

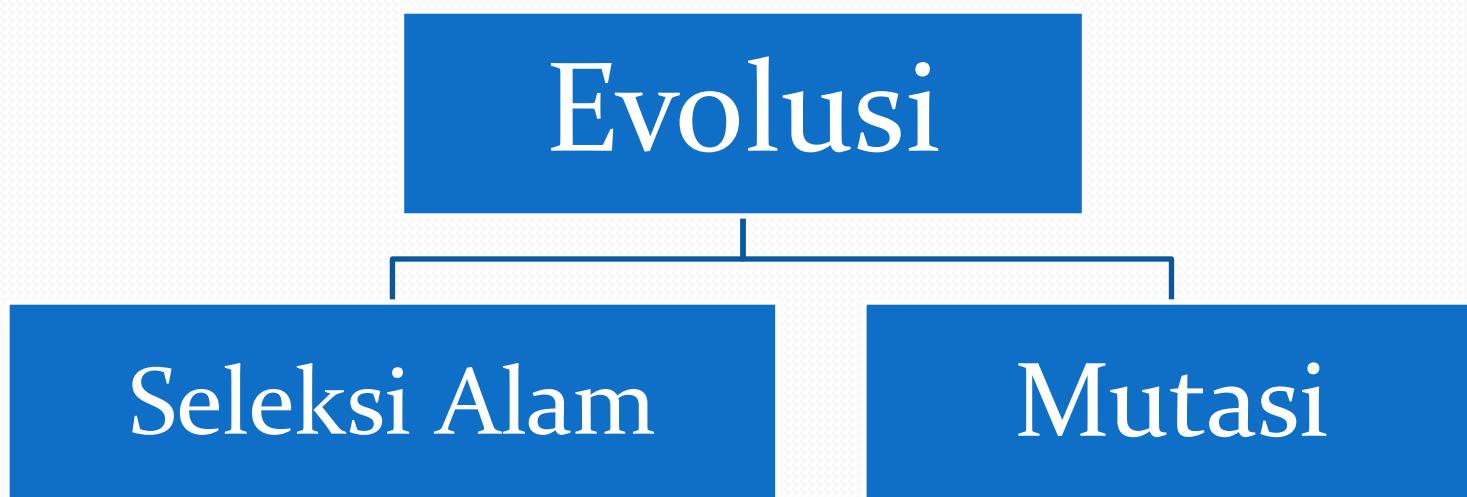
Pendahuluan

Dr. Suyanto, S.T., M.Sc.
HP/WA: 0812 845 12345

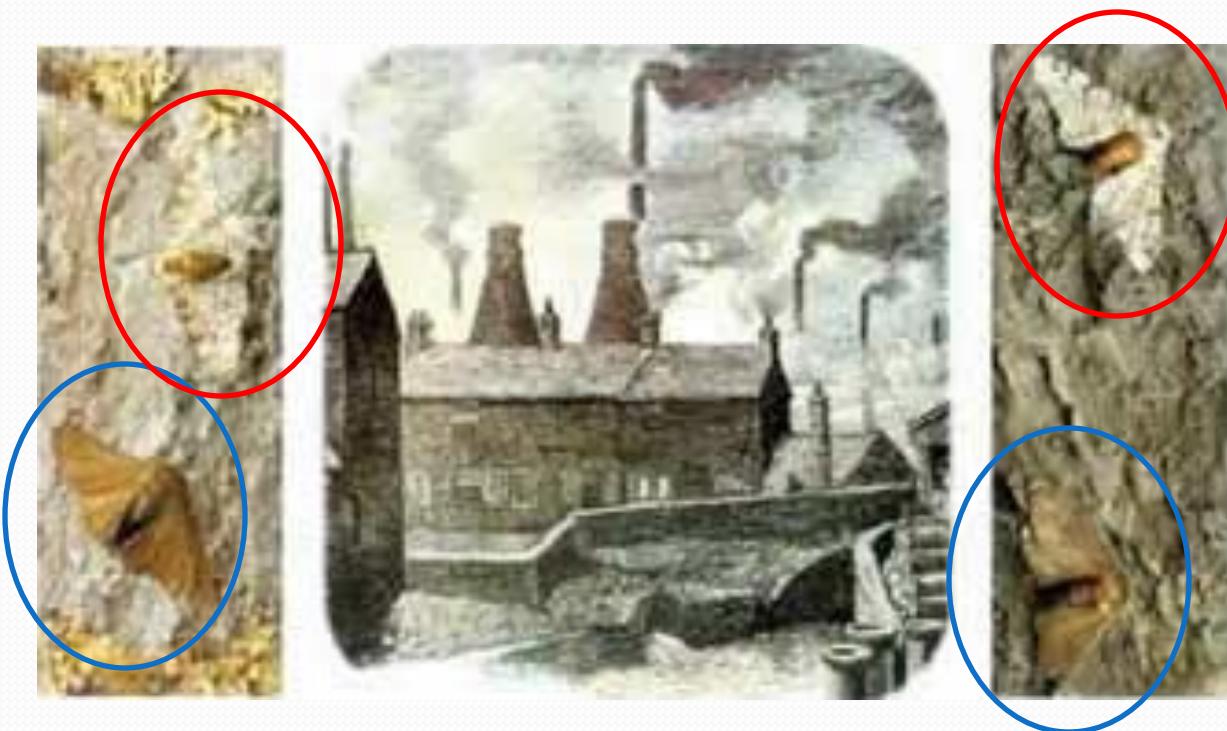
Intelligence Computing Multimedia (ICM)
Informatics faculty – Telkom University

Teori Evolusi

- Ilmuwan Berbeda pendapat
 - Pro: Alam tercipta secara acak
 - Kontra: Alam diciptakan oleh *intelligent designer* (Tuhan)
- Spesies ber-evolusi menjadi **spesies lain** yang **lebih baik???**



Ngengat cerah → gelap? Ilusi



Ref: [ADNo7]

Rusa ber-evolusi menjadi ...?

Rusa ber-evolusi menjadi spesies lain?

A photograph of a cheetah in a savanna setting, captured in mid-pounce as it chases a gazelle. The cheetah's distinctive yellow and black spots are clearly visible. The background shows other animals and green grass.

Evolusi

Seleksi Alam

Mutasi

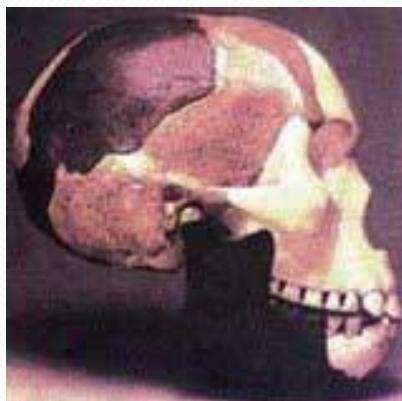
No doubt

?

Ref: [ADNo7]

Monyet → Manusia?

- Banyak ditemukan fosil palsu
- Jika benar, mengapa monyet masih ada hingga hari ini?

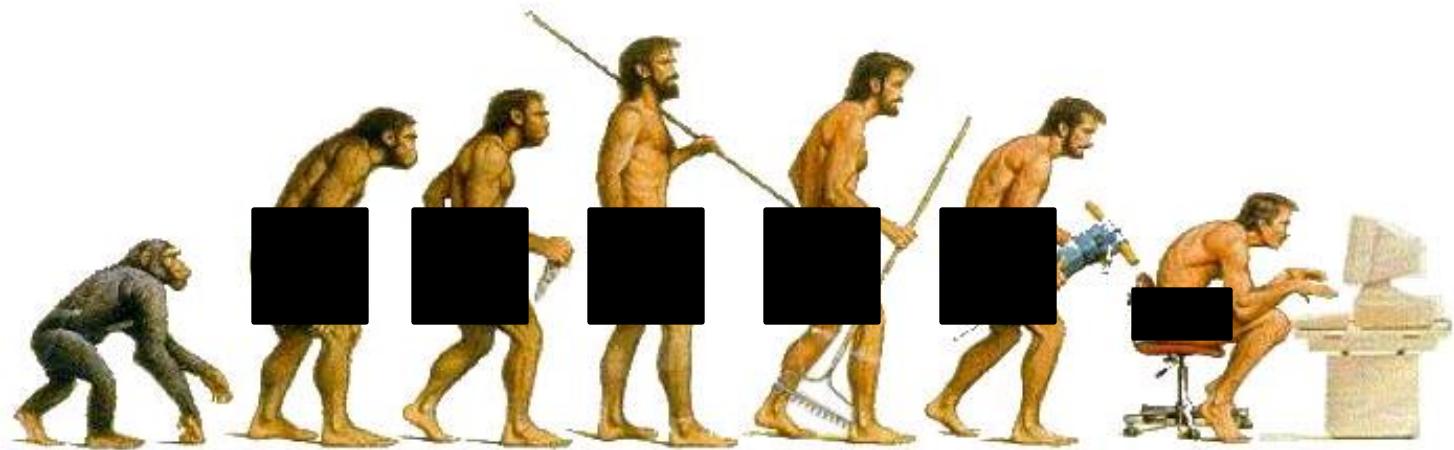


1912: Manusia Piltdown (Inggris) [Charles Dawson]

1953: Manusia Piltdown ternyata palsu [Oakley dkk]

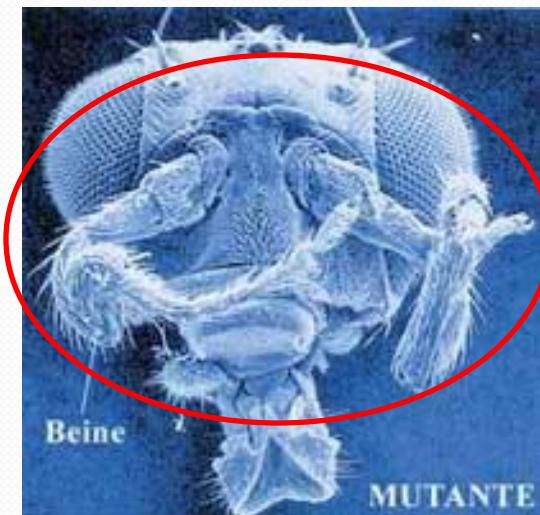
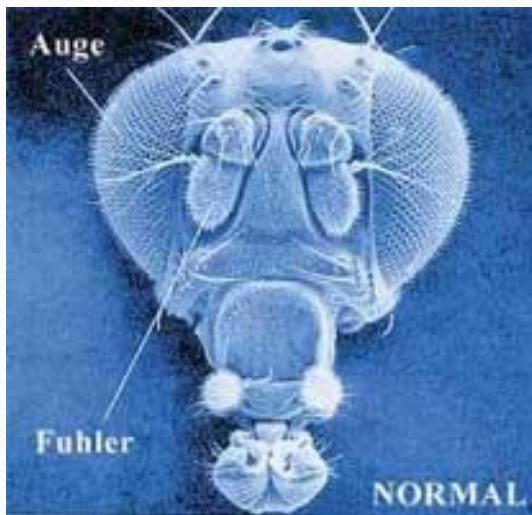
- Tengkorak Manusia + Rahang Orangutan
- Diberi **Dikhromat Potassium** → tampak kuno

Evolution



?????????

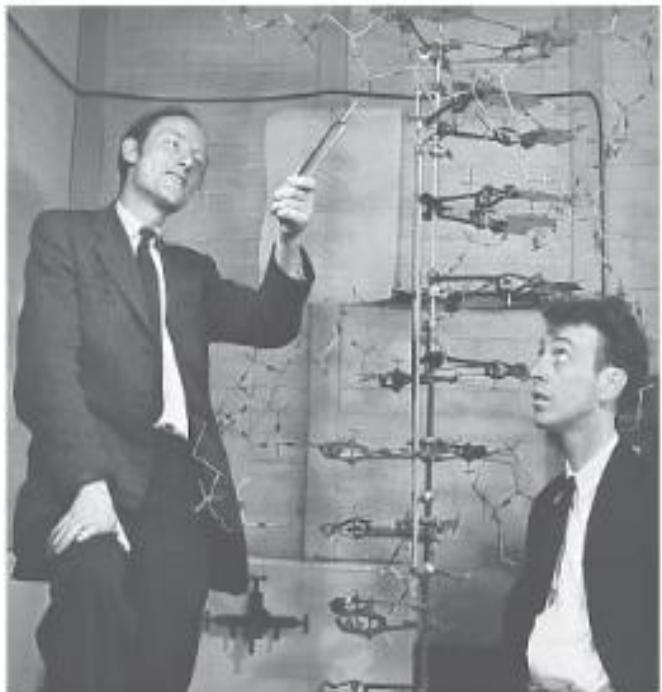
Mutasi: bisa lebih baik?



- Struktur DNA **amat sangat rumit !**
- Perubahan acak (mutasi) **selalu buruk !!**

Ref: [ADNo7]

(A)



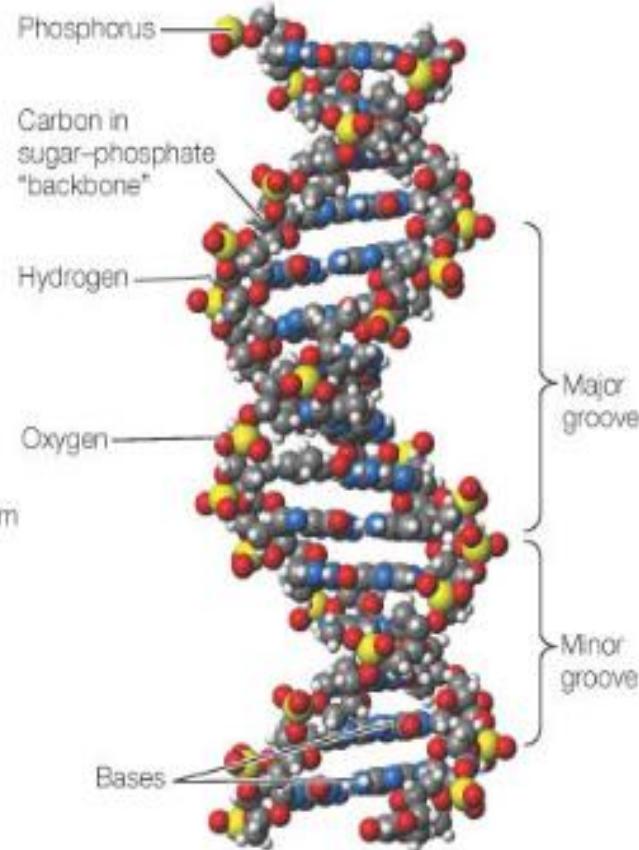
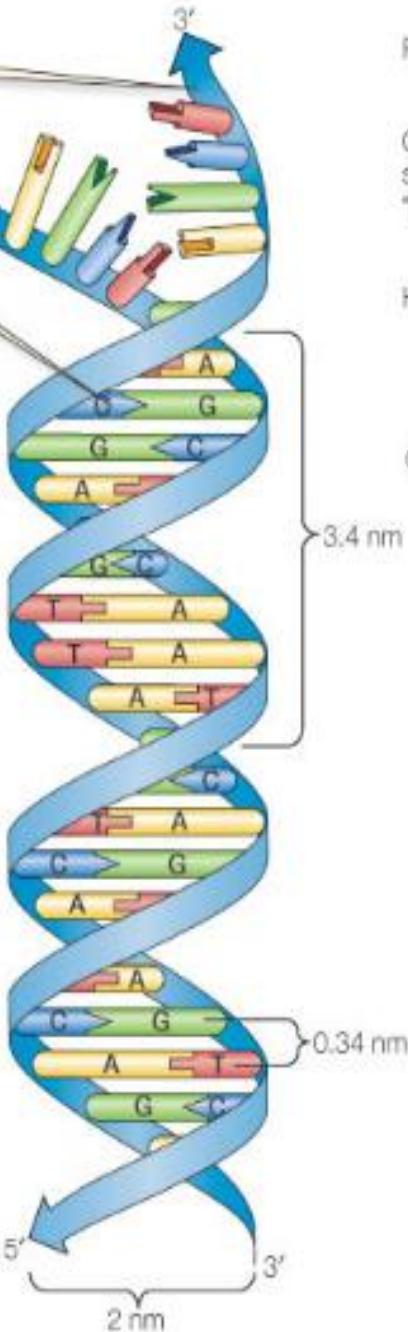
(B)

The blue bands represent the two sugar-phosphate chains.

Pairs of bases form horizontal connections between the chains.

The two chains run in opposite directions:

5' ↓ 3'
3' ↑ 5'



Mutasi: selalu lebih buruk!

- **Manusia Dulu** (jaman nabi Adam hingga nabi Nuh)
 - Usia: 950 tahun
 - Tinggi badan: 15-30 meter (**isu**)
 - Makhluk hidup hanya sedikit
- **Manusia sekarang**
 - Usia lebih pendek
 - Tinggi badan: 2 meter
 - Makhluk hidup semakin banyak
 - Bumi terasa semakin sempit

Manusia sekarang lebih pintar?

- Membuat pesawat terbang
- Komputer semakin hebat
- *Fourth Generation Network*
- Teknologi otomotif makin maju
- Peralatan elektronik
- Makanan semakin bervariasi

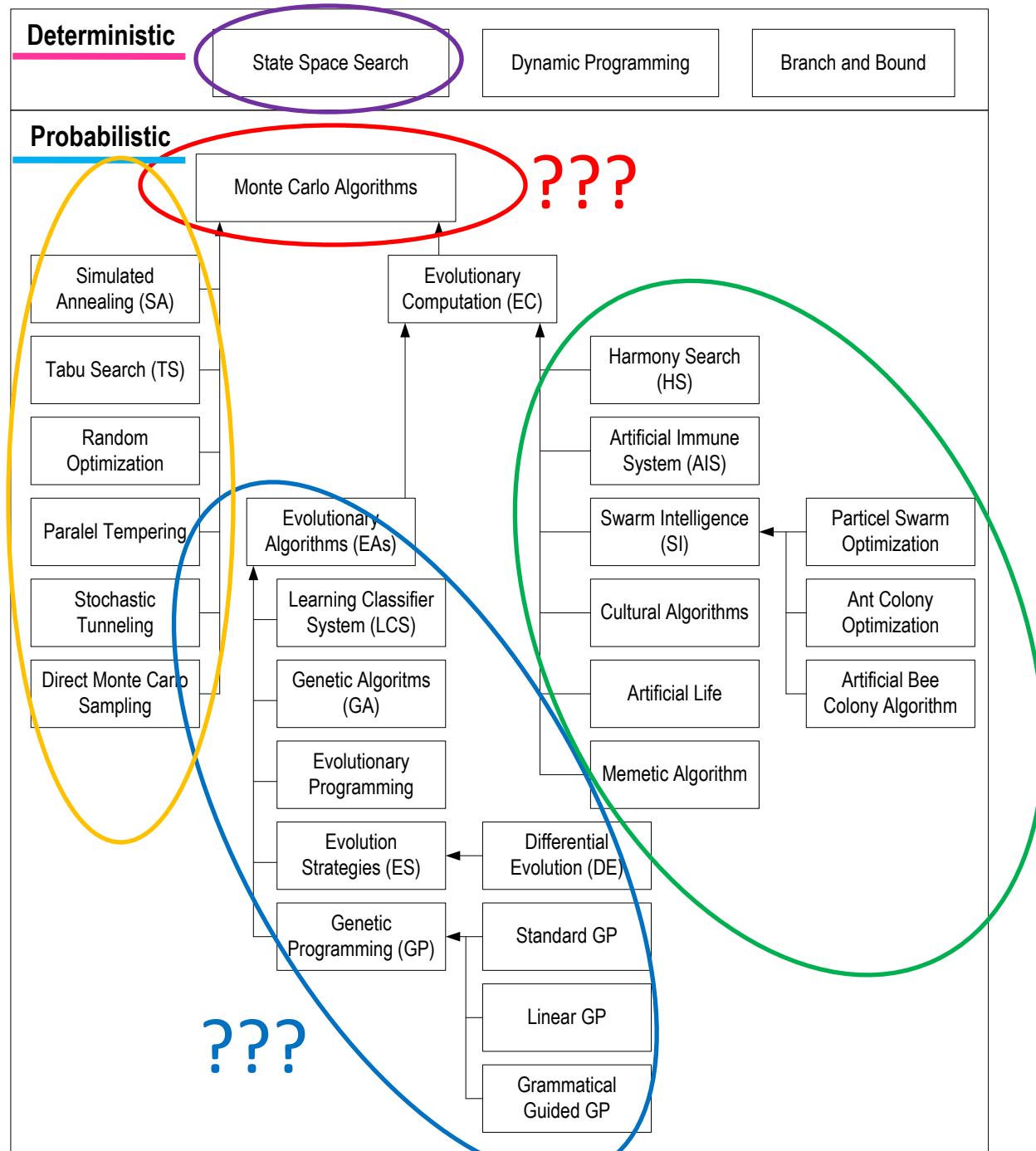
Teknologi manusia sekarang?

- Manusia sekarang mendapatkan teknologi dari manusia jaman dulu
 - Sains: Matematika, Fisika, Kimia
 - Kedokteran
 - Arsitektur: Piramid
- Teknologi: menguntungkan atau merugikan?
- Secara kecerdasan, apakah manusia sekarang lebih pintar dibandingkan manusia jaman dulu?

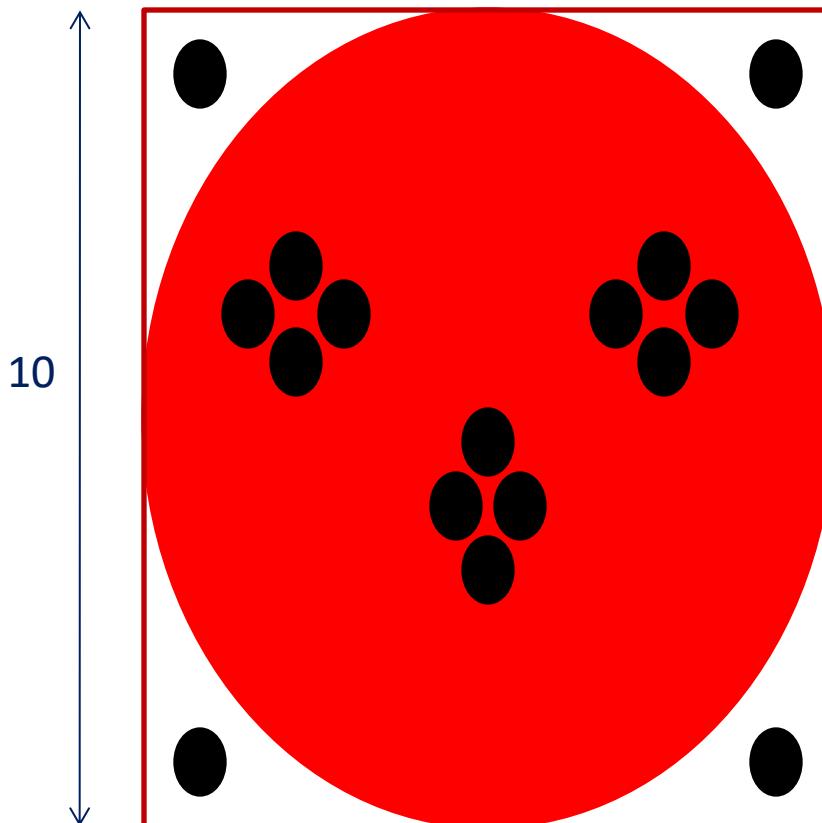
“Evolusi” & “Genetika”

- Dua teori lemah → EC yang powerful?
 - Dunia komputer berbeda dengan dunia nyata.
 - Banyak simplifikasi
- **OPTIMASI**
- **SEARCHING**
- **LEARNING**
- ...

Optimization Algorithms



Jika $d = 10$, berapa luas lingkaran?

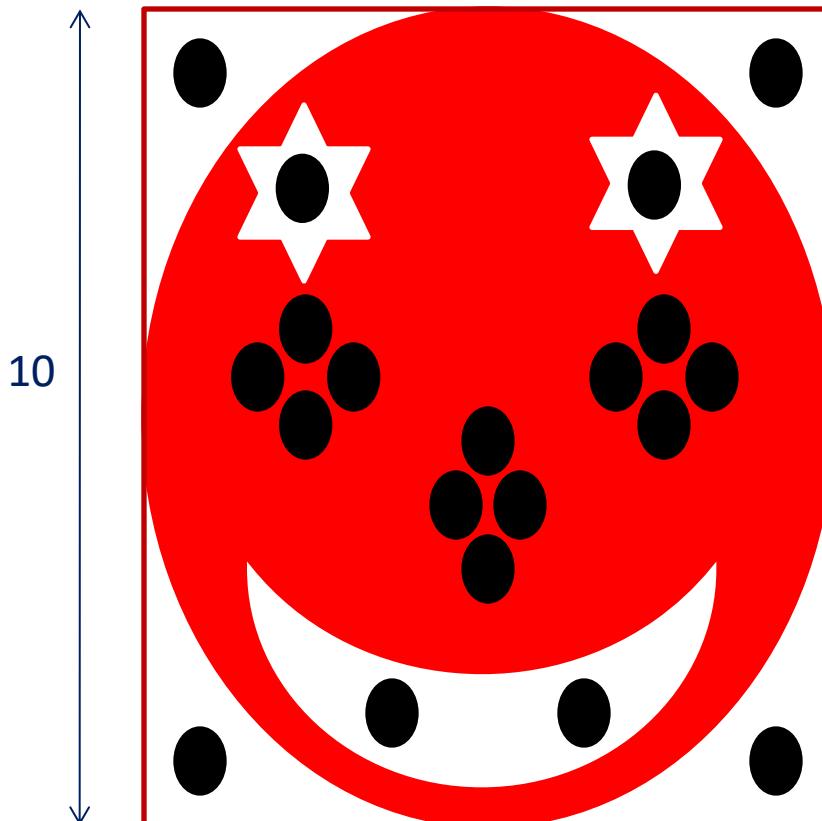


$$\pi r^2 = (22/7)(5^2) \\ = 78,57$$

$$\text{Luas} = (12/16).100 \\ = 75$$

Lebih akurat jika
ribuan titik.

Jika $d = 10$, berapa luas area merah?

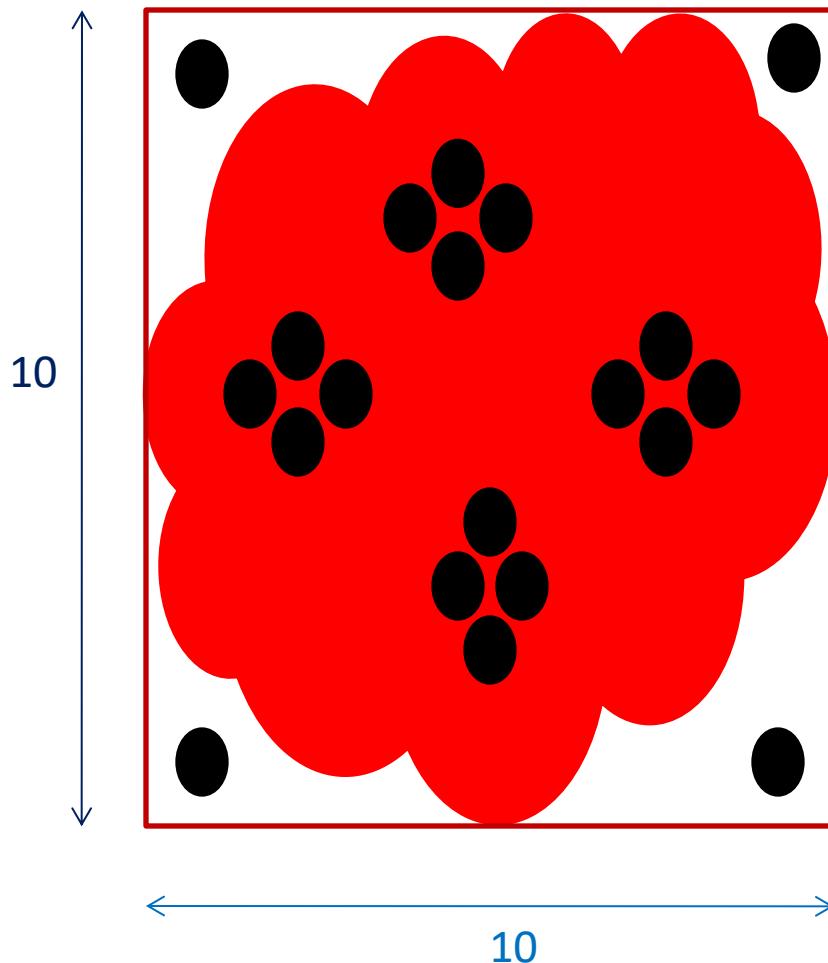


Luas lingkaran – luas
bulan sabit – 2 x luas
bintang = **SULIT ???**

$$\text{Luas} = (12/20).100
= 60$$

Lebih akurat jika
ribuan titik.

Berapa luas area merah?



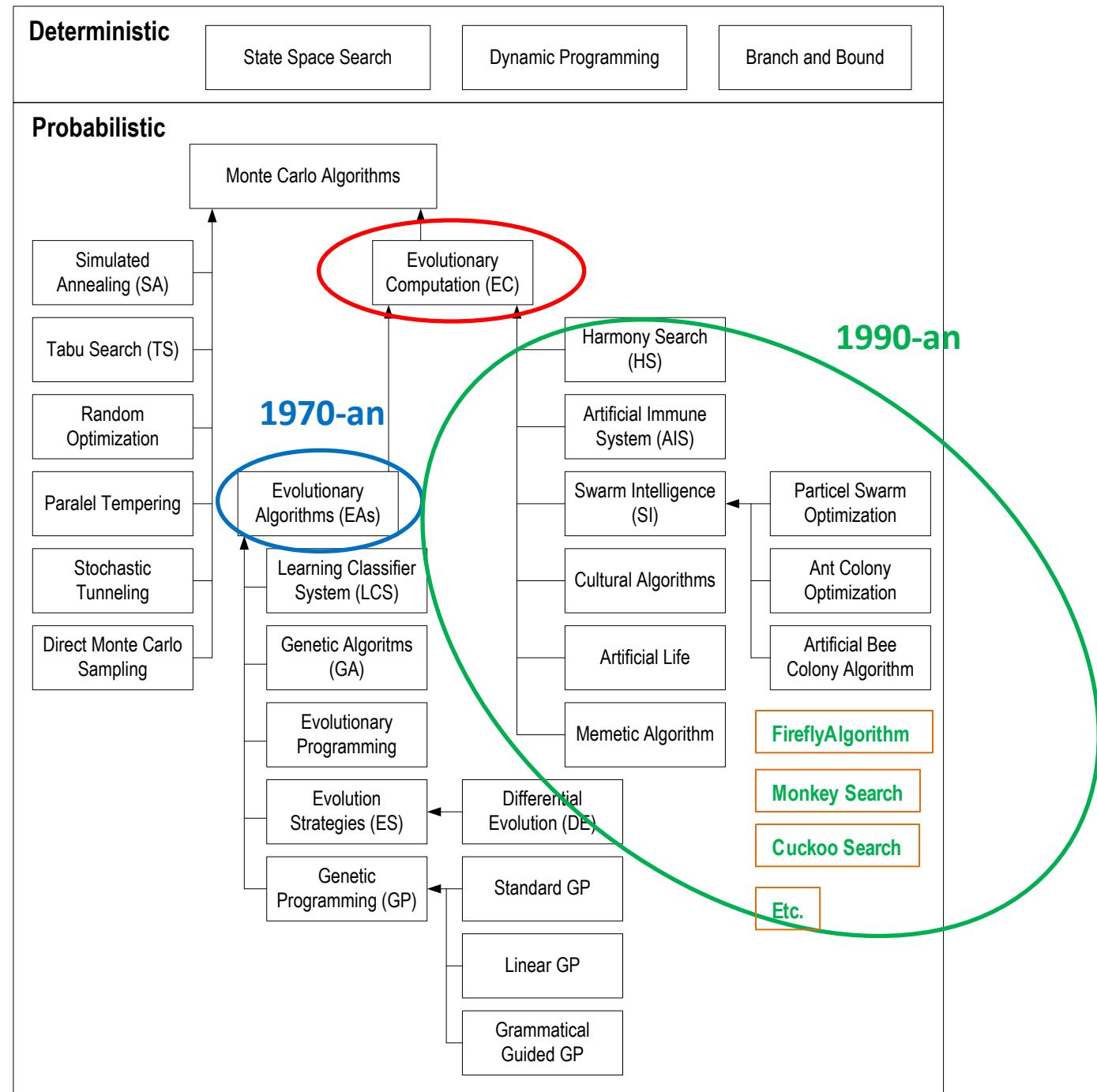
Luas = ???

Model matematis ??

$$\begin{aligned} \text{Luas} &= (16/20) \cdot 100 \\ &= 80 \end{aligned}$$

Lebih akurat jika
ribuan titik.

Optimization Algorithms



Swarm Intelligence: kecerdasan berkelompok

Satu ikan → kurang cerdas

Ribuan ikan → cerdas

Paus mencari makan ???



Fish Schooling: bergerombol



Bird Flocking: Formasi V

Tabrakan antar burung ???



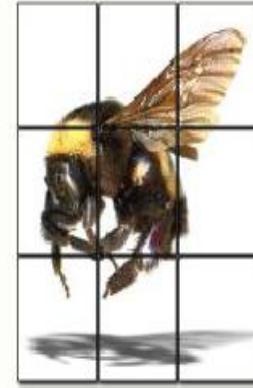
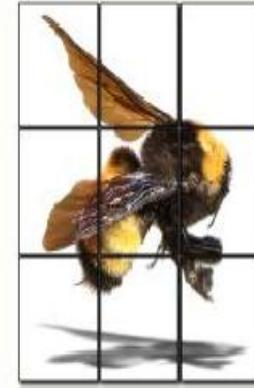
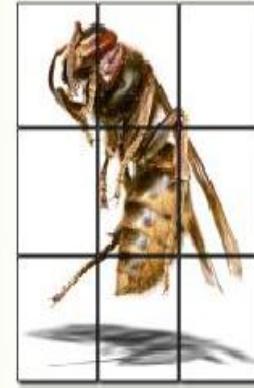
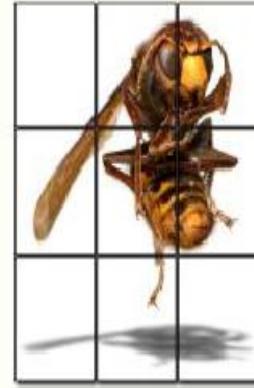
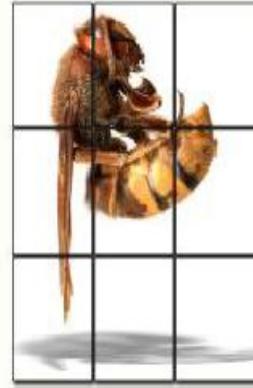
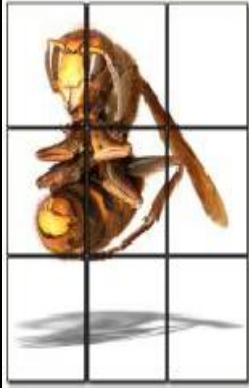


Jembatan Semut

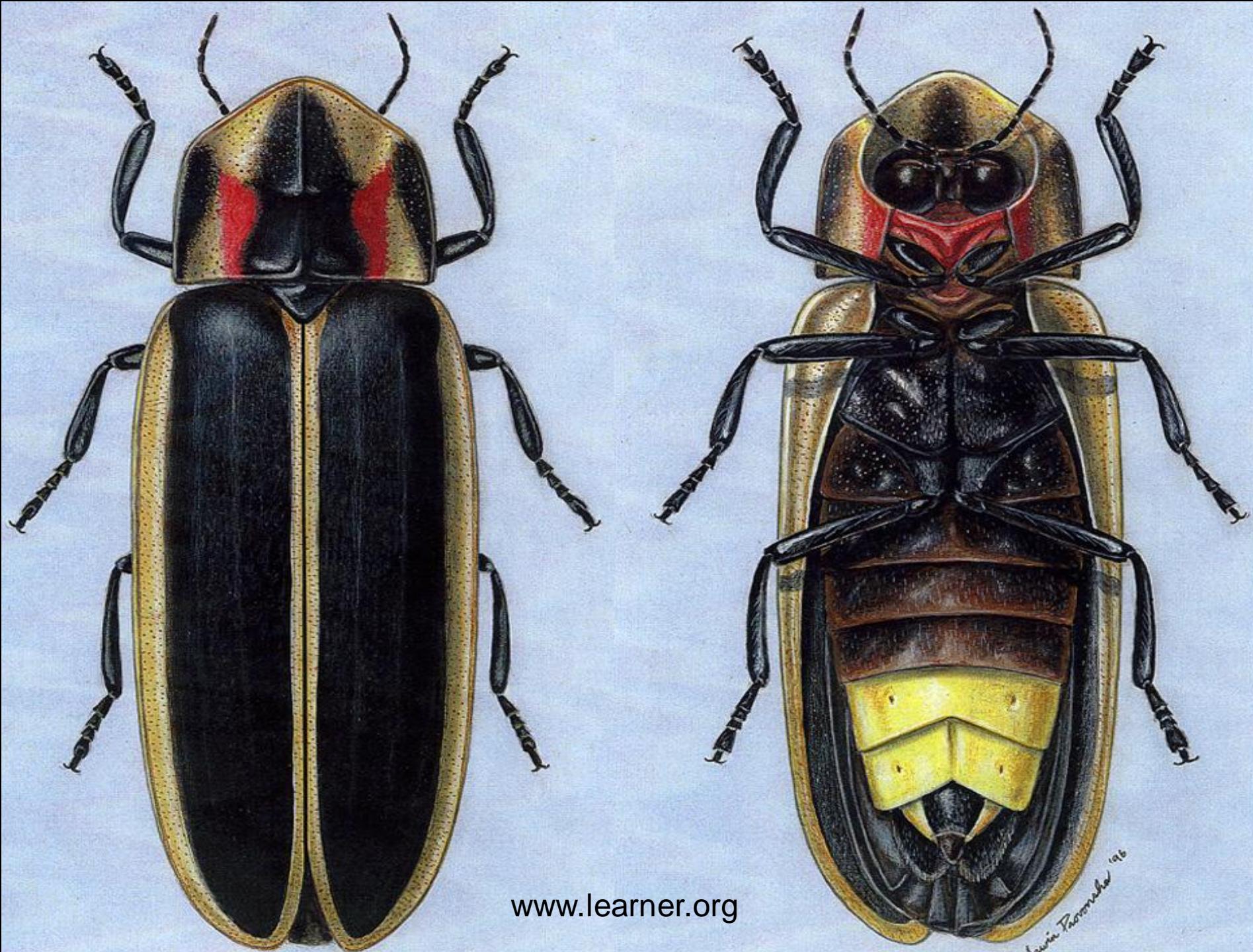


Jembatan Semut





- Pencarian makanan (*Foraging behaviours*)
- Perkawinan (*Marriage behaviours*)
- Konsep ratu lebah (*Queen bee concept*)
 - *Queen-Bee Evolution* [Sung, 2003].
 - *Bee Crossover* untuk memperbaiki GA [Kara, 2004].



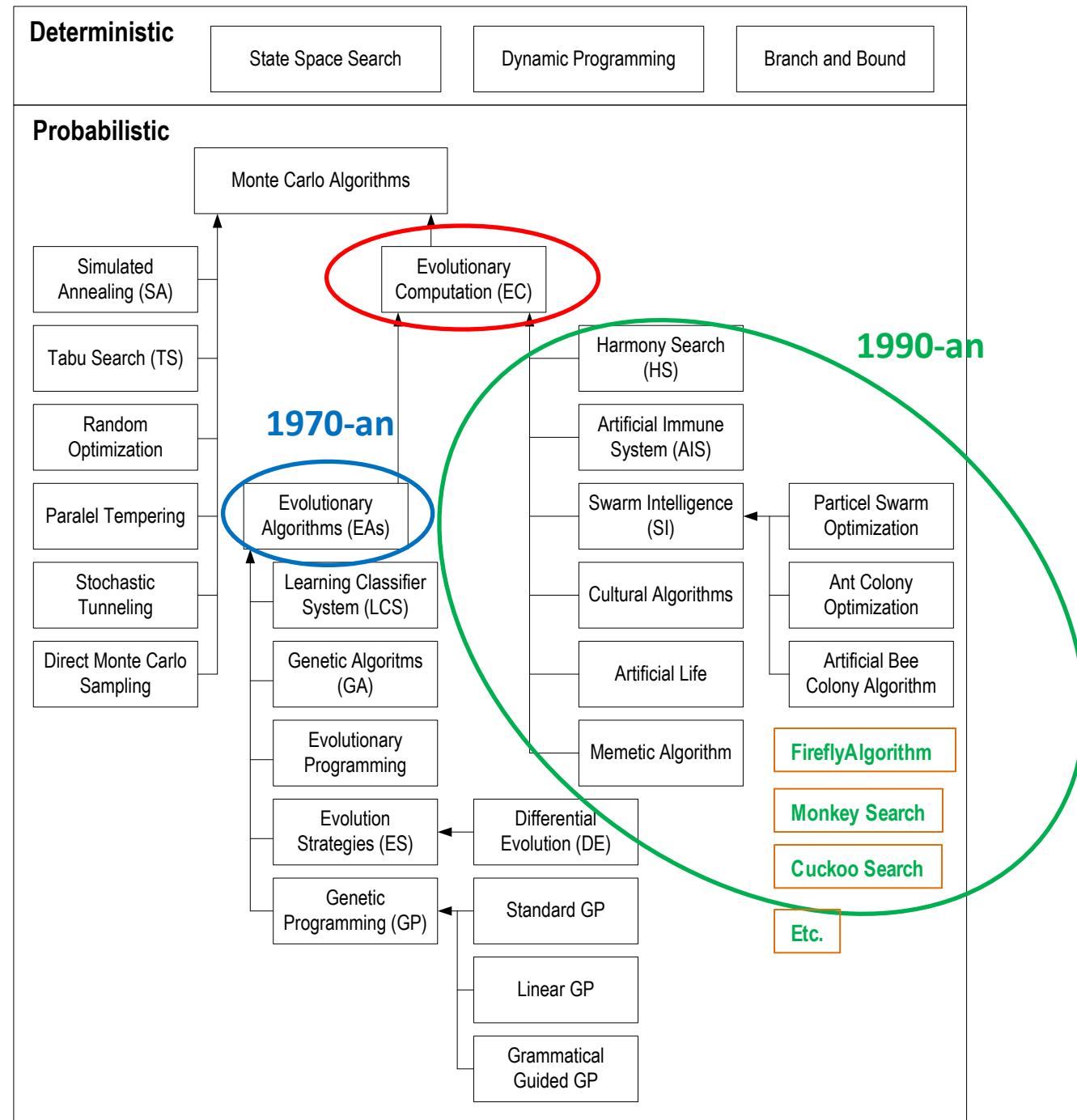






cheats-cuckoo

Optimization Algorithms

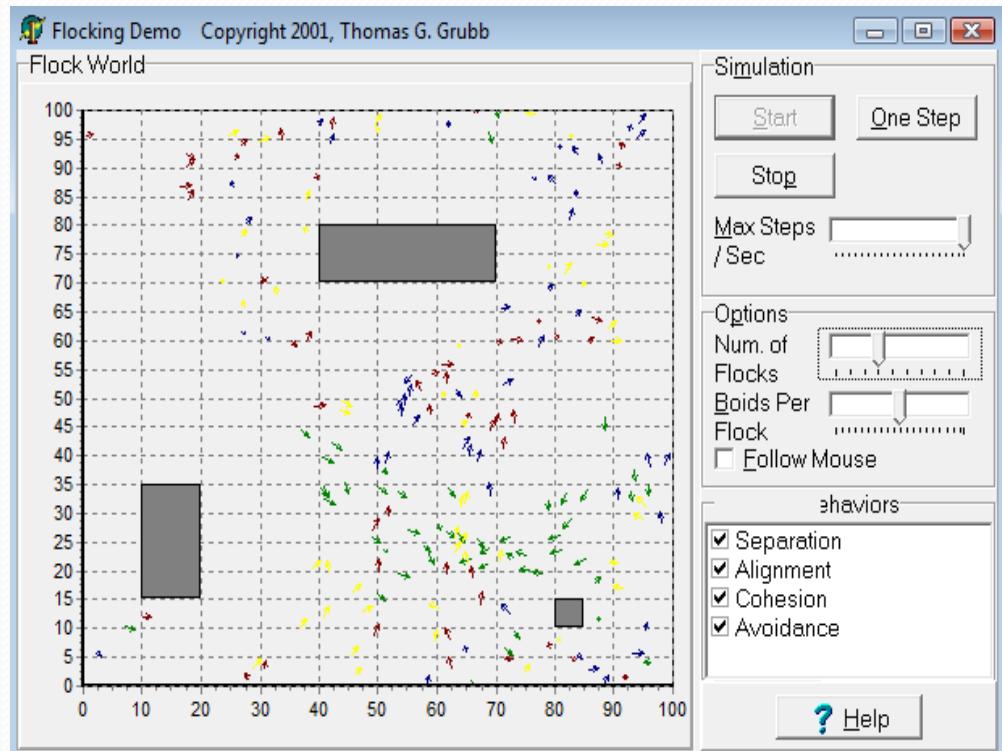


Swarm, Flock, School, Herd

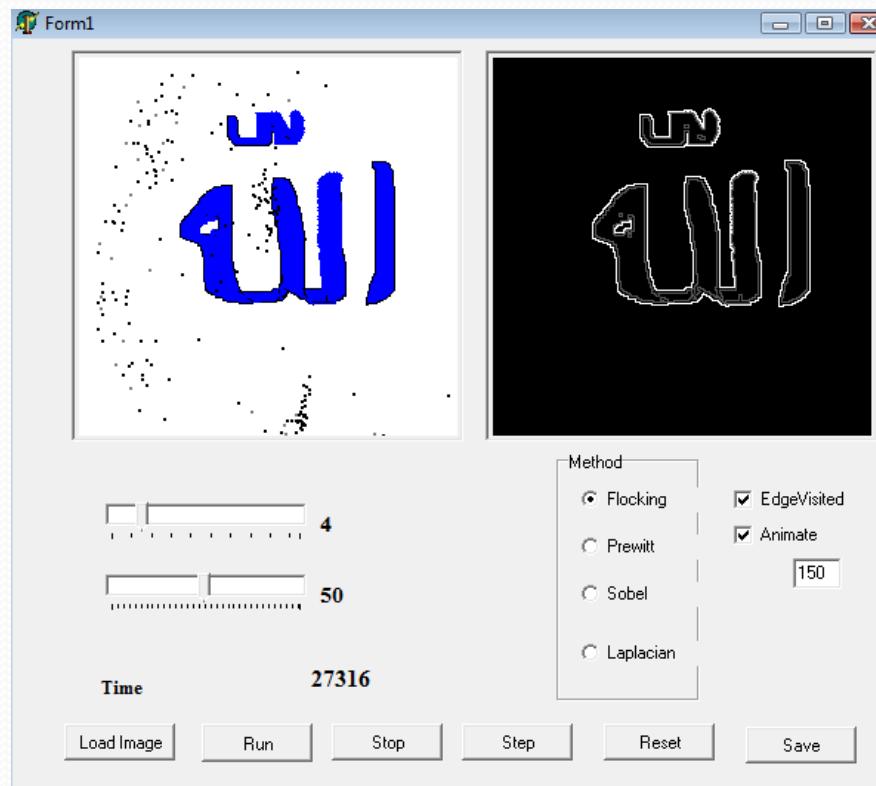


Simulasi & Animasi

- Simulasi pedestrian
 - Kapasitas ruangan
 - Gedung
 - Trotoar
- Animasi
 - Computer-graphic
- Pencarian tepi citra

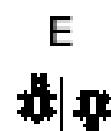


Edge detection

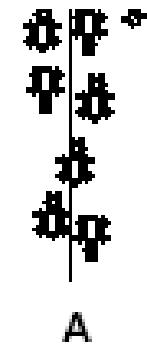
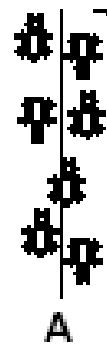
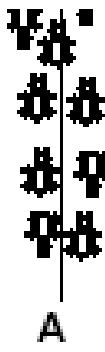


[**Addino Yudi Abdal - 113990156 - IMPLEMENTASI PROSES
PENDETEKSIAN SISI DENGAN TEKNIK FLOCKING]**

Ant Colony Optimization



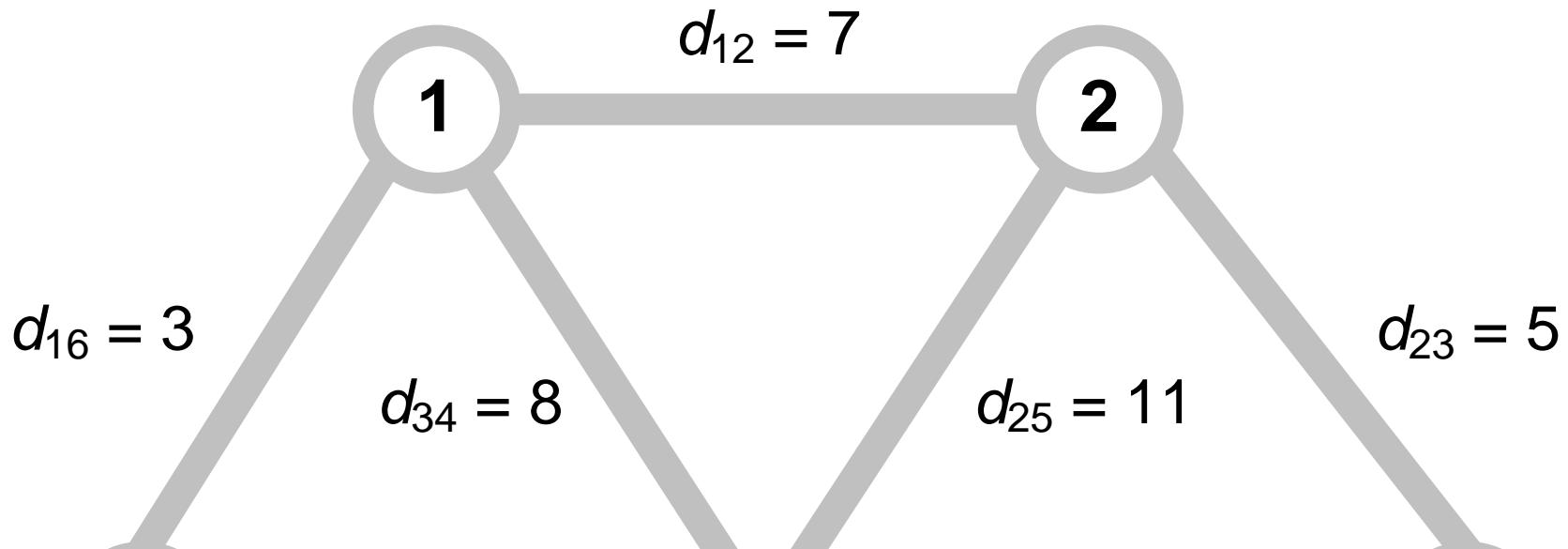
- Traveling Salesman Problem
- Masalah optimasi lainnya.



a)

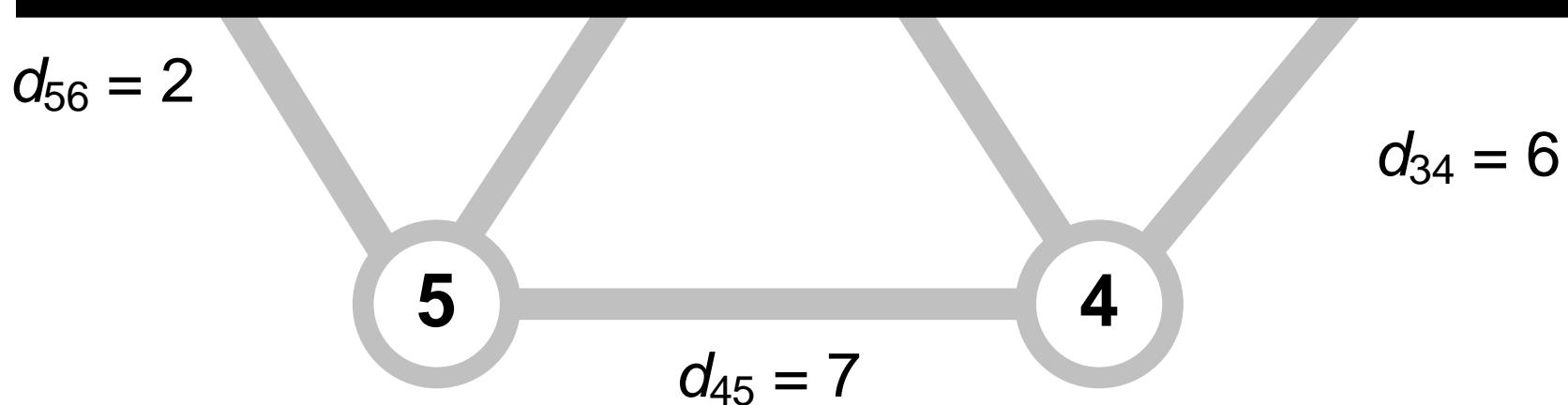
b)

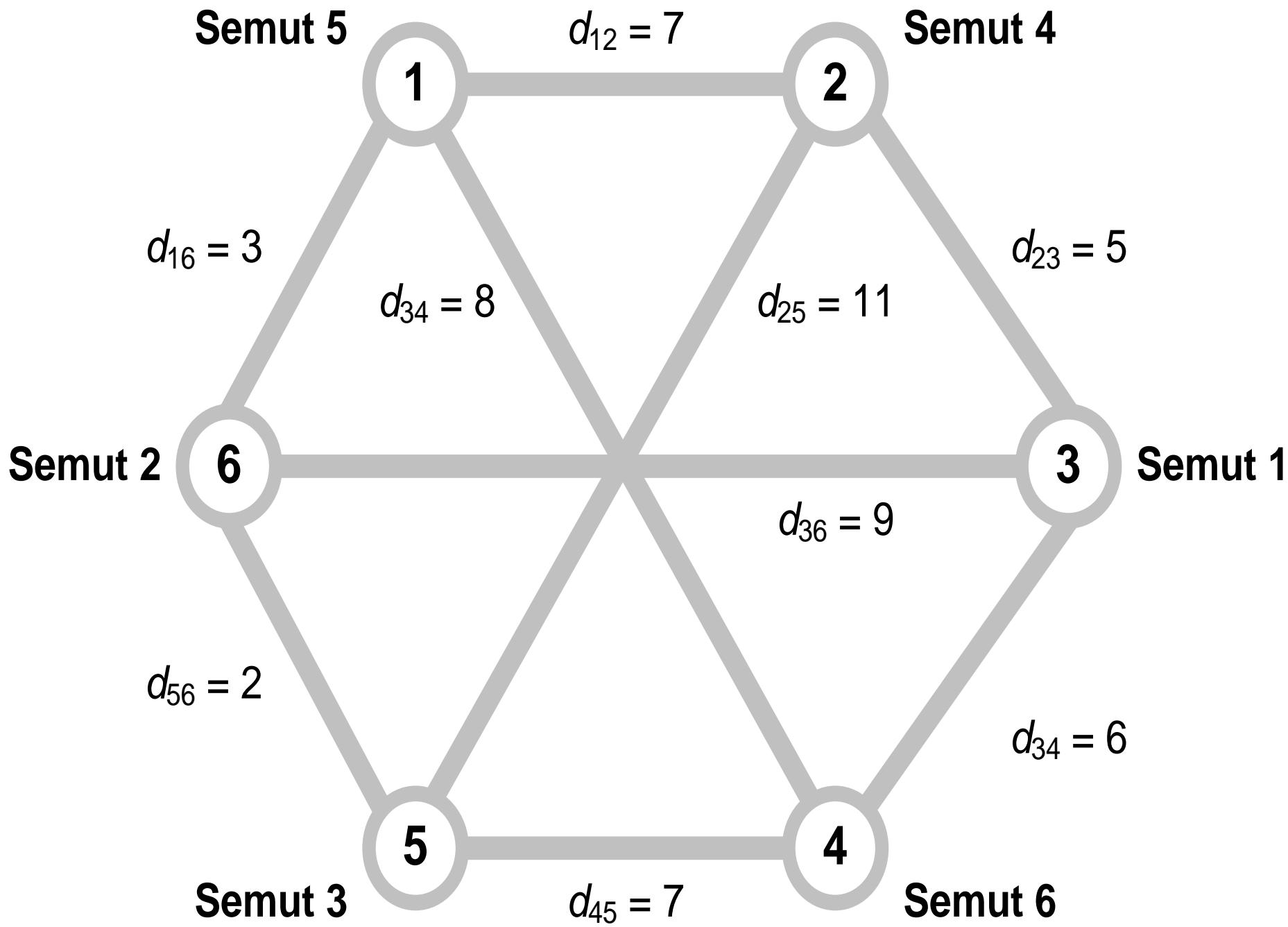
c)

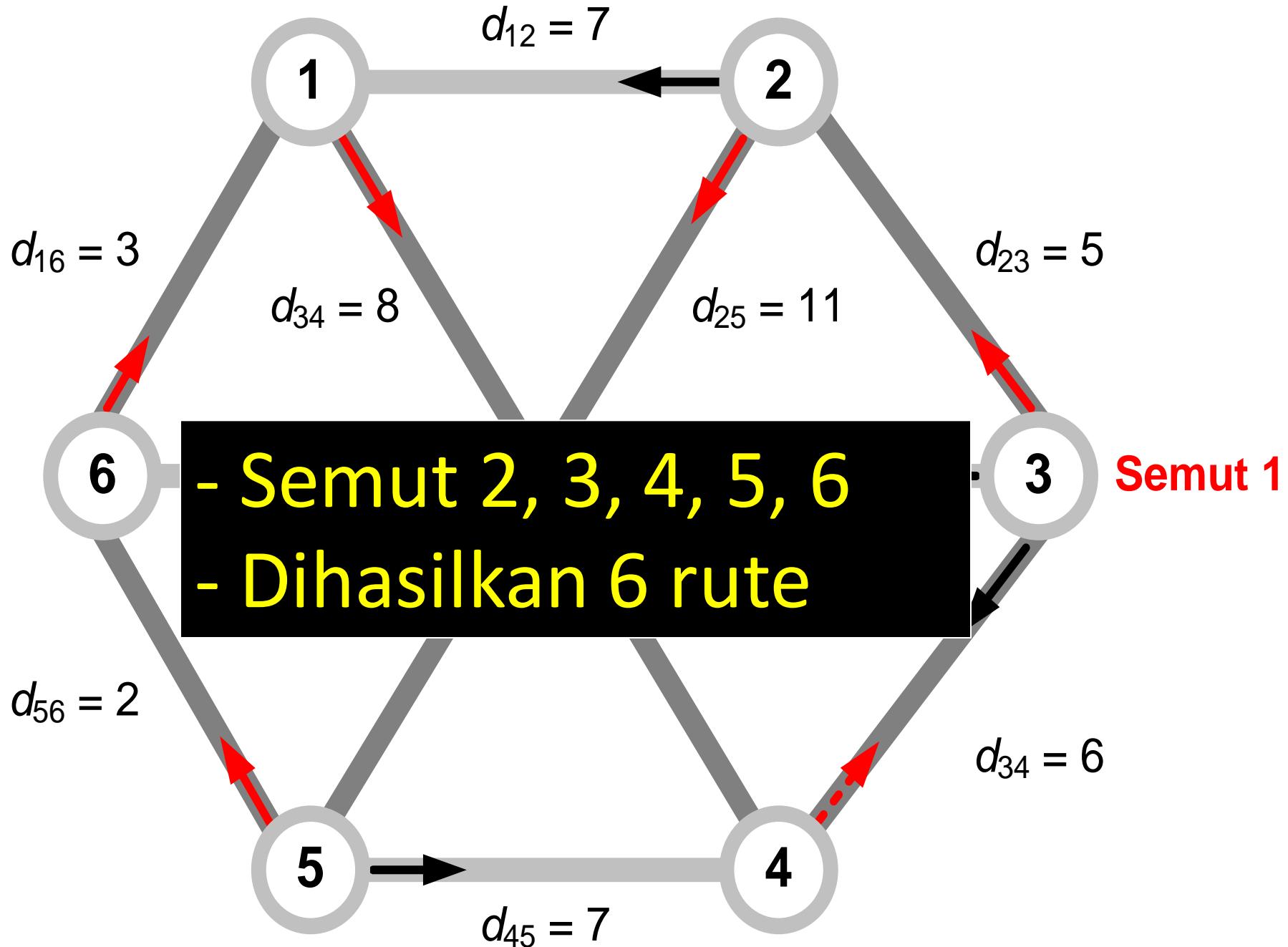


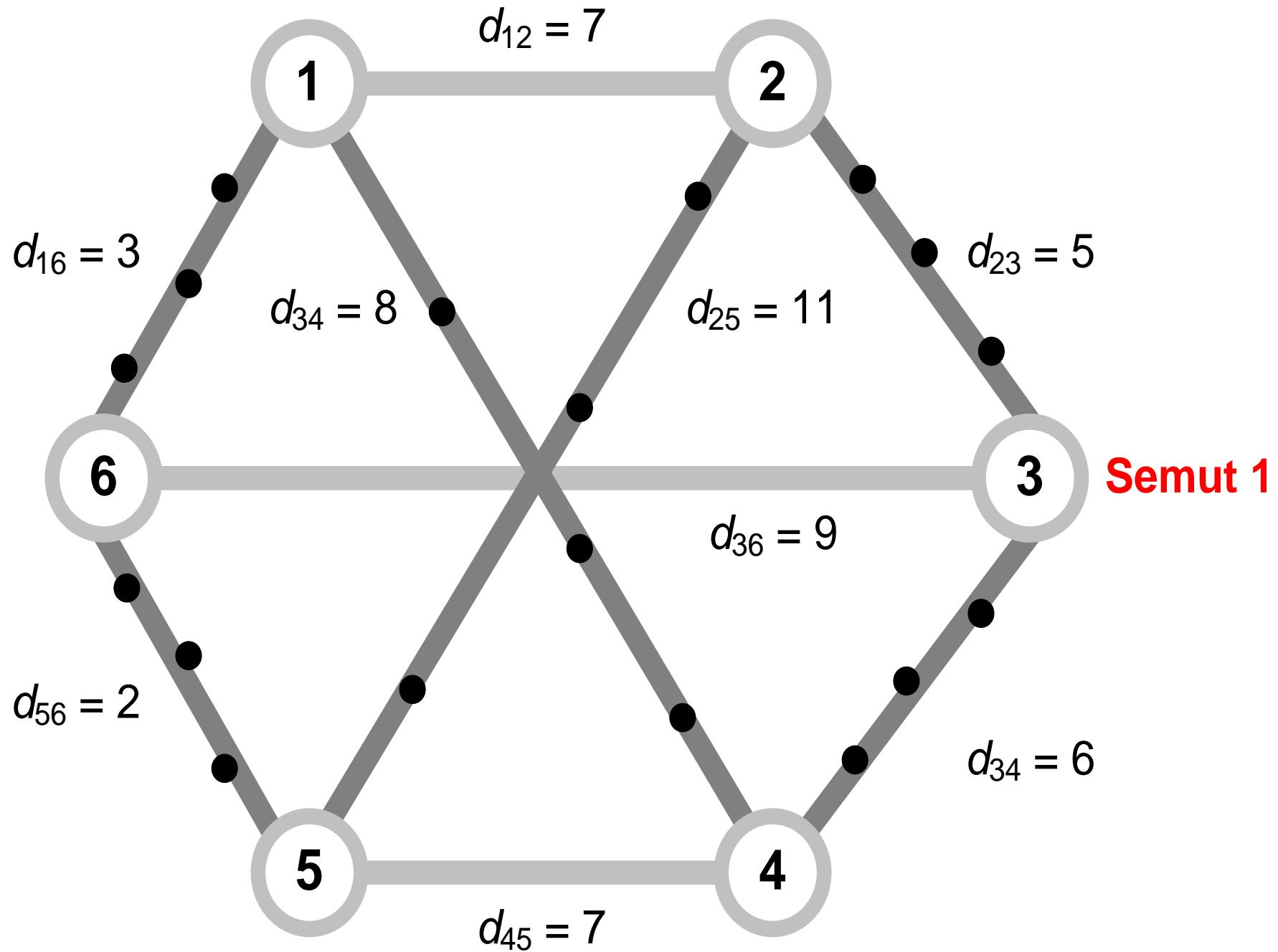
Rute kunjungan terbaik?

Koloni semut???

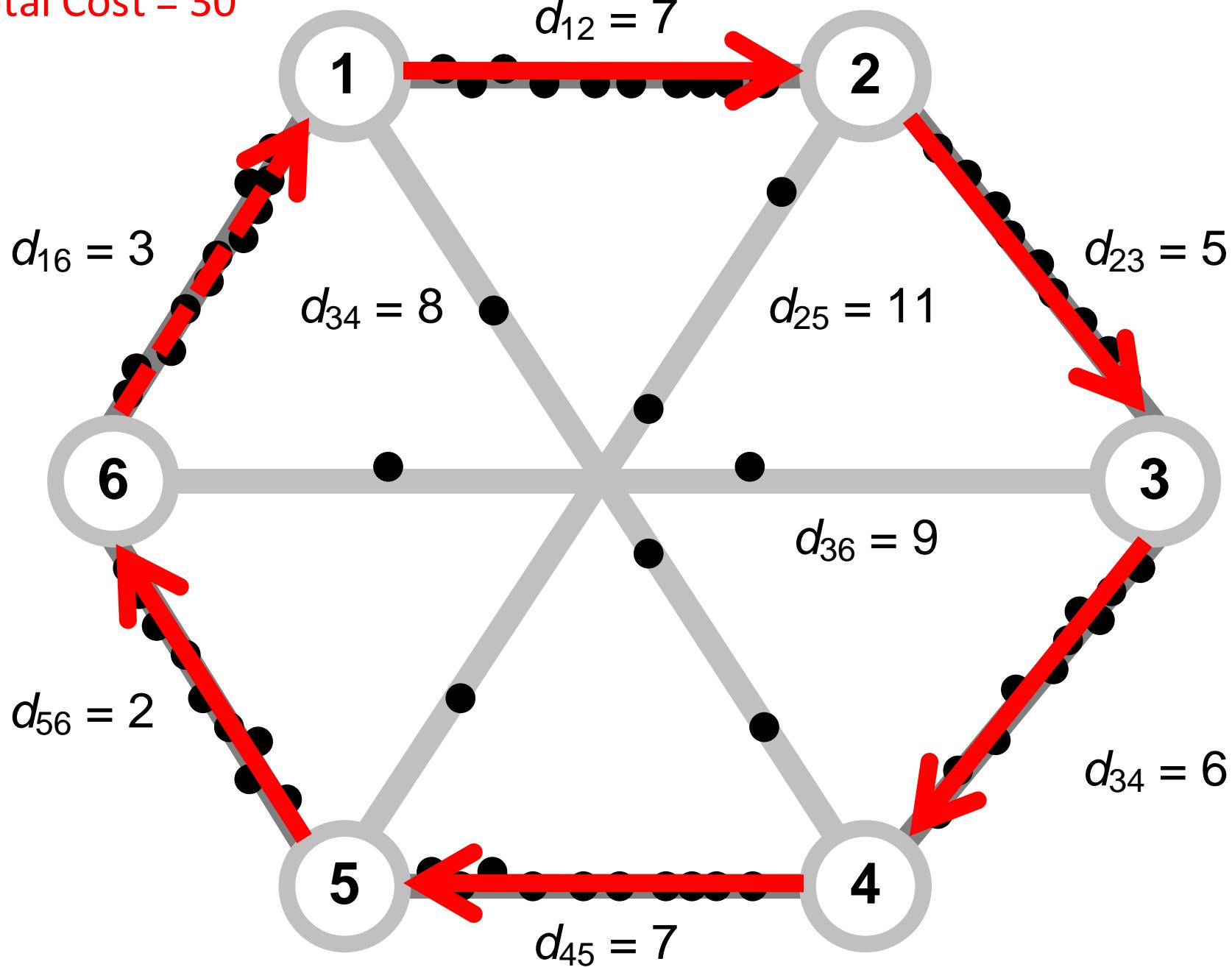






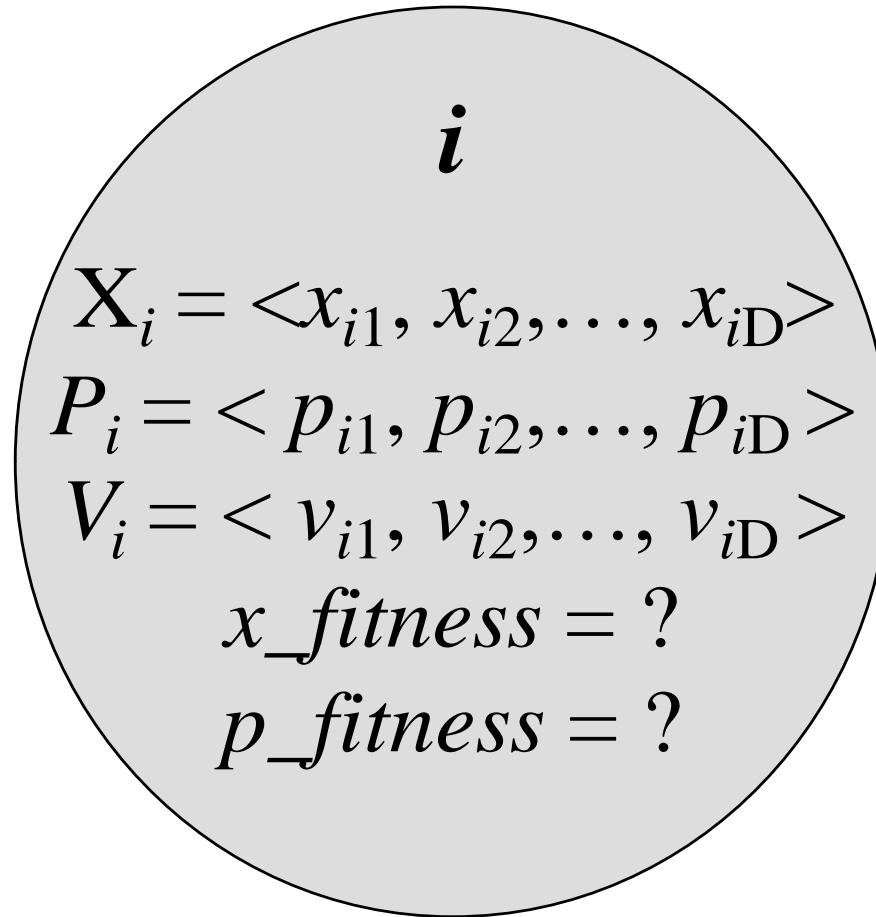


Total Cost = 30



Particle Swarm Optimization

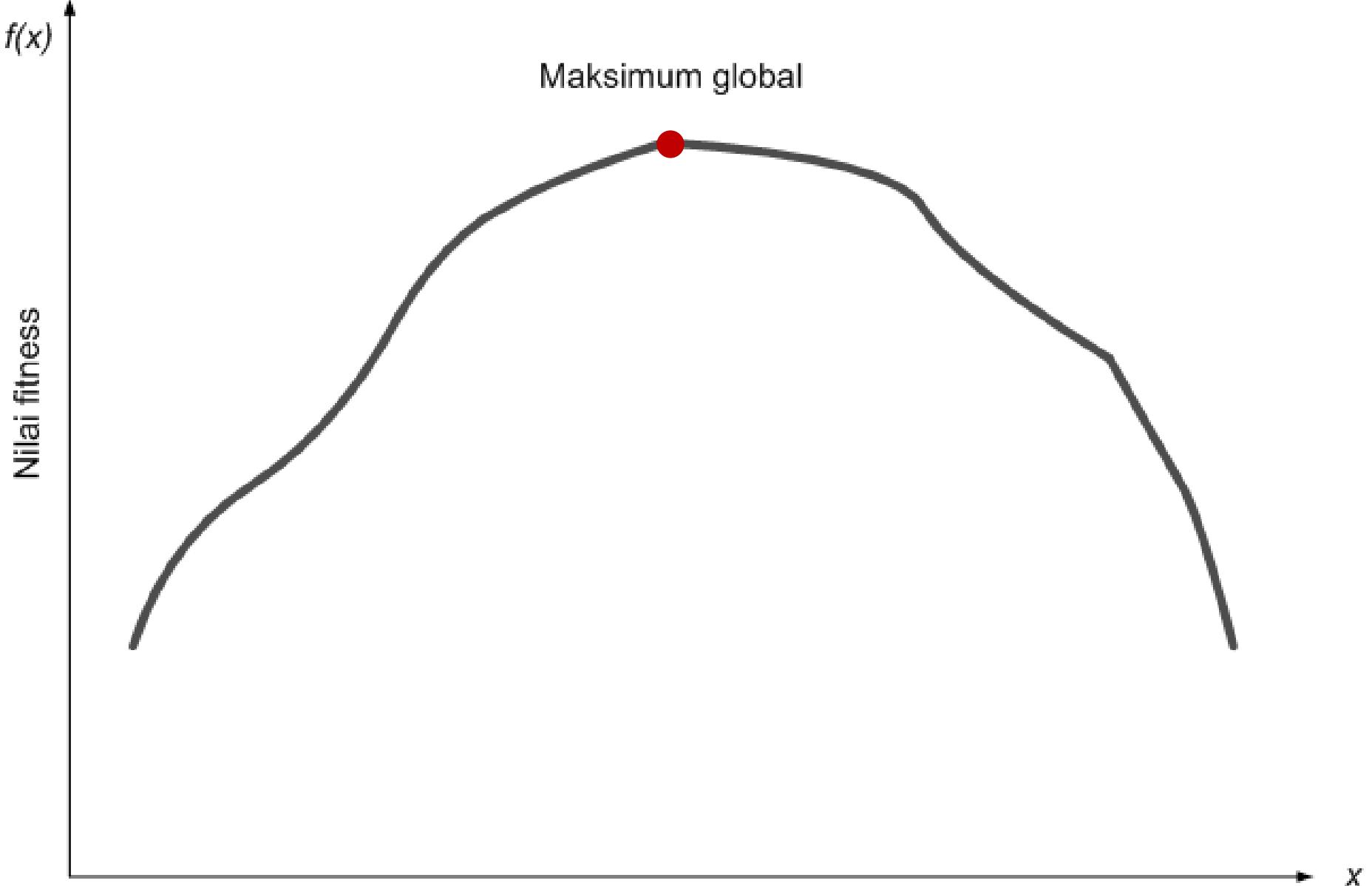
- James Kennedy dan Russ Eberhart (1995)
- *Bird flocking, Fish schooling, etc.*
- PSO dimulai dengan suatu populasi yang terdiri dari sejumlah individu (yang menyatakan solusi) yang dibangkitkan secara acak.
- Selanjutnya melakukan pencarian solusi optimum melalui perbaikan individu untuk sejumlah generasi tertentu.
- PSO tidak menggunakan operator-operator rekombinasi & mutasi.

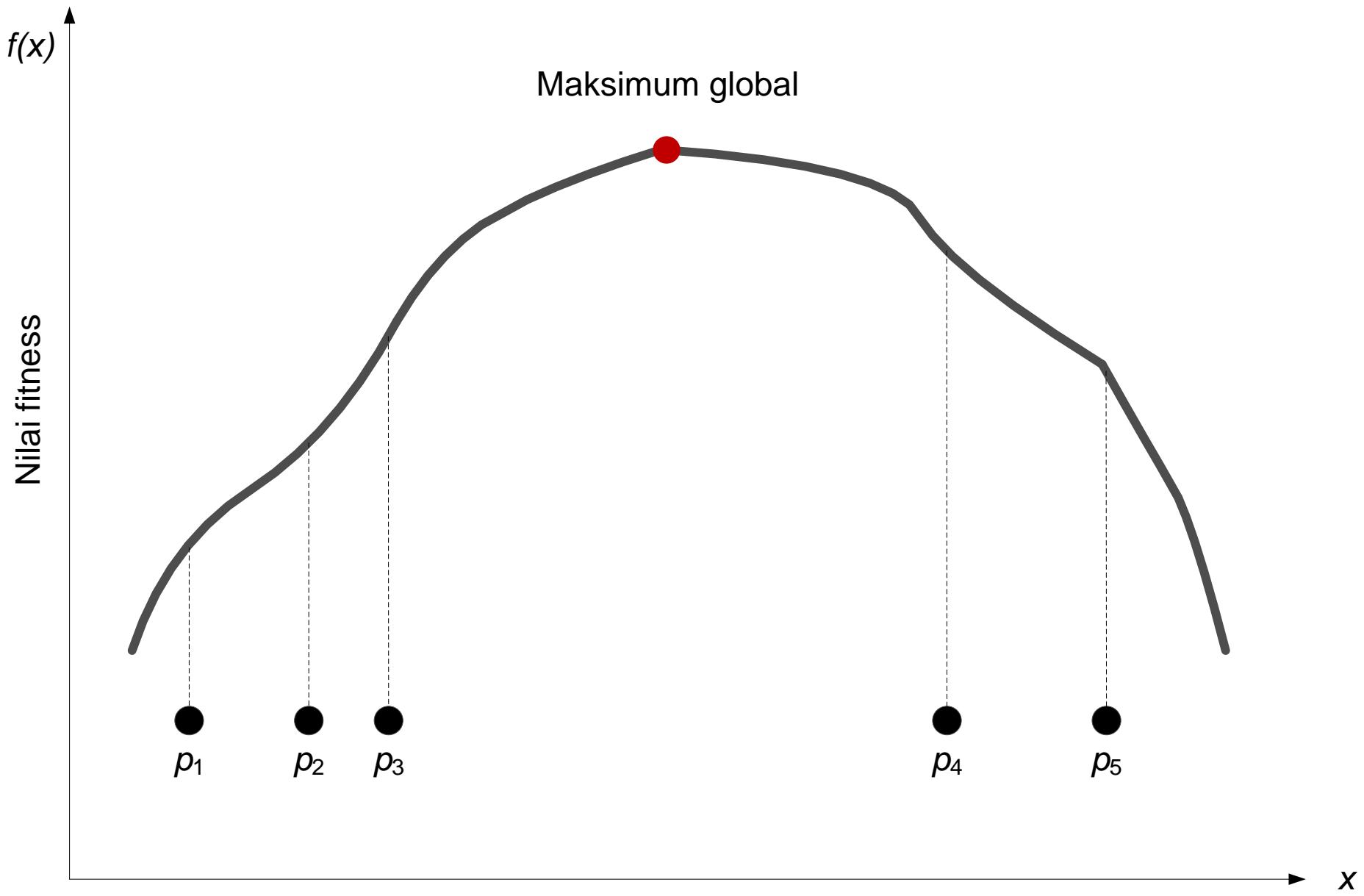


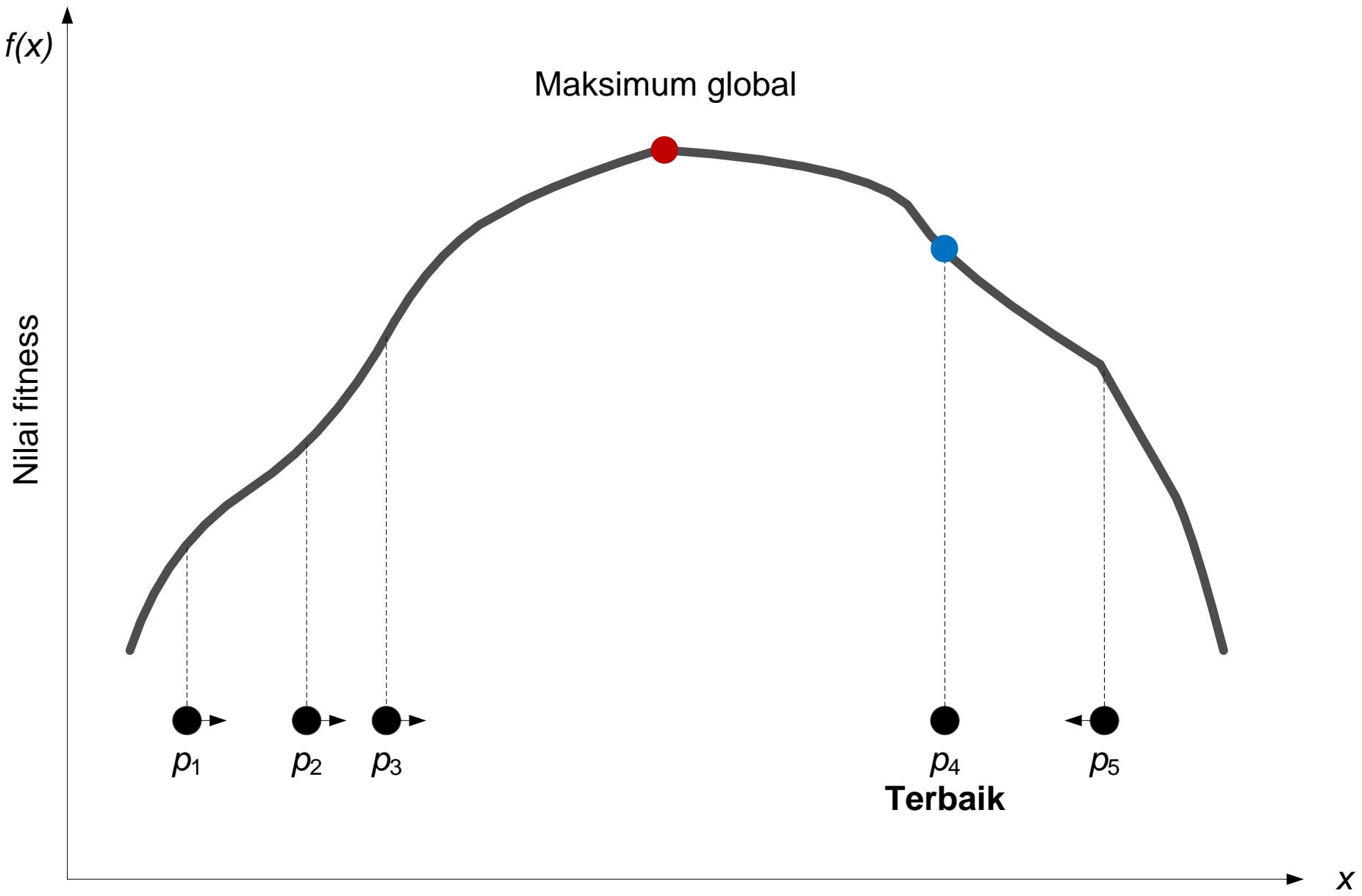
X : posisi partikel saat ini di dalam ruang pencarian

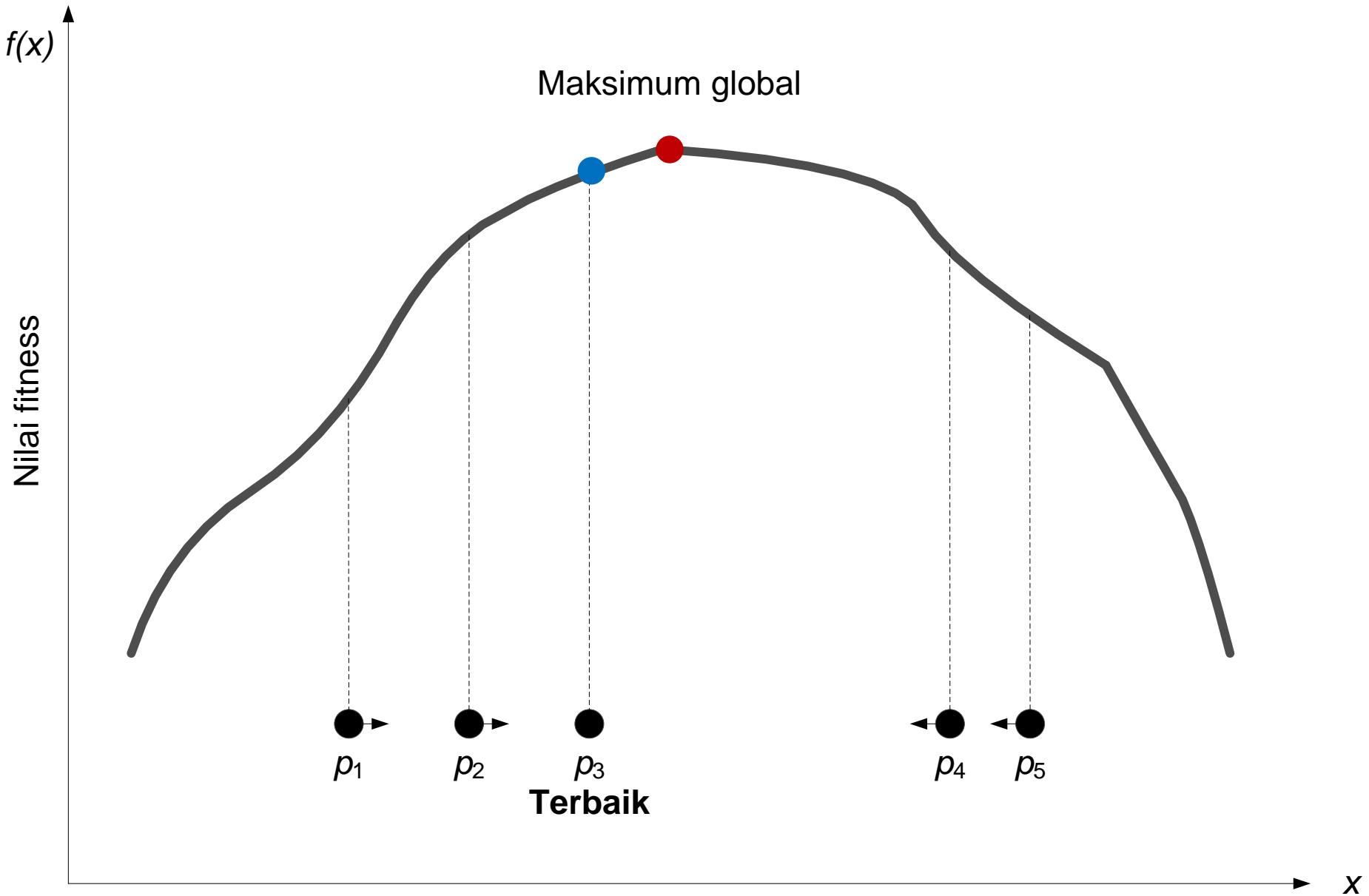
P : posisi solusi *best-so-far* dari partikel tersebut;

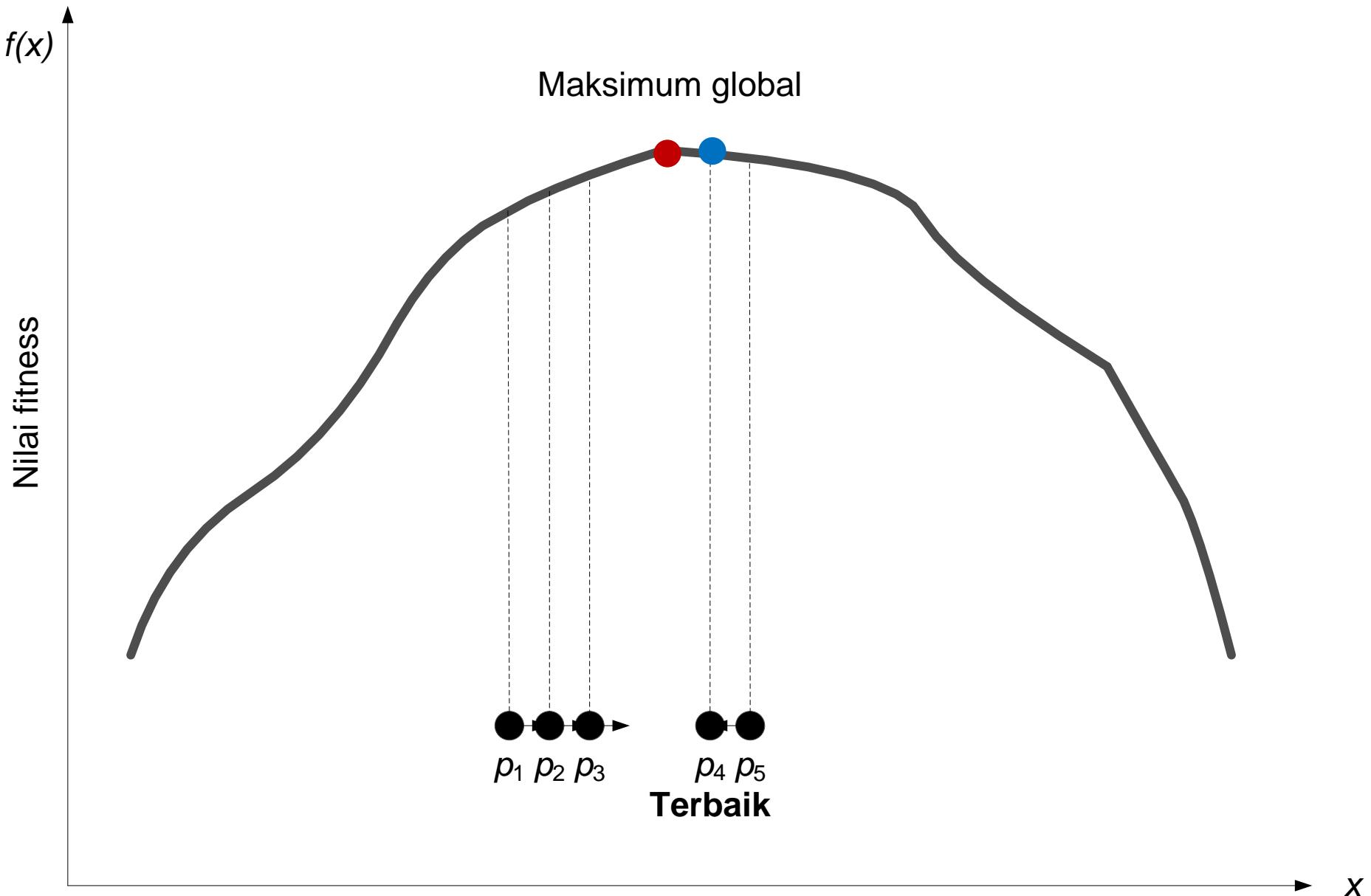
V : *gradien* (arah) kemana partikel akan “terbang”

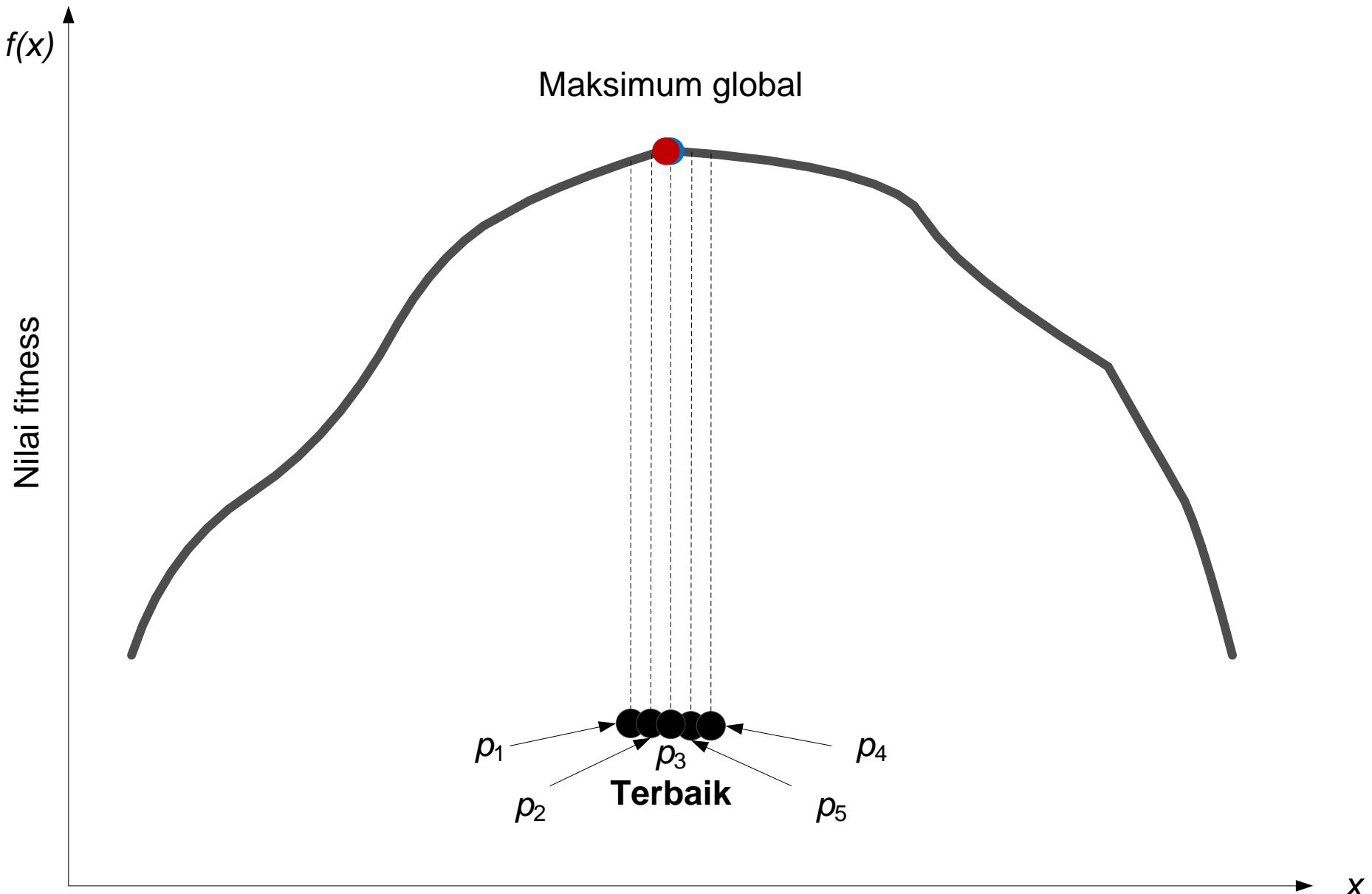


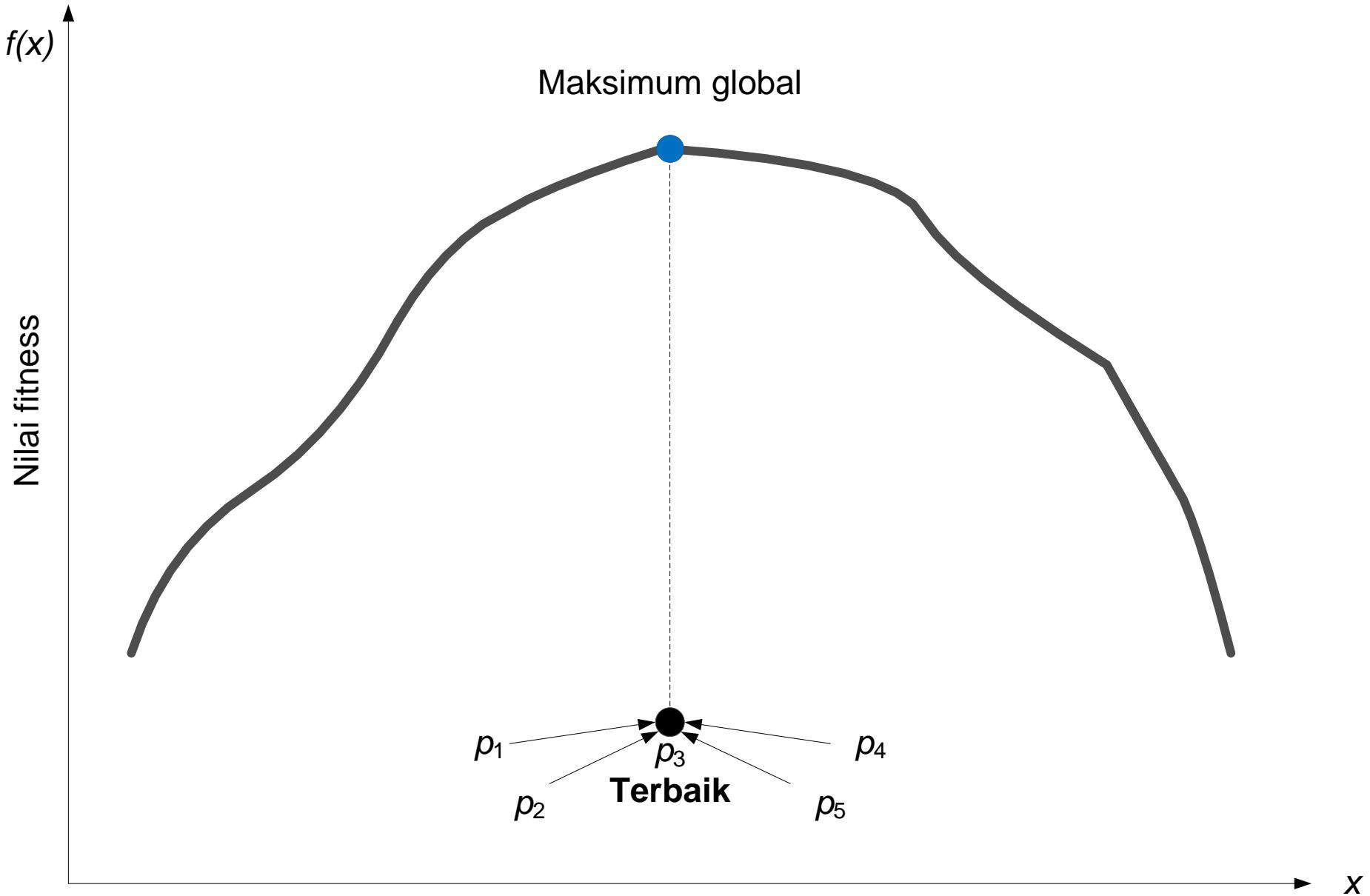


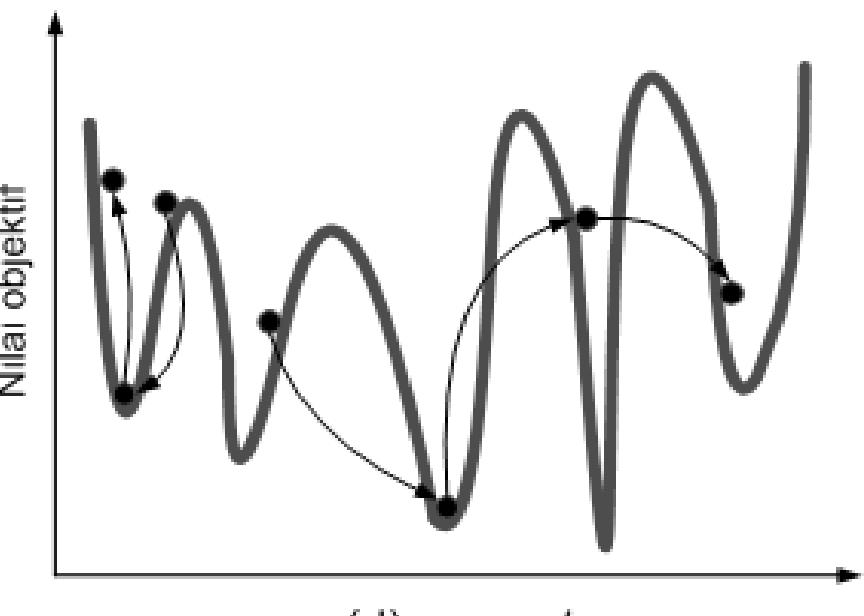
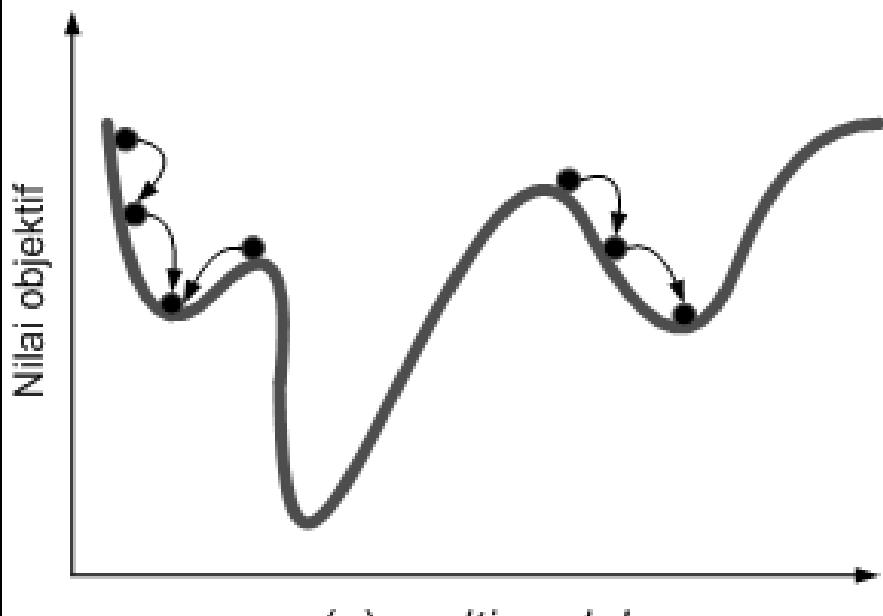
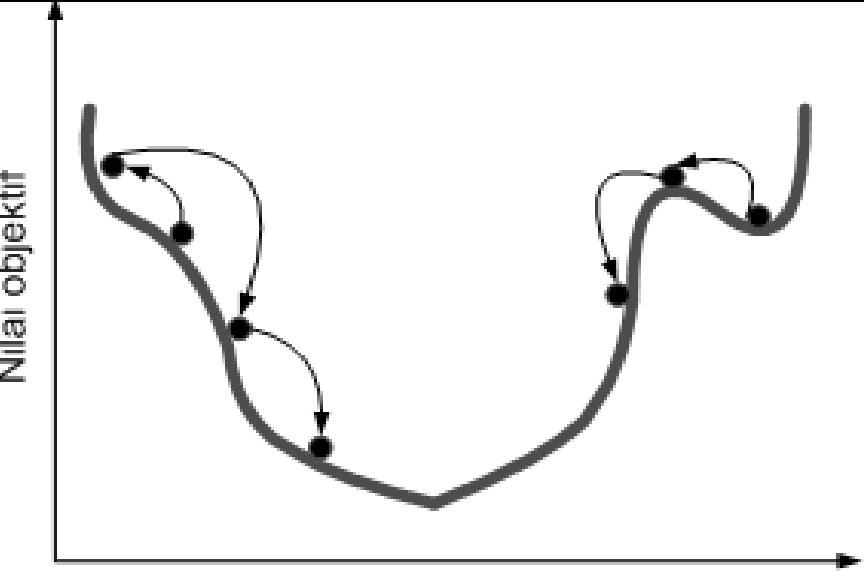
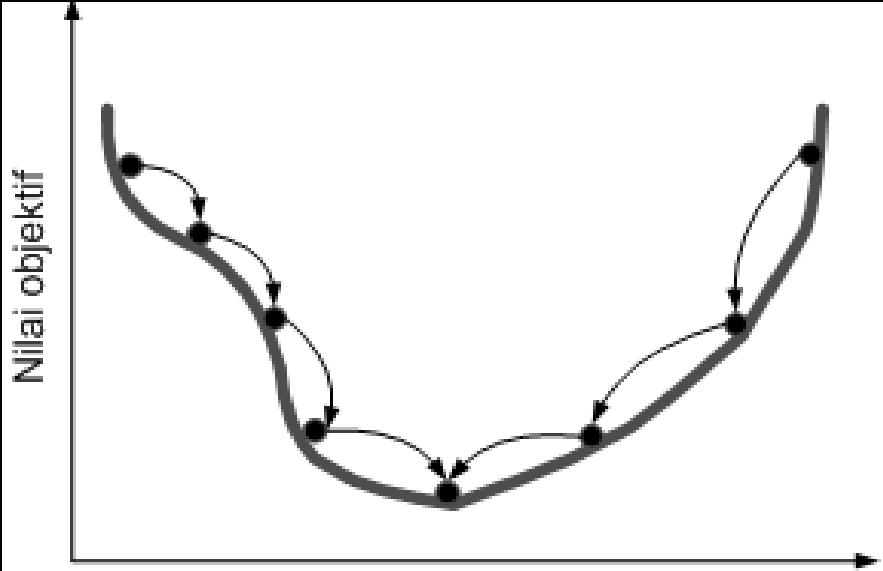


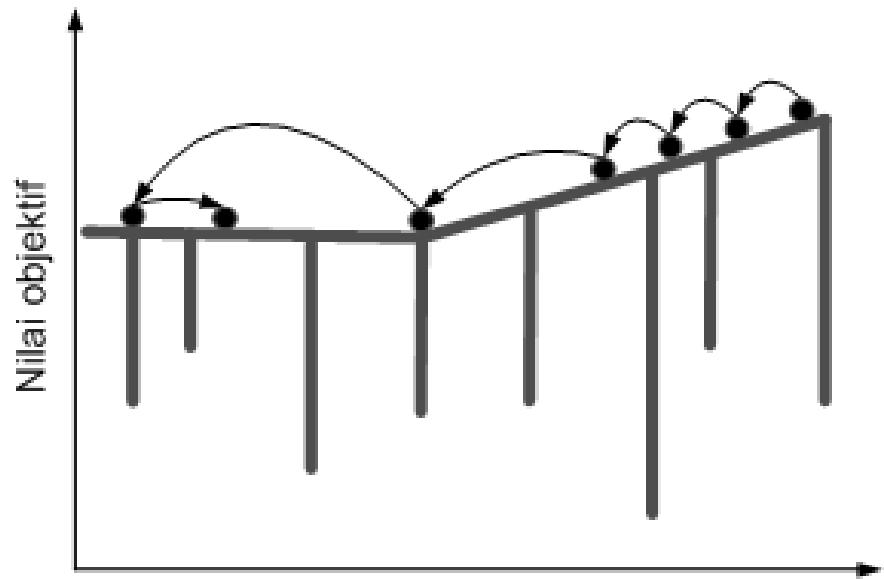
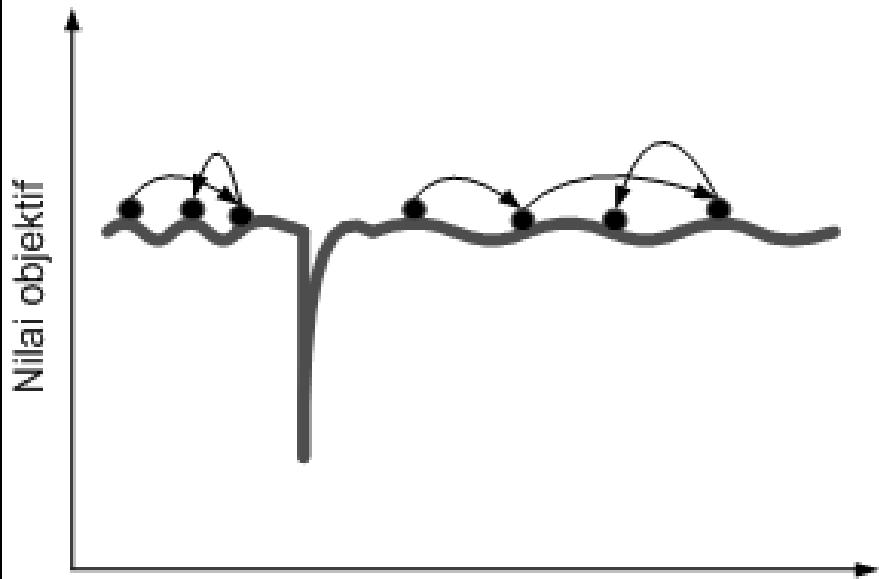
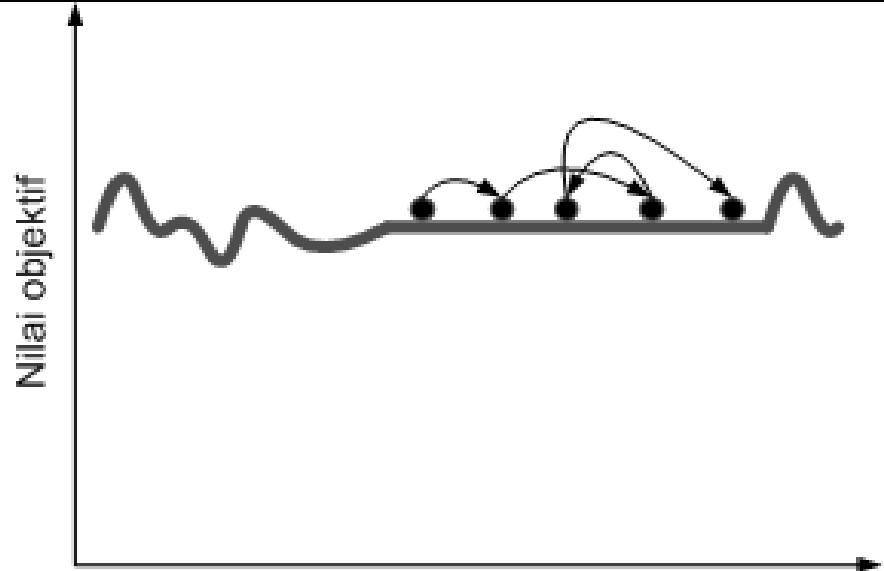
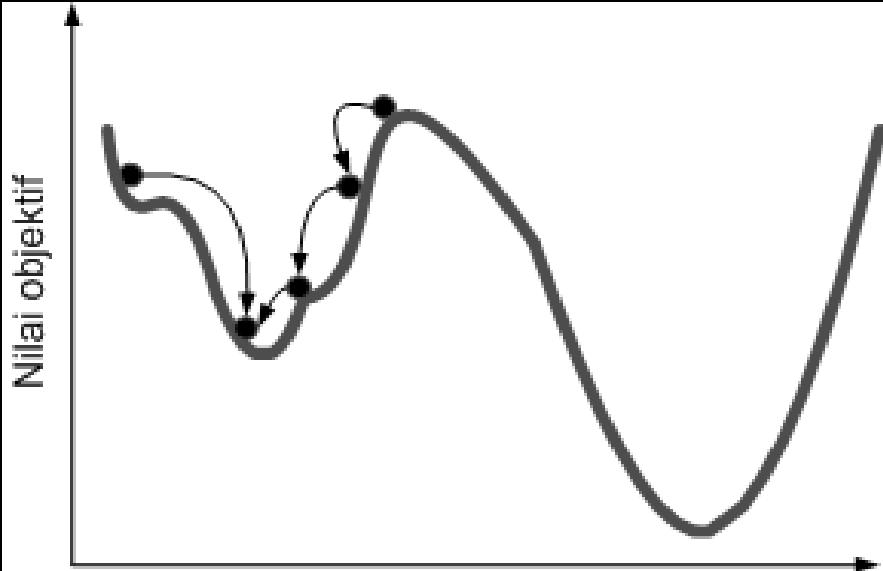














Apa itu EC?

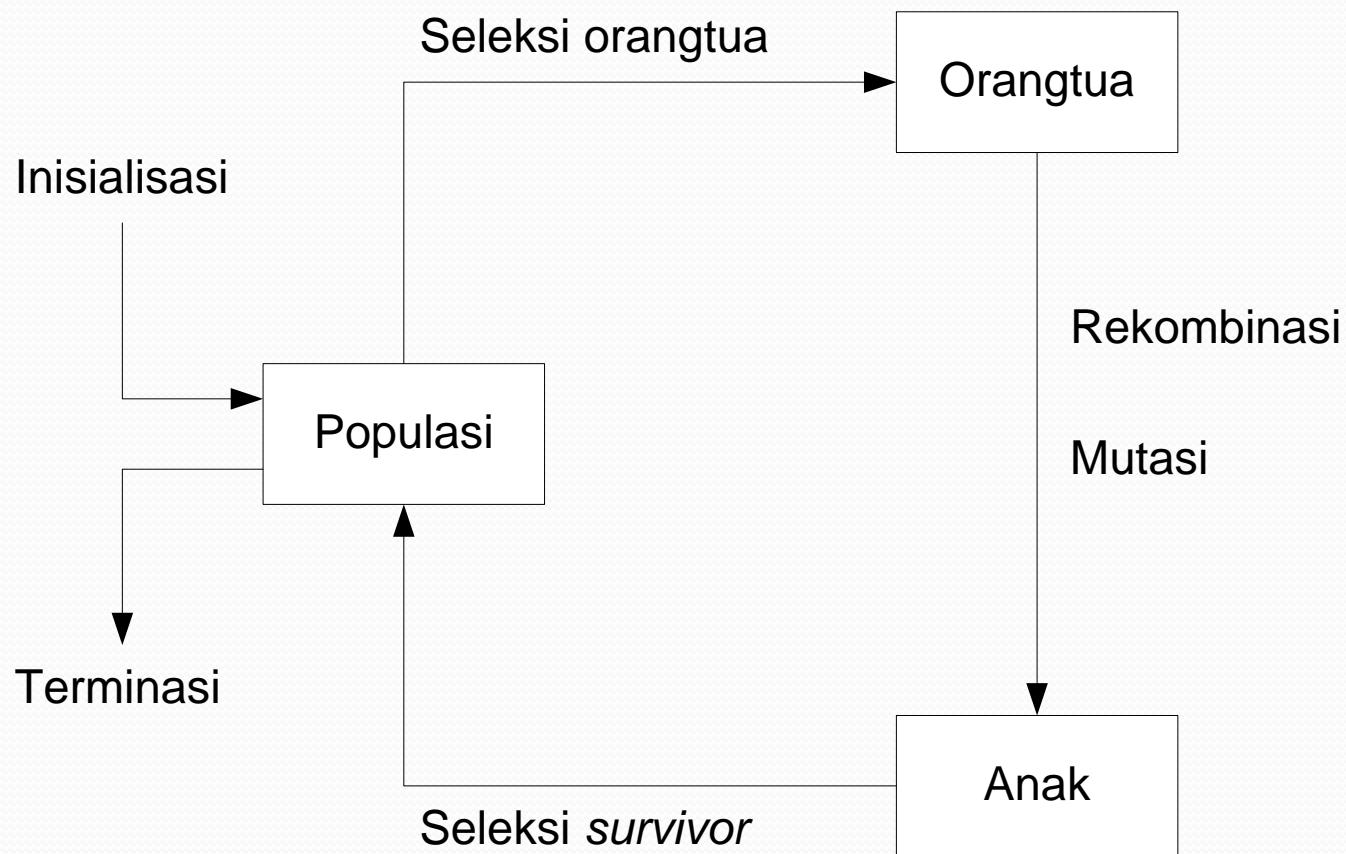
Evolutionary Computation is an abstraction from the theory of **biological evolution** that is used to create **optimization** procedures or methodologies, usually implemented on computers, that are used to solve problems“ [JULo7].

Apa itu EAs?

Evolutionary Algorithms are generic, **population-based meta-heuristic** optimization algorithms that use **biology-inspired** mechanisms like **mutation**, **crossover**, **natural selection** and **survival of the fittest**.

EAs = algoritma2 yang mengimplementasikan abstraksi **EC**

Skema umum EAs



TSP dengan 100 lokasi

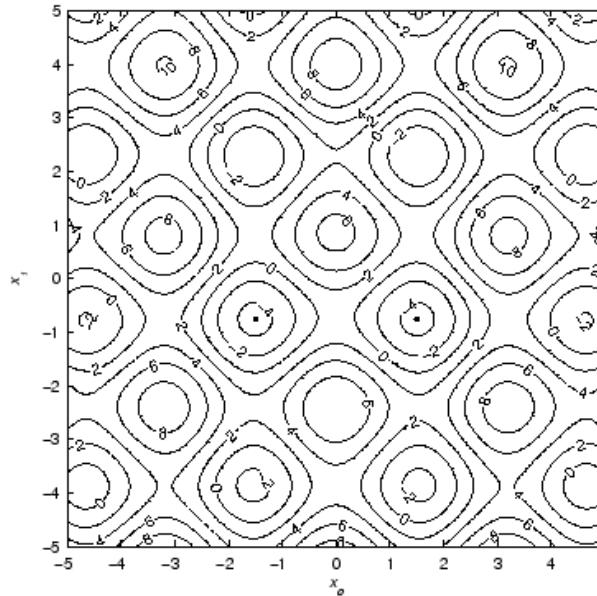
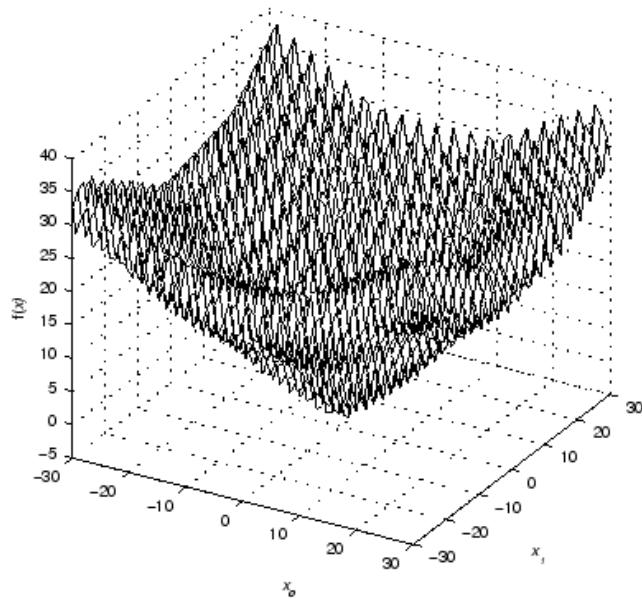
Seorang kurir punya waktu kerja: **8 jam**

Kriteria	Manual (berpikir)	Software A (Dijkstra)	Software B (GA)
Waktu running	0	2 jam	10 menit
Rute yang dihasilkan	11 jam	7 jam	7 jam 20 menit
Total Waktu	11 jam (lembur 3 jam)	9 jam (lembur 1 jam)	7,5 jam (tidak lembur)

Yang termasuk EAs:

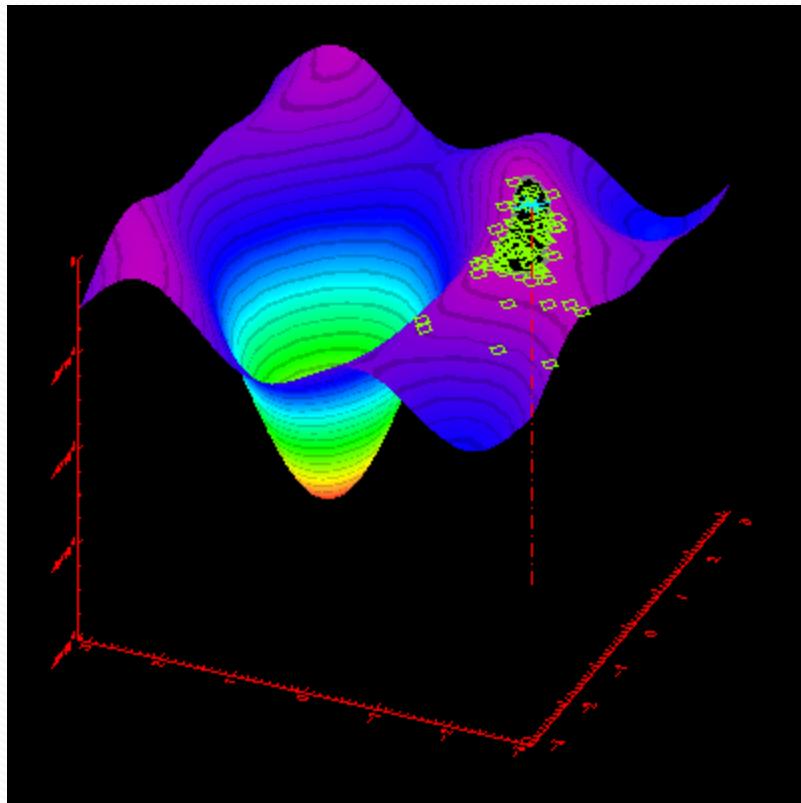
1. Genetic Algorithms (GA): binary strings
2. Evolution Strategies (ES): real-valued vectors
3. Evolutionary Programming (EP): finite state machines
4. Genetic Programming (GP): LISP trees
5. Differential Evolution (DE): Perkembangan dari ES
6. Grammatical Evolution (GE) ← Perkembangan GP

$$f(\vec{x}) = \sum_{i=0}^{D-1} \left(e^{-0.2} \sqrt{x_i^2 + x_{i+1}^2} + 3(\cos(2x_i) + \sin(2x_{i+1})) \right)$$

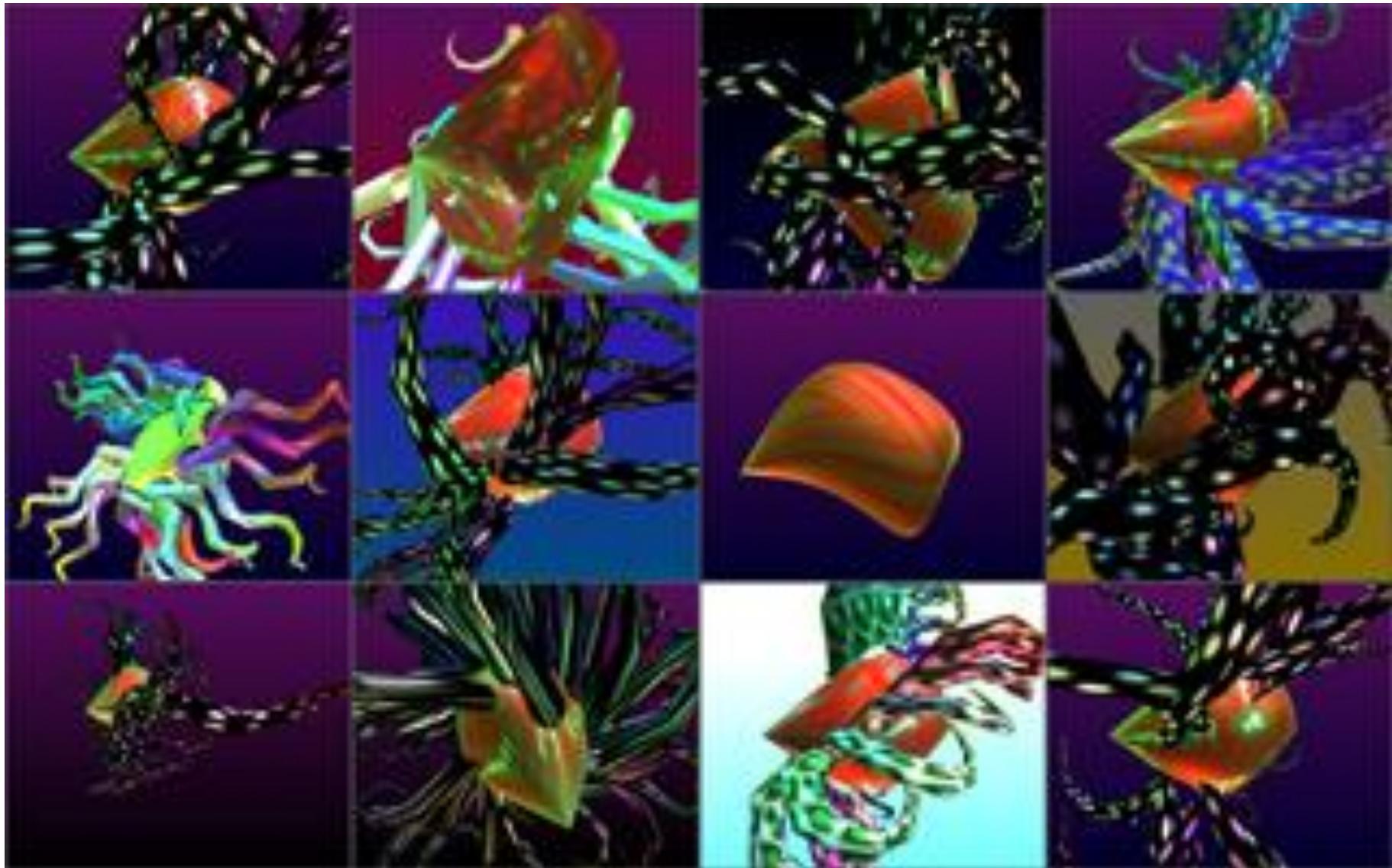


- Untuk presisi 10^{-9} → Berapa bit?
- Bisa menggunakan kromosom Real?

ES: *Self Adaptation*



Galapagos, Karl Sims 1997



Aplikasi-aplikasi EC

- **Optimasi**

- Penjadwalan Proyek, Perkuliahan, rumah sakit, dsb.
- Pengepakan Barang
- Pemotongan Bahan
- Instalasi Jaringan Telekomunikasi
- Instalasi Pipa Air
- Dsb.

Aplikasi-aplikasi EC

- **Pemodelan**

- ***Loan applicant creditibility***

Misalkan *British bank* yang membangun sistem untuk membuat model kredibilitas yang ber-evolusi untuk memprediksi tingkah laku nasabah-nasabah barunya dalam melakukan pembayaran hutang.

- **Prediksi penggunaan *bandwidth* jaringan**

Dengan menggunakan data-data pada masa lalu atau *historical data*, EC dapat digunakan untuk membangun model yang bisa memprediksi tingkah laku pelanggan menggunakan teknik *learning* (pembelajaran). Tentu saja data-data yang digunakan untuk proses learning harus representatif (menggambarkan kondisi berbagai macam kasus penggunaan *bandwidth* jaringan).

Aplikasi-aplikasi EC

- **Simulasi**

EC bisa digunakan untuk mensimulasikan perdagangan, ekonomi, kompetisi, dan sebagainya untuk melakukan kalibrasi model.

Dengan menggunakan model tersebut, kita bisa melakukan optimasi strategi dan pengambilan kebijakan. Salah satu contohnya adalah *Evolving Artificial Societies*.

EC masa depan

- Permasalahan semakin kompleks dan besar
- Beberapa diantaranya adalah:
 - Keuangan dan ekonomi
 - Robotika
 - Bioinformatika
 - Masalah optimasi dengan banyak tujuan atau ***multi-objective optimization.***

EC masa depan

- Bagaimana menemukan *forecasting rules*
- Membangun *bargaining strategies*
- Pemodelan ekonomi

EC masa depan

- Penggunaan *artificial markets* berbasis EC juga membantu kita memahami konsep-konsep dasar bidang ekonomi, seperti rasionalitas dan hipotesis pasar yang efisien.
- Di masa depan, ukuran dan kompleksitas data-data keuangan dan ekonomi global bisa dipastikan akan sangat besar.
- Jika perkembangan teknologi komputer, prosesor dan memori, tidak cukup cepat dalam mengimbangi besar dan kompleksitasnya permasalahan, maka EC akan menjadi satu-satunya teknik komputasi masa depan untuk bidang ini.

EC masa depan

- Pada bidang robotika, EC diharapkan bisa berperan banyak pada *RoboCup*, yaitu suatu proyek internasional yang mendorong lahirnya berbagai bidang riset seperti robotika dan *artificial intelligence*.
- Salah satu yang menarik adalah proyek *RoboCup soccer*, yakni membangun tim robot sepak bola.
- Tujuan akhir dari *RoboCup soccer* adalah mengalahkan tim (manusia) juara piala dunia sepak bola pada tahun 2050.

EC masa depan

- Lingkungan *RoboCup soccer* memberikan banyak masalah yang sangat menantang, seperti pemrosesan informasi secara waktu nyata (*real-time information processing*), penanganan data berderau (*noisy data handling*), kerjasama antar robot dalam bermain bola, dan strategi untuk memenangkan permainan.
- EC diharapkan bisa digunakan untuk membangun sistem strategi tim yang ber-evolusi (*evolving soccer team strategies*). Suatu strategi tim dikodekan sebagai untaian bilangan bulat (*integer string*) yang merepresentasikan sekumpulan aturan aksi dari sepuluh pemain [TOM07].

EC masa depan

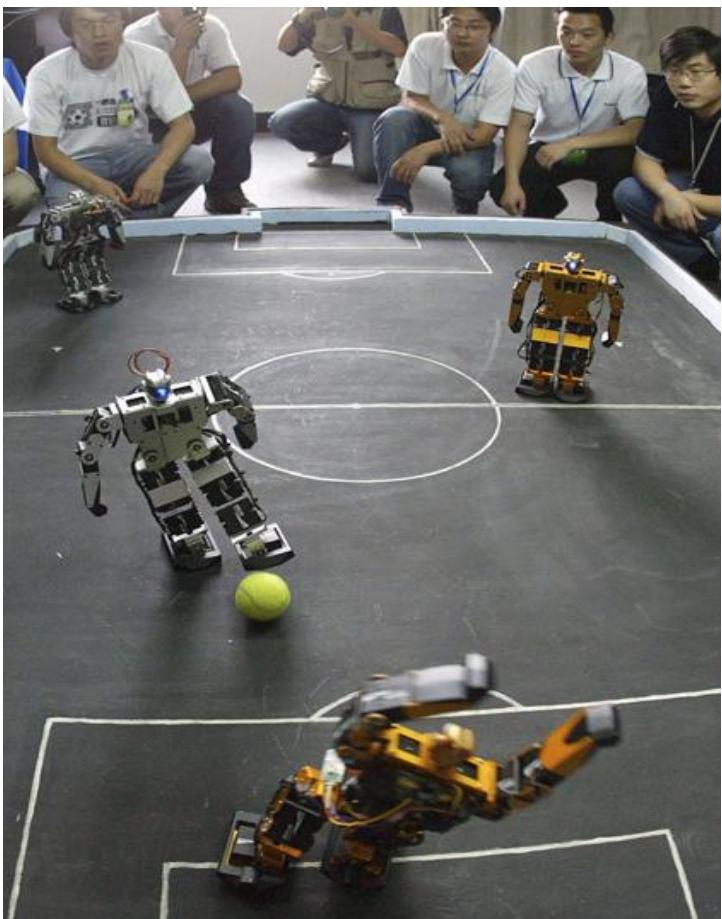
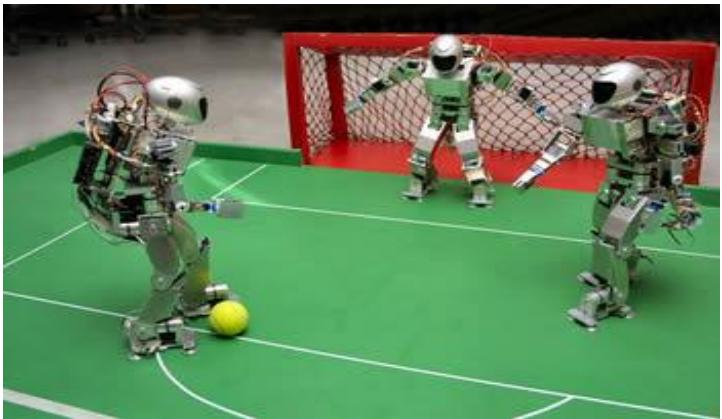
- EC digunakan untuk mencari sekumpulan aturan aksi yang optimal.
- Kalau kita bayangkan permainan sepak bola, jumlah strategi yang mungkin digunakan oleh masing-masing tim tentu saja sangat banyak atau bahkan tak terhingga. Selain itu, suatu tim bisa mengubah strategi setiap saat sehingga masalah ini bersifat *real-time*.
- Masalah dengan karakteristik tersebut tentu saja sangat sesuai untuk EC.

GP for Robot Soccer



WC-2050 is ours !!!





Kesimpulan

- EAs sangat powerful meski berpijak pada dua teori yang sangat lemah “Evolusi” & “Genetika”.
- Begitu berhasil membangun **kromosom** dan **fitness**, kita bisa menyelesaikan suatu masalah tanpa harus memikirkan analisa matematis dan algoritmanya
- Berbagai variasi EAs terus dikembangkan
- Beragam teknik pembangunan operator “evolusi” juga terus dikembangkan: representasi individu, seleksi orangtua, rekombinasi, mutasi, dan seleksi survivor.

Daftar Pustaka

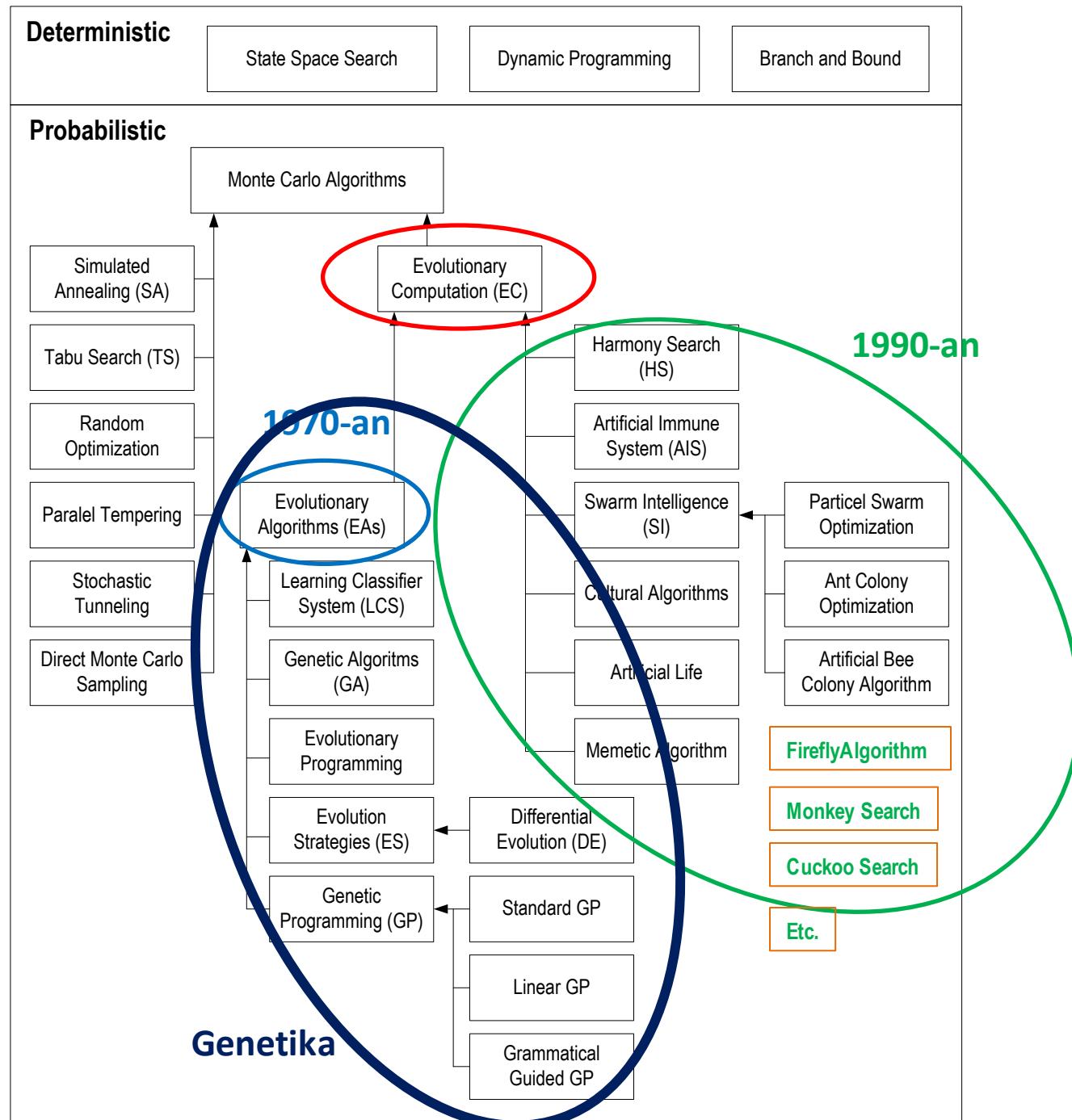
- [EIB03] Eiben, A.E. and Smith, J.E., 2003, "Introduction to Evolutionary Computing", Springer-Verlag Berlin Heidelberg.
- [ADN07] Adnan Oktar, 2007, "Mekanisme Khayalan Teori Evolusi", www.evolutiondeceit.com/indonesian/keruntuhan3.php
- [JUL07] Julie Leung, Keith Kern, Jeremy Dawson, 2007, "Genetic Algorithms and Evolution Strategies", presentation slides.
- [SUY08] Suyanto, 2008, Evolutionary Computation: Komputasi Berbasis "Evolusi" dan "Genetika", penerbit Informatika Bandung.
- [TOM07] Tomoharu Nakashima, 2007, "Evolving Soccer Teams for RoboCup Simulation", IEEE Congress on *Evolutionary Computation*, Singapore 25 – 28 September 2007.

Evolutionary Algorithms (EAs)

Dr. Suyanto, S.T., M.Sc.
HP/WA: 0812 845 12345

Intelligence Computing Multimedia (ICM)
Informatics faculty – Telkom University

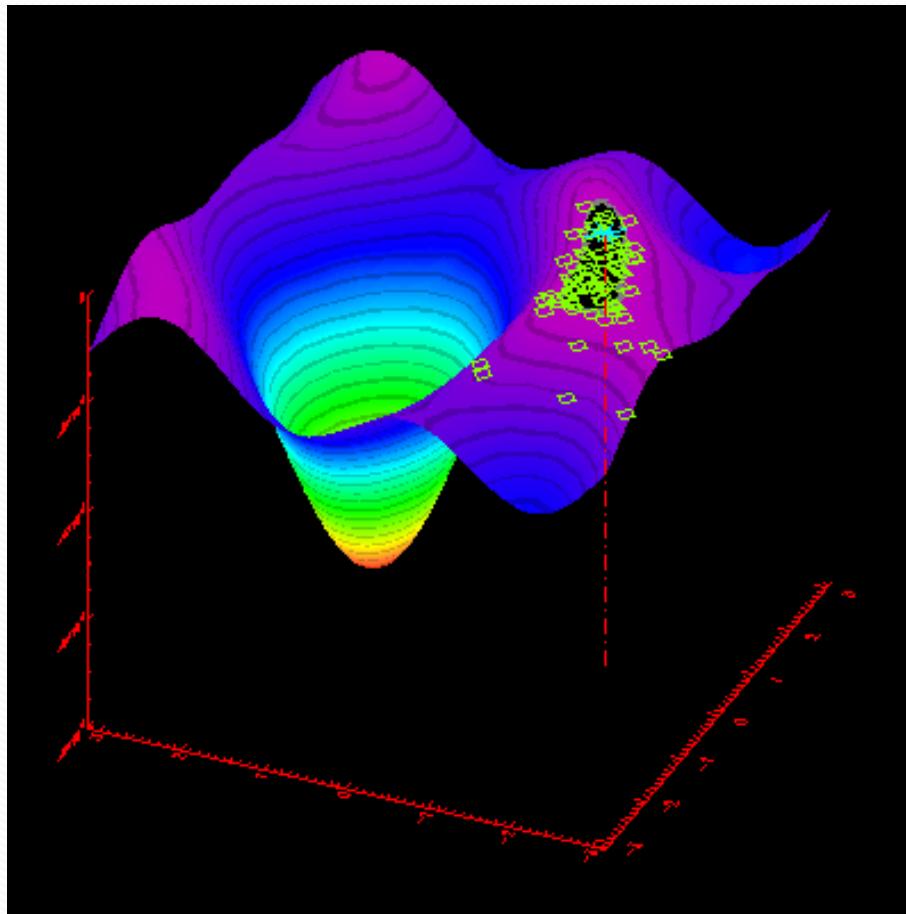
Optimization Algorithms



Swarm, Flock, School, Herd



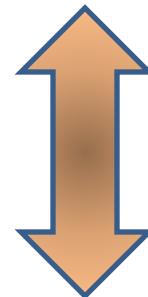
Self Adaptation



Terdapat kriteria tertentu → solusi atau bukan?

Searching

SPECIFIC



Optimization

GENERAL

Fungsi-fungsi objektif → konfigurasinya bagus atau tidak?

- **Searching:** terdapat kriteria tertentu yang menyatakan apakah suatu elemen x_i adalah solusi atau bukan.
- **Optimization:** mungkin tidak terdapat kriteria tersebut, melainkan hanya fungsi-fungsi objektif yang menggambarkan bagus atau tidaknya suatu konfigurasi yang diberikan.
- Fungsi-fungsi objektif bisa memberikan definisi masalah yang lebih umum → Optimization adalah **GENERALISASI** dari Searching.

Masalah Searching atau Optimasi?

1. Navigation System **Searching**
2. Travelling Salesman Problem (TSP) **Optimasi**
3. Vehicle Routing Problem (VRP) **Optimasi**
4. University Course Timetabling Problem **Optimasi**
5. Knapsack Problem **Searching**

Masalah searching bisa diselesaikan dengan algoritma Searching atau Optimasi. Begitu juga sebaliknya.

Apa itu EC?

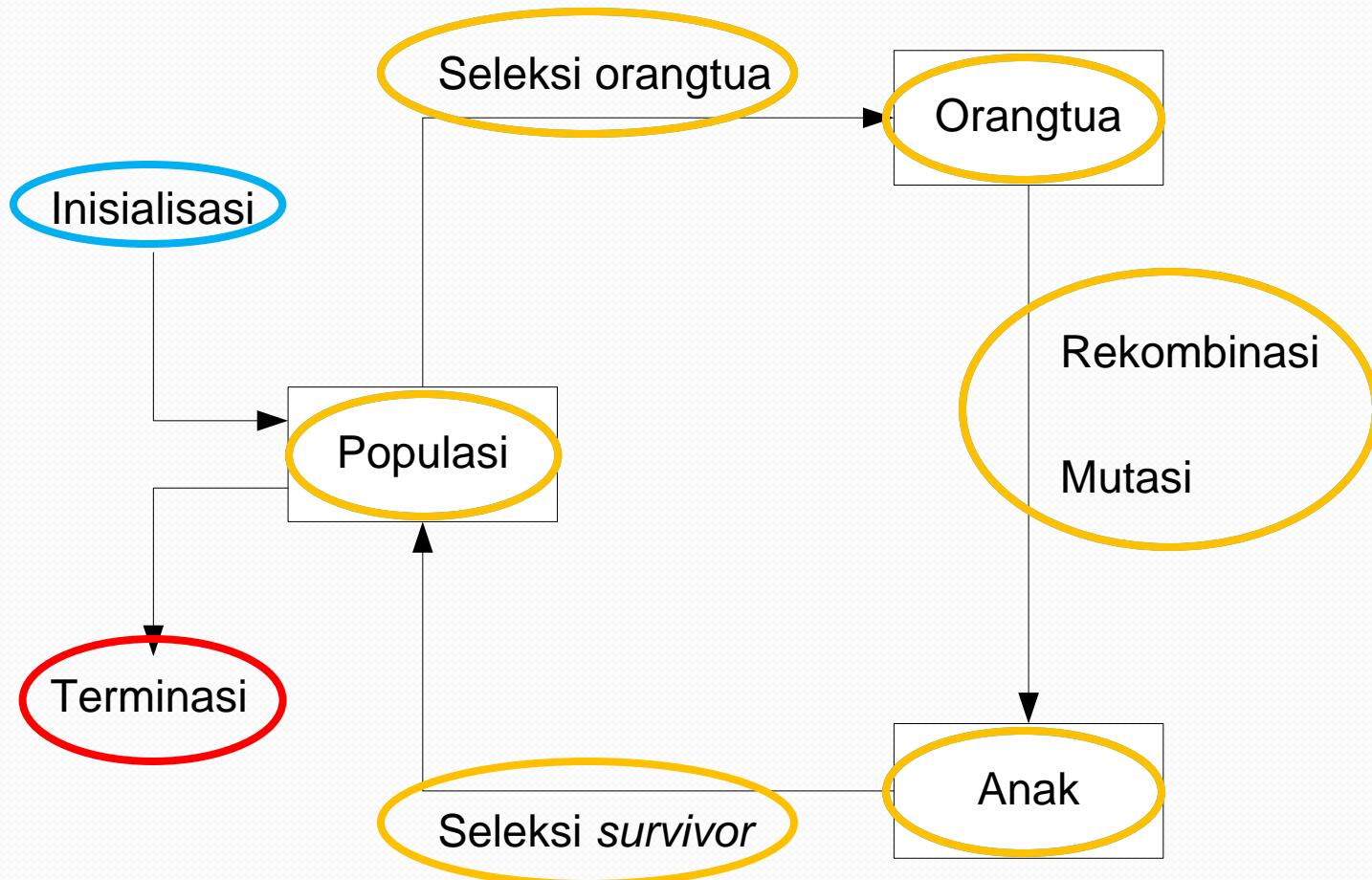
Evolutionary Computation is an abstraction from the theory of **biological evolution** that is used to create **optimization** procedures or methodologies, usually implemented on computers, that are used to solve problems“ [JULo7].

Apa itu EAs?

Evolutionary Algorithms are generic, **population-based meta-heuristic** optimization algorithms that use **biology-inspired** mechanisms like **mutation**, **crossover**, **natural selection** and **survival of the fittest**.

EAs = algoritma2 yang mengimplementasikan abstraksi **EC**

Skema umum EAs



EAs dengan *Steady State Replacement*

Bangkitkan populasi awal, N kromosom

random

Loop untuk N kromosom

individu = Dekode(kromosom)

fitness = Evaluasi(individu)

End

Loop sampai Kondisi Berhenti terpenuhi

Pilih dua kromosom sebagai orangtua P1 dan P2

random

[anak1, anak2] = Rekombinasi(P1, P2)

random

[anak1, anak2] = Mutasi(anak1, anak2)

random

Penggantian(populasi, anak1, anak2)

End

EAs dengan *Generational Replacement*

Bangkitkan populasi awal, N kromosom

random

Loop sampai Kondisi Berhenti terpenuhi

Loop untuk N kromosom

individu = Dekode(kromosom)

fitness = Evaluasi(individu)

End

Buat satu atau dua kopi kromosom terbaik

Loop sampai didapatkan N kromosom baru

Pilih dua kromosom sebagai orangtua P1 dan P2

random

[anak1, anak2] = Rekombinasi(P1, P2)

random

[anak1, anak2] = Mutasi(anak1, anak2)

random

End

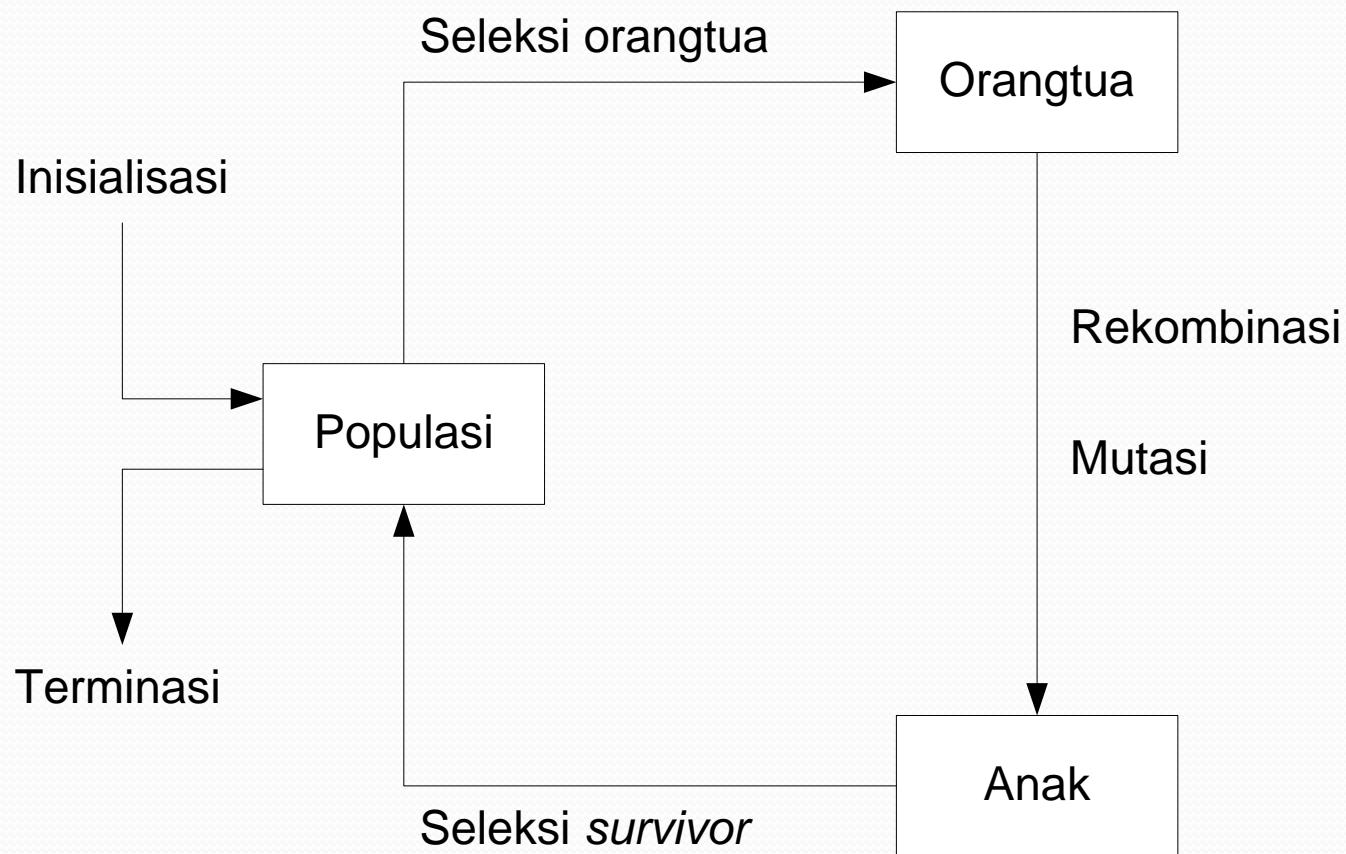
Ganti N kromosom lama dengan N kromosom baru

End

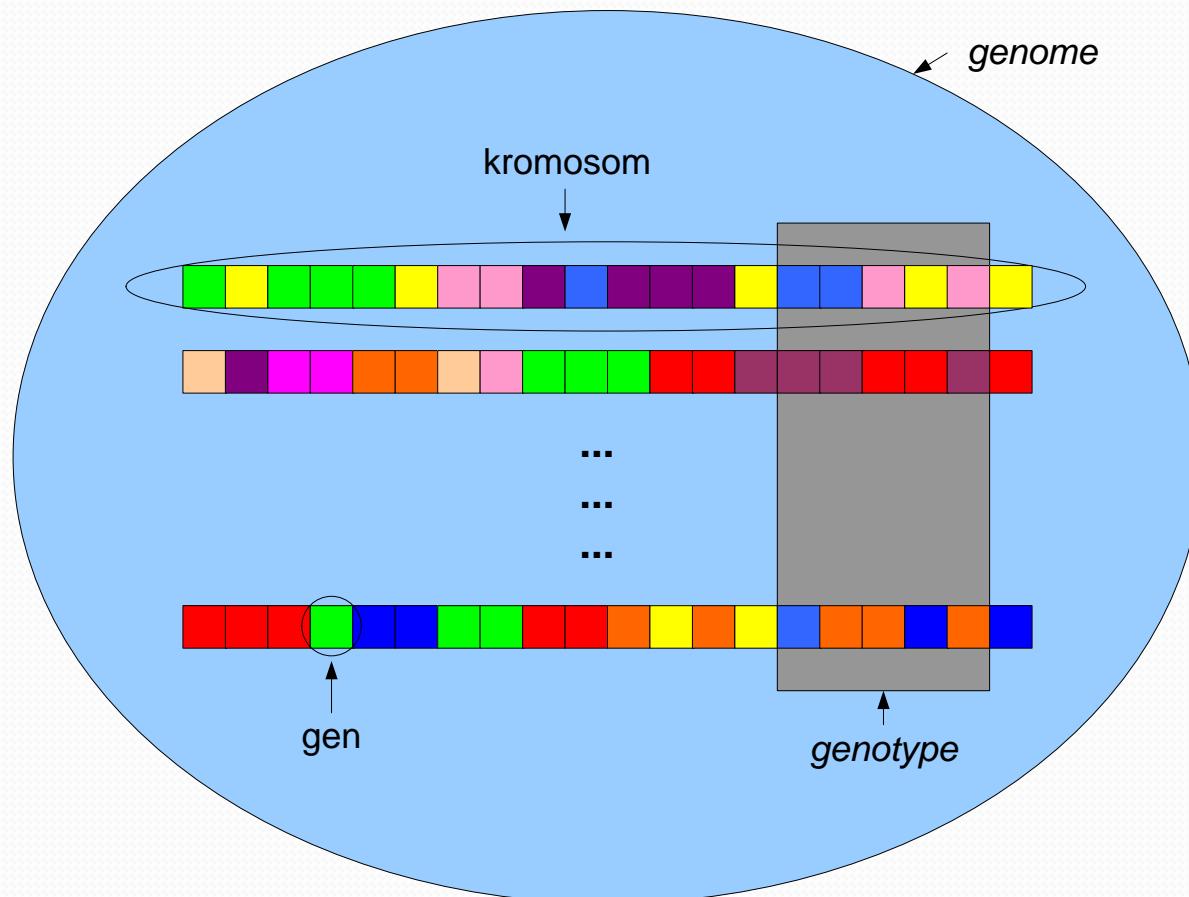
Yang termasuk EAs:

1. Genetic Algorithms (GA): binary strings
2. Evolution Strategies (ES): real-valued vectors
3. Evolutionary Programming (EP): finite state machines
4. Genetic Programming (GP): LISP trees
5. Differential Evolution (DE) → perkembangan ES
6. Grammatical Evolution (GE) → perkembangan GP

Skema umum EAs



Terminologi



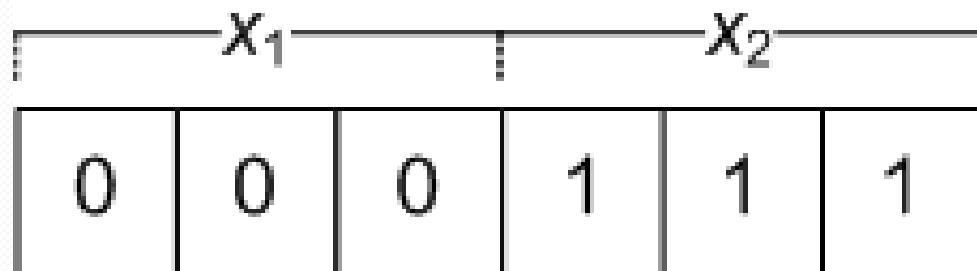
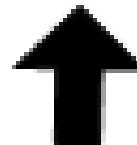
Pengkodean individu → kromosom

- Suatu **Individu** dikodekan ke dalam **Kromosom** dengan cara yang sesuai.
- Empat pengkodean yang umum adalah:
 - Biner
 - Integer
 - Real
 - Permutasi

1. Pengkodean Biner

Rentang Nilai: [-1,2]

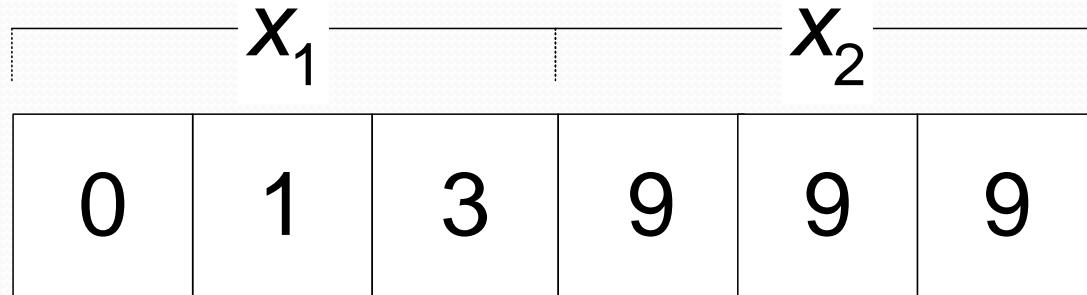
individu: $x_1 = -1$ dan $x_2 = 2$



2. Pengkodean Integer

Rentang Nilai: [-1,2]

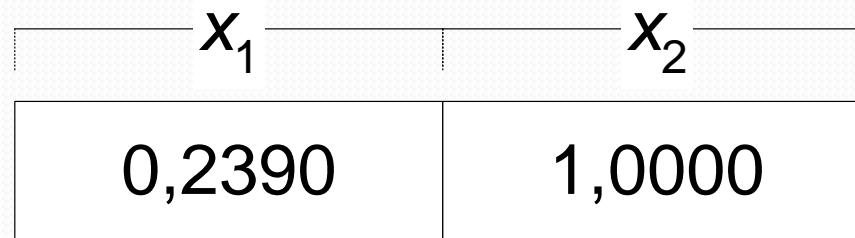
individu: $x_1 = -0,96096$ dan $x_2 = 2$



3. Pengkodean Real

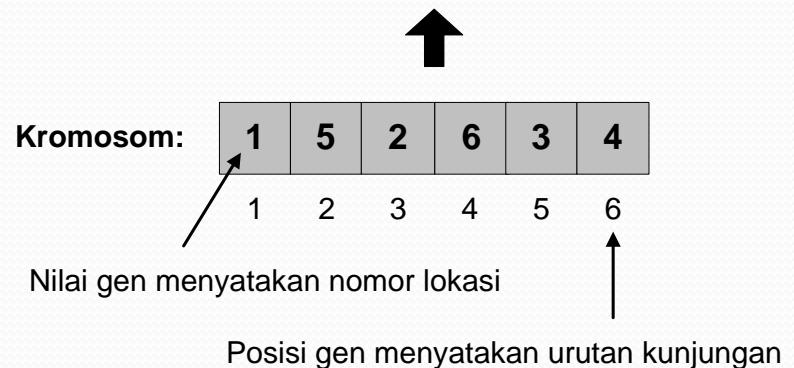
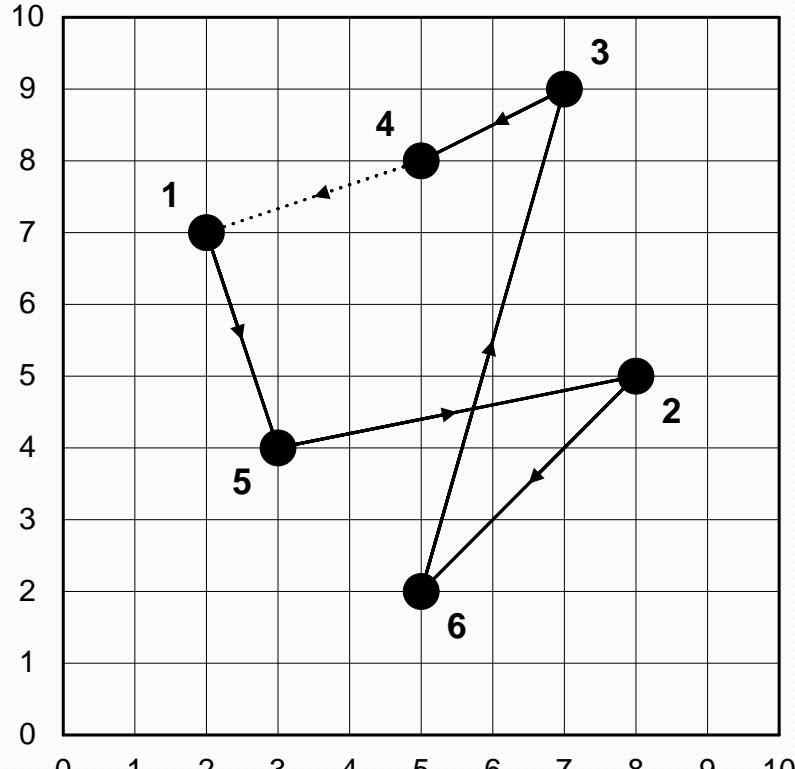
Rentang Nilai: [-1,2]

individu: $x_1 = -0,2830$ dan $x_2 = 2$



4. Pengkodean permutasi

Individu: urutan kunjungan semua lokasi



Pendekodean kromosom → individu

$$x = r_b + \frac{(r_a - r_b)}{N} (g_1 \cdot 2^{-1} + g_2 \cdot 2^{-2} + \dots + g_N \cdot 2^{-N}) \\ \sum_{i=1}^N 2^{-i}$$

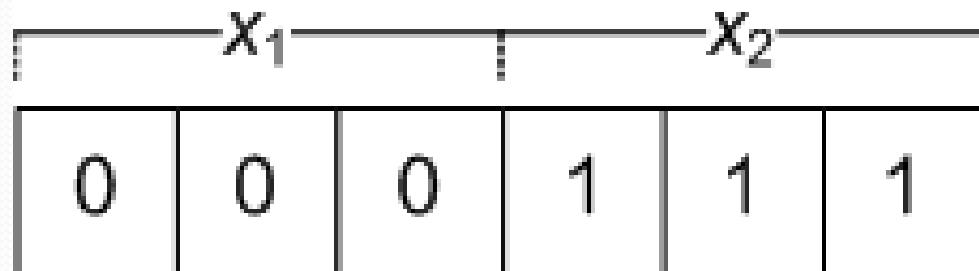
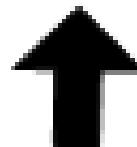
$$x = r_b + \frac{(r_a - r_b)}{N} (g_1 \cdot 10^{-1} + g_2 \cdot 10^{-2} + \dots + g_N \cdot 10^{-N}) \\ \sum_{i=1}^N 9 \cdot 10^{-i}$$

$$x = r_b + (r_a - r_b) (g_1 + g_2 + \dots + g_N)$$

1. Pengkodean Biner

Rentang Nilai: [-1,2]

individu: $x_1 = -1$ dan $x_2 = 2$



Nilai *Fitness*

- Maksimasi

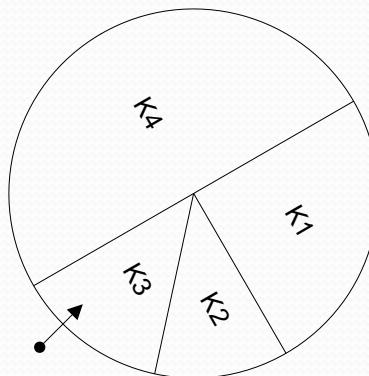
$$f = h$$

- Minimasi

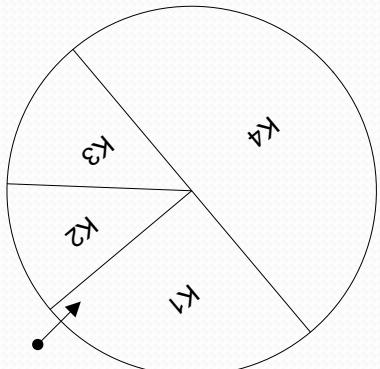
$$f = \frac{1}{(h + a)}$$

Seleksi Orangtua

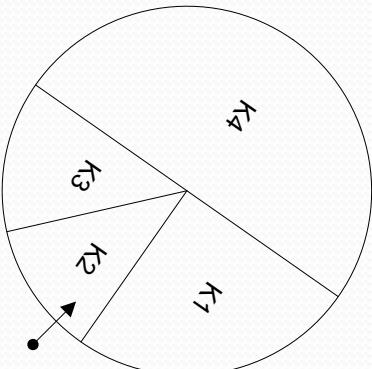
Kromosom	Fitness
K1	2
K2	1
K3	1
K4	4
Jumlah	8



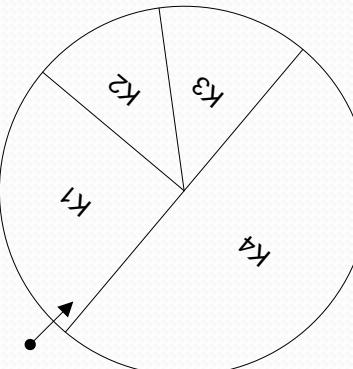
Putaran ke-1



Putaran ke-2



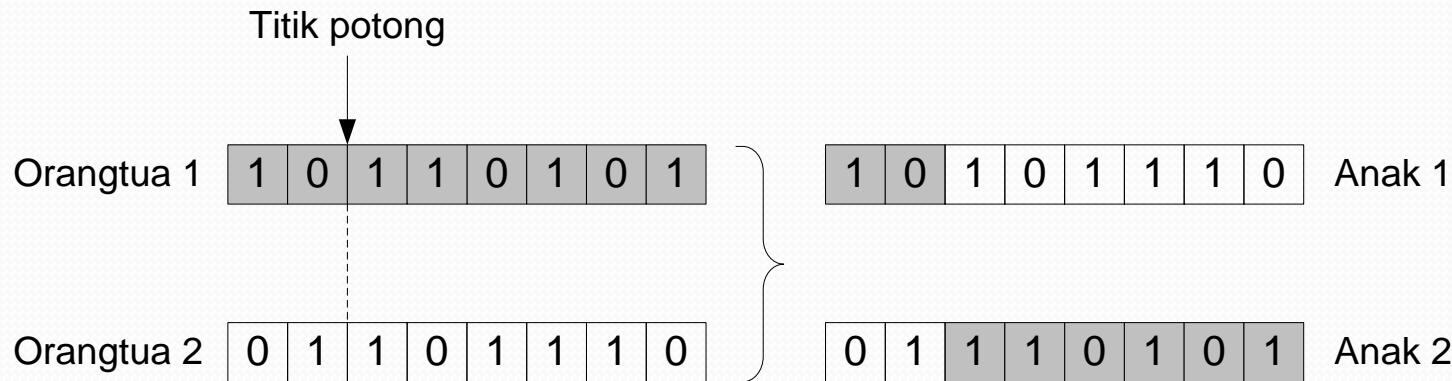
Putaran ke-3



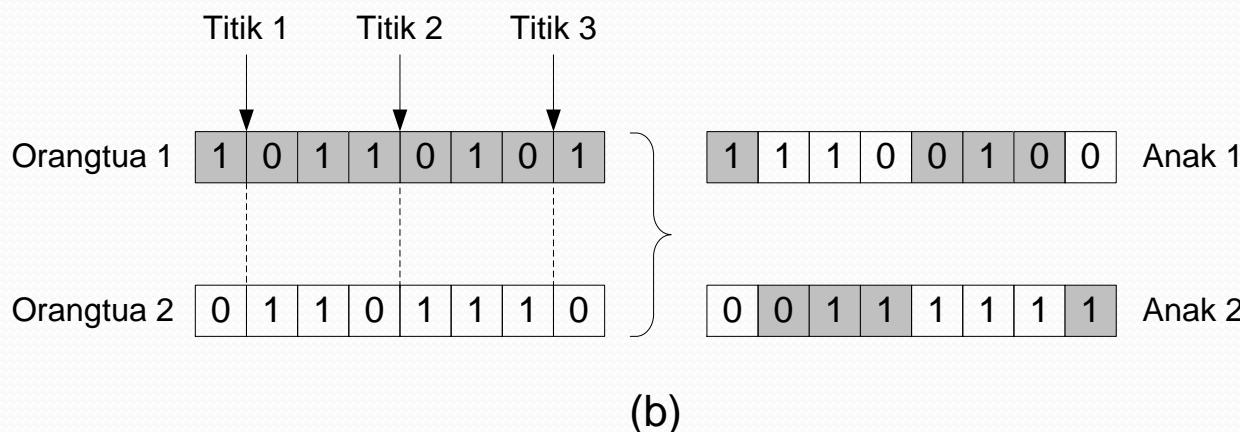
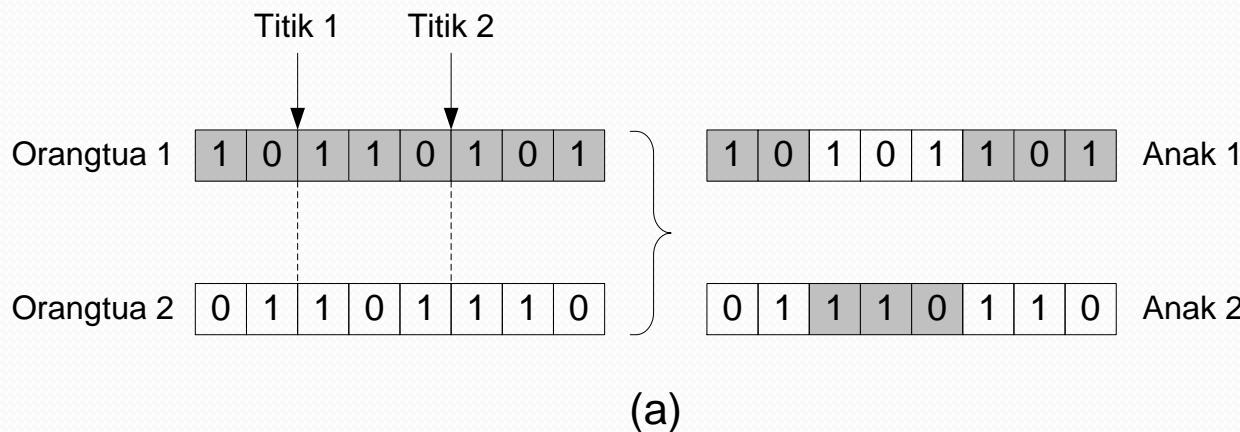
Putaran ke-4

Metode: *roulette wheel*

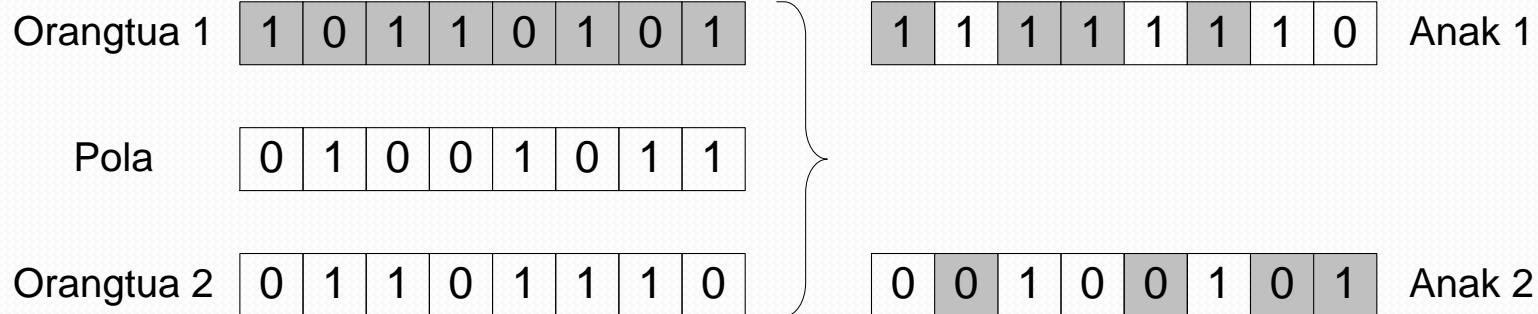
Rekombinasi/Crossover



Rekombinasi



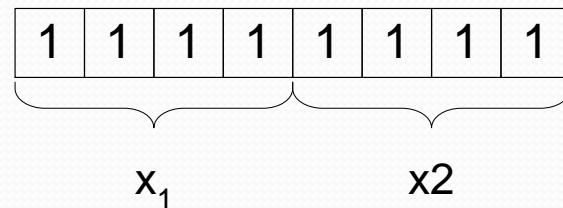
Rekombinasi



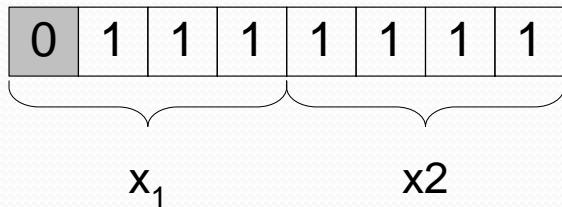
Mutasi

$h = 5x_1 + 2x_2 \rightarrow$ Maksimasi h dimana x_1 & x_2 : integer [0,15]

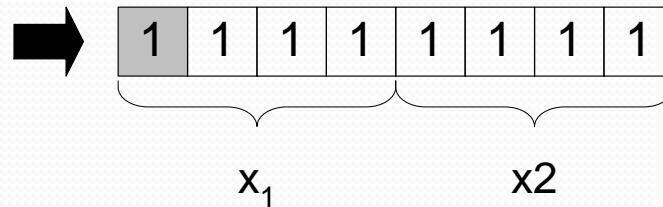
Kromosom yang menghasilkan nilai maksimum

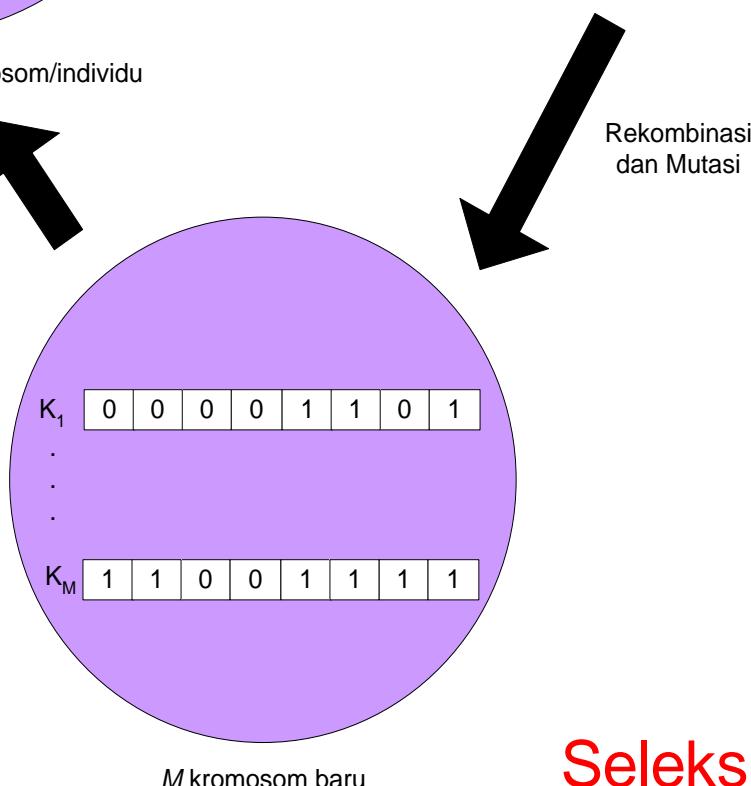
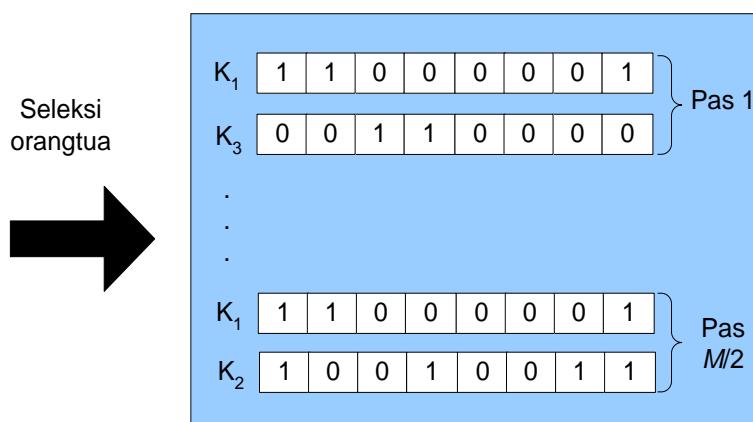
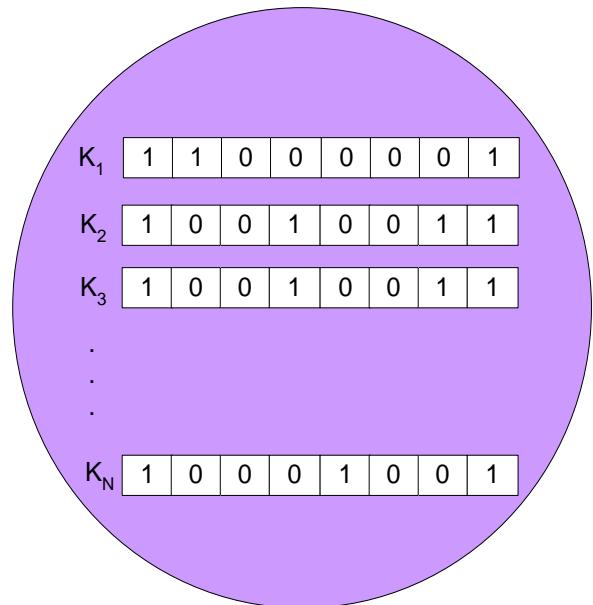


Kromosom awal

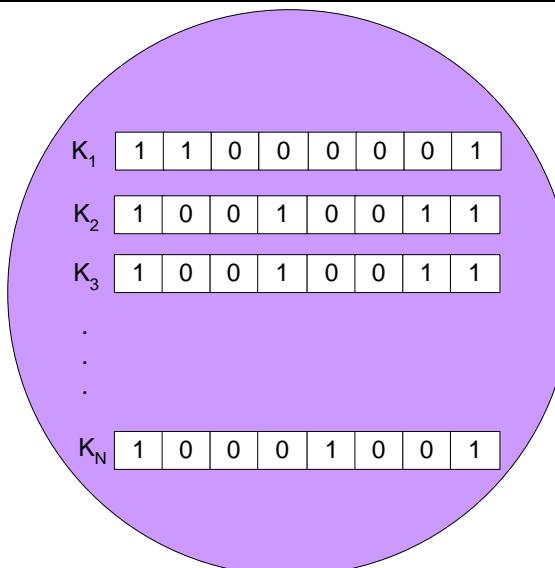


Kromosom hasil mutasi

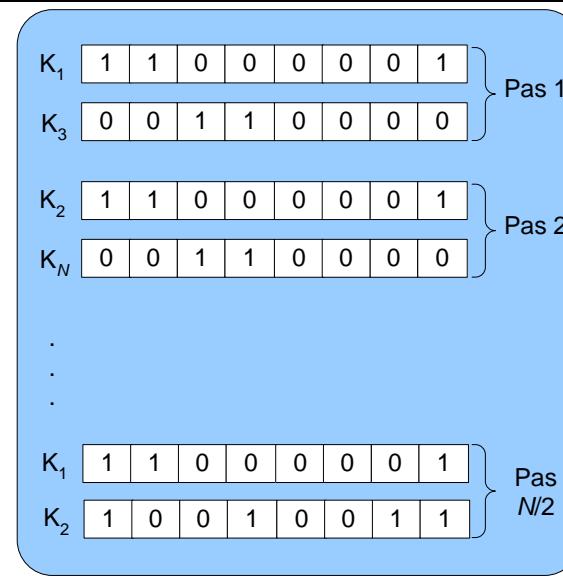
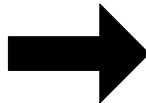




Seleksi Survivor: Steady State



Seleksi orangtua



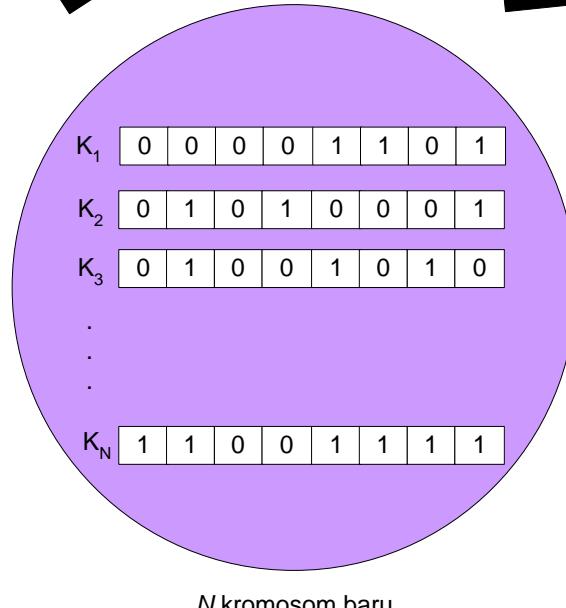
Pas 1

Pas 2

Pas $N/2$

Penggantian N kromosom

Rekombinasi dan Mutasi



Seleksi Survivor: *Generational*

Studi kasus 1: Maksimasi Fungsi

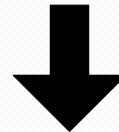
$$h(x_1, x_2) = 7x_1 - 3x_2$$

x_1, x_2 dalam interval [0, 15]

$h = \mathbf{105}$, dimana $x_1 = 15$ dan $x_2 = 0$

Phenotype → Genotype

individu: $x_1 = 5$ dan $x_2 = 3$



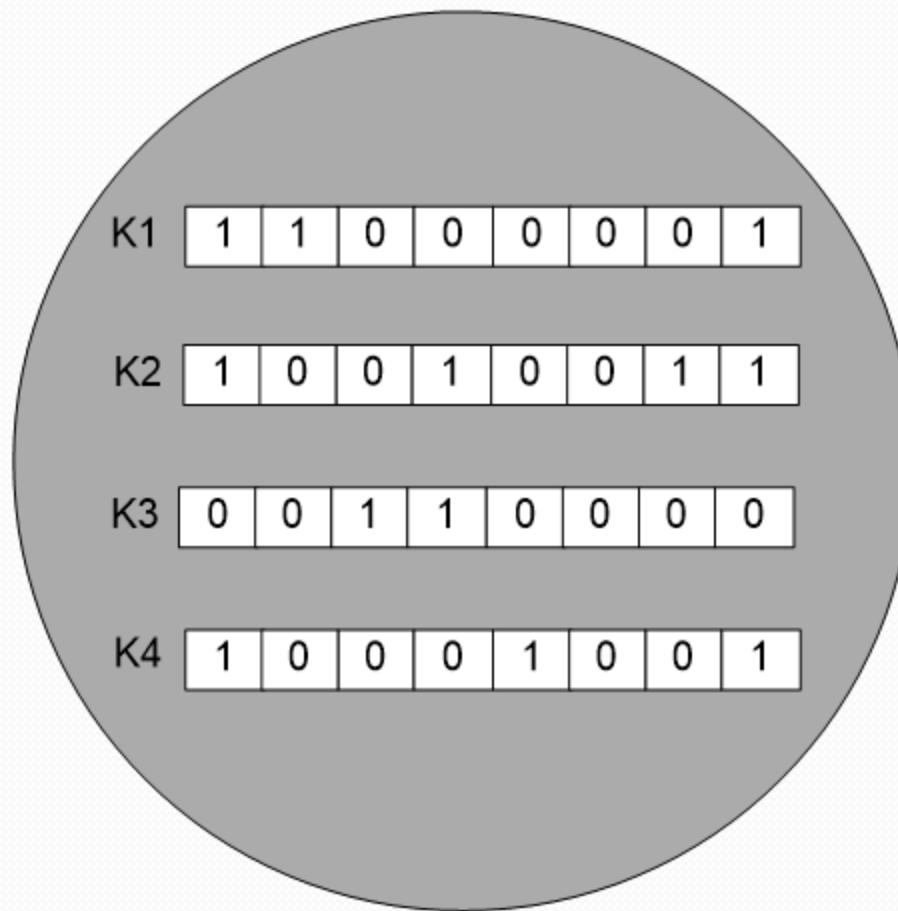
kromosom dengan *binary encoding*

x_1	x_2
0	1
0	1
1	0
0	0
1	1
1	1

Fungsi Fitness: Maksimasi

$$f = h$$

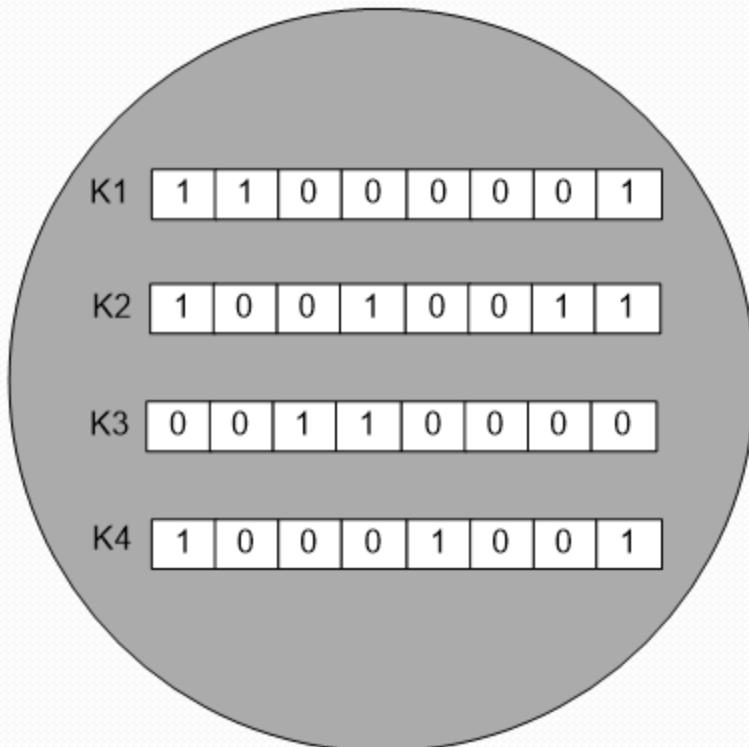
Generasi 1



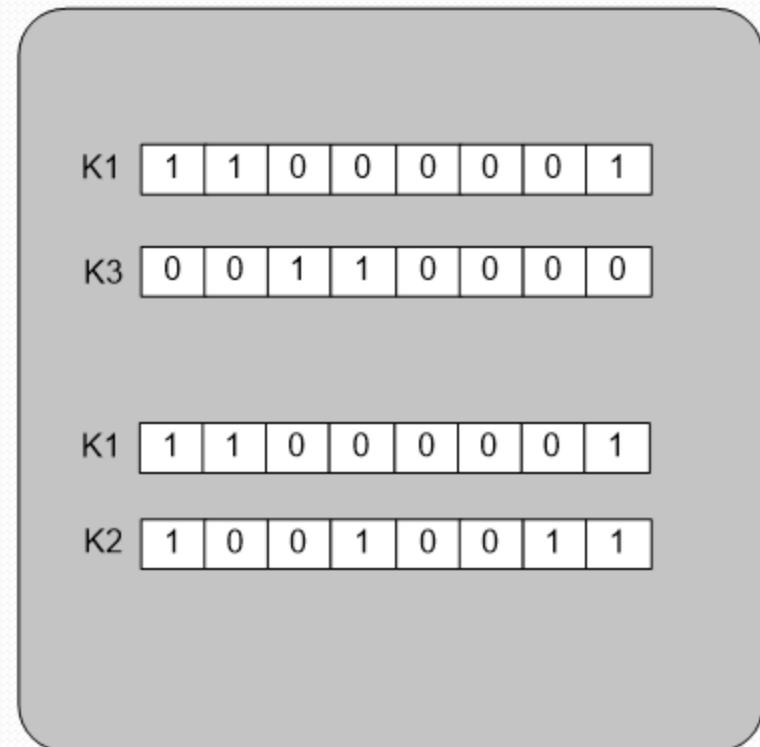
Seleksi Ortu: generasi 1

No	Kromosom	Individu (x_1, x_2)	Fitness ($7x_1 - 3x_2$)	Probabilitas terpilih	Jumlah yang diharapkan di <i>mating pool</i>	Jumlah aktual di <i>mating pool</i>
1	11000001	(12, 1)	81	0,41	1,64	2
2	10010011	(9, 3)	54	0,27	1,08	1
3	00110000	(3, 0)	21	0,11	0,44	1
4	10001001	(8, 5)	41	0,21	0,84	0
Jumlah			197	1,00	4,00	4
Rata-rata			49,25	0,24	1,00	1
Maksimum			81	0,41	1,64	2

Generasi 1



Populasi dengan 4 kromosom: K1 sampai K4



Mating pool: [K1, K3] dan [K1, K2]

Rekombinasi Ortu: generasi 1

No	Kromosom orangtua	Posisi titik rekombinasi	Kromosom anak hasil rekombinasi	Individu anak (x_1, x_2)	Fitness ($7x_1 - 3x_2$)
1	11 000001	2	11110000	(15, 0)	105
2	00 110000	2	00000001	(0, 1)	-3
3	110000 01	6	11000011	(12, 3)	75
4	100100 11	6	10010001	(9, 1)	60
Jumlah					244
Rata-rata					61
Maksimum					105

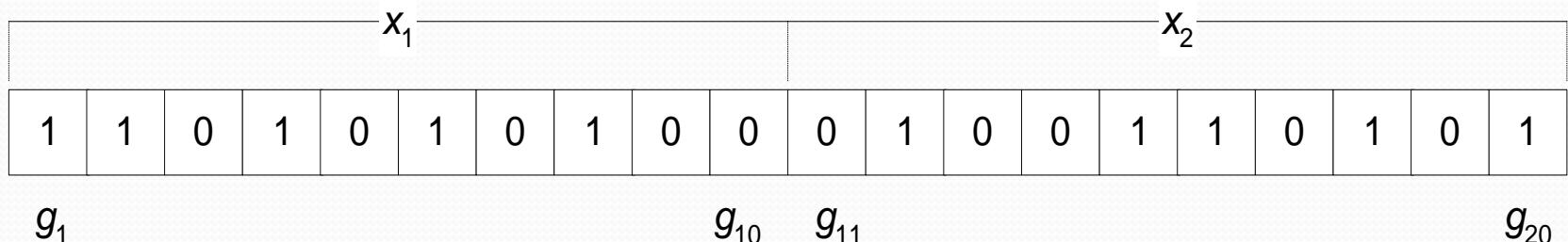
Studi kasus 2: Minimasi

Nilai minimum $h = ?$

$$h(x_1, x_2) = x_1^2 + x_2^2$$

$$x_1, x_2 \in [-5,12;5,12]$$

Pengkodean Individu → kromosom



Fitness

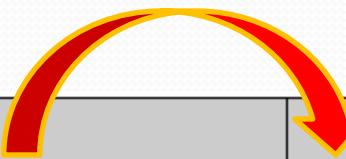
$$f = \frac{1}{(x_1^2 + x_2^2) + 0,01}$$

Jika nilai minimum = 0, nilai maks $f = ?$

Generasi 1

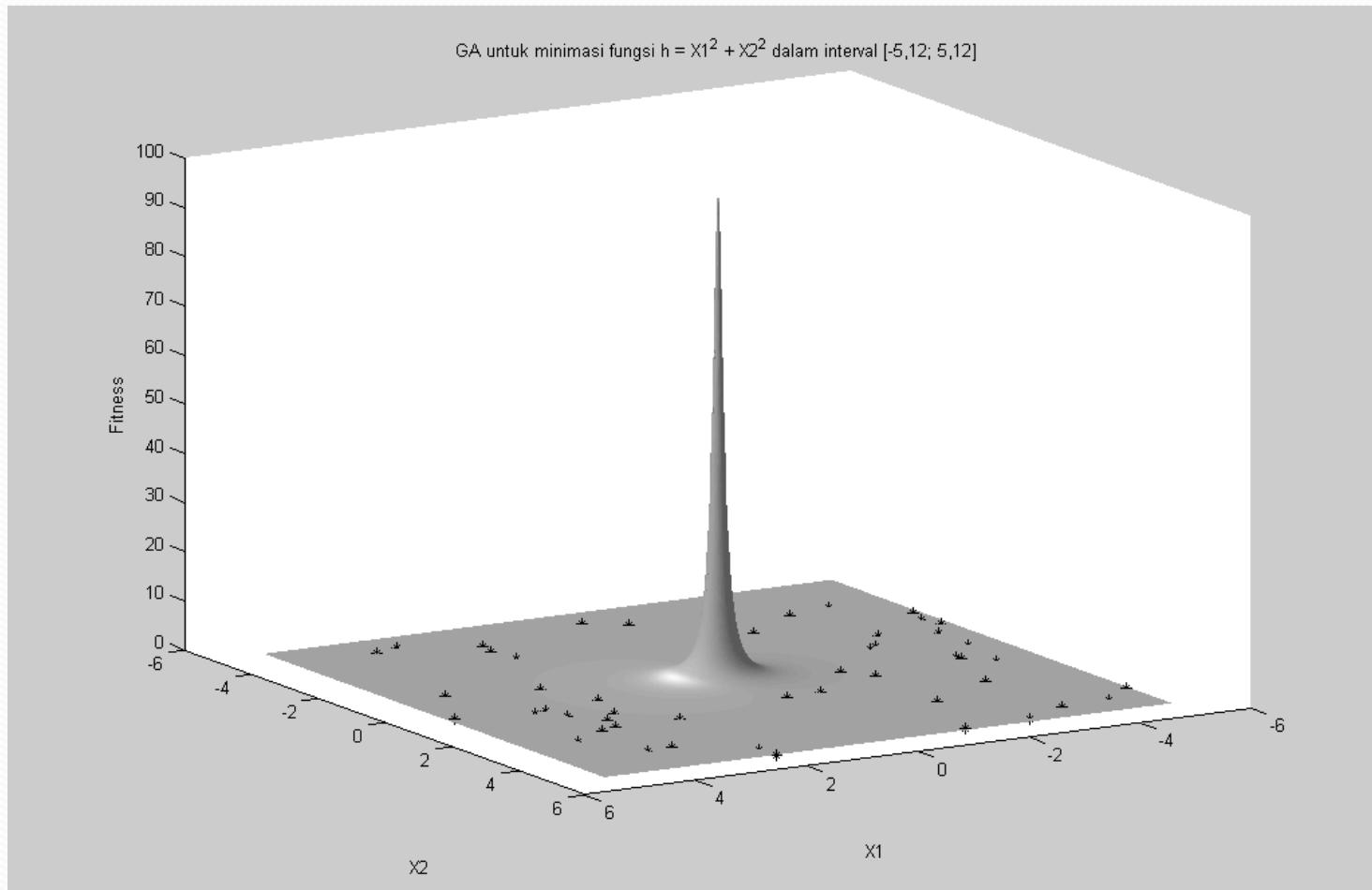
$$x = r_b + \frac{(r_a - r_b)}{\sum_{i=1}^N 2^{-i}} (g_1 \cdot 2^{-1} + g_2 \cdot 2^{-2} + \dots + g_N \cdot 2^{-N})$$

$$x_1, x_2 \in [-5,12; 5,12]$$



No	Genotype	Phenotype		Nilai fitness
	kromosom biner	X1	X2	
1	00010011011001101110	-4.35	1.1	0.049646
2	11001101110001000011	3.11	-4.45	0.033916
3	10110010111111001110	2.03	4.62	0.039254
4	11001110001101111101	3.12	3.81	0.041219
5	11001110101011011001	3.14	2.17	0.068594
6	00101110000110110110	-3.28	-0.74	0.08837
7	01111011111010110010	-0.17	1.78	0.31179
50	11010110011000111011	3.45	0.59	0.081562

Generasi 1



Generasi 10

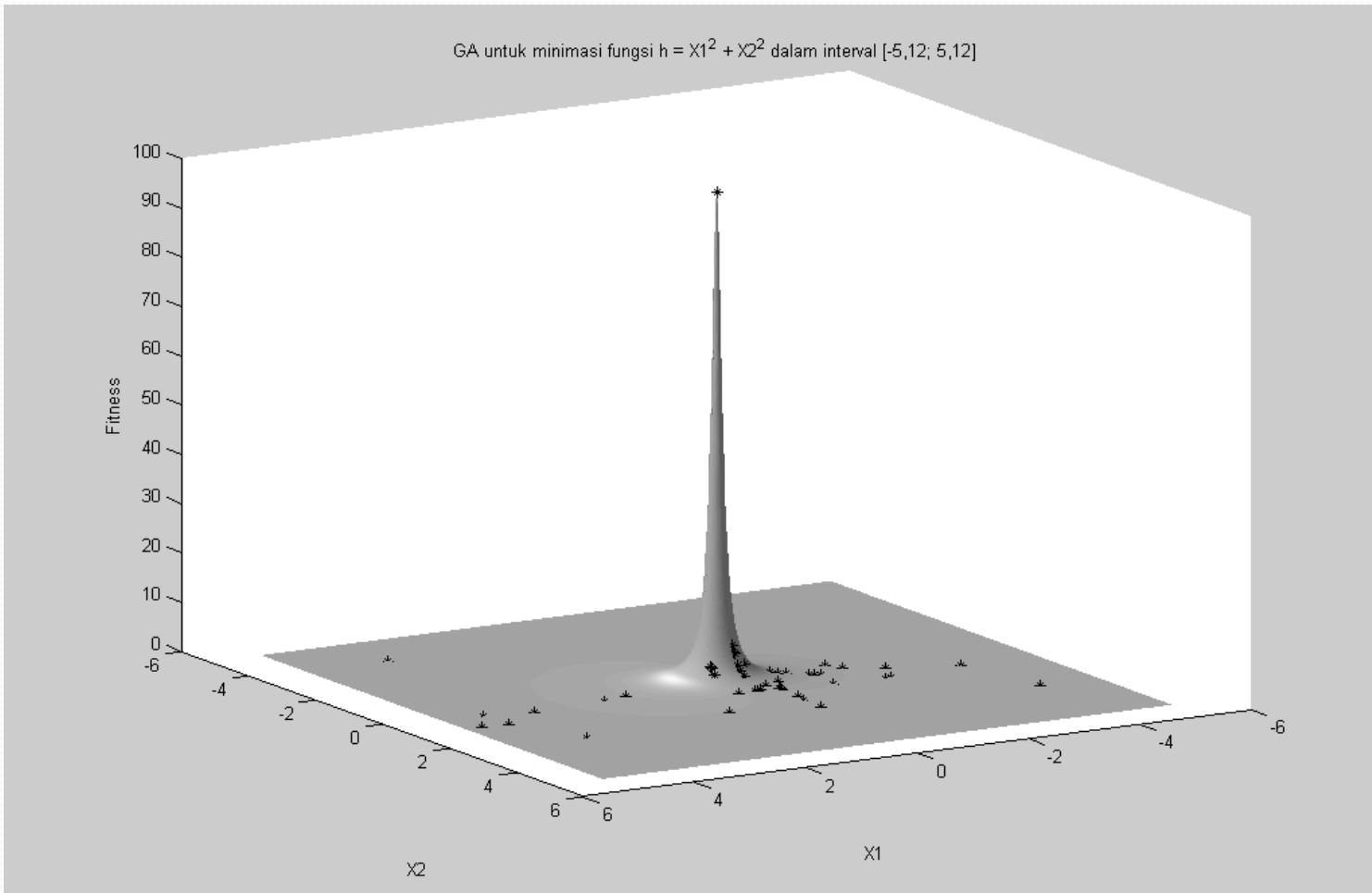
$$x = r_b + \frac{(r_a - r_b)}{\sum_{i=1}^N 2^{-i}} (g_1 \cdot 2^{-1} + g_2 \cdot 2^{-2} + \dots + g_N \cdot 2^{-N})$$

$$x_1, x_2 \in [-5,12; 5,12]$$

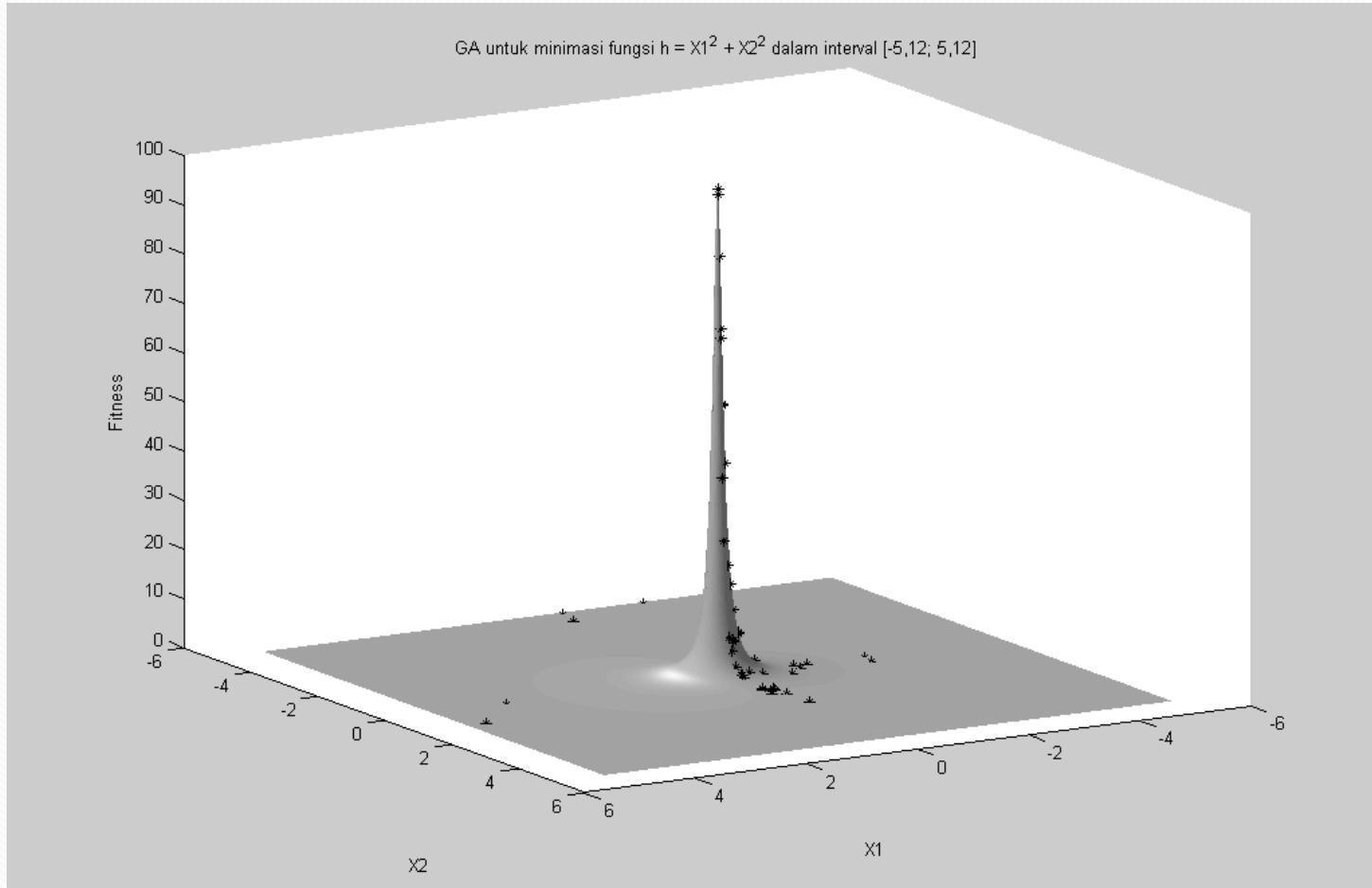


No	Genotype	Phenotype		Nilai fitness
	kromosom biner	X1	X2	
1	01111111110000000000	-0.01	0	99.01
2	01111111110000000000	-0.01	0	99.01
3	01111101010001000001	-3.77	1.03	0.065429
4	01111101011001110001	-2.4	1.2	0.1387
5	01111001111000100001	3.58	0.52	0.076355
6	01011101111000101010	4.83	1.01	0.041053
7	01111101111000100001	-1.38	1.2	0.29812
...				
50	01111101111000000001	-1.93	0.02	0.26772

Generasi 10



Generasi 100

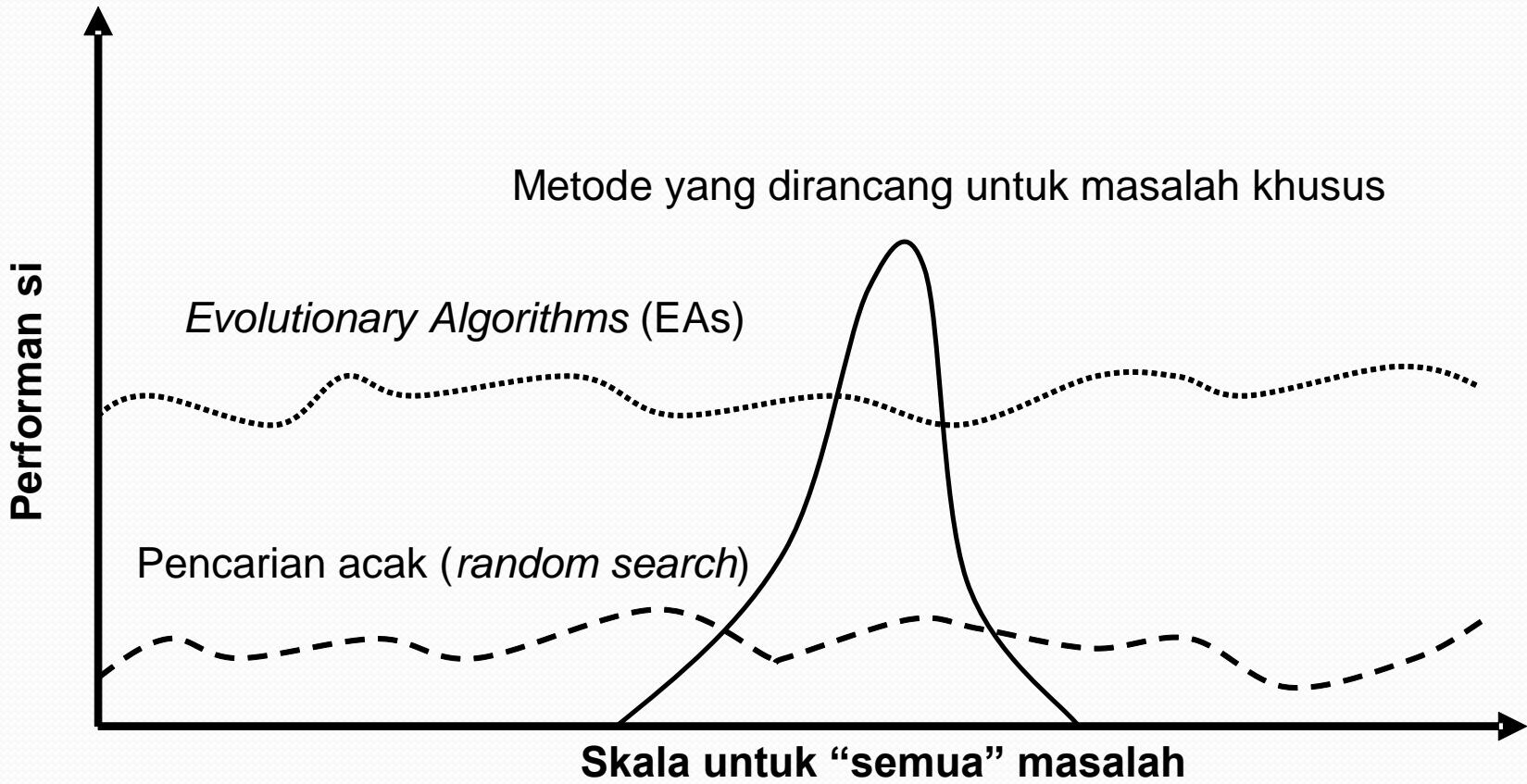


Mengapa EAs?

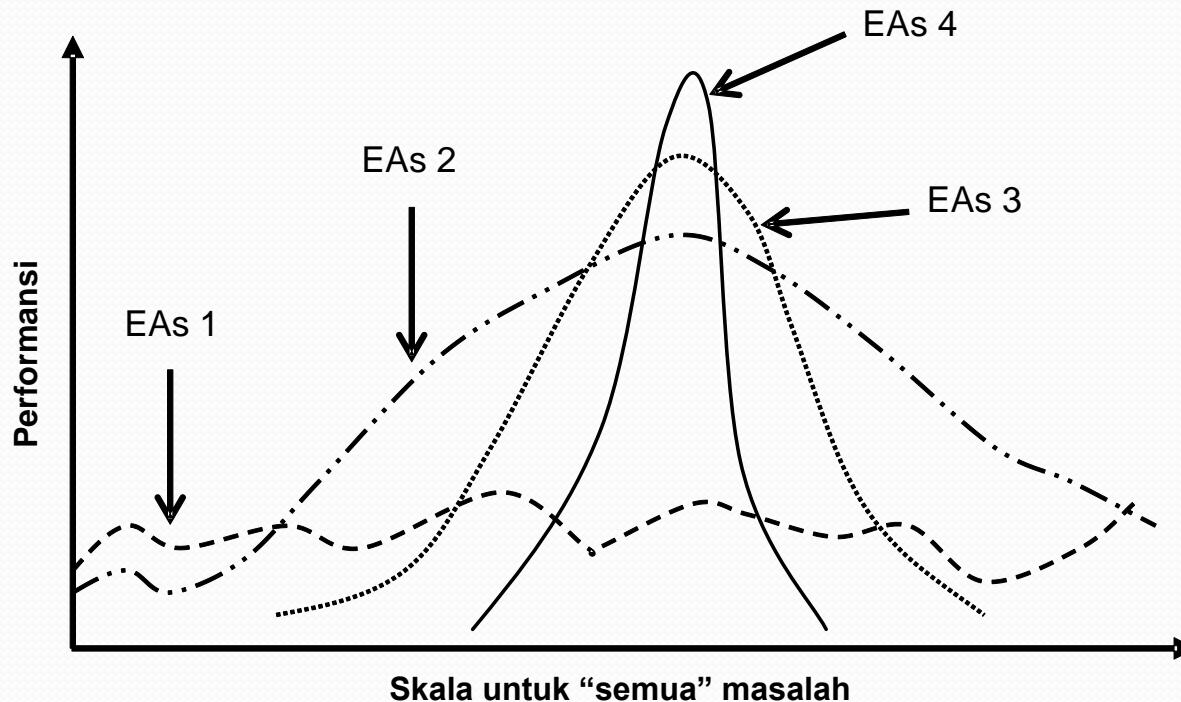
EAs sangat baik untuk permasalahan yang memiliki satu atau lebih ciri-ciri berikut ini:

- Ruang masalah sangat besar, kompleks, dan sulit dipahami;
- Tidak bisa diselesaikan menggunakan metode-metode konvensional;
- Terdapat batasan waktu, misalnya dalam sistem waktu nyata (*real time system*).
- Solusi yang diharapkan tidak harus paling optimal, tetapi 'bagus' atau bisa diterima;
- Kurang atau bahkan tidak ada pengetahuan yang memadai untuk merepresentasikan masalah ke dalam ruang pencarian yang lebih sempit;
- Tidak tersedia analisa matematika yang memadai;

Performansi EAs (Goldberg, 1989)



Performansi EAs (Michalewicz, 1996)



EAs 2, EAs 3, dan EAs 4 adalah EAs yang ditambahkan pengetahuan khusus yang memiliki akurasi lebih baik dibandingkan EAs 1 (tanpa pengetahuan).

Kapan EAs digunakan?

- Jika kita menghadapi masalah TSP untuk *graph* asimetris **10 node**, apakah kita harus menggunakan EAs? Jawabannya mungkin saja tidak perlu karena ada algoritma lain (misal Dijkstra atau dynamic programming) yang performansinya lebih baik.
- Tetapi, jika kita menghadapi masalah TSP untuk *graph* asimetris **1000 node**, apakah kita harus menggunakan EAs? Jawabannya mungkin “ya” karena algoritma lain membutuhkan waktu yang sangat lama.

Kesimpulan

- EAs adalah algoritma-algoritma yang mengimplementasikan abstraksi EC
- Terdapat dua variasi *survivor selection* atau *replacement scheme* , yaitu *Steady State* dan *Generational Replacement*.

Kesimpulan

- Jika pengetahuan yang ditambahkan semakin banyak, maka EAs akan memiliki performansi yang baik untuk berbagai masalah.
- Teori terbaru menyatakan bahwa “menemukan suatu algoritma yang bisa digunakan untuk semua masalah adalah mustahil”.

Daftar Pustaka

- [THO96] Thomas Bäck. 1996, “Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms”, Oxford University Press, ISBN: 0195099710, January 1996.
- [THO97] Thomas Bäck, David B. Fogel, and Zbigniew Michalewicz, editors, 1997, “Handbook of Evolutionary Computation”, Computational Intelligence Library. Oxford University Press in cooperation with the Institute of Physics Publishing, Bristol, New York, ringbound edition, ISBN: 0750303921, April 1997.
- [SUYo8] Suyanto, 2008, Evolutionary Computation: Komputasi Berbasis “Evolusi” dan “Genetika”, penerbit Informatika Bandung.
- [JUL07] Julie Leung, Keith Kern, Jeremy Dawson, 2007, “Genetic Algorithms and Evolution Strategies“, presentation slides.

Genetic Algorithm (GA)

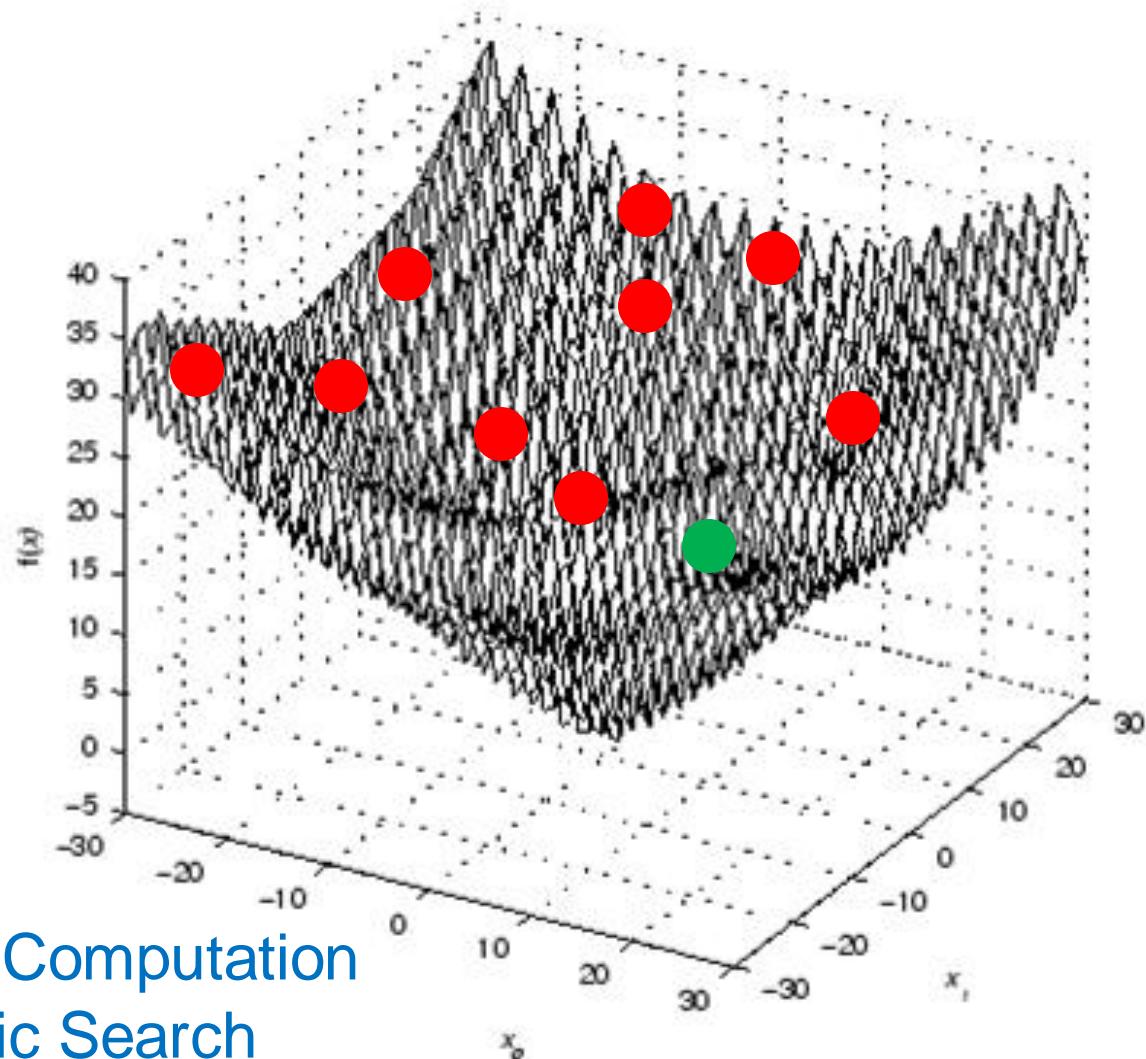
Dr. Suyanto, S.T., M.Sc.
HP/WA: 0812 845 12345

Intelligence Computing Multimedia (ICM)
Informatics faculty – Telkom University

Simple GA

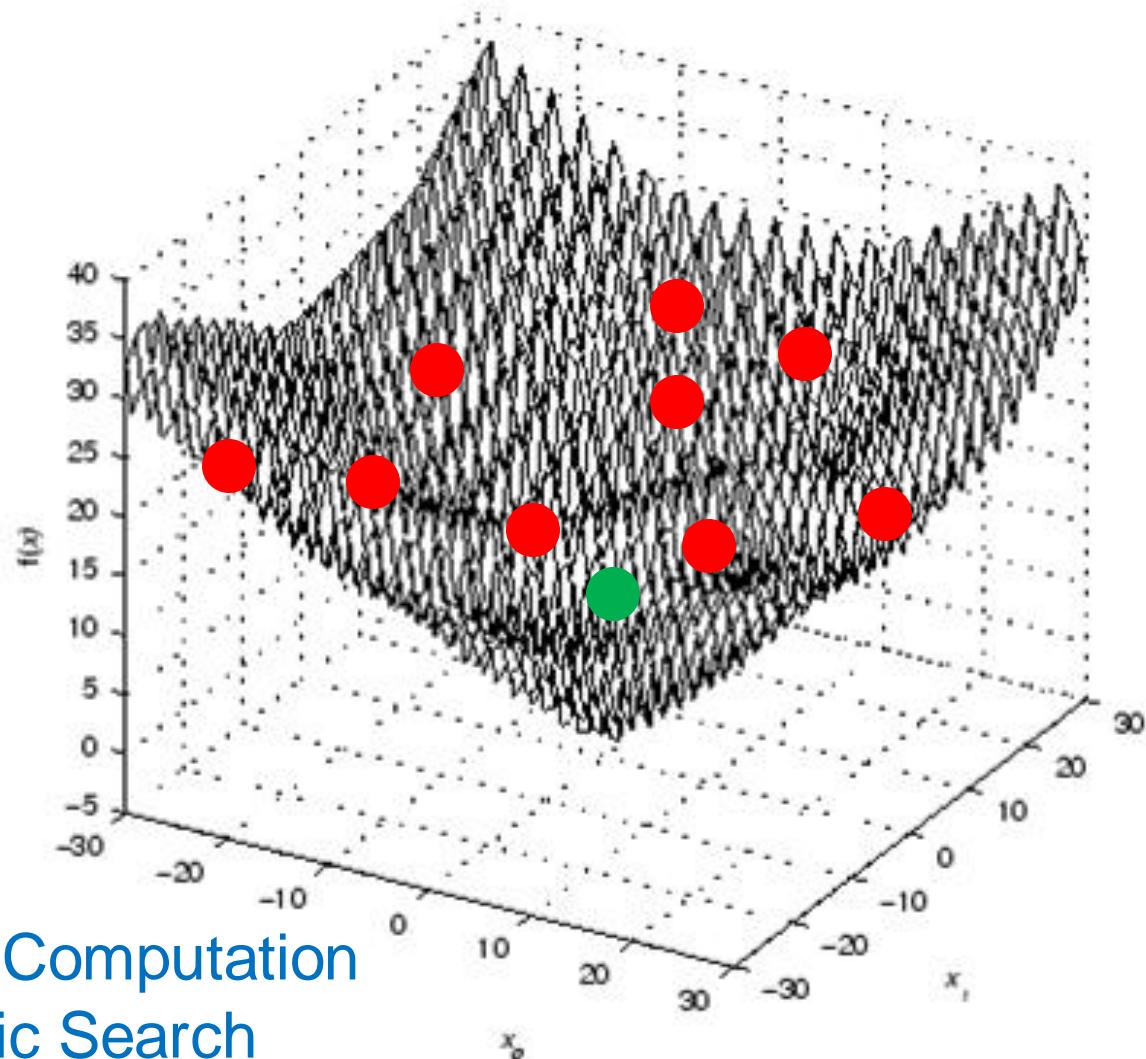
Representasi	Pengkodean biner (<i>binary encoding</i>)
Seleksi orangtua	Proporsional terhadap nilai <i>fitness</i>
Rekombinasi	<i>N-point</i> atau seragam (<i>uniform</i>)
Mutasi	Pembalikan bit dengan probabilitas tetap dan bersifat bebas (<i>independent</i>) pada masing-masing bit.
Seleksi survivor	Semua individu baru menggantikan semua individu lama (<i>generational replacement</i>)
Ciri khusus	Lebih menekankan pada rekombinasi

$$f(\vec{x}) = \sum_{i=0}^{D-1} \left(e^{-0.2\sqrt{x_i^2 + x_{i+1}^2}} + 3(\cos(2x_i) + \sin(2x_{i+1})) \right)$$



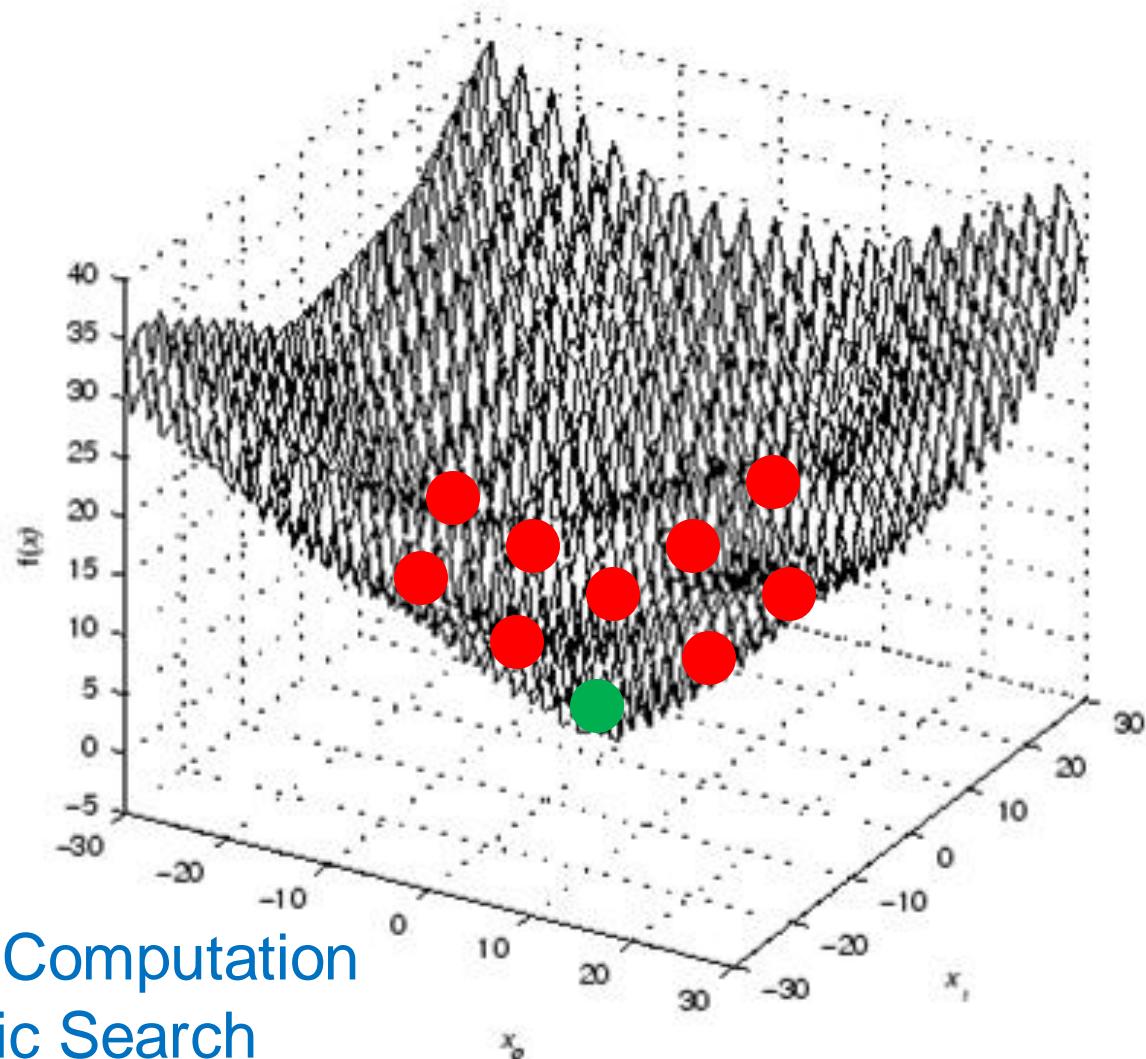
Evolutionary Computation
Meta Heuristic Search

$$f(\vec{x}) = \sum_{i=0}^{D-1} \left(e^{-0.2\sqrt{x_i^2 + x_{i+1}^2}} + 3(\cos(2x_i) + \sin(2x_{i+1})) \right)$$



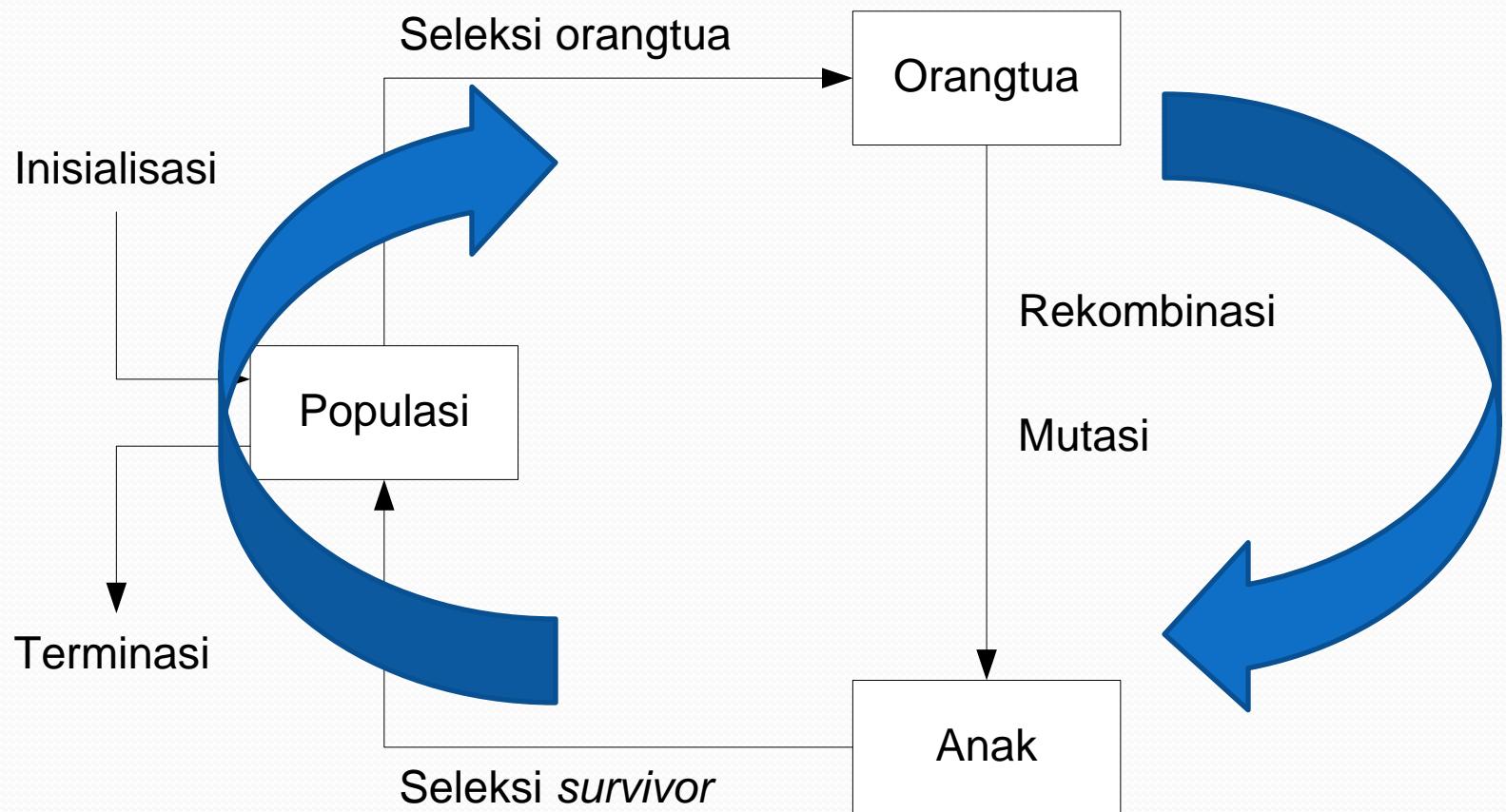
Evolutionary Computation
Meta Heuristic Search

$$f(\vec{x}) = \sum_{i=0}^{D-1} \left(e^{-0.2\sqrt{x_i^2 + x_{i+1}^2}} + 3(\cos(2x_i) + \sin(2x_{i+1})) \right)$$



Evolutionary Computation
Meta Heuristic Search

Skema umum EAs



Bekerja dengan GA

- Buat skema pengkodean Individu → Kromosom
- Bangun fungsi fitness
- Definisikan operator GA

Studi kasus: Minimasi fungsi

Nilai minimum $h = ?$

$$h(x_1, x_2) = x_1^2 + x_2^2$$

$$x_1, x_2 \in [-5,12; 5,12]$$

Individu

x ₁										x ₂									
1	1	0	1	0	1	0	1	0	0	0	1	0	0	1	1	0	1	0	1
g_1										g_{10}	g_{11}							g_{20}	

Fitness

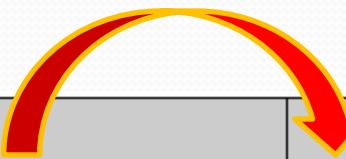
$$f = \frac{1}{(x_1^2 + x_2^2) + 0,01}$$

Jika nilai minimum = 0, nilai maks $f = ?$

Generasi 1

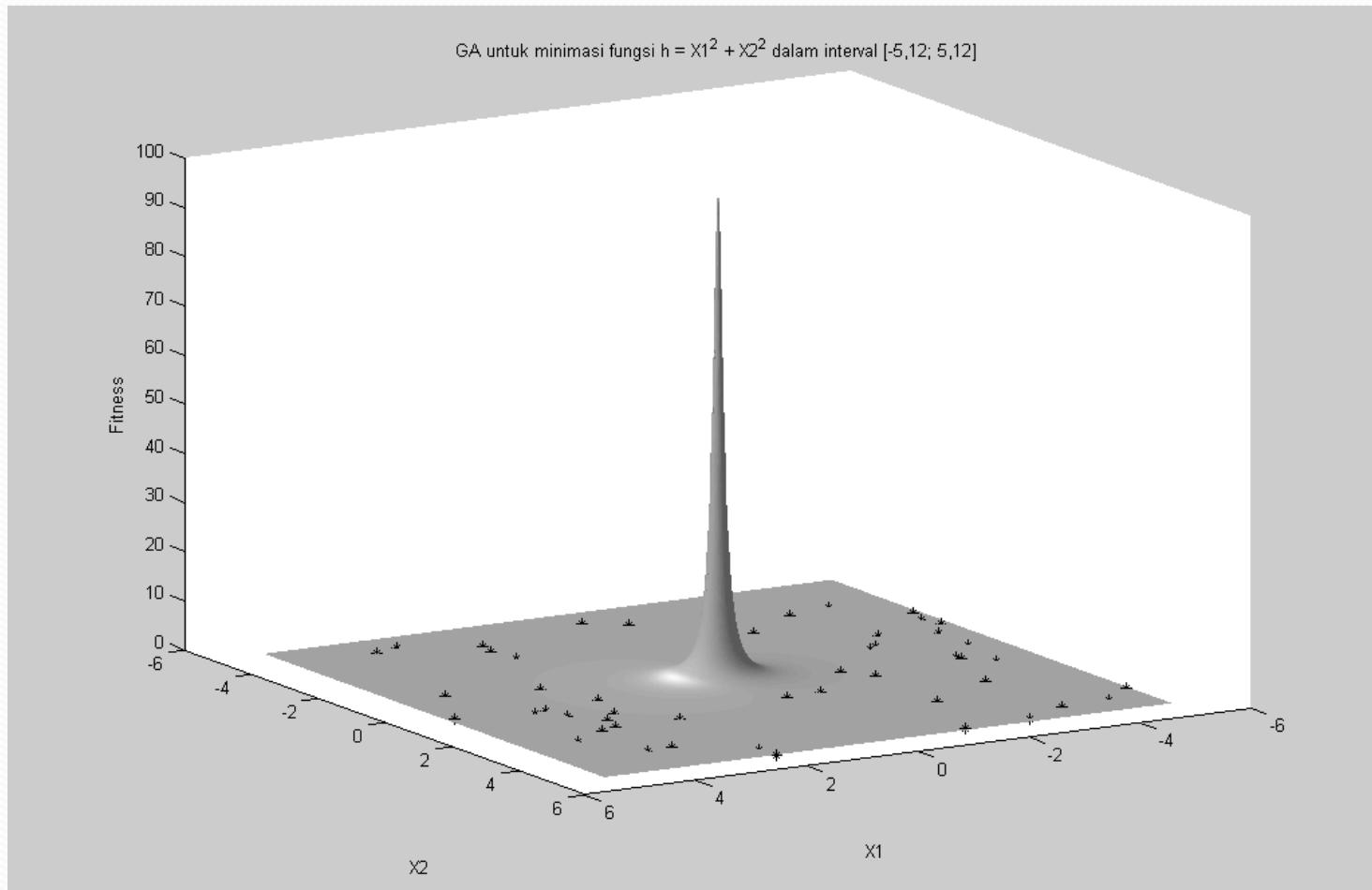
$$x = r_b + \frac{(r_a - r_b)}{\sum_{i=1}^N 2^{-i}} (g_1 \cdot 2^{-1} + g_2 \cdot 2^{-2} + \dots + g_N \cdot 2^{-N})$$

$$x_1, x_2 \in [-5,12; 5,12]$$



No	Genotype	Phenotype		Nilai fitness
	kromosom biner	X1	X2	
1	00010011011001101110	-4.35	1.1	0.049646
2	11001101110001000011	3.11	-4.45	0.033916
3	10110010111111001110	2.03	4.62	0.039254
4	11001110001101111101	3.12	3.81	0.041219
5	11001110101011011001	3.14	2.17	0.068594
6	00101110000110110110	-3.28	-0.74	0.08837
7	01111011111010110010	-0.17	1.78	0.31179
50	11010110011000111011	3.45	0.59	0.081562

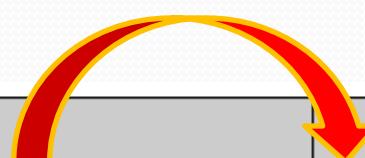
Generasi 1



Generasi 10

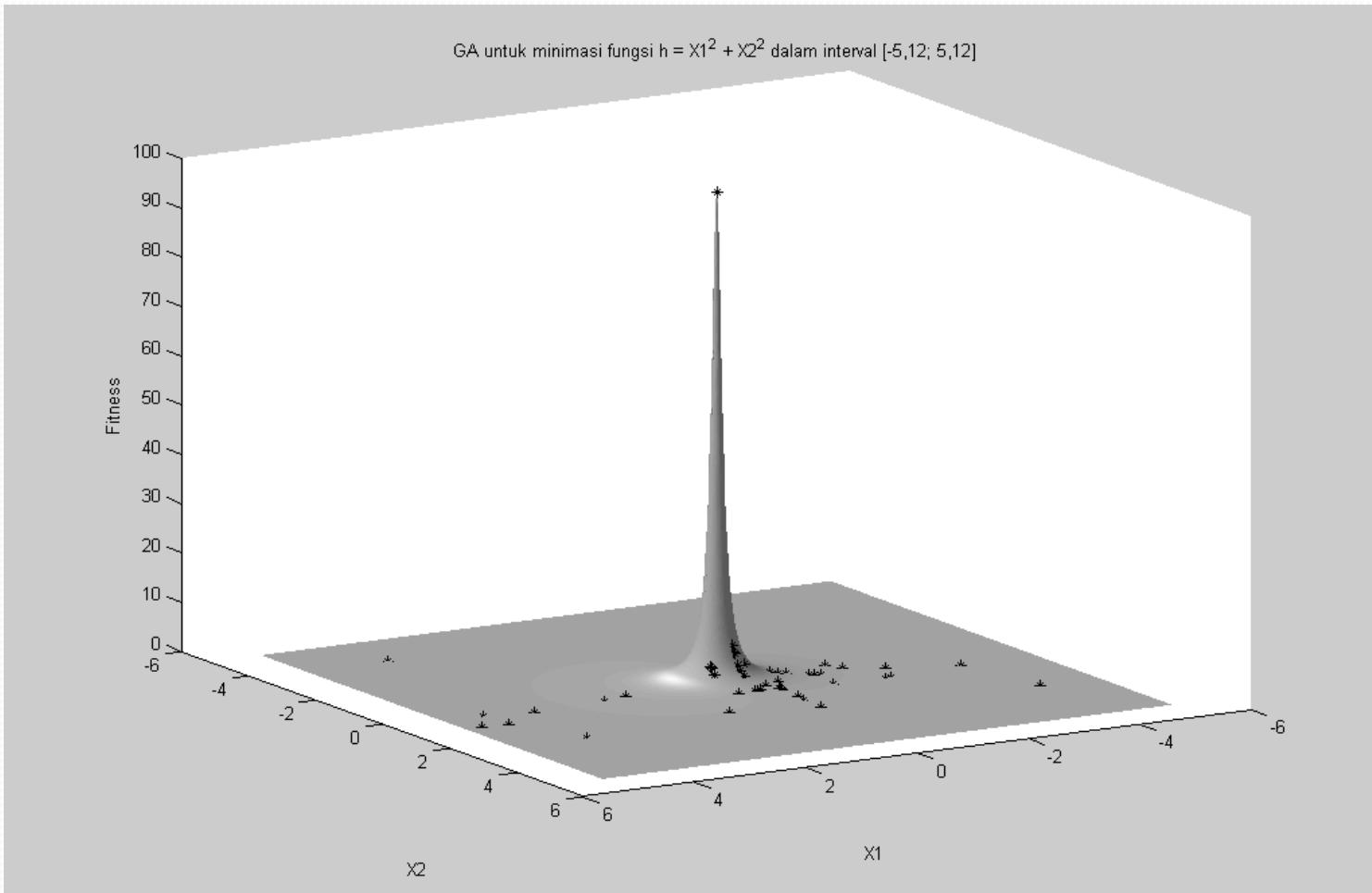
$$x = r_b + \frac{(r_a - r_b)}{\sum_{i=1}^N 2^{-i}} (g_1 \cdot 2^{-1} + g_2 \cdot 2^{-2} + \dots + g_N \cdot 2^{-N})$$

$$x_1, x_2 \in [-5,12; 5,12]$$

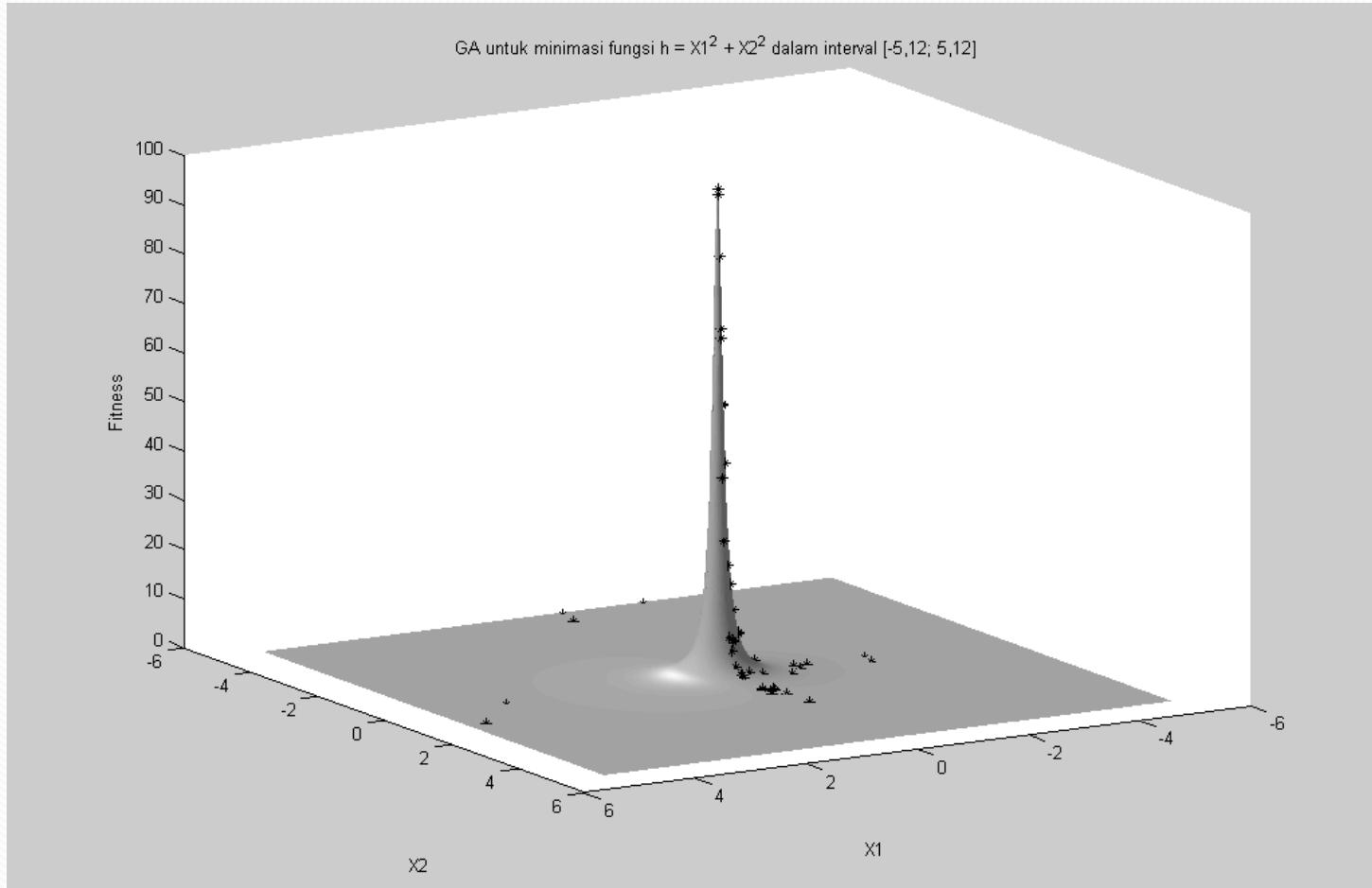


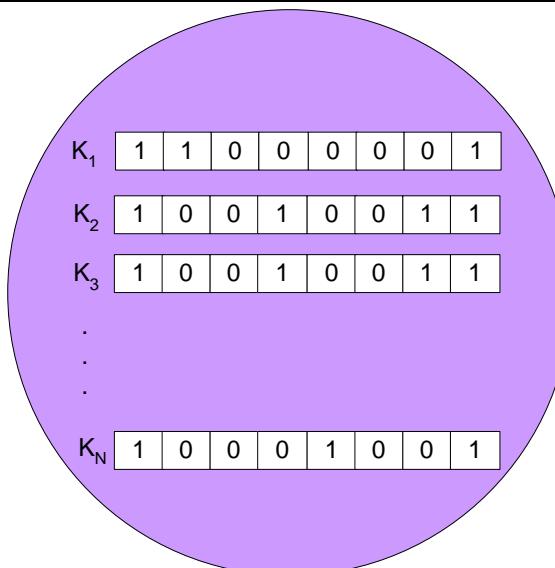
No	Genotype	Phenotype		Nilai fitness
	kromosom biner	X1	X2	
1	01111111110000000000	-0.01	0	99.01
2	01111111110000000000	-0.01	0	99.01
3	01111101010001000001	-3.77	1.03	0.065429
4	01111101011001110001	-2.4	1.2	0.1387
5	01111001111000100001	3.58	0.52	0.076355
6	01011101111000101010	4.83	1.01	0.041053
7	01111101111000100001	-1.38	1.2	0.29812
...				
50	01111101111000000001	-1.93	0.02	0.26772

Generasi 10

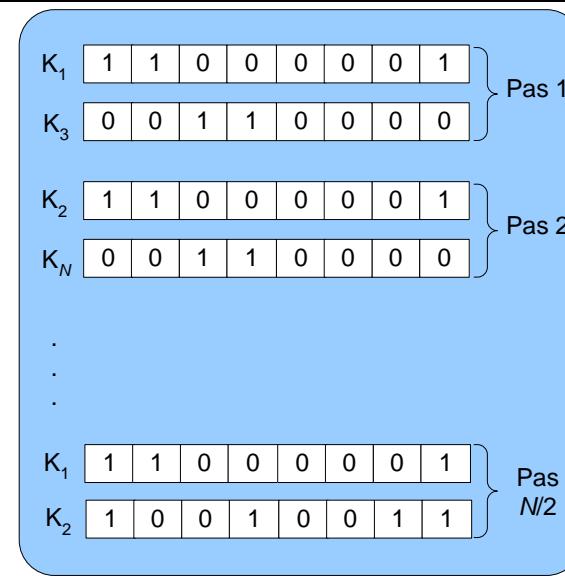
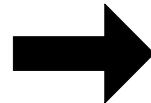


Generasi 100



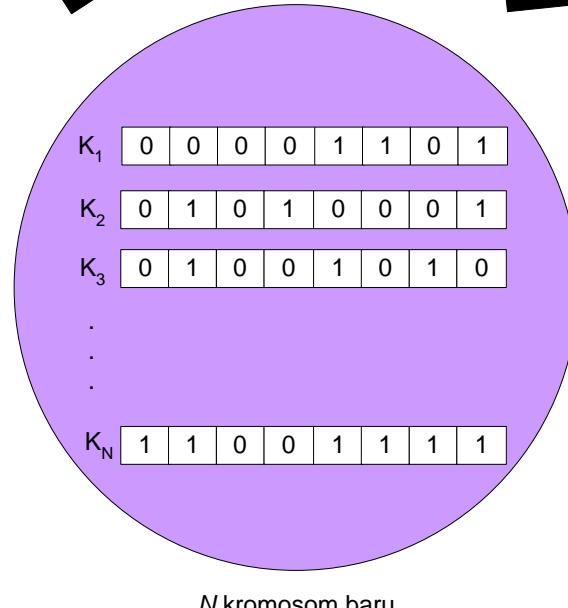


Seleksi orangtua



Penggantian N kromosom

Rekombinasi dan Mutasi



Seleksi Survivor: *Generational*

A photograph of a large, multi-tiered fountain in a city square. The fountain is active, with numerous jets of water shooting upwards from the base. In the center of the fountain is a bronze-colored statue depicting three figures, possibly a family or a group of people, in a dynamic, celebratory pose. The background features several modern skyscrapers and buildings under a blue sky with scattered white clouds.

Demo: Simple GA

Studi kasus: Minimasi fungsi

Nilai minimum $h = ?$

$$h(x_1, x_2) = x_1^2 + x_2^2$$

$$x_1, x_2 \in [-5,12;5,12]$$

Komponen GA

- Inisialisasi Populasi (N kromosom)
- Evaluasi Individu (berbasis fungsi fitness)
- Seleksi Ortu
- Rekombinasi
- Mutasi
- Seleksi Survivor

Implementasinya dalam Matlab?

Representasi Biner

function Populasi = InisialisasiPopulasiBiner(UkPop,JumGen)

Populasi = fix(2*rand(UkPop,JumGen));

Misal: UkPop = 5; JumGen = 10;

1	0	1	1	0	0	1	1	1	0
1	0	1	1	0	0	0	1	0	0
1	0	0	0	1	0	0	1	1	0
1	0	1	0	1	1	0	0	1	1
1	1	1	1	0	0	0	0	1	1

```
function x= DekodeKromosomBiner(Kromosom,Nvar,Ng,Ra,Rb)
```

```
MaxSum = 0;
```

```
for kk=1:Ng,
```

```
    MaxSum = MaxSum + 2^(-kk);
```

```
end
```

```
for ii=1:Nvar,
```

```
    x(ii) = 0;
```

```
for jj=1:Ng,
```

```
    x(ii) = x(ii) + Kromosom((ii-1)*Ng+jj)*2^(-jj);
```

```
end
```

```
    x(ii) = Rb + ((Ra-Rb)/MaxSum) * x(ii);
```

```
end
```

Individu, misal
X1 dan x2

kode genetik,
misal biner

Fitness

$$f = \frac{1}{(x_1^2 + x_2^2) + 0,01}$$

Jika nilai minimum = 0, nilai maks $f = ?$

Evaluasi Individu

```
function fitness = EvaluasiIndividu(x,BilKecil)
```

```
fitness = 1 / (x(1)^2 + x(2)^2 + BilKecil);
```

```
function Pindex = RouletteWheel(UkPop,LinearFitness)
```

```
JumFitness = sum(LinearFitness);
```

```
KumulatifFitness = 0;
```

```
RN = rand;
```

```
ii = 1;
```

```
while ii <= UkPop,
```

```
    KumulatifFitness = KumulatifFitness + LinearFitness(ii);
```

```
    if (KumulatifFitness/JumFitness) > RN,
```

```
        Pindex = ii;
```

```
        break;
```

```
    end
```

```
    ii = ii + 1;
```

```
end
```

Rekombinasi Biner

```
function Anak = RekombinasiBiner(Ortu1,Ortu2,JumGen)
```

```
TP = 1 + fix(rand*(JumGen-1));
```

```
Anak(1,:) = [Ortu1(1:TP) Ortu2(TP+1:JumGen)];
```

```
Anak(2,:) = [Ortu2(1:TP) Ortu1(TP+1:JumGen)];
```

Mutasi Biner

```
function MutKrom = MutasiBiner(Kromosom,JumGen,Pmutasi)
```

```
MutKrom = Kromosom;
```

```
for ii=1:JumGen,
```

```
    if (rand < Pmutasi),
```

```
        if Kromosom(ii)==0,
```

```
            MutKrom(ii) = 1;
```

```
        else
```

```
            MutKrom(ii) = 0;
```

```
        end
```

```
    end
```

```
end
```

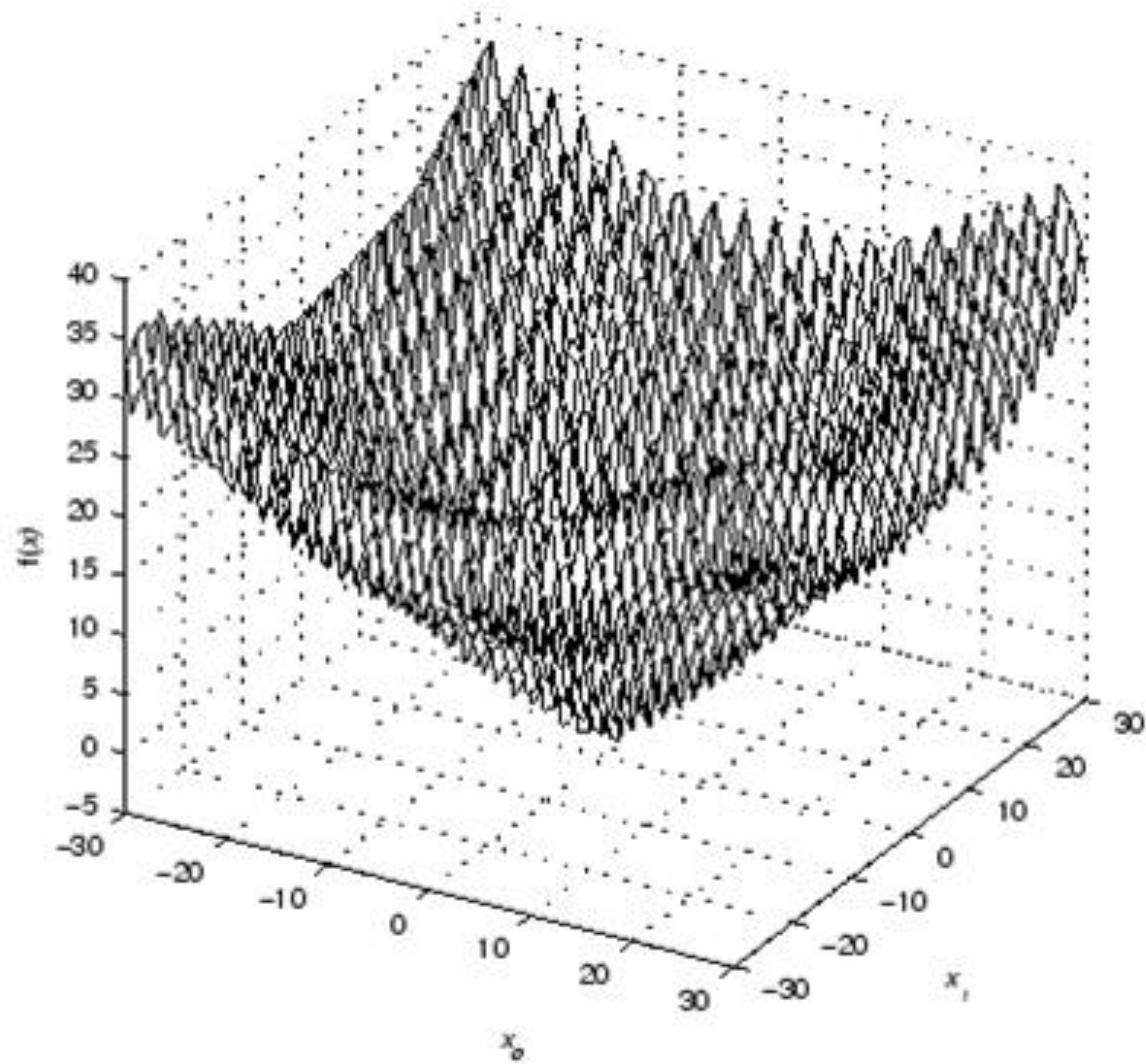
A photograph of a large, multi-tiered fountain in a city square. The fountain is active, with numerous jets of water shooting upwards from the base. In the center of the fountain is a bronze statue depicting three figures, possibly a family or a group of people, in a dynamic, celebratory pose. The background features several modern skyscrapers and buildings under a blue sky with scattered white clouds.

Demo: Simple GA



Question?

$$f(\vec{x}) = \sum_{i=0}^{D-1} \left(e^{-0.2\sqrt{x_i^2 + x_{i+1}^2}} + 3(\cos(2x_i) + \sin(2x_{i+1})) \right)$$



Representasi individu → kromosom

- Representasi Biner
- Representasi Integer
- Representasi Real
- Representasi Permutasi

Representasi Biner

function Populasi = InisialisasiPopulasiBiner(UkPop,JumGen)

Populasi = fix(2*rand(UkPop,JumGen));

Misal: UkPop = 5; JumGen = 10;

1	0	1	1	0	0	1	1	1	0
1	0	1	1	0	0	0	1	0	0
1	0	0	0	1	0	0	1	1	0
1	0	1	0	1	1	0	0	1	1
1	1	1	1	0	0	0	0	1	1

Representasi Integer

function Populasi = InisialisasiPopulasInteger(UkPop,JumGen)

Populasi = fix(9*rand(UkPop,JumGen));

Misal: UkPop = 5; JumGen = 10;

4	3	5	7	4	7	6	6	7	0
7	5	3	3	2	6	7	5	1	6
6	5	1	7	8	1	6	5	3	1
2	2	5	0	6	8	3	0	2	8
3	6	5	5	1	3	6	2	6	0

Representasi Real

function Populasi = InisialisasiPopulasiReal(UkPop,JumGen)

Populasi = rand(UkPop,JumGen);

Misal: UkPop = 5; JumGen = 10;

0,405102	0,313349	0,234897	0,84867	0,291679	0,249295	0,314046	0,305683	0,726449	0,61309
0,779813	0,985409	0,605045	0,486053	0,068964	0,196639	0,851981	0,294056	0,585907	0,622961
0,449634	0,197545	0,415529	0,998973	0,177705	0,467867	0,523253	0,481516	0,330538	0,225416
0,332314	0,339029	0,709216	0,031305	0,118845	0,252562	0,546119	0,499327	0,859106	0,752103
0,433246	0,134945	0,926929	0,466493	0,056166	0,188175	0,896785	0,665138	0,588188	0,435401

Representasi Permutasi

```
function Populasi = TSPInisialisasiPopulasi(UkPop,JumGen)
for ii=1:UkPop,
    [Xval,Ind] = sort(rand(1,JumGen));
    Populasi(ii,:) = Ind;
end
```

Misal: UkPop = 5; JumGen = 10;

10	2	9	3	6	4	5	8	1	7
1	5	6	8	4	9	7	3	2	10
6	10	3	4	2	8	1	7	5	9
9	5	8	3	6	2	7	10	1	4
2	1	8	10	4	5	6	9	3	7

```
function x= DekodeKromosomBiner(Kromosom,Nvar,Ng,Ra,Rb)
```

```
MaxSum = 0;
```

```
for kk=1:Ng,
```

```
    MaxSum = MaxSum + 2^(-kk);
```

```
end
```

```
for ii=1:Nvar,
```

```
    x(ii) = 0;
```

```
for jj=1:Ng,
```

```
    x(ii) = x(ii) + Kromosom((ii-1)*Ng+jj)*2^(-jj);
```

```
end
```

```
    x(ii) = Rb + ((Ra-Rb)/MaxSum) * x(ii);
```

```
end
```

Individu, misal
X1 dan x2

kode genetik,
misal biner

```
function x = DekodeKromosomInteger(Kromosom,Nvar,Ng,Ra,Rb)

MaxSum = 0;
for kk=1:Ng,
    MaxSum = MaxSum + (9 * 10^(-kk));
end
for ii=1:Nvar,
    x(ii) = 0;
    for jj=1:Ng,
        x(ii) = x(ii) + Kromosom((ii-1)*Ng+jj)*10^(-jj);
    end
    x(ii) = Rb + ((Ra-Rb)/MaxSum) * x(ii);
end
```

```
function x = DekodeKromosomReal(Kromosom,Nvar,Ng,Ra,Rb)  
  
for ii=1:Nvar,  
    x(ii) = Rb + ((Ra - Rb) * Kromosom(ii));  
end
```

Fitness

$$f = \frac{1}{(x_1^2 + x_2^2) + 0,01}$$

Jika nilai minimum = 0, nilai maks $f = ?$

Evaluasi Individu

```
function fitness = EvaluasiIndividu(x,BilKecil)
```

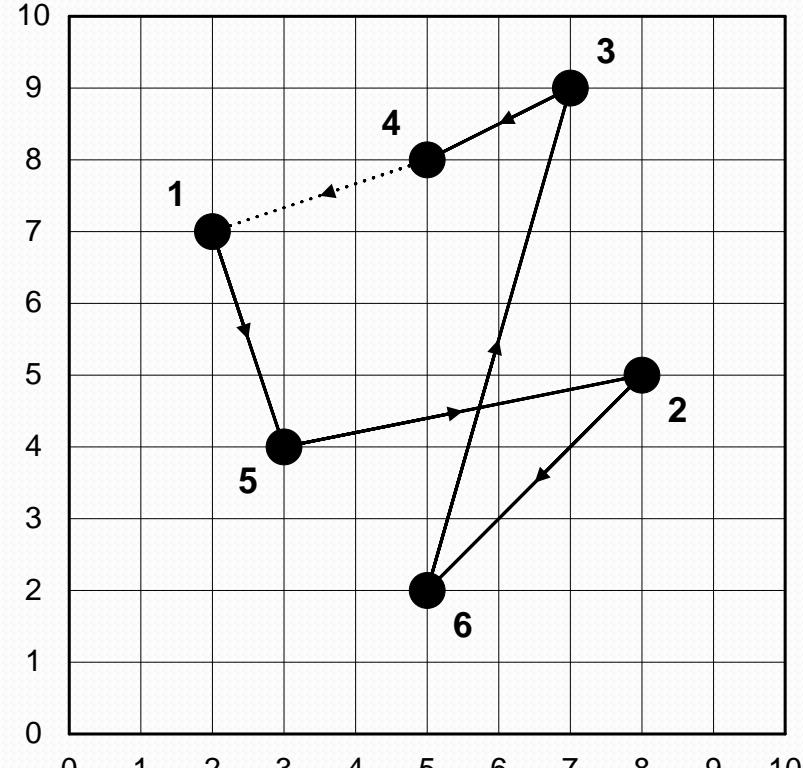
```
fitness = 1 / (x(1)^2 + x(2)^2 + BilKecil);
```

Kasus TSP

Fitness?

Total biaya kunjungan

Individu: urutan kunjungan semua lokasi



Nilai gen menyatakan nomor lokasi

Posisi gen menyatakan urutan kunjungan

```
function fitness = TSPEvaluasilIndividu(Kromosom,JumGen,XYkota)
```

```
TB = 0;
```

```
for ii=1:JumGen-1,
```

```
    TB = TB + norm(XYkota(Kromosom(ii),:) -  
        XYkota(Kromosom(ii+1),:));
```

```
end
```

```
% Jalur harus kembali ke kota asal
```

```
TB = TB + norm(XYkota(Kromosom(JumGen),:) -  
    XYkota(Kromosom(1),:));
```

```
fitness = 1 / TB;
```

```
function Pindex = RouletteWheel(UkPop,LinearFitness)
```

```
JumFitness = sum(LinearFitness);
```

```
KumulatifFitness = 0;
```

```
RN = rand;
```

```
ii = 1;
```

```
while ii <= UkPop,
```

```
    KumulatifFitness = KumulatifFitness + LinearFitness(ii);
```

```
    if (KumulatifFitness/JumFitness) > RN,
```

```
        Pindex = ii;
```

```
        break;
```

```
    end
```

```
    ii = ii + 1;
```

```
end
```

```

function IndTerpilih = TournamentSelection(UkPop, Fitness, UkTour, ProbTour)

for ii=1:UkTour, % pilih kontestan sebanyak UkTour
    IndTemp(ii) = 1 + fix(rand*UkPop); FitTemp(ii) = Fitness(IndTemp(ii));
end
[M, IndTerbaik] = max(FitTemp); IndLainnya = []; % cari individu terbaik
if IndTerbaik==1,
    IndLainnya = [IndLainnya 2:UkTour];
else
    IndLainnya = [IndLainnya 1:IndTerbaik-1];
    IndLainnya = [IndLainnya IndTerbaik+1:UkTour];
end
if rand < ProbTour, % Individu terbaik terpilih jika memenuhi ProbTour
    IndTerpilih = IndTemp(IndTerbaik);
else
    IndTerpilih = IndTemp(IndLainnya(1+fix(rand*(UkTour-1))));
```

end

Rekombinasi Biner/Integer

```
function Anak = RekombinasiBiner(Ortu1,Ortu2,JumGen)

TP = 1 + fix(rand*(JumGen-1));
Anak(1,:) = [Ortu1(1:TP) Ortu2(TP+1:JumGen)];
Anak(2,:) = [Ortu2(1:TP) Ortu1(TP+1:JumGen)];
```

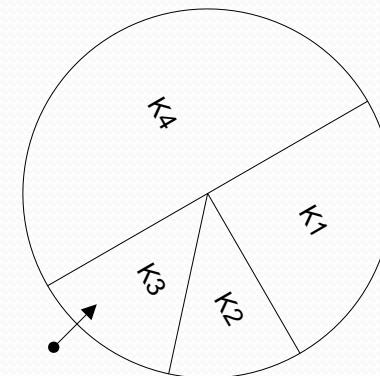
Seleksi Orangtua

- Fitness Proportionate Selection (FPS)
 - *Roulette wheel*
 - **Baker's SUS** (*Stochastic Universal Sampling*)
- Rank-Based Selection
 - Linear Ranking
 - Non-linear ranking
- Tournament Selection

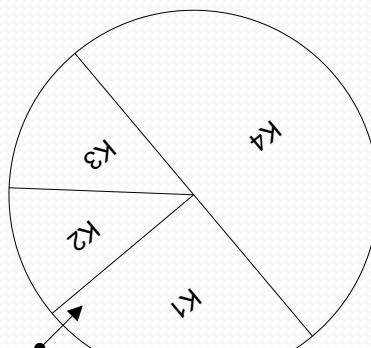
Fitness Proportionate Selection (FPS)

Metode: *roulette wheel*

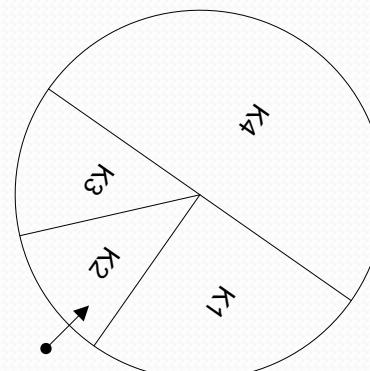
Kromosom	Fitness
K1	2
K2	1
K3	1
K4	4
Jumlah	8



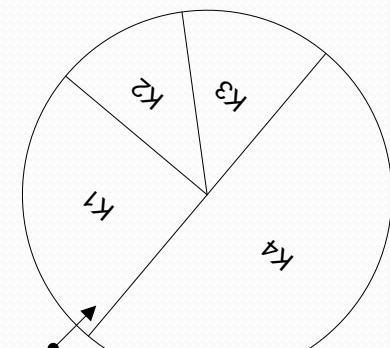
Putaran ke-1



Putaran ke-2



Putaran ke-3

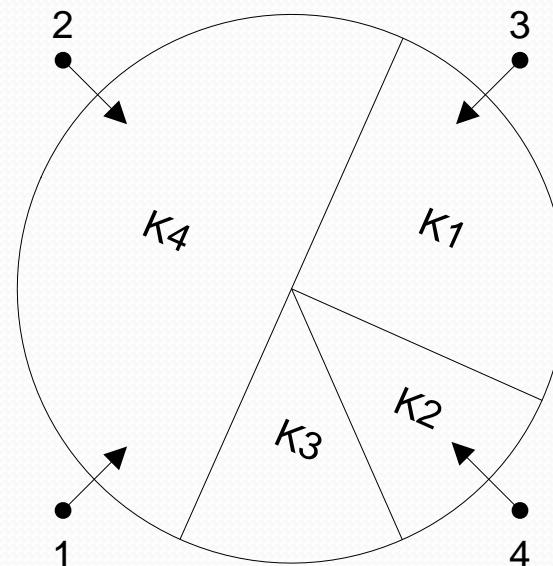


Putaran ke-4

Fitness Proportionate Selection (FPS)

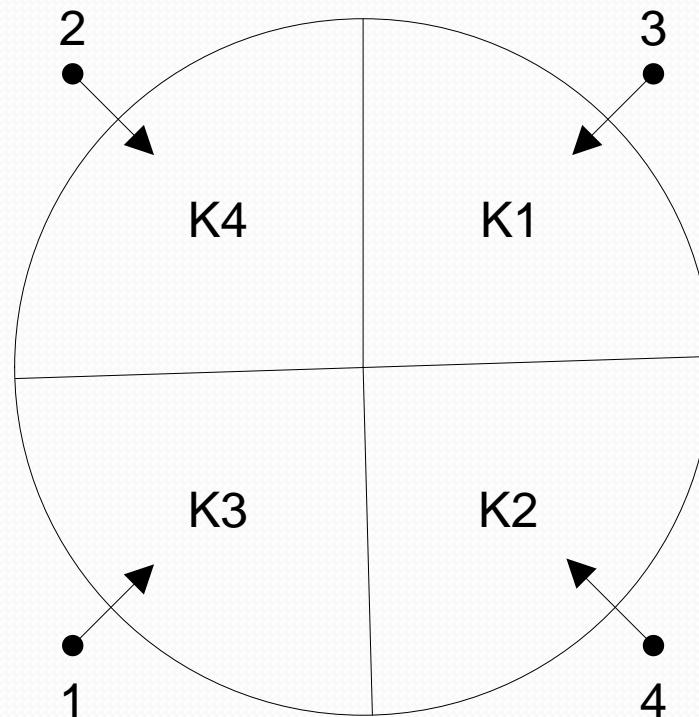
Metode: **Baker's SUS** (*Stochastic Universal Sampling*)

Kromosom	<i>Fitness</i>
K1	2
K2	1
K3	1
K4	4
Jumlah	8



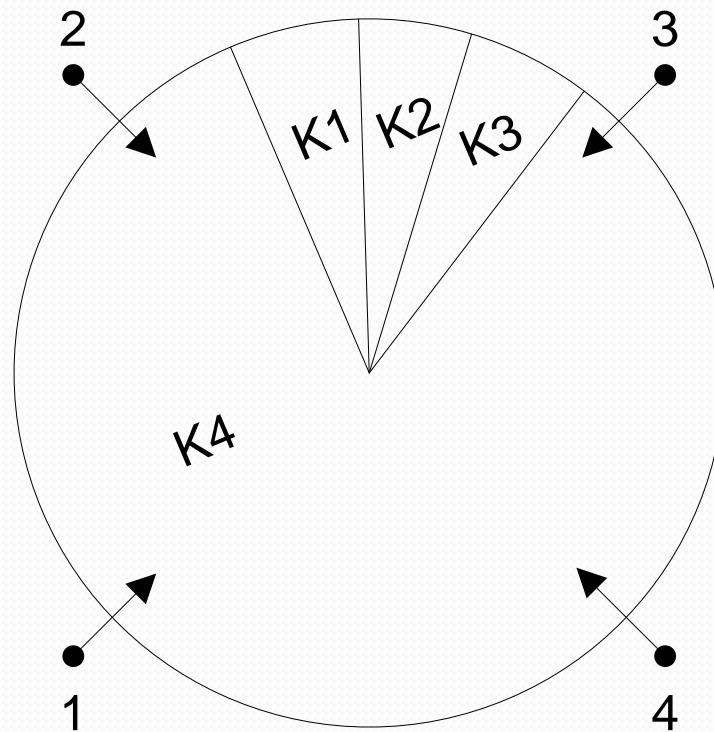
Kelemahan Baker's SUS

Kromosom	<i>Fitness</i>
K1	1,98
K2	2,01
K3	1,99
K4	2,02
Jumlah	8



Kelemahan Baker's SUS

Kromosom	<i>Fitness</i>
K1	1
K2	1
K3	1
K4	17
Jumlah	20



Untuk mengatasi Baker's SUS

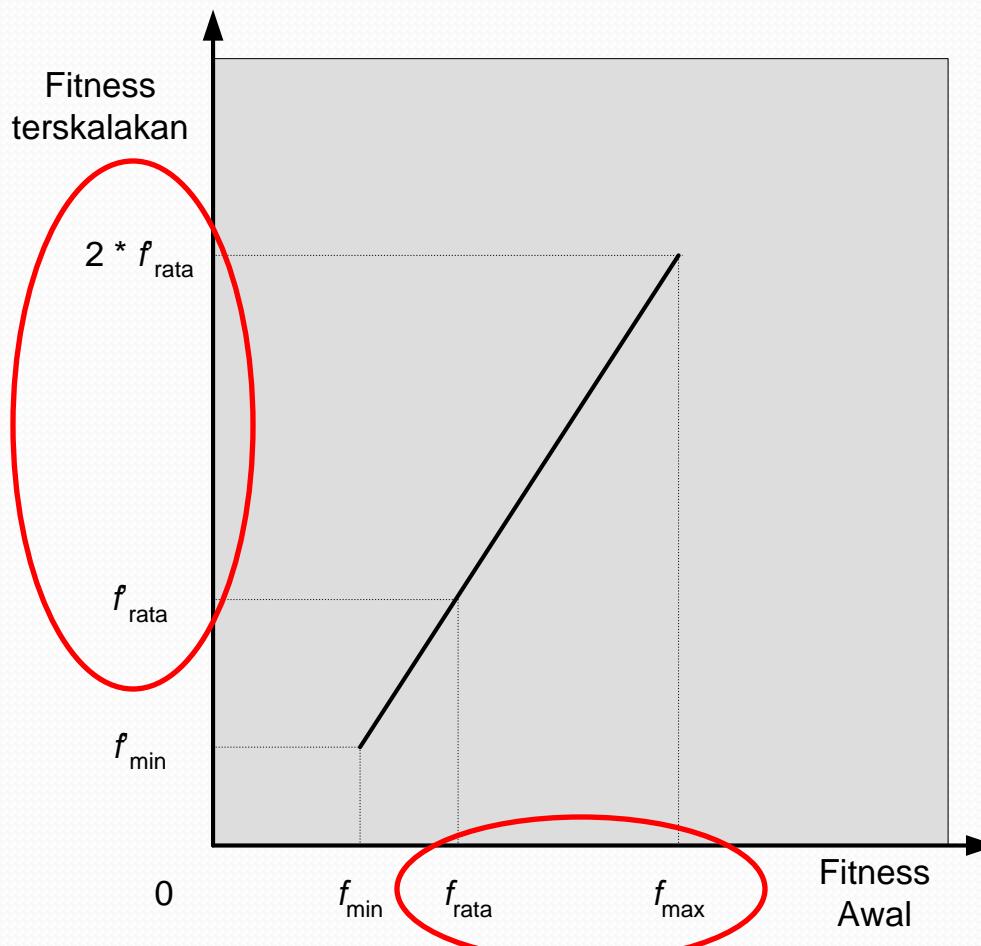
- *Linear Scaling*
- *Window Scaling*
- *Sigma Scaling*

Linear Scaling

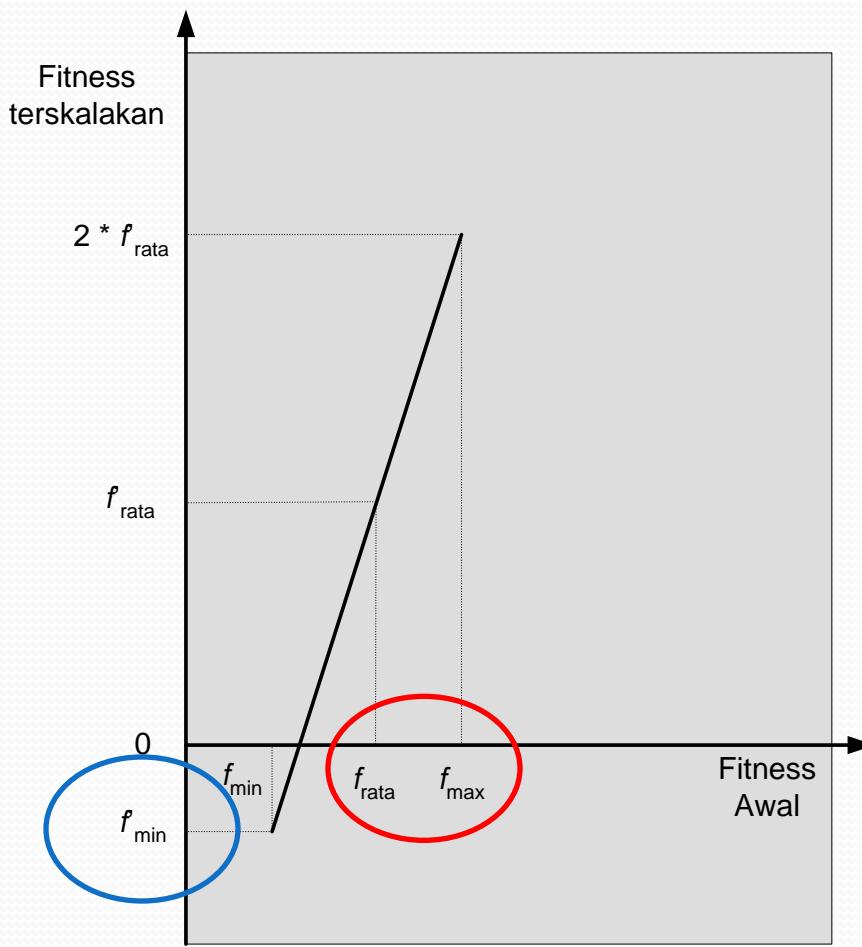
$$f'_i = af_i + b$$

- f_i = nilai *fitness* individu ke-*i*
- a dan b dipilih sedemikian hingga $\bar{f}' = \bar{f}$ dan
- $f'_{\max} = C_{mult} * \bar{f}$, dimana C_{mult} adalah nilai harapan berapa kali kromosom terbaik terpilih sebagai orangtua
- Untuk ukuran populasi sebesar 50 sampai 100 individu, biasanya C_{mult} antara 1,2 sampai 2.

Linear Scaling



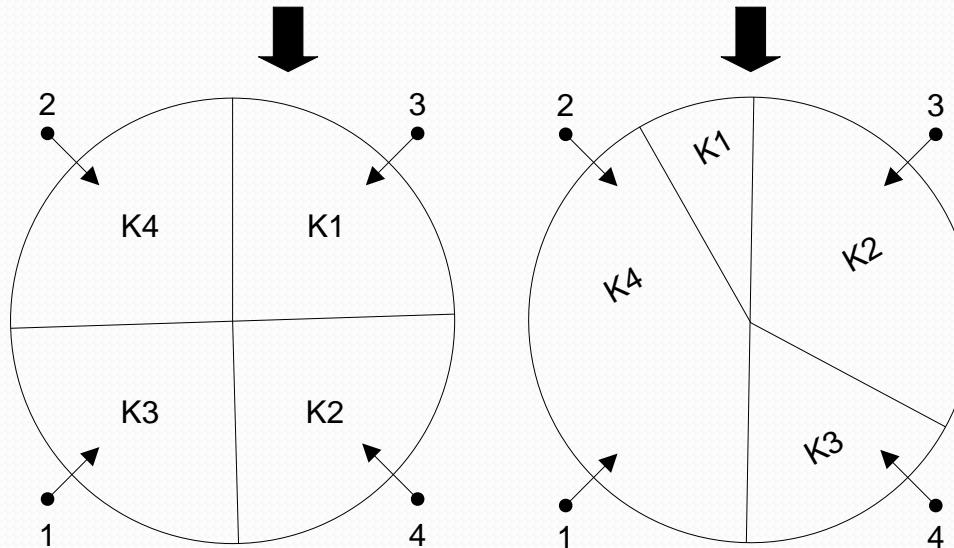
Kelemahan *Linear Scaling*



Window Scaling

Misalkan *fitness* terendah selama 5 generasi terakhir = 1,97

Kromosom	<i>f</i>	<i>f'</i>
K1	1,98	0,01
K2	2,01	0,04
K3	1,99	0,02
K4	2,02	0,05
Jumlah	8	0,12



Sigma Scaling

$$f'_i = f_i + (\bar{f} - c * \sigma)$$

- c = bilangan bulat kecil (biasanya sama dengan 2)
- Nilai-nilai $fitness f$ diskalakan menggunakan rata-rata dan standar deviasi

Rank-Based Selection

- *Linear Ranking*
- *Non-linear ranking*

Linear Ranking (LR)

- Semua individu yang berada dalam populasi diurutkan berdasarkan nilai fitnessnya secara *ascending*
- Nilai *fitness* hasil perankingan dihitung menggunakan rumus:

$$f'(Pos) = (2 - S) + 2(S - 1) \frac{(Pos - 1)}{(N - 1)}$$

- S adalah ***selective pressure*** (probabilitas terpilih individu terbaik dibandingkan dengan rata-rata probabilitas terpilih semua individu). S berada dalam interval $[1, 2]$.
- Pos adalah posisi individu dalam populasi, dimana individu terburuk (nilai *fitness*-nya paling rendah) berada di posisi 1 dan individu terbaik di posisi N .

Non Linear Ranking (NLR)

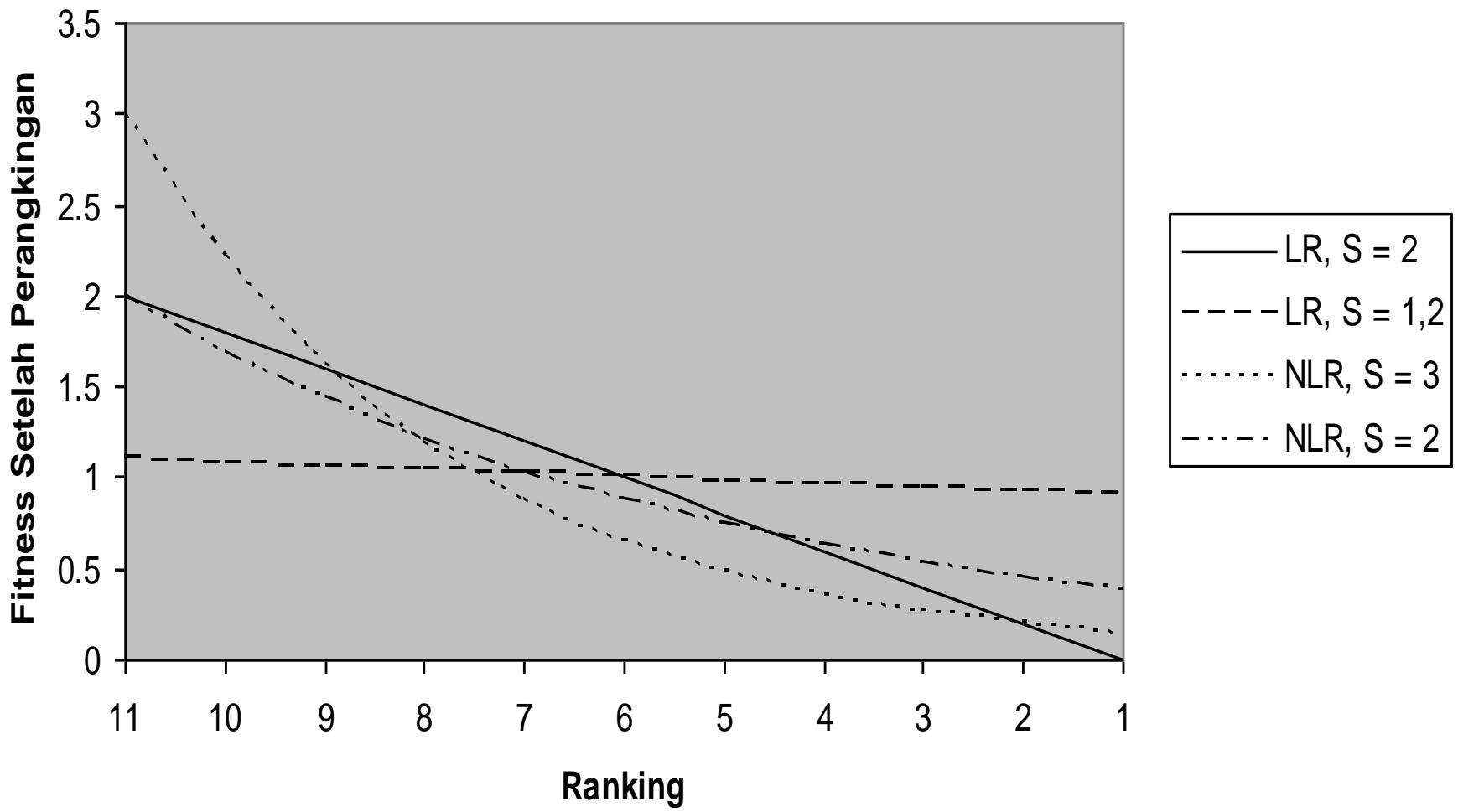
$$f'(Pos) = \frac{NX^{Pos-1}}{\sum_{i=1}^N X^{i-1}}$$

dimana X adalah akar dari polinomial:

$$0 = (S - N)X^{N-1} + SX^{N-2} + \dots + SX + S$$

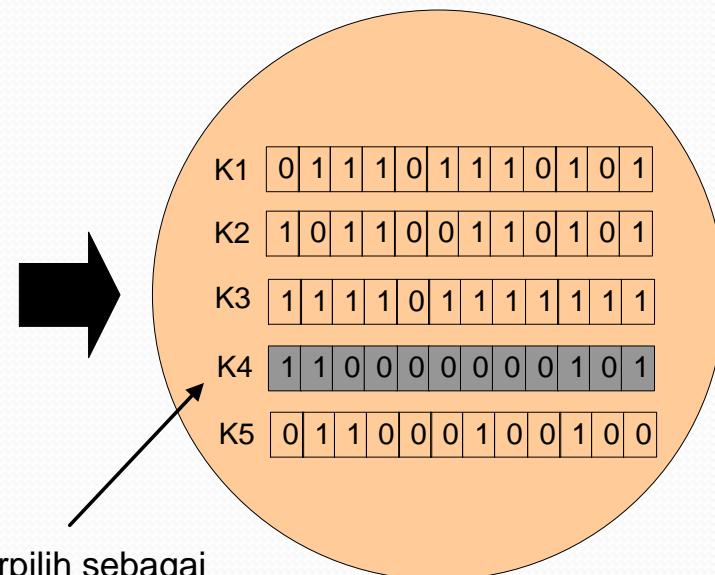
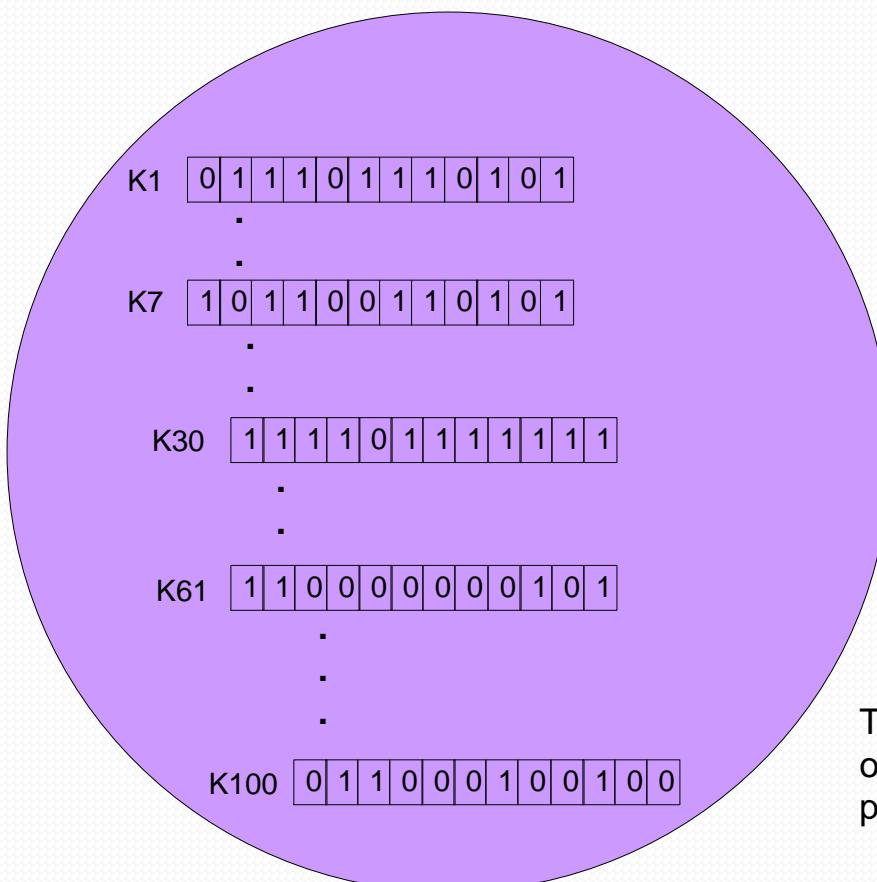
N = ukuran populasi

S = *selective pressure* dalam interval $[1, N-2]$.



<i>Fitness</i>	<i>Posisi</i>	<i>Fitness Perangkingan</i>			
		LR, S = 2	LR, S = 1,2	NLR, S = 3	NLR, S = 2
90,11	11	2	1,1	3	2
90,09	10	1,8	1,08	2,21	1,69
90,08	9	1,6	1,06	1,62	1,43
90,06	8	1,4	1,04	1,19	1,21
90,05	7	1,2	1,02	0,88	1,03
89,97	6	1	1	0,65	0,87
89,96	5	0,8	0,98	0,48	0,74
89,95	4	0,6	0,96	0,35	0,62
79,94	3	0,4	0,94	0,26	0,53
79,93	2	0,2	0,92	0,19	0,45
79,91	1	0	0,9	0,14	0,38

Tournament Selection

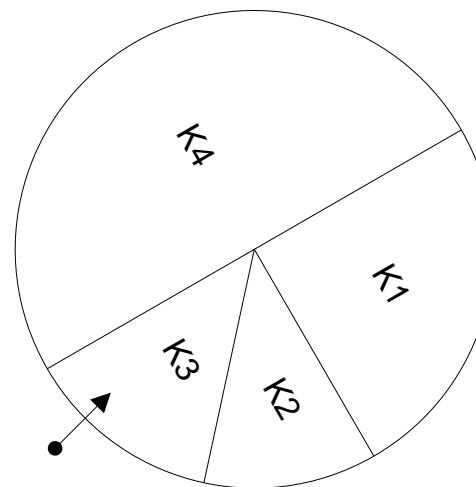


Tournament Selection

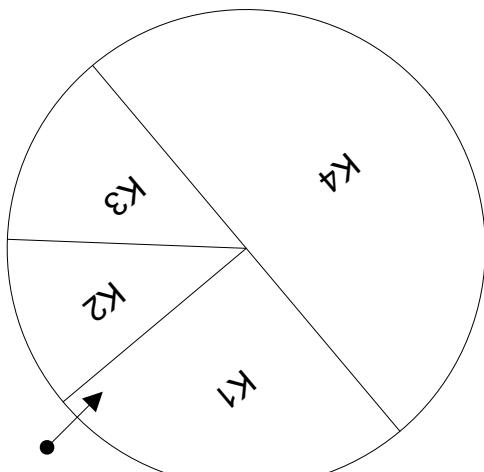
- Banyak cara yang bisa digunakan untuk memilih satu pemenang (orangtua) dari k kontestan tersebut.
- Ada empat hal yang bisa dijadikan acuan untuk membangun prosedur penentuan pemenang turnamen, yaitu:
 - Perlu perankingan atau tidak?
 - Berapa ukuran sampling k ?
 - Apakah kromosom yang sudah pernah terpilih sebagai kontestan bisa terpilih lagi?
 - Apakah kontestan terbaik (dengan *fitness* tertinggi) selalu menjadi pemenang (deterministik) atau bergantung pada suatu probabilistik tertentu?

Metode: roulette wheel

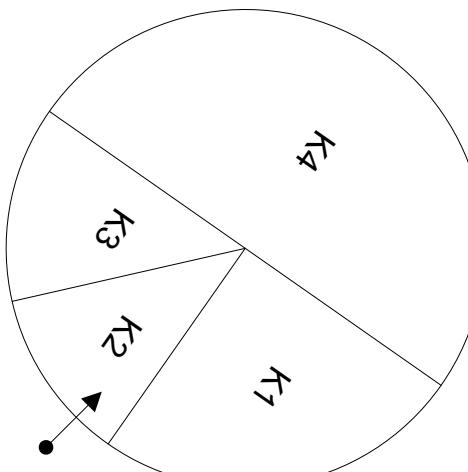
Kromosom	Fitness
K1	2
K2	1
K3	1
K4	4
Jumlah	8



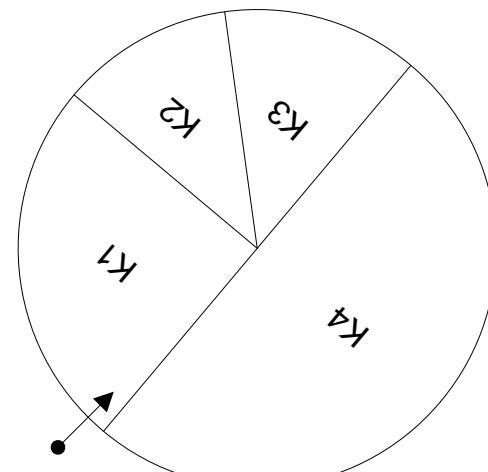
Putaran ke-1



Putaran ke-2



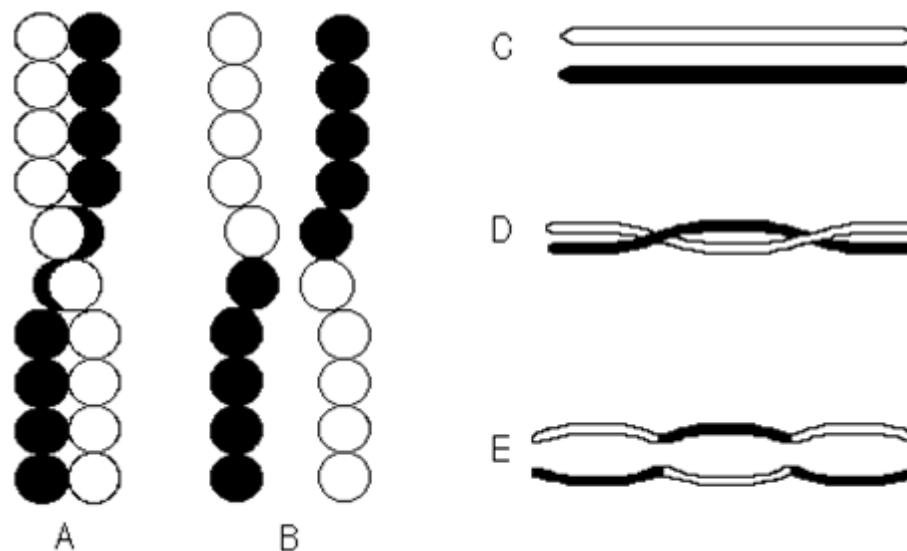
Putaran ke-3



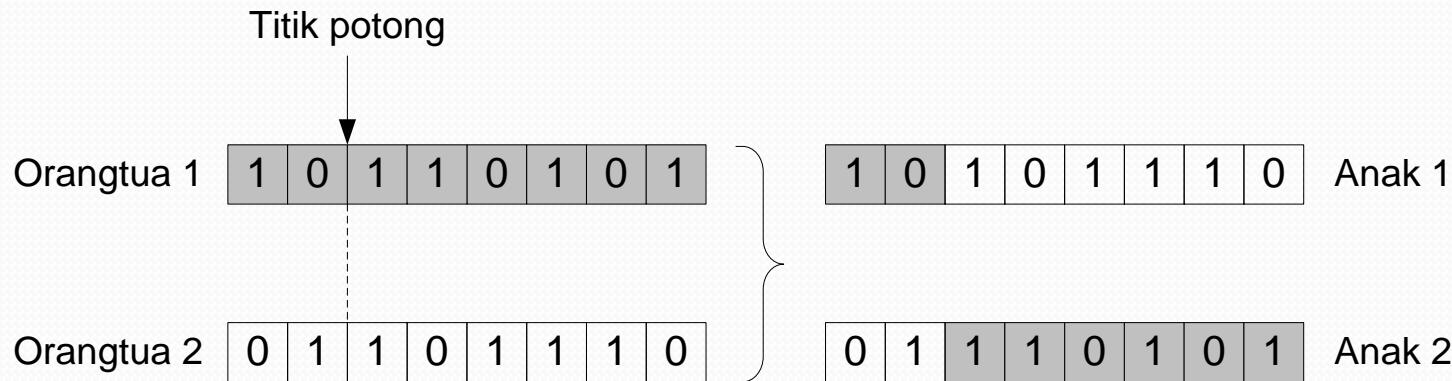
Putaran ke-4

Rekombinasi

Operator rekombinasi di GA menirukan apa yang terjadi di dunia nyata.



Rekombinasi



Rekombinasi

- Pada GA berjenis *generational replacement*, setelah mendapatkan N (ukuran populasi) kromosom di *mating pool* sebagai orangtua, GA menjalankan operator rekombinasi (atau *crossover/pindah silang*) terhadap pasangan orangtua berdasarkan probabilitas tertentu.
- Banyak metode rekombinasi yang telah diusulkan.
- Masing-masing metode memiliki ciri khusus dan mungkin saja hanya bisa digunakan pada jenis representasi tertentu. Misalnya, rekombinasi untuk representasi permutasi bersifat khusus untuk masalah permutasi.

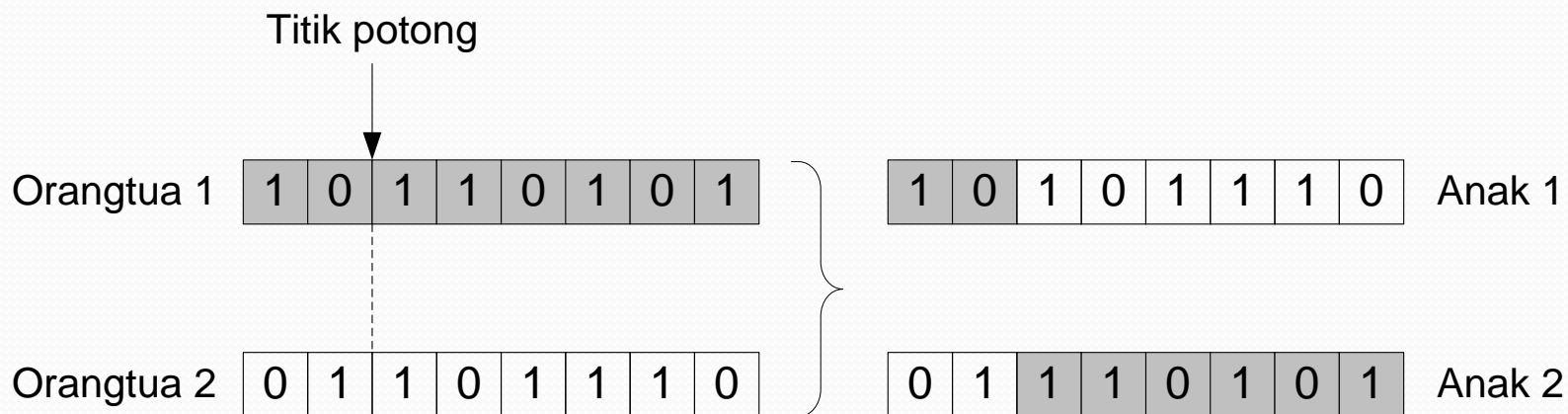
Rekombinasi

- Rekombinasi untuk representasi Biner
- Rekombinasi untuk representasi Integer
- Rekombinasi untuk representasi Real
- Rekombinasi untuk representasi Permutasi
- Rekombinasi *Path Relinking*
- Rekombinasi *Multi-parent*

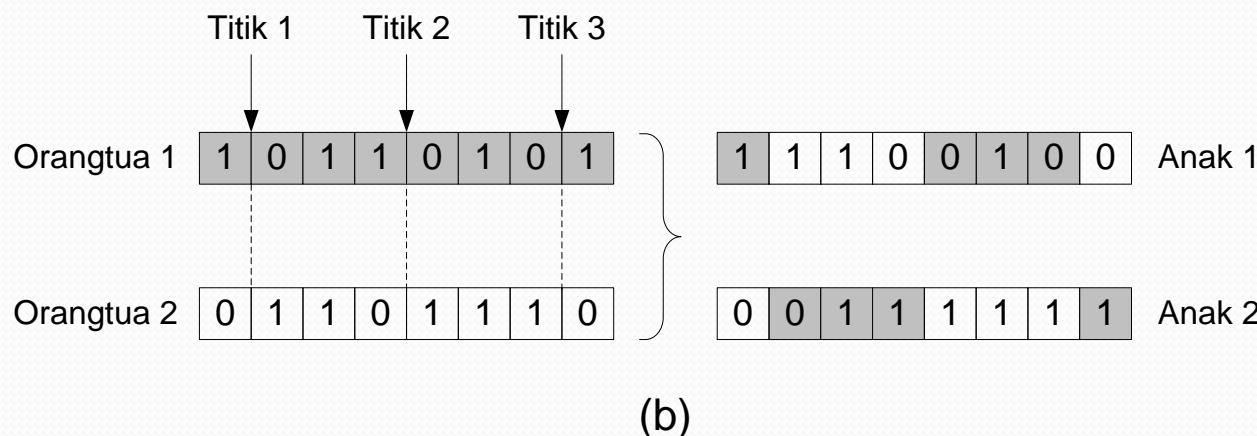
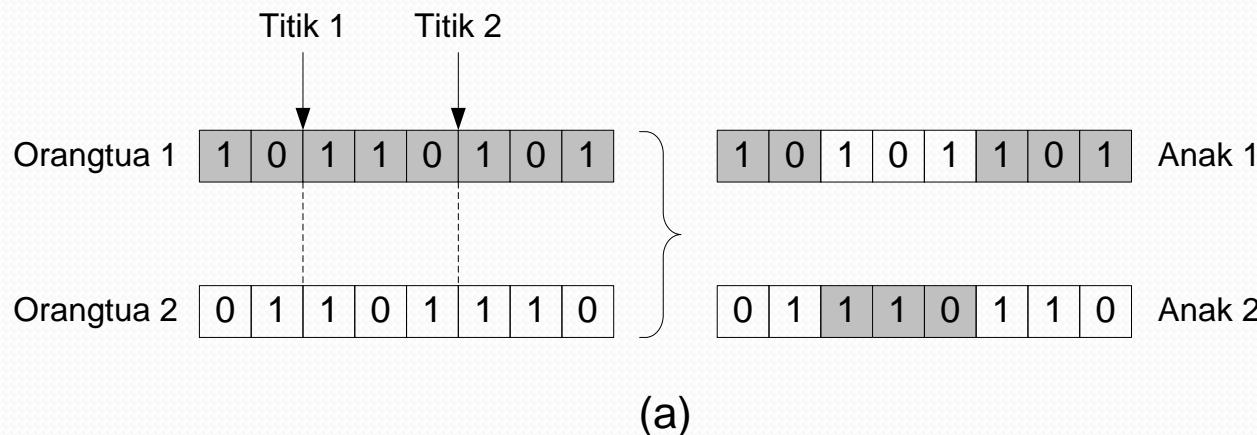
Rekombinasi untuk Rep. Biner

- Rekombinasi satu titik (1-point crossover)
- Rekombinasi banyak titik (Multipoint crossover)
- Rekombinasi seragam (uniform crossover)

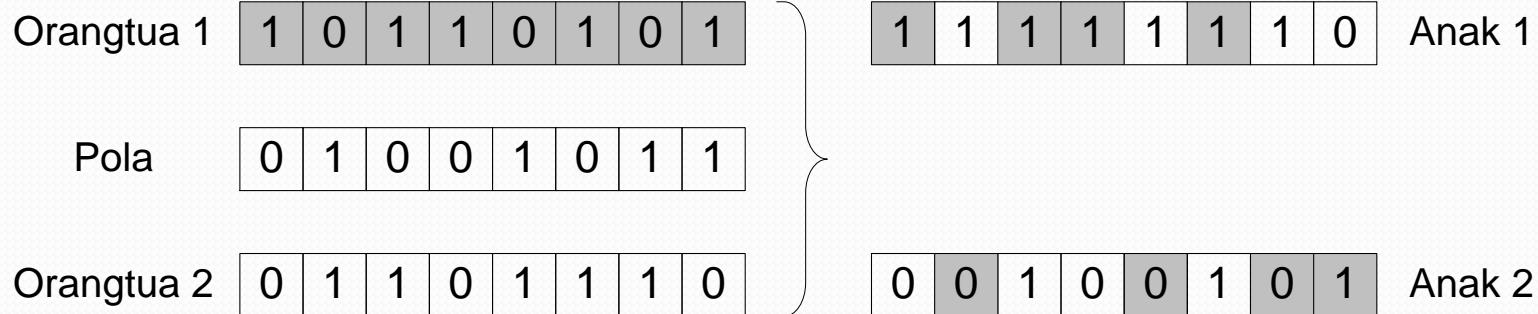
Rekombinasi satu titik



Rekombinasi banyak titik



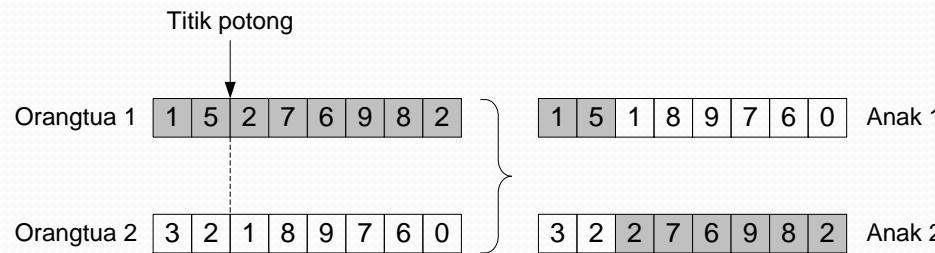
Rekombinasi Seragam



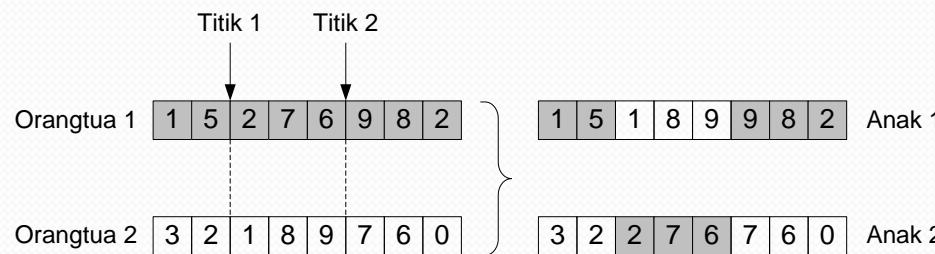
Rekombinasi untuk Rep. Integer

- Rekombinasi satu titik (1-point crossover)
- Rekombinasi banyak titik (Multipoint crossover)
- Rekombinasi seragam (uniform crossover)

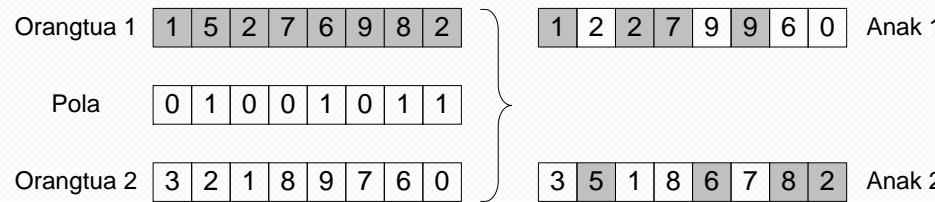
Rekombinasi untuk Rep. Integer



(a)



(b)



(c)

Rekombinasi untuk Rep. Real

- Kita bisa memahami bahwa tiga model rekombinasi di atas dapat digunakan untuk representasi biner dan integer.
- Hal ini disebabkan kedua representasi memiliki nilai-nilai gen dalam interval terbatas. Nilai-nilai gen pada representasi biner hanya berupa 0 atau 1.
- Pada representasi integer, mungkin juga hanya ada sedikit variasi nilai untuk setiap gen. Misalnya, representasi integer yang digunakan untuk merepresentasikan bilangan real hanya memiliki 10 kemungkinan nilai pada setiap gennya.
- Bagaimana jika setiap gen bisa memiliki nilai yang sangat bervariasi seperti representasi real? Untuk representasi real, rekombinasi bisa dilakukan dengan dua cara:
 - *discrete*
 - *intermediate*

Rekombinasi *Discrete*

- Setiap gen pada anak z berasal dari salah satu orangtuanya (x, y) dengan probabilitas yang sama, $z_i = x_i \text{ or } y_i$.
- Cara pemilihan posisi gen bisa menggunakan rekombinasi banyak titik atau rekombinasi seragam.

Rekombinasi *Intermediate*

- Memanfaatkan ide pembangunan anak yang berupa kromosom “antara” dari kedua orangtuanya.
- Oleh karena itu, rekombinasi jenis ini disebut juga *arithmetic crossover*.
- Setiap gen pada anak z diperoleh berdasarkan rumus

$$z_i = \alpha x_i + (1 - \alpha) y_i$$

dimana $0 \leq \alpha \leq 1$. Parameter α bisa dibuat konstan (*uniform arithmetical crossover*), variabel (misalnya, bergantung pada usia populasi), atau ditentukan secara acak pada setiap saat.

Rekombinasi *Intermediate*

Terdapat tiga model *arithmetic crossover*, yaitu:

- *single arithmetic crossover*,
- *simple arithmetic crossover*, dan
- *whole arithmetic crossover*.

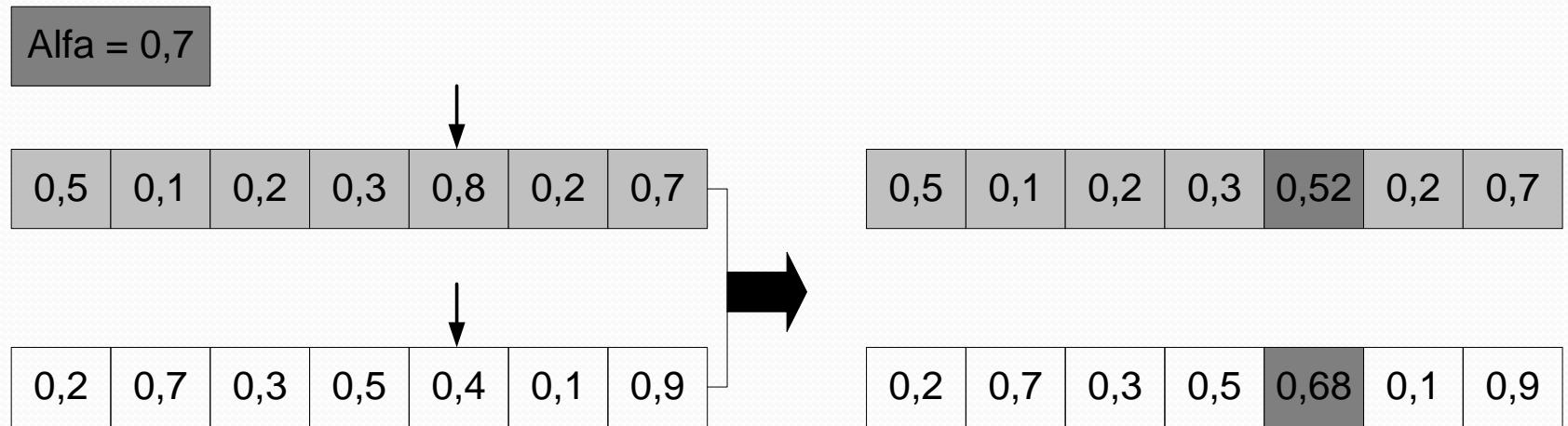
Single arithmetic crossover

- Misalkan dua kromosom orangtua dinyatakan sebagai $\langle x_1, \dots, x_n \rangle$ dan $\langle y_1, \dots, y_n \rangle$.
- Pilih satu gen secara acak, misal k . Selanjutnya, kedua anak dihasilkan dengan cara:

$$\text{Anak 1 : } \langle x_1, \dots, x_{k-1}, \alpha y_k + (1 - \alpha) x_k, \dots, x_n \rangle$$

$$\text{Anak 2 : } \langle y_1, \dots, y_{k-1}, \alpha x_k + (1 - \alpha) y_k, \dots, y_n \rangle$$

Single arithmetic crossover



$$\text{Anak 1 : } \langle x_1, \dots, x_{k-1}, \alpha y_k + (1 - \alpha)x_k, \dots, x_n \rangle$$

$$\text{Anak 2 : } \langle y_1, \dots, y_{k-1}, \alpha x_k + (1 - \alpha)y_k, \dots, y_n \rangle$$

Simple arithmetic crossover

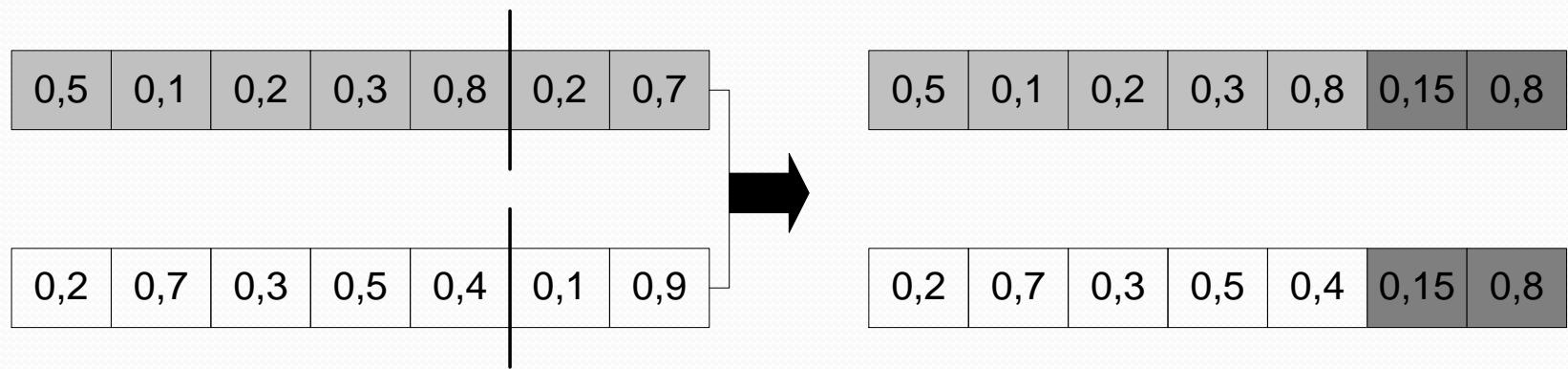
- Misalkan dua kromosom orangtua dinyatakan sebagai $\langle x_1, \dots, x_n \rangle$ dan $\langle y_1, \dots, y_n \rangle$.
- Pilih satu gen secara acak, misal k . Selanjutnya, kedua anak dihasilkan dengan cara:

Anak 1 : $\langle x_1, \dots, x_k, \alpha y_{k+1} + (1 - \alpha)x_{k+1}, \dots, \alpha y_n + (1 - \alpha)x_n \rangle$

Anak 2 : $\langle y_1, \dots, y_k, \alpha x_{k+1} + (1 - \alpha)y_{k+1}, \dots, \alpha x_n + (1 - \alpha)y_n \rangle$

Simple arithmetic crossover

Alfa = 0,5



$$\text{Anak 1 : } \langle x_1, \dots, x_k, \alpha y_{k+1} + (1 - \alpha)x_{k+1}, \dots, \alpha y_n + (1 - \alpha)x_n \rangle$$

$$\text{Anak 2 : } \langle y_1, \dots, y_k, \alpha x_{k+1} + (1 - \alpha)y_{k+1}, \dots, \alpha x_n + (1 - \alpha)y_n \rangle$$

Whole arithmetic crossover

- Misalkan dua kromosom orangtua dinyatakan sebagai $\langle x_1, \dots, x_n \rangle$ dan $\langle y_1, \dots, y_n \rangle$.
- Kedua anak dihasilkan dengan cara:

$$\text{Anak 1: } \alpha \cdot \bar{x} + (1 - \alpha) \cdot \bar{y}$$

$$\text{Anak 2: } \alpha \cdot \bar{y} + (1 - \alpha) \cdot \bar{x}$$

Whole arithmetic crossover

Alfa = 0,5

0,5	0,1	0,2	0,3	0,8	0,2	0,7
-----	-----	-----	-----	-----	-----	-----

0,35	0,4	0,25	0,4	0,6	0,15	0,8
------	-----	------	-----	-----	------	-----

0,2	0,7	0,3	0,5	0,4	0,1	0,9
-----	-----	-----	-----	-----	-----	-----

0,35	0,4	0,25	0,4	0,6	0,15	0,8
------	-----	------	-----	-----	------	-----



$$\text{Anak 1 : } \alpha \cdot \bar{x} + (1 - \alpha) \cdot \bar{y}$$

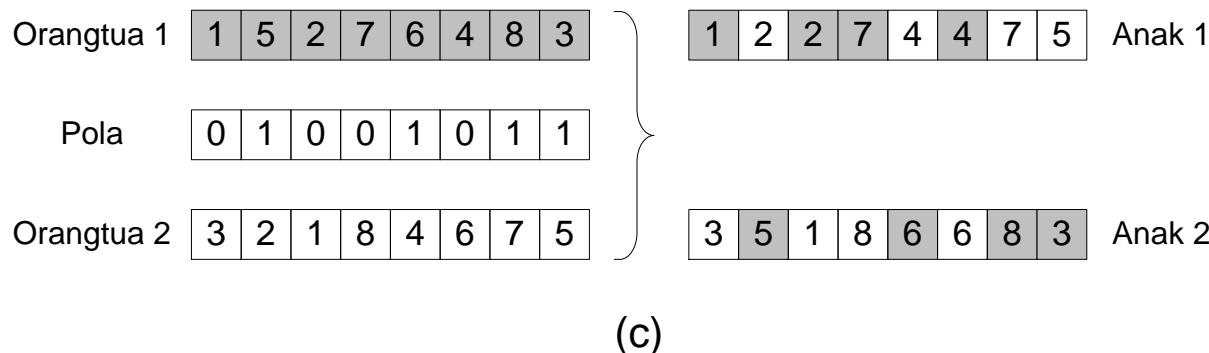
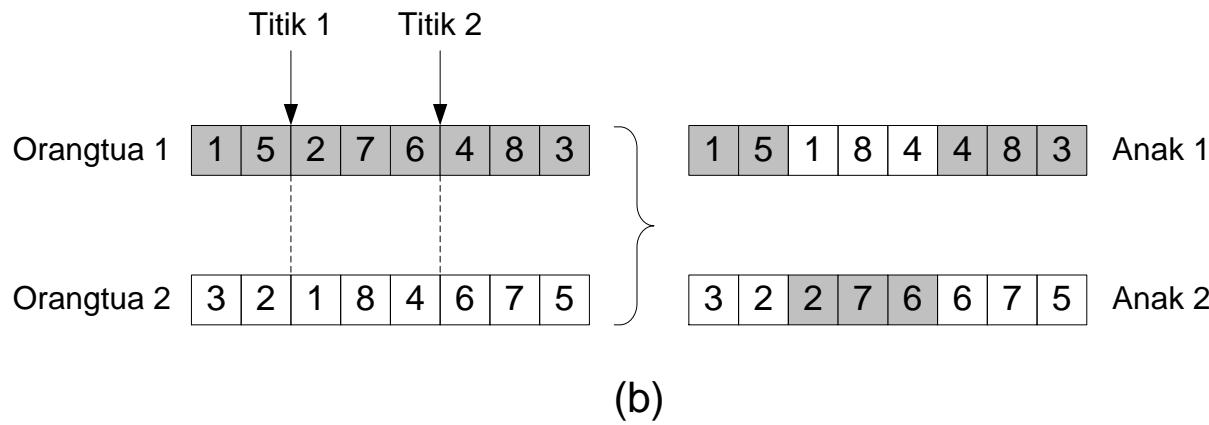
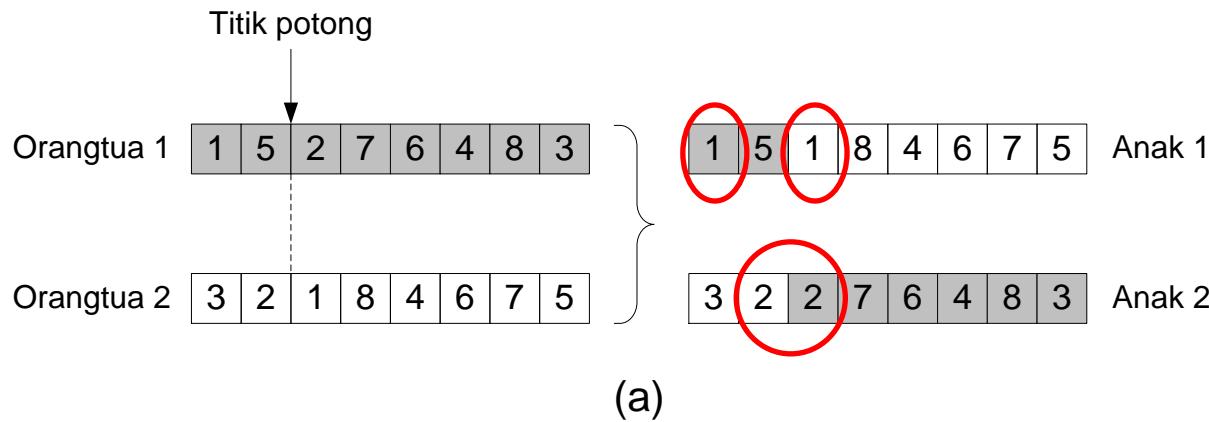
$$\text{Anak 2 : } \alpha \cdot \bar{y} + (1 - \alpha) \cdot \bar{x}$$

Rekombinasi untuk Rep. Permutasi

- Sesuai dengan namanya, representasi permutasi digunakan untuk masalah-masalah permutasi, seperti *Travelling Salesman Problem* (TSP) misalnya.
- Pada masalah TSP, representasi permutasi memanfaatkan posisi gen sebagai urutan kota. Hal ini menyebabkan metode-metode rekombinasi untuk representasi Biner, Integer maupun Real di atas tidak bisa digunakan untuk representasi ini.
- Mengapa tidak bisa?

Rekombinasi untuk Rep. Permutasi

- Karena ada kemungkinan anak-anak yang dihasilkan memiliki gen-gen yang **tidak valid**.
- Misalkan, ada dua gen yang bernilai sama dalam suatu kromosom yang berarti ada satu lokasi yang dikunjungi dua kali. Padahal pada masalah TSP ada batasan bahwa setiap lokasi hanya boleh dikunjungi maksimum satu kali.
- Sebaliknya, mungkin saja ada nomor lokasi yang tidak pernah muncul di dalam kromosom, yang berarti ada lokasi yang tidak pernah dikunjungi.



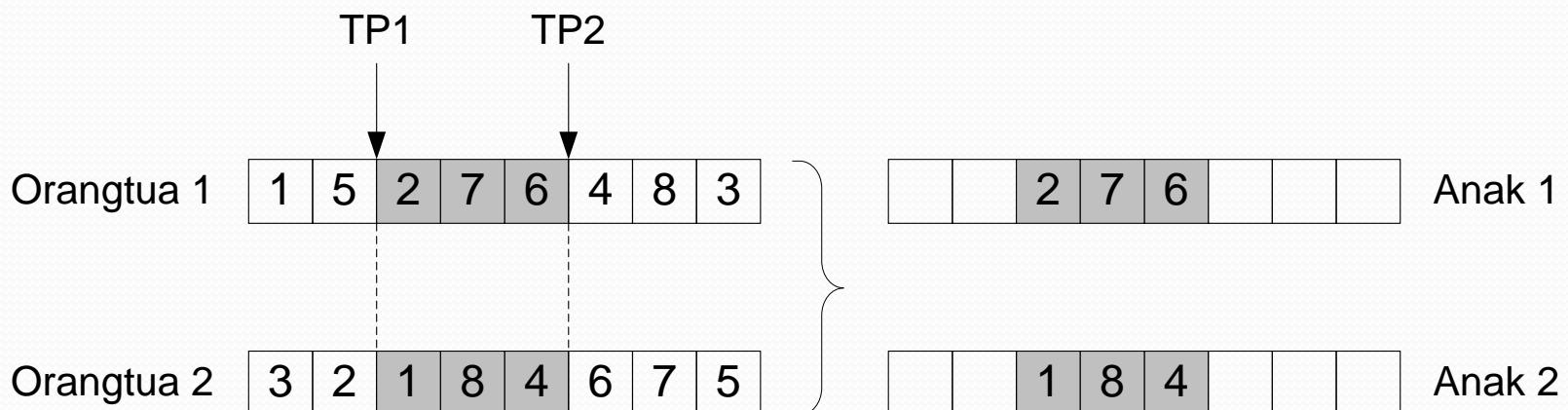
Rekombinasi untuk Rep. Permutasi

- *Order crossover*
- *Partially mapped crossover*
- *Cycle crossover*
- *Edge Recombination*

Order crossover

- Satu bagian kromosom dipertukarkan
- Tetapi urutan gen secara relatif yang bukan bagian dari kromosom tersebut tetap dijaga.

Order crossover



Gen orangtua 2 yang belum ada di Anak 1,
terurut setelah TP2: {5, 3, 1, 8, 4}

Gen orangtua 1 yang belum ada di Anak 2,
terurut setelah TP2: {3, 5, 2, 7, 6}

8 4 2 7 6 5 3 1 Anak 1

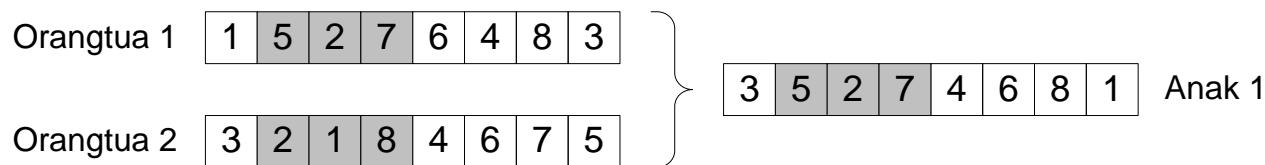
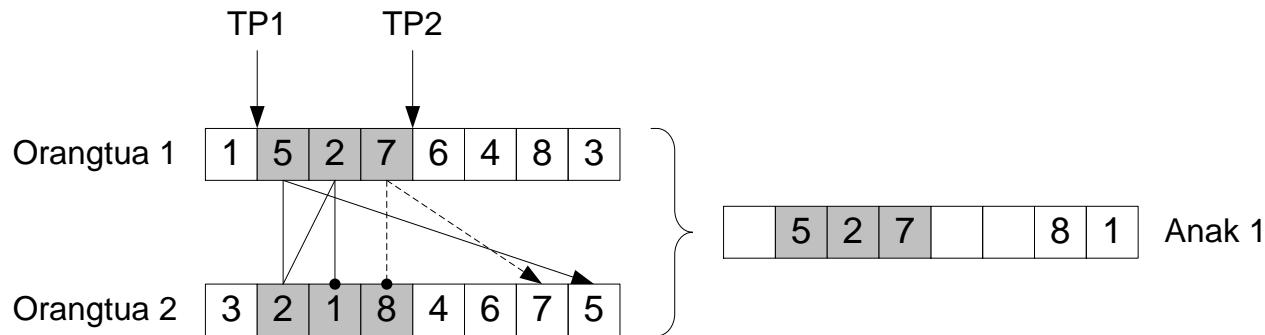
7 6 1 8 4 3 5 2 Anak 2

Algoritma *Order crossover*

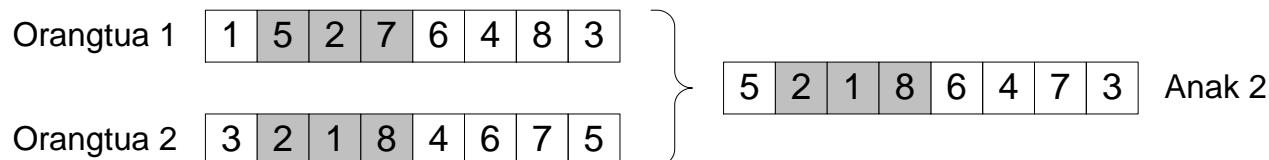
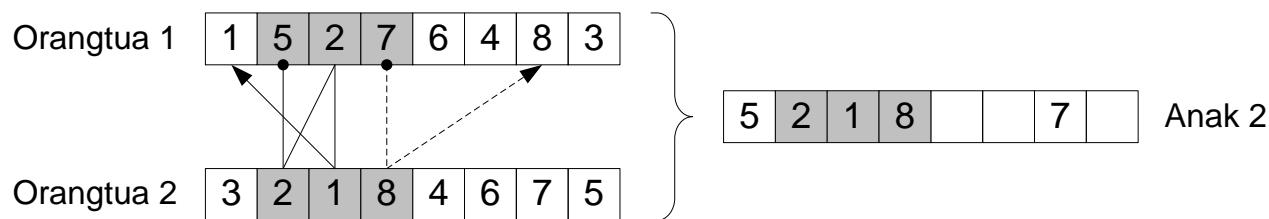
1. Pilih segmen kromosom dari kedua orangtua secara acak dengan cara membangkitkan dua titik potong, TP1 dan TP2.
2. Kopi bagian ini secara searah ke kedua anaknya. Artinya, segmen kromosom Orangtua 1 dikopi ke Anak 1 dan segmen kromosom Orangtua 2 dikopi ke Anak 2.
3. Kopi gen-gen Orangtua 2, yang tidak ada di Anak 1, ke Anak 1 dengan aturan:
 - Mulai dari posisi setelah TP2,
 - Memperhatikan urutan yang ada pada Orangtua 2
 - Kembali ke posisi awal setelah akhir kromosom (*wrapping*)
4. Lakukan langkah 3 dengan cara yang sama terhadap Orangtua 1 untuk menghasilkan Anak 2.

Partially mapped crossover

- Sama dengan metode *order crossover*, metode ini juga mewariskan sebagian gen orangtua secara searah dan sebagian lainnya secara menyilang kepada kedua anaknya.
- Pewarisan sebagian gen secara menyilang dilakukan dengan memanfaatkan posisi-posisi gen kedua orangtuanya yang memiliki nilai sama untuk dilakukan pemetaan (*mapping*).
- Karena pewarisan gen yang menggunakan pemetaan dilakukan hanya pada sebagian gen, maka metode rekombinasi ini dinamakan *partially mapped crossover*.



(a)



(b)

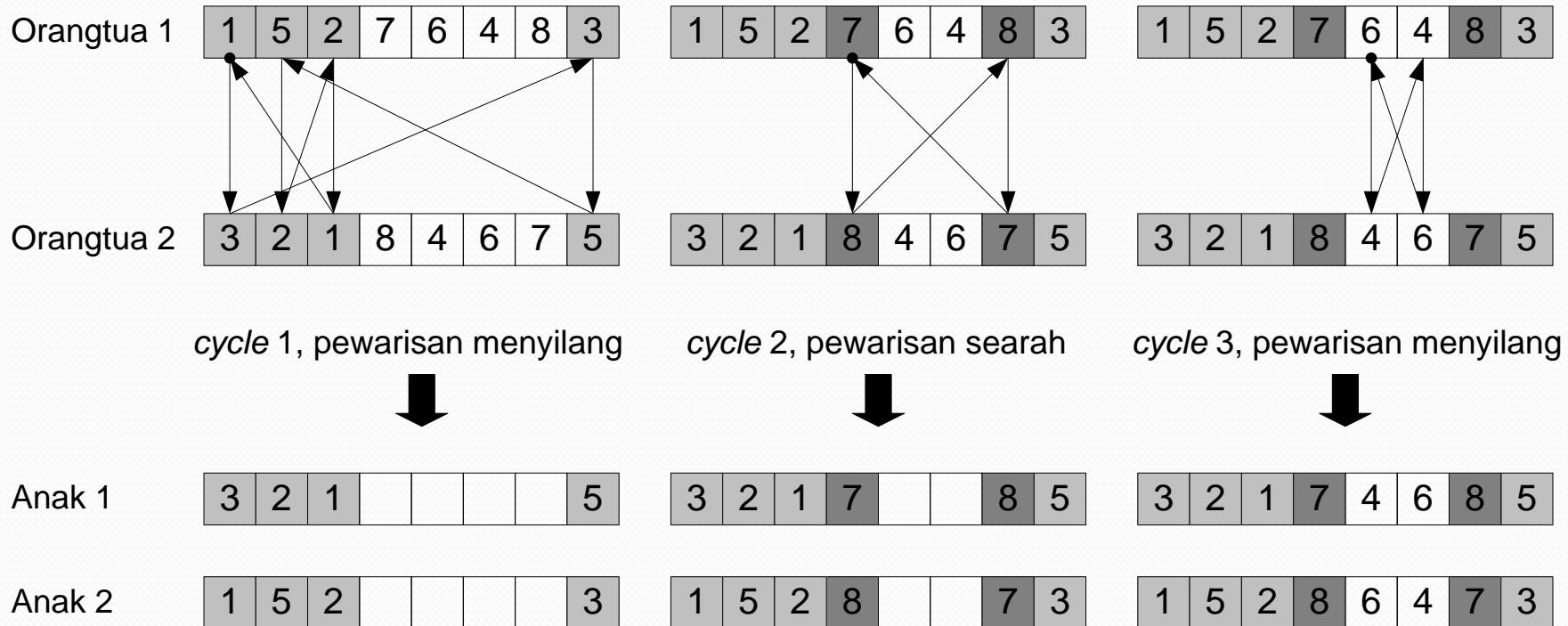
Algoritma *Partially mapped crossover*

1. Pilih segmen kromosom dari kedua orangtua secara acak dengan cara mebangkitkan dua titik, TP1 dan TP2.
2. Kopi segmen Orangtua 1 ke Anak 1
3. Mulai dari posisi TP1 lakukan pemetaan gen-gen yang ada di segmen Orangtua 2 tetapi tidak ada di segmen Orangtua 1.
4. Wariskan setiap gen tersebut ke Anak 1 pada posisi hasil pemetaan
5. Setelah semua gen di dalam segmen sudah diwariskan ke Anak 1, maka posisi-posisi gen Anak 1 yang masih kosong diisi dengan gen-gen Orangtua 2 pada posisi-posisi yang bersesuaian.
6. Lakukan hal sama pada untuk membangkitkan Anak 2.

Cycle crossover

- Sesuai dengan namanya, algoritma ini mencari siklus-siklus (*cycles*) yang terdapat pada kedua kromosom orangtua dan mewariskan *cycles* tersebut secara menyilang dan searah secara bergantian.
- Artinya, pada *cycle* pertama pewarisan dilakukan secara menyilang. Pada *cycle* ke-2 pewarisan dilakukan searah. Pada *cycle* ke-3 pewarisan kembali dilakukan secara menyilang, dan seterusnya.
- Dengan cara ini akan dihasilkan dua anak yang selalu valid.

Cycle crossover



Algoritma *Cycle crossover*

1. Cari cycle pada Orangtua 1 dengan cara:
 - a. Mulai dari posisi pertama Orangtua 1 yang belum diwariskan.
 - b. Buat panah ke posisi yang sama pada Orangtua 2.
 - c. Buat panah ke posisi gen yang bernilai sama pada Orangtua 1.
 - d. Tambahkan gen ini ke cycle.
 - e. Ulangi langkah b sampai d sampai panah kembali ke posisi awal.
2. Wariskan gen-gen orangtua yang berada pada cycle ini kepada kedua anaknya sesuai dengan posisinya dengan cara menyilang, searah, menyilang, searah, dan seterusnya.

Edge Recombination

- Algoritma ini bekerja dengan cara membangun suatu yang mendaftar *edges* (sisi-sisi) gen yang berada pada kedua orangtuanya.
- Untuk mendapatkan *edge* dari suatu gen , kromosom dianggap melingkar. Artinya, gen pada posisi 1 memiliki *edge* yang berupa gen posisi 2 dan gen posisi terakhir.
- Gen pada posisi terakhir memiliki *edge* yang berupa gen posisi sebelumnya dan gen posisi 1.
- Jika *edge* berada pada kedua orangtua, maka *edge* tersebut diberi tanda positif ‘+’ dan disebut sebagai *common edge*.
- Pewarisan gen-gen orangtua dilakukan berdasarkan tabel tersebut.

Orangtua 1

1	5	2	7	6	4	8	3
---	---	---	---	---	---	---	---

Orangtua 2

3	2	1	8	4	6	7	5
---	---	---	---	---	---	---	---

Pencarian edge untuk semua gen pada kedua Orangtua menghasilkan tabel sbb:

Elemen	Edge
1	3, 5, 2, 8
2	5, 7, 3, 1
3	8, 1, 5, 2
4	6+, 8+
5	1, 2, 7, 3
6	4+, 7+
7	2, 6+, 5
8	4+, 3, 1

Tabel harus di-update setelah suatu elemen terpilih. Setelah 3 terpilih, semua edge 3 dihapus dari tabel.

Elemen	Edge
1	5, 2, 8
2	5, 7, 1
3	8, 1, 5, 2
4	6+, 8+
5	1, 2, 7
6	4+, 7+
7	2, 6+, 5
8	4+, 1

...

Pilihan	Elemen terpilih	Alasan	Hasil
Semua	3	Pemilihan secara acak	{3}
8, 1, 5, 2	8	Daftar edge terpendek	{3,8}
4, 1	4	Common edge	{3,8,4}
6	6	Common edge	{3,8,4,6}
7	7	Common edge	{3,8,4,6,7}
2, 5	2	Pemilihan acak	{3,8,4,6,7,2}
5, 1	1	Pemilihan acak	{3,8,4,6,7,2,1}
5	5	Hanya ada satu pilihan	{3,8,4,6,7,2,1,5}

Anak1.
Anak2
dihasilkan
dengan cara
yang sama.
Karena acak,
maka Anak2
bisa berbeda
dengan Anak1.

Edge Recombination

- Pada gambar di atas, suatu tabel elemen dan *edge* dibangun berdasarkan susunan gen-gen yang berada pada kedua orangtuanya. Setelah tabel dibangun, suatu elemen awal dipilih secara acak dari semua elemen yang ada (1 sampai 8).
- Misalkan elemen 3 terpilih dan dimasukkan ke hasil. Setelah elemen 3 terpilih, maka tabel harus di-update dengan cara menghapus semua elemen 3 dari daftar *edge* (lihat tabel sebelah kanan).
- Selanjutnya, pilih elemen berikutnya dari daftar *edge* yang ada pada elemen 3, yaitu {8, 1, 5, 2}. Pemilihan elemen dilakukan berdasarkan aturan berprioritas sebagai berikut:
 - Jika ada elemen yang merupakan *common edge*, maka pilih elemen tersebut.
 - Jika tidak ada *common edge*, pilih elemen yang yang memiliki daftar *edge* terpendek (jumlah *edge* paling sedikit).
 - Jika semua elemen memiliki jumlah *edge* yang sama, maka pilih elemen secara acak.

Algoritma *Edge Recombination*

1. Pilih elemen awal secara acak dari semua elemen yang ada. Kemudian masukkan elemen terpilih ke Hasil.
2. Set elemen terpilih sebagai *current element*
3. Hapus *current element* dari semua daftar edge yang ada di tabel
4. Pilih satu elemen dari daftar edge yang ada di *current element* dengan aturan:
 - a. Jika ada elemen yang merupakan common edge, maka pilih elemen tersebut.
 - b. Jika tidak ada common edge, pilih elemen yang yang memiliki daftar edge terpendek (jumlah edge paling sedikit).
 - c. Jika semua elemen memiliki jumlah edge yang sama, maka pilih elemen secara acak.
5. Jika daftar edge yang ada di *current element* sudah kosong, maka
 - i. Ubah pemilihan elemen terakhir untuk mencari elemen lain yang memiliki daftar edge tidak kosong.
 - ii. Jika perubahan elemen tidak berhasil, pilih elemen baru secara acak.

Rekombinasi Path Relinking

- Rekombinasi dilakukan dengan membuat banyak anak yang memiliki perbedaan secara berurutan sehingga mirip suatu jalur. Kemudian pilih sejumlah anak yang memiliki *fitness* tertinggi.
- Bagaimana cara membuat anak-anak yang memiliki perbedaan secara berurutan?
- Mudah saja. Buat Anak 1 yang sebagian besar gen-nya (misal 90%) diambil dari Orangtua 1 dan sebagian kecil gen lainnya (10%) diambil dari Orangtua 2. Buat Anak 2 dengan porsi gen yang sedikit berbeda, misalnya 80% dari Orangtua 1 dan 20% dari Orangtua 2.
- Dengan demikian, bisa dikatakan Anak 1 mirip dengan Orangtua 1. Anak 2 mirip dengan Anak 1 dan seterusnya sehingga Anak *n* mirip dengan Orangtua 2.

Rekombinasi Path Relinking

Orangtua 1

1	0	0	1	0	0	1
---	---	---	---	---	---	---

Orangtua 2

0	0	1	0	1	0	0
---	---	---	---	---	---	---



Anak 1

1	0	0	1	0	0	0
---	---	---	---	---	---	---

Anak 2

1	0	0	1	1	0	0
---	---	---	---	---	---	---

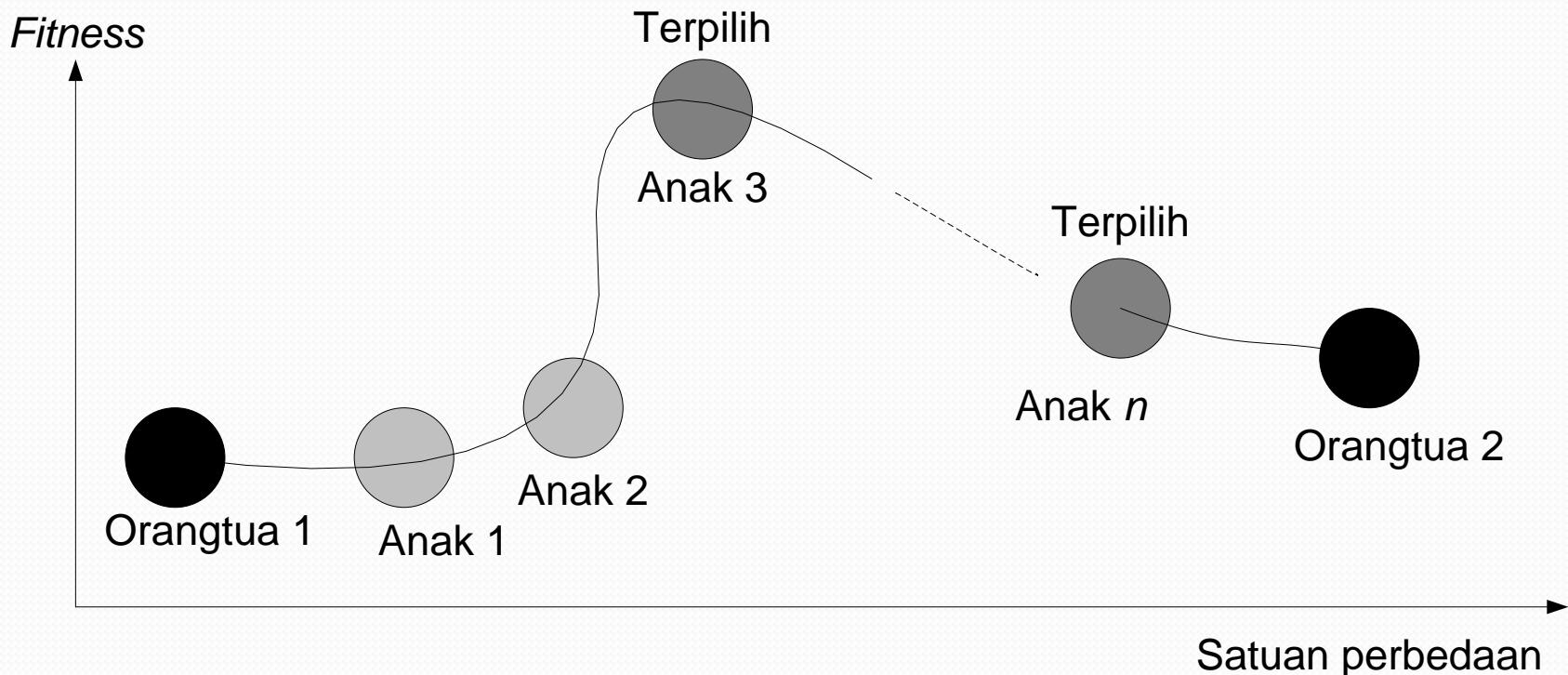
Anak 3

1	0	0	0	1	0	0
---	---	---	---	---	---	---

Anak 4

1	0	1	0	1	0	0
---	---	---	---	---	---	---

Rekombinasi Path Relinking



Rekombinasi Multi-parent

- Bagaimanapun, untuk membangun GA kita mungkin tidak perlu terlalu kaku dalam mengadopsi apa yang ada di dunia nyata.
- Mungkin saja penggunaan lebih dari dua kromosom orangtua dalam proses rekombinasi akan memberikan hasil yang lebih baik.
- Hal ini yang disebut sebagai rekombinasi banyak orangtua (*Multi-parent*).

Rekombinasi Multi-parent

- Bagaimana penerapan rekombinasi *Multi-parent* ini?
- Terdapat tiga pendekatan yang bisa digunakan, yaitu:
 - Berdasarkan frekuensi allele
 - Berdasarkan segmentasi dan rekombinasi
 - Berdasarkan operasi-operasi numerik pada allele bernilai real

Berdasarkan frekuensi allele

- Pendekatan ini merupakan generalisasi dari rekombinasi seragam (*uniform crossover*).
- Jika pada *uniform crossover* dilakukan pembangkitan pola dengan dua kemungkinan nilai (berdasarkan pelemparan koin), maka pada pendekatan ini pembangkitan pola dilakukan dengan kemungkinan nilai sebanyak jumlah orangtuanya.
- Jika orangtuanya berjumlah tiga, maka pembangkitan pola dilakukan untuk menghasilkan tiga nilai berbeda.

Berdasarkan frekuensi allele

Orangtua 1

0	0	1	1	0	1	0	1
---	---	---	---	---	---	---	---

Orangtua 2

1	1	0	1	1	0	1	0
---	---	---	---	---	---	---	---

Orangtua 3

0	1	1	0	1	1	1	0
---	---	---	---	---	---	---	---

Pola 1

2	1	3	3	2	1	1	3
---	---	---	---	---	---	---	---

Pola 2

1	1	3	2	2	1	2	3
---	---	---	---	---	---	---	---

Anak 1

1	0	1	0	1	1	0	0
---	---	---	---	---	---	---	---

Anak 2

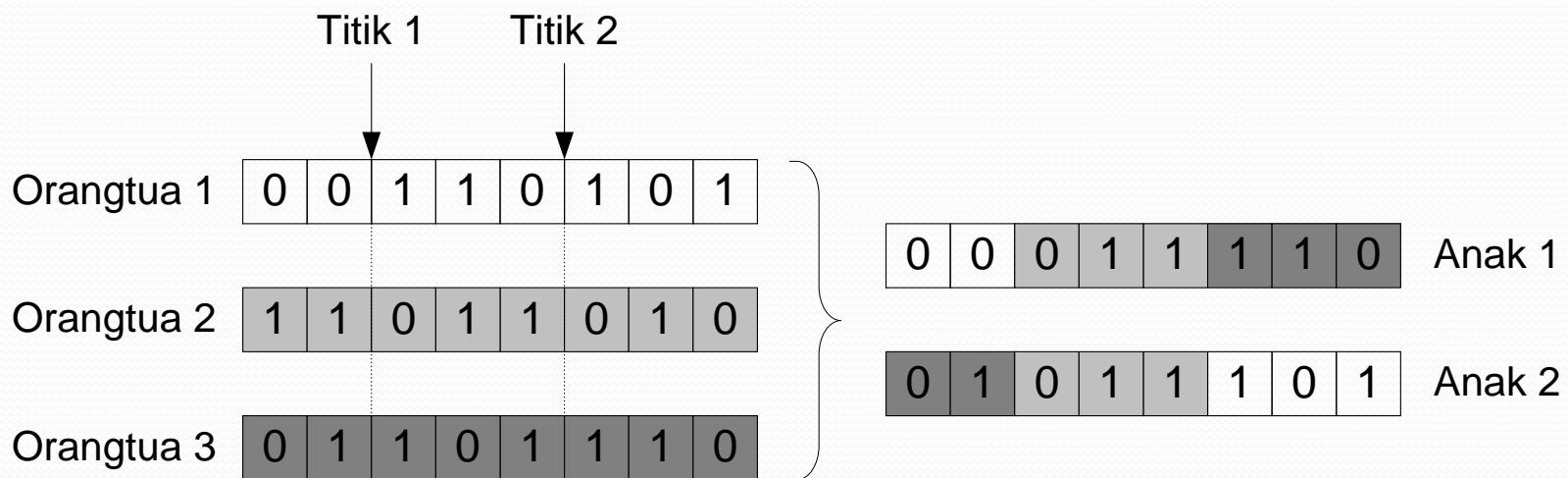
0	0	1	1	1	1	1	0
---	---	---	---	---	---	---	---

Anak 1 dihasilkan berdasarkan Pola 1
dan Anak 2 berdasarkan Pola 2.

Berdasarkan segmentasi dan rekombinasi

- Pendekatan ini merupakan generalisasi dari rekombinasi banyak titik (*n-point crossover*).
- Pewarisan gen dilakukan secara diagonal.

Berdasarkan segmentasi dan rekombinasi



Berdasarkan operasi-operasi numerik (pada allele bernilai real)

- Pendekatan ini merupakan generalisasi dari rekombinasi *Arithmetic Recombination*.
- Salah satu contohnya adalah rekombinasi pusat massa (*the center of mass crossover*).

Rekombinasi Biner/Integer

```
function Anak = RekombinasiBiner(Ortu1,Ortu2,JumGen)
```

```
TP = 1 + fix(rand*(JumGen-1));
```

```
Anak(1,:) = [Ortu1(1:TP) Ortu2(TP+1:JumGen)];
```

```
Anak(2,:) = [Ortu2(1:TP) Ortu1(TP+1:JumGen)];
```

Rekombinasi Real

```
function Anak = RekombinasiReal(Ortu1, Ortu2, JumGen)
```

```
Alfa = 0.8; % Alfa bisa diset sesuai kebutuhan
```

```
% Whole arithmetic crossover
```

```
Anak(1,:) = (Alfa .* Ortu1) + (1 - Alfa) .* Ortu2;  
Anak(2,:) = (Alfa .* Ortu2) + (1 - Alfa) .* Ortu1;
```

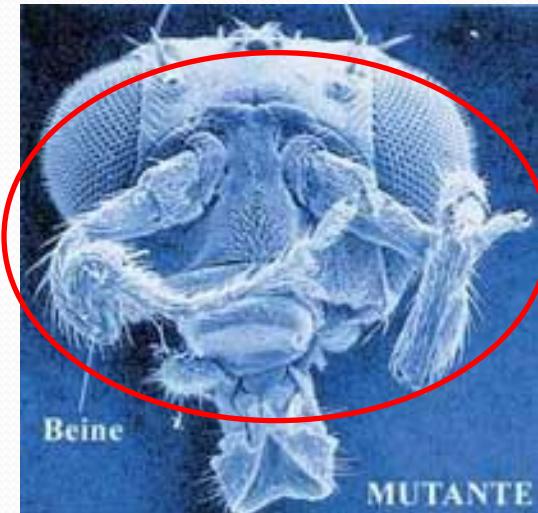
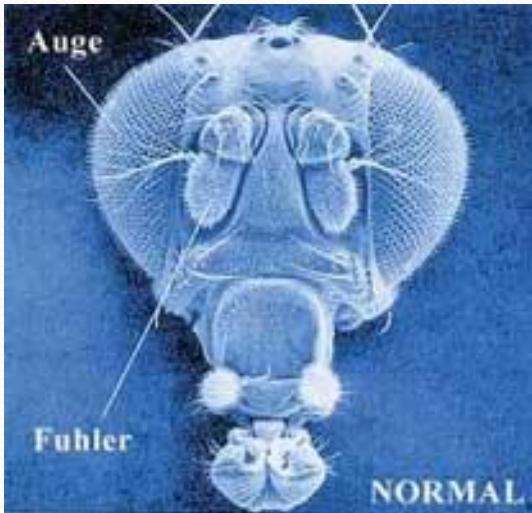
Mutasi

- Setelah tahap rekombinasi terhadap semua pasangan kromosom pada *mating pool* yang menghasilkan N (ukuran populasi) kromosom, maka GA menjalankan operator mutasi terhadap setiap kromosom tersebut.
- Banyak metode mutasi yang telah diusulkan.
- Masing-masing metode memiliki ciri khusus dan mungkin saja hanya bisa digunakan pada jenis representasi tertentu.
- Misalnya, mutasi untuk representasi permutasi memerlukan metode khusus yang menjamin kromosom hasil mutasi tetap valid.

Mutasi

- Mutasi bersifat kecil, acak, berbahaya, dan jarang terjadi. Jika terjadi, kemungkinan besar mutasi itu tidak berguna.
- Empat karakteristik mutasi ini menunjukkan bahwa mutasi tidak dapat mengarah pada perkembangan evolusioner.
- Suatu perubahan acak pada organisme yang sangat terspesialisasi bersifat tidak berguna atau membahayakan.
- Perubahan acak pada sebuah jam tidak dapat memperbaiki, malah kemungkinan besar akan merusaknya atau tidak berpengaruh sama sekali.
- Gempa bumi tidak akan memperbaiki kota, tetapi menghancurnya [RAN88].

Mutasi: bisa lebih baik?



- Struktur DNA **amat sangat rumit !**
- Perubahan acak (mutasi) **selalu buruk !!**

Kiri: seekor lalat buah (*drosophila*) normal. Tengah: seekor lalat buah dengan kaki tumbuh di kepala (mutasi akibat radiasi). Kanan: Bocah laki-laki korban kecelakaan instalasi nuklir Chernobyl yang mengakibatkan mutasi gen [ADNo7].

Mutasi pada EAs

- Mutasi untuk representasi Biner
- Mutasi untuk representasi Integer
- Mutasi untuk representasi Real
- Mutasi untuk representasi Permutasi

Mutasi: bisa lebih baik?

- Bisa. Mengapa?
- Representasi individu pada EAs jauh lebih sederhana
- Mutasi sebagian kecil gen mungkin menghasilkan individu yang lebih baik

Mutasi untuk Representasi Biner

Kromosom awal

0	1	1	0	1	1	1	0
---	---	---	---	---	---	---	---



$$a \leq P_m$$

Kromosom hasil mutasi

0	0	1	0	1	1	1	0
---	---	---	---	---	---	---	---

Membalik bit: 1 → 0 dan sebaliknya.

Mutasi untuk representasi Integer

- Membalik nilai integer
- Pemilihan nilai secara acak
- Mutasi *Creep* (Perlahan)

Mutasi untuk Representasi Integer

Membalik nilai integer

Kromosom awal

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



$$a \leq P_m$$

Kromosom hasil mutasi

3	6	8	0	5	1	2	9
---	---	---	---	---	---	---	---

Mutasi untuk Representasi Integer

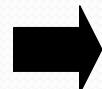
Pemilihan nilai secara acak

Kromosom awal

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



$$a \leq P_m$$



Kromosom hasil mutasi

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

Mutasi untuk Representasi Integer

Mutasi *Creep* (Perlahan)

Kromosom awal

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



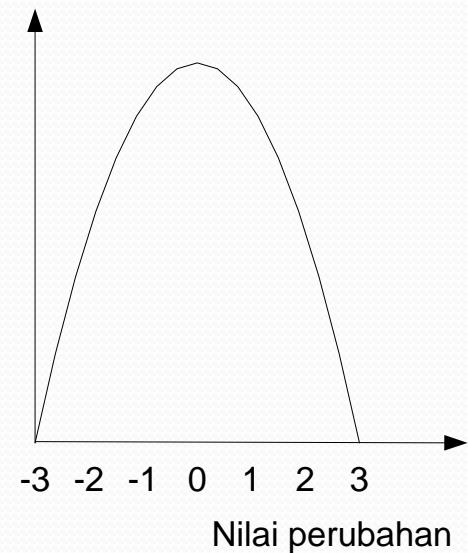
$$a \leq P_m$$

Kromosom hasil mutasi

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



Probabilitas



Mutasi untuk Representasi Real

- Representasi real memiliki karakteristik yang berbeda dengan biner ataupun integer.
- Nilai-nilai gen pada representasi real bersifat kontinyu sedangkan representasi biner dan integer bersifat diskrit.
- Oleh karena itu, representasi real memerlukan mutasi khusus yang berbeda dengan sebelumnya.

Mutasi untuk Representasi Real

- Mutasi *Uniform*

Nilai-nilai x_i didapat dari pembangkitan bilangan secara acak dengan distribusi seragam (*uniform distribution*)

- Mutasi *Non-uniform* dengan distribusi tetap

Mutasi jenis ini paling umum digunakan. Caranya mirip dengan metode *Creep* pada representasi integer. Bedanya di sini digunakan penambahan nilai real (bukan integer).

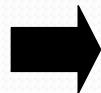
Mutasi untuk Rep. Permutasi

- Mutasi pada representasi permutasi harus menghasilkan kromosom yang valid.
- Sehingga, proses mutasi dilakukan dengan suatu cara tertentu yang menjamin kromosom hasil mutasi tetap valid.
- Banyak cara yang bisa digunakan, diantaranya adalah:
 - Mutasi pertukaran (*Swap Mutation*)
 - Mutasi penyisipan (*Insert Mutation*)
 - Mutasi pengacakan (*Scramble Mutation*)
 - Mutasi pembalikan (*Inversion Mutation*)

Mutasi pertukaran (Swap Mutation)

Kromosom awal

1	5	2	7	6	4	8	3
---	---	---	---	---	---	---	---



Kromosom hasil mutasi

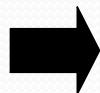
1	8	2	7	6	4	5	3
---	---	---	---	---	---	---	---

Mutasi pertukaran (*Swap Mutation*). Pemilihan dua posisi gen secara acak menghasilkan posisi 2 dan 7. Sehingga gen bernilai 5 dipertukarkan dengan gen bernilai 8.

Mutasi penyisipan (Insert Mutation)

Kromosom awal

1	5	2	7	6	4	8	3
---	---	---	---	---	---	---	---



Kromosom hasil mutasi

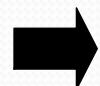
1	5	8	2	7	6	4	3
---	---	---	---	---	---	---	---

Mutasi penyisipan (*Insert Mutation*). Pemilihan dua posisi gen secara acak menghasilkan posisi 2 dan 7. Kemudian gen bernilai 8 disisipkan setelah gen bernilai 5.

Mutasi pengacakan (*Scramble Mutation*)

Kromosom awal

1	5	2	7	6	4	8	3
---	---	---	---	---	---	---	---

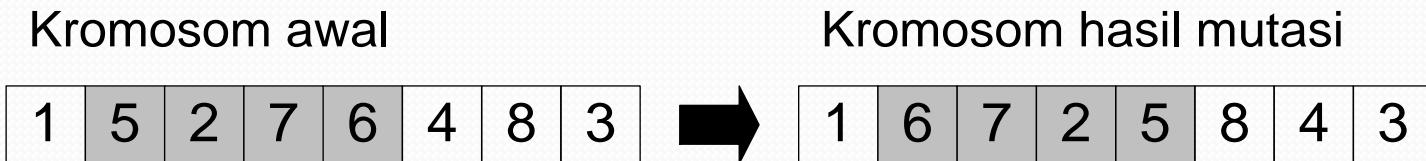


Kromosom hasil mutasi

1	2	6	5	7	8	4	3
---	---	---	---	---	---	---	---

Mutasi pengacakan (*Scramble Mutation*). Pemilihan segmen kromosom menghasilkan {5, 2, 7, 6}. Kemudian pengacakan gen dalam segmen menghasilkan {2, 6, 5, 7}

Mutasi pembalikan (Inversion Mutation)



Mutasi pembalikan (*Inversion Mutation*). Pemilihan segmen kromosom menghasilkan {5, 2, 7, 6}. Kemudian pembalikan posisi gen dalam segmen tersebut menghasilkan {6, 7, 2, 5}.

Mutasi Biner

```
function MutKrom = MutasiBiner(Kromosom, JumGen, Pmutasi)

MutKrom = Kromosom;
for ii=1:JumGen,
    if (rand < Pmutasi),
        if Kromosom(ii)==0,
            MutKrom(ii) = 1;
        else
            MutKrom(ii) = 0;
        end
    end
end
```

Mutasi Integer

```
function MutKrom = MutasiInteger(Kromosom, JumGen, Pmutasi)
```

```
MutKrom = Kromosom;  
for ii=1:JumGen,  
    if (rand < Pmutasi),  
        a = fix(randn * 2);  
        MutKrom(ii) = abs(a + Kromosom(ii));  
        if MutKrom(ii) > 9,  
            MutKrom(ii) = 9;  
    end  
end  
end
```

Mutasi Real

```
function MutKrom = MutasiReal(Kromosom, JumGen, Pmutasi)
CreepSize = 0.001;
MutKrom = Kromosom;
for ii=1:JumGen,
    if (rand < Pmutasi),
        a = (randn * CreepSize);
        MutKrom(ii) = a + Kromosom(ii);
        if MutKrom(ii) > 1,
            MutKrom(ii) = 1;
        elseif MutKrom(ii) < 0,
            MutKrom(ii) = 0;
    end
end
end
```

Permutasi: Swap mutation

```
function MutKrom = TSPMutasi (Kromosom, JumGen, Pmutasi)
```

```
MutKrom = Kromosom;
```

```
for ii=1:JumGen,
```

```
    if rand < Pmutasi,
```

```
        TM2 = 1 + fix(rand*JumGen);
```

```
        while TM2==ii,
```

```
            TM2 = 1 + fix(rand*JumGen);
```

```
        end
```

```
        temp = MutKrom(ii);
```

```
        MutKrom(ii) = MutKrom(TM2);
```

```
        MutKrom(TM2) = temp;
```

```
    end
```

```
end
```

Kromosom awal

1	5	2	7	6	4	8	3
---	---	---	---	---	---	---	---



Kromosom hasil mutasi

1	8	2	7	6	4	5	3
---	---	---	---	---	---	---	---

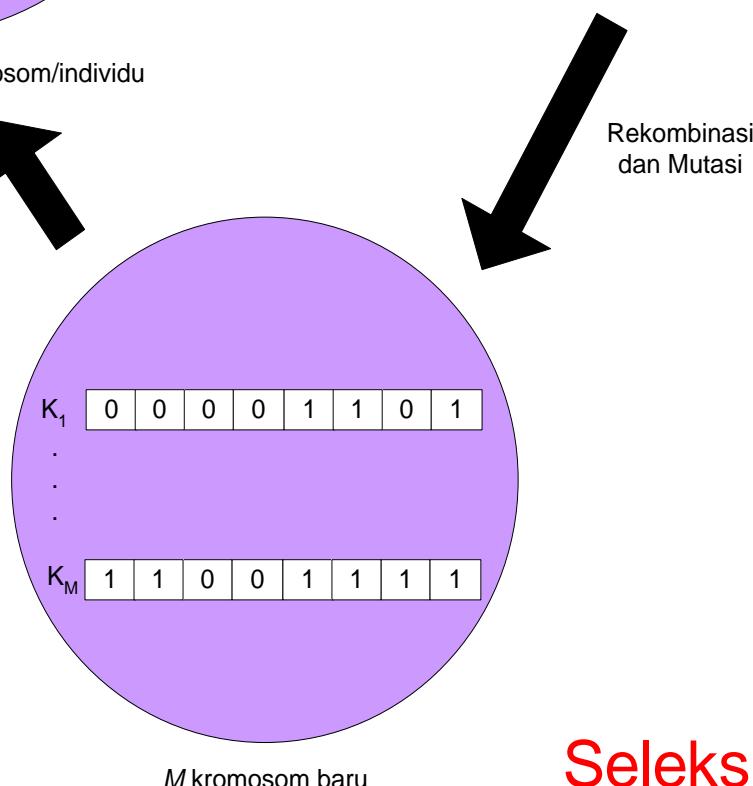
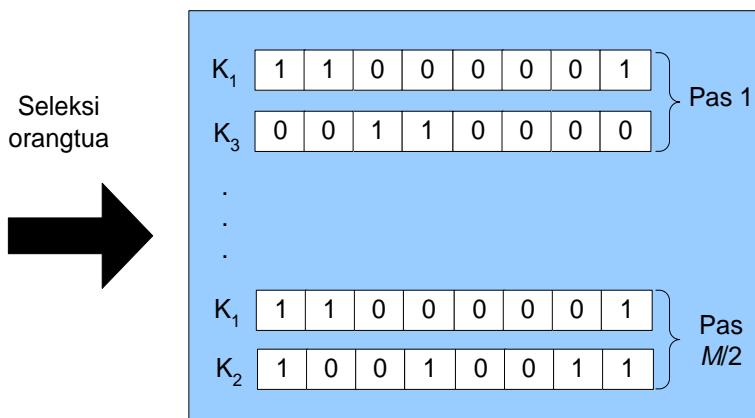
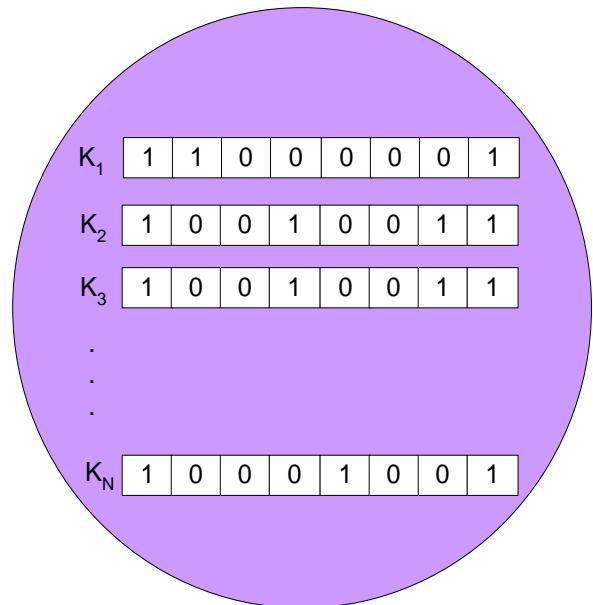
Seleksi Survivor

- Generational Model

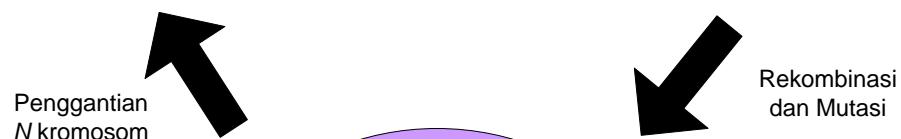
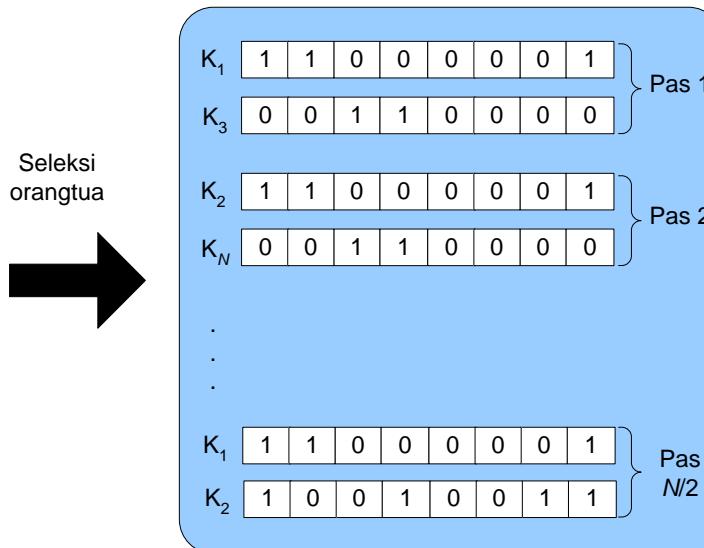
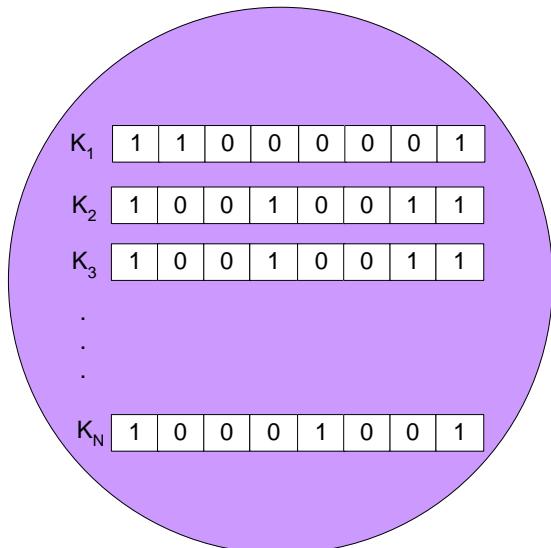
Suatu populasi berukuran N kromosom/individu pada suatu generasi diganti dengan N individu baru pada generasi berikutnya. Untuk menjaga kromosom terbaik, digunakan Elitisme.

- Steady State Model

Pada model ini, tidak semua kromosom diganti. Penggantian dilakukan hanya pada sejumlah kromosom tertentu, misal M . Dimana $M < N$.

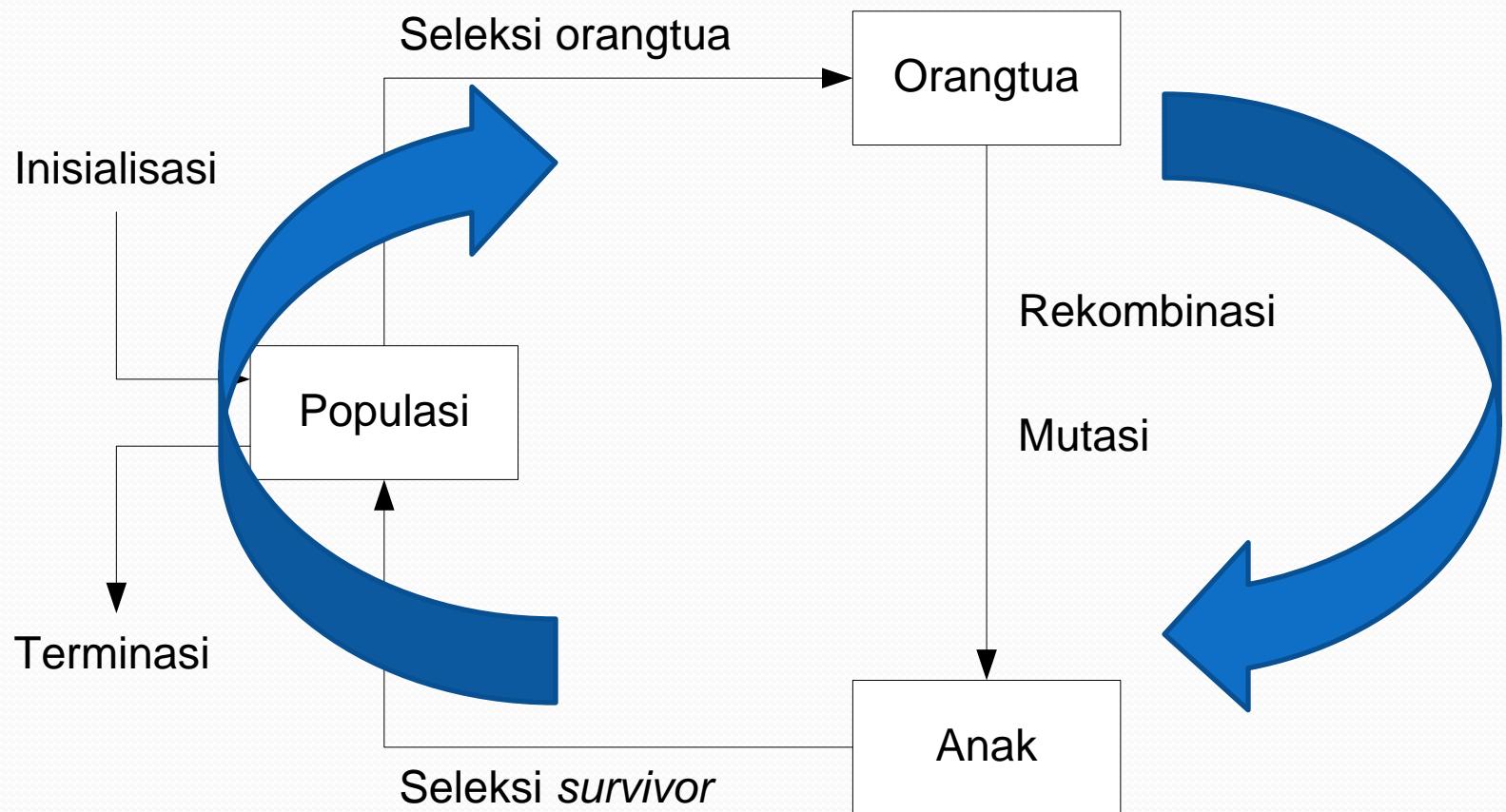


Seleksi Survivor: Steady State



Seleksi Survivor: *Generational*

Skema umum EAs



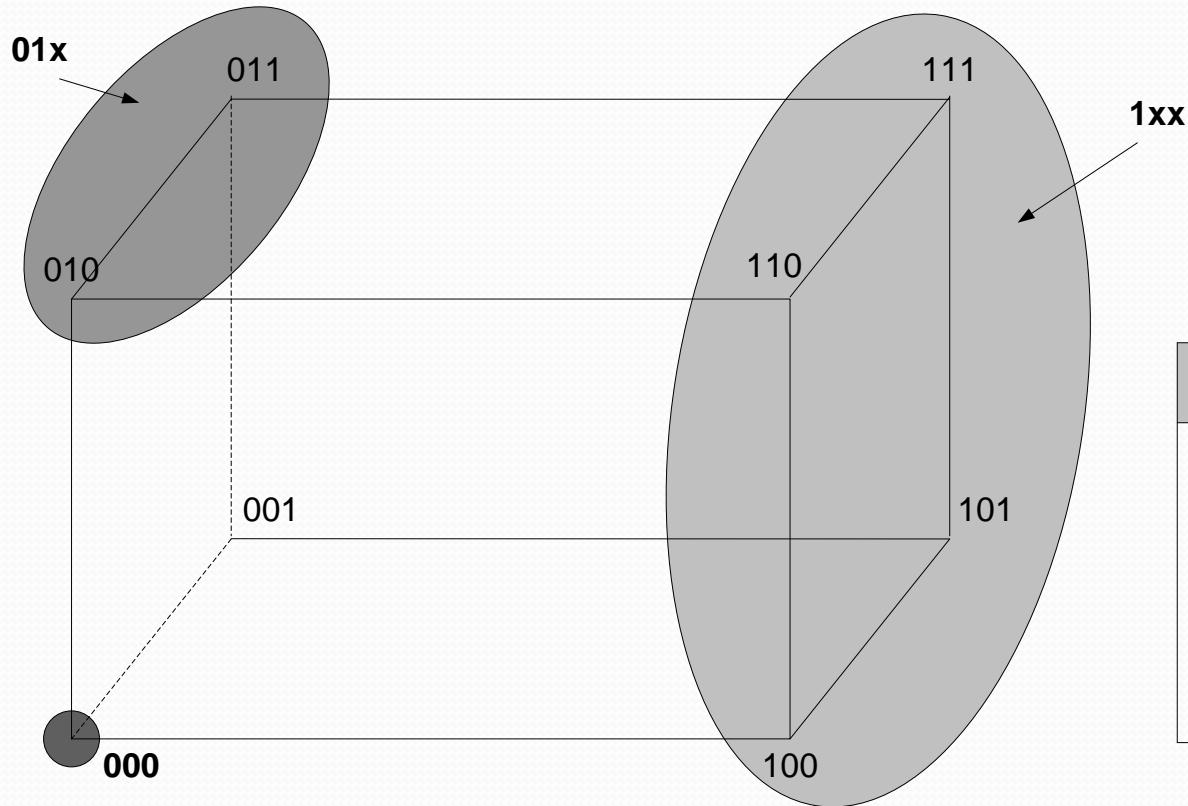
Pembuktian GA Secara Matematis

- *Schema Theorem*
- Pengaruh seleksi orangtua
- Pengaruh rekombinasi
- Pengaruh mutasi
- Pengaruh rekombinasi dan mutasi

Schema Theorem

- *Schema 1xx0* → kromosom 1000, 1010, 1100, 1110
- *Schema $S_1 = 1x1xx0xx$* memiliki
 - ***defining length*** (jarak antara simbol bukan x yang pertama dan terakhir) $D(S_1) = 6 - 1 = 5$.
 - ***order*** (jumlah simbol bukan x) $o(S_1) = 3$.

Schema Theorem



Schema S , Order O , dan Defining Length D

S	$O(S)$	$D(S)$
000	3	2
$01x$	2	1
$1xx$	1	0

Pengaruh seleksi orang tua

- Misalkan $\bar{f}(S)$ menyatakan rata-rata *fitness* dari suatu *schema* S dalam suatu populasi, yang didefinisikan sebagai rata-rata *fitness* dari semua kromosom yang termasuk dalam *schema* S tersebut.
- Dengan menggunakan seleksi orang tua yang proporsional terhadap nilai fitnessnya, maka probabilitas terpilihnya suatu kromosom dengan *fitness* f_i adalah f_i / f (dimana $f = \sum_{i=1}^N f_i$ adalah total *fitness* dari semua N kromosom dalam populasi tersebut)

Pengaruh seleksi orang tua

- Selanjutnya, misalkan \bar{f} menyatakan rata-rata *fitness* dalam populasi dengan N kromosom, yaitu

$$\bar{f} = f / N$$

- Banyaknya kopi dari *schema S* yang diharapkan pada generasi berikutnya ($g+1$) adalah

$$\Gamma(S, g + 1) = N \frac{\bar{f}(S)\Gamma(S, g)}{f}$$

dimana $\Gamma(S, g)$ menyatakan jumlah kopi dari *schema S* pada generasi g .

Pengaruh seleksi orang tua

- Dengan menggunakan fakta bahwa $N/f = 1/\bar{f}$, maka diperoleh persamaan berikut ini

$$\Gamma(S, g + 1) = \frac{\bar{f}(S)}{\bar{f}} \Gamma(S, g)$$

Pengaruh seleksi orang tua

- Jika suatu *schema* dihubungkan secara konsisten dengan rata-rata *fitness*-nya, yakni $\bar{f}(S)/\bar{f} = 1 + \alpha > 1$, dimana α adalah konstanta, maka jumlah kopi dari *schema* saat ini yang berada dalam populasi akan tumbuh secara **eksponensial** berdasarkan waktu k .

$$\Gamma(S, g+k) = \Gamma(S, g)(1+\alpha)^k$$

	g1	g2	g3	g4	g5	g6	g7	g8	g9	g10	Fitness
k1	1	1	1	1	1	0	0	0	0	0	22
k2	1	1	1	1	1	0	0	0	0	1	18
.	0	1	1	1	1	1	1	0	0	0	15
.	0	0	0	0	0	0	0	0	0	1	15
.	0	0	0	1	1	1	0	1	1	1	10
.	0	0	1	1	1	1	0	0	0	1	10
.	1	0	0	0	1	1	1	1	1	0	5
k8	1	0	0	0	1	1	1	1	0	0	5

$$\bar{f}(S) = \frac{22 + 18}{2} = 20$$

$$\bar{f} = \frac{100}{8} = 12,5$$

$$\Gamma(S, g+1) = \frac{\bar{f}(S)}{\bar{f}} \Gamma(S, g)$$

$$\bar{f}(S) / \bar{f} = 1 + \alpha > 1$$

S

	g1	g2	g3	g4	g5	g6	g7	g8	g9	g10	Fitness
k1	1	1	1	1	1	0	0	0	0	0	40
k2	1	1	1	1	1	0	0	0	0	1	20
.	0	1	1	1	1	1	1	0	0	0	15
.	0	0	0	0	0	0	0	0	0	1	15
.	0	0	0	1	1	1	0	1	1	1	10
.	0	0	1	1	1	1	0	0	0	1	10
.	1	0	0	0	1	1	1	1	0	0	5
k8	1	1	1	0	0	0	1	1	1	0	5

} s

$$\bar{f}(S) = \frac{40 + 20}{2} = 30$$

$$\bar{f} = \frac{120}{8} = 15$$

}

$$\Gamma(S, g+1) = \frac{\bar{f}(S)}{\bar{f}} \Gamma(S, g)$$

$$\bar{f}(S) / \bar{f} = 1 + \alpha > 1$$

Pengaruh seleksi orang tua

- *Schema* yang rata-rata *fitness*-nya **lebih tinggi** dibandingkan rata-rata *fitness* semua individu akan **naik secara eksponensial** berdasarkan waktu k .
- Jumlah *schema* yang rata-rata *fitness*-nya **lebih rendah** dibandingkan rata-rata *fitness* semua individu akan **menurun secara eksponensial juga**.

$$\Gamma(S, g + 1) = \frac{\bar{f}(S)}{\bar{f}} \Gamma(S, g)$$

$$\Gamma(S, g + k) = \Gamma(S, g)(1 + \alpha)^k$$

Pengaruh Rekombinasi

Probabilitas perusakan, selama rekombinasi, terhadap *schema* dengan *defining length* $D(S)$ adalah:

$$p_d = \frac{D(S)}{n - 1}$$

dimana n adalah panjang kromosom.

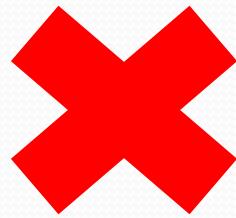
Semakin kecil $D(S)$ semakin kecil pula p_d .

Pengaruh Rekombinasi

Karena rekombinasi terjadi dengan probabilitas p_c maka *schema S* akan bertahan hidup dengan probabilitas sekitar

Yang di buku salah

$$\frac{1 - p_c D(S)}{(n - 1)}$$



$$1 - p_c \frac{D(S)}{(n - 1)}$$



dimana n adalah panjang kromosom.

Pengaruh Mutasi

Probabilitas bahwa *schema S* tidak akan termutasi pada gen-gen yang bukan x adalah sama dengan

$$(1 - p_m)^{O(S)}$$

dimana $O(S)$ adalah *order* dari *schema S* (jumlah simbol bukan x dari *schema S*).

Pengaruh rekombinasi dan mutasi

Dengan memasukkan pengaruh rekombinasi dan mutasi, jumlah kopi dari *schema S* pada generasi $g+1$ adalah

$$\Gamma(S, g+1) \geq \frac{\bar{f}(S)}{\bar{f}} \Gamma(S, g) \left(1 - p_c \frac{D(S)}{n-1}\right) (1 - p_m)^{O(S)}$$

$$\approx \frac{\bar{f}(S)}{\bar{f}} \Gamma(S, g) \left(1 - p_c \frac{D(S)}{n-1}\right) (1 - O(S)p_m)$$

$$\approx \frac{\bar{f}(S)}{\bar{f}} \Gamma(S, g) \left(1 - p_c \frac{D(S)}{n-1} - O(S)p_m\right)$$

Studi kasus: Minimasi fungsi

Nilai minimum $h = ?$

$$h(x_1, x_2) = x_1^2 + x_2^2$$

$$x_1, x_2 \in [-5,12;5,12]$$

Individu

x ₁										x ₂									
1	1	0	1	0	1	0	1	0	0	0	1	0	0	1	1	0	1	0	1
g_1										g_{10}	g_{11}							g_{20}	

Fitness

$$f = \frac{1}{(x_1^2 + x_2^2) + 0,01}$$

Jika nilai minimum = 0, nilai maks $f = ?$

Generasi 1

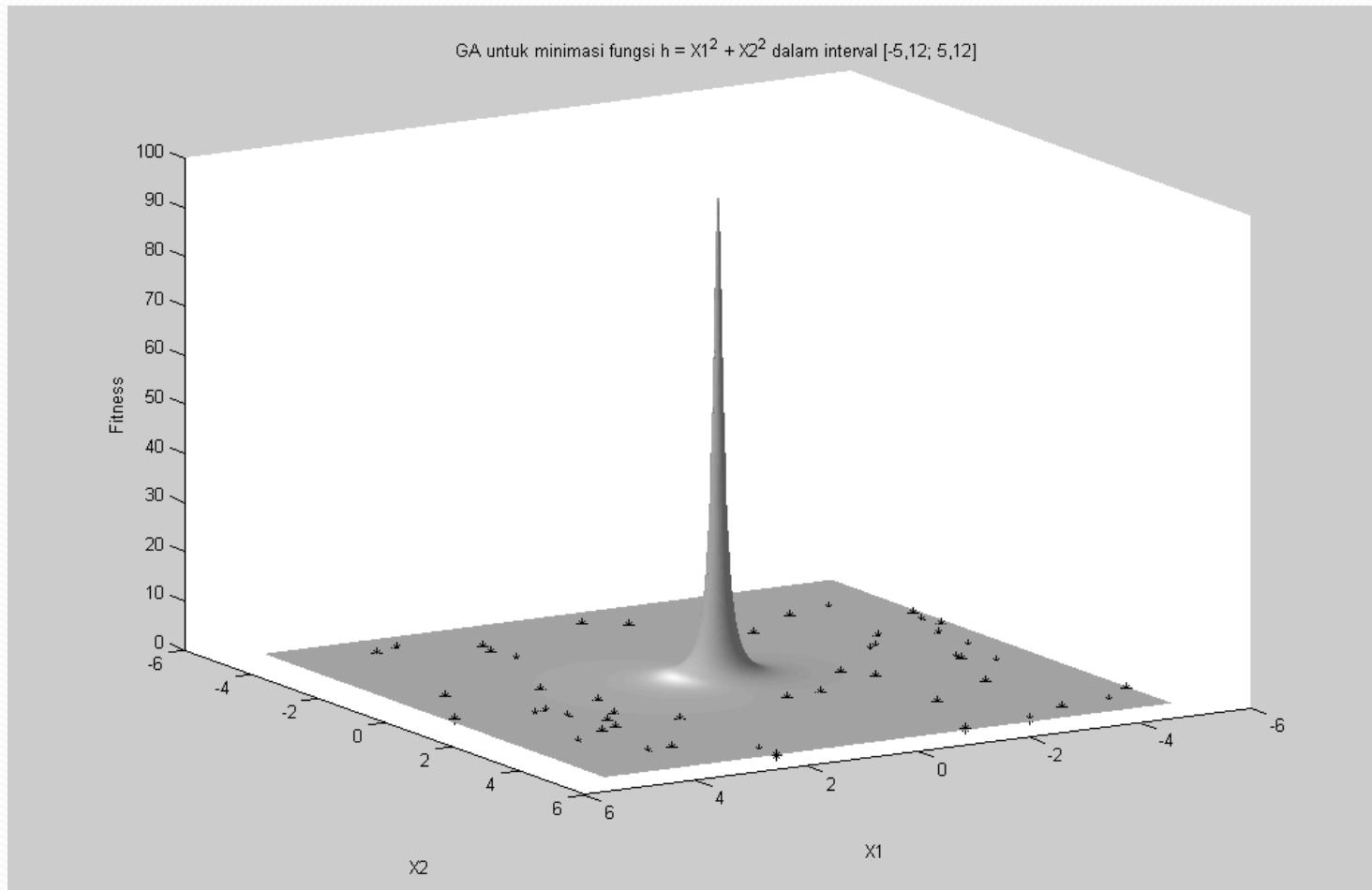
$$x = r_b + \frac{(r_a - r_b)}{\sum_{i=1}^N 2^{-i}} (g_1 \cdot 2^{-1} + g_2 \cdot 2^{-2} + \dots + g_N \cdot 2^{-N})$$

$$x_1, x_2 \in [-5,12; 5,12]$$



No	Genotype	Phenotype		Nilai fitness
	kromosom biner	X1	X2	
1	00010011011001101110	-4.35	1.1	0.049646
2	11001101110001000011	3.11	-4.45	0.033916
3	10110010111111001110	2.03	4.62	0.039254
4	11001110001101111101	3.12	3.81	0.041219
5	11001110101011011001	3.14	2.17	0.068594
6	00101110000110110110	-3.28	-0.74	0.08837
7	01111011111010110010	-0.17	1.78	0.31179
50	11010110011000111011	3.45	0.59	0.081562

Generasi 1



Generasi 10

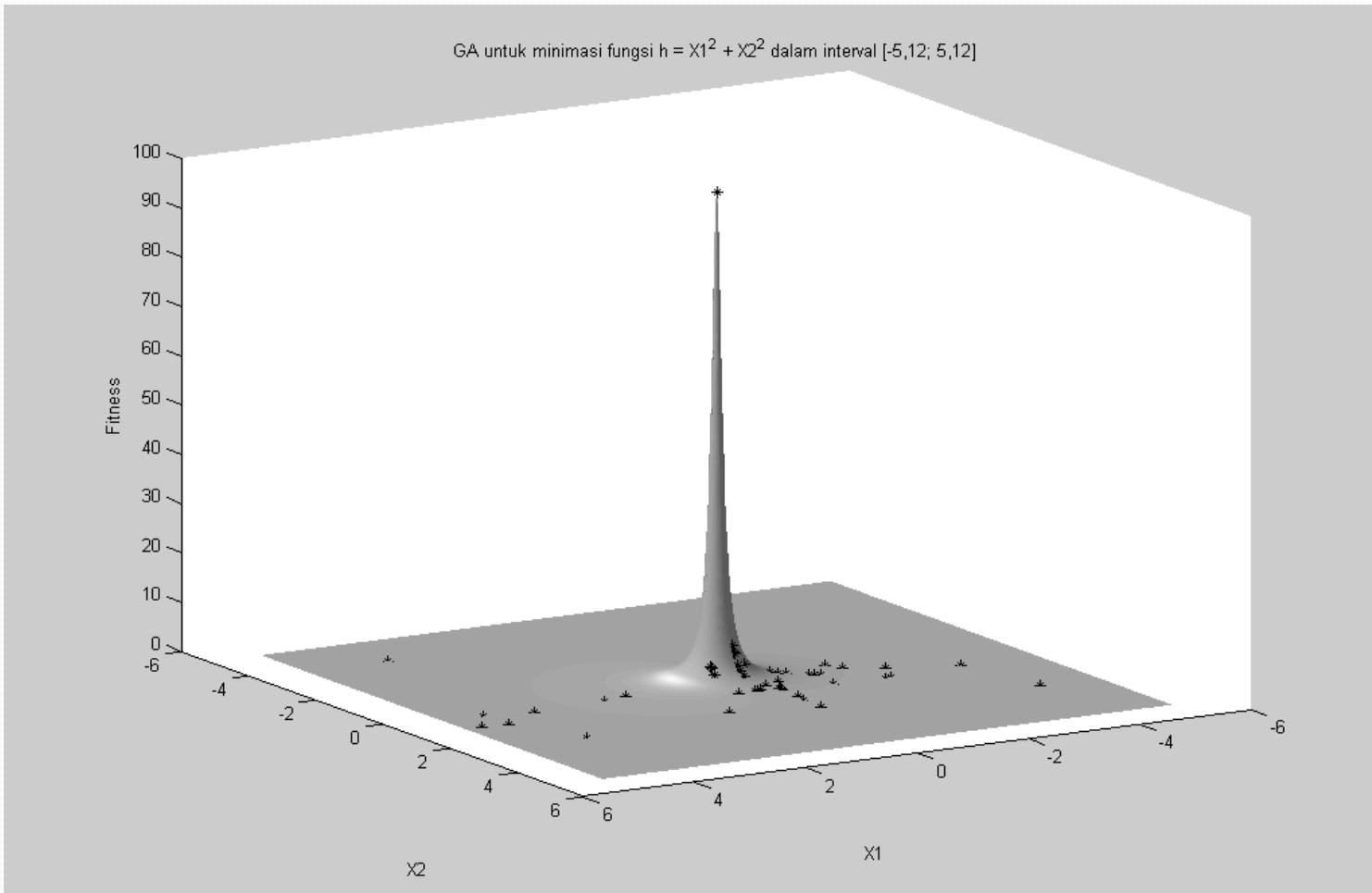
$$x = r_b + \frac{(r_a - r_b)}{\sum_{i=1}^N 2^{-i}} (g_1 \cdot 2^{-1} + g_2 \cdot 2^{-2} + \dots + g_N \cdot 2^{-N})$$

$$x_1, x_2 \in [-5,12; 5,12]$$

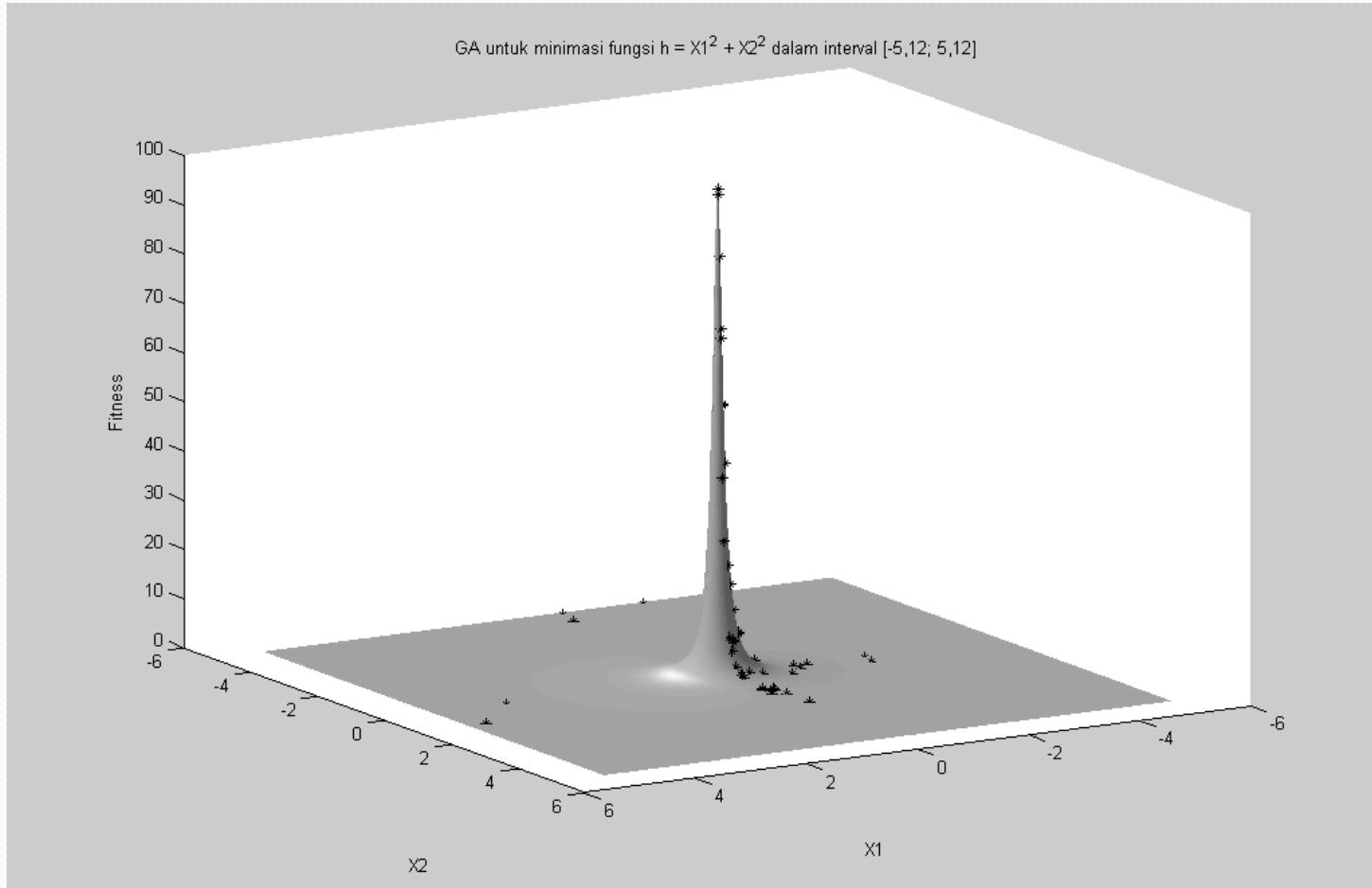


No	Genotype	Phenotype		Nilai fitness
	kromosom biner	X1	X2	
1	01111111110000000000	-0.01	0	99.01
2	01111111110000000000	-0.01	0	99.01
3	01111101010001000001	-3.77	1.03	0.065429
4	01111101011001110001	-2.4	1.2	0.1387
5	01111001111000100001	3.58	0.52	0.076355
6	01011101111000101010	4.83	1.01	0.041053
7	01111101111000100001	-1.38	1.2	0.29812
...				
50	01111101111000000001	-1.93	0.02	0.26772

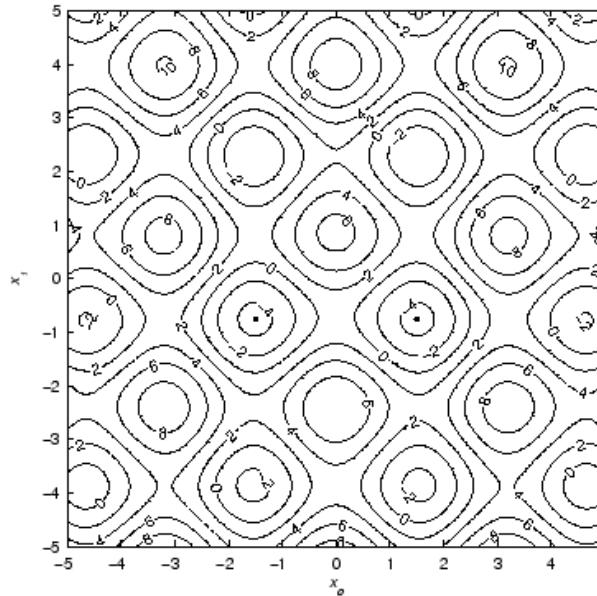
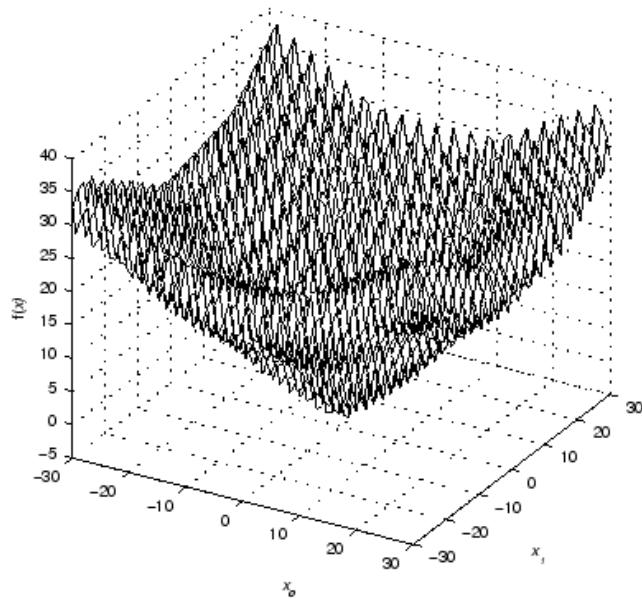
Generasi 10



Generasi 100



$$f(\vec{x}) = \sum_{i=0}^{D-1} \left(e^{-0.2} \sqrt{x_i^2 + x_{i+1}^2} + 3(\cos(2x_i) + \sin(2x_{i+1})) \right)$$



- Untuk presisi 10^{-9} → Berapa bit?
- Bisa menggunakan kromosom Real?

Setting parameter GA

- Tidak ada panduan yang pasti
- Hanya dengan Intuisi
- Umumnya:
 - Representasi kromosom = biner/integer/real/permutasi
 - Jumlah bit per variabel → presisi yang diinginkan
 - Ukuran Populasi = **50 - 100**
 - Probabilitas Crossover (Pc) = **0,8**
 - Probabilitas Mutasi (Pm) = **$1/NL$ sampai $1/L$**
 N = Ukuran Populasi
 L = Panjang Kromosom (Jumlah Gen)

Observasi parameter GA

- Minimasi fungsi $h = x_1^2 + x_2^2$, x_1 dan x_2 elemen $[-10, 10]$
- Fitness = $1/(x_1^2 + x_2^2 + 0.001)$

- Ukuran Populasi = [50 100 200]
- Jumlah bit = [10 50 90]
- Prob Rekombinasi = [0.5 0.7 0.9]
- Prob Mutasi = [0.5/JumGen 1/JumGen 2/JumGen]

- Jumlah Individu maksimum = 20000 (*fairness*)
- Jumlah running/percobaan = 30 (*valid*)

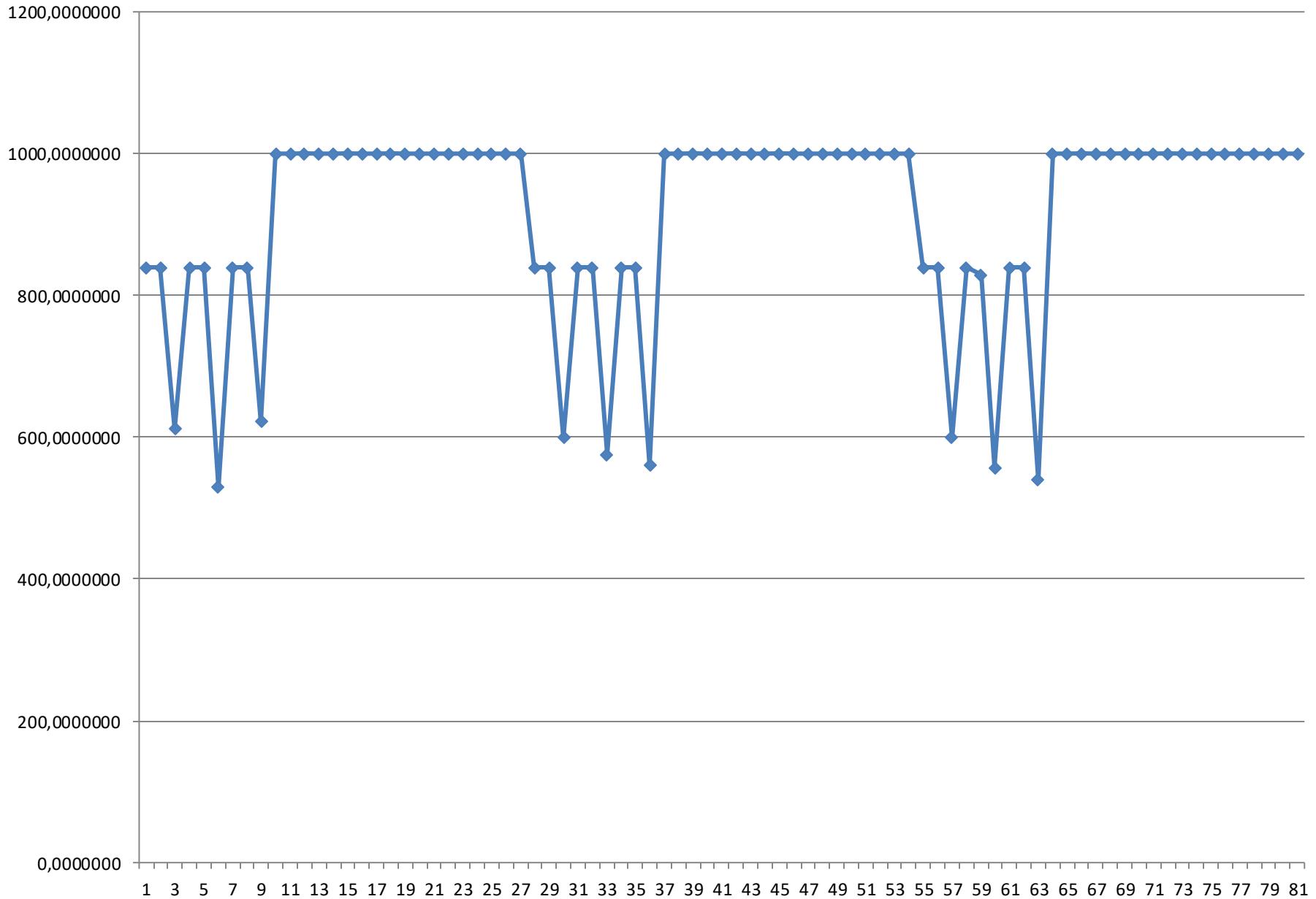
No Observasi	Ukuran populasi	Jumlah bit	Probabilitas Pdh Silang	Probabilitas Mutasi	Rata-rata Fitness terbaik	Rata-rata Jml Individu yang dievaluasi
1	50	10	0,5	0,0250	839,5544749	20000,0000
2	50	10	0,5	0,0500	839,5544749	20000,0000
3	50	10	0,5	0,1000	611,0770624	20000,0000
4	50	10	0,7	0,0250	839,5544749	20000,0000
5	50	10	0,7	0,0500	839,5544749	20000,0000
6	50	10	0,7	0,1000	528,7161733	20000,0000
7	50	10	0,9	0,0250	839,5544749	20000,0000
8	50	10	0,9	0,0500	839,5544749	20000,0000
9	50	10	0,9	0,1000	622,2201392	20000,0000
10	50	50	0,5	0,0050	1000,0000000	8301,6667
11	50	50	0,5	0,0100	1000,0000000	20000,0000
12	50	50	0,5	0,0200	999,9987777	20000,0000
13	50	50	0,7	0,0050	1000,0000000	8013,3333
14	50	50	0,7	0,0100	1000,0000000	20000,0000
15	50	50	0,7	0,0200	999,9982015	20000,0000
16	50	50	0,9	0,0050	1000,0000000	8133,3333
17	50	50	0,9	0,0100	1000,0000000	20000,0000
18	50	50	0,9	0,0200	999,9988782	20000,0000
19	50	90	0,5	0,0028	1000,0000000	8361,6667
20	50	90	0,5	0,0056	1000,0000000	8796,6667
21	50	90	0,5	0,0111	1000,0000000	20000,0000
22	50	90	0,7	0,0028	1000,0000000	8151,6667
23	50	90	0,7	0,0056	1000,0000000	8780,0000
24	50	90	0,7	0,0111	1000,0000000	20000,0000
25	50	90	0,9	0,0028	1000,0000000	7538,3333
26	50	90	0,9	0,0056	1000,0000000	8995,0000
27	50	90	0,9	0,0111	1000,0000000	20000,0000
28	100	10	0,5	0,0250	839,5544749	20000,0000
29	100	10	0,5	0,0500	839,5544749	20000,0000
30	100	10	0,5	0,1000	599,4452769	20000,0000

Paket parameter terbaik untuk kasus di atas

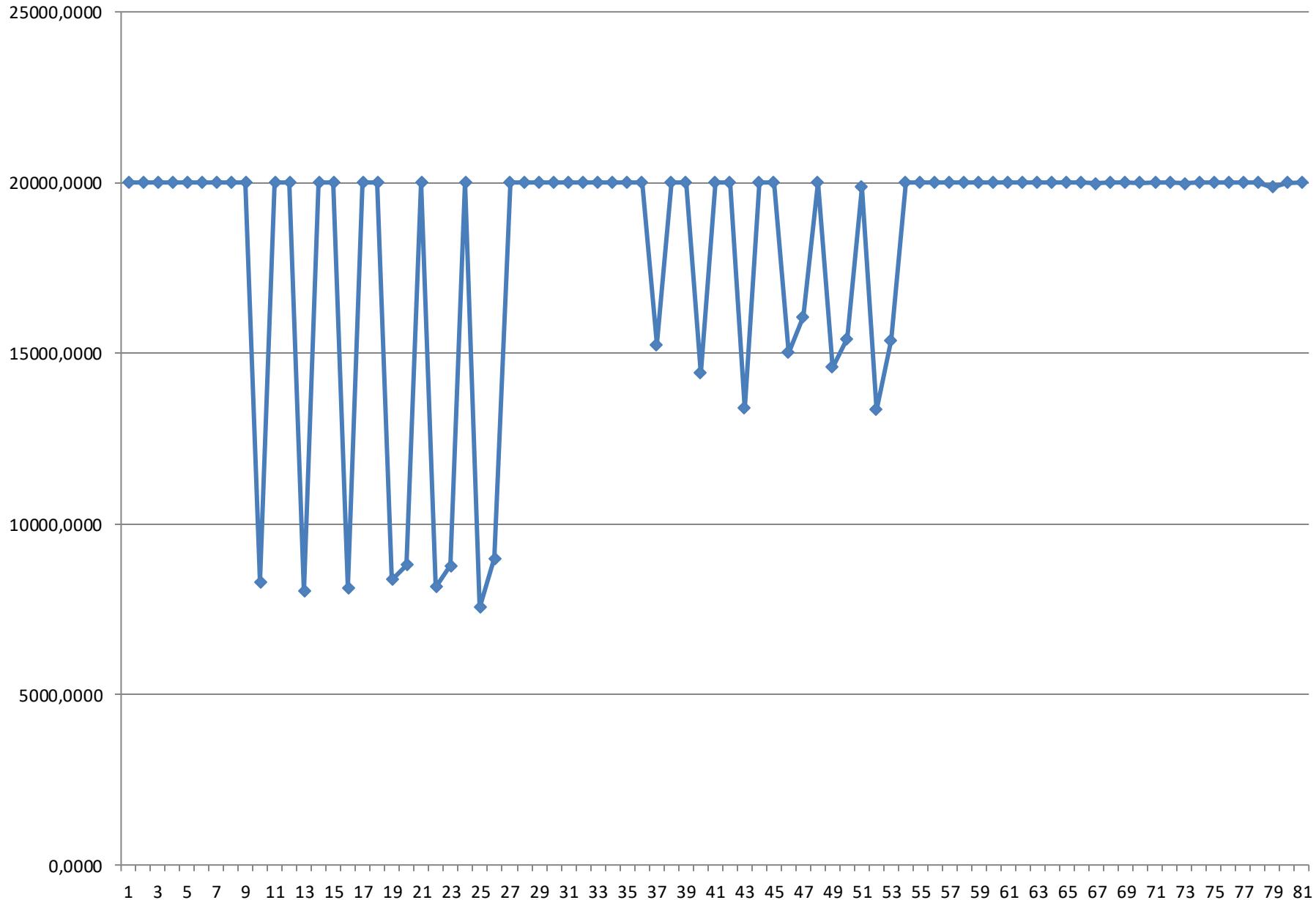
No Observasi	Ukuran populasi	Jumlah bit	Probabilitas Pdh Silang	Probabilitas Mutasi	Rata-rata Fitness terbaik	Rata-rata Jml Individu yang dievaluasi
31	100	10	0,7	0,0250	839,5544749	20000,0000
32	100	10	0,7	0,0500	839,5544749	20000,0000
33	100	10	0,7	0,1000	575,8472869	20000,0000
34	100	10	0,9	0,0250	839,5544749	20000,0000
35	100	10	0,9	0,0500	839,5544749	20000,0000
36	100	10	0,9	0,1000	559,6804844	20000,0000
37	100	50	0,5	0,0050	1000,0000000	15246,6667
38	100	50	0,5	0,0100	1000,0000000	20000,0000
39	100	50	0,5	0,0200	999,9986429	20000,0000
40	100	50	0,7	0,0050	1000,0000000	14416,6667
41	100	50	0,7	0,0100	1000,0000000	20000,0000
42	100	50	0,7	0,0200	999,9988459	20000,0000
43	100	50	0,9	0,0050	1000,0000000	13390,0000
44	100	50	0,9	0,0100	1000,0000000	20000,0000
45	100	50	0,9	0,0200	999,9987118	20000,0000
46	100	90	0,5	0,0028	1000,0000000	15010,0000
47	100	90	0,5	0,0056	1000,0000000	16056,6667
48	100	90	0,5	0,0111	1000,0000000	20000,0000
49	100	90	0,7	0,0028	1000,0000000	14580,0000
50	100	90	0,7	0,0056	1000,0000000	15430,0000
51	100	90	0,7	0,0111	1000,0000000	19860,0000
52	100	90	0,9	0,0028	1000,0000000	13346,6667
53	100	90	0,9	0,0056	1000,0000000	15390,0000
54	100	90	0,9	0,0111	1000,0000000	20000,0000
55	200	10	0,5	0,0250	839,5544749	20000,0000
56	200	10	0,5	0,0500	839,5544749	20000,0000
57	200	10	0,5	0,1000	599,0108676	20000,0000
58	200	10	0,7	0,0250	839,5544749	20000,0000
59	200	10	0,7	0,0500	828,6149185	20000,0000
60	200	10	0,7	0,1000	557,3828866	20000,0000

No Observasi	Ukuran populasi	Jumlah bit	Probabilitas Pdh Silang	Probabilitas Mutasi	Rata-rata Fitness terbaik	Rata-rata Jml Individu yang dievaluasi
61	200	10	0,9	0,0250	839,5544749	20000,0000
62	200	10	0,9	0,0500	839,5544749	20000,0000
63	200	10	0,9	0,1000	539,1055371	20000,0000
64	200	50	0,5	0,0050	1000,0000000	20000,0000
65	200	50	0,5	0,0100	999,9999995	20000,0000
66	200	50	0,5	0,0200	999,9986789	20000,0000
67	200	50	0,7	0,0050	1000,0000000	19966,6667
68	200	50	0,7	0,0100	999,9999997	20000,0000
69	200	50	0,7	0,0200	999,9947933	20000,0000
70	200	50	0,9	0,0050	1000,0000000	19986,6667
71	200	50	0,9	0,0100	999,9999996	20000,0000
72	200	50	0,9	0,0200	999,9939550	20000,0000
73	200	90	0,5	0,0028	999,9999988	19966,6667
74	200	90	0,5	0,0056	999,9999999	20000,0000
75	200	90	0,5	0,0111	999,9999976	20000,0000
76	200	90	0,7	0,0028	999,9999995	20000,0000
77	200	90	0,7	0,0056	1000,0000000	20000,0000
78	200	90	0,7	0,0111	999,9999979	20000,0000
79	200	90	0,9	0,0028	1000,0000000	19866,6667
80	200	90	0,9	0,0056	1000,0000000	19993,3333
81	200	90	0,9	0,0111	999,9999988	20000,0000

Rata-rata Fitness Terbaik



Rata-rata Jumlah Individu yg Dievaluasi



TSP

Masalah TSP dengan 20 kota. Terdapat **121.645 trilyun** solusi yang mungkin.

Solusi 1	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Solusi 2	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	20	19
•																				
•																				
•																				
121.645 trilyun																				
20 2 1 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19																				

Penyelesaian berbasis **EA** dengan populasi 100 individu

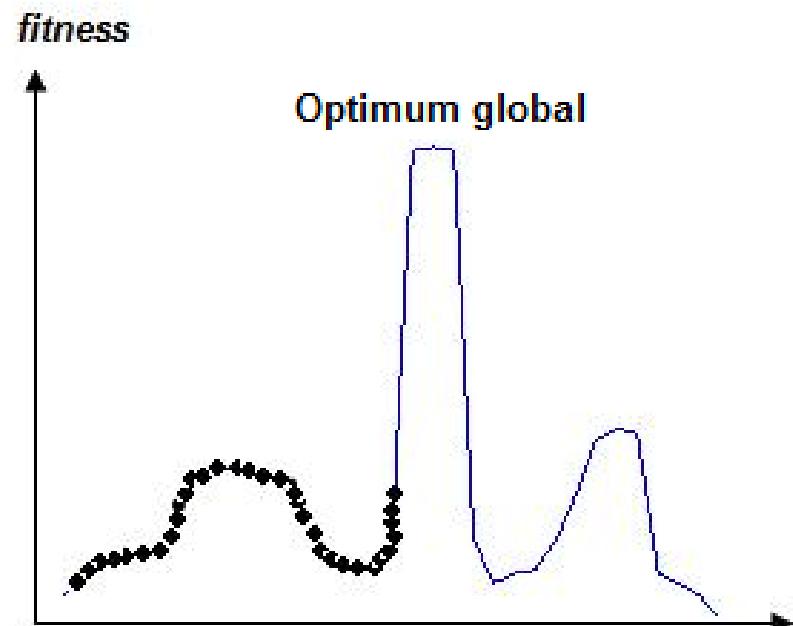
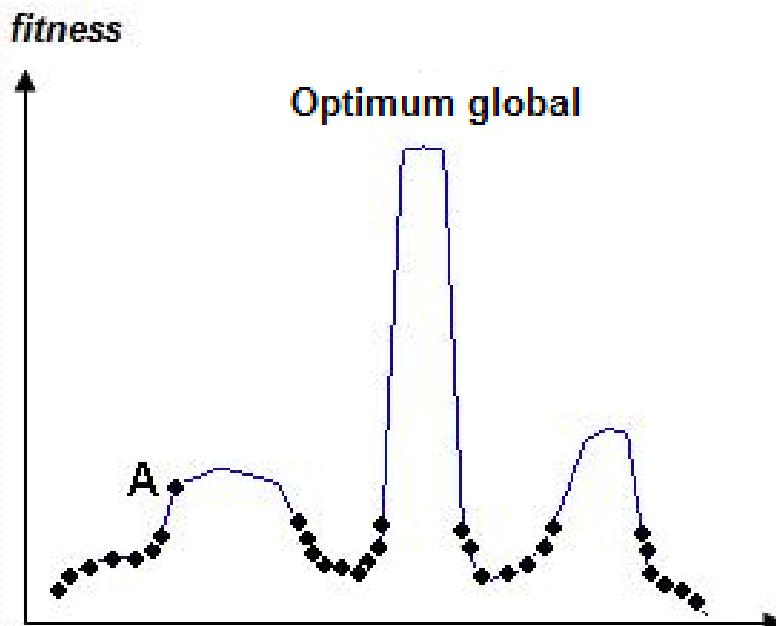
Individu 1	10	2	3	4	5	6	7	8	1	9	11	12	13	14	15	16	17	18	19	20
Individu 2	14	5	8	16	2	17	9	3	7	10	11	12	13	1	18	4	6	20	15	19
•																				
Individu 99	20	5	11	16	3	18	2	6	19	7	1	17	8	15	12	10	14	4	13	9
Individu 100	8	1	3	7	11	6	4	18	9	16	5	12	20	14	17	10	15	2	19	13

Dengan setting parameter yang tepat, GA bisa menemukan solusi secara cepat dan akurat.

Advanced GA

- Konvergensi Prematur (KP)
- *Gray Coding*
- *Messy Encoding*
- *Fitness Ranking*
- *Island Model*
- *Adaptive GA*
- *Grid-Based Crossover*
- *Grammatical Encoding*

Konvergensi Prematur (KP)



Pencegahan KP

- *Gray Coding*
- *Messy Encoding*
- *Fitness Ranking*
- *Island Model*
- *Adaptive GA*

Hamming Distance

Integer

14

0	1	1	1	0
---	---	---	---	---

Biner



Berbeda 1 bit

15

0	1	1	1	1
---	---	---	---	---



Berbeda 5 bit

16

1	0	0	0	0
---	---	---	---	---

Binary Coding

- Jika solusi maksimum yang dicari adalah **10000** (16)
- Individu terbaik saat ini **01111** (15)
- Sampai beberapa generasi berikutnya ternyata individu terbaik tetap **01111** (15).
- Mengapa?
- **01111** → **10000** memerlukan mutasi 5 gen.
- Padahal probabilitas mutasi biasanya dibuat sangat kecil, biasanya $1/NL$ sampai $1/L$, dimana N adalah ukuran populasi, L panjang kromosom.
- SOLUSINYA?

Gray Coding

Integer	0	1	2	3	4	5	6	7
Binary coding	000	001	010	011	100	101	110	111
Gray coding	000	001	011	010	110	111	101	100

Gray Coding

individu: $x_1 = 5$ dan $x_2 = 3$



x_1			x_2		
1	0	1	0	1	1

Binary coding



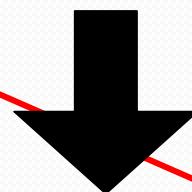
x_1			x_2		
1	1	1	0	1	0

Kromosom: *Gray coding*

Messy Encoding

Messy encoding:

3	0	8	0	2	1	6	1	1	0	4	1	5	1	7	1
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

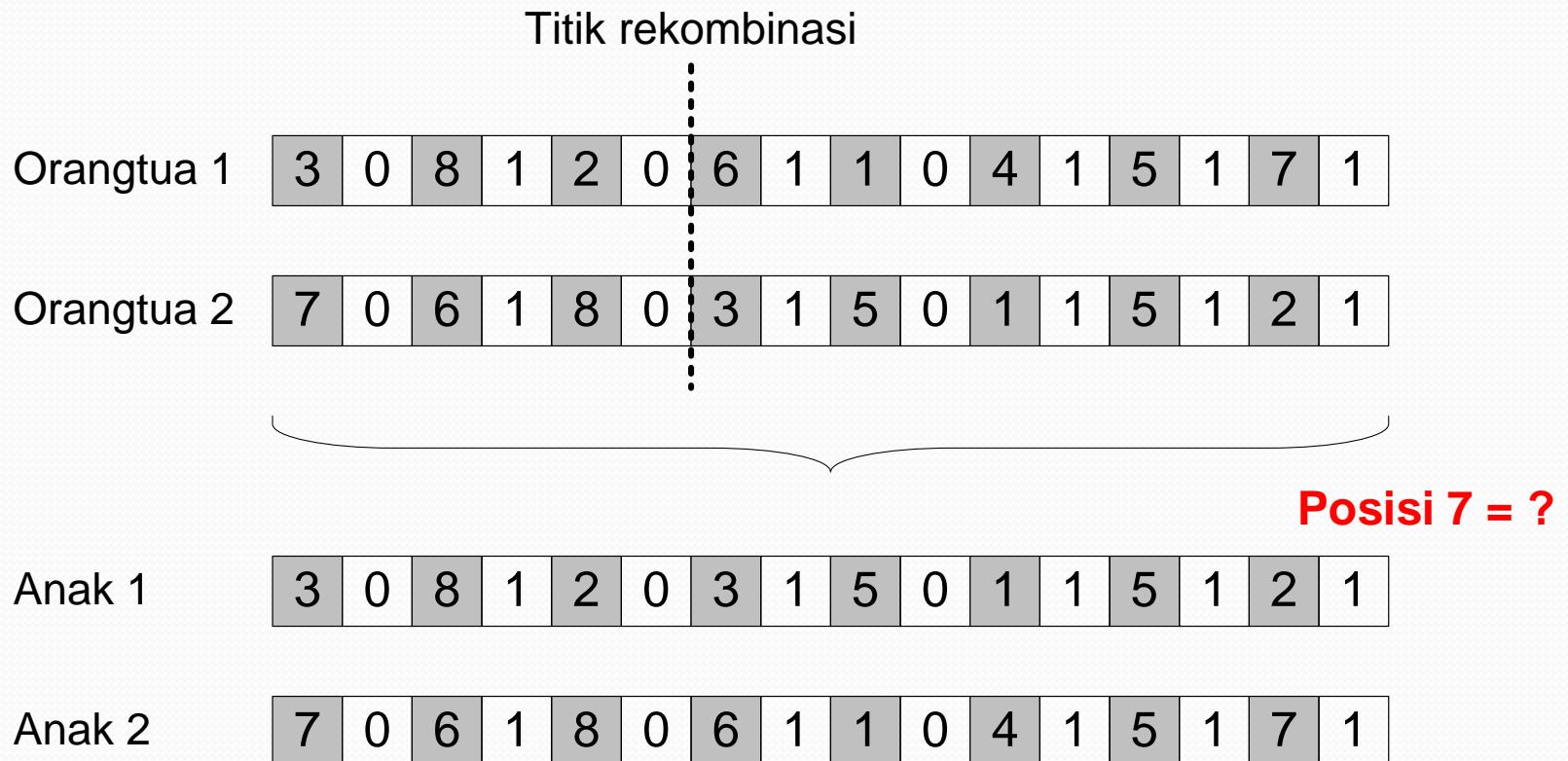


Binary encoding:

0	1	0	1	1	1	1	0
---	---	---	---	---	---	---	---

1 2 3 4 5 6 7 8

Crossover



Messy encoding:

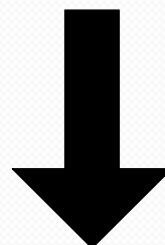
3	0	8	1	2	0	3	1	5	0	1	1	5	1	2	1
3	0	8	1	2	0	3	1	5	0	1	1	5	1	2	1

dipanjangkan
n kali semula

6	0	4	1	8	1	3	1	2	0	5	1	4	1	5	1
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

Binary encoding:

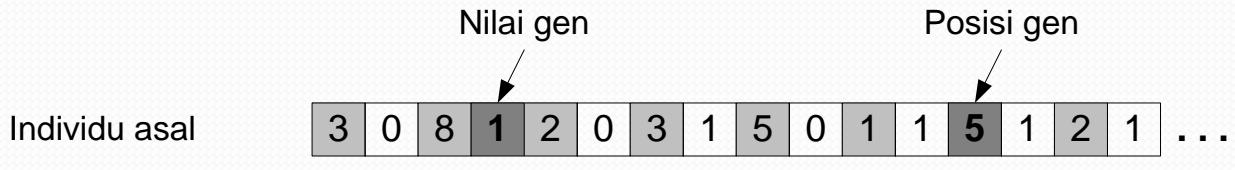
1	0	0	1	0	0	0	1
---	---	---	---	---	---	---	---



Nilai gen untuk posisi 7 = 0
(dibangkitkan secara acak)

Mutasi

Messy encoding:



Binary encoding:

1	0	0	1	1	0	0	1
---	---	---	---	---	---	---	---



1	0	0	1	1	0	1	0
---	---	---	---	---	---	---	---

Linear Fitness Ranking

Maksimasi h dimana x_1 dan x_2 adalah real $[-2, 2]$)?

$$h(x_1, x_2) = 100000 + 2x_1 + x_2$$

Linear Fitness Ranking

- Pada fungsi di atas, nilai-nilai h berada dalam interval 99994 sampai 100006.
- Dengan demikian, semua individu memiliki nilai *fitness* yang hampir sama dalam kisaran 100000.
- Hal ini akan berakibat buruk pada proses seleksi orangtua secara proporsional terhadap *fitness*-nya.
- Bagaimana Solusinya?

Linear Fitness Ranking

$$f_{LR}(i) = f_{\max} - (f_{\max} - f_{\min}) \left(\frac{R(i)-1}{N-1} \right)$$

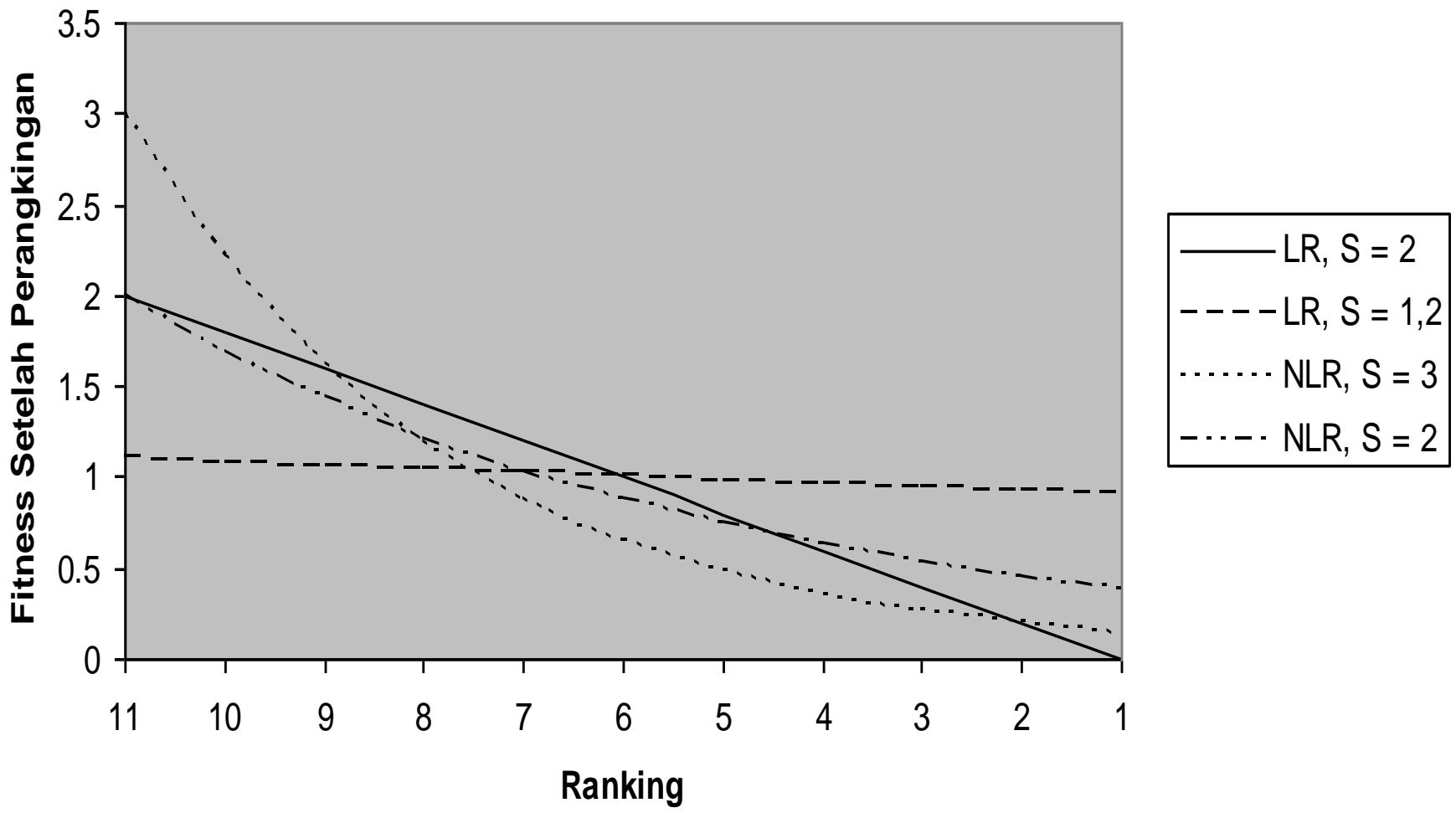
$f_{LR}(i) \rightarrow$ nilai *fitness* individu ke- i yang sudah diskalakan
 $N \rightarrow$ jumlah individu dalam populasi.

$R(i) \rightarrow$ ranking individu ke i .

$f_{\min} \rightarrow$ nilai *fitness* terkecil

$f_{\max} \rightarrow$ nilai *fitness* terbesar

Individu ke	<i>fitness</i>	Ranking $R(i)$	f_{LR}
1	100004,00	1	100004,00
2	100003,99	2	100003,56
3	100003,98	3	100003,12
4	100003,97	4	100002,68
5	100003,96	5	100002,26
6	100003,95	6	100001,82
7	100003,94	7	100001,38
8	100003,93	8	100000,84
9	100003,92	9	100000,44
10	100000,00	10	100000,00



<i>Fitness</i>	<i>Posisi</i>	<i>Fitness Perangkingan</i>			
		LR, S = 2	LR, S = 1,2	NLR, S = 3	NLR, S = 2
90,11	11	2	1,1	3	2
90,09	10	1,8	1,08	2,21	1,69
90,08	9	1,6	1,06	1,62	1,43
90,06	8	1,4	1,04	1,19	1,21
90,05	7	1,2	1,02	0,88	1,03
89,97	6	1	1	0,65	0,87
89,96	5	0,8	0,98	0,48	0,74
89,95	4	0,6	0,96	0,35	0,62
79,94	3	0,4	0,94	0,26	0,53
79,93	2	0,2	0,92	0,19	0,45
79,91	1	0	0,9	0,14	0,38

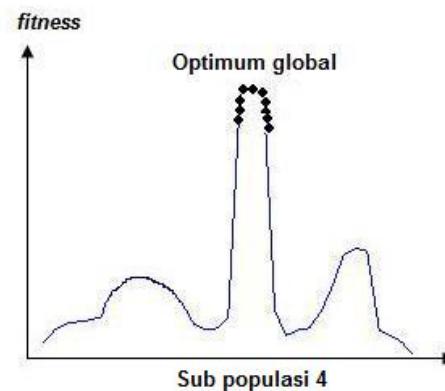
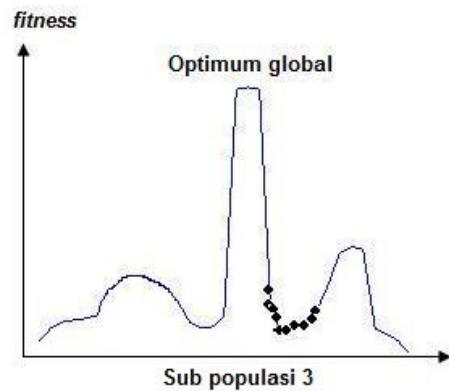
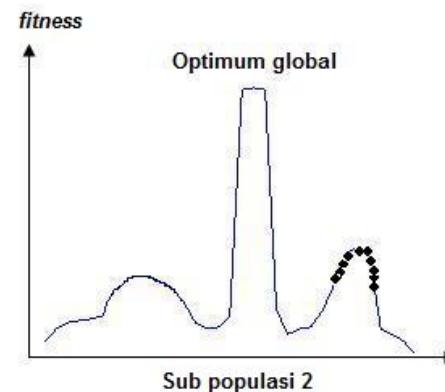
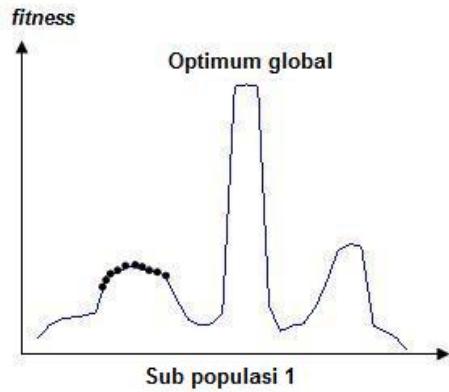
Island Models (Sub Population)

N kromosom dalam satu populasi dibagi menjadi N_k kelompok. Masing-masing kelompok berisi:

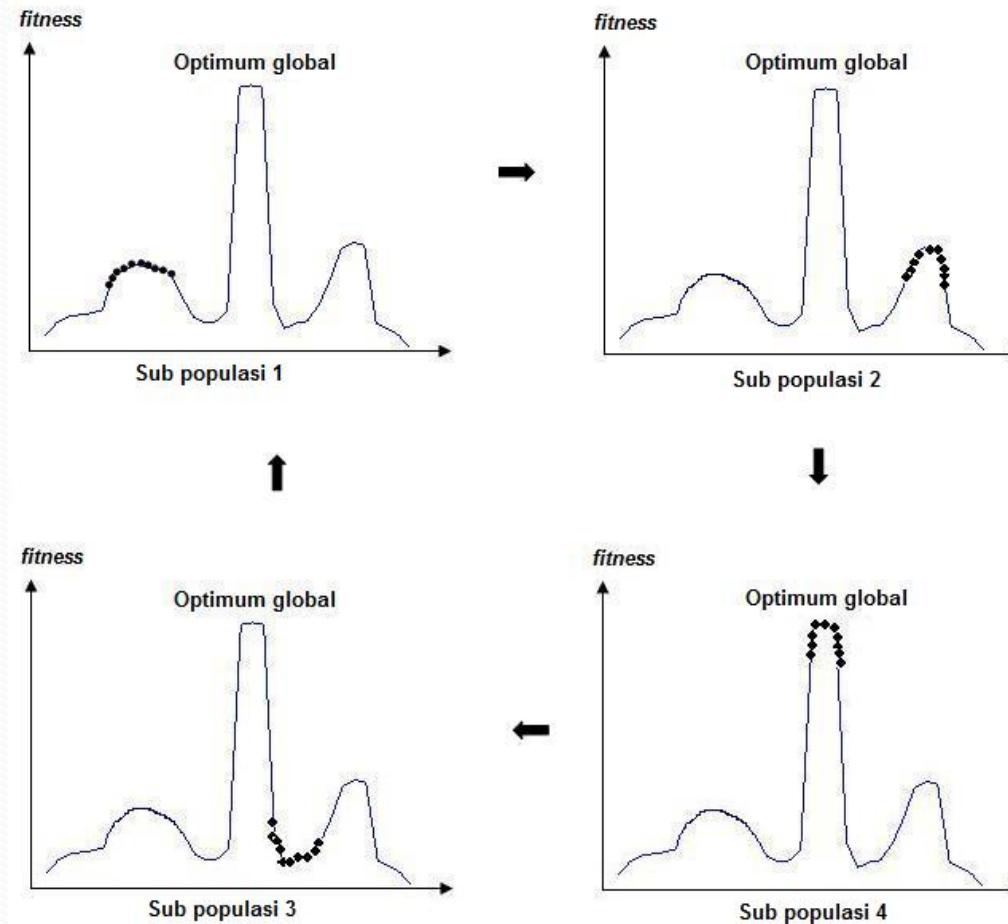
$$v = \frac{N}{N_k}$$

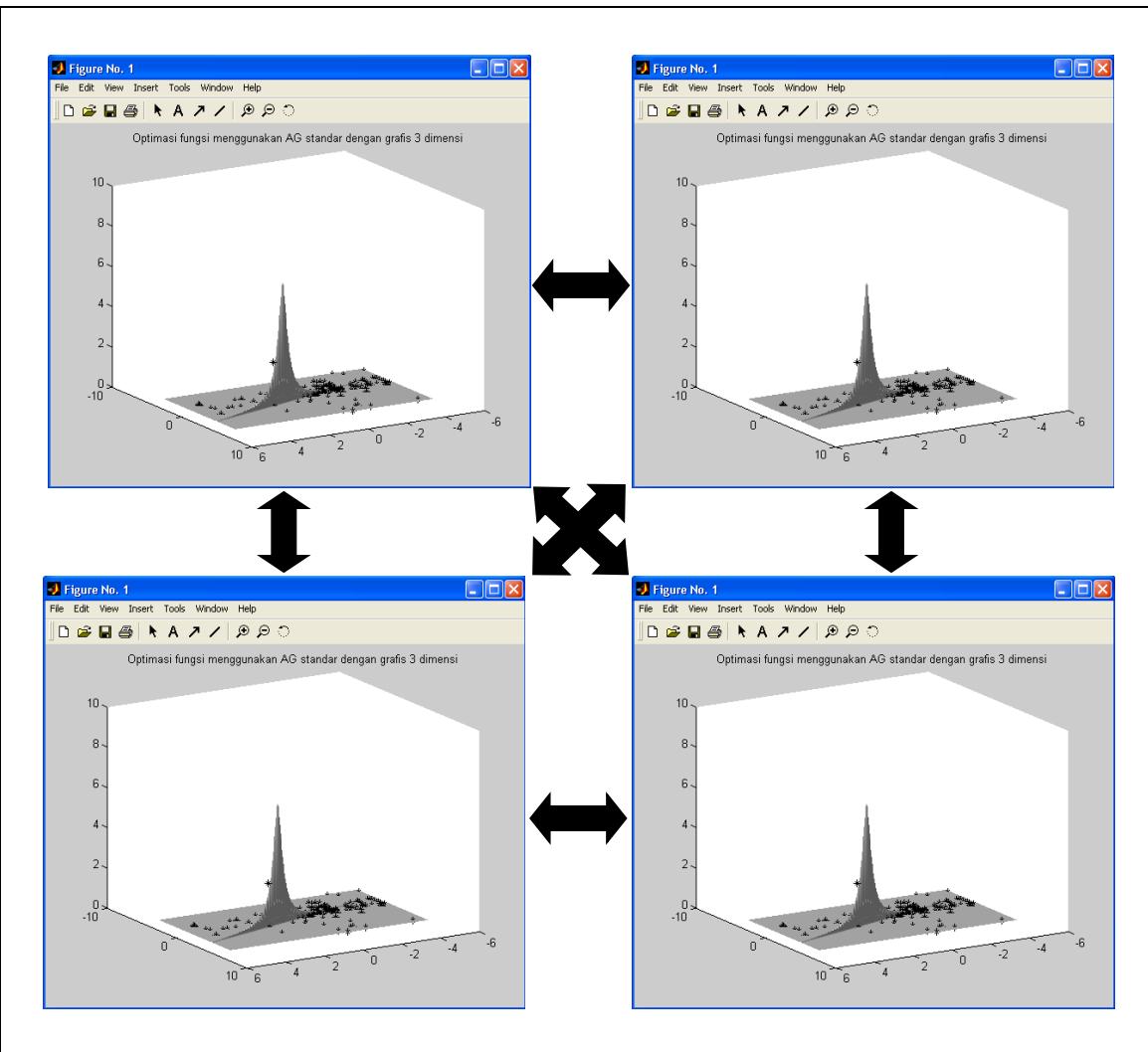
Suatu individu bisa dipindah ke sub populasi lain berdasarkan *tunneling probability* p_t

Island model EAs



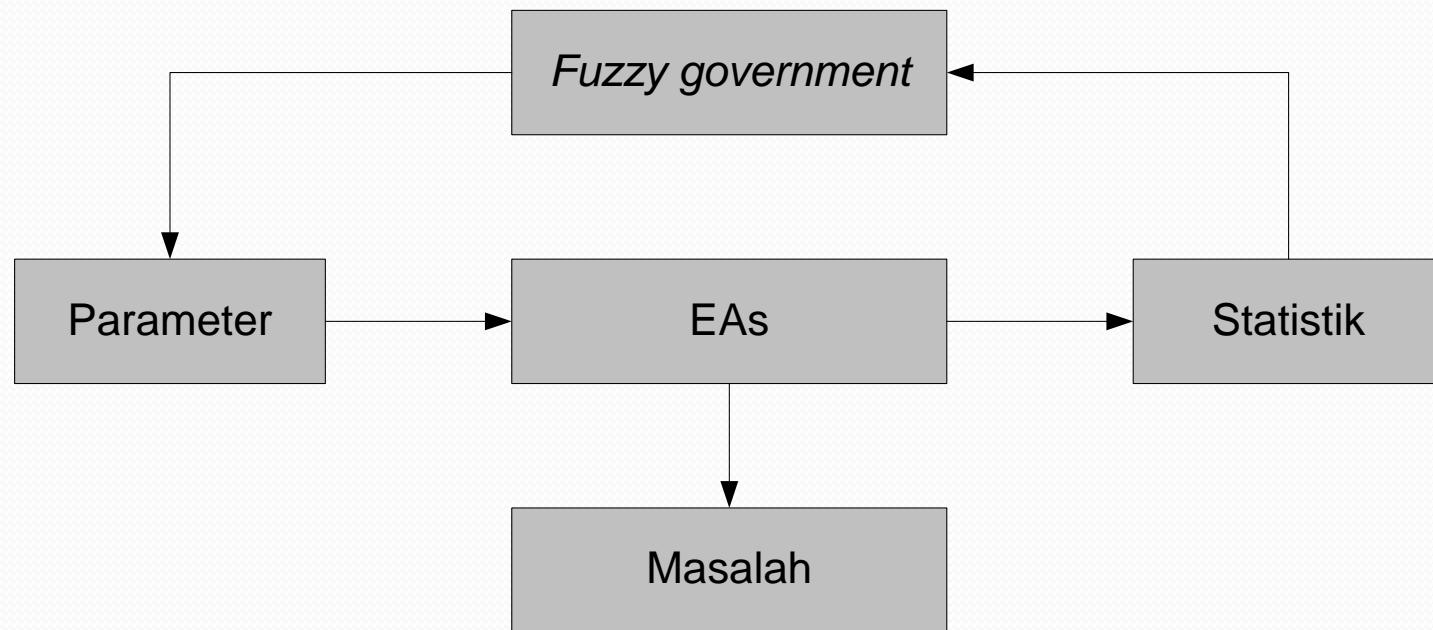
Island model EAs



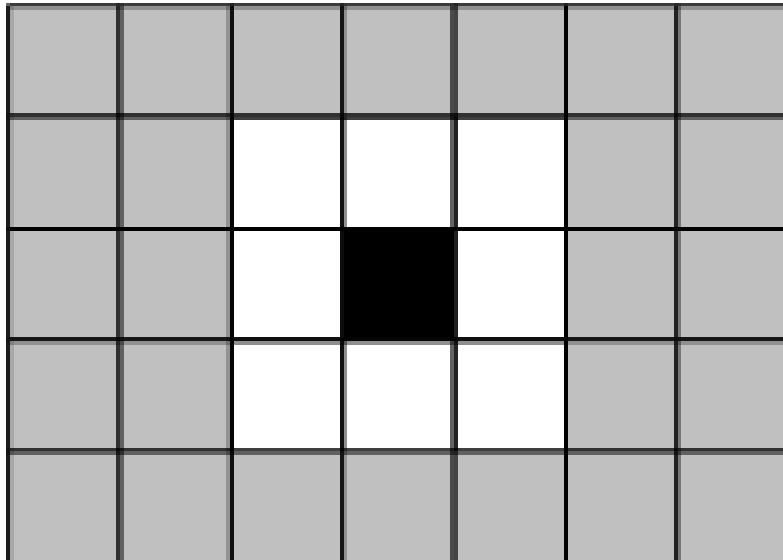


The best
individual

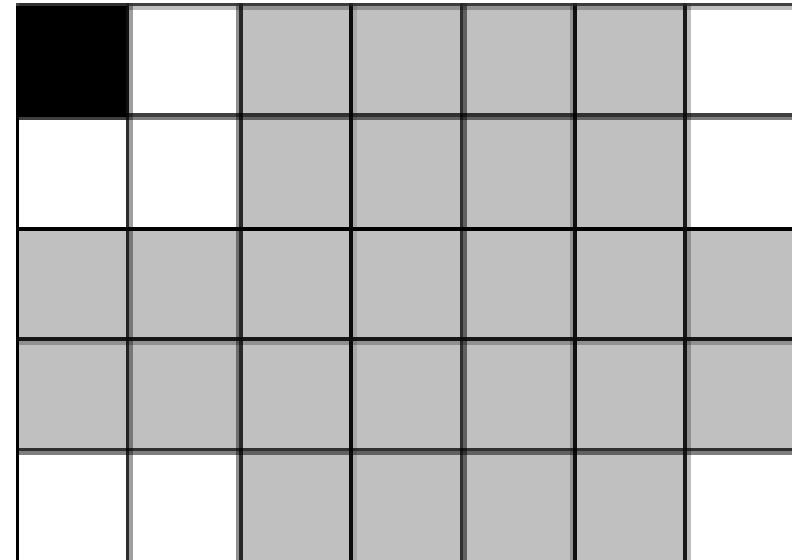
Adaptive EAs



Grid-based Crossover



(a)



(b)

- Individu-individu diletakkan dalam suatu ***toroidal space***, dimana ujung-ujung kotak tersebut disatukan dan membentuk ruang tiga dimensi seperti bola.
- Individu hanya bisa di-crossover dengan individu2 tetangganya.
- Individu hitam hanya boleh crossover dengan 1 dari 8 individu tetangga.

100	91	92	93	94	95	96	97	98	99	100	91
10	1	2	3	4	5	6	7	8	9	10	1
20	11	12	13	14	15	16	17	18	19	20	11
30	21	22	23	24	25	26	27	28	29	30	21
40	31	32	33	34	35	36	37	38	39	40	31
50	41	42	43	44	45	46	47	48	49	50	41
60	51	52	53	54	55	56	57	58	59	60	51
70	61	62	63	64	65	66	67	68	69	70	61
80	71	72	73	74	75	76	77	78	79	80	71
90	81	82	83	84	85	86	87	88	89	90	81
100	91	92	93	94	95	96	97	98	99	100	91
10	1	2	3	4	5	6	7	8	9	10	1

GA untuk melatih FFNN (MLP)

- Menemukan weights secara otomatis
- Menggunakan representasi biner
- Bagaimana performansi Advanced GA?

Masalah 3-Parity

Tabel Kebenaran XOR untuk tiga masukan X_1 , X_2 , dan X_3 .

X_1	X_2	X_3	Y
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

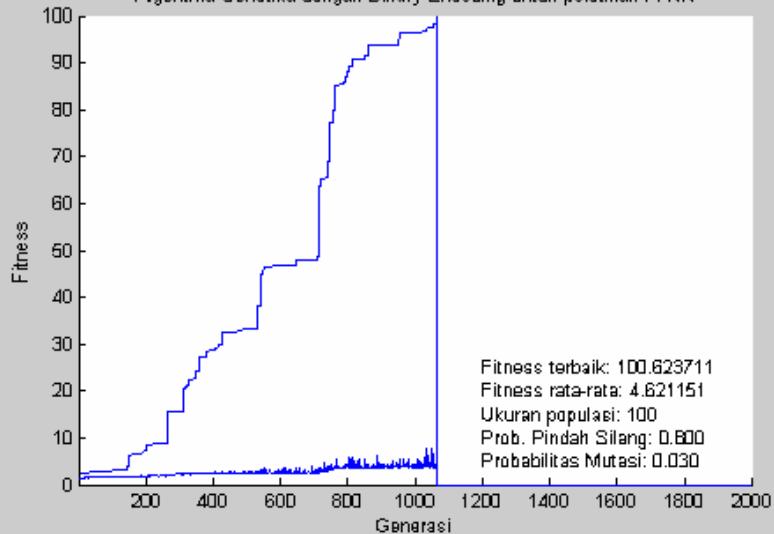
Fungsi Fitness

$$f = 1/delta$$

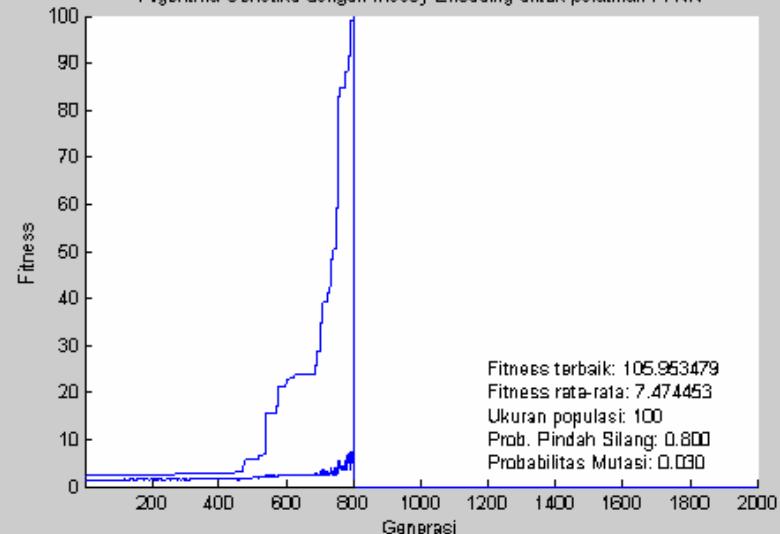
$$\text{delta} = \sqrt{\frac{1}{m} \sum_{i=1}^m (d(i) - y(i))^2}$$

- $m = 2^n = 2^3 = 8$
- $d(i)$ adalah output yang diharapkan.
- Pola masukan pertama, $X_1 = 0$, $X_2 = 0$, dan $X_3 = 0$, maka $d(1) = 0$.
- Pola masukan ke dua, $X_1 = 0$, $X_2 = 0$, dan $X_3 = 1$, maka $d(2) = 1$, dst.
- $y(i)$ adalah output aktual yang dihasilkan.

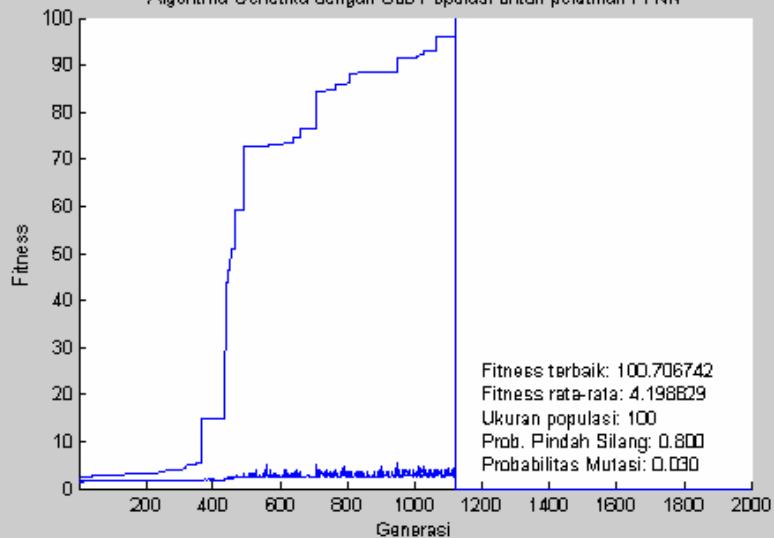
Algoritma Genetika dengan Binary Encoding untuk pelatihan FFNN



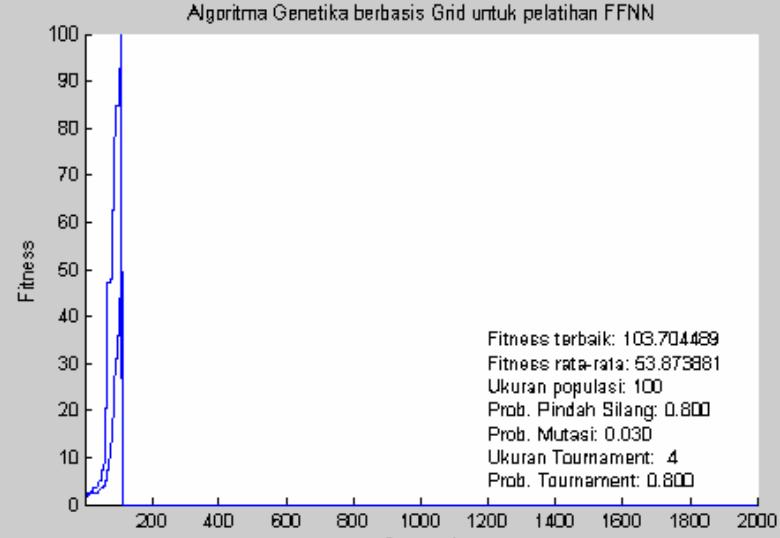
Algoritma Genetika dengan Messy Encoding untuk pelatihan FFNN



Algoritma Genetika dengan Sub Populasi untuk pelatihan FFNN



Algoritma Genetika berbasis Grid untuk pelatihan FFNN



Grammatical Encoding

- Otak manusia merupakan suatu komputer sangat kompleks yang terdiri dari sekitar 10^{11} elemen komputasi (*neurons*).
- Terdapat sekitar 10^{14} sampai 10^{15} koneksi antar *neurons*, atau sekitar 1000 sampai 10000 koneksi per *neuron*.
- Jika setiap koneksi dikodekan ke dalam kromosom, maka informasi yang mengisi kromosom akan sekitar **10⁵ GB**, dimana bobot-bobot sinaptik dikodekan menggunakan hanya 1 *byte*.
- Tetapi, pada kenyataanya ukuran *genome* manusia hanya sekitar **3 GB**.
- Oleh karena itu para peneliti percaya bahwa pengkodean otak manusia bukanlah menggunakan pengkodean langsung, melainkan pengkodean **prosedur** dimana otak dibentuk.

Grammatical Encoding

- Pada skema ini, kromosom dipandang sebagai kalimat yang diekspresikan menggunakan *grammar* (tata bahasa).
- Ketika sebuah kalimat dibaca (kromosom didekodekan), maka individu dibangkitkan menggunakan *grammar* tsb.
- Contoh: skema Kitano yang digunakan untuk mengkodekan ANN yang berisi maksimum 8 *neurons*

Skema Kitano

S	ACBA	Aadfb	Bbefd	Dfanp	Bahjm	Ckhgf	...
---	------	-------	-------	-------	-------	-------	-----

$$S \rightarrow \begin{pmatrix} A & C \\ B & A \end{pmatrix}$$

$$A \rightarrow \begin{pmatrix} a & d \\ f & b \end{pmatrix} \quad B \rightarrow \begin{pmatrix} b & e \\ f & d \end{pmatrix}$$

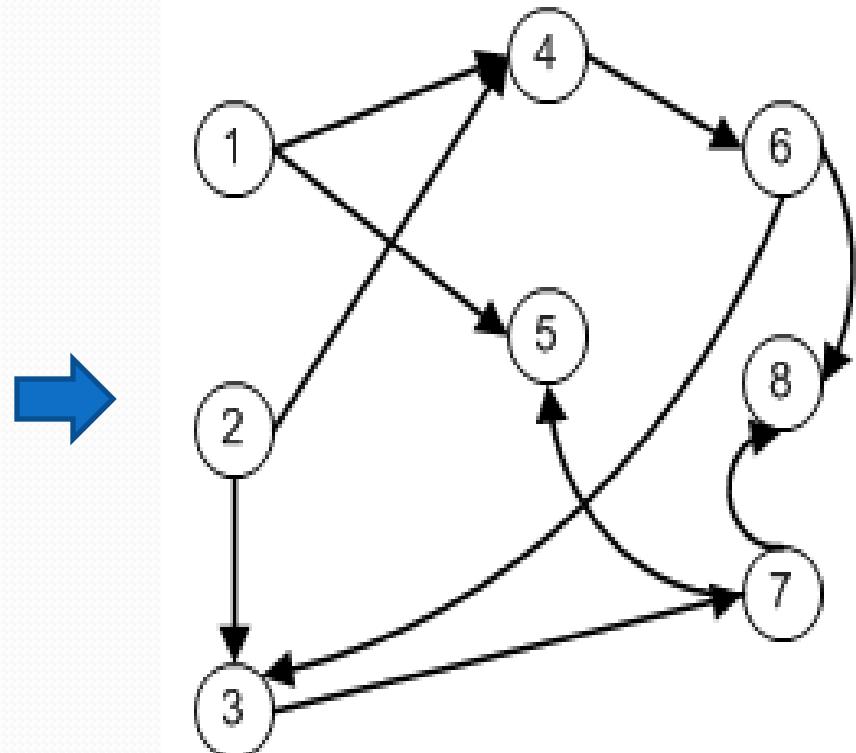
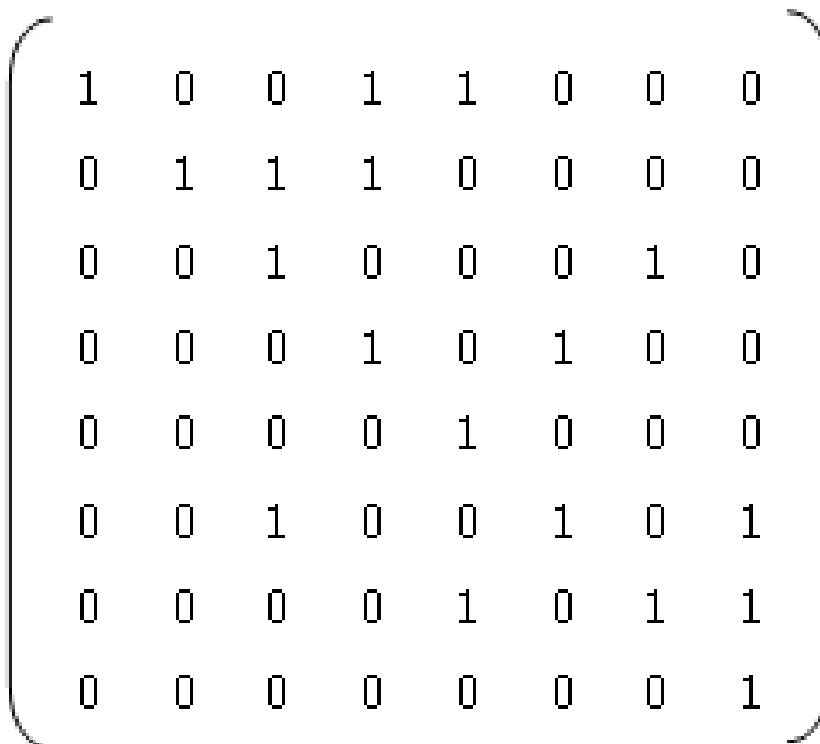
$$a \rightarrow \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}, \quad b \rightarrow \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}, \quad c \rightarrow \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix}, \dots p \rightarrow \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$$

Skema Kitano

S	ACBA	Aadfb	Bbefd	Dfanp	Bahjm	Ckhgf	...
---	------	-------	-------	-------	-------	-------	-----

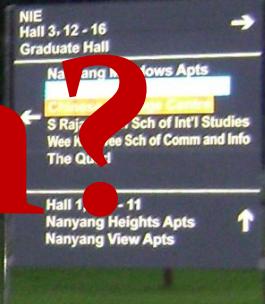
$$\begin{pmatrix} A & C \\ B & A \end{pmatrix} \xrightarrow{\quad} \begin{pmatrix} a & d & k & h \\ f & b & g & f \\ b & e & a & d \\ f & d & f & b \end{pmatrix} \xrightarrow{\quad} \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}$$

Skema Kitano



Demo Program

Question?



Evolution Learning?

- Baldwin Effect (1896)
- A new Factor of Evolution
- Learning → mempercepat evolusi?
 - Rusa (belajar) mempercepat larinya
 - Harimau (belajar) mempertajam taringnya
 - Dsb.

Evolution → Learning?

Learning → Evolution?



Hinton & Nowlan's model (1987)

- Kromosom: 20 gen biner
 - 5 gen bernilai 0
 - 5 gen bernilai 1
 - 10 gen bernilai ‘?’ (tidak diketahui 1 atau 0)
- Misalkan
 - Kromosom terbaik: **11111111110000000000**
 - Populasi: **1000** kromosom
 - Setiap kromosom **menebak** gen ‘?’ → **1000** trials
 - **Fitness** = $1 + ((19^*n)/M)$
 - M = jumlah trial maksimum
 - n = sisa trial

Contoh Populasi

- **Kromosom 1:** 111???001?0100???????
- **Kromosom 2:** 11111?????00000??????
- **Kromosom 3:** 1111100000????????????
- **Kromosom 4:** ????.?0001111100??????
- **Kromosom 5:** 00????00111110??????
- ...
- **Kromosom 1000:** ?000??????0011???111
- **Kromosom target:** **11111111110000000000**

Operator Evolusi

- **Seleksi ortu** : Roulette Wheel
- **Rekombinasi** : satu titik
- **Mutasi Biner** : terhadap gen 0 atau 1
- **Seleksi survivor:** Generational replacement

Contoh Trial

Kromosom target 11111111110000000000

Kromosom 111 ???001 ?0100???????

- **Trial 1** 11100000100100000011
- **Trial 2** 11111000110100001111
- **Trial 3** 11101000100100111100
- ...
- **Trial 1000** 11111100100100000000

Rekombinasi

Satu titik potong

Parent 1

110	????????????		0000000
-----	--------------	--	---------

Parent 2

1111111111	????		0???????
------------	------	--	----------

Offspring 1

110	????????????		0???????
-----	--------------	--	----------

Offspring 2

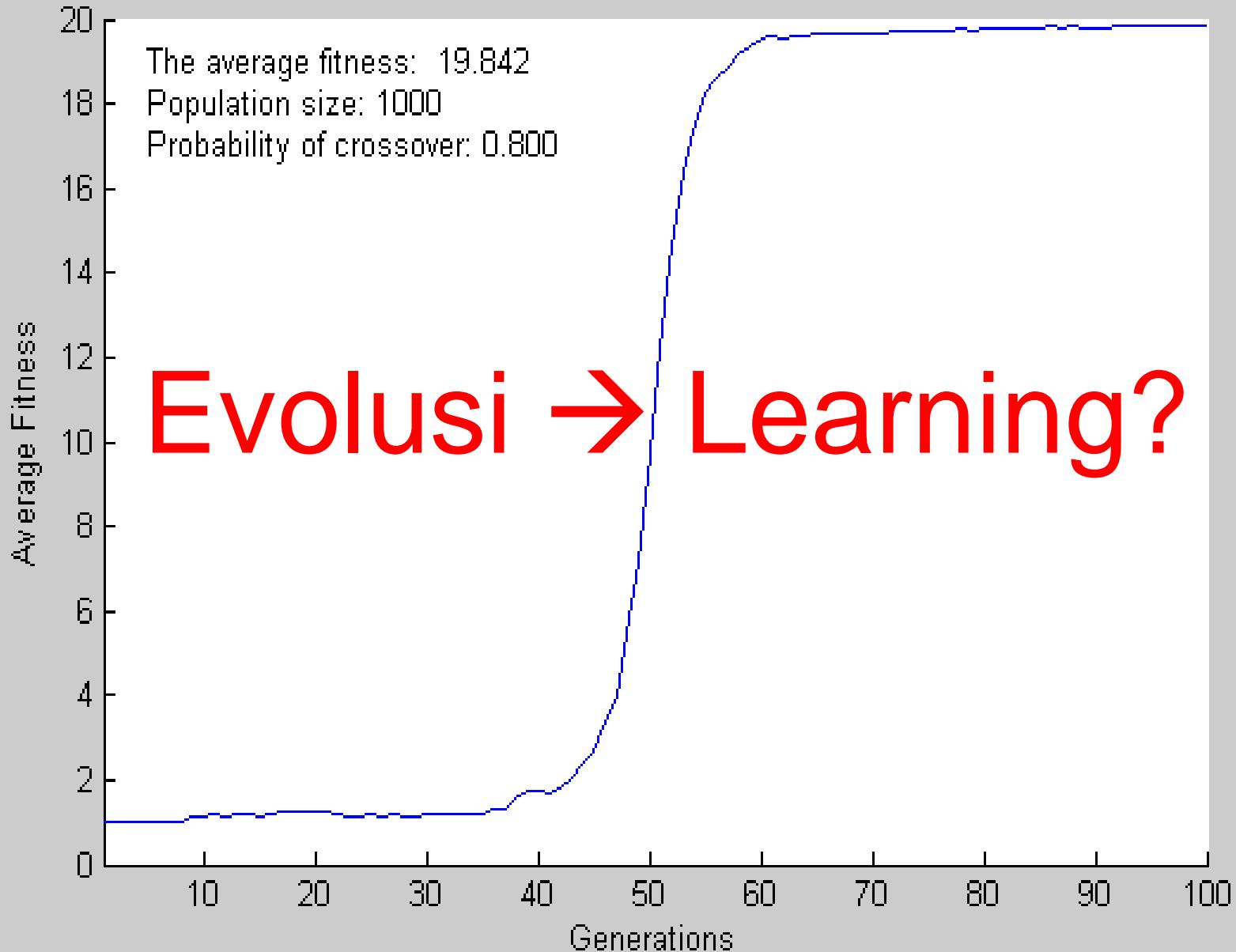
1111111111	????		0000000
------------	------	--	---------

Mutasi Biner

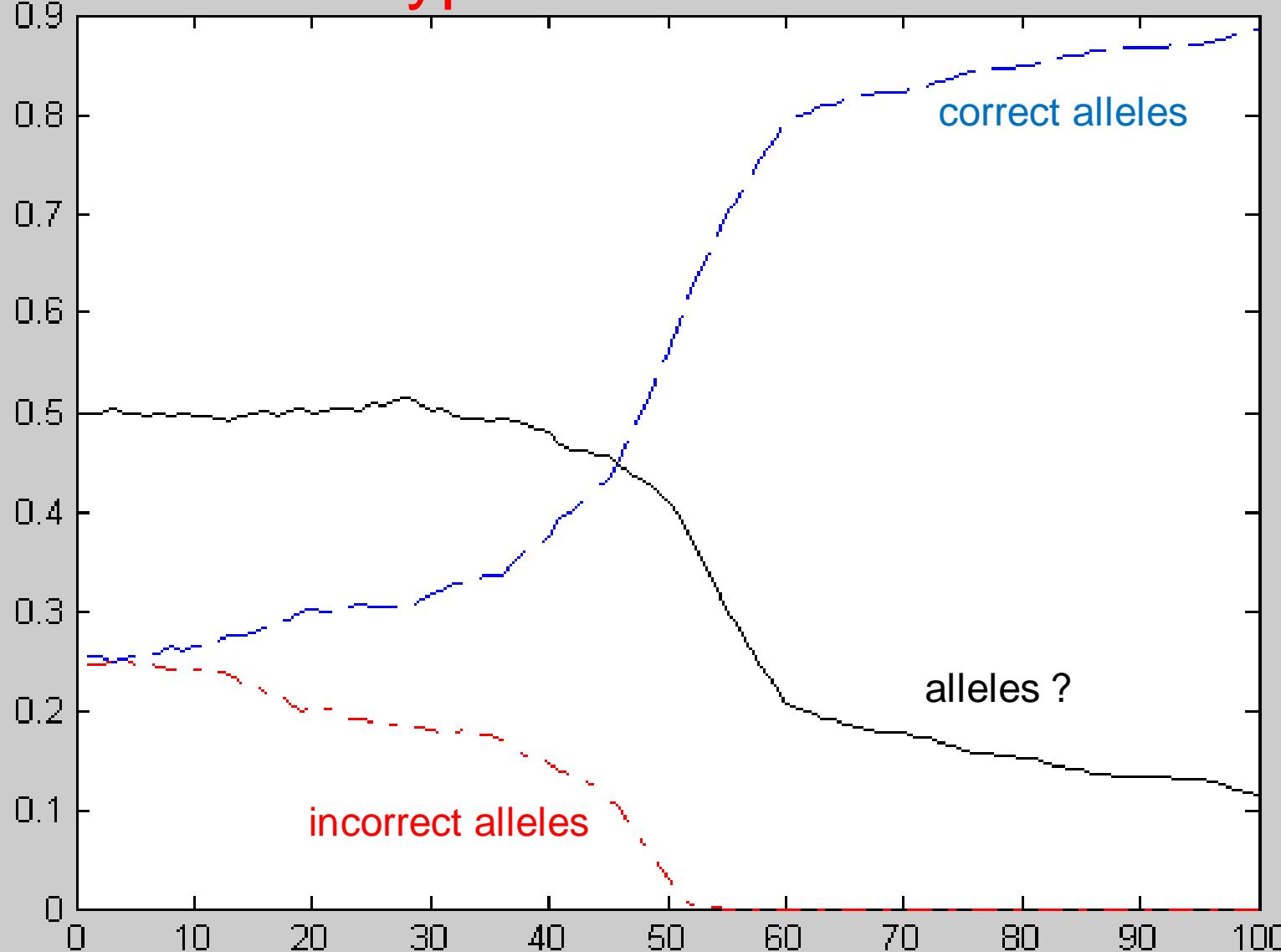
- Mutasi hanya untuk gen 0 atau 1 (tidak utk gen ?)
- 1 dimutasi menjadi 0
- 0 dimutasi menjadi 1

Kromosom 111111111????1000000

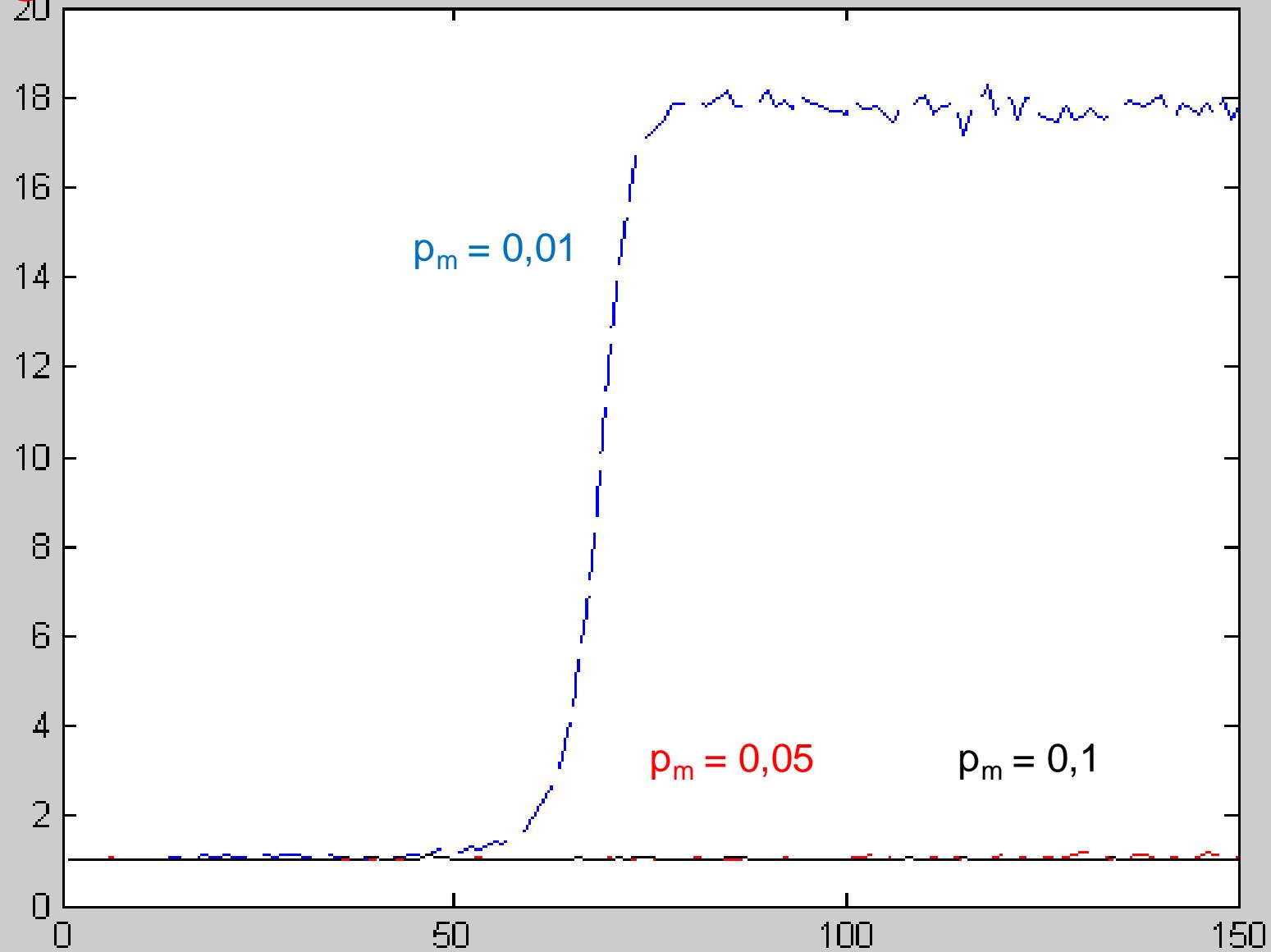
Hasil mutasi 111111111????0000000



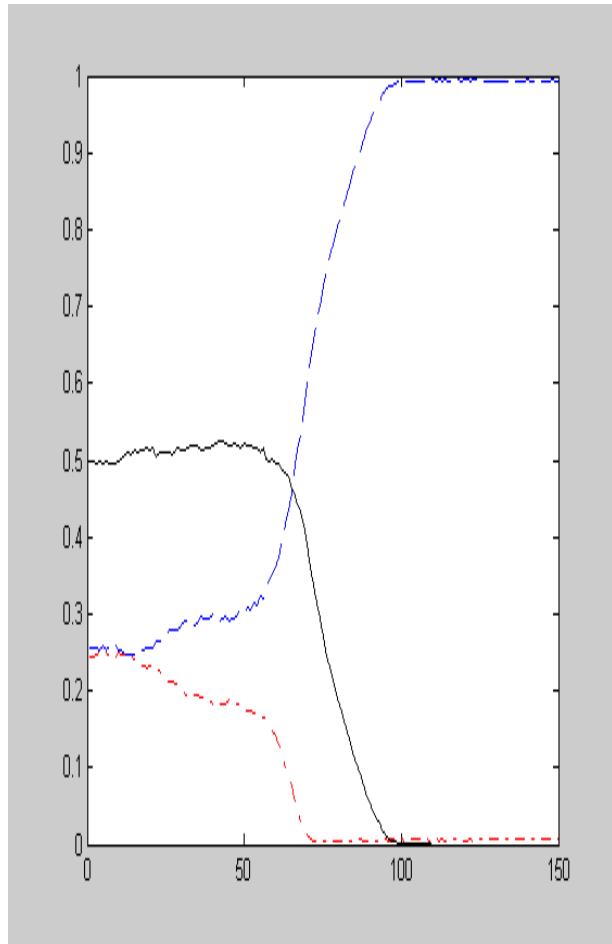
Dinamika Genotype



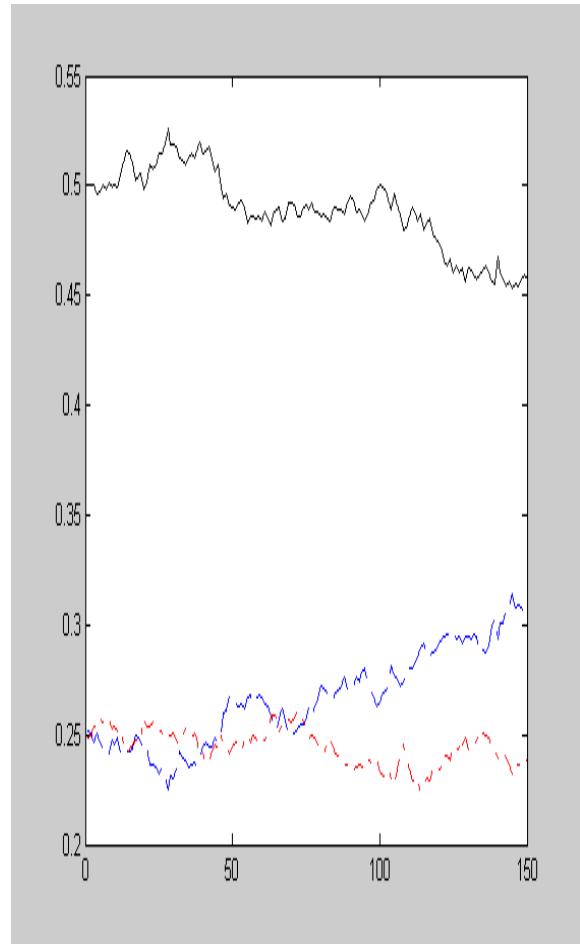
Pengaruh Mutasi



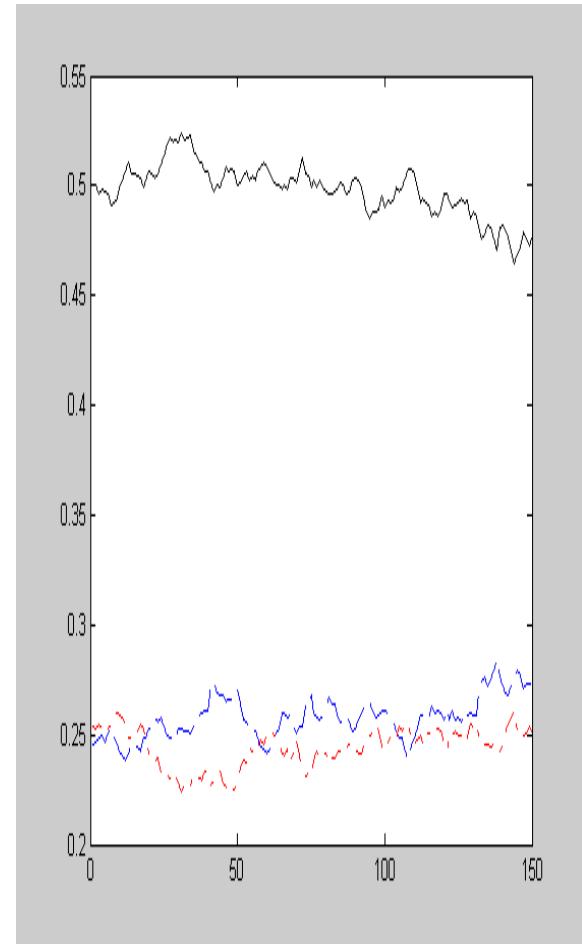
$p_m = 0,01$



$p_m = 0,05$



$p_m = 0,1$

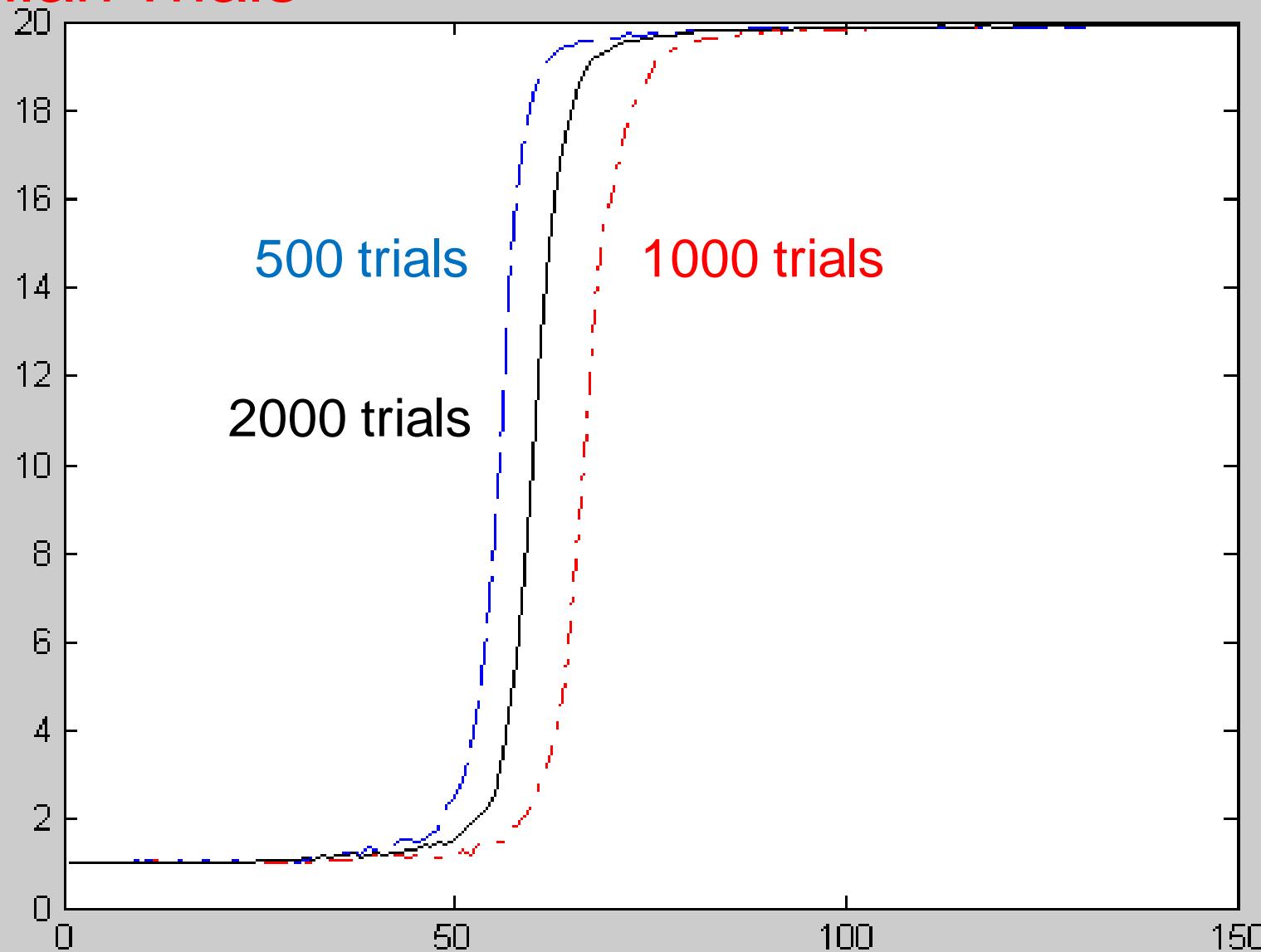


correct alleles

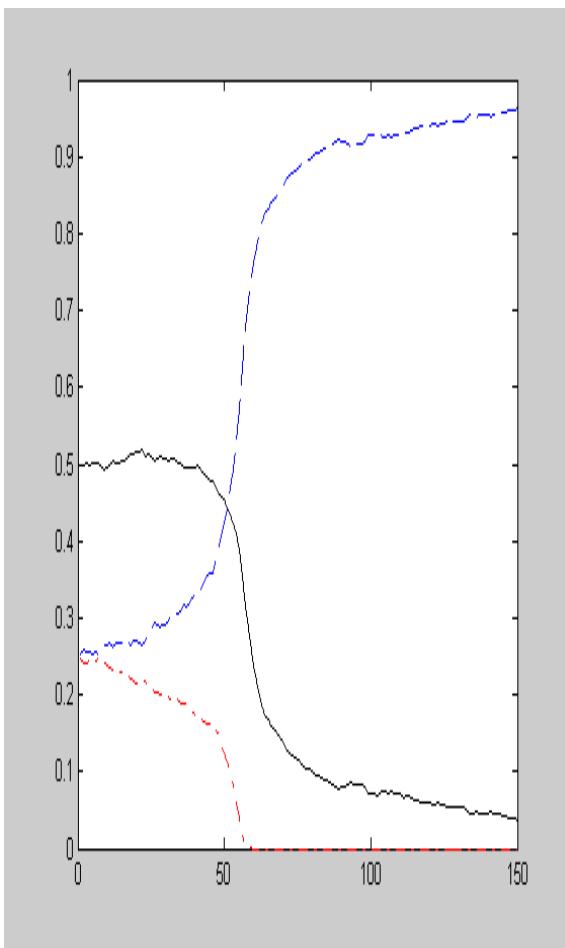
incorrect alleles

alleles ?

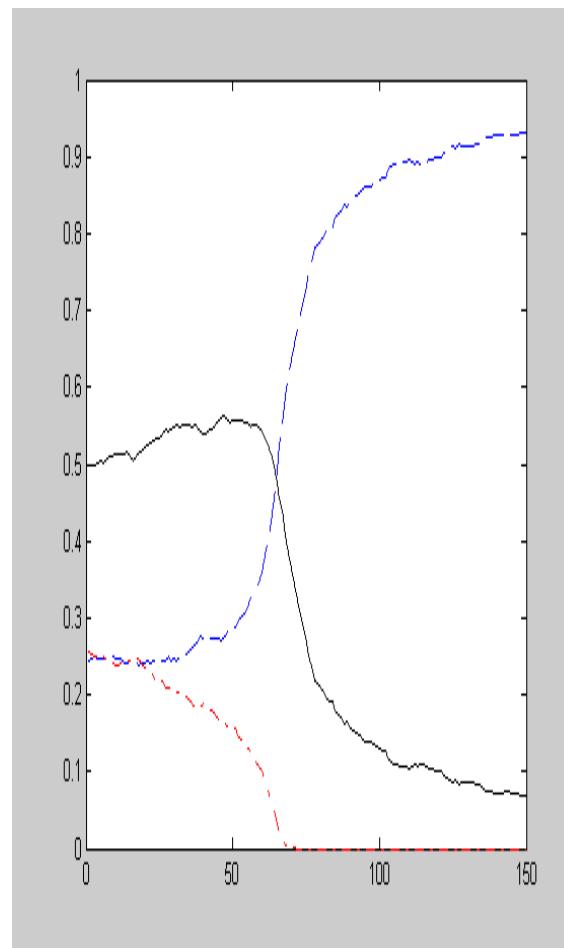
Jumlah Trials



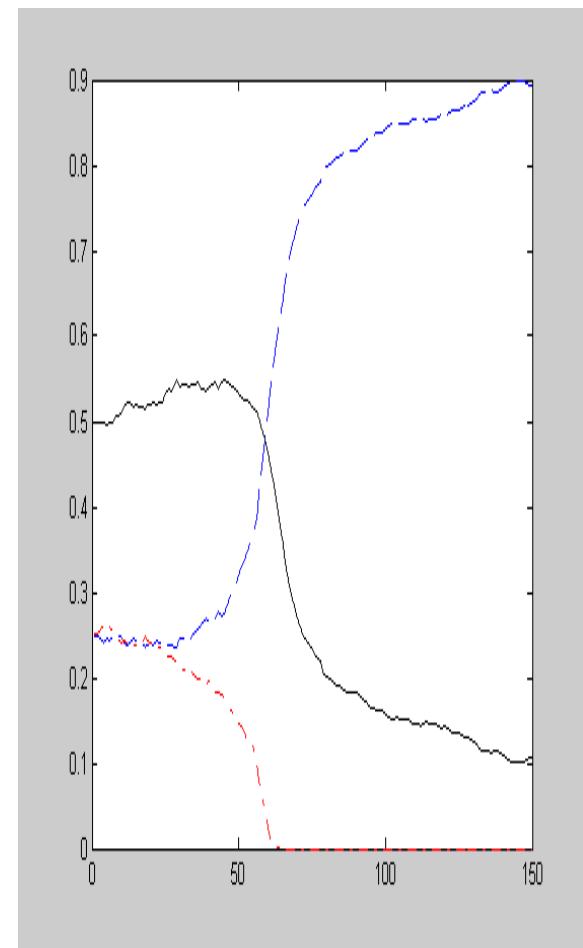
500 trials



1000 trials



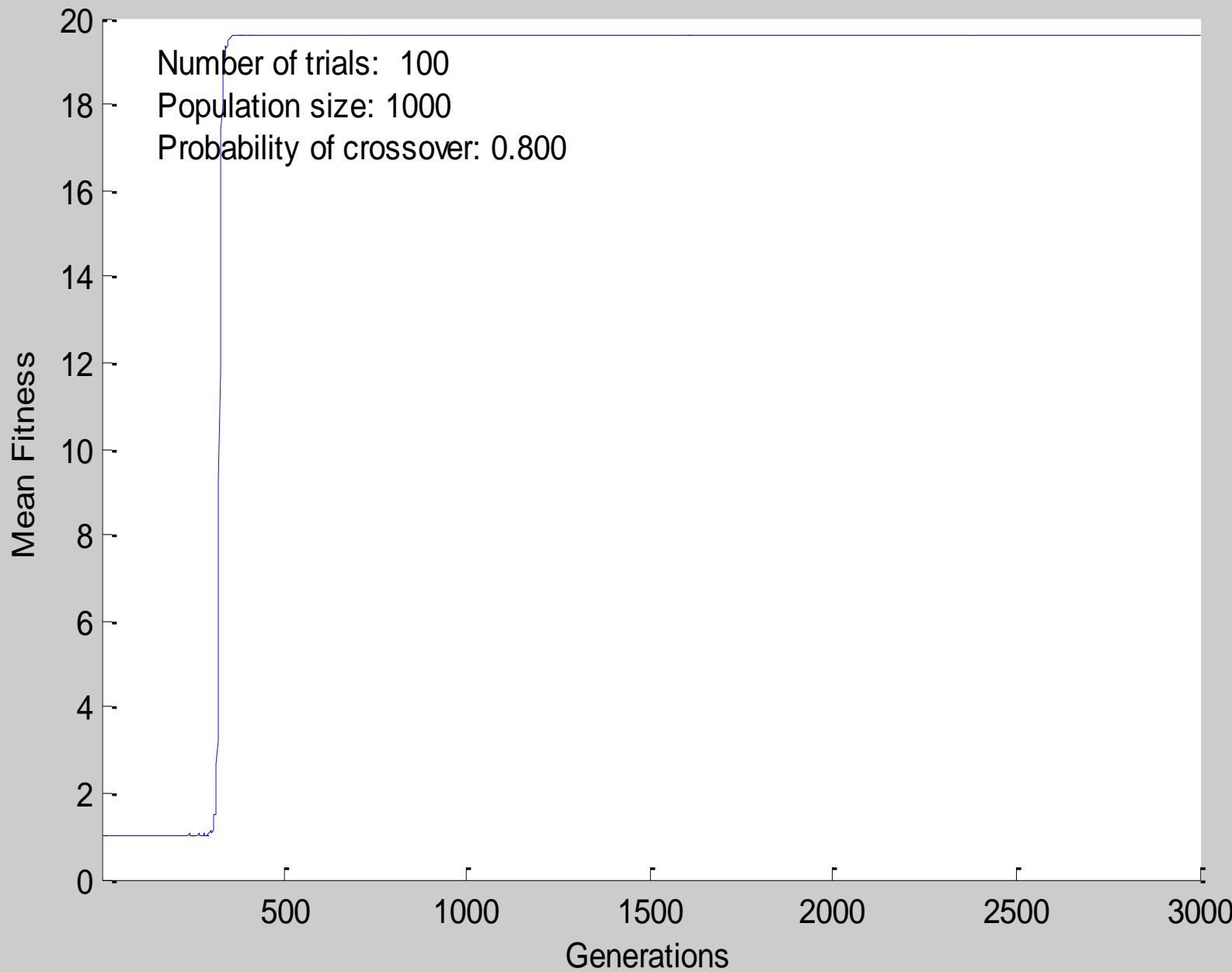
2000 trials

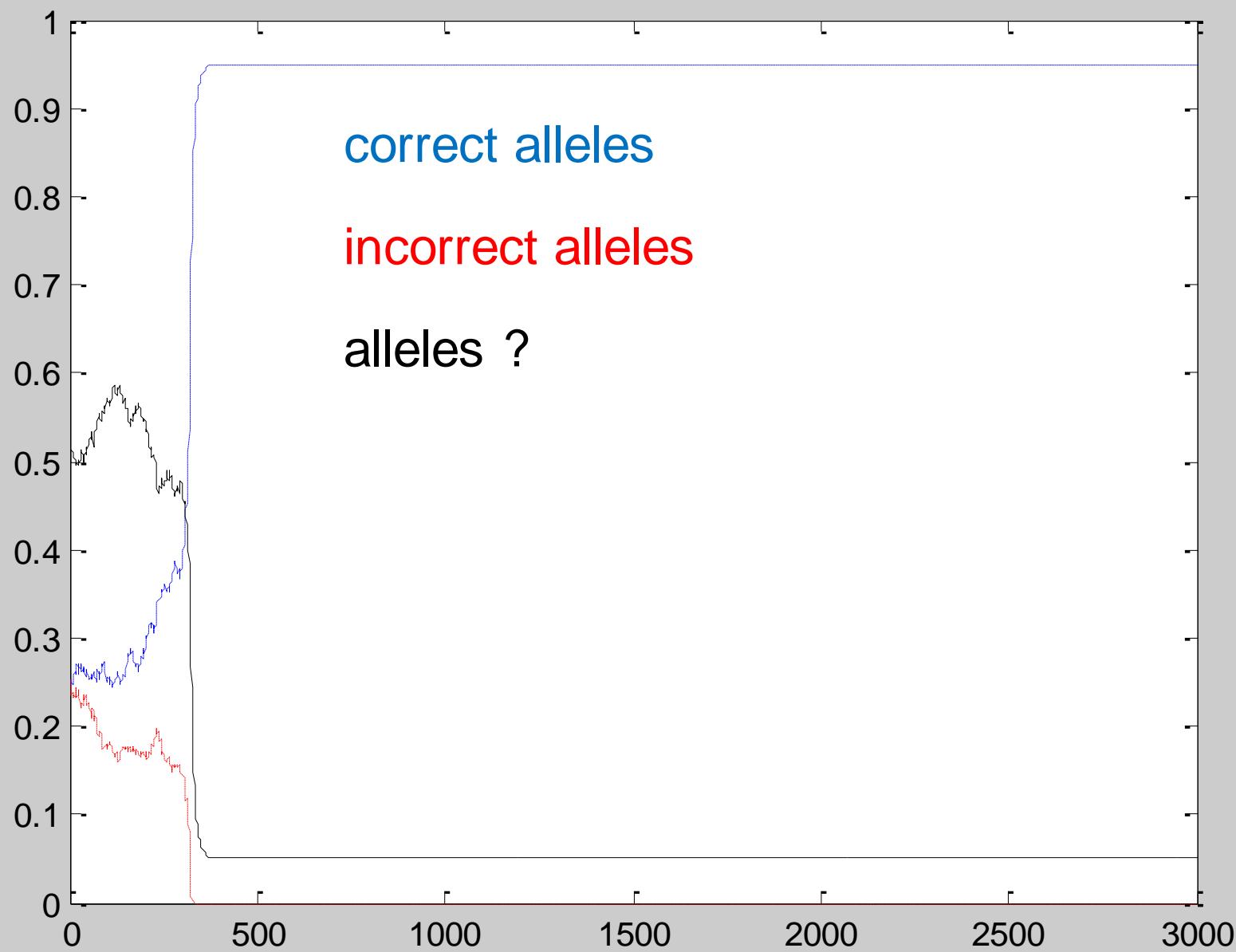


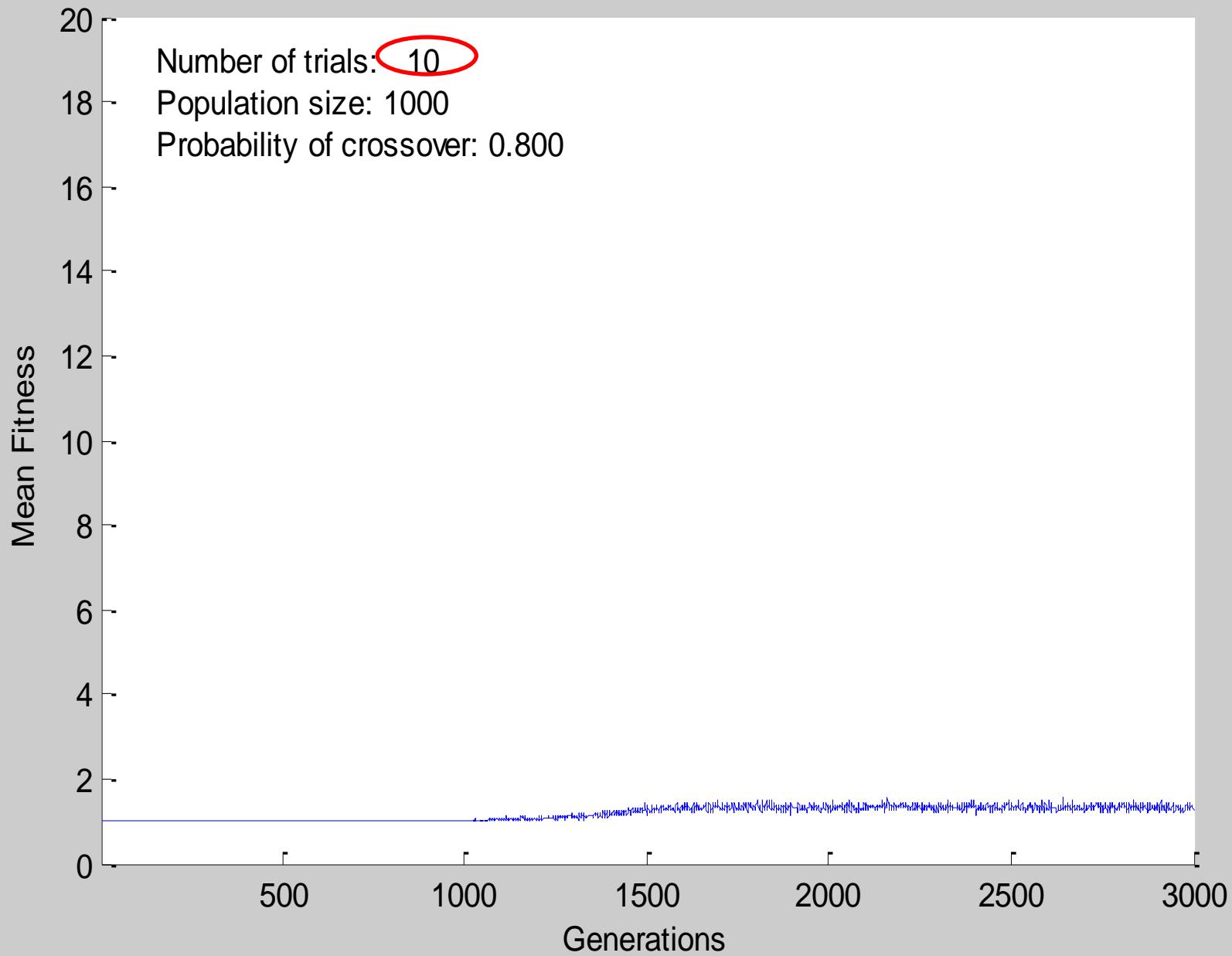
correct alleles

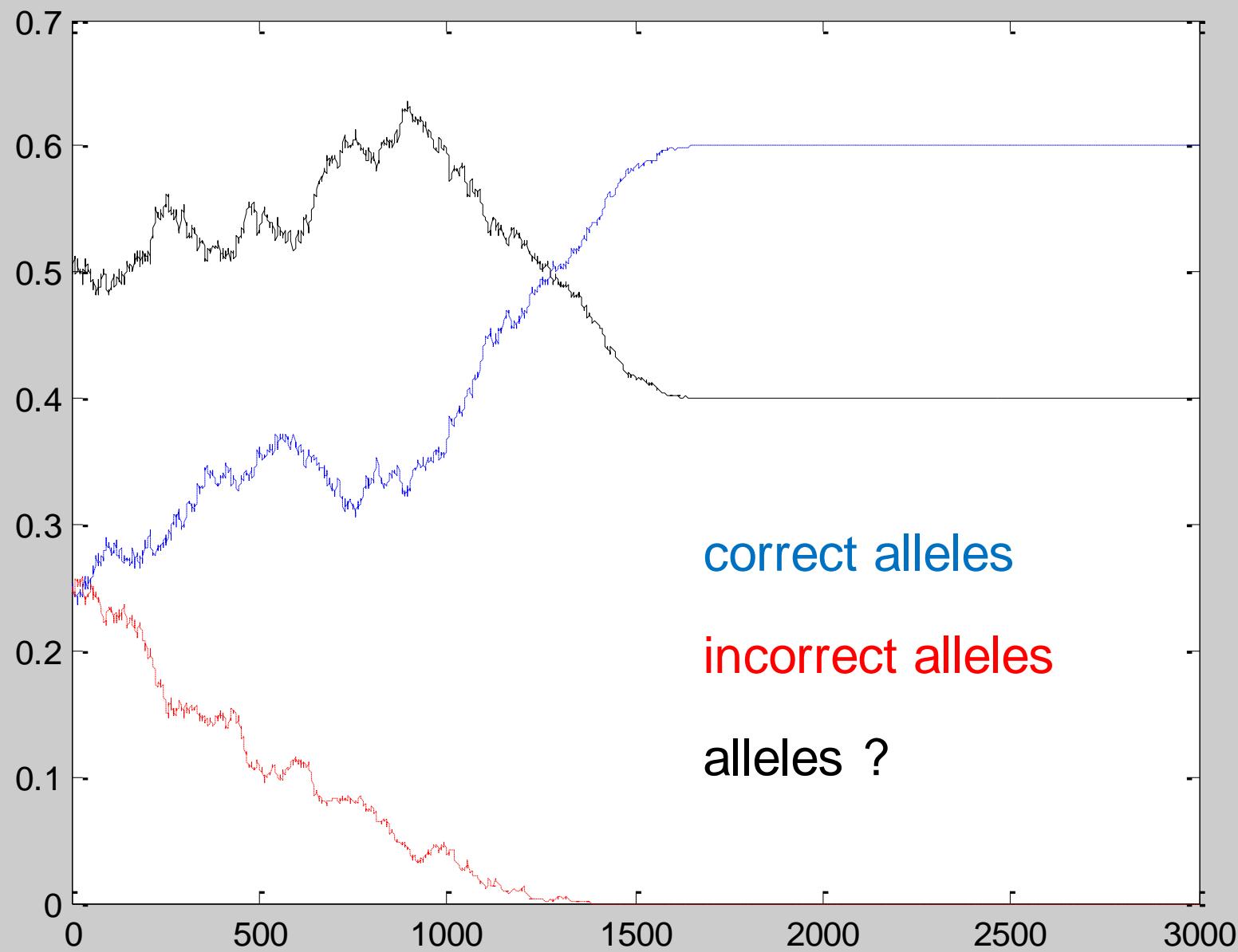
incorrect alleles

alleles ?

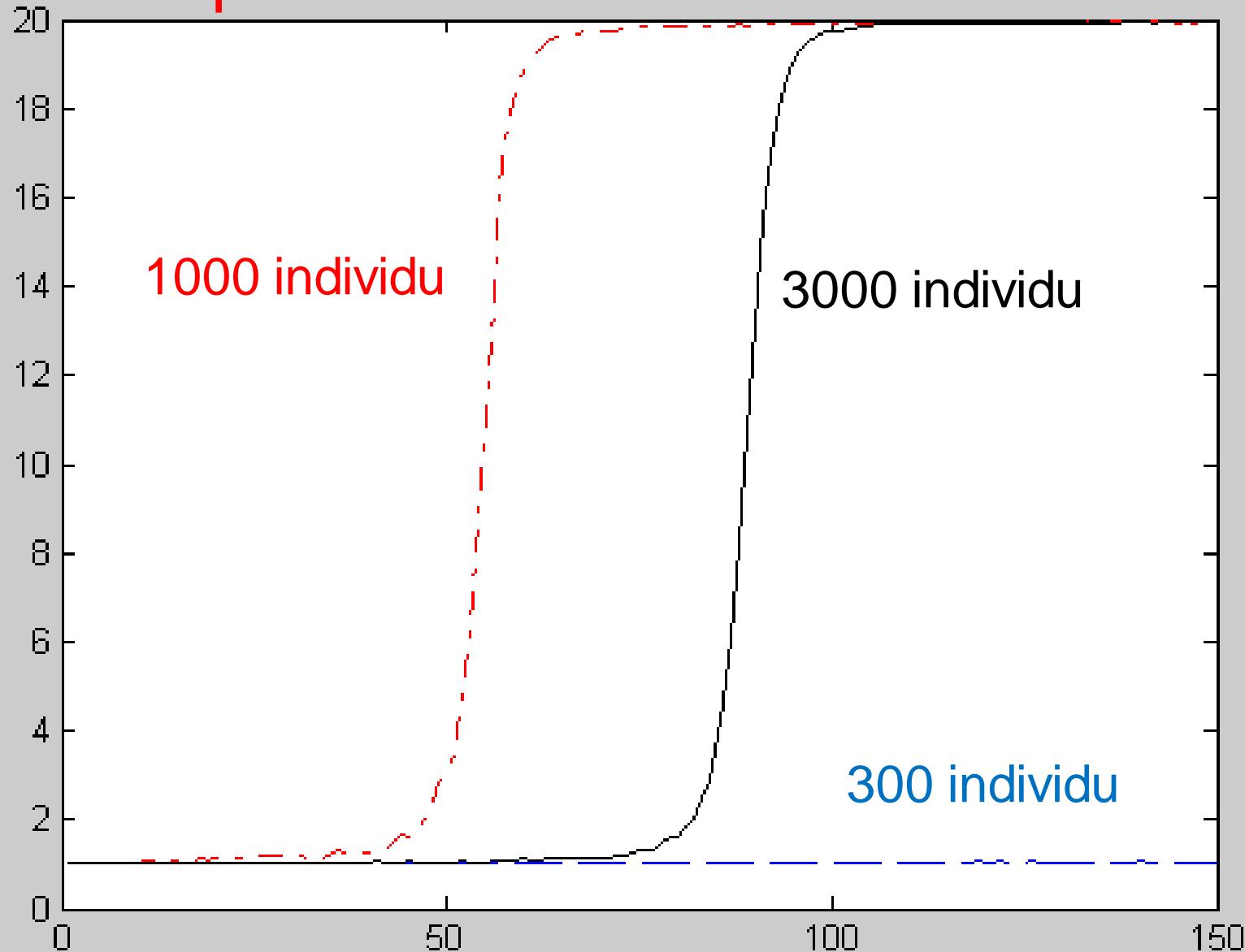




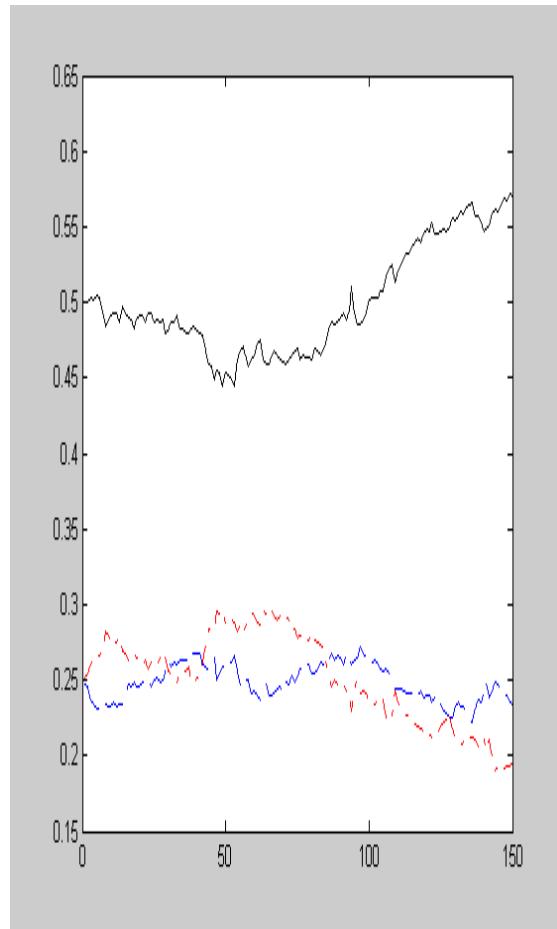




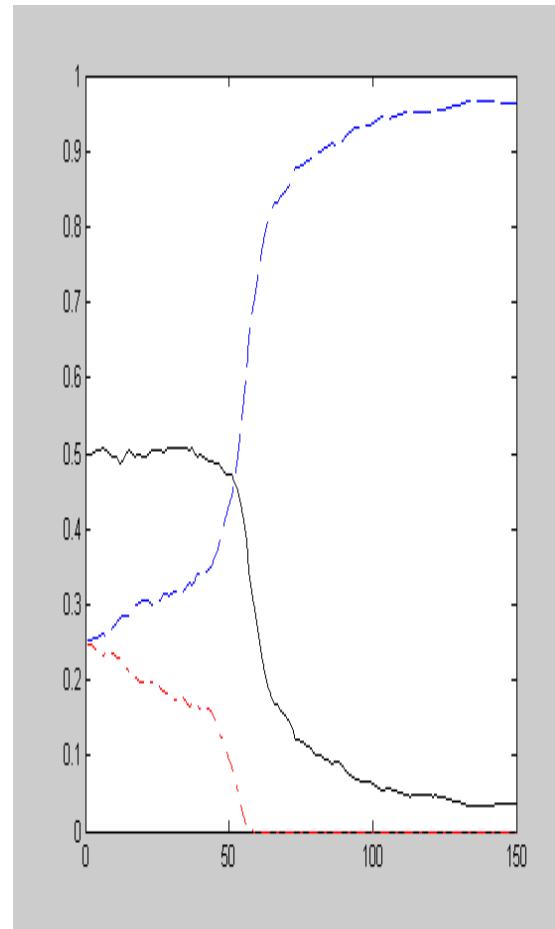
Ukuran Populasi



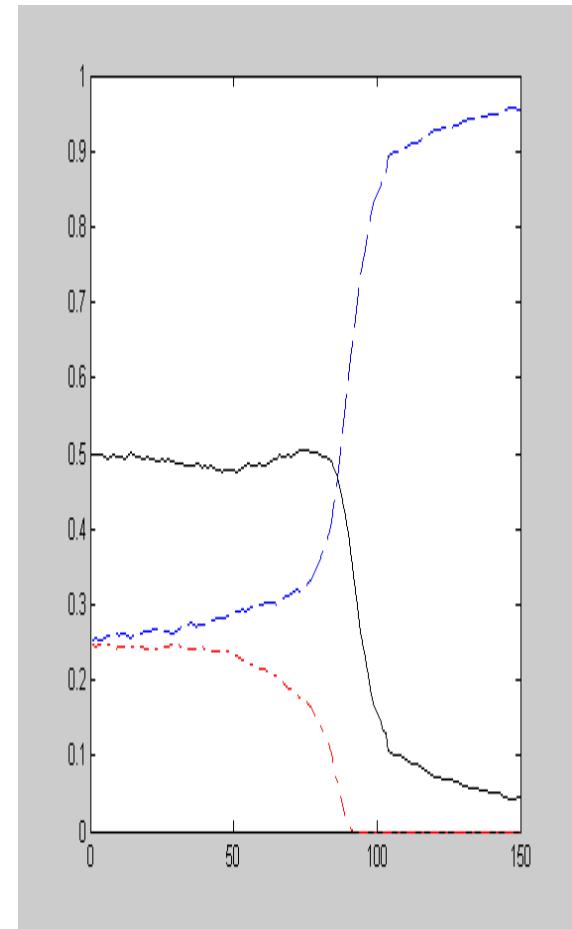
300 individu



1000 individu



3000 individu



correct alleles

incorrect alleles

alleles ?

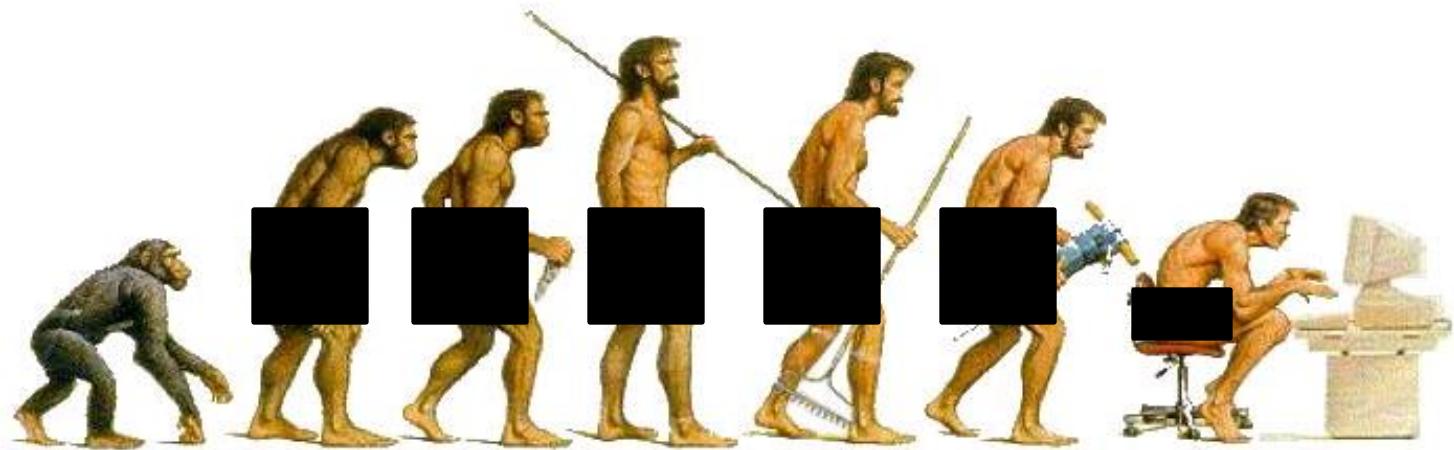
Evolution → Learning?

Learning → Evolution?



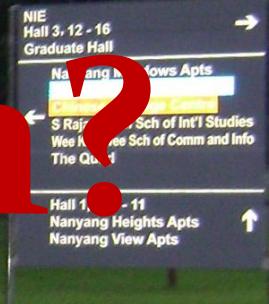
Berlatih terus bisa mempercepat evolusi?

Evolution



Makin Pintar?

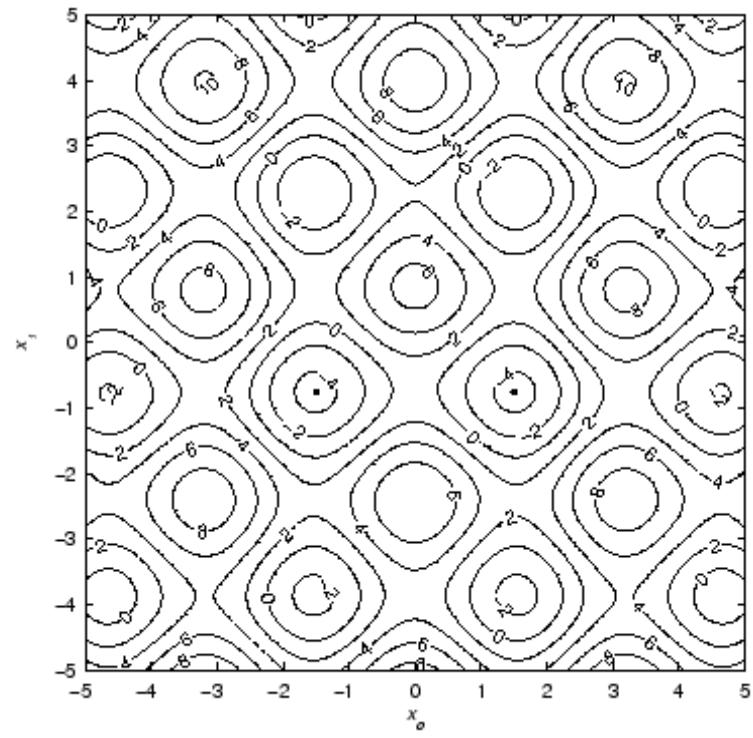
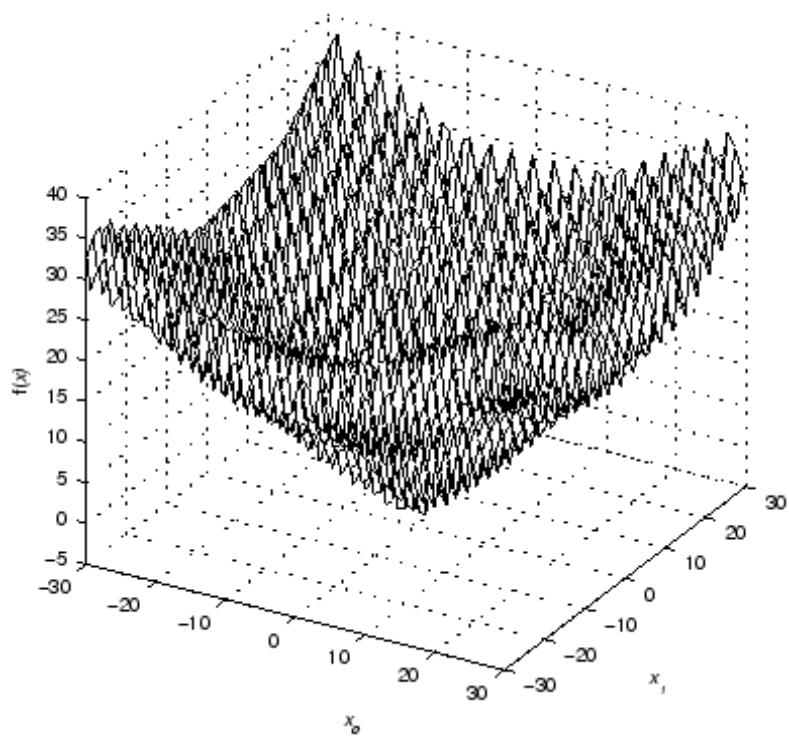
Question?



Studi Kasus: Kromosom & Fitness

1. Optimasi Fungsi Kompleks
2. *Graph Bisection*
3. *8-queens*
4. Pemotongan bahan
5. Peramalan Data *Time Series*
6. Penjadwalan kuliah
7. **Your problems**

$$f(\vec{x}) = \sum_{i=0}^{D-1} \left(e^{-0.2} \sqrt{x_{i-1}^2 + x_i^2} + 3(\cos(2x_{i-1}) + \sin(2x_i)) \right)$$



- Untuk presisi $10^{-9} \rightarrow$ Berapa bit?
- Bisa menggunakan kromosom Integer atau Real?

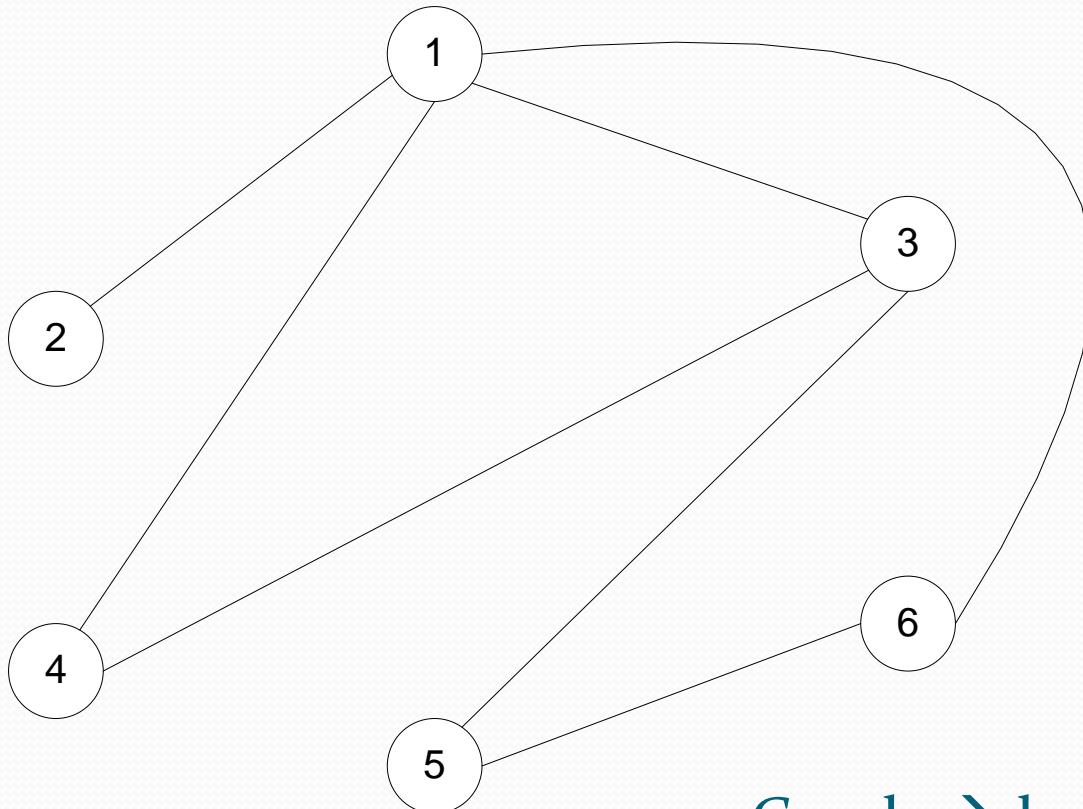
Fungsi *fitness*

$$f = \frac{1}{(h + a)}$$

h = nilai fungsi yang diminimasi

a = bilangan kecil untuk menghindari pembagian dengan nol

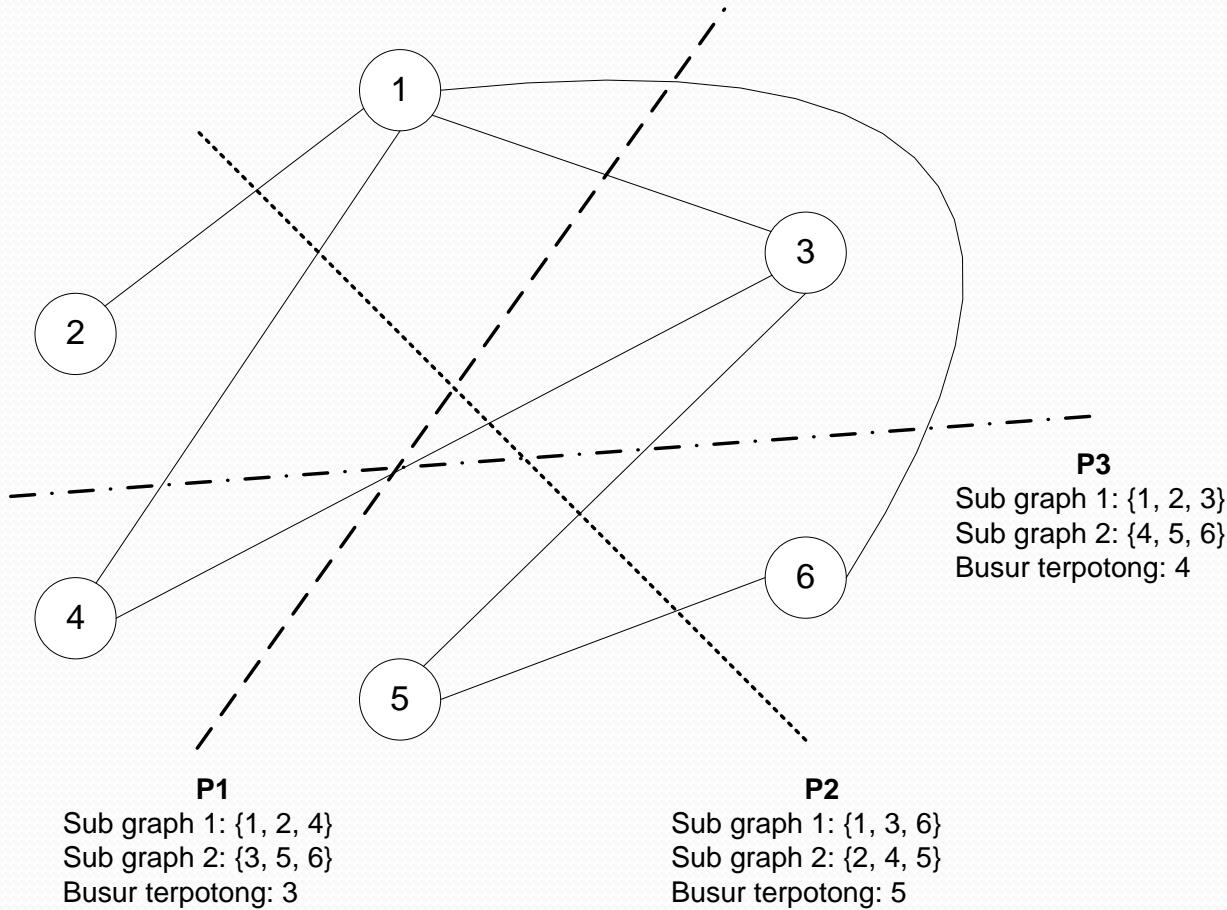
Graph bisection

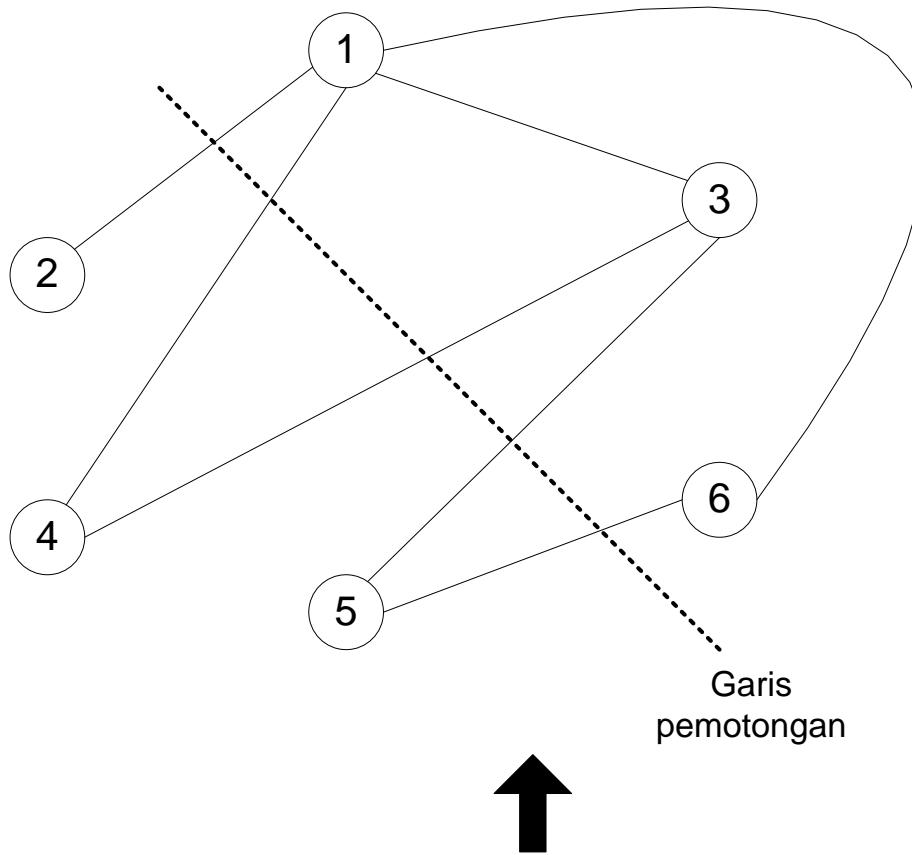


- VLSI design
- Task Scheduling
- Parallel Scientific Computing

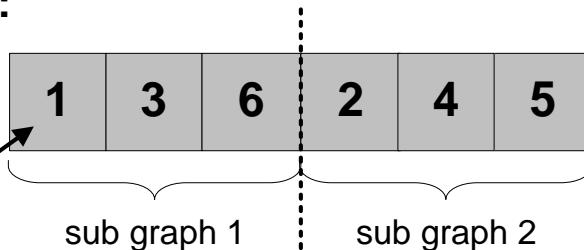
- Graph → dua sub graph sama besar?
- Minimasi busur terpotong

Graph bisection





Kromosom:



Nilai gen
menyatakan
nomor *node*

Permutasi
Biner?
Integer?
Real?

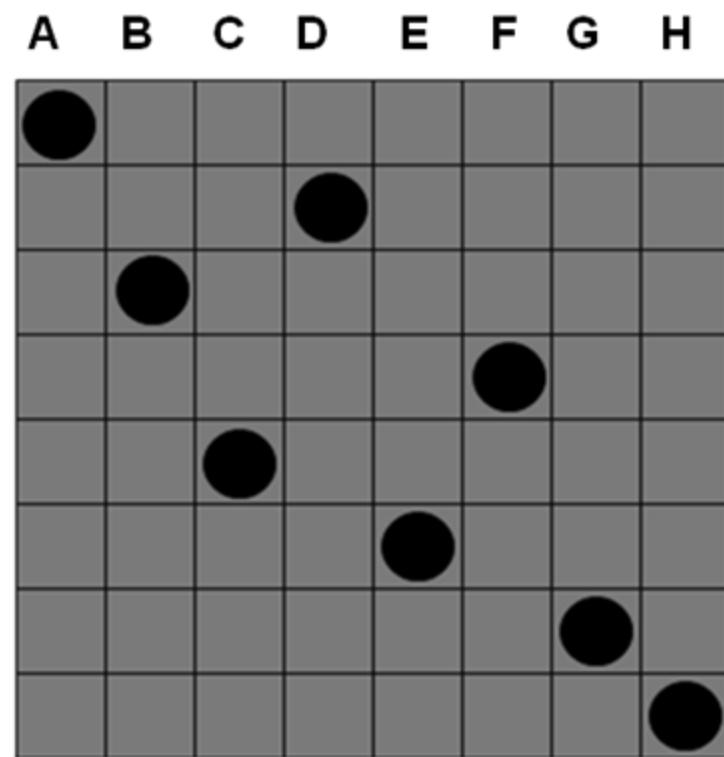
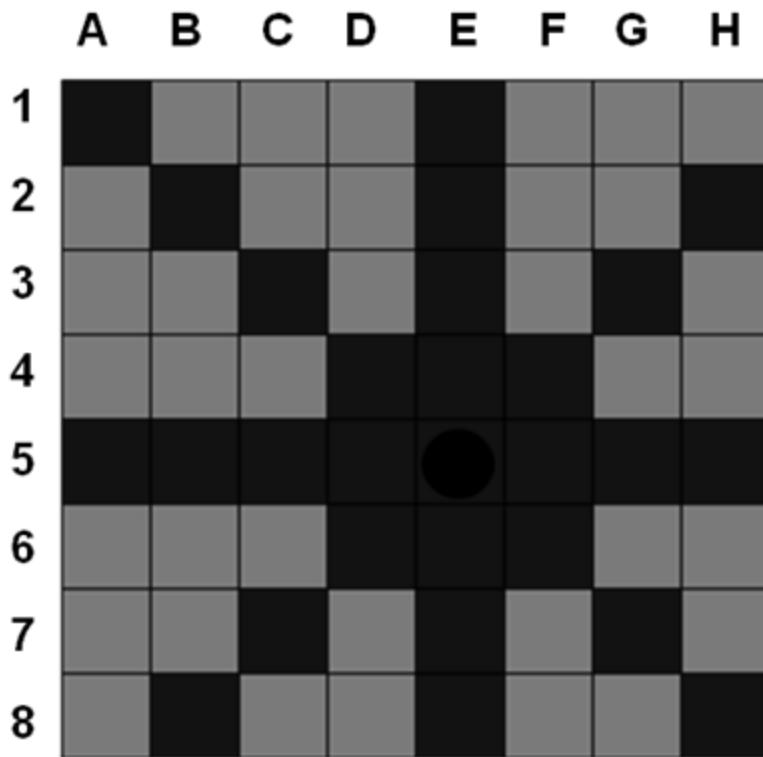
Fungsi *fitness*

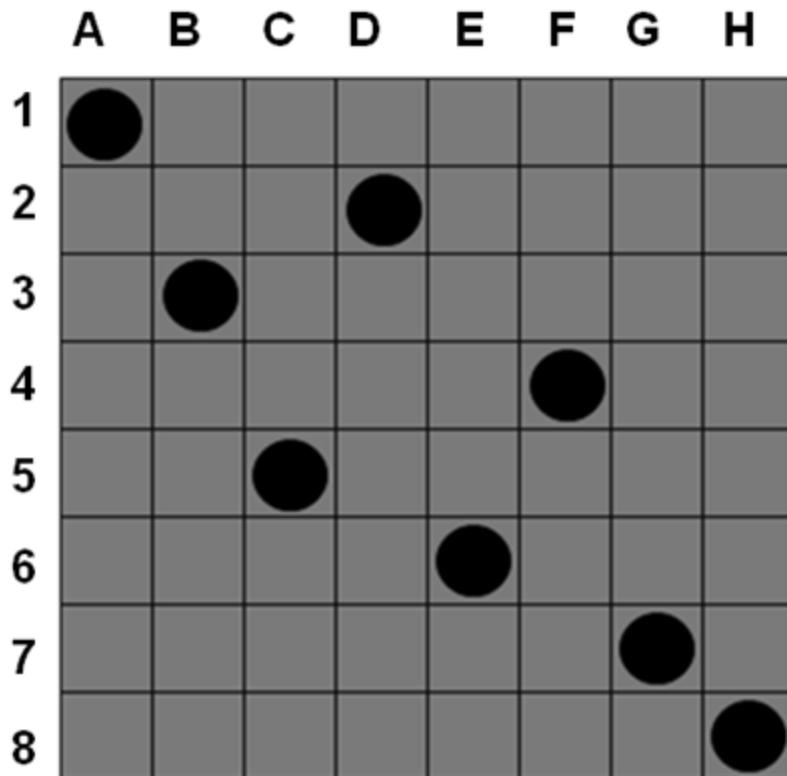
$$f = \frac{1}{(B + a)}$$

B = Jumlah busur yang terpotong

a = bilangan kecil untuk menghindari pembagian dengan nol

Masalah 8-queens





Kromosom:



Nilai gen
menyatakan
posisi baris

Permutasi
Biner?
Integer?
Real?

Fungsi *Fitness*

$$f = \frac{1}{(Q + a)}$$

$Q \rightarrow$ adalah jumlah queen yang saling mengancam
 $a \rightarrow$ bilangan yang diangap sangat kecil

Pemotongan bahan

11 cm

Awas !

Kertas ini mahal

15 cm

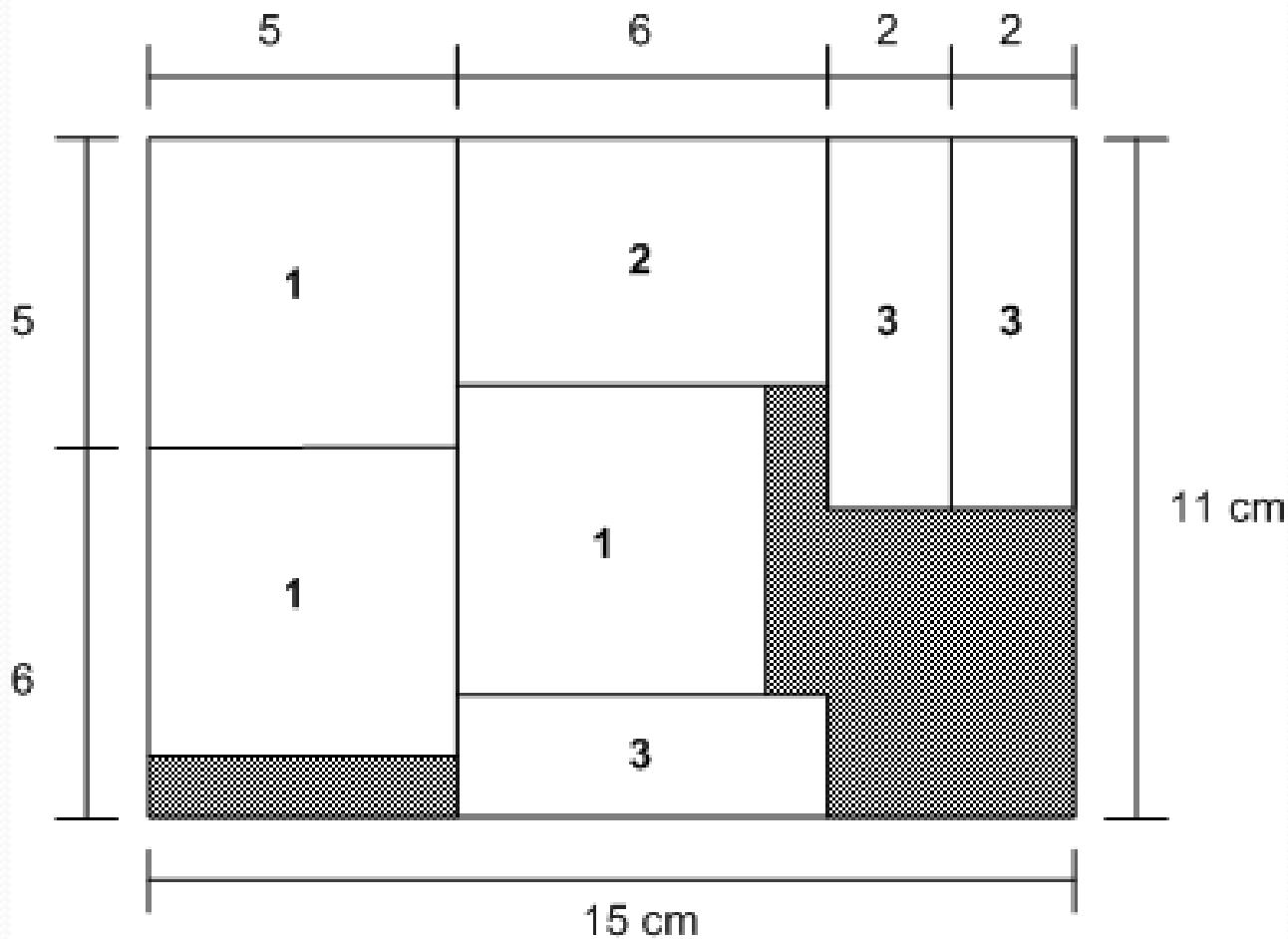
Order potongan

No.	Ukuran (cm ²)	Jml
1	5 x 5	3
2	4 x 6	2
3	2 x 6	3

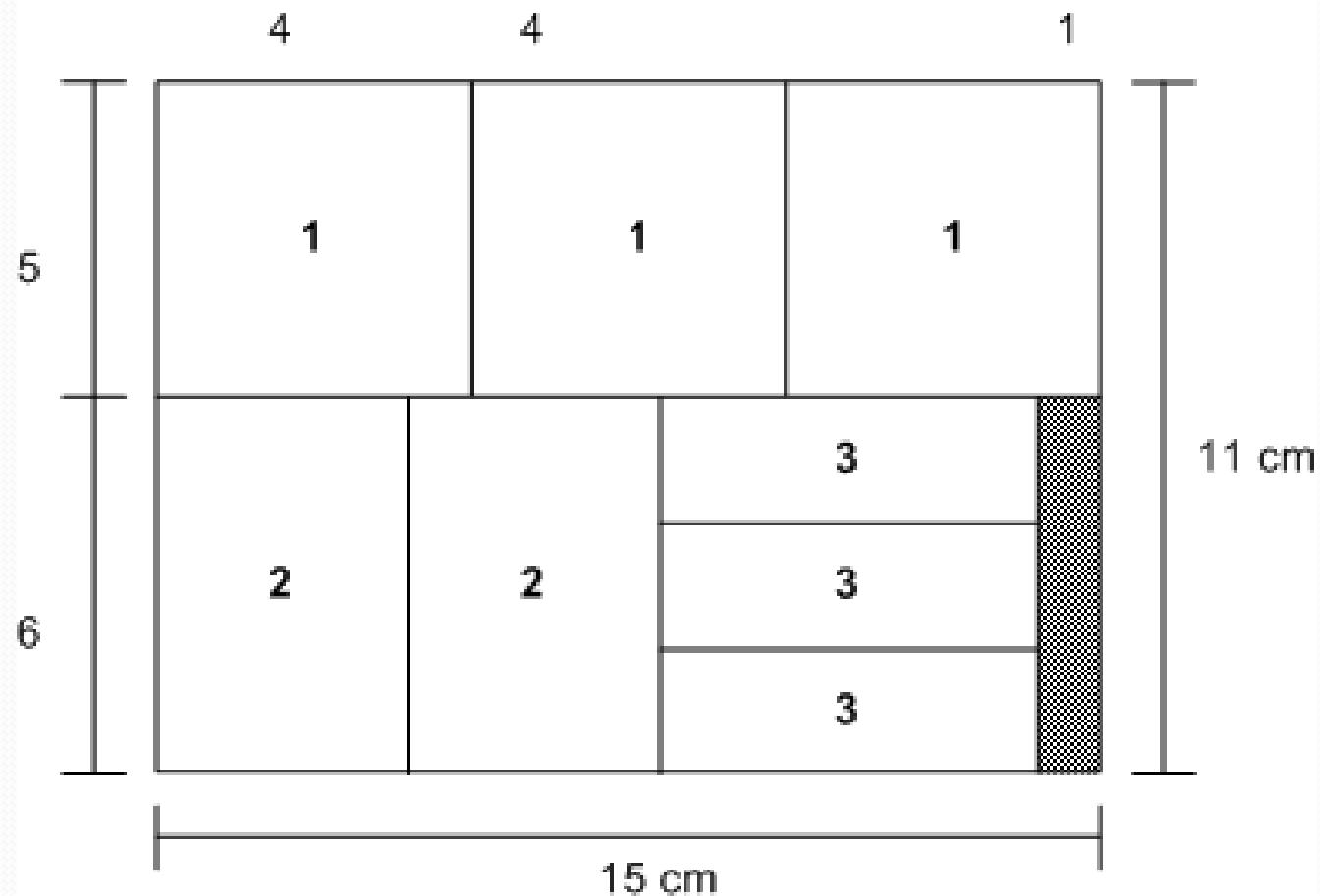
Pola potongan yang memaksimasi order dan meminimasi sisa bahan?

Kasus yang mirip:
Pemasangan Iklan

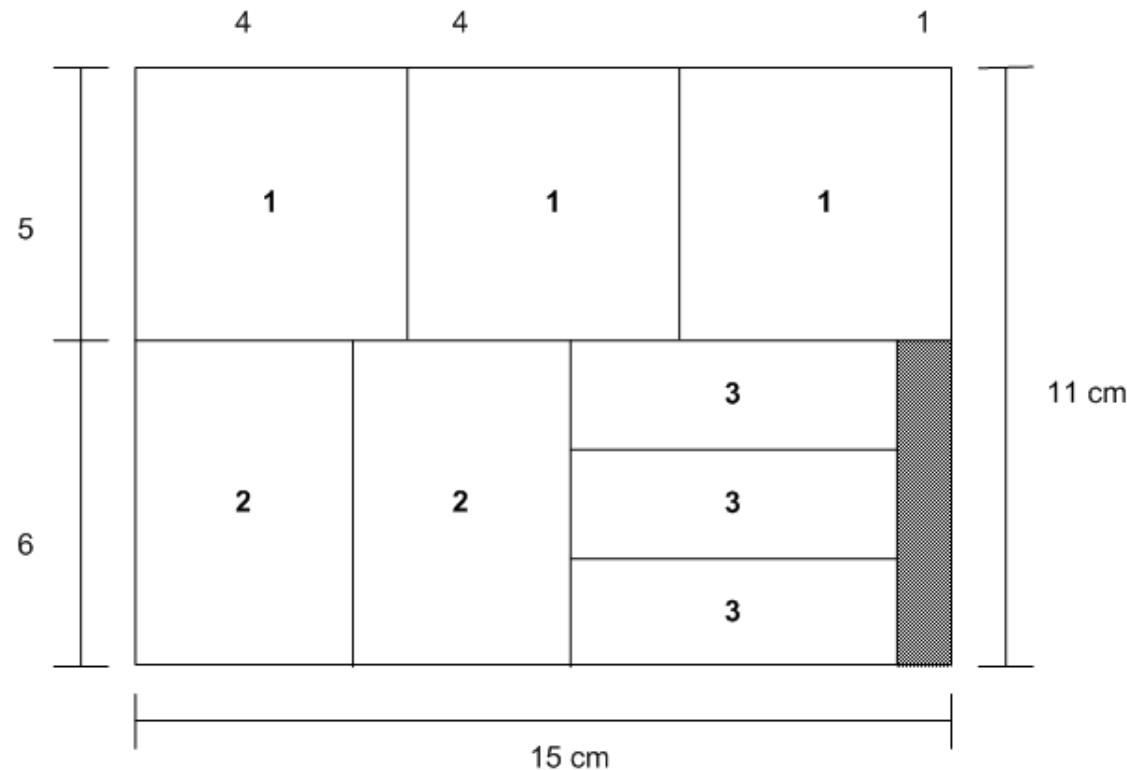
Pola pemotongan 1



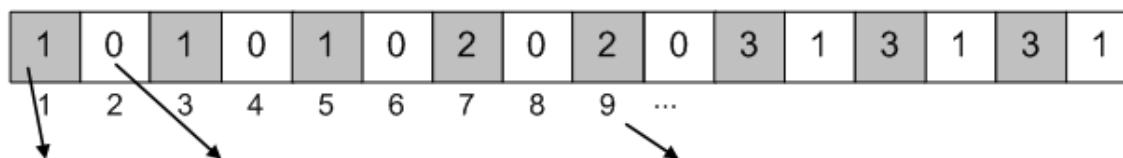
Pola pemotongan 2



Pola pemotongan:



Kromosom:



Nilai gen pada indeks ganjil (warna abu-abu) menyatakan **nomor order**

Nilai gen pada indeks genap (warna putih) menyatakan **orientasi order** untuk nomor order pada gen di sebelah kirinya.

Indeks ganjil menyatakan **urutan pemotongan** dari kiri atas ke kanan bawah. Jadi, urutan pemotongan adalah nomor order pada indeks 1, 3, 5, 7, 9, ...

Kromosom lain?

Fungsi *fitness*

$$f = \frac{P}{(S + a)}$$

P = Permintaan (order) yang terpenuhi

S = Sisa bahan

a = Bilangan kecil untuk menghindari pembagian dengan nol

Pemotongan Bahan

- **Urutan posisi**
 - Dari kiri atas ke kanan bawah?
 - Dari atas kiri ke bawah kanan?
 - Acak → kromosom perlu posisi baris kolom
- **Bentuk potongan**
 - Segitiga
 - Lingkaran
 - Bintang
- **Bahan**
 - Kain
 - Kulit
 - Kaca

Peramalan Data *Time Series*

Tanggal	Penjualan (miliar rupiah)
14 Dec 2007	99,9573
13 Dec 2007	99,8459
12 Dec 2007	98,8708
11 Dec 2007	98,7480
10 Dec 2007	98,3897
09 Dec 2007	97,6780
08 Dec 2007	97,3797
...	...
...	...
...	...
03 Jan 2006	90,7597
02 Jan 2006	90,5770
01 Jan 2006	89,3897

Model Matematis

$$z = a_0 + a_1 y_1 + a_2 y_2 + \dots + a_k y_k$$

Kromosom

a_0	a_1	...	a_k
0,0371	0,5133	...	0,4911

Real
Biner?
Integer?
Permutasi?

Fungsi *Fitness*

$$f = \frac{1}{(E + a)}$$

$E \rightarrow$ Tingkat Error (MAPE) untuk semua data histori.
 $a \rightarrow$ bilangan yang diangap sangat kecil

Penjadwalan Kuliah

No	Kode MK	Pertemuan ke-	Kode Kelas	Jumlah mhs	Kode Dosen	Status dosen	JFA
1	CS3143	1	IF-33-01	40	SBK	Dalam	Lektor
2	CS2314	1	IF-34-01	35	TWG	LB	Guru Besar
3	CS2314	2	IF-34-01	35	TWG	LB	Guru Besar
...
400	CS4923	2	IF-32-03	17	DSS	Dalam	Lektor

Kromosom

Ruang 1

	SN	SL	RB	KM	JM	SB
07-09	50		7	112	187	
09-11	32	1	101		400	
11-13		3		9		
13-15				6		
15-17	43	21	29	332		15

Gen ke-1

Ruang 36

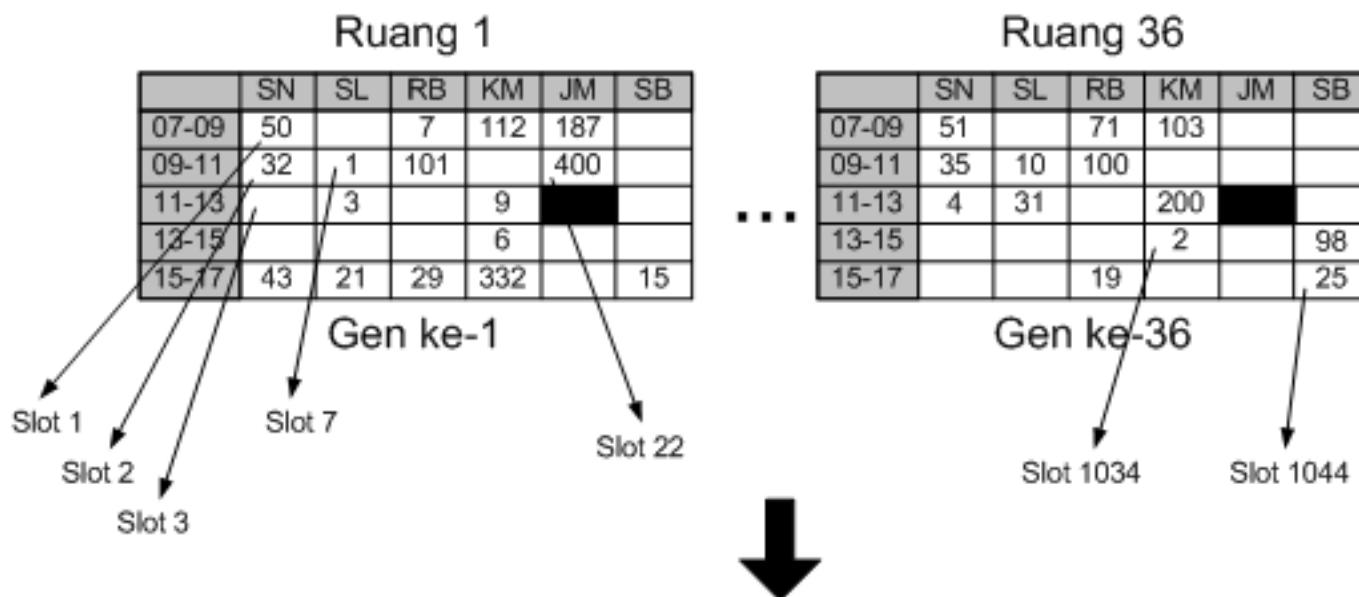
	SN	SL	RB	KM	JM	SB
07-09	51		71	103		
09-11	35	10	100			
11-13	4	31		200		
13-15				2		98
15-17			19			25

Gen ke-36

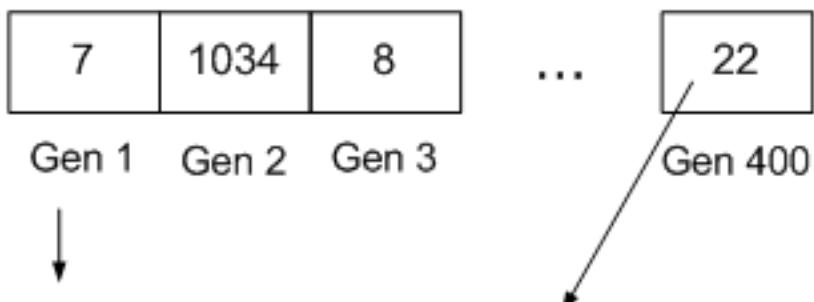
....

Kromosom lain?

Kromosom dengan 36 gen:



Kromosom dengan 400 gen:



Posisi gen menyatakan nomor urut pertemuan kuliah

Nilai-nilai gen menyatakan nomor slot

Batasan

Batasan	Nilai prioritas	Alasan
Tidak bentrok	24	Tidak bentrok merupakan suatu <i>hard constraint</i> (batasan keras) yang sebaiknya tidak dilanggar. Sehingga batasan ini lebih utama dibandingkan harus memenuhi permintaan dosen berjabatan akademik Guru Besar dan Lektor Kepala atau batasan-batasan yang lain. Misal batasan ini diberikan nilai prioritas sebesar 24.
Permintaan Lektor dan Asisten Ahli	4	Kemudian, batasan permintaan dosen berjabatan akademik Guru Besar dan Lektor Kepala diberi nilai prioritas sebesar 12.
Status Dosen	4	Misalkan ketiga batasan yang lainnya diberikan nilai prioritas yang sama, yaitu 4.
Ruang Kuliah	4	

Pelanggaran Batasan

- J_a = Jumlah jadwal pertemuan yang bentrok kali 24.
- J_b = Jumlah jadwal yang tidak sesuai permintaan dosen berjabatan Guru Besar dan Lektor Kepala dikali 12.
- J_c = Jumlah jadwal yang tidak sesuai permintaan dosen Lektor dan Asisten Ahli dikali 4.
- J_d = Jumlah jadwal yang tidak sesuai ruang kuliahnya dikali 4.
- J_e = Jumlah jadwal yang Tidak Sesuai Status dosen dikali 4.

Fungsi *Fitness*

$$f = \frac{1}{(B + a)}$$

$$B = J_a + J_b + J_c + J_d + J_e$$

Operator Evolusi?

- **Seleksi Ortu:**
 - Roullete Wheel, Tournament Selection, Ranking, Scaling, dst.
- **Rekombinasi**
 - Biner, Integer, Real, Permutasi, Gabungan dari rekombinasi tsb.
- **Mutasi**
 - Biner, Integer, Real, Permutasi, Gabungan dari rekombinasi tsb.
- **Seleksi Survivor**
 - Generational atau Steady State?
 - Deterministik atau Probabilistik?
- **EAs Model**
 - Simple Vs Advanced?

A photograph of the National Monument (Monas) in Jakarta, Indonesia. The monument is a tall, white, monolithic obelisk topped with a golden spire. It stands on a large, light-colored base. The sky is blue with scattered white clouds. In the foreground, there is a grassy area and a paved walkway. Several people are walking around the base of the monument. In the background, there are other buildings and some industrial structures like towers.

Your cases?

Question?



Kasus 1: Pengangkutan Barang

- Diambil dari TA mahasiswa S1-Informatika IT Telkom angkatan 2006: Naufal Mikhdzam Ar Rozi (113061051)
- Mengimplementasikan Algoritma Genetika untuk menemukan solusi optimal
- Melakukan analisis terhadap sistem Algoritma Genetika yang telah dibangun
- Dibangkitkan 3 studi kasus

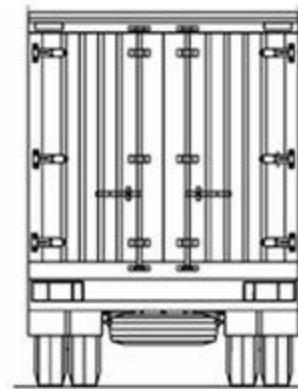
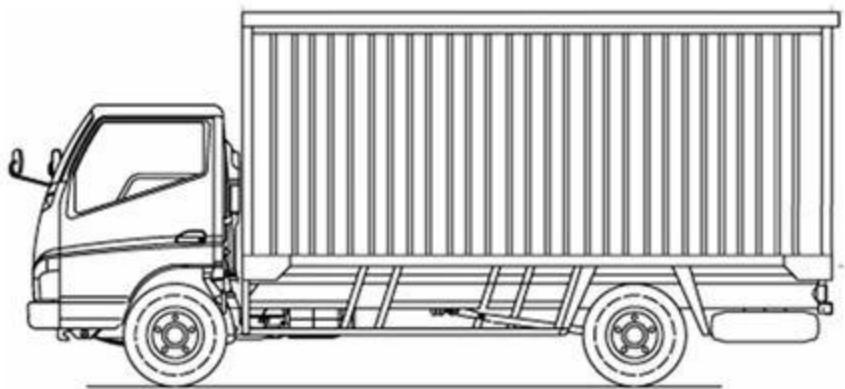
No.	Properti	Keterangan
1.	ID Barang	Identitas dari suatu barang
2.	Panjang	Panjang barang (cm)
3.	Lebar	Lebar barang (cm)
4.	Tinggi	Tinggi barang (cm)
5.	Volume	Volume barang (cm^3)
6.	Berat	Berat barang (kilogram)
7.	Kekuatan	Kekuatan barang terhadap beban yang menimpanya (kilogram). Apabila barang memiliki kekuatan 50 kg, maka barang tersebut dapat ditimpa oleh barang-barang dengan maksimal berat total barang yang menimpa adalah 50 kg.
8.	Tingkat Prioritas	Tingkat prioritas barang yang bernilai 1 sampai dengan 5. Tingkat prioritas tertinggi = 5.
9.	Jumlah sisi yang dapat dijadikan alas	Berapa jumlah sisi yang dapat dijadikan alas ketika barang diletakkan (Nilai = 1 sampai 6).
10.	Sisi-sisi yang dapat dijadikan alas	Sisi-sisi mana saja yang dapat dijadikan alas ketika barang diletakkan (Nilai = 1 sampai 6).

Kasus Tipe A



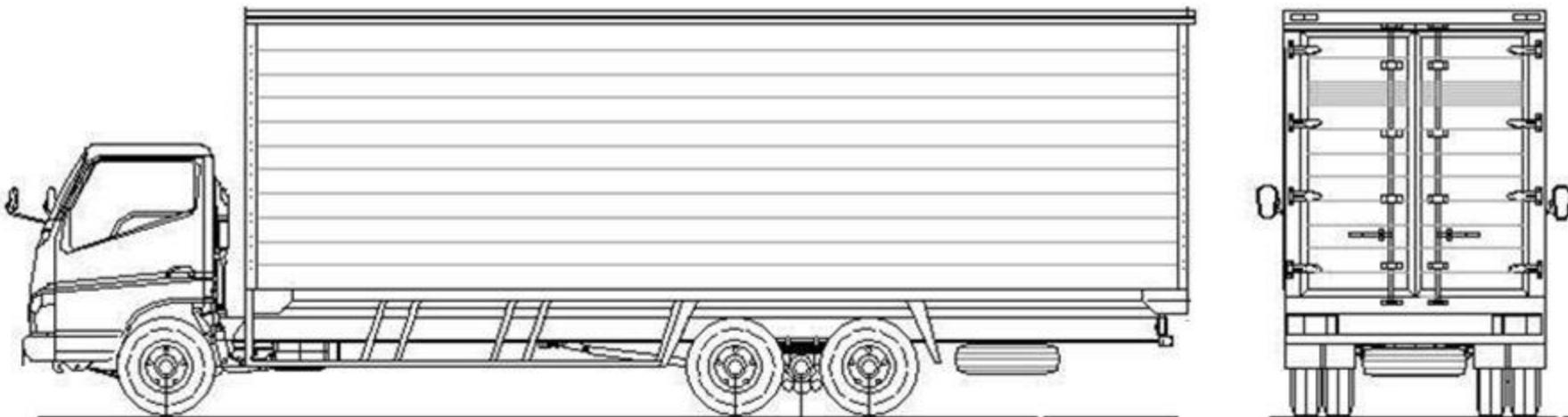
Nama Angkutan	Mitsubishi Colt L300
Panjang	220 cm
Lebar	155 cm
Tinggi	135 cm
Maksimum Beban	1000 kg [6]

Kasus Tipe B



Nama Angkutan	Mitsubishi Colt Diesel FE 73 HD
Panjang	430 cm
Lebar	197 cm
Tinggi	185 cm
Maksimum Beban	4000 kg [6]

Kasus Tipe C



Nama Angkutan	Mitsubishi Tronton FE 74
Panjang	710 cm
Lebar	207 cm
Tinggi	218 cm
Maksimum Beban	8000 kg [6]

Representasi Kromosom

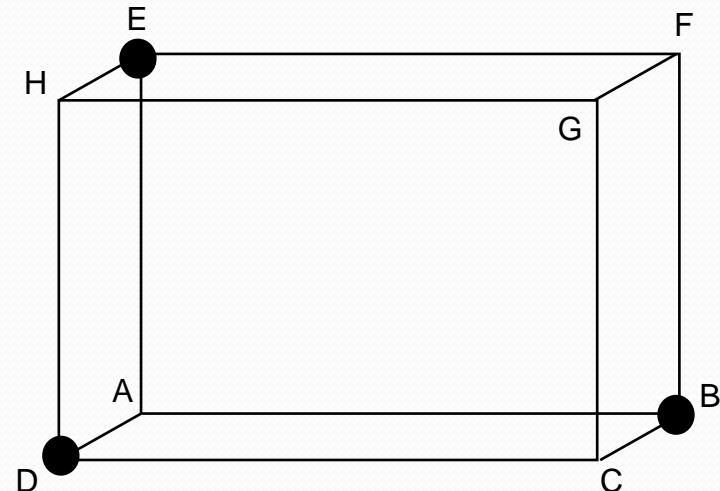
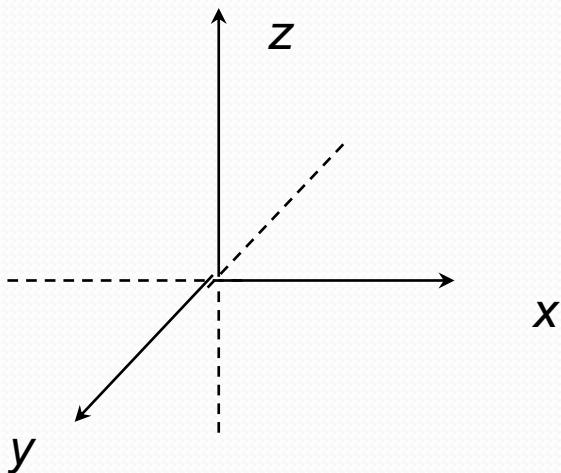
Bagian I

Bagian II

Bagian III

- Bagian I: representasi permutasi yang menggambarkan urutan peletakan barang.
- Bagian II: representasi integer yang menggambarkan sisi barang yang digunakan sebagai alas dalam peletakan.
- Bagian III: representasi biner yang menggambarkan posisi sisi yang dijadikan alas yaitu horizontal atau vertikal

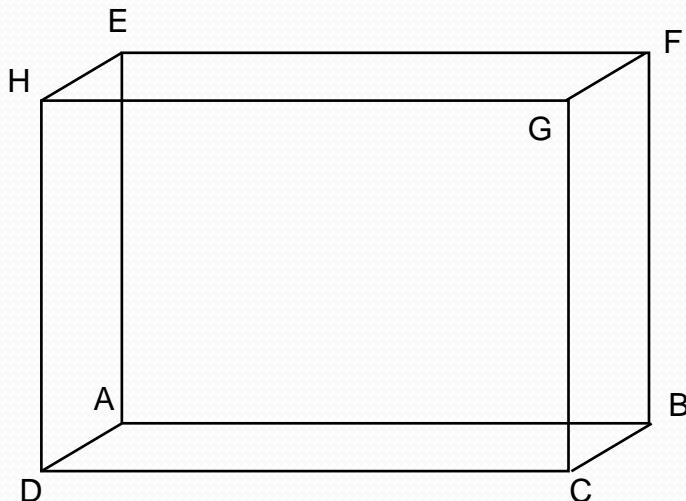
Peletakan barang



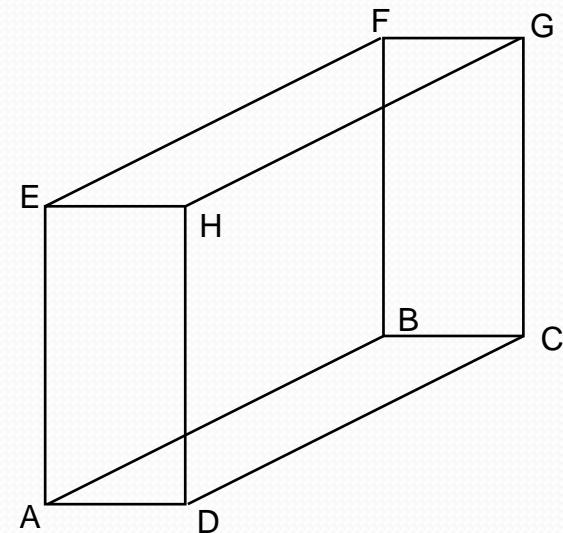
Penomoran sisi barang

No. Sisi	Sisi
1	ABCD
2	ABFE
3	BCGF
4	CDHG
5	DAEH
6	EFGH

Posisi peletakan: horizontal/vertikal

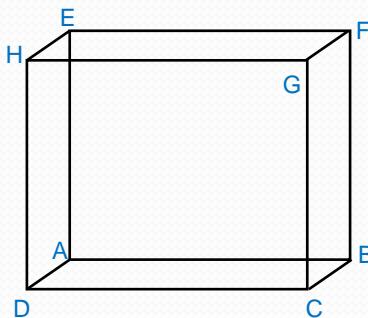


Horizontal

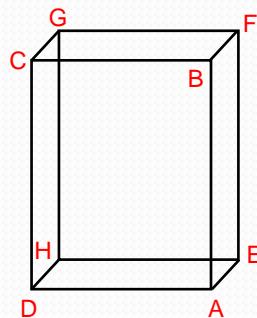


Vertikal

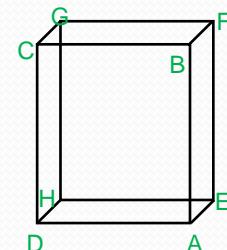
Individu Vs Kromosom



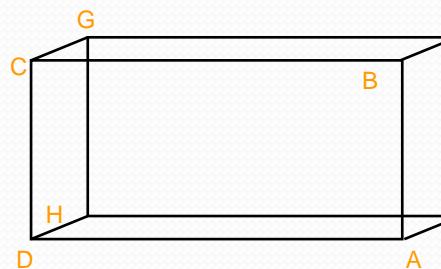
ID = 3



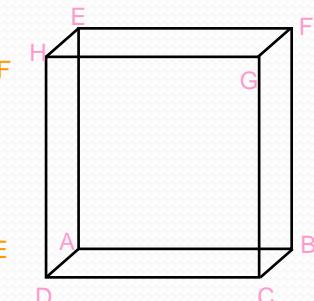
ID = 1



ID = 2



ID = 5

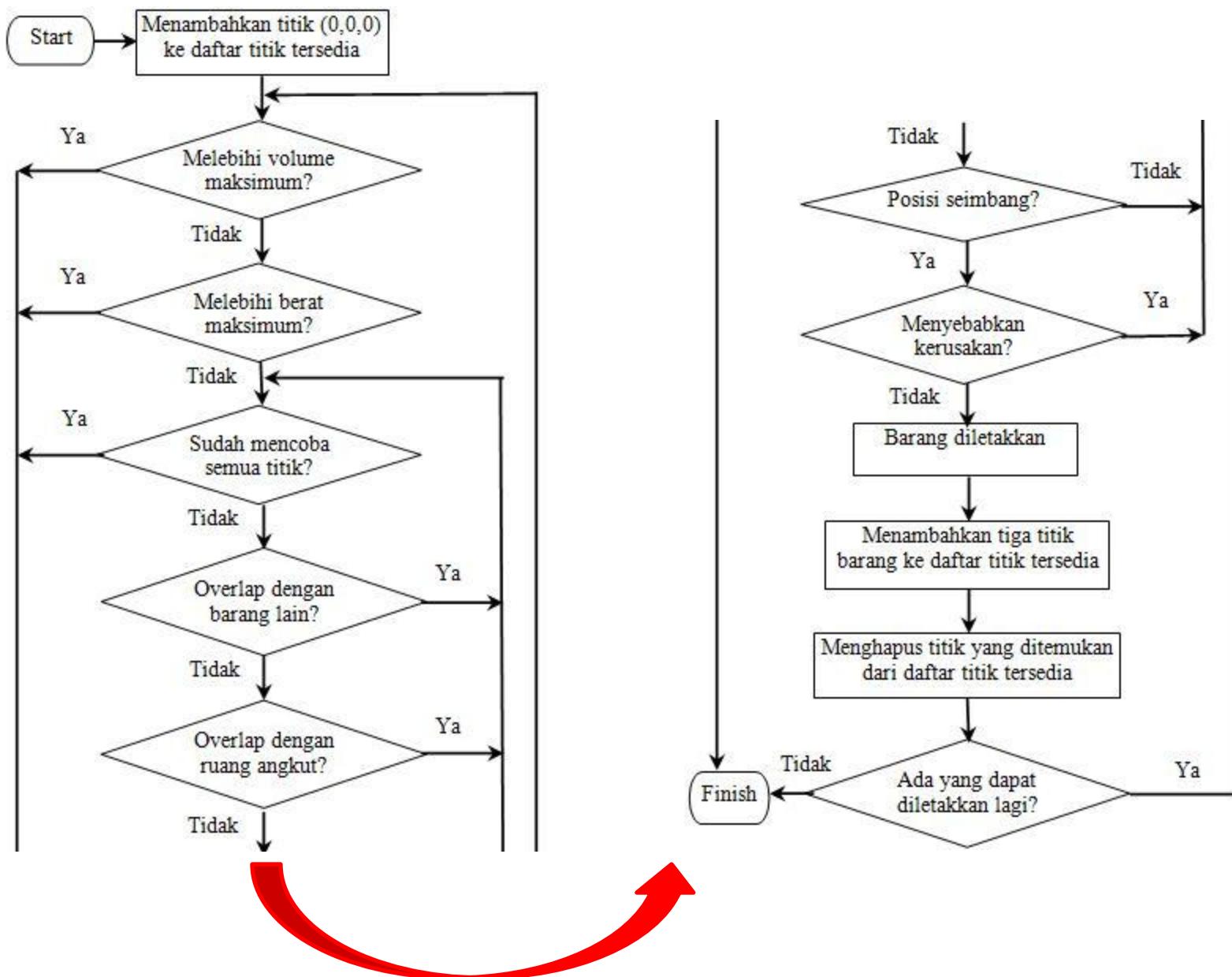


ID = 4



3	1	2	5	4	1	5	5	5	1	1	0	1	1	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

Pembangkitan kromosom



Fungsi Fitness

$$fitness = \frac{Value}{Maximum\ Value} \times 100$$

Value : Jumlah biaya dari barang-barang yang terangkut

Maximum Value: Jumlah biaya dari semua barang yang ada

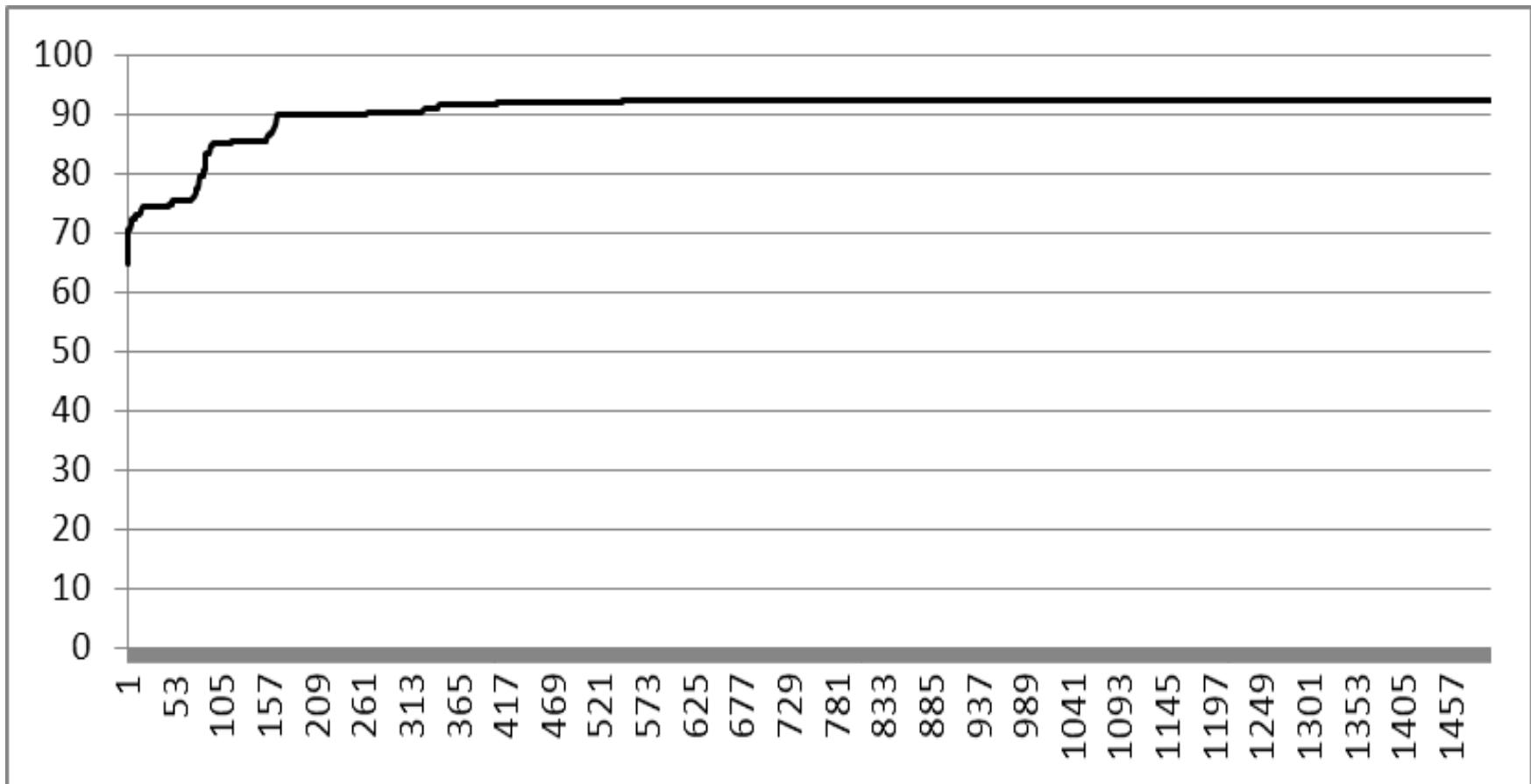
Biaya pengiriman tiap kilogram untuk tiap tingkat prioritas

Tingkat Prioritas	Biaya (Rp/kg)
1	5000
2	8000
3	15000
4	25000
5	50000

Operator Evolusi

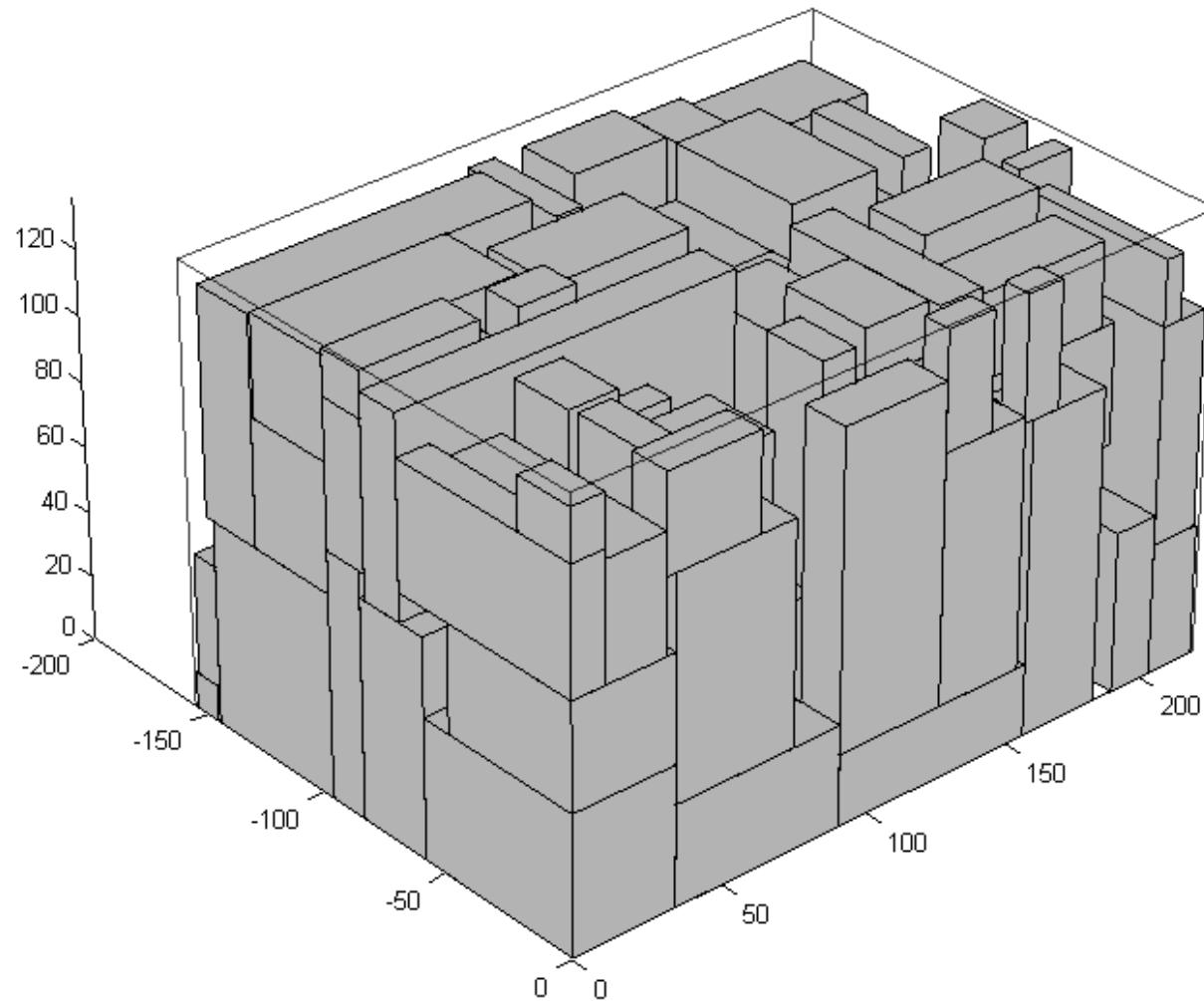
- **Seleksi Orangtua**
 - Tournament Selection
 - Tournament size = 10% dari ukuran populasi
- **Rekombinasi**
 - Bagian I: order crossover
 - Bagian II dan III: uniform crossover
- **Mutasi**
 - Bagian I: swap mutation
 - Bagian II: creep mutation
 - Bagian III: binary mutation
- **Seleksi survivor** : Generational model dengan elitisme

Hasil Pengujian Kasus A

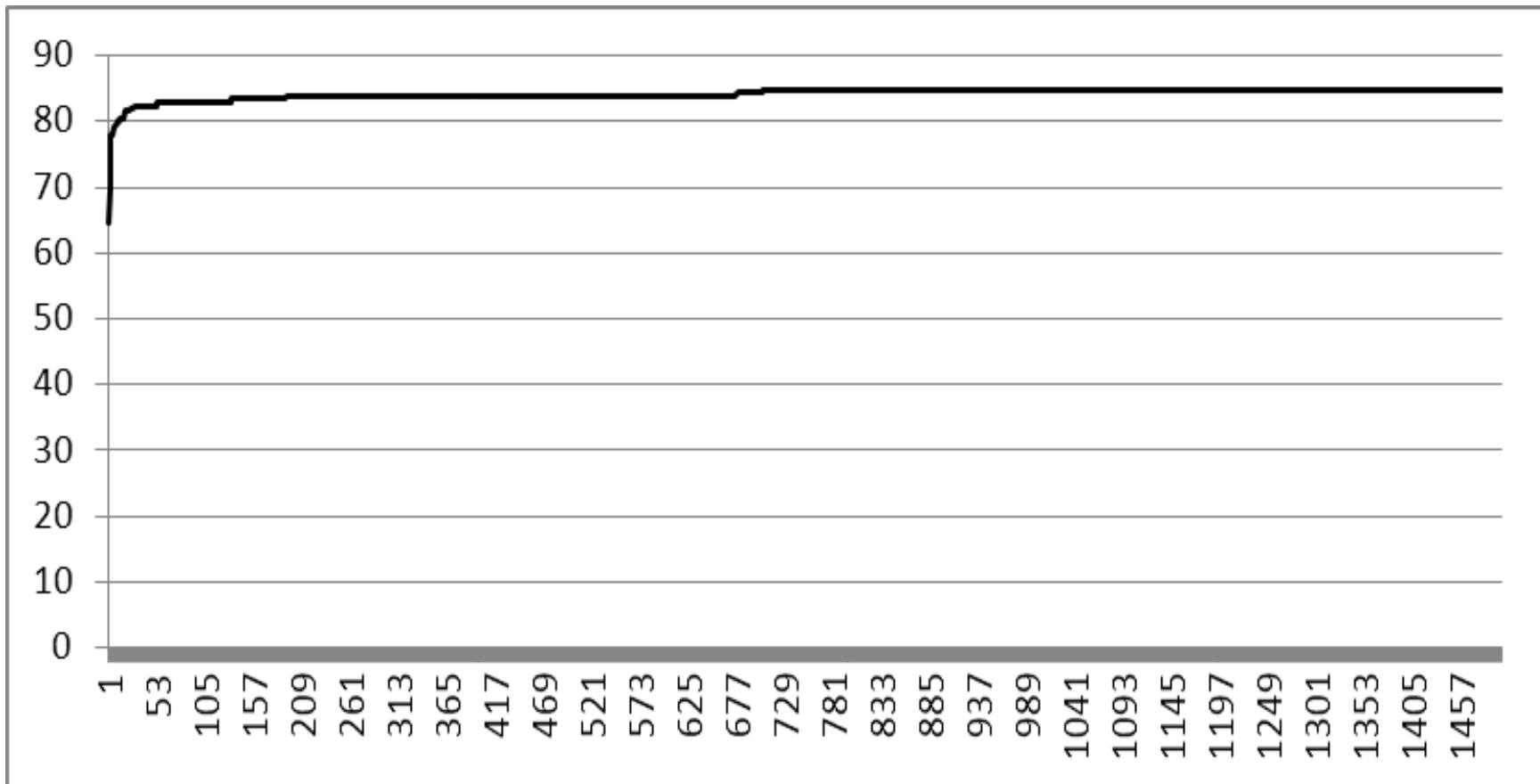


Fitness Terbaik 92.554

Visualisasi Kasus A

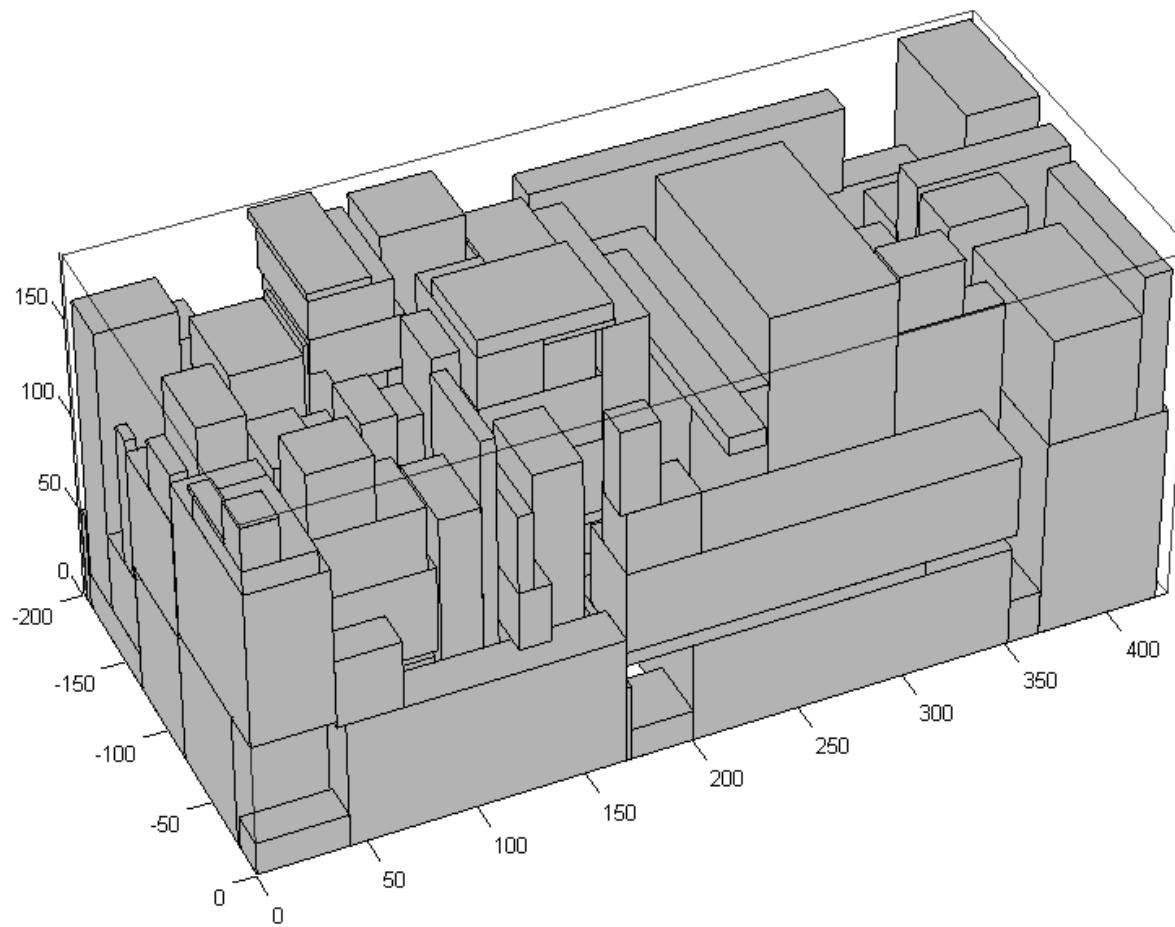


Hasil Pengujian Kasus B

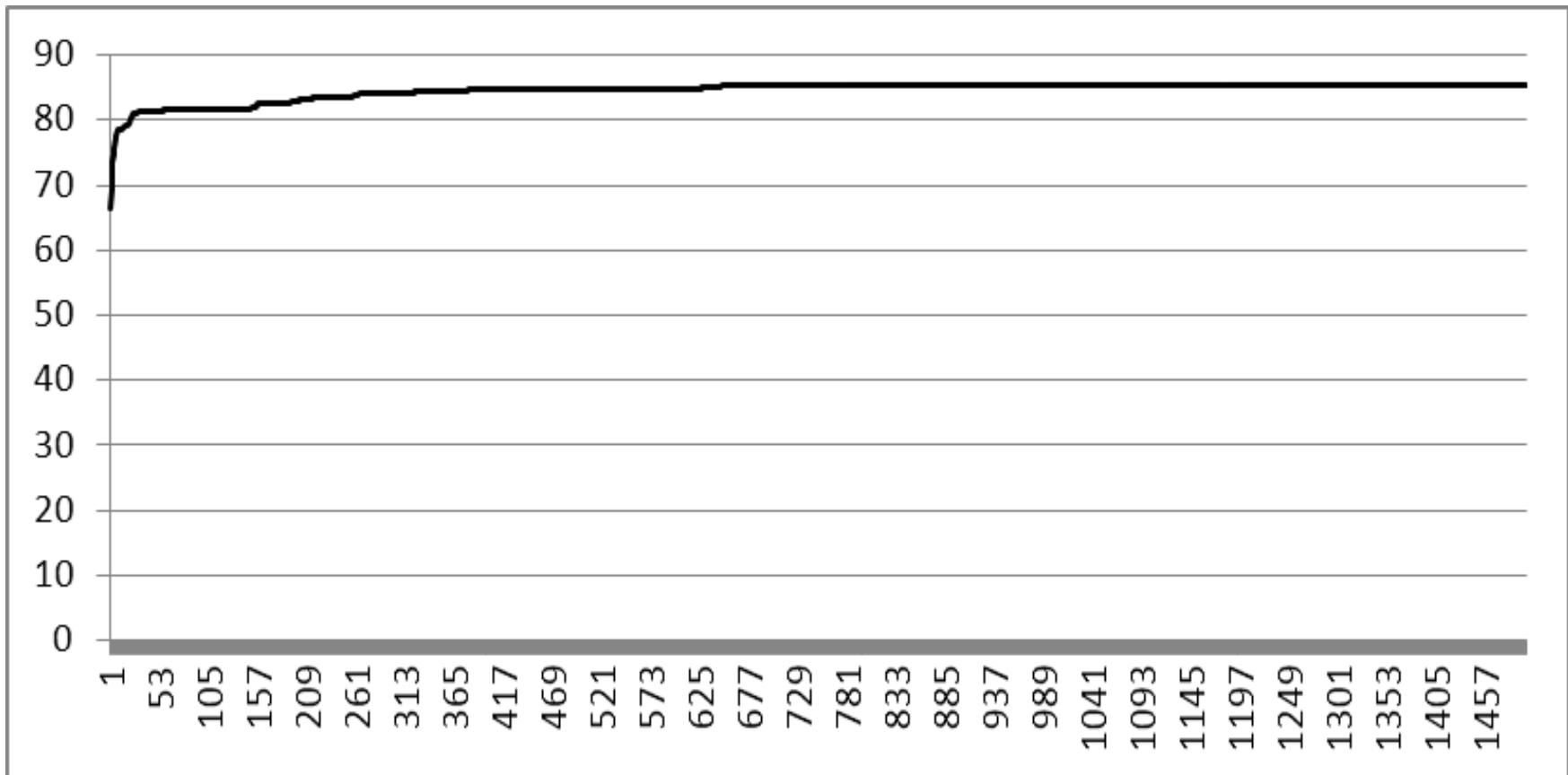


Fitness Terbaik 84.920

Visualisasi Kasus B

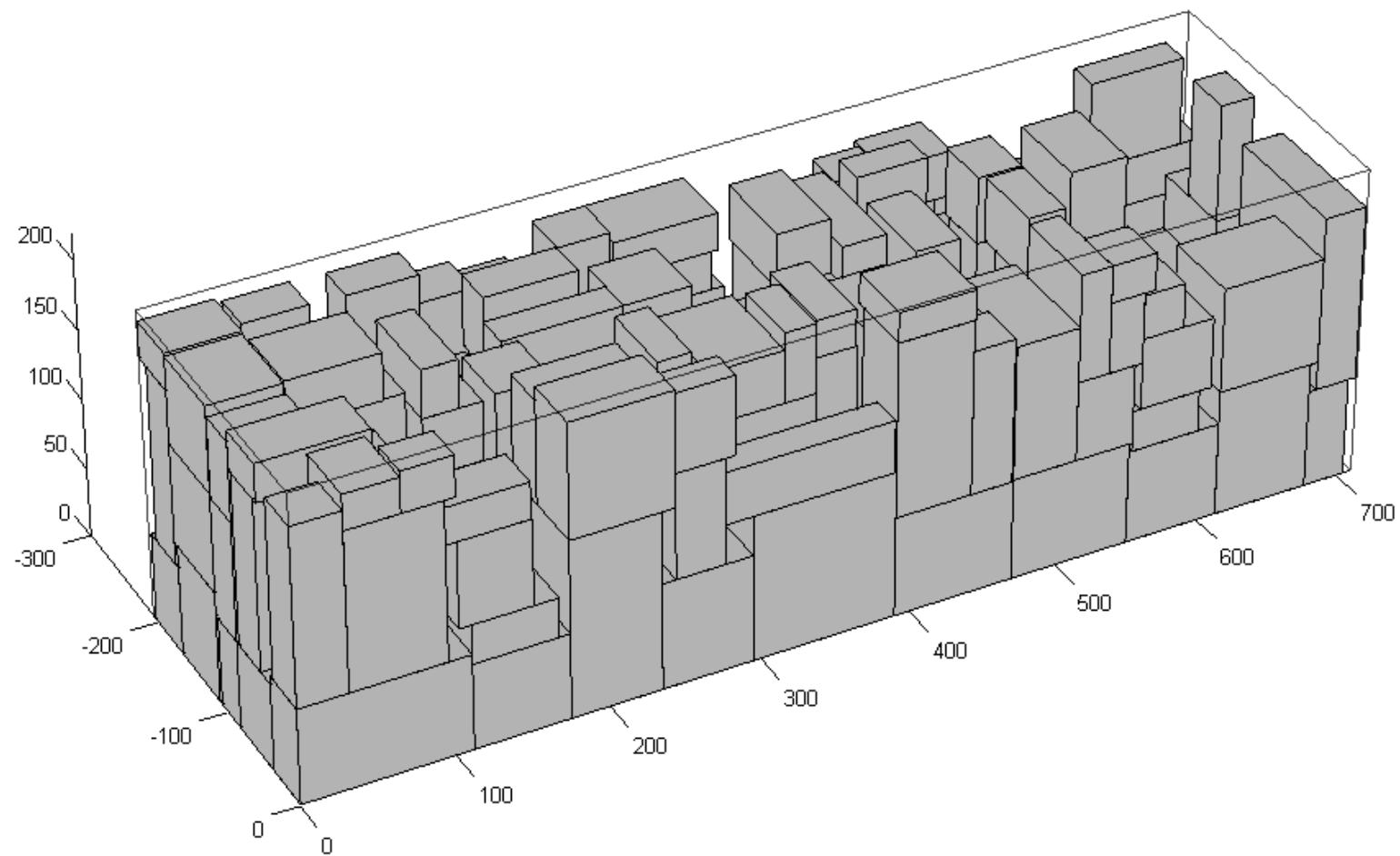


Hasil Pengujian Kasus C



Fitness Terbaik 85.366

Visualisasi Kasus C



Analisa Performansi GA

- Ruang Solusi

$$100! \times 6^{100} \times 2^{100} = 7.7291 \times 10^{265}$$

- Ruang yang dijelajahi 150.000 calon solusi

$$150000 \div (7.7291 \times 10^{265}) = 1.9407 \times 10^{-259}$$

- Performansi GA

- Tipe kasus A: 92.554
- Tipe kasus B: 84.920
- Tipe kasus C: 85.366

Aplikasi GA

- *Scheduling problems*
- *Chemistry chemical manufacturing*
- *Medicine*
- *Data mining and data analysis*
- *Geometry*
- *Finance and trade*
- *Optimizing distributed protocol*



Question?

Kesimpulan

- Individu pada GA bisa menggunakan empat representasi berbeda: Biner, Integer, Real, dan Permutasi.
- Operasi Seleksi Orang tua dan Seleksi Survivor tidak bergantung pada representasi individu.
- Rekombinasi dan Mutasi harus dipilih sesuai dengan representasi individu.
- Performansi GA bisa dibuktikan secara matematis meskipun masih sederhana dengan banyak asumsi dan pendekatan.
- Advanced GA sesuai untuk masalah yang sangat kompleks.

Daftar Pustaka

- [SUYo8] Suyanto, 2008, Evolutionary Computation: Komputasi Berbasis “Evolusi” dan “Genetika”, penerbit Informatika Bandung.
- [RAN88] Ranganathan, B. G, 1988, “Origins?”, Pennsylvania, The Banner Of Truth Trust.
- [ADN07] Adnan Oktar, 2007, "Mekanisme Khayalan Teori Evolusi", www.evolutiondeceit.com/indonesian/keruntuhan3.php

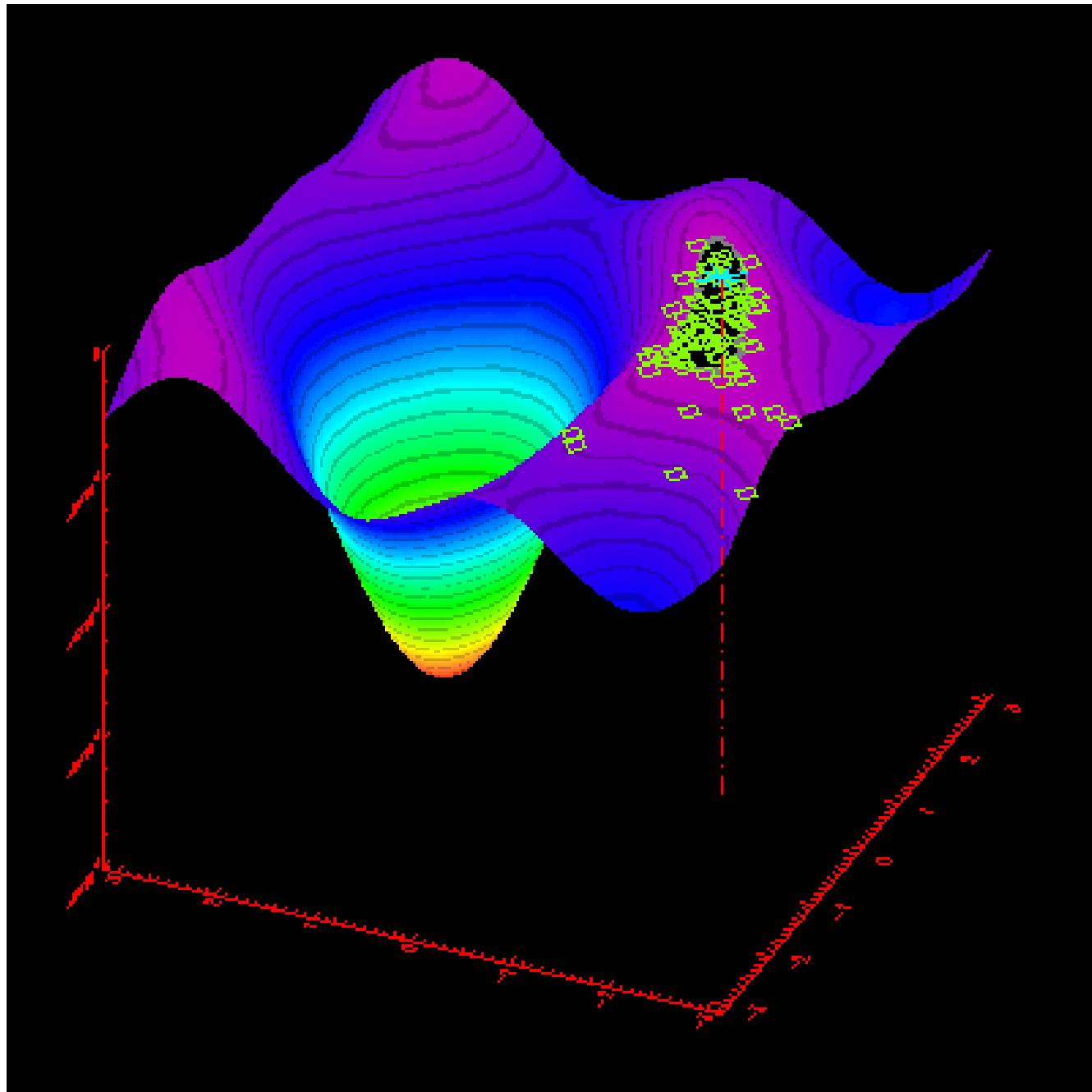
Evolution Strategies (ES)

Dr. Suyanto, S.T., M.Sc.
HP/WA: 0812 845 12345

Intelligence Computing Multimedia (ICM)
Informatics faculty – Telkom University

How the predators catch a prey?







Intro

- ES diperkenalkan pertama kali oleh Ingo Rechenberg di Jerman pada era 1970-an.
- Ide-ide pada ES sangat mirip dengan GA. Tetapi, Ingo Rechenberg mengembangkan ES secara terpisah dari GA.
- Berbeda dengan GA yang bisa menghasilkan perubahan signifikan, ES justru berbasis pada prinsip sebab akibat: “**Perubahan kecil menghasilkan efek-efek yang kecil juga**”.
- ES sering digunakan untuk eksperimen-eksperimen empiris, khususnya permasalahan optimasi numerik.
- Kecepatan proses ES lebih baik dibandingkan dengan GA untuk masalah optimasi bernilai **real**.

Intro

- Awalnya, ES menggunakan populasi yang hanya beranggotakan **satu kromosom** dan hanya menggunakan mutasi (**tanpa rekombinasi**) untuk menghasilkan satu anak.
- Jika anak yang dihasilkan lebih baik, maka anak tsb. menggantikan orangtuanya pada generasi berikutnya.
- Jadi, pada setiap generasi, populasi tetap beranggotakan **hanya satu kromosom**.

Spesifikasi teknis ES

Representasi	Vektor bernilai real
Seleksi orangtua	<i>Uniform random</i>
Rekombinasi	<i>Discrete</i> atau <i>Intermediary</i>
Mutasi	<i>Gaussian perturbation</i>
Seleksi survivor	(μ, λ) atau $(\mu + \lambda)$
Ciri khusus	<i>Self-adaptation</i> pada <i>mutation step sizes</i>

Pseudo-code ES

$t = 0$

Inisialisasi populasi: satu kromosom $x^t = x_1^t, \dots, x_n^t$

LOOP sampai kondisi berhenti dipenuhi

Ambil z_i secara acak dari distribusi normal untuk $i = 1, \dots, n$

$y_i^t = x_i^t + z_i$

IF $f(y^t) > f(x^t)$ **THEN** $x^{t+1} = y^t$ **ELSE** $x^{t+1} = x^t$

$t = t+1$

END LOOP

Pseudo-code ES

- Pada *pseudo-code* di atas, nilai-nilai z diambil secara acak dari distribusi normal $N(\xi, \sigma)$
- Nilai rata-rata ξ dibuat sama dengan o dan variansi σ disebut sebagai *mutation step size*
- σ dibuat bervariasi menggunakan aturan yang disebut “*1/5 success rule*”. Aturan ini melakukan proses perubahan σ pada setiap periode iterasi tertentu (misal k iterasi) menggunakan rumus:

$$\sigma = \frac{\sigma}{c} \text{ jika } p_s > \frac{1}{5}$$

$$\sigma = \sigma c \text{ jika } p_s < \frac{1}{5}$$

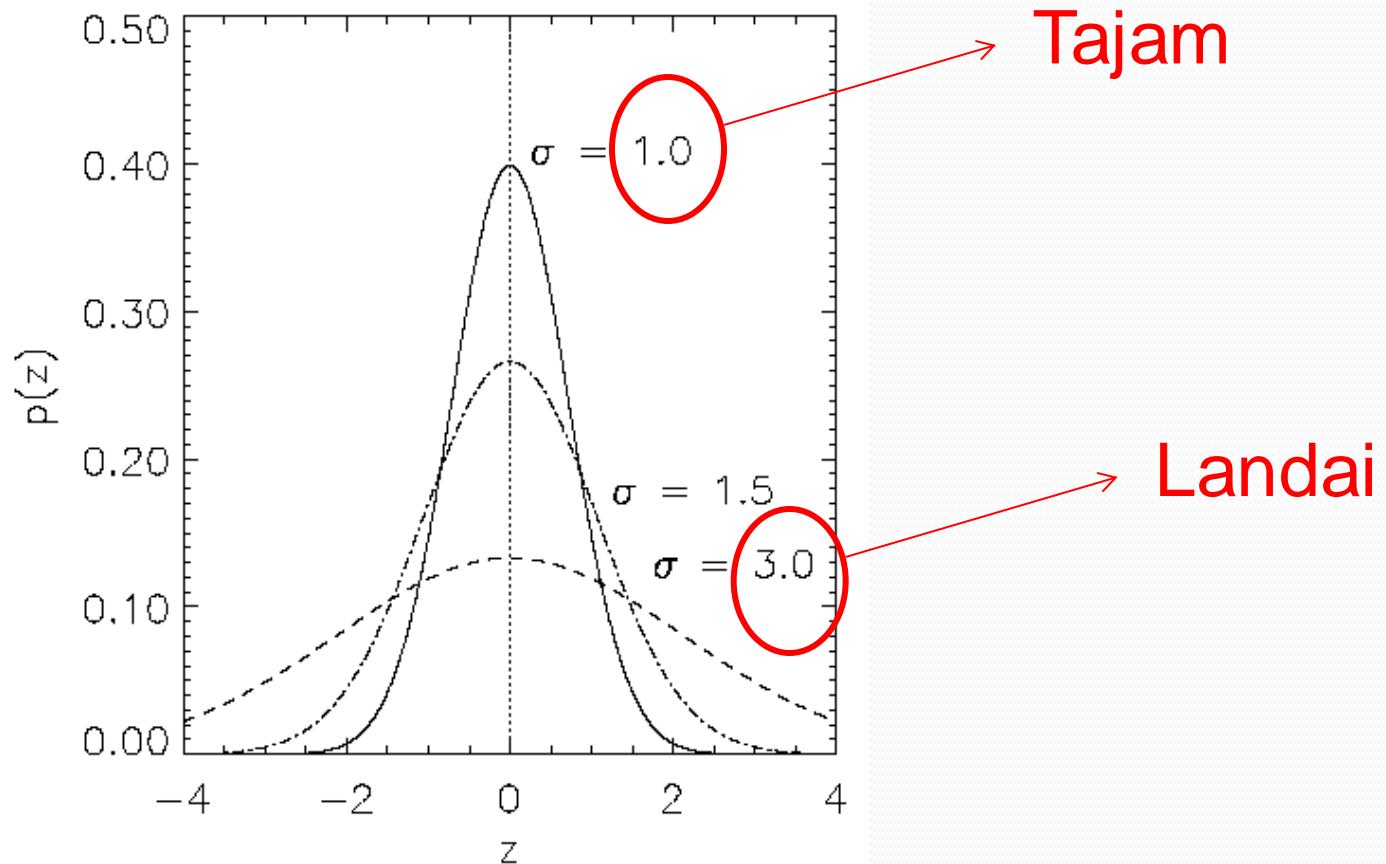
$$\sigma = \sigma \text{ jika } p_s = \frac{1}{5}$$

Jika sukses $> 20\%$, maka σ dinaikkan.
Sukses = solusi saat ini lebih baik dari solusi sebelumnya.

p_s = persentase dari mutasi yang sukses.

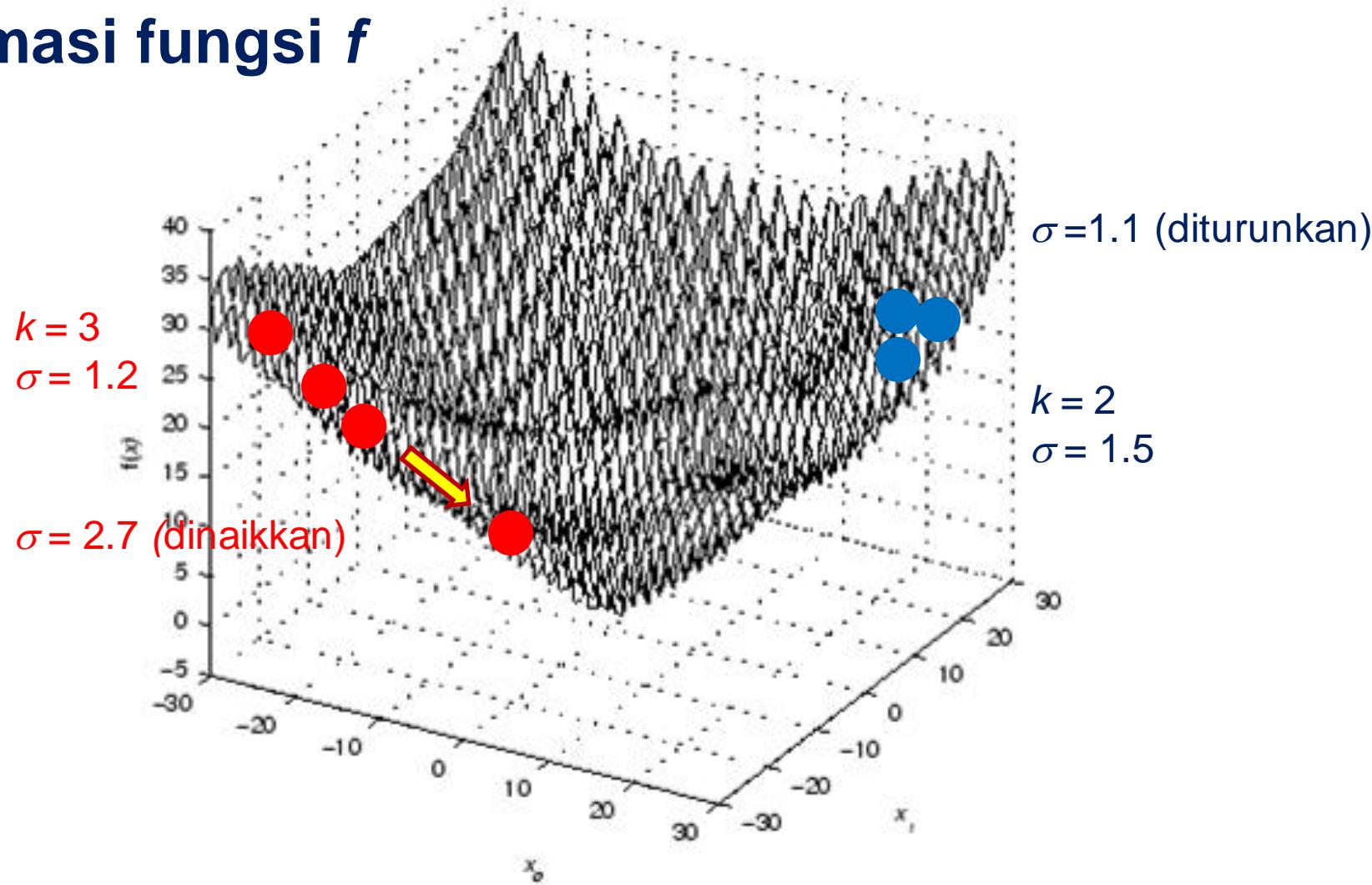
c = konstanta dengan batasan $0.8 \leq c \leq 1$.

Distribusi normal



$$f(\vec{x}) = \sum_{i=0}^{D-1} \left(e^{-0.2} \sqrt{x_{i-1}^2 + x_i^2} + 3(\cos(2x_{i-1}) + \sin(2x_i)) \right)$$

Minimasi fungsi f



Representasi Individu

- Variabel objek: x_1, \dots, x_n
- Parameter-parameter strategi:
 - Mutation step sizes: $\sigma_1, \dots, \sigma_n$
 - Sudut-sudut rotasi: $\alpha_1, \dots, \alpha_k$

Kromosom

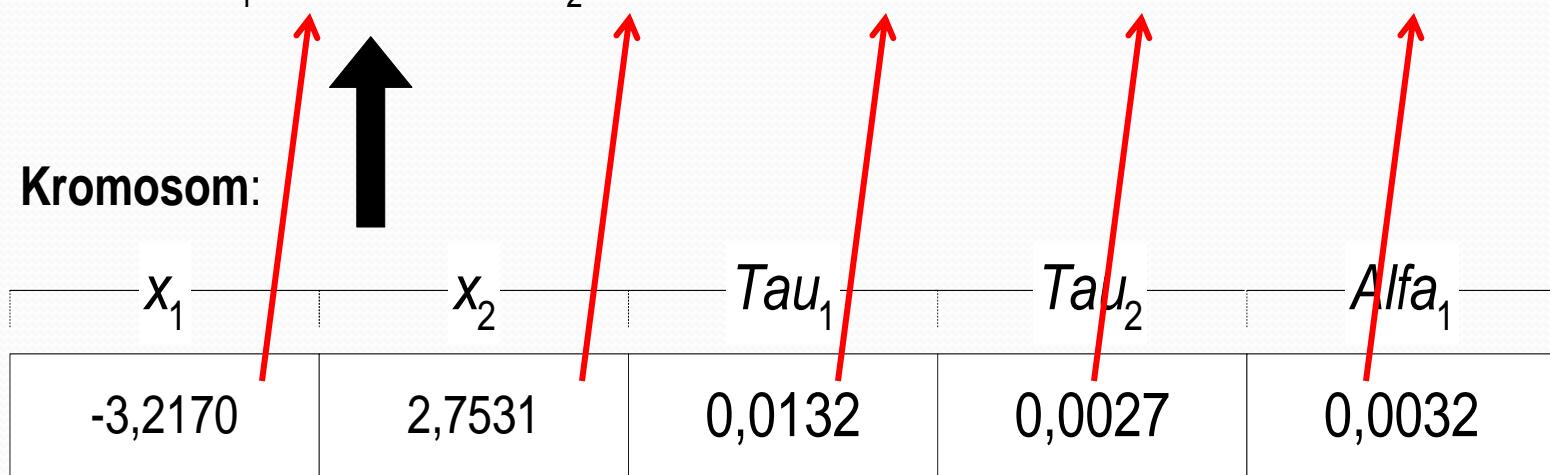
$$\langle x_1, \dots, x_n, \sigma_1, \dots, \sigma_n, \alpha_1, \dots, \alpha_k \rangle$$

- dimana $k = n(n-1)/2$ merupakan jumlah kombinasi pasangan i dan j dari n yang ada.
- Variabel objek mengkodekan nilai-nilai real secara langsung **tanpa konversi**.

Kromosom

Individu: $x_1 = -3,2170$ dan $x_2 = 2,7531$

Kromosom:



Variabel strategi

Seleksi Orangtua

- Kalau di GA, kita bisa menggunakan salah satu dari berbagai macam metode seleksi yang ada.
- Pada ES, proses seleksi orangtua dilakukan secara tidak bias.
- Artinya, setiap kromosom bisa terpilih sebagai orangtua dengan **probabilitas yang sama**.
- Caranya adalah dengan menggunakan distribusi ***uniform***.

Rekombinasi

	Dua orangtua tetap	Dua orangtua yang berubah-ubah untuk setiap gen ke- <i>i</i>
$z_i = (x_i + y_i) / 2$	<u>intermediary</u> <u>lokal</u>	<u>intermediary</u> <u>global</u>
$z_i = x_i$ atau y_i yang dipilih secara acak	<u>discrete</u> <u>lokal</u>	<u>discrete</u> <u>global</u>

Intermediary: z antara x dan y

discrete: z dipilih acak dari x atau y

Lokal: gen didapat dari ortu yg tetap

Global: gen didapat dari ortu yg berbeda2

Mutasi

- Sangat penting untuk menemukan solusi.
- Mengubah nilai gen dengan menambahkan bilangan random (distribusi normal).
- Dalam notasi matematika, suatu gen x_i dimutasi menggunakan rumus

$$x'_i = x_i + N(0, \sigma)$$

Mutasi

- *Mutation step sizes* σ adalah bagian dari kromosom dan σ juga dimutasi menjadi σ' .
- *Mutation step sizes* σ ber-evolusi secara bersama-sama (*co-evolution*) dengan variabel objektif x .
- Hal ini disebut sebagai *Net Mutation Effect* yang dituliskan sebagai

$$\langle x, \sigma \rangle \rightarrow \langle x', \sigma' \rangle$$

Mutasi

- Urutan mutasi merupakan hal yang sangat penting.
- Mutasi σ harus lebih dulu daripada x .
- Alasannya adalah $\langle x', \sigma' \rangle$ dievaluasi dua kali, yaitu:
 - **Primer:** x' adalah bagus jika $f(x')$ bagus.
 - **Sekunder:** σ' adalah bagus jika x' bagus.
- Jika urutannya dibalik, ES tidak bisa bekerja dengan baik untuk menemukan solusi.

Kromosom

Individu: $x_1 = -3,2170$ dan $x_2 = 2,7531$

Kromosom:

x_1	x_2	Tau_1	Tau_2	$Alfa_1$
-3,2170	2,7531	0,0132	0,0027	0,0032

Bagaimana terjadinya mutasi?

- Mutasi tanpa Korelasi menggunakan satu σ
- Mutasi tanpa Korelasi menggunakan σ sebanyak n
- Mutasi dengan Korelasi

Mutasi tanpa Korelasi menggunakan satu σ

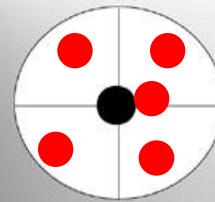
- Metode ini hanya menggunakan satu nilai σ untuk memutasi semua gen yang ada di dalam kromosom.
- Oleh karena itu, suatu kromosom direpresentasikan sebagai
$$\langle x_1, \dots, x_n, \sigma \rangle$$
- Mutasi σ dan x diperoleh dengan menggunakan rumus sebagai berikut:

$$\sigma' = \sigma \cdot \exp(\tau \cdot N(0,1))$$

$$x'_i = x_i + \sigma' \cdot N(0,1)$$

- dimana τ berfungsi seperti laju belajar (*learning rate*) dan biasanya τ diset mendekati $\frac{1}{\sqrt{n}}$.
- Tentu saja kita bisa menggunakan suatu aturan untuk membatasi nilai σ' pada suatu nilai tertentu, misalnya ε_0 . Jadi, jika $\sigma' < \varepsilon_0$ maka $\sigma' = \varepsilon_0$.

Local
maximum



Mutasi tanpa Korelasi menggunakan σ sebanyak n

- Suatu kromosom direpresentasikan sebagai

$$\langle x_1, \dots, x_n, \sigma_1, \dots, \sigma_n \rangle$$

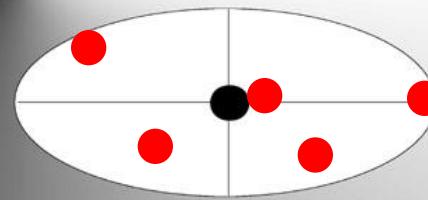
- Mutasi σ dan x diperoleh dengan menggunakan rumus

$$\sigma'_i = \sigma_i \cdot \exp(\eta \cdot N(0,1) + \tau \cdot N_i(0,1))$$

$$x'_i = x_i + \sigma'_i \cdot N_i(0,1)$$

- η adalah *learning rate* untuk semua gen (Biasanya diset $\frac{1}{\sqrt{2n}}$).
- τ adalah *learning rate* untuk setiap posisi gen ($\frac{1}{(2n)^{1/4}}$).

Local
maximum



Mutasi dengan Korelasi

- Metode ini selain menggunakan nilai σ sebanyak n yang masing-masing secara berurutan digunakan untuk memutasi gen-gen yang ada di dalam kromosom, juga menggunakan sudut-sudut rotasi α sebanyak k .
- Dimana $k = n(n-1)/2$ adalah jumlah kombinasi pasangan i dan j dari n yang ada.
- Oleh karena itu, suatu kromosom direpresentasikan sebagai

$$\langle x_1, \dots, x_n, \sigma_1, \dots, \sigma_n, \alpha_1, \dots, \alpha_k \rangle$$

Mutasi dengan Korelasi

Covariance matrix C didefinisikan sebagai

$$c_{ii} = \sigma_i^2$$

$c_{ij} = 0$ jika i dan j tidak berkorelasi

$c_{ji} = \frac{1}{2}(\sigma_i^2 - \sigma_j^2) \tan(2\alpha_{ij})$ jika i dan j berkorelasi

Mutasi dengan Korelasi

- Mutasi σ , α , dan x diperoleh dengan menggunakan rumus sebagai berikut:

$$\sigma'_i = \sigma_i \cdot \exp(\eta \cdot N(0,1) + \tau \cdot N_i(0,1))$$

$$\alpha'_j = \alpha_j + \beta \cdot N(0,1)$$

$$x' = x + N(0, C')$$

- x adalah vektor variabel objektif $\langle x_1, \dots, x_n \rangle$
- Sedangkan C' adalah matriks kovarian yang dihitung setelah mutasi sudut-sudut rotasi α .
- β adalah perubahan sudut yang biasanya berkisar 5° .
- Jika $|\alpha'_j| > \pi$, maka $\alpha'_j = \alpha'_j - 2\pi \text{sign}(\alpha'_j)$.

Mutasi dengan Korelasi

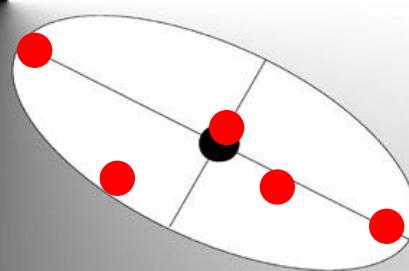
- η adalah *learning rate* untuk semua gen. Biasanya diset mendekati

$$\frac{1}{\sqrt{2n}}$$

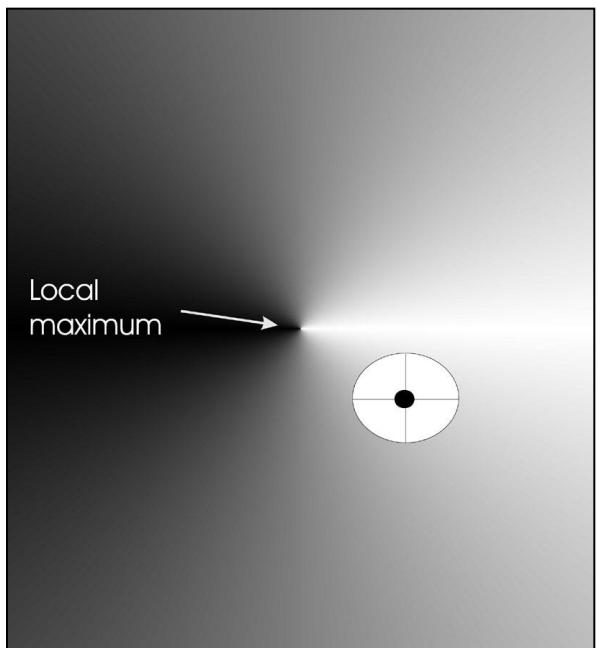
- τ adalah *learning rate* untuk setiap posisi gen. Biasanya diset mendekati

$$\frac{1}{(2n)^{1/4}}$$

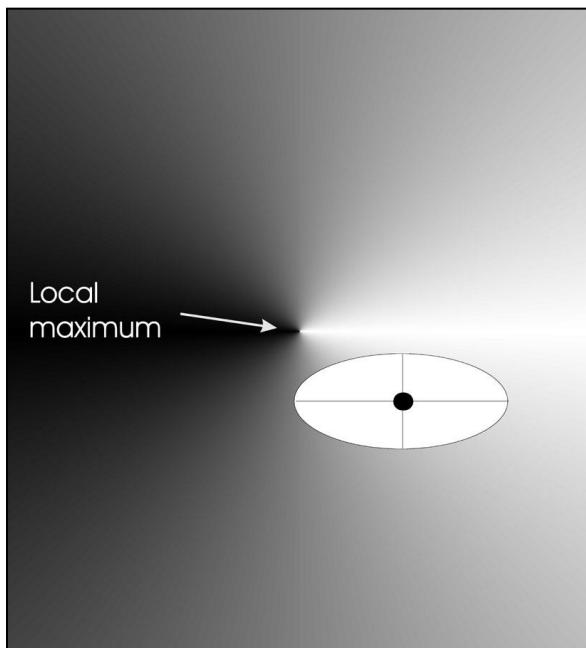
Local
maximum



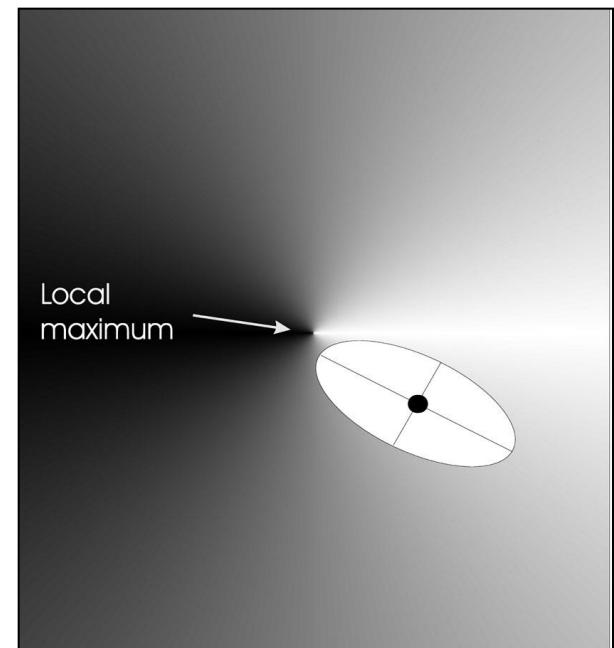
Mutasi tanpa Korelasi
menggunakan satu σ



Mutasi tanpa Korelasi
menggunakan $n \sigma$



Mutasi dengan
Korelasi



$f(x_1, x_2)$



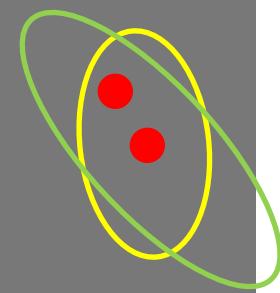
Mutasi dengan Korelasi

Populasi = 1



Global maximum

Generasi 1



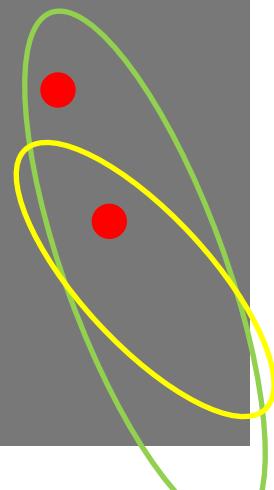
Mutasi dengan Korelasi

Populasi = 1



Global maximum

Generasi 2



Mutasi dengan Korelasi

Populasi = 1

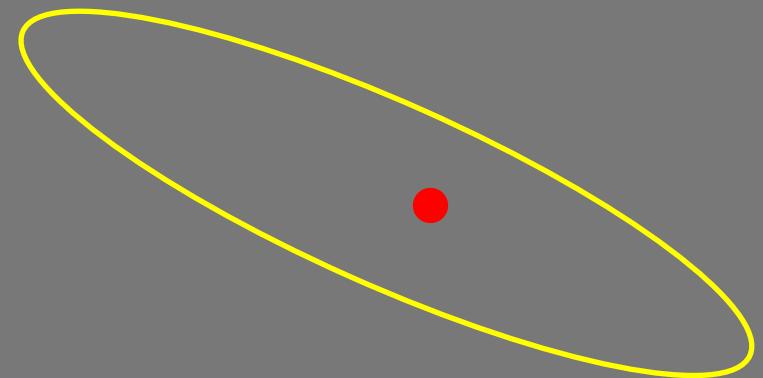
Global maximum

Generasi 3

Mutasi dengan Korelasi

Populasi = 1

Global maximum



Generasi 10

Mutasi dengan Korelasi

Populasi = 1



Generasi 50

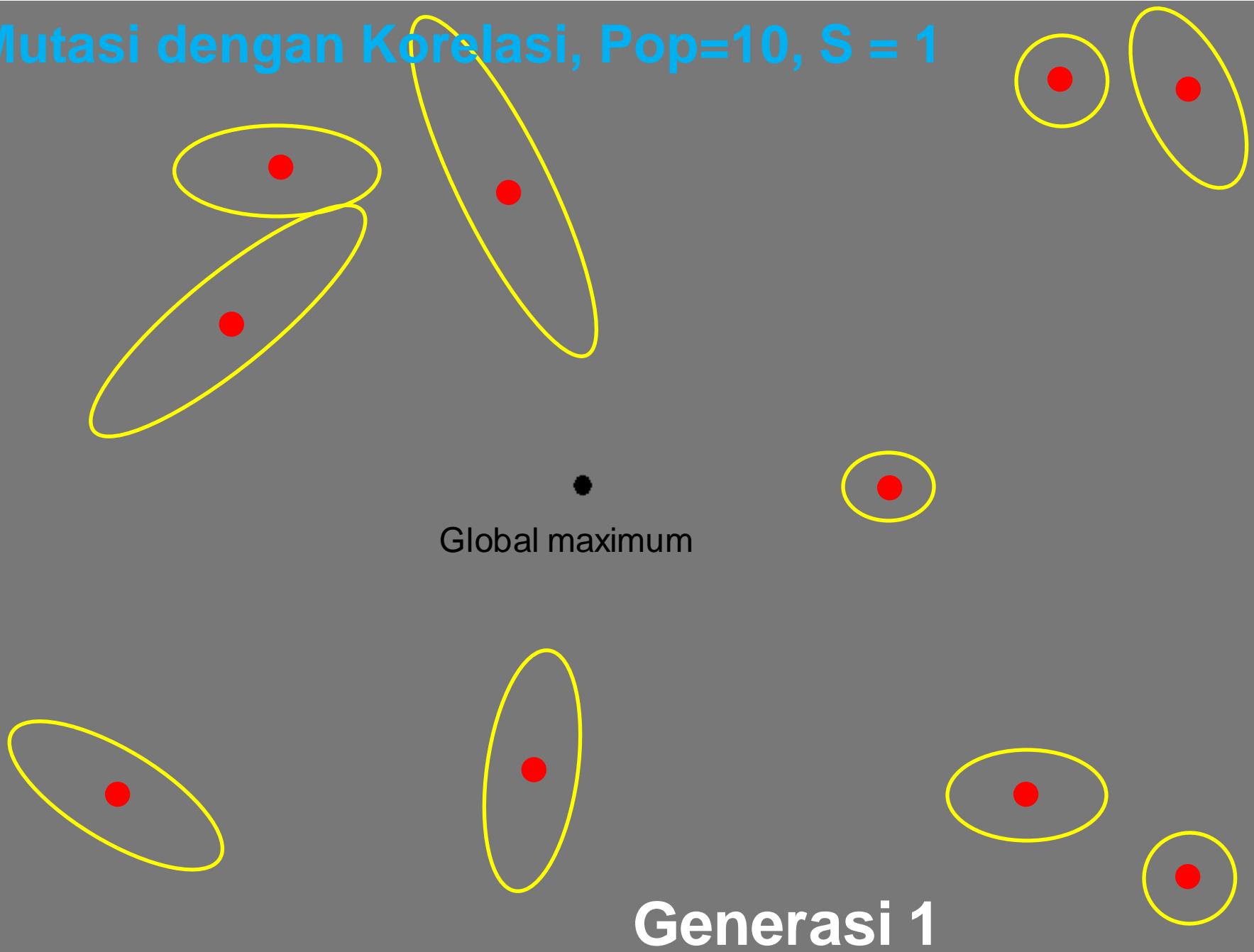
Mutasi dengan Korelasi

Populasi = 1

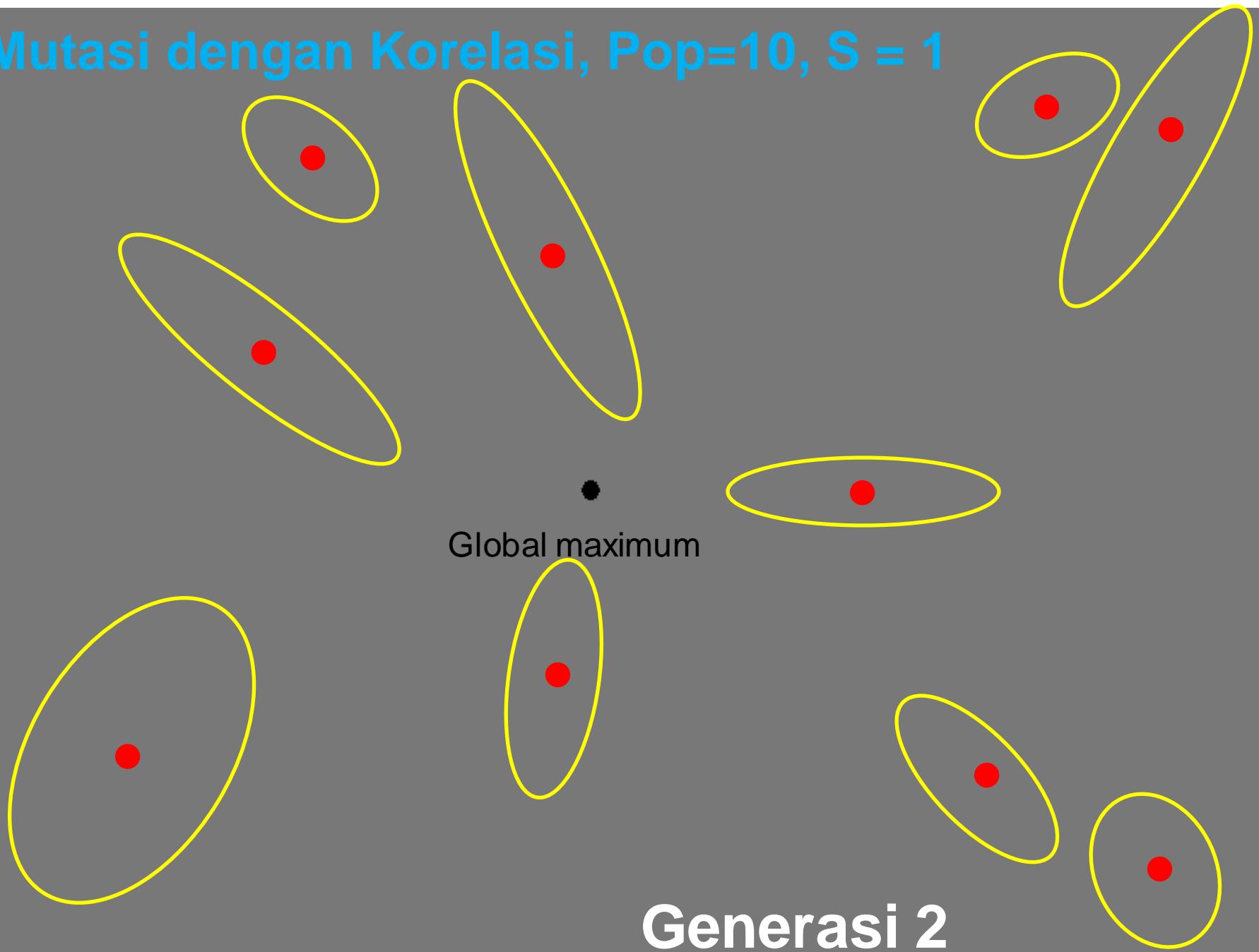


Generasi 100

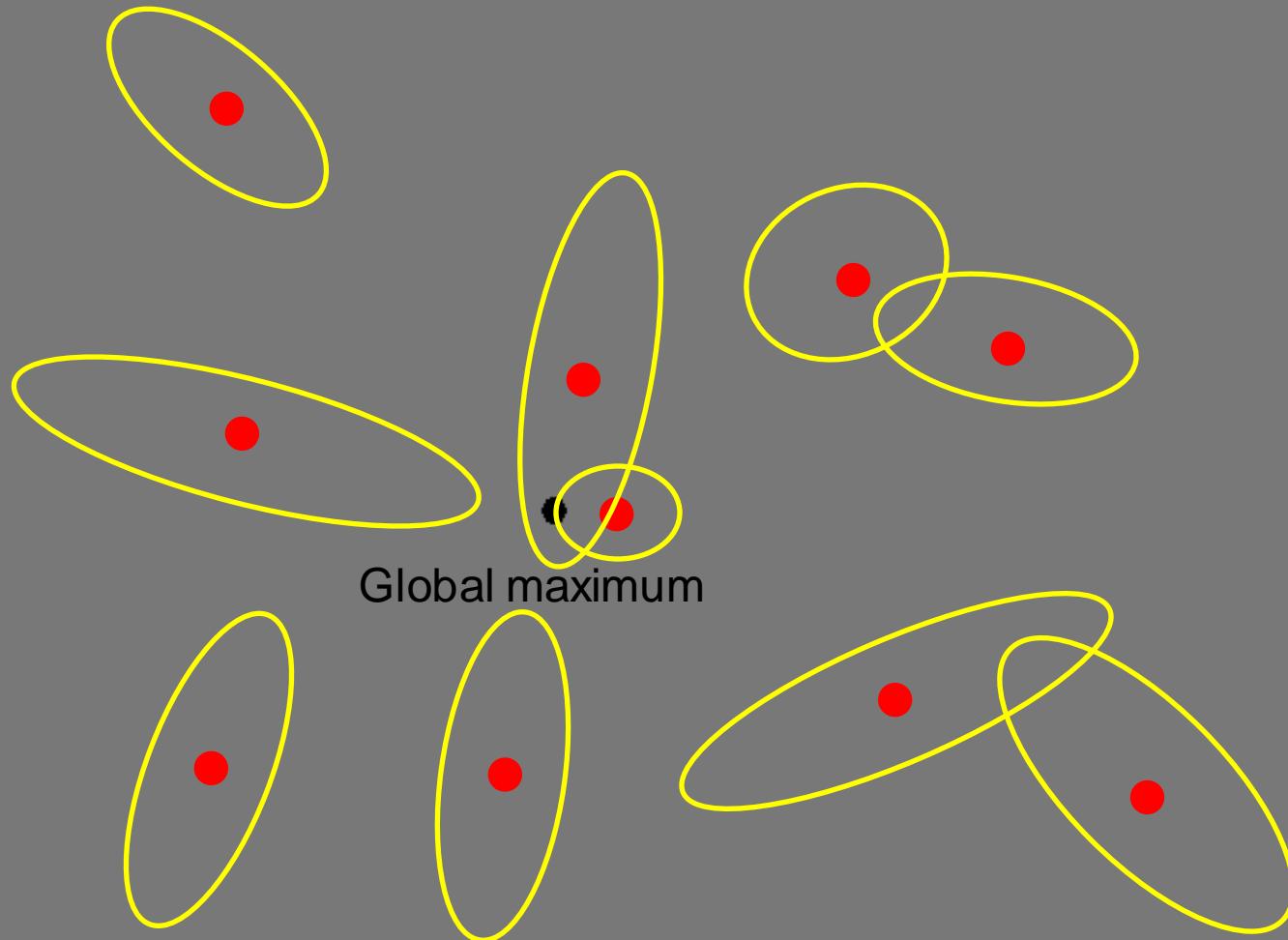
Mutasi dengan Korelasi, Pop=10, S = 1



Mutasi dengan Korelasi, Pop=10, S = 1

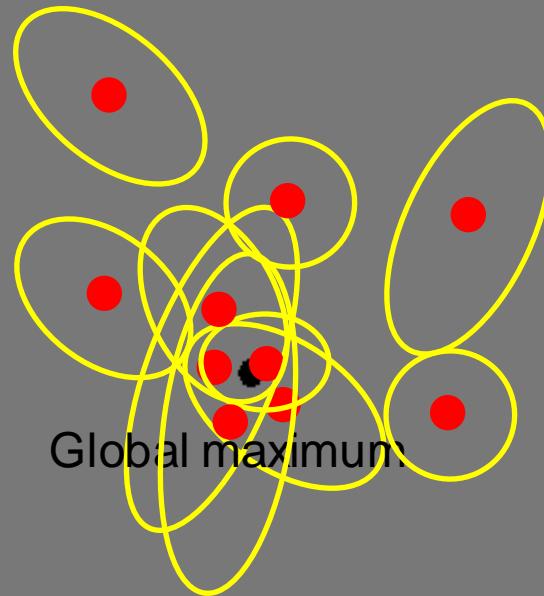


Mutasi dengan Korelasi, Pop=10, S = 1



Generasi 10

Mutasi dengan Korelasi, Pop=10, S = 1



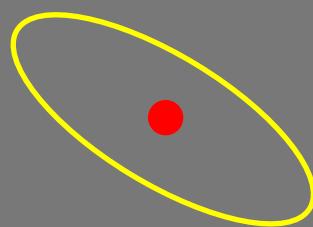
Generasi 20

Mutasi dengan Korelasi, Pop=10, S = 1

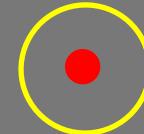


Generasi 20

Mutasi dengan Korelasi, Pop=2, S = 7

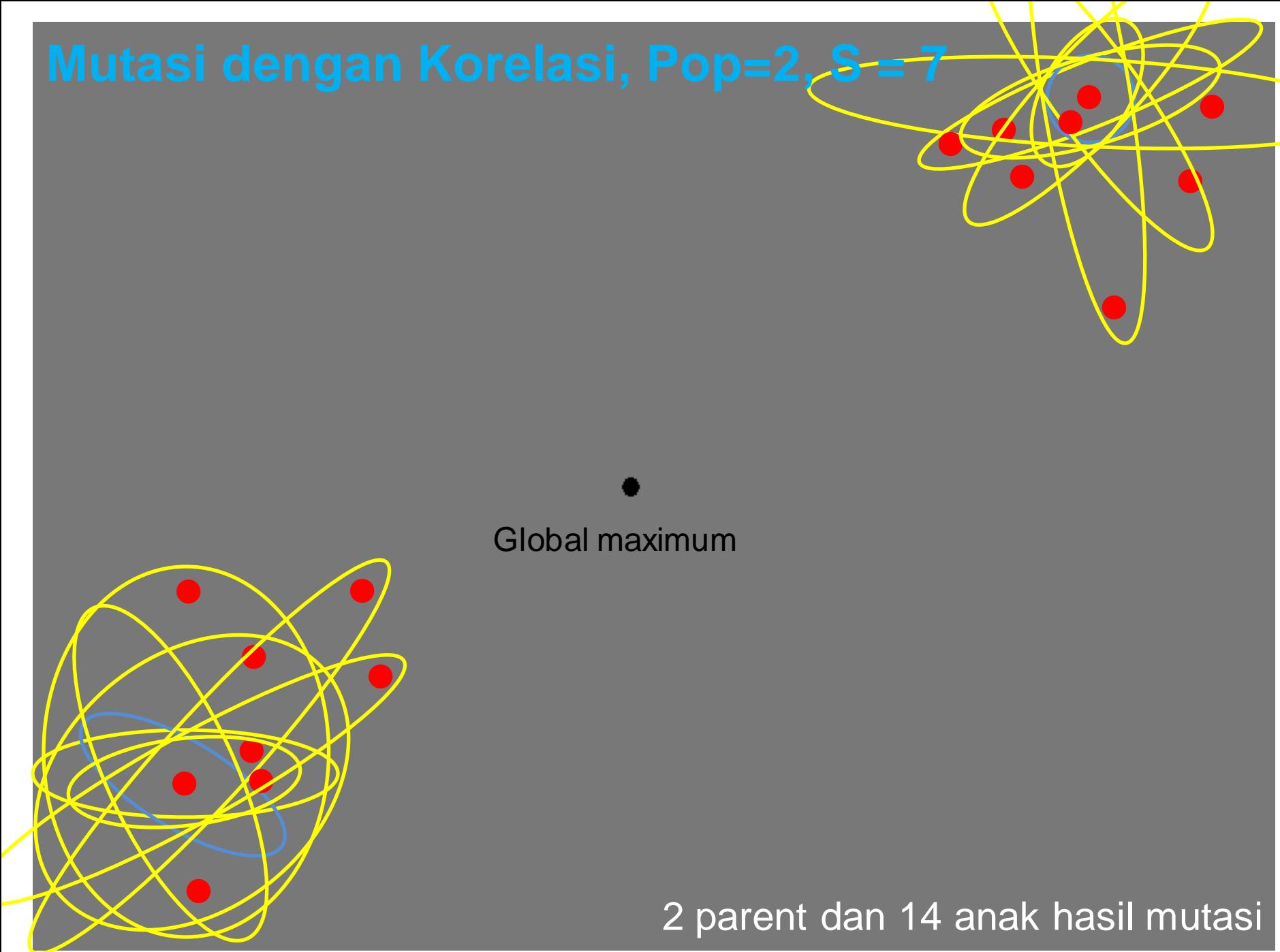


Global maximum

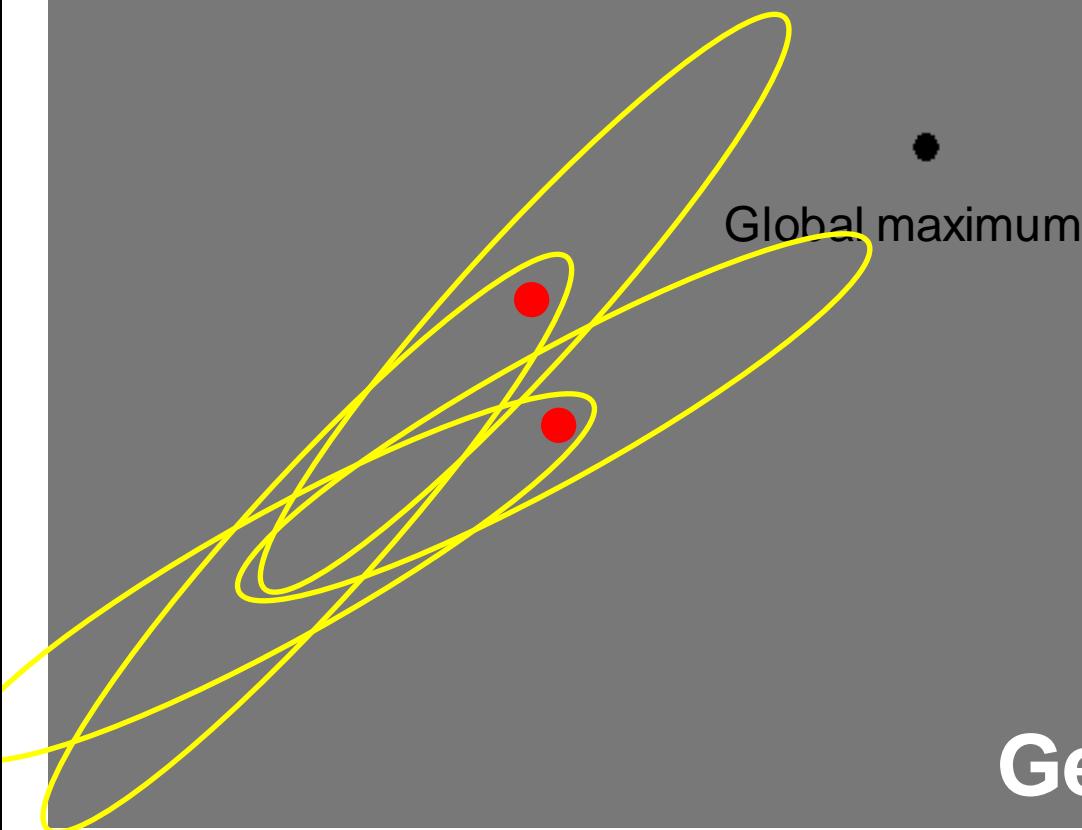


Generasi 1

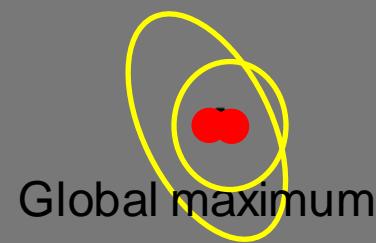
Mutasi dengan Korelasi, Pop=2, S = 7



Mutasi dengan Korelasi, Pop=2, S = 7



Mutasi dengan Korelasi, Pop=2, S = 7



Generasi 10

Rekombinasi

	Dua orangtua tetap	Dua orangtua yang berubah-ubah untuk setiap gen ke- <i>i</i>
$z_i = (x_i + y_i) / 2$	<u>intermediary lokal</u>	intermediary global
$z_i = x_i$ atau y_i yang dipilih secara acak	discrete lokal	discrete global

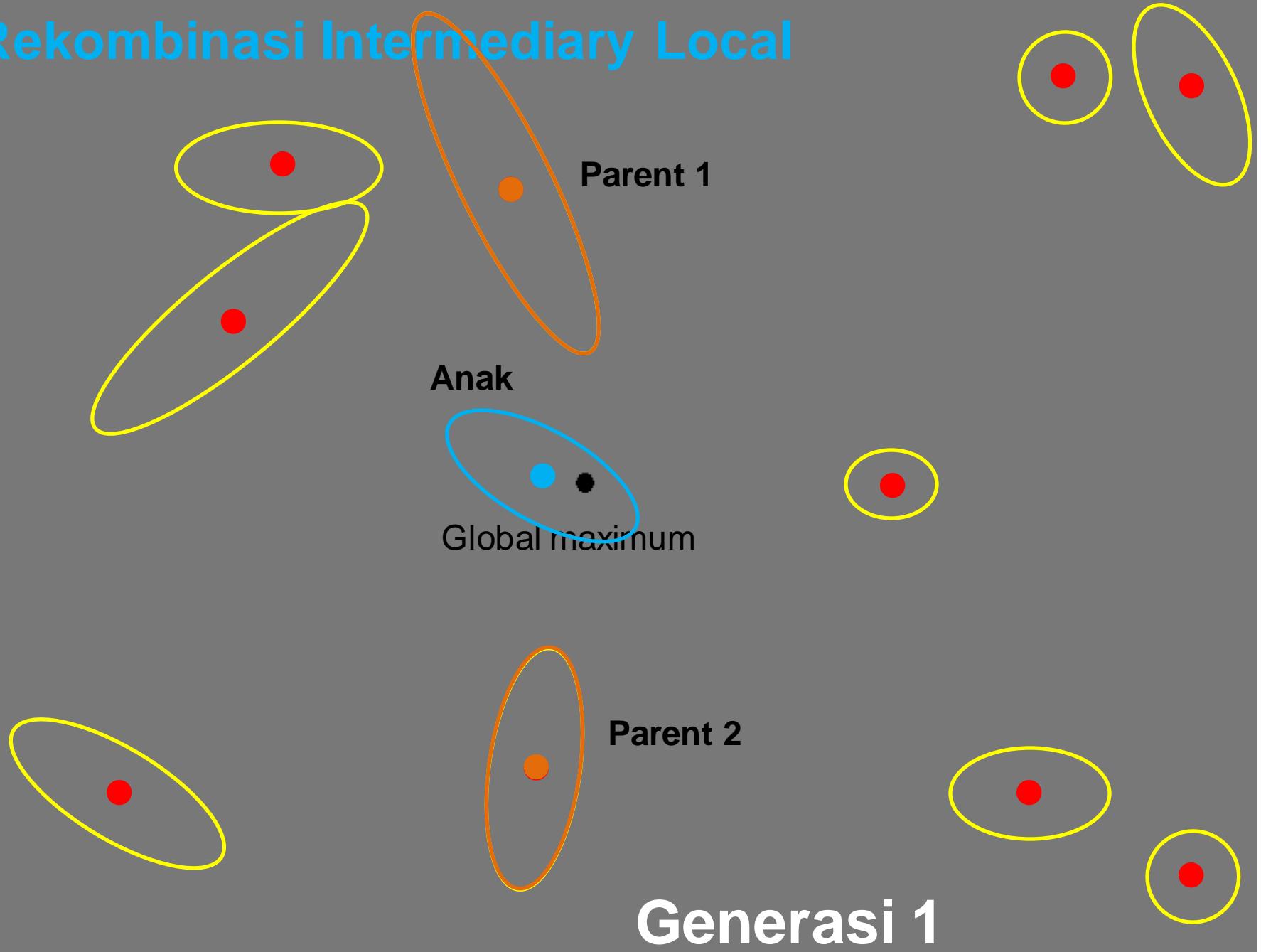
Intermediary: z antara x dan y

discrete: z dipilih acak dari x atau y

Lokal: gen didapat dari ortu yg tetap

Global: gen didapat dari ortu yg berbeda2

Rekombinasi Intermediary Local



Seleksi *Survivor*

- Misalkan jumlah kromosom dalam populasi adalah μ .
- Proses rekombinasi dan mutasi menghasilkan sejumlah kromosom anak, misalnya λ .
- Pada ES, seleksi *survivor* dilakukan secara deterministik dengan cara memilih sejumlah μ kromosom yang memiliki *fitness* paling tinggi.
- Proses pemilihan tersebut bisa dilakukan pada kromosom **anak saja** yang sering disebut sebagai **(μ,λ) -selection**.
- Atau dilakukan pada gabungan kromosom **orangtua** dan kromosom **anak** yang sering disebut sebagai **$(\mu+\lambda)$ -selection**.

Seleksi *Survivor*

- Metode $(\mu+\lambda)$ -*selection* bisa mempertahankan kromosom terbaik, sedangkan metode (μ,λ) -*selection* tidak bisa.
- Tetapi, metode (μ,λ) -*selection* lebih sering digunakan karena tiga alasan:
 - lebih baik dalam menghindari optimum lokal;
 - lebih baik untuk permasalahan dengan nilai optimum yang berubah-ubah;
 - bisa menghindari adanya nilai σ yang buruk tetapi bertahan hidup terlalu lama di dalam kromosom ketika x sangat dekat dengan solusi (*fit*).

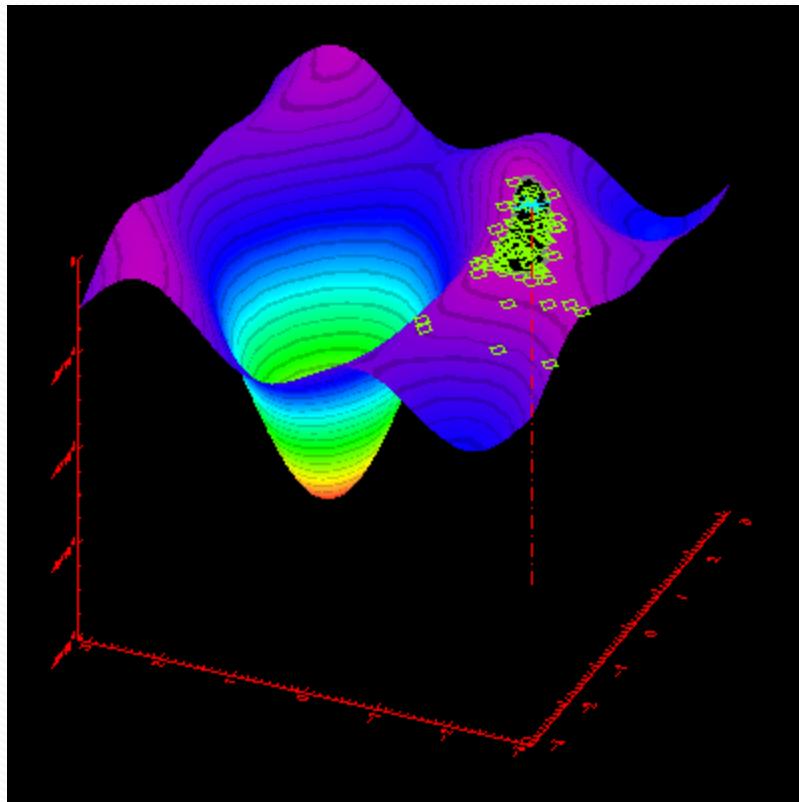
Seleksi Survivor

- Berbeda dengan GA, pada ES *selective pressure* biasanya dibuat sangat tinggi, misalnya $\lambda = 7\mu$.
- Jika kita menggunakan metode (μ,λ) -*selection*, maka untuk $\lambda = 7\mu$ berarti kita harus memilih 1 dari 7 kromosom sebagai individu terbaik.
- Hal ini berarti terjadi kompetisi yang sangat ketat (*selective*).

Self-Adaptation

- Kemampuan untuk mengikuti nilai optimum yang berubah-ubah dan kemampuan menentukan *mutation step size* σ secara adaptif.
- Misalkan terdapat suatu masalah optimasi fungsi yang nilai-nilai optimumnya sengaja dibuat berubah-ubah (*moving optimum*) pada periode sejumlah generasi tertentu, misalnya setiap 500 generasi.

Self-Adaptation



Self-Adaptation

- Pada gambar di atas, sekumpulan kotak kecil menyatakan individu/kromosom dalam populasi ES.
- Pada berbagai kasus, ES dapat segera menemukan nilai optimum.
- Sifat inilah yang disebut sebagai *self-adaptation*.

Self-Adaptation

Untuk bisa memiliki sifat *self-adaptation*, ES harus memenuhi persyaratan berikut:

- $\mu > 1$ untuk mendapatkan strategi-strategi yang berbeda,
- $\lambda > \mu$ untuk membangkitkan banyak anak (biasanya $\lambda \approx 7\mu$),
- (μ,λ) -selection untuk menghindari σ yang salah dalam beradaptasi,
- Saling menukarkan parameter-parameter strategi yang terdapat pada kromosom-kromosom menggunakan proses rekombinasi *intermediary* lokal atau global.

Proses Evolusi

- μ, λ

Sejumlah μ orangtua menghasilkan sejumlah λ anak hanya menggunakan mutasi (tanpa rekombinasi). Pada proses evolusi jenis ini, seleksi survivor dilakukan hanya terhadap sejumlah **λ anak** (μ orangtua tidak diperhatikan). Seleksi survivor menghasilkan sejumlah μ individu terbaik yang akan hidup pada generasi berikutnya. Dengan demikian, jumlah individu dalam populasi selalu tetap, yaitu sejumlah μ .

- $\mu/r, \lambda$

sama dengan μ, λ tetapi menggunakan rekombinasi (r).

Proses Evolusi

- $\mu + \lambda$

Sejumlah μ orangtua menghasilkan sejumlah λ anak hanya menggunakan mutasi (tanpa rekombinasi). Seleksi survivor dilakukan terhadap gabungan **anak dan orangtua**: $\lambda + \mu$. Seleksi survivor menghasilkan sejumlah μ individu terbaik yang akan hidup pada generasi berikutnya. Dengan demikian, jumlah individu dalam populasi selalu tetap, yaitu sejumlah μ .

- $\mu/r + \lambda$

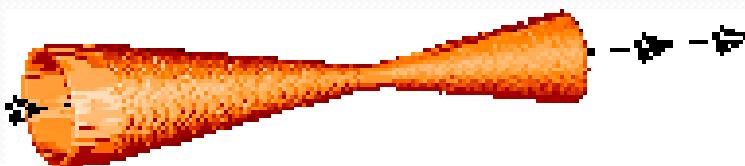
sama dengan $\mu + \lambda$ tetapi menggunakan rekombinasi (r).

Contoh Aplikasi ES

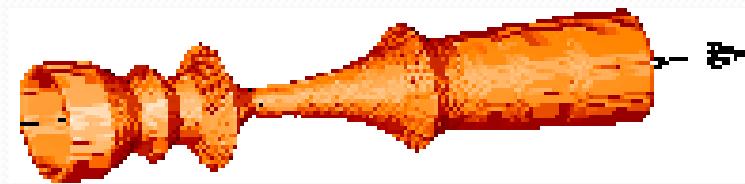
- Desain Jet Nozzle
- Desain Lensa
- Desain Jembatan
- Kotak Ajaib
- Optimasi swarming
- Optimasi Sistem Pipa
- Travelling Salesman Problem

Desain Jet Nozzle

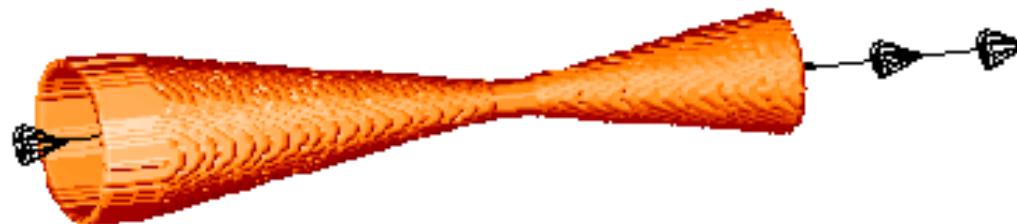
[evonet.iri.fr/CIRCUS2/node.php?node=72]



Starting



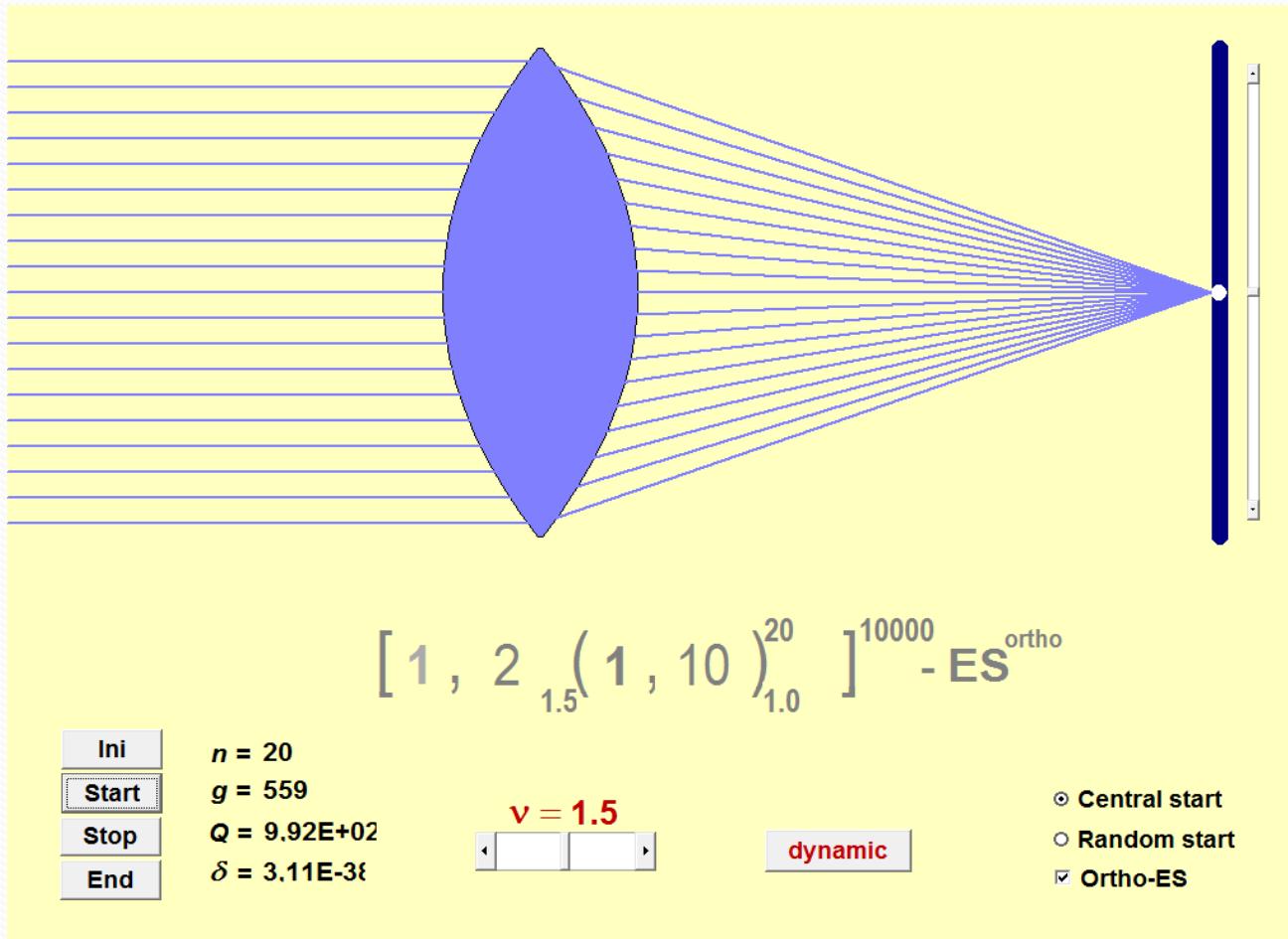
Resulting



Designing by ES

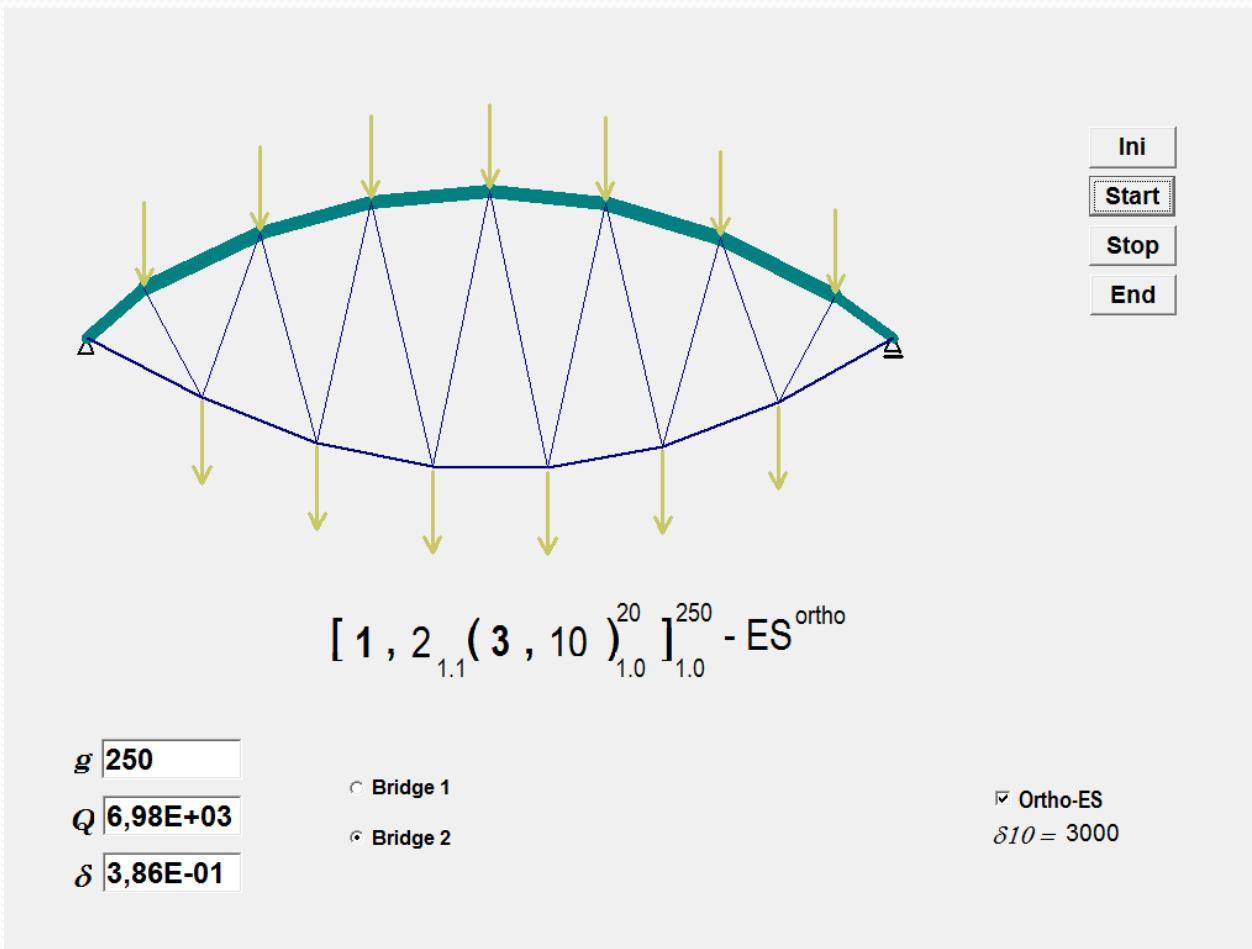
Desain Lensa Optik

[www.bionik.tu-berlin.de/institut/xs2anima]



Desain Jembatan

[www.bionik.tu-berlin.de/institut/xs2anima]



Kotak Ajaib

[www.bionik.tu-berlin.de/institut/xs2anima]

g

Q

δ

!

The Magic
21x21-Square with the
Magic Sum of 2006

72	99	116	93	95	59	103	97	76	128	94	122	91	98	123	104	94	96	76	89	81
65	104	96	92	93	104	106	93	104	138	48	109	88	91	128	48	107	81	123	102	86
85	86	109	82	85	84	148	116	69	155	91	83	68	95	117	63	127	105	119	84	35
73	118	86	111	102	64	106	91	96	116	75	122	68	84	108	50	146	120	93	82	101
72	114	77	101	139	59	118	86	132	99	54	106	64	109	121	70	118	107	84	114	62
58	88	100	122	108	123	71	70	81	88	68	117	79	102	132	101	121	97	117	71	92
71	101	81	121	92	59	142	62	78	106	83	110	86	97	110	83	89	136	125	96	78
174	174	2	2	133	154	196	0	0	175	113	216	0	0	216	151	155	6	6	6	127
148	2	98	159	2	165	0	171	127	0	127	0	173	150	0	99	6	144	157	166	112
127	147	117	138	2	151	0	156	120	0	141	0	155	138	0	110	6	146	116	138	98
97	178	151	2	176	131	0	152	156	0	162	0	185	197	0	140	6	6	6	175	86
122	134	2	122	165	186	0	122	146	0	139	0	132	160	0	149	6	127	149	6	139
122	2	172	151	155	130	0	127	147	0	111	0	175	141	0	107	6	179	165	6	110
153	2	2	2	2	130	198	0	0	228	171	207	0	0	222	154	190	6	6	177	156
82	82	162	95	104	94	130	98	88	135	65	102	68	65	93	68	85	66	96	124	104
113	89	95	88	107	85	101	75	105	115	58	152	93	93	119	80	126	76	56	92	88
42	126	106	74	82	51	133	111	102	87	63	112	140	71	115	99	136	67	139	86	64
98	53	114	120	117	19	117	75	113	96	84	98	100	95	106	116	92	113	102	97	81
32	105	141	106	109	20	124	111	87	126	83	141	104	75	93	67	131	104	94	90	63
93	108	51	117	54	80	102	90	88	128	63	114	53	103	83	103	142	108	113	98	115
107	94	128	108	84	58	117	103	91	86	113	95	84	42	120	44	117	116	64	107	128

Evolution of a Magic Square

3044

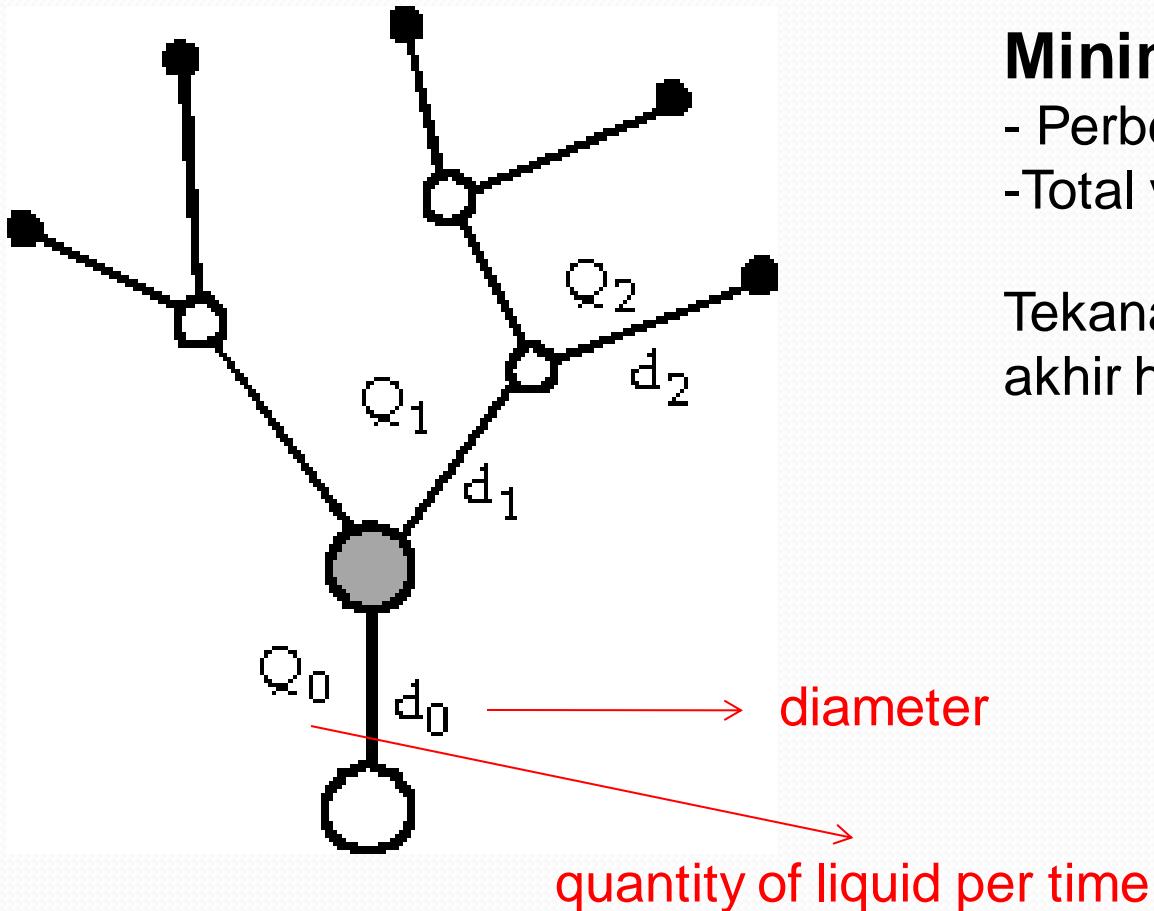
Optimasi Swarming

[www.bionik.tu-berlin.de/institut/xs2anima]



Optimasi Sistem Pipa

[Michael Herdy, Giannino Patone, Technical Report TR-94-05, 1994]



Minimasi

- Perbedaan tekanan
- Total volume pipa

Tekanan pada semua simpul akhir harus sama.

The quality function (Fitness)

$$Q = \sum_{\text{all final nodes}} (c_1 \cdot (p_i - \bar{p})^2 + c_2 \cdot p_i) + c_3 \cdot V_{\text{tot}}$$

constant

pressure at the i-th final node

average pressure at the final node

total volume of the system of pipes

Travelling Salesman Problem

[Michael Herdy, Giannino Patone, Technical Report TR-94-05, 1994]

- Representasi Kromosom?
- Permutasi → keluar dari pakem ES (umumnya real)
- Apa bedanya dengan GA?
- ES menggunakan **mutation step size**

Empat Operator Mutasi

1 2 3 4 5 6 7 8 9 10 11 12

Parents-Tour

1 5 4 3 2 6 7 8 9 10 11 12

Inversion of the tour segment 2-3-4-5

1 3 4 5 2 6 7 8 9 10 11 12

Insertion of town no.2 between no.5 and no.6

1 6 7 2 3 4 5 8 9 10 11 12

Displacement of the tour segment 2-3-4-5

1 5 3 4 2 6 7 8 9 10 11 12

Reciprocal Exchange of towns no.2 and no. 5

Kesimpulan

- ES sangat powerful untuk permasalahan dengan bilangan real
- ES cenderung lebih cepat dibandingkan GA
- ES bisa memiliki sifat *SelfAdaptation*

Daftar Pustaka

- [SUYo8] Suyanto, 2008, Evolutionary Computation: Komputasi Berbasis “Evolusi” dan “Genetika”, penerbit Informatika Bandung.
- evonet.lri.fr/CIRCUS2/node.php?node=72
- www.bionik.tu-berlin.de/institut/xs2anima
- Michael Herdy, Giannino Patone, Technical Report TR-94-05, 1994

Daftar Pustaka

- [EIB03] Eiben, A.E. and Smith, J.E., 2003, "Introduction to Evolutionary Computing", Springer-Verlag Berlin Heidelberg.
- [ADN07] Adnan Oktar, 2007, "Mekanisme Khayalan Teori Evolusi", www.evolutiondeceit.com/indonesian/keruntuhan3.php
- [JUL07] Julie Leung, Keith Kern, Jeremy Dawson, 2007, "Genetic Algorithms and Evolution Strategies", presentation slides.
- [SUY08] Suyanto, 2008, Evolutionary Computation: Komputasi Berbasis "Evolusi" dan "Genetika", penerbit Informatika Bandung.
- [TOM07] Tomoharu Nakashima, 2007, "Evolving Soccer Teams for RoboCup Simulation", IEEE Congress on *Evolutionary Computation*, Singapore 25 – 28 September 2007.
- [RYA98a] Ryan Conor and O'Neill Michael, 1998, "Grammatical Evolution: A Steady State approach". In Proceedings of the Second International Workshop on Frontiers in Evolutionary Algorithms 1998, pages 419-423.

Evolutionary Programming (EP)

Dr. Suyanto, S.T., M.Sc.
HP/WA: 0812 845 12345

Intelligence Computing Multimedia (ICM)
Informatics faculty – Telkom University

Intro

- EP diperkenalkan pertama kali oleh D. Fogel pada era 1960-an di Amerika Serikat
- Pada saat itu EP ditujukan untuk menghasilkan suatu bentuk kecerdasan (*intelligence*), dimana *intelligence* dipandang sebagai suatu tingkah laku yang adaptif (*adaptive behaviour*)
- Salah satu syarat dari *adaptive behaviour* adalah kemampuan memprediksi lingkungan yang dinamis

Intro

- Dengan demikian, **kemampuan memprediksi** adalah kunci menuju *intelligence*
- Bagaimana membangun kemampuan untuk memprediksi?
- D. Fogel mengusulkan EP yang menggunakan *Finite State Machines (FSM)* untuk menghasilkan suatu sistem *machine learning* yang memiliki kemampuan memprediksi

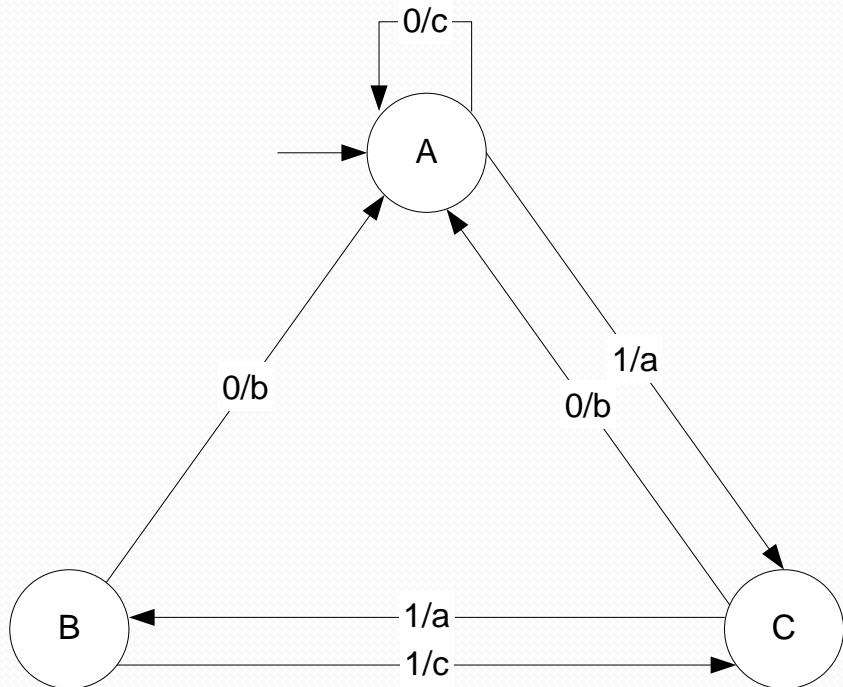
Intro

- FSM adalah mesin dengan sejumlah terbatas keadaan yang terdiri dari:
 - States (keadaan) S
 - Inputs I
 - Outputs O
 - Fungsi transisi $\delta : S \times I \rightarrow S \times O$

Intro

- FSM berfungsi mentransformasikan deretan input ke dalam deretan output.
- Dengan demikian, FSM dapat digunakan untuk memprediksi.
- Misal: “Jika inputnya x , maka outputnya y ”.

FSM



Contoh FSM

- Tiga state $S = \{A, B, C\}$
- Dua nilai input $I = \{0, 1\}$
- Tiga output $O = \{a, b, c\}$
- A adalah *initial state*
- Fungsi transisi digambarkan sebagai garis dengan tanda panah yang menunjukkan perpindahan dari suatu state ke state lainnya dengan menerima suatu input dan memberikan suatu output.

Kriteria FSM yang baik

1. Bisa memetakan sederetan input menjadi sederetan output dengan tingkat akurasi yang tinggi.
2. Memiliki jumlah state yang minimum.

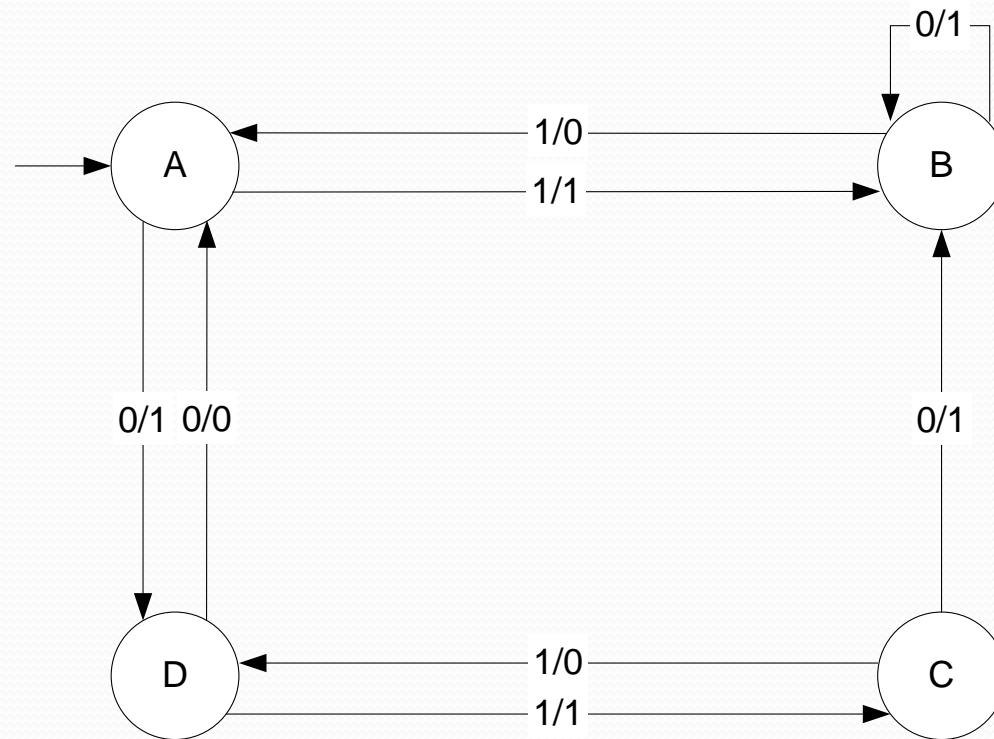
FSM

Input : 011010001

Output: 10011110

FSM yang bisa memetakan input-output tsb?

Aplikasi dunia nyata?
Prediksi data time series



FSM

- Pada FSM tersebut, A adalah *initial state*.
- FSM tersebut bisa memetakan setiap I_{i+1} menjadi O_i dengan akurasi 100%.
- Tetapi, mungkin terdapat FSM lain dengan jumlah *state* lebih sedikit dan memiliki akurasi 100% juga.

FSM

- Mengapa memilih FSM dengan minimum state?
- Jumlah state yang minimum tentu saja lebih efisien
- Jumlah state terlalu banyak → mungkin *overfit*

EP

- Pada perkembangannya, EP justru mengarah pada ES
- Kromosom Real
- Sedikit Berbeda dengan ES

Spesifikasi teknis EP

Representasi	Vektor bernilai real
Seleksi orangtua	Deterministik
Rekombinasi	Tidak ada
Mutasi	<i>Gaussian perturbation</i>
Seleksi survivor	Probabilistik ($\mu+\mu$)
Ciri khusus	<i>Self-adaptation</i> pada <i>mutation step sizes</i> (pada Meta-EP)

Representasi Individu

Kromosom yang lengkap pada EP hanya terdiri dari dua bagian, yaitu:

- Variabel objek: x_1, \dots, x_n
- Mutation step sizes: $\sigma_1, \dots, \sigma_n$

Kromosom

$$\langle x_1, \dots, x_n, \sigma_1, \dots \sigma_n \rangle$$

- n adalah jumlah variabel pada fungsi yang dioptimasi.
- Variabel objek mengkodekan nilai-nilai real secara langsung tanpa konversi.

Seleksi Orangtua

- Pada EP, setiap orangtua menghasilkan satu anak melalui proses mutasi (tanpa rekombinasi).
- Jadi, sejumlah μ orangtua menghasilkan sejumlah μ anak.
- Dengan demikian semua individu dalam populasi pasti terpilih sebagai orangtua tanpa memperhatikan nilai *fitness*-nya.
- Dengan kata lain, seleksi orangtua pada EP bersifat deterministik atau pasti tanpa dipengaruhi *fitness*-nya.
- Hal ini berbeda dengan seleksi orangtua pada GA yang bersifat probabilistik menggunakan metode *roulette wheel*, *tournament*, atau lainnya.

Rekombinasi

- Pada EP tidak terdapat rekombinasi
- Mengapa?
- Karena satu titik pada ruang pencarian berlaku sebagai suatu spesies (bukan sebagai suatu individu seperti pada GA maupun ES), sedangkan rekombinasi tidak mungkin dilakukan antar spesies
- Banyak perdebatan mengenai mutasi dan rekombinasi
- Mana yang lebih penting?
- Apakah keduanya sama-sama penting?
- Bisakah hanya menggunakan salah satu saja?

Mutasi

- Pada EP, mutasi merupakan satu-satunya proses untuk menghasilkan kromosom baru
- Mutasi dilakukan dengan cara mengubah nilai gen dengan menambahkan bilangan random yang dibangkitkan berdasarkan distribusi normal
- Proses mutasinya hampir sama dengan mutasi yang digunakan pada ES

Mutasi

- Suatu kromosom $\langle x_1, \dots, x_n, \sigma_1, \dots, \sigma_n \rangle$ bisa menghasilkan kromosom baru dengan adanya mutasi terhadap σ dan x yang diperoleh menggunakan rumus

$$\sigma'_i = \sigma_i \cdot (1 + \alpha \cdot N(0,1))$$

$$x'_i = x_i + \sigma'_i \cdot N_i(0,1) \quad \sigma'_i = \sigma_i \cdot \exp(\eta \cdot N(0,1) + \tau \cdot N_i(0,1))$$
$$x'_i = x_i + \sigma'_i \cdot N_i(0,1)$$

dimana α adalah suatu konstanta yang biasanya diset sekitar **0,2**.

Mutasi

- Urutan mutasi merupakan hal yang sangat penting.
- Mutasi terhadap σ harus lebih dulu daripada mutasi terhadap x .
- Jika urutannya dibalik, EP tidak bisa bekerja dengan baik untuk menemukan solusi.

Seleksi *Survivor*

- Misalkan suatu populasi P pada generasi t memiliki μ individu
- Karena seleksi orangtua pada EP adalah deterministik, maka semua individu pasti terpilih menjadi orangtua
- Dengan demikian kejadian ini bisa dilambangkan sebagai $P(t) = \mu$ orangtua

Seleksi Survivor

- Selanjutnya, karena setiap orangtua menghasilkan hanya satu anak melalui satu proses mutasi (tanpa rekombinasi), maka dari sejumlah μ orangtua akan menghasilkan sejumlah μ anak
- Sehingga $P'(t)$: μ anak menyatakan suatu populasi bayangan yang berisi sejumlah μ anak
- Pertanyaannya, bagaimana memilih sejumlah μ individu dari gabungan $P(t)$ dan $P'(t)$ untuk hidup pada generasi berikutnya atau $t + 1$?

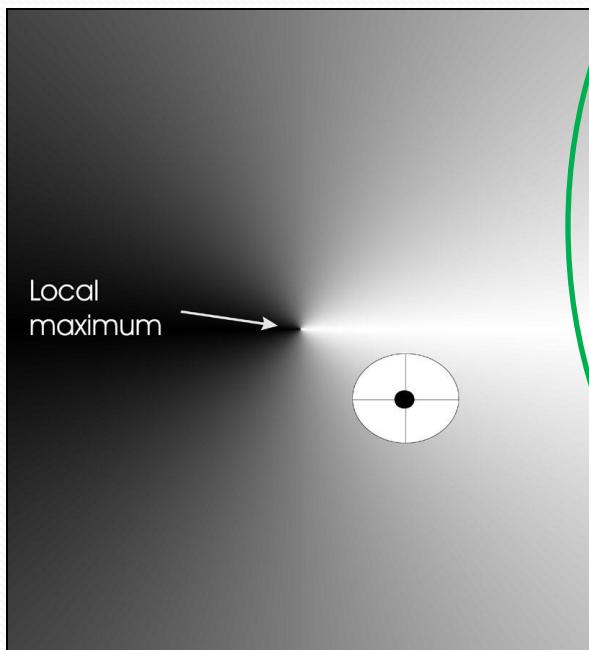
Seleksi Survivor

- EP menggunakan *Pairwise Competitions* dalam format *Round-Robin*, dengan cara:
 - Setiap solusi x dari gabungan $P(t) \cup P'(t)$ dievaluasi dengan cara membandingkan x terhadap sejumlah q solusi lain yang dipilih secara acak.
 - Untuk setiap perbandingan, suatu skor "menang" diberikan jika x lebih baik dibandingkan lawannya.
 - Sejumlah μ solusi dengan jumlah "menang" yang paling banyak akan bertahan hidup pada generasi berikutnya ($t + 1$).

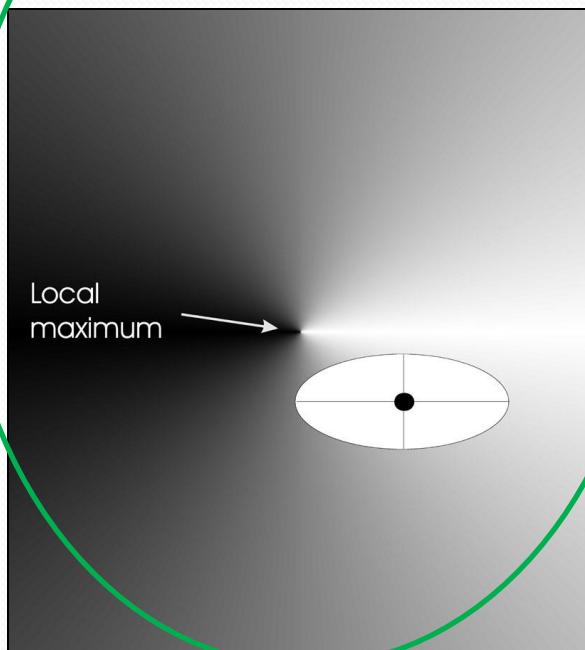
Pada cara di atas, parameter q digunakan untuk mengatur *selective pressure*. Biasanya q diset sekitar **10**.

Evolution Strategies

Mutasi tanpa Korelasi
menggunakan satu σ

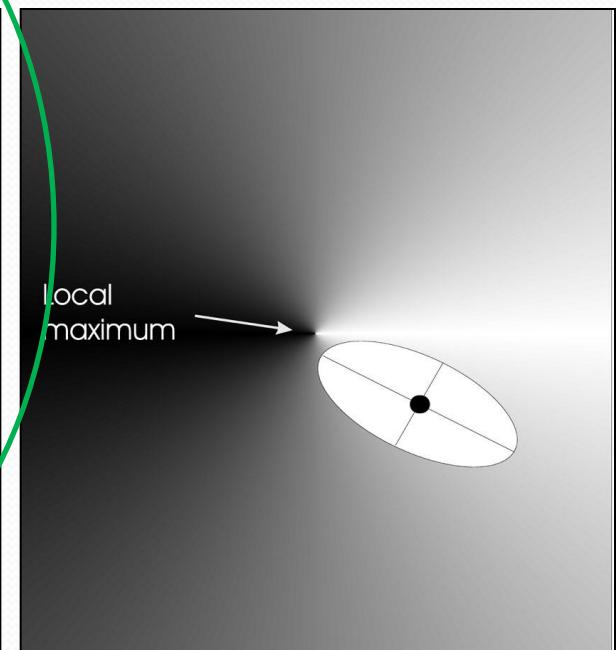


Mutasi tanpa Korelasi
menggunakan $n \sigma$



Evolution Programming
(tanpa Rekombinasi)

Mutasi dengan
Korelasi



$f(x_1, x_2)$



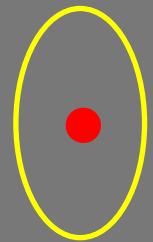
EP: Mutasi tanpa Korelasi dengan $n \sigma$

Populasi = 1



Global maximum

Generasi 1



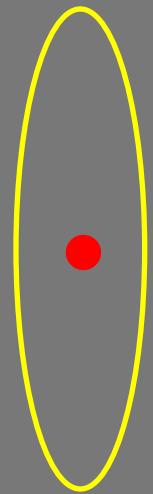
EP: Mutasi tanpa Korelasi dengan $n \sigma$

Populasi = 1



Global maximum

Generasi 2

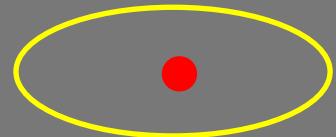


EP: Mutasi tanpa Korelasi dengan $n \sigma$

Populasi = 1



Global maximum

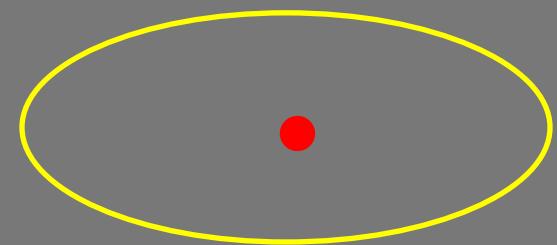


Generasi 3

EP: Mutasi tanpa Korelasi dengan $n \sigma$

Populasi = 1

Global maximum



Generasi 10

EP: Mutasi tanpa Korelasi dengan $n \sigma$

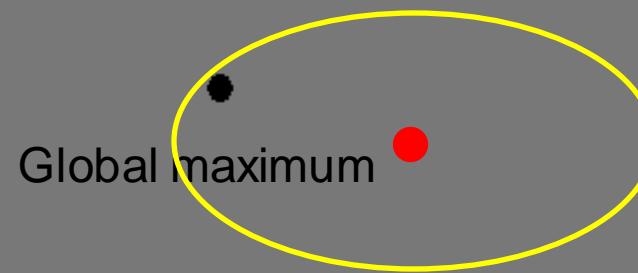
Populasi = 1



Generasi 50

EP: Mutasi tanpa Korelasi dengan $n \sigma$

Populasi = 1



Generasi 100

EP: Mutasi tanpa Korelasi dengan $n \sigma$

Populasi = 1



Global maximum

Generasi 200

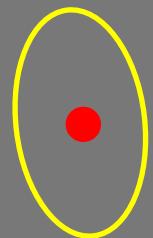
ES: Mutasi dengan Korelasi

Populasi = 1



Global maximum

Generasi 1

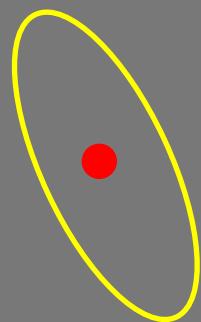


Mutasi dengan Korelasi

Populasi = 1



Global maximum



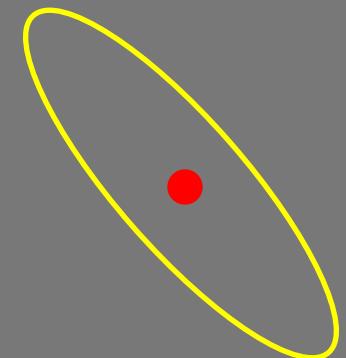
Generasi 2

Mutasi dengan Korelasi

Populasi = 1



Global maximum

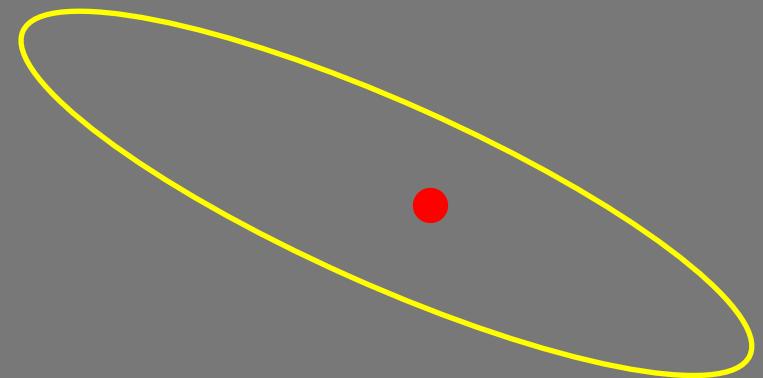


Generasi 3

Mutasi dengan Korelasi

Populasi = 1

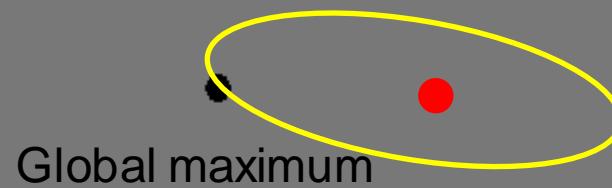
Global maximum



Generasi 10

ES: Mutasi dengan Korelasi

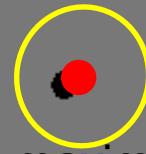
Populasi = 1



Generasi 20

ES: Mutasi dengan Korelasi

Populasi = 1

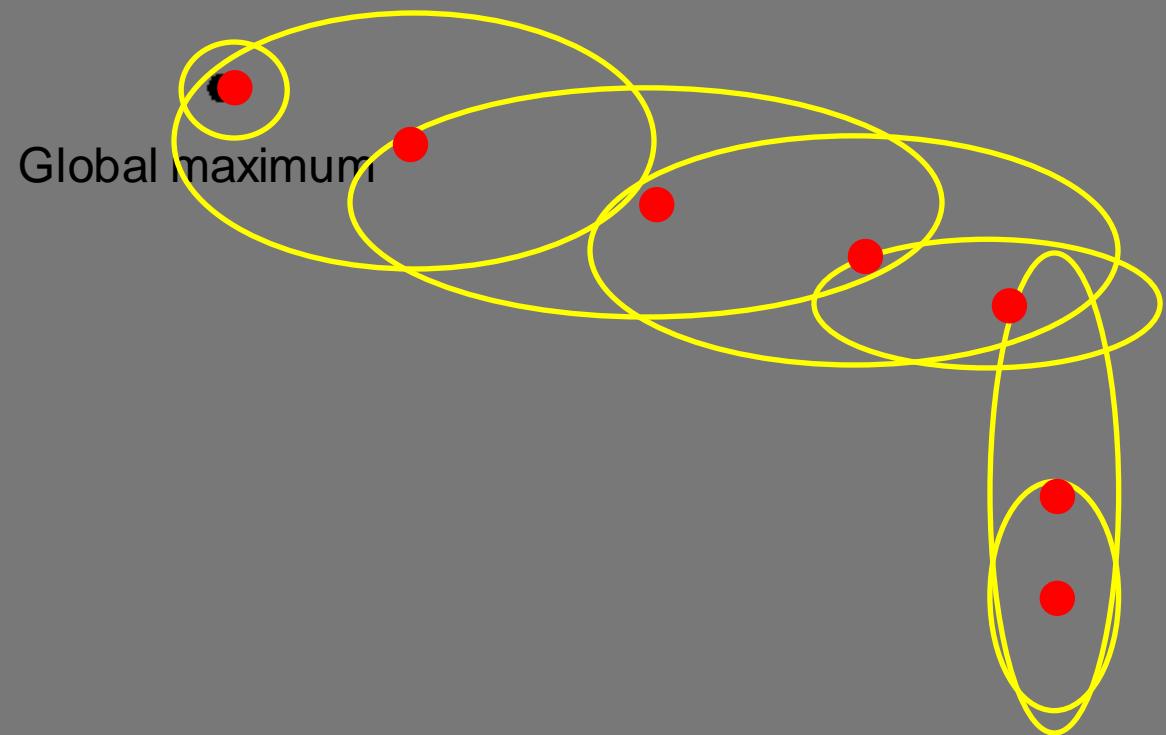


Global maximum

Generasi 50

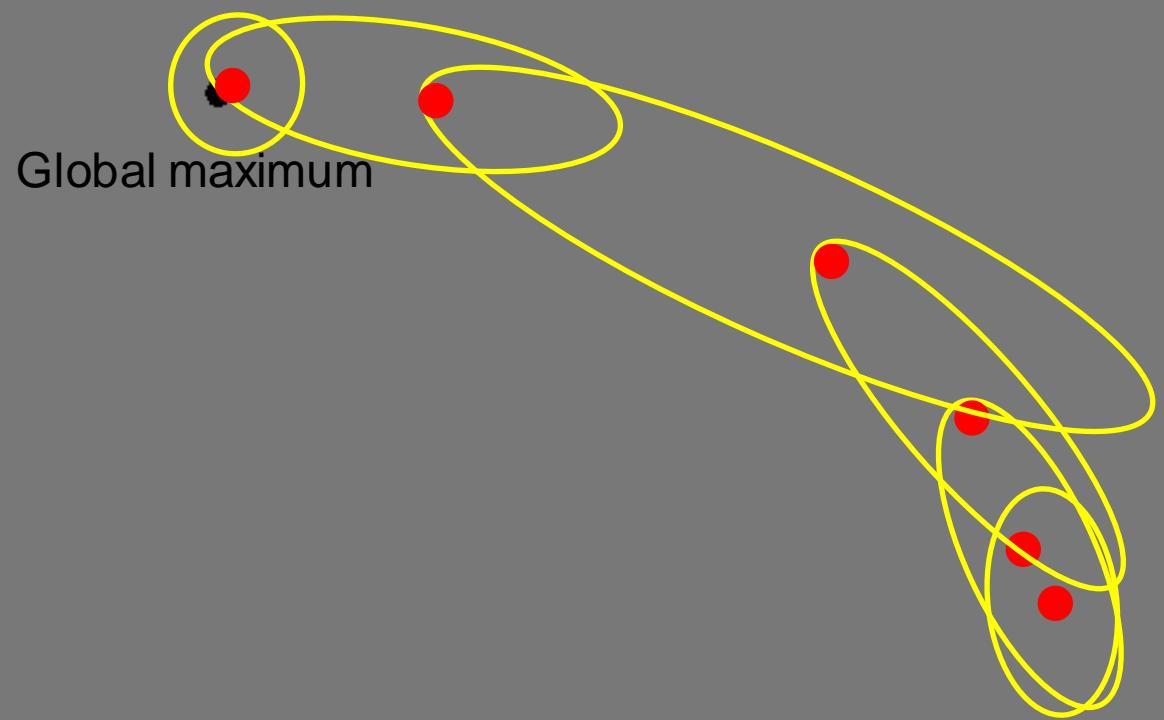
EP: Mutasi tanpa Korelasi dengan $n \sigma$

Populasi = 1



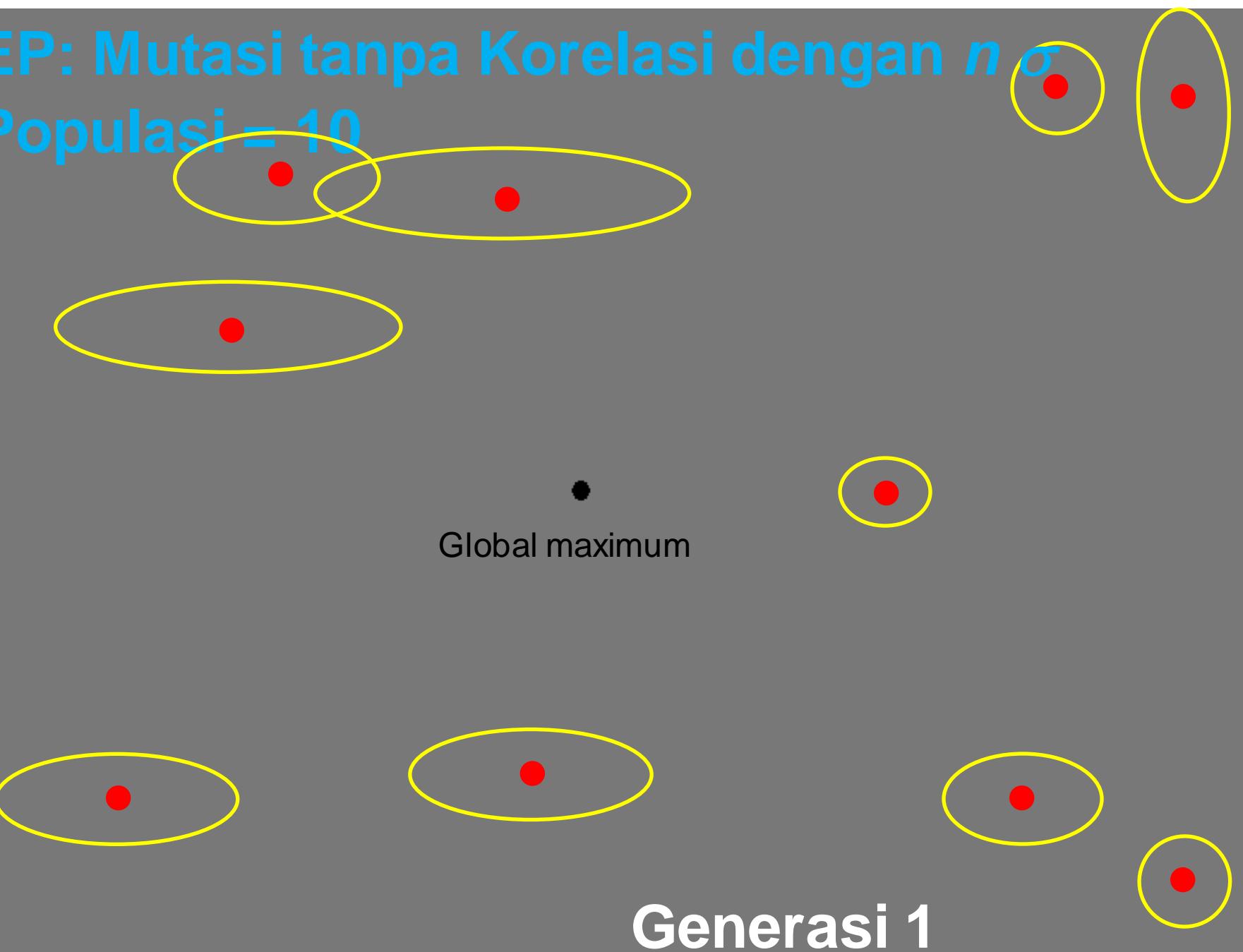
ES: Mutasi dengan Korelasi

Populasi = 1



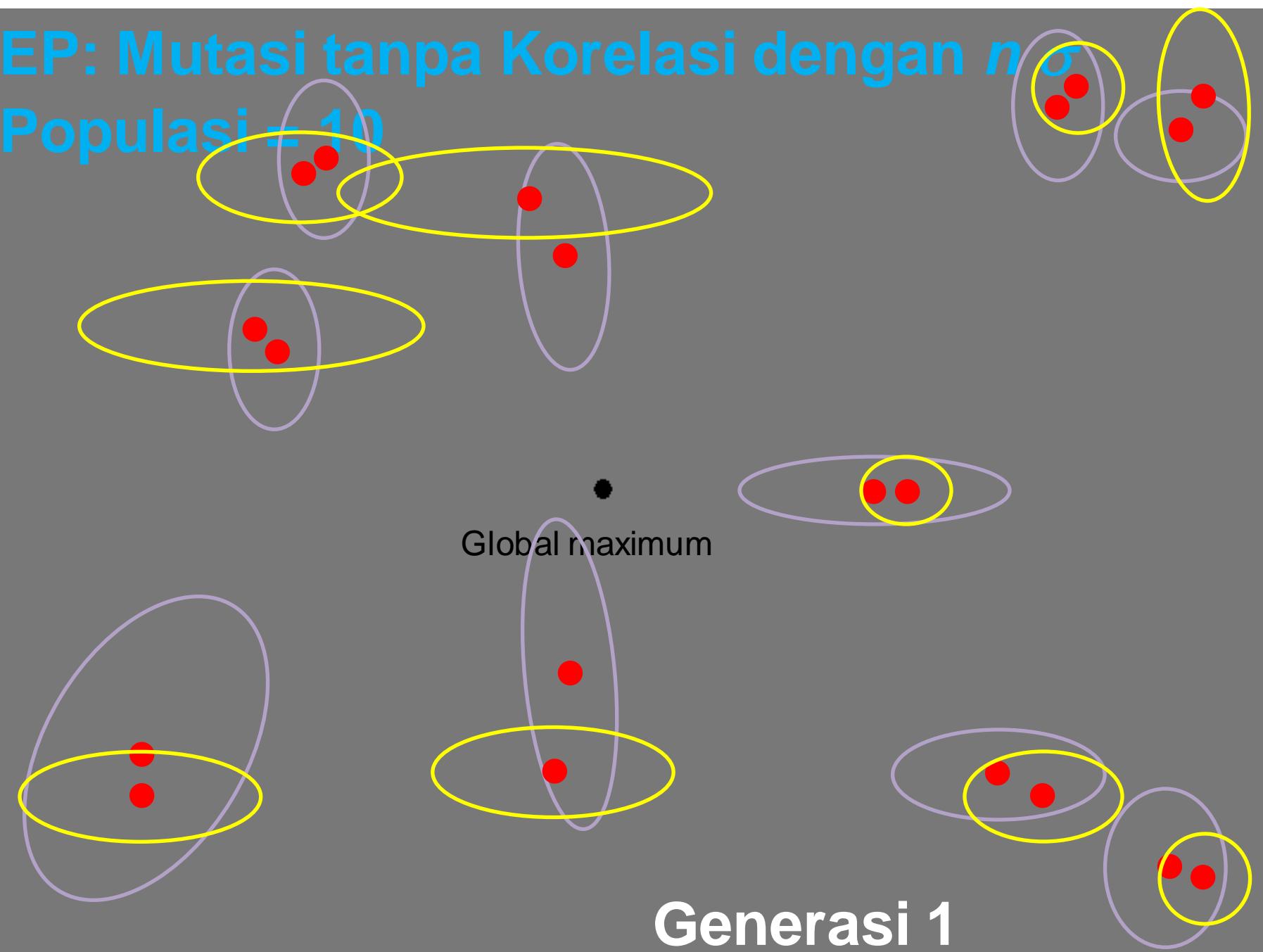
EP: Mutasi tanpa Korelasi dengan $n\sigma$

Populasi = 10



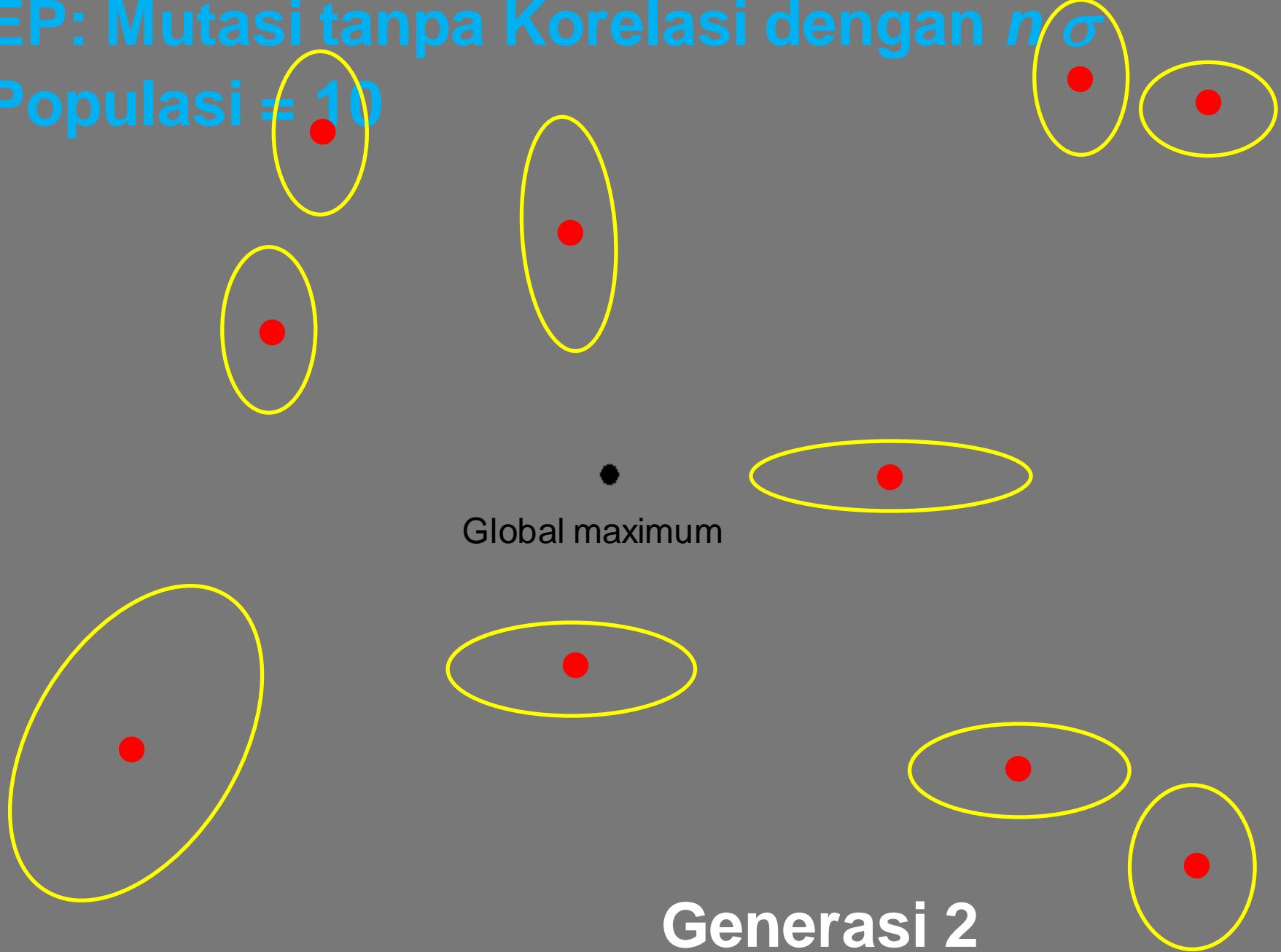
EP: Mutasi tanpa Korelasi dengan $n=5$

Populasi = 10



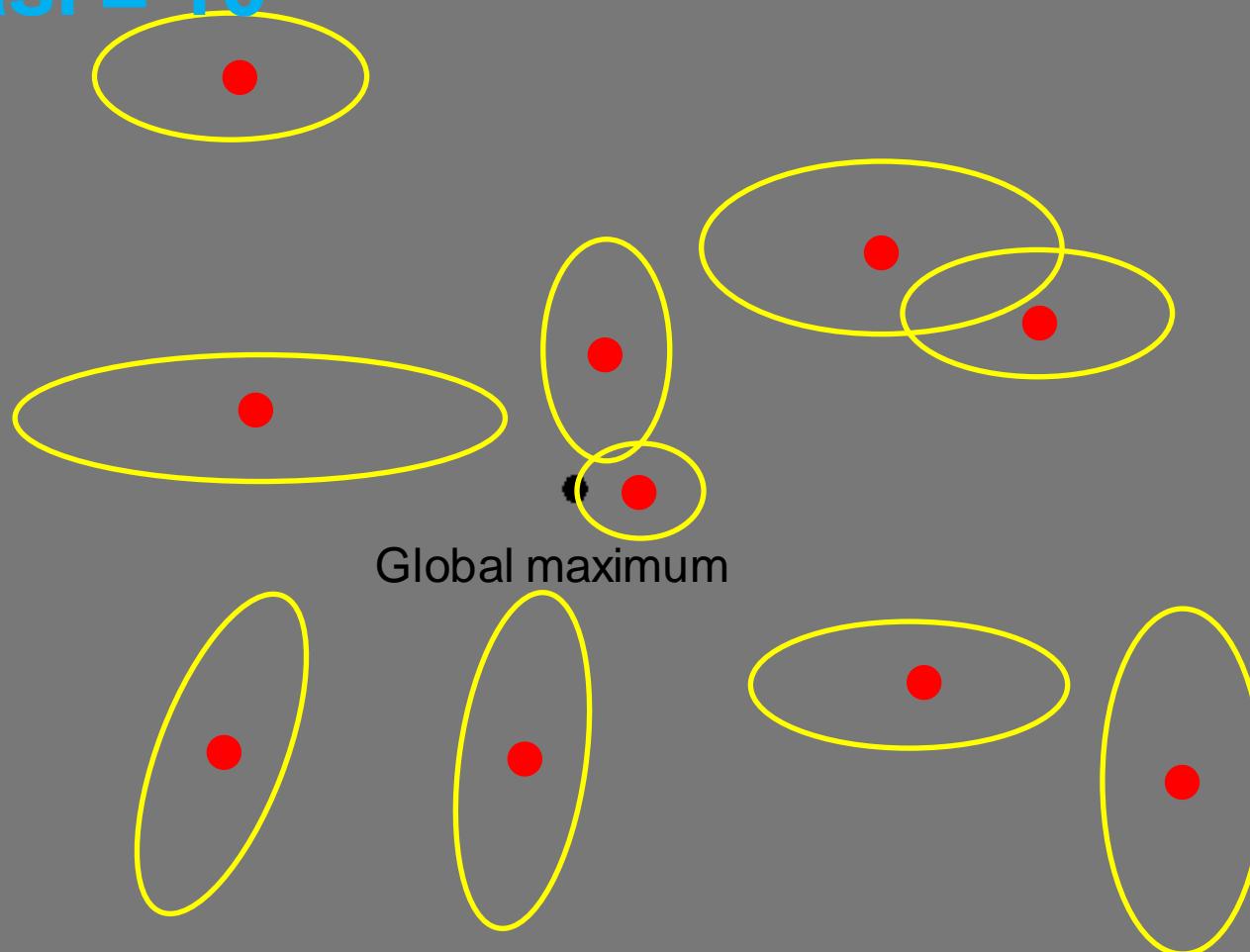
EP: Mutasi tanpa Korelasi dengan $n\sigma$

Populasi = 10



EP: Mutasi tanpa Korelasi dengan $n \sigma$

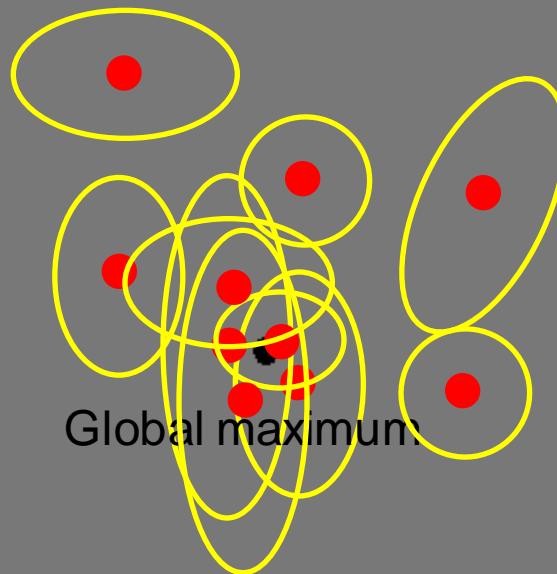
Populasi = 10



Generasi 10

EP: Mutasi tanpa Korelasi dengan $n \sigma$

Populasi = 10



Generasi 50

EP: Mutasi tanpa Korelasi dengan $n \sigma$

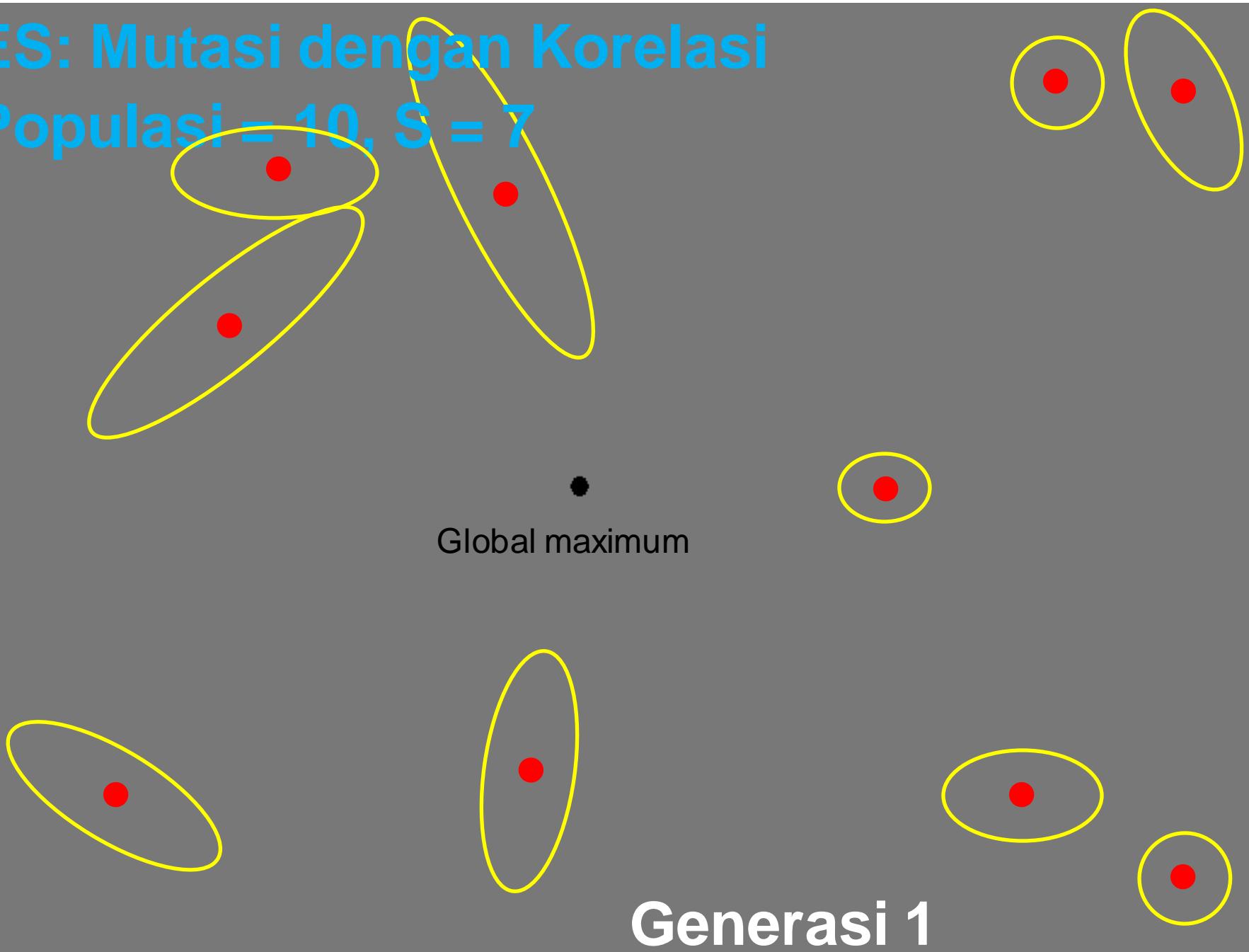
Populasi = 10

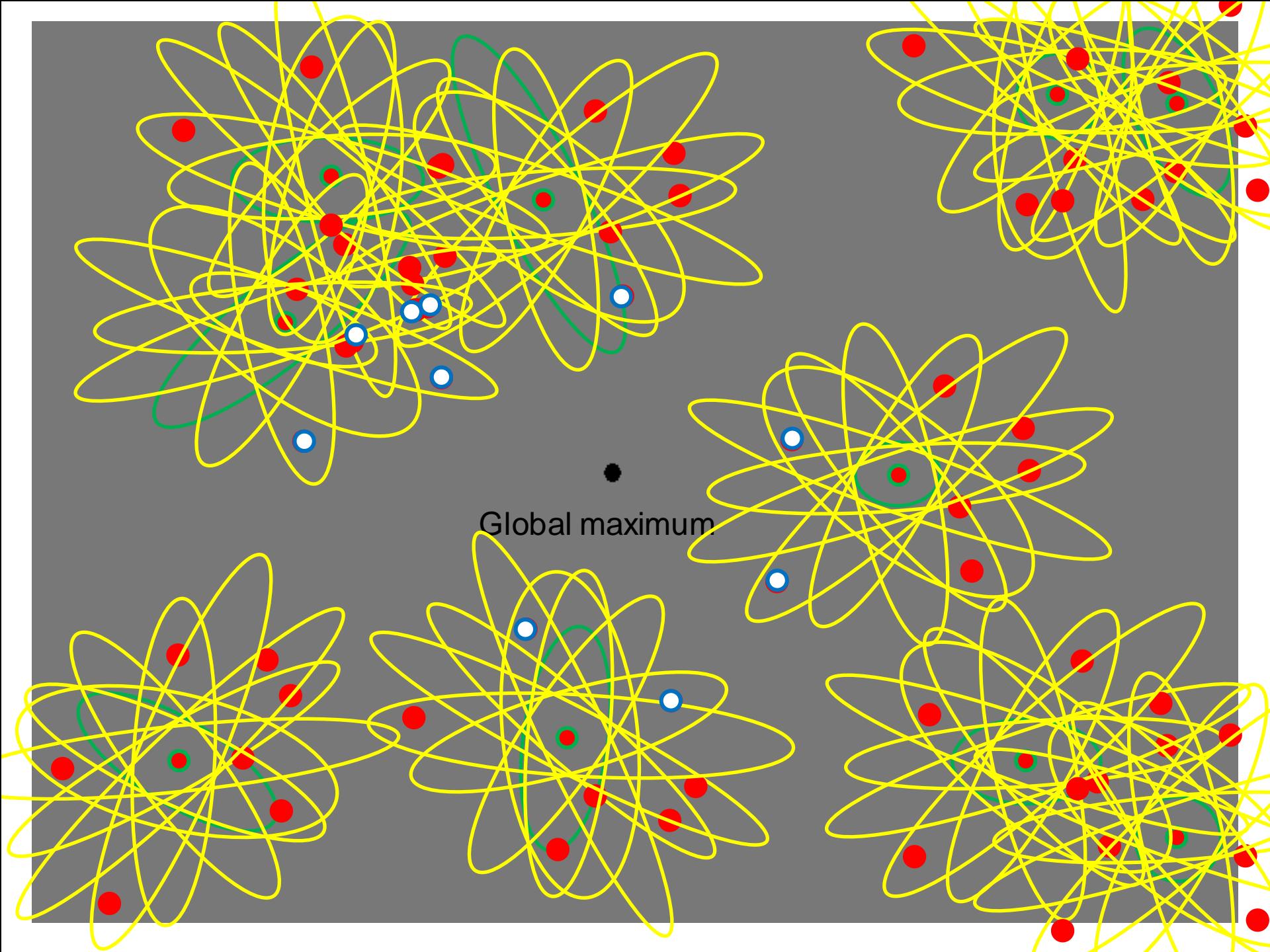


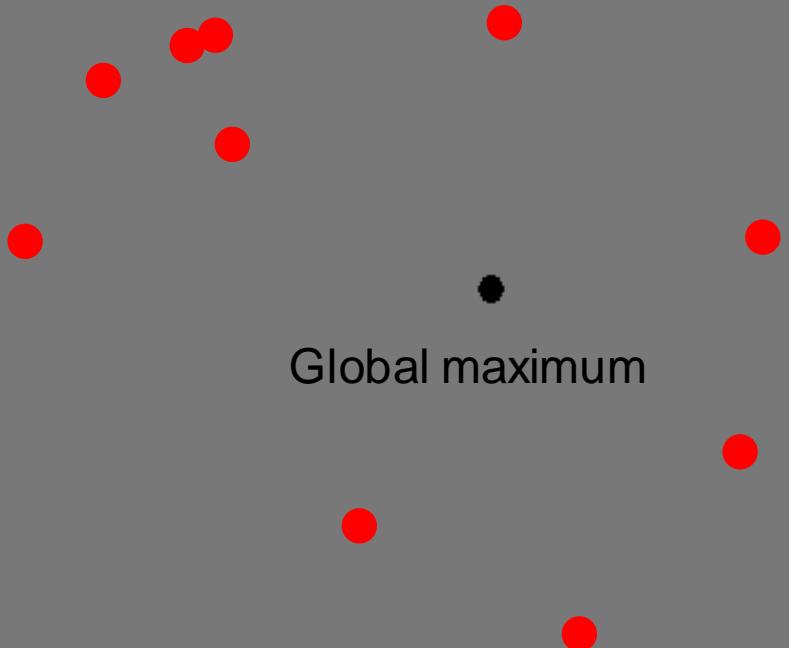
Generasi 100

ES: Mutasi dengan Korelasi

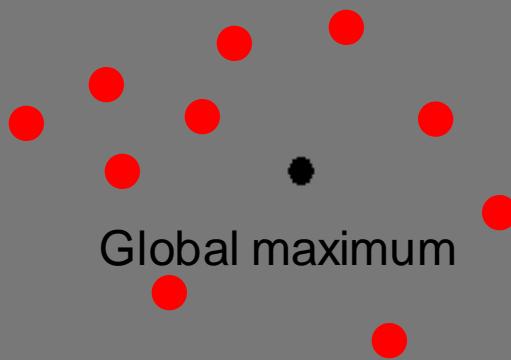
Populasi = 10, S = 7



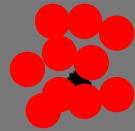




Generasi 2



Generasi 10



Global maximum

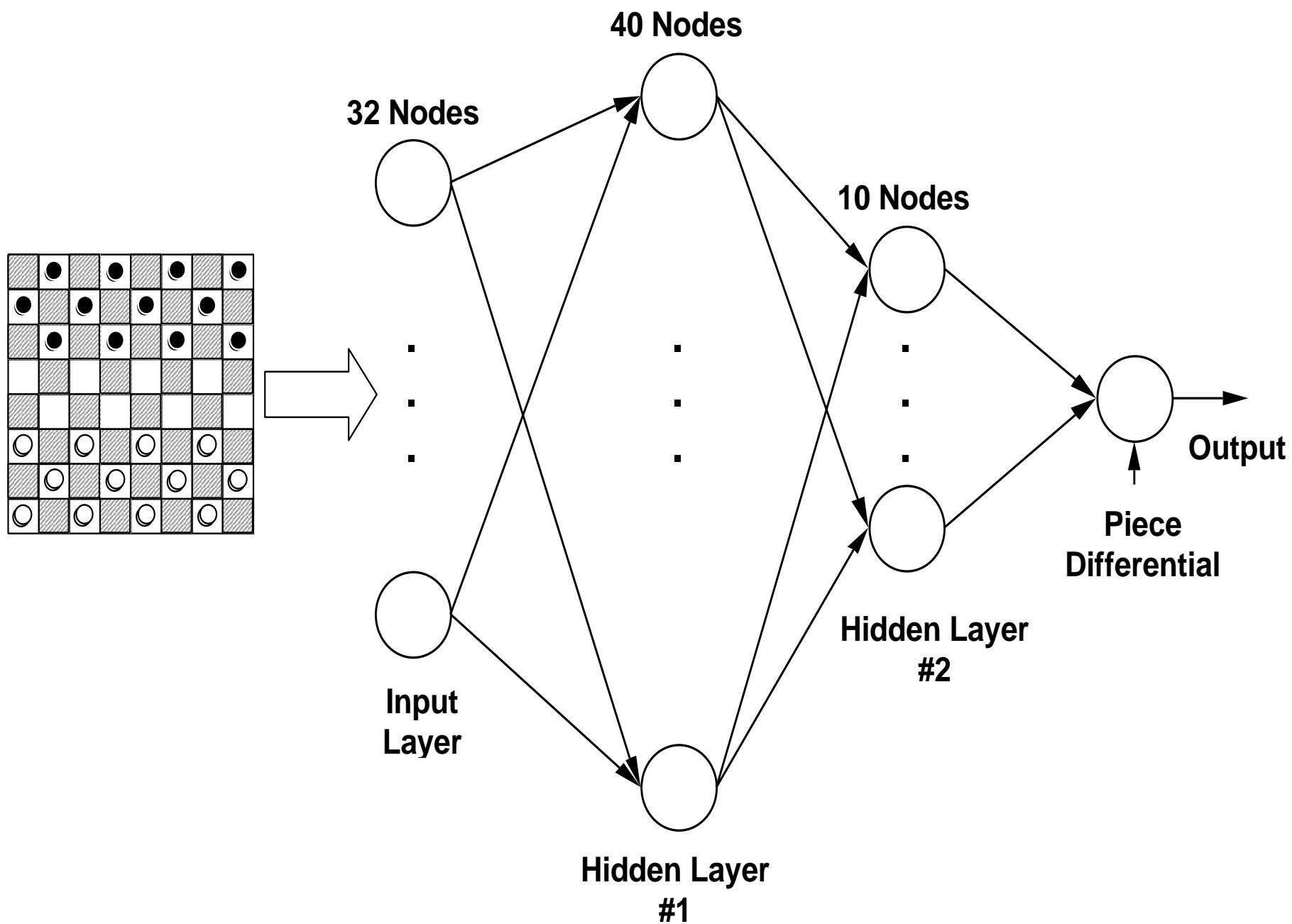
Generasi 20

Aplikasi EP

- *evolving game players*
- *training of artificial neural networks*
- *chemistry and biochemistry*
- *electronic and controller design*
- *fuzzy clustering*
- *general constraint optimization*
- *robotic motion control*

Evolving checkers players (Fogel'02)

- Neural nets for evaluating future values of moves are evolved
- NNs have fixed structure with 5046 weights, these are evolved + one weight for “kings”
- Representation:
 - vector of 5046 real numbers for object variables (weights)
 - vector of 5046 real numbers for σ 's
- Mutation:
 - Gaussian, lognormal scheme with σ -first
 - Plus special mechanism for the kings' weight
- Population size 15



Evolving checkers players (Fogel'02)

- Tournament size $q = 5$
- Programs (with NN inside) play against other programs, no human trainer or hard-wired intelligence
- After 840 generation (6 months!) best strategy was tested against humans via Internet
- Program earned “expert class” ranking outperforming 99.61% of all rated players

Kesimpulan

- Pada awalnya, EP untuk membangun FSM
- Tetapi, dalam perkembangannya EP justru menyerupai ES
- EP hanya menggunakan Mutasi untuk menghasilkan kromosom anak

Daftar Pustaka

- [SUYo8] Suyanto, 2008, Evolutionary Computation: Komputasi Berbasis “Evolusi” dan “Genetika”, penerbit Informatika Bandung.

Genetic Programming (GP)

Dr. Suyanto, S.T., M.Sc.
HP/WA: 0812 845 12345

Intelligence Computing Multimedia (ICM)
Informatics faculty – Telkom University

GA

ES

EP

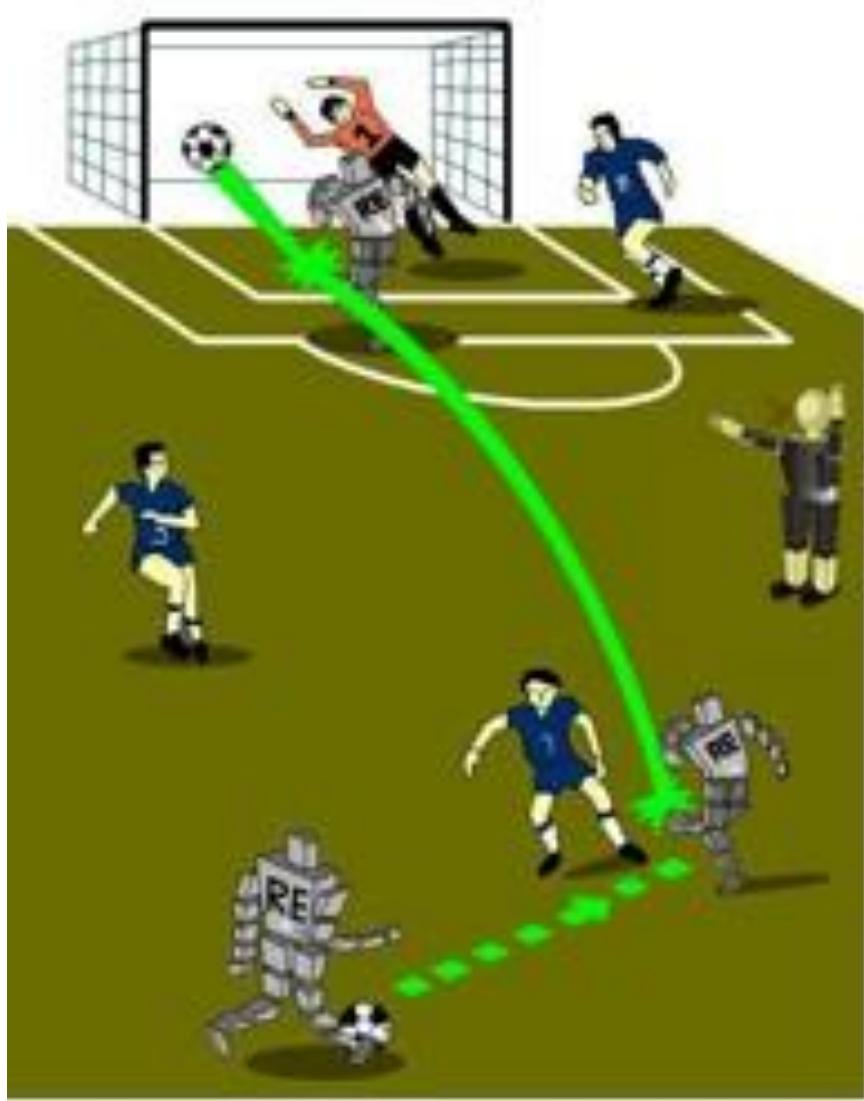
GP?

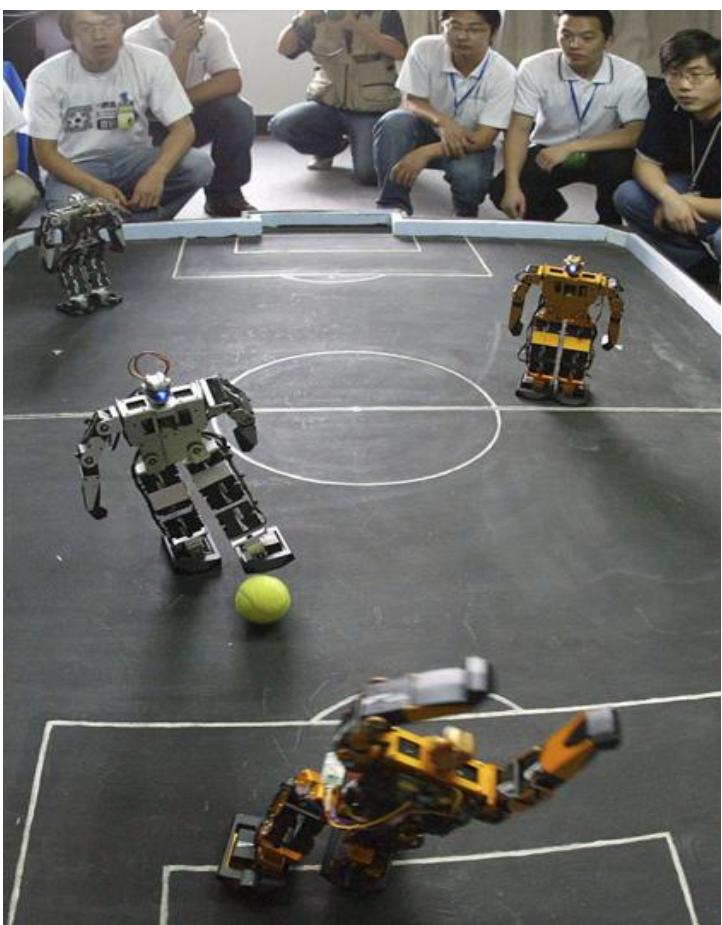


GP for Robot Soccer



WC-2050 is ours !!!





Intro

- Tujuan utama ilmu komputer: *automatic programming*
- Arthur Samuel (1959): “tujuan *automatic programming* adalah bagaimana komputer dapat dibuat mampu mengerjakan hal-hal yang perlu dikerjakan tanpa diberitahu secara pasti bagaimana cara mengerjakannya.
- Misalnya, kita memiliki sangat banyak pasangan data masukan dan keluaran. Bagaimana menemukan suatu program secara otomatis yang bisa memetakan masukan dan keluaran tersebut dengan benar?

Intro

- GP menggunakan representasi *tree* atau *graph*
- Diperkenalkan J. Koza pada era 1990-an di USA
- Untuk *machine learning*: prediksi, klasifikasi, dsb
- Ukuran populasi besar: ribuan individu
- GP lebih lambat dibandingkan GA, ES, dan EP

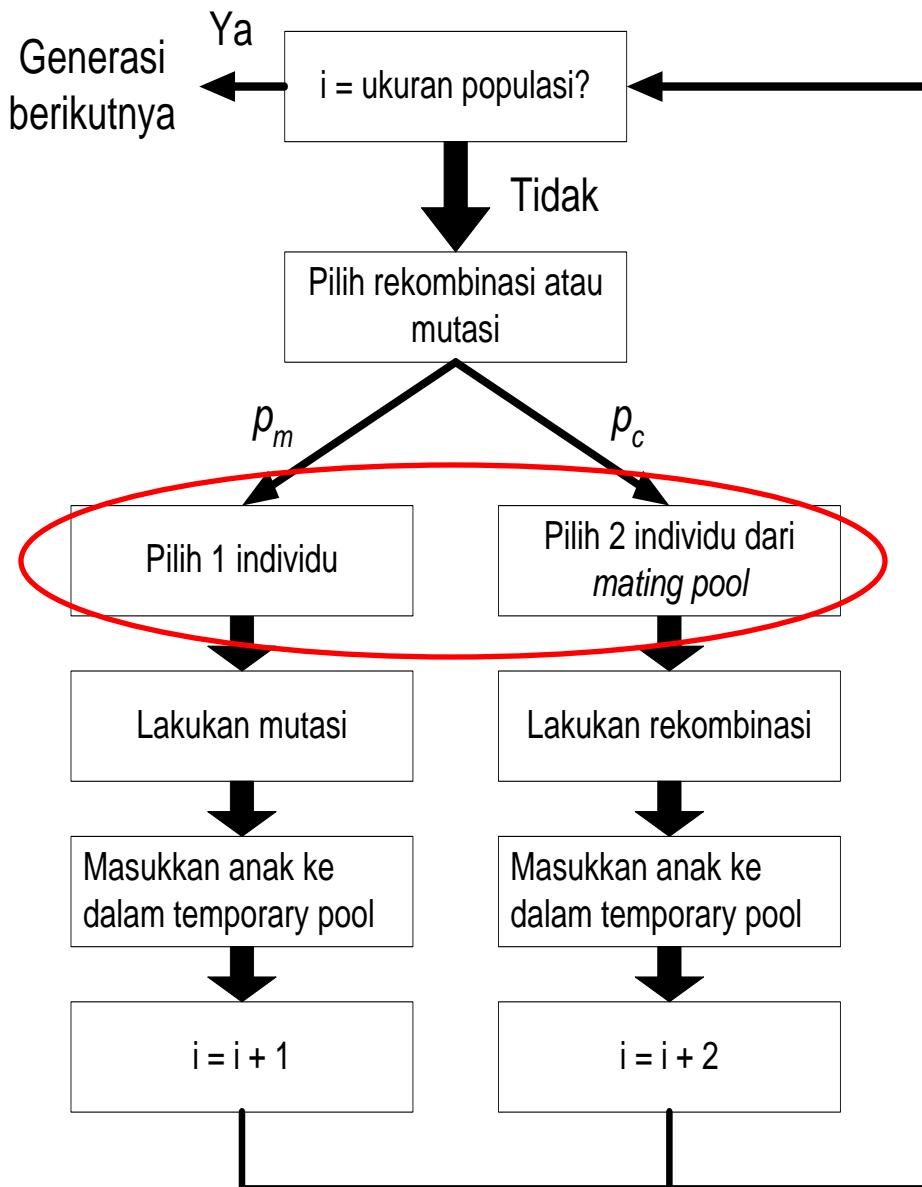
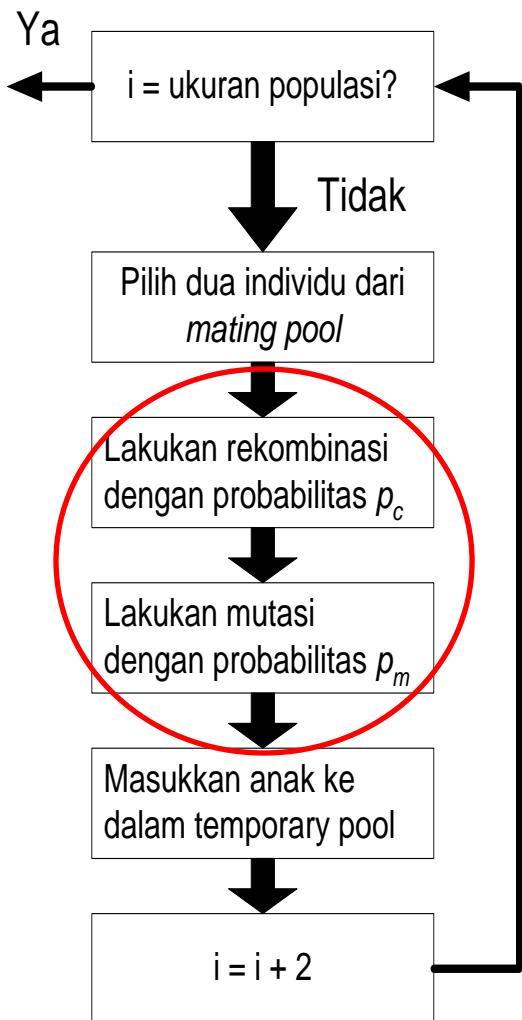
Intro

- Dalam satu populasi, kromosom-kromosom bisa memiliki panjang yang berbeda-beda
- GP bisa menggunakan proses rekombinasi dan mutasi dengan probabilitas tertentu
- Rekombinasi dilakukan dengan cara saling menukar ranting (*sub tree*)
- Sedangkan mutasi dilakukan dengan mengubah pohon secara acak

Spesifikasi teknis GP

Representasi	Struktur pohon (<i>tree</i>)
Seleksi orangtua	Proporsional terhadap <i>fitness</i>
Rekombinasi	Pertukaran ranting (<i>sub tree</i>)
Mutasi	Perubahan acak pada <i>tree</i>
Seleksi survivor	<i>Generational replacement</i>
Ciri khusus	Dalam satu populasi, kromosom-kromosom bisa memiliki panjang yang berbeda-beda

Generasi
berikutnya



GA

GP

Representasi Individu

- Misalkan si A menghadapi masalah dalam proses seleksi pegawai tahun ini
- Dia bingung bagaimana menemukan model yang tepat untuk proses seleksi tahun ini
- Aturan seleksi yang pernah digunakan tahun lalu tidak dia temukan
- Satu-satunya dokumen proses seleksi pegawai tahun lalu adalah data sebagai berikut:

Data histori penerimaan pegawai

Kandidat	IPK	Psikologi	Diterima
1	2,98	73	Tidak
2	3,08	97	Ya
3	2,56	83	Tidak
4	3,01	84	Ya
...

Kromosom

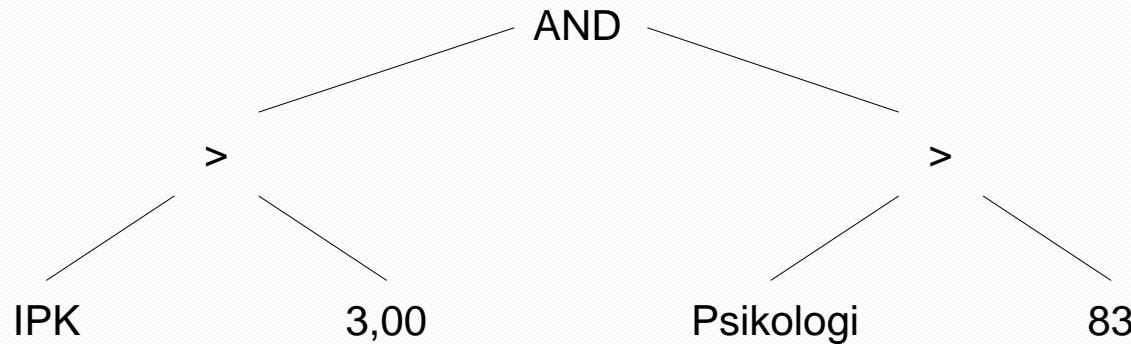
Misalkan, suatu model yang bisa digunakan untuk masalah tersebut adalah

IF (IPK > 3,00 AND Psikologi > 83)

THEN Diterima

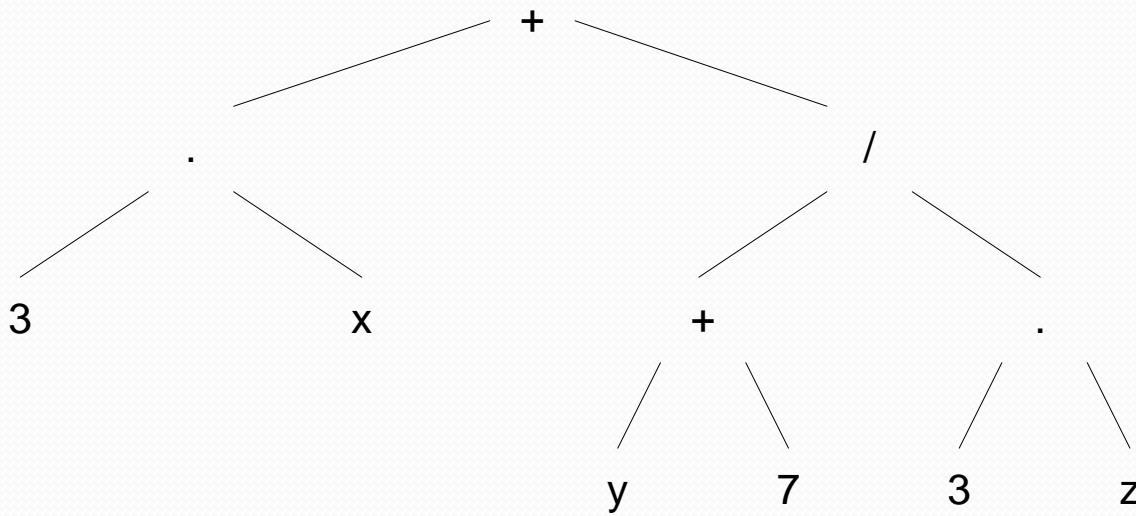
ELSE Tidak.

Kromosom



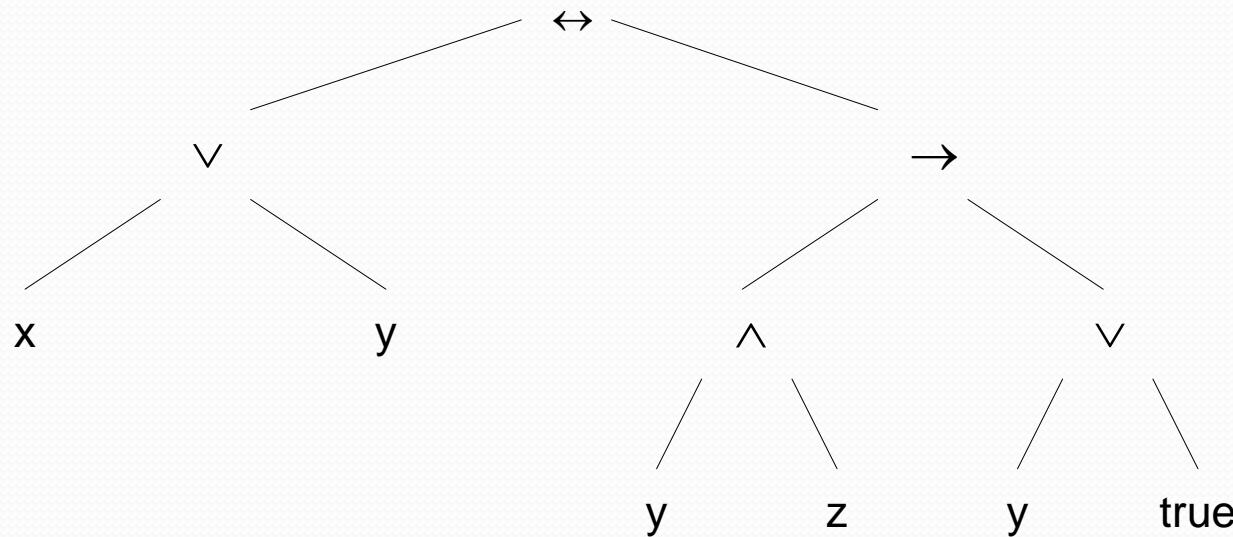
- Struktur pohon tersebut bisa dibayangkan sebagai suatu kromosom yang bisa ber-evolusi dengan proses rekombinasi dan mutasi.
- Untuk permasalahan yang lebih kompleks, misalnya jumlah atribut masukan lebih dari dua, tentu saja diperlukan waktu yang lama untuk menemukan model yang tepat.
- Proses evolusi pada GP sengaja dibuat untuk mengatasi masalah ini.

Kromosom: Formula Aritmetika



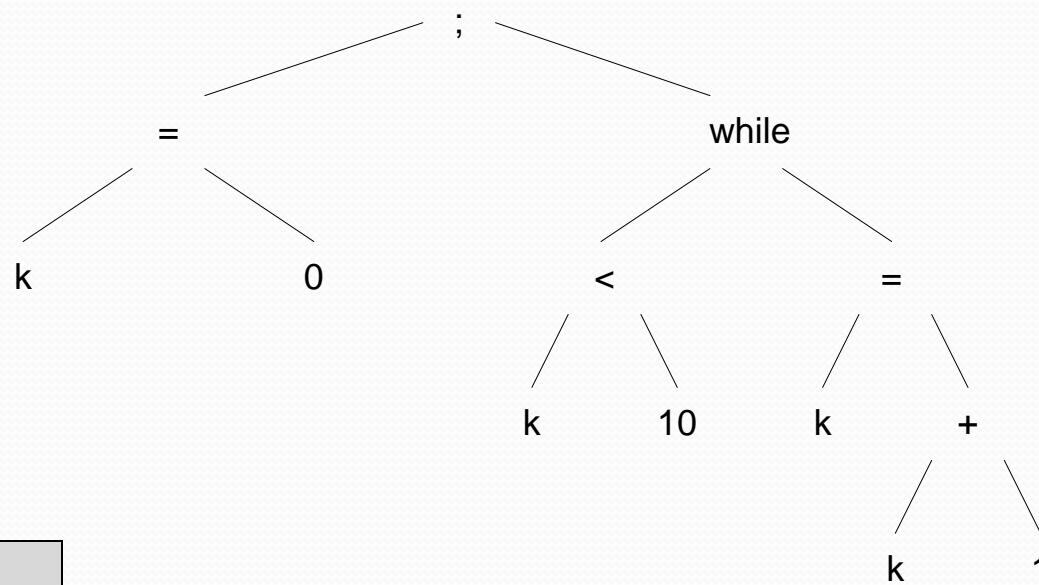
$$3x + \frac{(y + 7)}{3z}$$

Kromosom: Formula Logika



$$(x \vee y) \leftrightarrow ((y \wedge z) \rightarrow (y \vee \text{true}))$$

Kromosom: Program



```
k = 0;
while k < 10
{
    k = k +1
}
```

Mapping

- Bagaimana merepresentasikan suatu program ke dalam kromosom dan sebaliknya?
- Nichael Lynn Cramer (1985) mengusulkan cara pembangunan kromosom untuk meng-evolusi program dalam bahasa pemrograman PL
- Idenya: representasi integer seperti pada GA
- Terdapat dua pendekatan pemataan (*mapping*) yang diusulkan:
 - JB *mapping*
 - TB *mapping*

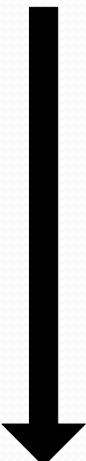
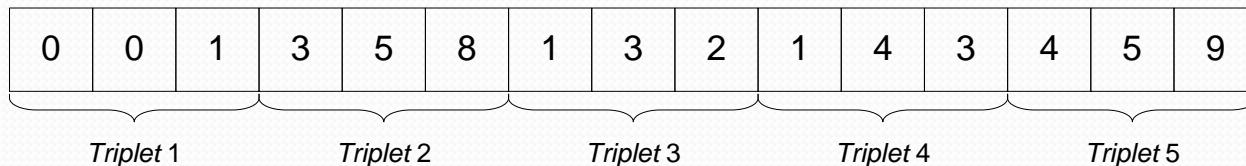
JB Mapping

Operasi bahasa pemrograman PL dan kode *integer*-nya

Kode integer	Operasi
0	BLOCK
1	LOOP
2	SET
3	ZERO
4	INC

JB Mapping

Genotype: representasi integer



(0 0 1) ;;main statement ? (: BLOCK AS0 AS1)
(3 5 8) ;;auxiliary statement 0 ? (: ZERO V5)
(1 3 2) ;;auxiliary statement 1 ? (: LOOP V3 AS2)
(1 4 3) ;;auxiliary statement 2 ? (: LOOP V4 AS3)
(4 5 9) ;;auxiliary statement 3 ? (: INC V5)

Phenotype: program PL

```
; ; Program perkalian: Kalikan V3 dengan V4 dan simpan hasilnya di V5
(: ZERO V5)
(: LOOP V3 (: LOOP V4 (: INC V5)))
```

TB *mapping*

- Kalau JB *mapping* menggunakan *auxiliary statement* (AS), TB *mapping* bekerja secara rekursif untuk mendekodekan ekspresi
- Hal ini membuat kromosom yang berbasis TB *mapping* menjadi lebih singkat

JB Mapping

Genotype: representasi integer

0	0	1	3	5	8	1	3	2	1	4	3	4	5	9
Triplet 1			Triplet 2			Triplet 3			Triplet 4			Triplet 5		

TB Mapping

(0 (3 5) (1 3 (1 4 (4 5))))

```
; Program perkalian: Kalikan V3 dengan V4 dan simpan hasilnya di V5
(: ZERO V5)
(: LOOP V3 (: LOOP V4 (: INC V5)))
```

GP untuk bahasa LISP

- Bagaimana penggunaan GP untuk bahasa pemrograman LISP (**LIS**t Programming), yaitu suatu bahasa pemrograman yang berbasis *list* (senarai)
- Pada masa-masa awal kemunculannya, GP banyak difokuskan pada bahasa LISP
- Pada LISP, suatu ekspresi program dituliskan secara *prefix*, yaitu menggunakan aturan **operasi operan operan**

GP untuk bahasa LISP

- Sebagai contoh, suatu ekspresi LISP berikut:

$(+ (* 5 x) (* 7 y))$

menyatakan ekspresi matematika

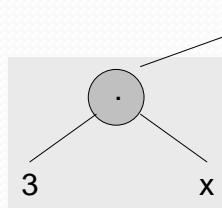
$5x + 7y$

- Berapa banyak kode operasi yang harus digunakan?
- Tentu saja bergantung pada masalah yang dihadapi.
- Semakin banyak operasi yang digunakan, maka GP bisa digunakan untuk menyelesaikan lebih banyak masalah tetapi tentu saja komputasinya semakin kompleks

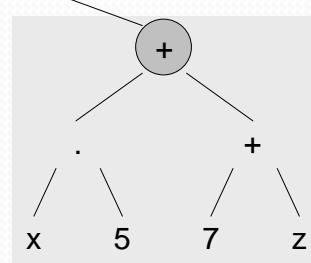
Seleksi Orangtua

- Pada GP, seleksi ortu biasanya dilakukan secara proporsional terhadap nilai *fitness*.
- GP biasanya menggunakan populasi yang berukuran sangat besar.
- Agar lebih efisien, seleksi ortu dilakukan dengan cara:
 - Buat perankingan berdasarkan *fitness* dan bagi individu ke dalam dua grup.
 - Grup 1 berisi $x\%$ individu terbaik dan Grup 2 berisi $(100-x)\%$ individu yang lain.
 - Lakukan 80% seleksi ortu pada Grup 1 dan 20% pada Grup 2.
 - Untuk ukuran populasi = 1000, 2000, 4000, dan 8000 gunakan secara berturut-turut $x = 32\%, 16\%, 8\%$ dan 4% .

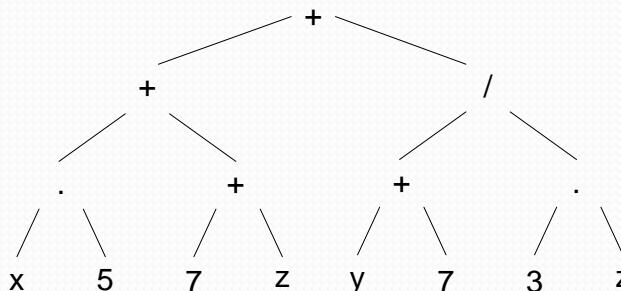
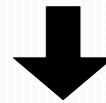
Rekombinasi



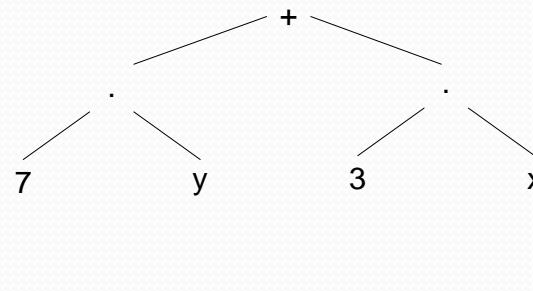
Orangtua 1



Orangtua 2

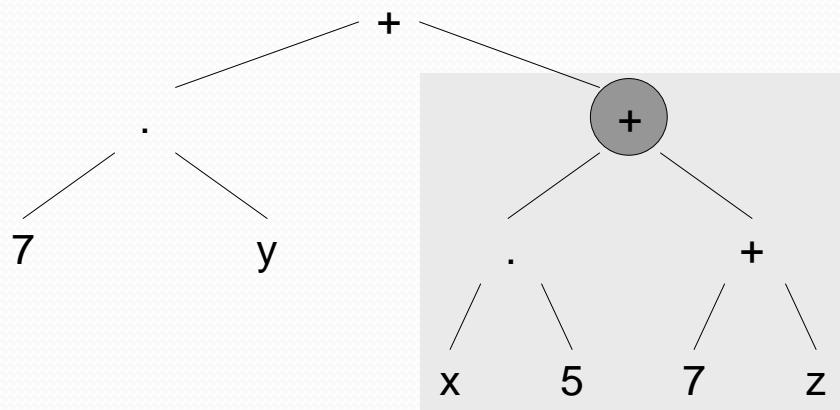


Anak 1

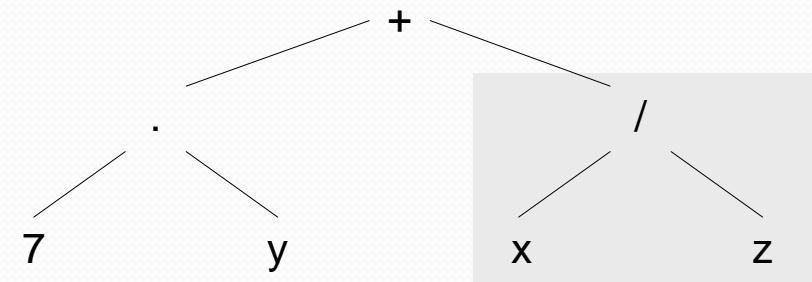


Anak 2

Mutasi



Kromosom awal



Kromosom hasil mutasi

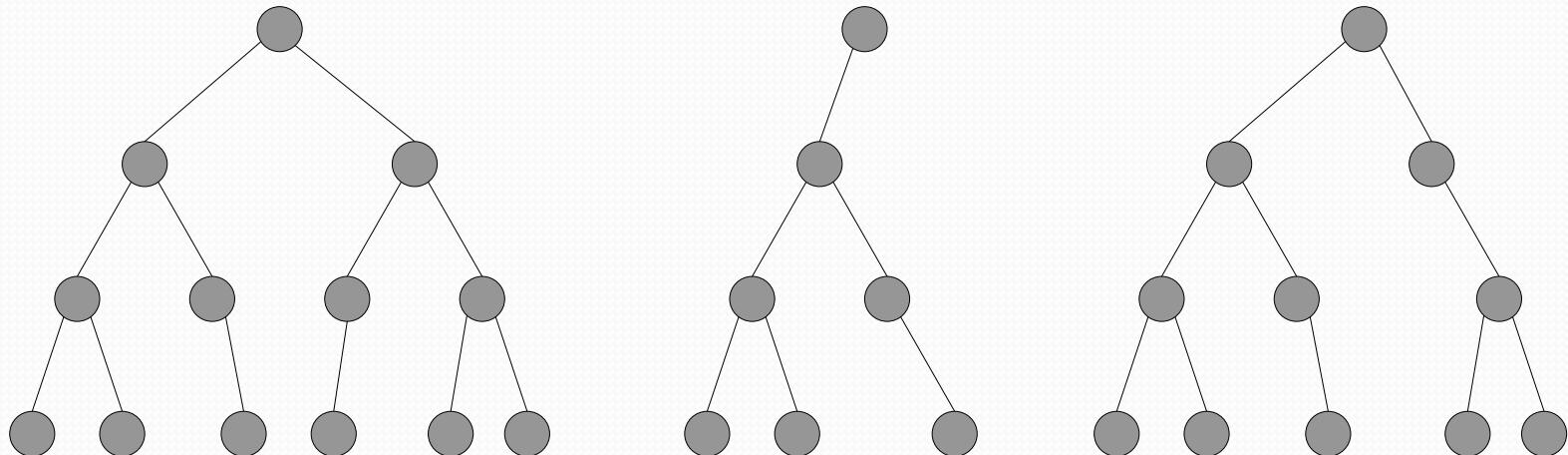
Seleksi Survivor

- Umumnya GP menggunakan model populasi *generational scheme* sehingga tidak ada seleksi survivor.
- Tetapi, akhir-akhir ini model populasi *steady-state* lebih populer dan banyak digunakan dibandingkan *generational scheme*.

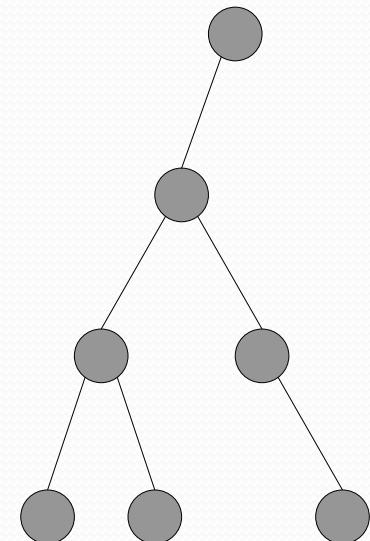
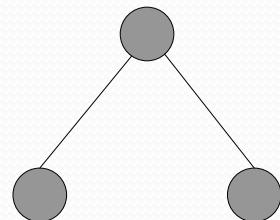
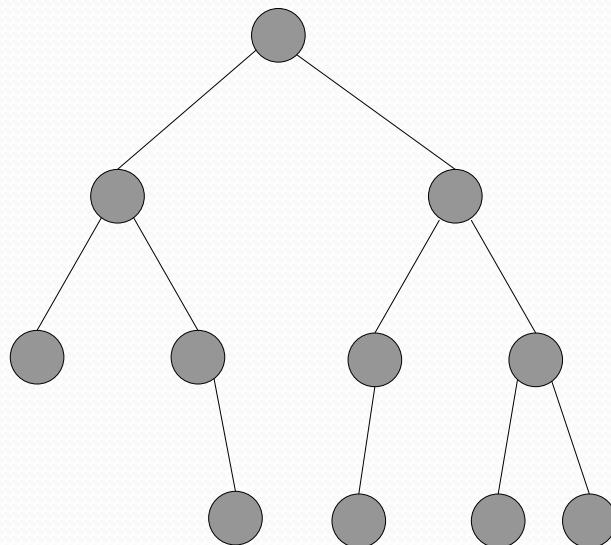
Inisialisasi

- Karena kromosom pada GP bisa memiliki ukuran yang berbeda-beda, maka pada generasi pertama kita bisa melakukan proses inisialisasi dengan membatasi kedalaman pohon.
- Misalkan, kedalaman pohon maksimum adalah D_{\max} . Selanjutnya, dengan menggunakan kedalaman maksimum ini kita bisa membangkitkan kromosom awal dengan cara:
 - ***Full method***
 - ***Grow method***
 - ***Ramped half-and-half***

Inisialisasi: Full method



Inisialisasi: Grow method



Inisialisasi: *Ramped half-and-half*

- Metode ini adalah yang paling umum digunakan pada GP.
- Pada metode ini pembangkitan kromosom dilakukan menggunakan kombinasi *full method* dan *grow method*.
- Caranya bisa bervariasi. Misalnya, 50% kromosom dibangkitkan menggunakan *grow method* dan 50% kromosom lainnya dibangkitkan menggunakan *full method*.
- Cara lain yang bisa digunakan adalah dengan membangkitkan bilangan acak, misalnya r (dimana r dalam interval $[2, D_{\max}]$), sebagai batas kedalaman pada setiap pembangkitan pohon.
- Dengan demikian, cara ini juga merupakan gabungan dari *full method* dan *grow method*.
- Metode *Ramped half-and-half* memang lebih disukai karena proses inisialisasi bisa menghasilkan pohon dengan kedalaman yang bervariasi.

Aplikasi GP

- *symbolic regression*
- *grammar induction*
- *data mining and data analysis*
- *logic function synthesis*
- *circuit design and layout*
- *high-level circuit design*
- *medicine*
- *breeding financial and trading rules*
- *microwave antenna design*
- *finding cellular automata rules*
- *learning of rules for geometric structures*
- *automated programming*

Kesimpulan

- GP digunakan untuk *autamtic programming*
- GP biasanya menggunakan populasi berukuran besar
- Kromosom merepresentasikan tree atau graph
- GP lebih fokus pada bahasa LISP

Daftar Pustaka

- [SUYo8] Suyanto, 2008, Evolutionary Computation: Komputasi Berbasis “Evolusi” dan “Genetika”, penerbit Informatika Bandung.

Differential Evolution (DE)

Dr. Suyanto, S.T., M.Sc.
HP/WA: 0812 845 12345

Intelligence Computing Multimedia (ICM)
Informatics faculty – Telkom University

Intro

- *Differential Evolution* (DE) merupakan suatu metode optimasi dengan pendekatan heuristik untuk mencari nilai minimum dari fungsi ruang kontinyu yang nonlinier dan *non-differentiable* [STO95a]
- Termasuk kelas *evolution strategies* (ES)
- Bisa menemukan minimum global dari fungsi multidimensional dan multimodal, yaitu fungsi yang memiliki lebih dari satu nilai minimum, dengan probabilitas tinggi.

Intro

- Apa yang membedakan DE dengan EAs yang lain?
- Satu kata kunci = *differential mutation*
- yang merupakan mutasi semi terarah dan bisa dianggap sebagai operasi pra-seleksi khusus

$f(x_1, x_2)$



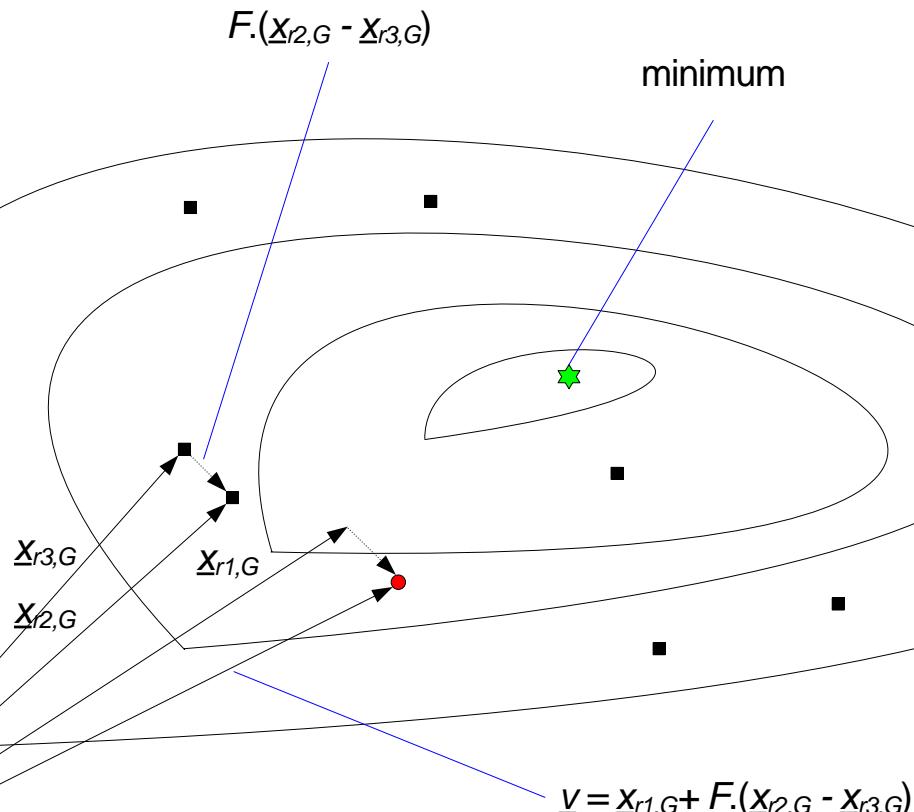
X_2

X_1

- Sejumlah NP vektor parameter pada generasi G
- Vektor parameter \underline{v} baru yang dihasilkan
- ★ Nilai minimum global yang dicari

Differential mutation

Skema DE1



X_1

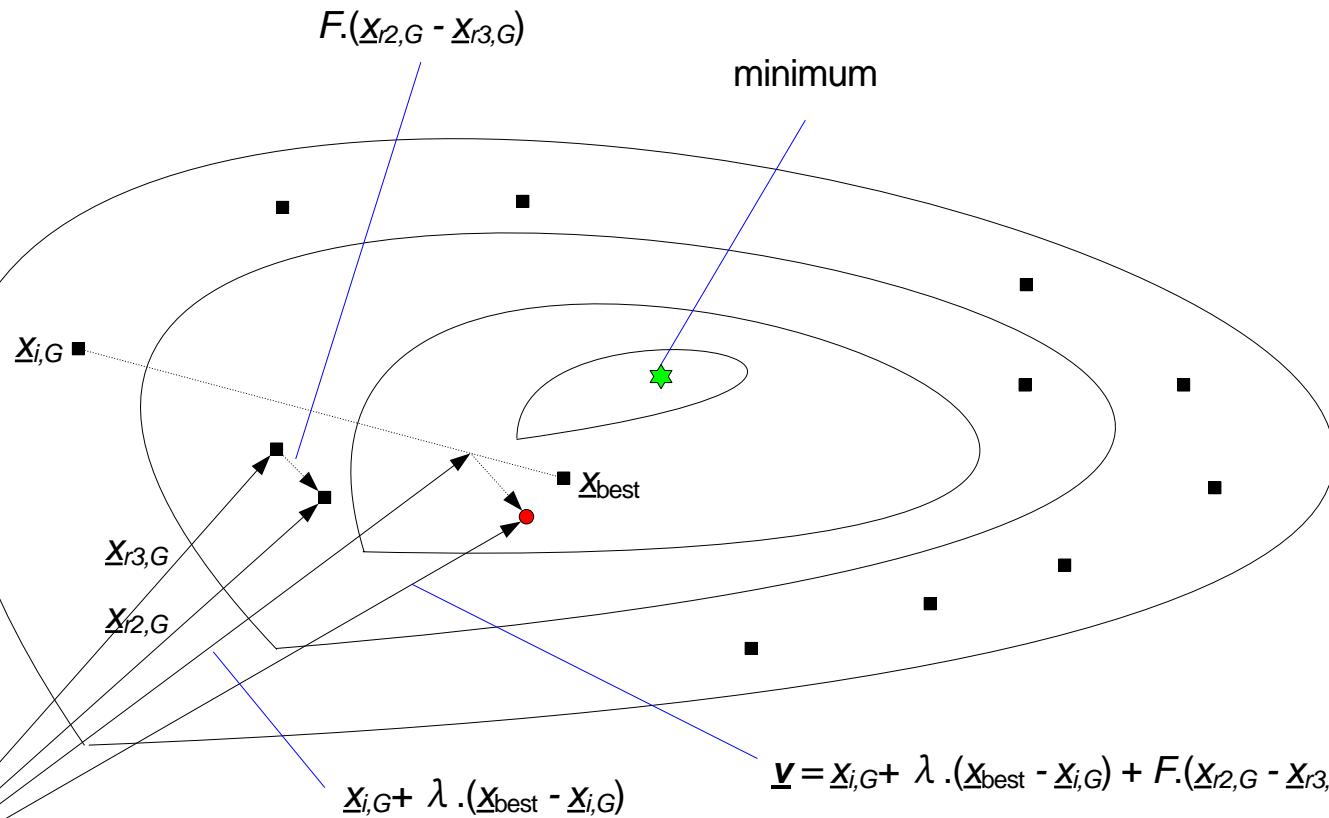
x_2

x_1

- Sejumlah NP vektor parameter pada generasi G
- Vektor parameter v baru yang dihasilkan
- ★ Nilai minimum global yang dicari

Differential mutation

Skema DE2



x_1

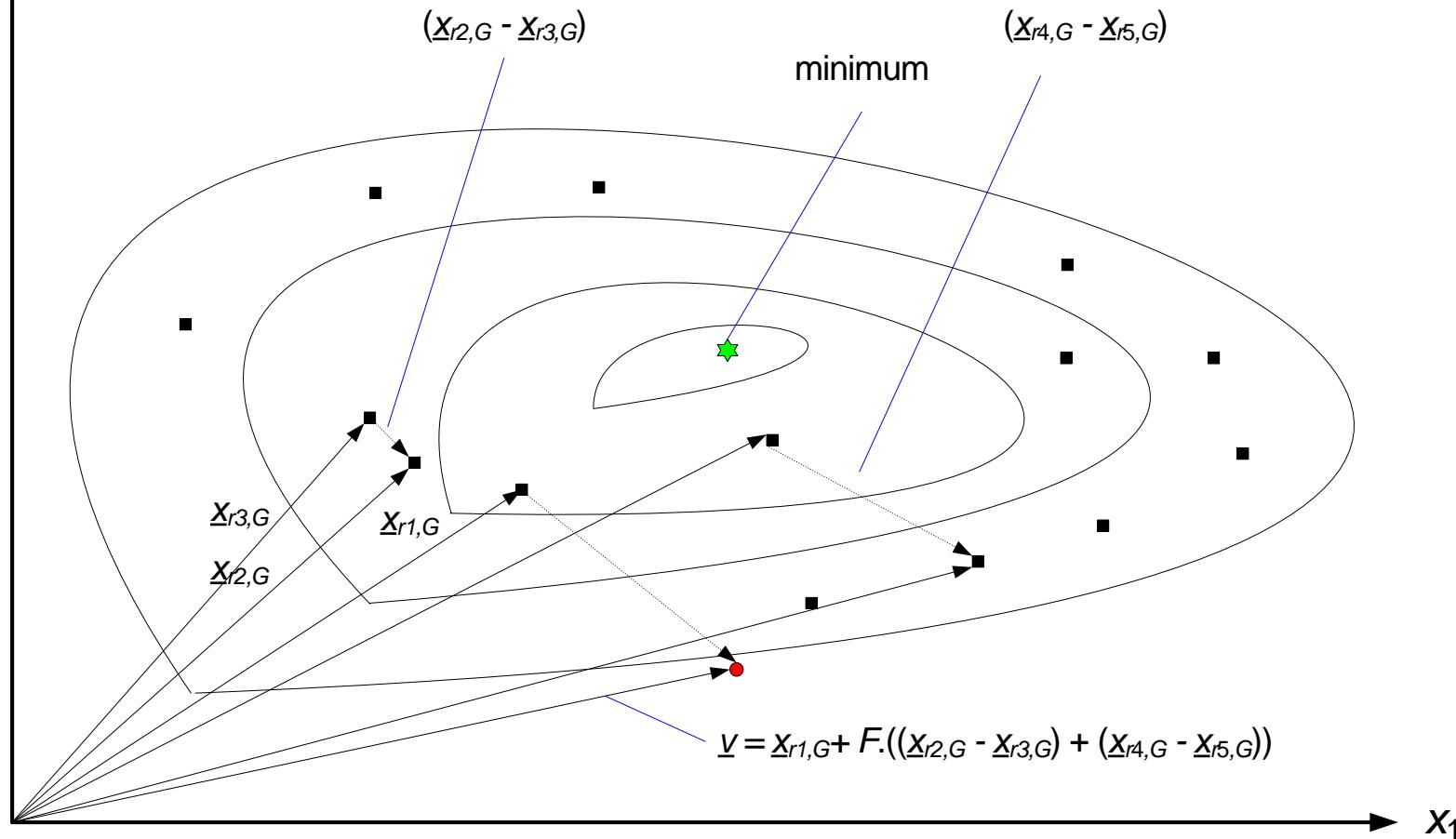
X_2

\uparrow

- Sejumlah NP vektor parameter pada generasi G
- Vektor parameter v baru yang dihasilkan
- ★ Nilai minimum global yang dicari

Differential mutation

Skema DE3



$f(x_1, x_2)$



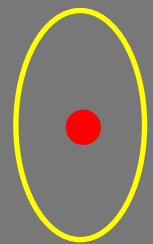
EP: Mutasi tanpa Korelasi dengan $n \sigma$

Populasi = 1



Global maximum

Generasi 1

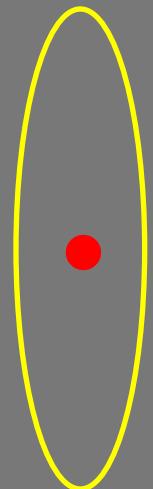


EP: Mutasi tanpa Korelasi dengan $n \sigma$

Populasi = 1



Global maximum



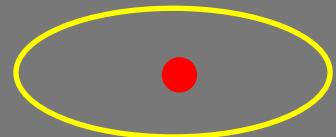
Generasi 2

EP: Mutasi tanpa Korelasi dengan $n \sigma$

Populasi = 1



Global maximum

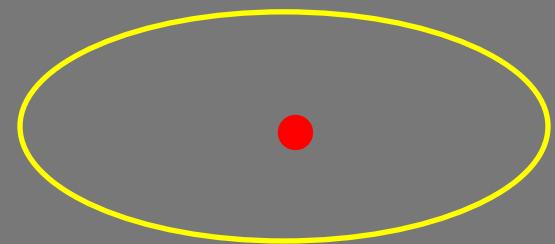


Generasi 3

EP: Mutasi tanpa Korelasi dengan $n \sigma$

Populasi = 1

Global maximum



Generasi 10

EP: Mutasi tanpa Korelasi dengan $n \sigma$

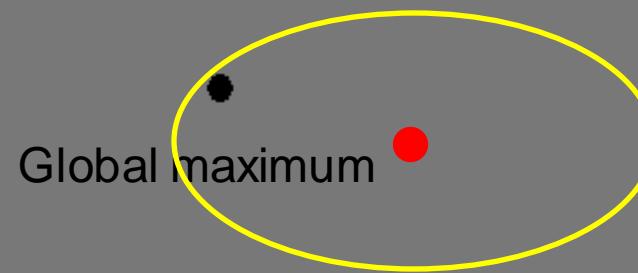
Populasi = 1



Generasi 50

EP: Mutasi tanpa Korelasi dengan $n \sigma$

Populasi = 1



Generasi 100

EP: Mutasi tanpa Korelasi dengan $n \sigma$

Populasi = 1



Global maximum

Generasi 200

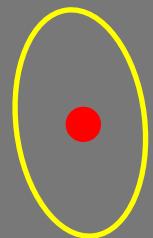
ES: Mutasi dengan Korelasi

Populasi = 1



Global maximum

Generasi 1

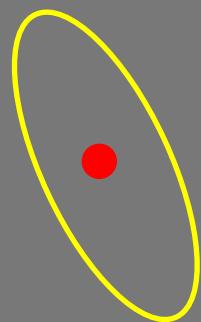


ES: Mutasi dengan Korelasi

Populasi = 1



Global maximum



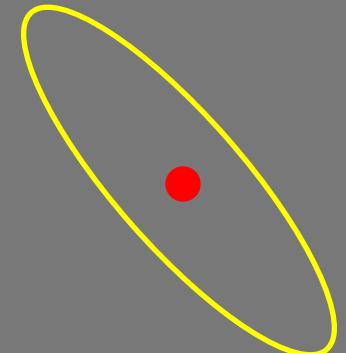
Generasi 2

ES: Mutasi dengan Korelasi

Populasi = 1



Global maximum

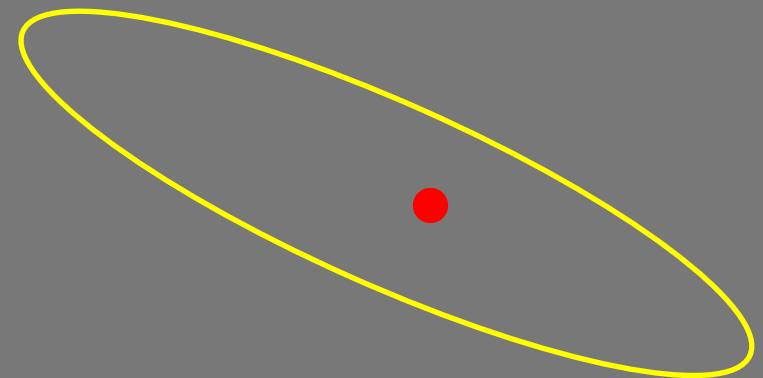


Generasi 3

ES: Mutasi dengan Korelasi

Populasi = 1

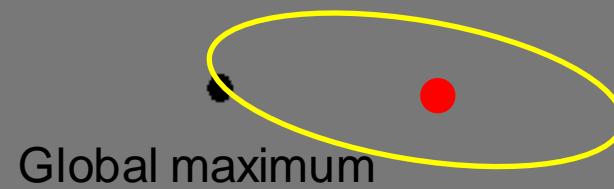
Global maximum



Generasi 10

ES: Mutasi dengan Korelasi

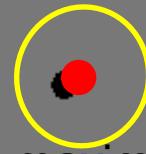
Populasi = 1



Generasi 20

ES: Mutasi dengan Korelasi

Populasi = 1

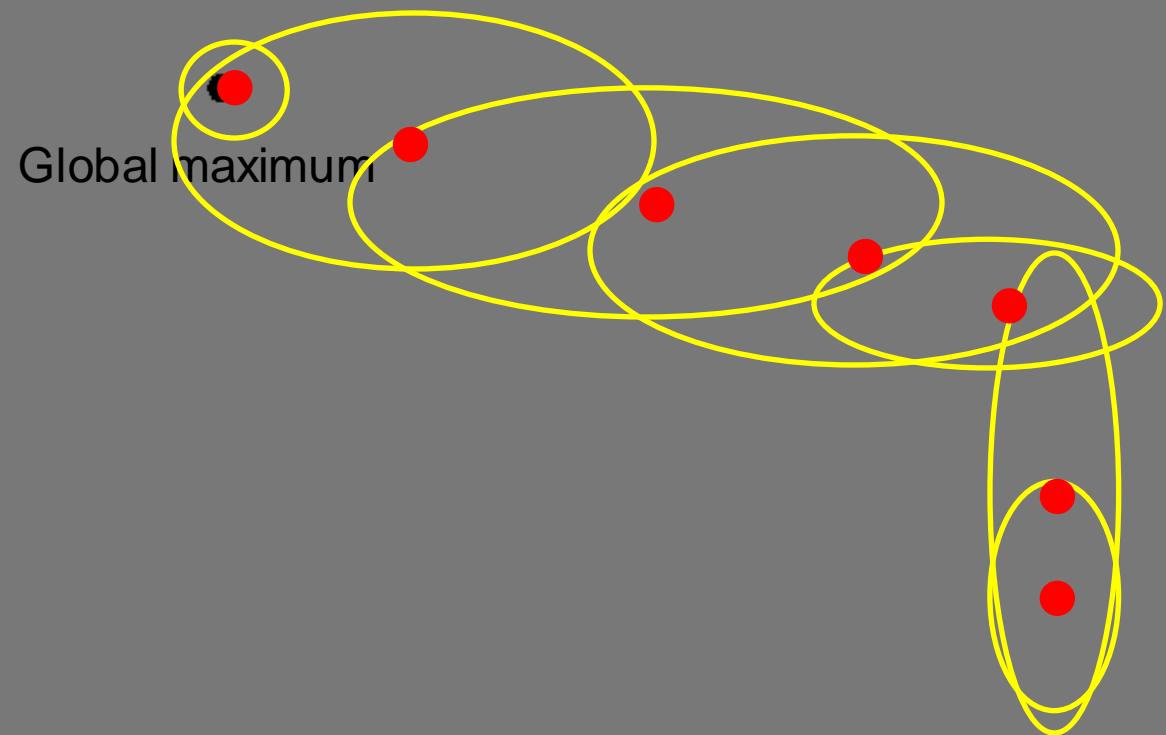


Global maximum

Generasi 50

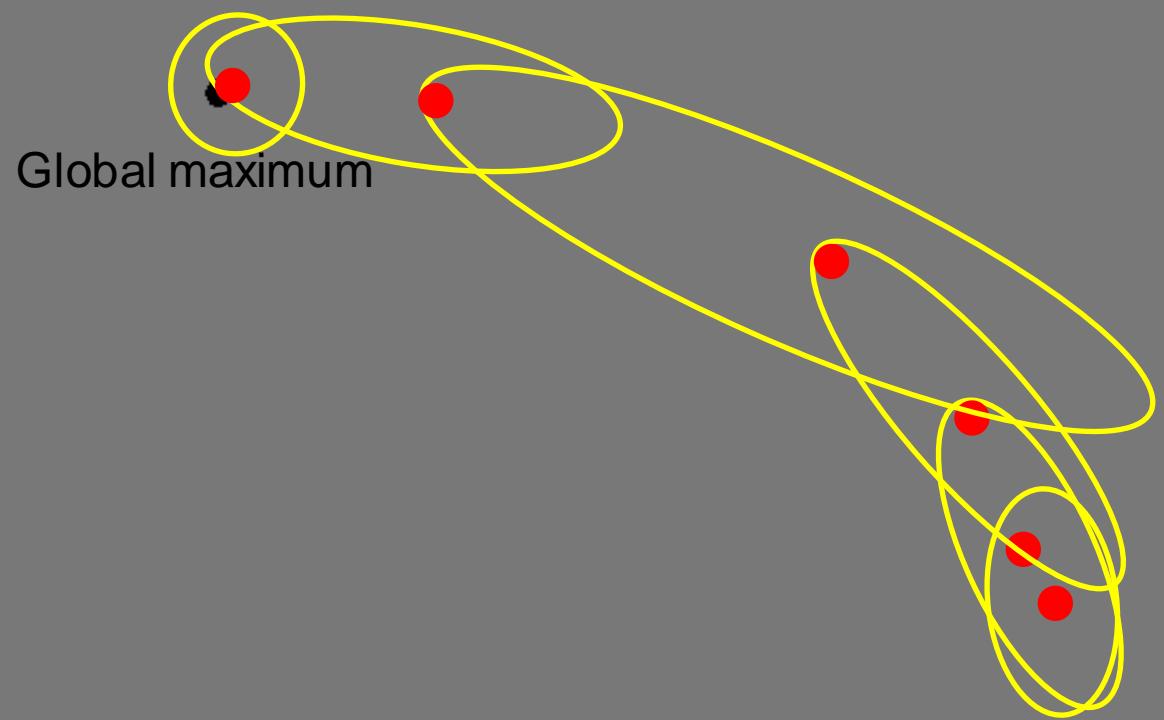
EP: Mutasi tanpa Korelasi dengan $n \sigma$

Populasi = 1



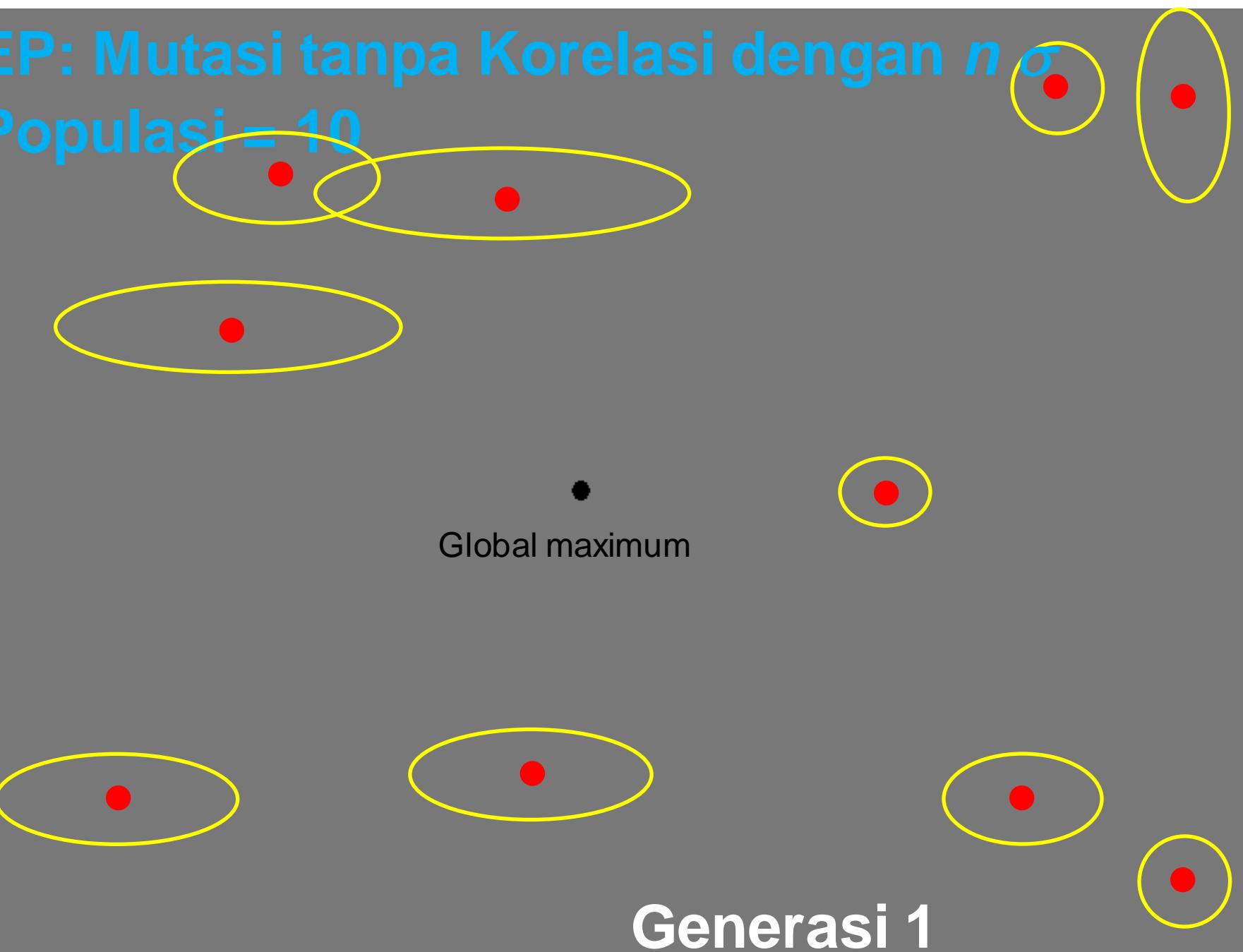
ES: Mutasi dengan Korelasi

Populasi = 1



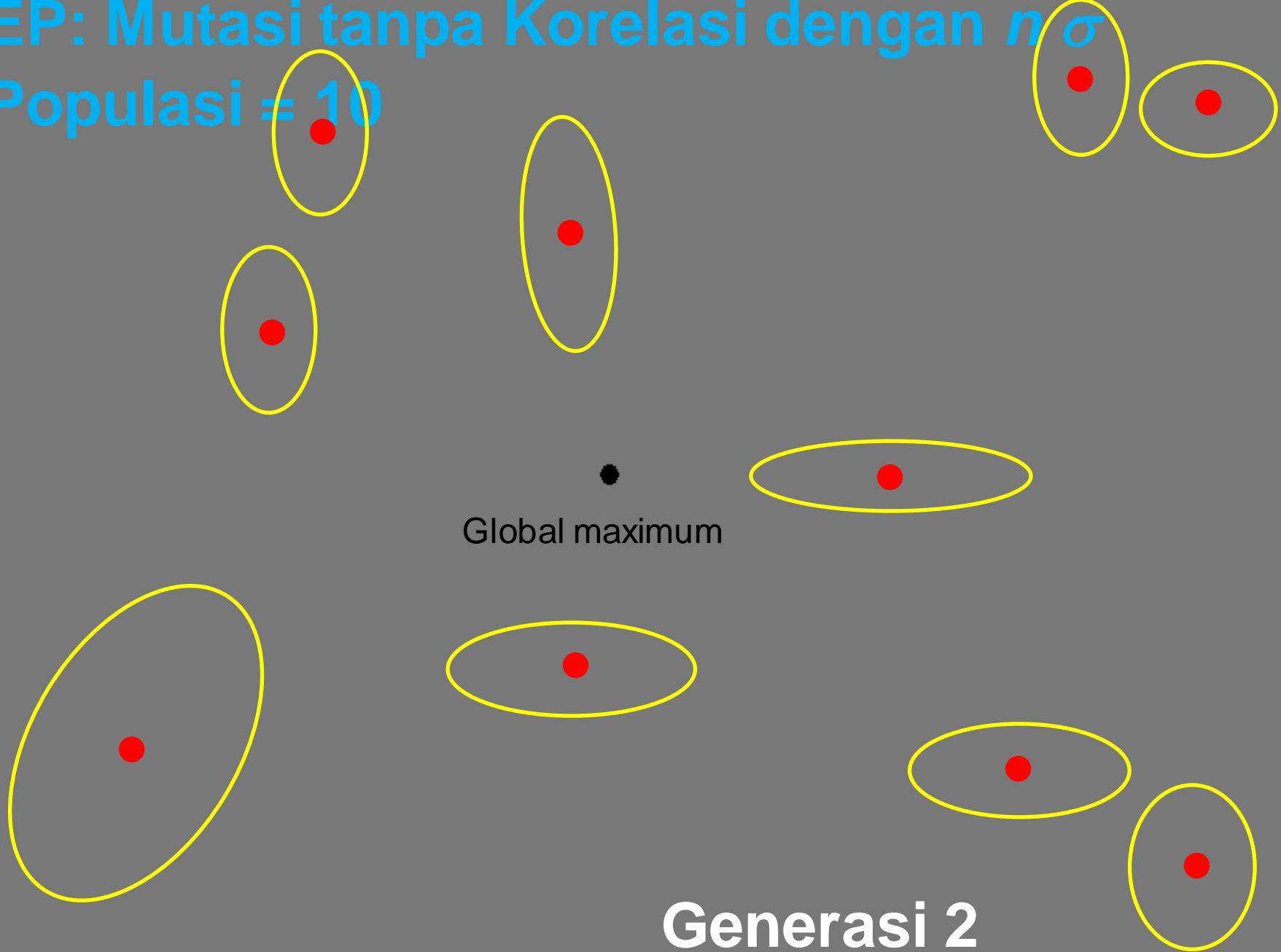
EP: Mutasi tanpa Korelasi dengan $n\sigma$

Populasi = 10



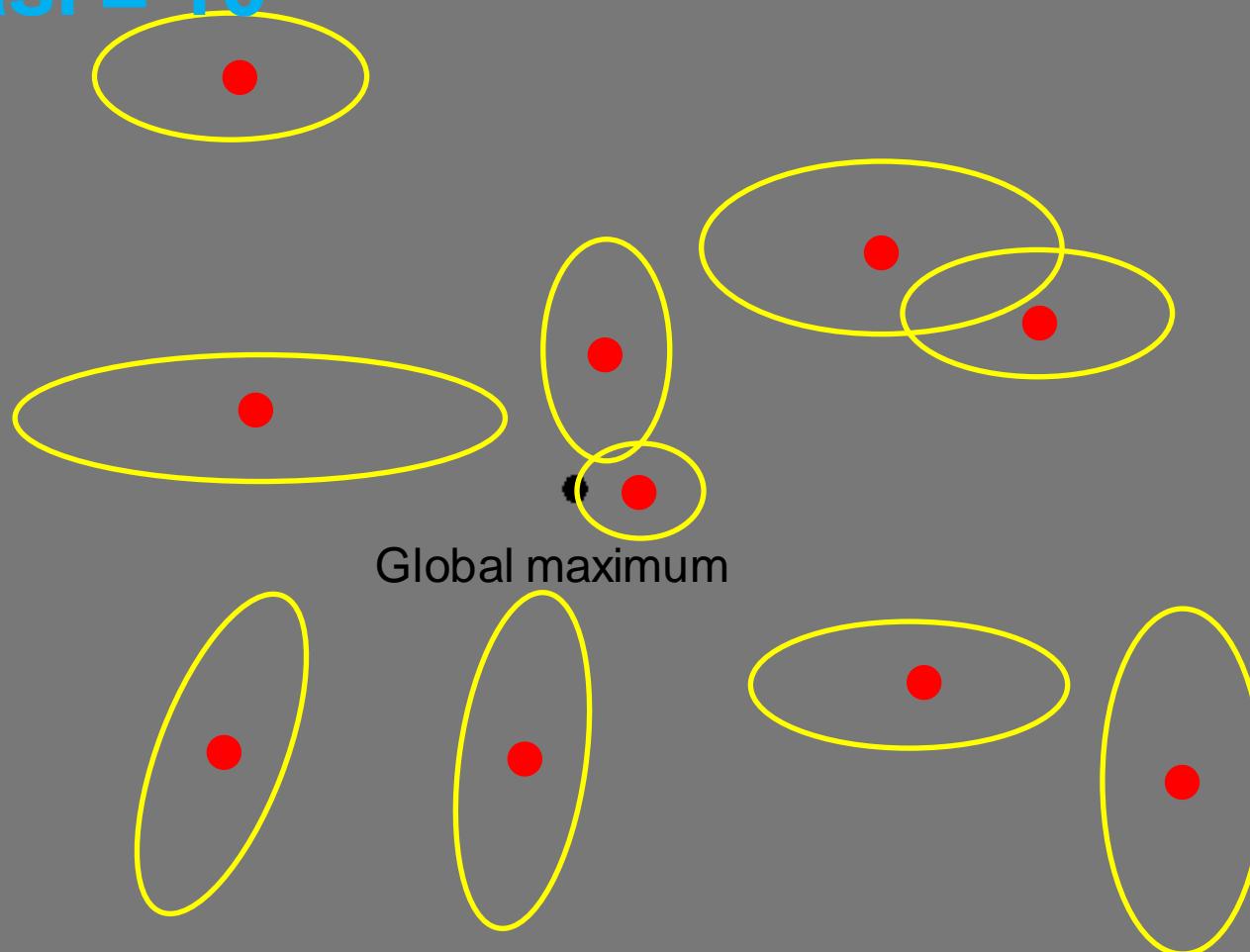
EP: Mutasi tanpa Korelasi dengan $n\sigma$

Populasi = 10



EP: Mutasi tanpa Korelasi dengan $n \sigma$

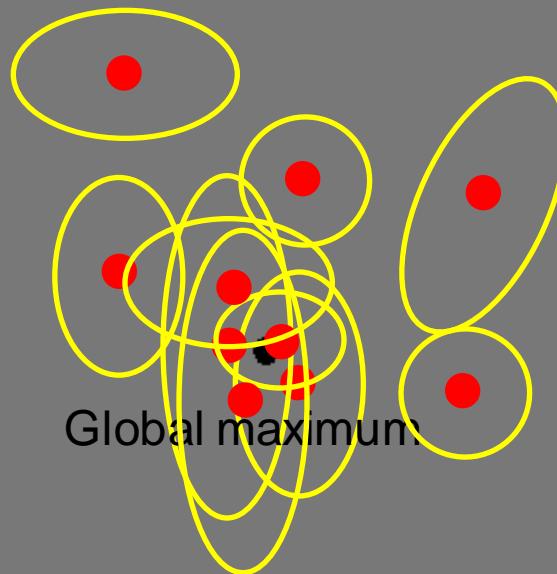
Populasi = 10



Generasi 10

EP: Mutasi tanpa Korelasi dengan $n \sigma$

Populasi = 10



Generasi 50

EP: Mutasi tanpa Korelasi dengan $n \sigma$

Populasi = 10



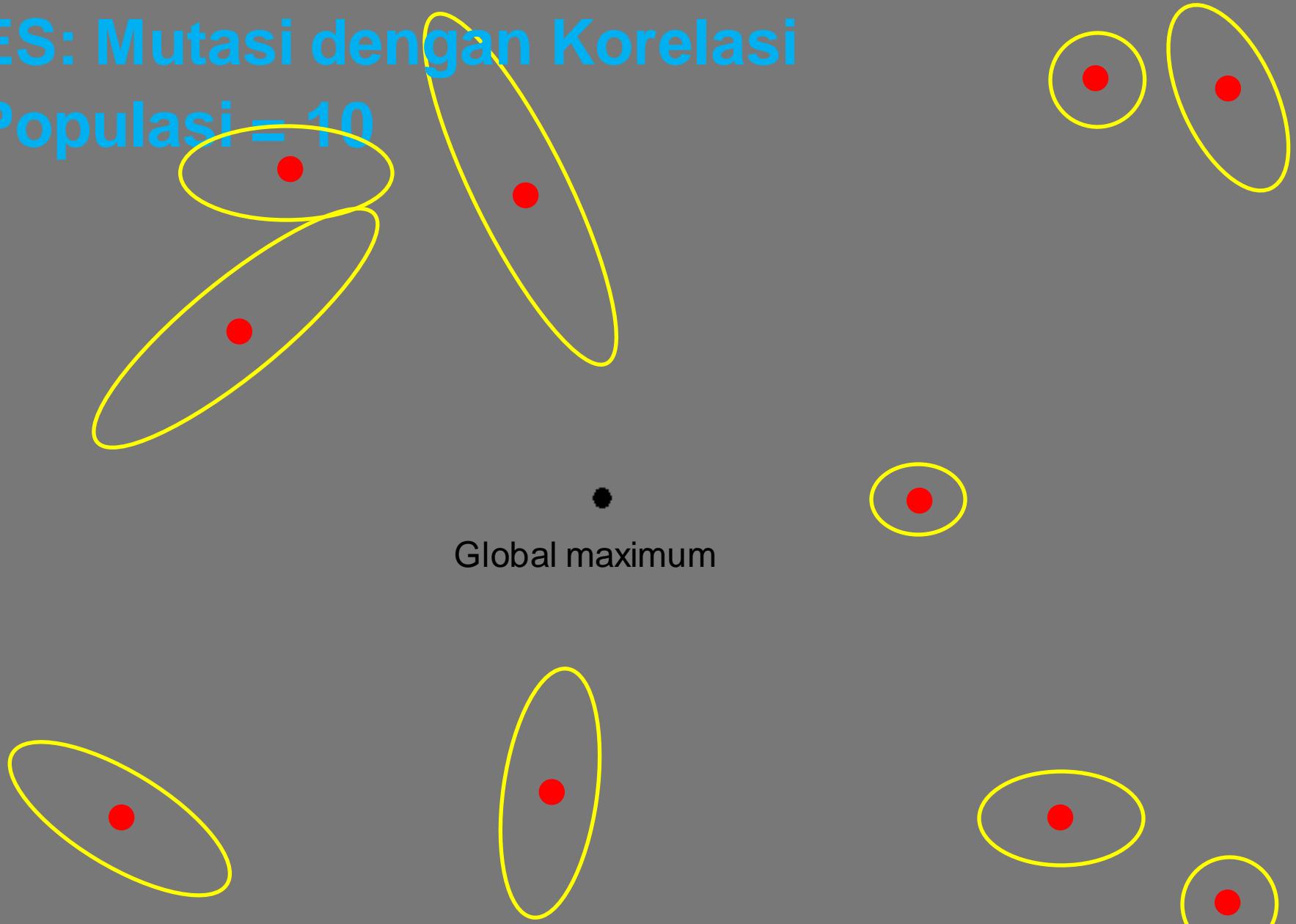
Generasi 100

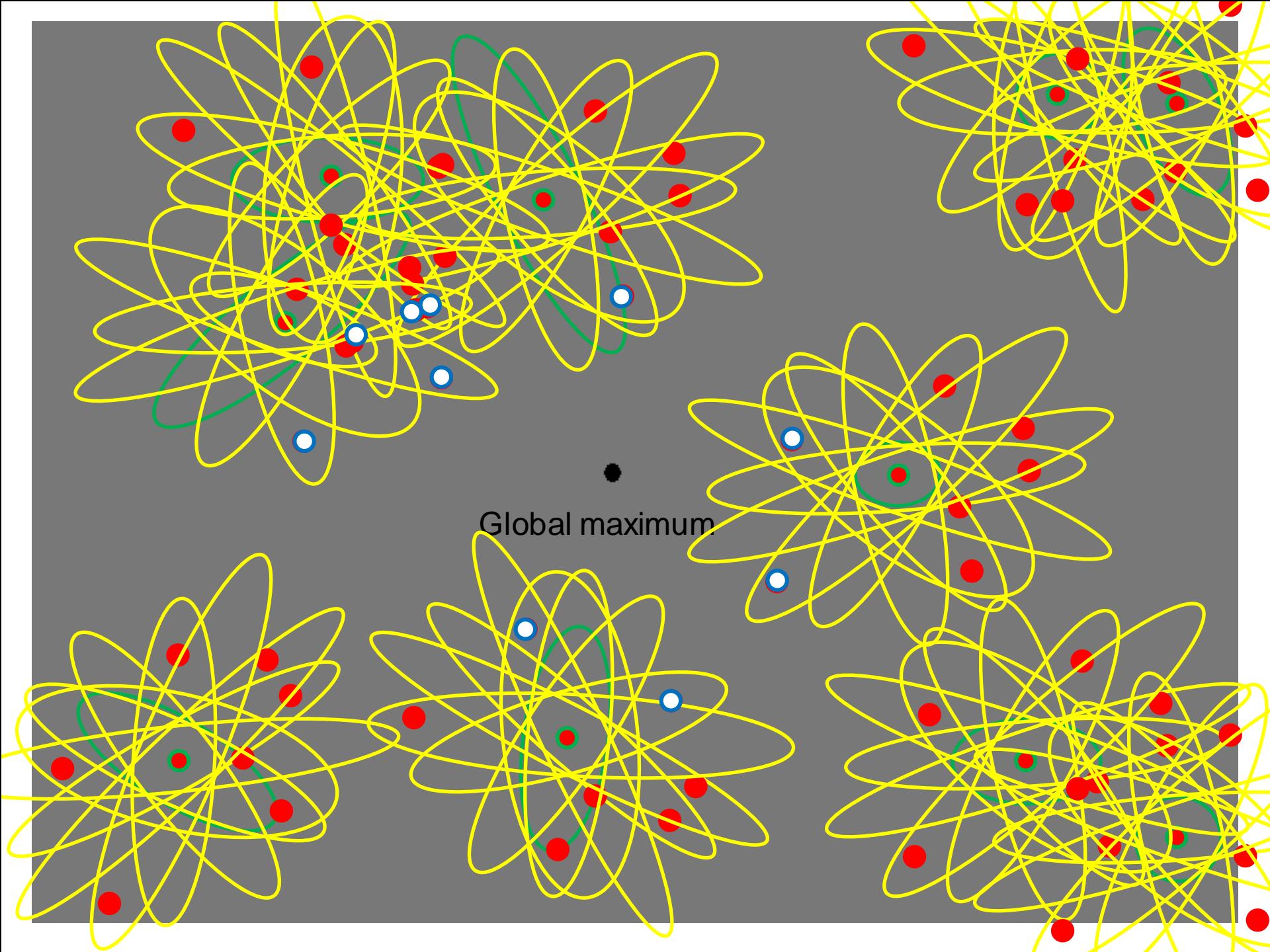
ES: Mutasi dengan Korelasi

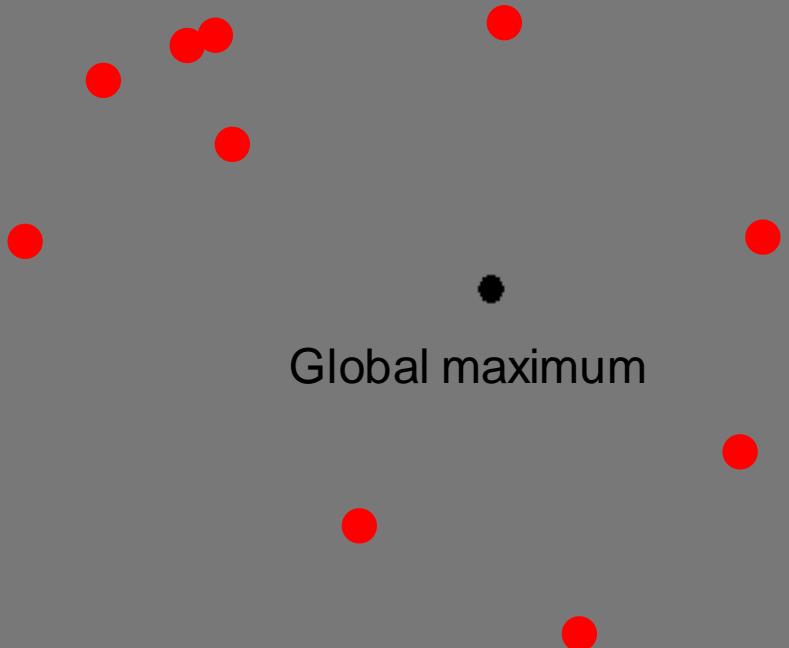
Populasi = 10

Global maximum

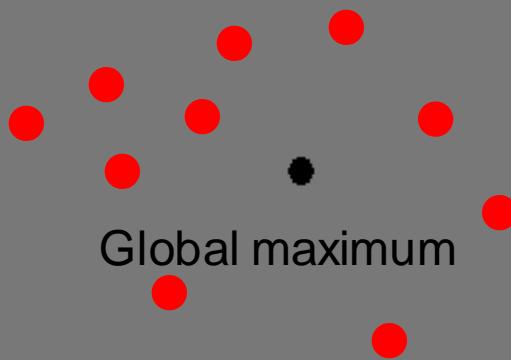
Generasi 1



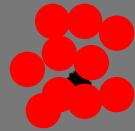




Generasi 2



Generasi 10



Global maximum

Generasi 20

Skema DE1

Untuk meningkatkan keberagaman vektor-vektor parameter, maka vektor \underline{v} direkombinasi dengan suatu vektor sembarang dalam populasi, misal $x_{i,G}$. Proses *crossover* ini menghasilkan vektor \underline{u} berikut ini

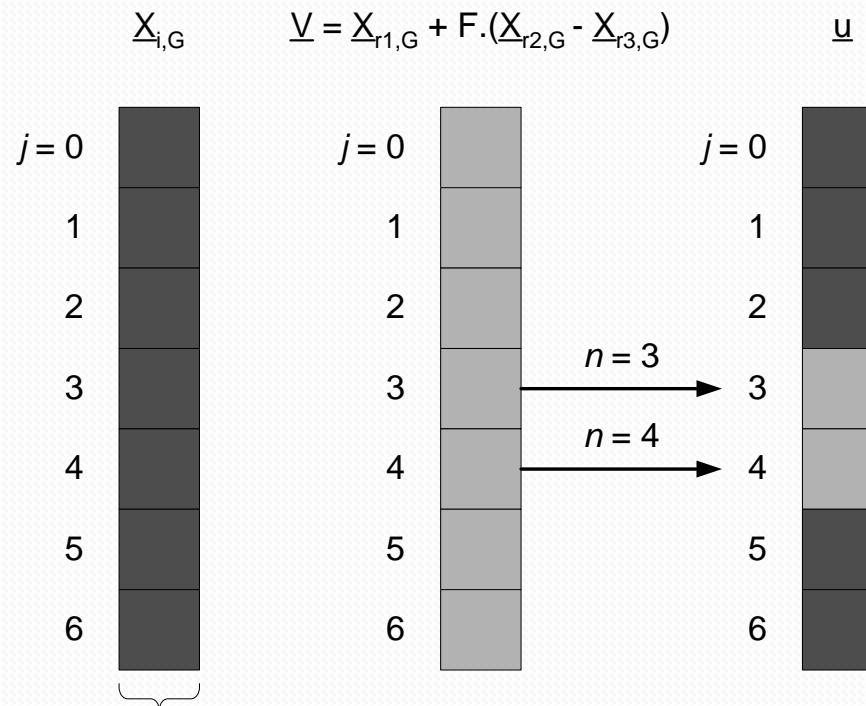
$$\underline{u} = (u_1, u_2, \dots, u_D)^T$$

dengan

$$u_j = \begin{cases} v_j & \text{untuk } j = \langle n \rangle_D, \langle n+1 \rangle_D, \dots, \langle n+L-1 \rangle_D \\ (x_{i,G})_j & \text{untuk } j \text{ yang lain} \end{cases}$$

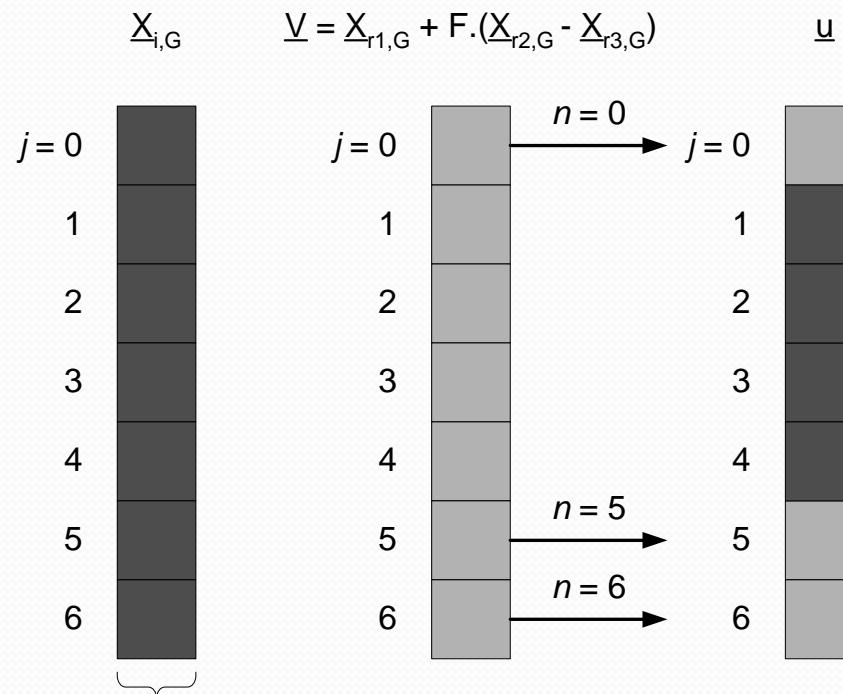
dimana simbol $\langle \rangle_D$ menyatakan fungsi modulo dengan modulus D .

Rekombinasi ($D = 7$, $n = 3$, dan $L = 2$)



Vektor parameter,
berisi parameter x_j ,
 $j = 0, 1, \dots, D-1$

Rekombinasi ($D = 7$, $n = 5$, dan $L = 3$)



Vektor parameter,
berisi parameter x_j ,
 $j = 0, 1, \dots, D-1$

Variasi-variasi DE

- Mutasi
 - Gunakan individu terbaik
 - Gunakan lebih banyak individu
- Crossover
 - Skema yang mirip *uniform crossover*
 - Banyak variasi yang bisa digunakan

Setting Parameter

- Ukuran Populasi?
 - 5 kali dimensi
 - 10 kali dimensi
- Konstanta F?
 - $0,2 < F \leq 1$
 - $0 < F \leq 2$
- CR?
 - $0,9 < CR \leq 1$

Performansi

- Pada banyak kasus, DE biasanya dapat menemukan minimum global.
- DE biasanya konvergen (menemukan minimum global) lebih cepat dibandingkan algoritma lain seperti *Annealed version of the Nelder&Mead strategy* (ANM) dan *Adaptive Simulated Annealing* (ASA), khususnya untuk fungsi-fungsi yang sulit diminimasi.

Performansi

- Berdasarkan penelitian Dervi§ Karaboga dan S. Ökdem, DE memiliki tiga kelebihan, yaitu:
 - biasanya menemukan minimum global tanpa terpengaruh oleh nilai-nilai parameter awal;
 - cepat konvergen (memerlukan sedikit generasi atau evaluasi fungsi untuk menemukan minimum global); dan
 - sangat mudah digunakan karena hanya terdapat satu parameter yang sensitif, yaitu faktor penskalaan F .

Kesimpulan

- DE menggunakan proses mutasi **semi deterministik**
- DE memiliki performansi yang sangat baik dibandingkan berbagai algoritma optimasi lainnya, terutama untuk permasalahan yang sulit diminimasi

Daftar Pustaka

- [SUYo8] Suyanto, 2008, Evolutionary Computation: Komputasi Berbasis “Evolusi” dan “Genetika”, penerbit Informatika Bandung.
- [STO95a] Storn, Rainer and Price, Kenneth (1995). Differential Evolution - a Simple and Efficient Adaptive Scheme for Global Optimization over Continuous Spaces. Technical Report TR-95-012, ICSI, March 1995.
- [WIKo7] Wikipedia, the free encyclopedia, 2007, “Differential Evolution”. Di-download pada bulan Desember 2007.

Grammatical Evolution (GE)

Dr. Suyanto, S.T., M.Sc.
HP/WA: 0812 845 12345

Intelligence Computing Multimedia (ICM)
Informatics faculty – Telkom University

Intro

- GE merupakan pengembangan dari *Genetic Programming* (GP).
- Perbedaan sangat signifikan di antara keduanya terletak pada **representasi individunya**.
- GE menggunakan representasi individu yang bisa digunakan untuk meng-“evolusi” program yang bebas bahasa. Sedangkan GP menggunakan representasi individu yang khusus untuk bahasa pemrograman LISP (*List Programming*).

Backus Naur Form (BNF)

- Adalah notasi untuk mengekspresikan *grammar* suatu bahasa dalam bentuk *production rules*.
- Tata bahasa (*grammar*) pada BNF terdiri dari:
 - terminal-terminal yang merupakan item-item yang dapat muncul dalam bahasa tersebut, yaitu: +, -, dan sebagainya;
 - non-terminal yang dapat dikembangkan (diperluas) ke dalam satu atau lebih terminal dan non-terminal.

Backus Naur Form (BNF)

- Grammar \rightarrow tuple $\{N, T, P, S\}$
- N adalah himpunan **non-terminal**;
- T adalah himpunan **terminal**;
- P adalah himpunan ***production rules*** yang memetakan elemen-elemen dalam N menjadi T ;
- S adalah simbol mulai (***start symbol***) yang berupa satu simbol non-terminal (anggota N).

N = {expr, op, pre_op}

T = {Sin, Cos, Tan, Log, +, -, /, *, X, ()}

S = <expr>

P dapat direpresentasikan sebagai:

(1) <expr> ::= <expr> <op> <expr> (A)

| (<expr> <op> <expr>) (B)

| <pre_op> (<expr>) (C)

| <var> (D)

(2) <op> ::= - (A)

| + (B)

| / (C)

| * (D)

(3) <pre_op> ::= Sin (A)

| Cos (B)

| Tan (C)

| Log (D)

(4) <var> ::= X

Contoh Grammar BNF

Contoh Grammar BNF

- Untuk melengkapi definisi BNF untuk fungsi dalam bahasa C, kita perlu memasukkan aturan-aturan berikut ini ke dalam definisi BNF sebelumnya:

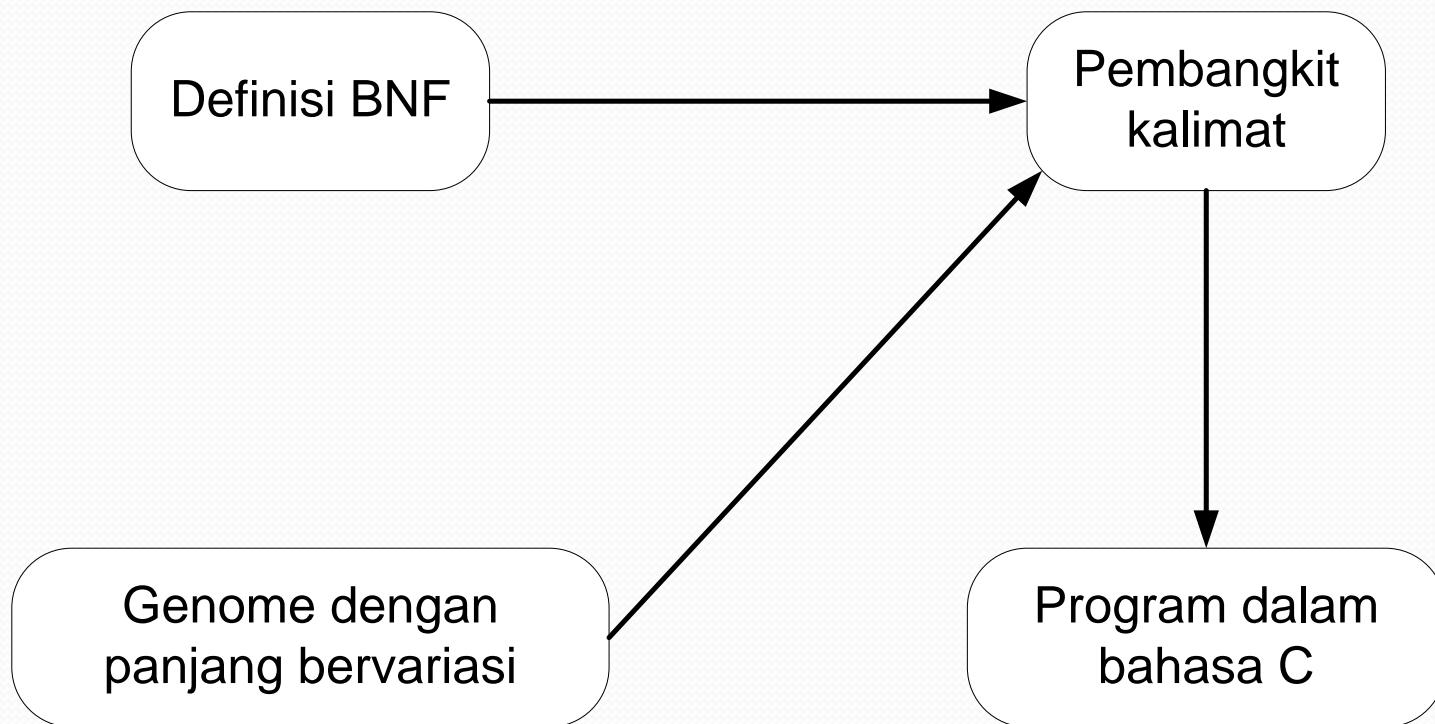
```
<func> ::= <header>
<header> ::= float symb(float X) {<body>}
<body> ::= <declarations> <code> <return>
<declarations ::= float a;
<code> ::= a = <expr>;
<return> ::= return(a);
```

Contoh Grammar BNF

Tetapi, fungsi di atas hanya terbatas untuk *code* yang hanya bisa membangkitkan satu baris tunggal. Kita bisa membangkitkan fungsi dengan panjang bervariasi dengan memodifikasi aturan *code* sehingga menjadi:

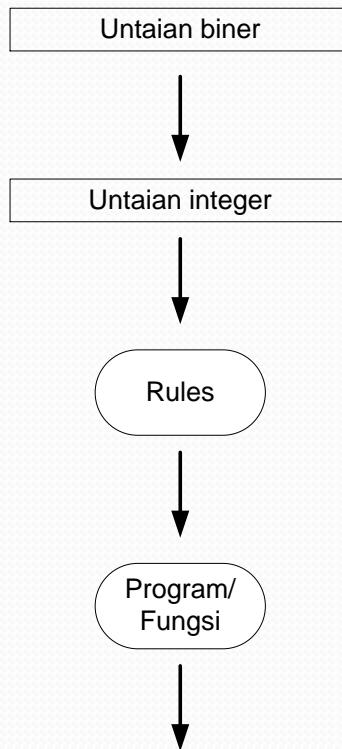
```
<code> ::= <line>; | <line>; <code>
<line> ::= <var> = <expr>
```

Sistem GE untuk bahasa C



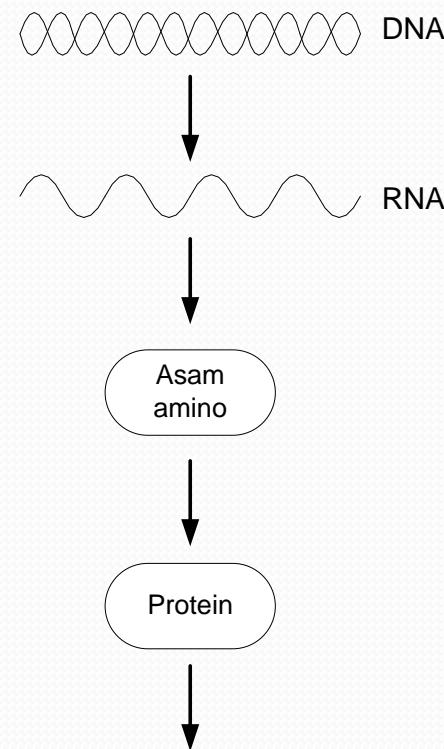
Representasi individu

Grammatical Evolution



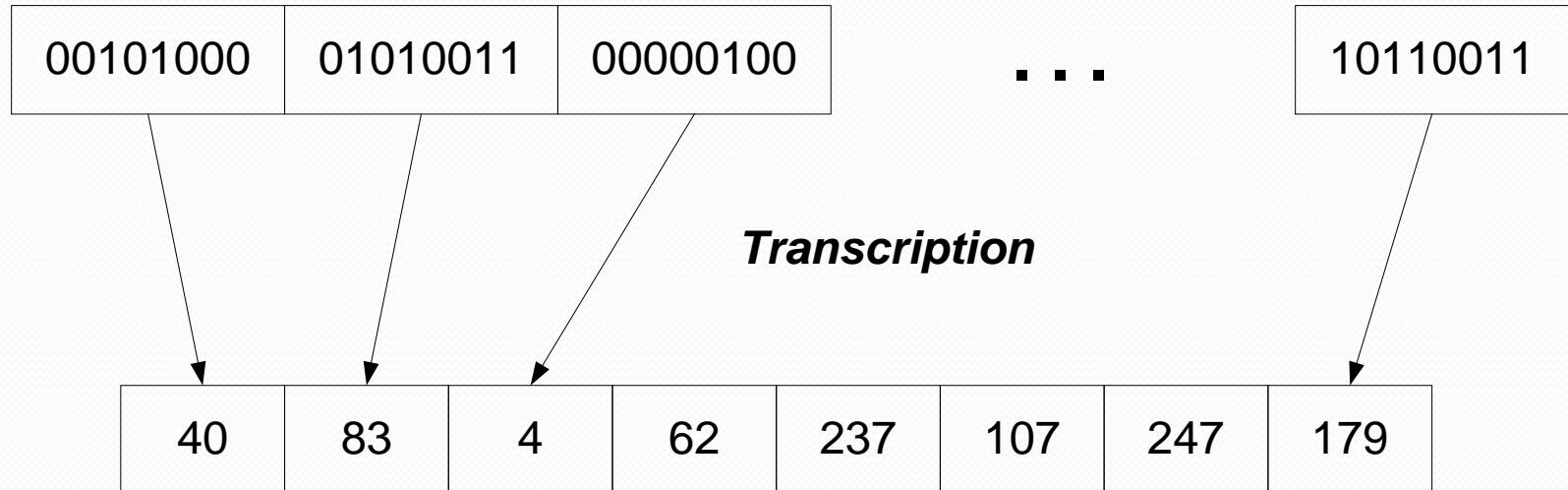
Transcription

Biologi di alam



Translation

Contoh individu GE (8 gen integer)



individu → program

Untuk menerjemahkan individu menjadi suatu program, setiap individu harus memiliki bentuk sebagai berikut:

```
float symb(float x)
{
    a = <expr>;
    return(a);
}
```

$$\{40, 83, 4, 62, 237, 107, 247, 179\} \rightarrow X - \cos(X)$$

Gen	Mod	Aturan terpilih	Ekspresi
		-	<expr>
40	0	1A	<expr> <op> <expr>
83	3	1D	<var> <op> <expr>
-	-	-	$X <\text{op}> <\text{expr}>$
4	0	2A	$X - <\text{expr}>$
62	2	1C	$X - <\text{pre_op}> (<\text{expr}>)$
237	1	3B	$X - \text{Cos } (<\text{expr}>)$
107	3	1D	$X - \text{Cos } (<\text{var}>)$
-	-	-	$X - \text{Cos } (X)$

N = {expr, op, pre_op}

T = {Sin, Cos, Tan, Log, +, -, /, *, X, ()}

S = <expr>

P dapat direpresentasikan sebagai:

(1) <expr> ::= <expr> <op> <expr> (A)

| (<expr> <op> <expr>) (B)

| <pre_op> (<expr>) (C)

| <var> (D)

(2) <op> ::= - (A)

| + (B)

| / (C)

| * (D)

(3) <pre_op> ::= Sin (A)

| Cos (B)

| Tan (C)

| Log (D)

(4) <var> ::= X

Contoh Grammar BNF

Operator evolusi

- Karena menggunakan representasi biner, maka GE menggunakan semua operator evolusi yang sama seperti yang ada pada GA: seleksi orangtua, rekombinasi, mutasi, dan seleksi survivor.
- Tetapi GE menggunakan dua operator tambahan, yaitu *Duplicate* dan *Prune*

Duplicate

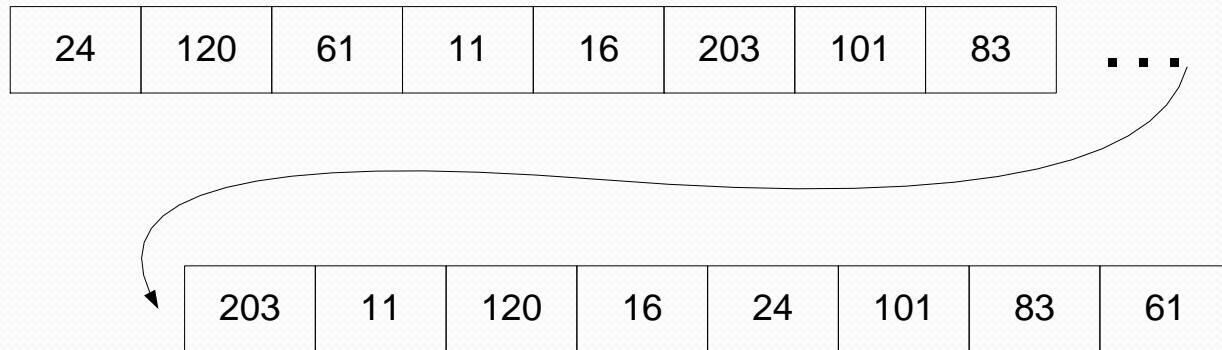
- Pada suatu kasus tertentu, mungkin saja dihasilkan suatu kromosom yang tidak valid yang jika ditranslasi akan menghasilkan suatu program yang tidak lengkap.

24	120	61	11	16	203	101	83
----	-----	----	----	----	-----	-----	----

- Kromosom di atas jika ditranslasi akan menjadi program yang tidak lengkap, yaitu:

$$(X + X) - <\text{var}> <\text{op}> <\text{expr}>$$

Duplicate



Kromosom 16 gen (*hasil duplicate*) di atas jika ditranslasi akan menghasilkan program yang lengkap, yaitu:

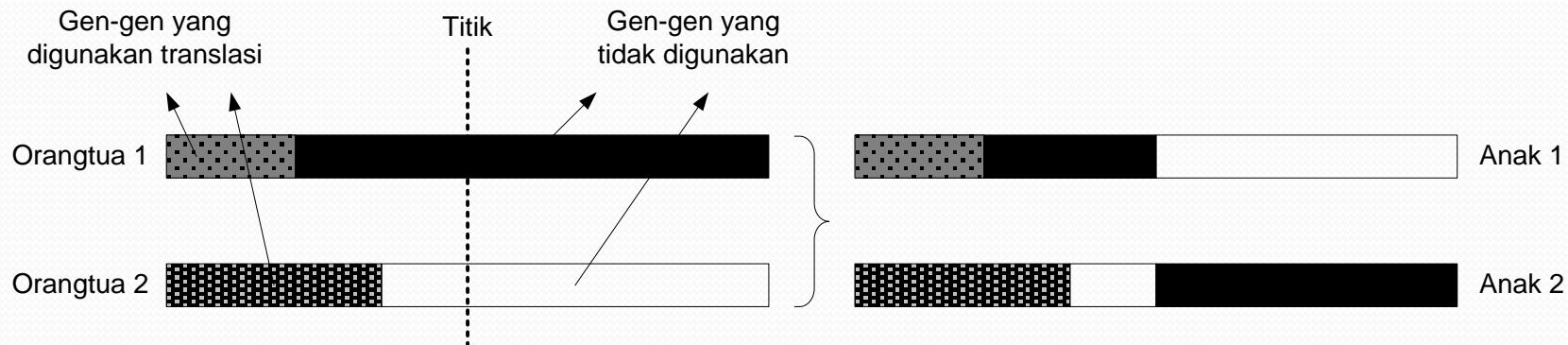
$$(X + X) - X * X$$

Duplicate

- Berapa banyak duplikasi gen yang sebaiknya dilakukan?
- Jawaban yang paling sederhana adalah sesuaikan saja dengan kebutuhan. JANGAN BERLEBIHAN !!!
- Kalau kebutuhan sudah terpenuhi, maka gen-gen yang tidak terpakai bisa dihapus (kromosomnya dipangkas sehingga lebih pendek).
- Pada kasus di atas, duplikasi dilakukan dua kali sehingga panjang kromosom menjadi 16 gen.
- Tetapi pada saat ditranslasi, ternyata hanya 10 gen yang digunakan dalam proses translasi. Gen yang tidak terpakai, 6 gen, bisa dihapus (*prune*).
- Mengapa harus dihapus? Supaya tidak mengganggu proses rekombinasi.

Prune

- Jika titik pindah silang yang dibangkitkan berada pada posisi gen-gen yang tidak digunakan, maka operator rekombinasi menjadi sia-sia.
- Mengapa? Karena kedua anak yang dihasilkan jika ditranslasi akan menghasilkan program yang sama dengan kedua orangtuanya.



Prune

- Bagaimana mengatasi masalah ini?
- Satu-satunya cara yang bisa digunakan adalah memangkas (menghapus) gen-gen yang tidak digunakan tersebut.
- Operator ini disebut *prune* (pemangkasan).
- Bagaimana caranya? Operator *Prune* dilakukan dengan probabilitas tertentu terhadap kromosom-kromosom yang tidak menggunakan semua gen-nya dalam proses translasi.
- Setiap gen yang tidak digunakan dalam proses translasi akan dihapus. Operator *prune* membuat rekombinasi menjadi lebih cepat dan lebih baik.

Performansi GE

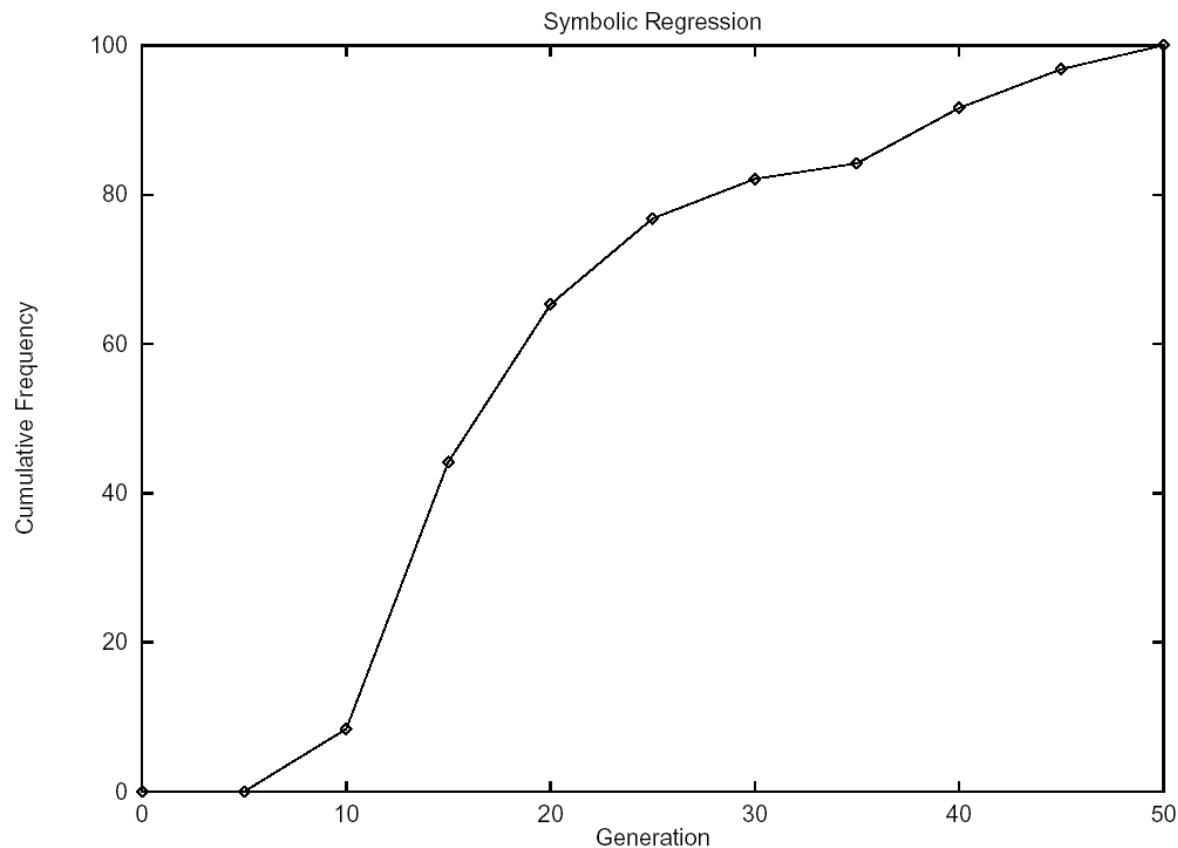
GE telah berhasil digunakan untuk menyelesaikan tiga masalah penting, yaitu:

- *Symbolic Regression;*
- *Trigonometric Identities;*
- *Symbolic Integration.*

GE untuk *Symbolic Regression*

- Pada [RYA98a], dinyatakan bahwa untuk masalah *Symbolic Regression*, GE dengan seleksi *steady state* bisa menemukan fungsi target $X + X^2 + X^3 + X^4$ secara cepat.
- Dari seratus kali running, prosentase GE dalam menemukan solusi pada generasi ≤ 50 bisa mencapai 95%. Bahkan GE pernah menemukan solusi pada generasi ke-10.

GE untuk *Symbolic Regression*



GE untuk *Trigonometric Identities*

- Untuk masalah *Trigonometric Identities*, GE diuji untuk menemukan fungsi yang merupakan *trigonometric identity* dari $\cos 2x$. Fungsi yang dicari tersebut adalah $1 - 2\sin^2 x$.
- Agar pencarian tidak selalu menemukan fungsi $\cos 2x$, maka \cos dikeluarkan dari *production rules P*.
- Dari eksperimen yang dilakukan oleh Conor Ryan and Michael O'Neill, GE dengan seleksi *steady state* bisa menemukan fungsi yang dicari, $1 - 2\sin^2 x$, secara cepat.
- Dari seratus kali running, prosentase GE dalam menemukan solusi pada generasi ≤ 50 bisa mencapai 87%.
- Bahkan GE pernah menemukan solusi pada generasi ke-5. Berikut hasil eksperimen selengkapnya yang dilakukan oleh Conor Ryan and Michael O'Neill [RYA98a].

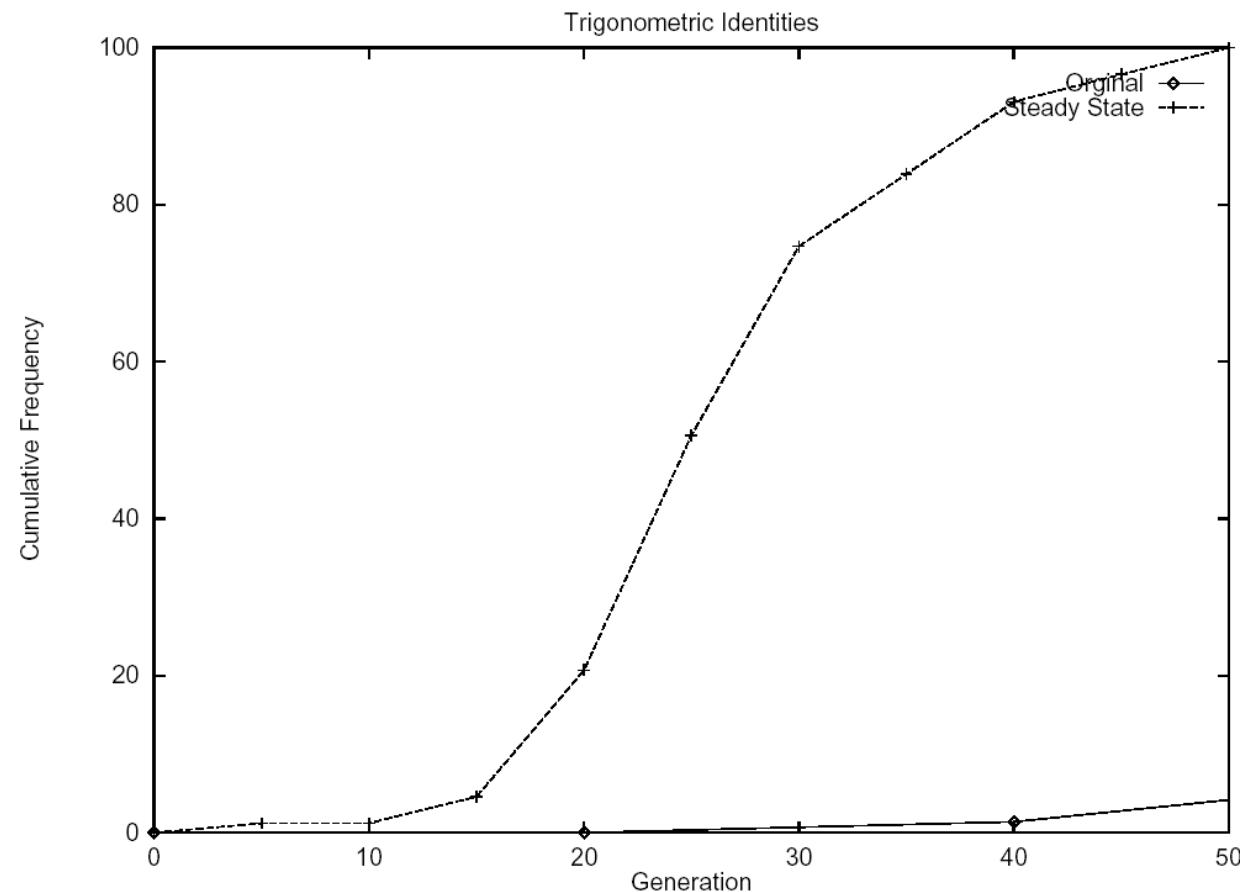
Data untuk Hitung Fitness

No.	x	Cos 2x	Fungsi baru	Error
1	0	1,00000	0,99998	0,00002
2	0,5	0,99996	0,99995	0,00001
...				
100	1	0,93969	0,93971	0,00002

$$\text{Fitness} = 1 / (\text{MAE} + a)$$

MAE = Mean Absolute Error = $\text{Mean}(|\text{Error}|)$
a = bilangan kecil, misal 0,000001

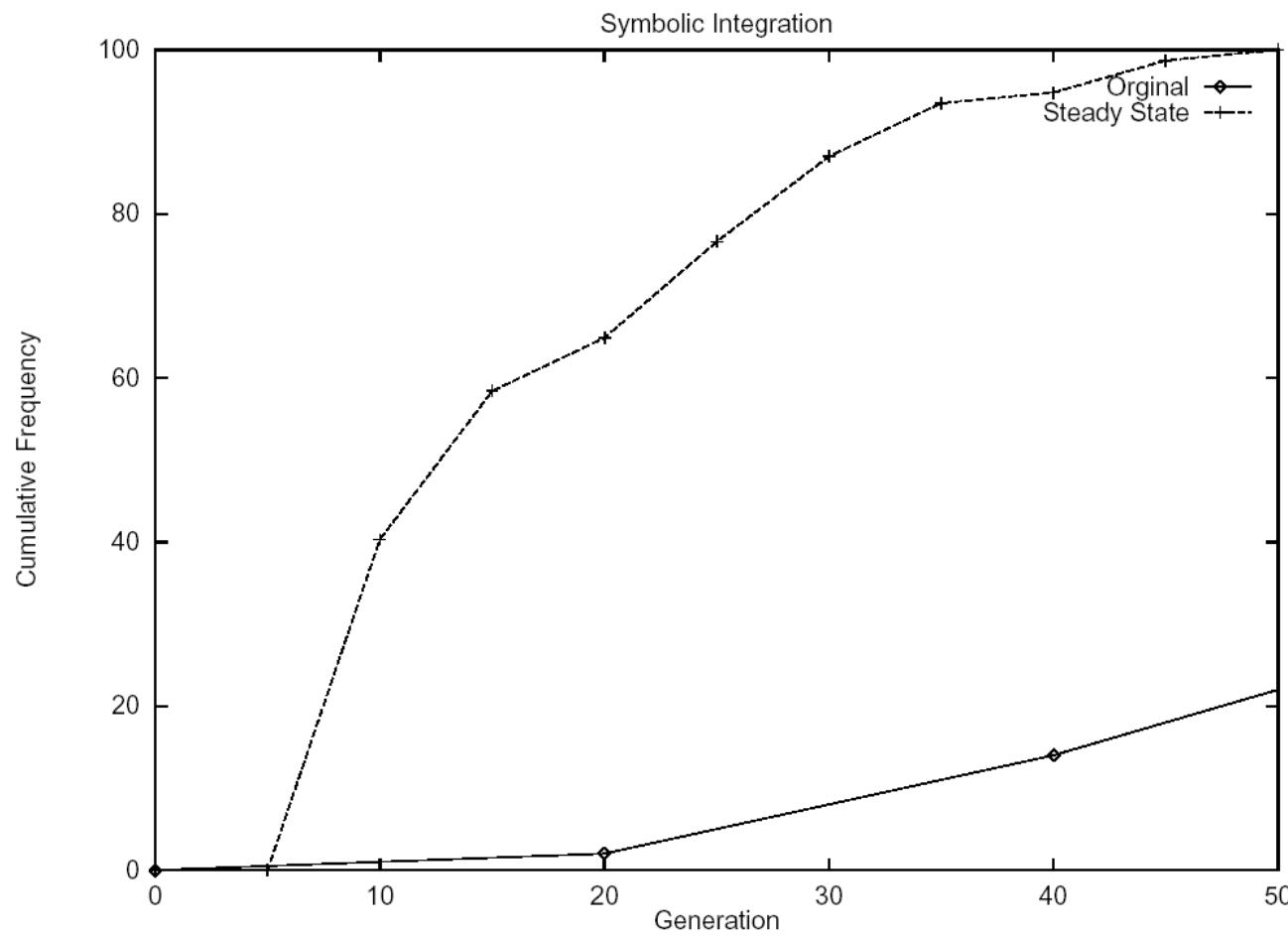
GE untuk *Trigonometric Identities*



GE untuk *Symbolic Integration*

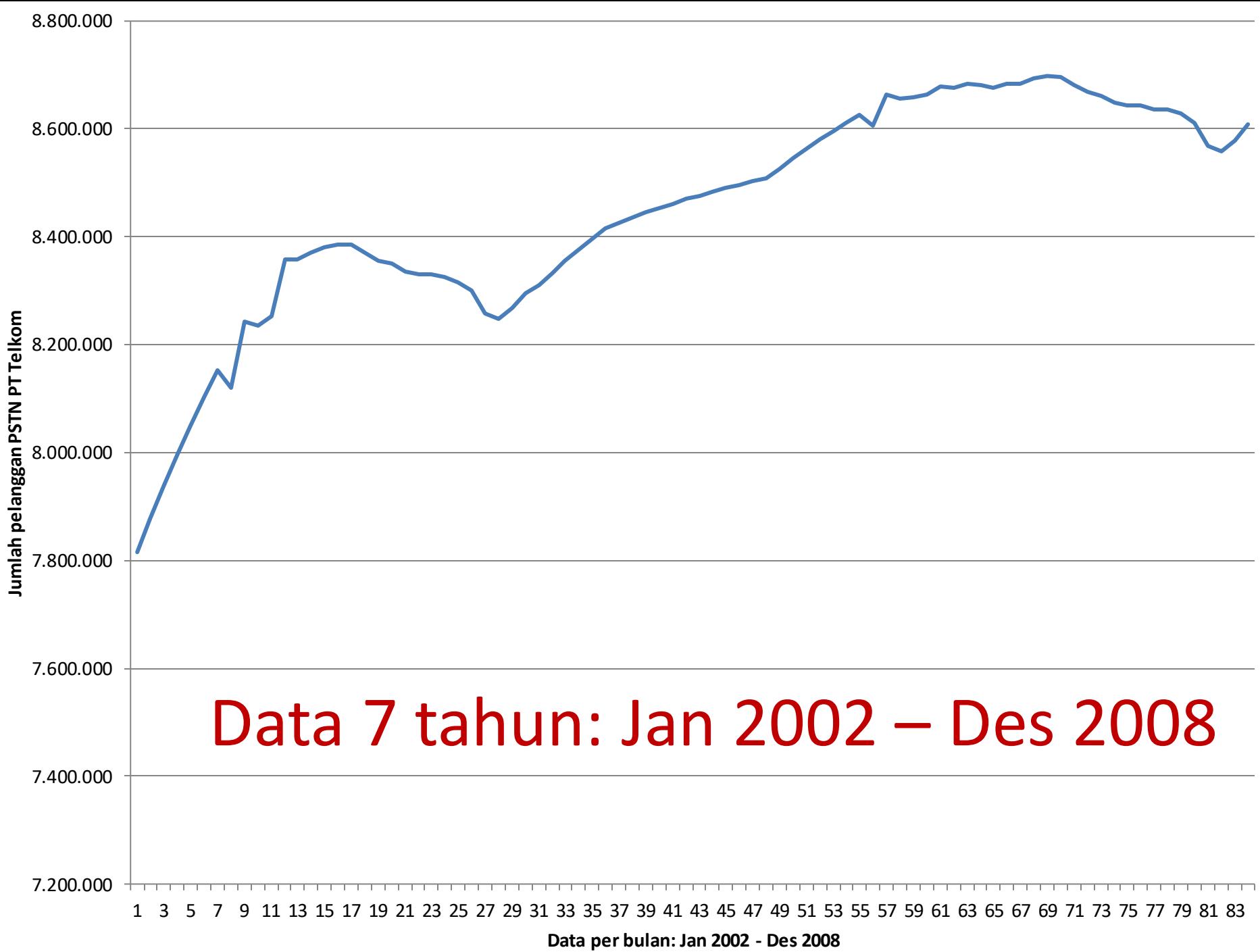
- Untuk masalah *Symbolic Integration*, GE diuji untuk menemukan suatu fungsi yang merupakan integral dari kurva yang diberikan.
- Sistem diberi sekumpulan pasangan *input* dan *output* dan harus menemukan suatu fungsi yang bisa memetakan input ke output secara benar.
- Fungsi yang diuji adalah $\cos(X) + 2X + 1$.
- Dari eksperimen yang dilakukan oleh Conor Ryan and Michael O'Neill, GE dengan seleksi *steady state* bisa menemukan fungsi yang dicari, $\sin(X) + X + X^2$, secara cepat.
- Dari seratus kali running, prosentase GE dalam menemukan solusi pada generasi ≤ 50 bisa mencapai 87%. GE pernah menemukan solusi secara cepat pada generasi ke-10. Berikut hasil eksperimen selengkapnya yang dilakukan oleh Conor Ryan and Michael O'Neill [RYA98a].

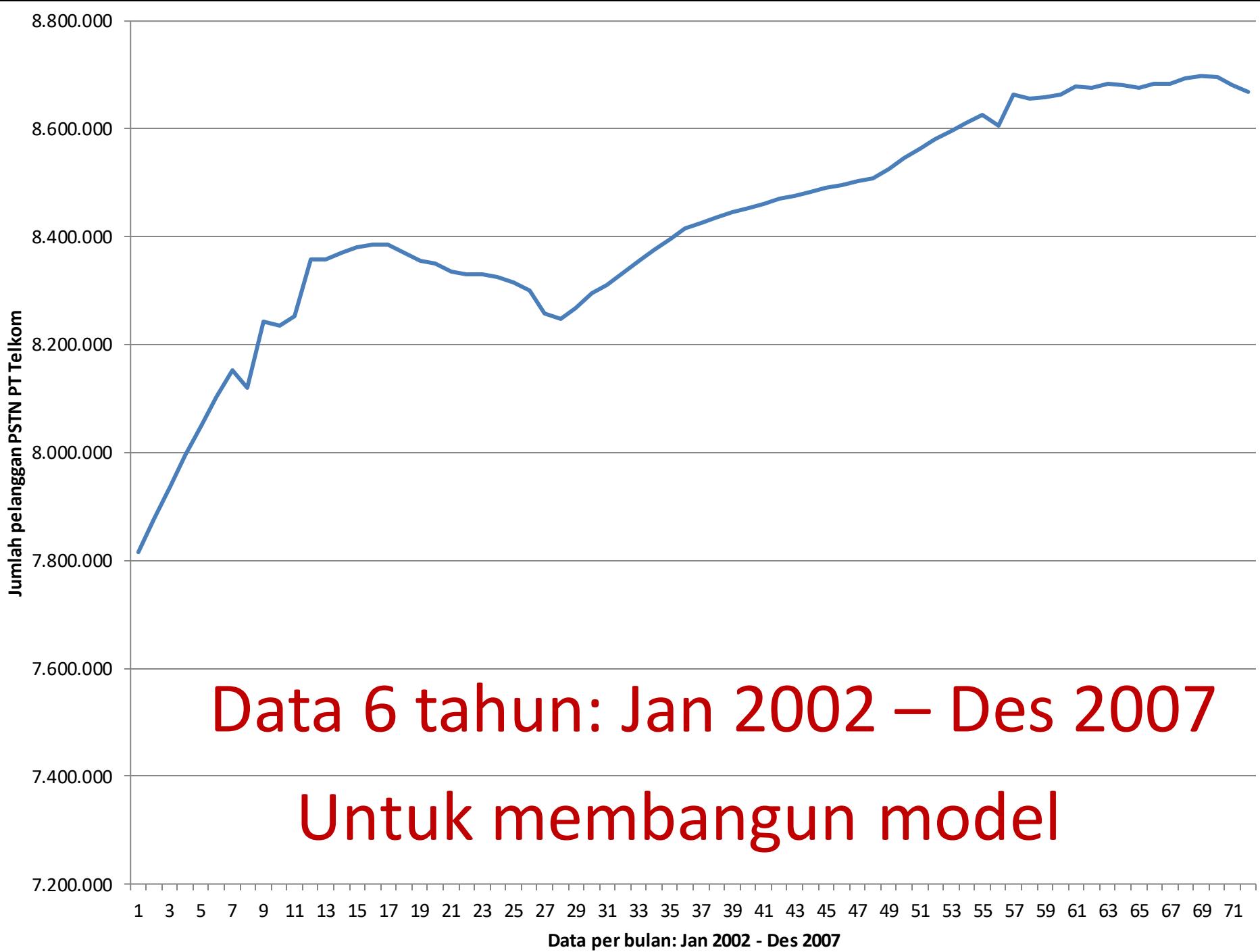
GE untuk *Symbolic Integration*

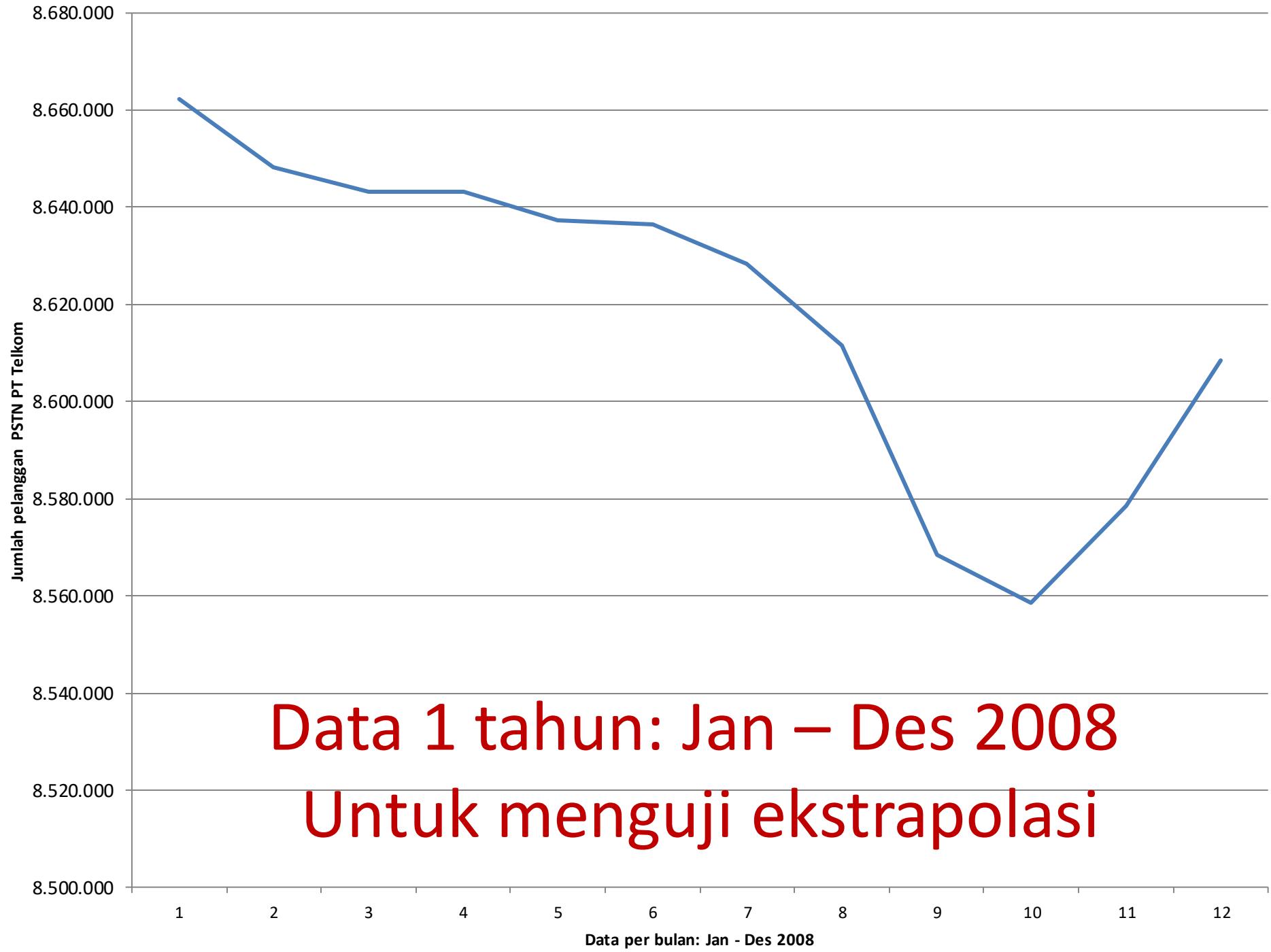


Studi Kasus

- Prediksi Data Time series Jumlah Pelanggan PSTN di PT Telkom
- Tugas Akhir mhs IF-2005: Dwi Tuti Supantari
- PSTN vs. Wireless (GSM & CDMA)
- Investasi PSTN sangat besar → belum BEP
- Kalah bersaing dengan GSM & CDMA
- Prediksi data PSTN → kebijakan yang tepat



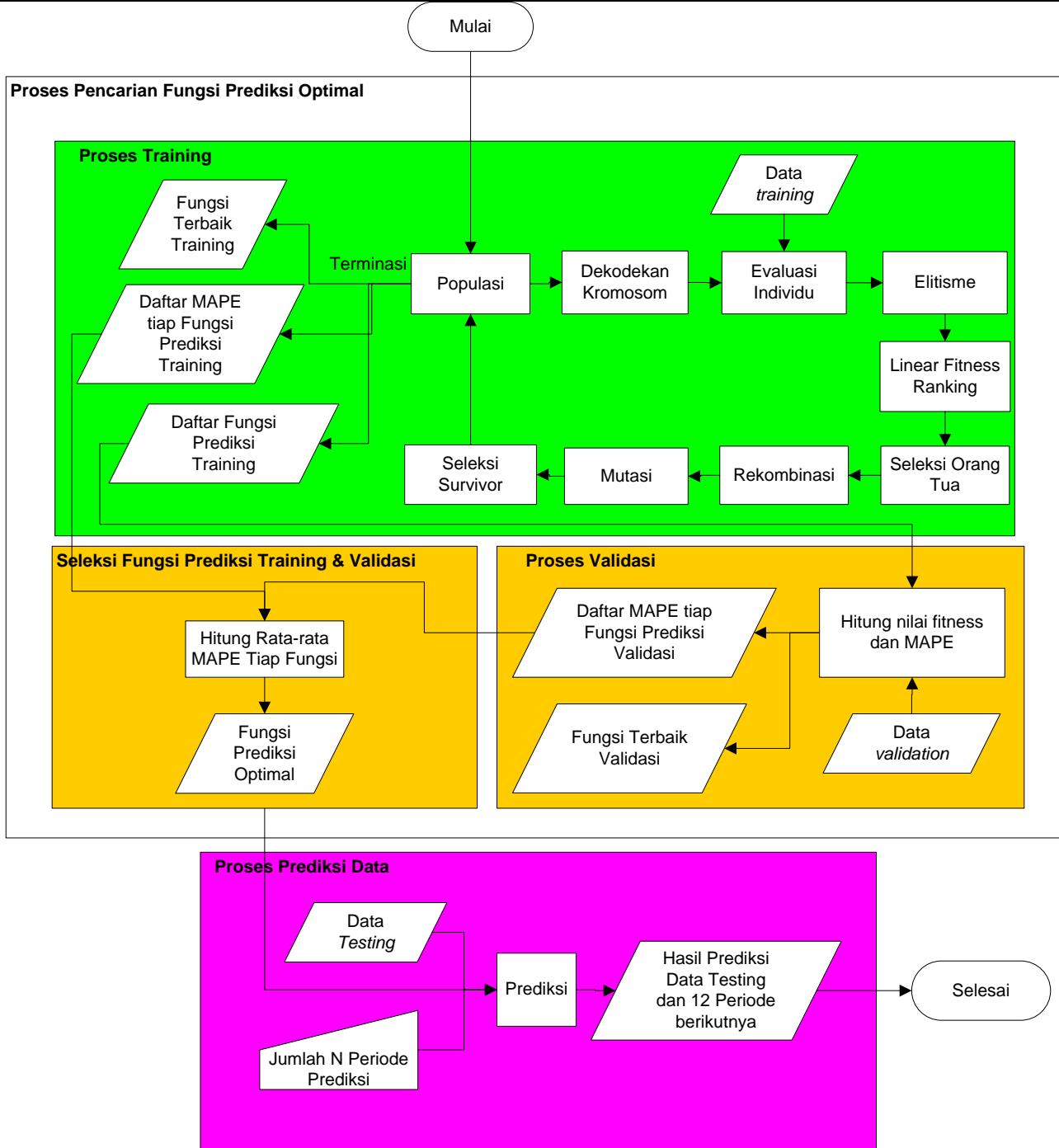




Data 1 tahun: Jan – Des 2008
Untuk menguji ekstrapolasi

Pembangunan Model

- **Skenario 1**
 - Data *training* : 24 bulan (2002 - 2003)
 - Data *validation* : 24 bulan (2004 - 2005)
 - Data *testing* : 24 bulan (2006 - 2007)
- **Skenario 2**
 - Data *training* : 12 bulan (2002)
 - Data *validation* : 12 bulan (2003)
 - Data *testing* : 48 bulan (2004 - 2007)
- **Skenario 3**
 - Data *training* : 48 bulan (2002 - 2005)
 - Data *validation* : 12 bulan (2006)
 - Data *testing* : 12 bulan (2007)



Normalisasi

$$Xn_i = \left(\frac{(X_i - \min(X))}{(\max(X) - \min(X))} \times 0,8 \right) + 0,1$$

- Xn_i = data aktual normalisasi ke-i
- X_i = data aktual dengan range data asli ke-i
- X = data aktual dengan range data asli

Denormalisasi

$$F_i = \left[\frac{((F'_i) - 0,1)}{0,8} \right] \times (\max(X) - \min(X)) + \min(X)$$

- F_i = nilai prediksi dengan range nilai asli
- F'_i = nilai prediksi dari hasil data yang dinormalisasi
- X = data aktual

BNF 1

N = {expr, op}

T = {+, -, 0.1, ..., 0.9, x1, x2, x3, x4, x5, e1, e2, e3, e4, e5, (,)}

S = <expr>

P dapat direpresentasikan sebagai :

1) <expr> := <expr> <op> <expr> (A)

(<const> * <var>) (B)

2) <op> := + (A)

- (B)

3) <const> := 0.1 (A) 0.6 (F)

0.2 (B) 0.7 (G)

0.3 (C) 0.8 (H)

e1 = selisih antara data sebenarnya dengan
data prediksi pada periode sebelumnya.

4) <var> := x1 (A) e1 (F)

x2 (B) e2 (G)

x3 (C) e3 (H)

x4 (D) e4 (I)

x5 (E) e5 (J)

BNF 2

N = {expr, op, pre_op}

T = {sin, cos, exp, +, -, *, /, ^, 0.1, ..., 0.9, 2, 3, x1, x2, x3, x4, x5, e1, e2, e3, e4, e5, (,) }

S = <expr>

P dapat direpresentasikan sebagai :

1) <expr>	:=	<expr><op><expr>	(A)	2) <op>	:=	+	(A)
		(<expr><op><expr>)	(B)			-	(B)
		<pre_op>(<expr>)	(C)			*	(C)
		<var>	(D)				
		<const>	(E)				

3) <pre_op>	:=	sin	(A)
		cos	(B)
		exp	(C)

4) <const>	:=	0.1	(A)	0.7	(G)
		0.2	(B)	0.8	(H)
		0.3	(C)	0.9	(I)

e1 = selisih antara data sebenarnya dengan data prediksi pada periode sebelumnya.

5) <var>	:=	x1	(A)	e1	(F)
		x2	(B)	e2	(G)
		x3	(C)	e3	(H)
		x4	(D)	e4	(I)
		x5	(E)	e5	(J)

Kombinasi Parameter

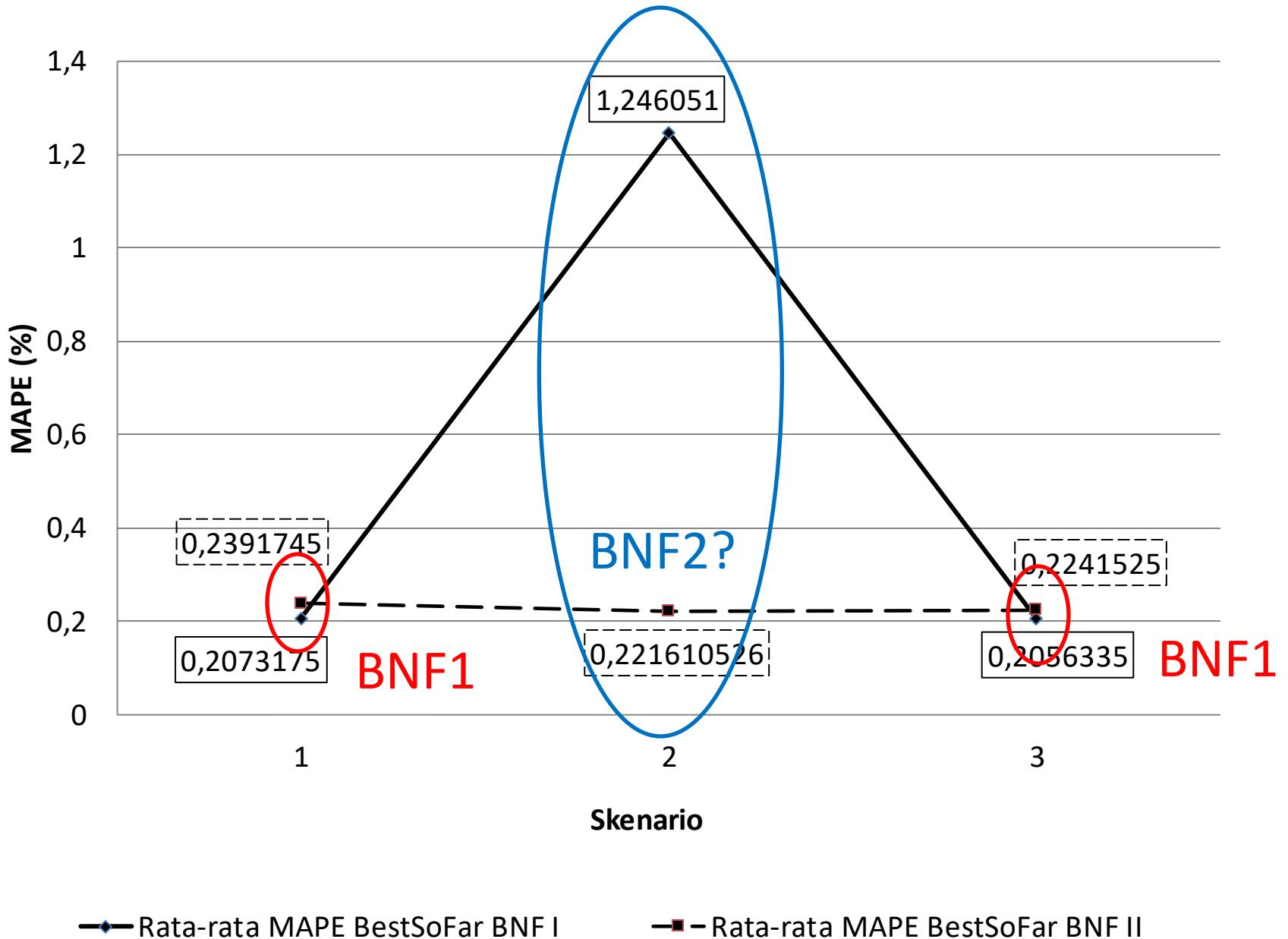
No Kombinasi Parameter	Ukuran Populasi	Generasi	Probabilitas crossover
1	100	1000	0.9
2	100	1000	0.7
3	50	2000	0.9
4	50	2000	0.7

Pengujian BNF 1

BNF	Skenario	No Kombinasi Parameter	Ukuran Populasi	Generasi	Probabilitas crossover
1	1	1	100	1000	0.9
		2	100	1000	0.7
		3	50	2000	0.9
		4	50	2000	0.7
	2	1	100	1000	0.9
		2	100	1000	0.7
		3	50	2000	0.9
		4	50	2000	0.7
	3	1	100	1000	0.9
		2	100	1000	0.7
		3	50	2000	0.9
		4	50	2000	0.7

Pengujian BNF 2

BNF	Skenario	No Kombinasi Parameter	Ukuran Populasi	Generasi	Probabilitas crossover
2	1	1	100	1000	0.9
		2	100	1000	0.7
		3	50	2000	0.9
		4	50	2000	0.7
	2	1	100	1000	0.9
		2	100	1000	0.7
		3	50	2000	0.9
		4	50	2000	0.7
	3	1	100	1000	0.9
		2	100	1000	0.7
		3	50	2000	0.9
		4	50	2000	0.7



Pembangunan Model

- **Skenario 1**

- Data *training* : 24 bulan (2002 - 2003)
- Data *validation* : 24 bulan (2004 - 2005)
- Data *testing* : 24 bulan (2006 - 2007)

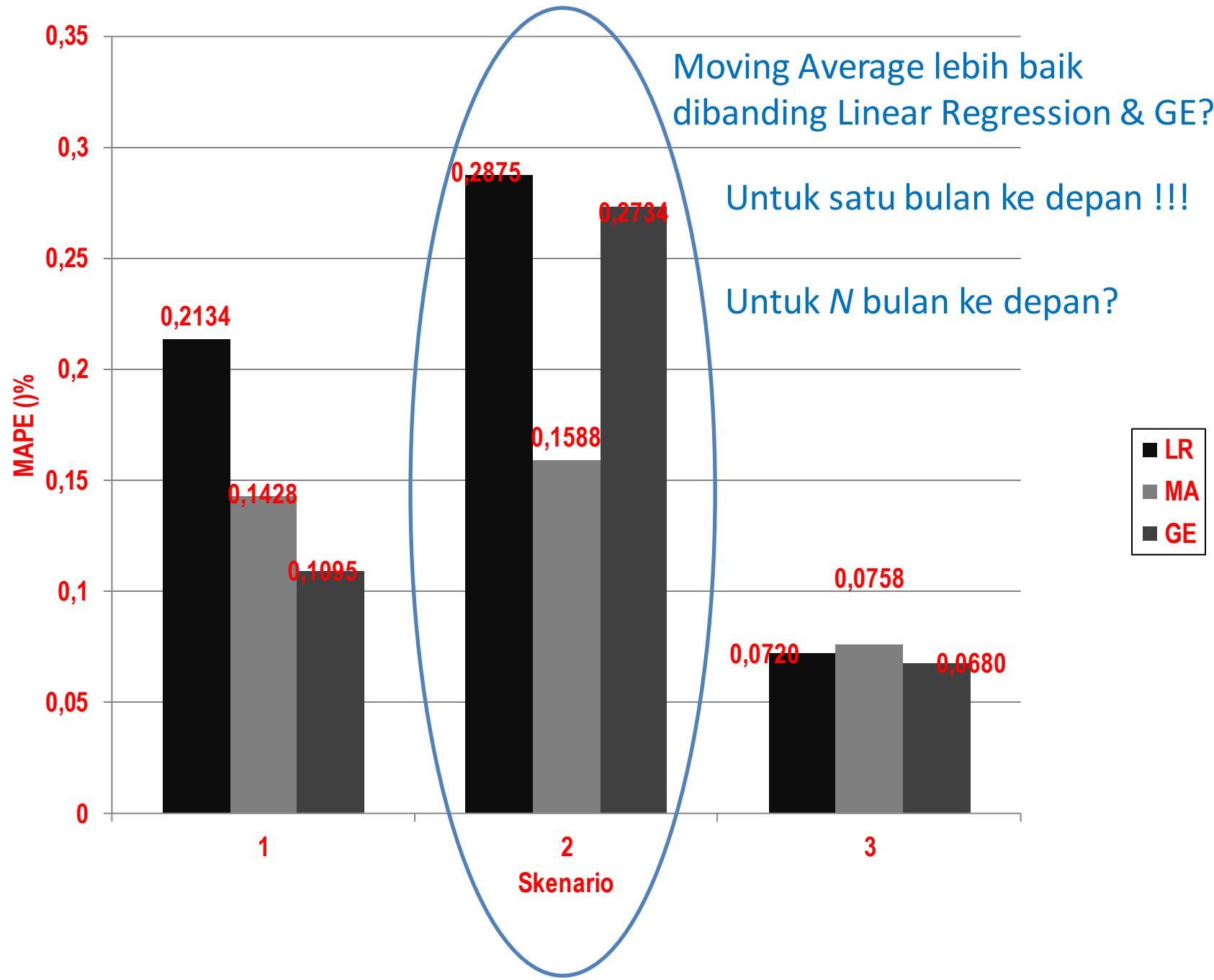
- **Skenario 2**

- Data *training* : 12 bulan (2002)
- Data *validation* : 12 bulan (2003)
- Data *testing* : 48 bulan (2004 - 2007)

- **Skenario 3**

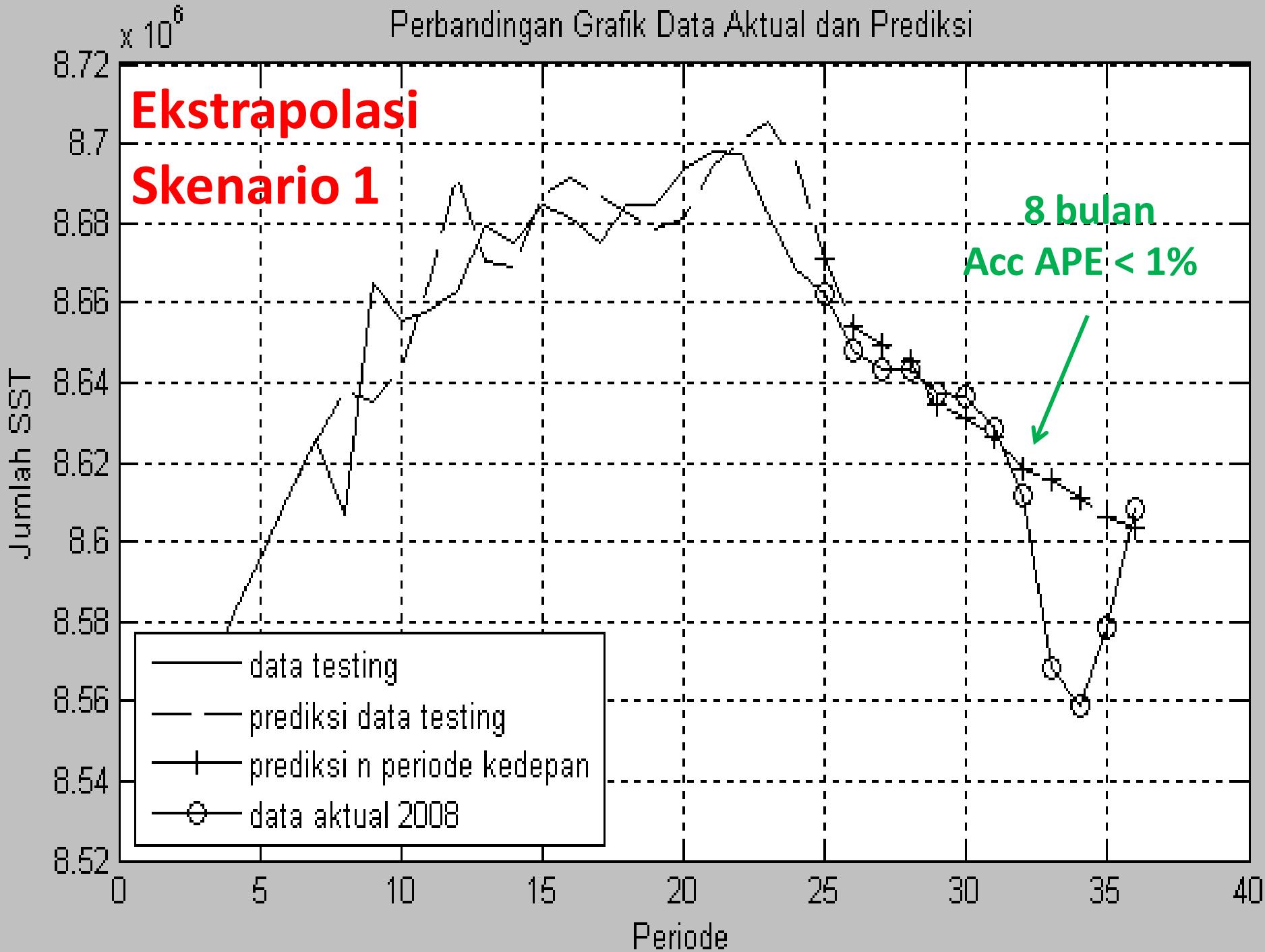
- Data *training* : 48 bulan (2002 - 2005)
- Data *validation* : 12 bulan (2006)
- Data *testing* : 12 bulan (2007)

BNF	Skenario	Ukuran Populasi	Generasi	Pc	Rata-rata MAPE BestSoFar	Standar Deviasi MAPE BestSoFar	Minimum MAPE BestSoFar	Maksimum MAPE BestSoFar
I	1	100	1000	0.9	0.2073175	0.026198929	0.16425	0.25167
		100	1000	0.7	0.469501	1.102284372	0.16011	5.1473
		50	2000	0.9	0.5020295	1.090989518	0.18624	5.1327
		50	2000	0.7	0.7214725	2.202654143	0.16278	10.0767
	2	100	1000	0.9	1.246051	2.035416117	0.20048	5.2621
		100	1000	0.7	3.5001885	3.630704359	0.21622	9.7619
		50	2000	0.9	3.2708075	3.68011751	0.22339	9.5804
		50	2000	0.7	4.7941555	4.616802348	0.2421	10.2182
	3	100	1000	0.9	0.209561	0.030683767	0.17222	0.28042
		100	1000	0.7	0.2056335	0.02752859	0.16704	0.26339
		50	2000	0.9	0.425015	0.58528908	0.17837	2.1343
		50	2000	0.7	0.419986	0.603236566	0.18534	2.2072
II	1	100	1000	0.9	0.250019	0.0280224	0.20973	0.30066
		100	1000	0.7	0.233899	0.026397873	0.20378	0.27668
		50	2000	0.9	0.2449035	0.027101195	0.20475	0.3041
		50	2000	0.7	0.2391745	0.026761996	0.2083	0.2867
	2	100	1000	0.9	0.221610526	0.024266181	0.1777	0.26499
		100	1000	0.7	0.237832	0.050575979	0.19711	0.39802
		50	2000	0.9	0.242081	0.046831149	0.18799	0.34969
		50	2000	0.7	0.2664384	0.056889061	0.17859	0.39802
	3	100	1000	0.9	0.2241525	0.016556672	0.20947	0.28582
		100	1000	0.7	0.218733	0.014143083	0.19211	0.25895
		50	2000	0.9	0.224914	0.015015264	0.19877	0.27292
		50	2000	0.7	0.226015294	0.019985903	0.19316	0.25895



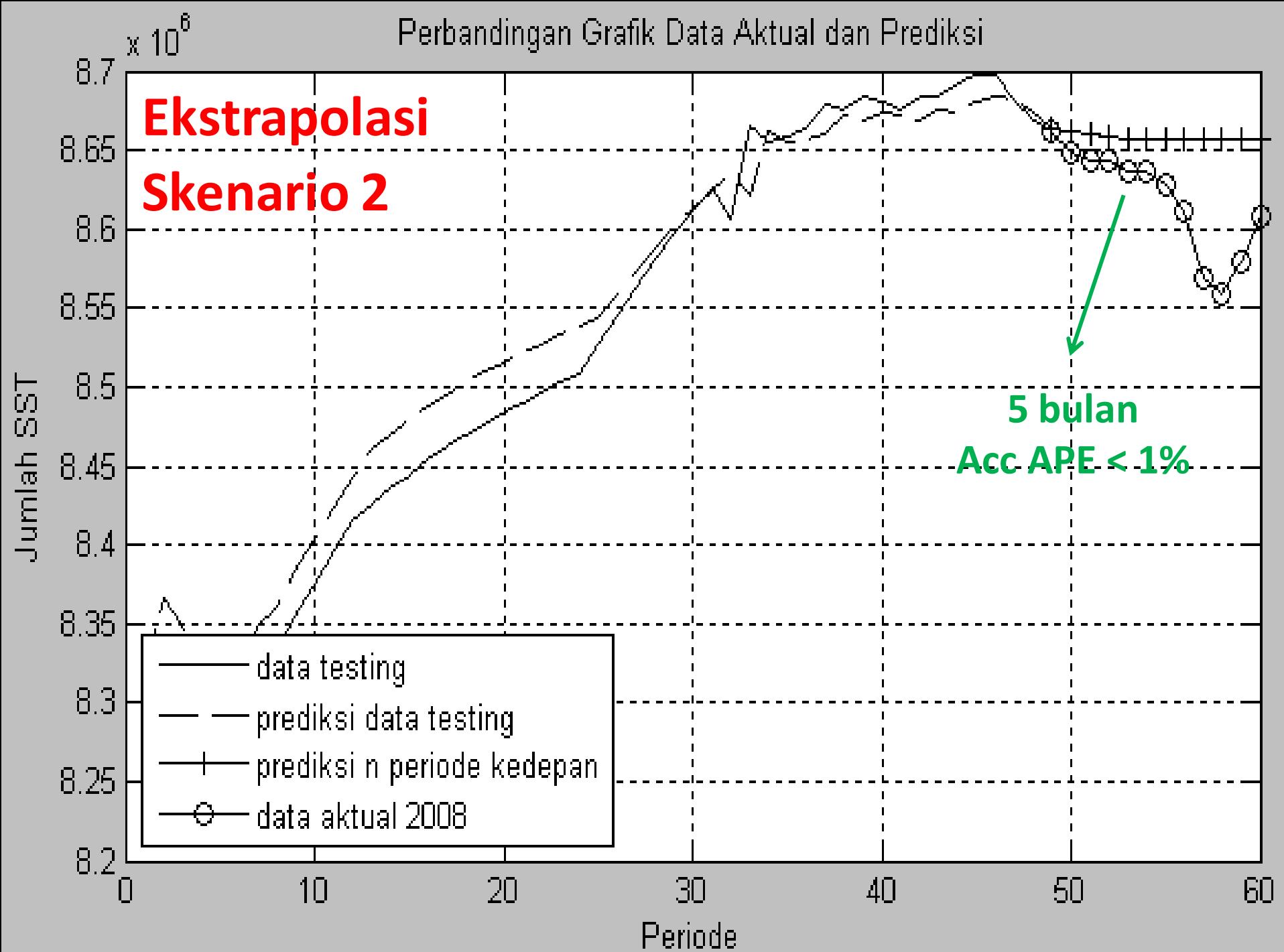
Skenario	Kombinasi Parameter GE	Fungsi Prediksi yang Menghasilkan Nilai Minimum MAPE <i>BestSoFar</i> (Fungsi Prediksi Optimal)
1	UkPop = 100 Generasi = 1000 Pc = 0.9 BNF I	$(0.6*x_3)-(0.4*x_5)-(0.4*x_4)+(0.3*x_1)+(0.9*x_2)+(0.2*e_1)$
2	UkPop = 100 Generasi = 1000 Pc = 0.9 BNF II	$\cos(0.6)-\sin(\cos(0.6))+\sin(x_1)$
3	UkPop = 100 Generasi = 1000 Pc = 0.7 BNF I	$(0.2*x_3)+(0.9*x_1)+(0.5*x_2)-(0.6*x_4)$

Perbandingan Grafik Data Aktual dan Prediksi



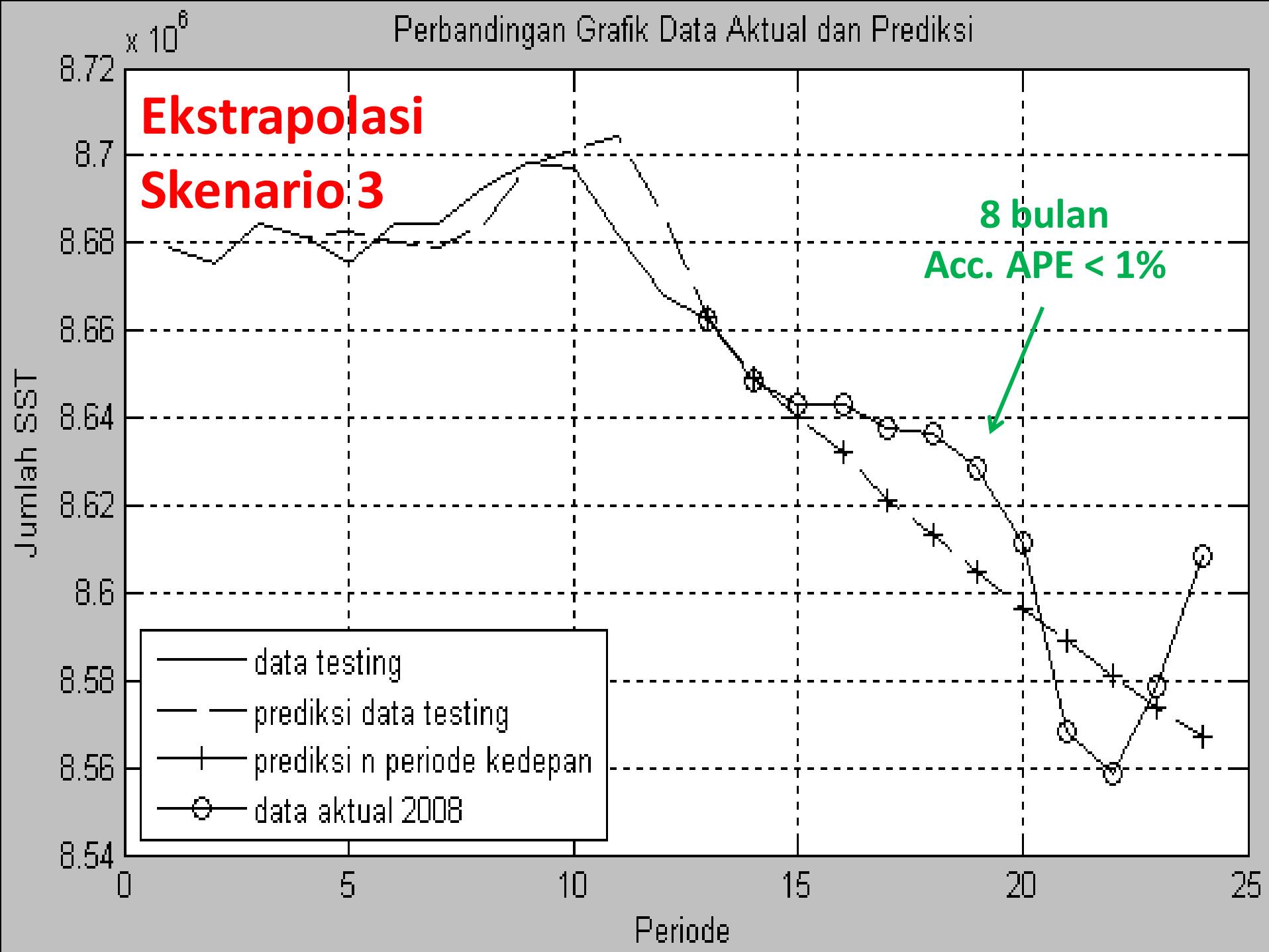
No	Data Aktual	Data Prediksi	Kesalahan	APE	Akumulasi APE
1	8662290	8670817	-8527	0.098438173	0.098438173
2	8648325	8653876	-5551	0.064185839	0.162624012
3	8643338	8649083	-5745	0.066467376	0.229091389
4	8643335	8645496	-2161	0.025001923	0.254093312
5	8637353	8634513	2840	0.032880444	0.286973756
6	8636355	8630875	5480	0.063452695	0.350426451
7	8628375	8626440	1935	0.022426007	0.372852458
8	8611418	8618597	-7179	0.083366061	0.456218519
9	8568525	8615896	-47371	0.552848944	1.009067464
10	8558550	8611211	-52661	0.615302826	1.624370289
11	8578500	8605900	-27400	0.319403159	1.943773448
12	8608425	8603382	5043	0.058582145	2.002355593

Perbandingan Grafik Data Aktual dan Prediksi



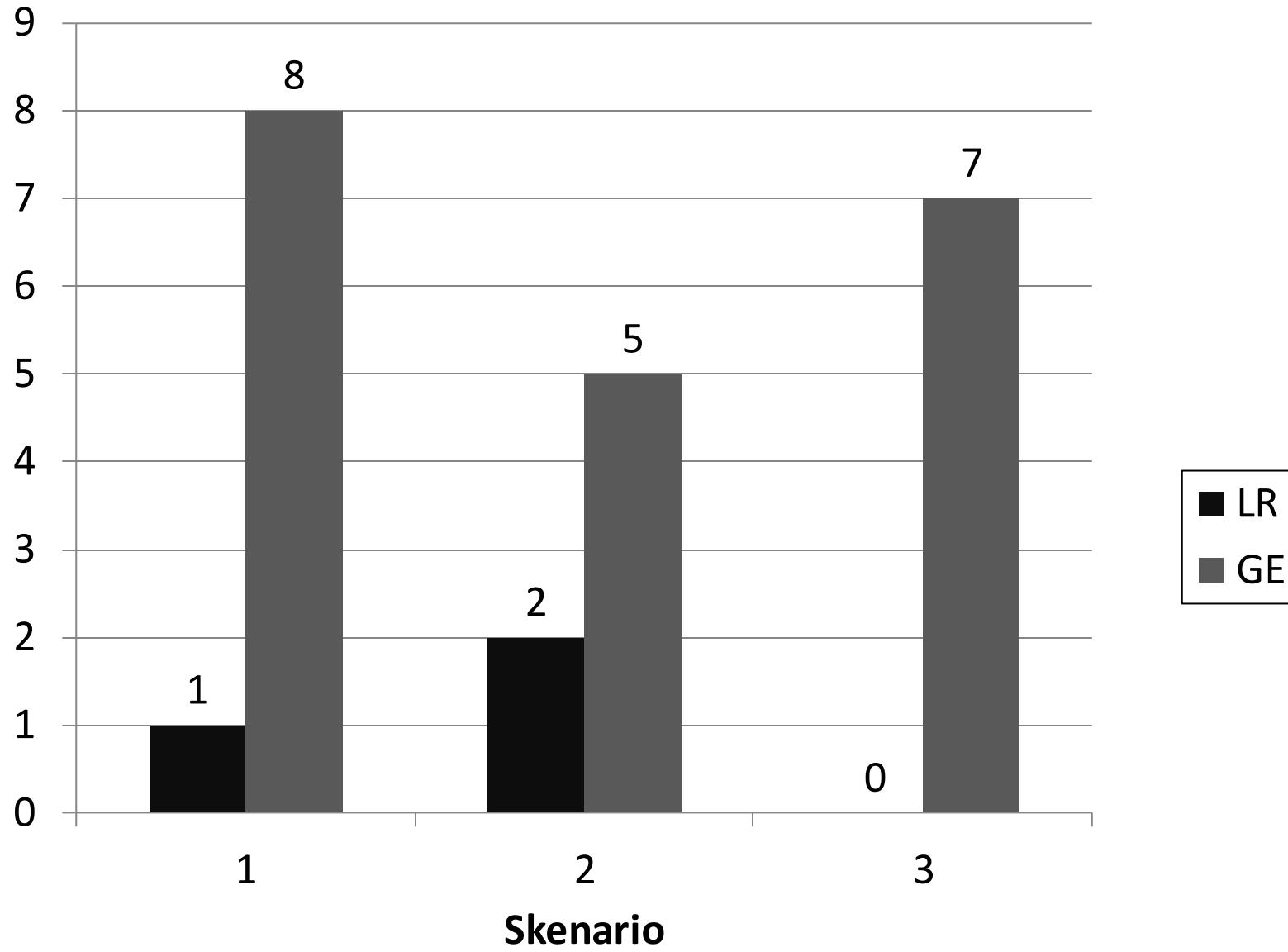
No	Data Aktual	Data Prediksi	Kesalahan	APE	Akumulasi APE
1	8662290	8664253	-1963	0.022661	0.022661444
2	8648325	8661578	-13253	0.153244	0.175904991
3	8643338	8659784	-16446	0.190274	0.366178703
4	8643335	8658582	-15247	0.176402	0.54258052
5	8637353	8657770	-20417	0.23638	0.778960809
6	8636355	8657222	-20867	0.241618	1.020578945
7	8628375	8656850	-28475	0.330016	1.350594736
8	8611418	8656596	-45178	0.524629	1.875223781
9	8568525	8656427	-87902	1.025871	2.901094628
10	8558550	8656314	-97764	1.142296	4.043390928
11	8578500	8656235	-77735	0.906161	4.949551679
12	8608425	8656184	-47759	0.554794	5.504345384

Perbandingan Grafik Data Aktual dan Prediksi



No	Data Aktual	Data Prediksi	Kesalahan	APE	Akumulasi APE
1	8662290	8663088	-798	0.009212345	0.009212345
2	8648325	8649043	-718	0.008302186	0.01751453
3	8643338	8639994	3344	0.038688757	0.056203287
4	8643335	8632169	11166	0.129186246	0.185389533
5	8637353	8620904	16449	0.190440289	0.375829821
6	8636355	8613469	22886	0.264996054	0.640825876
7	8628375	8605013	23362	0.270757819	0.911583695
8	8611418	8596125	15293	0.1775898	1.089173495
9	8568525	8589172	-20647	0.240963293	1.330136788
10	8558550	8581240	-22690	0.265115002	1.59525179
11	8578500	8573917	4583	0.053424258	1.648676049
12	8608425	8567303	41122	0.477694816	2.126370865

Jumlah Hasil Kelayakan Periode Prediksi (bulan)



Kesimpulan

- GE menggunakan representasi individu yang bisa digunakan untuk meng-“evolusi” program yang bebas bahasa.
- Representasi individunya menggunakan BNF.
- Dua operator penting GE: *Duplicate* dan *Prune*.
- GE memiliki performansi yang sangat baik untuk masalah *Symbolic Regression*, *Trigonometric Identities*, dan *Symbolic Integration*.

Daftar Pustaka

- [SUYo8] Suyanto, 2008, Evolutionary Computation: Komputasi Berbasis “Evolusi” dan “Genetika”, penerbit Informatika Bandung.
- [RYA98a] Ryan Conor and O'Neill Michael, 1998, “Grammatical Evolution: A Steady State approach”. In Proceedings of the Second International Workshop on Frontiers in Evolutionary Algorithms 1998, pages 419-423.

Paralelisasi dan Distribusi

Dr. Suyanto, S.T., M.Sc.
HP/WA: 0812 845 12345

Intelligence Computing Multimedia (ICM)
Informatics faculty – Telkom University

Intro

- **Paralelisasi:** mencari bagian algoritma yang memiliki potensi untuk dijalankan dalam waktu bersamaan (*concurrent*) oleh beberapa prosesor.
- **Distribusi:** kasus khusus dari paralelisasi dimana beberapa prosesor terletak di beberapa mesin dalam suatu jaringan komputer.

Intro

- Pada umumnya, satu PC paling banyak memiliki dua prosesor, sehingga paralelisasi lokal (hanya satu PC) tidak terlalu kuat pengaruhnya pada kecepatan proses.
- Sebaliknya, sistem terdistribusi bisa melibatkan sangat banyak PC sehingga bisa menghasilkan sistem yang sangat handal. Tetapi, pada sistem terdistribusi kita harus memperhitungkan biaya tambahan yang berupa transmisi data antar komputer.

Intro

- Banyak masalah optimasi di dunia nyata yang ruang solusinya amat sangat besar sehingga membutuhkan waktu sangat lama untuk menemukan solusi optimumnya.
- EAs sangat potensial untuk diimplementasikan ke dalam sistem paralel atau terdistribusi untuk menghasilkan *real-time systems*.
- Bagaimana menemukan bagian-bagian EAs yang bisa diparalelisasi?

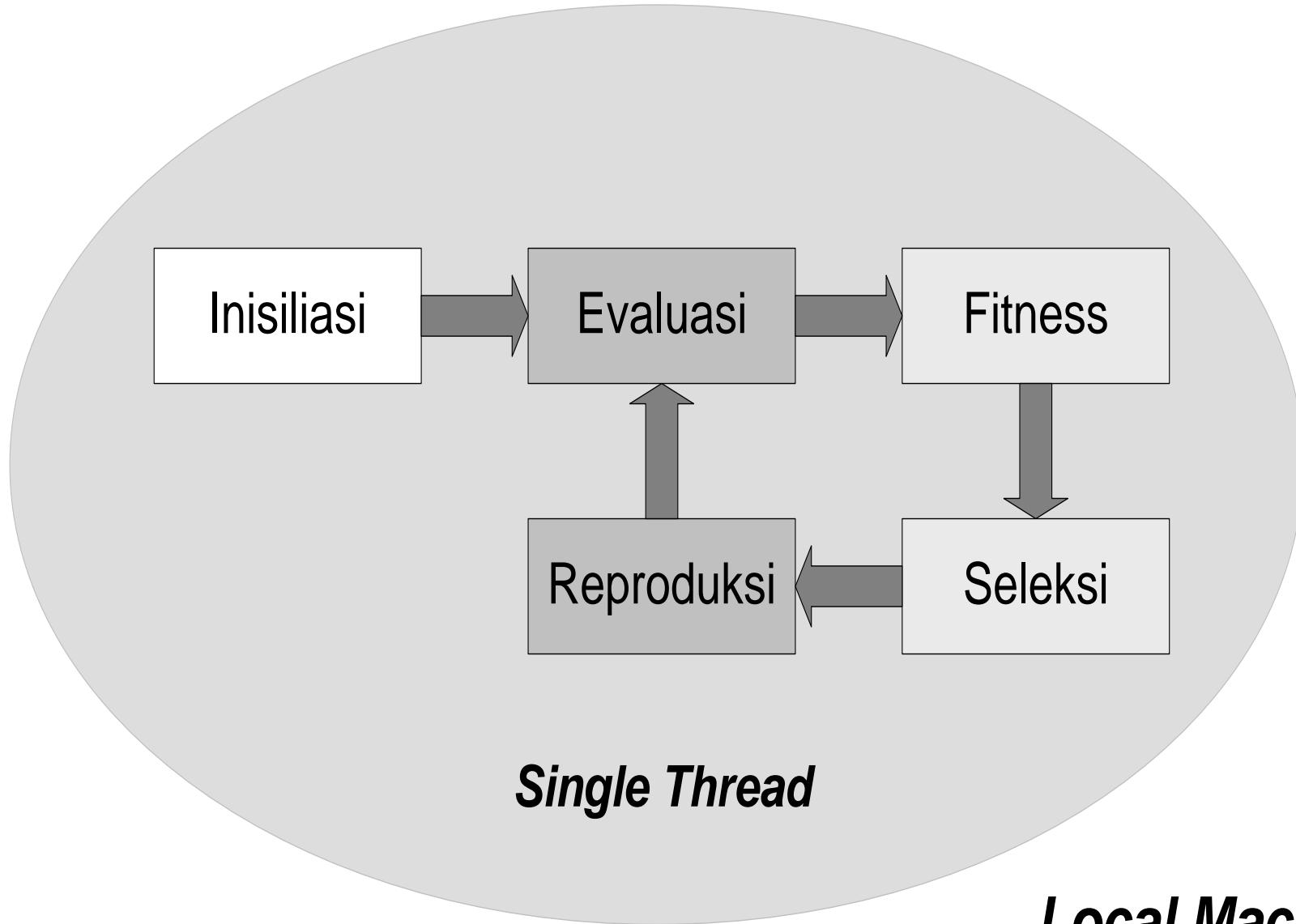
Analisis

- Pada EAs, bagian mana yang performansinya dapat ditingkatkan melalui paralelisasi?
- Inisialisasi?
- Evaluasi?
- Seleksi?
- Reproduksi?

Analisis

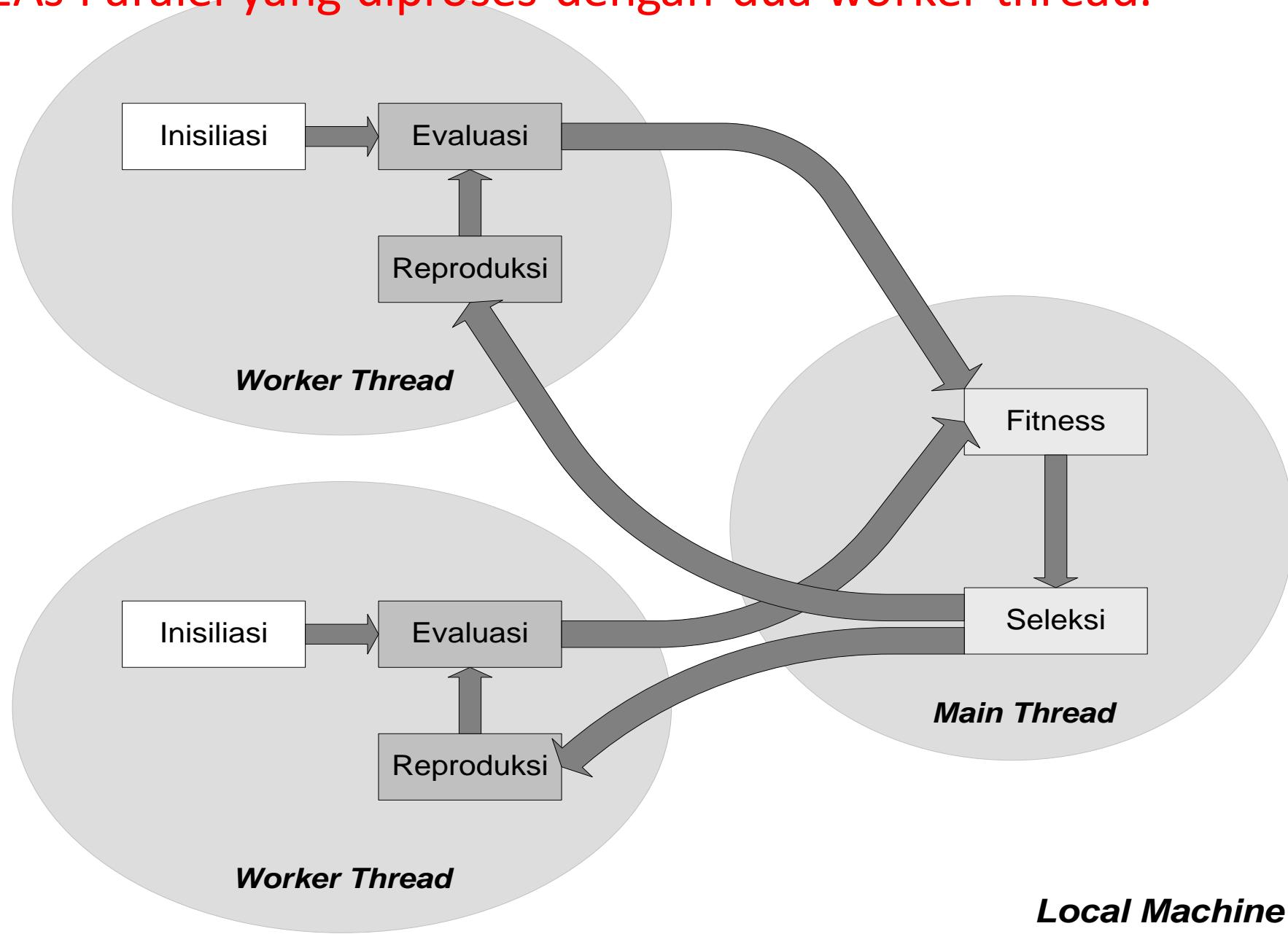
- **Evaluasi individu** dapat diparalelisasi karena prosesnya dilakukan pada setiap individu secara terpisah dan tidak bergantung pada individu lain dalam populasi.
- **Reproduksi** bisa diparalelisasi karena dilakukan secara terpisah.
- Reproduksi bisa dilakukan menggunakan operasi pembuatan individu baru ataupun rekombinasi dan/atau mutasi individu saat ini sehingga dihasilkan individu baru.

EAs sekuensial yang diproses dengan *single thread*.



Local Machine

EAs Paralel yang diproses dengan dua worker thread.



Distribusi

- Distribusi lebih baik dibandingkan Paralelisasi?
- **Belum tentu**
- Biaya transmisi pertukaran data antar komputer mungkin saja lebih besar dibandingkan biaya komputasi.
- Jadi, distribusi suatu algoritma seharusnya dilakukan hanya jika biaya transmisi pertukaran data jauh lebih kecil dibandingkan biaya komputasi.
- Sebagai contoh, pada penghitungan akar kuadrat dari suatu fungsi matematika $f(x)$, biaya transmisi vektor parameter x ke komputer lain mungkin akan membutuhkan waktu yang jauh lebih lama dibandingkan biaya komputasi fungsi $f(x)$ secara lokal.
- Untuk kasus ini, distribusi merupakan suatu langkah yang sia-sia atau merugikan.
- Jadi, sebelum memutuskan penggunaan distribusi, kita harus memperhitungkan masalah biaya secara detail dan hati-hati.

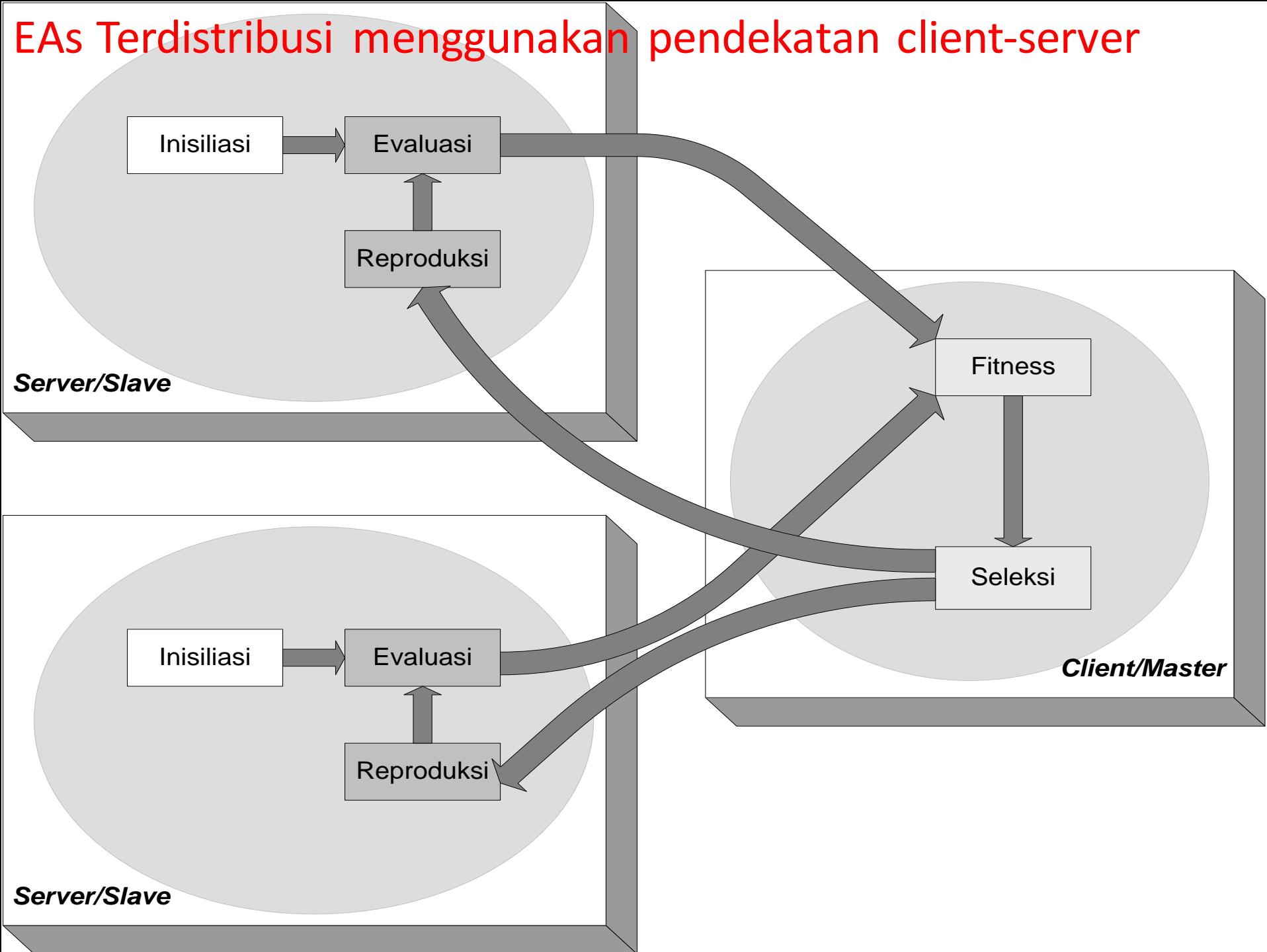
Distribusi

- *Client-Server*
- *Island Model*
- *Mixed Distribution*

Client-Server

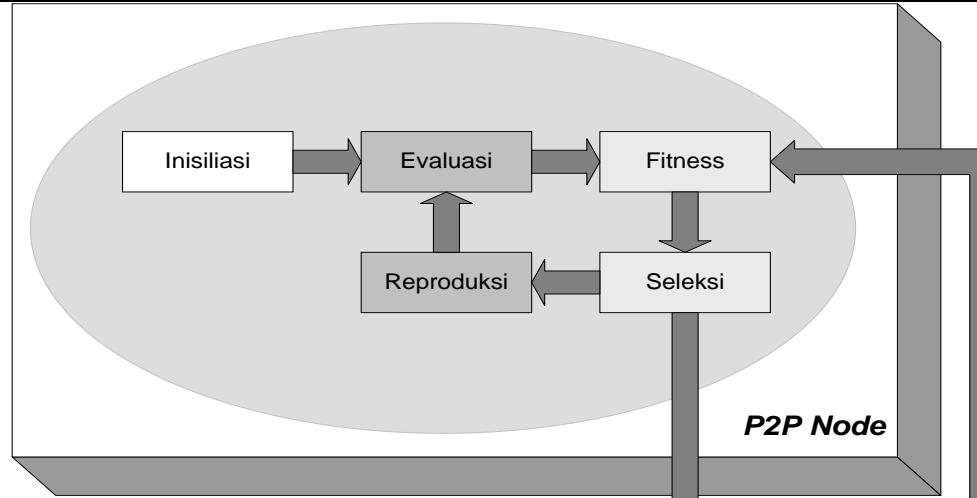
- Pada EAs, jika evaluasi fungsi *fitness* membutuhkan waktu yang sangat lama, pendekatan paling mudah untuk mendistribusikan proses adalah menggunakan skema ***client-server*** atau disebut juga ***master-slave***.

EAs Terdistribusi menggunakan pendekatan client-server

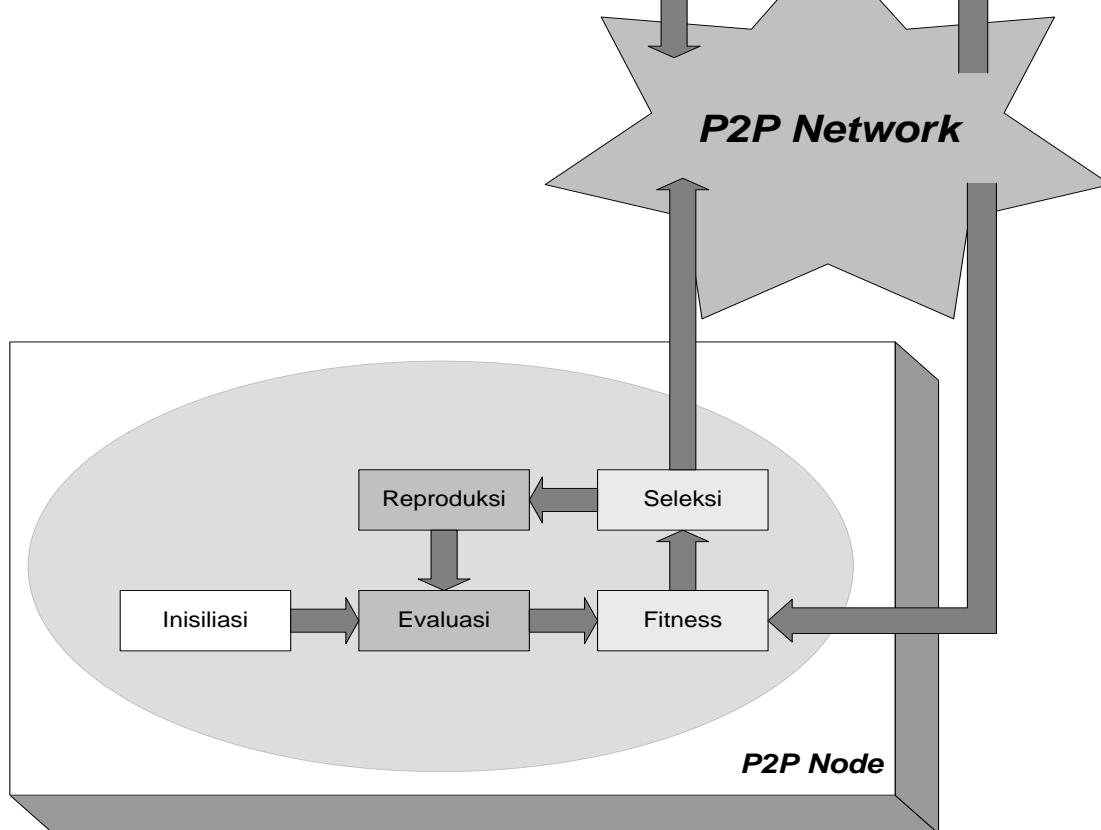


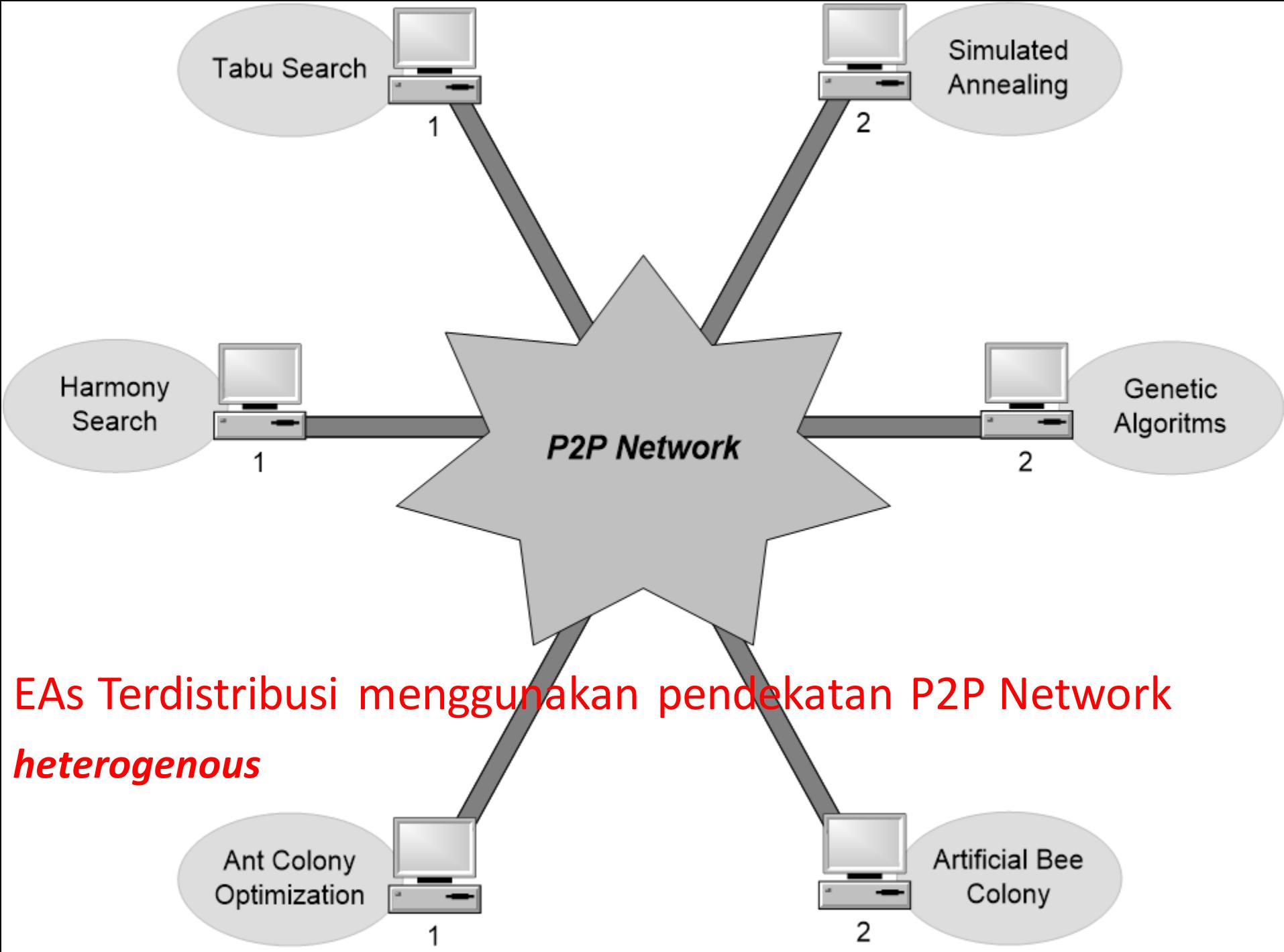
Island Model

- *homogenous*
- *heterogenous*



EAs Terdistribusi menggunakan pendekatan P2P Network
homogenous





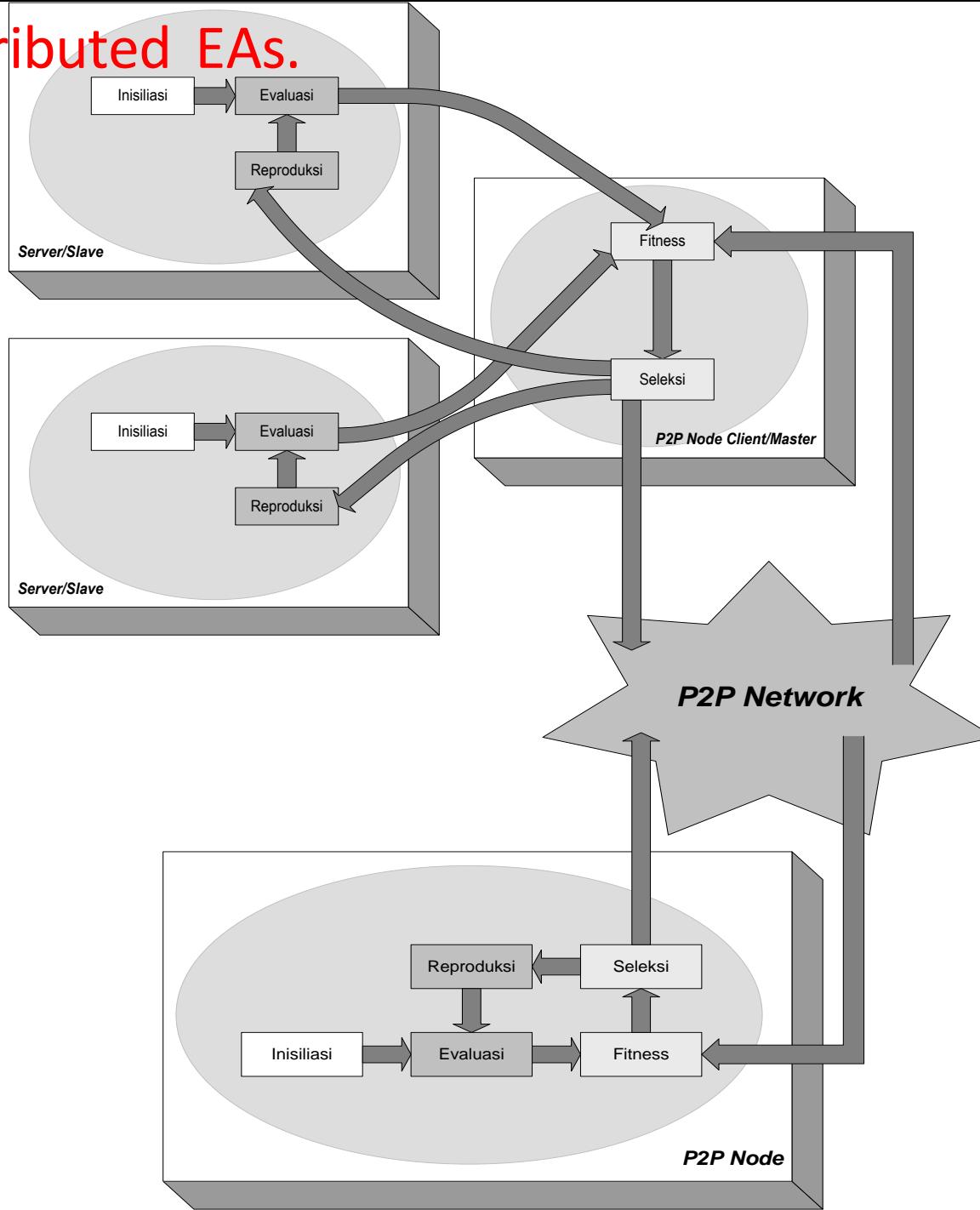
Island Model

- Mana yang lebih baik *Homogenous* atau *heterogenous*?
- *Heterogenous*
- Tidak ada satupun algoritma yang memiliki performansi bagus untuk semua masalah.
- *Local search*, seperti *Simulated Annealing* (SA) dan *Tabu Search* (TS), sangat sesuai untuk masalah yang memiliki satu nilai optimum tunggal (tidak ada optimum lokal).
- Algoritma yang bekerja secara paralel (berbasis populasi), seperti *Harmony Search* (HS), *Genetic Algorithm* (GA), *Ant Colony Optimization* (ACO), dan *Artificial Bee Colony* (ABC) bisa mengatasi masalah yang memiliki banyak optimum lokal.
- Dengan menggabungkan banyak algoritma menjadi *heterogenous island model*, tentu saja beragam masalah lebih mudah diselesaikan dibandingkan jika kita menggunakan *homogenous island model*.

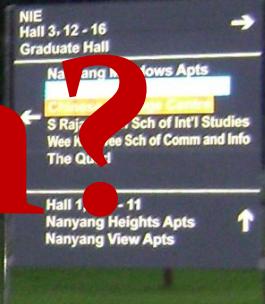
Mixed Distribution

- Distribusi campuran yang memiliki *P2P network* sekaligus *client-server*
- Sangat sesuai untuk menyelesaikan masalah yang memerlukan populasi besar dengan individu-individu yang membutuhkan waktu evaluasi sangat lama.

Mixed distributed EAs.



Question?



Kesimpulan

- Paralelisasi maupun distribusi memungkinkan EC menjadi sangat powerful
- Perlu analisis EC secara detail untuk menentukan jenis paralelisasi maupun distribusi

Daftar Pustaka

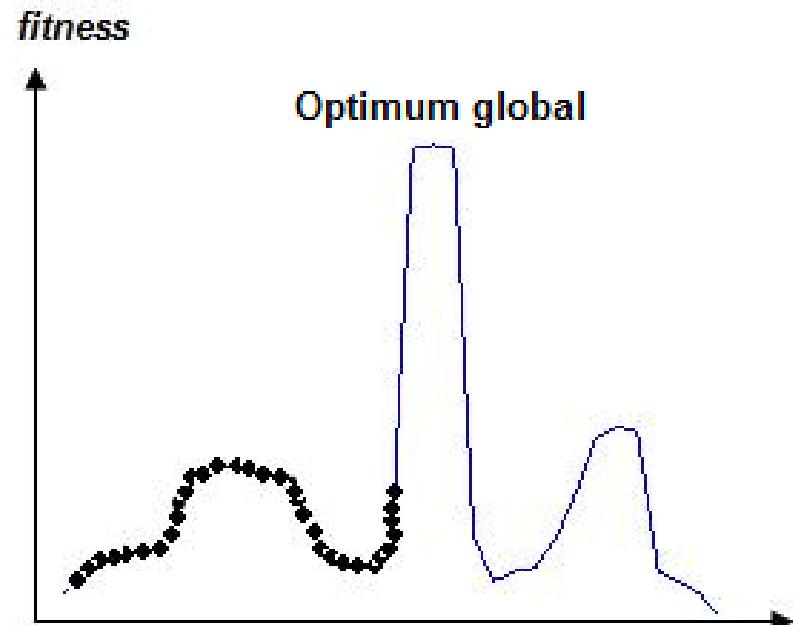
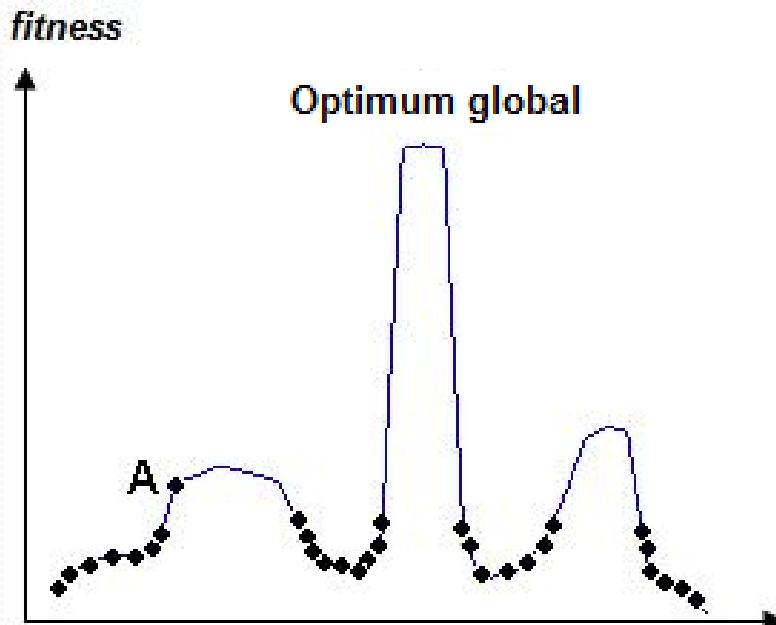
- [SUYo8] Suyanto, 2008, Evolutionary Computation: Komputasi Berbasis “Evolusi” dan “Genetika”, penerbit Informatika Bandung.

Konvergensi Prematur dan Pencegahannya

Dr. Suyanto, S.T., M.Sc.
HP/WA: 0812 845 12345

Intelligence Computing Multimedia (ICM)
Informatics faculty – Telkom University

Konvergensi Prematur



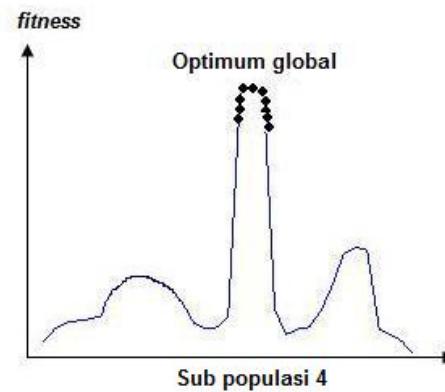
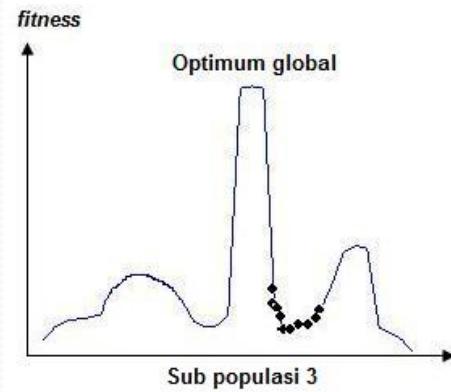
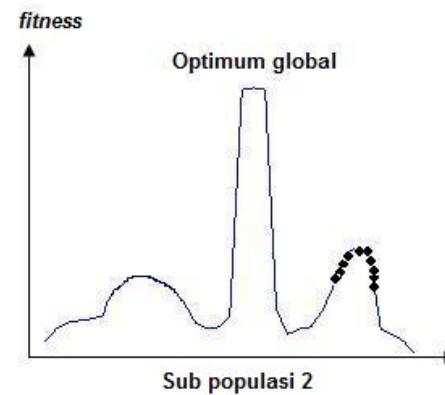
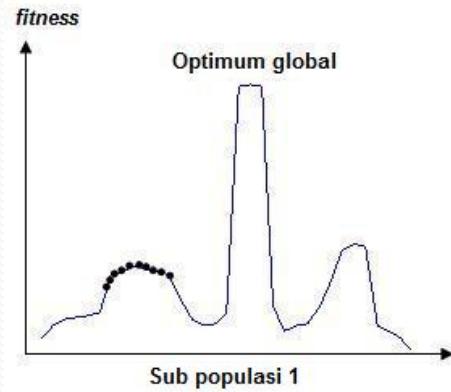
Solusi

- *Island model* EAs
- Representasi Rumit (*Messy Encoding*)
- *Adaptive* EAs

Island model EAs

- Pada metode ini, sejumlah N individu dalam suatu populasi dibagi menjadi N_k kelompok (sub populasi). Masing-masing kelompok berisi n individu [SUYo5a].
- Pada model ini, pindah silang dan/atau mutasi hanya terjadi di dalam setiap sub populasi saja.
- Dengan sub populasi, dapat dikurangi adanya kemungkinan bahwa semua sub populasi konvergen pada lokal optimum yang sama.
- Hal ini kemungkinan bisa menghindarkan EAs dari konvergensi prematur.

Island model EAs



Pertukaran individu antar sub populasi

- Jika tidak ada interaksi antar individu dari sub populasi yang berbeda, maka EAs berbasis sub populasi akan sama saja dengan mengerjakan sejumlah N_k sub populasi yang masing-masing berisi v individu.
- Tanpa interaksi antar individu dari sub populasi berbeda, kemungkinan sub populasi EAs akan cepat konvergen juga.
- Oleh karena itu, perlu ada pertukaran individu-individu antar sub populasi.
- Hal ini bisa meningkatkan variasi individu pada sub populasi sehingga bisa mencegah konvergensi prematur.

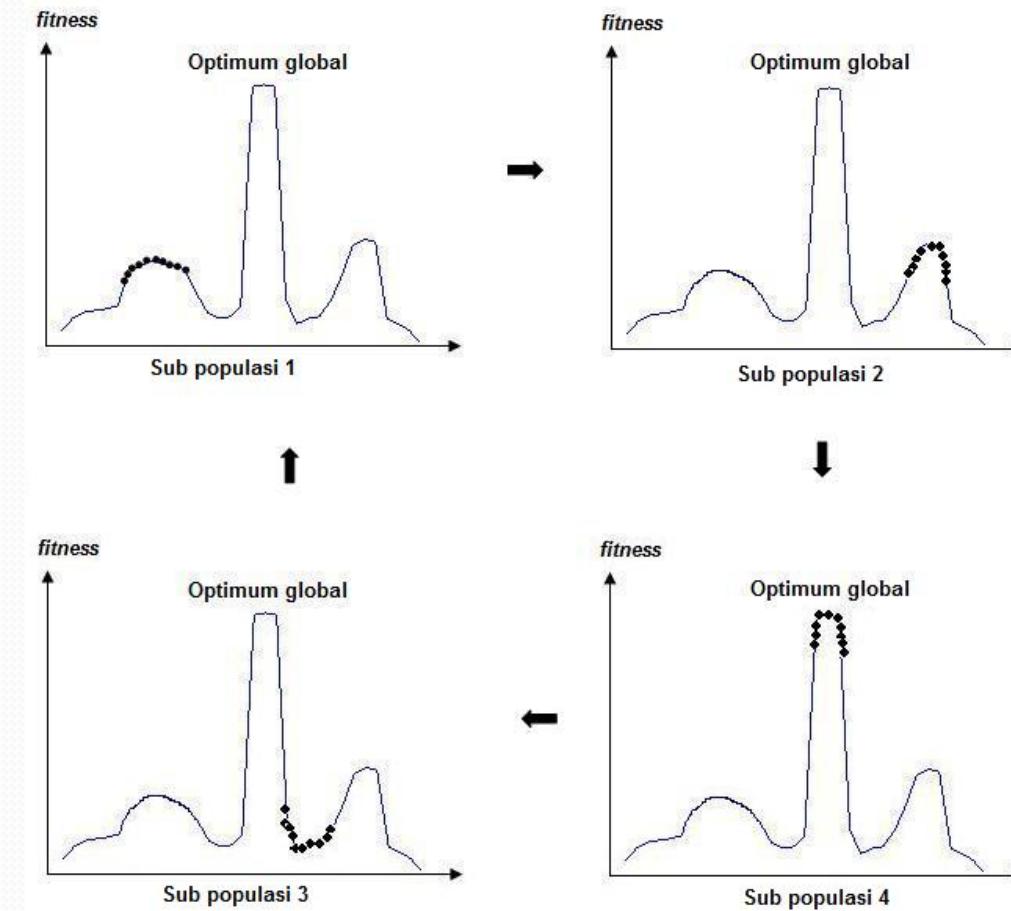
Pertukaran individu antar sub populasi

- **Bagaimana cara pertukaran individu yang baik?** Secara periodik pada sejumlah generasi tertentu dan berdasarkan *tunneling probability* p_t , dua individu dipilih secara acak dari dua sub populasi berbeda kemudian dipertukarkan.
- **Seberapa besar periode pertukaran yang baik?** Jika terlalu cepat, sub populasi bisa konvergen pada optimum lokal yang sama. Jika terlalu lama, tentu saja hanya membuang-buang waktu. Kebanyakan praktisi EAs menggunakan periode pertukaran antara 25 sampai 150 generasi. Tetapi periode ini bisa didefinisikan secara adaptif.
- **Berapa jumlah individu yang sebaiknya dipertukarkan?** Biasanya hanya 2 sampai 5 individu yang dipertukarkan. Tetapi, tentu saja bisa lebih besar dari rentang tersebut dan disesuaikan pada ukuran populasi.

Pertukaran individu antar sub populasi

- **Yang dipertukarkan harus individu terbaik?**
 - Martin, WN menemukan bahwa individu-individu yang dipilih secara acak untuk dipertukarkan lebih baik dibandingkan pertukaran yang dilakukan pada individu-individu terbaik saja [MAR97].
 - Interaksi antar sub populasi biasanya dilakukan secara melingkar. Ketika diperlukan pertukaran individu, sejumlah n individu pada sub populasi 1 yang terpilih secara acak menggantikan n individu pada sub populasi 2 (bisa dipilih yang paling rendah nilai *fitness*-nya). Kemudian n individu pada sub populasi 2 yang terpilih secara acak menggantikan n individu pada sub populasi 3. Demikian seterusnya sampai n individu pada sub populasi terakhir yang terpilih secara acak menggantikan n individu pada sub populasi 1.

Pertukaran individu antar sub populasi



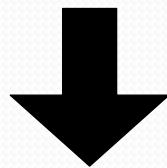
Representasi Rumit (*Messy Encoding*)

- *Schema* $S_1 = 1xxxxxx0$ sangat mudah dirusak
- Bagaimana mempertahankan S_1 dari kerusakan?
 - Suatu representasi individu yang disebut pengkodean rumit (*messy encoding*) bisa digunakan untuk tujuan ini [SUYo5a].
 - Idenya adalah dengan membuat representasi kromosom yang tidak bergantung pada posisi.
 - Jika *schema* S_1 diubah menjadi xxxxxxx1 (bit pertama dipindah ke belakang), maka *schema* tersebut tidak akan mudah rusak karena *defining length* menjadi jauh lebih pendek (yang tadinya 7 menjadi 1).
 - Bagaimana caranya? Masukkan posisi gen sebagai bagian dari kromosom.

Representasi Rumit (*Messy Encoding*)

Messy encoding:

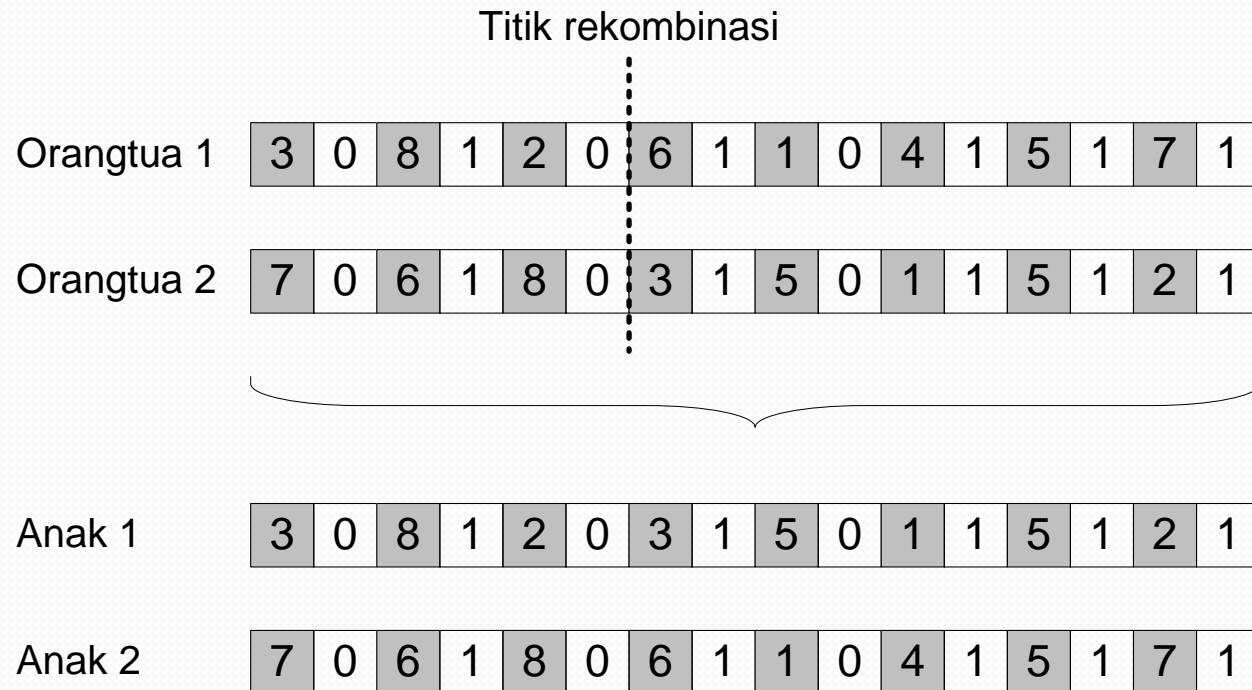
3	0	8	0	2	1	6	1	1	0	4	1	5	1	7	1
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---



Binary encoding:

0	1	0	1	1	1	1	0
---	---	---	---	---	---	---	---

Representasi Rumit (*Messy Encoding*)

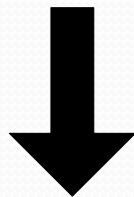


Representasi Rumit (*Messy Encoding*)

Messy encoding:

3	0	8	1	2	0	3	1	5	0	1	1	5	1	2	1	...
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	-----

6	0	4	1	8	1	3	1	2	0	5	1	4	1	5	1
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---



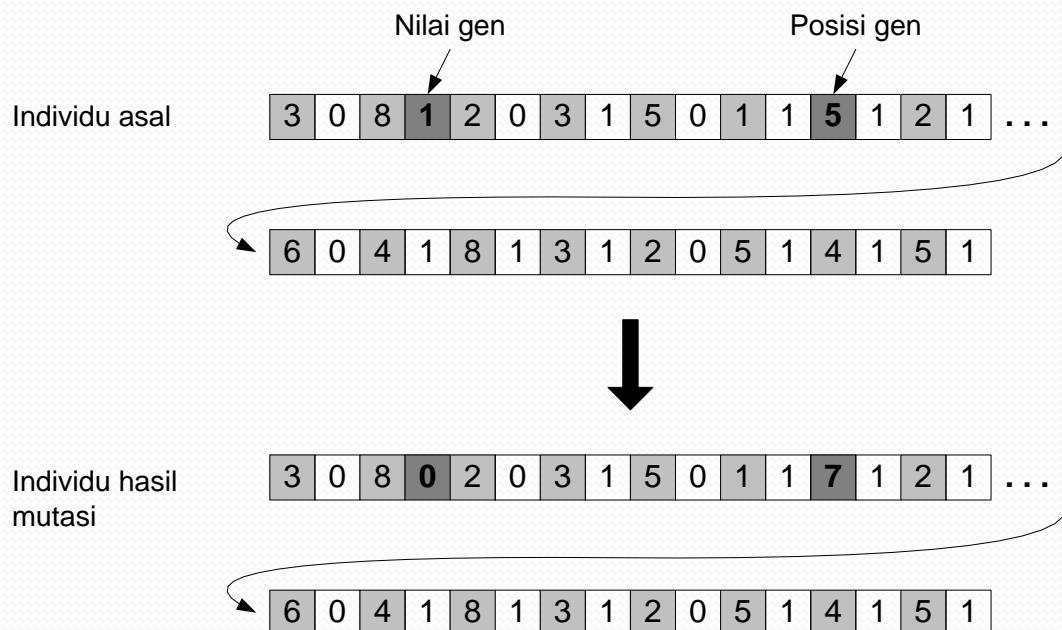
Binary encoding:

1	0	0	1	0	0	0	1
---	---	---	---	---	---	---	---

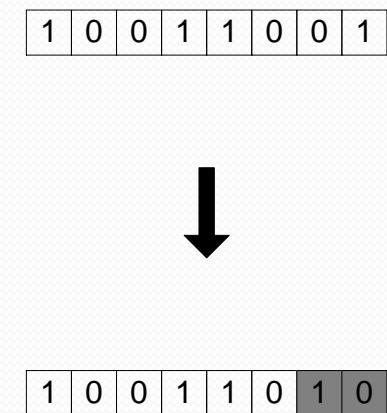
Nilai gen untuk posisi 7 = 0
(dibangkitkan secara acak)

Representasi Rumit (*Messy Encoding*)

Messy encoding:



Binary encoding:

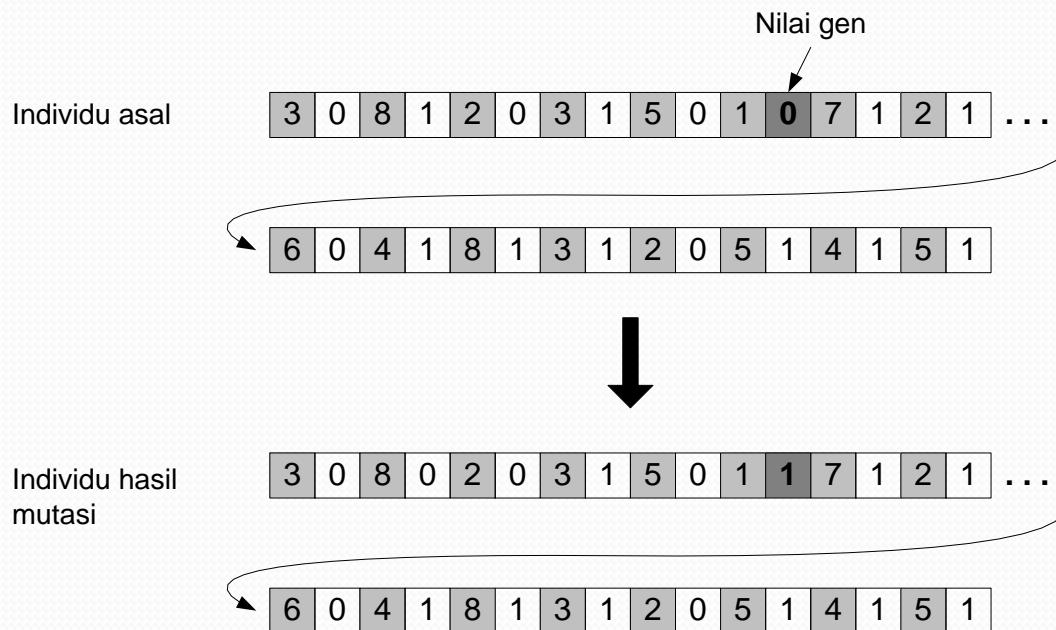


Representasi Rumit (*Messy Encoding*)

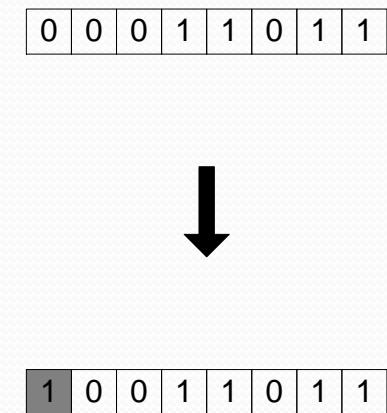
- Keuntungan penggunaan *messy encoding* sangat terlihat pada proses mutasi.
- Kemungkinan untuk terjadinya mutasi pada suatu gen tertentu menjadi lebih besar. Misalkan, individu terbaik pada suatu populasi adalah {0, 0, 0, 1, 1, 0, 1, 1} dan individu terbaik yang merupakan optimum global adalah {1, 0, 0, 1, 1, 0, 1, 1}.
- Artinya, hanya diperlukan mutasi satu gen saja (yaitu gen ke-1) untuk mendapatkan solusi optimum global. Jika kita menggunakan *binary encoding*, probabilitas termutasi gen ke-1 adalah sebesar $1/8$ dikali dengan probabilitas mutasi ($p_m/8$). Hal ini karena posisi gen pada *binary encoding* bersifat statis. Gen ke-1 selalu berada di posisi ke-1 kromosom.
- Tetapi, jika kita menggunakan *messy encoding*, maka probabilitas termutasi gen ke-1 akan lebih besar karena posisi gen ke-1 bisa berada dimana saja (tidak harus di posisi ke-1 kromosom). Hal ini karena posisi gen pada *messy encoding* bersifat dinamis.

Representasi Rumit (*Messy Encoding*)

Messy encoding:



Binary encoding:



Adaptive EAs

- Algoritma-algoritma pada EAs memiliki parameter-parameter yang harus didefinisikan secara hati-hati agar bisa konvergen pada optimum global dengan cepat.
- Terdapat empat parameter yang cukup sensitif terhadap performansi EAs, yaitu: ukuran populasi, *selective pressure* pada pemilihan orangtua, probabilitas crossover dan probabilitas mutasi.
- Seringkali kita merasa kesulitan dalam menentukan nilai parameter tersebut karena nilai parameter sangat bergantung pada kasus yang dihadapi.

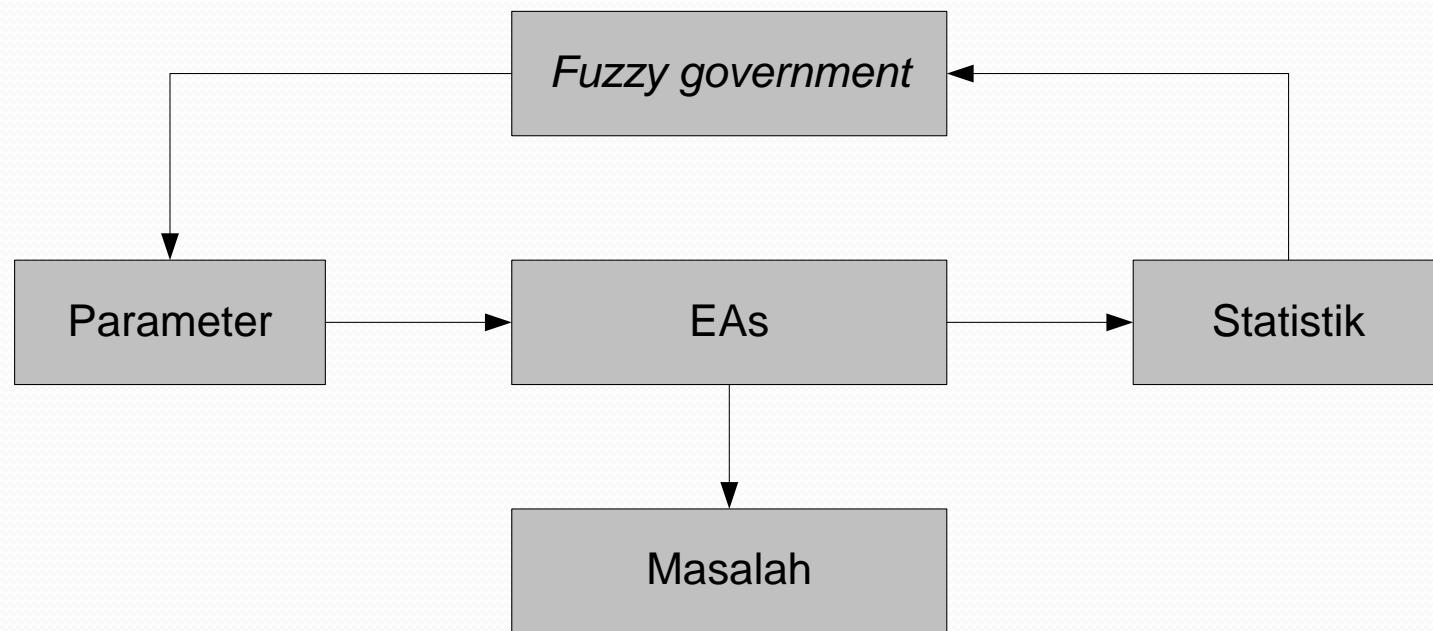
Adaptive EAs

- Interaksi yang terjadi pada parameter-parameter tersebut sangat kuat, kompleks dan sulit dipahami. Sehingga bisa dikatakan parameter-parameter tersebut seolah-olah menjadi satu kesatuan paket.
- Misalnya, suatu paket parameter (ukuran populasi = 100, probabilitas *crossover* = 0,8 dan probabilitas mutasi = 0,01) bisa memberikan performansi yang sangat baik untuk suatu masalah. Solusi optimum global selalu ditemukan dengan konvergensi yang sangat cepat. Tetapi, untuk masalah yang sama dengan kasus yang sedikit berbeda, paket paremeter tersebut justru mengakibatkan konvergensi prematur maupun konvergensi yang sangat lambat.

Adaptive EAs

- Bagaimana solusinya?
- Bangun EAs adaptif menggunakan *Fuzzy Government*.
- *Fuzzy Government* (FG) adalah kumpulan *fuzzy rules* dan *routines* yang berfungsi untuk: mengontrol proses evolusi, mendeteksi kemunculan solusi, mengubah-ubah (*tuning*) parameter EAs pada saat *running* sehingga dapat mencegah konvergensi prematur maupun konvergensi yang sangat lambat.

Adaptive EAs



Adaptive EAs

- Masukan (*input*) untuk FG adalah data-data statistik EAs yang diperoleh secara periodik pada sejumlah generasi tertentu (*sampling rate*), misalnya r generasi.
- Statistik EAs bisa berupa:
 - *genotype statistic*, yang merupakan kesimpulan atas aspek-aspek yang berhubungan dengan *genotypes* dari individu-individu dalam suatu populasi tanpa memperhatikan arti dari aspek-aspek tersebut. *Genotype statistic* yang umum digunakan adalah *fuzzy similarity measure*.
 - *phenotype statistic*, yang fokus pada nilai *fitness* individu untuk masalah yang dihadapi. *Phenotype statistic* bisa berupa *phenotype diversity measure* yang berupa *fitness range*, rasio *fitness terbaik* terhadap rata-rata *fitness*, variansi *fitness*, dan sebagainya.

Adaptive EAs

- Keluaran (*output*) dari FG adalah nilai-nilai parameter EAs yang paling sesuai untuk kondisi populasi saat ini.
- Parameter-parameter EAs yang bisa menjadi output FG adalah ukuran populasi, probabilitas *crossover*, probabilitas mutasi, dan *selective pressure* (jika memungkinkan).
- Keluaran FG ini bisa berubah-ubah pada setiap periode generasi sehingga membuat EAs bersifat adaptif.

Contoh Himpunan aturan fuzzy untuk pengontrolan probabilitas crossover (p_c) [TET01]

p_c	Ukuran populasi		
Generasi	Kecil	Sedang	Besar
Singkat	Sedang	Kecil	Kecil
Sedang	Besar	Besar	Sedang
Lama	Sangat besar	Sangat besar	Besar

Contoh Himpunan aturan fuzzy untuk pengontrolan probabilitas *crossover* (p_c) [TET01]

1. IF Generasi = Singkat AND Ukuran populasi = Kecil
THEN p_c = Sedang
2. IF Generasi = Singkat AND Ukuran populasi = Sedang
THEN p_c = Kecil
- ...
9. IF Generasi = Lama AND Ukuran populasi = Besar
THEN p_c = Besar

Himpunan aturan fuzzy untuk pengontrolan probabilitas mutasi (p_m) [TET01]

p_m	Ukuran populasi		
Generasi	Kecil	Sedang	Besar
Singkat	Besar	Sedang	Kecil
Sedang	Sedang	Kecil	Sangat kecil
Lama	Kecil	Sangat kecil	Sangat kecil

Himpunan aturan fuzzy pengontrolan *exploitation-oriented crossover rate* [TET01]

Δp_e	<i>Phenotype diversity</i>		
<i>Genotype diversity</i>	Rendah	Sedang	Tinggi
Rendah	Sedang	Kecil	Kecil
Sedang	Besar	Besar	Sedang
Tinggi	Besar	Besar	Sedang

Himpunan aturan fuzzy untuk pengontrolan *selective pressure* [TET01]

$\Delta\eta_{\min}$	<i>Phenotype diversity</i>		
<i>Genotype diversity</i>	Rendah	Sedang	Tinggi
Rendah	Kecil	Sedang	Besar
Sedang	Kecil	Besar	Besar
Tinggi	Kecil	Kecil	Besar

Kesimpulan

- Masalah utama EAs adalah Konvergensi Prematur
- Cara pencegahannya:
 - Island model EAs
 - Messy encoding
 - Adaptive EAs

Daftar Pustaka

- [SUYo8] Suyanto, 2008, Evolutionary Computation: Komputasi Berbasis “Evolusi” dan “Genetika”, penerbit Informatika Bandung.
- [SUYo5a] Suyanto, 2005, “Algoritma Genetika dalam MATLAB”, Andi Publisher, Yogyakarta, Indonesia, ISBN: 979-731-727-7.
- [TETo1] Tettamanzi A., Tomassini M., ”Soft Computing”. Springer-Verlag Berlin Heidelberg, 2001. Printed in Germany.
- [MAR97] Martin WN, Liegnic Jens, and Cohon James P, 1997, “Handbook of Evolutionary Computation”, IOP publishing Ltd and Oxford University Press.

Studi Kasus

Dr. Suyanto, S.T., M.Sc.
HP/WA: 0812 845 12345

Intelligence Computing Multimedia (ICM)
Informatics faculty – Telkom University

Intro

- “Untuk masalah seperti ini, apakah saya bisa menggunakan EAs?
- Kalau ya, algoritma EAs yang mana yang harus saya gunakan?
- Operator evolusi mana yang paling tepat?
- Bagaimana melakukan *setting* parameter yang baik?”

Strategi menggunakan EAs

- Kenali masalah yang anda hadapi
- Ruang masalah: seberapa besar?
- Analisa matematika: ada atau tidak?
- Jenis solusi: harus paling baik?
- Batasan waktu: ada atau tidak?
- Biner, Real, Automata atau Program?
- Nilai optimum atau permutasi?
- Ada pengetahuan untuk masalah yang dihadapi?

Strategi menggunakan EAs

- *Setting* parameter EAs
 - Observasi parameter
 - Pengontrolan parameter secara adaptif

Studi Kasus

- Peramalan Data *Time Series*
- TSP dengan batasan
- Penjadwalan kuliah

Peramalan Data *Time Series*

- Misalkan kita ingin membangun sistem peramalan hasil penjualan komputer di suatu perusahaan vendor besar.
- Misalkan kita memiliki data-data penjualan harian dari tanggal 01 Januari 2006 sampai 14 Desember 2007 yang ditunjukkan pada tabel berikut.

Hasil penjualan komputer per hari (01 Jan 2006 - 14 Dec 2007)

Tanggal	Penjualan (miliar rupiah)
14 Dec 2007	99,9573
13 Dec 2007	99,8459
12 Dec 2007	98,8708
11 Dec 2007	98,7480
10 Dec 2007	98,3897
09 Dec 2007	97,6780
08 Dec 2007	97,3797
...	...
...	...
...	...
03 Jan 2006	90,7597
02 Jan 2006	90,5770
01 Jan 2006	89,3897

Peramalan untuk Data *Time Series*

- Pada kasus peramalan ini, kita bisa melihat bahwa peramalan yang dimaksud adalah peramalan menggunakan data *time series*.
- Artinya, untuk memprediksi penjualan pada hari H , kita hanya menggunakan hasil-hasil penjualan pada hari-hari sebelumnya, $H-1$, $H-2$, dan seterusnya.
- Dengan demikian, masalah ini dapat dimodelkan secara linier sebagai berikut

Peramalan untuk Data *Time Series*

Dengan demikian, masalah ini dapat dimodelkan secara linier sebagai berikut

$$z = a_0 + a_1 y_1 + a_2 y_2 + \dots + a_k y_k$$

dimana y_1 sampai y_k adalah masukan yang berupa hasil-hasil penjualan pada hari-hari sebelumnya, $H-1$, $H-2$, ..., $H-k$.

Representasi individu GA

$$z = 0,1157 + 0,7315y_1 + 0,3995y_2$$



a_0	a_1	a_2
0,1157	0,7315	0,3995

Bagaimana dengan GE?

$N = \{\text{expr, op, pre_op}\}$

$T = \{\text{Sin, Cos, Tan, Log, +, -, /, *, y}_1, y_2, y_3, y_4, 0,5, 1, 1,5, 2, ()\}$

$S = \langle \text{expr} \rangle$

P dapat direpresentasikan sebagai:

- (1) $\langle \text{expr} \rangle ::= \langle \text{expr} \rangle \langle \text{op} \rangle \langle \text{expr} \rangle \quad (\text{A})$
| $(\langle \text{expr} \rangle \langle \text{op} \rangle \langle \text{expr} \rangle) \quad (\text{B})$
| $\langle \text{pre_op} \rangle (\langle \text{expr} \rangle) \quad (\text{C})$
| $\langle \text{var} \rangle \quad (\text{D})$

- (2) $\langle \text{op} \rangle ::= + \quad (\text{A})$
| - $\quad (\text{B})$
| / $\quad (\text{C})$
| * $\quad (\text{D})$

- (3) $\langle \text{pre_op} \rangle ::= \text{Sin} \quad (\text{A})$
| Cos $\quad (\text{B})$
| Tan $\quad (\text{C})$
| Log $\quad (\text{D})$

- (4) $\langle \text{var} \rangle ::= y_1 \quad (\text{A})$
| $y_2 \quad (\text{B})$
| $y_3 \quad (\text{C})$
| $y_4 \quad (\text{D})$
| $0,5 \quad (\text{E})$
| $1 \quad (\text{F})$
| $1,5 \quad (\text{G})$
| $2 \quad (\text{H})$

Bagaimana dengan GE?

- Grammar tersebut dibuat dengan asumsi bahwa model-model peramalan hanya memperhatikan maksimum 4 data masukan, y_1 sampai y_4 . Variabel a_0 sampai a_4 diasumsikan hanya bisa bernilai 0,5; 1; 1,5 atau 2. Dengan grammar di atas, kromosom-kromosom pada GE bisa menghasilkan model-model peramalan yang sangat bervariasi, seperti:

$$z = 0,5 + 2y_1 - y_2 + (0,5 * y_3)$$

atau

$$z = 1,5 + \sin(y_1) - \log(y_2)$$

atau

$$z = \frac{0,5 + y_1}{(y_2 + (0,5 * y_3) - \cos(y_4))}$$

Fungsi *Fitness*

$$f = \frac{1}{(K + b)}$$

dimana b merupakan suatu bilangan yang dianggap sangat kecil untuk menghindari pembagian dengan 0, sedangkan K adalah rata-rata kesalahan peramalan untuk semua data penjualan.

Fungsi *Fitness*

Kesalahan peramalan merupakan harga mutlak dari selisih hasil peramalan menggunakan model tersebut (z) dengan data penjualan yang sebenarnya (z^*). Dengan demikian, rata-rata kesalahan peramalan dapat dituliskan sebagai

$$K = \frac{1}{N} \sum_{i=1}^N |z_i - z_i^*|$$

dimana N adalah jumlah semua data peramalan.

Traveling Salesman Problem

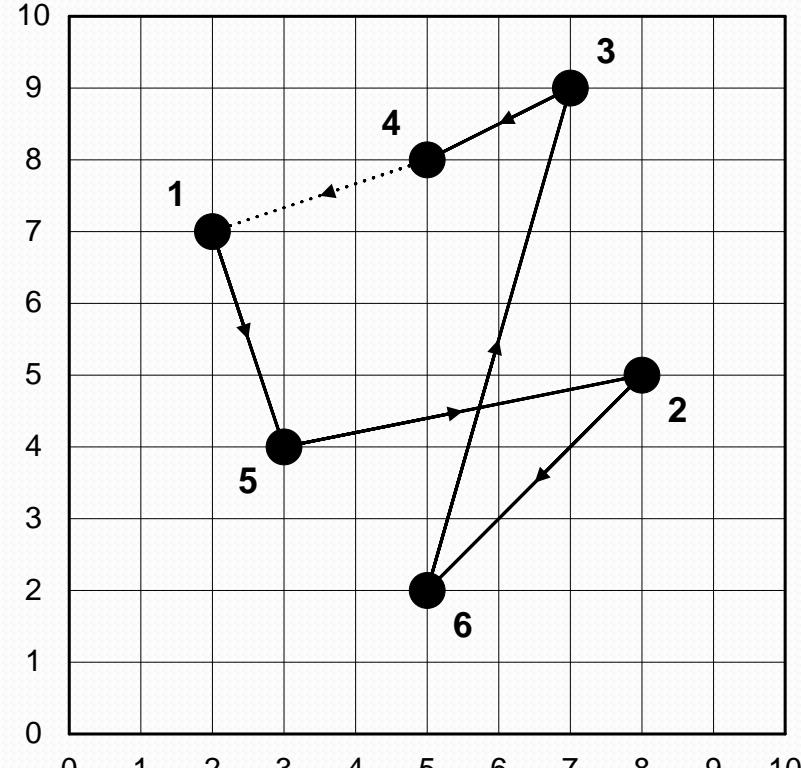
- Bagaimana menemukan urutan kunjungan
- Solusi → permutasi dengan biaya terendah
- Termasuk masalah Diskrit
- Algoritma EAs yang bisa dipakai?
- GA ?
- ES, EP & DE ??
- GP & GE ???

TSP dengan GA

Fungsi fitness?

1 / total biaya

Individu: urutan kunjungan semua lokasi



Nilai gen menyatakan nomor lokasi

Posisi gen menyatakan urutan kunjungan

TSP dengan ES, EP dan DE

- Konversi kontinyu menjadi diskrit
- Variabel strategies
- Komputasi vektor pada DE

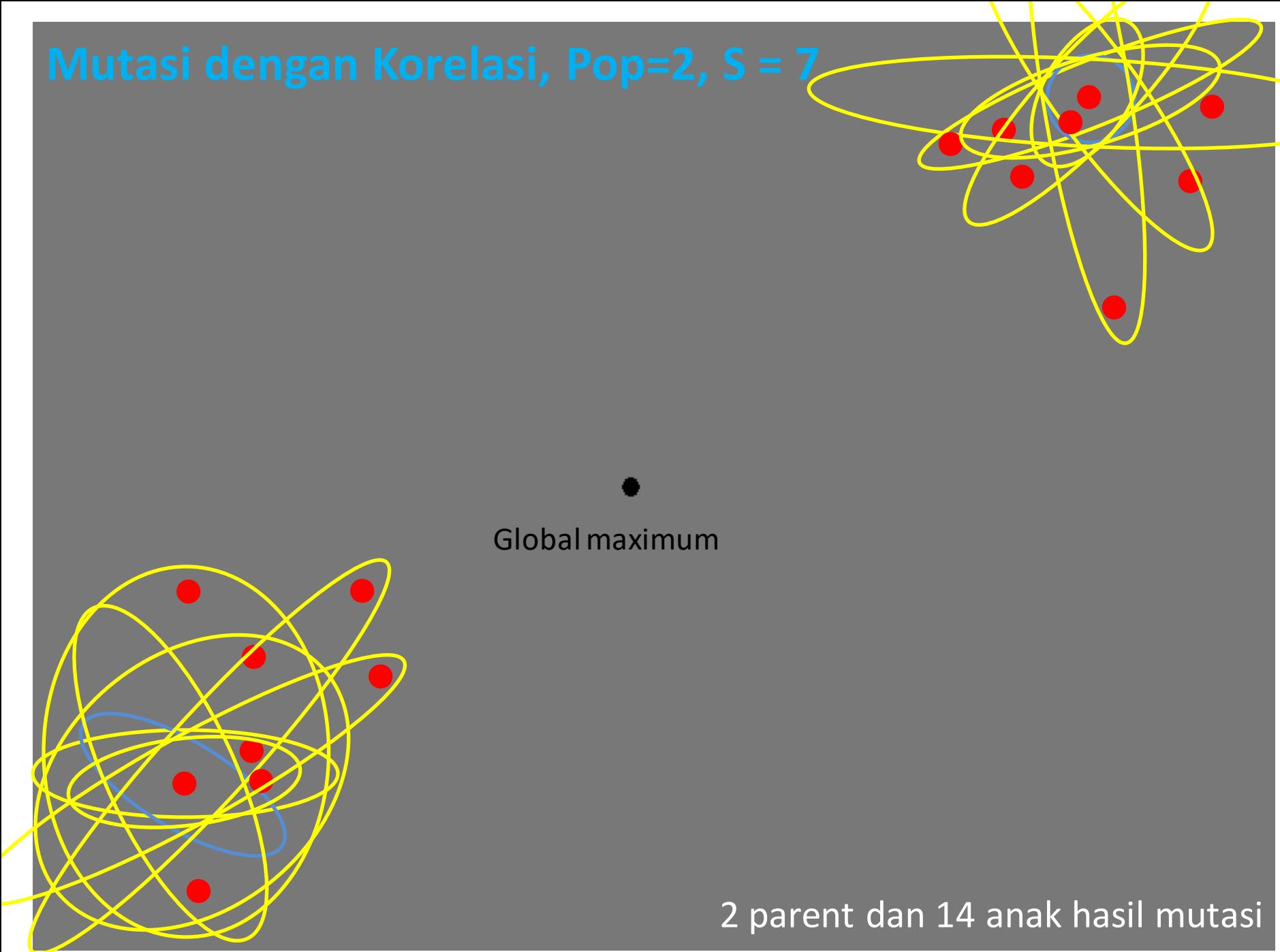
Kromosom

Individu: $x_1 = -3,2170$ dan $x_2 = 2,7531$

Kromosom:

x_1	x_2	Tau_1	Tau_2	$Alfa_1$
-3,2170	2,7531	0,0132	0,0027	0,0032

Mutasi dengan Korelasi, Pop=2, S = 7



x_2

x_1

- Sejumlah NP vektor parameter pada generasi G
- Vektor parameter \underline{v} baru yang dihasilkan
- ★ Nilai minimum global yang dicari

$$F.(x_{r2,G} - x_{r3,G})$$

minimum

$x_{i,G}$

$x_{r3,G}$

$x_{r2,G}$

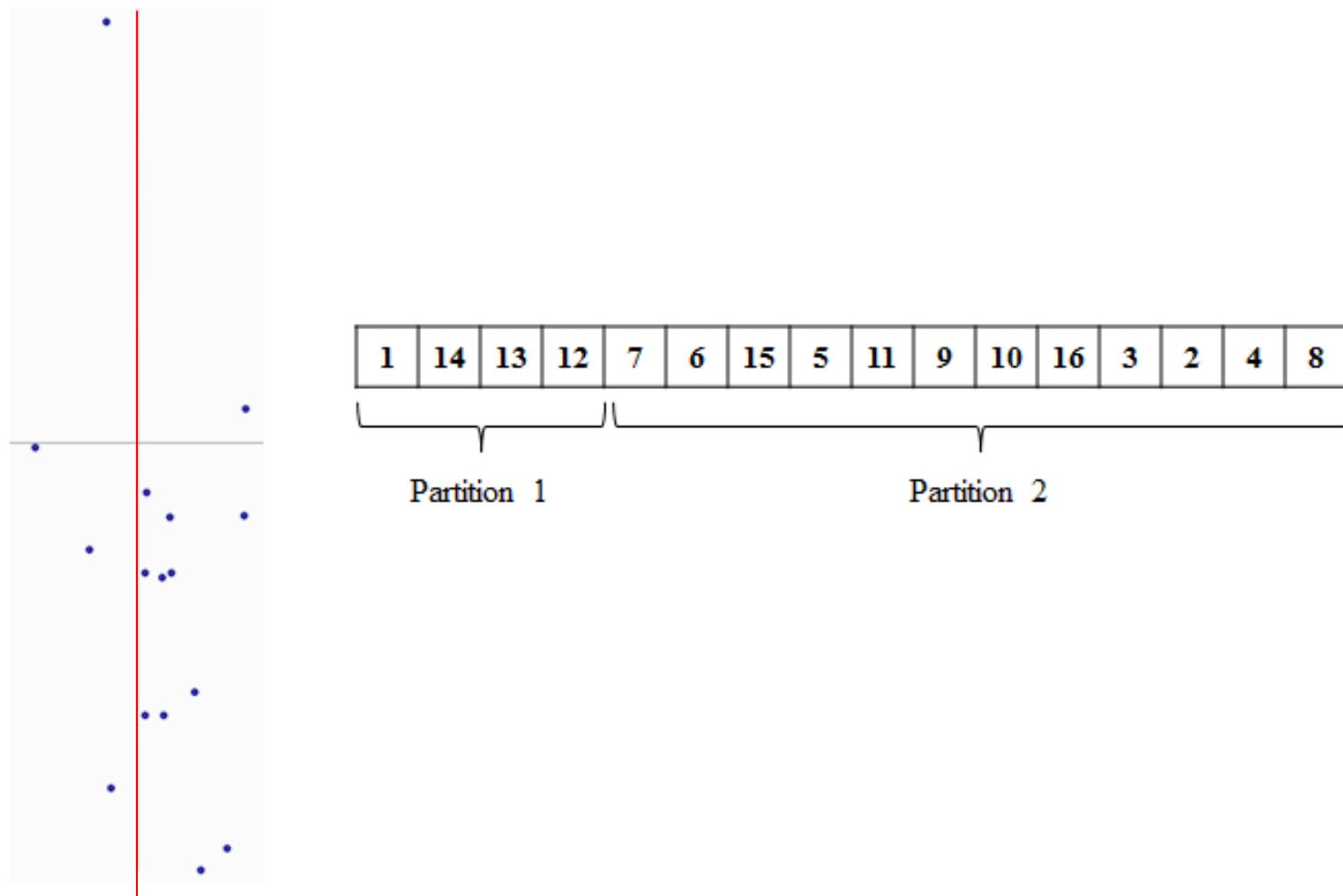
x_{best}

$$\underline{x}_{i,G} + \lambda . (\underline{x}_{best} - \underline{x}_{i,G})$$

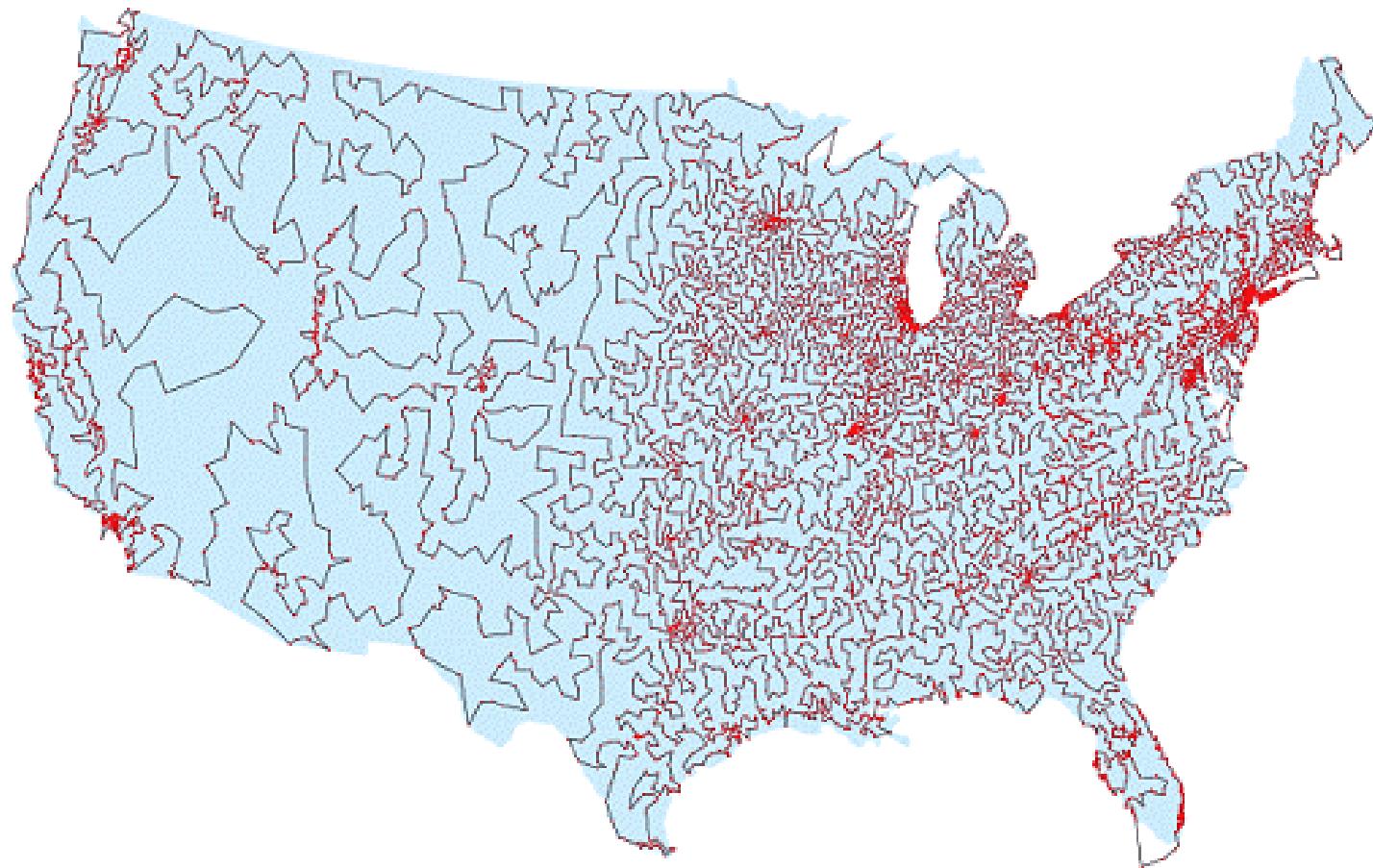
$$\underline{v} = \underline{x}_{i,G} + \lambda . (\underline{x}_{best} - \underline{x}_{i,G}) + F.(x_{r2,G} - x_{r3,G})$$

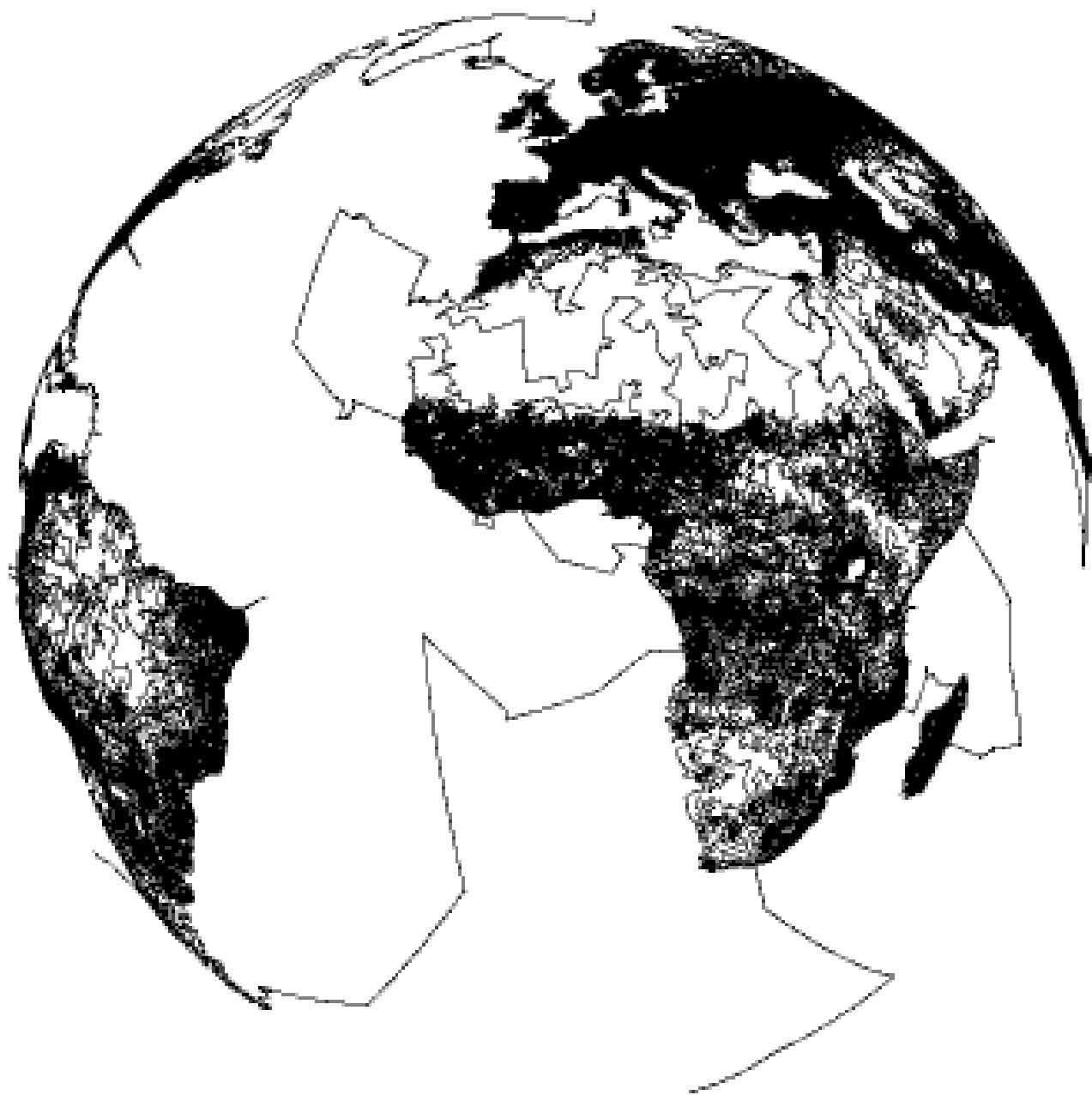
x_1

Two sub partitions by Y axis



13,509 cities in the US





1,904,711-city instance of locations, <http://www.tsp.gatech.edu/world/images/world.anim5a.gif>

TSP dengan batasan

- *Traveling Salesperson Problem* (TSP) adalah suatu masalah optimasi kombinatorial.
- Pada kasus tertentu, mungkin saja terdapat batasan untuk kombinasi. Misalkan, kita ingin menemukan rute penerbangan dengan total jarak paling pendek untuk 60 kota besar di tiga benua berbeda, yaitu 20 kota di *North America*, 20 kota di *Europe*, dan 20 kota di *Sout-East Asia* (Jepang, Cina, Korea Selatan, dan Taiwan) [SUY05b].
- Untuk masalah ini, misalkan rute yang valid adalah rute yang pada setiap benua hanya mengunjungi maksimum dua kota (tidak boleh lebih).

Daftar 60 kota di tiga benua berbeda dengan posisi *Latitude* dan *Longitudenya*

No.	Nama kota	Latitude	Longitude
1	Amarillo	35.207	- 101.834
2	Atlanta	33.752	- 84.393
3	Berkeley	37.869	-122.271
4	Boston	42.356	-71.056
5	Buffalo	42.881	-78.872
...
60	Tianjin	39.13	117.20

Inisialisasi populasi

- Untuk membangkitkan individu yang valid, kita bisa memilih secara acak satu atau dua kota dari 60 kota yang ada menggunakan dua aturan berikut ini:
 - Setelah meletakkan satu atau dua kota dari suatu benua, pilih secara acak satu atau dua kota dari benua yang lain.
 - Jumlah kota (satu atau dua kota) dari suatu benua dipilih berdasarkan suatu probabilitas, misalkan P_k , yang nilainya dalam interval $[0, 1]$. Jumlah kota yang dipilih adalah satu jika bilangan acak yang dibangkitkan adalah lebih kecil dari P_k . Tetapi, jika bilangan acak yang dibangkitkan adalah lebih besar atau sama dengan P_k , maka jumlah kota yang dipilih adalah dua.

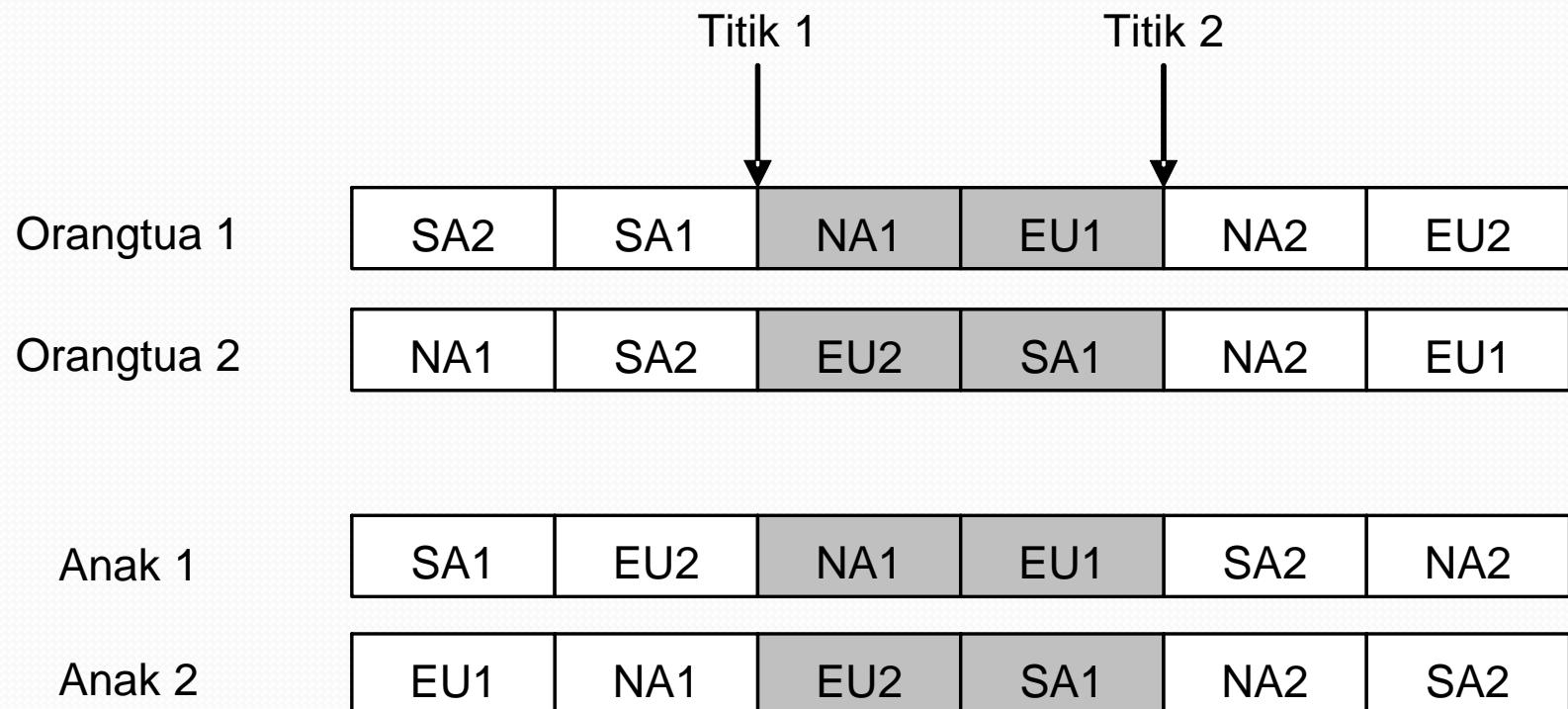
Inisialisasi populasi

	Gen 1	Gen 2	Gen 3	Gen 4	Gen 5	Gen 6		Gen 59	Gen 60
a	NA1	EU5	SA3	NA3	EU1	SA2	...	EU11	SA9
b	NA1	NA6	EU1	EU5	SA5	SA1	...	SA7	SA12
c	NA3	NA5	SA1	EU3	NA2	NA7	...	EU4	EU7

Rekombinasi

- Untuk masalah TSP dengan batasan ini, kita bisa menggunakan rekombinasi untuk representasi permutasi, seperti: *order crossover*, *partially mapped crossover*, *cycle crossover*, maupun *edge recombination*.
- Tetapi, diperlukan sedikit modifikasi agar anak-anak yang dihasilkan tetap valid. Misalkan kita menggunakan *order crossover*. Modifikasi seperti apa yang harus kita lakukan?
- Kita bisa memberikan batasan bahwa peletakkan kota harus mengikuti aturan seperti pada inisialisasi.

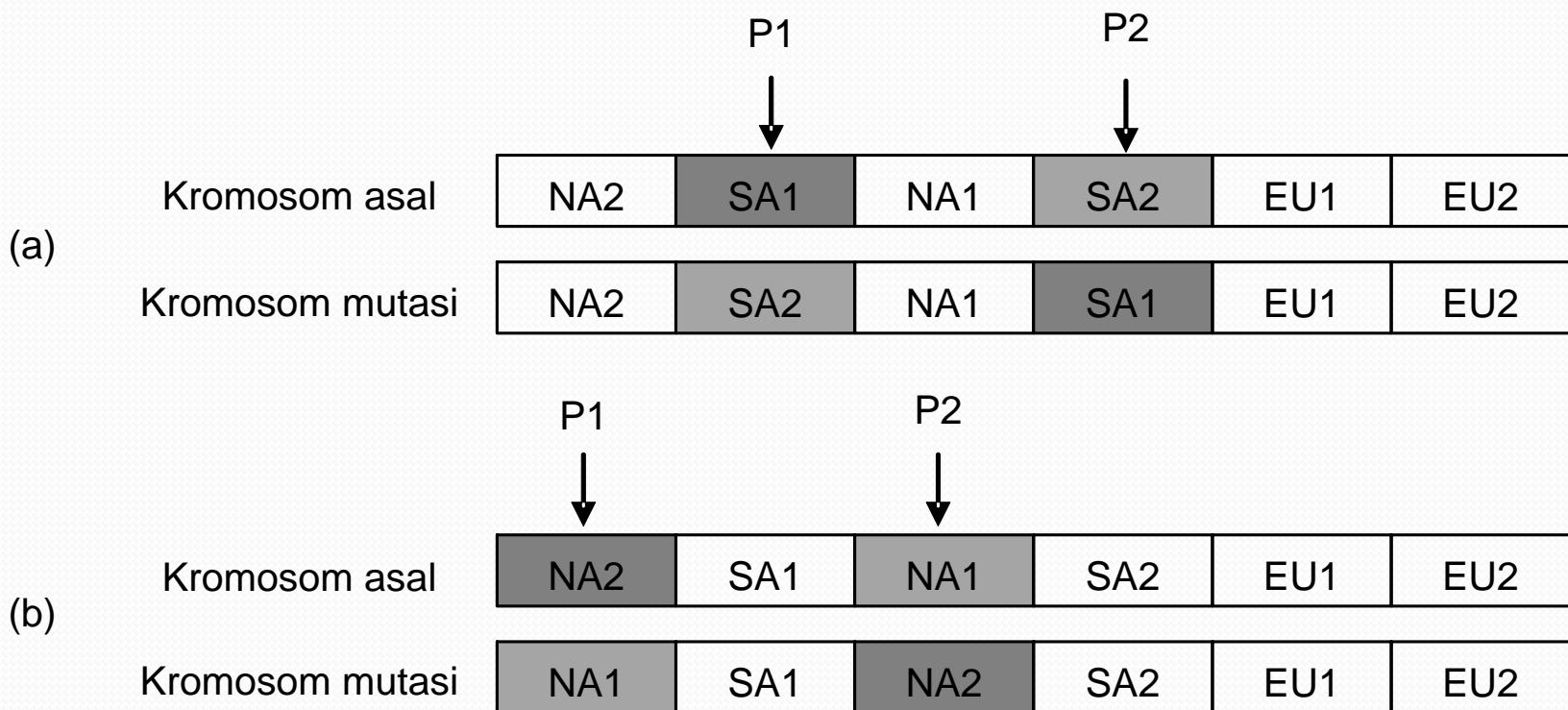
Rekombinasi



Mutasi

- Untuk operator mutasi, kita bisa menggunakan *swap mutation* yang diberikan sedikit batasan, yaitu jika dua gen yang dipertukarkan harus berupa kota yang berasal dari benua yang sama.
- Artinya, jika gen pertama yang terpilih adalah kota dari benua SA, maka gen kedua yang akan dipertukarkan harus berupa kota yang berasal dari SA juga.

Mutasi



Fungsi *fitness*

- Jarak tempuh antara dua kota dihitung berdasarkan perhitungan jarak yang disebut *the great circle distance* (untuk lebih detail, silahkan lihat di situs www.zipcodeworld.com):

$$d = 69.1 \left(\frac{180}{\pi} \right) \arccos[\sin(Lat_1)\sin(Lat_2) + \dots \\ \cos(Lat_1)\cos(Lat_2)\cos(Long_2 - Long_1)]$$

dimana d adalah jarak antara dua kota dalam km .

Fungsi *fitness*

Karena masalahnya adalah minimasi, maka fungsi *fitness*-nya adalah:

$$f = \frac{1}{c(d)}$$

Mengapa tidak menggunakan bilangan kecil a ?
Karena total biaya perjalanan tidak mungkin 0.

Penjadwalan kuliah

- Penjadwalan kuliah merupakan suatu masalah yang sangat kompleks.
- Misalnya, suatu fakultas di sebuah universitas memiliki 36 ruang kuliah yang terdiri dari 24 ruangan besar dengan kapasitas di atas 50 mahasiswa dan 12 ruangan kecil dengan kapasitas 30 mahasiswa.
- Misalkan, setiap semester terdapat 400 pertemuan kuliah (setiap pertemuan membutukan waktu 2 jam) yang harus ditentukan jadwal kuliah dan ruangannya dalam satu minggu (Senin sampai Sabtu).

Penjadwalan kuliah

- Hari Senin sampai Kamis dan hari Sabtu, jam kuliah adalah 07.00-15.00 (8 jam). Hari Jum'at, jam kuliah adalah 07.00-11.00 dan 13.00-15.00 (6 jam). Sehingga jumlah jam kuliah dalam satu minggu adalah 46 jam. Karena setiap pertemuan kuliah membutuhkan waktu 2 jam, maka dalam satu minggu terdapat maksimum 23 pertemuan.
- Status dosen dibedakan menjadi dua, yakni dosen Dalam dan dosen Luar Biasa (LB). Seorang dosen mungkin mengajar lebih dari satu kelas kuliah.

Penjadwalan kuliah

Batasan-batasan yang digunakan:

- Tidak boleh terjadi bentrok, baik waktu maupun ruangan, untuk dosen maupun kelas kuliah yang sama.
- Dosen Dalam maupun dosen LB yang mempunyai jabatan akademik boleh meminta jadwal kuliah, dengan aturan bahwa dosen dengan jabatan akademik Guru Besar dan Lektor Kepala lebih diprioritaskan tiga kali lebih besar dibanding dosen dengan jabatan akademik Lektor dan Asisten Ahli. Jadwal yang dihasilkan se bisa mungkin dapat memenuhi permintaan tersebut.

Penjadwalan kuliah

Batasan-batasan yang digunakan:

- Jadwal kuliah untuk dosen Dalam, diutamakan hari Senin sampai Kamis dengan nilai prioritas empat kali lebih besar dibanding hari Jum'at dan Sabtu. Sedangkan untuk dosen LB, diutamakan hari Jum'at dan Sabtu dengan nilai prioritas empat kali lebih besar dibanding hari Senin sampai Kamis. Kecuali atas permintaan dosen yang bersangkutan (lihat aturan b di atas).
- Ruang kuliah yang besar diutamakan untuk kelas kuliah yang pesertanya lebih dari 30 mahasiswa. Nilai prioritasnya adalah empat kali lebih besar dibanding kelas kuliah yang pesertanya kurang dari 30 mahasiswa.

Penjadwalan kuliah

- Bagaimana menyelesaikan tersebut?
- Pertama kita lihat dulu seberapa besar ruang masalahnya? Dari 36 ruangan, 400 kelas kuliah, dan 23 pertemuan kuliah per minggu, maka jumlah solusi yang mungkin adalah sebanyak $400^{(23 \times 36)}$.
- Suatu jumlah yang sangat besar jika harus diselesaikan dengan algoritma konvensional.
- Misalkan kita ingin menyelesaikan masalah ini menggunakan EAs. Dua hal yang harus kita definisikan adalah: representasi individu dan fungsi *fitness*-nya.

Representasi individu

Basis data Pertemuan Kuliah

No	Kode MK	Kode Kelas	Kuota mhs	Kode Dosen	Waktu	Ruangan
1	CS3143	IF-33-01	40	SBK		
2	CS2315	IF-34-01	35	TWG		
...						
400	CS4923	IF-32-03	17	DSS		

Representasi kromosom

Ruang 1

	SN	SL	RB	KM	JM	SB
07-09	50		7	112	187	
07-09	32	1	101		400	
11-13		3		9		
13-15				6		
15-17	43	21	29	332		15

Gen ke-1

Ruang 36

	SN	SL	RB	KM	JM	SB
07-09	51		71	103		
07-09	35	10	100			
11-13	4	31		200		
13-15				2		98
15-17			19			25

Gen ke-36

...

Kromosom dengan 36 gen:

Ruang 1

	SN	SL	RB	KM	JM	SB
07-09	50		7	112	187	
07-09	32	1	101		400	
11-13		3		9		
13-15				6		
15-17	43	21	29	332		15

Gen ke-1

The diagram consists of three labels arranged vertically: "Slot 1" at the top, "Slot 2" in the middle, and "Slot 3" at the bottom. Each label is preceded by a black arrow pointing downwards towards it.

Ruang 36

	SN	SL	RB	KM	JM	SB
07-09	51		71	103		
07-09	35	10	100			
11-13	4	31		200		
13-15				2		98
15-17			19	/		25

Gen ke-36

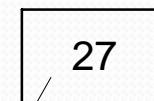
Slot 818

Slot 828

Kromosom dengan 400 gen:



Gen 1 Gen 2 Gen 3

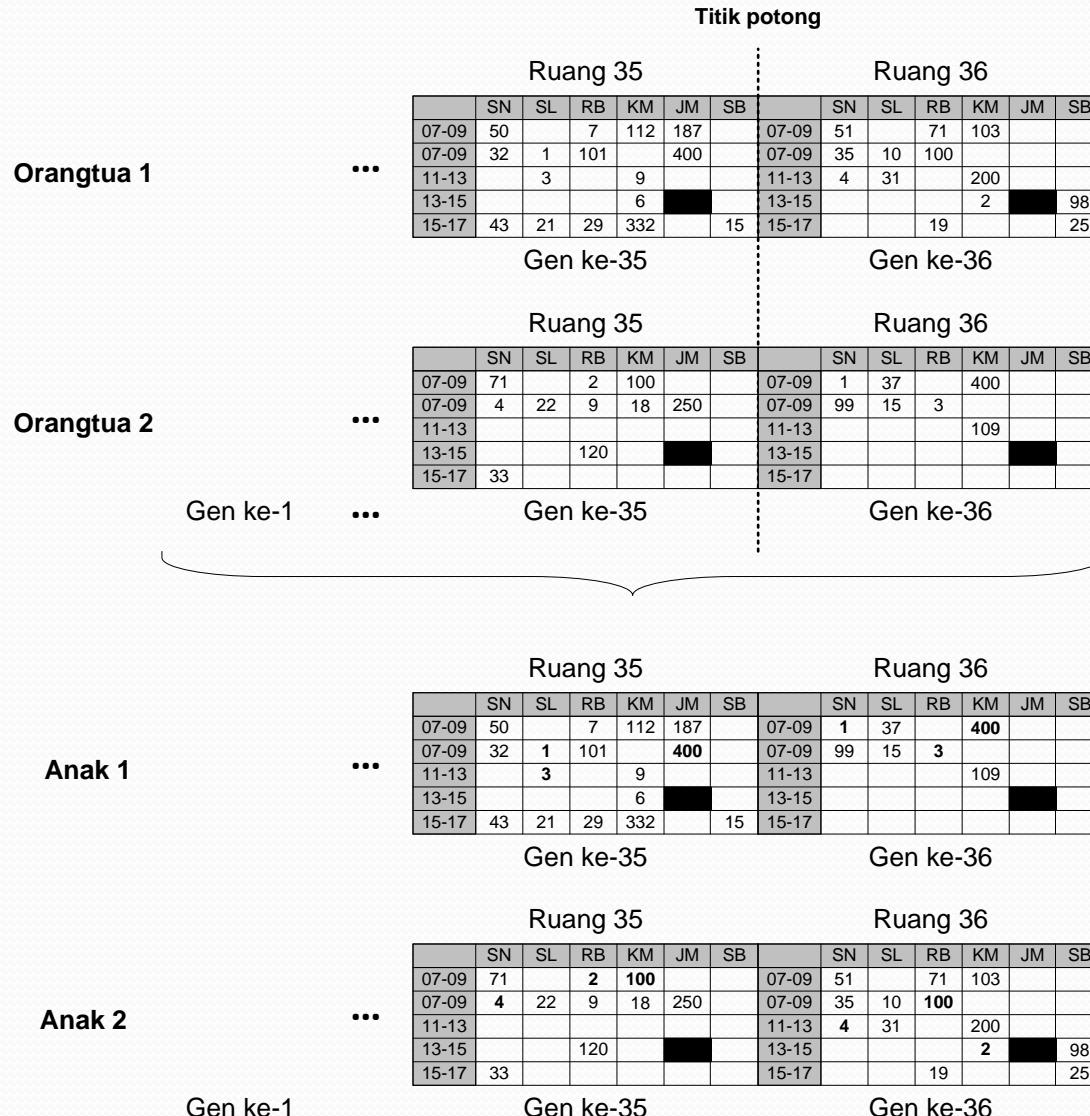


Gen 400

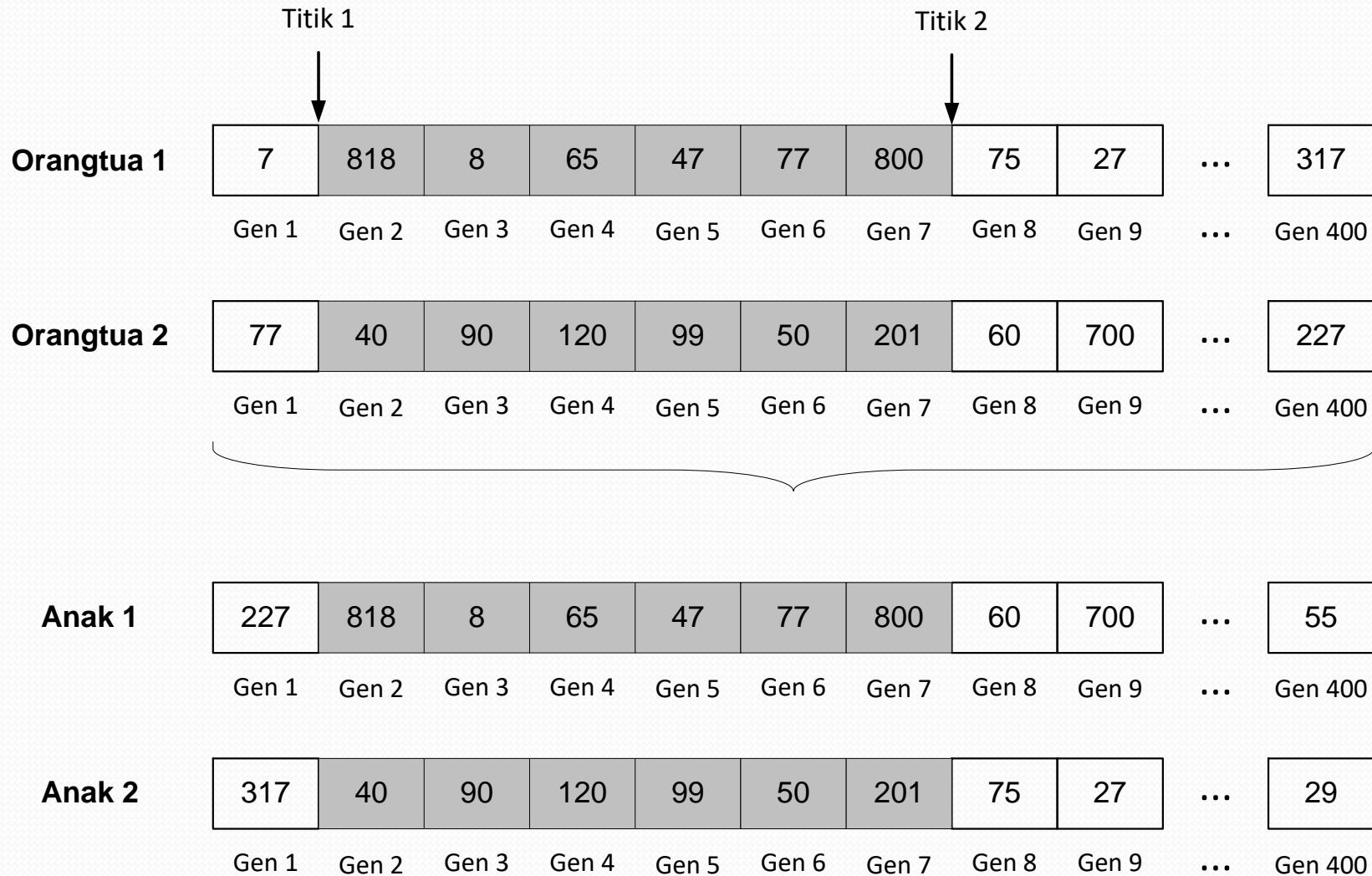
Posisi gen menyatakan nomor urut pertemuan kuliah

Nilai-nilai gen
menyatakan nomor slot

Rekombinasi 1



Rekombinasi 2



Mutasi 1

Kromosom asal

	SN	SL	RB	KM	JM	SB
07-09	50		7	112	187	
07-09	32	1	101		400	
11-13		3		9		
13-15				6		
15-17	43	21	29	332		15

Gen ke-1

Slot 1

Ruang 1

Ruang 36

	SN	SL	RB	KM	JM	SB
07-09	51		71	103		
07-09	35	10	100			
11-13	4	31		200		
13-15				2		98
15-17			19			25

Gen ke-36

...

	SN	SL	RB	KM	JM	SB
07-09	51		71	103		
07-09	35	10	100			
11-13	4	31		400		
13-15				2		98
15-17			19			25

Ruang 36

...

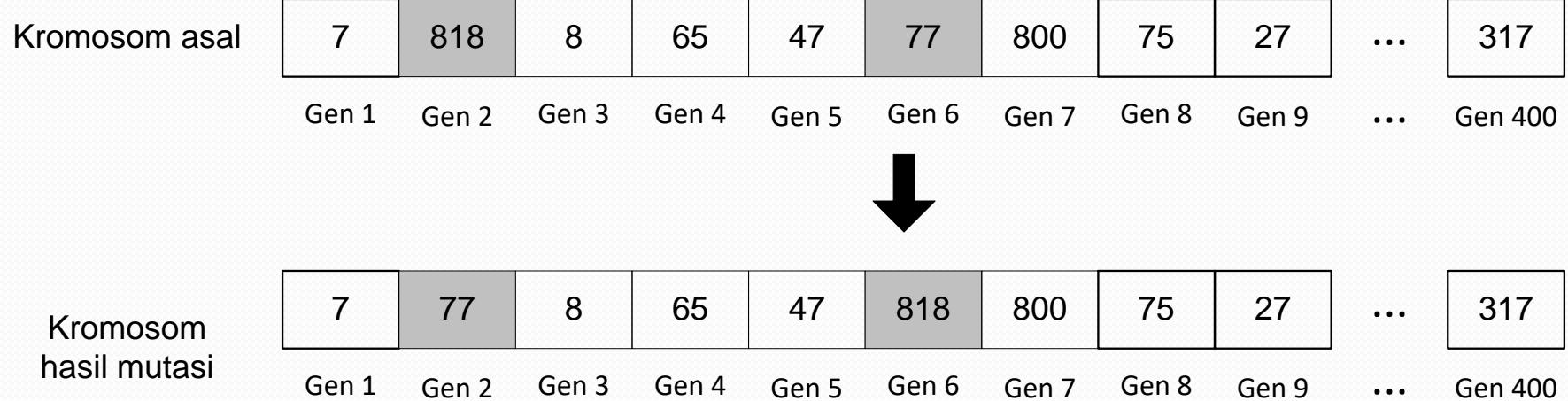
Kromosom hasil mutasi

	SN	SL	RB	KM	JM	SB
07-09			7	112	187	
07-09	32	1	101		200	
11-13		3		9		
13-15			50	6		
15-17	43	21	29	332		15

Gen ke-1

Slot 14

Mutasi 2 (Swap)



Fungsi *fitness*

Batasan	Penalti	Alasan
Tidak bentrok	24	
Permintaan Guru Besar dan Lektor Kepala	12	
Permintaan Lektor dan Asisten Ahli	4	
Status Dosen	4	
Ruang Kuliah	4	Tidak bentrok merupakan suatu <i>hard constraint</i> (batasan keras) yang sebaiknya tidak dilanggar. Sehingga batasan ini lebih utama dibandingkan harus memenuhi permintaan dosen berjabatan akademik Guru Besar dan Lektor Kepala atau batasan-batasan yang lain. Misal batasan ini diberikan nilai prioritas sebesar 24. Kemudian, batasan permintaan dosen berjabatan akademik Guru Besar dan Lektor Kepala diberi nilai prioritas sebesar 12. Misalkan ketiga batasan yang lainnya diberikan nilai priors yang sama, yaitu 4.

Fungsi *fitness*

Pemberian Penalty atau pembobotan seperti pada tabel di atas belum tentu yang paling baik. Secara praktis, kita bisa melakukan observasi pembobotan tersebut selama melakukan *running GA*.

Fungsi *fitness*

$$f = \frac{1}{(B + a)}$$

$$B = J_a + J_b + J_c + J_d + J_e$$

- J_a = Jumlah jadwal pertemuan yang bentrok dikali Penalti (24),
- J_b = Jumlah jadwal yang tidak sesuai permintaan dosen berjabatan Guru Besar dan Lektor Kepala dikali Penalti (12),
- J_c = Jumlah jadwal yang tidak sesuai permintaan dosen Lektor dan Asisten Ahli dikali Penalti (4),
- J_d = Jumlah jadwal yang tidak sesuai ruang kuliahnya dikali Penalti (4)
- J_e = Jumlah jadwal yang Tidak Sesuai Status dosen Dalam atau Luar Biasa dikali Penalti (4).

Kesimpulan

- Sebelum memutuskan menggunakan EAs, sebaiknya suatu masalah dipahami secara benar.
- Keenam algoritma pada EAs memiliki karakteristik khusus yang harus diperhatikan dengan baik sehingga bisa memutuskan algoritma mana yang sebaiknya digunakan.

Daftar Pustaka

- [SUYo8] Suyanto, 2008, Evolutionary Computation: Komputasi Berbasis “Evolusi” dan “Genetika”, penerbit Informatika Bandung.
- [SUYo5b] Suyanto and Sanaullah, 2005, “The Traveling Salesperson Problem Using Evolutionary Algorithm”, technical report, Chalmers University of Technology, Sweden.