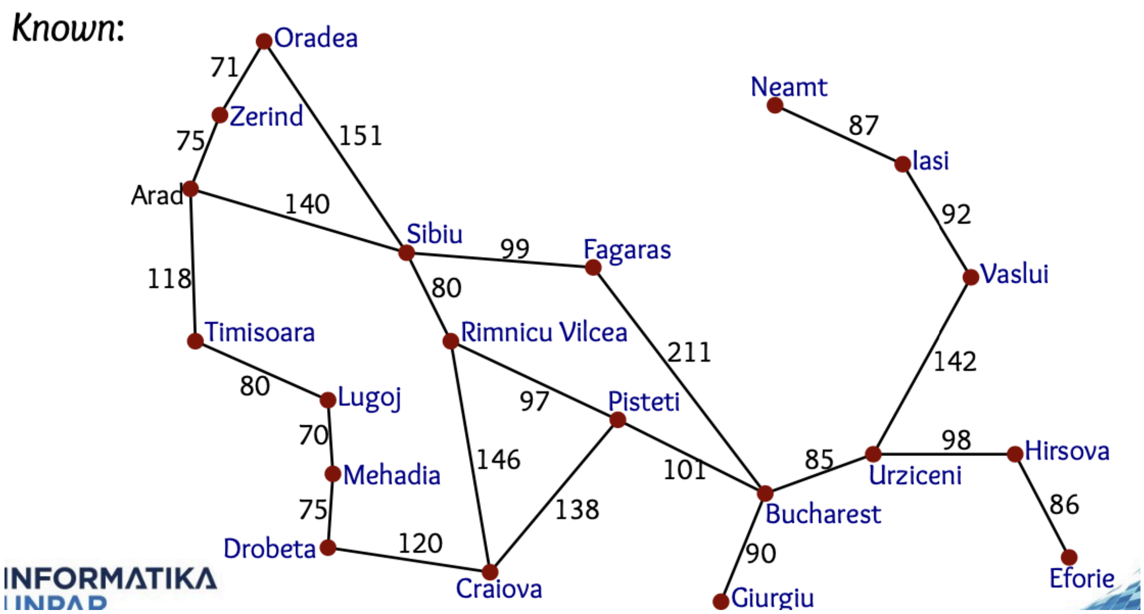
Mangat sobat gpt

Materi 3. Searching

Problem Solving Agent

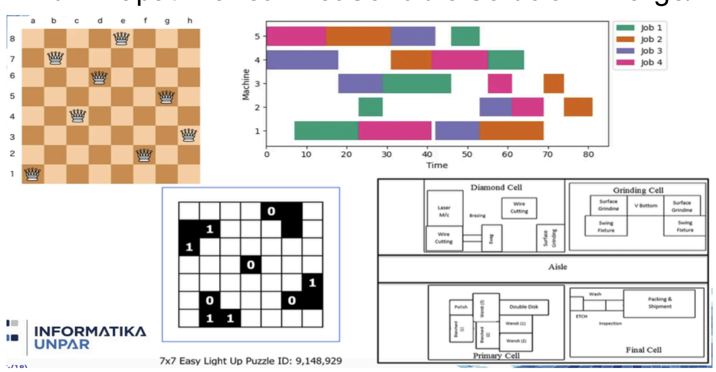
- **Agent** yang **mempertimbangkan** sequence of **actions** untuk **mencapai goal**
- Unknown = **memilih randomly** di antara beberapa neighbouring **cities**
- Known
    a. Goal Formulation = objectives/goals **membatasi** agent's **behavior**
    b. Problem Formulation = **deskripsi** dari **states** dan **actions**
    c. Search = **simulasi** sequence of **actions** yang **mencapai goal**
    d. Execution = **execute** actions dalam solution
- Fixed sequence of actions = if the environment is **known**, **fully observable**, **deterministic**



Materi 4. Complex Search

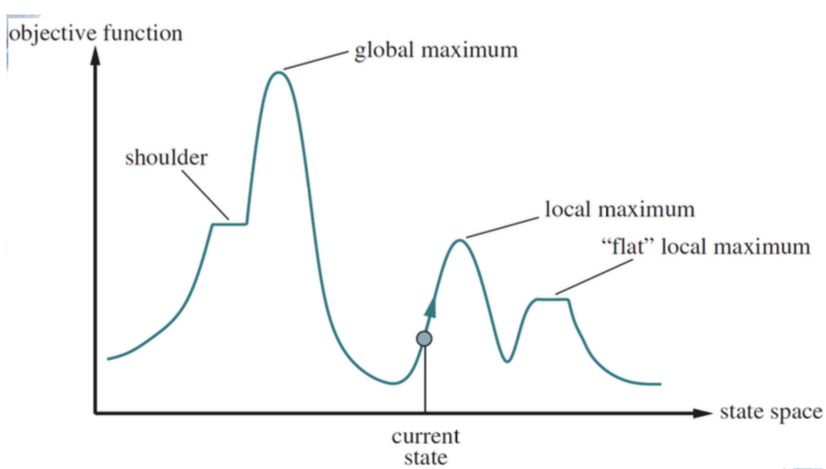
Local Search

- Hanya membutuhkan **final state**, **not** the **path**
- Dalam optimization problem, the aim is **mencari best state** yang **maximize** or **minimize objective function**
- Algorithm
    a. **Searching** from start state to neighboring states
    b. **Tidak keeping track** of the paths atau states yang reached
    c. Use **very little** computer **memory**
    d. Dapat **mencari reasonable solution** in large/infinite state spaceship



State-space Landscape

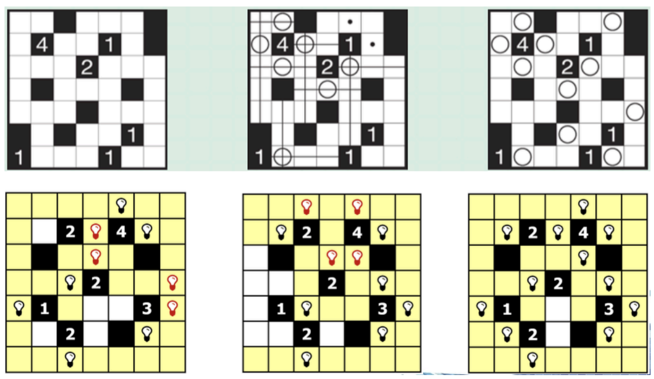
- Hill climbing = find global **maximum**
- Gradient descent = find global **minimum**



Hill Climbing

Algorithm

- **Start** di **random** state
- Setiap iterasi, **moves to** neighboring state with the **highest value**
- **Berakhir** saat mencapai “**peak**” (**no** neighbor has a **higher value**)
- Complete state formulation = setiap state memiliki semua components of a solution, tapi mungkin not in the right place



Kekurangan

- Local maxima = **stuck** at a **peak**, **higher than** each of its **neighboring** states but **lower than global maximum**
- Ridges = sequence of **local maxima** yang **sulit dinavigasi**
- Plateaus = **flat area**. Bisa **local maximum** or **shoulder**. **Shoulder** (solved by allowing **sideways move**), flat **local minimum** (solved by **limit** number of **sideways moves**)

Variants

- Stochastic = chooses **uphill moves randomly** dengan **probability** yang **related** dengan **steepness**. Usually converges (menuju satu titik/memusat) **slowly** than steepest ascent, but **might finds better solutions**
- First-choice = stochastic, tetapi **generating successor randomly sampai** one is **better** than **current** state. Good strategy when a state has **many neighboring** states
- Random-restart = **consecutive hill climbing** from **randomly** chosen initial states, **sampai goal** ditemukan

Random Walk

- Moves to **successor** state with **no khawatir** about **value**, pada akhirnya **reach global maximum**, but **extremely inefficient**

Simulated Annealing

- **Combine hill climbing** and **random walk** yang menghasilkan efficiency and completeness
- Annealing = proses **temper** or harden metals and **glass** dengan memanaskan with **high temperature** dan **then cooling**, maka material **mencapai** low energy **crystalline state**
- Consider gradient descent = **shake surface** to get ball **fall** into the **deepest celah** in a bumpy surface. Shake to **escape local minima**
- Starts by **shaking hard** and gradually **decrease** the **intensity**

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

Local Beam Search

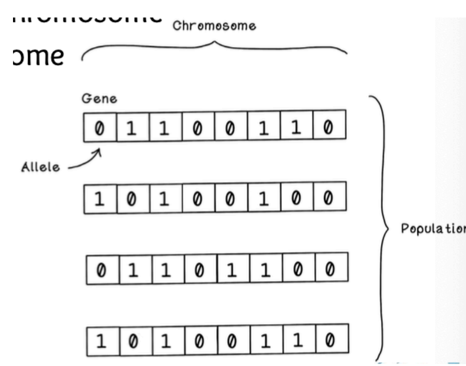
- **Mulai** dari **random** state and **keeps track** of states in parallel
- Not random-restart HC in parallel = useful **information** is **passed among** the parallel **search threads**, seperti **move** to the most **promising states**
- Lack of diversity -> **slower version** of **Hill Climbing**
- Stochastic beam search = increase diversity by **milih neighbor** dengan **probability proportional** terhadap their **values**, **instead** of milih **best neighbor**

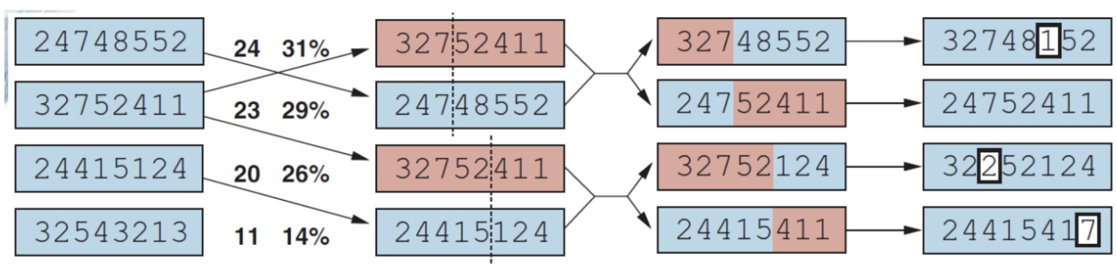
Evolutionary Algorithm

- Population-based **metaheuristic optimization** algorithm
- Problems are modeled in:
    a. Individual = state
    b. Fit better = higher value of a state
    c. Offspring = successor/neighboring state
    d. Recombination = generate another state
- Elements:
    a. **Size** of population
    b. **Fitness function** menentukan quality of individual
    c. **Selection** process untuk **memilih individuals** who will **reproduce**
    d. **Mutation rate** menentukan seberapa sering **offspring have** random **mutations**

Genetic Algorithm

- Sub-class of evolutionary algorithms dimana setiap **individu** is a **string** of **finite alphabet**
- Population = collection of **chromosomes**
- Chromosome of genotype = candidate **solution**
- Phenotype = actual **representation** of chromosome
- Gene = **logical type** of unit in a chromosome
- Alele = **actual value** of a gene
- Operator:
    a. Selection = selection of two parents
    b. Crossover = reproduction
    c. Mutation = to maintain genetic diversity

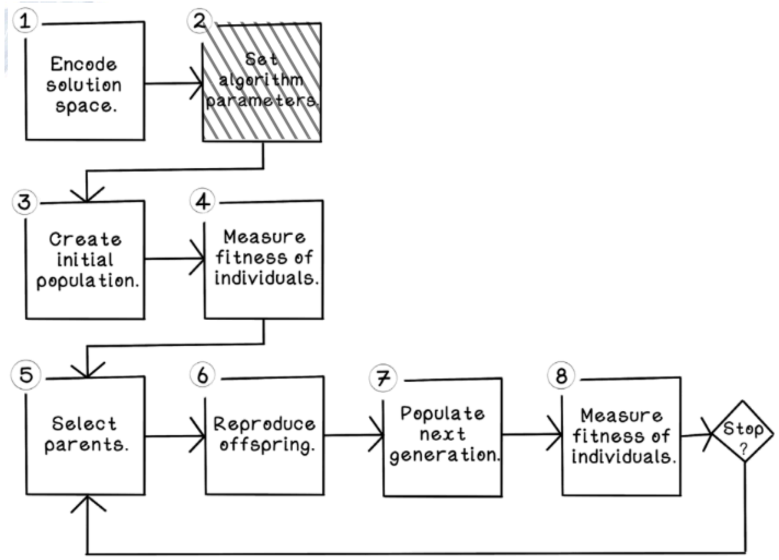




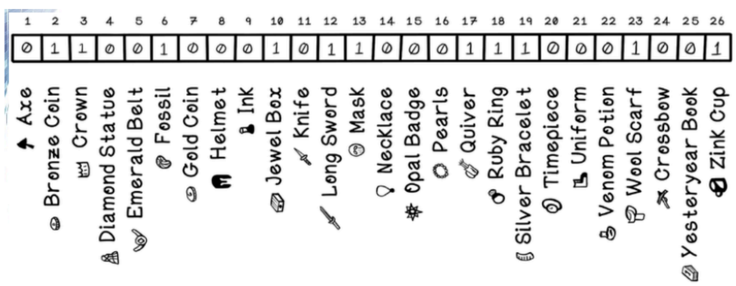
(a) Initial Population (b) Fitness Function (c) Selection (d) Crossover (e) Mutation

a. Populasi awal
b. Fitness function menentukan quality of individual
c. Memilih individu terbaik dari populasi berdasarkan nilai fitness
d. Kombinasi dua orang tua
e. Perubahan acak pada individu to maintain diversity, menghindari **local optimum**

Genetic Algorithm Life Cycle



Step 1. Encode Solution Space



Step 2. Set Algorithm Parameters

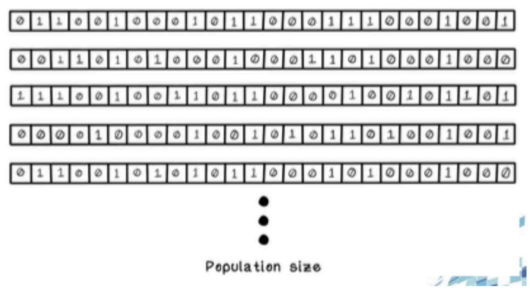
Configuring parameter

- Chromosome **encoding**
- Population **size**
- Population **initialization**
- Number of **offspring**
- Parent **selection** method
- **Crossover** method and rate
- **Mutation** method and rate
- **Generation** selection method
- **Stopping** condition

Step 3. Create Initial Population

- **Generate** some possible **solutions**
- Initialize **random** but **valid potential solutions**

```
generate_initial_population (population_size, individual_size)
  let population be an empty array
  for individual in range 0 to population_size
    let current_individual be an empty array
    for gene in range 0 to individual_size
      let random_gene be 0 or 1 randomly
      append random_gene to current_individual
    append current_individual to population
  return population
```



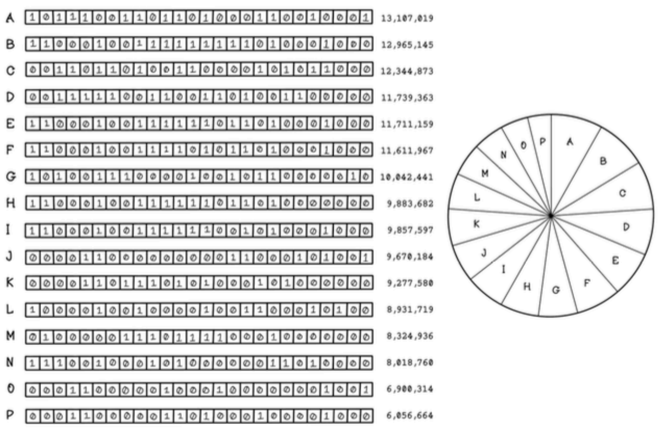
Step 4. Measure Fitness of Individuals

- Defines how well a solution performs
- Contoh:
    a. Minimize (traveling salesman)
        - Fitness = 1/cost
        - Semakin **kecil** cost (jarak perjalanan/waktu), semakin **besar** fitness
    b. Maximize (knapsack)
        - Fitness = total value of items
        - Semakin **besar** value (barang yang dapat dimuat dalam tas), semakin **besar** fitness
    c. Persamaan (equation)
        - Fitness = 1 / |target value - calculated value| + 1
        - Semakin **kecil** perbedaan target dengan calculated, semakin **besar** fitness
    d. Klasifikasi atau pengelompokan (machine learning)
        - Fitness = accuracy or fitness = 1/error

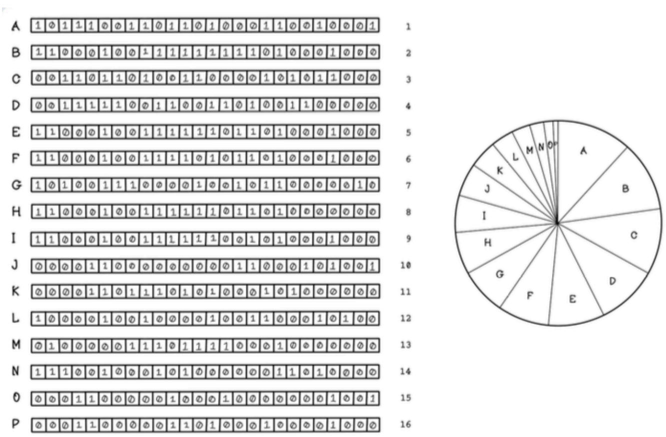
Step 5. Select Parents

- Selecting parents based on their fitness
- To control diversity
    a. Steady state = replace a **portion** of the **population**, **weaker** individuals will be **removed**
    b. Generational = replace the **entire population**
- Rank selection = **ranking** individuals based on their fitness
- Contoh lain = Roulette-wheel selection. **Each individual have portions**. **Selecting** parents and creating offsprings **with different parents** selected (dengan kemungkinan individu yang sama become a parent more than once) **hingga** jumlah **offspring** yang diinginkan **tercapai**
- Tournament selection = chooses individuals and **places** them **in** a **group**

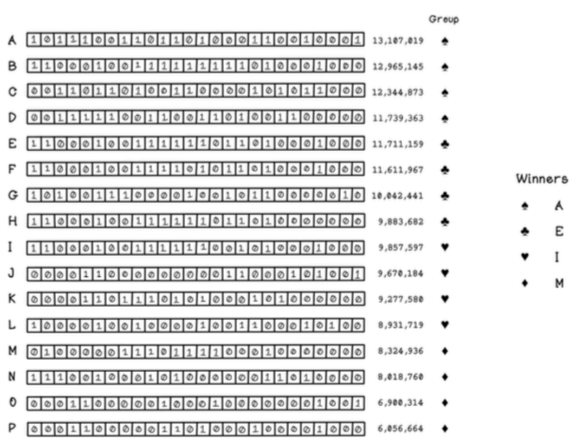
Roulette-wheel Selection



Rank Selection

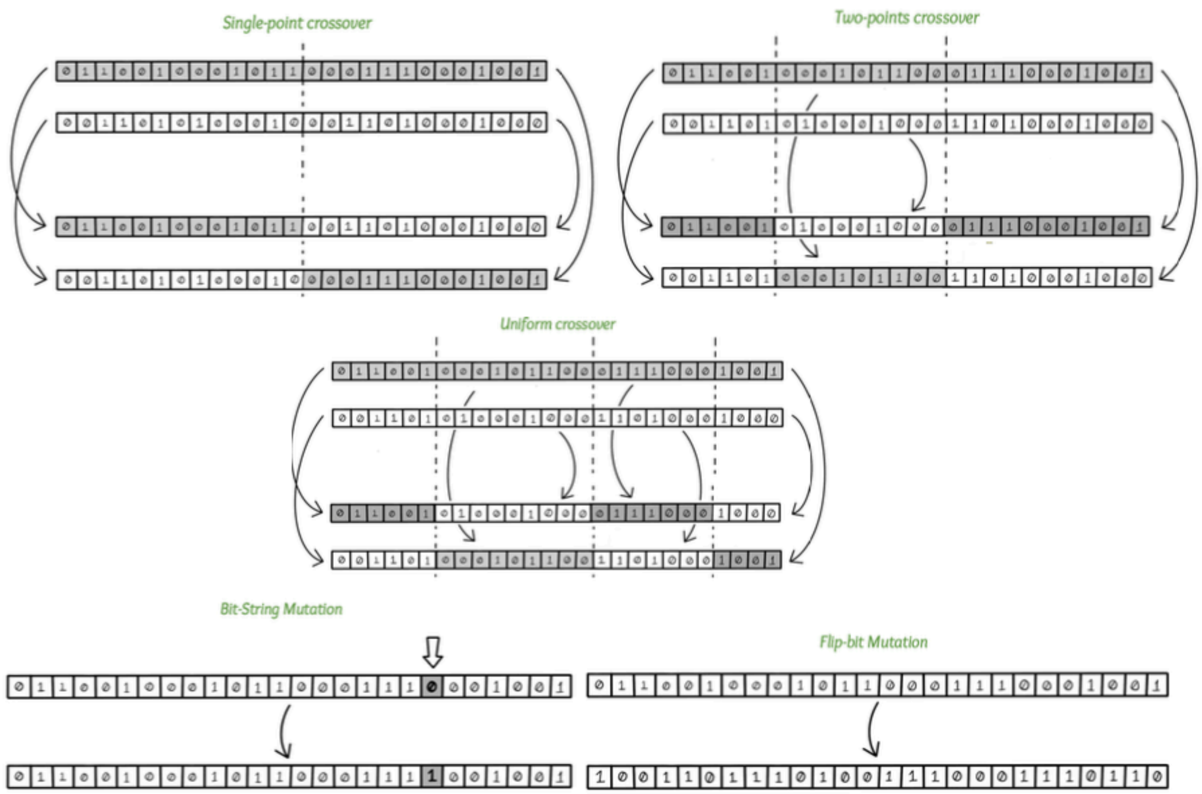


Tournament Selection



Step 6. Reproduce Offspring

- Crossover = **mixing** part of the **chromosome**
- Mutation = **changing offspring** sedikit untuk diversity



Step 7. Populate Next Generation

- Selecting **which** individuals **live** on the **next generation**
- Elitism = **mempertahankan strong**-performing **individuals**

Step 8. Measure Fitness of Individuals

- Measuring the performance of the solutions
- How well do these solutions solve the problem

Step 9. Stopping Condition

- Set a **constant value** as number of generations
- **Stagnasi** = sedikit perubahan atau perkembangan for several generations

Materi 5. Swarm Intelligence

Swarm Intelligence

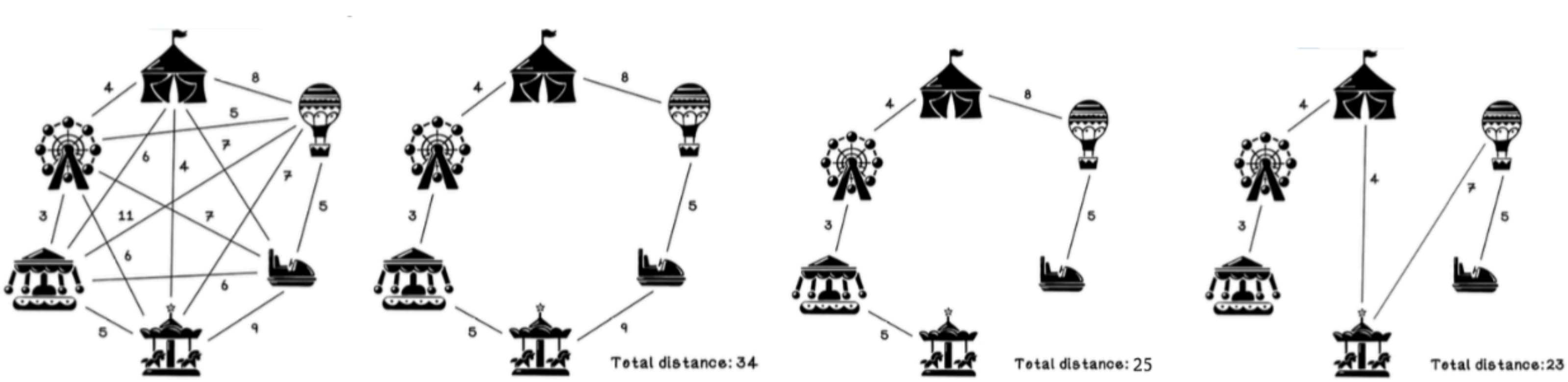
- Swarm = **group** of moving **individuals** that **communicate** with each other, either **directly** or by **acting** on their local environment
- Uses
    a. Clustering = determine **groupings** of **data** with similar features
    b. Optimization = determine global **optimal value**
    c. Scheduling = for **efficient** outcome
    d. Routing = determine **efficient paths**

Ant Colony Optimization

- Can be used in
    a. Route Optimization
    b. Job Scheduling
    c. Image Processing

The Carnival Problem

- Find **shortest path** between **all attractions** while visit the carnival

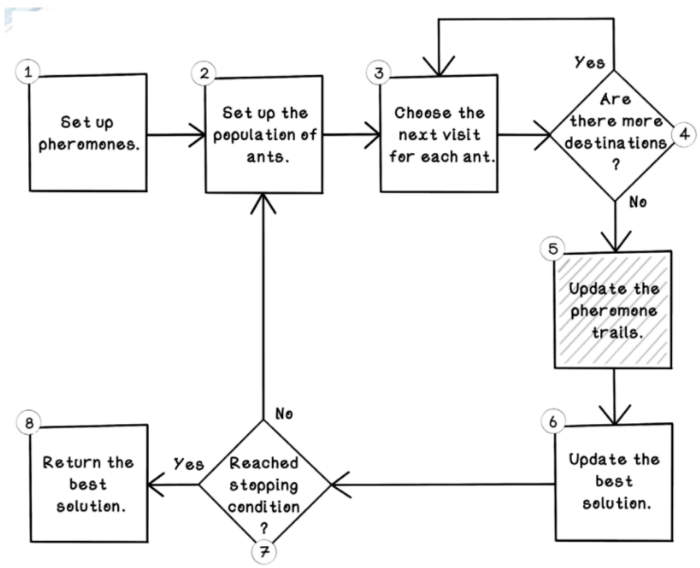


Distances on the paths | A possible route to all attractions | A possible route to all attractions | The best possible route to all attractions

Ant Representation

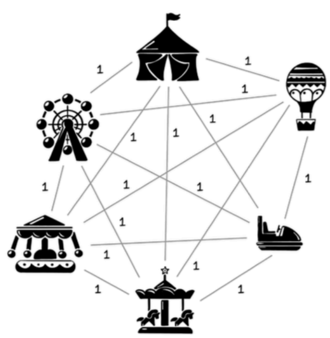
- 3 Basic properties
    a. Memory = list of **attractions** already **visited**
    b. Best fitness = **shortest total** distance
    c. Action = **choose next** attractions and **drop pheromones** sepanjang jalan

Ant Colony Optimization Life Cycle



Step 1. Set Up Pheromones

- Important = **set** all pheromone **trails** to **1**, so tidak ada trail yang memiliki advantage over others



Pheromones initialize at 1

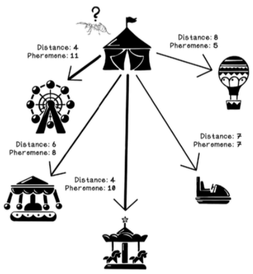
Step 2. Set Up the Population of Ants

- Ants **start** at **randomly** assigned attractions

Step 3. Choose the Next Visit for Each Ant

- **Decides next** destination based on **pheromone** trails and **heuristic**
- The influence is **controlled** by **a** and **b**

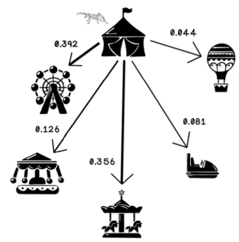
$$\frac{(\text{pheromones on path } x)^a * (1 / \text{heuristic for path } x)^b}{\sum_{\text{sum of n available destinations}} ((\text{pheromones on path } n)^a * (1 / \text{heuristic for path } n)^b)}$$

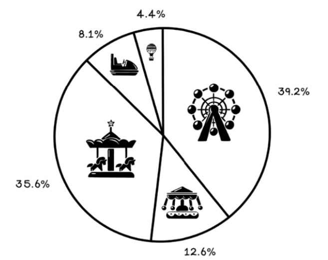


$((\text{pheromones on path } x)^a * (1 / \text{heuristic for path } x)^b)$

Ferris wheel:	$11 * (1/4)^2 = 0.688$
Swings:	$8 * (1/6)^2 = 0.222$
Carousel:	$10 * (1/4)^2 = 0.625$
Bumper cars:	$7 * (1/7)^2 = 0.143$
Balloons:	$5 * (1/8)^2 = 0.078$

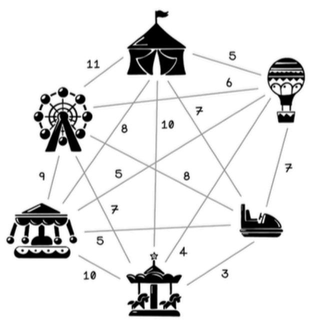
Ferris wheel:	**0.688 / 1.756 = 0.392**
Swings:	0.222 / 1.756 = 0.126
Carousel:	**0.625 / 1.756 = 0.356**
Bumper cars:	0.143 / 1.756 = 0.081
Balloons:	0.078 / 1.756 = 0.044



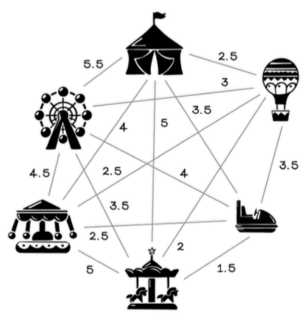


Step 4. Are There More Destinations?

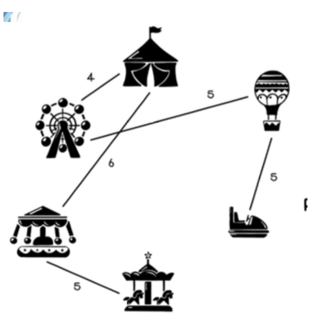
Step 5. Update the Pheromone Trails



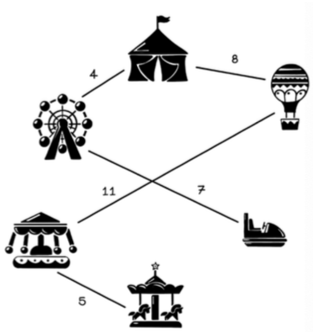
Pheromones on paths



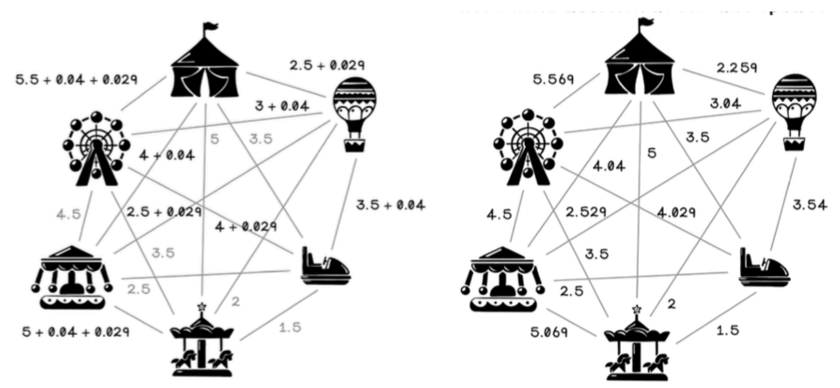
Pheromones on paths after 50% evaporation



Ant A total: 25
1/25 = 0.04



Ant B total: 35
1/35 = 0.029



Pheromone addition after ant update   Pheromones after ant update

Step 6. Update the Best Solution

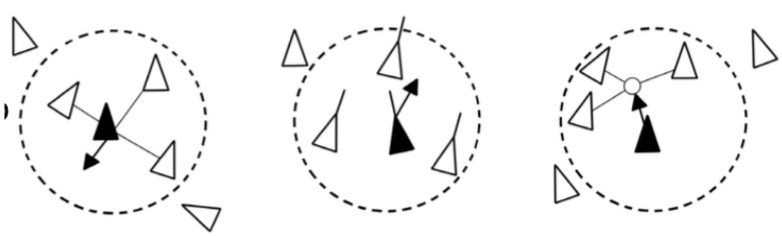
Step 7. Reached Stopping Condition?

- Options
    a. Number of **iterations** is **reached**
    b. When the best solutions **stagnates**

Step 8. Return the Best Solution

Flock of Birds

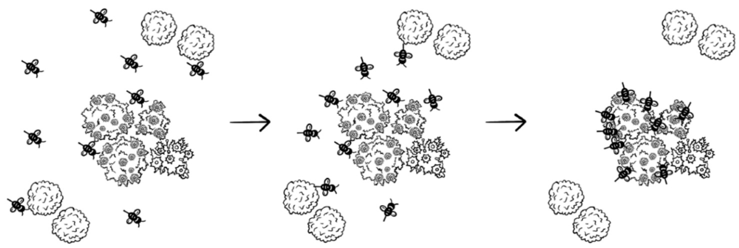
- Behaviors that guide the group
    a. Separation = avoid **colliding**
    b. Alignment = **maintain** the **formation**
    c. Cohesion = ensure **travels** in a **similar direction**



Separation   Alignment   Cohesion

Particle

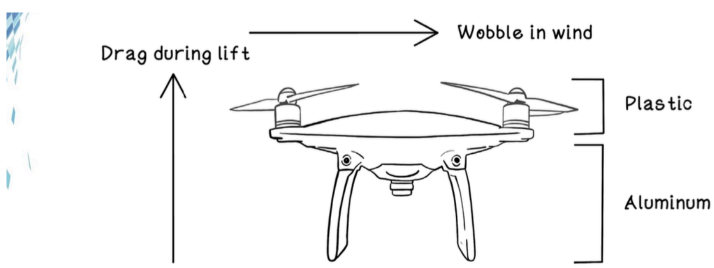
- Group of individuals flown through
- Contoh = swarm of bees **looking** for **flowers** and gradually **converges** on area that has **most density** of flowers



- Can be used in
    a. Optimizing weights in an artificial neural network
    b. Motion tracking in videos
    c. Speech enhancement in audio

The Drone Problem

- Find a good ratio of plastic to aluminum that reduces drag during lift and wobble in the wind



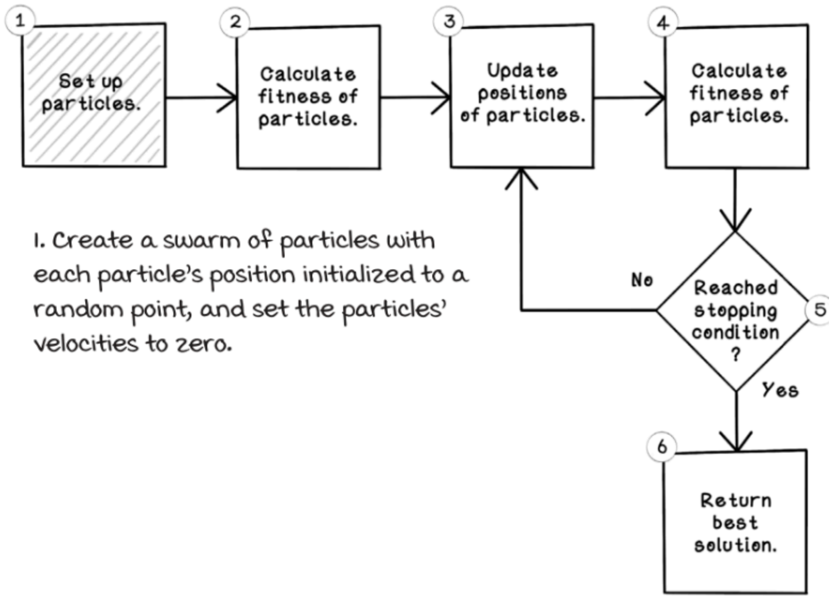
The fitness function: the ratio between aluminum (x) and plastic (y):

$$f(x, y) = (x + 2y - 7)^2 + (2x + y - 5)^2$$

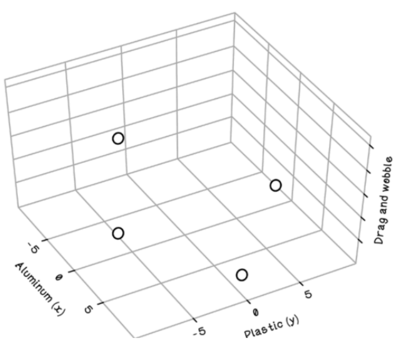
Particle Representation

- Current position
- Best position using fitness function
- Current velocity

Particle Swarm Optimization Life Cycle



Step 1. Set Up Particles



- Number of particles
- Starting position = random position distributed
- Starting velocity = initialize to 0

Step 2. Calculate Fitness of Particles

$$f(x, y) = (x + 2y - 7)^2 + (2x + y - 5)^2$$

$$f(7,1) = (7 + 2(1) - 7)^2 + (2(7) + 1 - 5)^2 = 104$$

$$f(-1,9) = (-1 + 2(9) - 7)^2 + (2(-1) + 9 - 5)^2 = 104$$

$$f(-10,1) = (-10 + 2(1) - 7)^2 + (2(-10) + 1 - 5)^2 = 801$$

$$f(-2,-5) = (-2 + 2(-5) - 7)^2 + (2(-2) - 5 - 5)^2 = 557$$

Particle	Velocity	Current aluminum (x)	Current plastic (y)	Current fitness	Best aluminum (x)	Best plastic (y)	Best fitness
1	0	7	1	104	7	1	104
2	0	-1	9	104	-1	9	104
3	0	-10	1	80	-10	1	80
4	0	-2	-5	365	-2	-5	365

Step 3. Update Positions of Particles

Step 4. Calculate Fitness of Particles

- Three components to calculate new velocity
    a. Inertia
        - **Resistensi** terhadap **movement** atau **change** in **direction**
        - Component = Inertia * current velocity
    b. Cognitive
        - **Ability** of a **specific particle**
        - **Use** its known **best position** in swarm **to influence** its **movement**
        - Component = Cognitive acceleration = cognitive constant * random cognitive number
        - Cognitive acceleration * (particle best position - current position)
    c. Social
        - **Ability** of a **particle** to **interact** with **swarm**
        - Component = Social acceleration = social constant * random social number
        - Social acceleration * (Social best position - social position)
- New velocity = inertia component + cognitive component + social component

Particle	Velocity	Current aluminum	Current plastic	Current fitness	Best aluminum	Best plastic	Best fitness
1	0	7	1	104	7	1	104
2	0	-1	9	104	-1	9	104
3	0	-10	1	80	-10	1	80
4	0	-2	-5	365	-2	-5	365

Inertia = 0.2
Cognitive constant = 0.35 Random cognitive number = 0.2
Social constant = 0.45 Random social number = 0.3

Particle	Velocity	Current aluminum	Current plastic	Current fitness	Best aluminum	Best plastic	Best fitness
1	0	7	1	104	7	1	104
2	0	-1	9	104	-1	9	104
3	0	-10	1	801	-10	1	801
4	0	-2	-5	365	-2	-5	365

Particle	Velocity	Current aluminum	Current plastic	Current fitness	Best aluminum	Best plastic	Best fitness
1	2.295	7	1	104	7	1	104
2	1.626	-1	9	104	-1	9	104
3	2.043	-10	1	801	-10	1	801
4	1.35	-2	-5	365	-2	-5	365

Particle	Velocity	Current aluminum	Current plastic	Current fitness	Best aluminum	Best plastic	Best fitness
1	2.295	9.925	3.295	419.776	7	1	104
2	1.626	0.626	10.626	268.662	–1	9	104
3	2.043	–7.957	3.043	398.067	–7.957	3.043	398.067
4	1.35	-0.65	-3.65	322.505	-0.65	-3.65	322.505

Step 5. Reached Stopping Condition?

- Options
    a. Number of iterations is reached
    b. When the best solutions stagnates

Step 6. Return Best Solutions

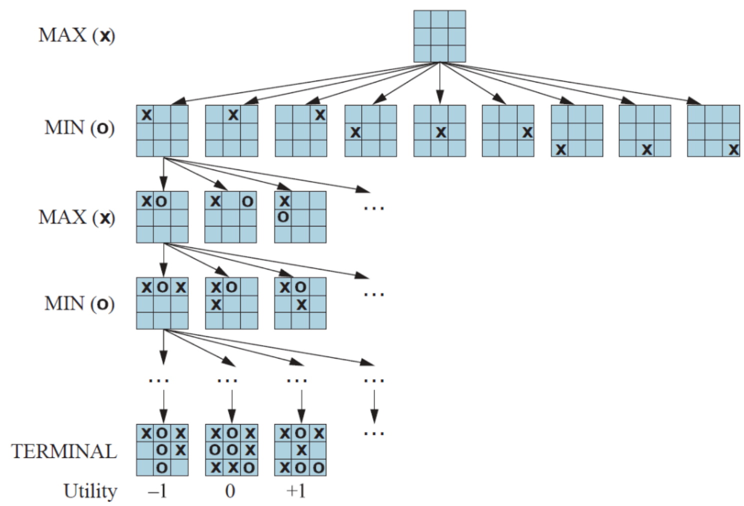
Materi 6. Adversarial Search

Background

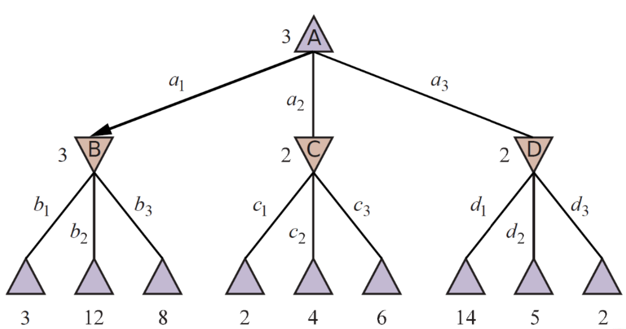
- Involving **people opposing** or **disagreeing** with **each other**
- Competitive environment = two or more **agents** have **conflicting goals**
- Zero-sum game with environment = **deterministic**, **fully observable**, **sequential** (taking turn)
- Tic-Tac-Toe, Chess, Go, Checkers

State-space Graph in a Game

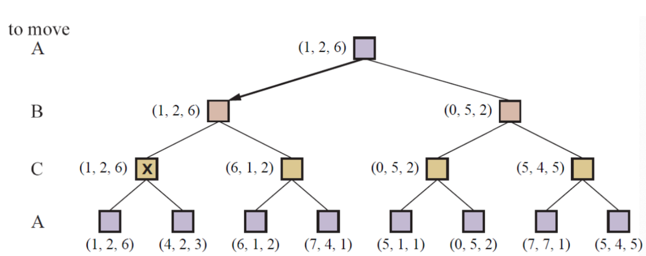
Terminal = the **game** is **over**
Utility = **final numerical value** to players **when** the **game ends**



Minimax Search



Level 2 mengambil nilai MIN dari daun paling bawah
Level 1 mengambil nilai MAX dari level 2 (3, 2, 2)



Level 3 lihat angka ke-3 dari dalam kurung dari daun paling bawah, ambil MAX (6 > 3)
Level 2 lihat angka ke-2 dari dalam kurung dari level 3, ambil MAX (2 > 1)
Level 2 lihat angka ke-1 dari dalam kurung dari level 2, ambil MAX (1 > 0)