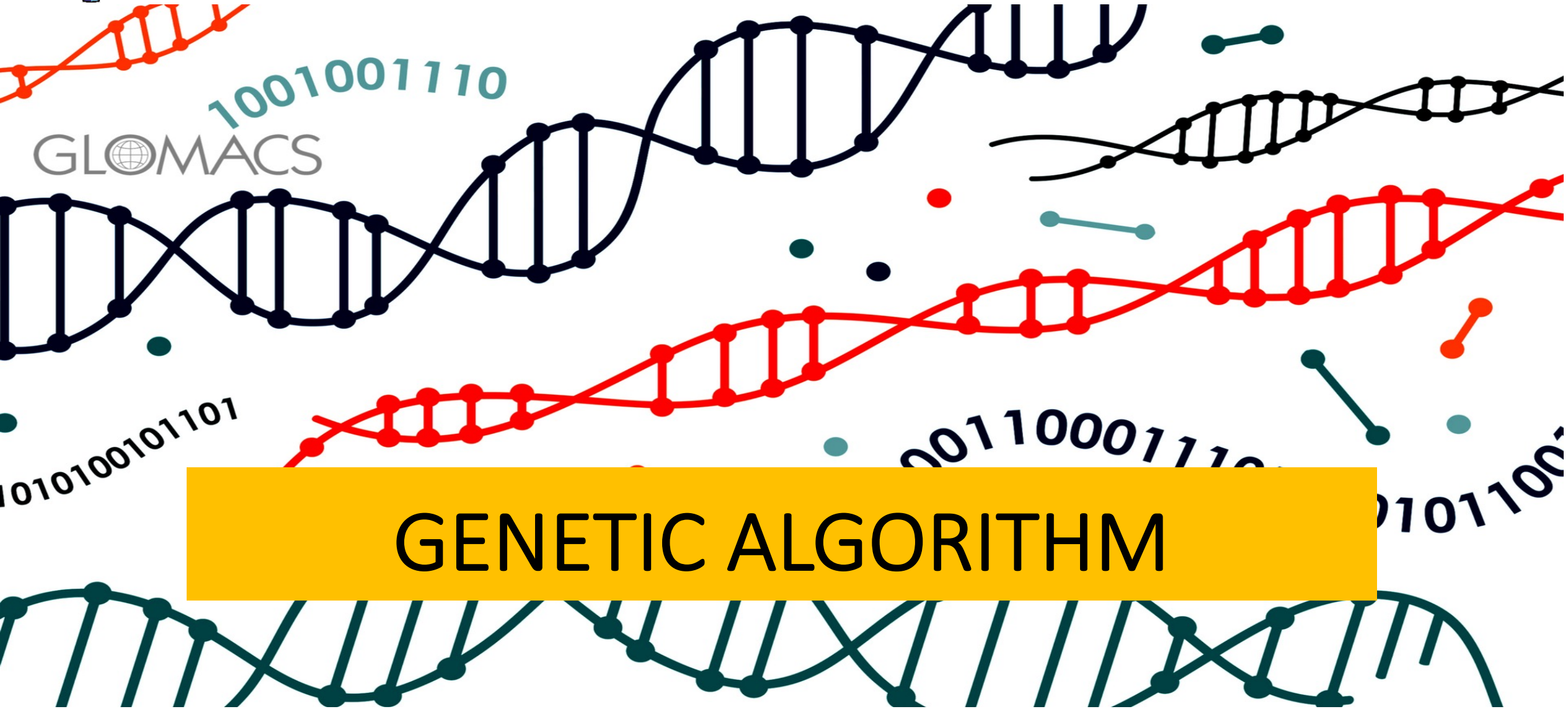
Local Search & Optimization Problem

Chastine Fatichah
Departemen Teknik Informatika
April 2023

Capaian Pembelajaran Matakuliah

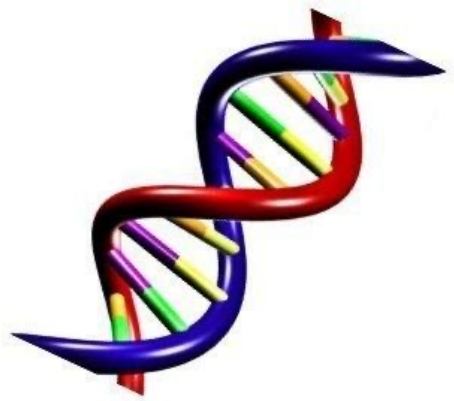
Mahasiswa mampu menjelaskan, mengidentifikasi, merancang, dan menerapkan *intelligent agent* untuk *problem* yang sesuai dengan memanfaatkan algoritma pencarian yang meliputi *uninformed search*, *informed search*, *heuristic search*, local search, *adversarial search*, serta algoritma *search* untuk *Constraint Satisfaction Problem*

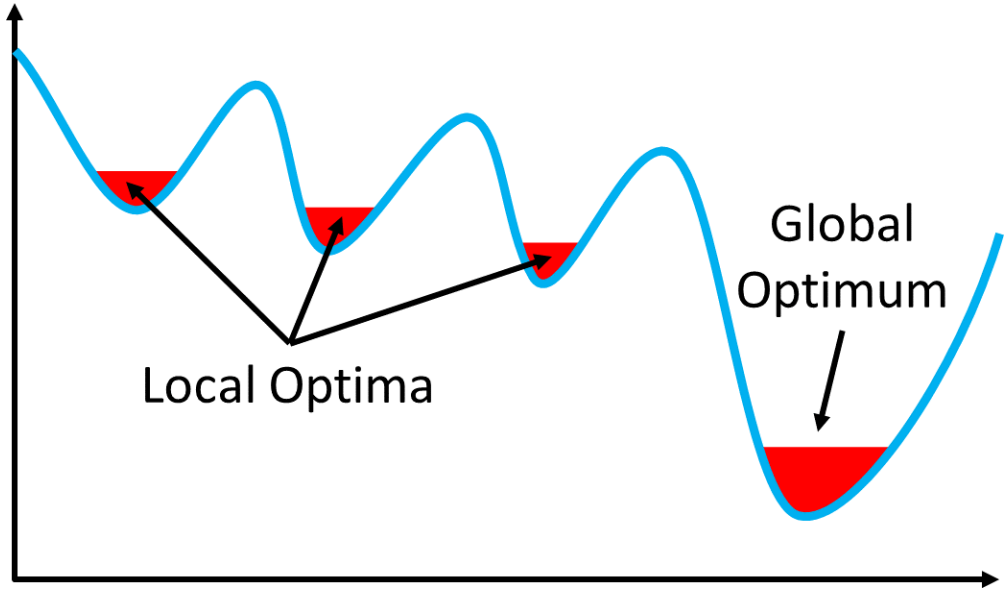


GENETIC ALGORITHM

Konsep Genetic Algorithm

- ***Genetic algorithm*** (**GA**) adalah salah satu teknik pencarian yang digunakan untuk menemukan atau mengaproksimasi solusi yang optimal.
- GA dikelompokkan sebagai *global search heuristics*
- GA merupakan salah satu *evolutionary algorithms* yang menggunakan teknik-teknik yang terinspirasi oleh *evolutionary biology* seperti *inheritance, mutation, selection, recombination*



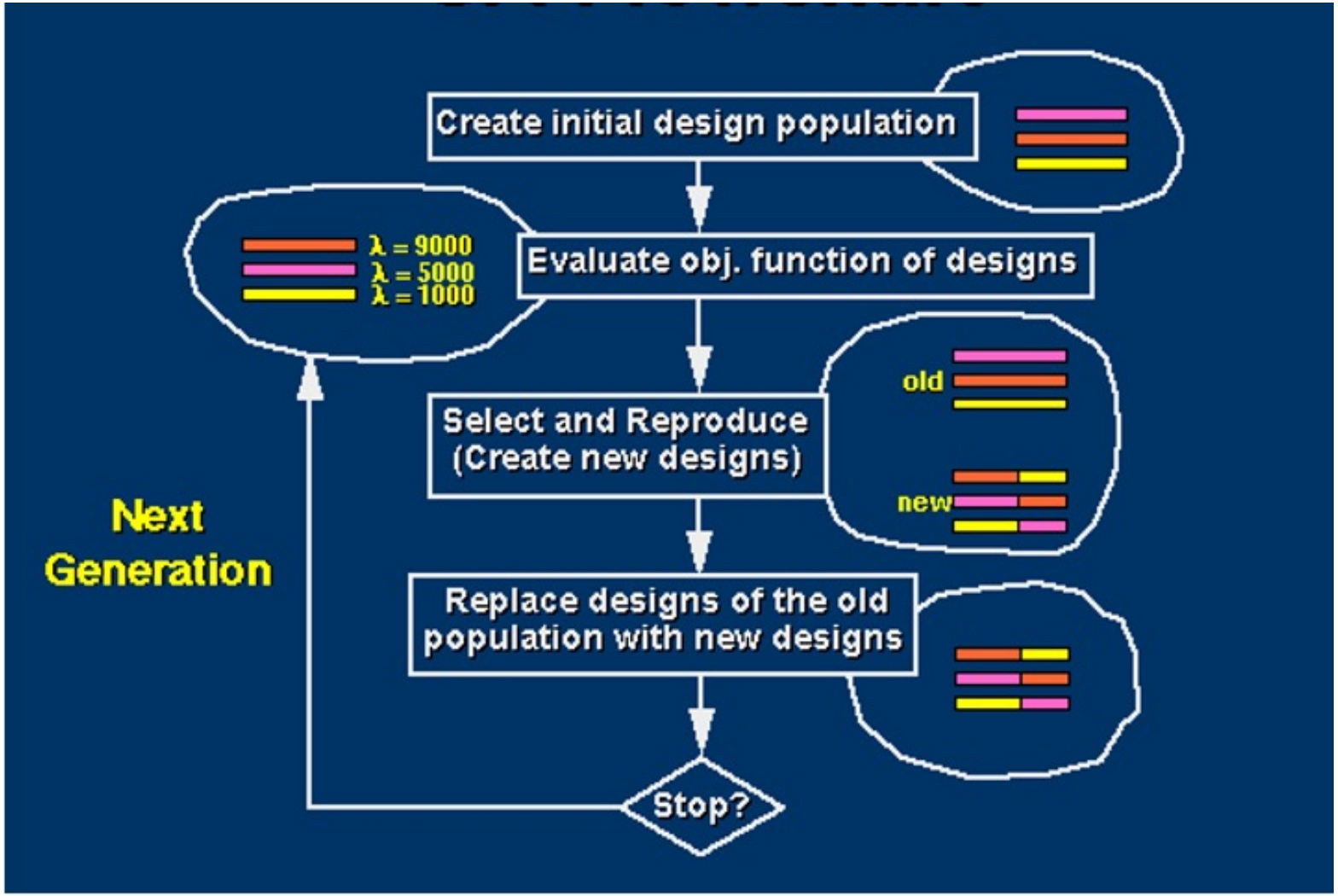


Konsep Genetic Algorithm

Secara umum algoritma genetika (GA) memerlukan dua hal yang harus didefinisikan yaitu:

1. Representasi *genetic* sebagai domain solusi
2. Fungsi *fitness* untuk mengevalusi domain solusi

Tahapan Genetic Algorithm



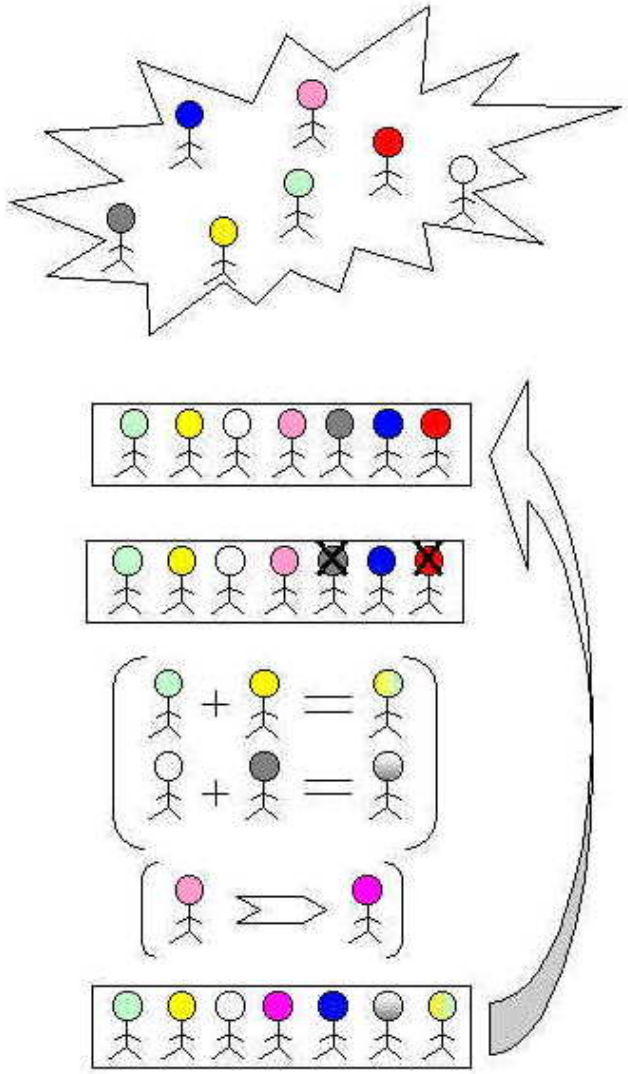
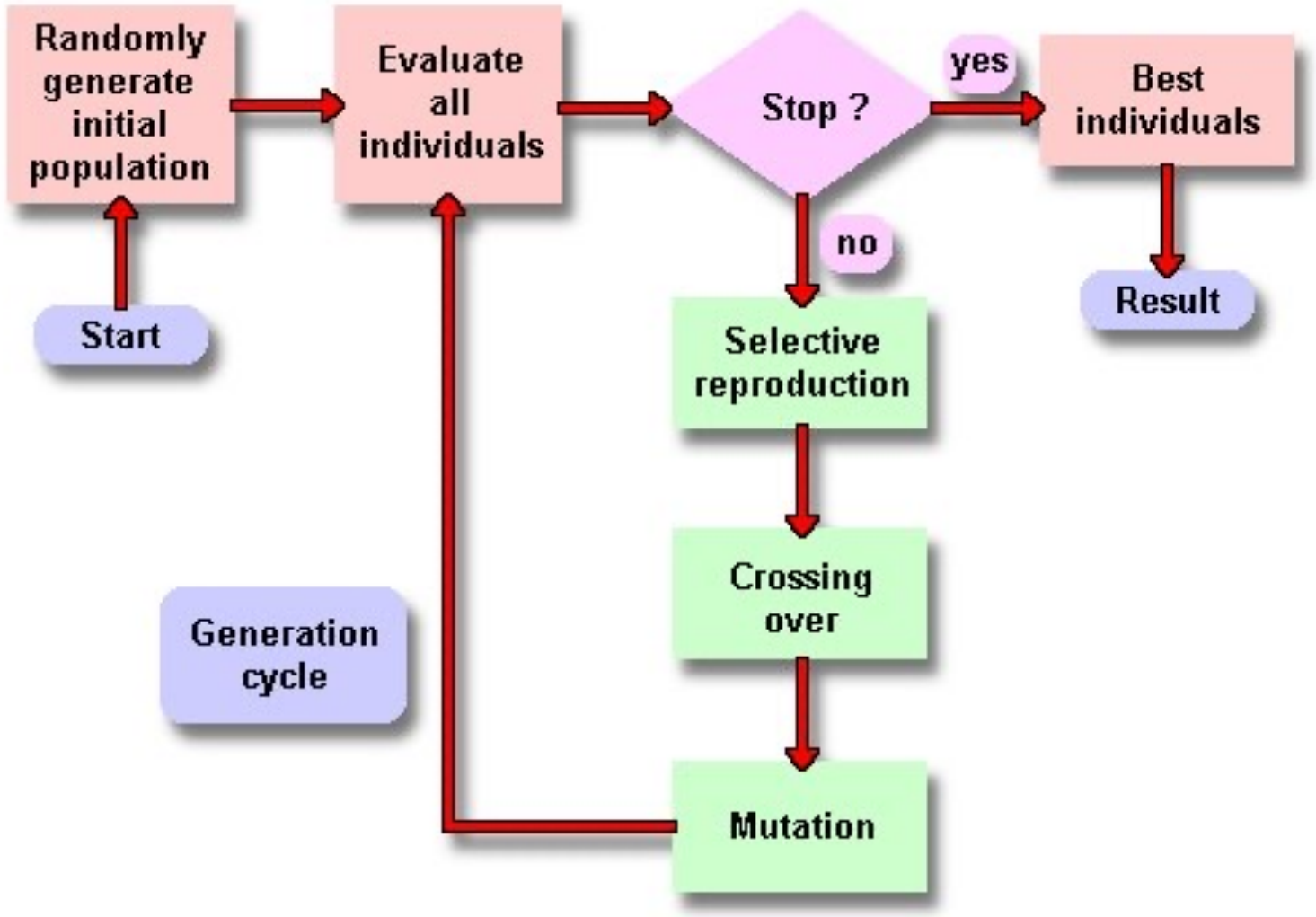


Diagram Alir Genetic Algorithm

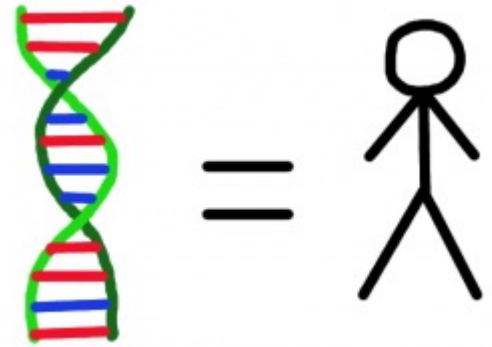


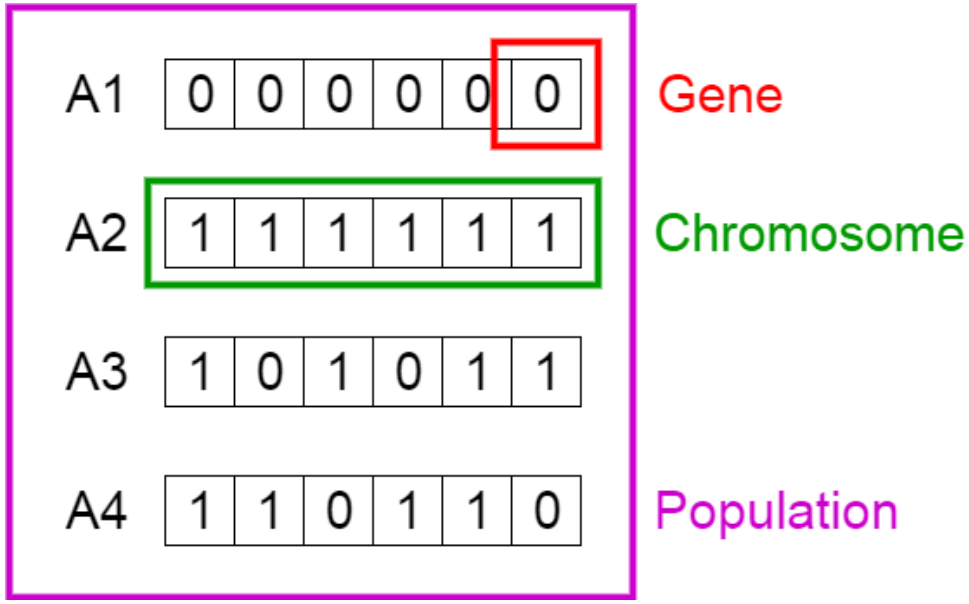
1. Mulai dengan "population" yang dibangkitkan secara random sebagai representasi domain solusi
2. Secara iteratif melakukan:
   - Evaluasi setiap domain solusi
   - Simpan solusi yang terbaik
   - Menggunakan domain solusi yang ada untuk mengenerate populasi baru
3. Berhenti ketika solusi sudah ditemukan (atau maksimum iterasi sudah tercapai)

Generate Initial Population

Chromosomes (individu) dapat berupa:

- Bit strings (0101 ... 1100)
- Real numbers (43.2 -33.1 ... 0.0 89.2)
- Permutations of element (E11 E7 ... E1 E15)
- Lists of rules (R1 R2 R3 ... R22 R23)
- ... any data structure ...





Evaluasi Solusi (Fungsi *Objective*)

- Fungsi *objective* pada umumnya direpresentasikan sebagai fungsi fitness
- Fungsi *fitness* merupakan hal yang penting pada algoritma genetika, karena menentukan kualitas dari *chromosome* (individu)
- Fungsi fitness umumnya diukur untuk mengetahui seberapa baik *chromosome* (individu) mencapai tujuan
  - Contoh pada kasus *Travelling Salesmen Problem* (TSP), maka fungsi *fitness* adalah biaya atau jarak yang dicapai dalam mengunjungi antar kota. Semakin kecil biaya atau jarak yang didapatkan semakin baik solusinya.

Selection

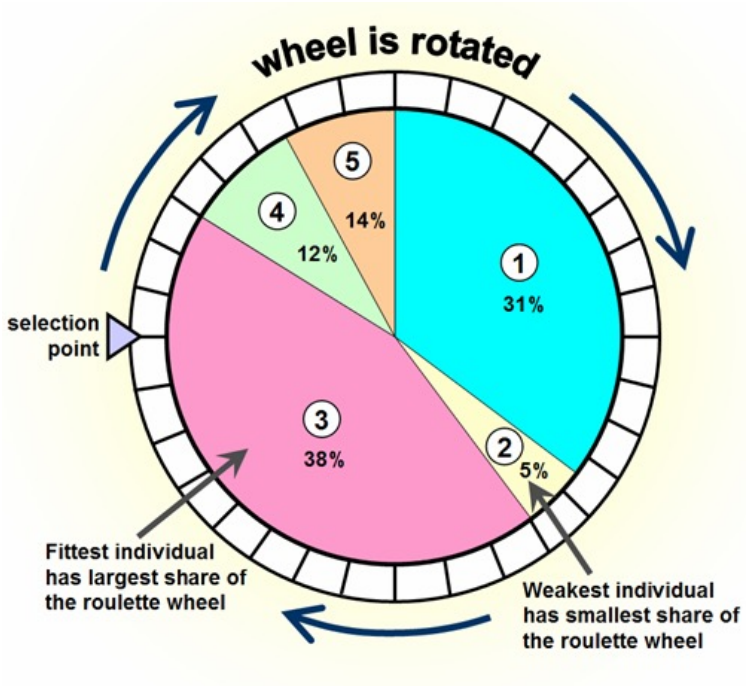
- *Selection* merupakan sebuah prosedur memilih *parent chromosome* untuk menghasilkan *off-spring* (keturunan)

- Tipe dari teknik *selection*:
  - ***Random Selection*** – pemilihan dilakukan secara random dari populasi
  - ***Proportional Selection*** – probabilitas pemilihan setiap *chromosome* (individu) dihitung berdasarkan:

$$P(\mathbf{x}_i) = f(\mathbf{x}_i)/\Sigma f(\mathbf{x}_j) \qquad \text{for all } j$$

  - ***Rank Based Selection*** – menggunakan perangkingan berdasarkan nilai *fitness*

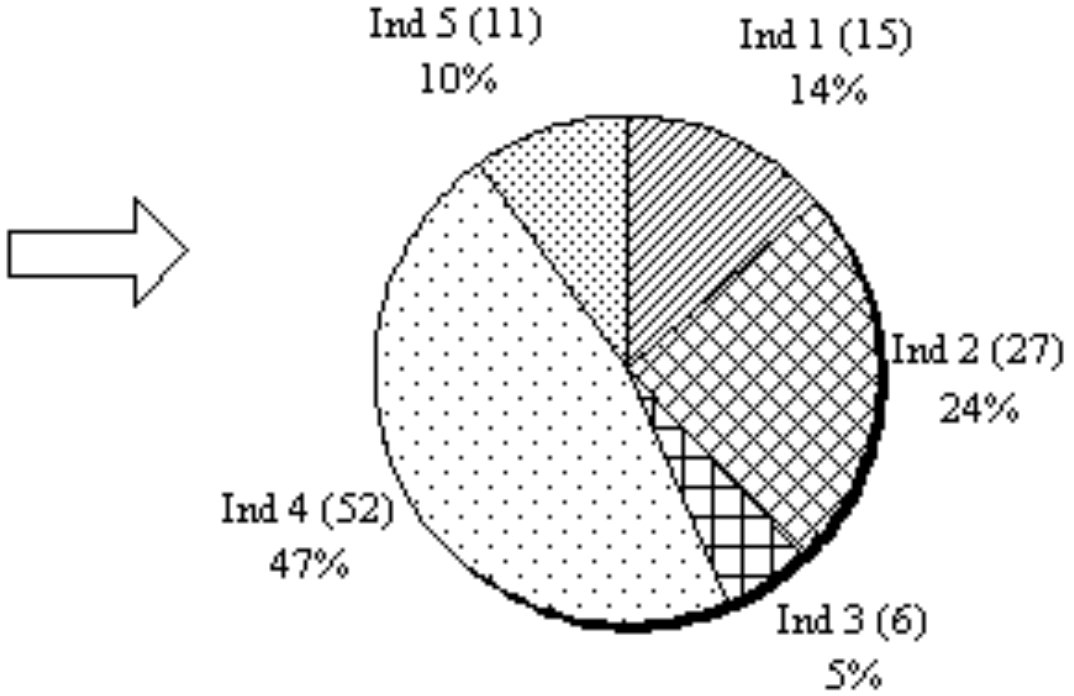
Roulette Wheel Selection



Let $i = 1$, where i denotes chromosome index;
Calculate $P(\mathbf{x_i})$ using proportional selection;
$sum = P(\mathbf{x_i})$;
choose $r \sim U(0,1)$;
while $sum < r$ **do**
    $i = i + 1$; i.e. next chromosome
    $sum = sum + P(\mathbf{x_i})$;
end
return $\mathbf{x_i}$ as one of the selected parent;
repeat until all parents are selected

Roulette Wheel Selection

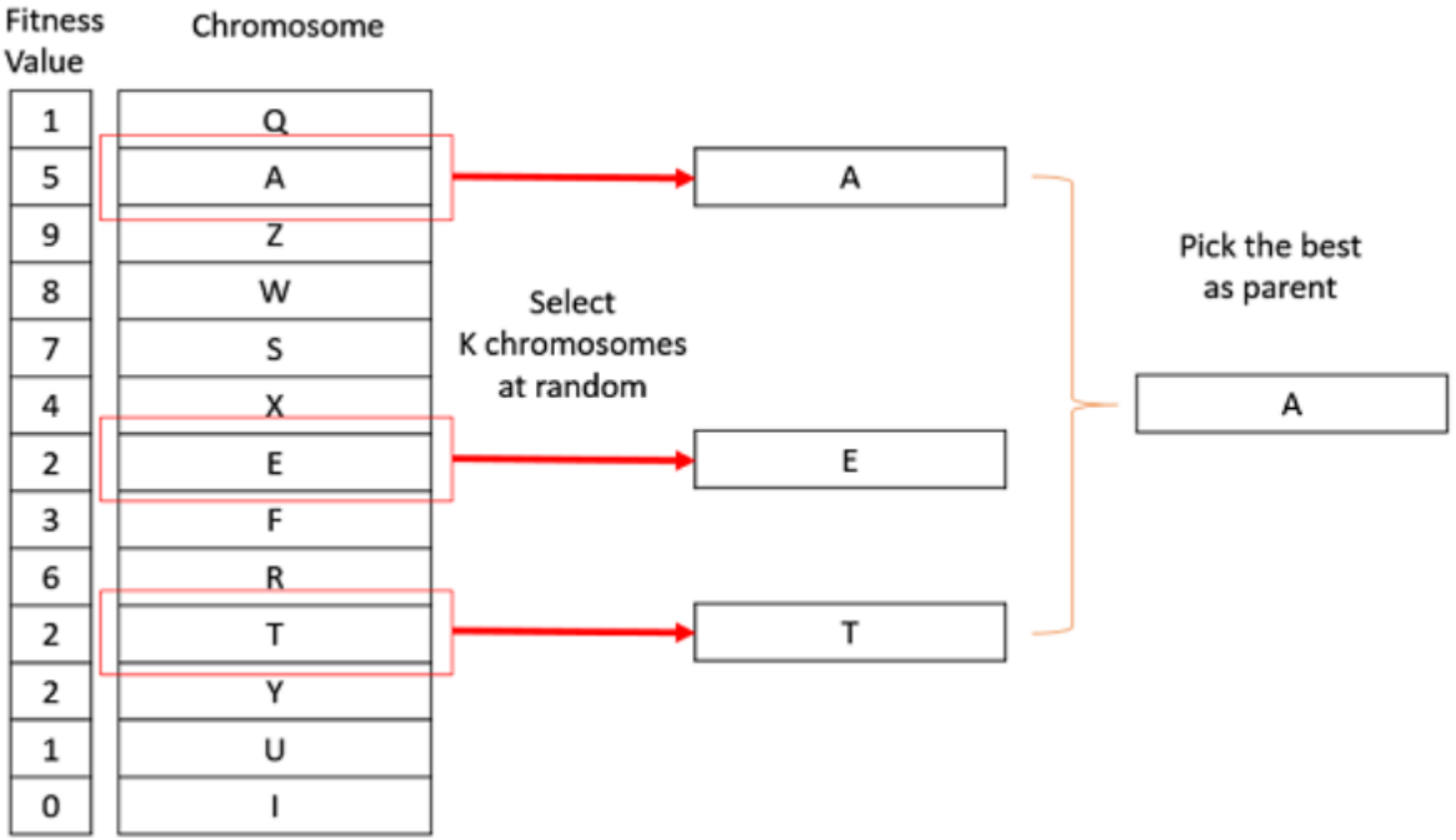
Population	*Fitness*
Individual 1	15
Individual 2	27
Individual 3	6
Individual 4	52
Individual 5	11



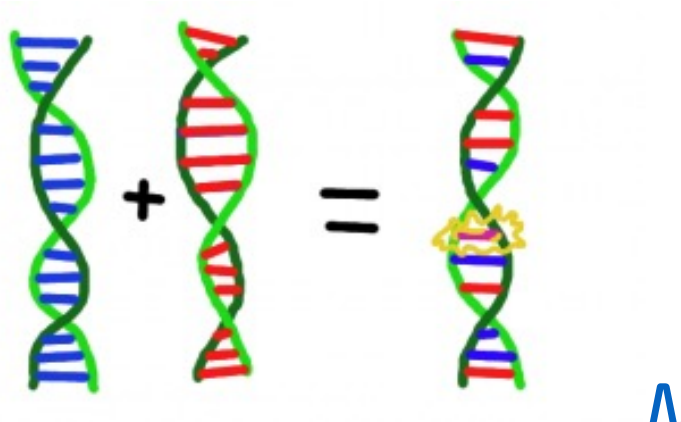
Randomly generated number = 21

Individual 2 is selected

Tournament Selection



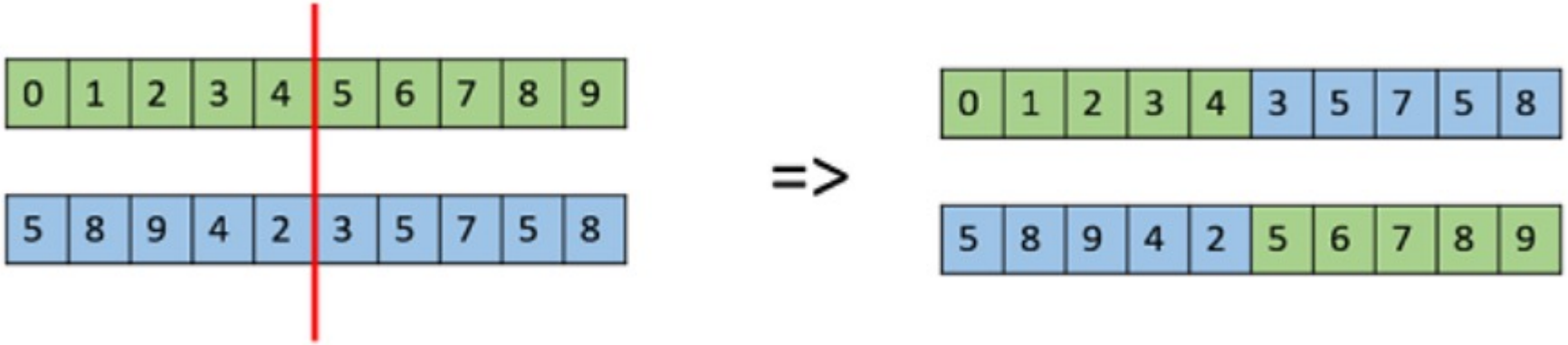
Reproduction



- *Crossover*
  - Dua *parent* menghasilkan dua *offspring*
  - Terdapat probabilitas *chromosome* dari dua *parent* direkombinasi (crossover) secara random untuk menghasilkan *offspring*
  - Secara umum, probabilitas *crossover* antara 0.6 dan 1.0
- *Mutation*
  - Terdapat probabilitas bahwa *gene* dari anak dirubah secara random
  - Umumnya, probabilitas *mutation* adalah rendah (Misal: 0.001)

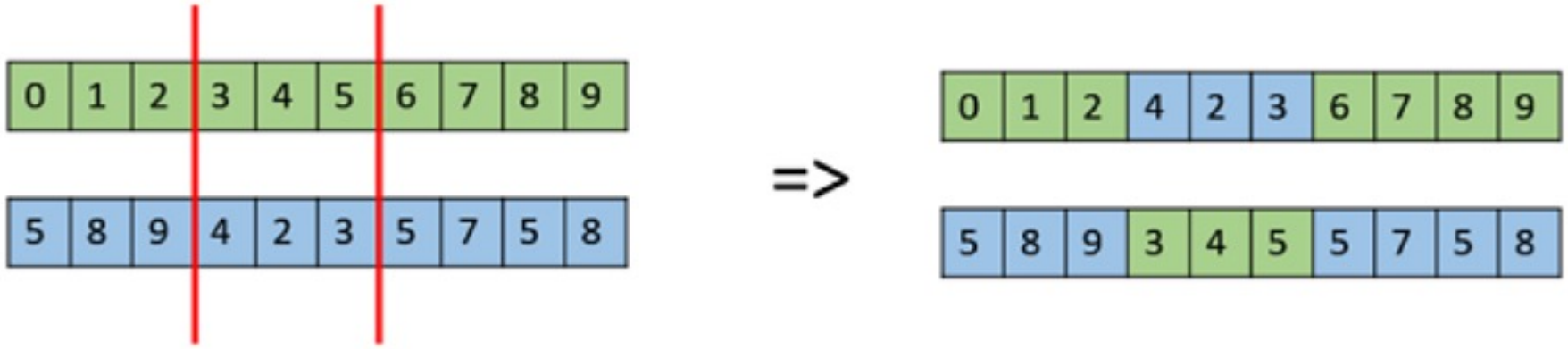
Crossover

- *Single point crossover*
  - Dipilih secara random sebuah posisi pada *chromosome*
  - Anak 1 merupakan kepala dari *chromosome parent* 1 dengan ekor dari *chromosome parent* 2
  - Anak 2 merupakan kepala dari *chromosome parent* 2 dengan ekor *chromosome parent* 1



Crossover

- *Twopoint crossover (Multi point crossover)*
  - Dipilih dua posisi secara random pada chromosomes
  - Mempertahankan kepala dan ekor pada chromosome ketika *recombined*



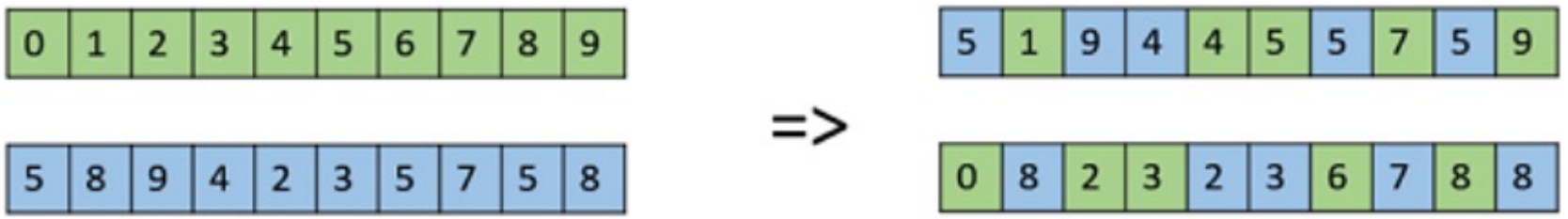
Crossover

- *Uniform crossover*
  - Generate *mask* secara random
  - Menentukan bit mana yang di *copy* dari satu parent dan dari parent lainnya

Mask: **0110011000** (***Randomly generated***)

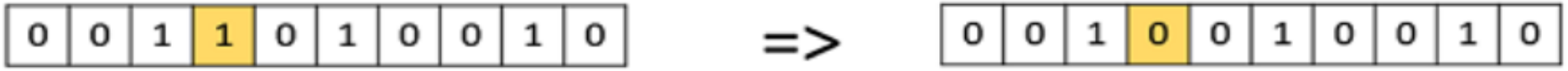
Parents: **1010001110** **0011010010**

Offspring: **0011001010** **1010010110**

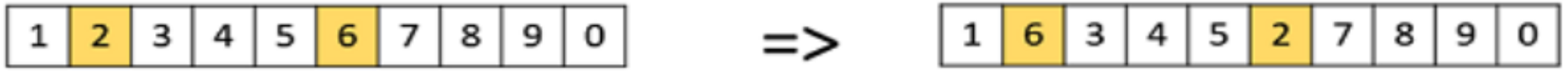


Mutation

- *Bit Flip Mutation*
  - Memilih secara random satu atau lebih bit dan mengganti nilainya
  - Digunakan pada GA dengan *binary encoded*



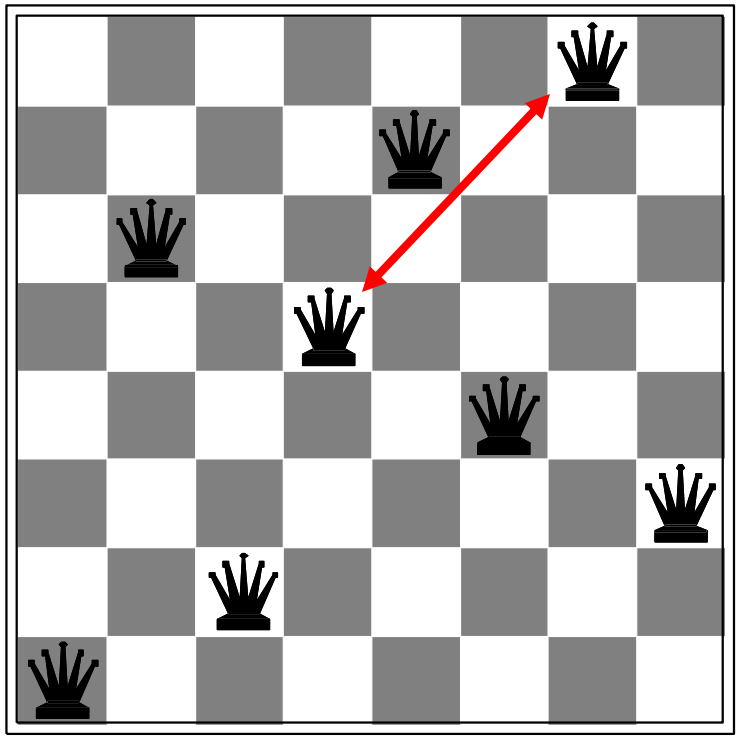
- *Random Resetting*
  - Memilih secara random satu atau lebih bit dan mengganti nilainya
  - Digunakan pada GA dengan representasi integer
- *Swap Mutation*
  - Memilih secara random dua position pada *chromosome* dan kedua nilai tersebut ditukar.
  - Umumnya digunakan pada GA dengan *permutation-based encodings*



Tujuan *Crossover* dan *Mutation*

- ***Exploration***: proses menemukan beberapa *promising area* pada ruang pencarian (*search space*)
- ***Exploitation***: melakukan optimasi didalam sebuah *promising area*
- *Crossover* adalah ***explorative***, yang membuat *big* jump ke area lain antara dua (*parent*) area
- *Mutation* adalah ***exploitative***, menciptakan *random small diversions*, sehingga masih berada pada sekitar area dari *parent*

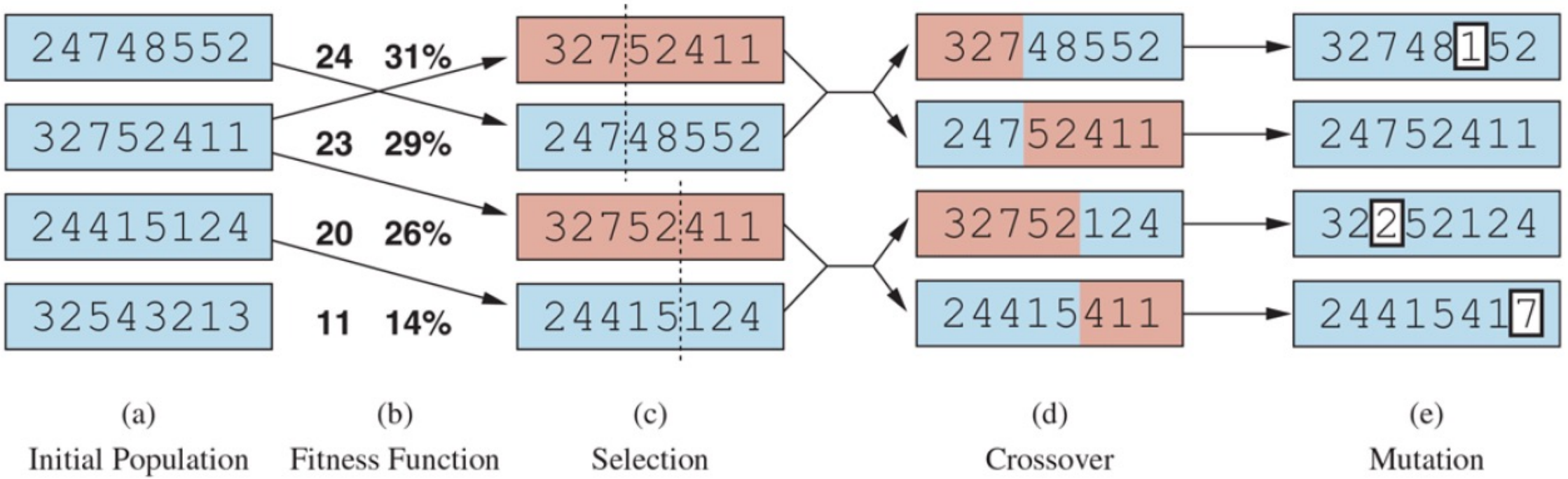
Contoh: Penyelesaian 8-queens problem menggunakan Genetic Algorithm



Representasi Genetik:

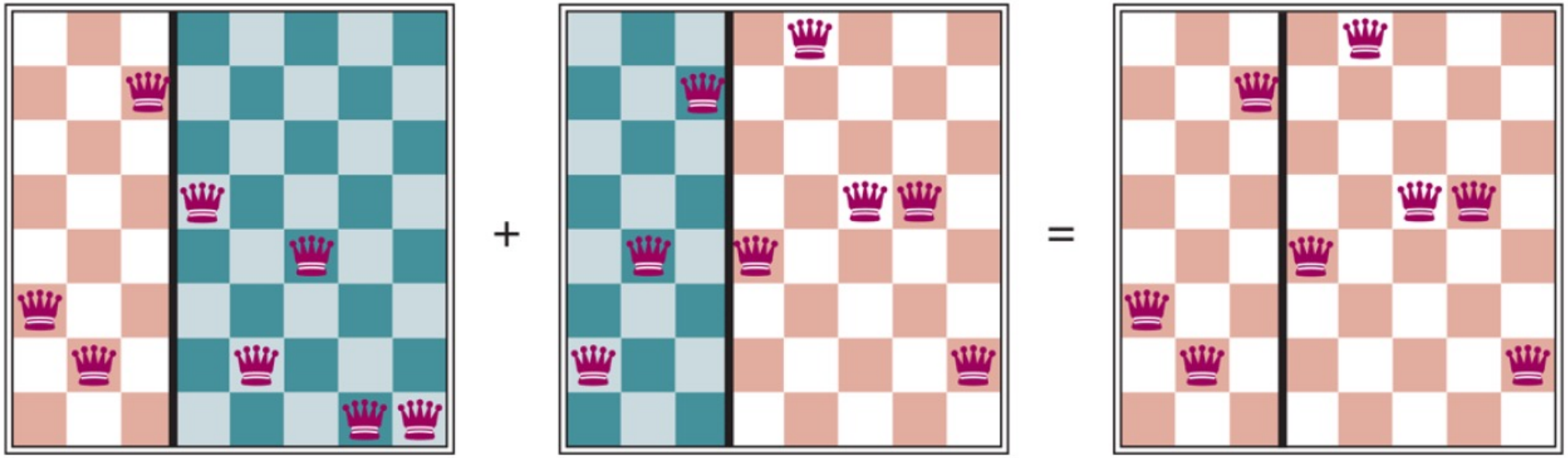
- Individu: 8-digit strings dimana posisi digit merepresentasikan nomor kolom dan nilainya merepresentasikan nomor baris. Misalnya: 83742516
- Fungsi fitness: banyaknya pasangan ratu yang tidak saling menyerang. Misalnya: 7+6+5+4+3+2=27
- Asumsi jumlah populasi adalah 8
- Selection menggunakan ranking berdasarkan fungsi fitness
- Crossover menggunakan single point

Contoh: Penyelesaian 8-queens problem menggunakan Genetic Algorithm



A genetic algorithm, illustrated for digit strings representing 8-queens states. The initial population in (a) is ranked by a fitness function in (b) resulting in pairs for mating in (c). They produce offspring in (d), which are subject to mutation in (e).

Contoh: Penyelesaian 8-queens problem menggunakan Genetic Algorithm



57257688 + 75251447 = 57251447

Contoh hasil crossover (offspring) dari 2 parents (individu) pada 8-queens problem

Contoh: Optimasi parameter persamaan polinomial

- Suppose you have (x, y) data points
  - For example, (1.0, 4.1), (3.1, 9.5), (-5.2, 8.6), ...
- You would like to fit a polynomial (of up to degree 5) through these data points
  - A formula $y = ax^5 + bx^4 + cx^3 + dx^2 + ex + f$ that gives you a reasonably good fit to the actual data
  - Here's the usual way to compute goodness of fit:
    - Compute the sum of $(\text{actual } y - \text{predicted } y)^2$ for all the data points
    - The lowest sum represents the best fit
- **Using Genetic Algorithm**
  - "genes" are a, b, c, d, e, and f
  - "chromosome" is the array [a, b, c, d, e, f]
  - evaluation function for *one* array is:
    - For *every* actual data point (x, y),
      - Compute $\acute{y} = ax^5 + bx^4 + cx^3 + dx^2 + ex + f$
      - Find the sum of $(y - \acute{y})^2$ over all x
      - The sum is your measure of "badness" (larger numbers are worse)
    - Example: For [0, 0, 0, 2, 3, 5] and the data points (1, 12) and (2, 22):
      - $\acute{y} = 0x^5 + 0x^4 + 0x^3 + 2x^2 + 3x + 5$ is $2 + 3 + 5 = 10$ when x is 1
      - $\acute{y} = 0x^5 + 0x^4 + 0x^3 + 2x^2 + 3x + 5$ is $8 + 6 + 5 = 19$ when x is 2
      - $(12 - 10)^2 + (22 - 19)^2 = 2^2 + 3^2 = 13$
      - If these are the only two data points, the "badness" of [0, 0, 0, 2, 3, 5] is 13

Contoh: Optimasi parameter persamaan polinomial

- The GA algorithm might be as follows:
  - Create 100 six-element arrays of random numbers
  - Repeat 500 times (or any other number):
    - For each of the 100 arrays, compute its badness (using all data points)
    - Keep the ten best arrays (discard the other 90)
    - From each array you keep, generate nine new arrays as follows:
      - Pick a random element of the six
      - Pick a random floating-point number between 0.0 and 2.0
      - Multiply the random element of the array by the random floating-point number
  - After all 500 trials, pick the best array as the final answer

Contoh: Optimasi Traveling Salesman Problem

- The Traveling Salesman Problem:
- Find a tour of a given set of cities so that
  - each city is visited only once
  - the total distance traveled is minimized
- Representation is an ordered list of city numbers known as an *order-based* GA.

1) London 3) Dunedin 5) Beijing 7) Tokyo
2) Venice 4) Singapore 6) Phoenix 8) Victoria

CityList1 (3 5 7 2 1 6 4 8)

CityList2 (2 5 7 6 8 1 3 4)

Contoh: Optimasi Traveling Salesman Problem

- Crossover combines inversion and recombination:

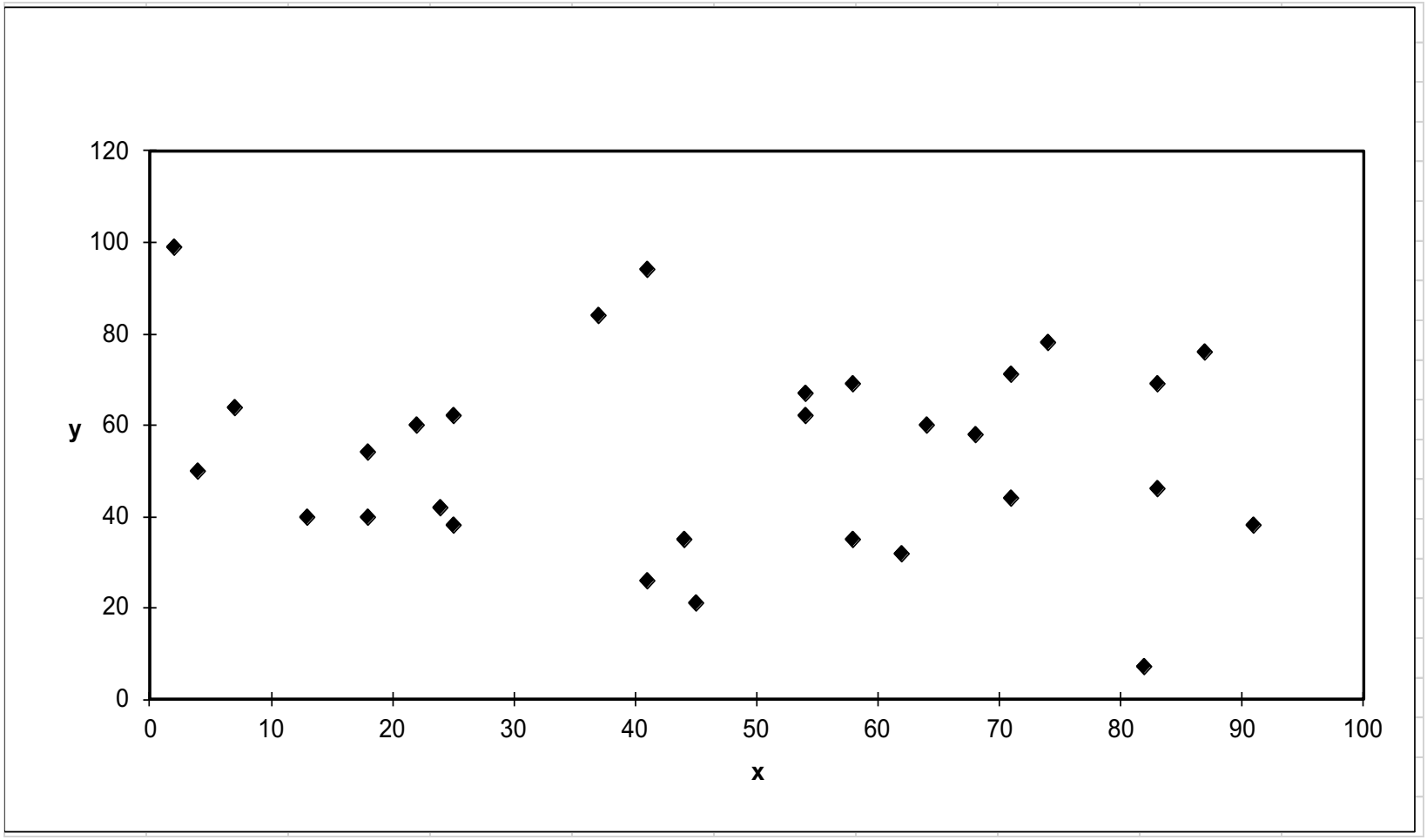
```
                    *       *
Parent1    (3  5  7  2  1  6  4  8)
Parent2    (2  5  7  6  8  1  3  4)

Child      (5  8  7  2  1  6  3  4)
```

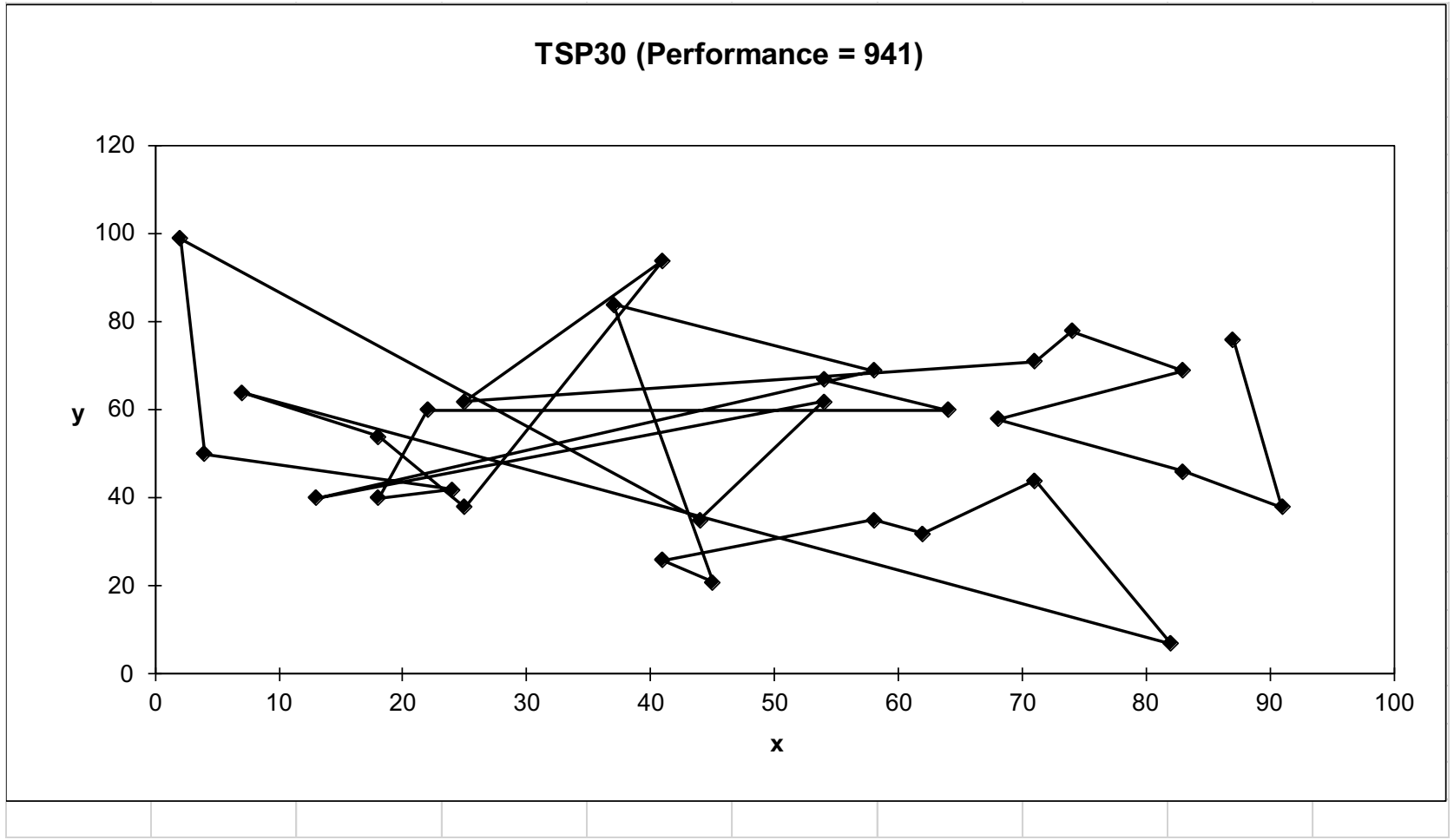
- Mutation involves reordering of the list:

```
                    *       *
Before:    (5  8  7  2  1  6  3  4)
                   <------------->
After:     (5  8  6  2  1  7  3  4)
```

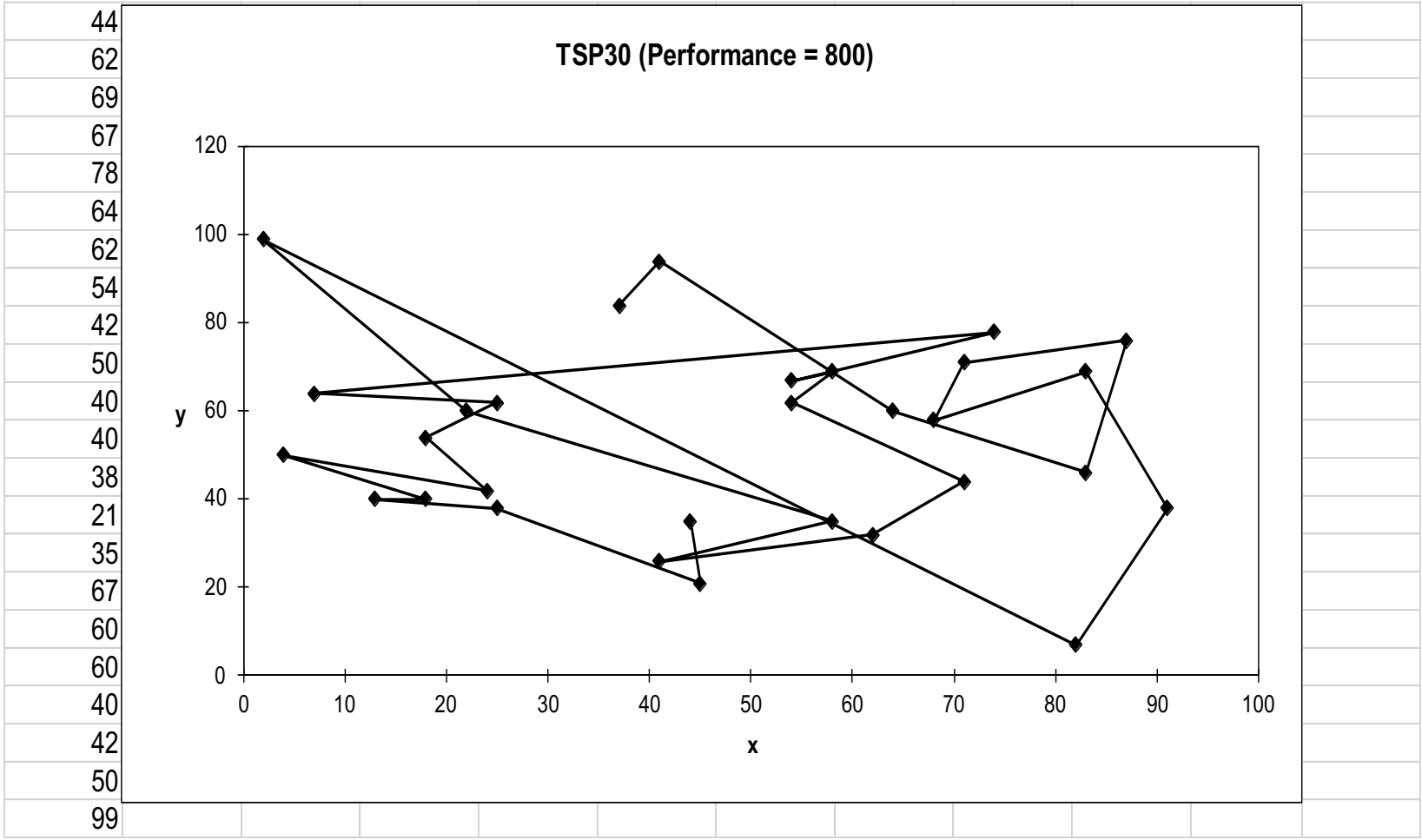
TSP Example: 30 Cities



Solution (Distance = 941)

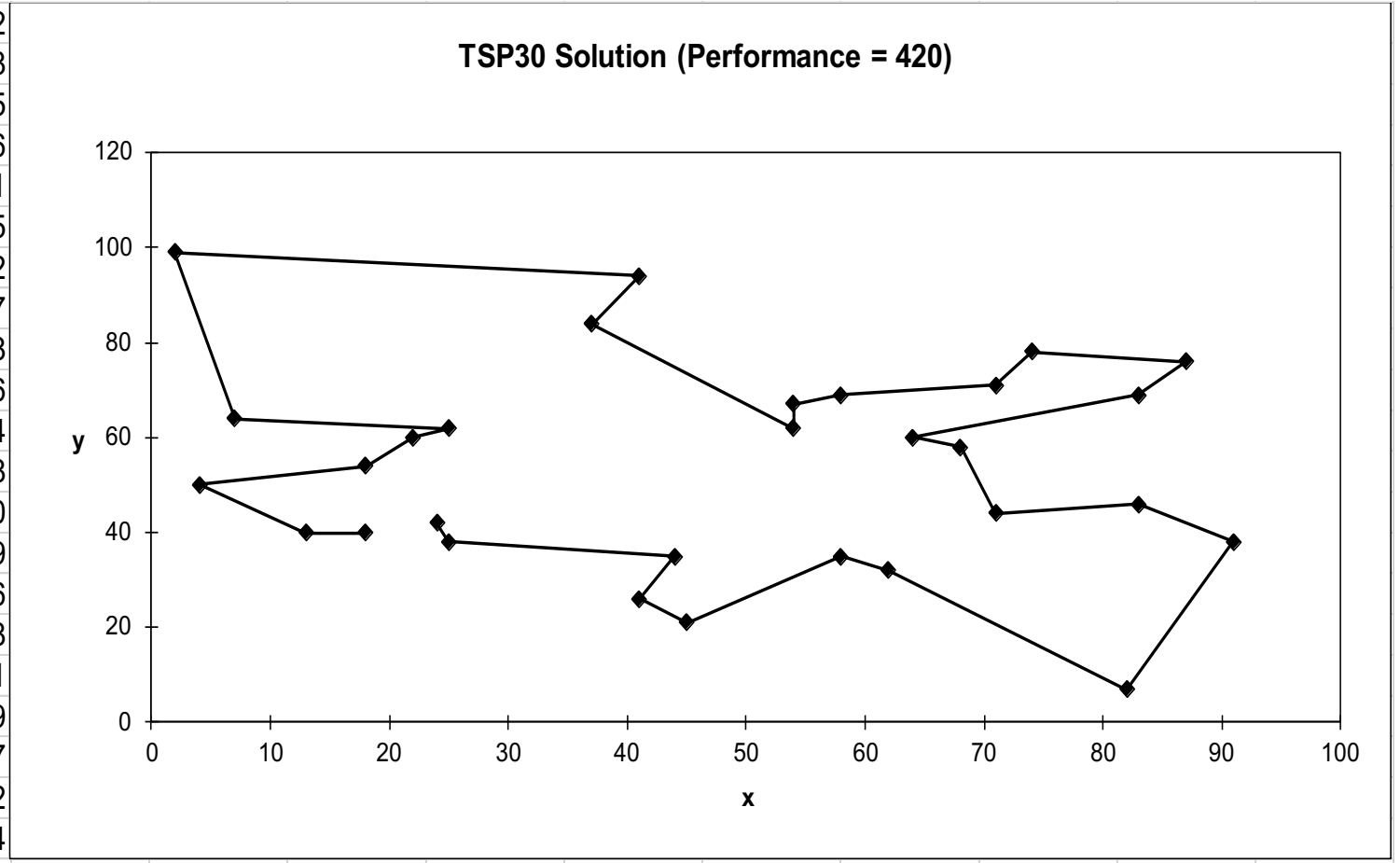


Solution (Distance = 800)



Best Solution (Distance = 420)

42
38
35
26
21
35
32
7
38
46
44
58
60
69
76
78
71
69
67
62
84
94



- TERIMA KASIH -