Modul : Introduction to AI

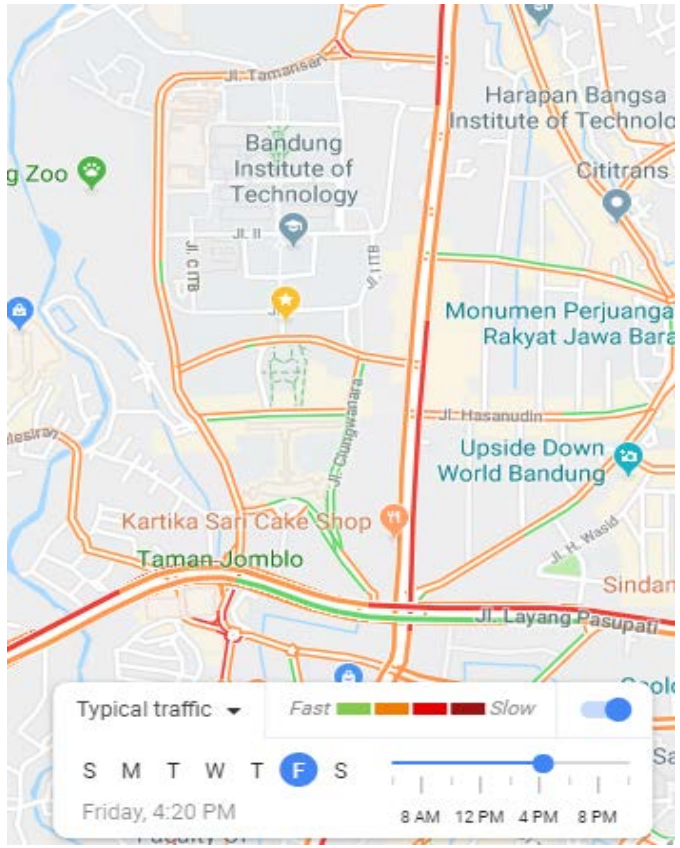
AI Application

KK IF - Teknik Informatika- STEI ITB

Inteligensi Buatan
(Artificial Intelligence)

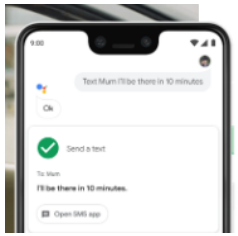
AI in Life: Path Finding

Path Finding / Direction

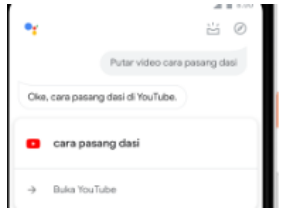


Traffic pattern

AI in Life: AI Assistant



"Hey Google, text Mum I'll be there in 10 minutes"

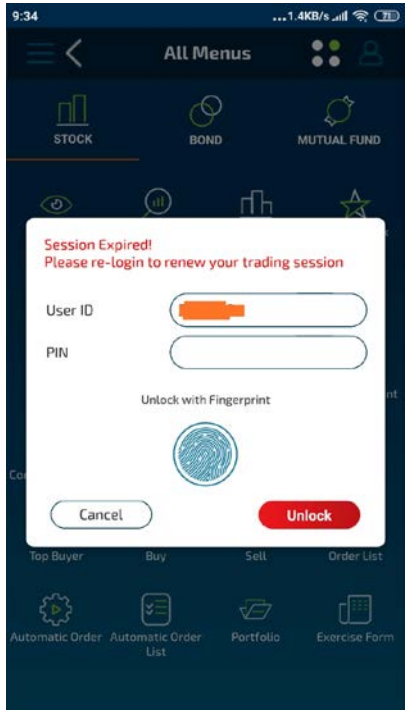


"Hey Google, play my morning playlist"

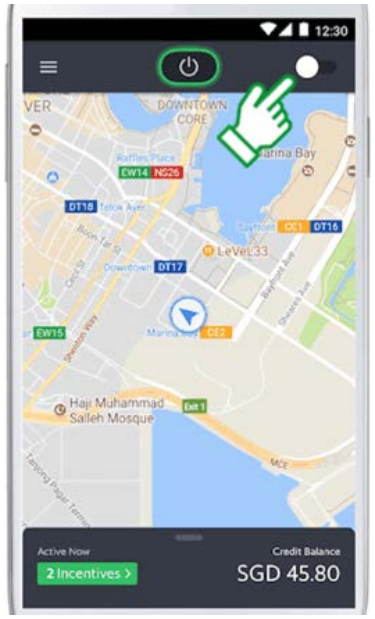
"Hey Google, dim the bedroom lights"

"Hey Google, set the temperature to 20 degrees"

AI in Life: Biometric Application



Fingerprint verification



Face verification

AI in Life: Transform Your Face

Pumarola, A., Agudo, A., Martinez, A.M., Sanfeliu, A., & Moreno-Noguer, F. (2019). *GANimation: One-Shot Anatomically Consistent Facial Animation. In International Journal of Computer Vision*

AI in Life: OpenAI Five for Dota 2

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

APRIL 15, 2019 • 7 MINUTE READ

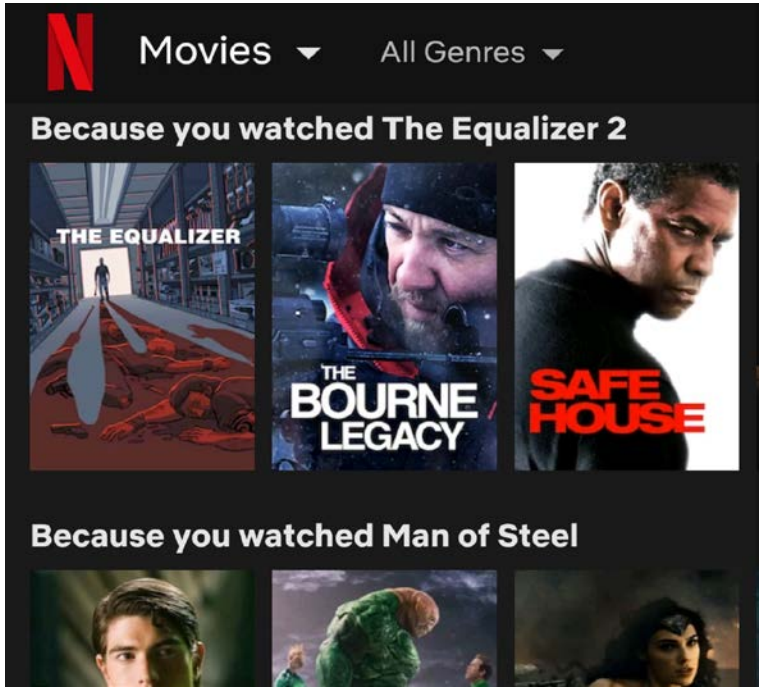
OpenAI Five Defeats Dota 2 World Champions

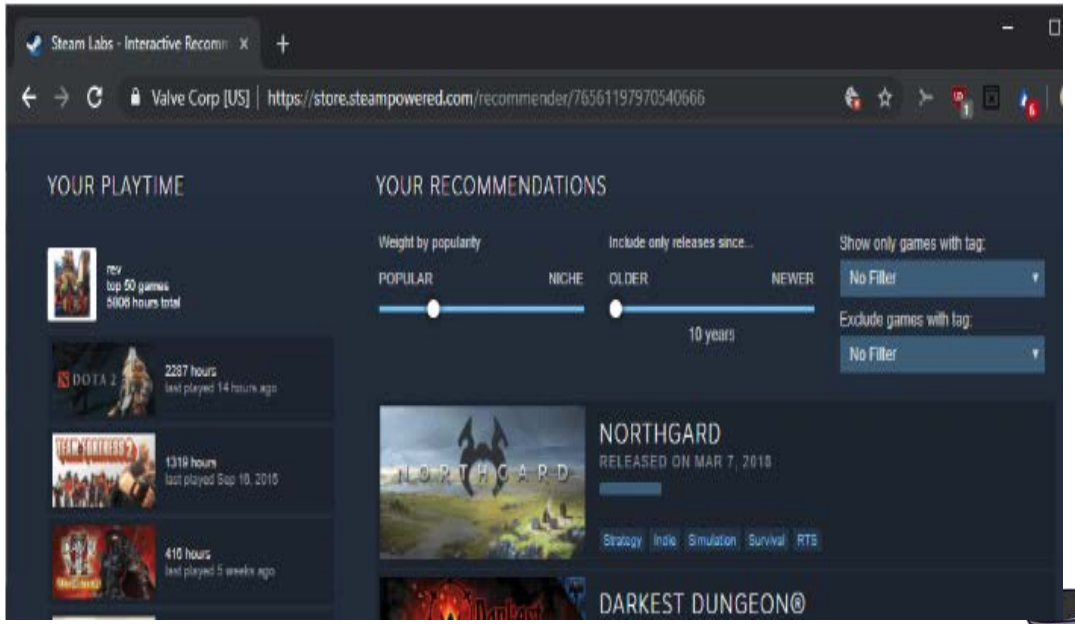
OpenAI Five is the first AI to beat the world champions in an esports game, having won two back-to-back games versus the world champion Dota 2 team, OG, at Finals this weekend. Both OpenAI Five and DeepMind's AlphaStar had previously beaten good pros privately but lost their live pro matches, making this also the first time an AI has beaten esports pros on livestream.

https://openai.com/blog/openai-five-defeats-dota-2-world-champions/

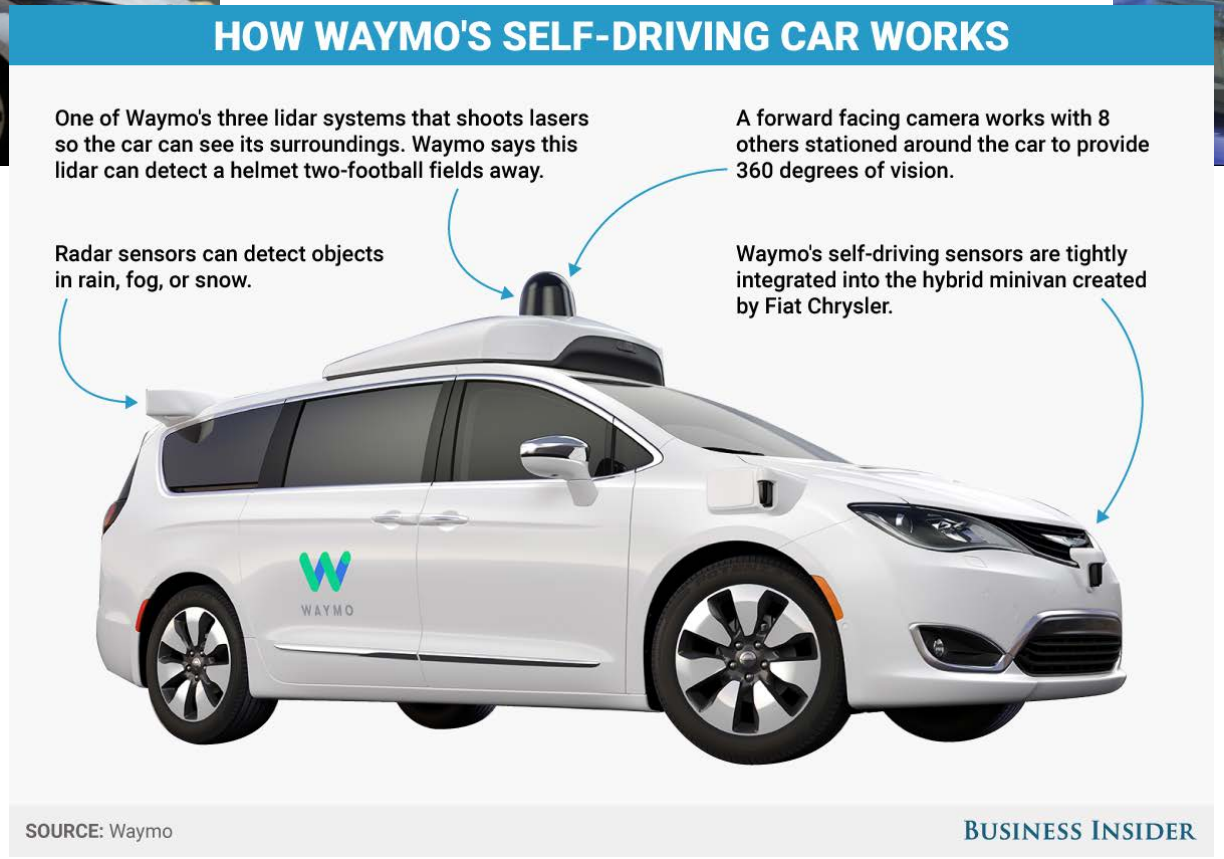
AI in Life: Recommender System

Spotify





AI in Life: Self Driving Car



http://www.businessinsider.sg/how-does-googles-waymo-self-driving-car-work-graphic-2017-1/?r=US&IR=T

VARIOUS AI APPLICATION

- AI for HEALTHCARE
- AI for EDUCATION
- AI for ENERGY INDUSTRY
- AI for MANUFACTURING

Modul: Introduction to AI

What is AI (1): Acting Humanly

KK IF – Teknik Informatika – STEI ITB

Inteligensi Buatan
(Artificial Intelligence)

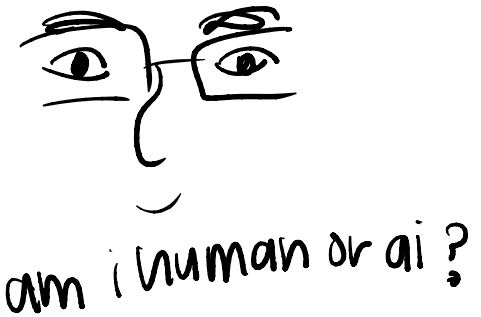
4 Approaches in AI Definition

	Humanly	or	Rationally
Thinking or Acting	Thinking Humanly		Thinking Rationally
	Acting Humanly		Acting Rationally

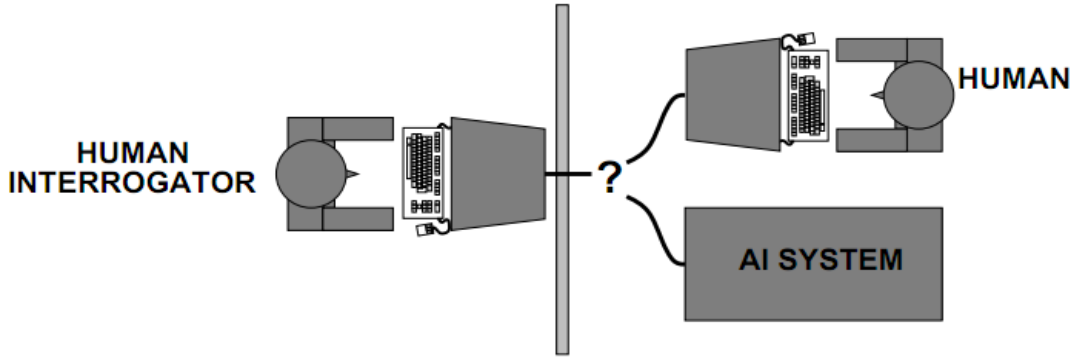
1st Approach of What is AI: ACTING HUMANLY

The art of creating machines that perform functions that require intelligence when performed by people (Kurzweil, 1990)

The study of how to make computers do things at which, at the moment, people are better (Rich and Knight, 1991)



1st Approach of What is AI: Acting Humanly (2)



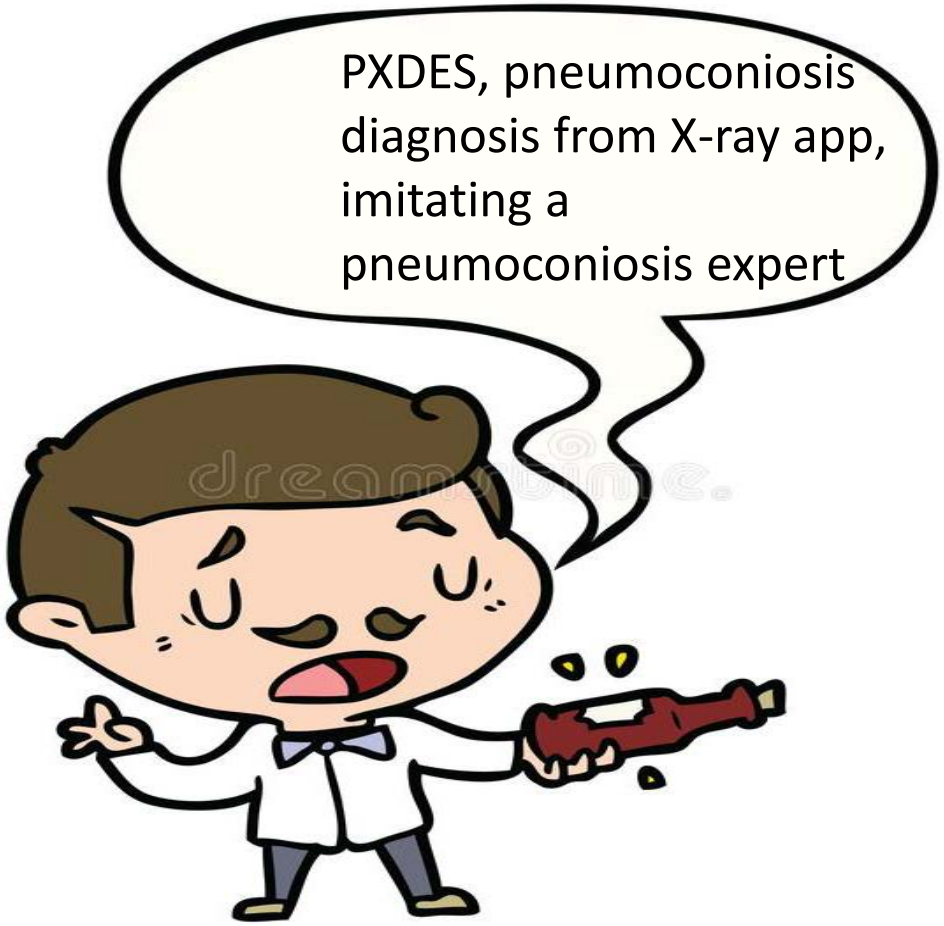
TURING TEST

LOEBNER PRIZE

Annual Turing Test Competition

https://www.pandorabots.com/mitsuku/

Acting Humanly Approach: Applications Examples



Modul: Introduction to AI

What is AI (2): Thinking Humanly

KK IF – Teknik Informatika – STEI ITB

Inteligensi Buatan
(Artificial Intelligence)

4 Approaches in AI Definition

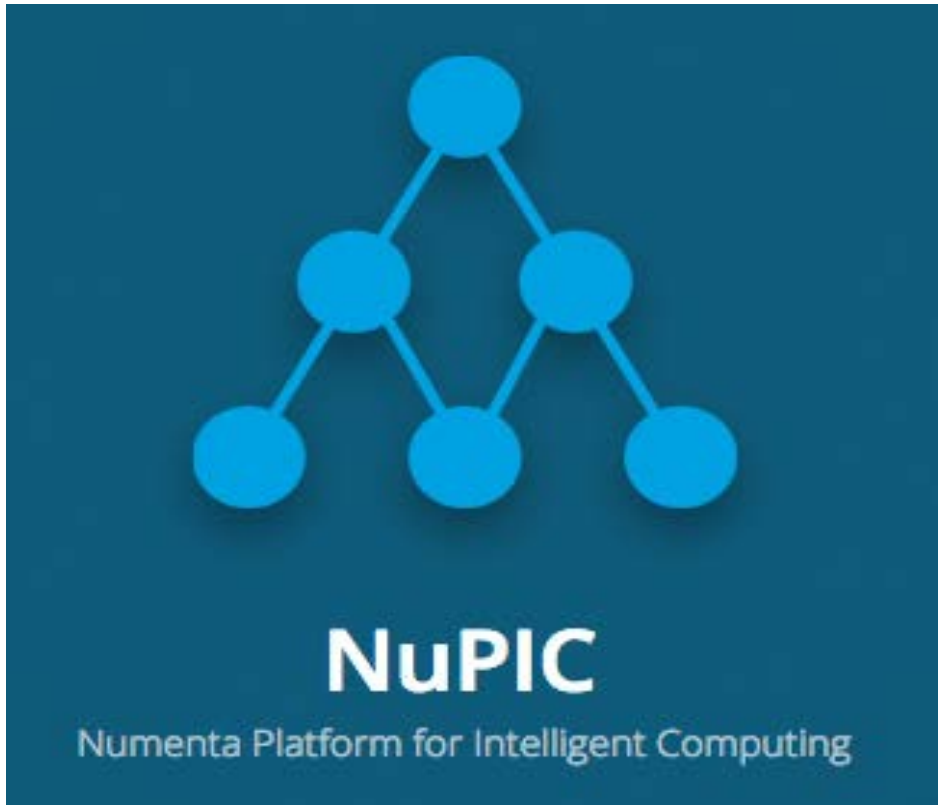
Thinking or Acting	Humanly	Rationally
	Thinking Humanly	Thinking Rationally
	Acting Humanly	Acting Rationally

2nd Approach of What is AI: THINKING HUMANLY

[The automation of] activities that we associate with human thinking, activities such as decision-making, problem solving, learning ... (Bellman, 1978)

The exciting new effort to make computers think ... machines with minds, in the full and literal sense (Haugeland, 1985)

Thinking Humanly Approach: Applications Examples



https://numenta.org/

Sighthound VIDEO
intelligent security camera software

Our patented deep learning computer vision technology teaches your cameras to see video just like the human brain.

https://www.sighthound.com/products/sighthound-video

Modul: Introduction to AI

What is AI (3): Thinking Rationally

KK IF – Teknik Informatika – STEI ITB

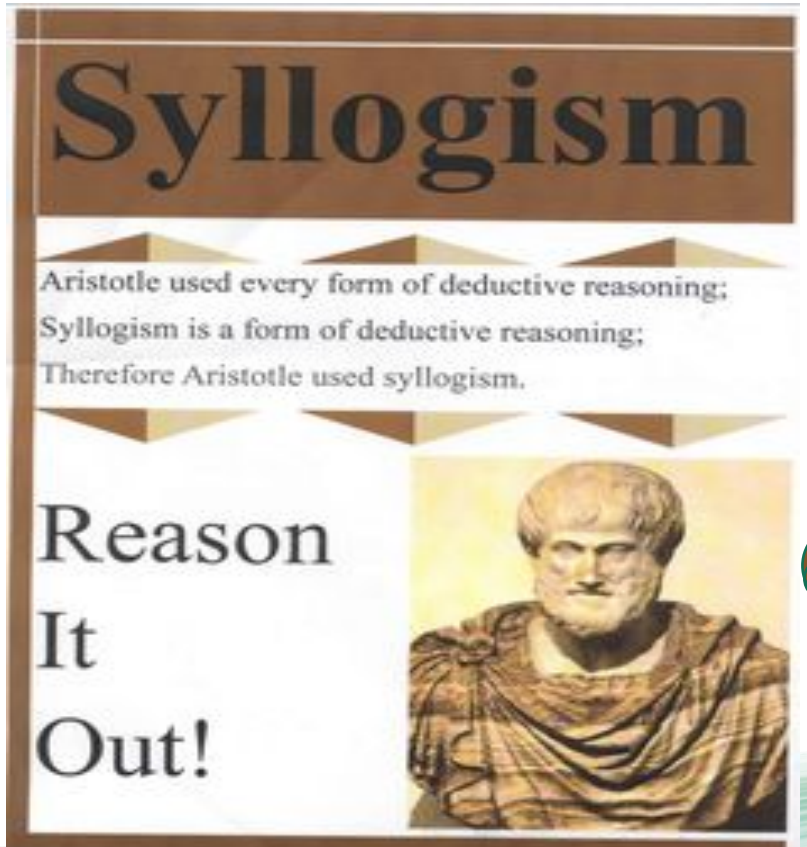
Inteligensi Buatan
(Artificial Intelligence)

4 Approaches in AI Definition

Thinking or Acting	Humanly	Rationally
	Thinking Humanly	Thinking Rationally
	Acting Humanly	Acting Rationally

3rd Approach of What is AI: THINKING RATIONALLY

The study of mental faculties through the use of computational models (Charniak and McDermott, 1985)



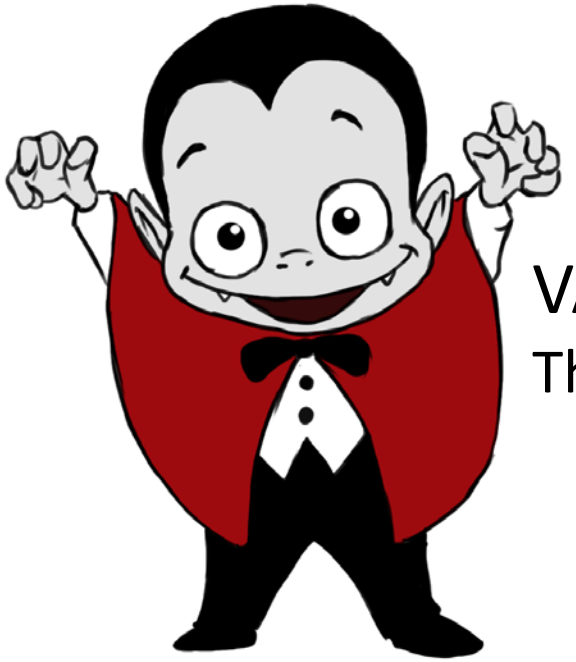
The study of the computations that make it possible to perceive, reason, and act (Winston, 1992)

Thinking Rationally Approach: Applications Examples

LEAN
THEOREM PROVER

Microsoft Research

http://leanprover.github.io/



VAMPIRE
Theorem Prover

https://vprover.github.io/index.html

Modul: Introduction to AI

What is AI (4): Acting Rationally

KK IF – Teknik Informatika – STEI ITB

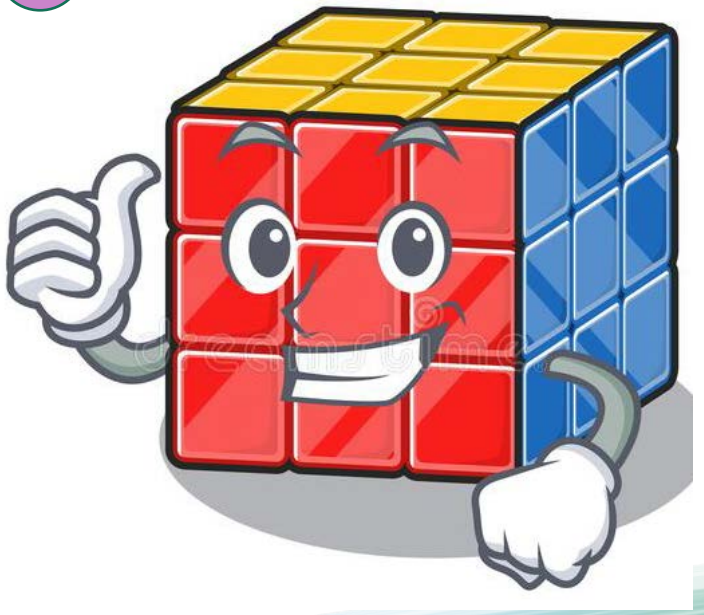
Inteligensi Buatan
(Artificial Intelligence)

4 Approaches in AI Definition

	Humanly	or	Rationally
Thinking or Acting	Thinking Humanly		Thinking Rationally
	Acting Humanly		Acting Rationally

4th Approach of What is AI: ACTING RATIONALLY

Computational intelligence is the study of the design of intelligent agents(Poole et al., 1998)



AI...is concerned with intelligent behavior in artifacts (Nilsson, 1998)

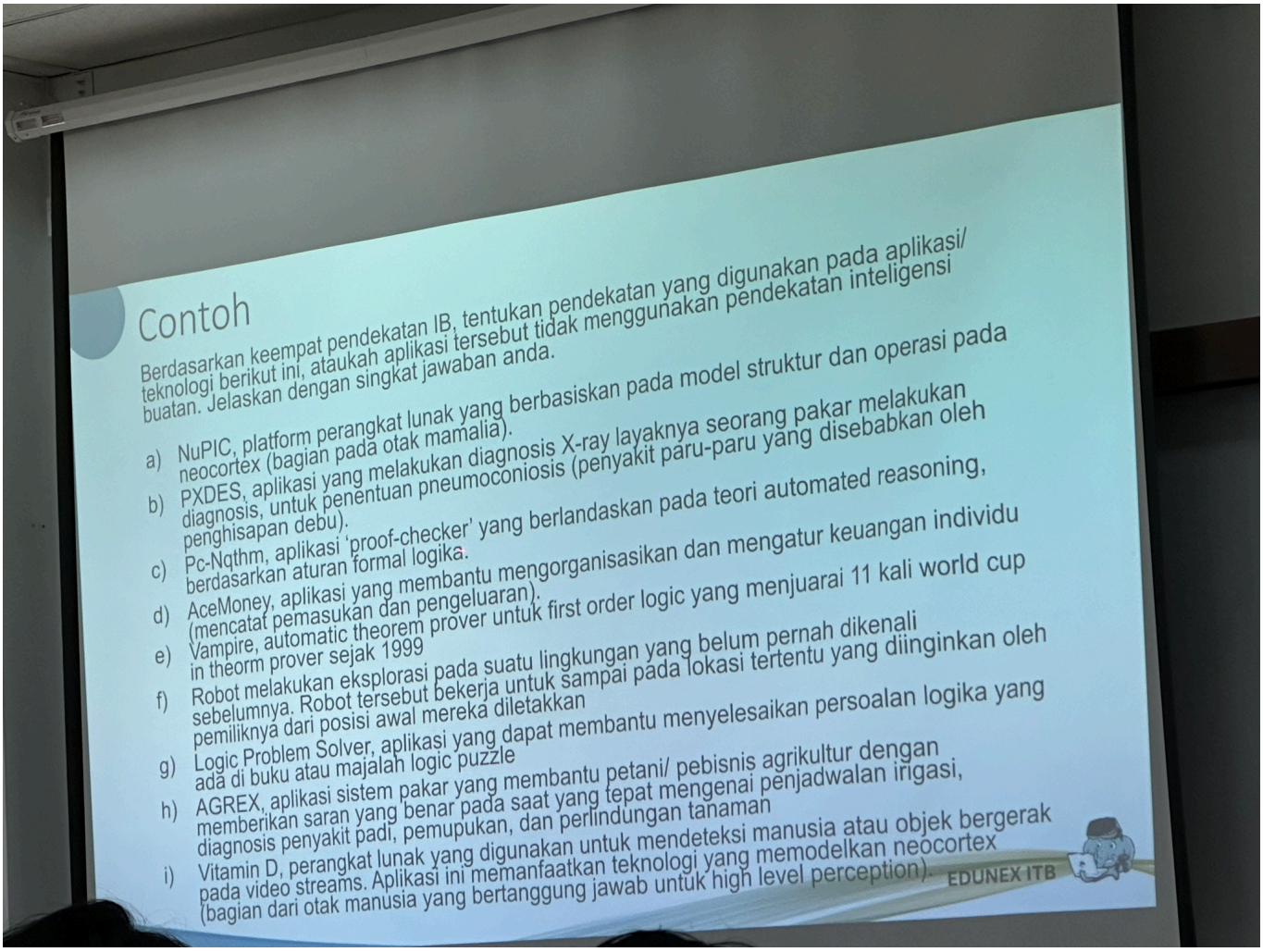
Acting Rationally Approach: Applications Examples

JobShop visual
Visual Production Scheduling

http://www.jobshop.72.sk/?m=0HU

https://blog.dota2.com/?l=english

THANK YOU



d) thinking humanly → meniru otak

b) acting humanly → meniru perilaku pakar

c) thinking rationally

d) bukan AI

e) thinking rationally

f) acting rationally

g) gak dinyatakan menyelesaikan persoalan pake pendekatan apa, bisa aja searching biasa

h) acting humanly

i) thinking humanly

↳ pakai hukum² logika, bgmn sebenarnya manusia berpikir

acting rationally
→ dfs, bfs, a*
→ model matematis yg telah ditemukan sebelumnya.

I love densu <3 thx

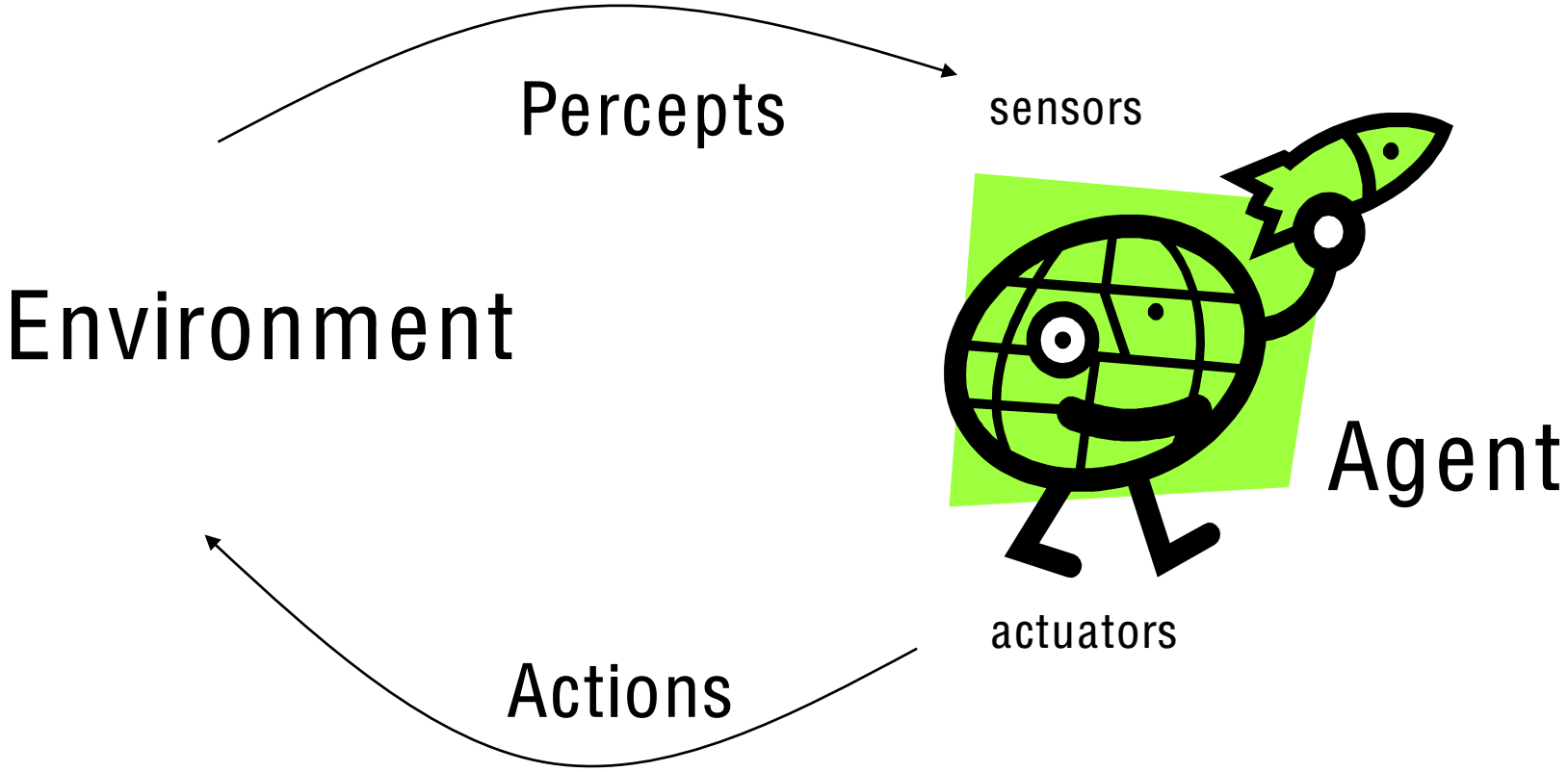
Modul : Intelligent Agent

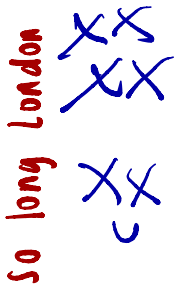
Agent & Environment

KK IF - Teknik Informatika- STEI ITB

Inteligensi Buatan
(Artificial Intelligence)

Agent & Environment





What is Agent?

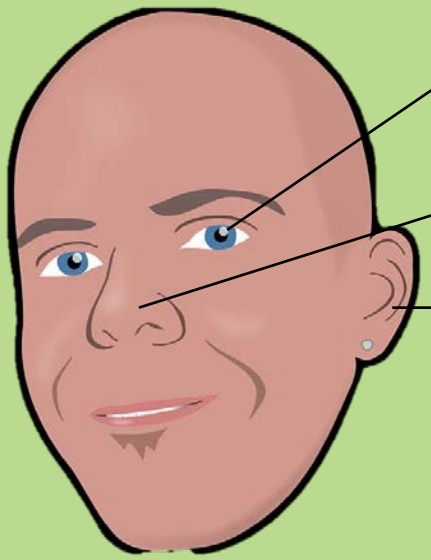
Anything that can be viewed as perceiving its environment through sensors and acting upon that environment through actuators.

- A robot
- A factory
- A web shopping program
- …

Computational agents that behave autonomously

Example: Human Agent

Sensors



Eyes

Nose

Ears

Actuators

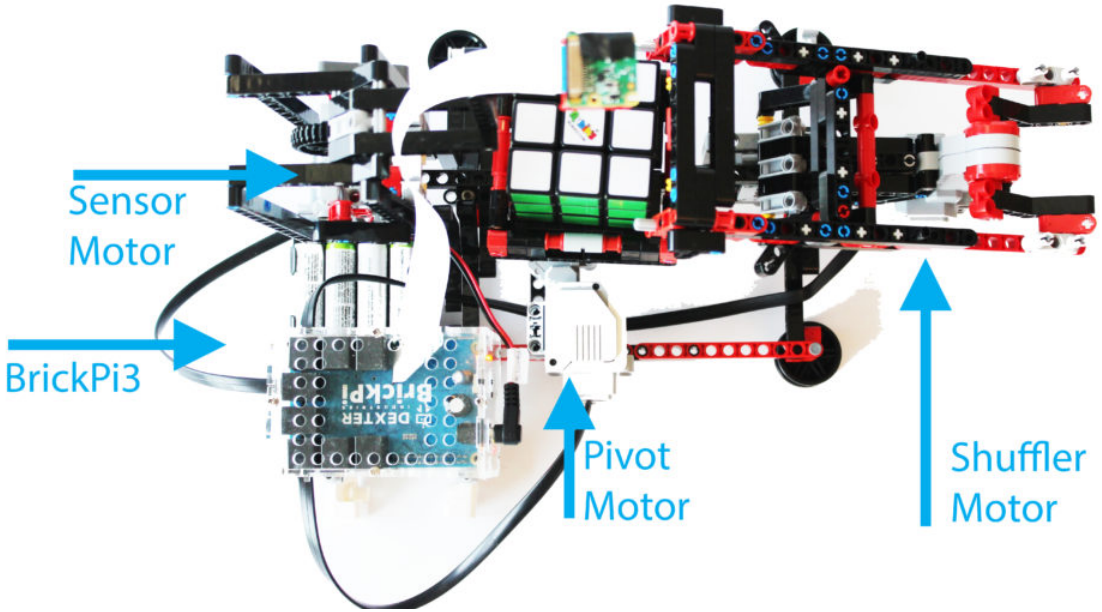
Hands

Legs

Other Example: Rubic Solver Robot Agent

Sensors

Actuators



https://www.dexterindustries.com/projects/brickuber-project-raspberry-pi-rubiks-cube-solving-robot-project/

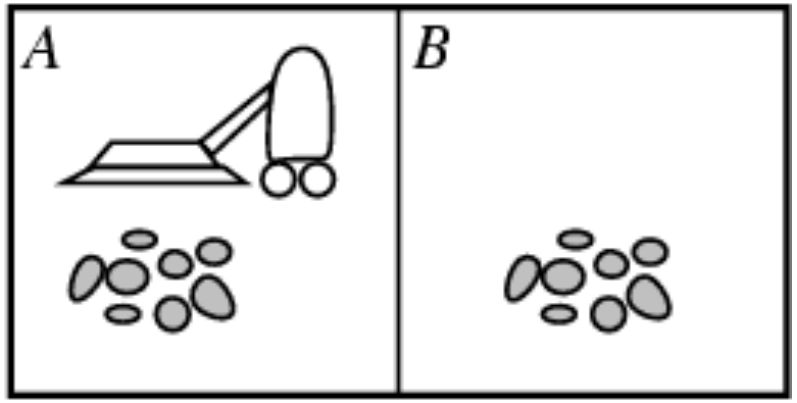
Modul : Intelligent Agent

Agent Model

KK IF - Teknik Informatika- STEI ITB

Inteligensi Buatan
(Artificial Intelligence)

Vacuum-cleaner World



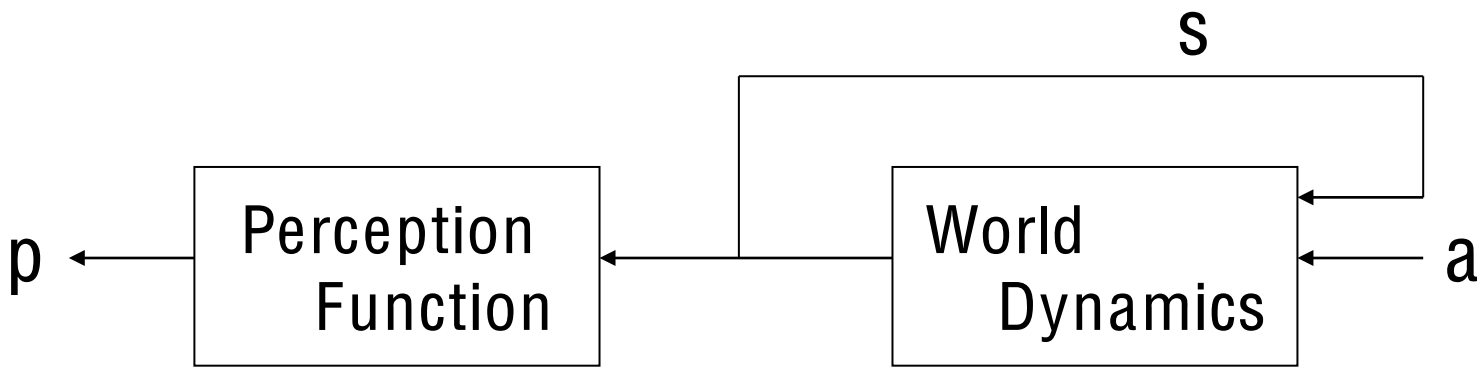
Percepts: something that is perceived by the agent **sensors**
→ location and contents: [A, Dirty]

Action: something that is carried out by the agent **actuators**
→ *Left*, *Right*, *Suck*, *NoOp*

intelligent agent = cuman lakuin yg udah di program aja, kalau dia bikin yg lain = rusak

How to Model the Agent World ??

- Percept Space
- Action Space
- Internal State
- Perception Function: S → P
- World Dynamics: S x A → S



Agent Design

Utility Function (U):
S → real
S* → real

Problem:
Find P* → A

Mapping sequence of percept (P*) to action (A)

Maximize U
(sequences of states S)

Remember:
S x A → S

Modul : Intelligent Agent

Rational Agent

KK IF - Teknik Informatika- STEI ITB

Inteligensi Buatan
(Artificial Intelligence)

Rational Agent

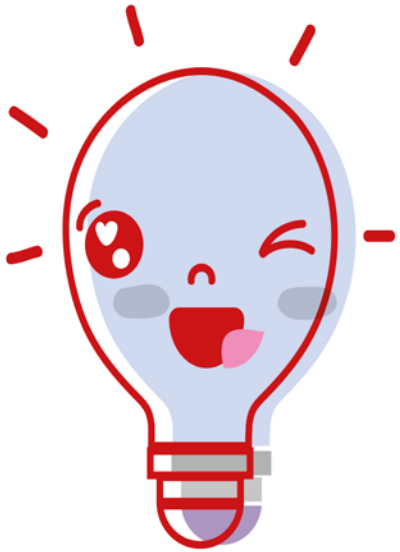
Strive to **DO THE RIGHT THING**

- Based on what it can perceived
- Based on what it can perform

Performance Measure:
Objective Criterion for Success of an Agent's behaviour

Rational Agent (2)

Sequences of Percepts



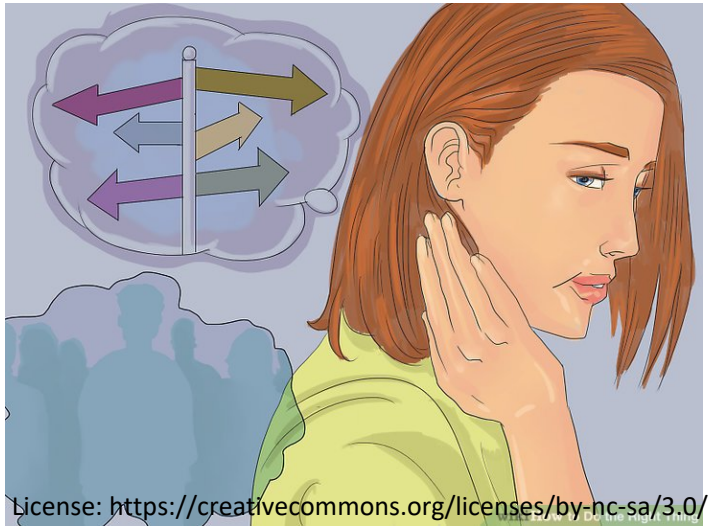
https://www.vecteezy.com/free-vector/kawaii

Actions

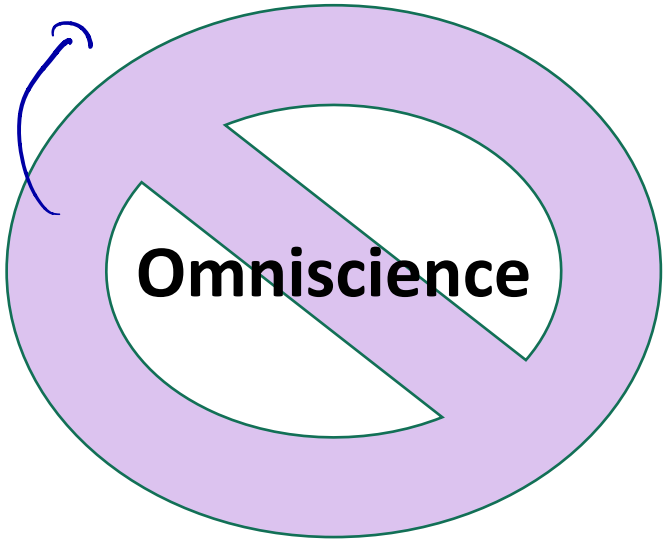
Performance Measurement

“Knowledge”

Rationality



blm tentu tau semua, berganung sama info yg kita kasih



Limited Rationality

Rationality limitation:
Computational Constraint

Agent Design

Problem:
Find P* → A

Mapping sequence of percept (P*) to action (A)

Maximize U
(sequences of states S)
Subject to Computational Constraints

P E A S

THANK YOU

Modul : Intelligent Agent

P E A S

KK IF - Teknik Informatika- STEI ITB

Inteligensi Buatan
(Artificial Intelligence)

P E A S

Performance Measure

Environment

Actuators

Sensors

By: Strader

Example: Designing an Automated Taxi Driver

http://www.gettyimages.com/detail/83988175/Stone

- **P** — Safe, fast, legal, comfortable trip, maximize profits
- **E** — Roads, other traffic, pedestrians, customers
- **A** — Steering wheel, accelerator, brake, signal, horn
- **S** — Cameras, sonar, speedometer, GPS, odometer, engine sensors, keyboard

Example: Medical Diagnosis System Agent

P

Healthy patient, minimize costs, lawsuits

E

Patient, hospital, staff

S

Keyboard (entry of symptoms, findings, patient's answers)

A

Screen display (questions, tests, diagnoses, treatments, referrals)

Modul : Intelligent Agent

Task Environments

KK IF - Teknik Informatika- STEI ITB

Inteligensi Buatan
(Artificial Intelligence)

Task Environment

Fully vs Partially Observable

fully : semua kondisi environment diketahui

partially : ada kondisi env yg tdk diketahui

Deterministic vs Stochastic

udh pasti partially

d : aksi yg menghasilkan hal pasti

s : hasil aksi berupa probabilitas

ex : main kartu, kita gtw next nya kartu orang berikan apa

Known vs Unknown

pengetahuan agent dlm melakukan aksi

agent tdk punya pengetahuan dlm melakukan aksi selanjutnya meski lingkungannya observable

Episodic vs Sequential

episodic : aksi sekarang gk mempengaruhi aksi selanjutnya

sequential : aksi skrg mempengaruhi yg selanjutnya

ex : main catur

Static vs Dynamic

static : kondisi environmentnya ga berubah

dynamic : kondisi environment berubah selama agent nya mikir

ex : autonomous driving, eh ada yg nyebrang, tiba2 red light

Semidynamic

environmentnya sbnrnya ga berubah tp ada pertimbangan luar yg bikin agent jd bikin hal yg

Discrete vs Continuous

discrete : lingkungannya bisa dipisah (statenya pasti)

continuous : tdk bisa dipisahkan secara discrete lingkungannya

cth : automated driver (lingkungannya beda2 terus)

Single vs Multi Agent

single : lingkungan berubah hanya berdasarkan aksi

multi agent : lingkungan berubah karena ga cuman aksi tapi ada faktor luar

Examples

	Chess with a clock	Chess without a clock	Taxi driving
Fully vs Partially Observable	Fully	Fully	Partially
Deterministic vs Stochastic	Deterministic	Deterministic	Stochastic
Episodic vs Sequential	Sequential	Sequential	Sequential
Static vs Dynamic	Semidynamic	Static	Dynamic
Discrete vs Continuous	Discrete	Discrete	Continuous
Single vs Multi Agent	Multi Agent	Multi Agent	Multi Agent

Agent Structure

Architecture → **Computing Devices +Sensors + Actuators**

+

Program: Percept* → Actions

- **Simple Reflex Agent**
  spontan (kondisi if-then)
  tdk menyimpan info apapun
  cth : lgsg menghindar pas mau ketabrak mobil
- **Model-based Reflex Agent**
  tdk semua diberikan, ada informasi lingkungan yg hrs disimpan & dibentuk jd model
- **Goal-based Agent**
  mempertimbangkan goal selain persepsi
- **Utility-based Agent**
  ada bbrp aksi, hrs pilih aksi yg memaksimalkan nilai utilitas
- **Learning Agent**
  menilai aksi & memilih aksi yg nilainya plg besar, ada jg generator utk menilai aksi dia & agent aksi lain yg pernah dilakukan

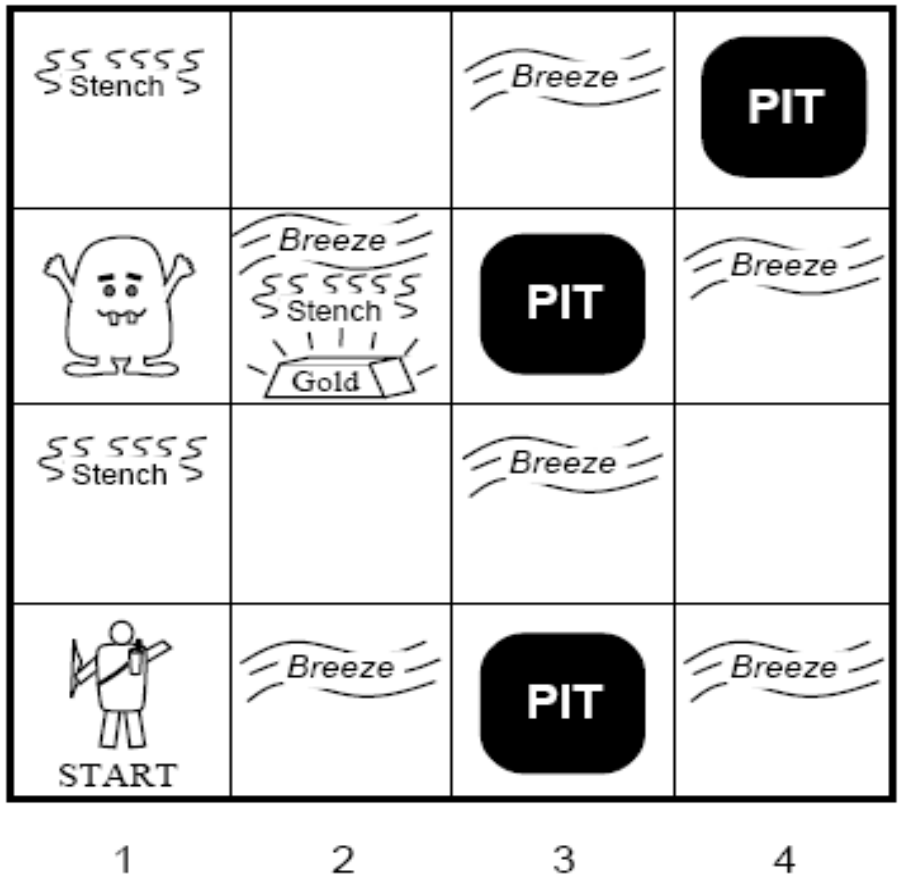
Modul : Intelligent Agent

Agent Level

KK IF - Teknik Informatika- STEI ITB

Inteligensi Buatan
(Artificial Intelligence)

Wumpus World



Performance Measure: gold +1000, death -1000, -1 per step, -10 for using the arrow

Environment: cave, rooms, Wumpus, gold

Actuators: motor to move Left, Right, Forward, hands to Grab, Release, and Shoot arrow

Sensors: sensor to capture [Stench, Breeze, Glitter, Bump, Scream]

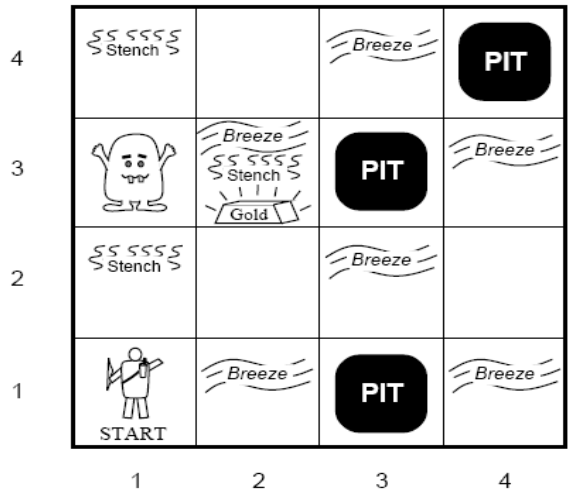
Level 1: Problem Solving Agent

agent punya informasi state scr lengkap

Agent has information about all of the states in Wumpus World

Agent has to 'search', the path that can lead agent to the goal, as fast as possible

There are many searching algorithms, and each algorithm is suitable for certain problem



DFS, BFS, IDS, UCS
A*, Greedy Best First,
Minmax search,
Genetic Algorithm, Hill Climbing,
Simulated Annealing,
Etc...

Level 2: Knowledge Based Agent

yg diberikan ke agent hanya knowledge saja

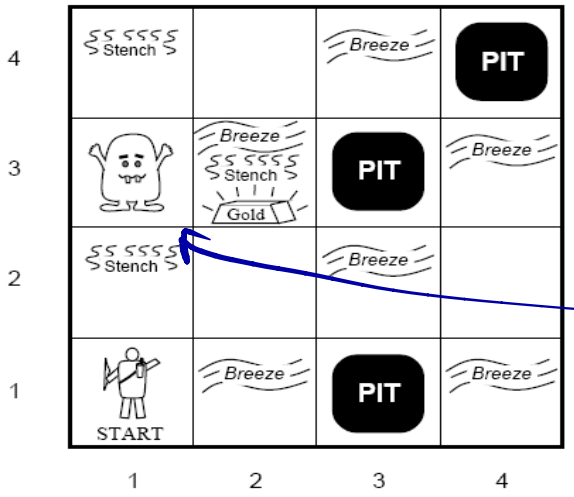
agent gapunya info state lengkap,

jadi tiap ada state baru dia hrs mikir based on hal" yg udh diketahui

Agent doesn't have information about all of the states in Wumpus World. It only has 'basic knowledge/ premises'

When agent percept a state in a room, it will try to reason new facts/ states, this is how agent will step by step collecting all of the states of wumpus world in order to achieve its goal

Reasoning has to be done by Agent→ by deducting the premises with percepted fact.



contoh : if ada bau, maka ada wumpus

ini ya wumpus

Level 3: Learning Agent

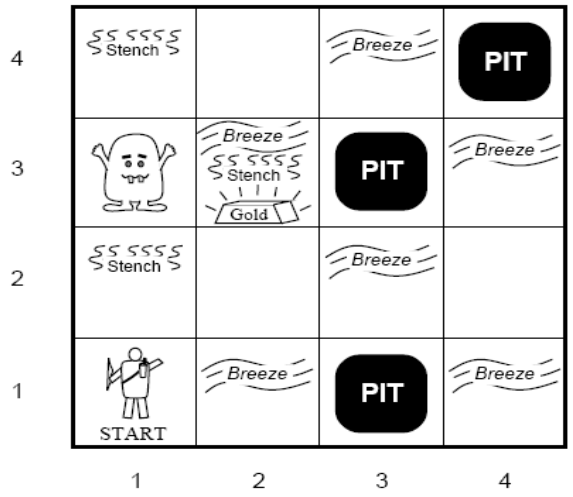
bodoh di awal, pintar di akhir

agent gapunya state lengkap & gapunya knowledge jd agent belajar bnr" dr tiap aksi yg dilakukan di awal, agent bnr" gatau apa"

Agent doesn't have the information of all of the states and doesn't even have the basic knowldge of the wumpus world

Agent plays several times (perhaps dies several times) → The **observation data** from playing several times is the 'input' for learning process

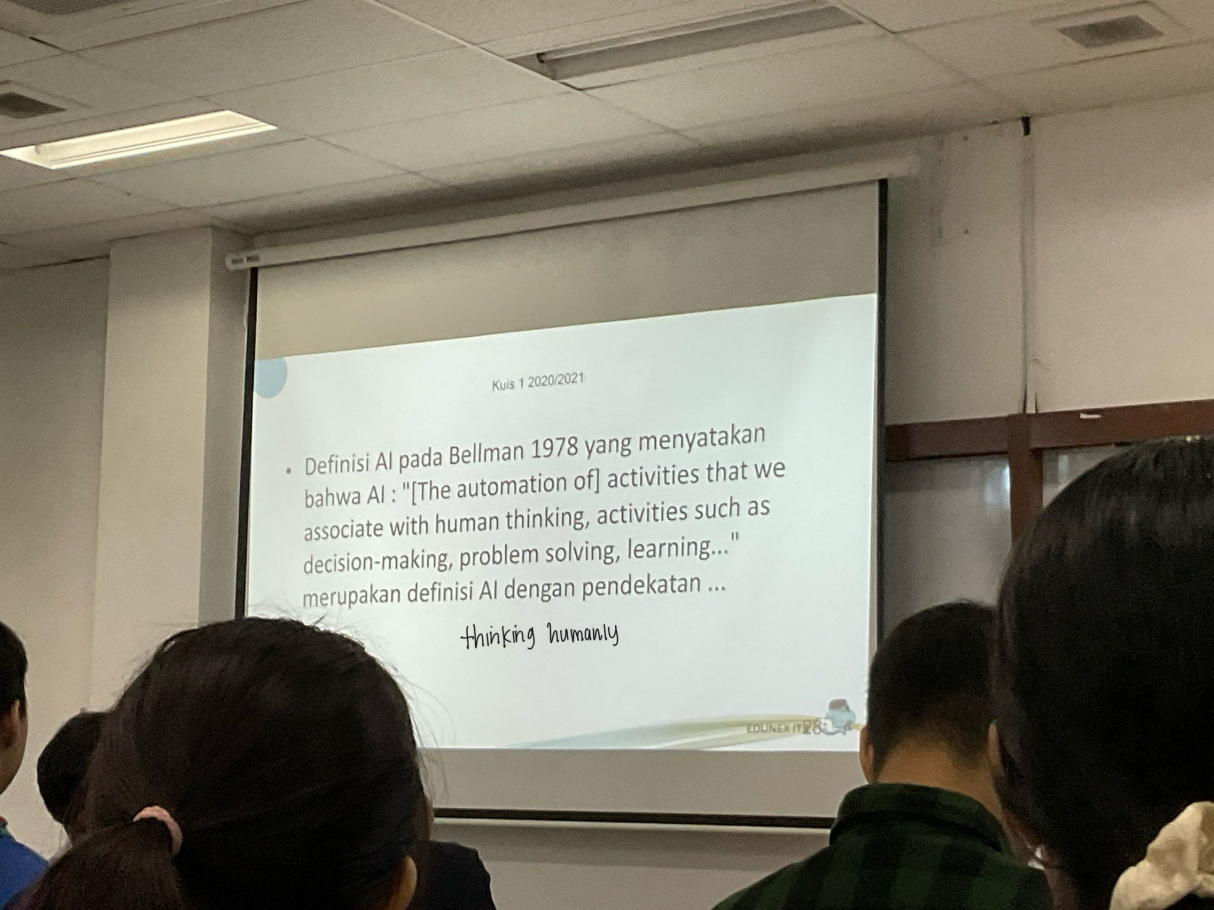
The result of the learning process, agent will have basic knowledge, e.g. Squares adjacent to pit are breezy

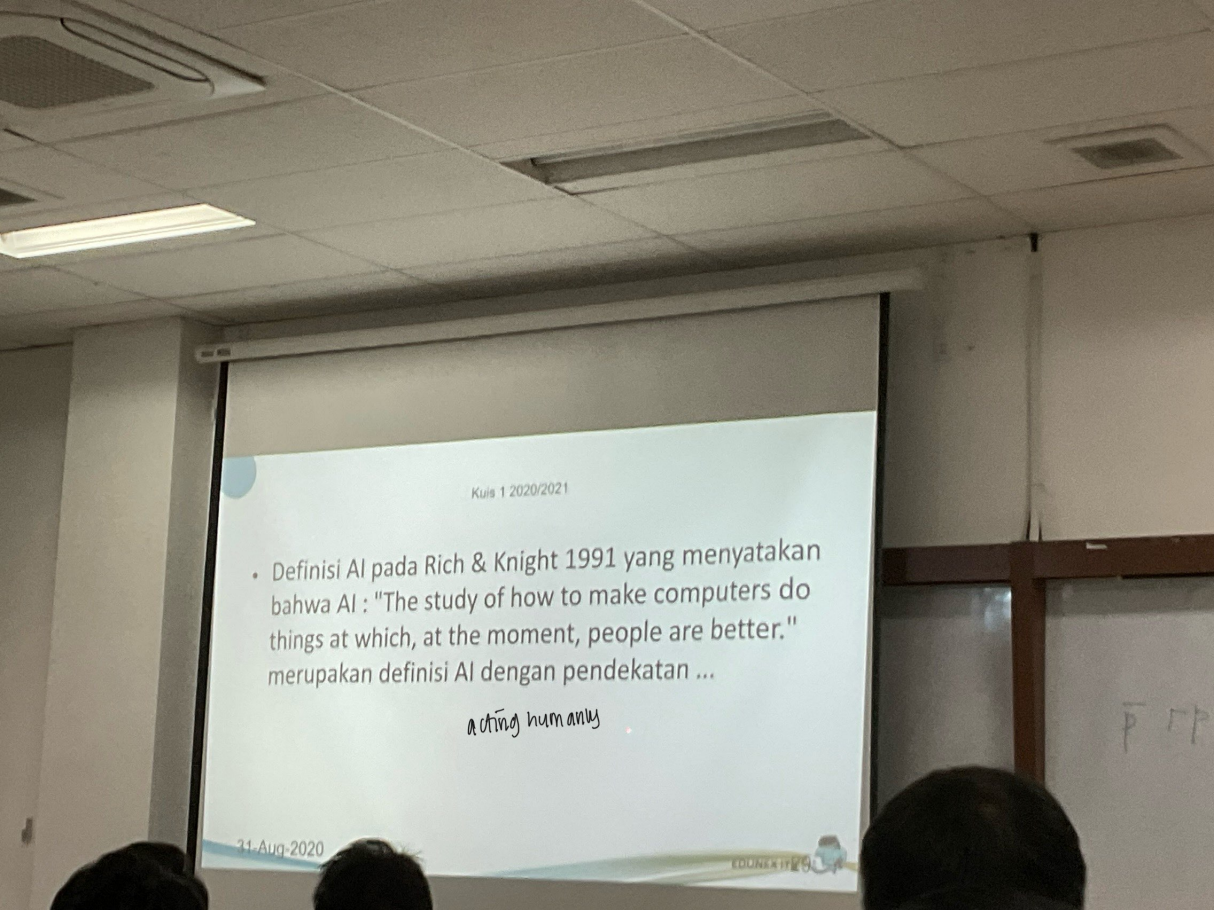


There are many learning algorithms, that suitable for certain purposes, and the 'availability' of the data/ feedback

- Supervised learning
- Unsupervised learning
- Reinforcement learning

THANK YOU


Kuis 1 2020/2021
Definisi AI pada Bellman 1978 yang menyatakan bahwa AI : "[The automation of] activities that we associate with human thinking, activities such as decision-making, problem solving, learning..." merupakan definisi AI dengan pendekatan ...
thinking humanly


Kuis 1 2020/2021
• Definisi AI pada Rich & Knight 1991 yang menyatakan bahwa AI : "The study of how to make computers do things at which, at the moment, people are better." merupakan definisi AI dengan pendekatan ...
acting humanly
31-Aug-2020

No	Kalimat	Benar/ Salah (Nilai 1)	Alasan (Nilai 2)
a	Sebuah aplikasi chatbot harus dapat berpikir seperti manusia agar lolos Turing Test. → acting humanly	= salah	
b	Persoalan teka-teki logika yang diselesaikan dengan memanfaatkan algoritma Backtracking Search, merupakan contoh aplikasi berbasis AI dengan pendekatan thinking reationally. → Logika (law of thoughts)	: salah,	dia acting rationally
c	Bahasan Intelligent Agent dalam kuliah ini adalah agen yang bisa bersifat rational, artinya agen yang bekerja pada lingkungan tugas dengan properti 'partially observable' tidak akan bisa bersifat rational. ↳ tetap bisa bersifat rasional	: salah	

Terdapat sebuah *intelligent agent* yang dibangun sebagai aplikasi web untuk membantu pembuktian suatu teorema matematika, dengan langkah sesedikit mungkin dan waktu secepat mungkin. Pembuktian ini memanfaatkan kaidah-kaidah inferensi yang sudah terdefinisi dalam domain matematika. Tentukan lingkungan tugas (task environment) PEAS dan 6 properti lingkungan tugas dari agen tersebut, dengan mengisi tabel berikut ini. Jawaban disertai alasan dengan singkat.

P : Ketepatan cepat, langkah
performance measure

E : orang (dia yg masukin teorema), web, kaidah² inferensinya
environment

A : layar, modul yg menerapkan kaidah inferensinya untuk membuktikan teorema math
actuator

S : keyboard
sensor

fully (di soal dh ketatan semua kaidah² yg dh tau)

deterministic (hasilnya udh pasti)

sequential (bisa pakai kaidah² sblmnya)

~~static (pas agent mikir, lingkungannya ga berubah)~~

semidynamic karena harus secepat mungkin

diskrit (lingkungan bs dipisah² tahapannya jelas, pas di apply kaidah udh pasti hasilnya)

single

EQUINEXITB

Lingkungan Tugas	Jawaban dengan alasan singkat
P	Pemilihan dan penerapan kaidah tepat, waktunya secepat mungkin dengan langkah sesedikit mungkin
E	Web, orang yang akan membuktikan teorema, kaidah
A	Modul inferensi yang dilengkapi dengan kaidah-kaidah, display untuk state antara dan hasil
S	Keyboard atau touch screen, bergantung asumsi device yang digunakan.

Masayu Leylia Khodra

KK IF – Teknik Informatika – STEI ITB

Modul 3: Beyond Classical Search

Classical vs Local Search

Inteligensi Buatan
(*Artificial Intelligence*)

Classical Search

transisi state → kalau ada aksi, selama bisa sampe goal, dia pindah state

Problem

semua state sudah diketahui

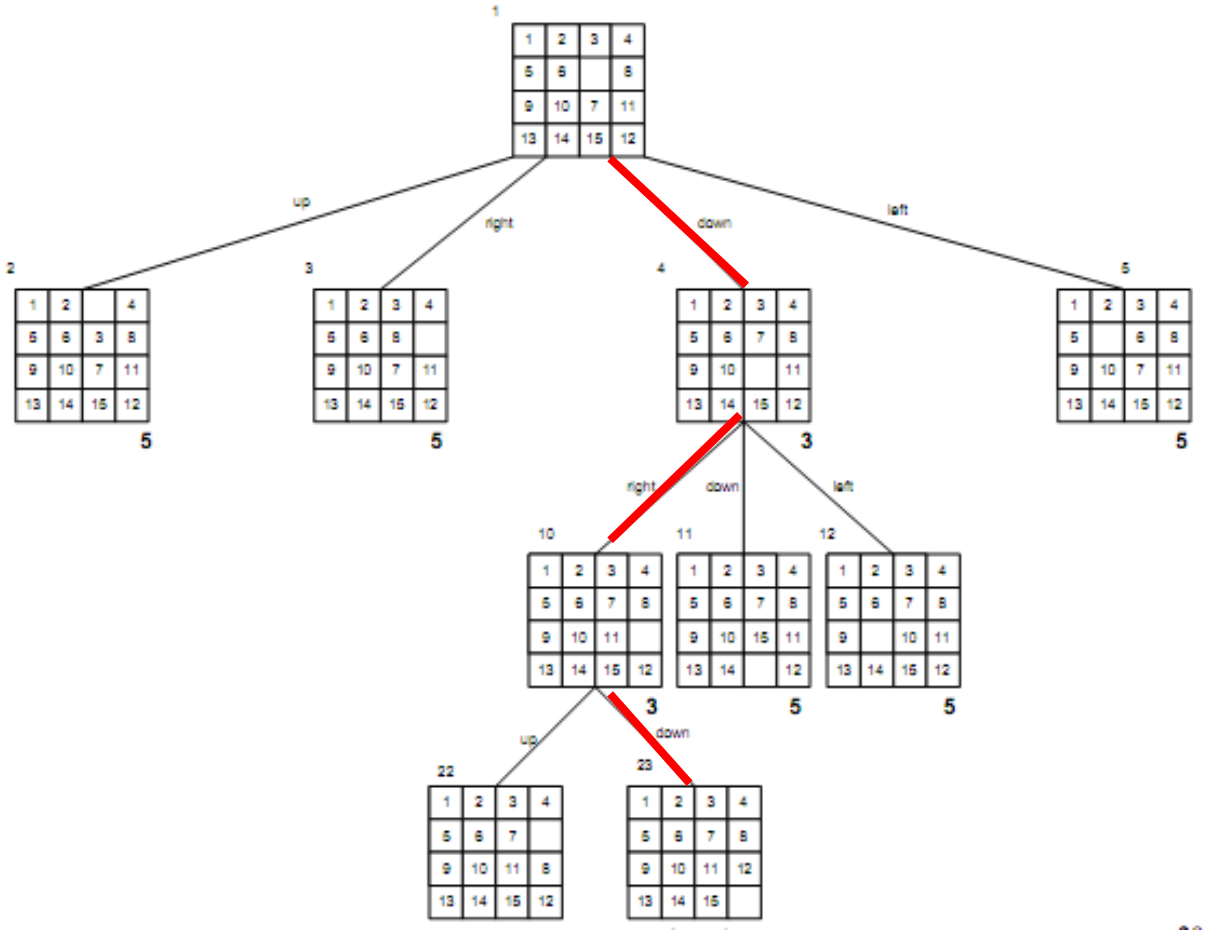
Known environment (observable, deterministic)
States (initial), operators, path cost, goal test

Explore search space systematically

Solution (path to goal)

Solution: sequence of actions

N-Puzzle Problem



Solution: down → right → down

Path to Goal as Solution

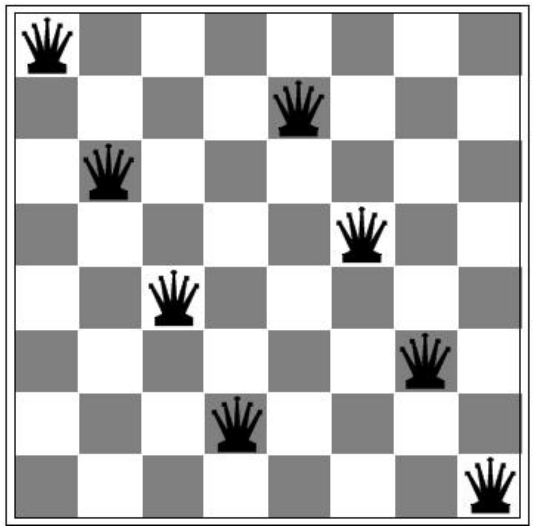
- In many problems, the path to goal is irrelevant.
- Solution: X=(8,6,4,2,7,5,3,1)
- Q1=8 → Q2=6 → … → Q8=1

path relevant : path jadi bagian dari solusi
contoh : rubiks, water jug.

path irrelevant : path dari initial state ke goal state tidak menjadi bagian dari solusi. contoh : N-queens, graph-coloring, knapsack, cryptarithmetic (contoh huruf mana aja dulu doesn't matter)

variable : setiap karakter
↳ nilai yg diassign itu angka

N-Queen Problem



objective : menempatkan queen ke bidak agar tidak bisa saling makan.

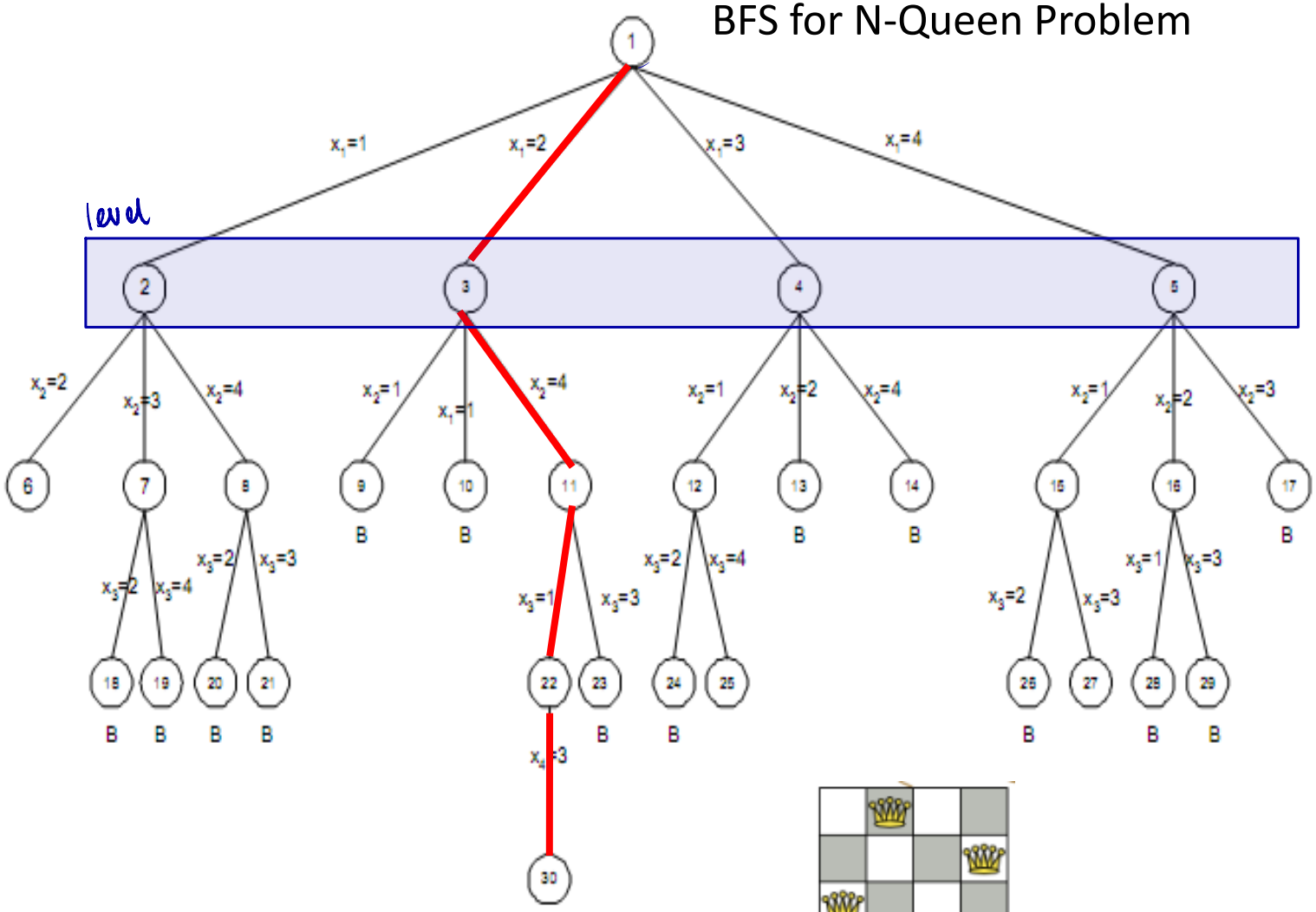
variabel : queen1, queen2, ... queen-N.
↳ nilai yg di assign ke variabel itu posisinya.

n-queens are on the board, none attacked. one queen per column

Path to Goal as Solution (2)

- For example, in n-queen
  - BFS solution: x1=2 → x2=4 → x3=1 → x4=3
- What matters is the final configuration of queens, not the order in which they are added.

BFS for N-Queen Problem



First solution 4-queen: X=(2,4,1,3)

Local Search

Jika path ke goal nya TIDAK PENTING (path irrelevant)

pada initial state, complete configuration → tiap variable sudah diisi nilai secara random, pasti ada nilai dlm state. tiap statenya jd perbaiki nilai var nya smpe dh ga ada yg melanggar constraints

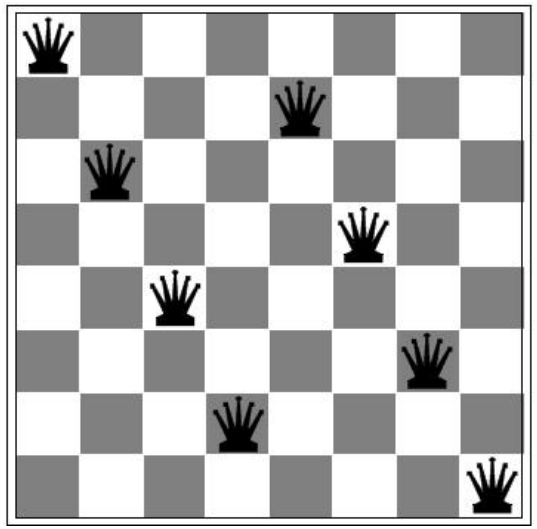
- If path to goal does not matter, we might consider different class of algorithms.
- Local search: complete state formulation
  - State: "complete" configurations
  - Keep single current state, not paths.
  - Action: move only to neighbors of current state.
  - No path cost → state value: value according to objective function or heuristic cost function
  - No goal test → maximum state value (berhenti pas nilai statenya sudah maksimum)
  - Solution: final state.

Problem
States (initial), neighbors
Path to goal is irrelevant

Find the best state by moving to neighbors of current state

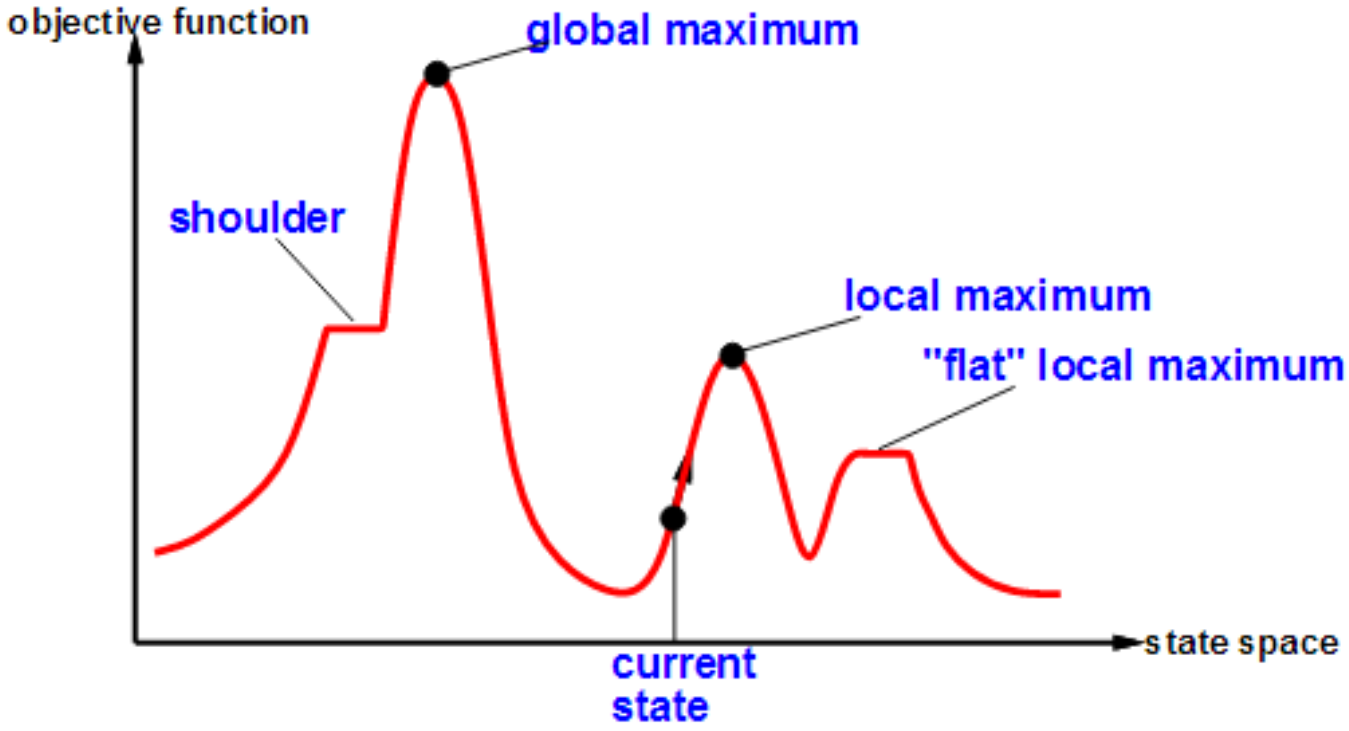
Solution
Final state/configuration

N-Queen: Classical vs Local Search



Classical search	Local search
State: any arrangement of **0 to n** queens on the board (incremental)	State: any arrangement of **n** queens on the board (complete)
Initial state: no queens on the board. Goal test: n-queens are on the board, none attacked	Initial state: a [random] state. Goal test: state value = global maximum
Action: add a queen to any empty square	Action: move a single queen to another square (neighbor)
Solution: path to goal	Solution: final state

Local Search Explore State-space Landscape



A one-dimensional state-space landscape in which elevation corresponds to the objective function. The aim is to find the global maximum. (Russel & Norvig, 2010)

- A landscape has "location" (defined by state) and "elevation" (value of objective or heuristic cost value).
- Local search aims to find global optimum.
- Problem: depending on initial state, can get stuck in local optimum.

Summary: Local Search

Keep single current state
State: "complete" configurations
Complete formulation

Find the best state (global optimum)
Action: move to neighbors

Path to goal is irrelevant
Solution: final state

Next:

- State
- Successor
- Neighbors

Masayu Leylia Khodra

KK IF – Teknik Informatika – STEI ITB

Modul 3: Beyond Classical Search

State: Value, Successor, and Neighbor

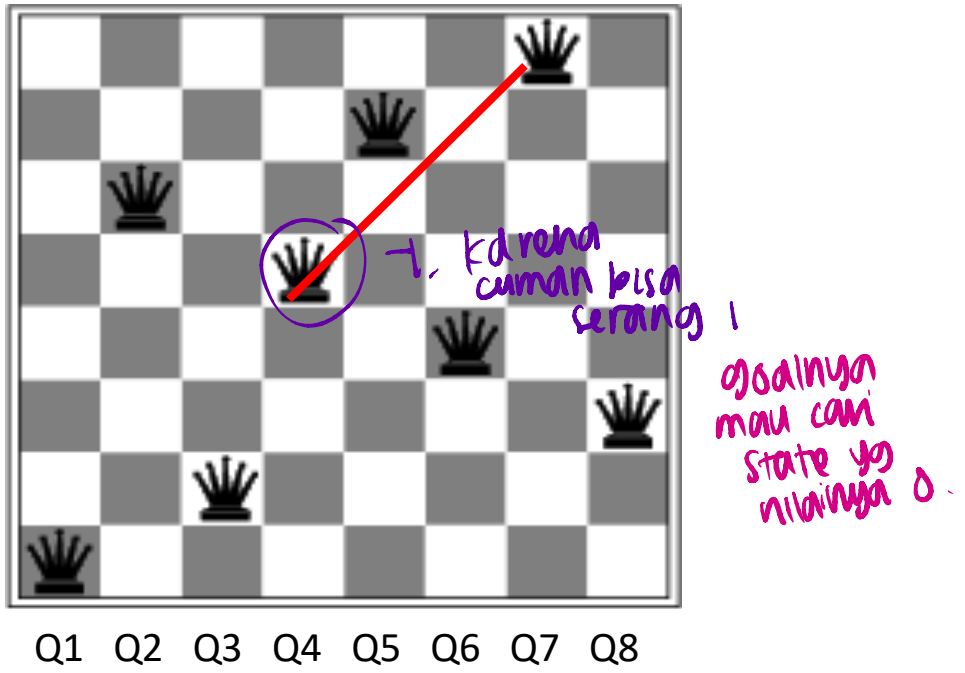
Inteligensi Buatan
(*Artificial Intelligence*)

Local Search: State

Keep single current state
State: "complete" configurations (one queen per column)
Solution: final state

Find the best state (global optimum)
Action: move to neighbors

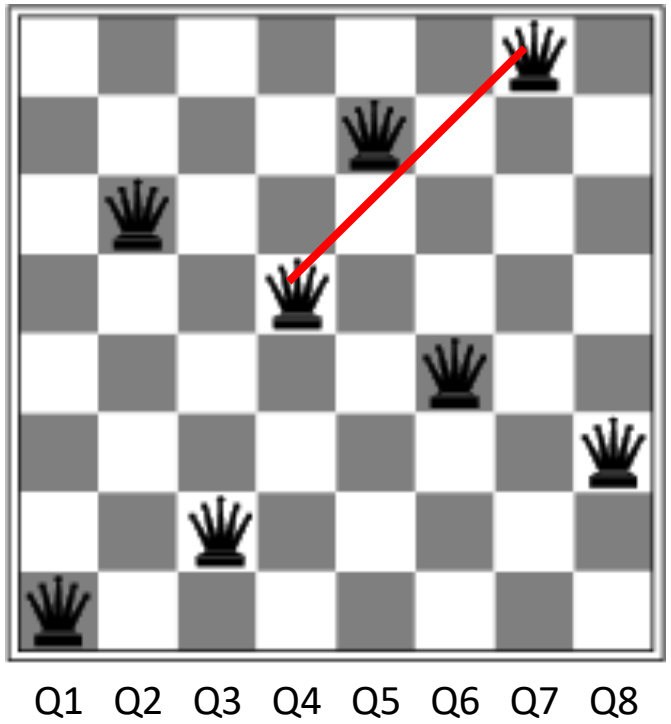
State for 8-queens problem



Each state has state value based on heuristic cost function.

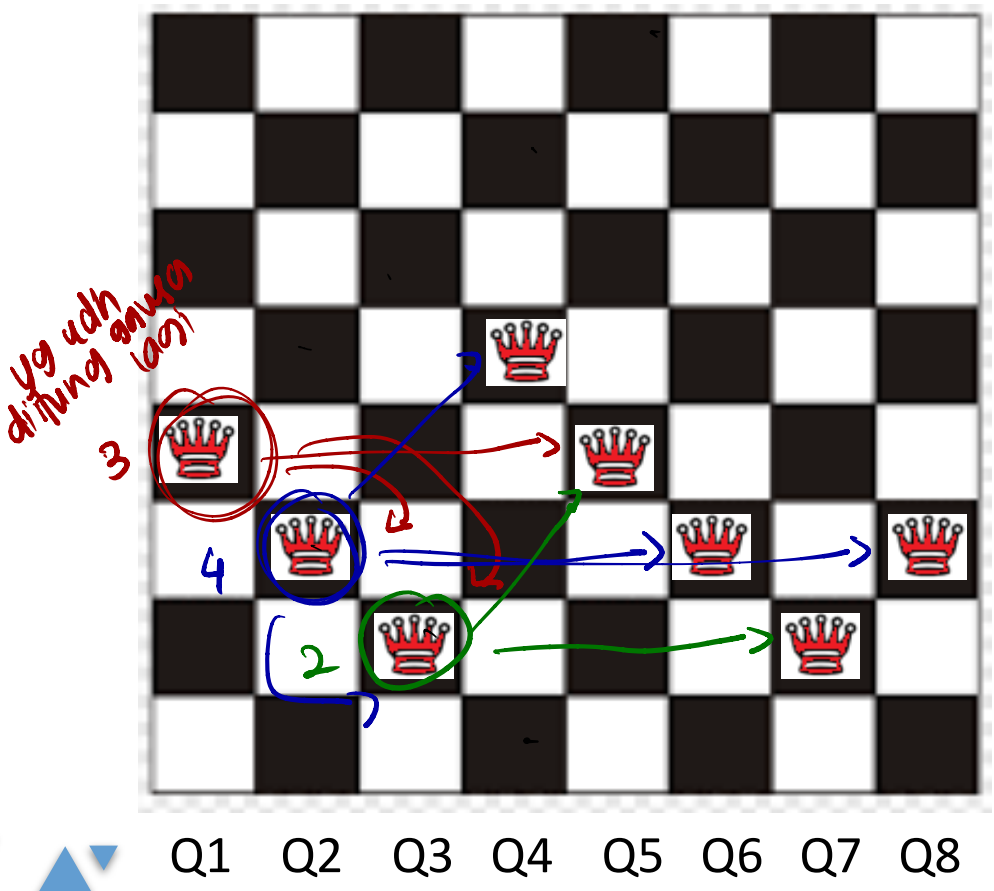
State Value (h) for 8-queens problem

- *h* = - number of pairs of queens that are attacking each other, either directly or indirectly.
- h=-1: only 1 pair of queens (Q4 attacks Q7)
- Optimum solution has global maximum, h=0.



h = *-1* for the above state

Exercise: Determine h



List pairs of queens that are attacking each other, either directly or indirectly.

h = - number of pairs of queens that are attacking each other, either directly or indirectly.

ada 56 tetangga yg mungkin
(1 ratu bs punya 7 kondisi lain)

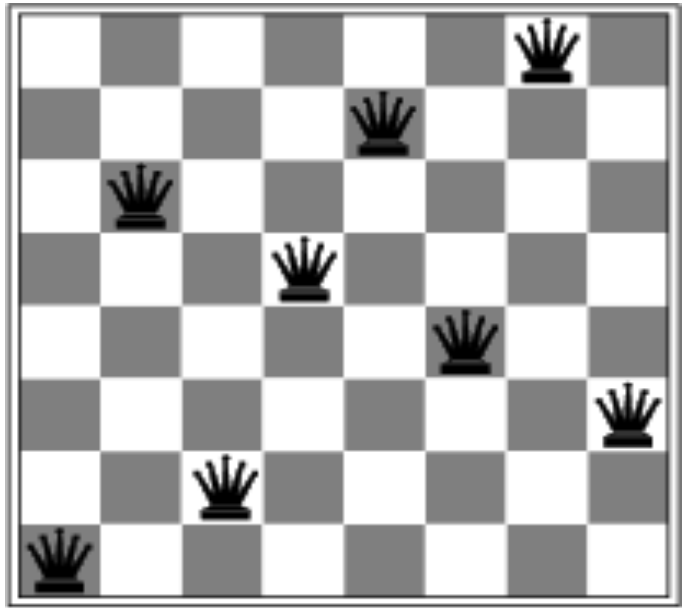
Local Search: Action

Keep single current state
State: "complete" configurations (one queen per column)
Solution: final state

Find the best state (global optimum)
Action: move to neighbor

What is neighbor of a state ?

Neighbor and Successors for 8-queens



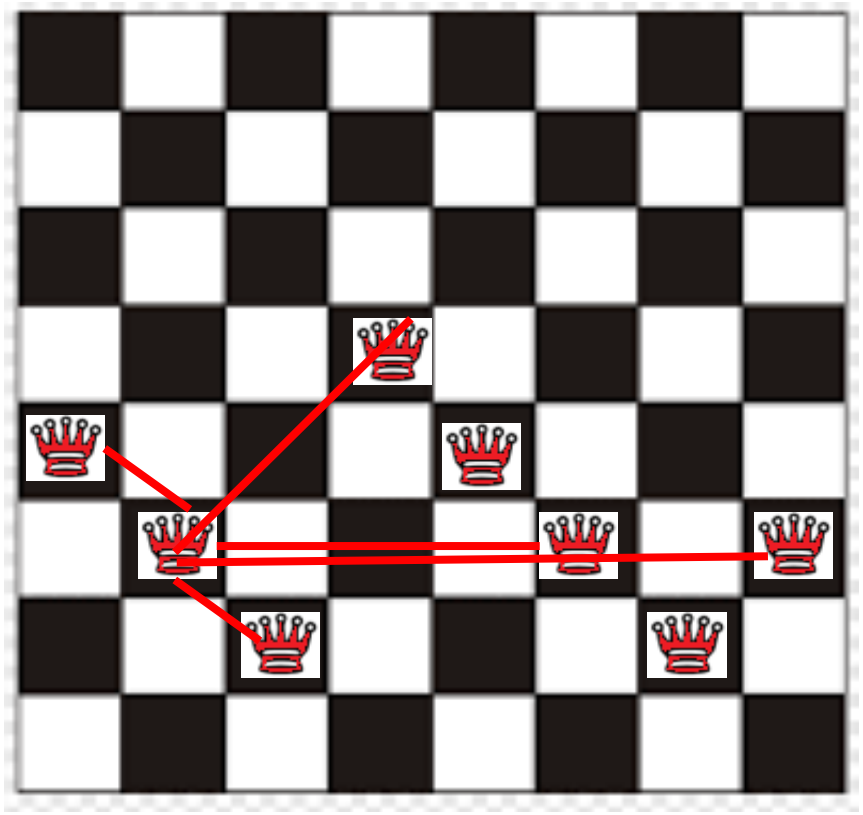
Neighbor: highest-valued successor

- Successor function returns all possible states generated by moving a single queen to another square in the same column.
- Each state has 8*7 = 56 successors
- Choose randomly among the set of best successors if there is more than one.

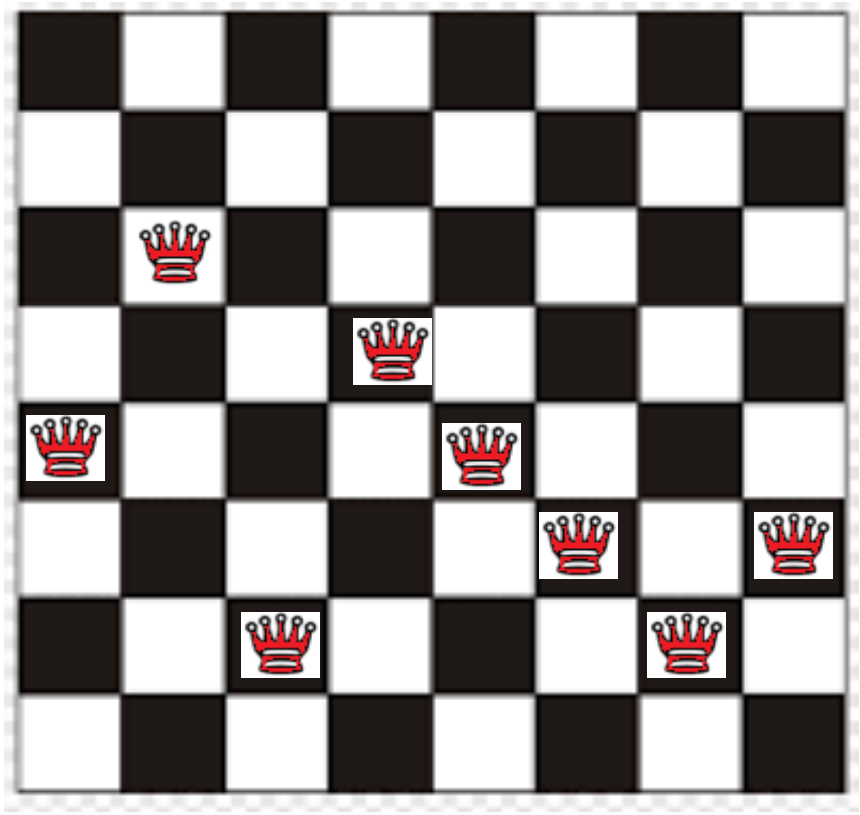
Neighbor: random successor

- Successor function returns a random state generated by moving a random single queen to another square in the same column.

Neighbor: Random Successor



Column: Q2
Row: 3➔6



Neighbor: Highest-valued Successor

Each number indicates h if we move a queen in its corresponding column

State value dari setiap successor belum diberi tanda negative (cost)

Summary: State, Neighbor

Keep single current state
State: "complete" configurations (one queen per column)
Solution: final state

Find the best state (global optimum)
Action: move to neighbors

Next:

- Hill climbing Search
- Simulated annealing
- Genetic algorithm

Masayu Leylia Khodra

KK IF – Teknik Informatika – STEI ITB

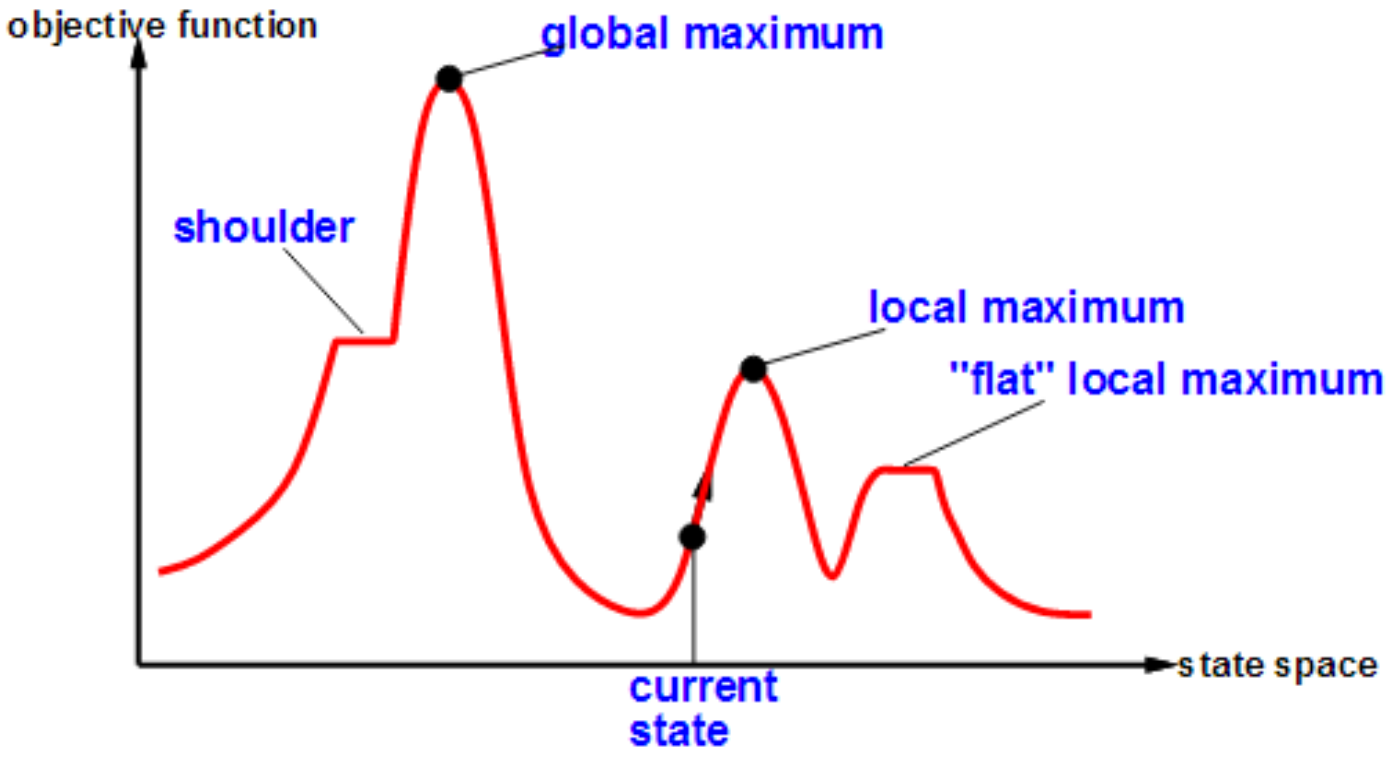
Modul 3: Beyond Classical Search

Hill-climbing Search

Inteligensi Buatan
(*Artificial Intelligence*)

Hill-climbing Search

"Like climbing Everest in thick fog with amnesia"



Starting from a randomly generated initial state

Loop that continually moves in the direction of increasing value (objective) or decreasing value (cost)

Terminates when it reaches a "peak" where no neighbor has a higher value

Hill-climbing Search: Steepest Ascent

(Russel & Norvig, 2010)

function HILL-CLIMBING(*problem*) **returns** a state that is a local maximum

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

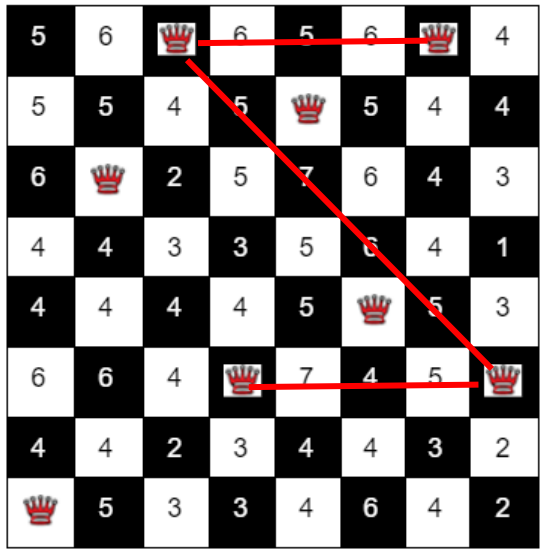
Starting from a randomly generated initial state

Loop that continually moves in the direction of increasing value (objective) or decreasing value (cost)

Terminates when it reaches a **"peak"** including **"flat"** where no neighbor has a higher value

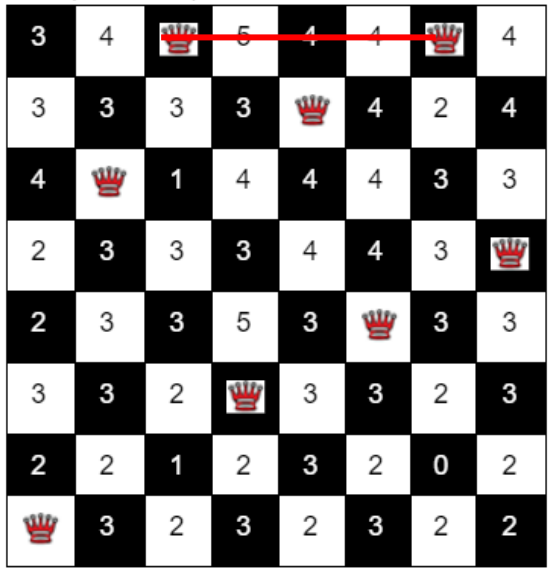
Hill-climbing: Illustration

h=-3



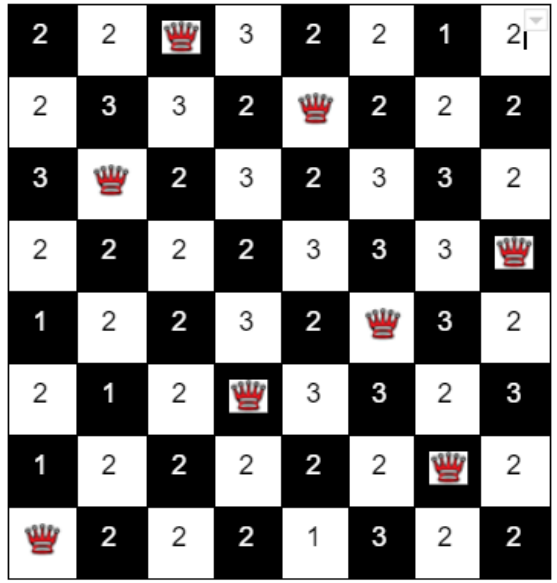
Q8: 3→5

h=-1 (Q8: 3 →5)



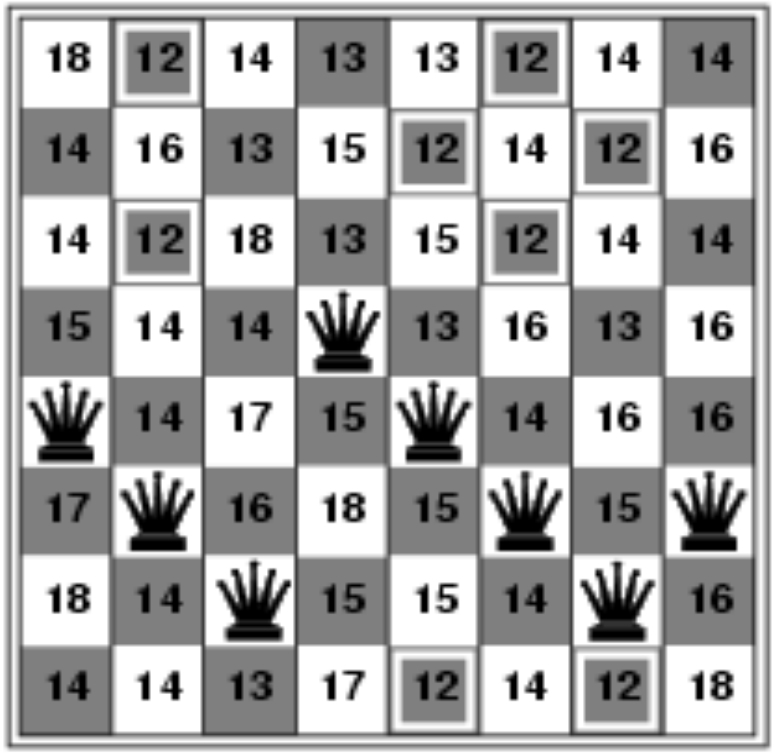
Q7: 8→2

h=0 (Q7: 8 →2)



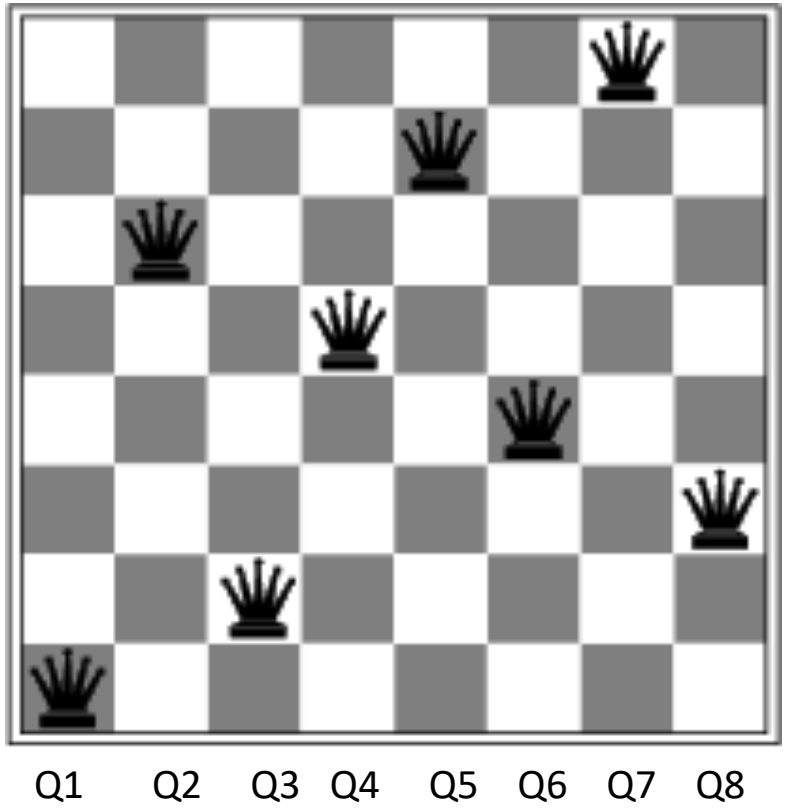
stop in global optimum (solution)

Hill-climbing: Stuck In Local Optimum



6 step

Step 1: h=-17 → h=-12
Step 2: h=-12 → h=-7
Step 3: h=-7 → h=-4
Step 4: h=-4 → h=-3
Step 5: h=-3 → h=-1
Step 6: h=-1 stop



Hill-climbing for 8-Queen Problem

function HILL-CLIMBING(*problem*) **returns** a state that is a local maximum

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

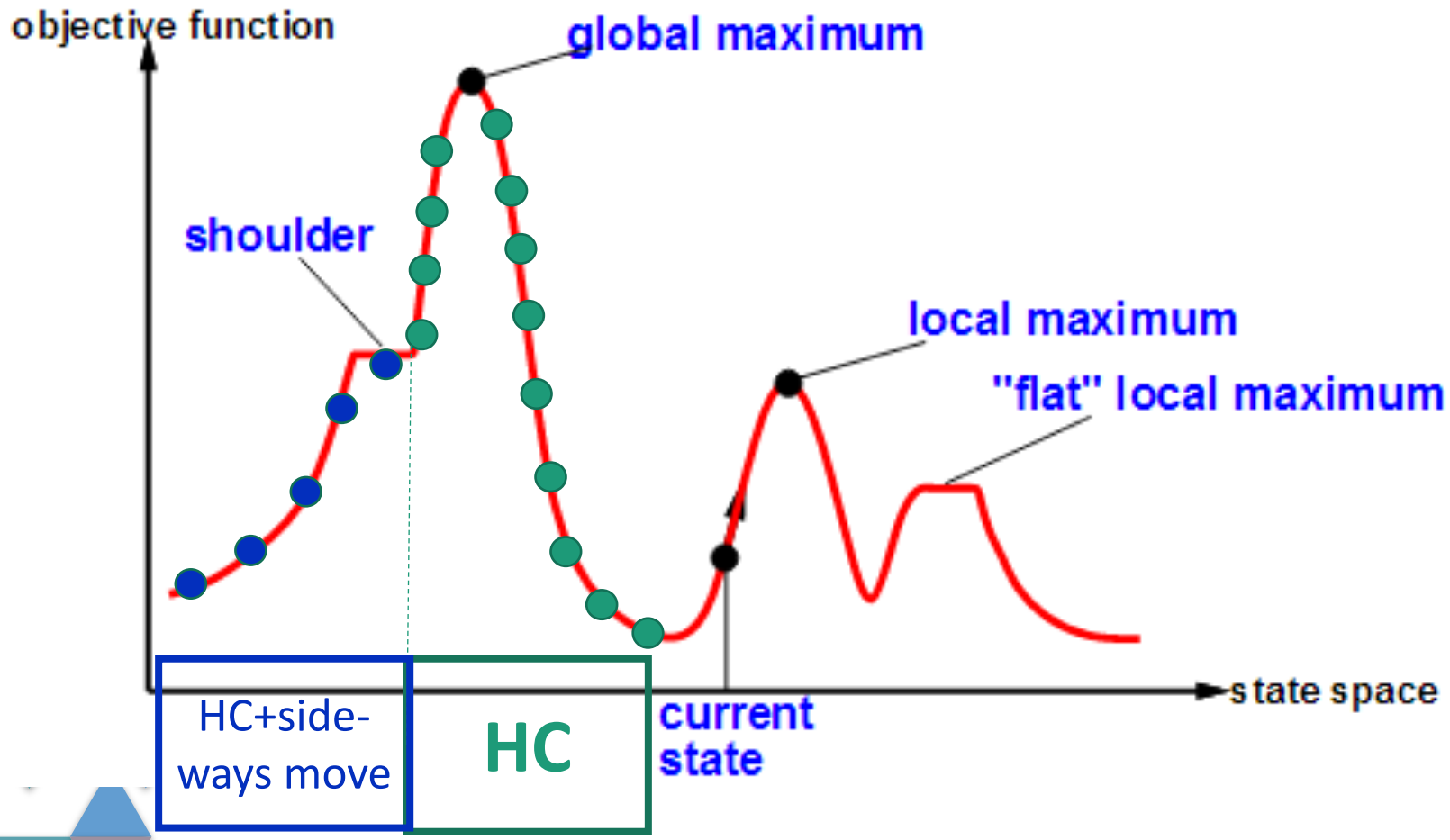
State space:
$8^8 \approx 16.8$ million states

Average case: works quickly
when **success**: avg 4 steps
when **stuck**: avg 3 steps

Best case:
initial state = goal state
Prob: $92 / 8^8 = 0.00054\%$

Get **stuck** 86%,
solving only 14% of
problem instances

Hill-climbing Search: Final State



Success: global maximum

Stuck: 1) local maximum, 2) flat local maximum, 3) shoulder

Shoulder: still possible to global max.
Variant HC: + **sideways move** with limit on number of consecutive ways

Variant 1: Hill-climbing with Sideways Move

boleh in tetangga nilai nya sama, jd gn langsung berhenti

function HILL-CLIMBING(*problem*) **returns** a state that is a local maximum

current ← MAKE-NODE(*problem*.INITIAL-STATE)
loop do
neighbor ← a highest-valued successor of *current*
if neighbor.VALUE < current.VALUE **then return** *current*.STATE
current ← *neighbor*

Terminates when it reaches a "peak" ~~including "flat"~~

Increase success for 8-queens problem.
Limit=100:
14% ➔ 94% success

Works **slower**:
when **success**: avg 4 ➔ 21 steps
when **stuck**: avg 3 ➔ 64 steps

Variant 2: Random Restart Hill-climbing

If at first you don't succeed, try, try again. It conducts **a series of hill climbing** searches (random initial states, until a goal is found)

Expected nb of restarts = `1/p` → p=0.14: 7 restart
Expected nb of steps = `s+f(1-p)/p` → p=0.14, s=4, f=3: 22 steps.

Very effective for n-queens problem: solving 3 million queens in under a minute (Luby et al., 1993)

hanya membangkitkan 1 successor secara random.

Variant 3: Stochastic Hill-climbing

cek nilai tetangga lebih bagus. kalau worse (≤), statenya tetap selama iterasinya belum sampai nmax.

function HILL-CLIMBING(*problem*) **returns** a state that is a local maximum

current ← MAKE-NODE(*problem*.INITIAL-STATE)
repeat nmax times
  neighbor ← a **random** successor of *current*
  if neighbor.VALUE **>** current.VALUE **then** *current* ← *neighbor*

Terminates when it reaches **nmax iteration**

Generating a successor randomly (not all successor)

Move to neighbor if it is better than current state.

Works **slower** (more steps)

Summary: Hill-climbing

continually moves in the direction of increasing value (objective) or decreasing value (cost)

Depending on initial state, can get stuck in local maxima.

Variant: steepest ascent HC, HC with sideways move, stochastic HC, random restart HC

Next:

- Simulated annealing

THANK YOU

Modul 3: Beyond Classical Search

Masayu Leylia Khodra

KK IF – Teknik Informatika – STEI ITB

Simulated Annealing

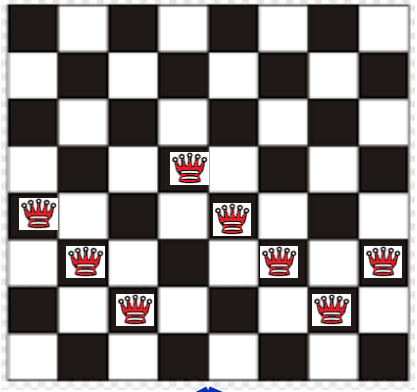
Inteligensi Buatan
(*Artificial Intelligence*)

SA: Combining Completeness and Efficiency

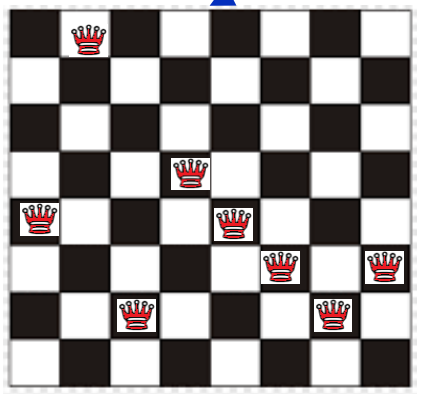
- Simulated annealing combines hill climbing (efficient) with a random walk (complete)
- Idea: escape local maxima by allowing some "bad" moves but gradually decrease their frequency
- Simulated annealing is a version of stochastic hill climbing where some downhill moves are allowed.

Neighbor: One Random Successor of Current

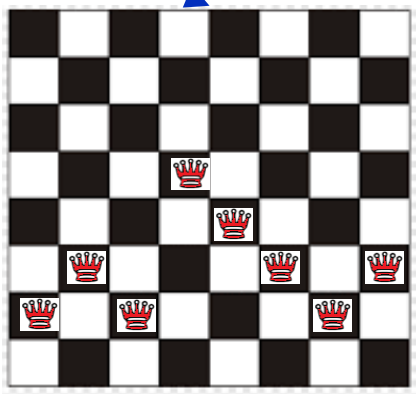
Current State: h=-17



56 possible random successor



Better neighbor: h=-12



Worse neighbor: h=-18

- Stochastic hill-climbing:
  - Only move to better neighbor, and skip worse neighbor.
- Simulated annealing:
  - move to better neighbor, and allow move to worse neighbor that has probability

$$e^{\Delta E/T}$$

nilai / utilitas state tetangga terpilih − nilai / utilitas state sekarang.

temperatur

hanya digunakan ketika state tetangga yg dibangkitkan lebih buruk.

T ↑, nilai e jd makin besar · karena ΔE negatif → prob. ↑

Simulated Annealing

ΔE selalu negatif karena dia lebih buruk.

https://id.m.wikipedia.org/wiki/Berkas:Annealing_a_silver_strip.JPG

Annealing: heat (metal or glass) and allow it to cool slowly, in order to remove internal stresses and toughen it.

- T is the "temperature" of annealing that gradually decreases.
- ΔE=neighbor.value-current.value
- If better neighbor (ΔE>0): move to neighbor (stochastic HC)

kalo lebih > 0 pasti pindah

- If worse neighbor (ΔE<0): probability move $e^{\Delta E/T}$
  - When T is high (e.g. ΔE=-5, T=100, prob=0.95), there is a lot of random motion ➔ random walk
  - When T approaches 0 (e.g. ΔE=-5, T=1, prob=0.007), randomness is decreased ➔ stochastic hill climbing.

Simulated Annealing (Russel & Norvig, 2010)

function SIMULATED-ANNEALING(*problem*, *schedule*) **returns** a solution state
  inputs: *problem*, a problem
    schedule, a mapping from time to "temperature"

  current ← MAKE-NODE(*problem*.INITIAL-STATE)
  for $t = 1$ **to** ∞ **do**
    $T \leftarrow schedule(t)$
    if $T = 0$ **then return** *current*
    next ← a randomly selected successor of *current*
    $\Delta E \leftarrow next.\text{VALUE} - current.\text{VALUE}$
    if $\Delta E > 0$ **then** *current* ← *next*
    else *current* ← *next* only with probability $e^{\Delta E/T}$

T: temperature as a function of time t

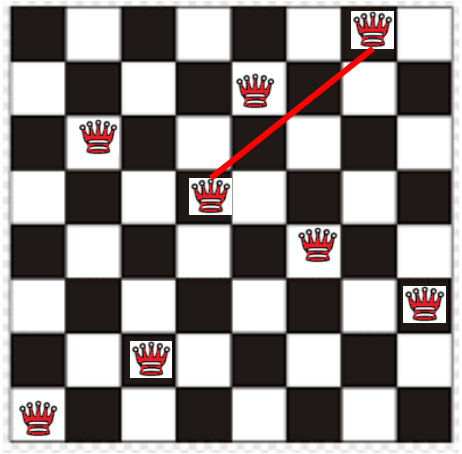
Terminates T=0

Move to better neighbor (stochastic HC)

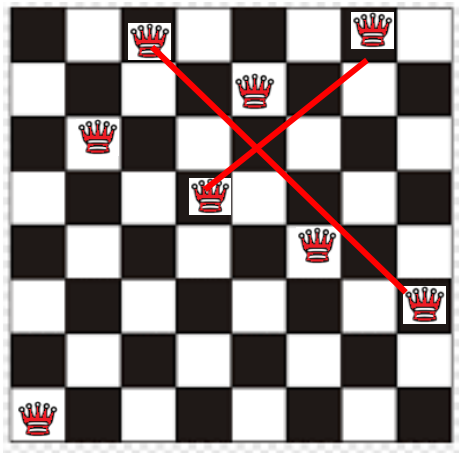
allowing some "bad" moves, depends on probability

Move to Worse State

Current State: h=-1



Column: C3
Row: 2➔8



$current \leftarrow next$ only with probability $e^{\Delta E/T}$

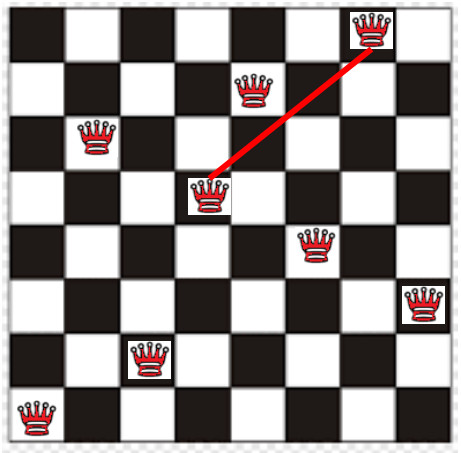
Compare to static value:
move probability > 0.5

Compare to random value:
move probability > random(0,1)

Next: h=-2, T=10
$\Delta E<0$: prob=$e^{(-2-(-1))/10}=e^{-0.1}$ =0.9

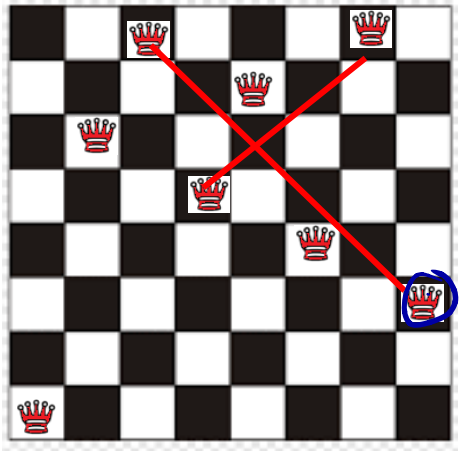
Simulated Annealing: Illustration

Current State: h=-1



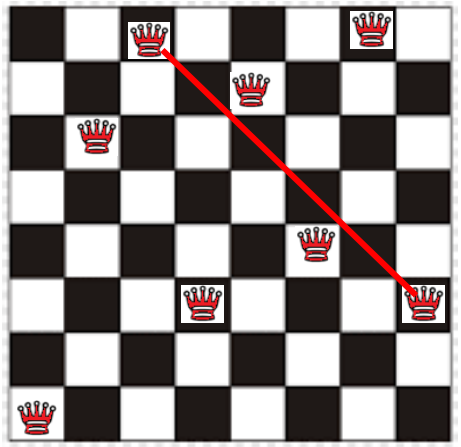
Column: C3
Row: 2➔8

Next: h=-2, T=10
Prob=$e^{(-2-(-1))/10}=e^{-0.1}$ =0.9
Current ← next



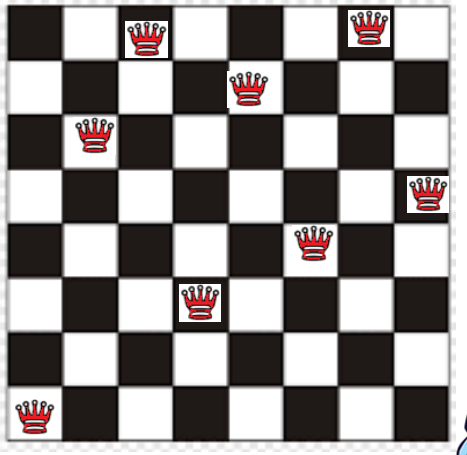
Column: C4
Row: 5➔3

Next: h=-1
Current ← next

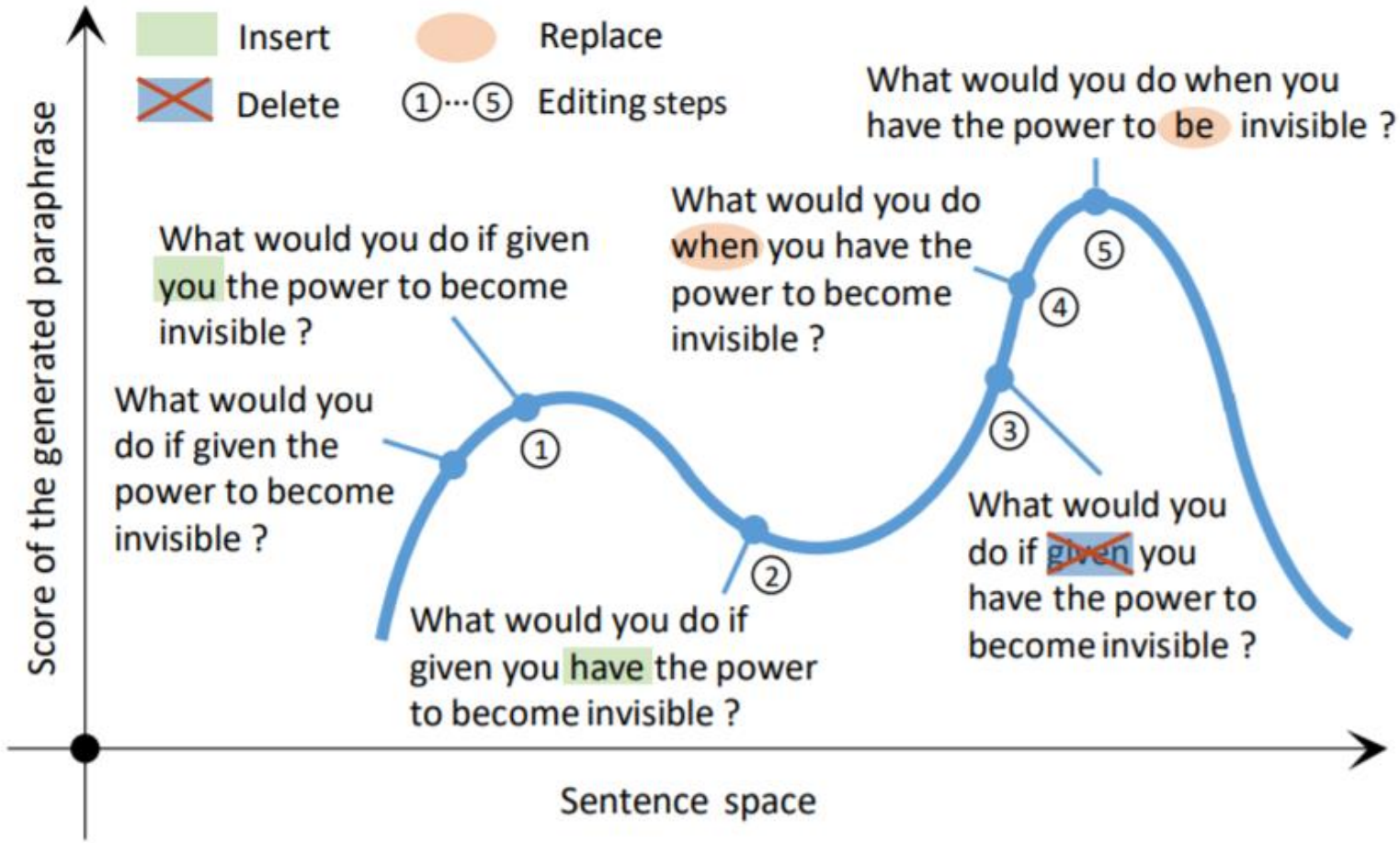


Column: C8
Row: 3➔5

Next: h=0
Current ← next



Application: Simulated Annealing for Paraphrase



Liu, X., Mou, L., Meng, F., Zhou, H., Zhou, J., & Song, S. (2020). Unsupervised paraphrasing by simulated annealing. Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 302–312 July 5 - 10, 2020. c 2020 Association for Computational Linguistics. *arXiv preprint arXiv:1909.03588*.

Properties of simulated annealing search

One can prove: If *T* decreases slowly enough, then simulated annealing search will find a global optimum with probability approaching 1

Widely used in VLSI layout, airline scheduling, etc

Summary: Simulated Annealing

Simulated annealing is a version of stochastic hill climbing where some downhill moves are allowed.

If *T* decreases slowly enough, then simulated annealing search will find a global optimum with probability approaching 1

Next:

- Genetic Algorithm

THANK YOU


INSTITUT TEKNOLOGI BANDUNG
1920

Modul 3: Beyond Classical Search

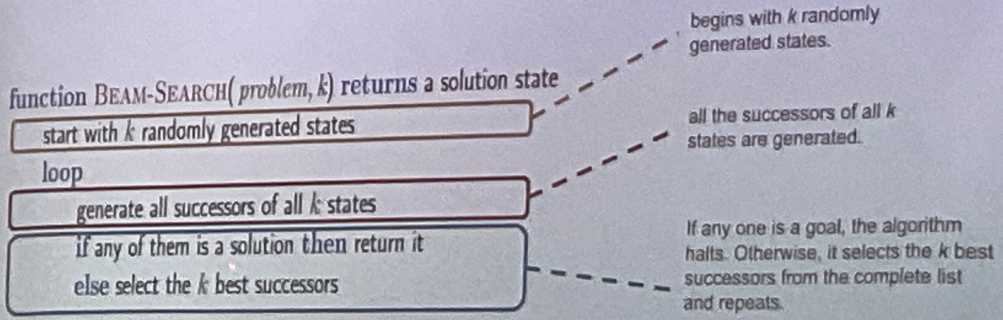
Genetic Algorithm

KK IF – Teknik Informatika – STEI ITB

Inteligensi Buatan
(*Artificial Intelligence*)

Genetic Algorithm

Local Beam Search → masih ada kemungkinan stuck di local optima



In a local beam search, useful information is passed among the parallel search k threads

Genetic Algorithm: Illustration (Russel & Norvig, 2010)

State / Individual

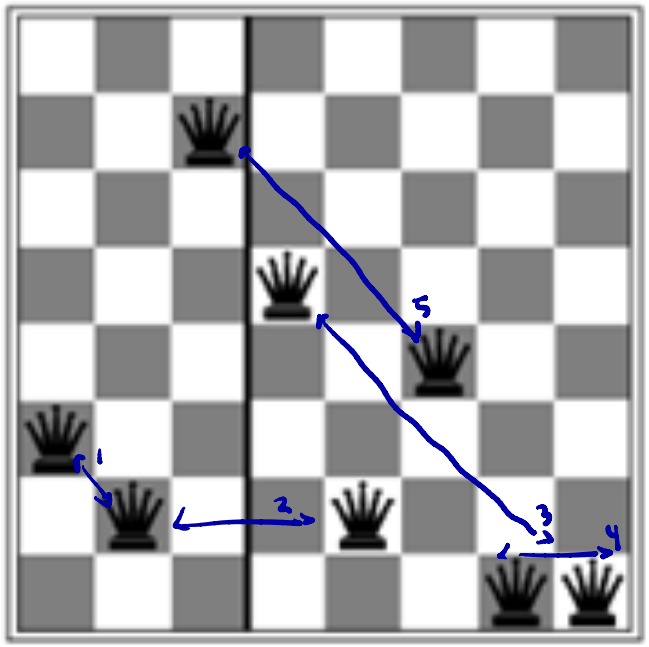
A state is represented as a string over a finite alphabet (often a string of 0s and 1s)

One character ~ one variable

State Value: Fitness Function

- Evaluation function (fitness function).
- Higher values for better states.
- Fitness function for n-queen: number of non-attacking pairs of queens
  - min = 0
  - max = $(8 \times 7)/2 = 28$ (global maximum)

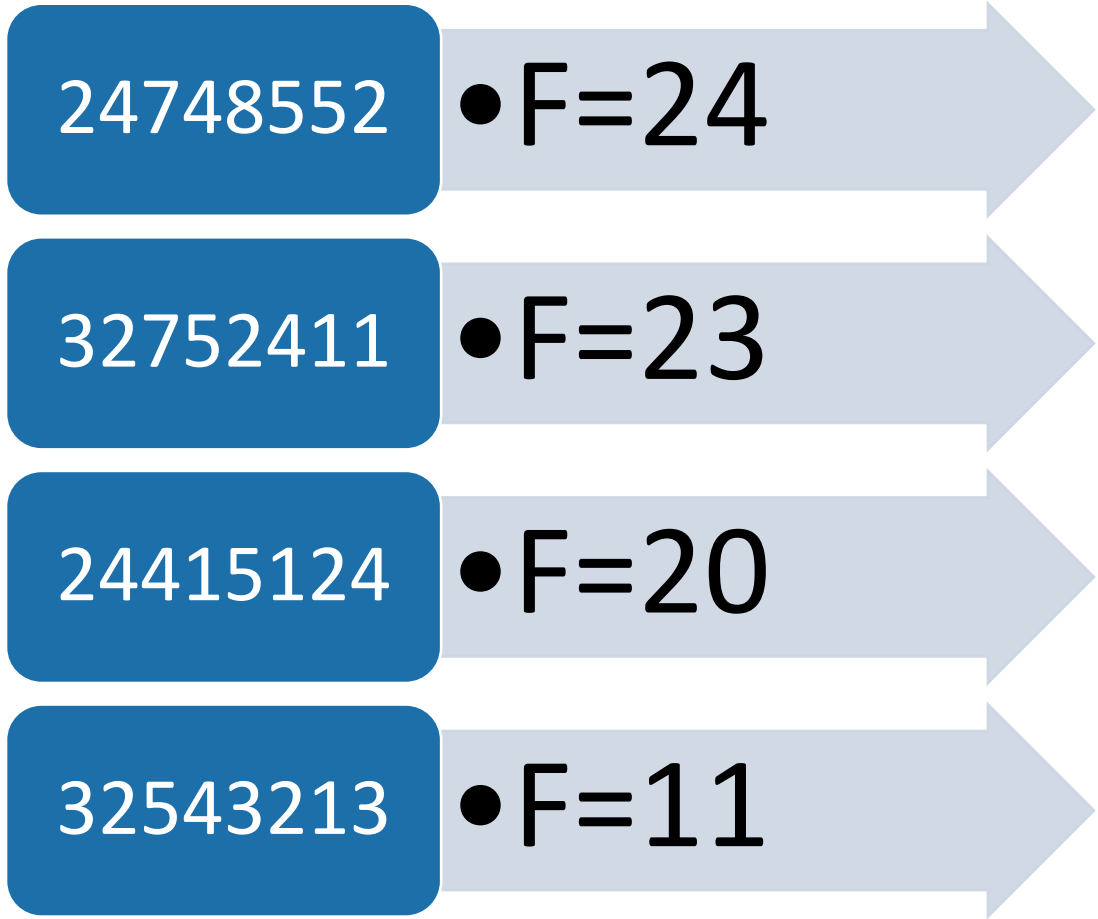
32752411



F(32752411)=28-5=23

Initial Population & Successor Function

- GA starts with *k* randomly generated states.
- A successor state is generated by combining two parent states
- Produce the next generation of states by selection, crossover, and mutation

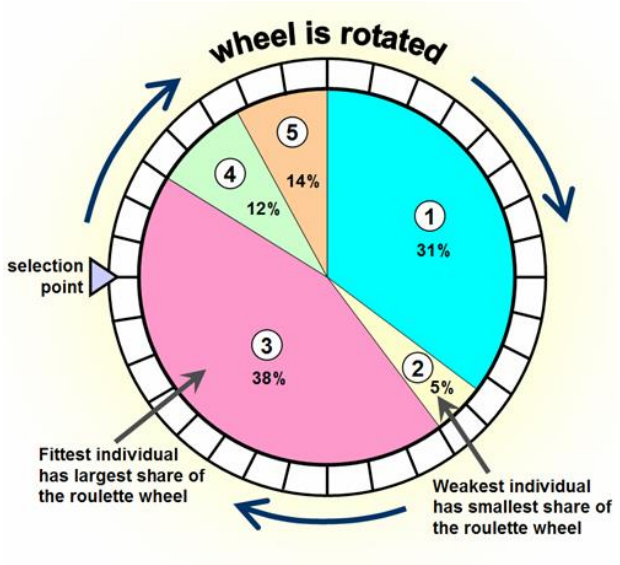


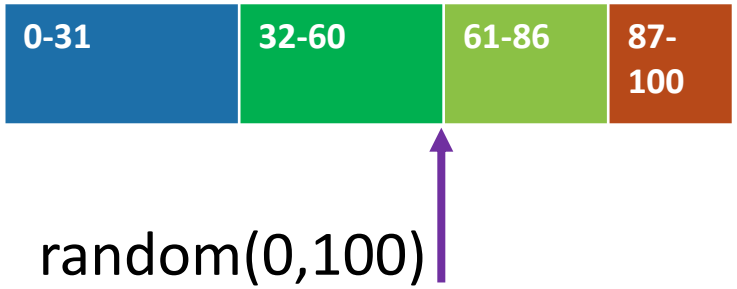
pendekatan — threshold
roulette wheel.

Random Selection of Parent States

Probability of random selection

24748552	• 24/(24+23+20+11) • = 31%
32752411	• 23/(24+23+20+11) • =29%
24415124	• 20/(24+23+20+11) • =26%
32543213	• 11/(24+23+20+11) • =14%





32752411

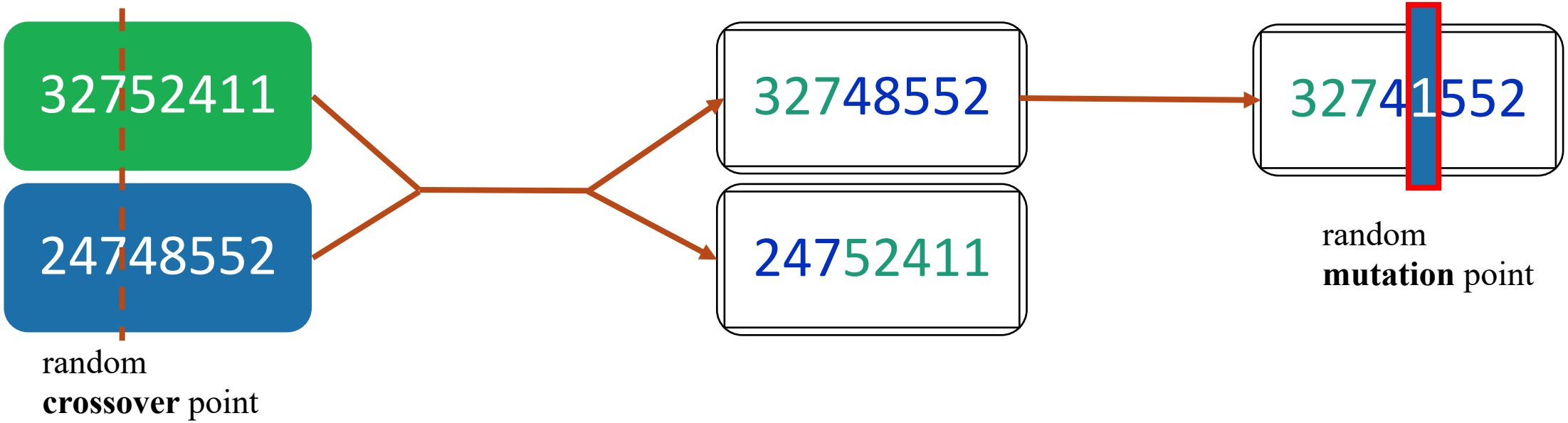
24748552

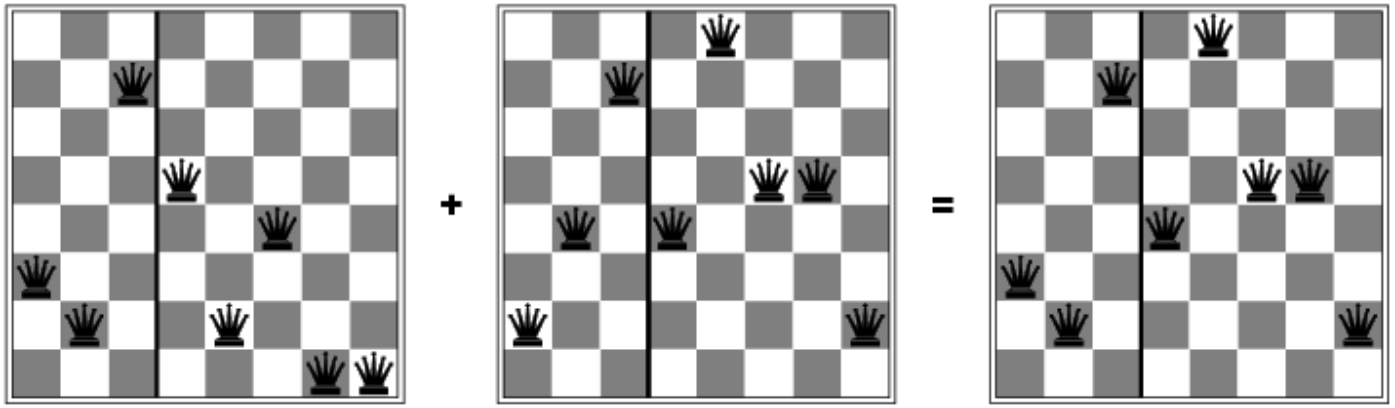
32752411

24415124

EDUNEX ITB

Cross Over / Reproduce & Mutation





In more popular version, each mating of two parents produces only one offspring, not two (Russel & Norvig, 2010)

Genetic Algorithm: Illustration (Russel & Norvig, 2010)

Genetic Algorithm

function GENETIC-ALGORITHM(*population*, FITNESS-FN) **returns** an individual
  inputs: *population*, a set of individuals
    FITNESS-FN, a function that measures the fitness of an individual

  repeat
    new_population ← empty set
    for *i* = 1 **to** SIZE(*population*) **do**
      x ← RANDOM-SELECTION(*population*, FITNESS-FN)
      y ← RANDOM-SELECTION(*population*, FITNESS-FN)
      child ← REPRODUCE(*x*, *y*)
      if (small random probability) **then** *child* ← MUTATE(*child*)
      add *child* to *new_population*
    population ← *new_population*
  until some individual is fit enough, or enough time has elapsed
  return the best individual in *population*, according to FITNESS-FN

Summary

k randomly generated states (population: k individual)

Fitness function (state value)

successor function: combining two parent states (selection, cross-over, mutation)

Terima Kasih


INSTITUT TEKNOLOGI BANDUNG
1920

1. Teknik local search yang mengenumerasi semua state tetangga (successor) dan current state pindah ke state tetangga terbaik jika nilai evaluasi state tetangga terbaik lebih baik dari nilai evaluasi current state adalah:
a. steepest-ascend hill-climbing, b. hill-climbing with sideways move; c. stochastic hill-climbing

2. Simulated annealing (SA) merupakan kombinasi dari hill-climbing dan random walk. Manakah karakteristik dari SA ?
a. Jika penurunan temperatur T cukup lambat, SA menjamin akan menemukan optimum global dengan peluang mendekati 1.
b. SA memiliki peluang yang membesar secara eksponensial untuk memilih state tetangga yang lebih buruk nilai evaluasinya.
c. SA membangkitkan secara random satu state tetangga, bukan semua state tetangga untuk setiap iterasinya.

3. Misalkan current state memiliki nilai evaluasi h, dan objektifnya mencari nilai h maksimum. Manakah perpindahan current state ke tetangga yang benar.
a. Pada steepest-ascend hill climbing, current state akan selalu pindah ke tetangga terbaik dari semua tetangga yang ada, dengan nilai evaluasi h_next jika h_next >h. Jika h_next <=h, masih ada peluang untuk berpindah state.
b. Pada simulated annealing, current state akan selalu pindah ke tetangga yang diambil secara acak dengan nilai evaluasi h_next jika h_next>h. Jika h_next <=h, masih ada peluang untuk berpindah state.
c. Pada algoritma genetika, current state akan selalu pindah ke tetangga dengan nilai evaluasi h_next yang dihasilkan dari proses seleksi, cross-over, dan mutasi tanpa mempedulikan apakah h_next >h.

Pada simulated annealing, diberikan sejumlah kondisi nilai temperatur (T), current.value, dan neighbor.value.

Diketahui batas move probability adalah 0.6. Pada ketiga kasus berikut, tentukanlah peluang neighbor state akan

dipilih menggantikan current state, serta current state diset dengan apa. (Nilai 10)

a. T = 10, current.value = 5, neighbor.value = 3 → pindah ke neighbor → tetangga is worse, $e^{\frac{\Delta E}{T}} = e^{\frac{3-5}{10}} = e^{\frac{-2}{10}} = 0.8$ an.

b. T = 3, current.value = 3, neighbor.value = 5 → pindah ke neighbor → tetangga is better. Iqsa pindah.

c. T = 0, current.value = 5, neighbor.value = 3 → ga ngapa² in

soalnya T nya = 0., iqsa return parent.

- Pada algoritma genetika dengan populasi 4 individu (A,B,C,D), misalkan nilai fitness setiap individu secara berturut-turut adalah 10,8,7,5. Jika menggunakan roulette wheel dengan urutan individu tetap, lakukanlah proses seleksi pembentukan populasi untuk cross-over dengan urutan bilangan acak 0.55; 0.25; 0.85; 0.35

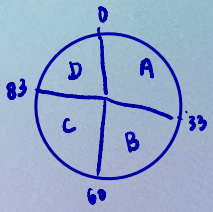
cari probability.
proporsi fitness function

A : $\frac{10}{30} = 0.33$

B : $\frac{8}{30} : 0,27$

C : $\frac{7}{30} : 0,23$

D : $\frac{5}{30} : 0,17$.



A	B	C	D
0-33	34-60	61-83	84-100

0,55 → B
0.25 → A
0,85 → D
0,35 → B


INSTITUT TEKNOLOGI BANDUNG
1920

Modul 4: Adversarial Search

Adversarial Search

Masayu Leylia Khodra
(masayu@informatika.org)

KK IF – Teknik Informatika – STEI ITB

Inteligensi Buatan
(*Artificial Intelligence*)

Adversarial Search

Non-Adversarial Search

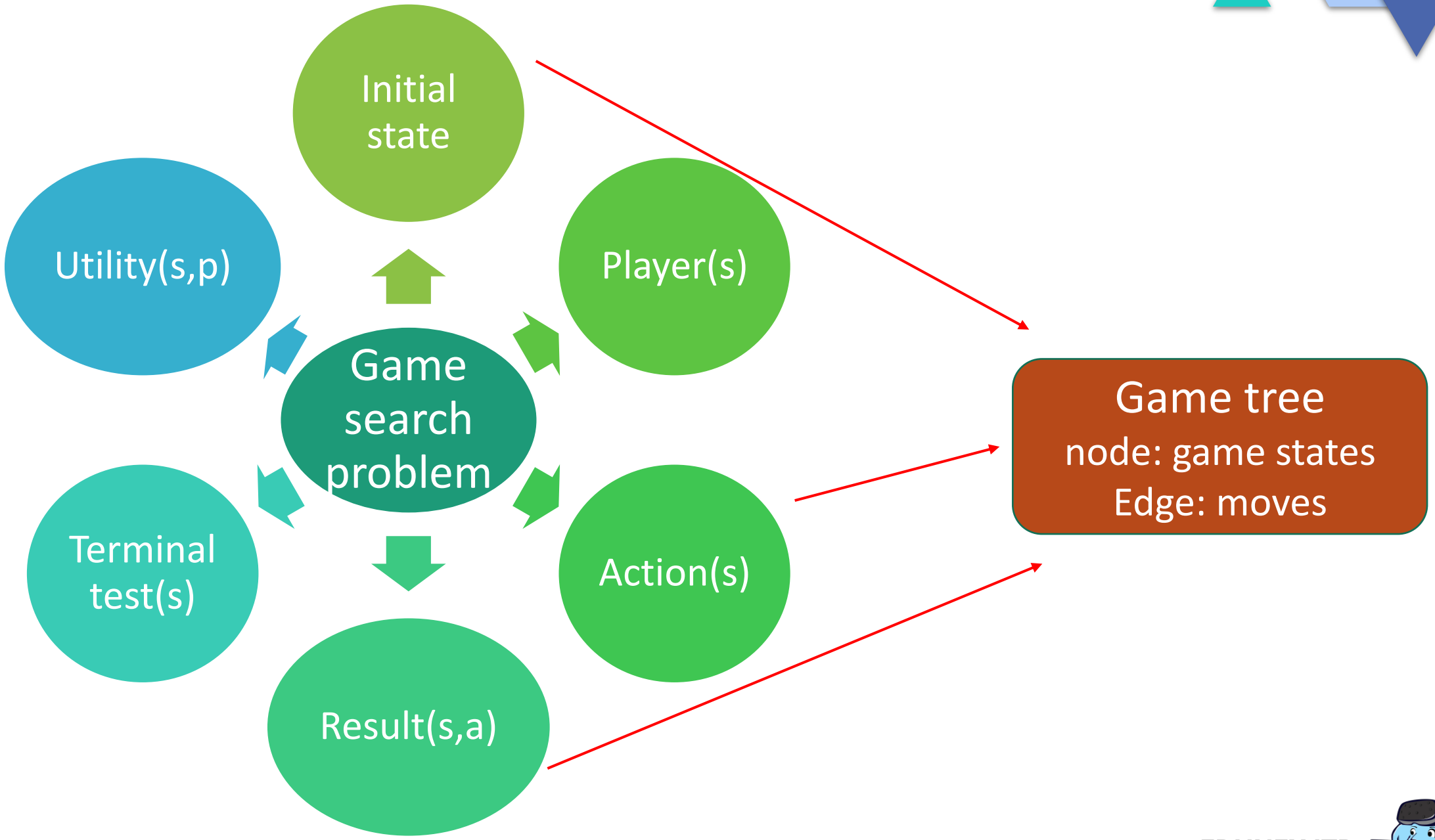
Approximate solution: strategy

Multiagent: competitive environment (games)

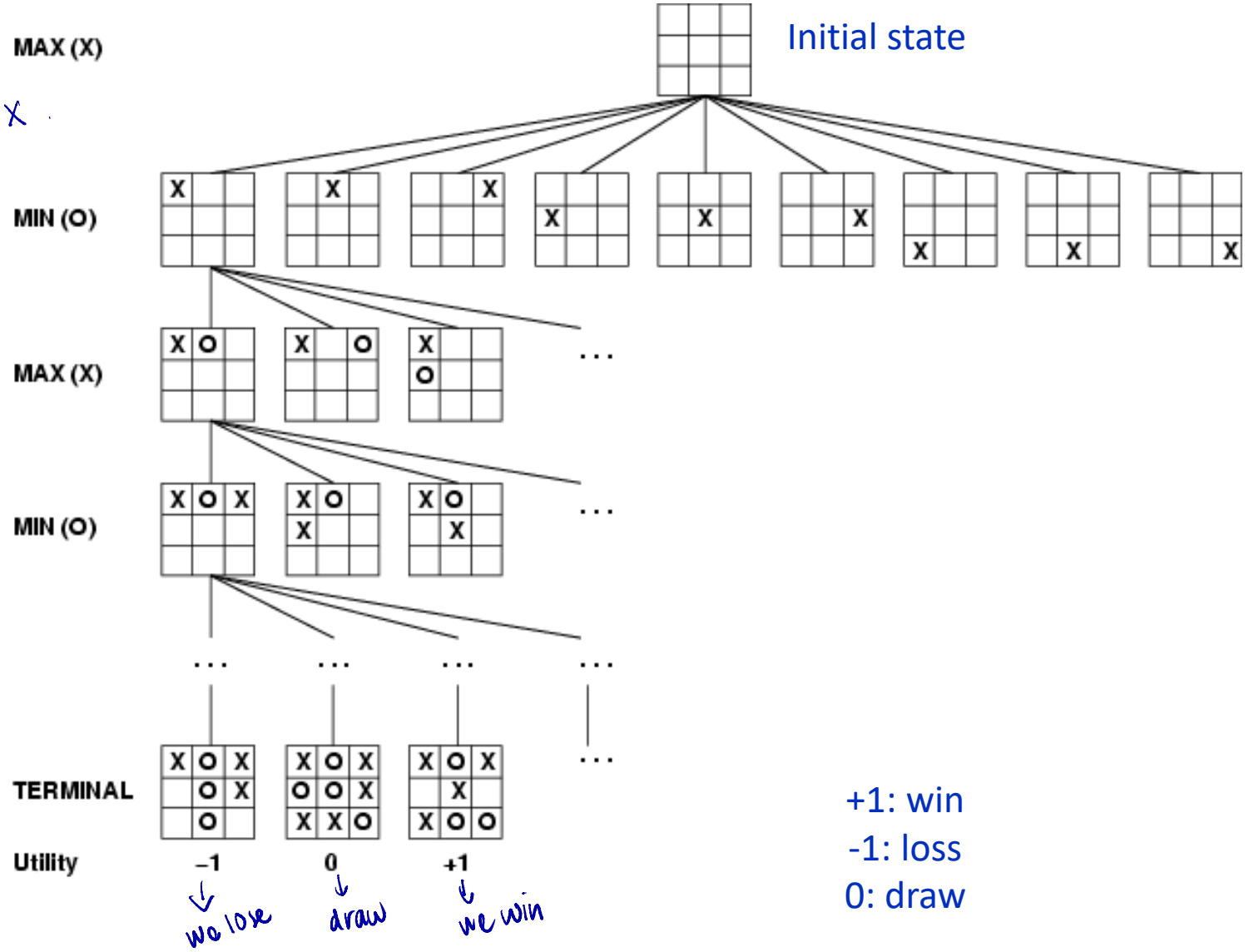
Optimize solution: path to goal or solution state

Deterministic and fully observable

Turn-taking


Initial state
Utility(s,p)
Player(s)
Game search problem
Terminal test(s)
Action(s)
Result(s,a)
Game tree
node: game states
Edge: moves

Game tree (2-players, tic-tac-toe)



node: game states
Edge: moves

+1: win
-1: loss
0: draw

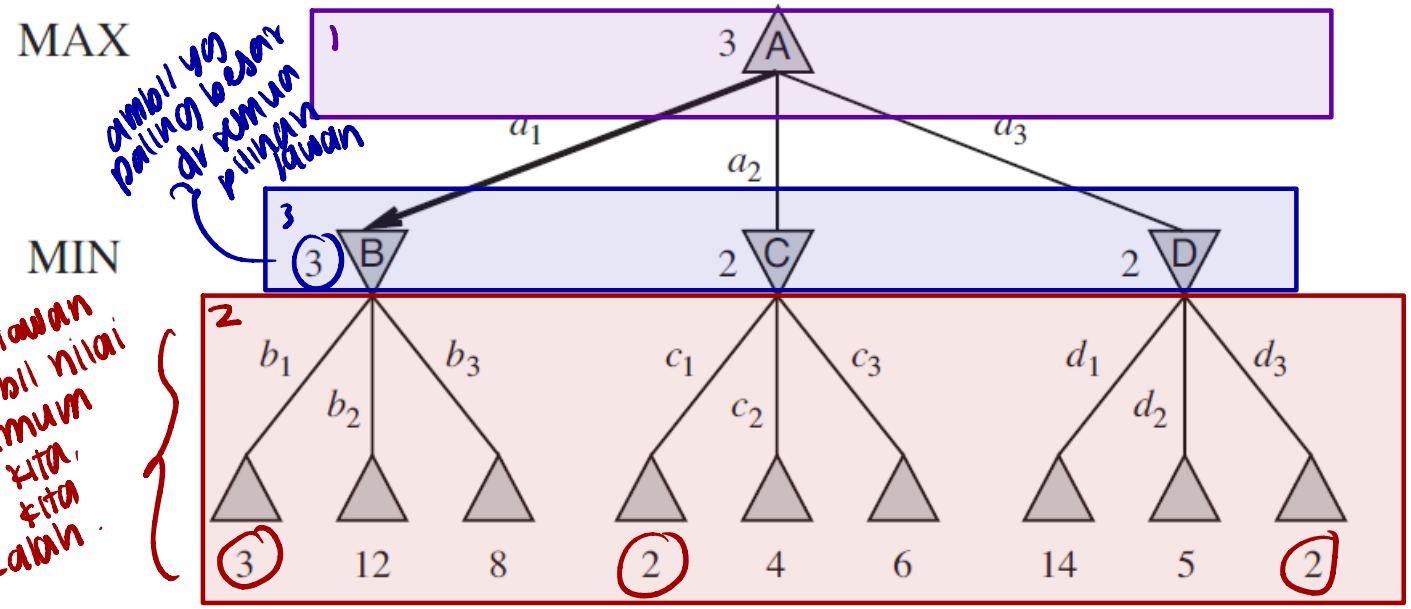
Optimal Decision with Minimax Algorithm

function MINIMAX-DECISION($state$) **returns** *an action*
  return $\arg\max_{a \in \text{ACTIONS}(s)}$ MIN-VALUE(RESULT($state, a$))

function MIN-VALUE($state$) **returns** *a utility value*
  if TERMINAL-TEST($state$) **then return** UTILITY($state$)
  $v \leftarrow \infty$
  for each a **in** ACTIONS($state$) **do**
    $v \leftarrow$ MIN(v, MAX-VALUE(RESULT(s, a)))
  return v

function MAX-VALUE($state$) **returns** *a utility value*
  if TERMINAL-TEST($state$) **then return** UTILITY($state$)
  $v \leftarrow -\infty$
  for each a **in** ACTIONS($state$) **do**
    $v \leftarrow$ MAX(v, MIN-VALUE(RESULT(s, a)))
  return v

Minimax: 2-ply Game Tree

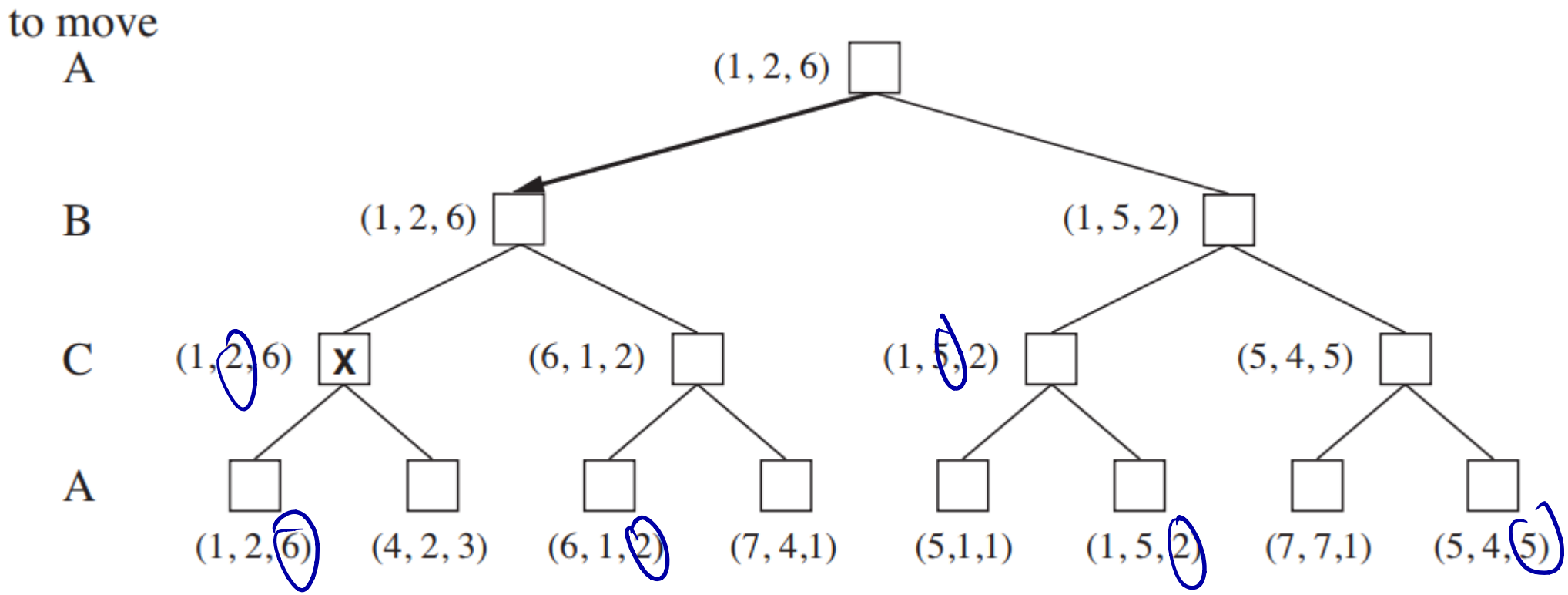


Minimax(B)=min(
Minimax(result(B,b1)),
Minimax(result(B,b2)),
Minimax(result(B,b3)))
=min(3,12,8)=3

Minimax(A)=max(Minimax(B),
Minimax(C),Minimax(D))
=max(3,2,2)=3

$$\text{MINIMAX}(s) = \begin{cases} \text{UTILITY}(s) & \text{if TERMINAL-TEST}(s) \\ \max_{a \in Actions(s)} \text{MINIMAX}(\text{RESULT}(s,a)) & \text{if PLAYER}(s) = \text{MAX} \\ \min_{a \in Actions(s)} \text{MINIMAX}(\text{RESULT}(s,a)) & \text{if PLAYER}(s) = \text{MIN} \end{cases}$$

Minimax 3-ply in Multiplayer Games



Minimax Properties

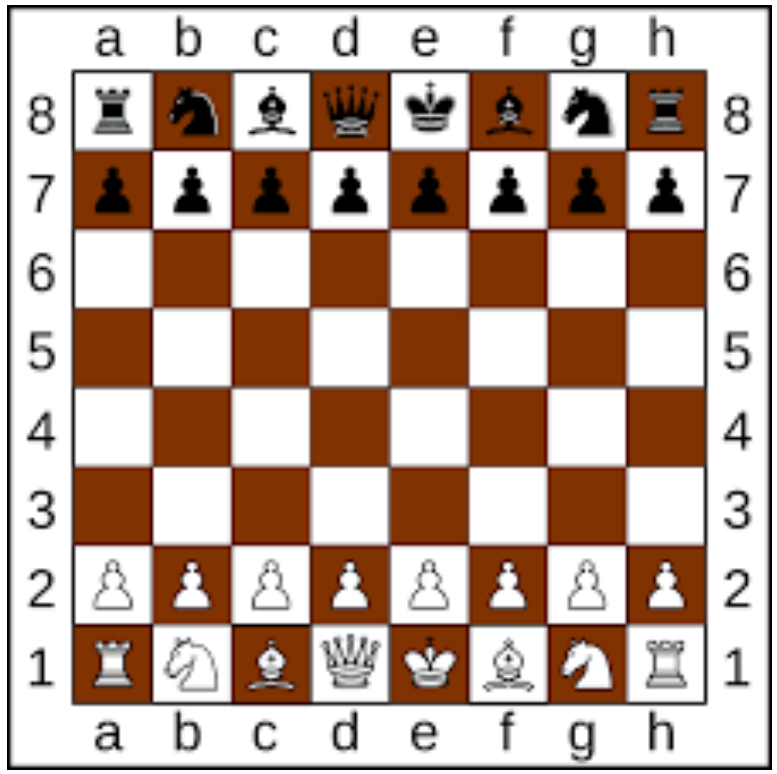
Complete?
Yes (if tree is finite)

Optimal?
Yes (against an optimal opponent)

Time complexity?
$O(b^m)$

Space complexity?
O(bm) (depth-first exploration)

Minimax for Chess

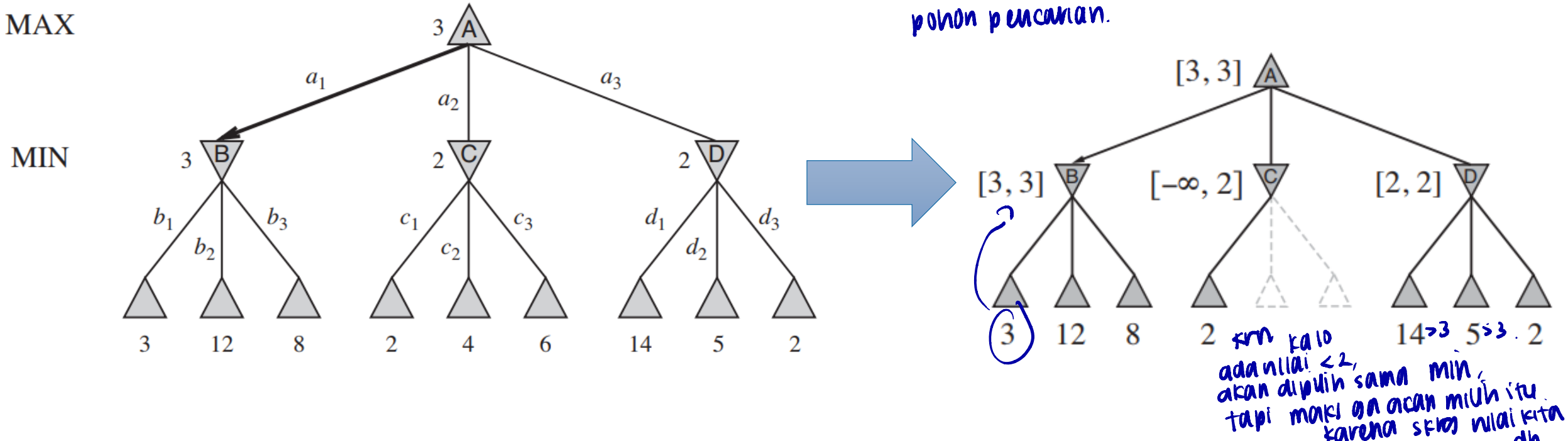


https://commons.wikimedia.org/wiki/File:AAA_SVG_Chessboard_and_chess_pieces_02.svg

- Branching factor: 35 (avg)
- Games often 50 moves for each player → m=100
- Game states is exponential in the depth of game tree.
- Exact solution is completely infeasible

αβSearch: Minimax with Pruning

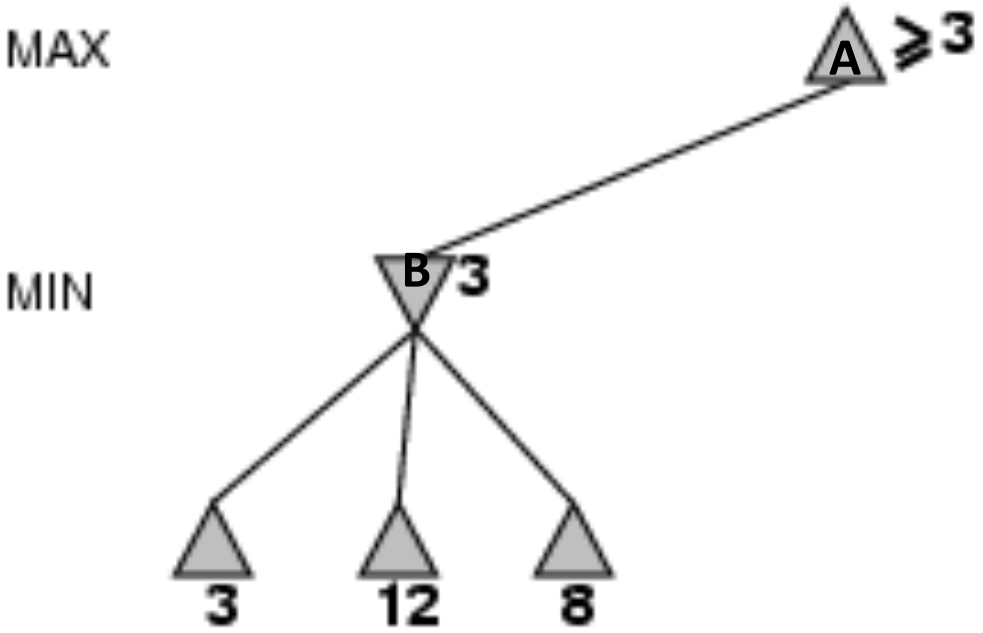
tidak mengubah hasil akhir, hanya memangkas pohon pencarian.



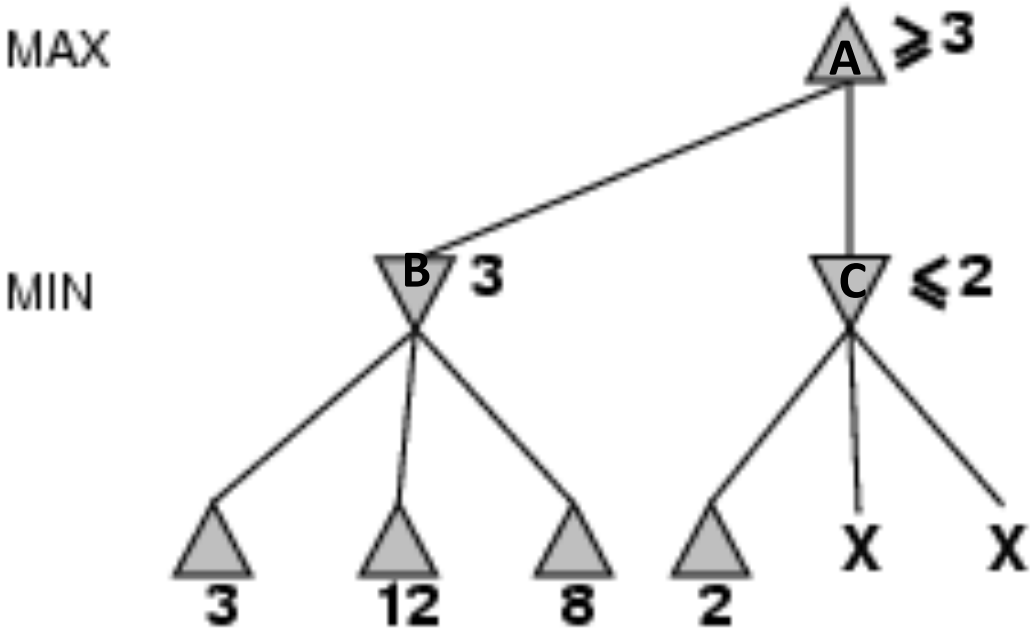
kalo ada nilai < 2, akan dipilih sama min, tapi maks gak akan milih itu karena sdh ada nilai kita 3.

Pruning does not affect final result. It returns the same move as minimax would, but prunes away branches that cannot possibly influence the final decision.

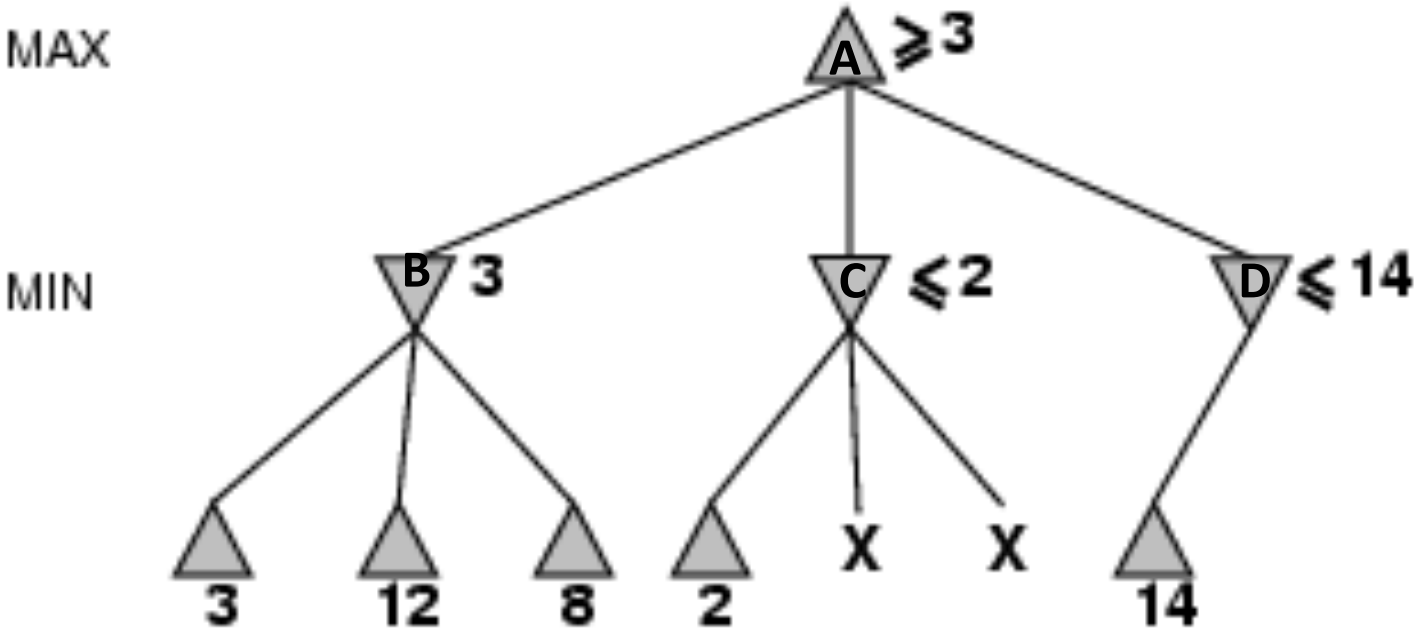
α-β pruning example



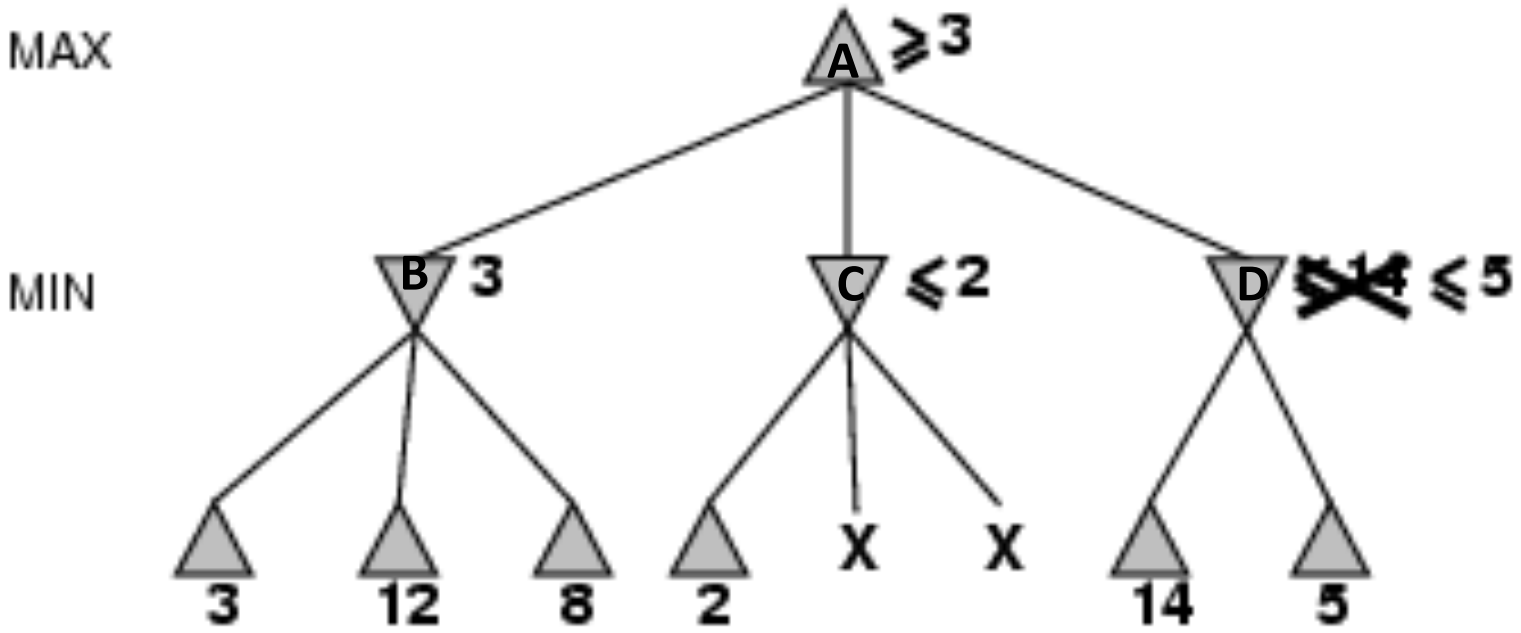
α-β pruning example



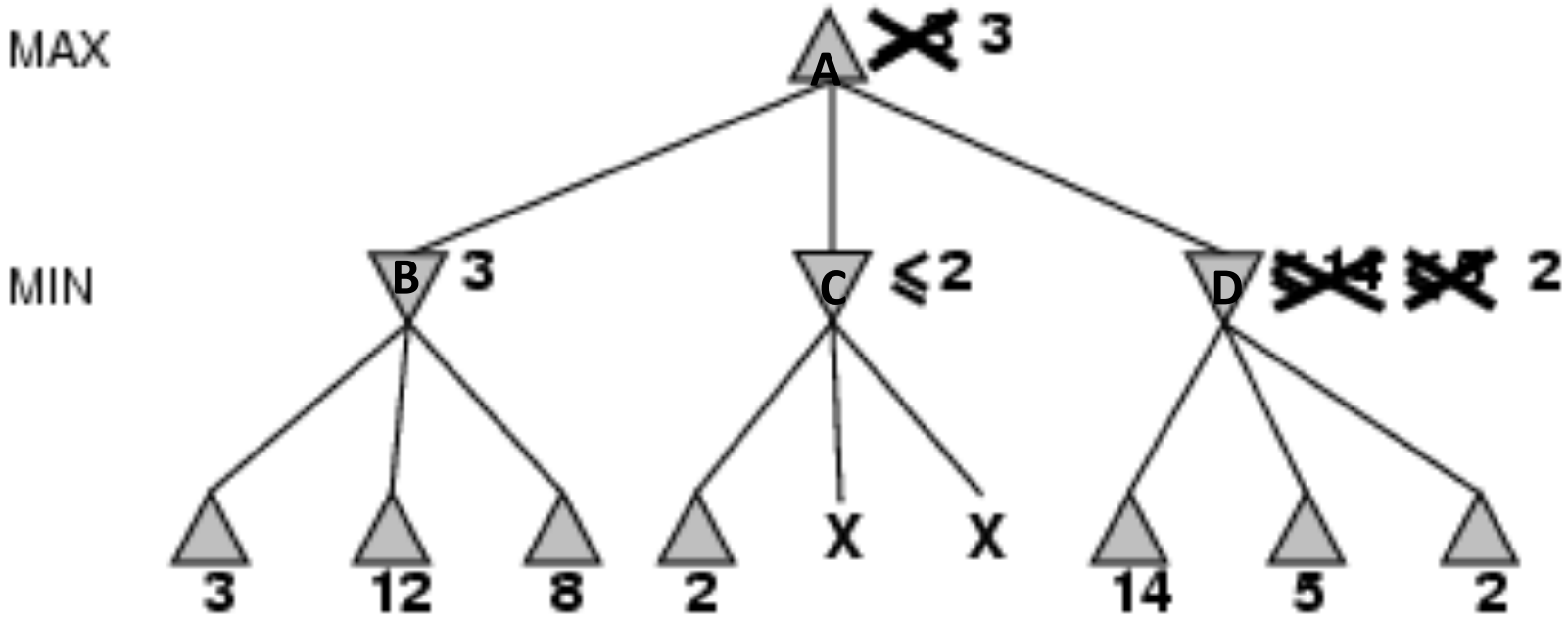
α-β pruning example



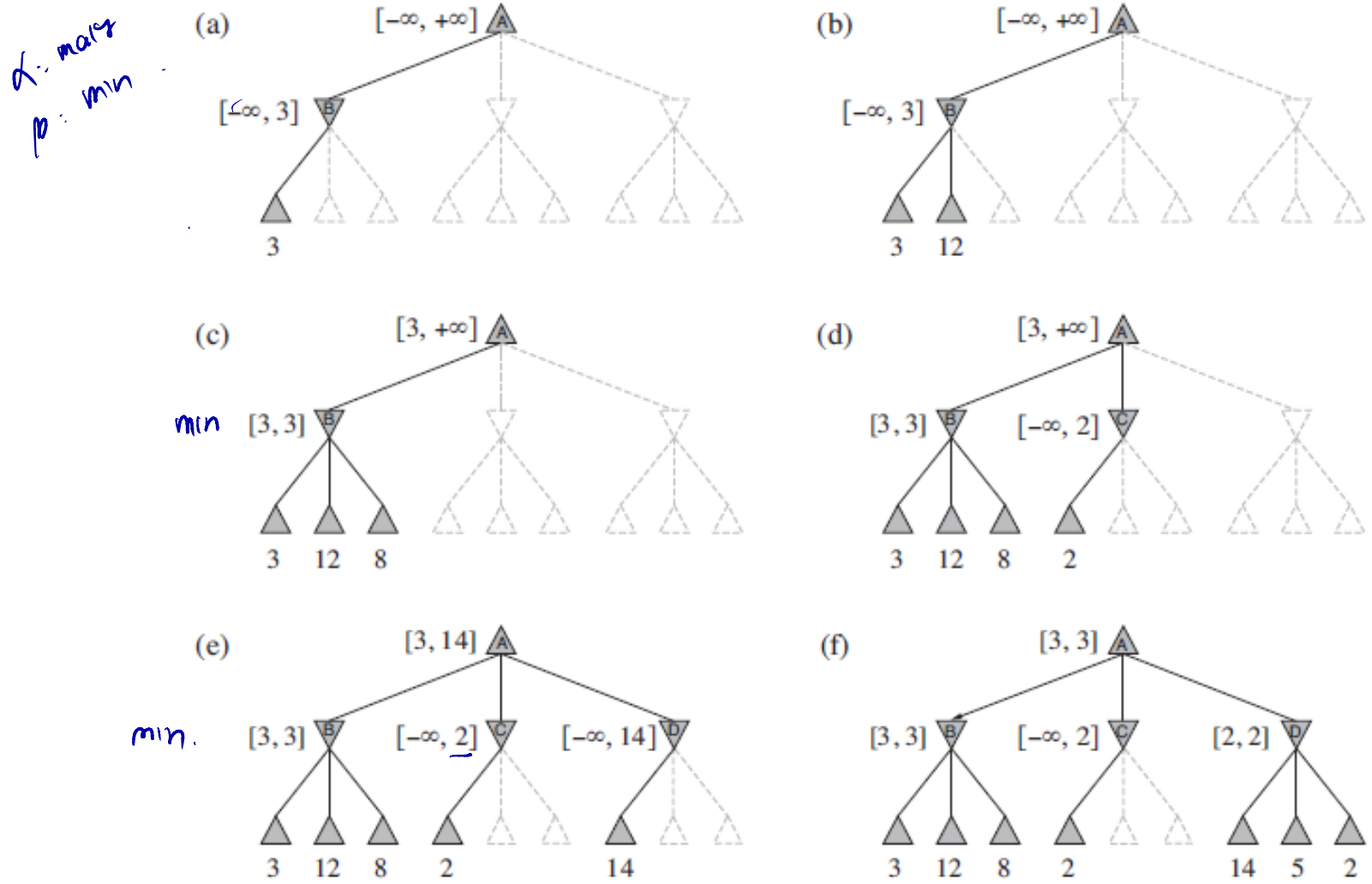
α-β pruning example



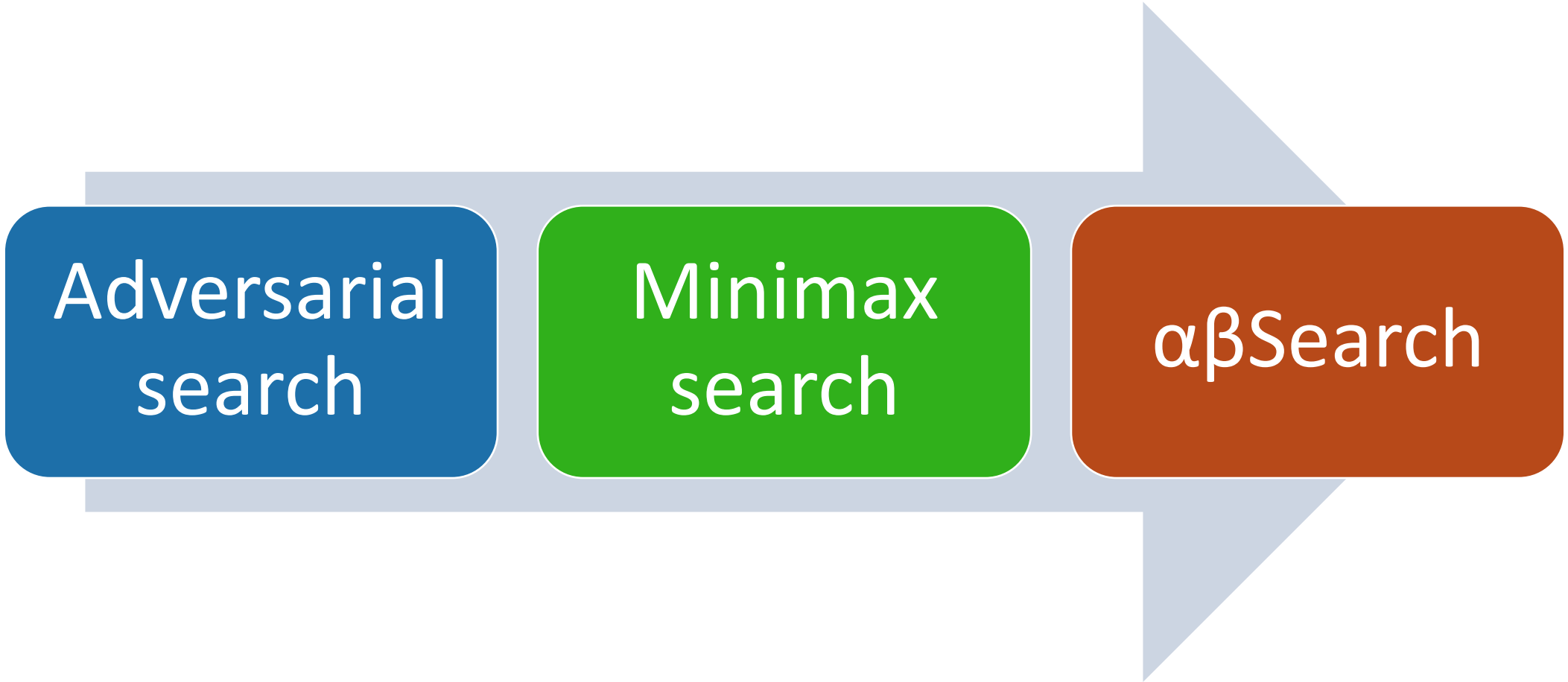
α-β pruning example



Algorithm illustration with α-β value



Summary



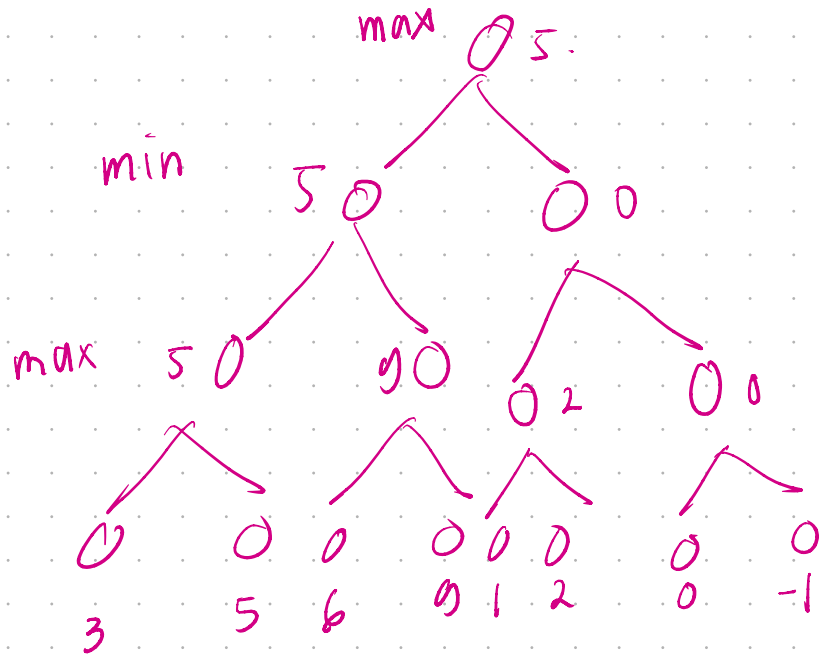

INSTITUT TEKNOLOGI BANDUNG
1920

Soal Adversarial Search

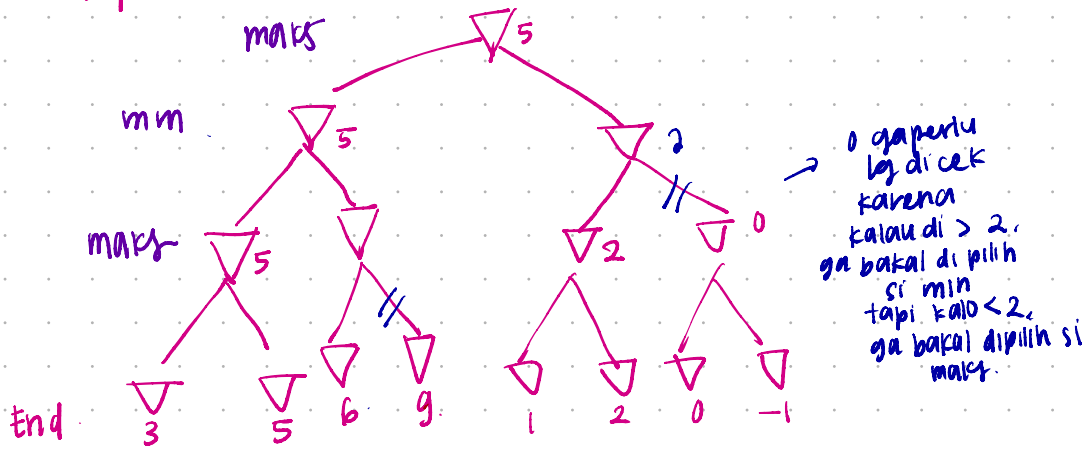
Terdapat suatu permainan berupa *3-ply game* sebagai berikut. Langkah pertama MAX yang bergerak dan memiliki dua kemungkinan langkah yaitu L atau R. Langkah berikutnya dilakukan MIN dengan dua kemungkinan langkah juga yaitu L atau R. Langkah terakhir dilakukan oleh MAX dengan dua kemungkinan langkah yaitu L atau R. Banyaknya kemungkinan urutan langkah adalah 8. Nilai pay-offs untuk MAX pada setiap kemungkinan urutan langkah adalah sebagai berikut: LLL = 3; LLR = 5; LRL = 6; LRR = 9; RLL = 1; RLR = 2; RRL = 0; dan RRR = -1. MAX berusaha memaksimalkan nilai payoffs nya, sedangkan MIN berusaha meminimalkan nilai payoffs MAX.

a) Gambarkan pohon *3-ply game* tersebut lengkap dengan nilai payoffs dari MAX pada setiap simpul pohon, dengan asumsi semua pemain memilih aksi secara rasional.

b) Terapkan *alpha-beta pruning* saat melakukan pencarian, dan ilustrasikan dengan gambar pohon untuk **setiap *pruning*** yang mungkin dilakukan pada cabang pohon dan jelaskan alasannya. Aksi L dieksplorasi terlebih dahulu sebelum aksi R dalam tahapan pencarian.


max
5
min
5
0
max
5
9
2
1
3
5
6
9
1
2
0
-1

alpha beta-pruning



→ 0 gaperlu lg dicek karena kalau di > 2, ga bakal dipilih si min tapi kalo < 2, ga bakal dipilih si maks.

kalo misal dia > 6 ga bakal dipilih min.

< 6, ga bakal dipilih maks.

α mengubah nilai maks.

Modul : Constraint Satisfaction Problem (CSP)

Terminologi Dalam CSP

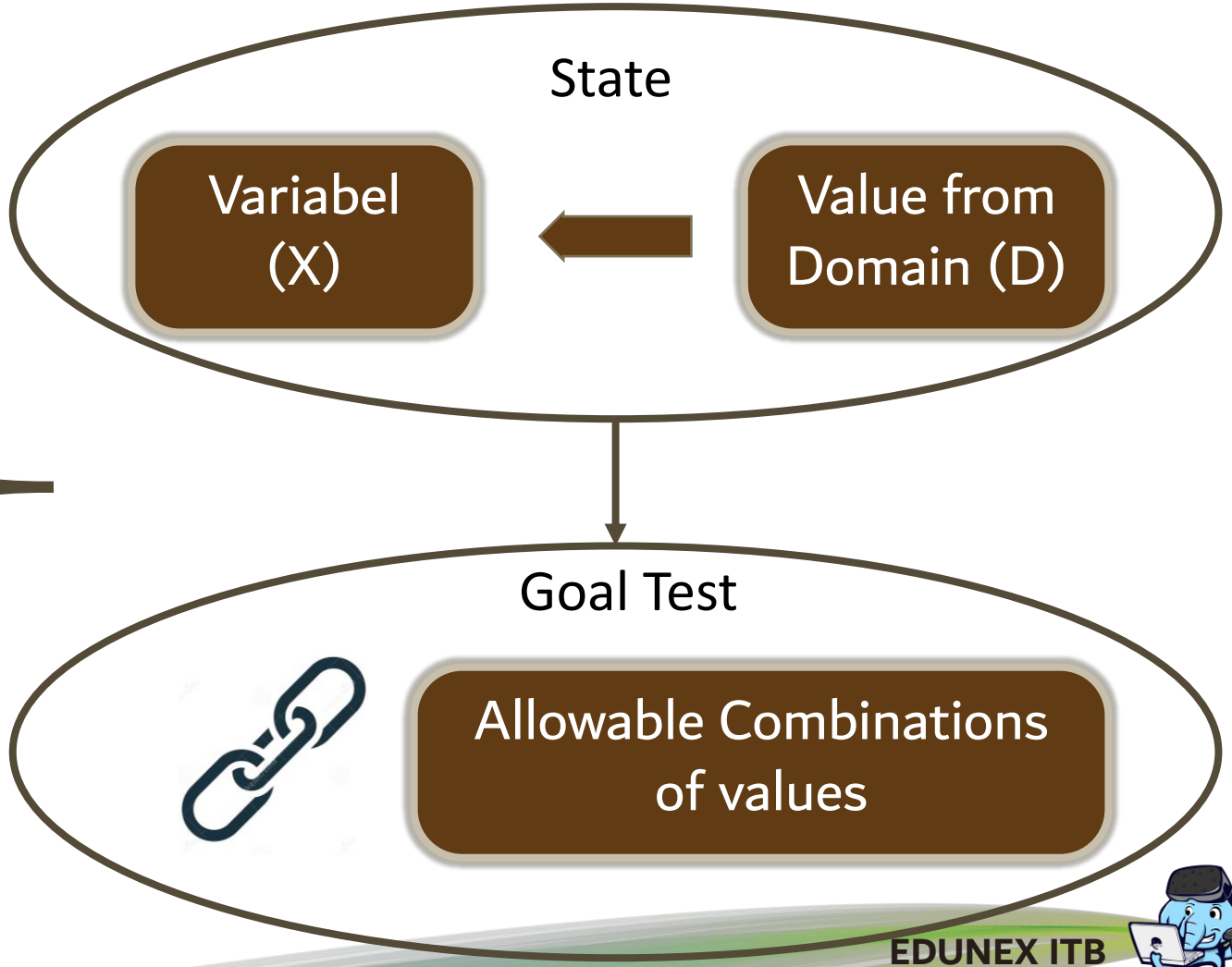
Nur ULFA Maulidevi

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

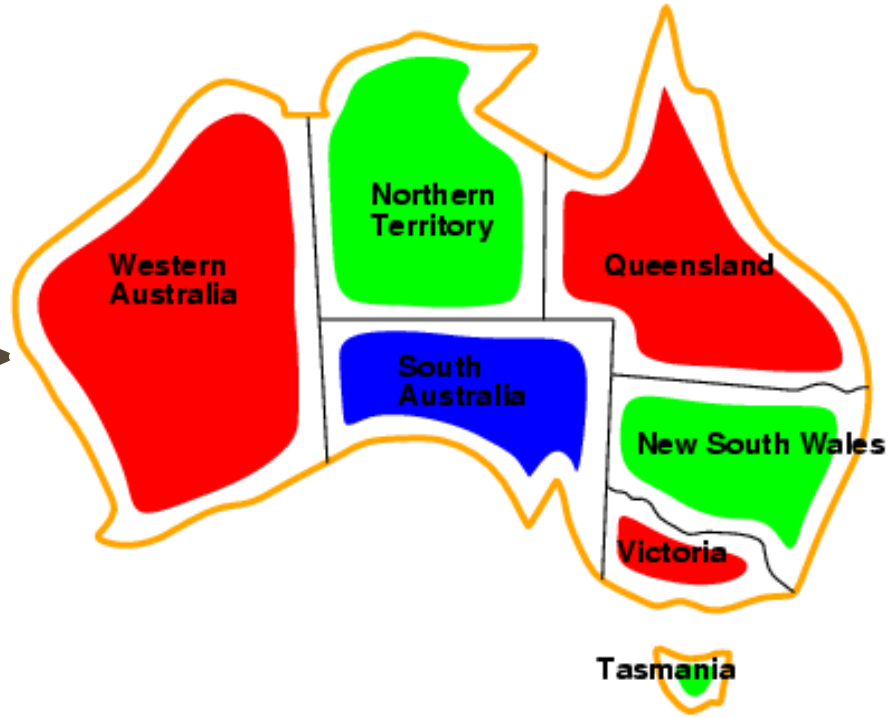


Example: Map-Coloring Problem

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

Domain:
D_i = {red,green,blue}

Constraints:
adjacent regions must have different colors



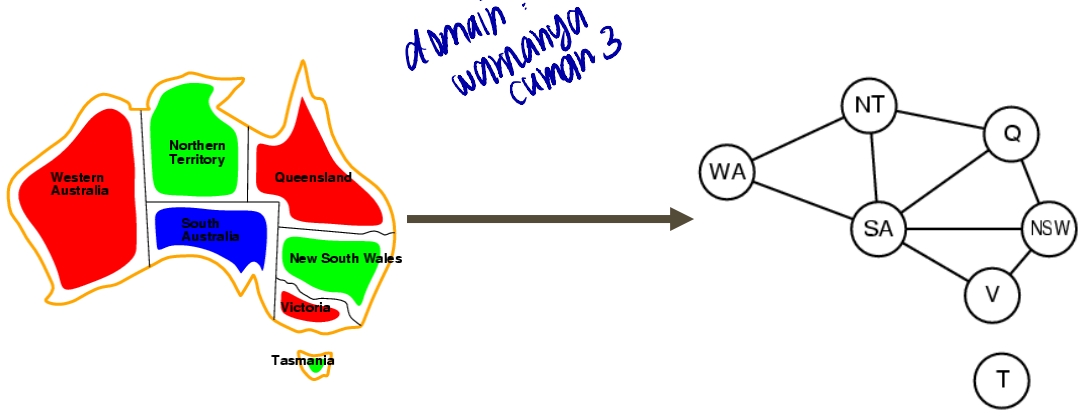
Solution: complete and consistent assignments

CSP Visualization

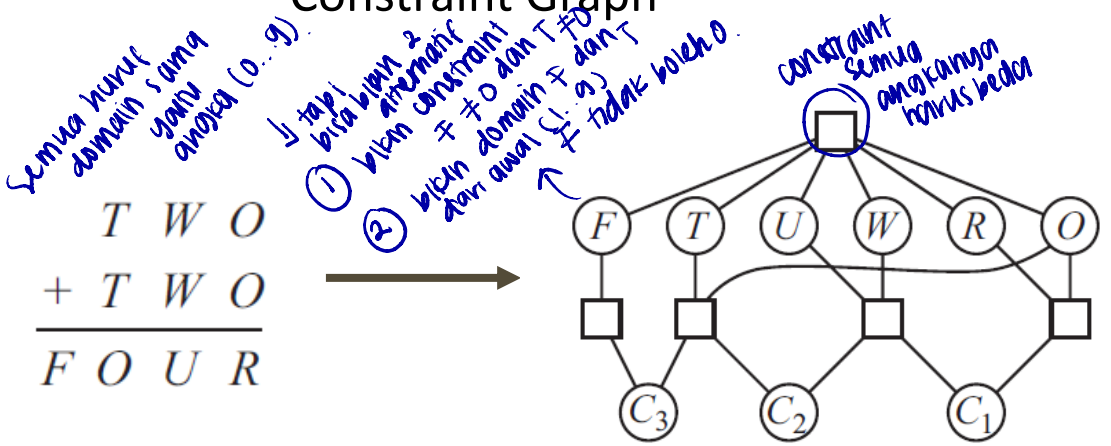
Variables → Nodes

Constraints → Link/ Arc

Constraints → Box

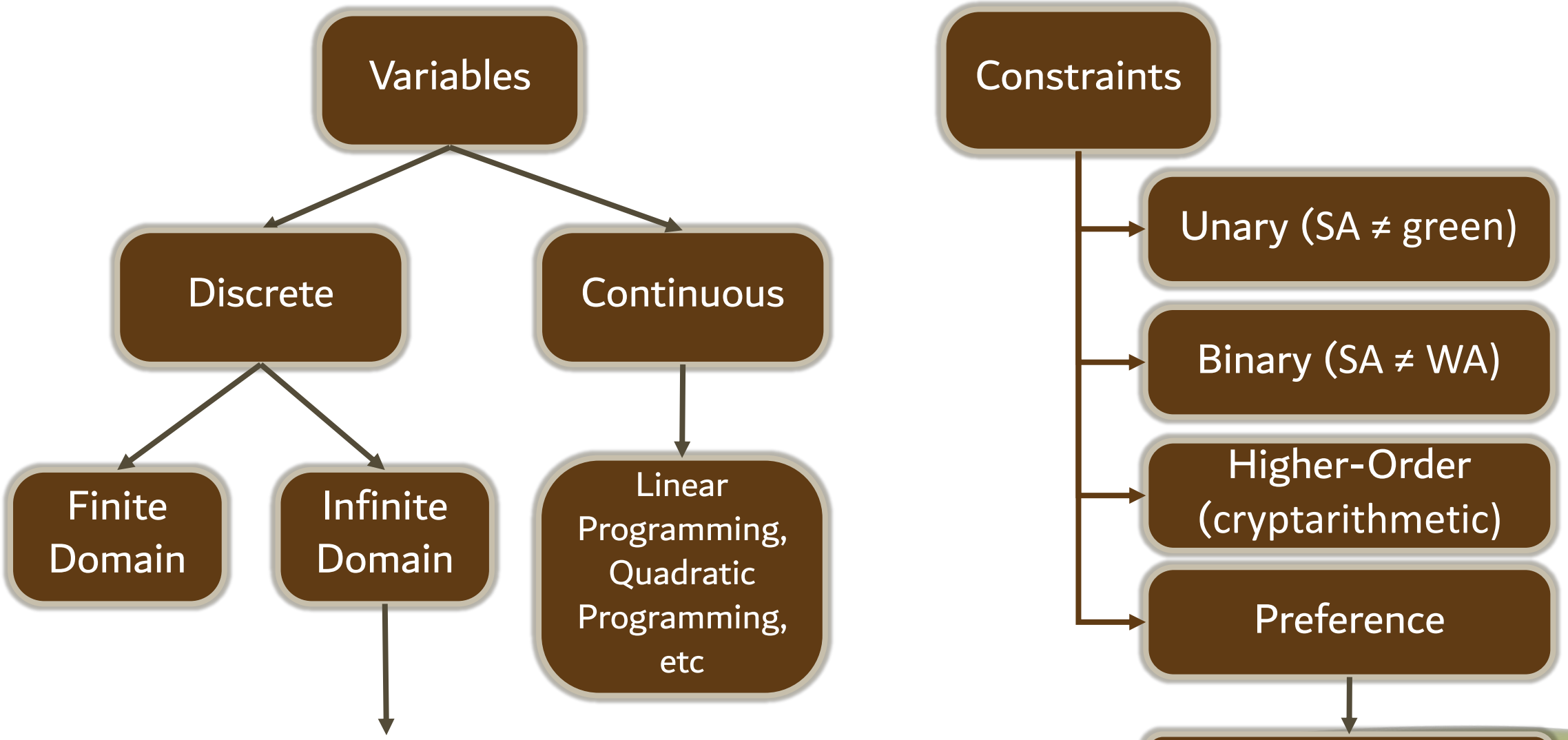


Constraint Graph



Constraint Hypergraph

Variations of CSP Formalism



Constraint Language: $StartJob_1 + 5 \leq StartJob_3$

Example: Cryptarithmetic Puzzle

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

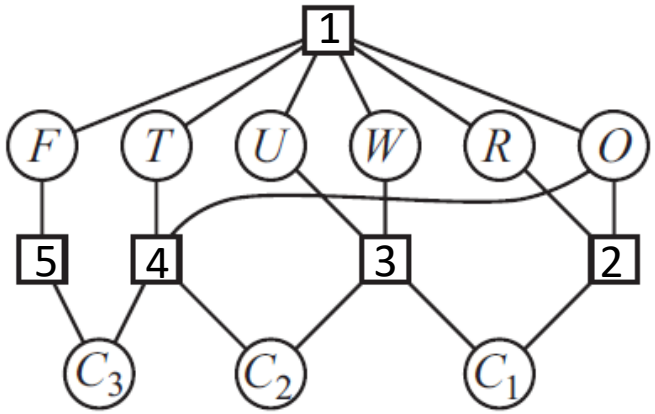
Variables

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)

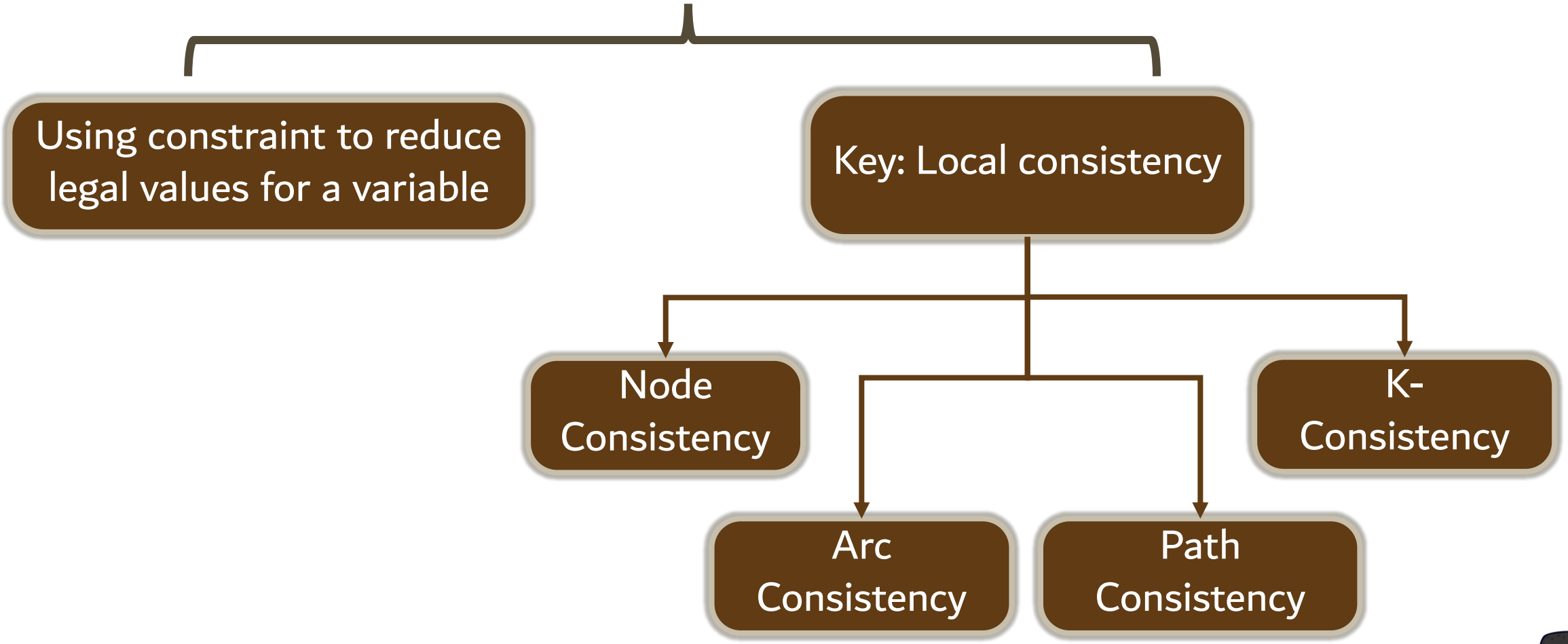
Inference in CSP

Nur ULFA Maulidevi

KK IF - Teknik Informatika- STEI ITB

Inteligensi Buatan
(Artificial Intelligence)

Constraint Propagation

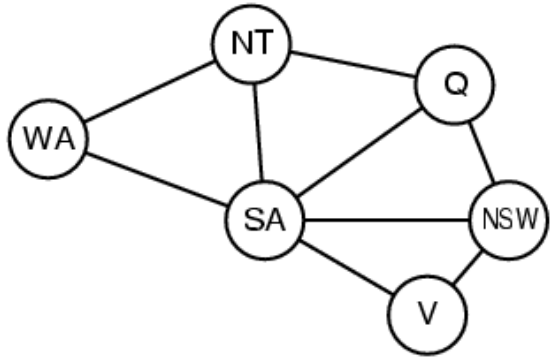


Node Consistency

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

Example: SA ≠ green

SA = {red, blue}

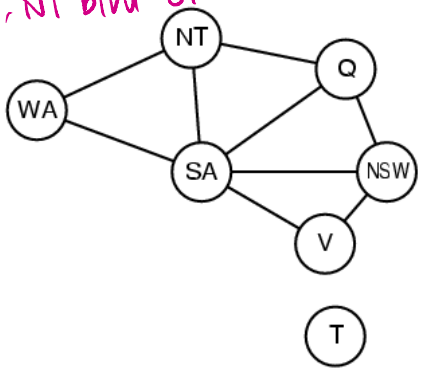


T mau warna apapun bebas (selama ada dlm domain) bcs dia ga terikat ke manapun

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

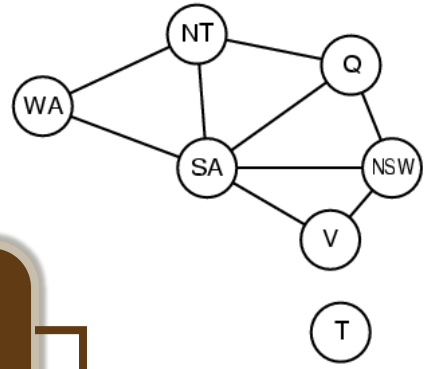
(SA,WA) = {(red , green), (red , blue), (green, red), (green, blue), (blue, red), (blue, green)}

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

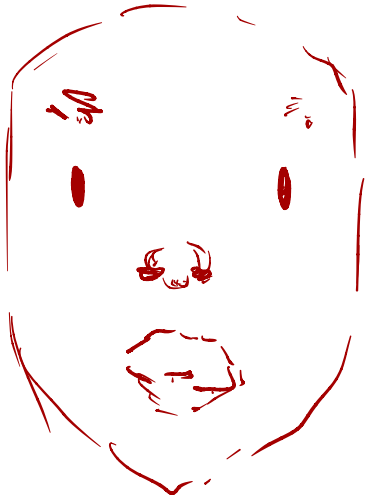
Algorithm

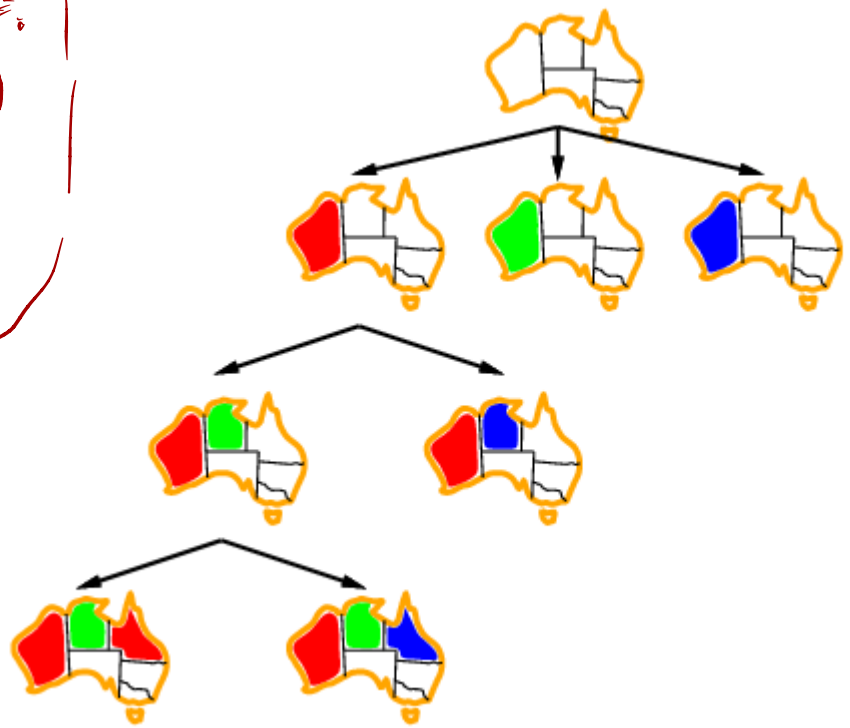
```
function BACKTRACKING-SEARCH(csp) returns a solution or failure
    return BACKTRACK(csp, { })

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
```

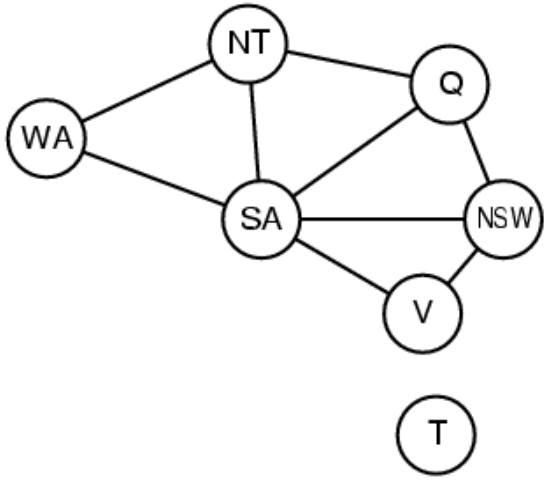
misal SA udah merah, WA dan NT kan gabolh merah, jadi merah tu diapus dr domain

inference : menghapus nilai yg membuat inconsistent dr domain

Example: Map Coloring Problem





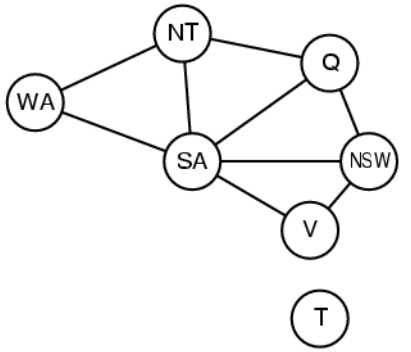


Improving Backtracking Efficiency

- 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)

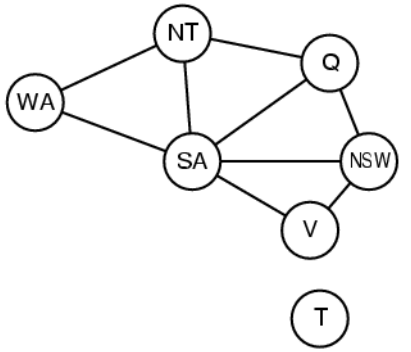
1. Static Variable Ordering: {*WA, NT, SA, Q, NSW, V, T* }

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



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

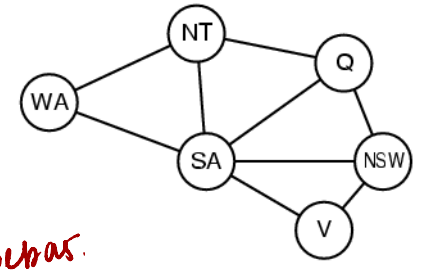
2. Minimum Remaining Values Heuristic : First Variable to Assign?

- 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.

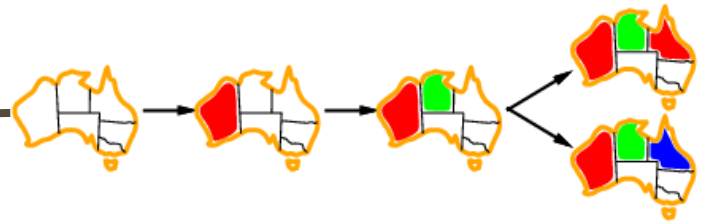
misal: urutan bebas.
isi WA merah
NT green
Q bisa merah/biru.
→ kalau Q biru, maka ketika inferensi SA habis.
→ kalau Q merah, SA masih ada.
jd pilih merah.

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

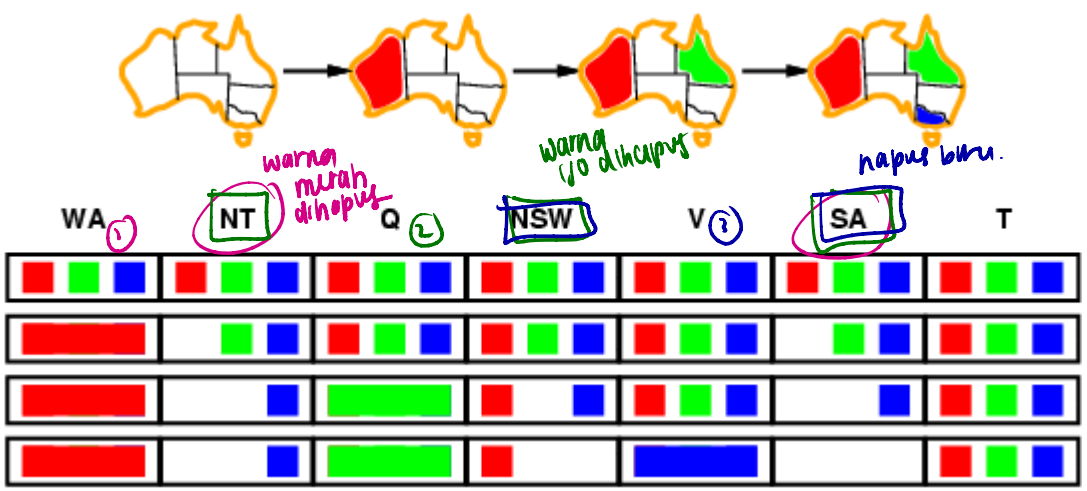
- Forward Checking: establishes arc consistency for binary constraint
- Constraint Propagation → Maintaining Arc Consistency (MAC)

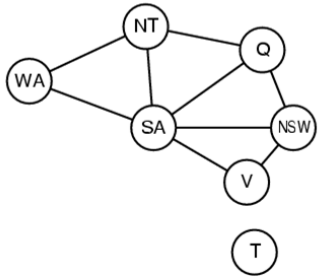
Let's see an example

Example: Interleave Search and Inference

Keep track of remaining legal values for unassigned variables

Terminate search when any variable has no legal values





FAIL!

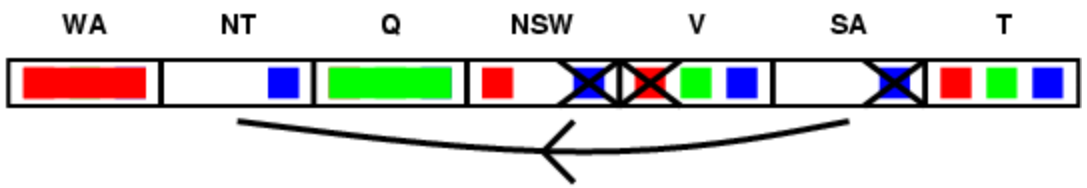
MAC: NT and SA cannot both be blue → MAC: repeatedly enforces constraints locally

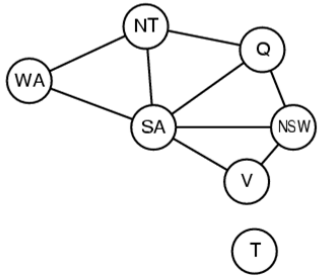
Example: Interleave Search and Inference

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

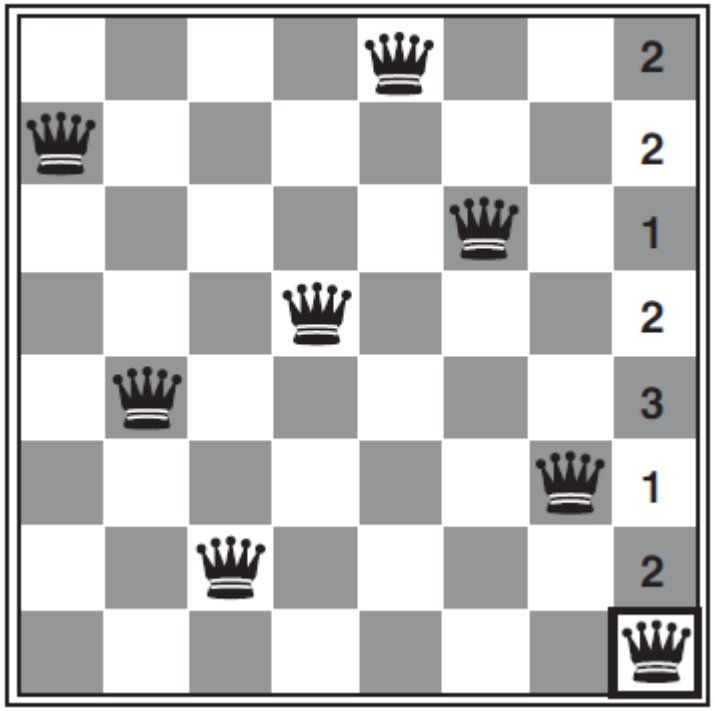
Value selection by min-conflicts heuristic

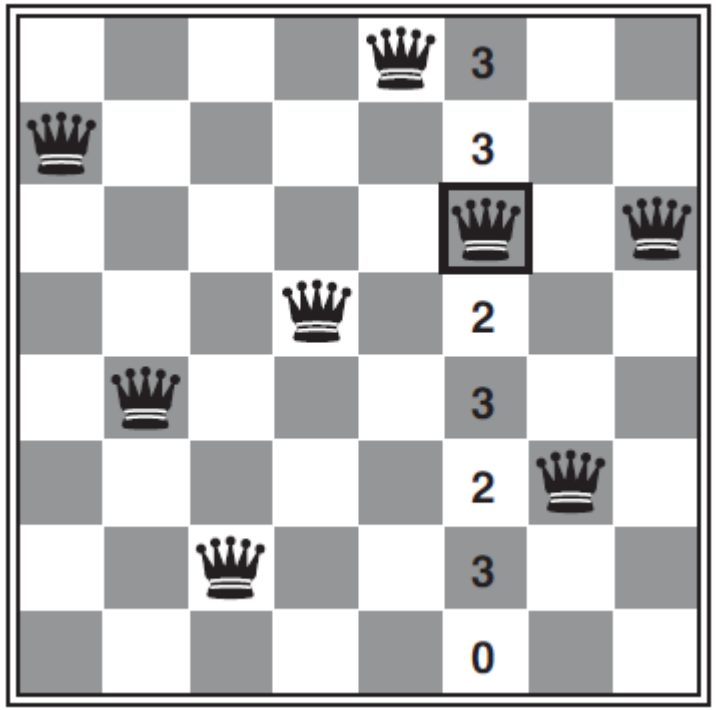
Min-Conflict Heuristic

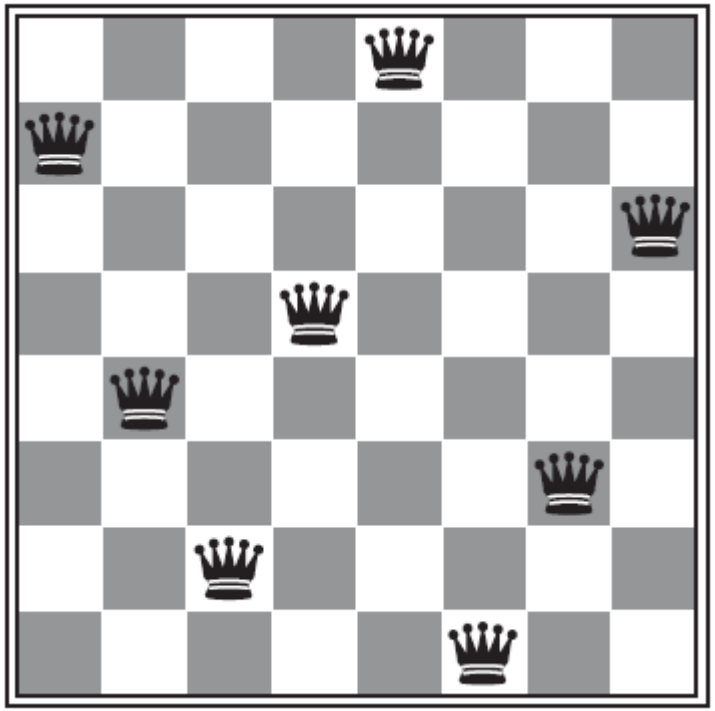
function MIN-CONFLICTS(csp, max_steps) **returns** a solution or failure
  inputs: csp, a constraint satisfaction problem
    max_steps, the number of steps allowed before giving up

  $current \leftarrow$ 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

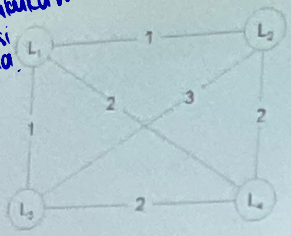
THANK YOU

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

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