FACULTY OF SCIENCES

DEPARTMENT OF APPLIED MATHEMATICS, COMPUTER SCIENCE AND STATISTICS
RESEARCH GROUP DATA MINING AND MODELING FOR BIOMEDICINE

ZOEKHEURISTIEKEN – 2

prof. dr. Yvan Saeys (yvan.saeys@ugent.be)
Bureau: Sterre S9, 1e verdiep (naast leslokaal 1.1)

GHENT UNIVERSITY

OVERZICHT

- **Constraint satisfaction problems (CSPs)**
  - Backtracking
  - Heuristieken
  - CSPs en boomstructuren
  - Local search

GHENT UNIVERSITY

ZOEKEN VOOR CSPS

- Identificatie
  - Toewijzing van waarden aan variabelen…
    - terwijl we voldoen aan bepaalde restricties
- Oplossing vinden is belangrijk, niet hoe we er geraken (pad)
  - Toestand: variabelen X_i met waarden uit “domein” D_i
  - Doeltest: verzameling restricties die toelaatbare toewijzingen aan de variabelen definieren
- Voor de meeste formuleringen
  - Alle paden hebben dezelfde diepte

CSP FORMULERING

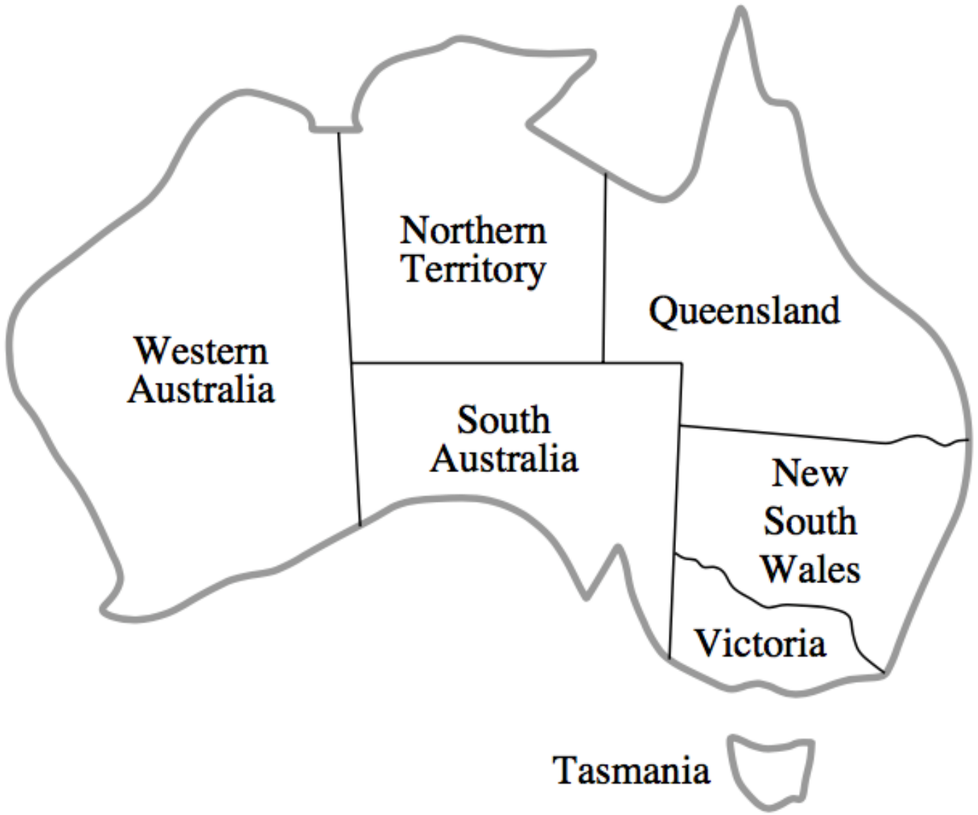
- Gegeven:
  - Een verzameling variabelen $\{X_1, X_2, \ldots, X_n\}$
  - Een verzameling restricties $\{C_1, C_2, \ldots, C_m\}$
- Elke variabele X_i heeft een niet-ledig ***domein*** D_i van mogelijke ***waarden***
- Elke restrictie C_i heeft betrekking op een deelverzameling van variabelen en specificeert toelaatbare combinaties van waarden voor deze deelverzameling

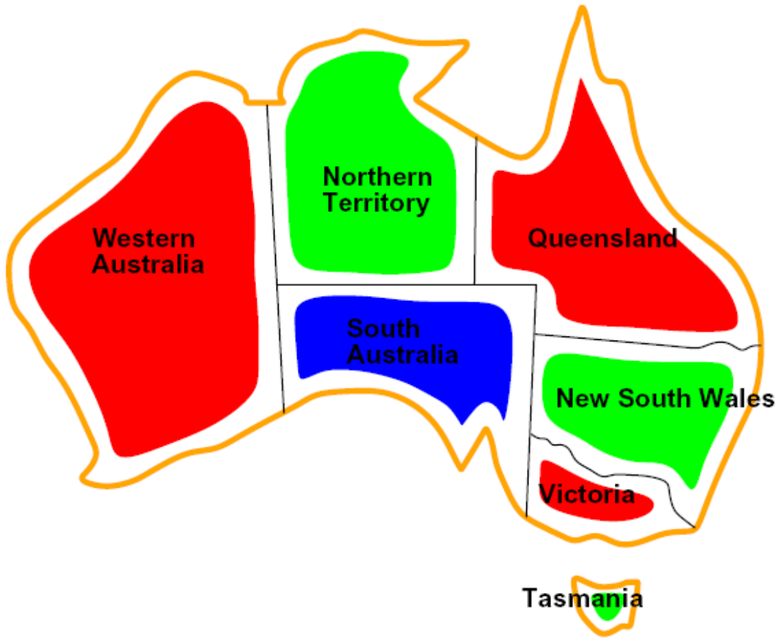
CSP FORMULERING

- Een toestand is gedefinieerd door een **toewijzing** van waarden aan sommige of alle variabelen, $\{X_i = v_i, X_j = v_j , \ldots\}$
- Een toewijzing die geen restricties schendt is een **consistente** toewijzing
- Sommige CSP formuleringen maken gebruik van een **objectief-functie** die gemaximaliseerd wordt

GHENT UNIVERSITY

VOORBEELD: GRAAF-KLEUREN

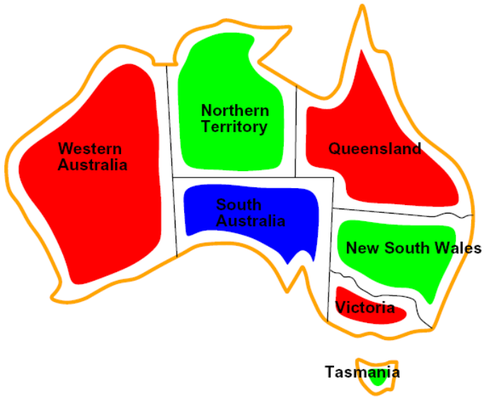




GHENT UNIVERSITY

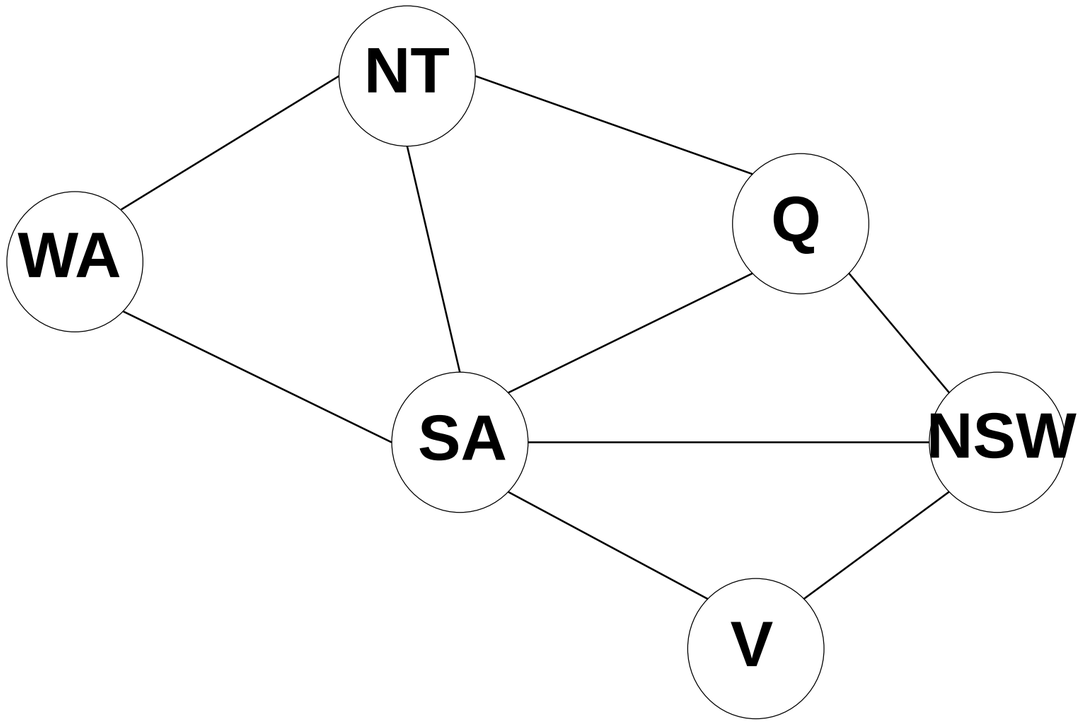
GRAAF-KLEUREN: FORMULERING

- Variabelen:
  V = {WA, NT , Q, NSW , V , SA, T}
- Domeinen:
  D = {R,G,B}
- Restricties:
  Naburige gebieden moeten verschillende kleuren hebben: WA≠NT,…
- Oplossingen:
  Bvb
  {WA=R,NT=G,Q=R,NSW=G,V=R,SA=B,T=G}

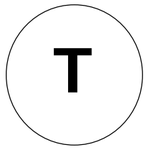


GHENT UNIVERSITY

CONSTRAINT GRAPH



Tasmania is een onafhankelijk sub-probleem



GHENT UNIVERSITY

VOORBEELD: N KONINGINNEN-PROBLEEM

- Variabelen:

  Q_k

- Domeinen:

  {1,2,…,N}

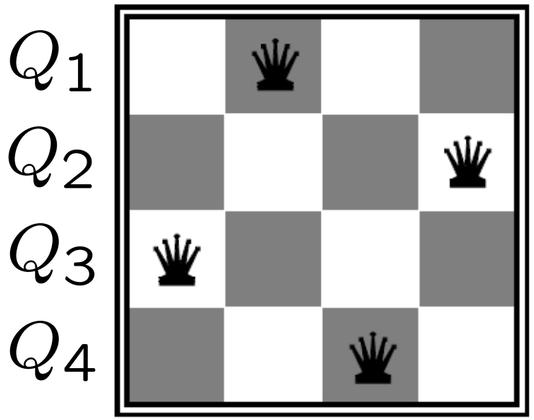
- Restricties:

  Impliciet:

  $$\forall i, j \;\; \text{non-threatening}(Q_i, Q_j)$$

  Expliciet

  $$(Q_1, Q_2) \in \{(1,3), (1,4), \ldots\}$$
  $$\ldots$$



GHENT UNIVERSITY

CSPS MET DISCRETE VARIABELEN EN EINDIGE DOMEINEN

GHENT UNIVERSITY

TYPES VAN RESTRICTIES

- Unaire restricties
  - Beperken de waarde van één variabele
  - Voorbeeld: SA ≠ groen
- Binaire restricties
  - Gaan over twee variabelen
  - Voorbeeld: SA ≠ NSW
- Hogere orde restricties
  - >2 variabelen
  - Voorbeeld: Alldiff (F, T, U, W, R, O)
- „Soft constraints“
  - „Rood is beter dan groen“
  - Vaak voorgesteld adhv een kostfunctie

VOORBEELD: CRYPTARITMETISCHE PUZZEL

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

Restricties

$$O + O = R + 10 \cdot X_1$$

$$X_1 + W + W = U + 10 \cdot X_2$$

$$X_2 + T + T = O + 10 \cdot X_3$$

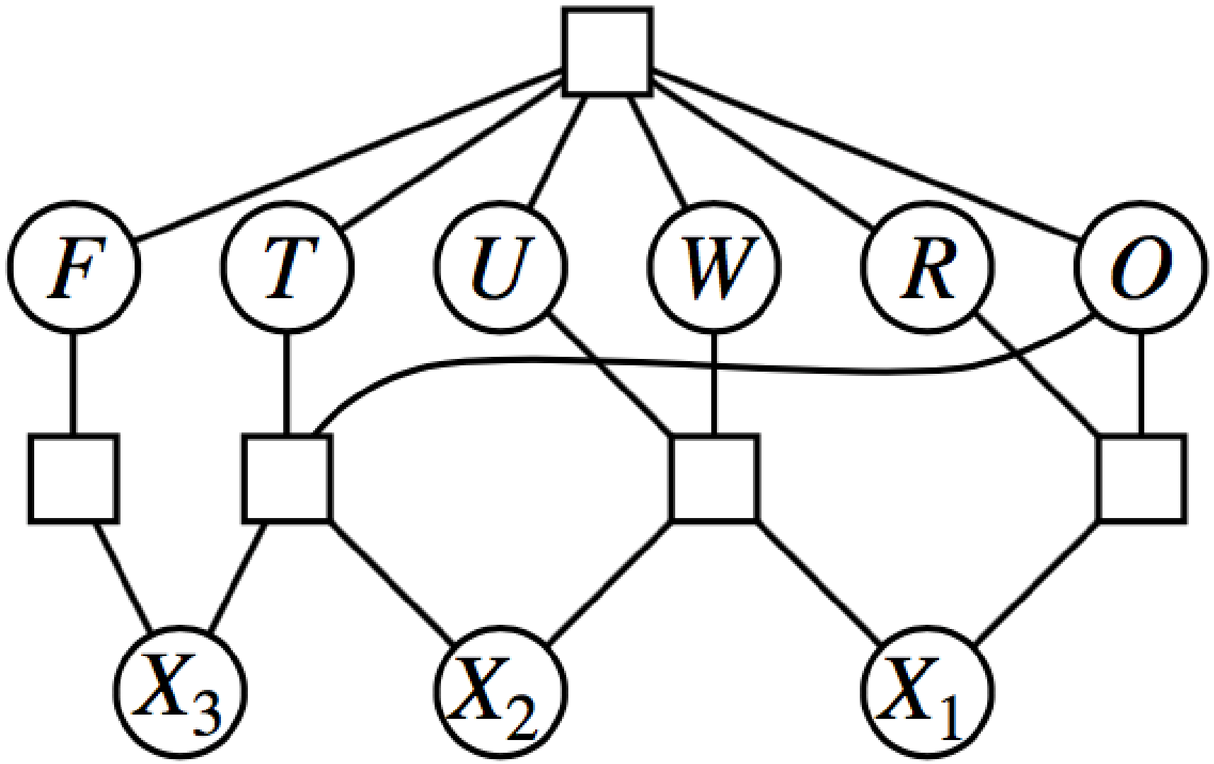
$$X_3 = F$$

$$Alldiff\,(F, T, U, W, R, O)$$

V = {T, W, O, F, U, R, X_1, X_2, X_3}

GHENT UNIVERSITY

CONSTRAINT HYPERGRAPH



Elke boog in de hypergraph (vierkant) stelt een restrictie voor op een verzameling van variabelen

GHENT UNIVERSITY

VOORBEELDEN VAN CSPS

- Toekenningsproblemen
  - Voorbeeld: wie geeft welke les
- Inroosteringsproblemen
  - Welke les wordt wanneer en waar gegeven
- Hardware configuratie
- Transport-problemen/scheduling
- Fabrieksplanning
- In veel van deze problemen: continue variabelen

GHENT UNIVERSITY

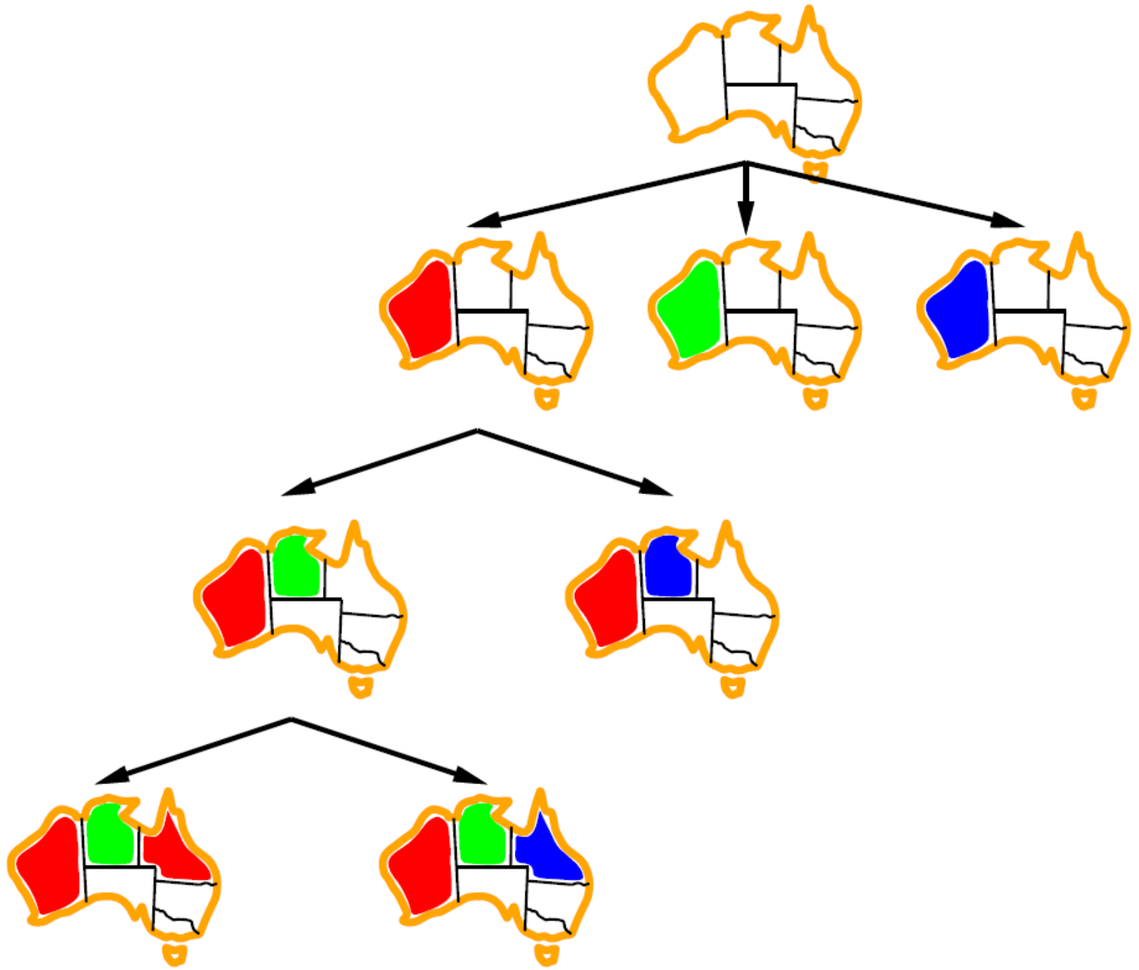
BACKTRACKING SEARCH

- Basis zoekmethode voor CSP
- Idee 1: exploreer 1 variabele per keer
  - Kies een vaste volgorde

  [WA = rood dan NT = groen] = [NT = groen dan WA = rood]
  - Telkens toewijzing aan 1 variabele per stap
- Idee 2: check restricties on-the-fly
  - Beschouw enkel waarden die niet conflicteren met reeds toegekende variabelen
  - Check restricties+goal test
- Equivalent met een depth-first search met 2 extra aanpassingen = backtracking

GHENT UNIVERSITY

VOORBEELD: GRAAF-KLEUREN



GHENT
UNIVERSITY

BACKTRACKING: PSEUDOCODE

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

function BACKTRACK(assignment, csp) returns a solution, or failure
    if assignment is complete then return assignment
    var ← SELECT-UNASSIGNED-VARIABLE(csp)
    for each value in ORDER-DOMAIN-VALUES(var, assignment, csp) do
        if value is consistent with assignment then
            add {var = value} to assignment
            inferences ← INFERENCE(csp, var, value)
            if inferences ≠ failure then
                add inferences to assignment
                result ← BACKTRACK(assignment, csp)
                if result ≠ failure then
                    return result
        remove {var = value} and inferences from assignment
    return failure
```

Waar zijn er verbeteringen mogelijk ?

GHENT UNIVERSITY

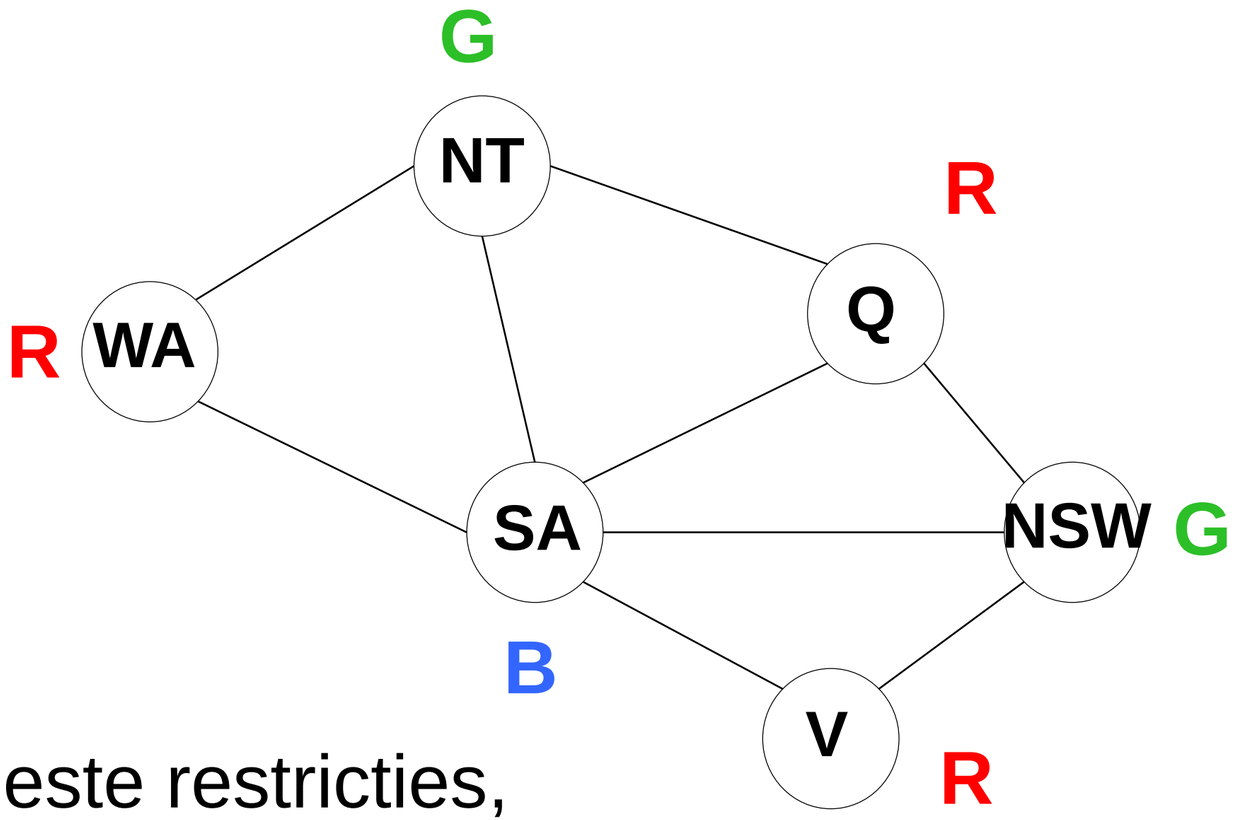
HEURISTIEKEN VOOR CSPS

- Invloed van de volgorde ?
  - Welke variabelen eerst toekennen ?
  - Welke waarden eerst toekennen ?
- Kunnen we de zoekboom snoeien ?
  - Filtering: vermijden van oplossing die sowieso tot fout zal leiden
  - Kunnen we snel tot een fout komen ?
- Kunnen we specifieke structuur-informatie gebruiken ?

MINIMUM REMAINING VALUES HEURISTIEK (MRV)

- Kies de variabele met het minst mogelijke waarden
  - Ook wel “most constrained variable” of “fail-first” heuristiek genaamd
- Als er een variabele X met 0 mogelijke goede toekenning overblijft, zal MRV X kiezen, en onmiddelijk een fout detecteren
  - Zorgt voor direct snoeien in de zoekruimte

CONSTRAINT GRAPH



SA heeft meeste restricties,
Dus wijs toe aan SA ipv Q

GHENT UNIVERSITY

SPEEDUP

- Aantal consistency checks nodig om tot een oplossing te komen:

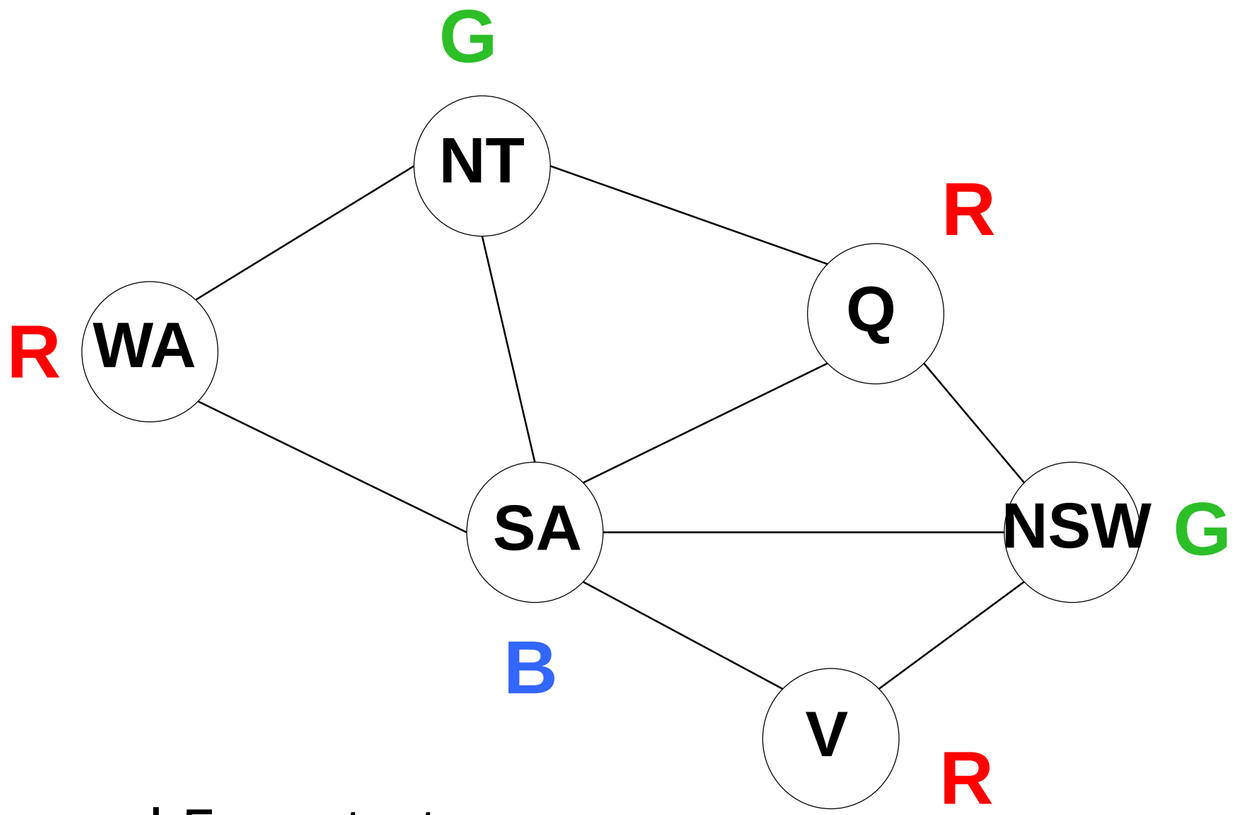
Problem	Backtracking	BT+MRV
USA	(> 1,000K)	(> 1,000K)
n-Queens	(> 40,000K)	13,500K
Zebra	3,859K	1K
Random 1	415K	3K
Random 2	942K	27K

GHENT UNIVERSITY

DE GRAAD-HEURISTIEK

- MRV heuristiek helpt niet bij het kiezen waar we starten
- De graad-heuristiek kiest de variabele die het meeste restricties heeft met andere niet-toegekende variabelen
- MRV heuristiek is typisch beter, maar graad-heuristiek kan nuttig zijn of bij gelijkspel

VOORBEELD

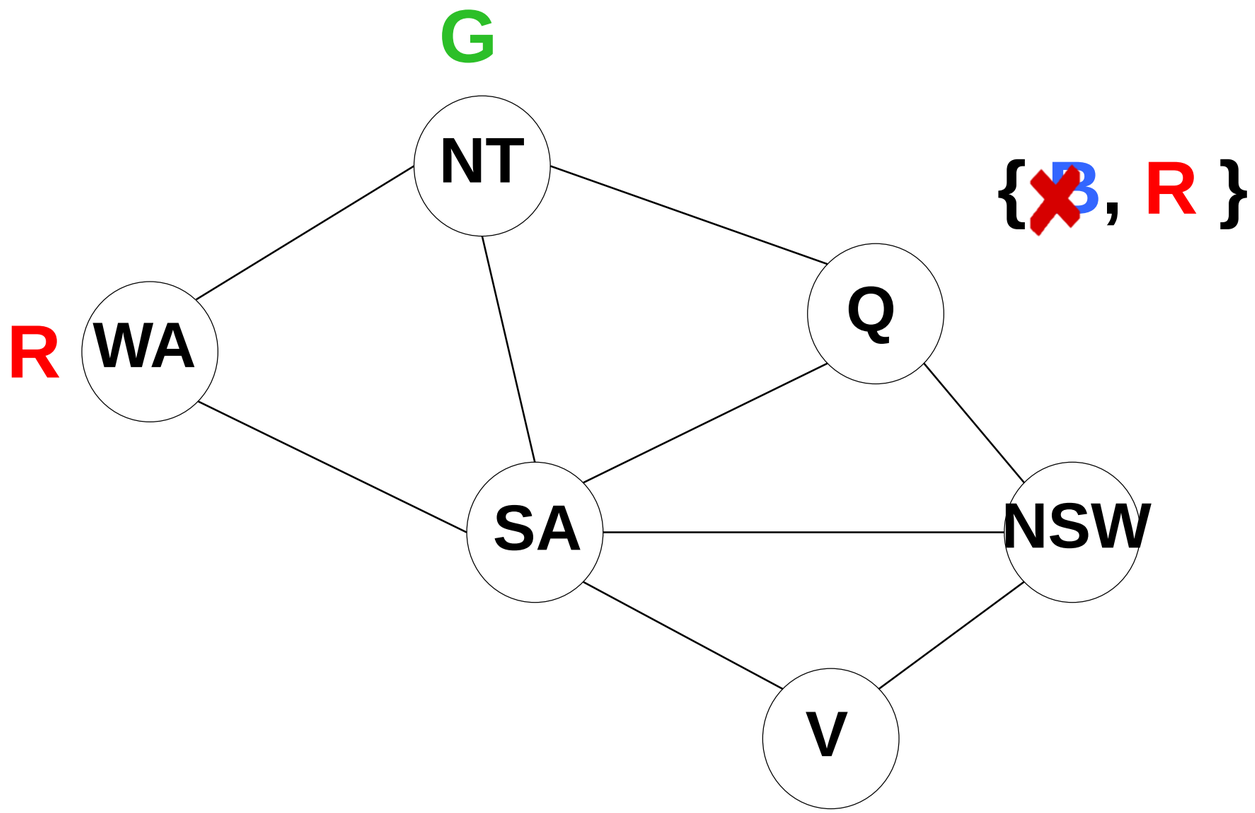


SA heeft graad 5 -> start

LEAST-CONSTRAINING-VALUE HEURISTIEK
LCV

- Eens we een variabele toegekend hebben, moeten we kiezen in welke volgorde we de mogelijke waarden gaan toekennen
- LCV verkiest de waarde die de meeste opties open houdt voor naburige variabelen in de constraint graph

LEAST-CONSTRAINING-VALUE HEURISTIC



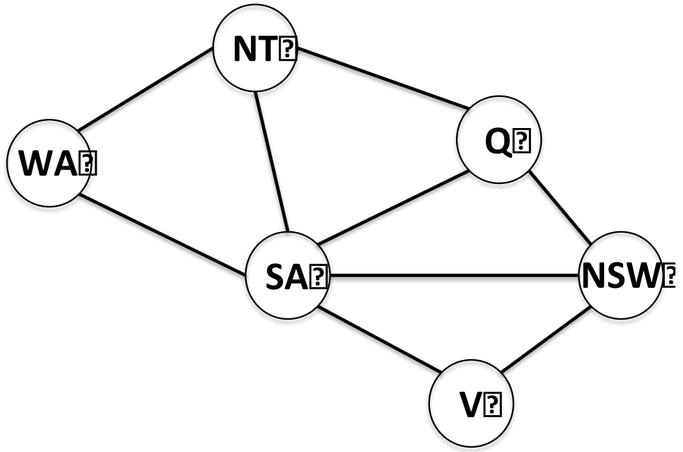
Hou opties open voor SA

GHENT UNIVERSITY

FILTERING: FORWARD CHECKING

- Stel dat variabele X toegewezen is
- Kijk vervolgens naar elke niet-toegewezen variabele Y die geconnecteerd is met X door een restrictie
- Verwijder uit domein van Y die waarden die inconsistent zijn met de gekozen waarde voor X

VOORBEELD



	WA	*NT*	*Q*	*NSW*	*V*	*SA*	*T*
Initial domains	R G B	R G B	R G B	R G B	R G B	R G B	R G B
After *WA=red*	Ⓡ	G B	R G B	R G B	R G B	G B	R G B

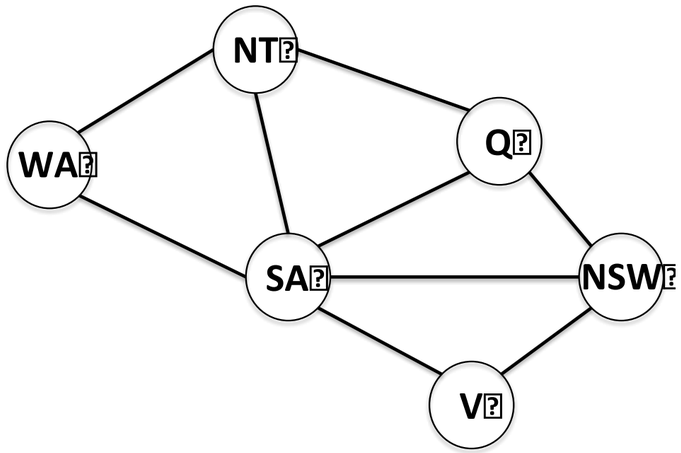
- Forward checking is een efficiente manier om incrementeel de informatie te berekenen die de MRV heuristiek nodig heeft
- Na WA zal MRV automatisch SA selecteren en dan NT

GHENT UNIVERSITY

SPEEDUP

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

BEPERKING VAN FORWARD CHECKING



Maar NT en SA zijn geconnecteerd, dus moeten verschillende kleur hebben -> we hadden al kunnen snoeien (en backtracken)

	WA	*NT*	*Q*	*NSW*	*V*	*SA*	*T*
Initial domains	R G B	R G B	R G B	R G B	R G B	R G B	R G B
After *WA=red*	(R)	G B	R G B	R G B	R G B	G B	R G B
After *Q=green*	(R)	B	(G)	R B	R G B	B	R G B

Zowel NT als SA moeten blauw kiezen

GHENT UNIVERSITY

PROPAGATIE VAN RESTRICTIES

- Idee: propageer de implicaties van een restrictie op 1 variabele al naar andere variabelen
- Moet snel berekend kunnen worden
  - Het heeft niet veel zin om meer tijd te steken in het berekenen van constraint propagatie dan om echt de zoekboom te doorzoeken

BOOGCONSISTENTIE (AC3)

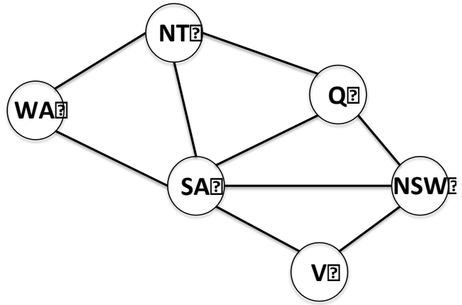
- Meest simpele vorm van propagatie maakt elke boog consistent:

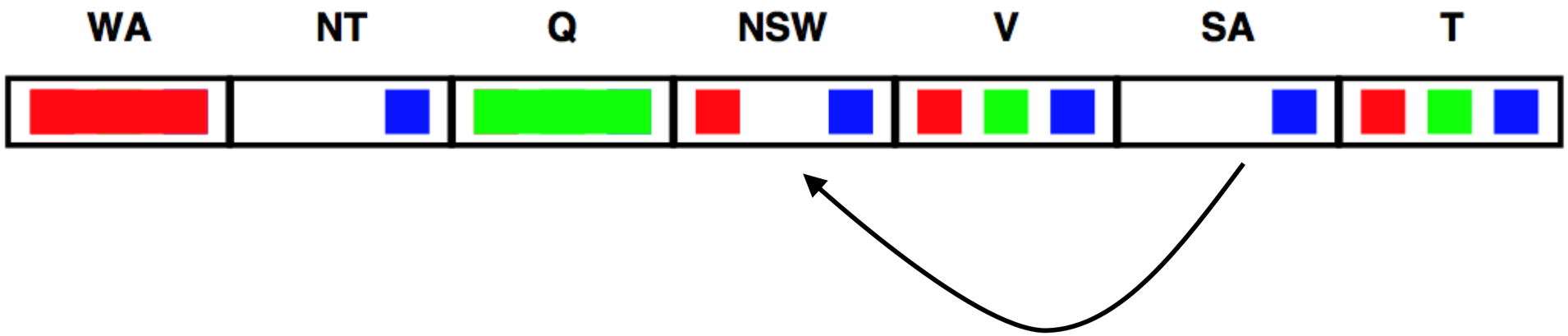
X → Y is ***consistent*** asa voor elke waarde x van X er een toegelaten waarde y van Y is

	WA	*NT*	*Q*	*NSW*	*V*	*SA*	*T*
Initial domains	R G B	R G B	R G B	R G B	R G B	R G B	R G B
After *WA=red*	(R)	G B	R G B	R G B	R G B	G B	R G B
After *Q=green*	(R)	B	(G)	R B	R G B	B	R G B

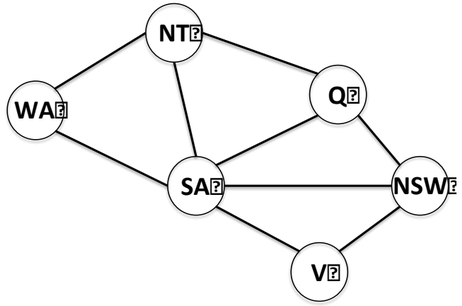
GHENT UNIVERSITY

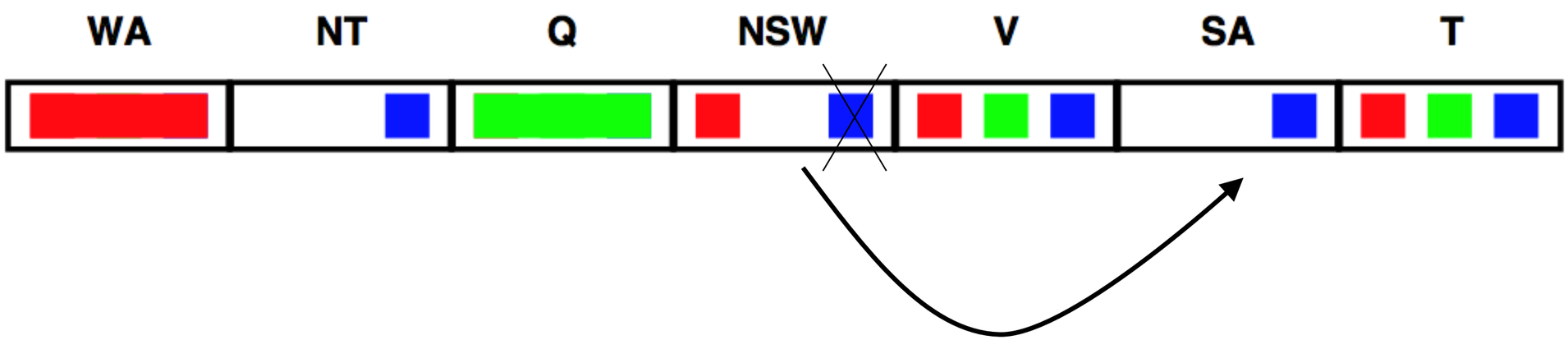
BOOGCONSISTENTIE





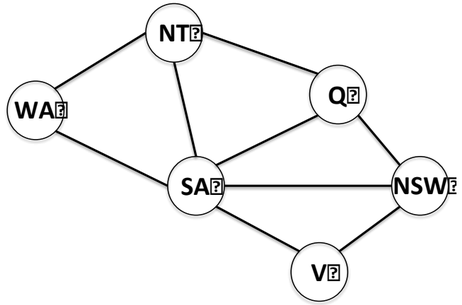
BOOGCONSISTENTIE

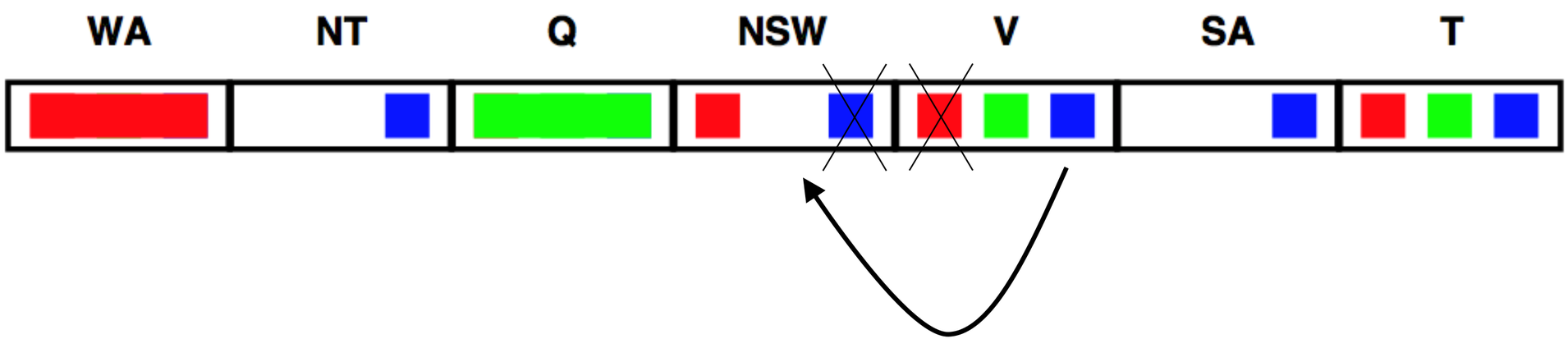




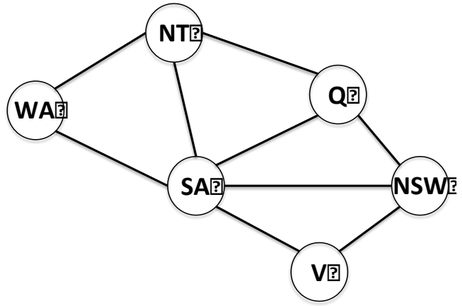
Als we van X een waarde elimineren moeten we de buren van X opnieuw controleren

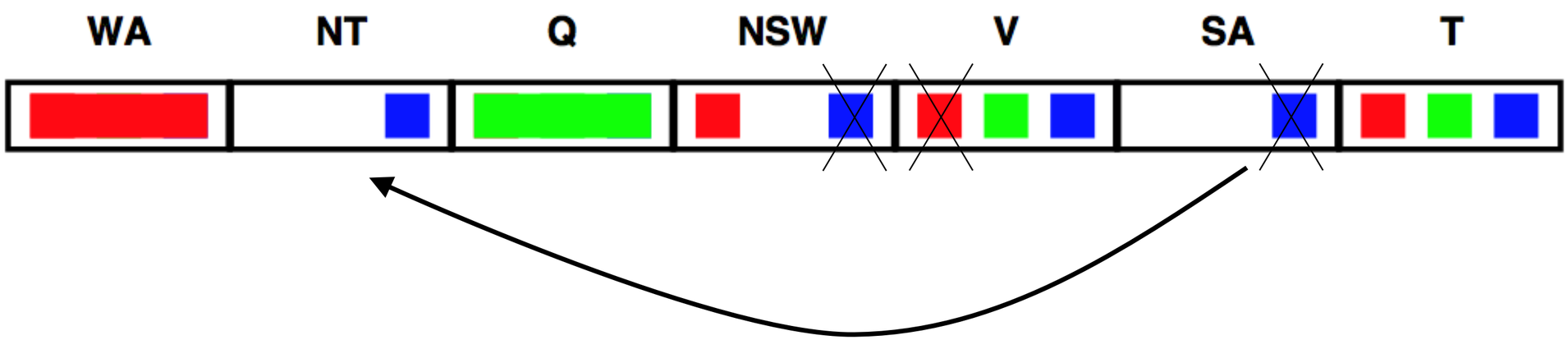
BOOGCONSISTENTIE





BOOGCONSISTENTIE





- Boogconsistentie detecteert fout sneller dan forward checking
- Kan als pre-processor gerund worden voor elke toekenning

GHENT UNIVERSITY

BOOGCONSISTENTIE: PSEUDOCODE

function AC-3(*csp*) **returns** the CSP, possibly with reduced domains
  inputs: *csp*, a binary CSP with variables $\{X_1, X_2, \ldots, X_n\}$
  local variables: *queue*, a queue of arcs, initially all the arcs in *csp*

  while *queue* is not empty **do**
    $(X_i, X_j) \leftarrow$ REMOVE-FIRST(*queue*)
    if REMOVE-INCONSISTENT-VALUES(X_i, X_j) **then**
      for each X_k **in** NEIGHBORS[X_i] **do**
        add (X_k, X_i) to *queue*

---

function REMOVE-INCONSISTENT-VALUES(X_i, X_j) **returns** true iff succeeds
  removed $\leftarrow$ *false*
  for each x **in** DOMAIN[X_i] **do**
    if no value y in DOMAIN[X_j] allows (x,y) to satisfy the constraint $X_i \leftrightarrow X_j$
      then delete x from DOMAIN[X_i]; *removed* $\leftarrow$ *true*
  return *removed*

GHENT UNIVERSITY

Consistency in networks of relations AK Mackworth, Artificial intelligence 8 (1), 99-118

AC3 COMPLEXITEIT

- N=aantal variabelen, d=max. domeingrootte
- Een binair CSP heeft ten hoogste $O(N^2)$ bogen
- Elke boog(X_k,X_i) kan maximum d keer in de wachtrij toegevoegd worden (d waarden)
- Checken van de consistentie van een boog kan gedaan worden in $O(d^2)$ tijd
- Totale worst-case tijd is $O(N^2d^3)$
  - Hoewel dit duurder is dan forward checking, is deze extra kost meestal wel de moeite waard

STERKERE VORMEN VAN CONSTRAINT PROPAGATIE

- Een CSP is ***k-consistent*** als, voor elke verzameling van k − 1 variabelen en voor elke consistente toewijzing aan die variabelen, er een consistente toewijzing kan gebeuren aan elke k^e variabele
- 1-consistency = node consistency
- 2-consistency = arc consistency
- 3-consistency = path consistency
  - Elk paar aanliggende variabelen kan steeds uitgebreid worden naar een derde aanliggende variabele

STRUCTUUR UITBUITEN

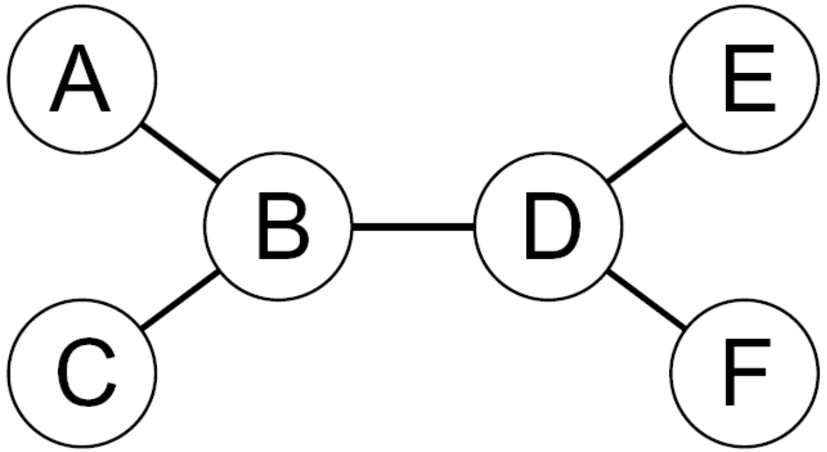
- Extreem geval: onafhankelijke deelprobleem
  - Voorbeeld: Tasmania
- Onafhankelijke deelproblemen zijn elk een component van de constraint graaf
  - Stel dat de graaf met n variabelen kan opgedeeld worden in onafhankelijke deelproblemen van elk c variabelen
  - Worst-case kost is $O((n/c)(d^c))$, lineair in n
  - Voorbeeld: n = 80, d = 2, c =20

    2^{80} = 4 miljard jaar (10 miljoen knopen/sec)

    $(4)(2^{20})$ = 0.4 seconds (10 miljoen knopen/sec)

GHENT
UNIVERSITY

CSPS MET BOOMSTRUCTUUR

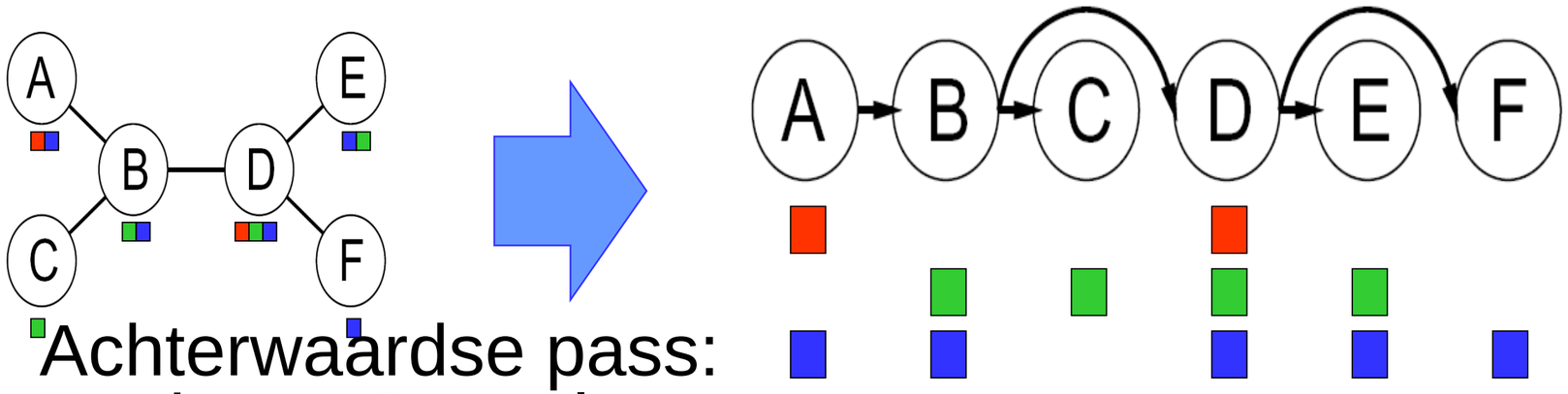


- Theorema: als de constraint graaf geen cykels bevat kan het CSP opgelost worden in $O(n\ d^2)$ tijd
  - Algemeen geval was $O(d^n)$

GHENT UNIVERSITY

CSPS MET BOOMSTRUCTUUR: ALGORITME

1. Volgorde: kies een wortel-variable, en rangschik zodat ouder-knopen kind-knopen voorafgaan

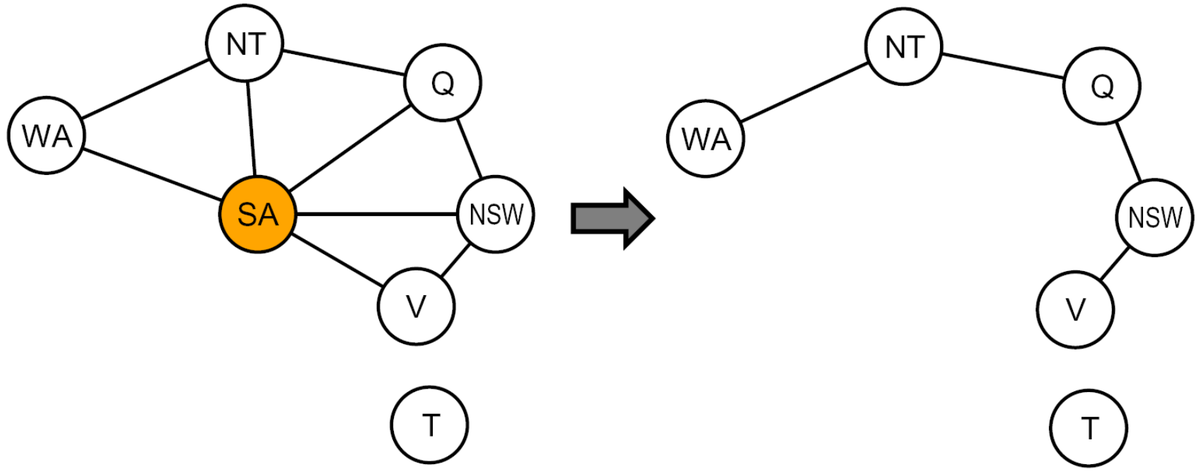


1. Achterwaardse pass:
   For i = n : 2, apply
   RemoveInconsistent(Parent(X_i),X_i)
2. Voorwaartse pass:
   For i = 1 : n, assign X_i consistently with
   Parent(X_i)

Runtime: $O(n\ d^2)$

GHENT UNIVERSITY

CSPS MET “BIJNA” BOOMSTRUCTUUR



- We gebruiken een trucje: conditionering
  - Ken een waarde toe aan een variabele, snoei domeinen van zijn buren
- “Cutset conditioning”
  - Overloop alle toekenningen aan een verzameling van variabelen zodat de resterende constraint graaf een boomstructuur heeft
- Cutset van grootte c resulteert in tijd O((d^c) (n-c) d^2)
  - Snelheidswinst voor kleine waarden van c

GHENT UNIVERSITY

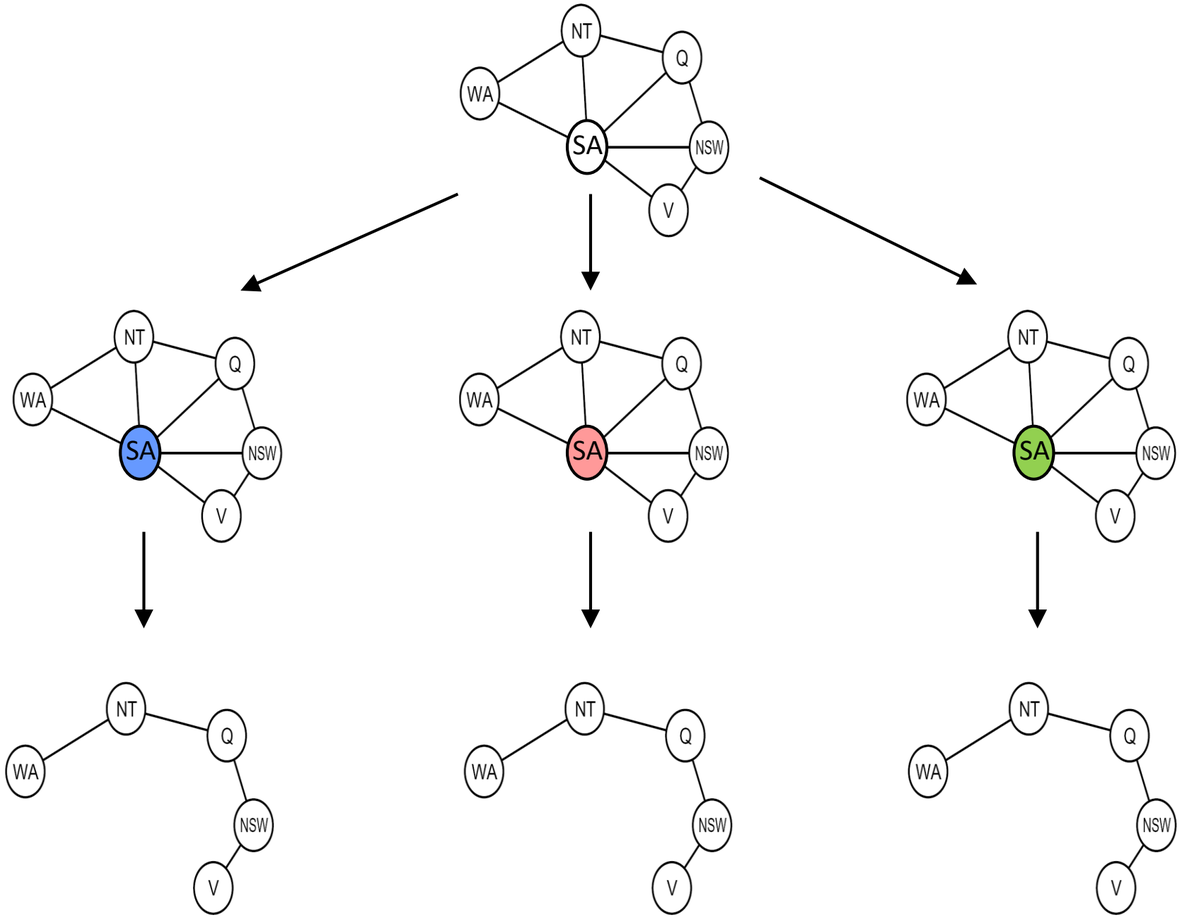
CUTSET CONDITIONING

Kies een cutset

Ken waarden toe aan cut set (alle mogelijkheden)

Bereken resterend CSP probleem

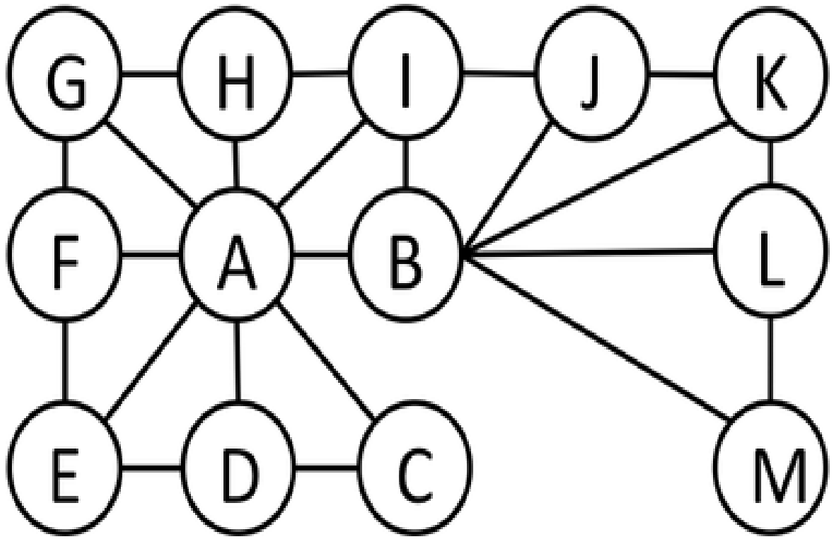
Los CSP probleem op



GHENT UNIVERSITY

CUTSET QUIZ

- Wat is de kleinste cutset voor onderstaande graaf ?



GHENT
UNIVERSITY

COMPLEXERE TYPES CSPS

- Discrete variabelen met oneindig domein
  - E.g. alle integers
  - Lineaire constraints of niet-lineaire constraints
  - Scheduling problemen
    - $StartJob_1 + 5 \leq StartJob_3$
- Continue variabelen
  - Convexe programmering
  - Lineaire programmering
  - Quadratische programmering
  - Second order cone programming (SOCP)

LOKALE ZOEKMETHODEN VOOR CSPS

GHENT UNIVERSITY

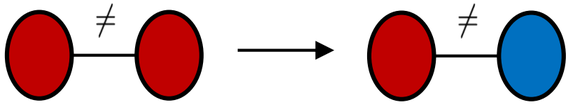
OVERZICHT

- **Constraint satisfaction problems (CSPs)**
  - Backtracking
  - Heuristieken
  - CSPs en boomstructuren
  - **Local search**

GHENT UNIVERSITY

ITERATIEVE ALGORITMEN VOOR CSPS

- Lokale zoekmethoden werken typisch op complete toekenningen (alle variabelen hebben een waarde toegekend)
- Voor CSPs:
  - Neem een toewijzing waarvan de restricties nog niet allemaal vervuld zijn
  - Operatoren kiezen nieuwe waarden voor de variabelen
  - Geen “zoekfront”

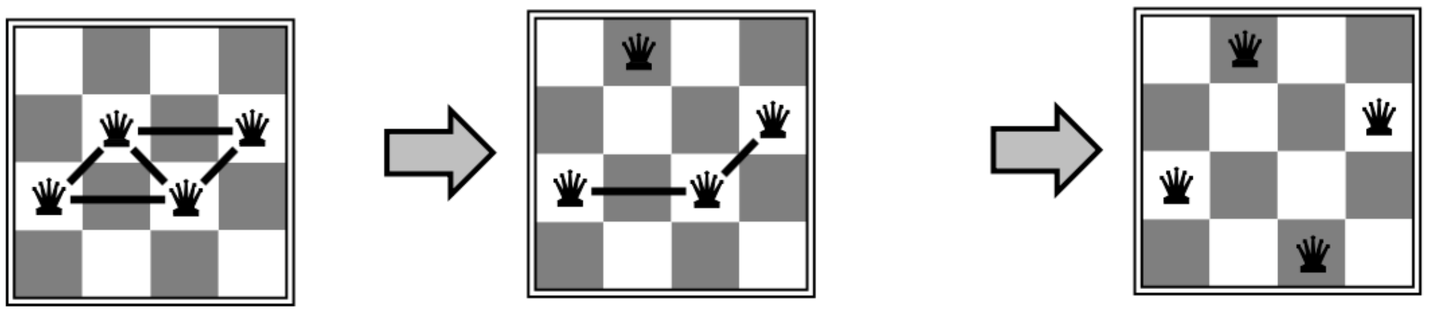


- Algoritme:
  - While(no solution) do
    - Selecteer variabele: kies willekeurig een variabele met een conflict
    - Kies een nieuwe waarde voor de variabele:
      - Bvb min-conflicts heuristiek
        - Kies een waarde die het minste constraints schendt
      - Voorbeeld: hill climbing met h(n) = totaal aantal geschonden constraints

GHENT UNIVERSITY

VOORBEELD: HILL-CLIMBING VOOR N KONINGINNEN

- Toestanden:
  - 4 koninginnen in 4 kolommen (4^4 = 256 states)
- Operatoren:
  - Verplaats 1 koningin in 1 kolom
- Doeltest: geen aanvallen mogelijk
- Evaluatiefunctie: h(n) = aantal aanvallen
  - Minimaliseer



GHENT UNIVERSITY

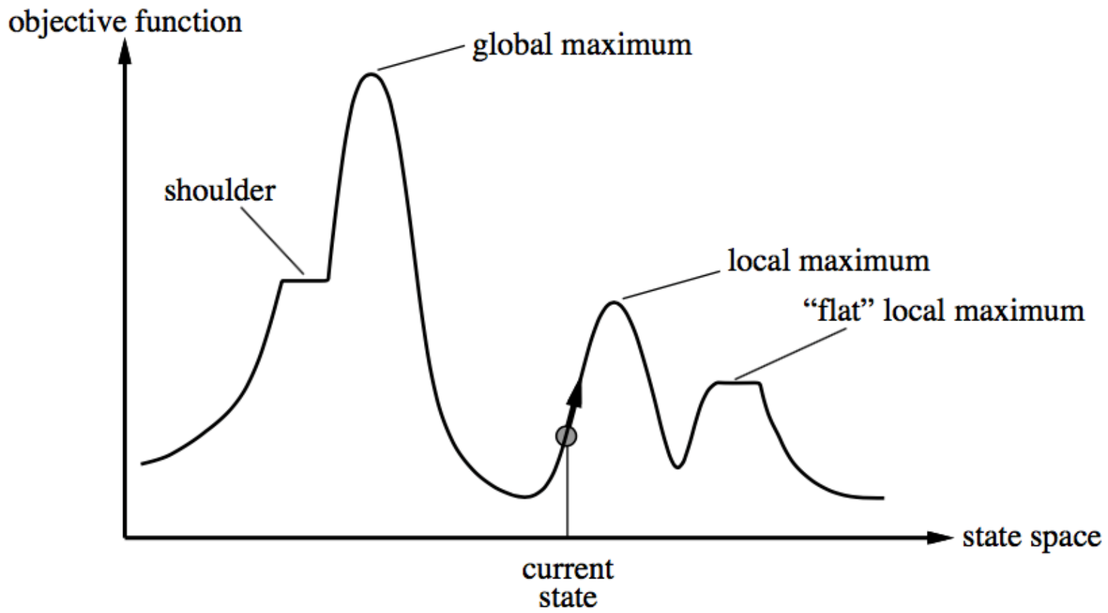
HILL CLIMBING ALGORITME (AKA STEEPEST ASCENT, AKA GREEDY LOCAL SEARCH)

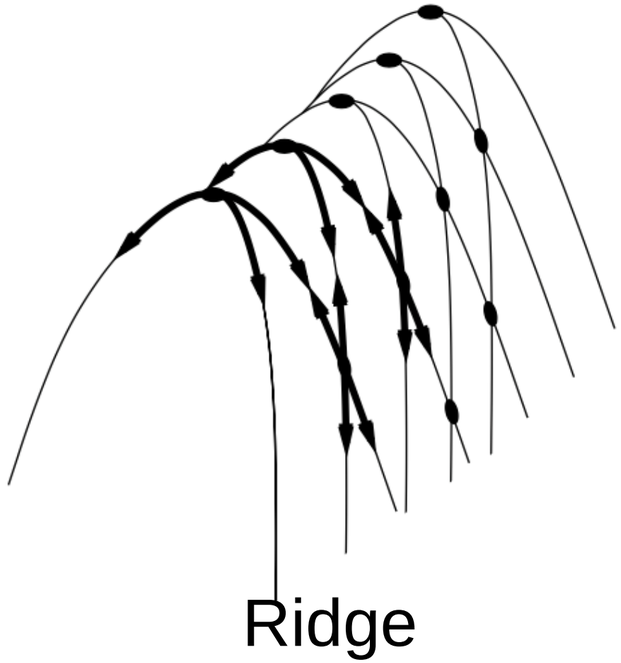
```
function HILL-CLIMBING( problem) returns a state that is a local maximum
    inputs: problem, a problem
    local variables: current, a node
                     neighbor, a node

    current ← MAKE-NODE(INITIAL-STATE[problem])
    loop do
        neighbor ← a highest-valued successor of current*
        if VALUE[neighbor] ≤ VALUE[current] then return STATE[current]
        current ← neighbor
    end
```

* Als we een heuristiek h gebuiken, dan zouden we de buur met de beste h waarde kiezen

GHENT UNIVERSITY

NADELEN VAN HILL-CLIMBING





- Compleet ? Optimaal ?
  - Lokale maxima
  - Ridges
- Plateau's

GHENT UNIVERSITY

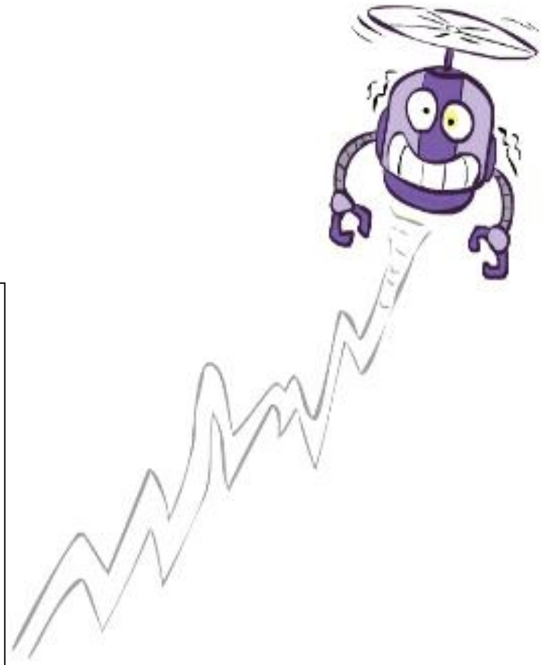
HILL CLIMBING: AANPASSINGEN

- Stochastische hill climbing
  - Kies een willekeurig punt tussen alle mogelijke oplossingen die de kostfunctie verbeteren
- “First-choice” hill climbing
  - Genereer willekeurige opvolgers totdat we een tegenkomen die beter is dan de huidig toestand
- “Random-restart” hill climbing
  - Herhaal het zoekalgoritme een aantal keer, te starten van van verschillende willekeurig gekozen startpunten

SIMULATED ANNEALING

- Idee: ontsnap uit een lokaal maximum door stappen “bergaf” toe te laten
  - Maar neem kleinere en minder frequente stappen bergaf naarmate de tijd vordert

```
function SIMULATED-ANNEALING(problem, schedule) returns a solution state
  inputs: problem, a problem
          schedule, a mapping from time to "temperature"
  local variables: current, a node
                   next, a node
                   T, a "temperature" controlling prob. of downward steps

  current ← MAKE-NODE(INITIAL-STATE[problem])
  for t ← 1 to ∞ do
      T ← schedule[t]
      if T = 0 then return current
      next ← a randomly selected successor of current
      ΔE ← VALUE[next] – VALUE[current]
      if ΔE > 0 then current ← next
      else current ← next only with probability e^(Δ E/T)
```



GHENT UNIVERSITY

SIMULATED ANNEALING: EIGENSCHAPPEN

- De probabiliteit om een slechtere oplossing te aanvaarden daalt exponentieel met de kwaliteit van de oplossing, ΔE
- Deze probabiliteit daalt ook naarmate de temperatuur T daalt:
  - “slechte” sprongen zijn vaker toegelaten in het begin van het algoritme
  - Ze worden minder frequent naarmate T daalt
- Wanneer T traag genoeg daalt kan gegarandeerd worden dat een globaal optimum gevonden wordt

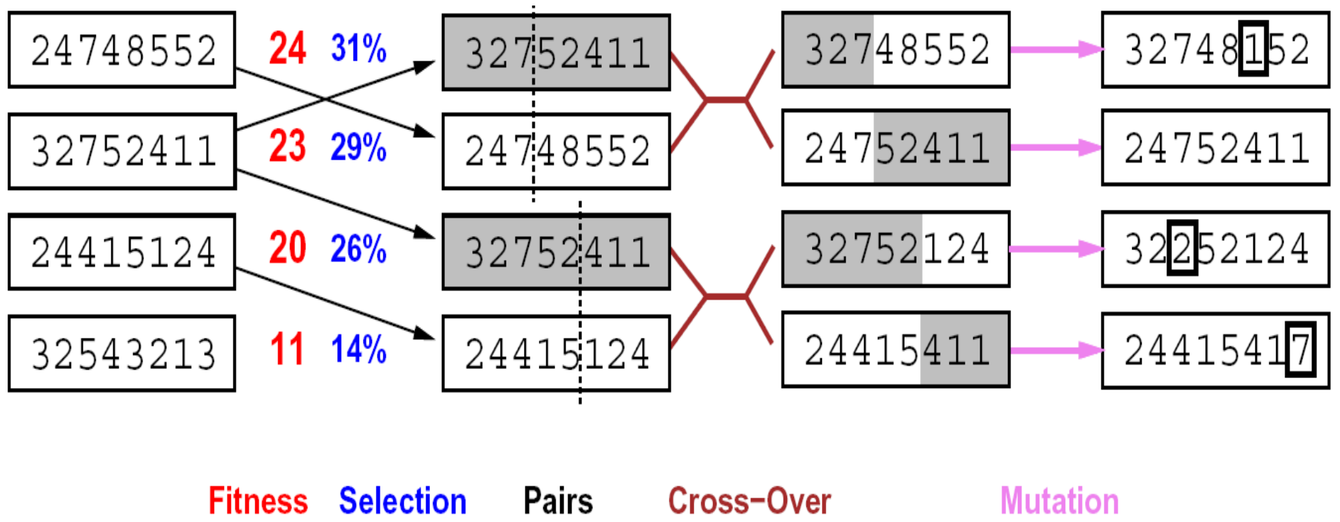
BEAM SEARCH

- Idee: in plaats van 1 oplossing iteratief aan te passen houden we nu k oplossingen bij
- Algoritme:
  1. Start met k willekeurig gegenereerde toestanden
  2. Genereer alle opvolgers van alle k toestanden
  3. Als een doeltoestand gevonden wordt, geef deze terug, zoniet kies de beste k van deze opvolger toestanden en itereer
- Dit is niet hetzelfde als k parallelle zoekopdrachten te lopen !

ANDERE METAHEURISTISCHE ZOEKMETHODEN

GHENT UNIVERSITY

GENETISCHE ALGORITMEN



- Gebaseerd op het principe van “evolutie”
  - Hou de beste N oplossingen over bij elke iteratie (selectie), gebaseerd op een “fitness” functie
  - Introduceer diversiteit: crossover en mutatie operatoren

GHENT UNIVERSITY

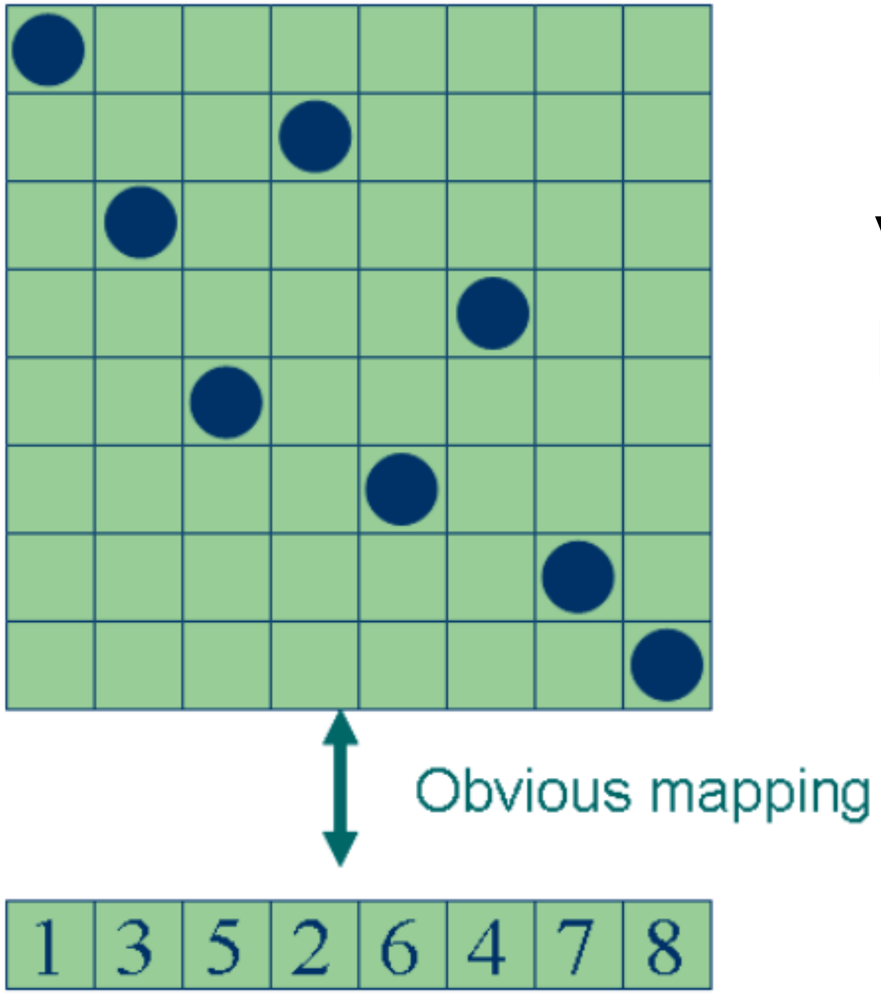
GA: PSEUDOCODE

1. *Randomly generate an initial population;*
2. *Evaluate each individual;*
3. *Do until termination criterion is met:*
   - 3.1 **Select** *some individuals for reproduction;*
   - 3.2 *Create offspring by* **crossing** *individuals;*
   - 3.3 *Occasionally* **mutate** *some individuals;*
   - 3.4 *Evaluate the new individuals;*
   - 3.5 *Compute new population;*
4. *Return a solution;*

GENETISCHE ALGORITMEN: EIGENSCHAPPEN

- In plaats van 1 oplossing, houden we een verzameling van oplossingen bij (populatie)
- We updaten deze verzameling op basis van de fitness functie en de genetische operatoren:
  - Selectie
  - Crossover (combineer eigenschappen van goede ouders om betere kinderen te maken)
  - Mutatie (exploratie van de zoekruimte)

VOORBEELD: 8 KONINGINNENPROBLEEM



Voorstelling:
Permutatie van 1-8

GHENT UNIVERSITY

EVALUATIEFUNCTIE

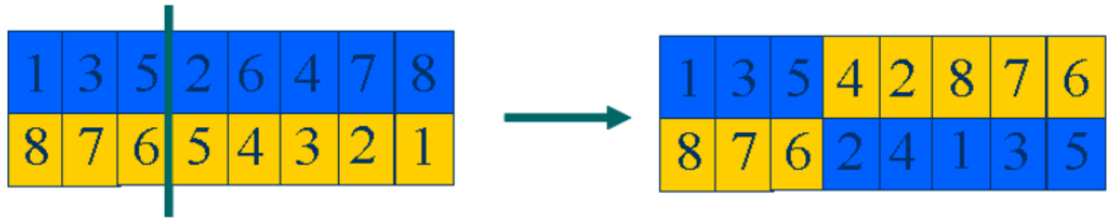
- Penalty voor elke koningin: het aantal andere koninginnen die ze kan schaken
- Totale penalty: som van alle penalties van alle individuele koninginnen
- Doel: vind een (unieke?) configuratie die de totale penalty minimaliseert (idealiter 0)
- Fitness: inverse van de totale penalty

OPERATOR: SELECTIE

- Keuze van de ouders:
  - Kies 5 ouders, en neem de beste 2 om crossover en mutatie toe te passen
- Volgende populatie
  - Wanneer een nieuw kind gegenereerd wordt, kies een individu om vervangen te worden door het kind:
    - Sorteer de hele populatie op fitness
    - Vervang het eerste individu dat een lagere fitness heeft dan het kind door het kind

OPERATOR: CROSSOVER

- Combineer 2 permutaties tot een nieuwe permutatie:
  - Kies een willekeurig kruisingspunt
  - Kopieer deel 1 naar de kinderen
  - Maak het tweede deel af door aan te vullen met de andere ouder:
    - In volgorde van voorkomen
    - Begin na het kruisingspunt
    - Sla waarden over die al in het kind aanwezig waren



GHENT UNIVERSITY

OPERATOR: MUTATIE

- Idee: genereer een kleine variatie van de permutatie
- Voorbeeld:
  - Kies 2 willekeurige posities en wissel hun waarden om

GHENT UNIVERSITY

ANDERE BIOLOGISCH GEÏNSPIREERDE ZOEK/OPTIMALISATIE-ALGORITMEN

- Geavanceerde evolutionaire algoritmen:
  - Niching
  - Co-evolutie
  - Eilandmodellen
- Swarm intelligence (PSO)
- Ant colony optimization (ACO)
- Artificial life

GHENT UNIVERSITY

ARTIFICIAL LIFE: VOORBEELDEN

https://www.youtube.com/watch?v=JBgG_VSP7f8

https://www.youtube.com/watch?v=CrWj_l-UrN4&t=98s

GHENT UNIVERSITY