

Проблеми кои треба да задоволат услови*

Проблеми кои задоволуваат услови - CSP

- Просторот на состојби има стандардна, структурна репрезентација
- Алгоритмите за пребарување ги користат карактеристиките на просторот
- Наместо хеуристики за специфични проблеми, се користат алгоритми со општа хеуристика

Дефиниција

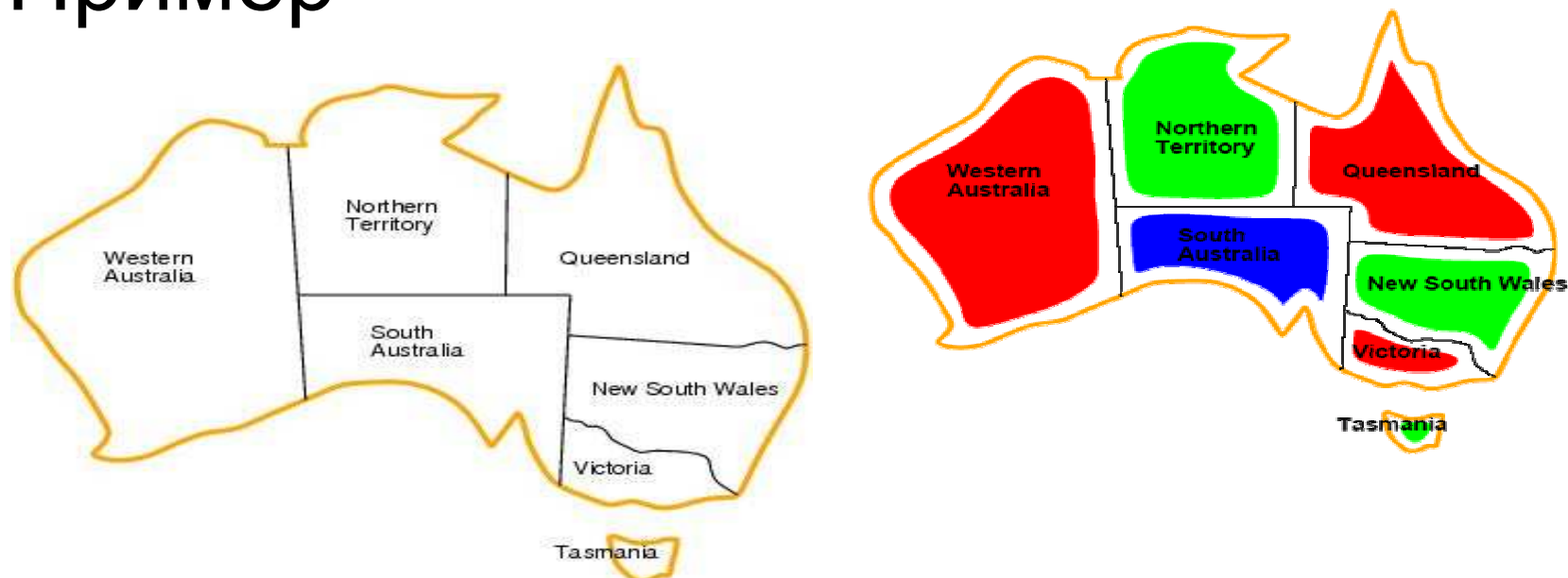
Проблемот се дефинира со:

- множество на променливи V_1, V_2, \dots, V_n
 - секоја променлива дефинирана со домен на дозволени вредности $D_{V1}, D_{V2}, \dots, D_{Vn}$
- множество на услови C_1, C_2, \dots, C_m
 - секој се однесува на подмножество променливи и ги дефинира дозволените вредности на подмножеството
- Просторот на состојби е дефиниран со доделување на вредности (assignment of values) на дел или сите променливи

Својства

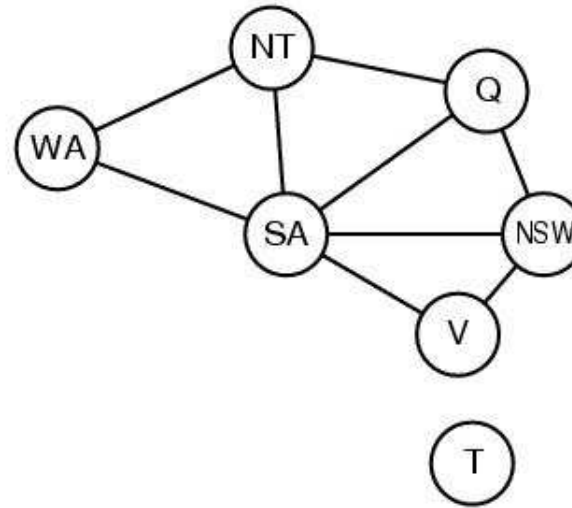
- Доделувањето вредности кое ги задоволува сите зададени услови е **КОНЗИСТЕНТНО**
- Доделувањето вредности во кое се спомнува секоја променлива е **комплетно**
- Решение за даден проблем е **комплетно** ако ги задоволува сите услови

Пример



- Да се обојат државите во Австралија со црвена, зелена или сина боја, но да го задоволуваат условот соседните држави да не бидат со иста боја
- Домен на секоја променлива е $\{red, green, blue\}$, условите како нееднаквости $WA \neq NT$
- Постојат повеќе решенија
 $\{WA = red, NT = green, Q = red, NSW = green, V = red, SA = blue, T = red\}$.

Пример



- Граф на услови
- Јазлите се променливи
- Врските се бинарни услови
- Пребарување на граф

Друга дефиниција на проблеми кои задоволуваат услови

1. Почетна состојба т.е. ниту една променлива нема доделена вредност
2. Функција на наследник: доделување на недефинирана променлива со вредност која не е во спротивност со претходно доделените променливи
3. Тест за целна состојба: дали тековните доделувања се комплетни
4. Цена на патот: константна за секој чекор

Типови на проблеми

- Дискретни променливи со конечни домени (боење на карта, проблем на 8 кралици)
- Дискретни променливи со бесконечни домени (временска распределба / план на работа).
- Континуирани домени (распоред на часови)
- Типови на услови: унарни, бинарни, од повисок степен (криптоаритметика)

Криптоаритметика

- Криптоаритметика е тип на математичка загатка во која цифрите се заменуваат со букви или други симболи.

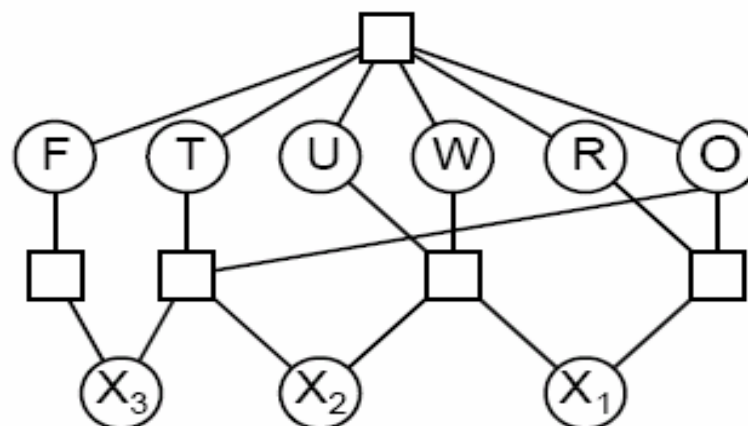
$$\begin{array}{r} \text{S E N D} \\ 9 \ 5 \ 6 \ 7 \\ + \text{M O R E} \\ 1 \ 0 \ 8 \ 5 \\ \hline \text{M O N E Y} \\ 1 \ 0 \ 6 \ 5 \ 2 \end{array}$$

Комутативност

- Проблемите кои задоволуваат услови се комутативни
 - Редоследот на било која низа акции не влијае врз резултатот
- пример: $[WA = \text{red then } NT = \text{green}]$ е исто како и $[NT = \text{green then } WA = \text{red}]$
- Сите алгоритми за пребарување може да генерираат наследник земајќи ги предвид само вредностите на една единствена променлива во секој јазел

Криптоаритметика

$$\begin{array}{r} \text{ T W O} \\ + \text{ T W O} \\ \hline \text{ F O U R} \end{array}$$



Variables: $F T U W R O X_1 X_2 X_3$

Domains: $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$

Constraints

$alldiff(F, T, U, W, R, O)$

$O + O = R + 10 \cdot X_1$, etc.

Насоки за решавање

Правила:

- Секоја буква или симбол може да преставува само една цифра;
- Кога сите букви ќе се заменат со нивниот пар од множеството цифри, резултантната аритметичка операција треба да биде задоволена;
- Нумеричка основа е вообичаено 10;
- Бројките не смеат да почнуваат со 0;
- Мора да постои само ЕДНО ЕДИНСТВЕНО решение на проблемот.

Да се најде “0”

- Почнуваме со 0, 1 и 9.
- Стратегија за наоѓање на 0 или 9 е да се најдат колони со две или повеќе исти букви.
- Колоните $A + A = A$ и $B + A = B$ посочуваат дека $A=0$.

$$\begin{array}{r} * * * A \\ + * * * A \\ \hline * * * A \end{array}$$

$$\begin{array}{r} * * * B \\ + * * * A \\ \hline * * * B \end{array}$$

Да се најде “0”/”9”

- Но, ако истите колони се наоѓаат на друго место, можно е $A=0$, но и $A=9$.
- Зависи дали имало пренос на единица од претходната колона
- Всушност, 9 е како 0 секогаш кога имаме пренос на 1.

$$\begin{array}{r} \begin{array}{ccccc} & * & A & * & * \\ + & * & A & * & * \\ \hline & * & A & * & * \end{array} & \begin{array}{ccccc} & * & B & * & * \\ + & * & A & * & * \\ \hline & * & B & * & * \end{array} \end{array}$$

Да се најде “1”

- Најлевата цифра, ако е единствена
- Во првата загатка M може да биде само 1, бидејќи е пренос од претходната колона.

$$\begin{array}{r} \text{S E N D} \\ + \text{M O R E} \\ \hline \text{M O N E Y} \end{array}$$

Генерирај и тестирај

1. Наведи ги сите цифри кои не се идентификувани;
2. Избери буква за да ја почнеш процедурата (генерирај);
3. Следи циклус на чекори и тестирања:
Од листата на неискористени цифри избери една и додели ја за замена на избраната буква (избриши ја од листата); продолжи да ги заменуваш другите букви;
Ако не е конзистентно решението, повторно оди на чекор 3, ако е конзистентно, решението е пронајдено.

Пример

T A K E	T 9 K 1	$C1 + K + K = T + 10$
A	9	$C3 + T + C = K$
+ C A K E	+ C 9 K 1	
-----	-----	$\Rightarrow C1=1 \text{ \& } C3=1$
K A T E	K 9 T 1	

T 9 8 1	C Y C L E	A	E	K	C	T
9	=====					
+ 0 9 8 1	#1	9	1	8	[0]	

8 9 T 1						

Пример

	T 9 7 1	CYCLE	A	E	K	C	T
	9	=====					
+	1 9 7 1	#1	9	1	8	[0]	
	-----	#2	9	1	7	[1]	
	7 9 T 1						

	3 9 6 1	CYCLE	A	E	K	C	T
	9	=====					
+	2 9 6 1	#1	9	1	8	[0]	
	-----	#2	9	1	7	[1]	
	6 9 3 1	#3	9	1	6	2	3

Пребарување со враќање наназад backtracking

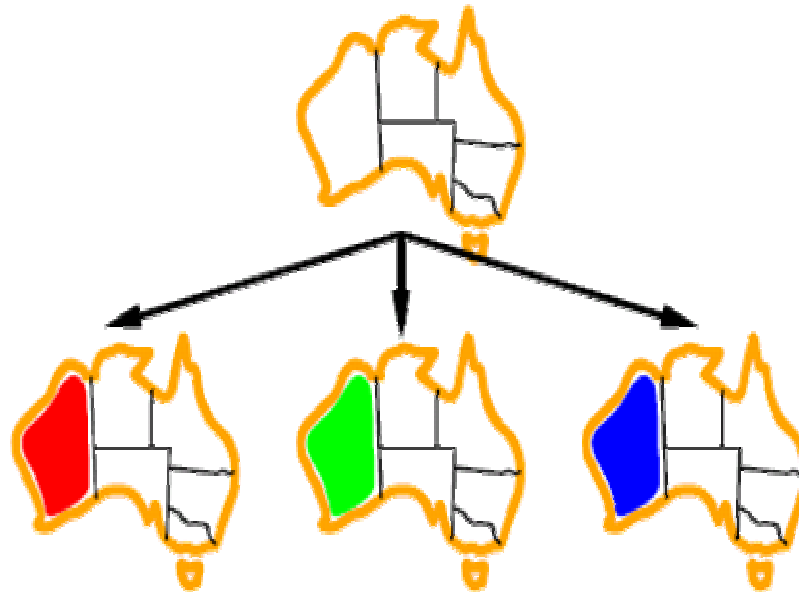
- Слично со пребарување по прва длабочина
- Одбира вредности за секоја променлива поединчно и се враќа наназад кога променливата нема веќе дозволени вредности за доделување
- Неинформирано пребарување
 - нема добри перформанси

Пребарување со враќање наназад backtracking

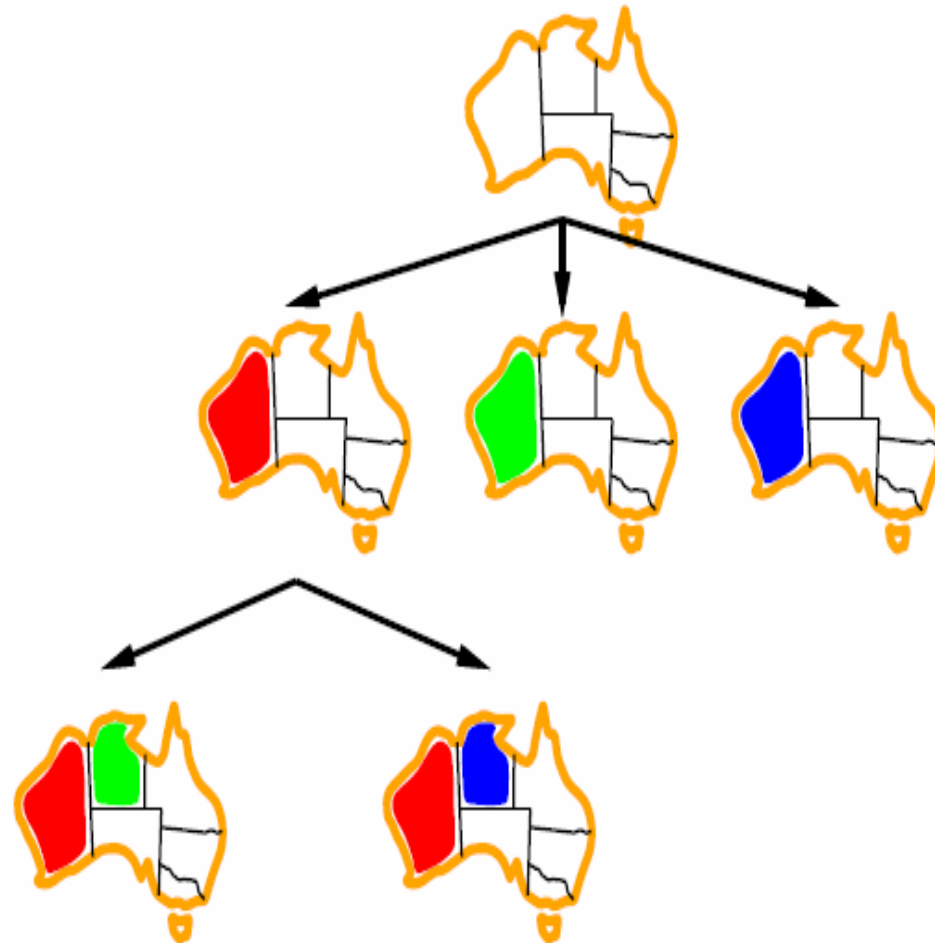
```
function BACKTRACKING-SEARCH(csp) return a solution or failure  
  return RECURSIVE-BACKTRACKING({} , csp)
```

```
function RECURSIVE-BACKTRACKING(assignment, csp) return a solution  
  or failure  
  if assignment is complete then return assignment  
  var ← SELECT-UNASSIGNED-  
  VARIABLE(VARIABLES[csp],assignment,csp)  
  for each value in ORDER-DOMAIN-VALUES(var, assignment, csp) do  
    if value is consistent with assignment according to  
    CONSTRAINTS[csp] then  
      add {var=value} to assignment  
      result ← RRECURSIVE-BACKTRACKING(assignment, csp)  
      if result ≠ failure then return result  
      remove {var=value} from assignment  
  return failure
```

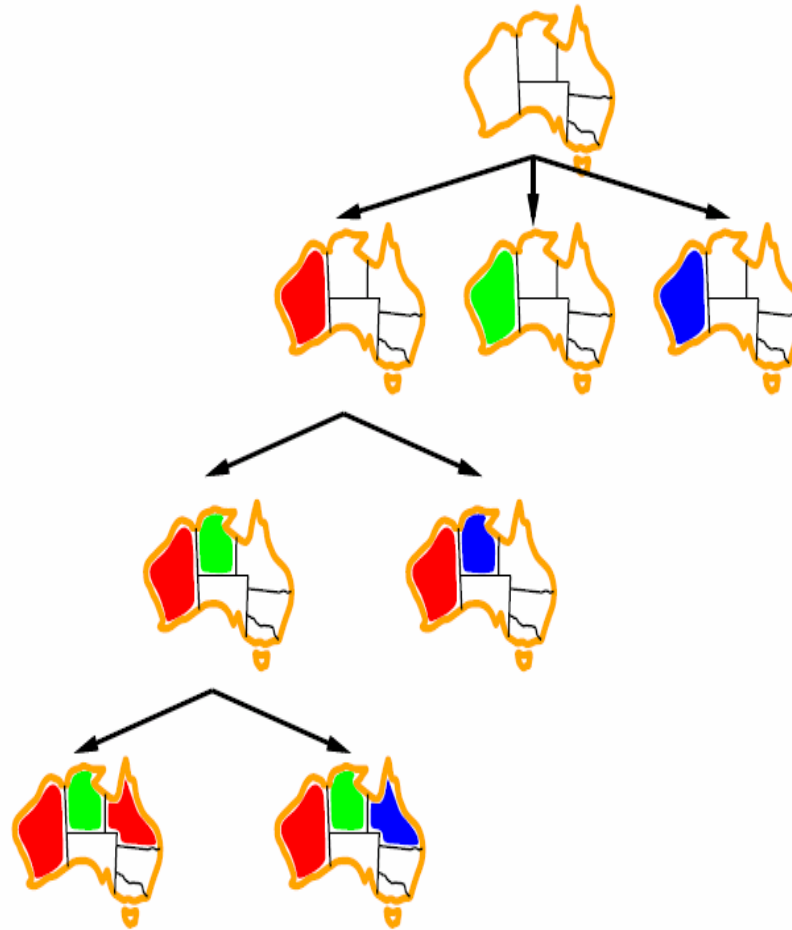
Пребарување со враќање назад: Пример



Пребарување со враќање назад: Пример



Пребарување со враќање назад: Пример



Споредба на алгоритми за пребарување за различни проблеми

Problem	Backtracking	BT+MRV	Forward Checking	FC+MRV	Min-Conflicts
USA	(> 1,000K)	(> 1,000K)	2K	60	64
<i>n</i> -Queens	(> 40,000K)	13,500K	(> 40,000K)	817K	4K
Zebra	3,859K	1K	35K	0.5K	2K
Random 1	415K	3K	26K	2K	
Random 2	942K	27K	77K	15K	

Просечен број на проверки за конзистентност
Заградите означуваат дека не е пронајдено решение

Подобрувања на ефикасноста на алгоритмите

- Воведување на хеуристика
- Општи методи ја подобруваат брзината
 - На кои променливи треба прво да им се додели вредност?
 - По кој редослед треба да се испробаат вредностите?
 - Дали може навреме да се идентификуваат неуспешните обиди?
 - Дали може да се искористи структурата на проблемот?

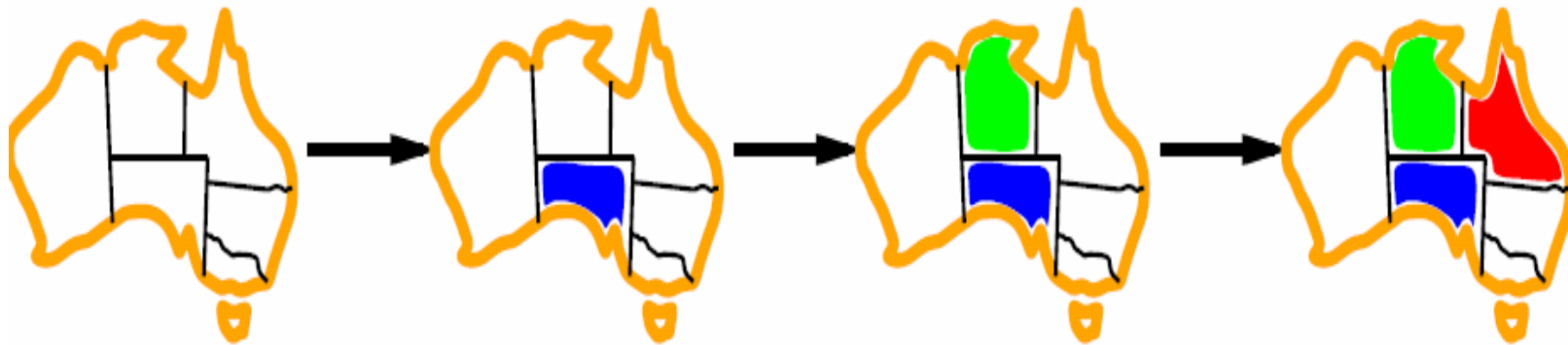
Хеуристика 1



$var \leftarrow \text{SELECT-UNASSIGNED-VARIABLE}(\text{VARIABLES}[csp], \text{assignment}, csp)$

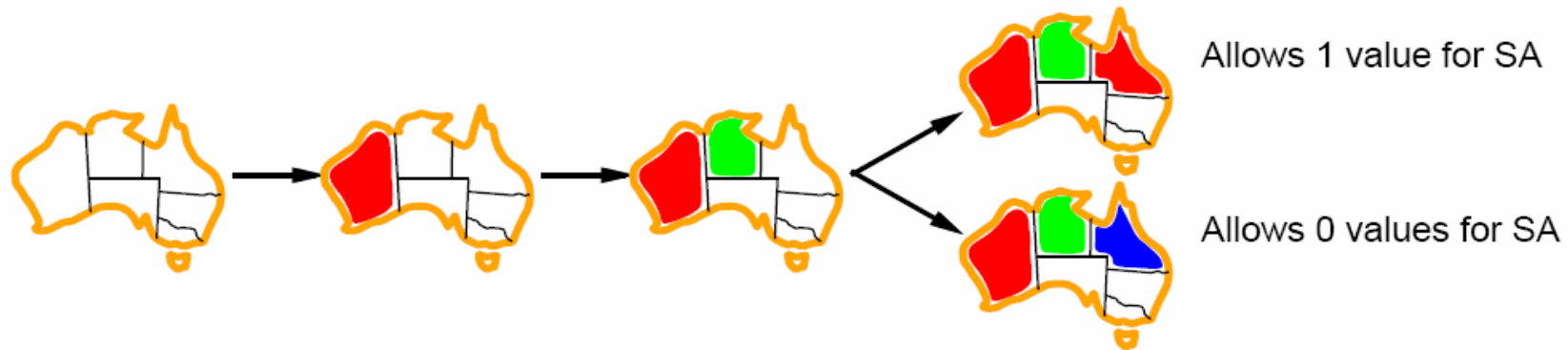
- Избери променлива која има најмал број на дозволени вредности

Хеуристика 2



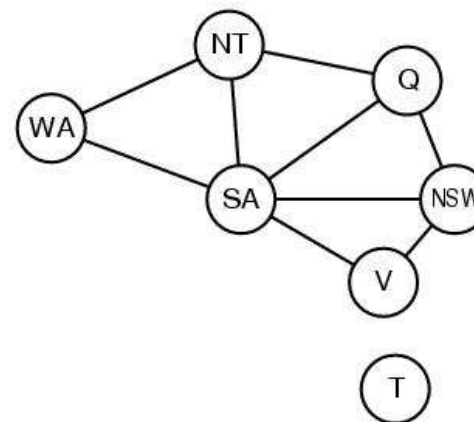
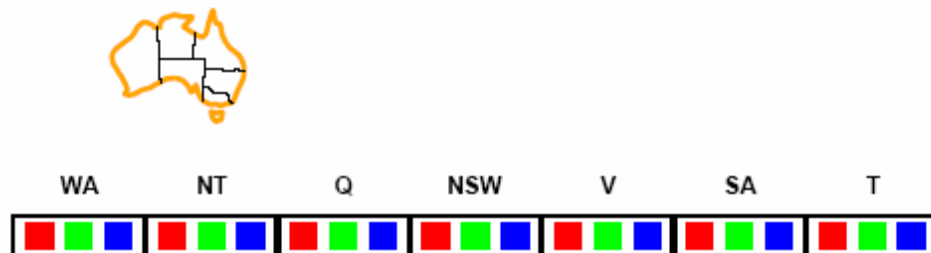
- Избери променлива која учествува во најголемиот број на услови за променливите на кои не им се доделени вредности
- Помага кога ќе настане застој

Хеуристика 3



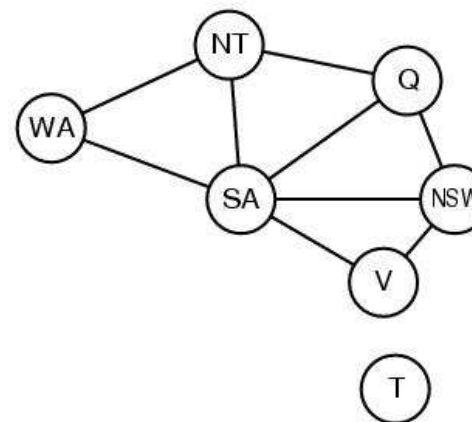
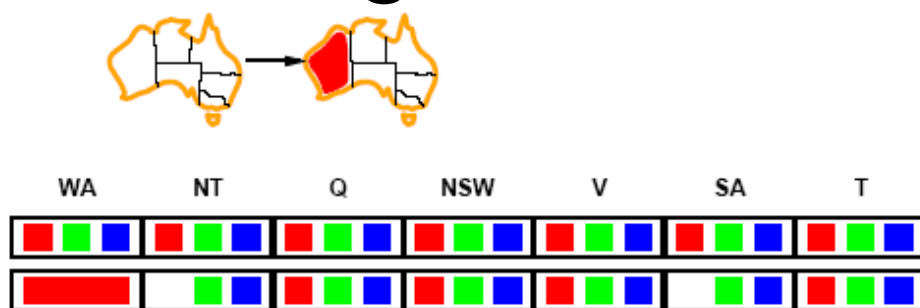
- Избери променлива која има најмал број на дозволени вредности (според условите)

Проверка напред Forward checking



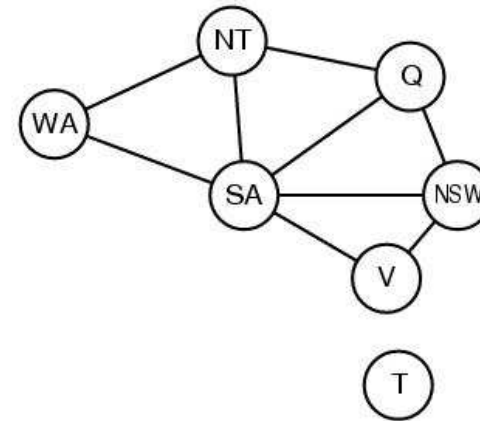
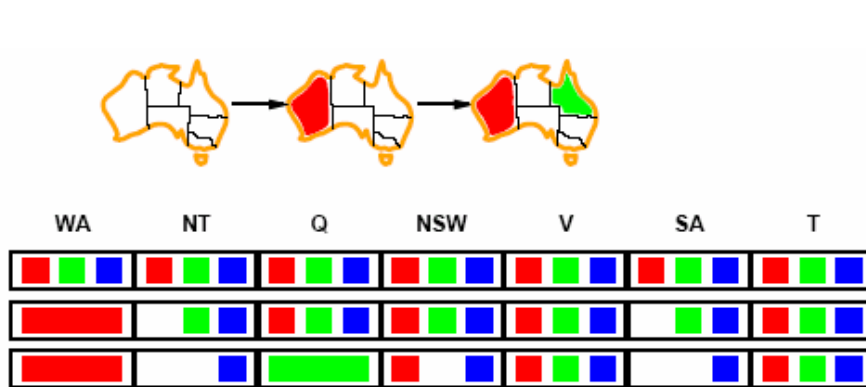
- Може ли да предвидиме неуспех и да го избегнеме?
- Пристап: Води сметка (чувај записи) за дозволените вредности кои преостануваат за променливите на кои не им се доделени вредности
- Прекини го пребарувањето кога некоја променлива ќе остане без дозволени вредности

Проверка напред Forward checking



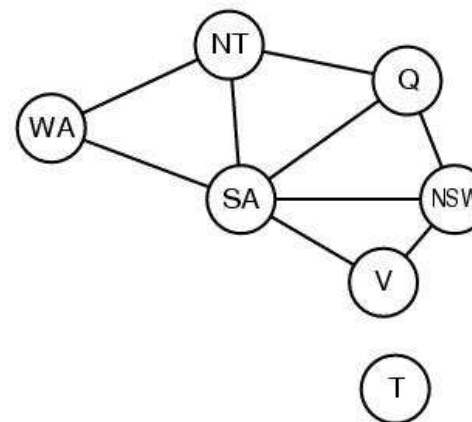
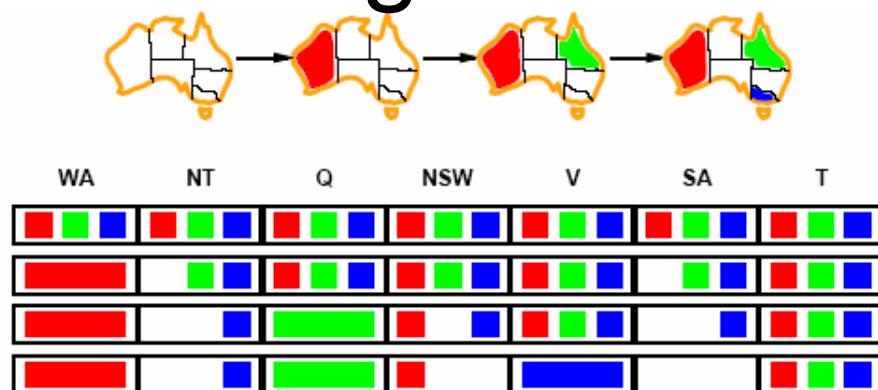
- Доделување $\{WA = red\}$
- Кои се последиците за останатите променливи врзани со услов со WA пр.
 - NT не може да биде црвено,
 - SA не може да биде црвено

Проверка напред Forward checking



- Доделување $\{Q = \text{green}\}$
- Кои се последиците за останатите променливи врзани со услов со WA
 - ☐ NT не може да биде зелено
 - ☐ NSW не може да биде зелено
 - ☐ SA не може да биде зелено,
- Хеуристиката 1 веднаш би ги избрала NT или SA за следни

Проверка нанапред Forward checking



- Доделување V = сино
- Кои се последиците за останатите променливи врзани со услов со WA
 - NSW не може да биде сино
 - SA е празно
- Проверката нанапред открива неконзистентност и затоа се враќа наназад

Albert Einstein: Who Owns The Fish? Part 1

- There are five houses in a row and in five different colours.
- In each house lives a person from a different country.
- Each person drinks a certain drink, plays a certain sport, and keeps a certain pet.
- No two people drink the same drink, play the same sport, or keep the same pet.

Albert Einstein: Who Owns The Fish? Part 2

- The Brit lives in a red house
- The Swede keeps dogs
- The Dane drinks tea
- The green house is on the left of the white house
- The green house owner drinks coffee
- The person who plays polo rears birds
- The owner of the yellow house plays hockey
- The man living in the house right in the centre drinks milk

Albert Einstein: Who Owns The Fish? Part 3

- The Norwegian lives in the first house
- The man who plays baseball lives next to the man who keeps cats
- The man who keeps horses lives next to the one who plays hockey
- The man who plays billiards drinks beer
- The German plays soccer
- The Norwegian lives next to the blue house
- The man who plays baseball has a neighbour who drinks water.



Прашања?