Modul : Constraint Satisfaction Problem (CSP)

Terminologi Dalam CSP

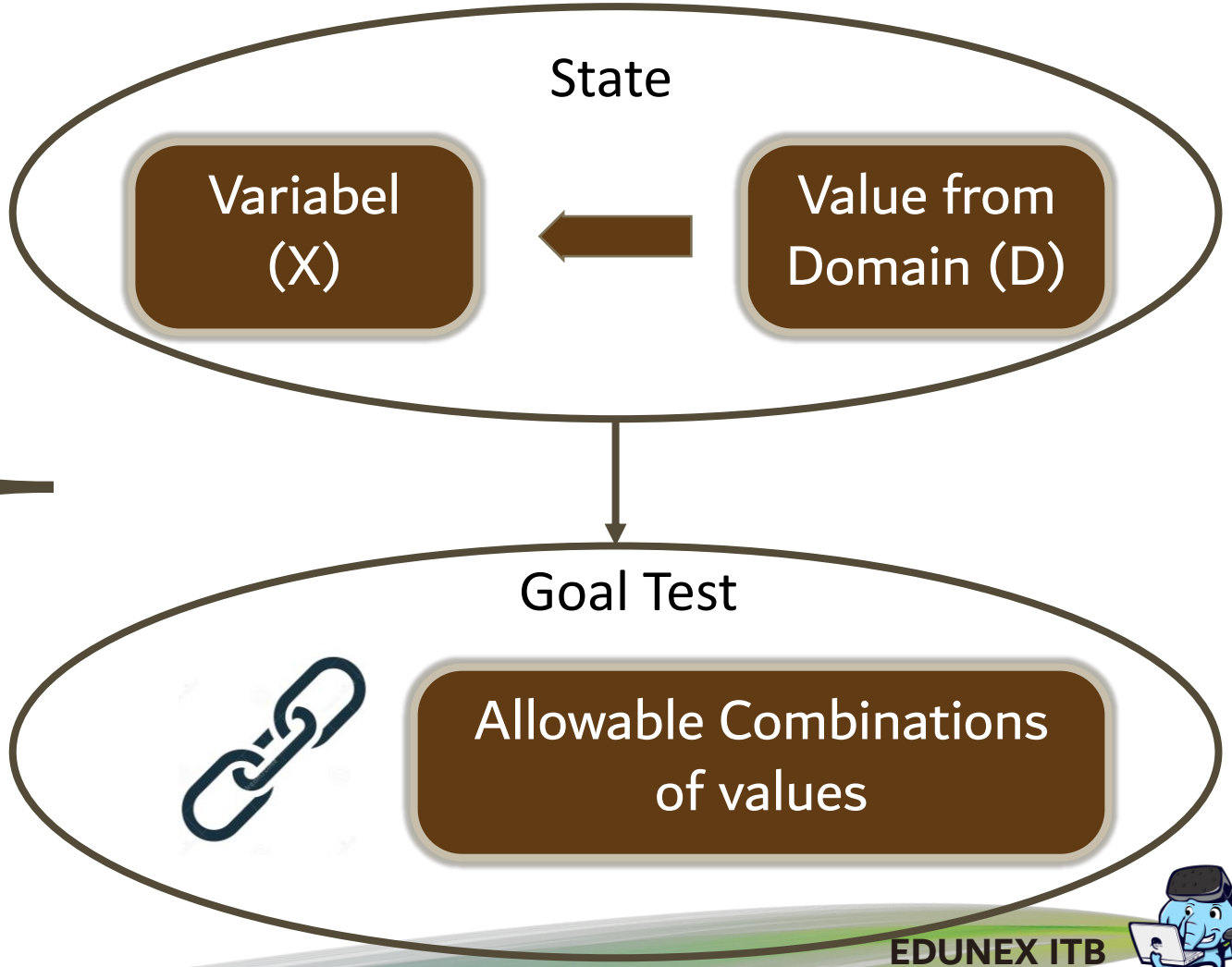
Nur ULFA Maulidevi

KK IF - Teknik Informatika- STEI ITB

Inteligensi Buatan
(Artificial Intelligence)

Constraint Satisfaction Problem (CSP)

- Termasuk dalam Problem Solving
- Formal Representation Language
- Allow General-Purpose Algorithm with more power

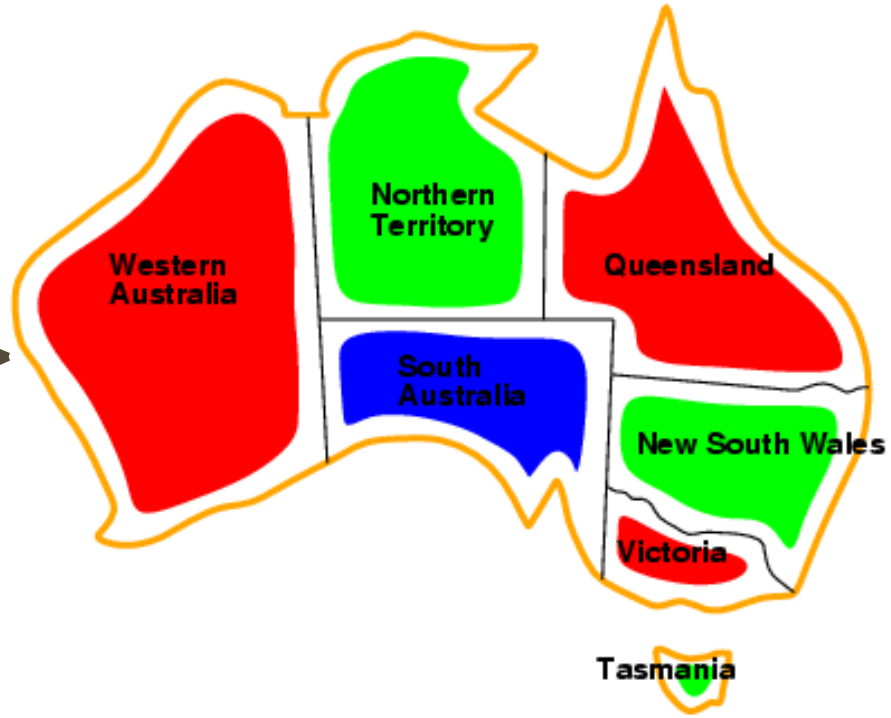


Example: Map-Coloring Problem

Variables:
WA, NT, Q, NSW, V, SA, T

Domain:
D_i = {red,green,blue}

Constraints:
adjacent regions must have different colors



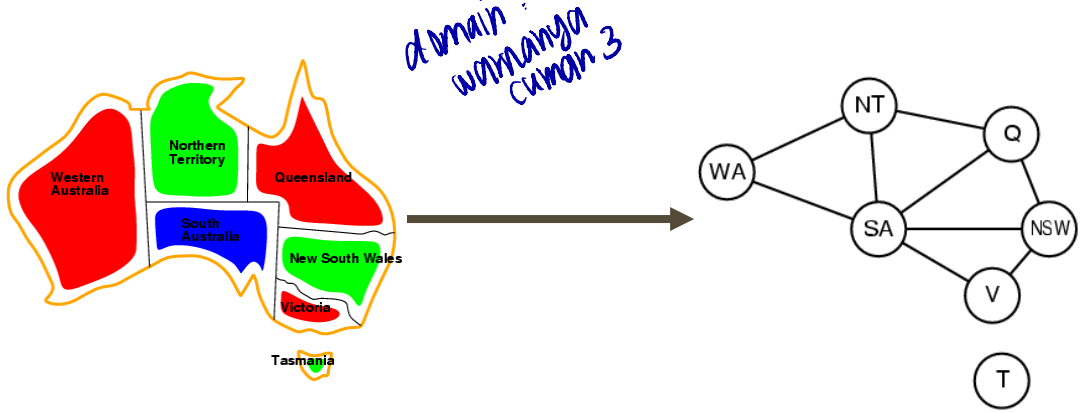
Solution: complete and consistent assignments

CSP Visualization

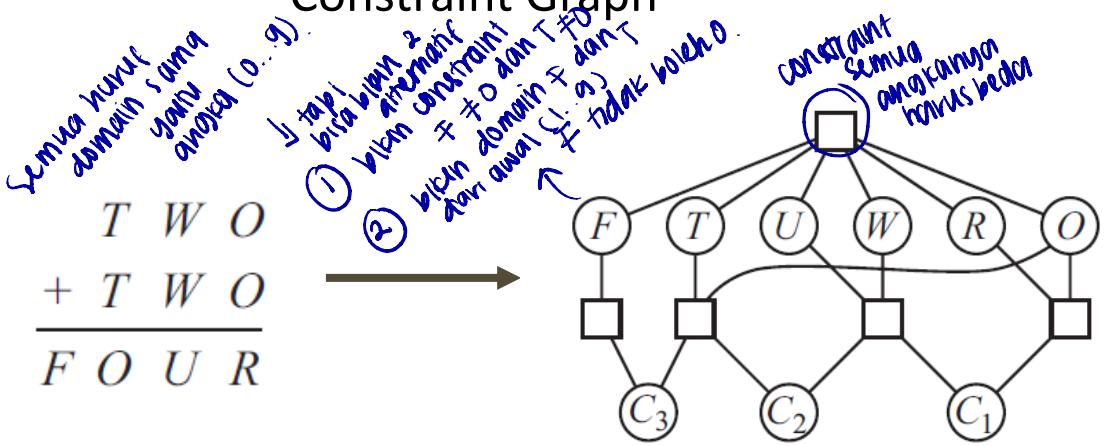
Variables → Nodes

Constraints → Link/ Arc

Constraints → Box

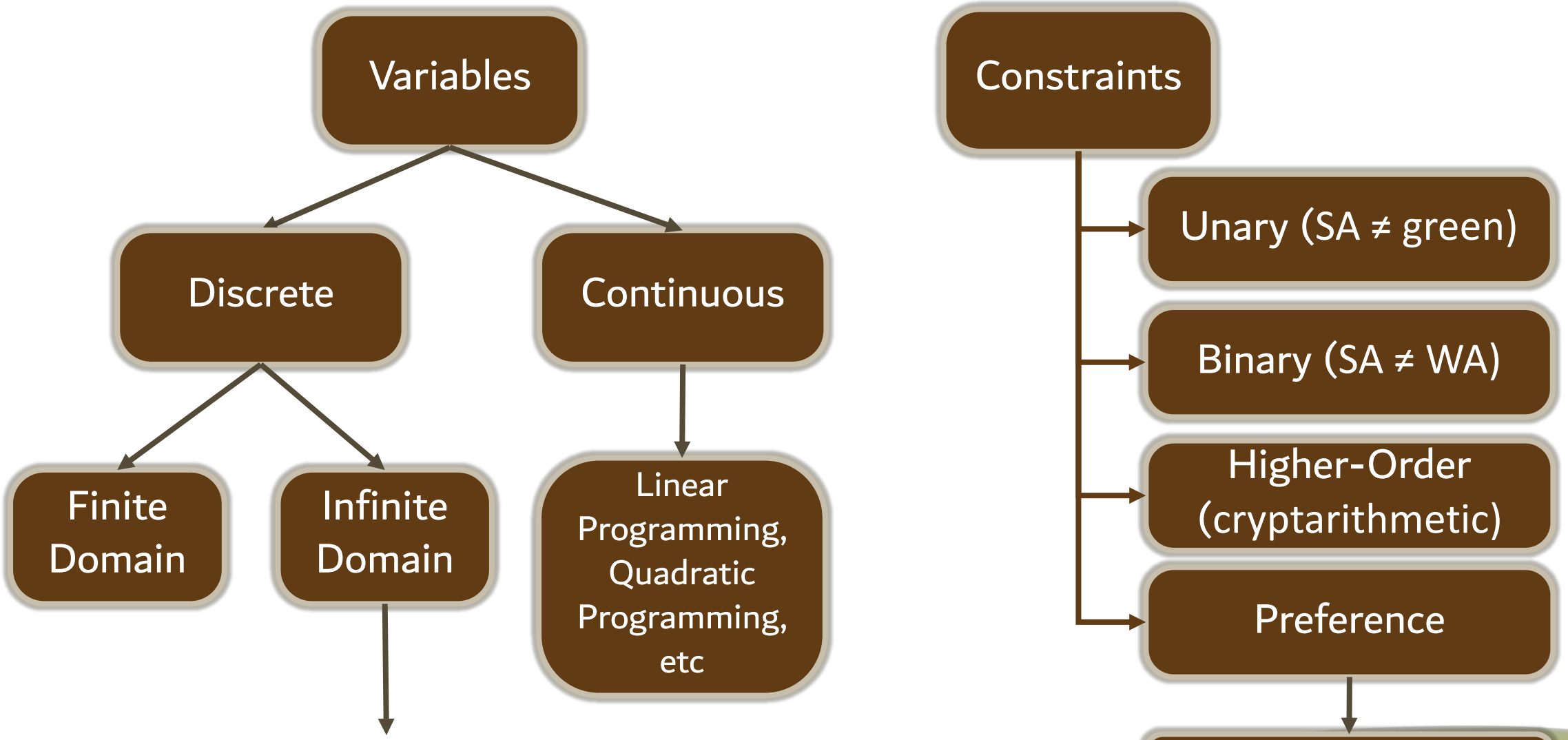


Constraint Graph



Constraint Hypergraph

Variations of CSP Formalism



Constraint Language: $StartJob_1 + 5 \leq StartJob_3$

Example: Cryptarithmetic Puzzle

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

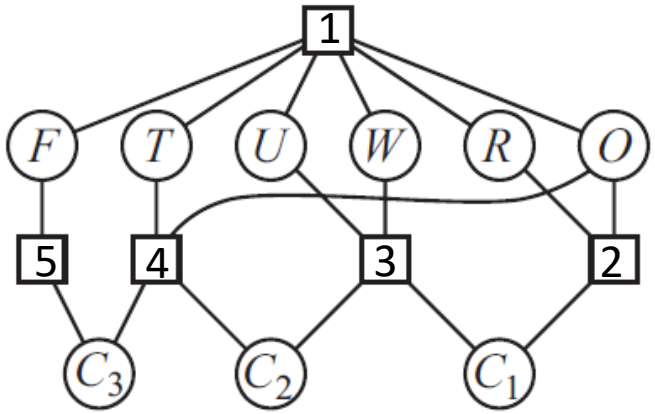
Variables

F,T,U,W,R,O

C_1, C_2, C_3 : auxiliary variables

Domain

F,T,U,W,R,O
={0,1,2,3,4,5,6,7,8,9}



Constraints

1. Alldiff *(F,T,U,W,R,O)*
2. $O + O = R + 10 \cdot C_1$
3. $W + W + C_1 = U + 10 \cdot C_2$
4. $T + T + C_2 = O + 10 \cdot C_3$
5. F = C_3 , $T \neq 0$, $F \neq 0$

Modul : Constraint Satisfaction Problem (CSP)

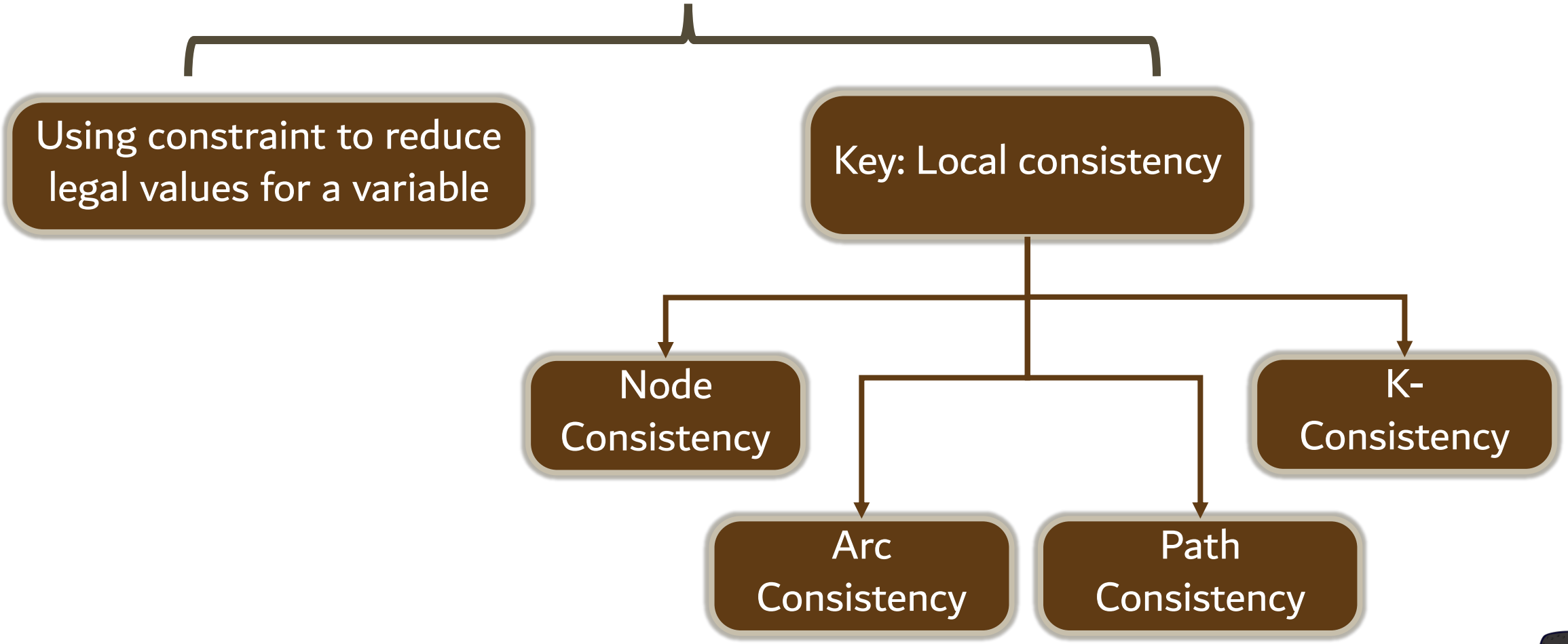
Inference in CSP

Nur ULFA Maulidevi

KK IF - Teknik Informatika- STEI ITB

Inteligensi Buatan
(Artificial Intelligence)

Constraint Propagation

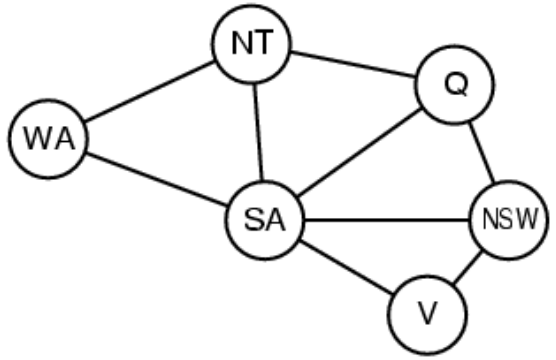


Node Consistency

All the values in the variable's domain satisfy the variable's **unary constraints**

Example: SA ≠ green

SA = {red, blue}

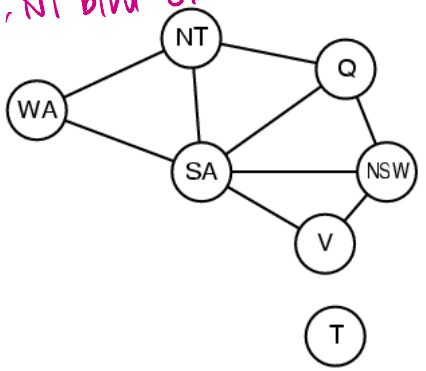


T mau warna apapun bebas (selama ada dlm domain) bcs dia ga terikat ke manapun

A network is node-consistent if every variable in the network is node-consistent

Arc Consistency (AC)

SA merah, WA biru OK
SA merah, NT biru OK



A variable in a CSP is **arc-consistent** if every value in its domain satisfies the variable's binary constraints

Example: SA ≠ WA

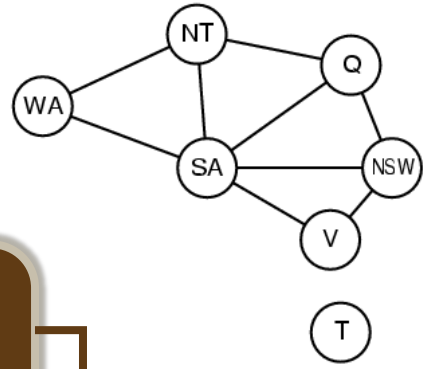
(SA,WA) = {(red , green), (red , blue), (green, red), (green, blue), (blue, red), (blue, green)}

Has no effect in this example (no reduction in the domain)

A network is arc-consistent if every variable is arc consistent with every other variable

Path Consistency (PC)

Arc Consistency: solve the problem if each variable has only 1 value left after the process OR finds that CSP can not be solved



Does not work for map coloring with only 2 values in the domain → (Path Consistency)

{Xi,Xj} is path-consistent to Xm if:
- Assignment {Xi = a,Xj = b} consistent with constraints on {Xi,Xj}
- There is assignment to Xm that satisfies constraints on {Xi,Xm} and {Xm,Xj}.

Example: Coloring Map with 2 colors (red, blue) → PC: {WA,SA} wrt NT → {WA = red ,SA = blue} or {WA = blue,SA = red} / No valid choice for NT → Eliminate both assignment → No solution

K-Consistency

A CSP is k-consistent if: any set of k − 1 variables & any consistent assignment to those variables, there is a consistent value to be assigned to k^{th} variable

- 1-consistency: given empty set, can make any set of one variable consistent
- 2-consistency = Arc Consistency
- 3-consistency = Path Consistency

Modul : Constraint Satisfaction Problem (CSP)

Backtracking Search for CSP

Nur ULFA Maulidevi

KK IF - Teknik Informatika- STEI ITB

Inteligensi Buatan
(Artificial Intelligence)

Backtracking Search

Use Depth First Search → Solution for n variables at depth n

classical dimulai dengan kosong (tdk ada variabel yg diassign)

local dimulai dengan random configuration

Path is irrelevant → variable assignment commutative

Only consider assignments to a single variable at each node

Basic uninformed algorithm for CSPs

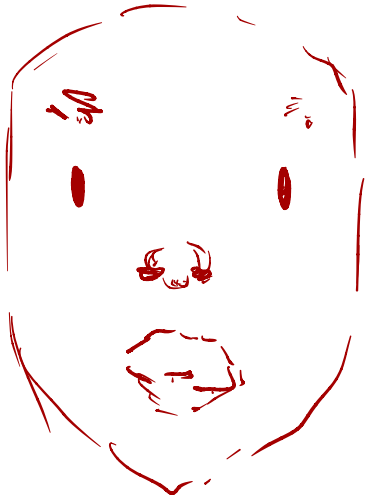
Algorithm

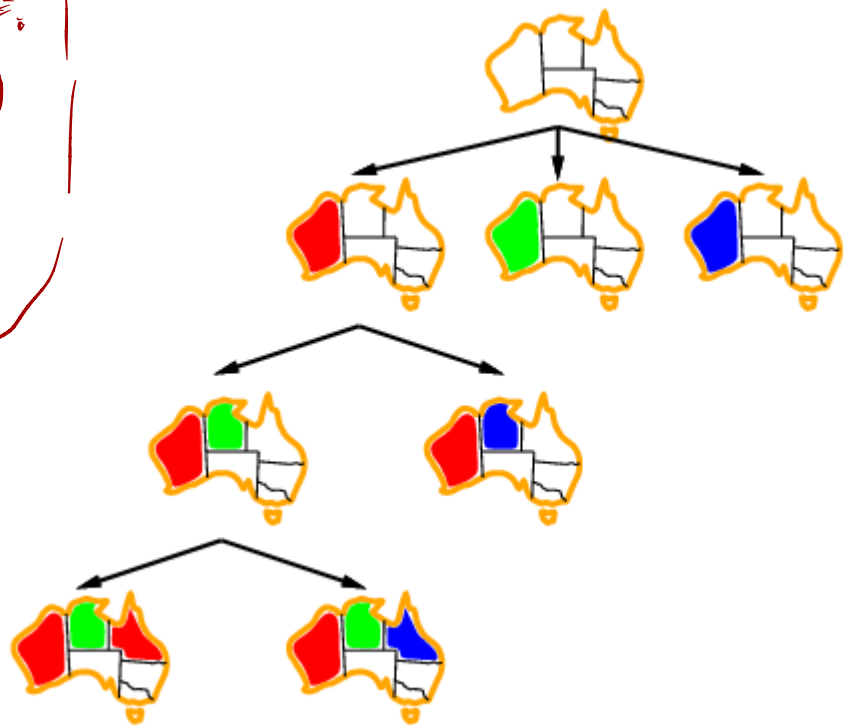
```
function BACKTRACKING-SEARCH(csp) returns a solution or failure
    return BACKTRACK(csp, { })

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

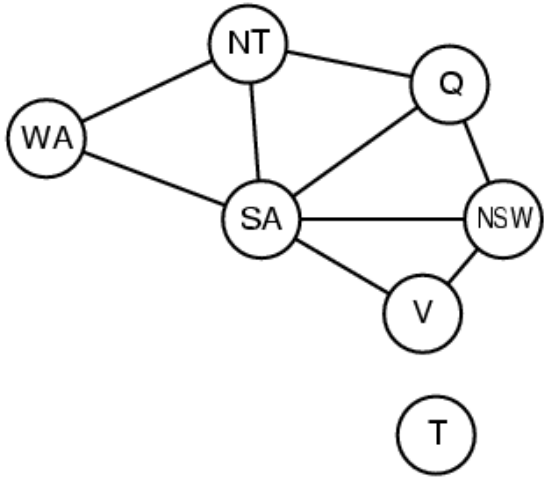
misal SA udah merah, WA dan NT kan gabolh merah, jadi merah tu diapus dr domain

inference : menghapus nilai yg membuat inconsistent dr domain

Example: Map Coloring Problem





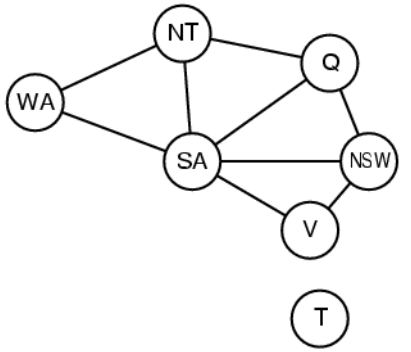


Improving Backtracking Efficiency

- Which variable should be assigned next?
- In what order should its values be tried?
- Detect inevitable failure early?

Without Domain Specific Knowledge

Variable Ordering



$var \leftarrow$ SELECT-UNASSIGNED-VARIABLE(csp)

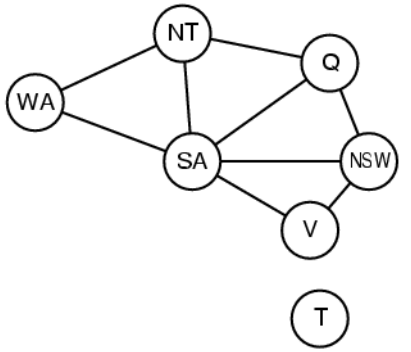
1. Static Variable Ordering: {*WA, NT, SA, Q, NSW, V, T* }

2. Minimum Remaining Values Heuristic

sudah assign nilai. ada nilai yg hilang dr domain variabel tertentu ⇒ inference
→ jd pilih variabel yg nilai dalam domainnya paling dikit

- Most Constrained Variable
- After WA=red and NT=green, SA or Q?
  - SA={blue} Q={blue, red}
  - Choose SA (has minimum remaining value)

Variable Ordering - 2



$var \leftarrow \text{SELECT-UNASSIGNED-VARIABLE}(csp)$

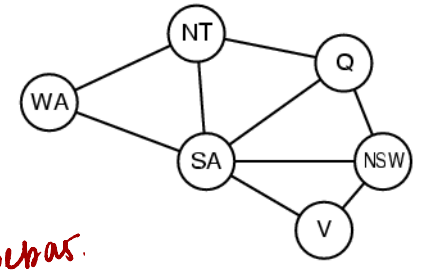
2. Minimum Remaining Values Heuristic : First Variable to Assign?

- Use degree heuristic → Number of constraints involved in a variable on other unassigned variable

SA di assign duluan karena constraintnya paling besar (5). T belakangan aja bcs no constraint.

- SA: 5
  NT, Q, NSW: 3
  WA, V: 2
  T: 0 → Choose SA First!!

Value Ordering



Least Constraining Value Heuristic

member kebebasan lebih buat variabel selanjutnya/yg belum di assign.

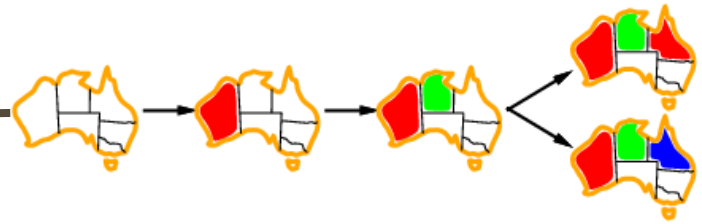
misal: urutan bebas.
isi WA merah
NT green
Q bisa merah/biru.
→ kalau Q biru, maka ketika inferensi SA habis.
→ kalau Q merah, SA masih ada.
jd pilih merah.

Prefer value that rules out fewest choice for neighboring variables

After WA=red and NT=green,
What color for Q?

Q=blue → SA = { }
Q=red → SA = {blue}

Choose value red for Q



Value Ordering is irrelevant if we want to have all possible solutions

Modul : Constraint Satisfaction Problem (CSP)

Interleaving Search and Inference in CSP

Nur ULFA Maulidevi

KK IF - Teknik Informatika- STEI ITB

Inteligensi Buatan
(Artificial Intelligence)

Interleaving search and inference

bisa dilakukan sblm pencarian juga, lho!

Inference can be done before searching

Interleaving search and inference → detect failure early

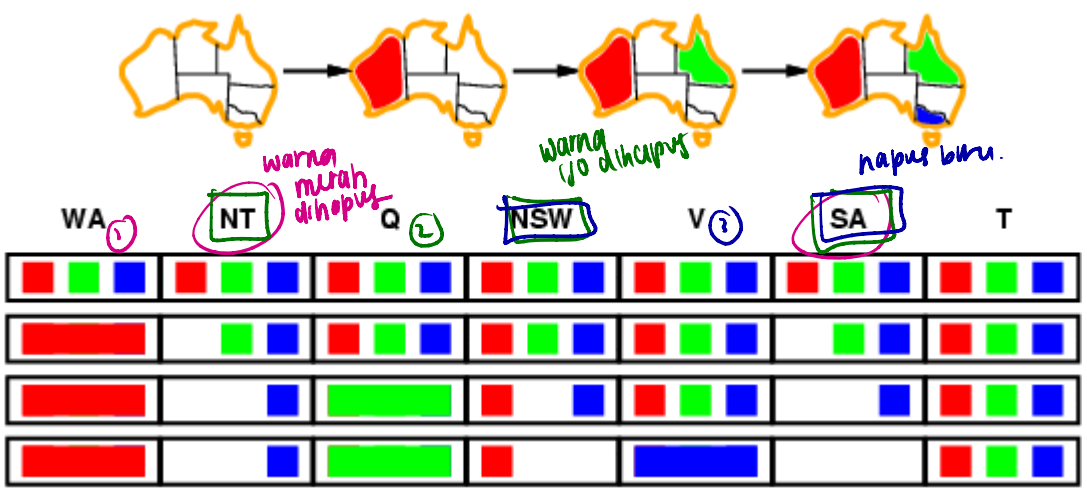
- Forward Checking: establishes arc consistency for binary constraint
- Constraint Propagation → Maintaining Arc Consistency (MAC)

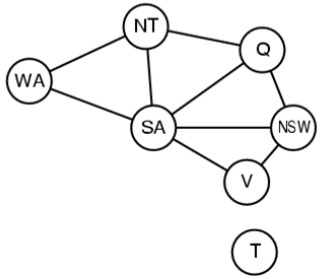
Let's see an example

Example: Interleave Search and Inference

Keep track of remaining legal values for unassigned variables

Terminate search when any variable has no legal values





FAIL!

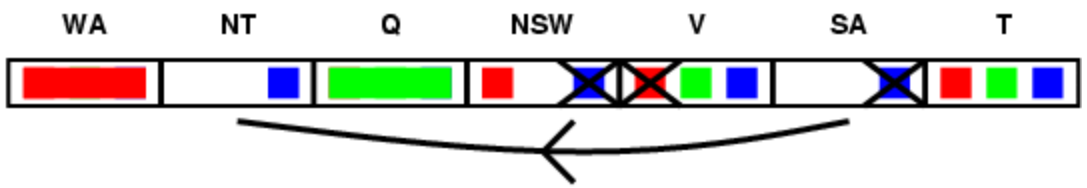
MAC: NT and SA cannot both be blue → MAC: repeatedly enforces constraints locally

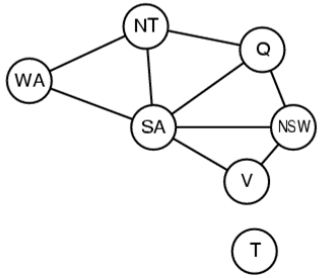
Example: Interleave Search and Inference

Keep track of remaining legal values for unassigned variables

Terminate search when any variable has no legal values

MAC: repeatedly enforces constraints locally





mencegah dari awal assignment yg gagal dari awal.

FAIL!

misal WA merah Q jdo NT sisa biru. SA yg sisa biru } uda pasti gabisa?

If variable *X* loses a value, neighbors of *X* need to be rechecked

artinya Q jgn di assign jdo.

Modul : Constraint Satisfaction Problem (CSP)

Local Search for CSP

Nur ULFA Maulidevi

KK IF - Teknik Informatika- STEI ITB

Inteligensi Buatan
(Artificial Intelligence)

Local Search

Complete-state formulation → initial state assigns a value to every variable

The search changes the value of one variable at a time

Variable selection: randomly select any conflicted variable

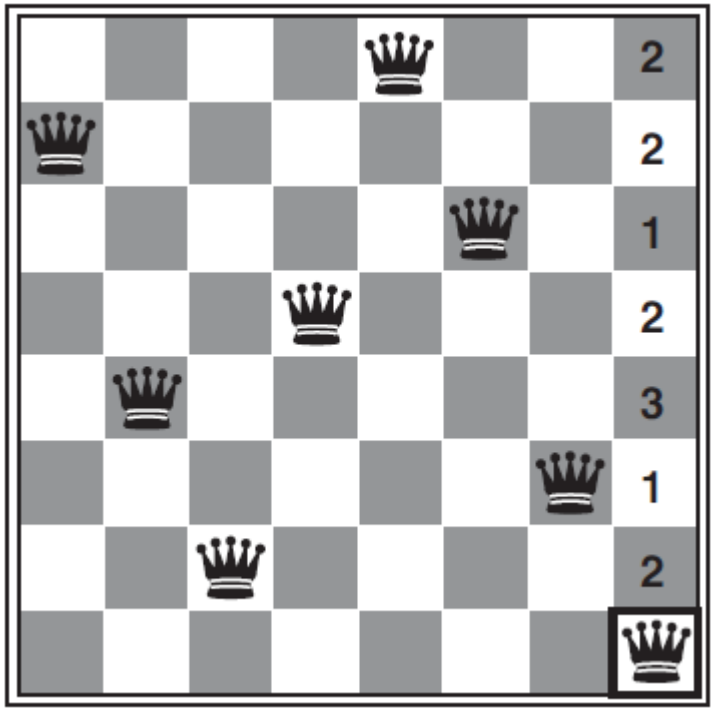
Value selection by min-conflicts heuristic

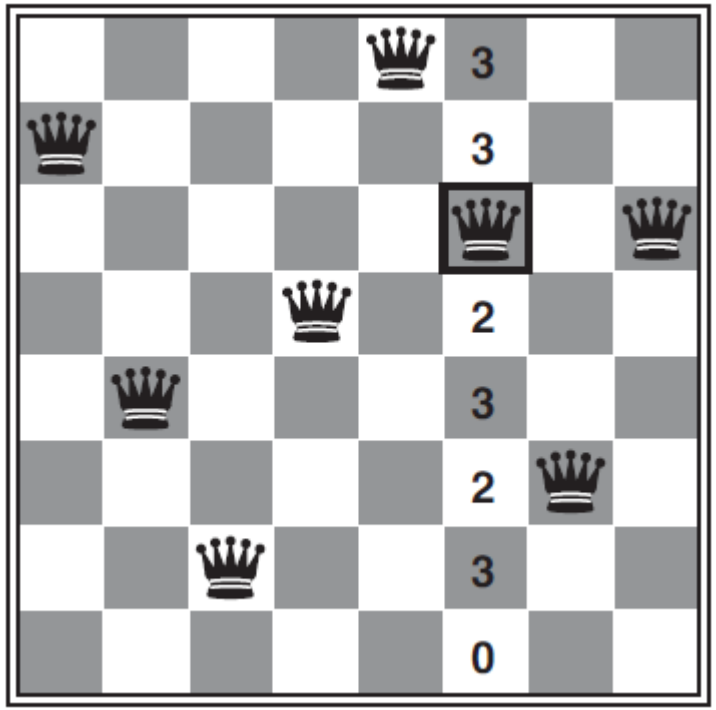
Min-Conflict Heuristic

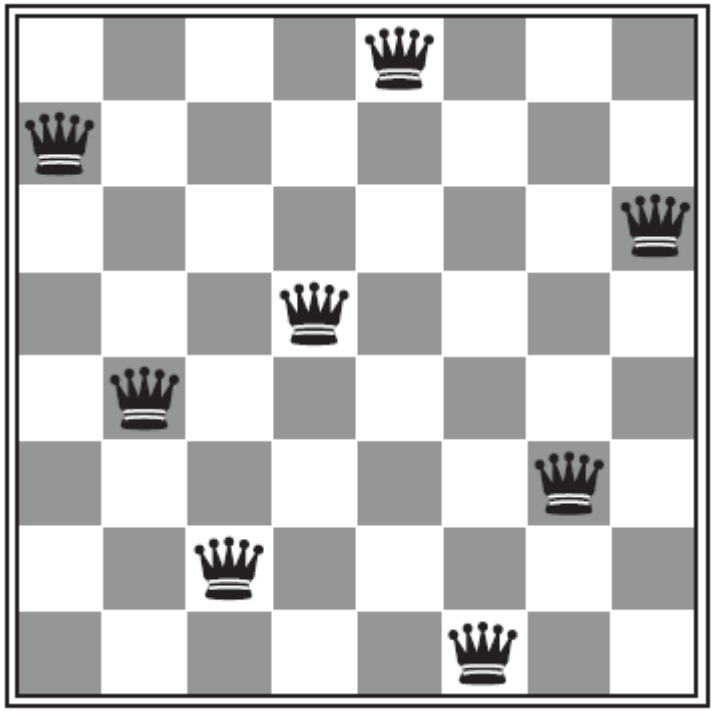
function MIN-CONFLICTS(csp, max_steps) **returns** a solution or failure
  inputs: csp, a constraint satisfaction problem
    max_steps, the number of steps allowed before giving up

  $current \leftarrow$ an initial complete assignment for csp
  for i = 1 to max_steps **do**
    if $current$ is a solution for csp **then return** $current$
    $var \leftarrow$ a randomly chosen conflicted variable from csp.VARIABLES
    $value \leftarrow$ the value v for var that minimizes CONFLICTS($var, v, current, csp$)
    set $var = value$ in $current$
  return *failure*

Example: n-Queens Problem







Applications

Solve *n*-queens in almost constant time for arbitrary *n* with high probability

Online setting → scheduling

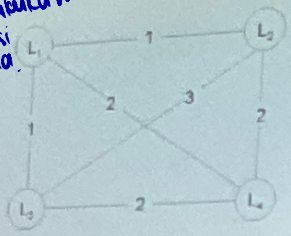
THANK YOU

Meeting Scheduling Problem:

tidak ada meeting dilaksanakan di lokasi yg sama

Meeting	Location	Attending agents
m_1	L_1	A_1, A_3
m_2	L_2	A_2, A_3, A_4
m_3	L_3	A_1, A_4
m_4	L_4	A_1, A_2

Gambar 1.



Gambar 2.

- Durasi tiap pertemuan: 1 time slot
- Jarak tempuh antar lokasi: sesuai Gambar 2 (dalam time slot)
- Dicari alokasi waktu tiap pertemuan dalam 1 hari

Tentukan:

a. Variable
b. Domain
c. Constraints

ga mencari jarak terdekat

$|m_i - m_j| \geq 1 + jarak_{i-j}$