PRINCIPLES OF SOFT COMPUTING

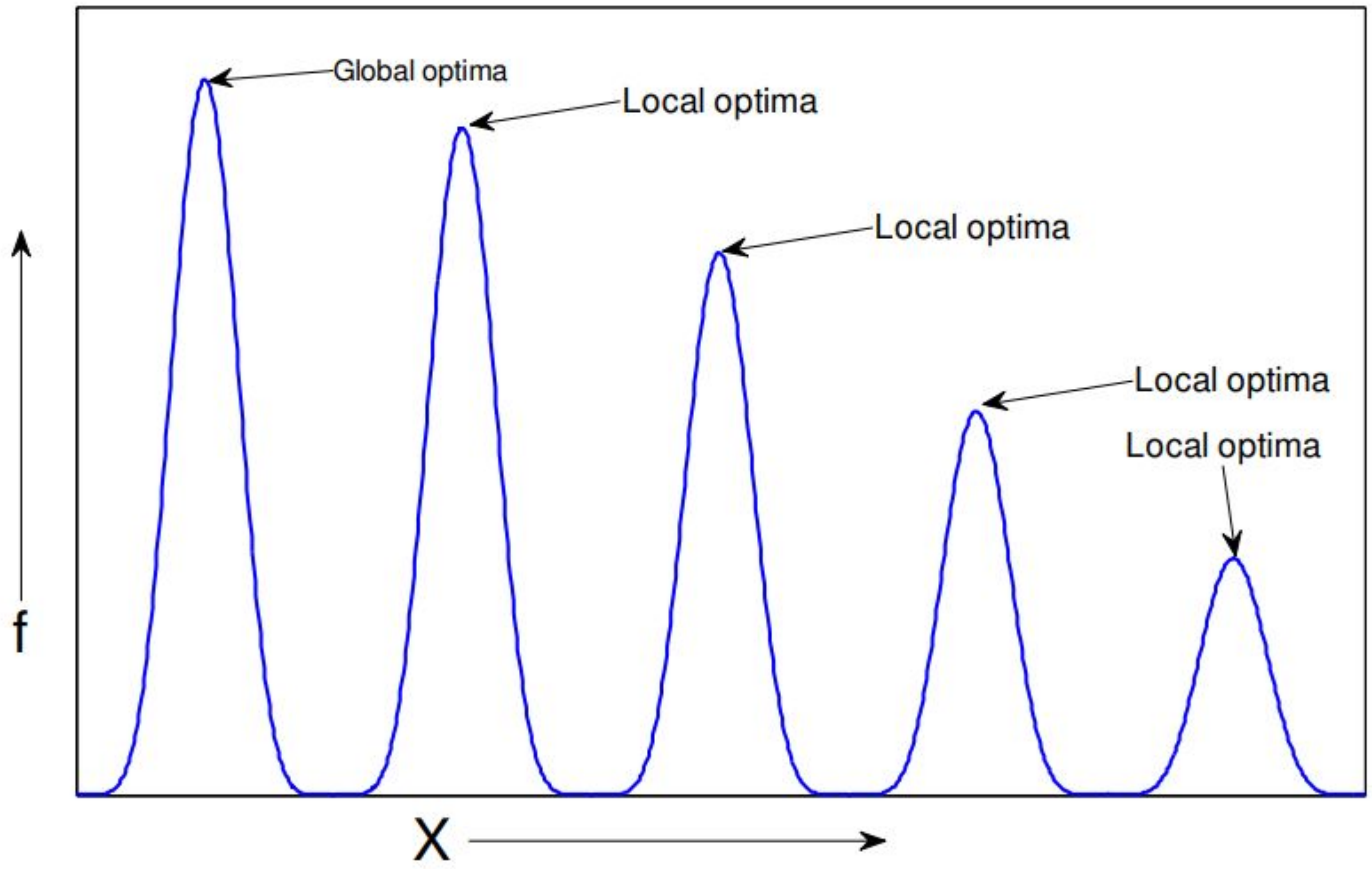
Dr. Neeraj kumar sharma

UNIT IV: GENETIC ALGORITHMS

TOPICS

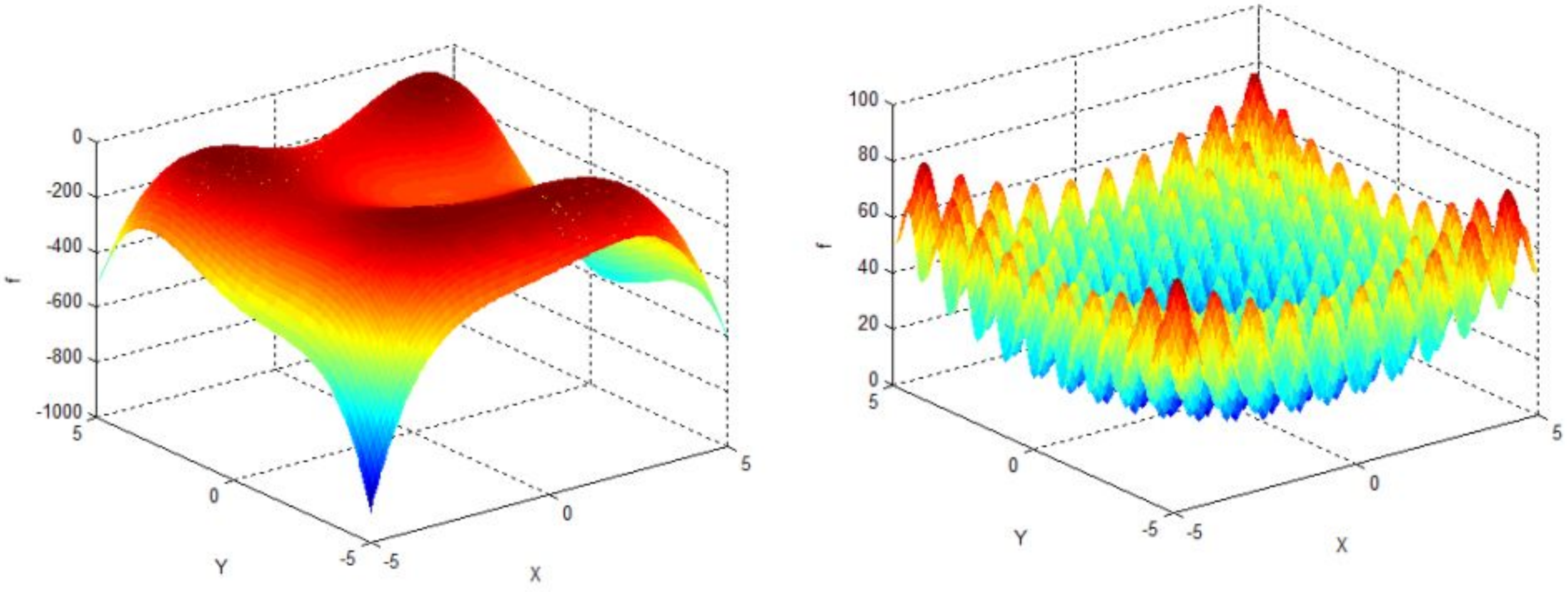
Fundamentals of genetic algorithms: Encoding, Fitness functions, Reproduction. Genetic Modeling: Cross cover, Inversion and deletion, Mutation operator, Bit-wise operators, Bitwise operators used in GA. Convergence of Genetic algorithm. Applications, Real life Problems. Particle Swarm Optimization and its variants.

Introduction to optimization



Introduction to optimization

Multiple optimal solutions

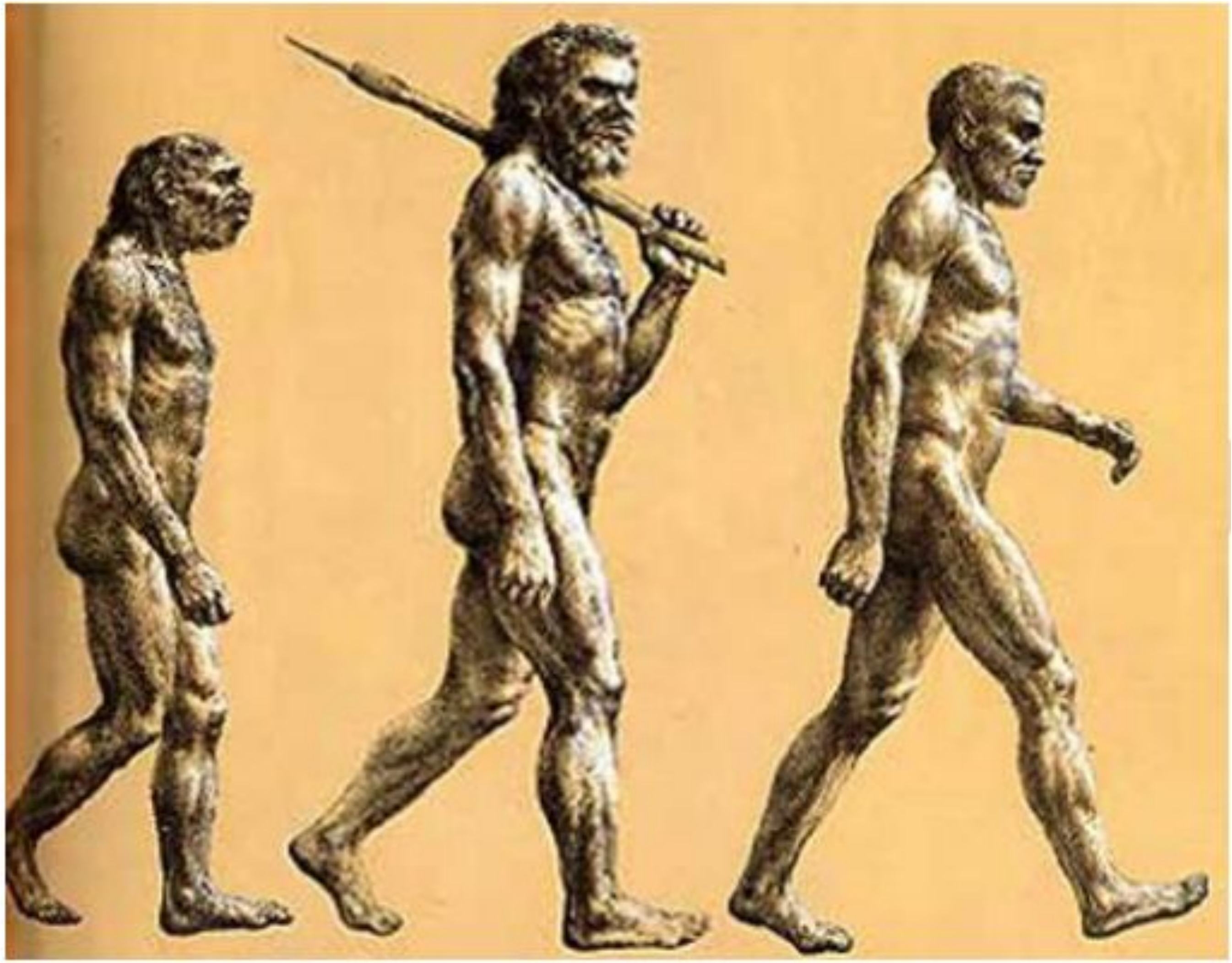


Genetic Algorithms

Genetic Algorithms are the heuristic search and optimization techniques that mimic the process of natural evolution.

Principle Of Natural Selection

"Select The Best, Discard The Rest"

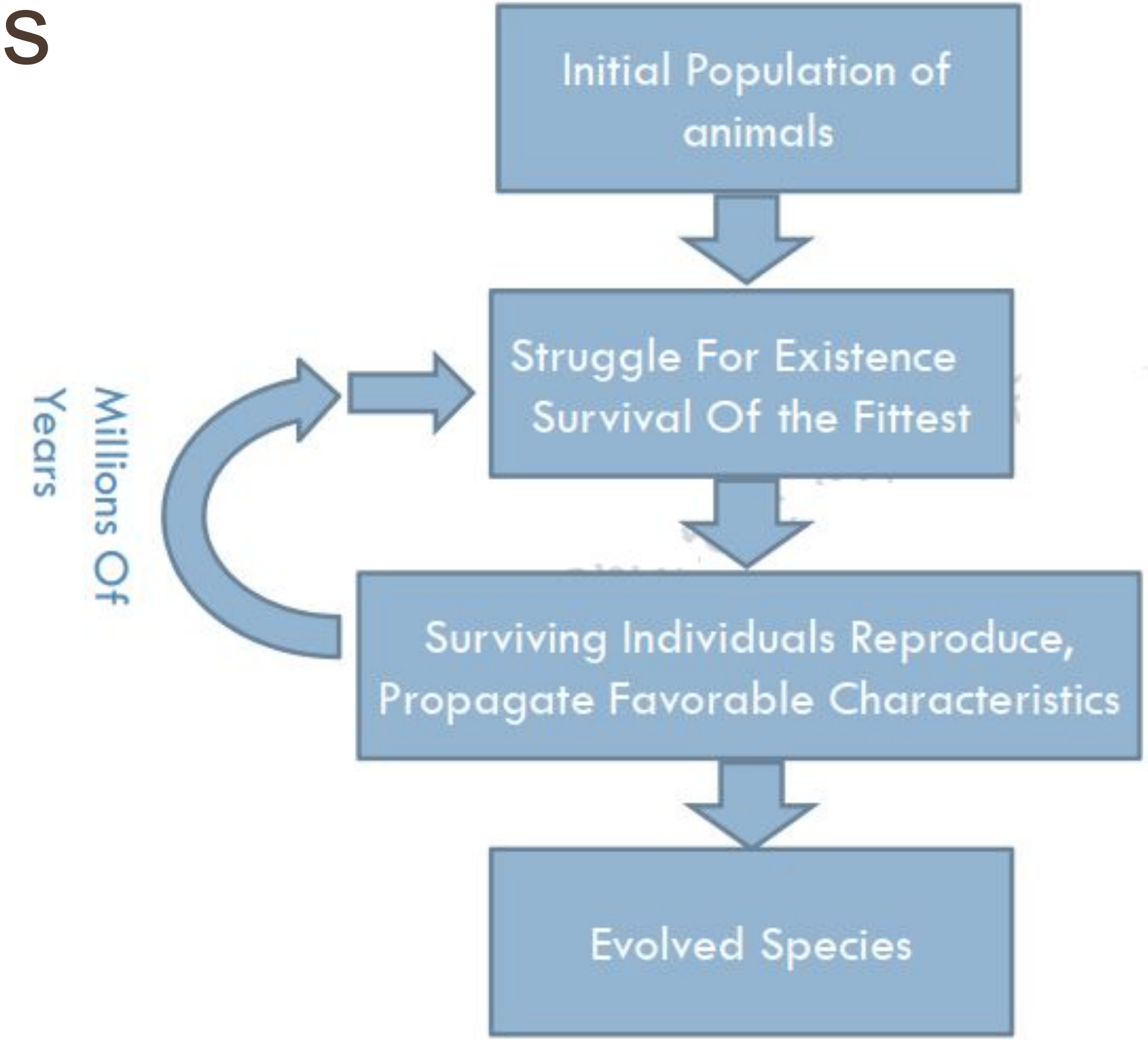


Giraffes have long necks

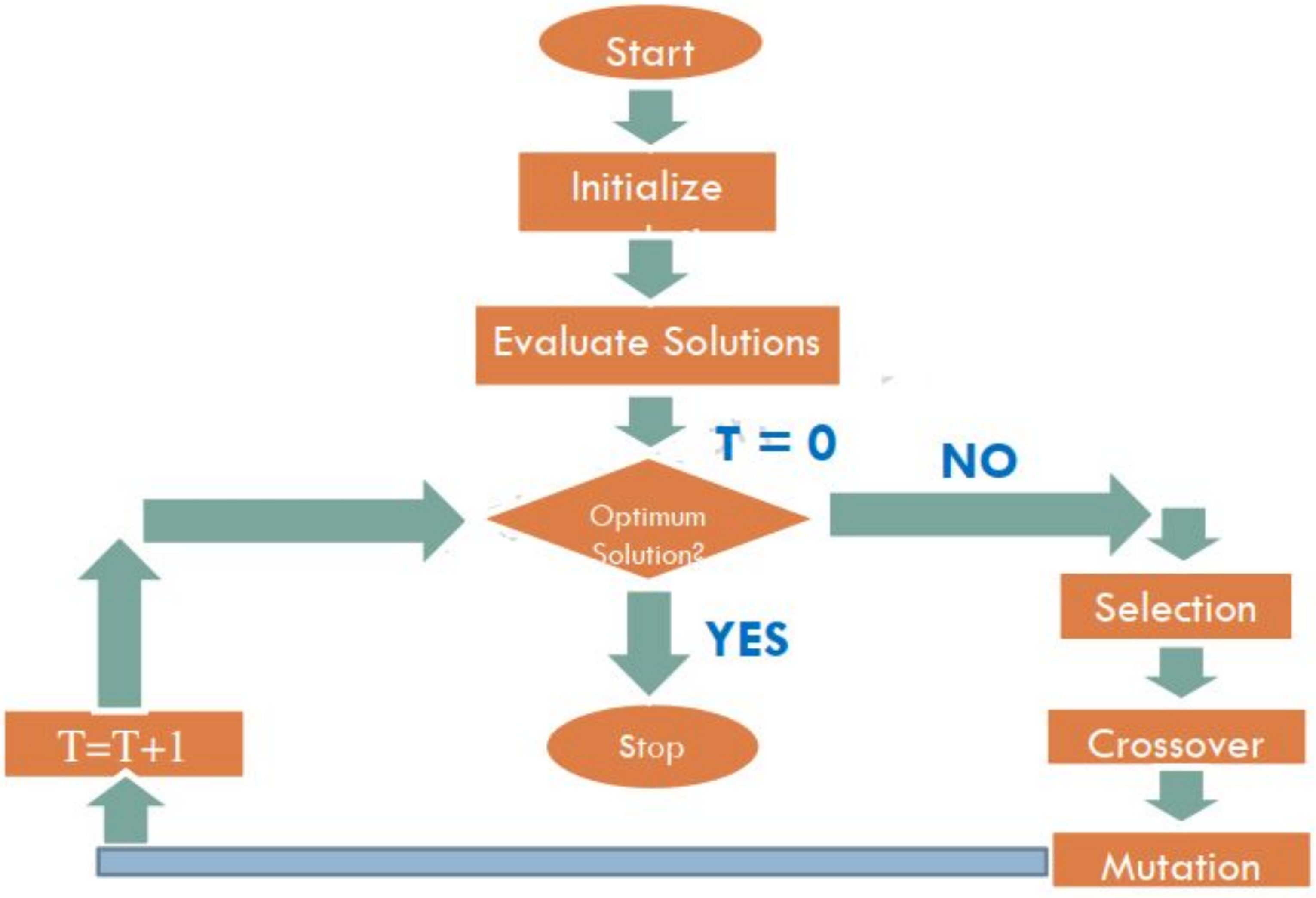
- Giraffes with slightly longer necks could feed on leaves of higher branches when all lower ones had been eaten off.
- They had a better chance of survival.
- Favorable characteristic propagated through generations of giraffes.
- Now, evolved species has long necks.
- This longer necks may have due to the effect of mutation initially. However as it was favorable, this was propagated over the generations.

Evolution of species

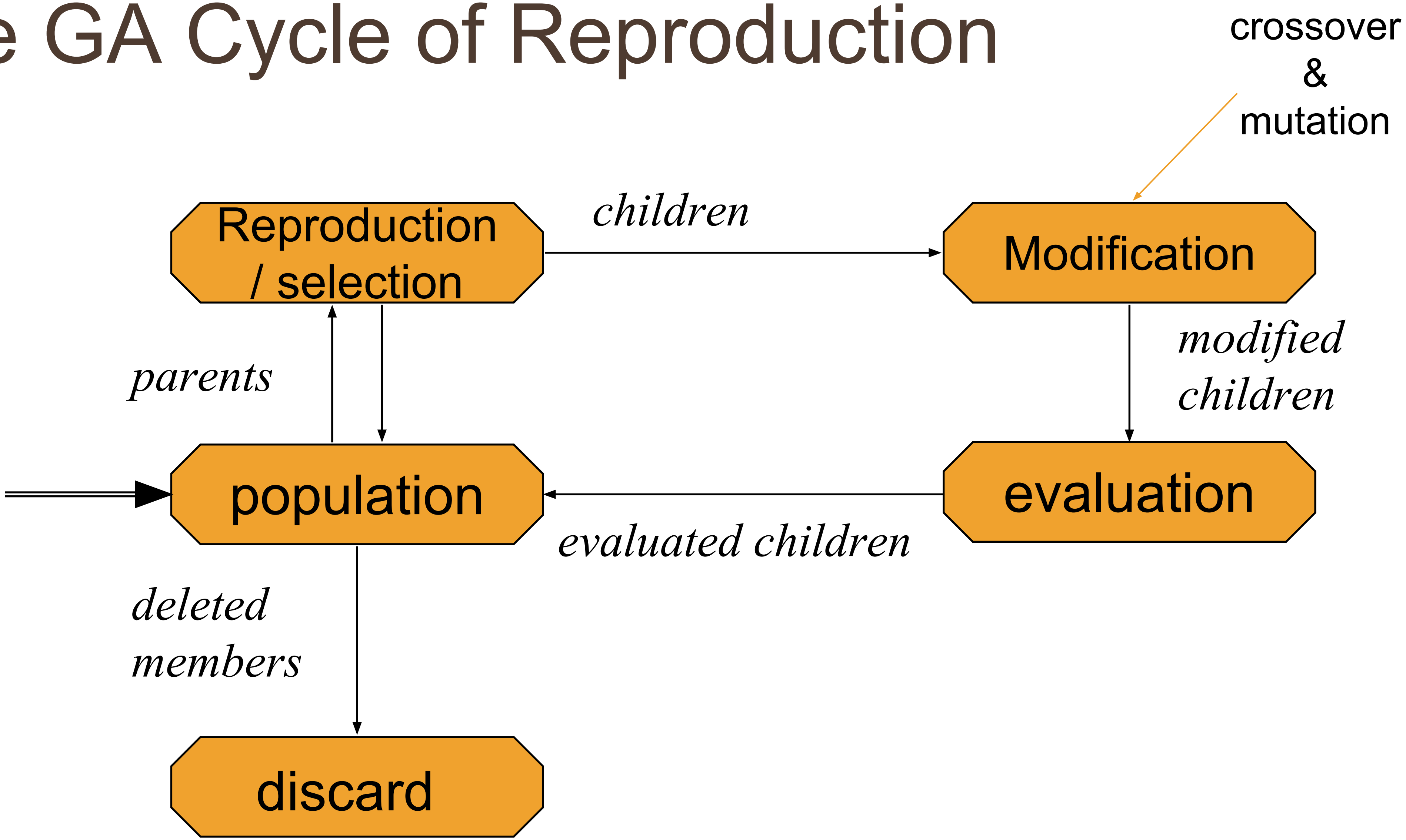
Thus genetic algorithms implement the optimization strategies by simulating evolution of species through natural selection



Simple Genetic Algorithms



The GA Cycle of Reproduction



Simple Genetic Algorithm

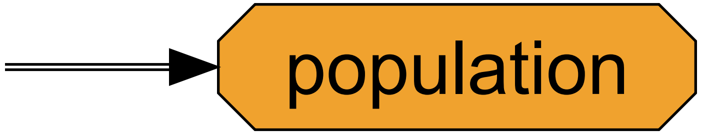
```
function sga ()
    {
        Initialize population;
        Calculate fitness function;
        While(fitness value != termination criteria)
        {
            Selection;
            Crossover;
            Mutation;
            Calculate fitness function;
        }
    }
```

Components of a GA

A problem to solve, and ...

- Encoding technique (*gene, chromosome*)
- Initialization procedure (*creation*)
- Evaluation function (*environment*)
- Selection of parents (*reproduction*)
- Genetic operators (*mutation, recombination*)
- Parameter settings (*practice and art*)

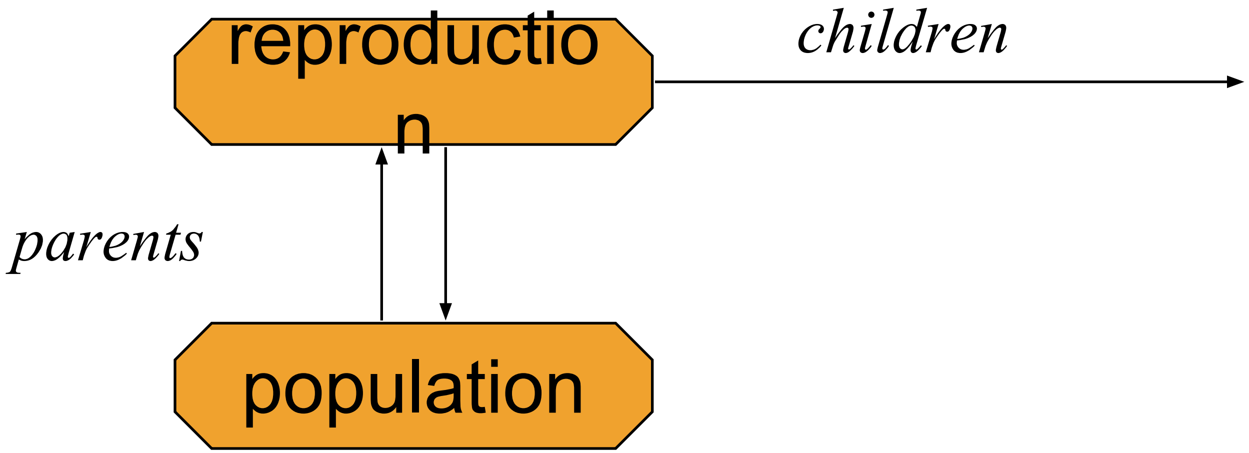
Population



Chromosomes could be:

- Bit strings (0101 ... 1100)
- Real numbers (43.2 -33.1 ... 0.0 89.2)
- Permutations of element (E11 E3 E7 ... E1 E15)
- Lists of rules (R1 R2 R3 ... R22 R23)
- Program elements (genetic programming)
- ... any data structure ...

Reproduction/selection



- Parents are selected at random with selection chances biased in relation to chromosome evaluations.

Selection

- The process that determines which solutions are to be preserved and allowed to reproduce and which ones deserve to die out.
- The primary objective of the selection operator is to emphasize the good solutions and eliminate the bad solutions in a population while keeping the population size constant.
  “Selects the best, discards the rest”

Functions of Selection operator

- Identify the good solutions in a population
- Make multiple copies of the good solutions
- Eliminate bad solutions from the population so that multiple copies of good solutions can be placed in the population
- Now how to identify the good solutions?

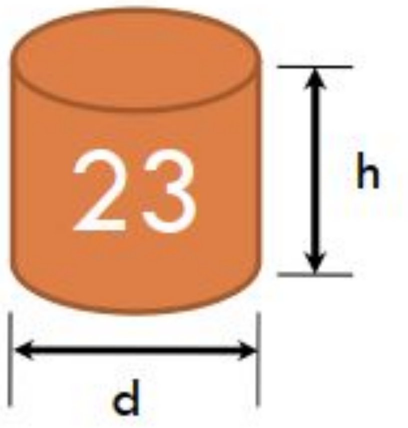
Fitness function

- A fitness function value quantifies the optimality of a solution. The value is used to rank a particular solution against all the other solutions
- A fitness value is assigned to each solution depending on how close it is actually to the optimal solution of the problem
- A fitness value is assigned to each solution depending on close it is actually to the optimal solution of the problem

Assigning a fitness value

$$\begin{aligned}
\text{Minimize} \quad & f(d,h) = c((\pi d^2/2) + \pi dh), \\
\text{Subject to} \quad & g_1(d,h) \equiv (\pi d^2 h/4) \geq 300, \\
\text{Variable bounds} \quad & d_{\min} \leq d \leq d_{\max}, \\
& h_{\min} \leq h \leq h_{\max}.
\end{aligned}$$

Considering $c = 0.0654$



$$\begin{aligned}
F(s) &= 0.0654(\pi(8)^2/2 + \pi(8)(10)), \\
&= 23,
\end{aligned}$$

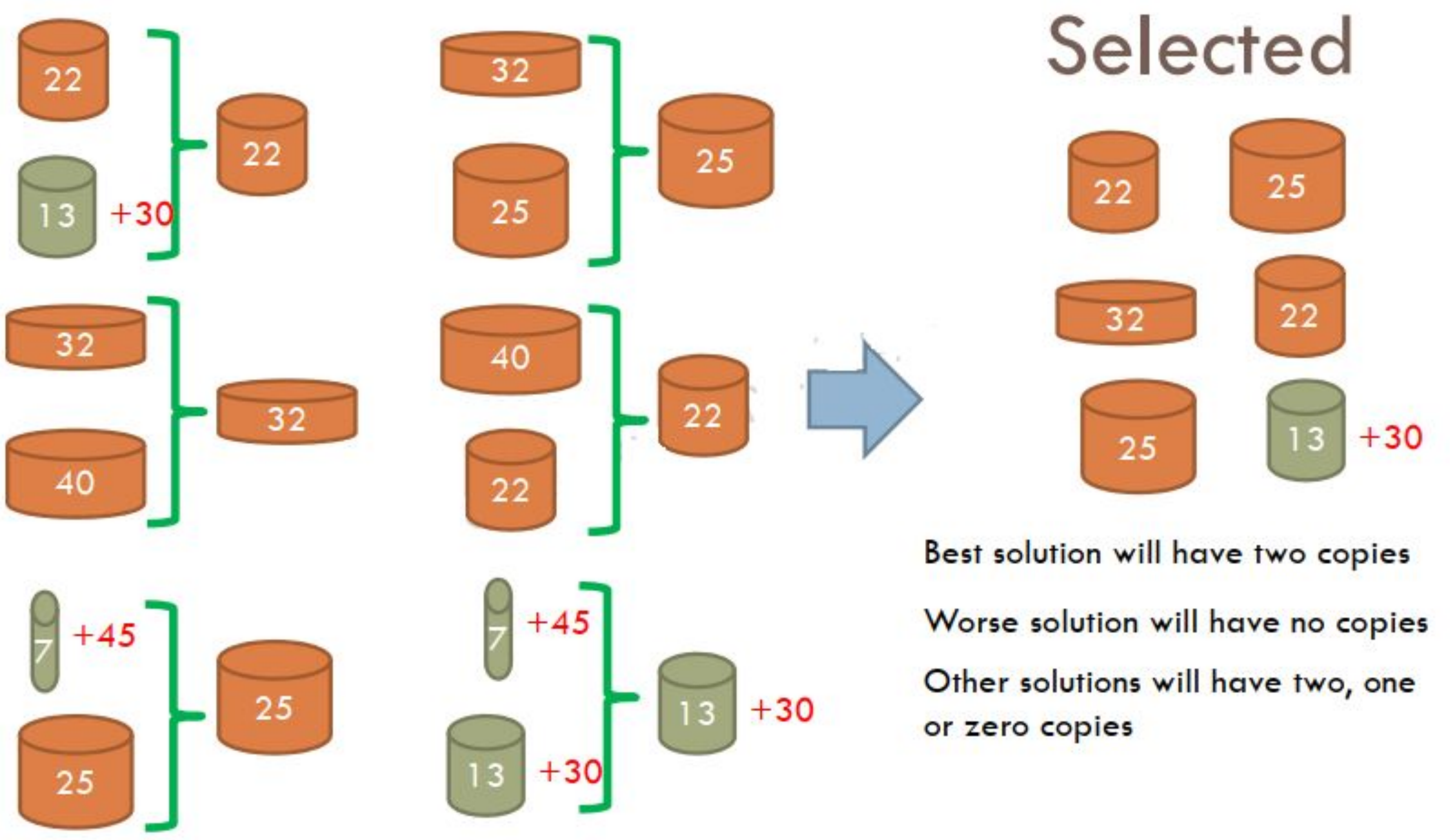
Selection Operator

- There are different techniques to implement selection in Genetic Algorithms.
- They are:
  - Tournament selection
  - Roulette wheel selection
  - Proportionate selection
  - Rank selection
  - Steady state selection, etc

Tournament selection

- In tournament selection several tournaments are played among a few individuals. The individuals are chosen at random from the population.
- The winner of each tournament is selected for next generation.
- Selection pressure can be adjusted by changing the tournament size.
- Weak individuals have a smaller chance to be selected if tournament size is large.

Tournament selection

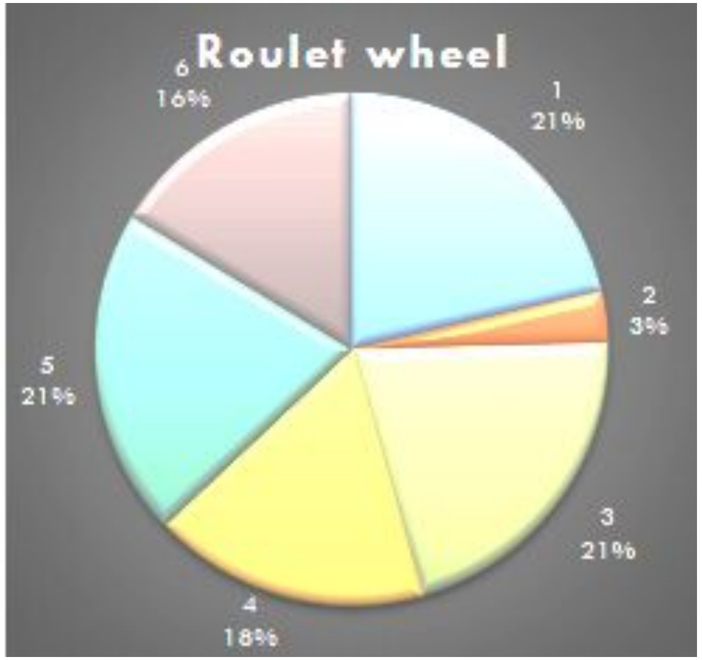


Best solution will have two copies

Worse solution will have no copies

Other solutions will have two, one or zero copies

Roulette wheel and proportionate selection



Chrom #	Fitness	% of RW	EC	AC
1	50	26.88	1.61	2
2	6	3.47	0.19	0
3	36	20.81	1.16	1
4	30	17.34	0.97	1
5	36	20.81	1.16	1
6	28	16.18	0.90	1
	186	100.00	6	6

- Parents are selected according to their fitness values
- The better chromosomes have more chances to be selected

Rank selection

Chrom #	Fitness
1	37
2	6
3	36
4	30
5	36
6	28

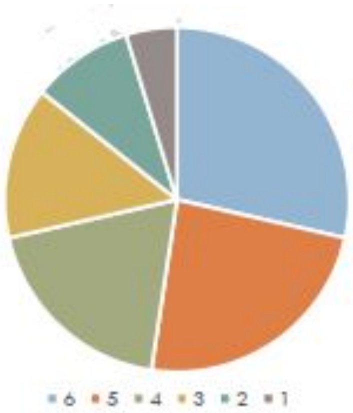
Sort according to fitness

Chrom #	Fitness
1	37
3	36
5	36
4	30
6	28
2	6

Assign raking

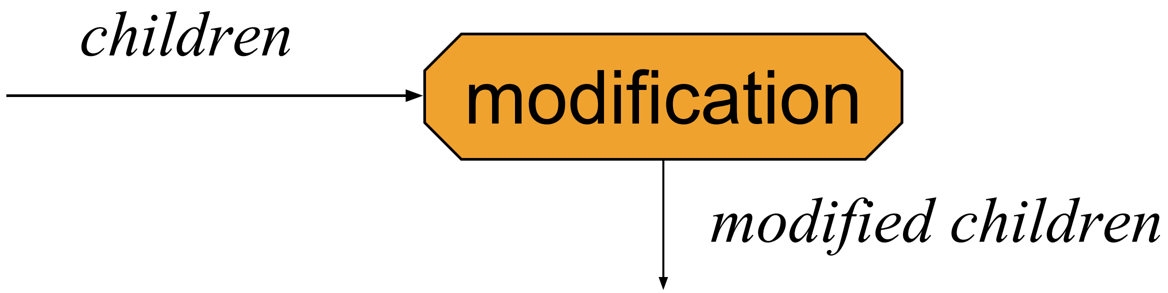
Chrom #	Rank
1	6
3	5
5	4
4	3
6	2
2	1

Chrom #	% of RW
1	29
3	24
5	19
4	14
6	10
2	5



Chrom #	EC	AC
1	1.714	2
3	1.429	1
5	1.143	1
4	0.857	1
6	0.571	1
2	0.286	0

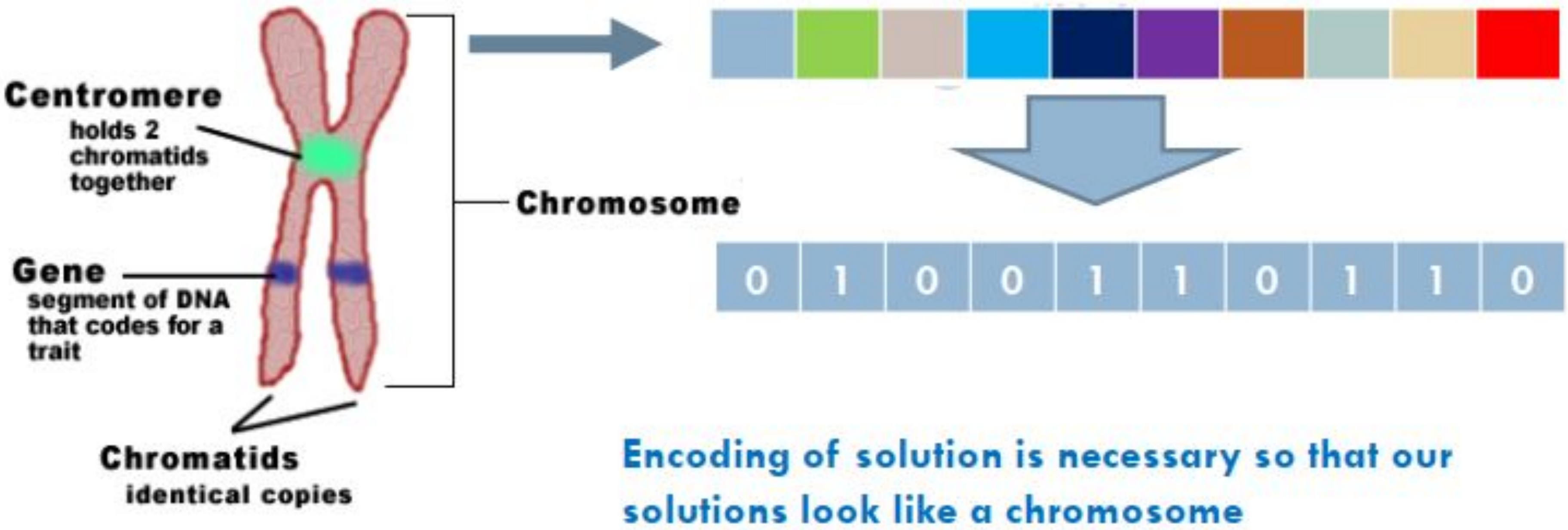
Chromosome Modification/ Crossover



- Modifications are stochastically triggered
- Operator types are:
  - Mutation
  - Crossover (recombination)

How to implement crossover

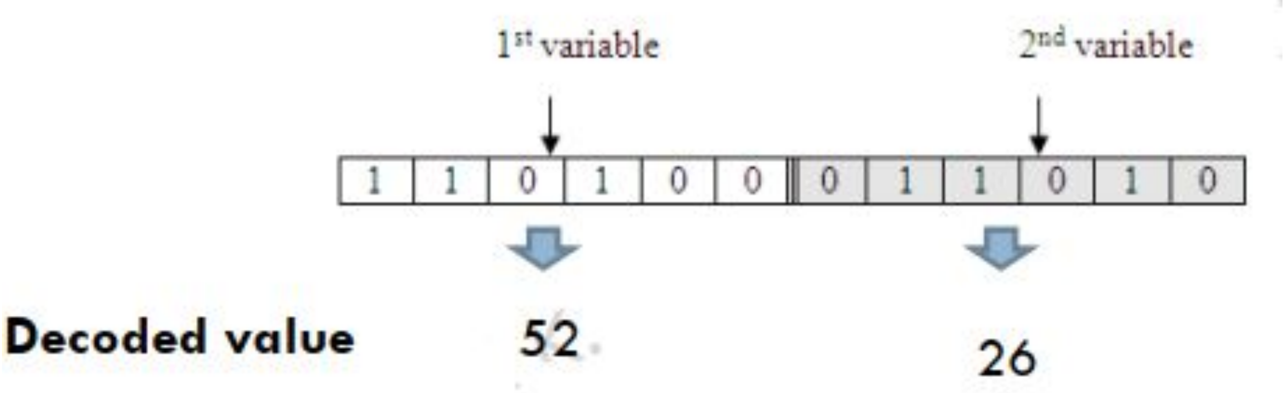
- The crossover operator is used to create new solutions from the existing solutions available in the mating pool after applying selection operator.
- This operator exchanges the gene information between the solutions in the mating pool.



Encoding

- The process of representing a solution in the form of a string that conveys the necessary information.
- Just as in a chromosome, each gene controls a particular characteristic of the individual, similarly, each bit in the string represents a characteristic of the solution.
- Most common method of encoding is binary coded. Chromosomes are strings of 1 and 0 and each position in the chromosome represents a particular characteristic of the problem.

Encoding

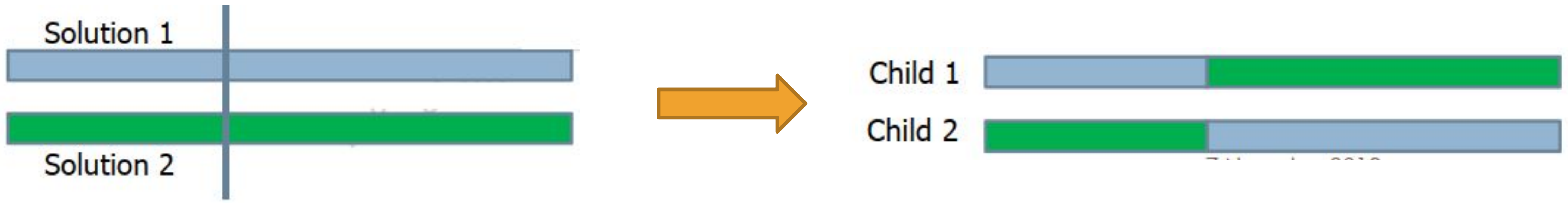


Mapping between decimal and binary value

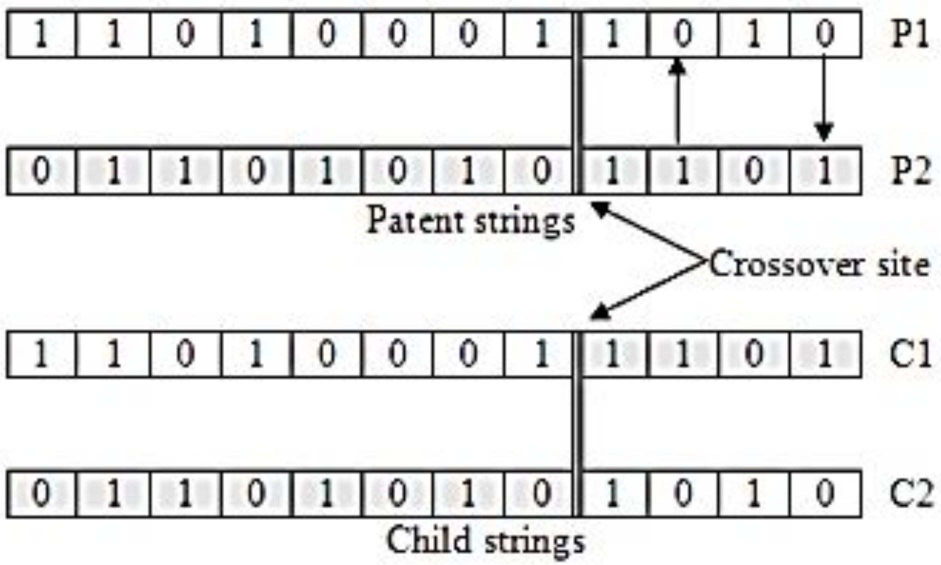
$$x_i = x_i^{\min} + \frac{x_i^{\max} - x_i^{\min}}{2^{l_i} - 1} \mathrm{DV}(s_i)$$

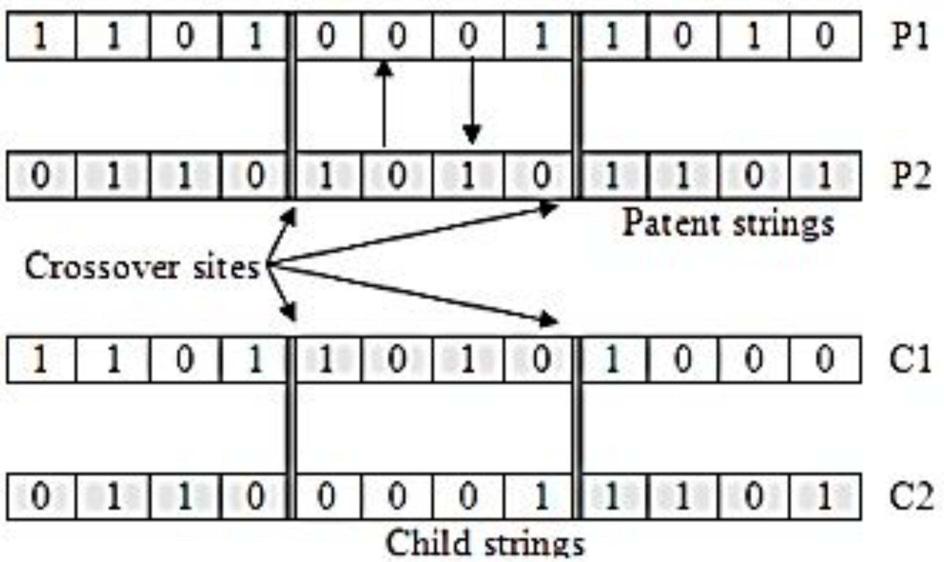
Crossover operator

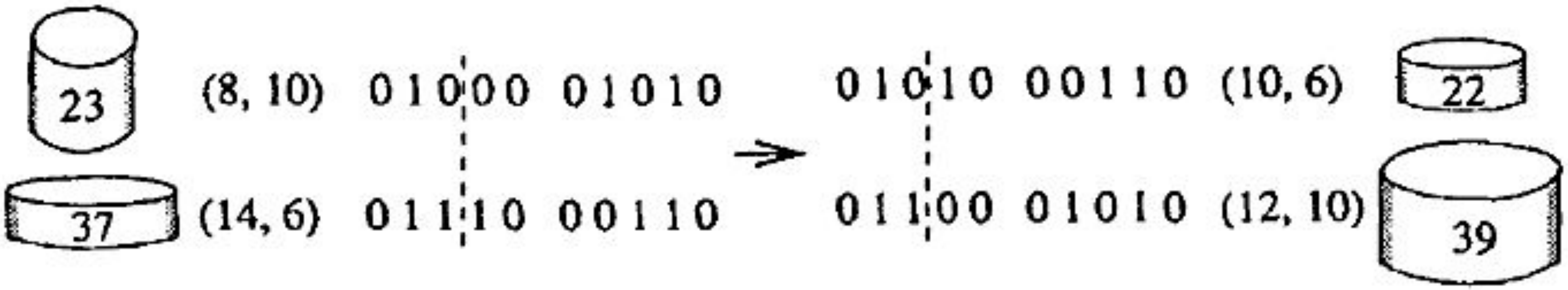
- The most popular crossover selects any two solutions strings randomly from the mating pool and some portion of the strings is exchanged between the strings.
- The selection point is selected randomly.
- A probability of crossover is also introduced in order to give freedom to an individual solution string to determine whether the solution would go for crossover or not.



Binary Crossover

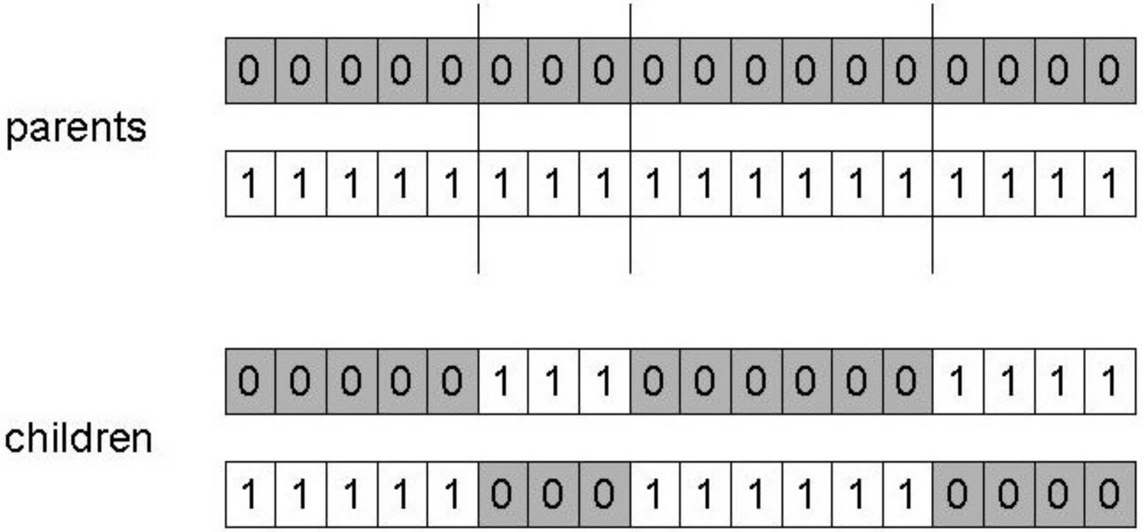






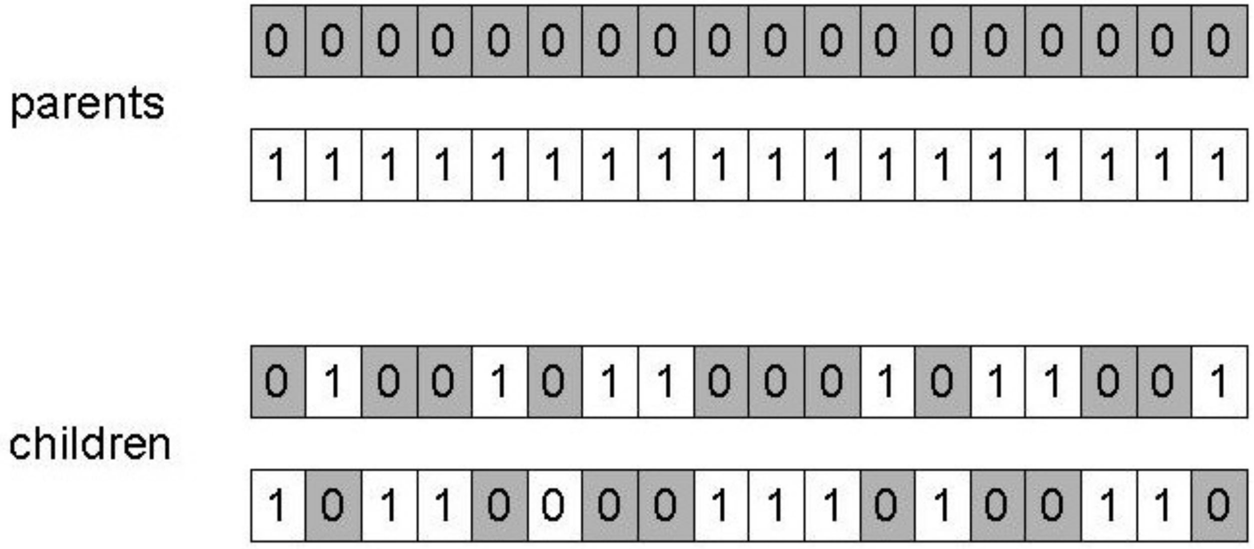
N-point crossover

- Choose n random crossover points
- Split along those points
- Glue parts, alternating between parents
- Generalisation of 1 point (still some positional bias)



Uniform crossover

- Assign 'heads' to one parent, 'tails' to the other
- Flip a coin for each gene of the first child
- Make an inverse copy of the gene for the second child
- Inheritance is independent of position



Crossover

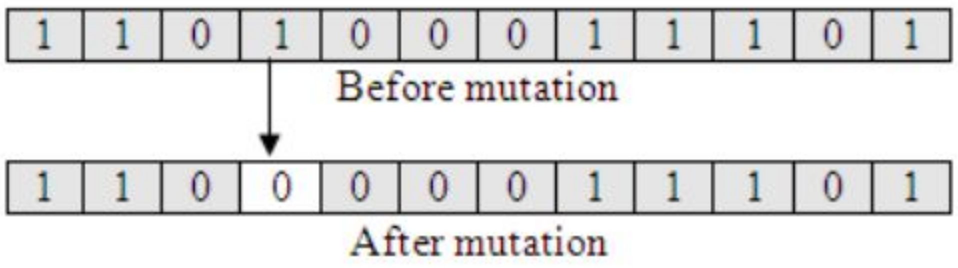
Crossover is a critical feature of genetic algorithms:

- It greatly accelerates search early in evolution of a population
- It leads to effective combination of schemata (sub-solutions on different chromosomes)

Mutation operator

- Though crossover has the main responsibility to search for the optimal solution, mutation is also used for this purpose.
- Mutation is the occasional introduction of new features in to the solution strings of the population pool to maintain diversity in the population.

- The mutation probability is generally kept low for steady convergence.



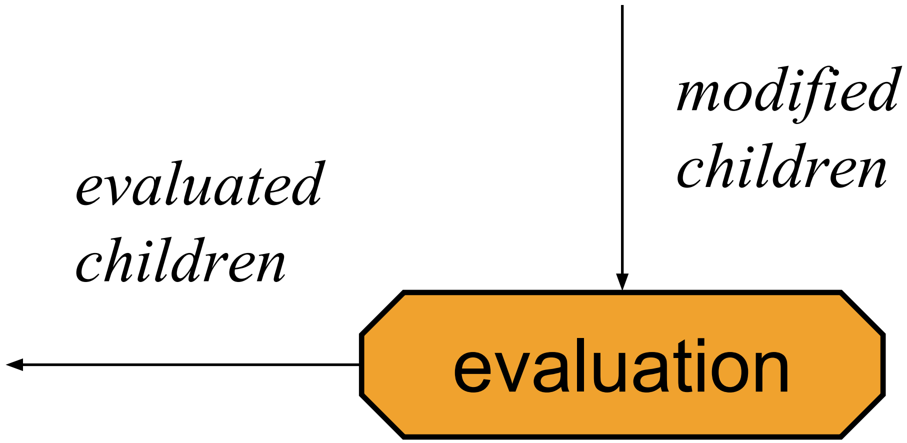
Mutation: Local Modification

Before:	(1 0 1 1 0 1 1 0)
After:	(0 1 1 0 0 1 1 0)

Before:	(1.38 -69.4 326.44 0.1)
After:	(1.38 -67.5 326.44 0.1)

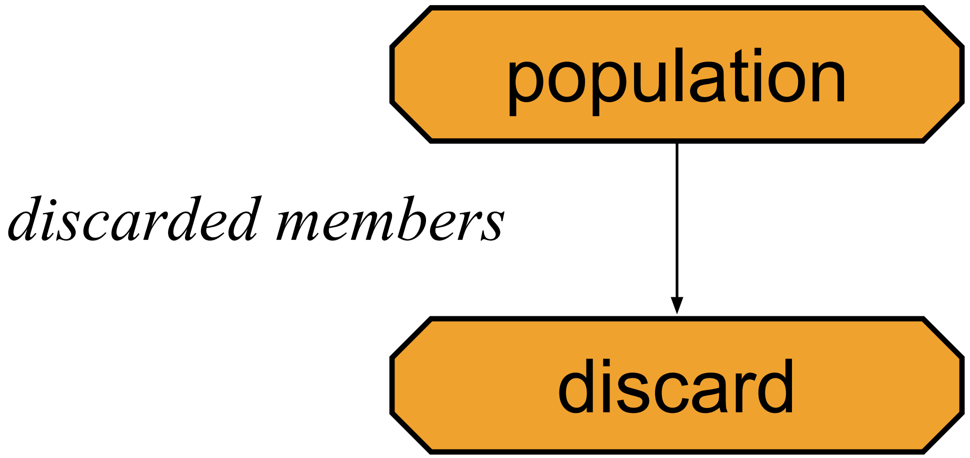
- Causes movement in the search space (local or global)
- Restores lost information to the population

Evaluation



- The evaluator decodes a chromosome and assigns it a fitness measure
- The evaluator is the only link between a classical GA and the problem it is solving

Deletion



- *Generational* GA: entire populations replaced with each iteration
- *Steady-state* GA: a few members replaced each generation

Example

- Simple problem: maximize (x^2) over {0,1,…,31}
- GA approach:
  - Representation: binary code, e.g. 01101 ↔ 13
  - Population size: 4
  - 1-point xover, bitwise mutation
  - Roulette wheel selection
  - Random initialisation
- We show one generational cycle done by hand

x2 example: selection

String no.	Initial population	x Value	Fitness $f(x) = x^2$	$Prob_i$	Expected count	Actual count
1	0 1 1 0 1	13	169	0.14	0.58	1
2	1 1 0 0 0	24	576	0.49	1.97	2
3	0 1 0 0 0	8	64	0.06	0.22	0
4	1 0 0 1 1	19	361	0.31	1.23	1
Sum			1170	1.00	4.00	4
Average			293	0.25	1.00	1
Max			576	0.49	1.97	2

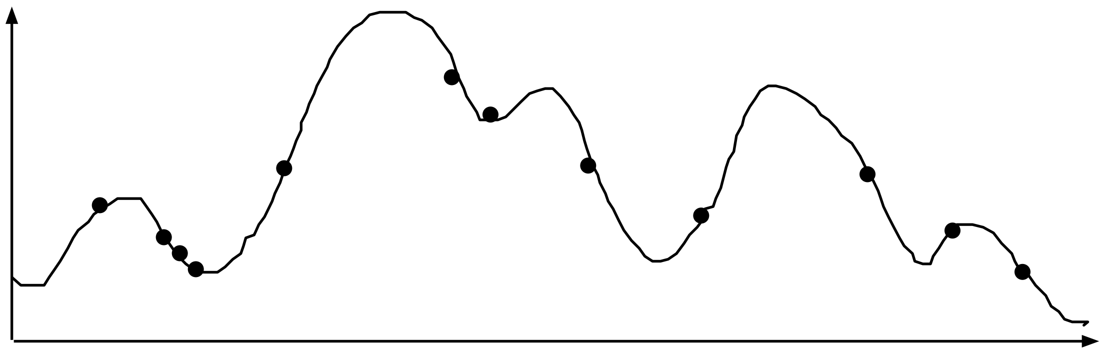
X2 example: crossover

String no.	Mating pool	Crossover point	Offspring after xover	x Value	Fitness $f(x) = x^2$
1	0 1 1 0 | 1	4	0 1 1 0 0	12	144
2	1 1 0 0 | 0	4	1 1 0 0 1	25	625
2	1 1 | 0 0 0	2	1 1 0 1 1	27	729
4	1 0 | 0 1 1	2	1 0 0 0 0	16	256
Sum					1754
Average					439
Max					729

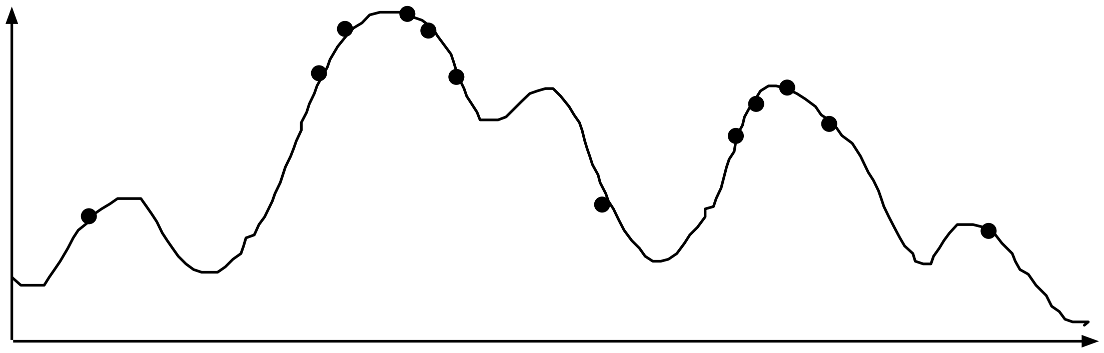
X2 example: mutation

String no.	Offspring after xover	Offspring after mutation	x Value	Fitness $f(x) = x^2$
1	0 1 1 0 0	$\boxed{1}$ 1 1 0 0	26	676
2	1 1 0 0 1	1 1 0 0 1	25	625
2	1 1 0 1 1	1 1 0 1 1	27	729
4	1 0 0 0 0	1 0 $\boxed{1}$ 0 0	18	324
Sum				2354
Average				588.5
Max				729

An Abstract Example



Distribution of Individuals in Generation 0



Distribution of Individuals in Generation N

A Simple Example

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

- 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)

Crossover

Crossover combines inversion and recombination:

$$
\begin{array}{ll}
 & \quad\quad\quad\quad\; * \quad\quad\quad\quad\; * \\
\text{Parent1} & (3\ \ 5\ \ \boxed{7\ \ 2\ \ 1\ \ 6}\ \ 4\ \ 8) \\
\text{Parent2} & (2\ \ 5\ \ 7\ \ 6\ \ 8\ \ 1\ \ 3\ \ 4) \\
\hline
\text{Child} & (2\ \ 5\ \ \boxed{7\ \ 2\ \ 1\ \ 6}\ \ 3\ \ 4)
\end{array}
$$

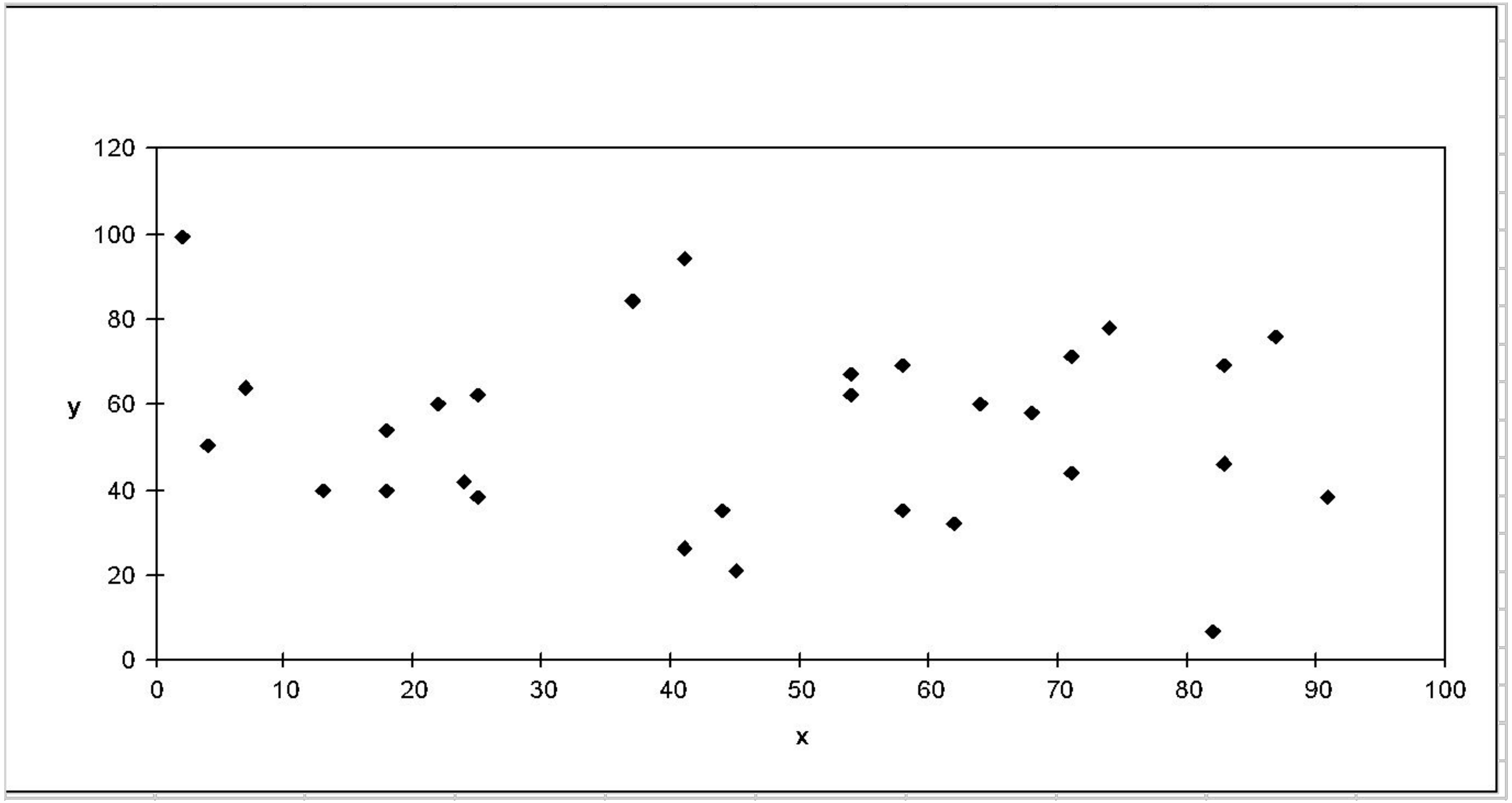
This operator is called the *two-point* crossover.

Mutation

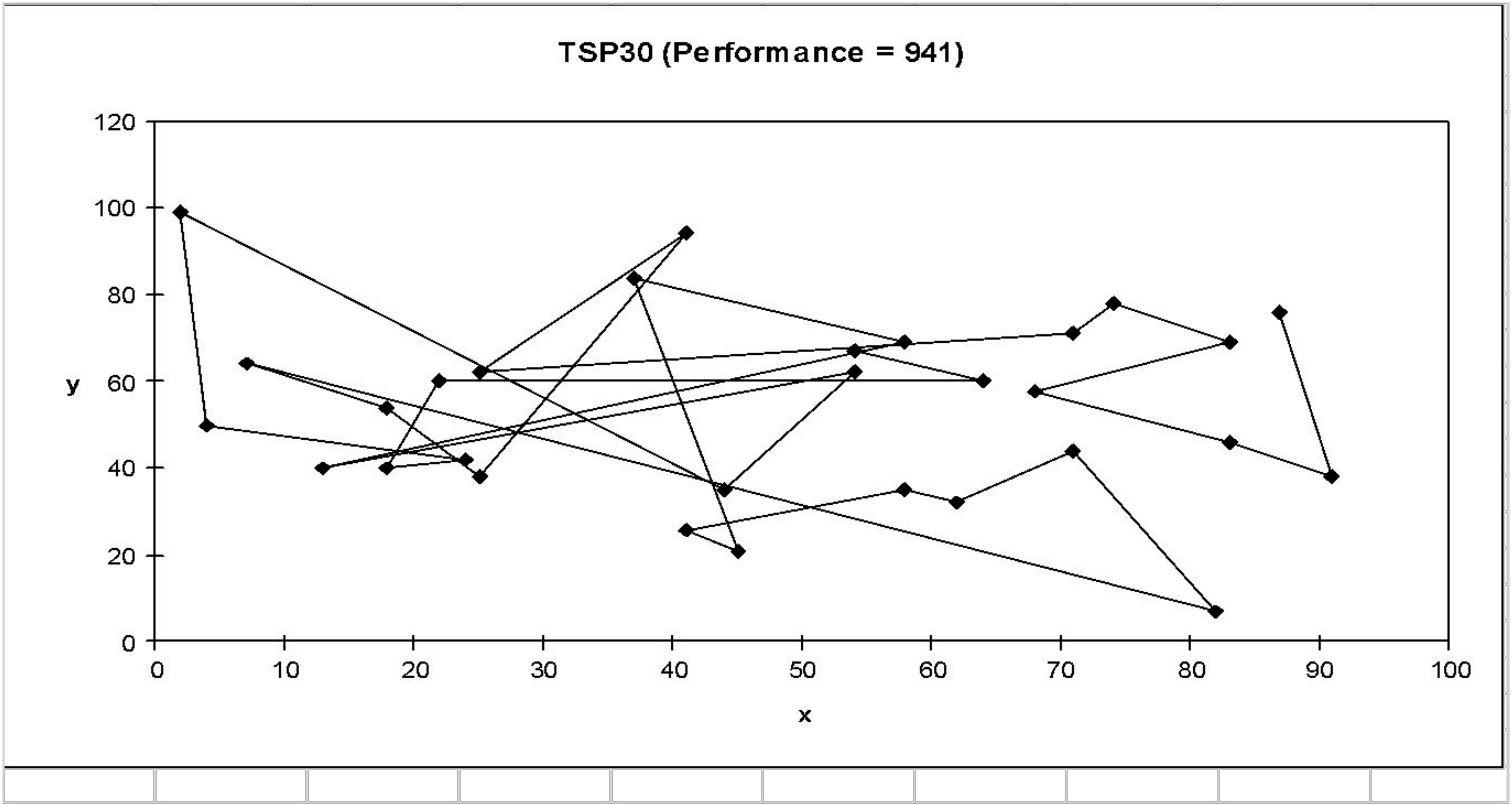
Mutation involves reordering of the list:

			*			*		
Before:	(5	8	7	2	1	6	3	4)
After:	(5	8	6	2	1	7	3	4)

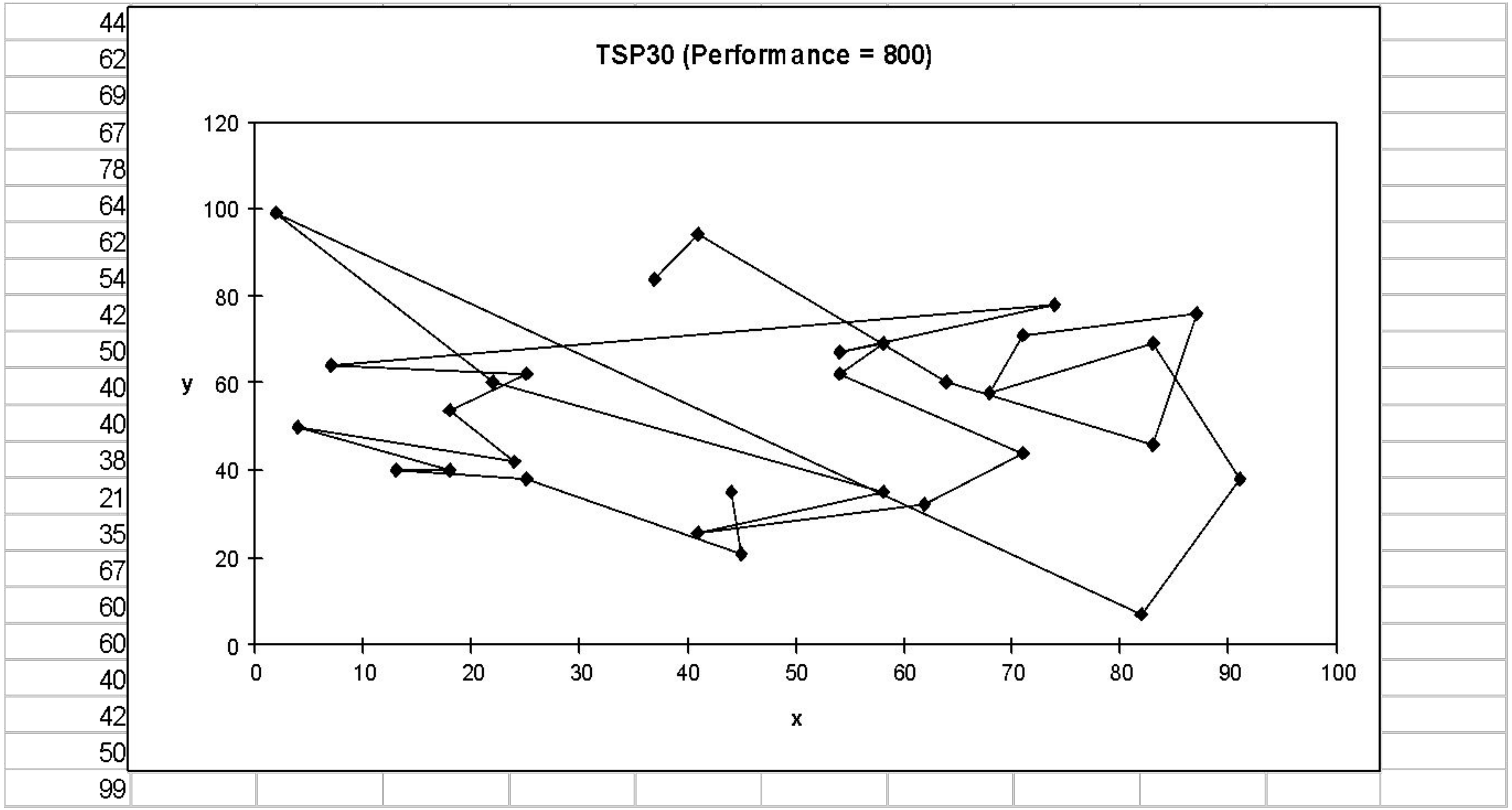
TSP Example: 30 Cities



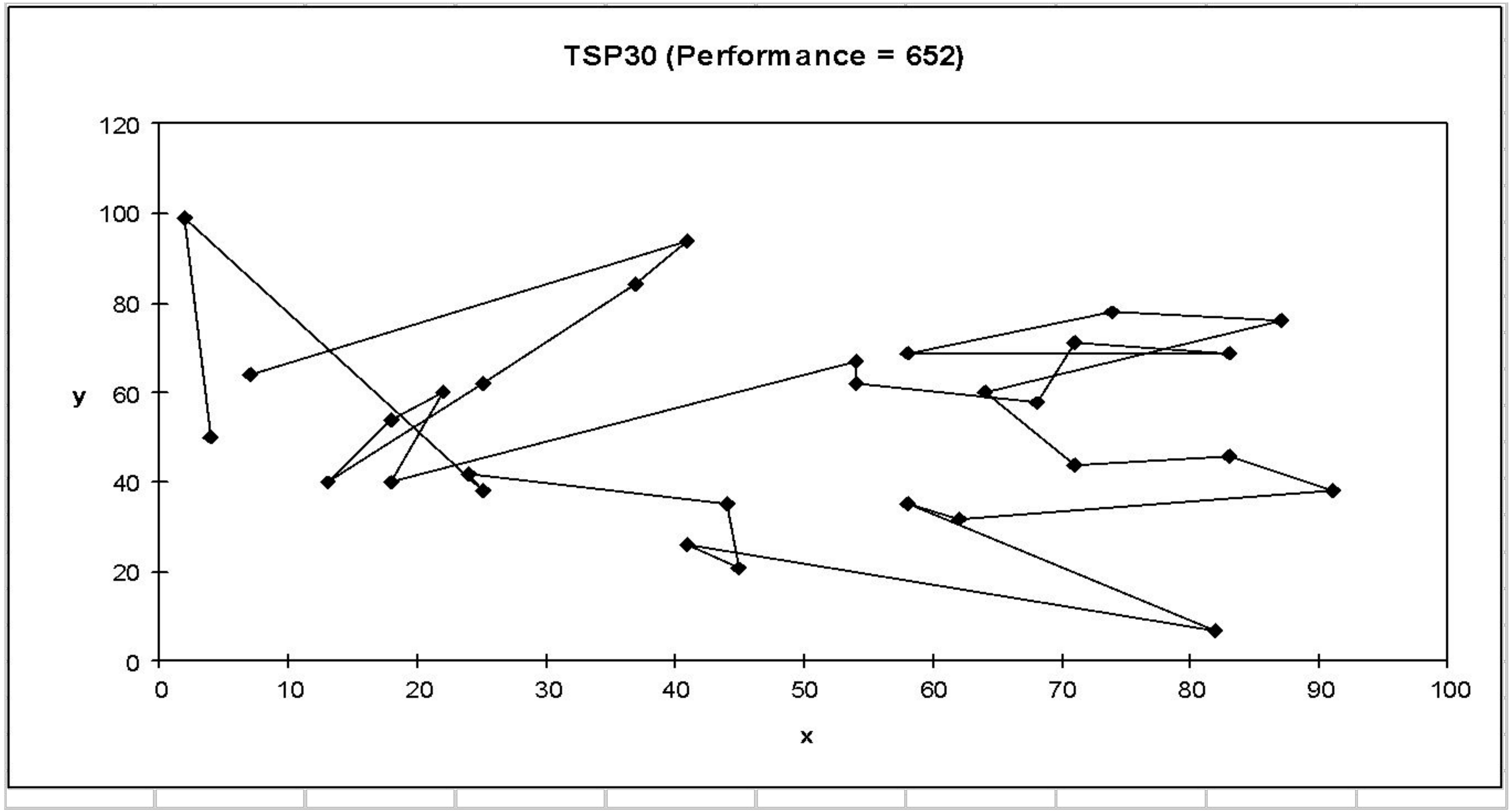
Solution $_i$ (Distance = 941)



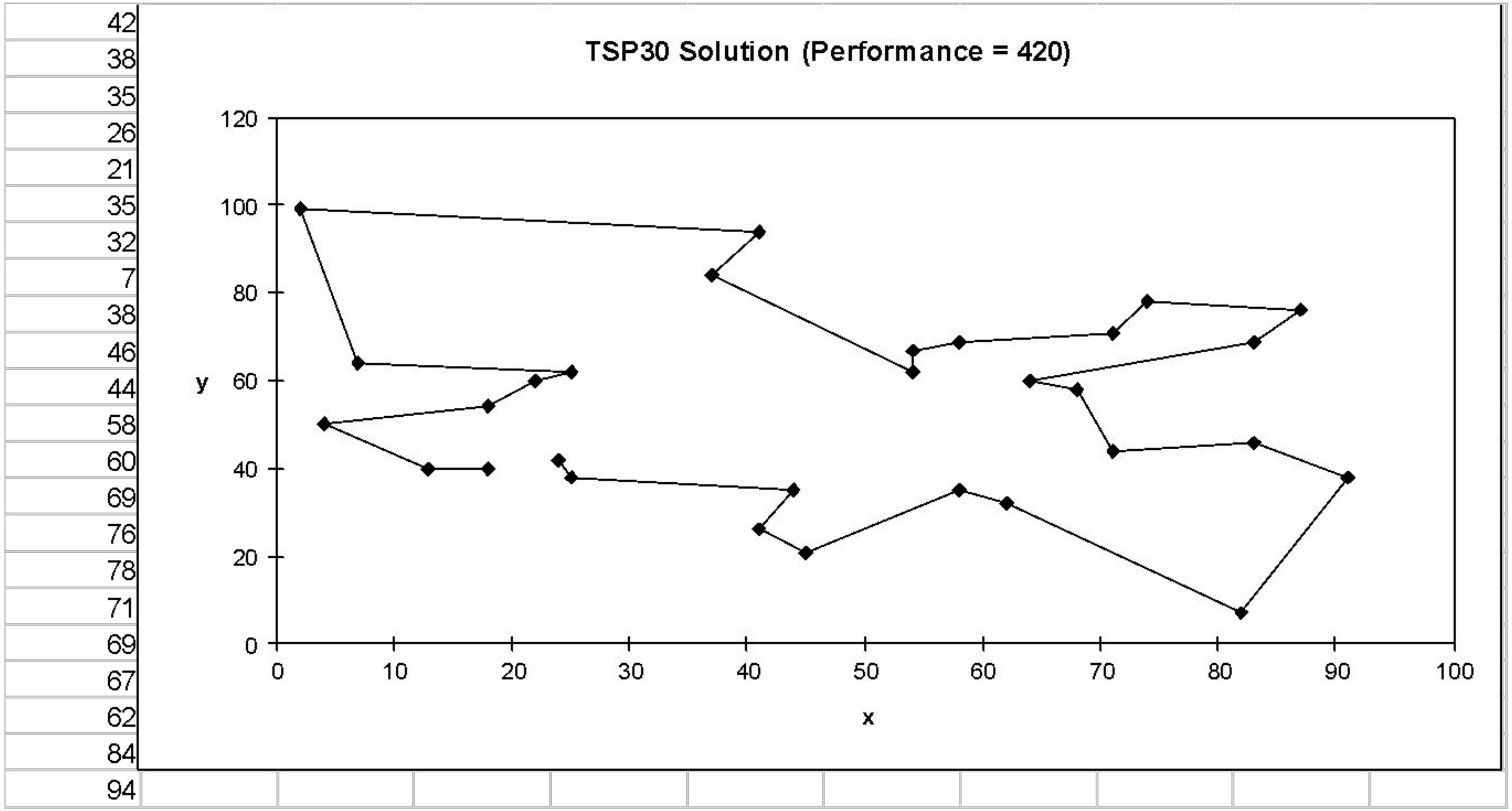
Solution $_j$(Distance = 800)



Solution $_k$(Distance = 652)



Best Solution (Distance = 420)



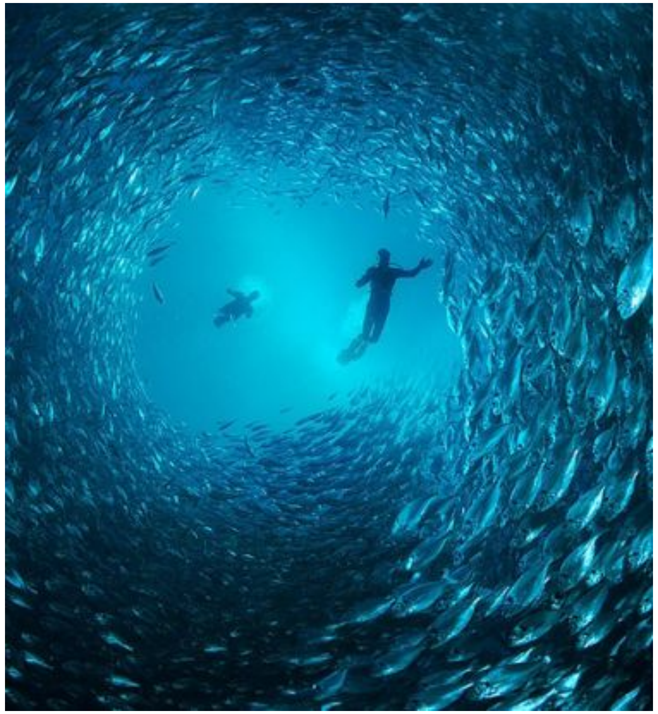
When to Use a GA

- Alternate solutions are too slow or overly complicated
- Need an exploratory tool to examine new approaches
- Problem is similar to one that has already been successfully solved by using a GA
- Want to hybridize with an existing solution
- Benefits of the GA technology meet key problem requirements

Particle Swarm Optimization

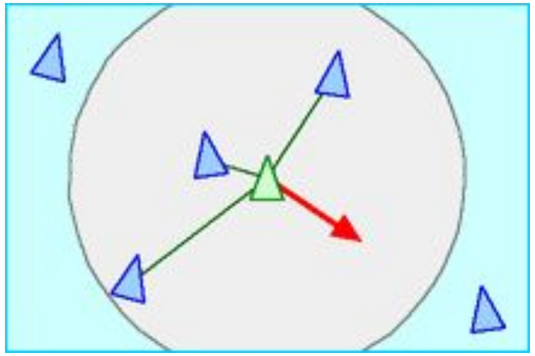
PSO: Origins

- Inspired from the nature social behavior and dynamic movements with communications of insects, birds and fish



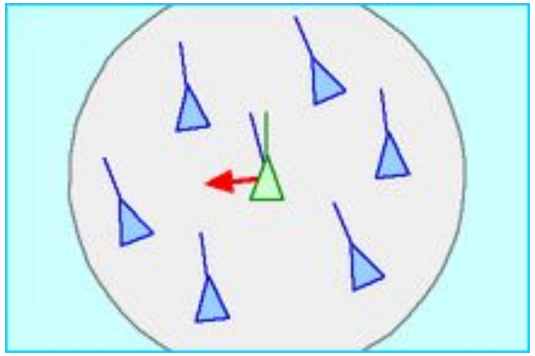
PSO: Origins

- In 1986, Craig Reynolds described this process in 3 simple behaviors:



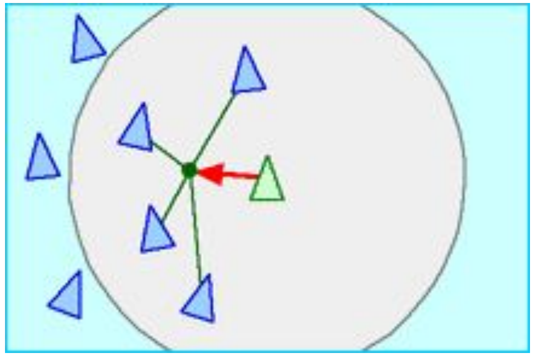
<u>Separation</u>

avoid crowding local flockmates



<u>Alignment</u>

move towards the average heading of local flockmates



<u>Cohesion</u>

move toward the average position of local flockmates

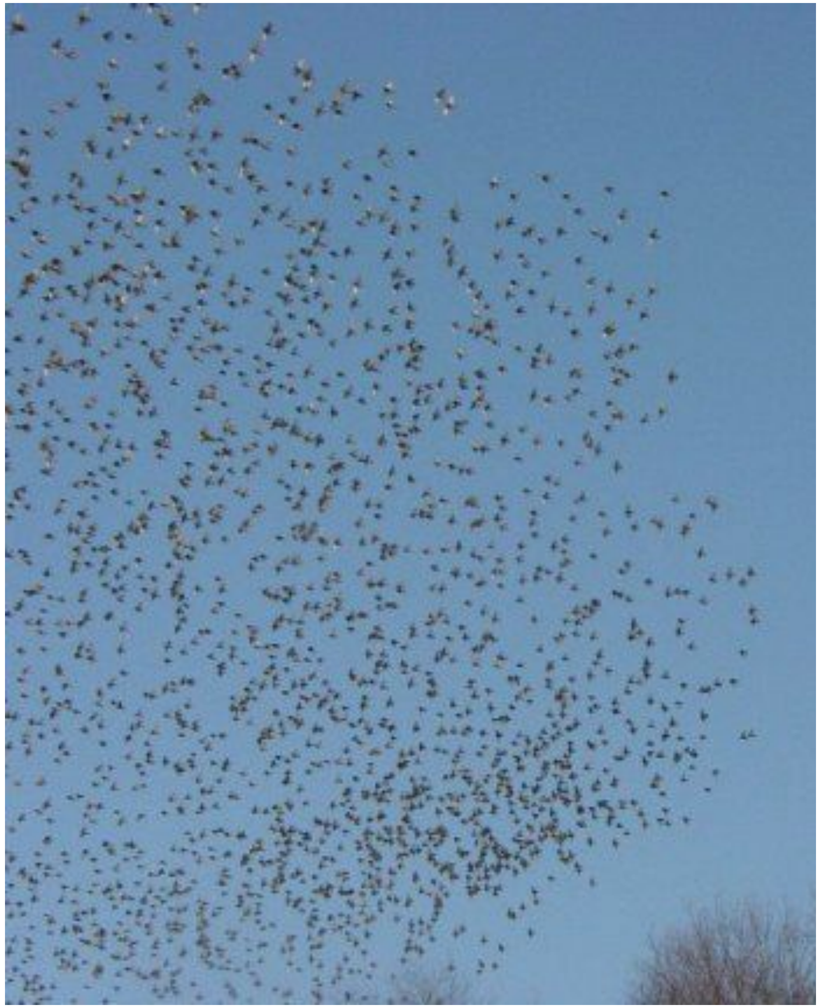
PSO: Origins

- Application to optimization: Particle Swarm Optimization
- Proposed by James Kennedy & Russell Eberhart (1995)
- Combines self-experiences with social experiences

PSO: Concept

- Uses a number of agents (particles) that constitute a swarm moving around in the search space looking for the best solution

- Each particle in search space adjusts its "flying" according to its own flying experience as well as the flying experience of other particles



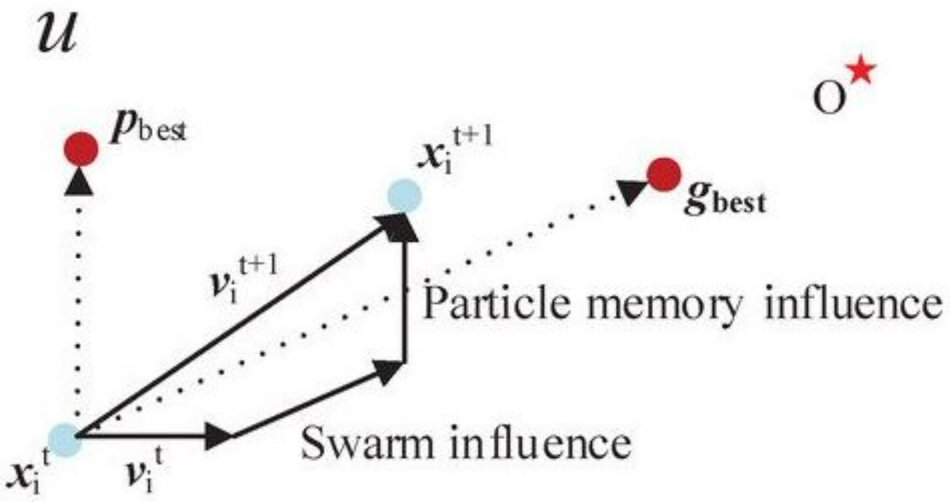
PSO: Concept

- Collection of flying particles (swarm) - Changing solutions
- Search area - Possible solutions
- Movement towards a promising area to get the global optimum
- Each particle keeps track:
  - its best solution, personal best, *pbest*
  - the best value of any particle, global best, *gbest*

PSO: Concept

- Each particle adjusts its travelling speed dynamically corresponding to the flying experiences of itself and its colleagues

- Each particle modifies its position according to:
  - its current position
  - its current velocity
  - the distance between its current position and pbest
  - the distance between its current position and gbest



PSO: Algorithm - Parameters

- Algorithm parameters
  - ***A*** : Population of agents
  - p_i : Position of agent a_i in the solution space
  - ***f*** : Objective function
  - v_i : Velocity of agent's a_i
  - $V(a_i)$: Neighborhood of agent a_i (fixed)
- The neighborhood concept in PSO is not the same as the one used in other meta-heuristics search, since in PSO each particle's neighborhood never changes (is fixed)

PSO: Algorithm

- Particle update rule

$$\boldsymbol{p = p + v} \qquad (1)$$

  - with

$$\boldsymbol{v = v + c_1 * rand * (pBest - p) + c_2 * rand * (gBest - p)} \qquad (2)$$

  - where

- p: particle's position
- v: velocity
- c_1: weight of local information
- c_2: weight of global information
- *pBest*: best position of the particle
- *gBest*: best position of the swarm
- *rand*: random variable

What a particle does

- In each timestep, a particle has to move to a new position. It does this by adjusting its velocity.
  - The adjustment is essentially this:
  - The current velocity PLUS
  - A weighted random portion in the direction of its personal best PLUS
  - A weighted random portion in the direction of the neighborhood best.
- Having worked out a new velocity, its position is simply its old position plus the new velocity.

The PSO algorithm pseudocode

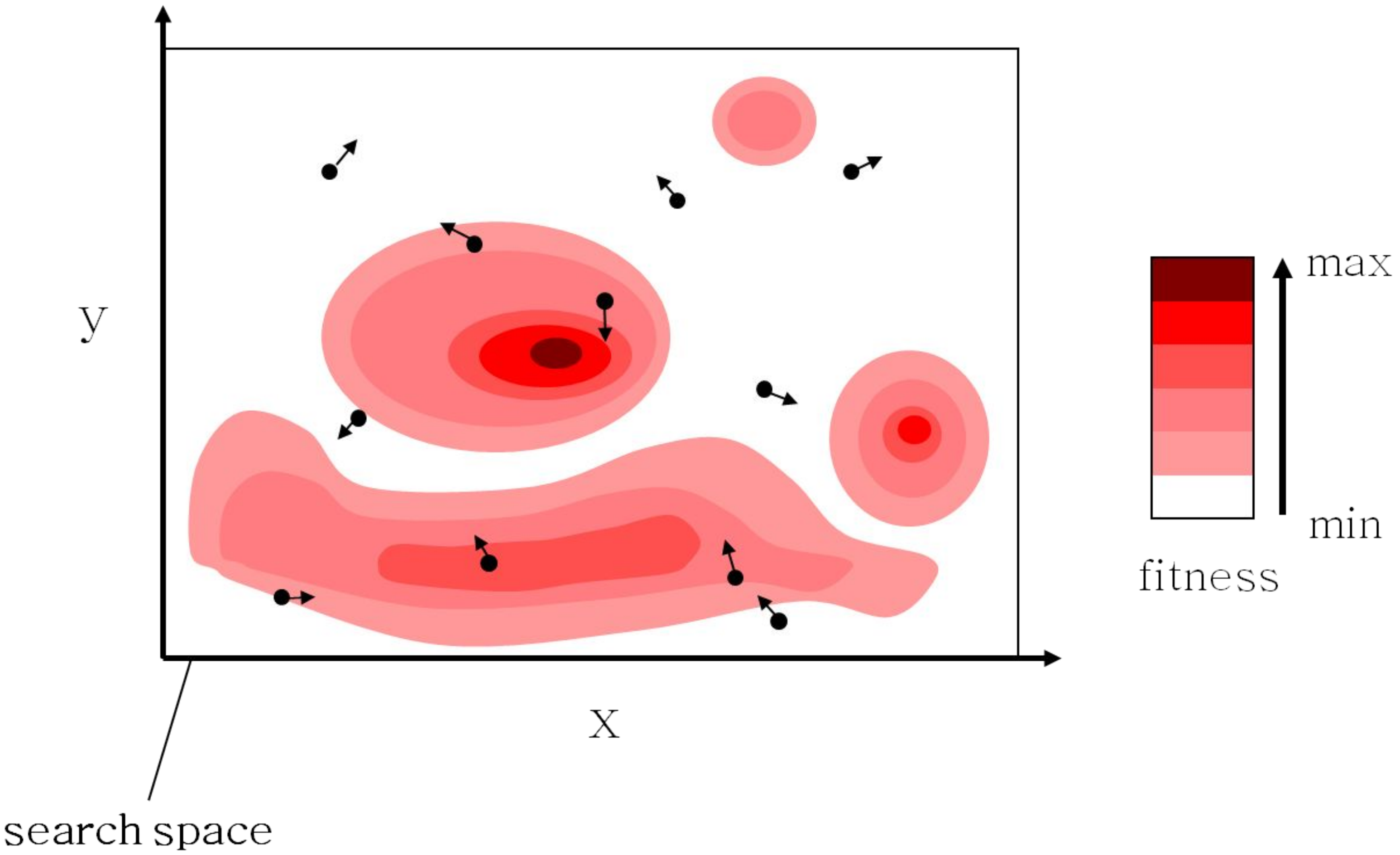
Input: Randomly initialized position and velocity of Particles:

$X_{i}(0)$ and $V_{i}(0)$

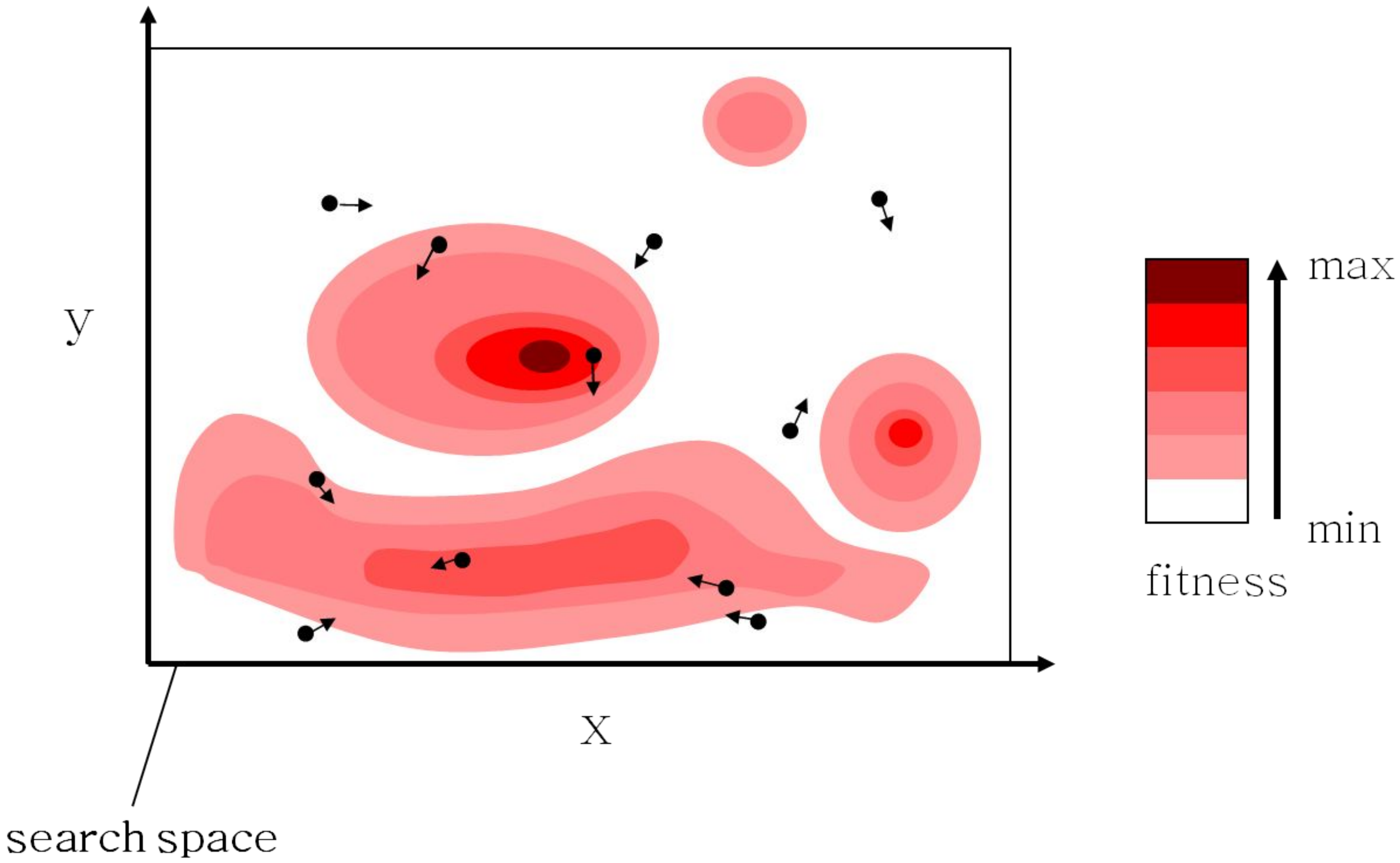
Output: Position of the approximate global minimum X^{*}

while terminating condition is not reached do
    for i = 1 to number of particles **do**
        Calculate the fitness function f
        Update personal best and global best of each particle
        Update velocity of the particle using Equation 2
        Update the position of the particle using equation 1
    end for
end while

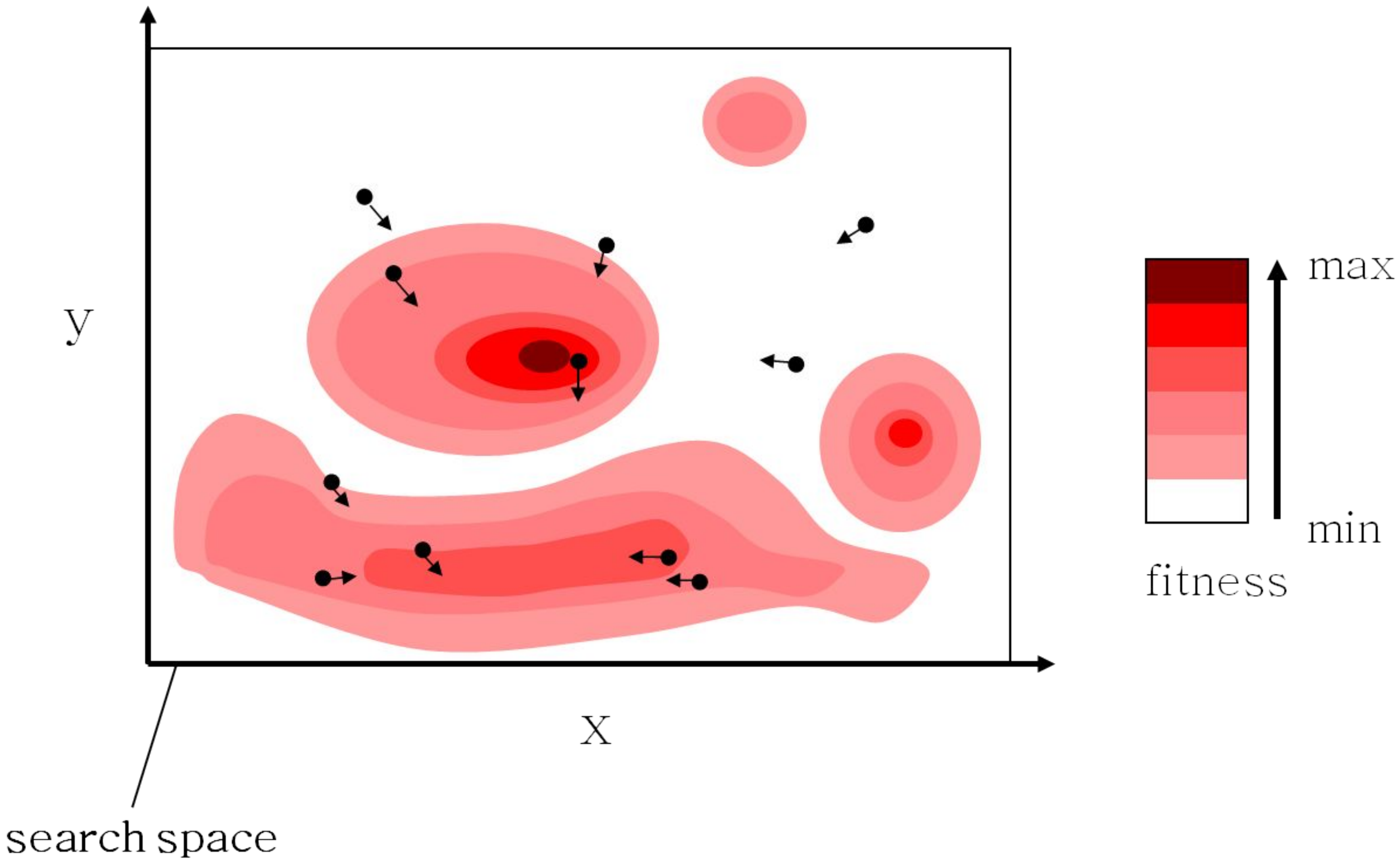
PSO: Algorithm - Example



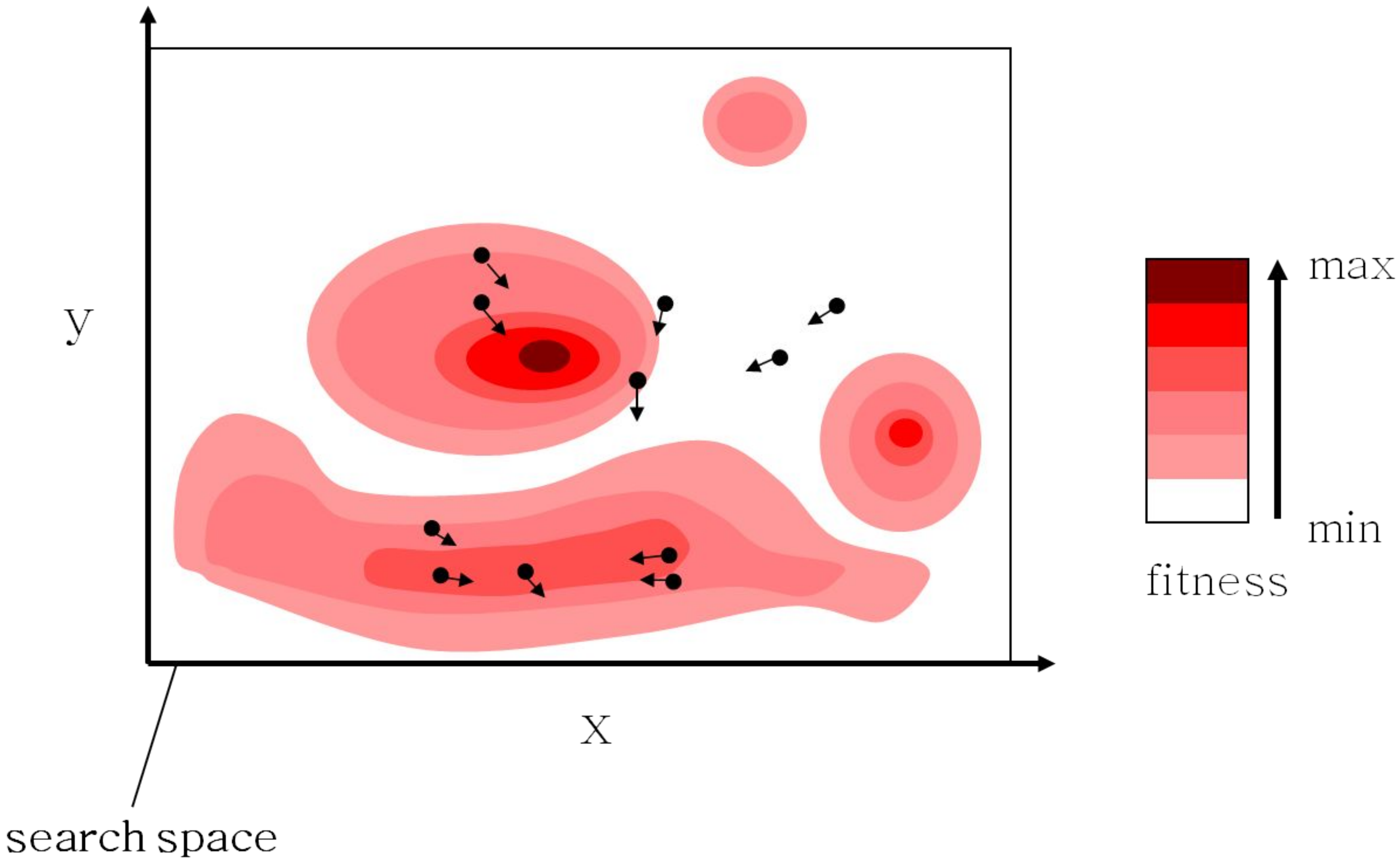
PSO: Algorithm - Example



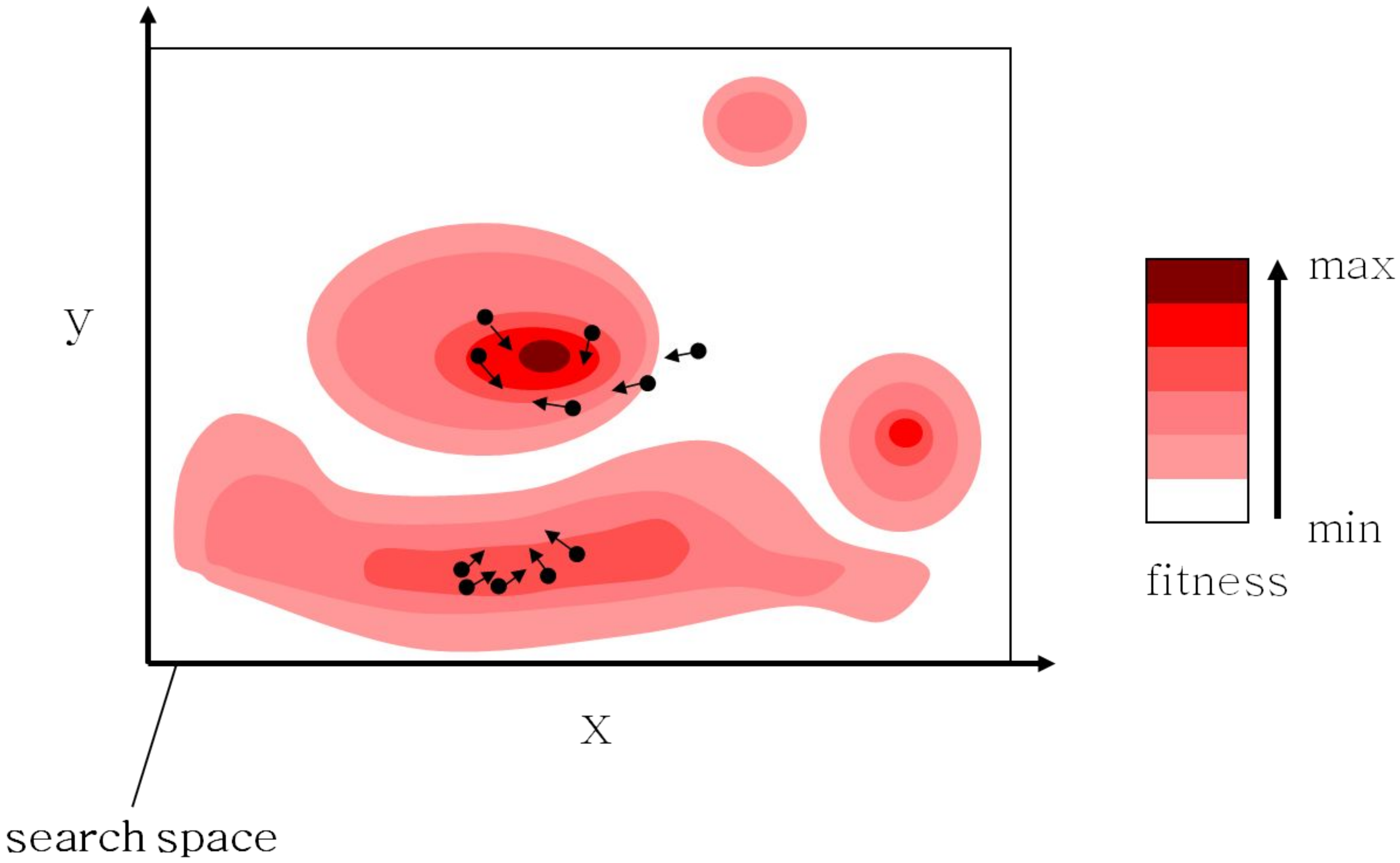
PSO: Algorithm - Example



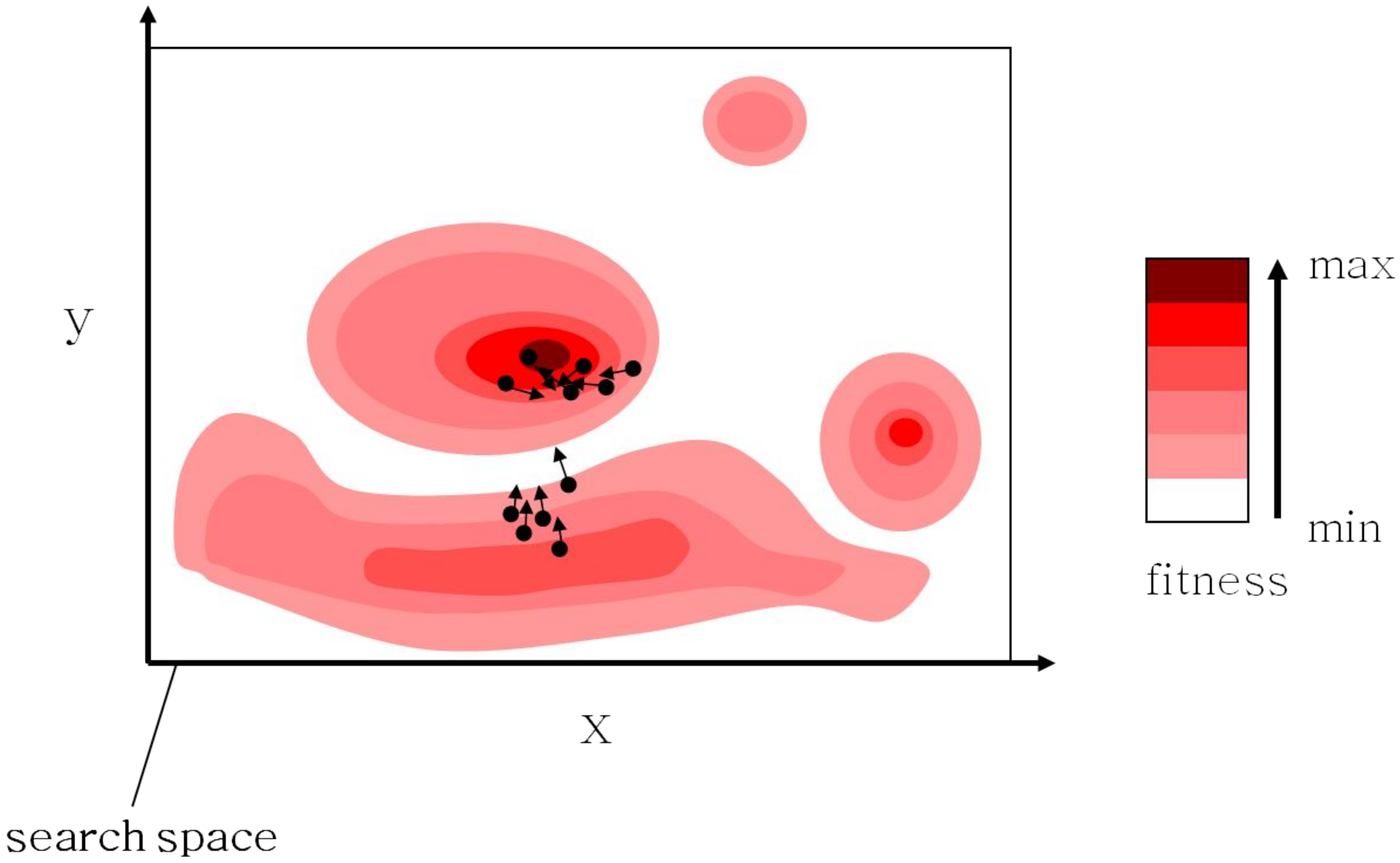
PSO: Algorithm - Example



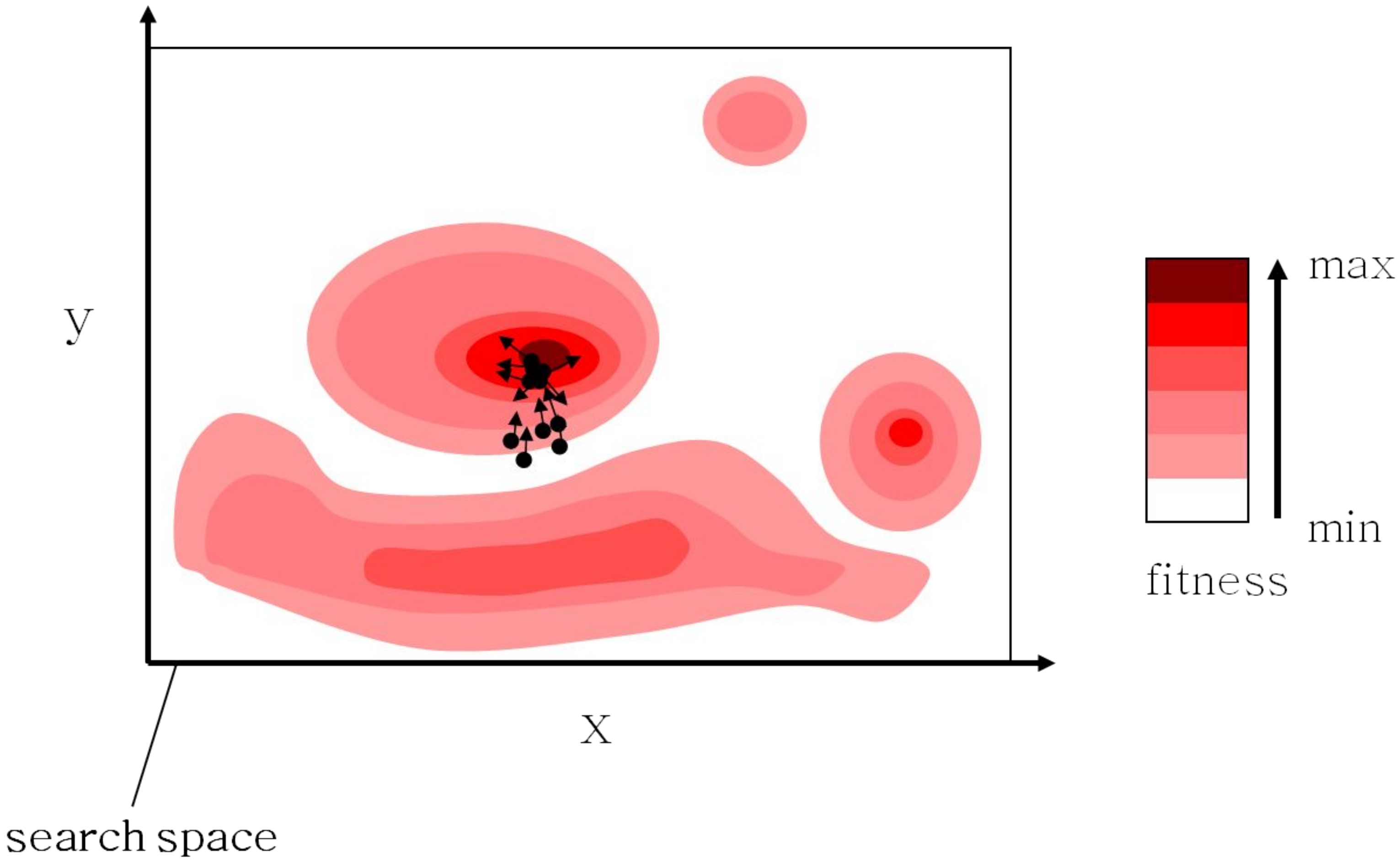
PSO: Algorithm - Example



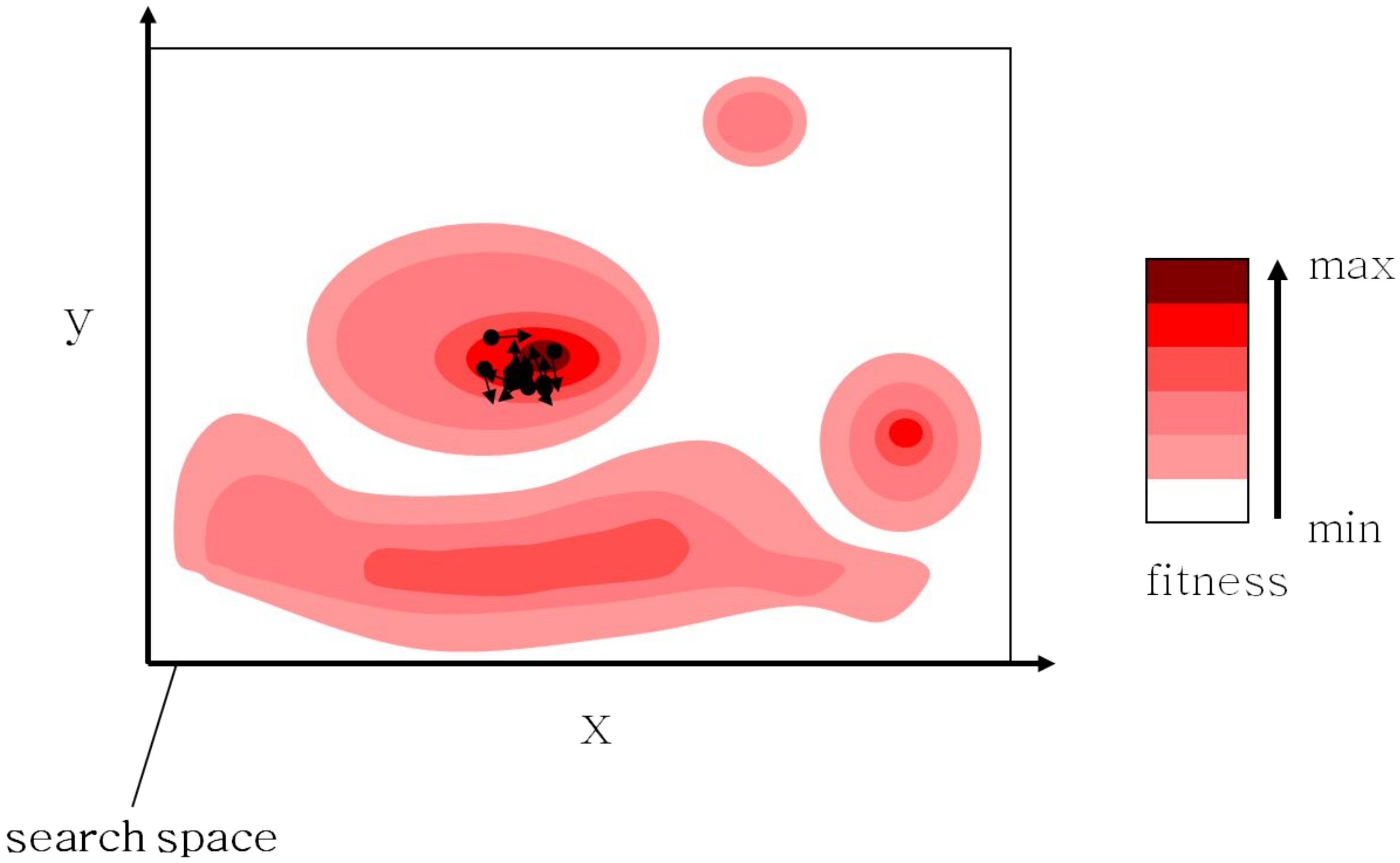
PSO: Algorithm - Example



PSO: Algorithm - Example



PSO: Algorithm - Example



Variants of PSO

- Standard PSO (Eberhart et al. (1996))

$$v_{t+1} = v_i + \varphi_1 \beta_1 (p_i - x_i) + \varphi_2 \beta_2 (p_g - x_i)$$

$$x_{t+1} = x_t + v_{t+1}$$

- Modified PSO (Shi and Eberhart (1998))

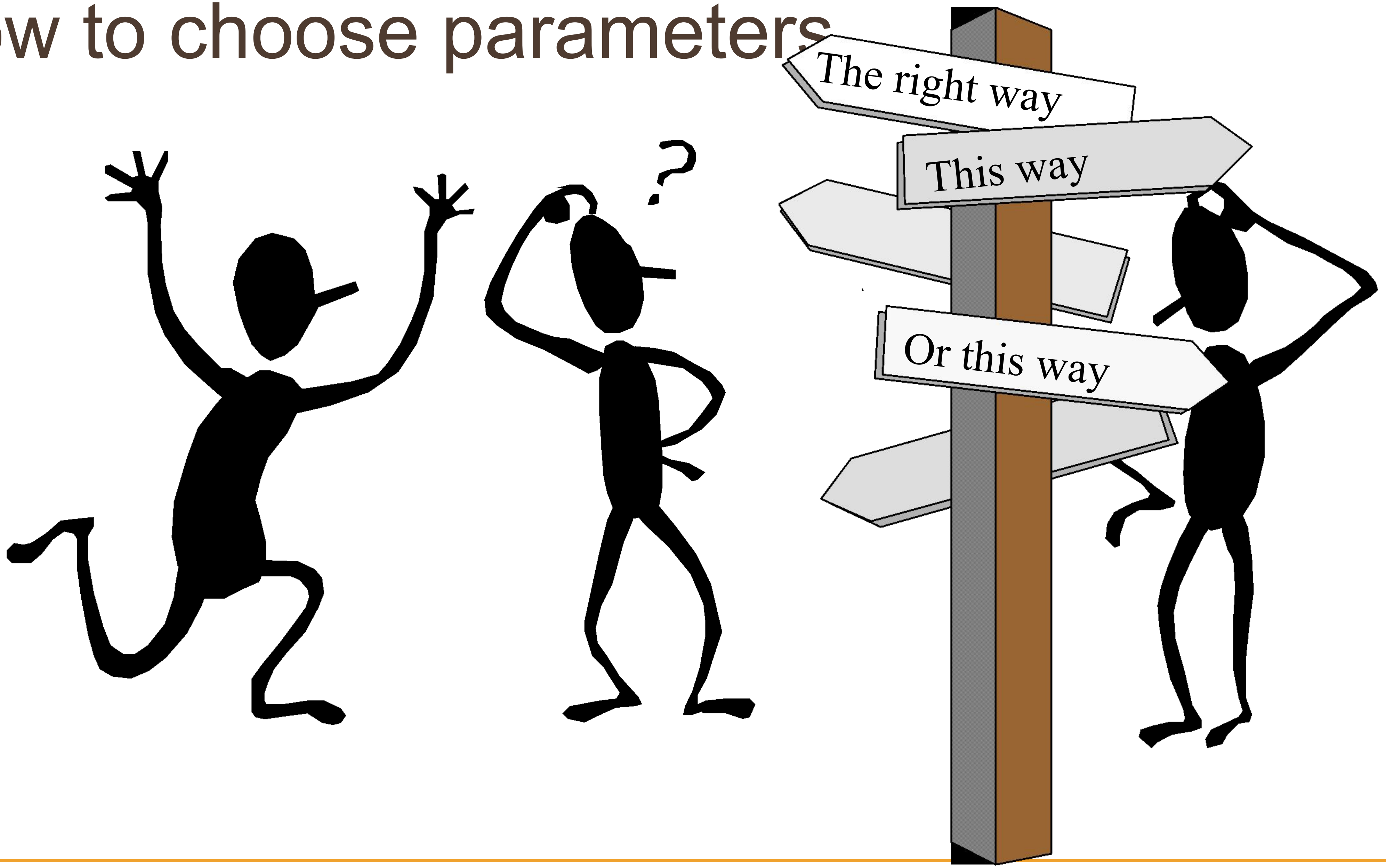
$$v_{t+1} = \omega v_t + \varphi_1 \beta_1 (p_i - x_i) + \varphi_2 \beta_2 (p_g - x_i)$$

- PSO(Clerc and Kennedy (developed 1999)- applied a constriction factor, v,

$$v_{t+1} = \chi \{ v_t + \varphi_1 \beta_1 (p_i - x_i) + \varphi_2 \beta_2 (p_g - x_i) \}$$

$$\chi = \frac{2}{\left| 2 - \varphi - \sqrt{\varphi^2 - 4\varphi} \right|} \quad \text{where} \quad \varphi = \varphi_1 + \varphi_2, \varphi > 4$$

How to choose parameters



Parameters

- Number of particles: (10—50) are reported as usually sufficient.
- C1 (importance of personal best)
- C2 (importance of neighborhood best)

  Usually C1+C2 = 4. No good reason other than empiricism
- v – too low: too slow
- v – too high: too unstable

PSO: Algorithm Characteristics

- Advantages
  - Insensitive to scaling of design variables
  - Simple implementation
  - Easily parallelized for concurrent processing
  - Derivative free
  - Very few algorithm parameters
  - Very efficient global search algorithm

- Disadvantages
  - Tendency to a fast and premature convergence in mid optimum points
  - Slow convergence in refined search stage (weak local search ability)