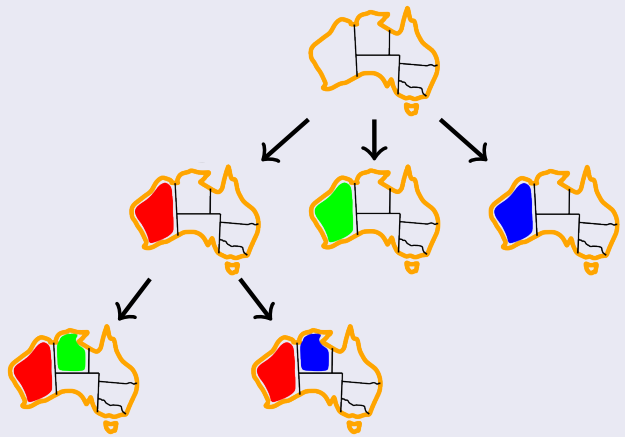
Fonte: Adaptado dos slides de AIMA. Russell & Norvig.

CSP – Backtracking Search

Algoritmo – Funcionamento

Função *BUSCA(csp)*: **associação**

- Empilhe a raiz {} em *prox*;
- **enquanto** *prox* $\neq$ *vazia* **faça**
  - *associação* $\leftarrow$ desempilhe uma associação de *prox*;
  - **se** *associação for completa* **então**
    - **retorna** *associação*
  - *var* $\leftarrow$ seleciona uma variável sem valor atribuído em *associação*;
  - **para cada** *valor no domínio de var* **faça**
    - **se** *valor for consistente com a associação* **então**
      - Empilhe $\{var = valor\} \cup$ *associação* em *prox*;
- **retorna** *falha*



{WA=vm, NT=vd}
{WA=vm, NT=az}
{WA=vd}
{WA=az}

prox

var | NT $\leftarrow$ vm

associação | {WA=vm}

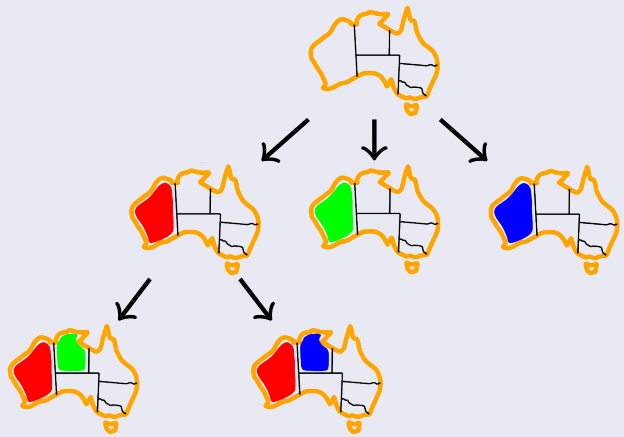
Fonte: Adaptado dos slides de AIMA. Russell & Norvig.

CSP – Backtracking Search

Algoritmo – Funcionamento

Função *BUSCA(csp)*: **associação**

- Empilhe a raiz {} em *prox*;
- **enquanto** *prox* $\neq$ *vazia* **faça**
  - *associação* $\leftarrow$ desempilhe uma associação de *prox*;
  - **se** *associação for completa* **então**
    - **retorna** *associação*
  - *var* $\leftarrow$ seleciona uma variável sem valor atribuído em *associação*;
  - **para cada** *valor no domínio de var* **faça**
    - **se** *valor for consistente com a associação* **então**
      - Empilhe $\{var = valor\} \cup$ *associação* em *prox*;
- **retorna** *falha*



{WA=vm, NT=vd}
{WA=vm, NT=az}
{WA=vd}
{WA=az}

prox

var | NT $\leftarrow$ vm

associação | {WA=vm}

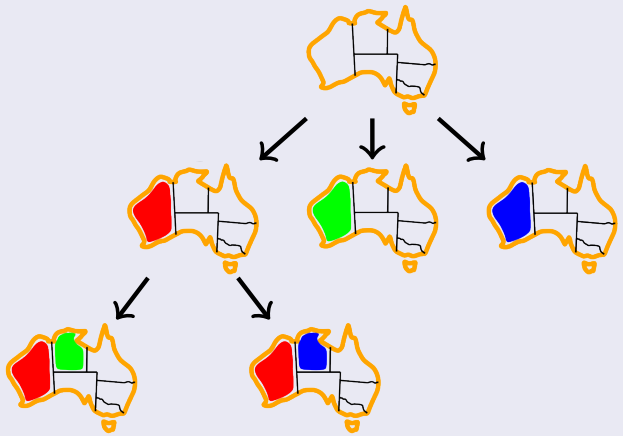
Fonte: Adaptado dos slides de AIMA. Russell & Norvig.

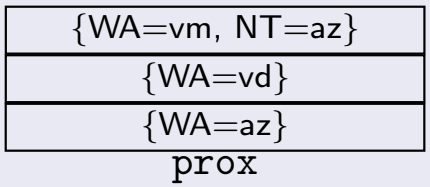
CSP – Backtracking Search

Algoritmo – Funcionamento

Função *BUSCA(csp)*: **associação**

- Empilhe a raiz {} em *prox*;
- **enquanto** *prox* $\neq$ *vazia* **faça**
  - *associação* $\leftarrow$ desempilhe uma associação de *prox*;
  - **se** *associação for completa* **então**
    - **retorna** *associação*
  - *var* $\leftarrow$ seleciona uma variável sem valor atribuído em *associação*;
  - **para cada** *valor no domínio de var* **faça**
    - **se** *valor for consistente com a associação* **então**
      - Empilhe $\{var = valor\} \cup$ *associação* em *prox*;
- **retorna** *falha*





var: NT $\leftarrow$ vm

associação: {WA=vm, NT=vd}

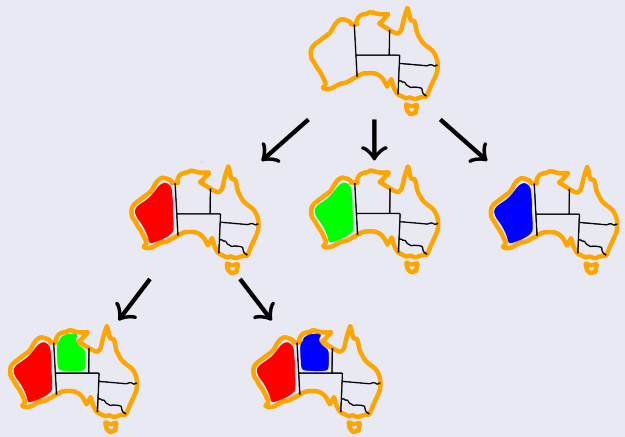
Fonte: Adaptado dos slides de AIMA. Russell & Norvig.

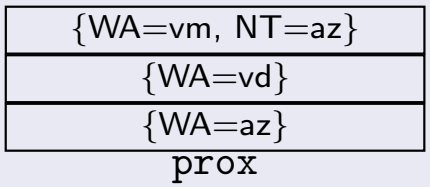
CSP – Backtracking Search

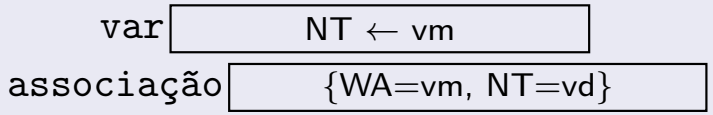
Algoritmo – Funcionamento

Função *BUSCA(csp)*: **associação**

- Empilhe a raiz {} em *prox*;
- **enquanto** *prox* $\neq$ *vazia* **faça**
  - *associação* $\leftarrow$ desempilhe uma associação de *prox*;
  - **se** *associação for completa* **então**
    - **retorna** *associação*
  - *var* $\leftarrow$ seleciona uma variável sem valor atribuído em *associação*;
  - **para cada** *valor no domínio de var* **faça**
    - **se** *valor for consistente com a associação* **então**
      - Empilhe $\{var = valor\} \cup$ *associação* em *prox*;
- **retorna** *falha*







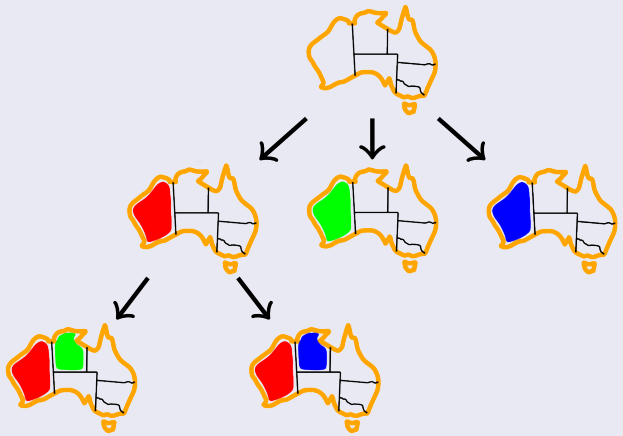
Fonte: Adaptado dos slides de AIMA. Russell & Norvig.

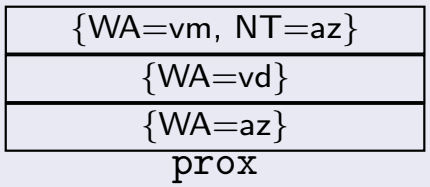
CSP – Backtracking Search

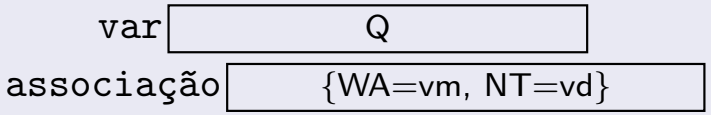
Algoritmo – Funcionamento

Função *BUSCA(csp)*: **associação**

- Empilhe a raiz {} em *prox*;
- **enquanto** *prox* $\neq$ *vazia* **faça**
  - *associação* $\leftarrow$ desempilhe uma associação de *prox*;
  - **se** *associação for completa* **então**
    - **retorna** *associação*
  - *var* $\leftarrow$ seleciona uma variável sem valor atribuído em *associação*;
  - **para cada** *valor no domínio de var* **faça**
    - **se** *valor for consistente com a associação* **então**
      - Empilhe $\{var = valor\} \cup$ *associação* em *prox*;
- **retorna** *falha*







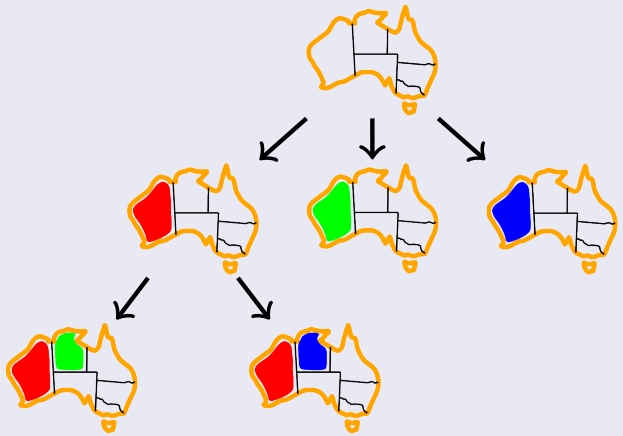
Fonte: Adaptado dos slides de AIMA. Russell & Norvig.

CSP – Backtracking Search

Algoritmo – Funcionamento

Função *BUSCA(csp)*: **associação**

- Empilhe a raiz {} em *prox*;
- **enquanto** *prox* $\neq$ *vazia* **faça**
  - *associação* $\leftarrow$ desempilhe uma associação de *prox*;
  - **se** *associação for completa* **então**
    - **retorna** *associação*
  - *var* $\leftarrow$ seleciona uma variável sem valor atribuído em *associação*;
  - **para cada** *valor no domínio de var* **faça**
    - **se** *valor for consistente com a associação* **então**
      - Empilhe $\{var = valor\} \cup$ *associação* em *prox*;
- **retorna** *falha*



{WA=vm, NT=az}
{WA=vd}
{WA=az}

prox

var | Q $\leftarrow$ az

associação | {WA=vm, NT=vd}

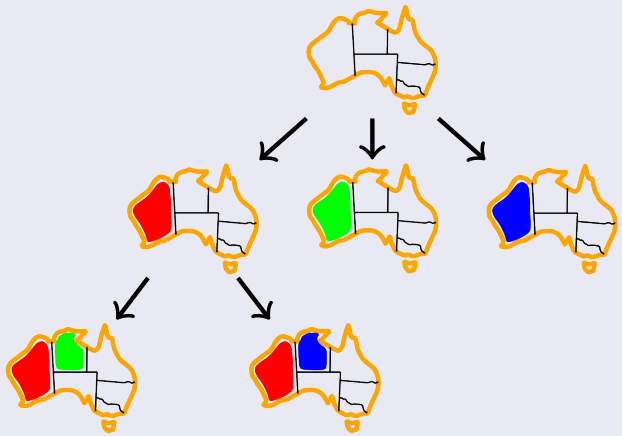
Fonte: Adaptado dos slides de AIMA. Russell & Norvig.

CSP – Backtracking Search

Algoritmo – Funcionamento

Função *BUSCA(csp)*: **associação**

- Empilhe a raiz {} em *prox*;
- **enquanto** *prox* $\neq$ *vazia* **faça**
  - *associação* $\leftarrow$ desempilhe uma associação de *prox*;
  - **se** *associação for completa* **então**
    - **retorna** *associação*
  - *var* $\leftarrow$ seleciona uma variável sem valor atribuído em *associação*;
  - **para cada** *valor no domínio de var* **faça**
    - **se** *valor for consistente com a associação* **então**
      - Empilhe $\{var = valor\} \cup$ *associação* em *prox*;
- **retorna** *falha*



{WA=vm, NT=az}
{WA=vd}
{WA=az}

prox

var: Q $\leftarrow$ az

associação: {WA=vm, NT=vd}

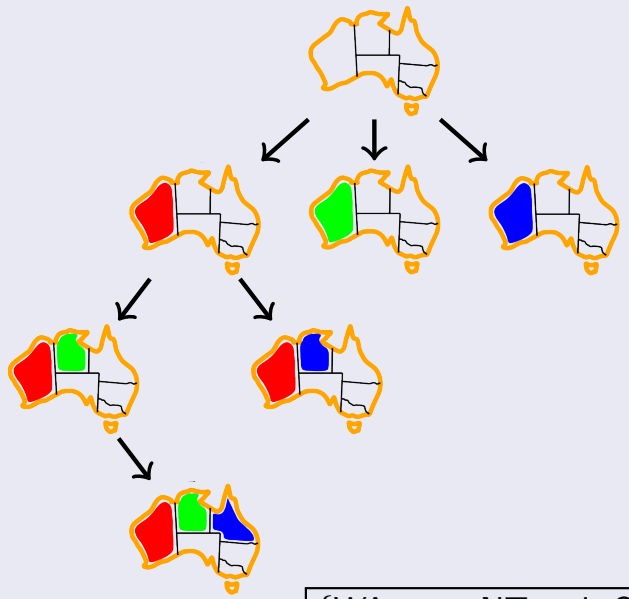
Fonte: Adaptado dos slides de AIMA. Russell & Norvig.

CSP – Backtracking Search

Algoritmo – Funcionamento

Função *BUSCA(csp)*: **associação**

- Empilhe a raiz {} em *prox*;
- **enquanto** *prox* $\neq$ *vazia* **faça**
  - *associação* $\leftarrow$ desempilhe uma associação de *prox*;
  - **se** *associação for completa* **então**
    - **retorna** *associação*
  - *var* $\leftarrow$ seleciona uma variável sem valor atribuído em *associação*;
  - **para cada** *valor no domínio de var* **faça**
    - **se** *valor for consistente com a associação* **então**
      - Empilhe $\{var = valor\} \cup$ *associação* em *prox*;
- **retorna** *falha*



{WA=vm, NT=vd, Q=az}
{WA=vm, NT=az}
{WA=vd}
{WA=az}

prox

var	Q ← az
associação	{WA=vm, NT=vd}

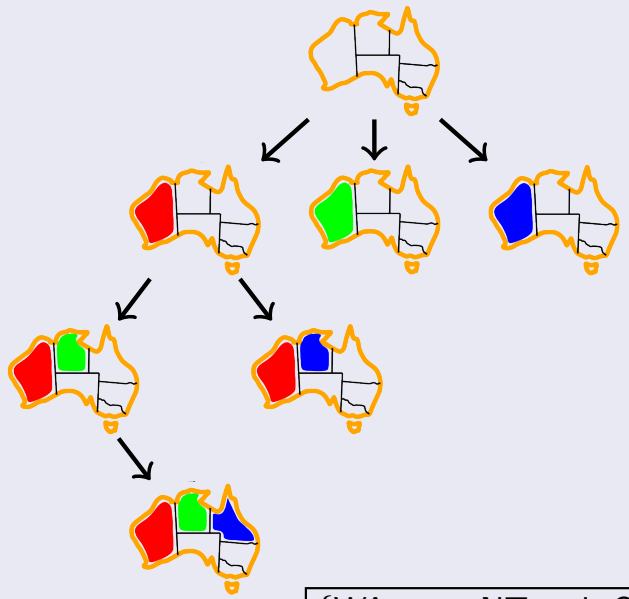
Fonte: Adaptado dos slides de AIMA. Russell & Norvig.

CSP – Backtracking Search

Algoritmo – Funcionamento

Função *BUSCA(csp)*: **associação**

- Empilhe a raiz {} em *prox*;
- **enquanto** *prox* $\neq$ *vazia* **faça**
  - *associação* $\leftarrow$ desempilhe uma associação de *prox*;
  - **se** *associação for completa* **então**
    - **retorna** *associação*
  - *var* $\leftarrow$ seleciona uma variável sem valor atribuído em *associação*;
  - **para cada** *valor no domínio de var* **faça**
    - **se** *valor for consistente com a associação* **então**
      - Empilhe $\{var = valor\} \cup$ *associação* em *prox*;
- **retorna** *falha*



{WA=vm, NT=vd, Q=az}
{WA=vm, NT=az}
{WA=vd}
{WA=az}

prox

var	Q ← vd
associação	{WA=vm, NT=vd}

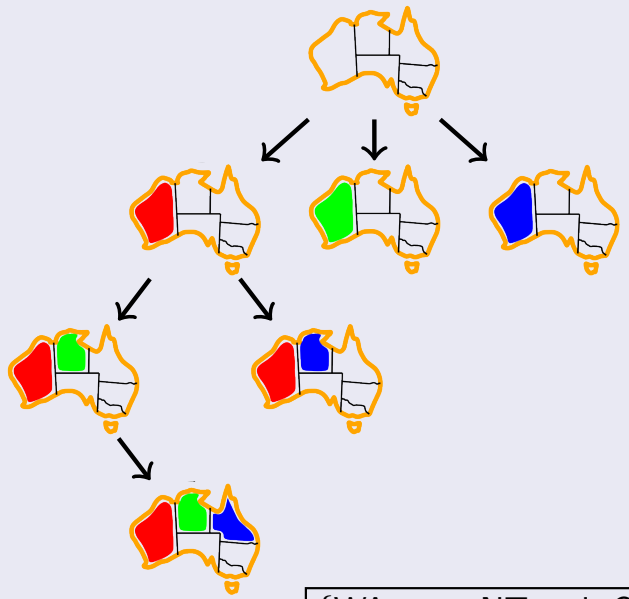
Fonte: Adaptado dos slides de AIMA. Russell & Norvig.

CSP – Backtracking Search

Algoritmo – Funcionamento

Função *BUSCA(csp)*: **associação**

- Empilhe a raiz {} em *prox*;
- **enquanto** *prox* $\neq$ *vazia* **faça**
  - *associação* $\leftarrow$ desempilhe uma associação de *prox*;
  - **se** *associação for completa* **então**
    - **retorna** *associação*
  - *var* $\leftarrow$ seleciona uma variável sem valor atribuído em *associação*;
  - **para cada** *valor no domínio de var* **faça**
    - **se** *valor for consistente com a associação* **então**
      - Empilhe $\{var = valor\} \cup$ *associação* em *prox*;
- **retorna** *falha*



{WA=vm, NT=vd, Q=az}
{WA=vm, NT=az}
{WA=vd}
{WA=az}

prox

var	Q $\leftarrow$ vd
associação	{WA=vm, NT=vd}

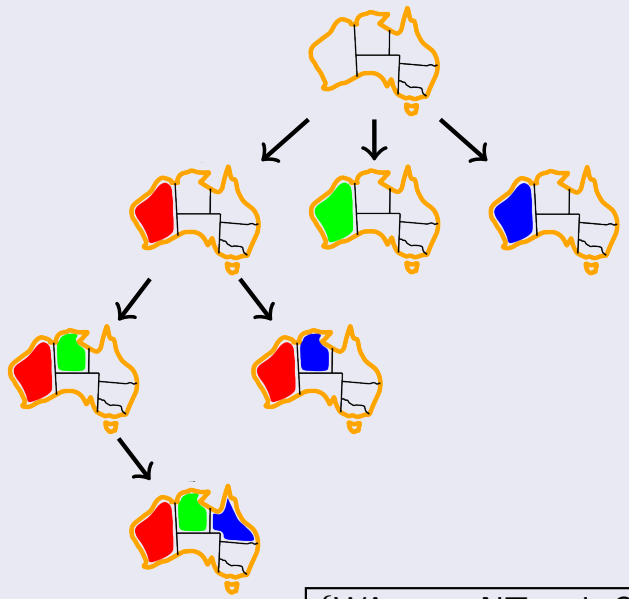
Fonte: Adaptado dos slides de AIMA. Russell & Norvig.

CSP – Backtracking Search

Algoritmo – Funcionamento

Função *BUSCA(csp)*: **associação**

- Empilhe a raiz {} em *prox*;
- **enquanto** *prox* $\neq$ *vazia* **faça**
  - *associação* $\leftarrow$ desempilhe uma associação de *prox*;
  - **se** *associação for completa* **então**
    - **retorna** *associação*
  - *var* $\leftarrow$ seleciona uma variável sem valor atribuído em *associação*;
  - **para cada** *valor no domínio de var* **faça**
    - **se** *valor for consistente com a associação* **então**
      - Empilhe $\{var = valor\} \cup$ *associação* em *prox*;
- **retorna** *falha*



prox
{WA=vm, NT=vd, Q=az}
{WA=vm, NT=az}
{WA=vd}
{WA=az}

var	Q $\leftarrow$ vm
associação	{WA=vm, NT=vd}

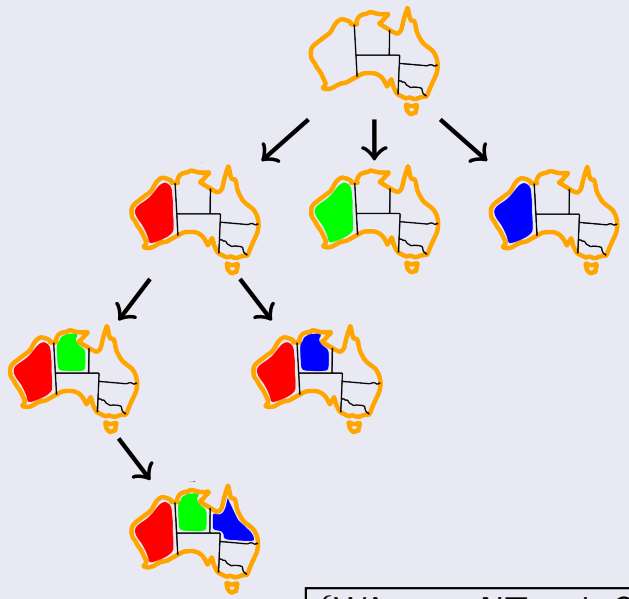
Fonte: Adaptado dos slides de AIMA. Russell & Norvig.

CSP – Backtracking Search

Algoritmo – Funcionamento

Função *BUSCA(csp)*: **associação**

- Empilhe a raiz {} em *prox*;
- **enquanto** *prox* $\neq$ *vazia* **faça**
  - *associação* $\leftarrow$ desempilhe uma associação de *prox*;
  - **se** *associação for completa* **então**
    - **retorna** *associação*
  - *var* $\leftarrow$ seleciona uma variável sem valor atribuído em *associação*;
  - **para cada** *valor no domínio de var* **faça**
    - **se** *valor for consistente com a associação* **então**
      - Empilhe $\{var = valor\} \cup$ *associação* em *prox*;
- **retorna** *falha*



{WA=vm, NT=vd, Q=az}
{WA=vm, NT=az}
{WA=vd}
{WA=az}

prox

var	Q ← vm
associação	{WA=vm, NT=vd}

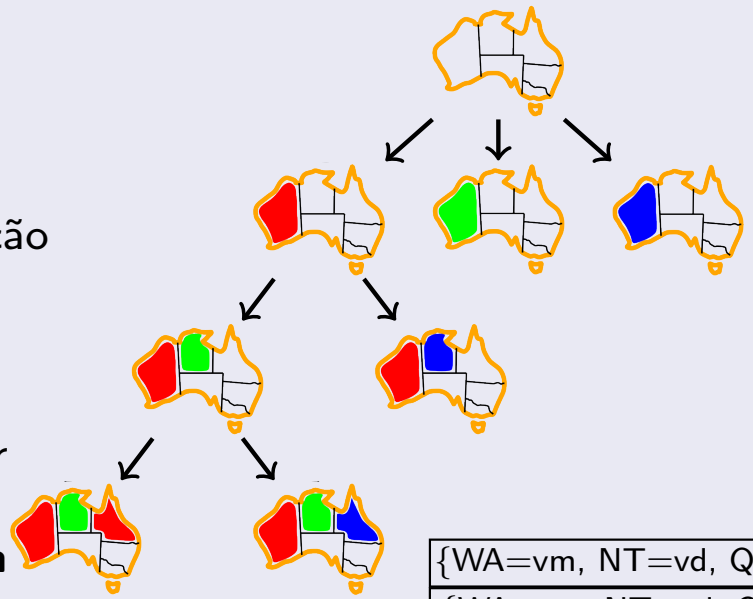
Fonte: Adaptado dos slides de AIMA. Russell & Norvig.

CSP – Backtracking Search

Algoritmo – Funcionamento

Função *BUSCA(csp)*: **associação**

- Empilhe a raiz {} em *prox*;
- **enquanto** *prox* $\neq$ *vazia* **faça**
  - *associação* $\leftarrow$ desempilhe uma associação de *prox*;
  - **se** *associação for completa* **então**
    - **retorna** *associação*
  - *var* $\leftarrow$ seleciona uma variável sem valor atribuído em *associação*;
  - **para cada** *valor no domínio de var* **faça**
    - **se** *valor for consistente com a associação* **então**
      - Empilhe $\{var = valor\} \cup$ *associação* em *prox*;
- **retorna** *falha*



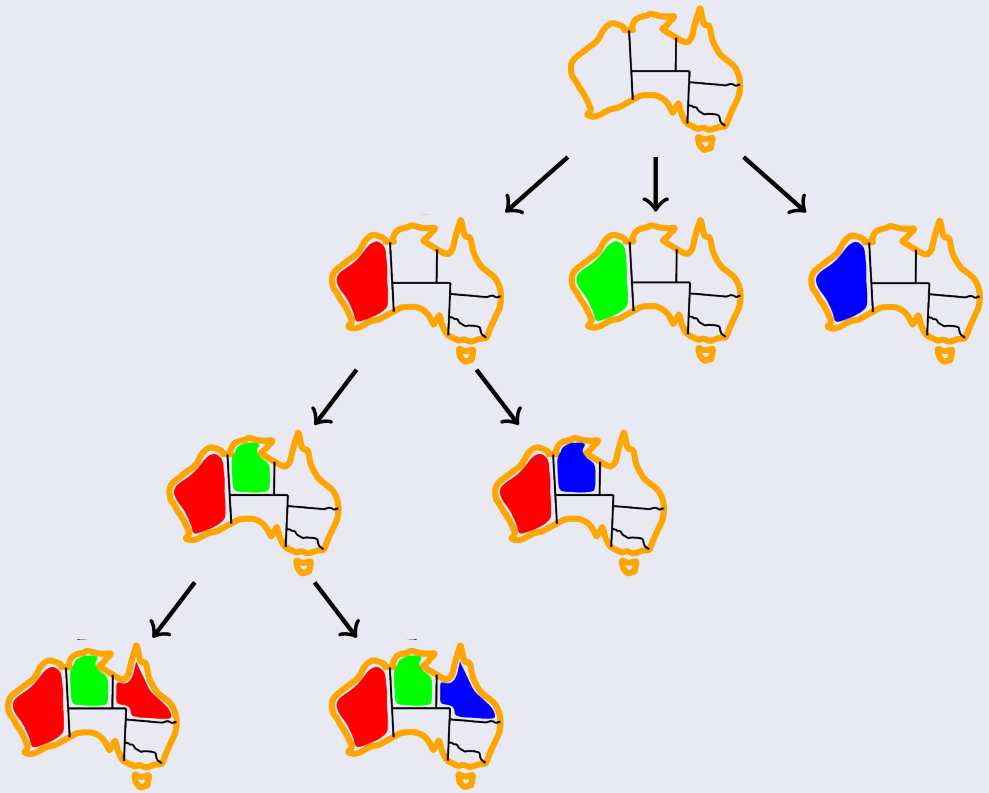
prox
{WA=vm, NT=vd, Q=vm}
{WA=vm, NT=vd, Q=az}
{WA=vm, NT=az}
{WA=vd}
{WA=az}

var	Q $\leftarrow$ vm
associação	{WA=vm, NT=vd}

Fonte: Adaptado dos slides de AIMA. Russell & Norvig.

CSP – Backtracking Search

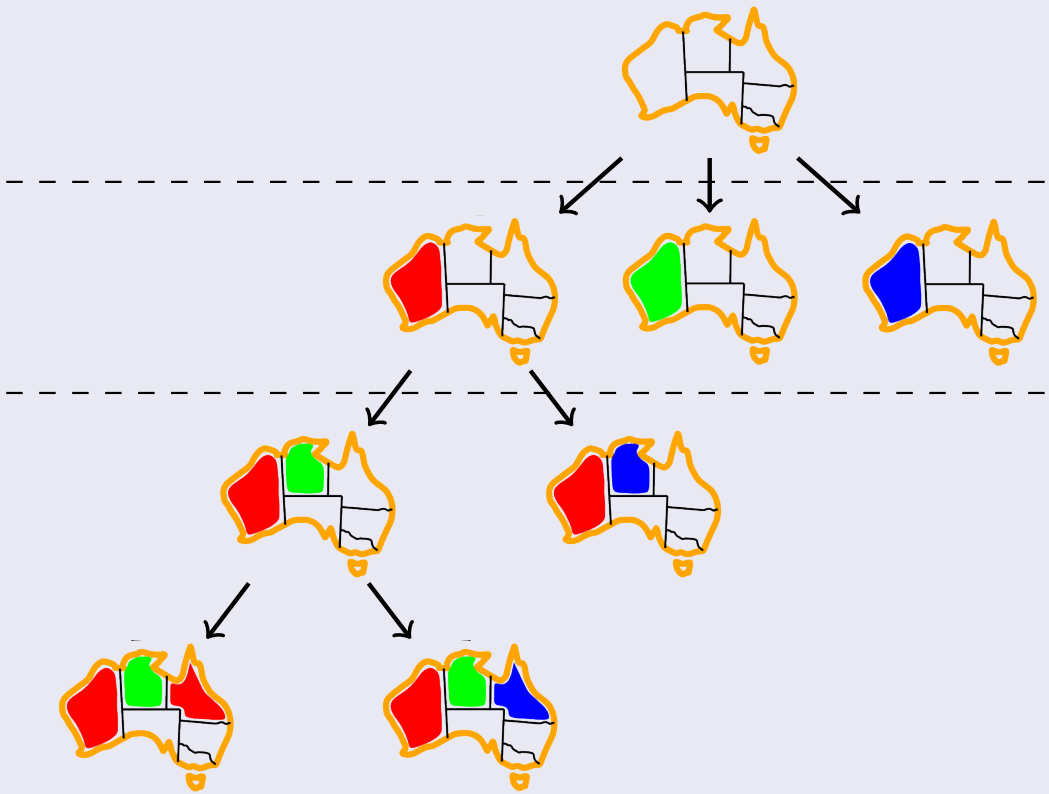
Algoritmo – Funcionamento



Fonte: Adaptado dos slides de AIMA. Russell & Norvig.

CSP – Backtracking Search

Algoritmo – Funcionamento

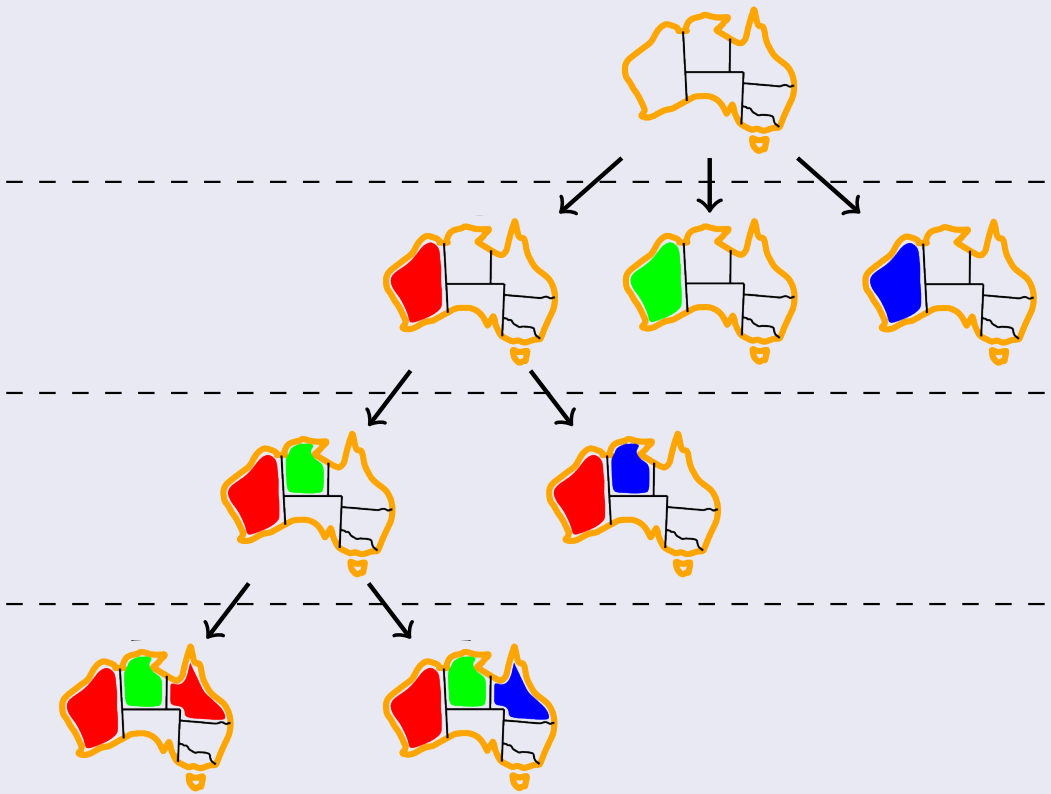


Primeiro nível: olhamos apenas a primeira variável

Fonte: Adaptado dos slides de AIMA. Russell & Norvig.

CSP – Backtracking Search

Algoritmo – Funcionamento



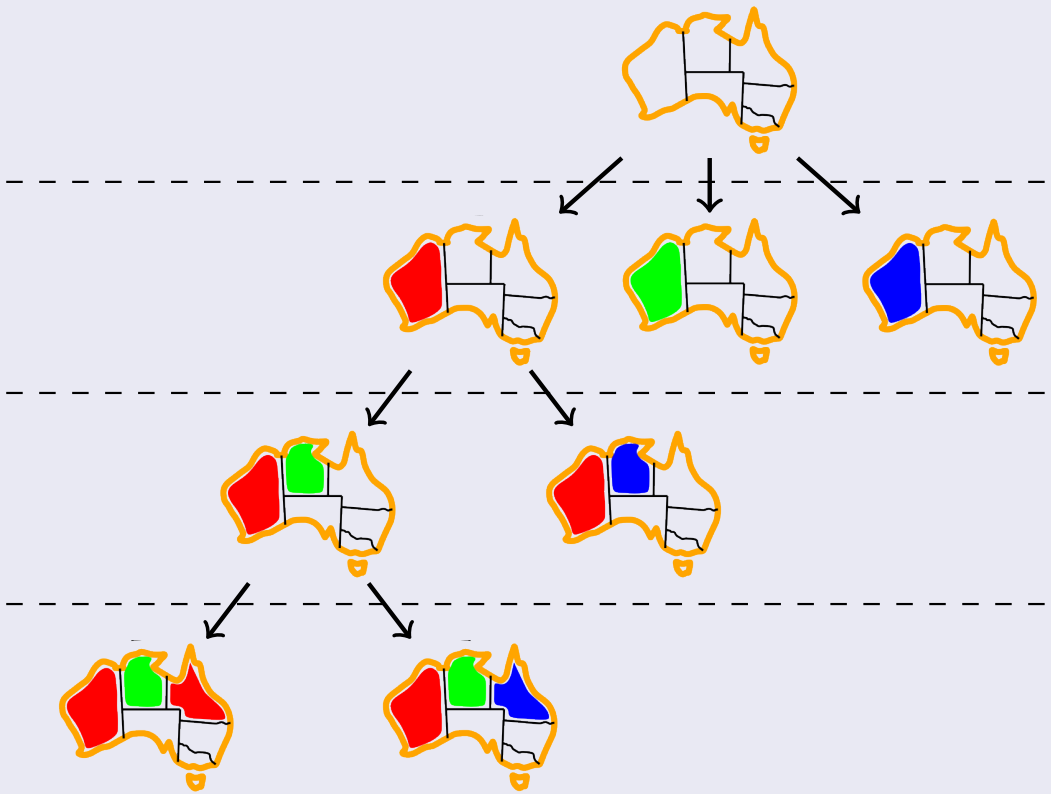
Primeiro nível: olhamos apenas a primeira variável

Segundo nível: olhamos apenas a segunda variável

Fonte: Adaptado dos slides de AIMA. Russell & Norvig.

CSP – Backtracking Search

Algoritmo – Funcionamento



Primeiro nível: olhamos apenas a primeira variável

Segundo nível: olhamos apenas a segunda variável

E assim por diante...

Fonte: Adaptado dos slides de AIMA. Russell & Norvig.

CSP – Backtracking Search

Algoritmo – Funcionamento

- Trata-se de uma busca em profundidade:

CSP – Backtracking Search

Algoritmo – Funcionamento

- Trata-se de uma busca em profundidade:
  - Escolhe valores para uma única variável a cada vez

CSP – Backtracking Search

Algoritmo – Funcionamento

- Trata-se de uma busca em profundidade:
  - Escolhe valores para uma única variável a cada vez
  - Volta na árvore (*backtracking*) quando a variável não consegue assumir nenhum valor legal

CSP – Backtracking Search

Algoritmo – Funcionamento

- Trata-se de uma busca em profundidade:
  - Escolhe valores para uma única variável a cada vez
  - Volta na árvore (*backtracking*) quando a variável não consegue assumir nenhum valor legal
- Note que o algoritmo representa 1 único estado

CSP – Backtracking Search

Algoritmo – Funcionamento

- Trata-se de uma busca em profundidade:
  - Escolhe valores para uma única variável a cada vez
  - Volta na árvore (*backtracking*) quando a variável não consegue assumir nenhum valor legal
- Note que o algoritmo representa 1 único estado
  - Alterando essa representação, em vez de criar novas

CSP – Backtracking Search

Algoritmo – Funcionamento

- Trata-se de uma busca em profundidade:
  - Escolhe valores para uma única variável a cada vez
  - Volta na árvore (*backtracking*) quando a variável não consegue assumir nenhum valor legal
- Note que o algoritmo representa 1 único estado
  - Alterando essa representação, em vez de criar novas
- É o algoritmo não-informado básico para CSP

CSP – Backtracking Search

Algoritmo – Funcionamento

- Trata-se de uma busca em profundidade:
  - Escolhe valores para uma única variável a cada vez
  - Volta na árvore (*backtracking*) quando a variável não consegue assumir nenhum valor legal
- Note que o algoritmo representa 1 único estado
  - Alterando essa representação, em vez de criar novas
- É o algoritmo não-informado básico para CSP
  - Pode ser usado para qualquer outra busca, no entanto

CSP – Backtracking Search

Questões importantes que afetam a eficiência

CSP – Backtracking Search

Questões importantes que afetam a eficiência

- Ordem das variáveis

CSP – Backtracking Search

Questões importantes que afetam a eficiência

- Ordem das variáveis
  - Que variável deve ser observada a cada passo?

CSP – Backtracking Search

Questões importantes que afetam a eficiência

- Ordem das variáveis
  - Que variável deve ser observada a cada passo?
- Ordem dos valores

CSP – Backtracking Search

Questões importantes que afetam a eficiência

- Ordem das variáveis
  - Que variável deve ser observada a cada passo?
- Ordem dos valores
  - Em que ordem deveriam ser os valores testados?

CSP – Backtracking Search

Questões importantes que afetam a eficiência

- Ordem das variáveis
  - Que variável deve ser observada a cada passo?
- Ordem dos valores
  - Em que ordem deveriam ser os valores testados?
- Podemos tirar vantagem da estrutura do problema?

CSP – Backtracking Search

Questões importantes que afetam a eficiência

- Ordem das variáveis
  - Que variável deve ser observada a cada passo?
- Ordem dos valores
  - Em que ordem deveriam ser os valores testados?
- Podemos tirar vantagem da estrutura do problema?
- Podemos detectar falhas inevitáveis antes de acontecerem?

Backtracking Search – Heurísticas

Ordem das variáveis

- Padrão: usar a próxima variável sem valor, em uma fila $\rightarrow$ Ineficiente

Backtracking Search – Heurísticas

Ordem das variáveis

- Padrão: usar a próxima variável sem valor, em uma fila $\rightarrow$ Ineficiente
- Heurística: **Valores restantes mínimos**

Backtracking Search – Heurísticas

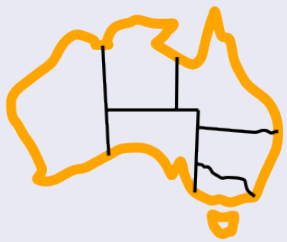
Ordem das variáveis

- Padrão: usar a próxima variável sem valor, em uma fila $\rightarrow$ Ineficiente
- Heurística: **Valores restantes mínimos**
  - Escolha a variável com o menor número de valores legais restantes (ou seja, a que tem mais chance de falhar)

Backtracking Search – Heurísticas

Ordem das variáveis

- Padrão: usar a próxima variável sem valor, em uma fila $\rightarrow$ Ineficiente
- Heurística: **Valores restantes mínimos**
  - Escolha a variável com o menor número de valores legais restantes (ou seja, a que tem mais chance de falhar)
  - No início não há restrição $\rightarrow$ escolhemos qualquer uma

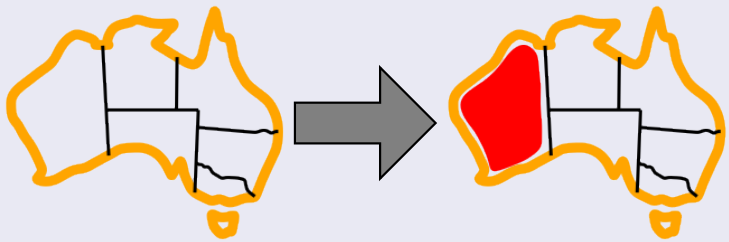


Fonte: Adaptado dos slides de AIMA. Russell & Norvig.

Backtracking Search – Heurísticas

Ordem das variáveis

- Padrão: usar a próxima variável sem valor, em uma fila $\rightarrow$ Ineficiente
- Heurística: **Valores restantes mínimos**
  - Escolha a variável com o menor número de valores legais restantes (ou seja, a que tem mais chance de falhar)
  - No início não há restrição $\rightarrow$ escolhemos qualquer uma

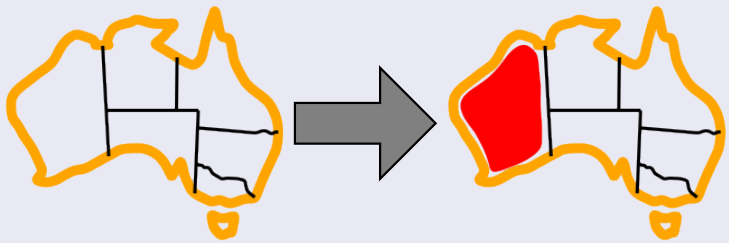


Fonte: Adaptado dos slides de AIMA. Russell & Norvig.

Backtracking Search – Heurísticas

Ordem das variáveis

- Padrão: usar a próxima variável sem valor, em uma fila $\rightarrow$ Ineficiente
- Heurística: **Valores restantes mínimos**
  - Escolha a variável com o menor número de valores legais restantes (ou seja, a que tem mais chance de falhar)
  - $WA = vm$ limita as escolhas em NT e SA, escolhemos uma

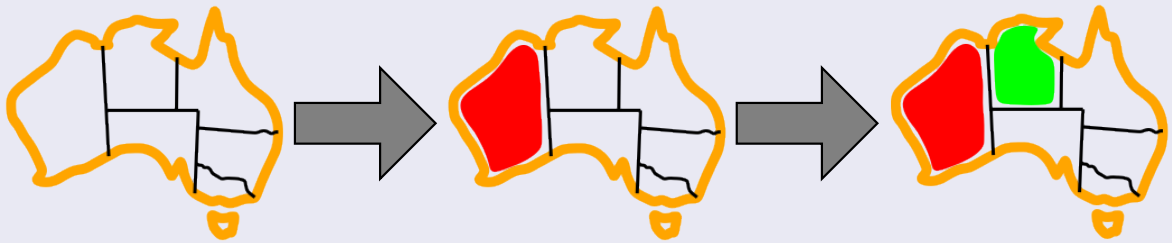


Fonte: Adaptado dos slides de AIMA. Russell & Norvig.

Backtracking Search – Heurísticas

Ordem das variáveis

- Padrão: usar a próxima variável sem valor, em uma fila $\rightarrow$ Ineficiente
- Heurística: **Valores restantes mínimos**
  - Escolha a variável com o menor número de valores legais restantes (ou seja, a que tem mais chance de falhar)
  - $WA = vm$ limita as escolhas em NT e SA, escolhemos uma

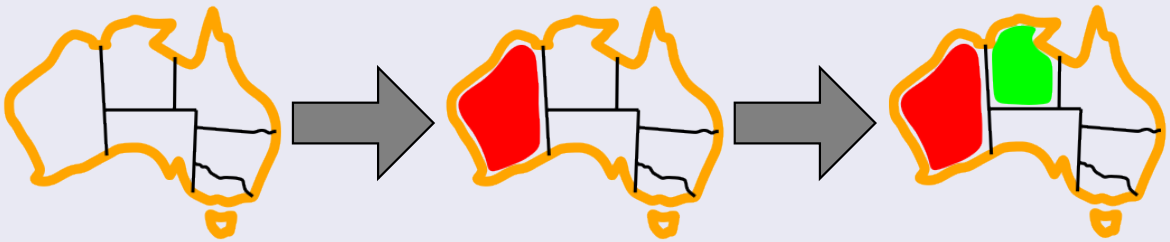


Fonte: Adaptado dos slides de AIMA. Russell & Norvig.

Backtracking Search – Heurísticas

Ordem das variáveis

- Padrão: usar a próxima variável sem valor, em uma fila $\rightarrow$ Ineficiente
- Heurística: **Valores restantes mínimos**
  - Escolha a variável com o menor número de valores legais restantes (ou seja, a que tem mais chance de falhar)
  - Após $WA = vm$ e $NT = vd$, há somente um valor para SA

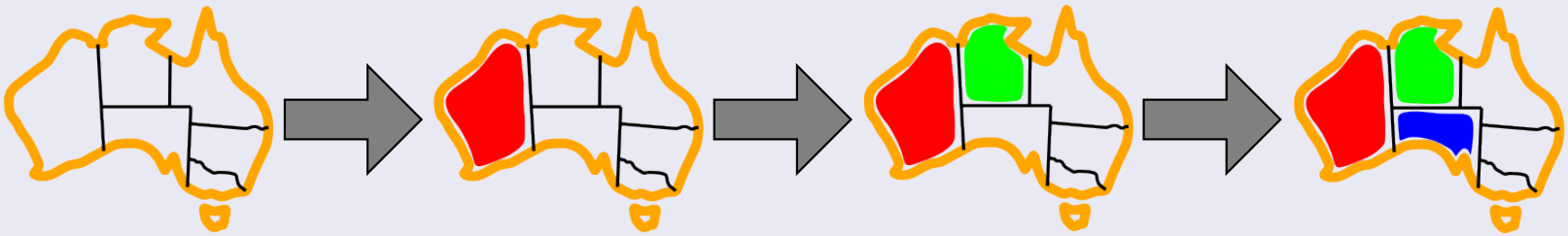


Fonte: Adaptado dos slides de AIMA. Russell & Norvig.

Backtracking Search – Heurísticas

Ordem das variáveis

- Padrão: usar a próxima variável sem valor, em uma fila $\rightarrow$ Ineficiente
- Heurística: **Valores restantes mínimos**
  - Escolha a variável com o menor número de valores legais restantes (ou seja, a que tem mais chance de falhar)
  - Após $WA = vm$ e $NT = vd$, há somente um valor para SA



Fonte: Adaptado dos slides de AIMA. Russell & Norvig.

Backtracking Search – Heurísticas

Ordem das variáveis

- A valores restantes mínimos não ajuda na hora de escolher a primeira região para colorir

Backtracking Search – Heurísticas

Ordem das variáveis

- A valores restantes mínimos não ajuda na hora de escolher a primeira região para colorir
  - Uma vez que todas possuem o mesmo número de valores restantes

Backtracking Search – Heurísticas

Ordem das variáveis

- A valores restantes mínimos não ajuda na hora de escolher a primeira região para colorir
  - Uma vez que todas possuem o mesmo número de valores restantes
- Quando a VRM encontra um empate, podemos desempatar usando a heurística do grau

Backtracking Search – Heurísticas

Ordem das variáveis

- A valores restantes mínimos não ajuda na hora de escolher a primeira região para colorir
  - Uma vez que todas possuem o mesmo número de valores restantes
- Quando a VRM encontra um empate, podemos desempatar usando a heurística do grau
- **Heurística do grau**:

Backtracking Search – Heurísticas

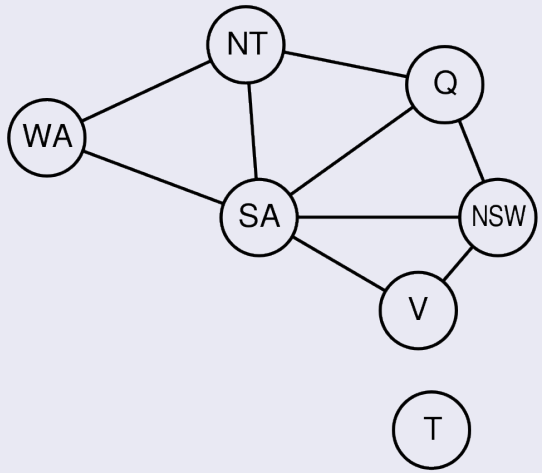
Ordem das variáveis

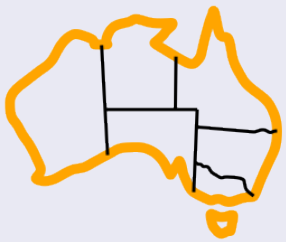
- A valores restantes mínimos não ajuda na hora de escolher a primeira região para colorir
  - Uma vez que todas possuem o mesmo número de valores restantes
- Quando a VRM encontra um empate, podemos desempatar usando a heurística do grau
- **Heurística do grau**:
  - Selecione a variável (ainda sem valor associado) com o maior número de restrições com outras variáveis sem valor associado

Backtracking Search – Heurísticas

Ordem das variáveis – Heurística do grau

- Ex:



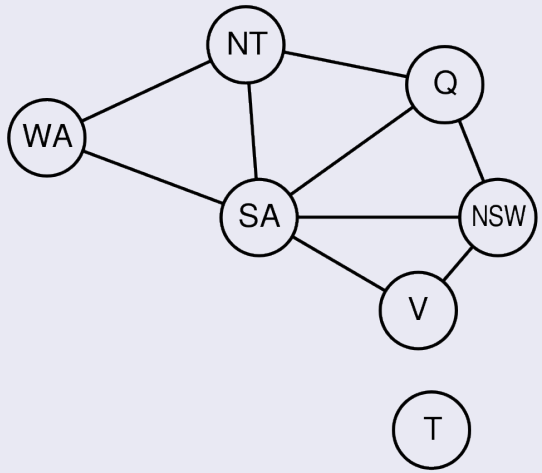


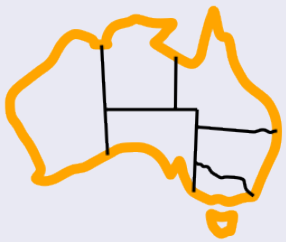
Fonte: Adaptado dos slides de AIMA. Russell & Norvig.

Backtracking Search – Heurísticas

Ordem das variáveis – Heurística do grau

- Ex:
  - SA tem o maior número de arestas $\rightarrow$ maior número de restrições



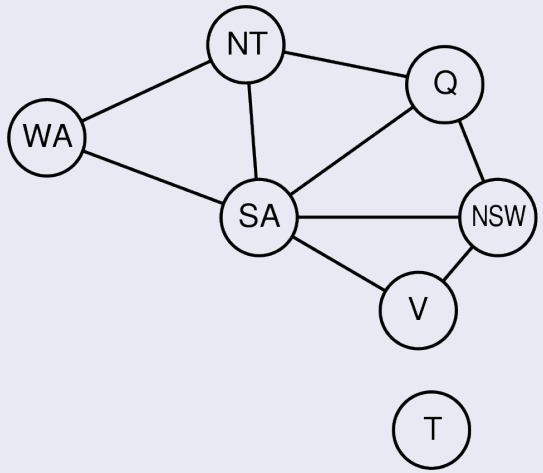


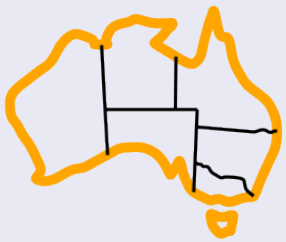
Fonte: Adaptado dos slides de AIMA. Russell & Norvig.

Backtracking Search – Heurísticas

Ordem das variáveis – Heurística do grau

- Ex:
  - SA tem o maior número de arestas $\rightarrow$ maior número de restrições
  - Começamos por ele



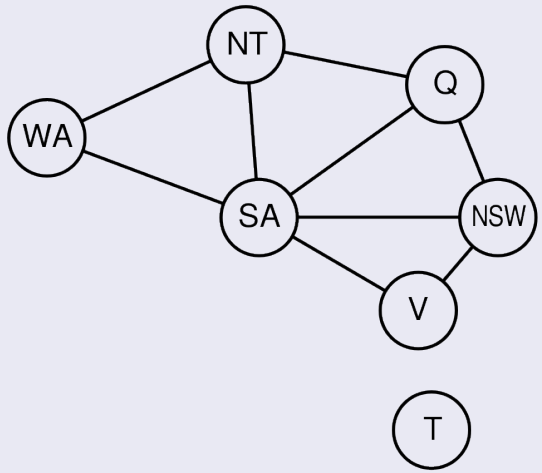


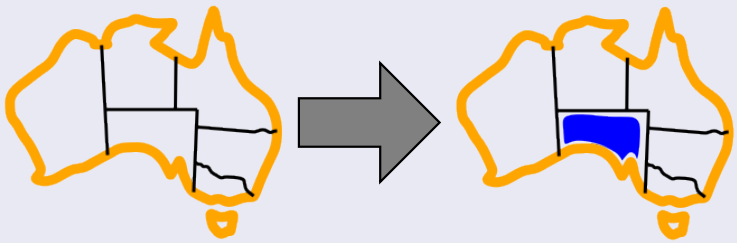
Fonte: Adaptado dos slides de AIMA. Russell & Norvig.

Backtracking Search – Heurísticas

Ordem das variáveis – Heurística do grau

- Ex:
  - SA tem o maior número de arestas $\rightarrow$ maior número de restrições
  - Começamos por ele



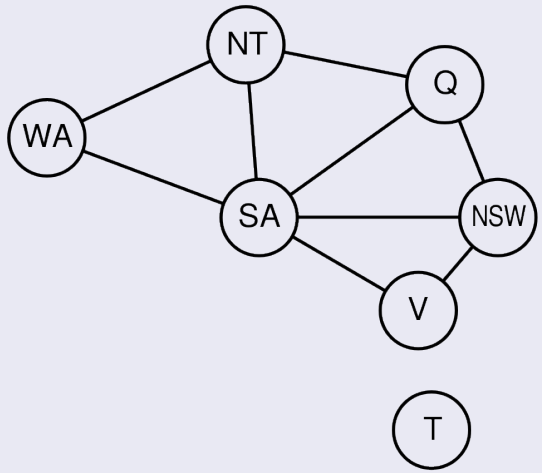


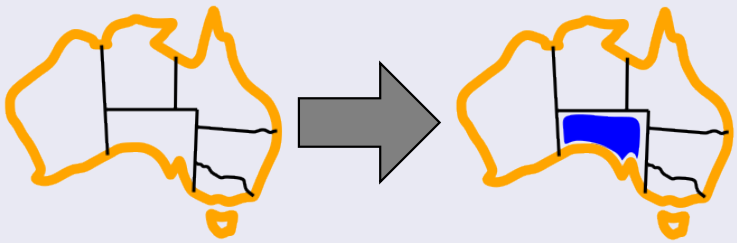
Fonte: Adaptado dos slides de AIMA. Russell & Norvig.

Backtracking Search – Heurísticas

Ordem das variáveis – Heurística do grau

- Ex:
  - SA tem o maior número de arestas $\rightarrow$ maior número de restrições
  - Começamos por ele
  - Sobram NT, Q e NSW empatados no número de valores, e com 2 restrições (com variáveis restantes) cada



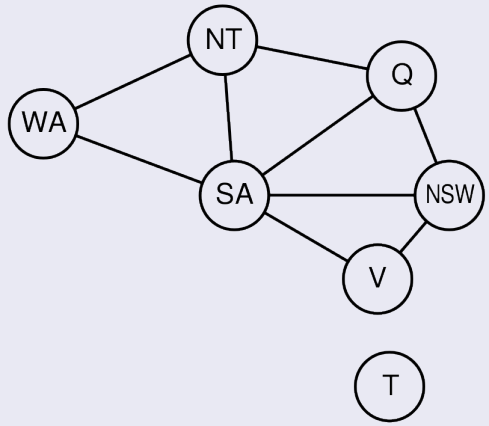


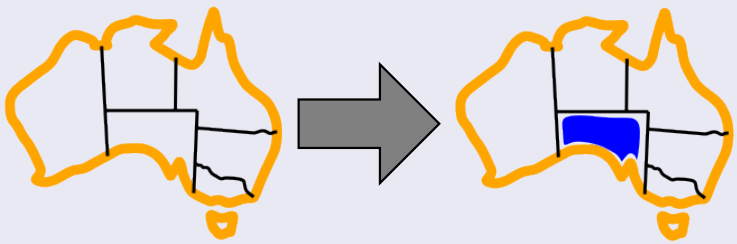
Fonte: Adaptado dos slides de AIMA. Russell & Norvig.

Backtracking Search – Heurísticas

Ordem das variáveis – Heurística do grau

- Escolhemos um deles



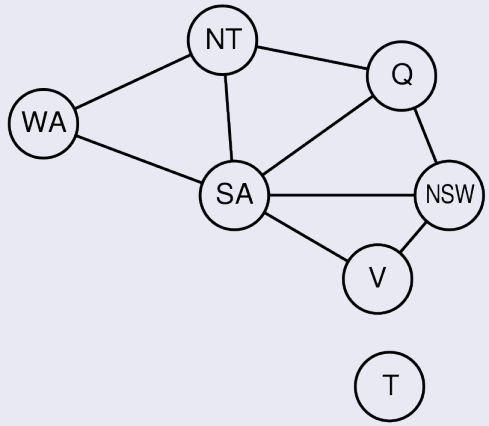


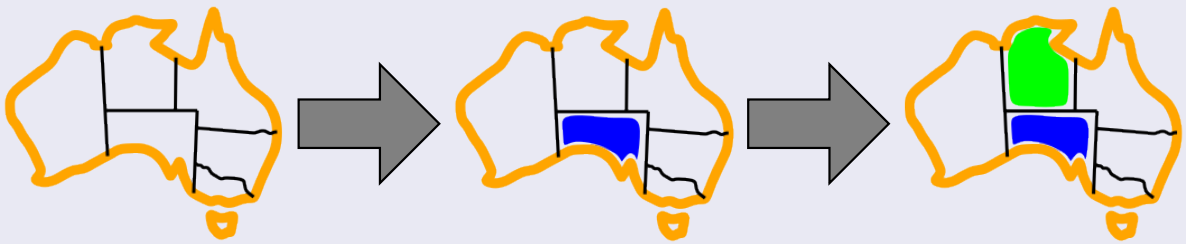
Fonte: Adaptado dos slides de AIMA. Russell & Norvig.

Backtracking Search – Heurísticas

Ordem das variáveis – Heurística do grau

- Escolhemos um deles



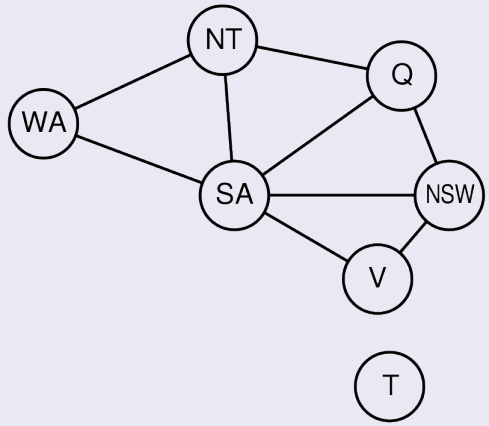


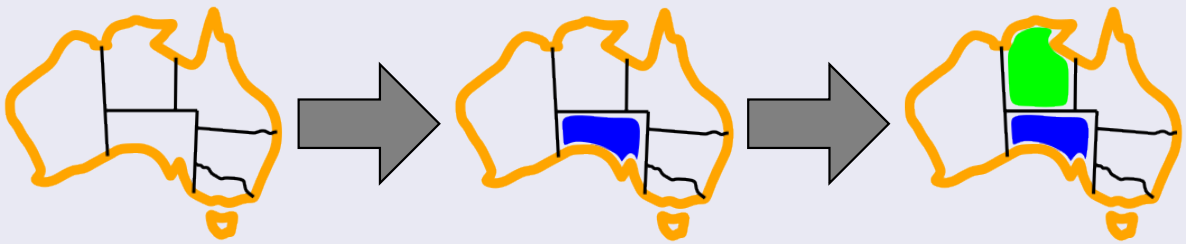
Fonte: Adaptado dos slides de AIMA. Russell & Norvig.

Backtracking Search – Heurísticas

Ordem das variáveis – Heurística do grau

- Escolhemos um deles
- WA e Q têm apenas 1 opção de cor (empate), mas Q tem grau maior com as variáveis restantes (WA=0 e Q=1)



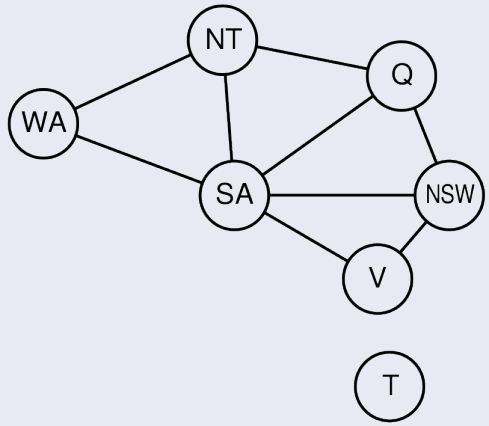


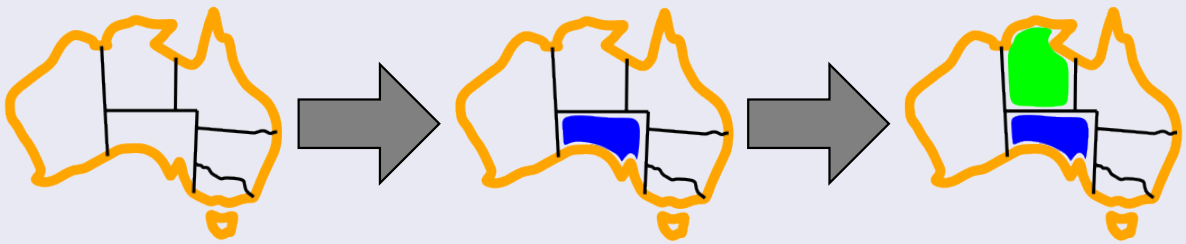
Fonte: Adaptado dos slides de AIMA. Russell & Norvig.

Backtracking Search – Heurísticas

Ordem das variáveis – Heurística do grau

- Escolhemos um deles
- WA e Q têm apenas 1 opção de cor (empate), mas Q tem grau maior com as variáveis restantes (WA=0 e Q=1)
- Escolhemos este então



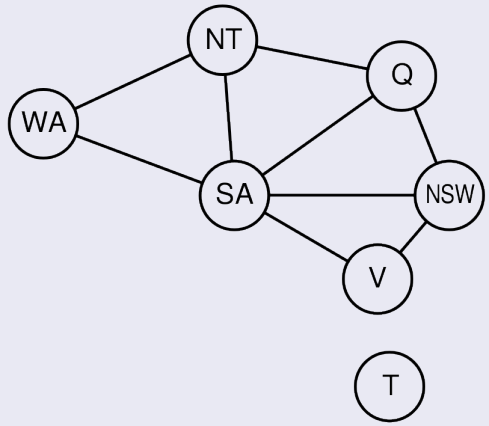


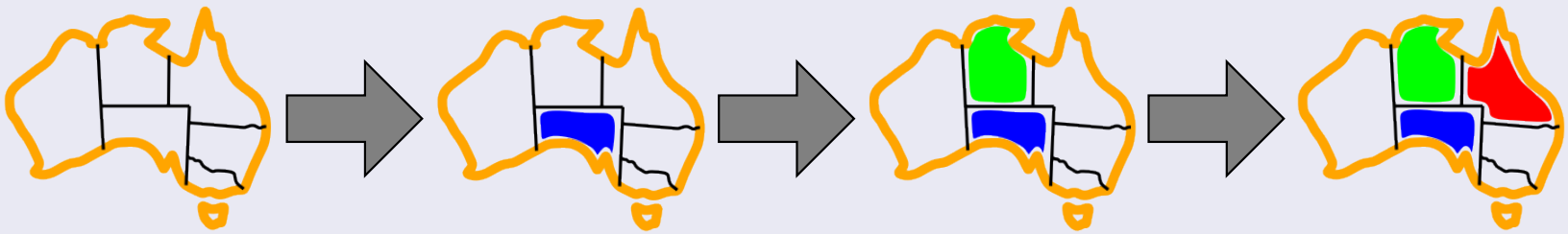
Fonte: Adaptado dos slides de AIMA. Russell & Norvig.

Backtracking Search – Heurísticas

Ordem das variáveis – Heurística do grau

- Escolhemos um deles
- WA e Q têm apenas 1 opção de cor (empate), mas Q tem grau maior com as variáveis restantes (WA=0 e Q=1)
- Escolhemos este então



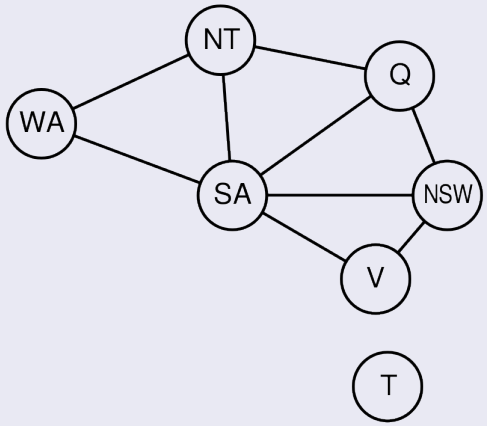


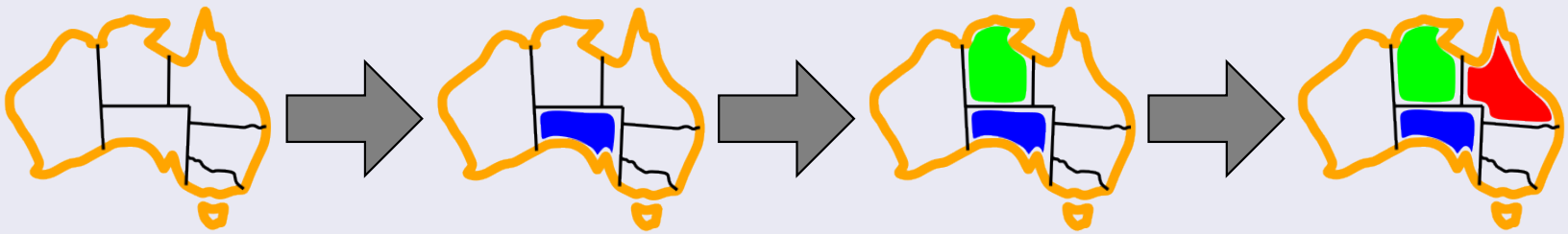
Fonte: Adaptado dos slides de AIMA. Russell & Norvig.

Backtracking Search – Heurísticas

Ordem das variáveis – Heurística do grau

- Escolhemos um deles
- WA e Q têm apenas 1 opção de cor (empate), mas Q tem grau maior com as variáveis restantes (WA=0 e Q=1)
- Escolhemos este então
- E assim seguimos...





Fonte: Adaptado dos slides de AIMA. Russell & Norvig.

Backtracking Search – Heurísticas

Ordem das variáveis

- Valores mínimos restantes é geralmente um guia mais poderoso

Backtracking Search – Heurísticas

Ordem das variáveis

- Valores mínimos restantes é geralmente um guia mais poderoso
  - Mas a heurística do grau pode ser bastante útil para desempates

Backtracking Search – Heurísticas

Ordem dos valores – Valor Menos Restritivo

- Uma vez escolhida a próxima variável, resta ainda decidir qual valor atribuir a ela primeiro

Backtracking Search – Heurísticas

Ordem dos valores – Valor Menos Restritivo

- Uma vez escolhida a próxima variável, resta ainda decidir qual valor atribuir a ela primeiro
- Heurística do **Valor Menos Restritivo**

Backtracking Search – Heurísticas

Ordem dos valores – Valor Menos Restritivo

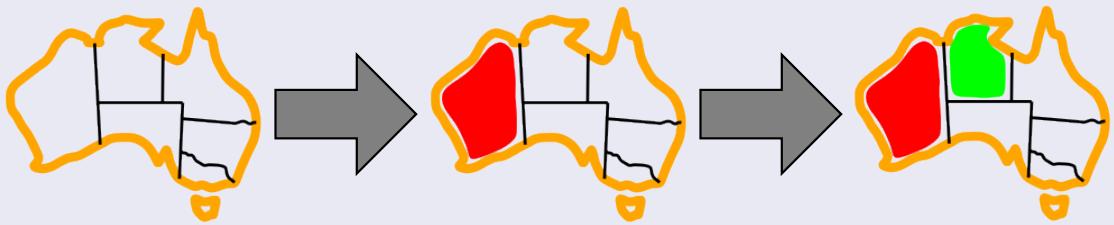
- Uma vez escolhida a próxima variável, resta ainda decidir qual valor atribuir a ela primeiro
- Heurística do **Valor Menos Restritivo**
  - Escolha o valor que elimina o menor número de escolhas para as variáveis vizinhas no grafo de restrições

Backtracking Search – Heurísticas

Ordem dos valores – Valor Menos Restritivo

- Uma vez escolhida a próxima variável, resta ainda decidir qual valor atribuir a ela primeiro
- Heurística do **Valor Menos Restritivo**
  - Escolha o valor que elimina o menor número de escolhas para as variáveis vizinhas no grafo de restrições

Ex: suponha que já escolhemos $WA = vm$ e $NT = vd$, e vamos escolher um valor para Q:



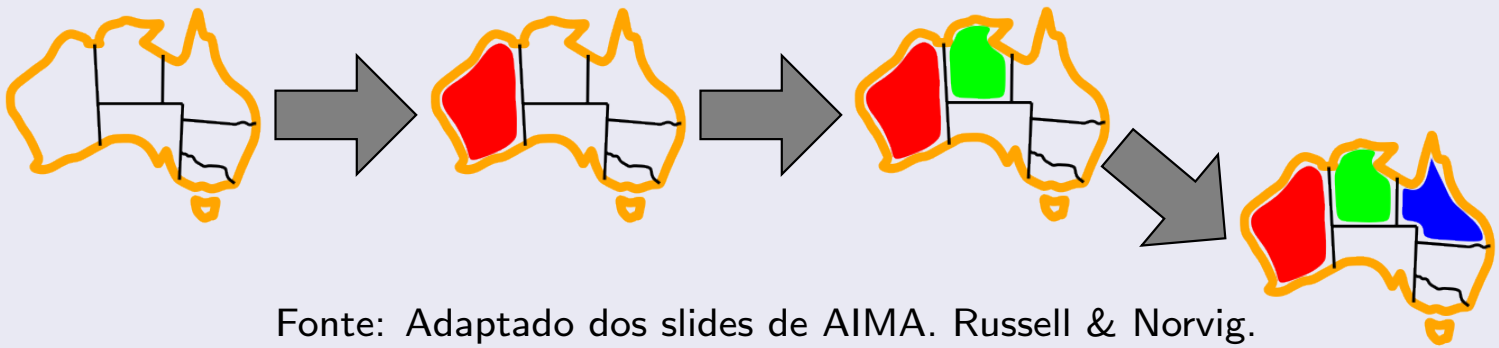
Fonte: Adaptado dos slides de AIMA. Russell & Norvig.

Backtracking Search – Heurísticas

Ordem dos valores – Valor Menos Restritivo

- Uma vez escolhida a próxima variável, resta ainda decidir qual valor atribuir a ela primeiro
- Heurística do **Valor Menos Restritivo**
  - Escolha o valor que elimina o menor número de escolhas para as variáveis vizinhas no grafo de restrições

Escolher Q = *az* deixa 0 valor para *SA*



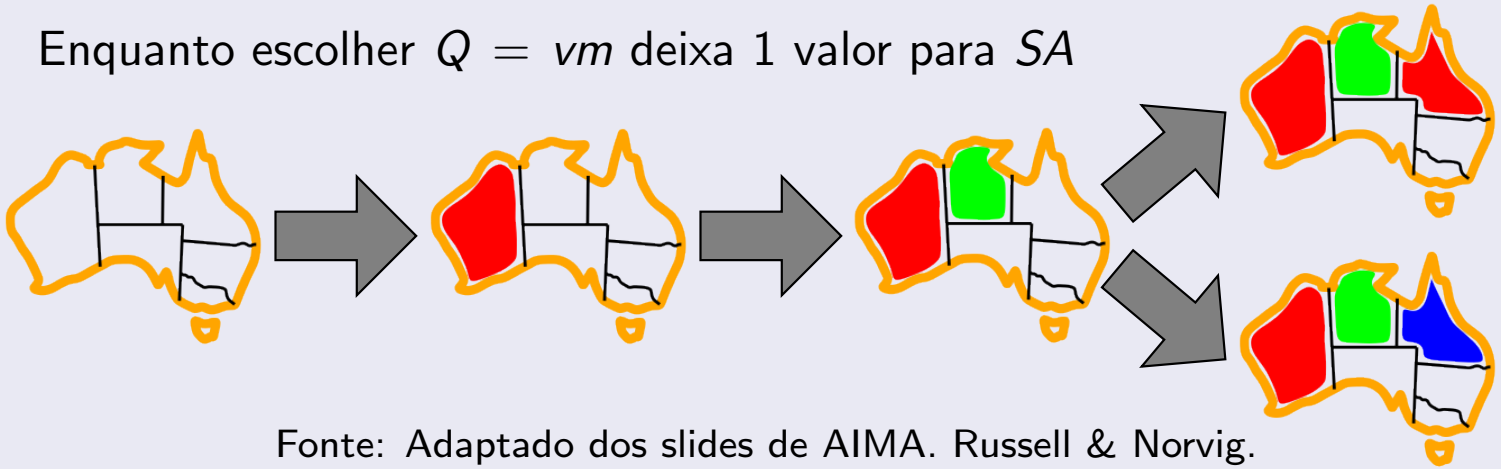
Fonte: Adaptado dos slides de AIMA. Russell & Norvig.

Backtracking Search – Heurísticas

Ordem dos valores – Valor Menos Restritivo

- Uma vez escolhida a próxima variável, resta ainda decidir qual valor atribuir a ela primeiro
- Heurística do **Valor Menos Restritivo**
  - Escolha o valor que elimina o menor número de escolhas para as variáveis vizinhas no grafo de restrições

Enquanto escolher $Q = vm$ deixa 1 valor para SA



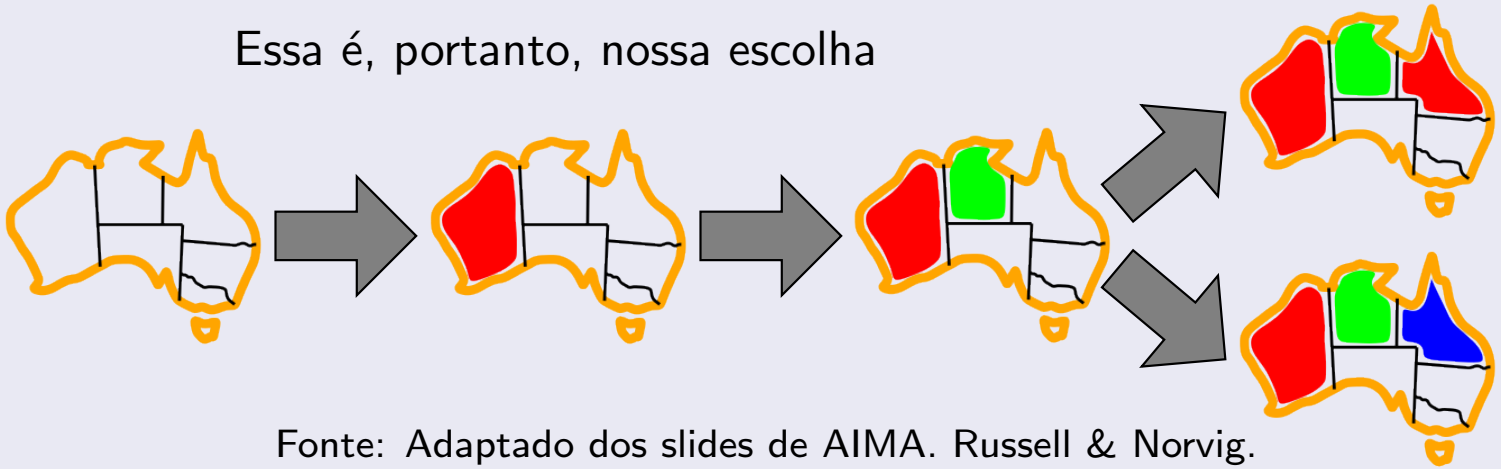
Fonte: Adaptado dos slides de AIMA. Russell & Norvig.

Backtracking Search – Heurísticas

Ordem dos valores – Valor Menos Restritivo

- Uma vez escolhida a próxima variável, resta ainda decidir qual valor atribuir a ela primeiro
- Heurística do **Valor Menos Restritivo**
  - Escolha o valor que elimina o menor número de escolhas para as variáveis vizinhas no grafo de restrições

Essa é, portanto, nossa escolha



Fonte: Adaptado dos slides de AIMA. Russell & Norvig.

Backtracking Search – Heurísticas

Ordem dos valores – Valor Menos Restritivo

- A heurística tenta assim dar a maior flexibilidade possível para atribuições subsequentes

Backtracking Search – Heurísticas

Ordem dos valores – Valor Menos Restritivo

- A heurística tenta assim dar a maior flexibilidade possível para atribuições subsequentes
- Em geral, usar Busca Retroativa combinando essas 3 heurísticas acelera consideravelmente a obtenção do resultado

CSP – Constraint Satisfaction Problems

Usando a Estrutura do Problema

- Podemos usar a estrutura do problema para auxiliar na busca de soluções

CSP – Constraint Satisfaction Problems

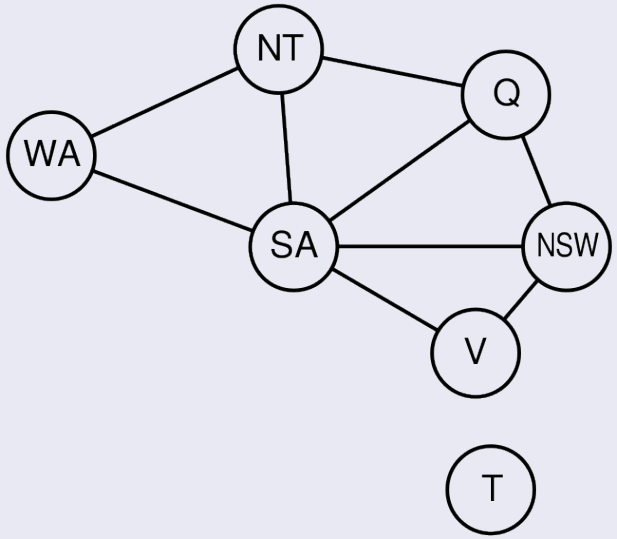
Usando a Estrutura do Problema

- Podemos usar a estrutura do problema para auxiliar na busca de soluções
  - Usando o grafo de restrição

CSP – Constraint Satisfaction Problems

Usando a Estrutura do Problema

- Podemos usar a estrutura do problema para auxiliar na busca de soluções
  - Usando o grafo de restrição
- Ex:

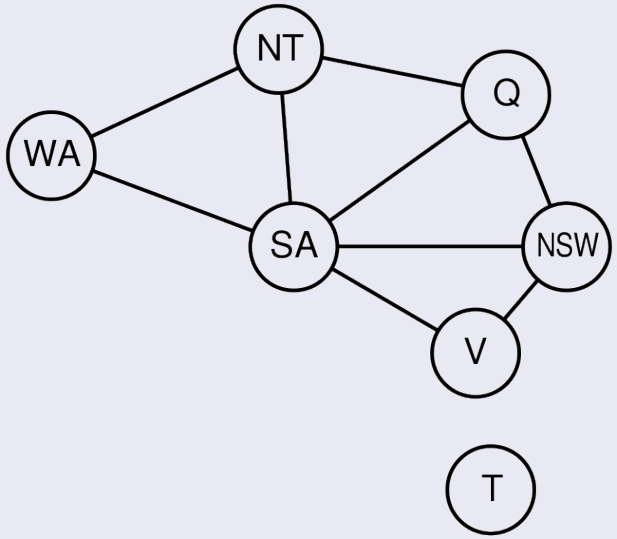


Fonte: Slides de AIMA. Russell & Norvig.

CSP – Constraint Satisfaction Problems

Usando a Estrutura do Problema

- Podemos usar a estrutura do problema para auxiliar na busca de soluções
  - Usando o grafo de restrição
- Ex:
  - Note que T não está conectado

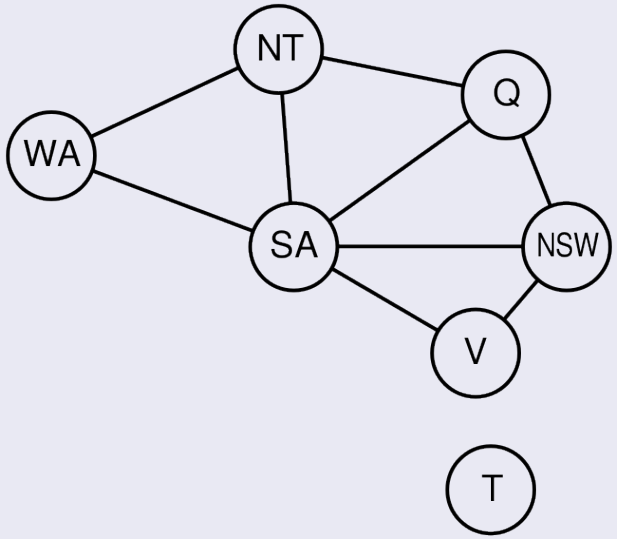


Fonte: Slides de AIMA. Russell & Norvig.

CSP – Constraint Satisfaction Problems

Usando a Estrutura do Problema

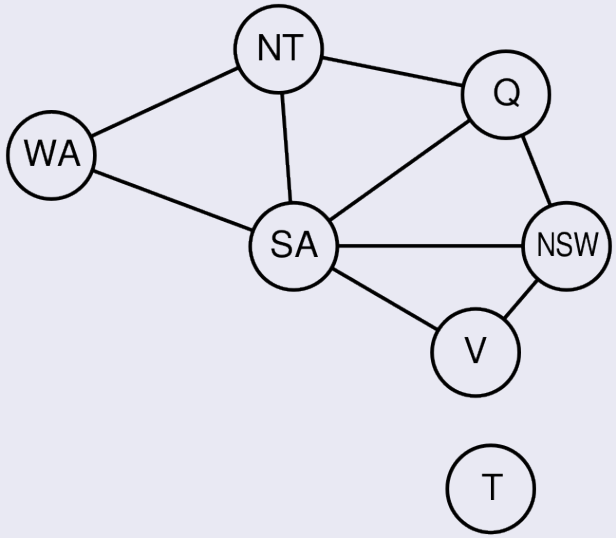
- Podemos usar a estrutura do problema para auxiliar na busca de soluções
  - Usando o grafo de restrição
- Ex:
  - Note que T não está conectado
  - A Tasmânia e a ilha principal são subproblemas independentes, com menos variáveis em cada um



Fonte: Slides de AIMA. Russell & Norvig.

Usando a Estrutura do Problema

- Podemos, então, começar buscando componentes conexos no grafo

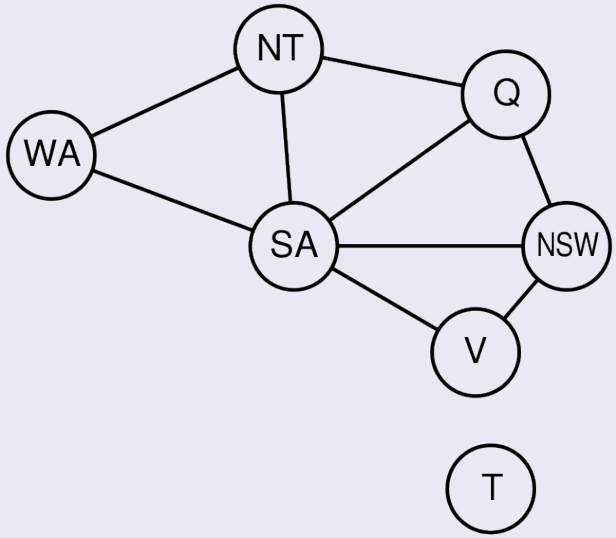


Fonte: Slides de AIMA. Russell & Norvig.

CSP – Constraint Satisfaction Problems

Usando a Estrutura do Problema

- Podemos, então, começar buscando componentes conexos no grafo
  - Identificando, assim, regiões independentes

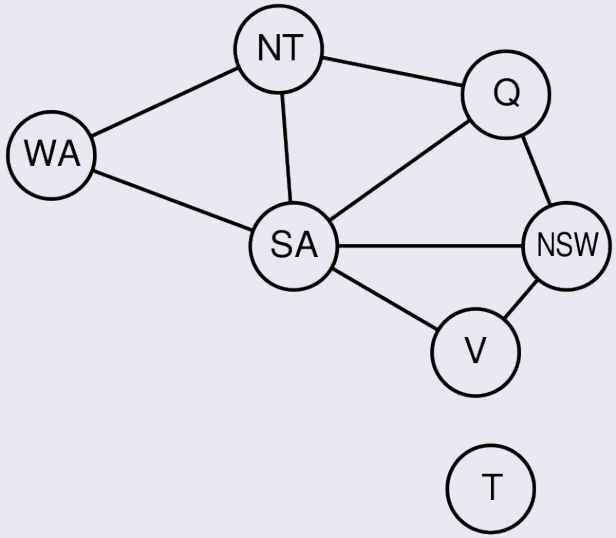


Fonte: Slides de AIMA. Russell & Norvig.

CSP – Constraint Satisfaction Problems

Usando a Estrutura do Problema

- Podemos, então, começar buscando componentes conexos no grafo
  - Identificando, assim, regiões independentes
- Cada componente corresponderá a um subproblema

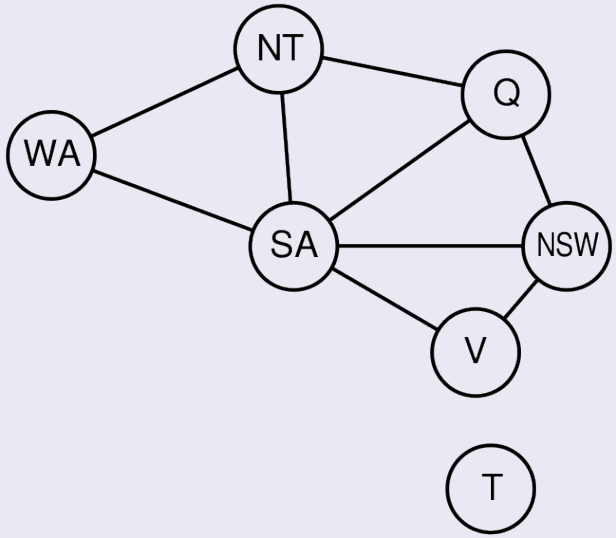


Fonte: Slides de AIMA. Russell & Norvig.

CSP – Constraint Satisfaction Problems

Usando a Estrutura do Problema

- Podemos, então, começar buscando componentes conexos no grafo
  - Identificando, assim, regiões independentes
- Cada componente corresponderá a um subproblema
  - Com menos variáveis para busca e totalmente paralelizável



Fonte: Slides de AIMA. Russell & Norvig.

CSP – Constraint Satisfaction Problems

Propagação de Restrições

- Até agora, consideramos as restrições em uma variável somente quando a selecionamos

CSP – Constraint Satisfaction Problems

Propagação de Restrições

- Até agora, consideramos as restrições em uma variável somente quando a selecionamos
  - Uma melhoria seria ver os efeitos das restrições em uma variável nas demais (que ainda permanecem sem valor)

CSP – Constraint Satisfaction Problems

Propagação de Restrições

- Até agora, consideramos as restrições em uma variável somente quando a selecionamos
  - Uma melhoria seria ver os efeitos das restrições em uma variável nas demais (que ainda permanecem sem valor)
- Técnica chamada de **Propagação de restrições**

CSP – Constraint Satisfaction Problems

Propagação de Restrições

- Até agora, consideramos as restrições em uma variável somente quando a selecionamos
  - Uma melhoria seria ver os efeitos das restrições em uma variável nas demais (que ainda permanecem sem valor)
- Técnica chamada de **Propagação de restrições**
  - Usamos as restrições para reduzir o número de valores legais para uma variável

CSP – Constraint Satisfaction Problems

Propagação de Restrições

- Até agora, consideramos as restrições em uma variável somente quando a selecionamos
  - Uma melhoria seria ver os efeitos das restrições em uma variável nas demais (que ainda permanecem sem valor)
- Técnica chamada de **Propagação de restrições**
  - Usamos as restrições para reduzir o número de valores legais para uma variável
  - E então propagamos essa redução pelo grafo

CSP – Constraint Satisfaction Problems

Propagação de Restrições

- Até agora, consideramos as restrições em uma variável somente quando a selecionamos
  - Uma melhoria seria ver os efeitos das restrições em uma variável nas demais (que ainda permanecem sem valor)
- Técnica chamada de **Propagação de restrições**
  - Usamos as restrições para reduzir o número de valores legais para uma variável
  - E então propagamos essa redução pelo grafo
  - Métodos: *Forward checking* e Consistência de arestas

CSP – Propagação de Restrições

Backtracking Search com *Forward Checking*

CSP – Propagação de Restrições

Backtracking Search com *Forward Checking*

- Quando atribuímos um valor a uma variável X:

CSP – Propagação de Restrições

Backtracking Search com *Forward Checking*

- Quando atribuímos um valor a uma variável X:
  - Olhamos cada variável Y, ainda sem valor, conectada a X por uma restrição

CSP – Propagação de Restrições

Backtracking Search com *Forward Checking*

- Quando atribuímos um valor a uma variável X:
  - Olhamos cada variável Y, ainda sem valor, conectada a X por uma restrição
  - Removemos do domínio de Y qualquer valor inconsistente com o valor escolhido para X

CSP – Propagação de Restrições

Backtracking Search com *Forward Checking*

- Quando atribuímos um valor a uma variável X:
  - Olhamos cada variável Y, ainda sem valor, conectada a X por uma restrição
  - Removemos do domínio de Y qualquer valor inconsistente com o valor escolhido para X
- A busca termina quando alguma variável fica sem valor possível $\rightarrow$ fazemos o *backtracking*

CSP – Propagação de Restrições

Backtracking Search com *Forward Checking*

- Quando atribuímos um valor a uma variável X:
  - Olhamos cada variável Y, ainda sem valor, conectada a X por uma restrição
  - Removemos do domínio de Y qualquer valor inconsistente com o valor escolhido para X
- A busca termina quando alguma variável fica sem valor possível $\rightarrow$ fazemos o *backtracking*
  - A ideia é então manter registro dos valores legais restantes para variáveis ainda sem valor

CSP – Propagação de Restrições

Backtracking Search com *Forward Checking*

- Quando atribuímos um valor a uma variável X:
  - Olhamos cada variável Y, ainda sem valor, conectada a X por uma restrição
  - Removemos do domínio de Y qualquer valor inconsistente com o valor escolhido para X
- A busca termina quando alguma variável fica sem valor possível $\rightarrow$ fazemos o *backtracking*
  - A ideia é então manter registro dos valores legais restantes para variáveis ainda sem valor
  - E terminar a busca quando uma variável estiver sem valor

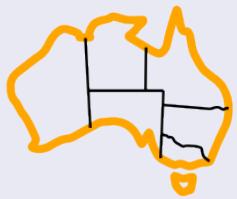
B.S. com *Forward Checking*

Exemplo (sem heurísticas)

B.S. com *Forward Checking*

Exemplo (sem heurísticas)

- Inicialmente, o domínio de todas as variáveis tem todos os possíveis valores

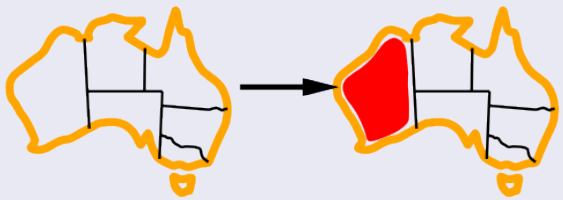


Fonte: Slides de AIMA. Russell & Norvig.

B.S. com *Forward Checking*

Exemplo (sem heurísticas)

- Uma vez definida a cor para *WA*, essa cor é removida do domínio de *NT* e *SA*

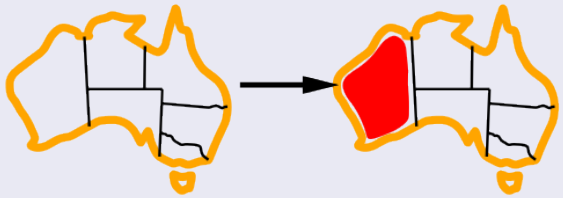


Fonte: Slides de AIMA. Russell & Norvig.

B.S. com *Forward Checking*

Exemplo (sem heurísticas)

- O método propaga então informação das variáveis com valor para as sem valor definido

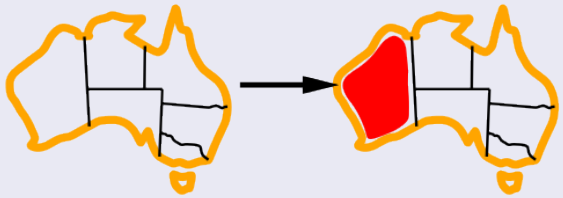


Fonte: Slides de AIMA. Russell & Norvig.

B.S. com *Forward Checking*

Exemplo (sem heurísticas)

- O método propaga então informação das variáveis com valor para as sem valor definido
  - Suponha que o próximo escolhido seja Q

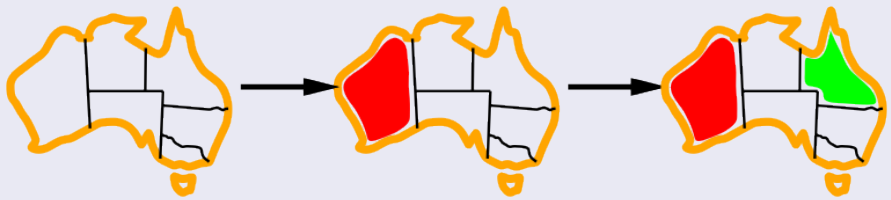


Fonte: Slides de AIMA. Russell & Norvig.

B.S. com *Forward Checking*

Exemplo (sem heurísticas)

- Atribuído verde a Q, essa cor é removida de seus vizinhos ainda sem cor (NT, SA e NSW)

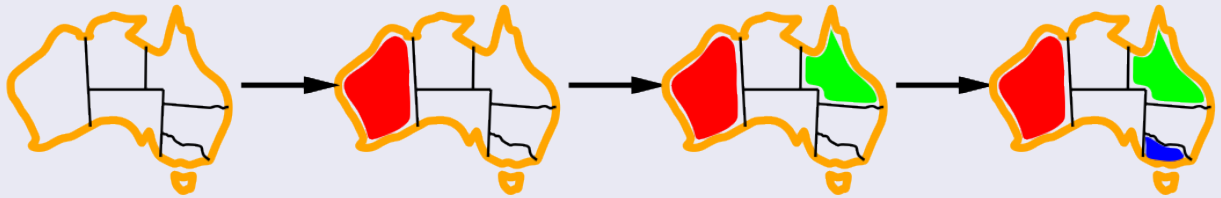


Fonte: Slides de AIMA. Russell & Norvig.

B.S. com *Forward Checking*

Exemplo (sem heurísticas)

- Escolhido $V = azul$, o método pára, pois SA ficou com o domínio vazio (não há valor consistente)

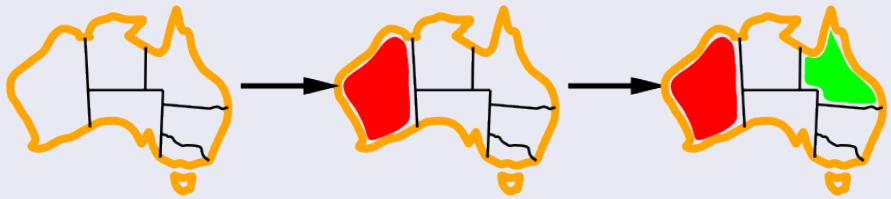


Fonte: Slides de AIMA. Russell & Norvig.

B.S. com *Forward Checking*

Exemplo (sem heurísticas)

- O algoritmo deve fazer então um *backtracking* para a decisão anterior e atribuir outra cor a V

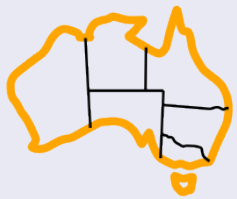


Fonte: Slides de AIMA. Russell & Norvig.

B.S. com *Forward Checking*

Forward Checking + Valores Restantes Mínimos

- Para muitos problemas, a busca é mais efetiva se combinarmos a heurística dos valores restantes mínimos com *forward checking*

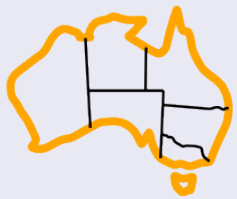


Fonte: Slides de AIMA. Russell & Norvig.

B.S. com *Forward Checking*

Forward Checking + Valores Restantes Mínimos

- Para muitos problemas, a busca é mais efetiva se combinarmos a heurística dos valores restantes mínimos com *forward checking*
  - Considere, por exemplo, quando fizemos $WA = vermelho$

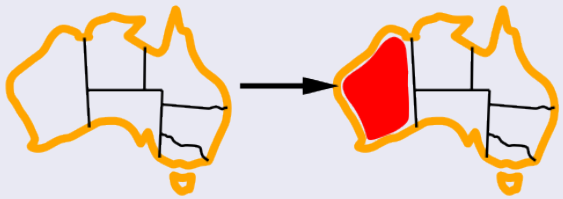


Fonte: Slides de AIMA. Russell & Norvig.

B.S. com *Forward Checking*

Forward Checking + Valores Restantes Mínimos

- Para muitos problemas, a busca é mais efetiva se combinarmos a heurística dos valores restantes mínimos com *forward checking*
  - Considere, por exemplo, quando fizemos *WA* = *vermelho*

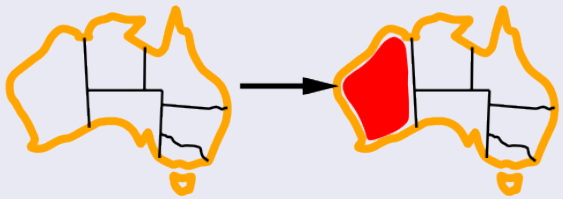


Fonte: Slides de AIMA. Russell & Norvig.

B.S. com *Forward Checking*

Forward Checking + Valores Restantes Mínimos

- Nesse ponto, pela heurística dos VRMs, teríamos que escolher entre *NT* e *SA*

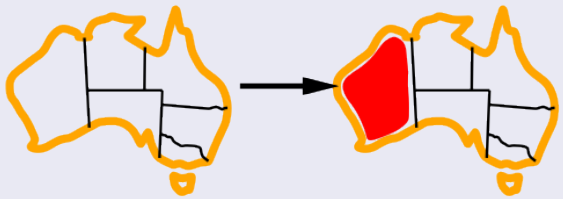


Fonte: Slides de AIMA. Russell & Norvig.

B.S. com *Forward Checking*

Forward Checking + Valores Restantes Mínimos

- Nesse ponto, pela heurística dos VRMs, teríamos que escolher entre *NT* e *SA*
  - E o algoritmo não falharia

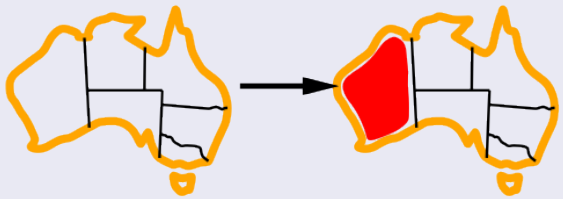


Fonte: Slides de AIMA. Russell & Norvig.

B.S. com *Forward Checking*

Forward Checking + Valores Restantes Mínimos

- Temos então que *forward checking* é, de fato, um modo eficiente de calcular incrementalmente a informação que a VRM precisa para fazer seu trabalho



Fonte: Slides de AIMA. Russell & Norvig.

Referências

- Russell, S.; Norvig P. (2010): Artificial Intelligence: A Modern Approach. Prentice Hall. 3a ed.
  - Slides do livro: `http://aima.eecs.berkeley.edu/slides-pdf/`
- `http://ocw.mit.edu/OcwWeb/Electrical-Engineering-and-Computer-Science/6-034Spring-2005/LectureNotes/index.htm`