Swarm and Evolutionary Algorithms

Unit 4: Genetics & Evolution, Genetic Algorithms, Ant Colony & Particle Swarm Optimization

Dr. Gopal Chandra Jana

Dept. of CSE, Sharda University
Soft Computing (CSA301) — Unit 4

July 20, 2026

Course Page:
https://www.gcjana.in/courses/shardauniversity/2601/soft-computing/

Outline

Learning Outcomes of Unit 4

After this unit, a student will be able to...

1. **Explain** the biological ideas of genetics and evolution and map them to probabilistic search.
2. **Describe** the basic Genetic Algorithm (GA) framework and its variants.
3. **Apply** the GA operators — encoding, selection, crossover, mutation — and **solve** single-objective optimisation problems numerically.
4. **Explain** Ant Colony Optimization (ACO) and its transition rule.
5. **Explain** Particle Swarm Optimization (PSO) and its velocity/position update.
6. **Compare** evolutionary and swarm methods and choose appropriately.

Mapping to the Syllabus

A. Genetics & evolution $\rightarrow$ probabilistic search. **B.** GA framework, architectures, operators; single-objective optimisation. **C.** Swarm optimisation: ACO, PSO.

From Biology to Optimisation

Darwinian evolution

Nature optimises species over generations through:

- **Variation** — individuals differ.
- **Inheritance** — traits pass to offspring.
- **Selection** — the fitter survive and reproduce.

Over time the population **adapts** to its environment.

Genetics — the mechanism

- **Gene**: a unit of heredity.
- **Chromosome**: a string of genes.
- **Crossover**: parents recombine chromosomes.
- **Mutation**: random change in a gene.

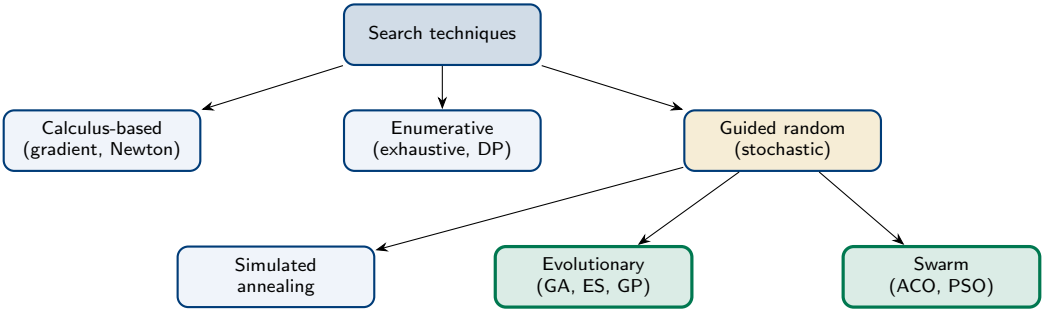
Biology ↔ GA vocabulary

Biology	**Genetic Algorithm**
Individual	Candidate solution
Chromosome	Encoded string
Gene	Variable / bit
Allele	Value of a gene
Fitness	Objective value
Population	Set of solutions
Generation	Iteration

The leap

If nature can *search* the space of organisms by evolution, we can search the space of *solutions* the same way.

Search Techniques — Where GAs Fit



Why not gradient methods?

They need derivatives, get stuck in **local optima**, and fail on discontinuous, noisy or combinatorial landscapes.

Probabilistic search

GAs and swarms use **randomness with direction**: they sample many points in parallel and bias the search toward better regions — robust and derivative-free.

Exploration vs. Exploitation

Two competing needs

- **Exploration**: search new, unknown regions (diversity) — provided by **mutation** and randomness.
- **Exploitation**: refine around known-good regions — provided by **selection** and **crossover**.

The balance

Too much exploration ⇒ random walk (never converges).
Too much exploitation ⇒ **premature convergence** to a local optimum.
Good algorithms *balance* the two.

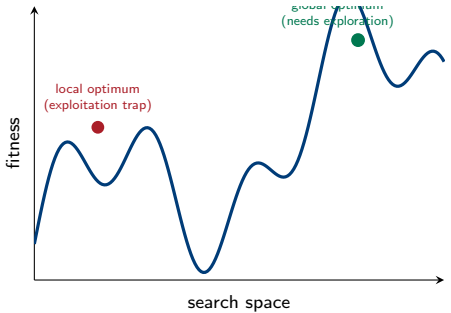


Figure: A multi-modal landscape — exploration is needed to escape the local optimum and find the global one.

The Basic GA Framework

Pseudocode

1. **Initialise** a random population.
2. **Evaluate** the fitness of each individual.
3. **Repeat** until a stopping criterion:
   - **Select** parents (fitter $\rightarrow$ more likely).
   - **Crossover** parents $\rightarrow$ offspring.
   - **Mutate** offspring.
   - **Evaluate** & form the new generation.
4. **Return** the best individual found.

Stopping criteria

max generations, fitness threshold reached, or no improvement for k generations.

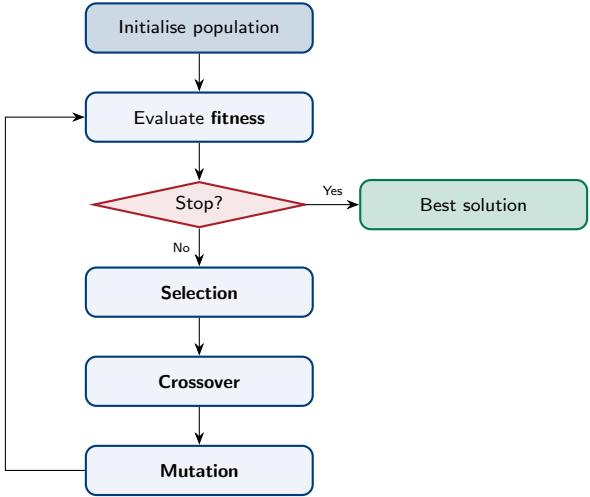


Figure: The GA cycle — selection, crossover, mutation repeat each generation.

GA Operator 1 — Encoding (Representation)

Common encodings

- **Binary**: `1 0 1 1 0` — classic, simple crossover.
- **Real-valued**: $[2.7,\ 0.4,\ -1.1]$ — for continuous problems.
- **Permutation**: $(3, 1, 4, 2)$ — for ordering (TSP, scheduling).
- **Tree**: expressions — genetic programming.

Binary example

A 5-bit string encodes an integer $x \in [0, 31]$:
`01101` $\rightarrow 0{\cdot}16 + 1{\cdot}8 + 1{\cdot}4 + 0{\cdot}2 + 1{\cdot}1 = \mathbf{13}$.

Decoding a bounded real variable

For $x \in [x_{\min}, x_{\max}]$ with an L-bit string of decimal value d:

$$x = x_{\min} + \frac{d}{2^L - 1}\,(x_{\max} - x_{\min})$$

Worked

$L = 5$, $x \in [0, 10]$, string `10110` $\Rightarrow d = 22$:

$$x = 0 + \frac{22}{31} \times 10 \approx \mathbf{7.10}$$

Design rule

The encoding must make *small genetic changes* produce *small solution changes* (good "locality") — otherwise search becomes random.

GA Operator 2 — Selection

Roulette-wheel (fitness proportionate)

Probability of selecting individual i:

$$p_i = \frac{f_i}{\sum_j f_j}$$

Fitter individuals occupy a larger slice of the wheel.

Other schemes

- **Tournament**: pick k at random, keep the best.
- **Rank**: select by rank, not raw fitness (avoids domination by a super-individual).
- **Elitism**: copy the best few unchanged to the next generation.

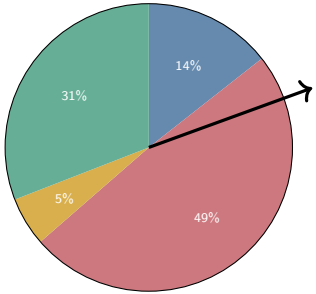


Figure: Roulette wheel for the $f = x^2$ example (next slide). The pointer lands in a slice with probability $\propto$ fitness.

Numerical Example — Selection (Maximise $f(x) = x^2$)

Population of four 5-bit strings on $x \in [0, 31]$

String	x	$f(x) = x^2$	$p_i = f_i / \sum f$	Expected copies $p_i \cdot n$	% of wheel
01101	13	169	0.1444	0.58	14.4
11000	24	576	0.4923	1.97	49.2
01000	8	64	0.0547	0.22	5.5
10011	19	361	0.3085	1.23	30.9
Sum		**1170**	1.000	4.00	100

Reading the result

The best string 11000 ($x = 24$) claims $\approx 49\%$ of the wheel, so it is expected to be selected about **twice**; the weakest, 01000 ($x = 8$), almost **disappears**.

Cumulative (for spinning)

$0.1444 \to 0.6368 \to 0.6915 \to 1.0000$.
A random $r \in [0, 1)$ selects the first cumulative bin $\geq r$.

GA Operator 3 — Crossover (Recombination)

Binary crossover types

- **Single-point**: split at one locus, swap tails.
- **Two-point**: swap the middle segment.
- **Uniform**: swap each gene with probability 0.5.

Applied with crossover probability p_c (typically 0.6–0.9).

Single-point example ($k = 4$)

P1: 0110 1 $(x = 13)$
P2: 1100 0 $(x = 24)$

C1: 0110 0 $(x = \mathbf{12})$
C2: 1100 1 $(x = \mathbf{25})$

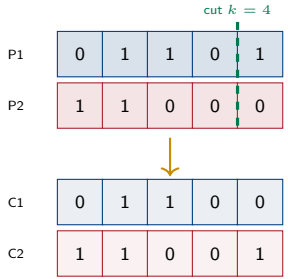


Figure: Single-point crossover swaps the tails after the cut point.

GA Operator 4 — Mutation

Purpose

Mutation injects **diversity**, letting the search reach genes that selection and crossover alone could never produce — it prevents **premature convergence**.

Forms

- **Bit-flip** (binary): flip a bit with probability p_m.
- **Gaussian** (real): add small random noise.
- **Swap / inversion** (permutation): exchange positions.

Typical p_m is small: $\approx 1/L$ (one bit per string on average).

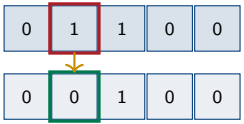
Bit-flip example

Take child 01100 ($x = 12$); flip bit position 2:

$$01\underline{1}00 \;\rightarrow\; 00\underline{1}00 = 4$$

Balance again

p_m too high $\Rightarrow$ random search (loses good genes).
p_m too low $\Rightarrow$ stagnation, stuck in a local optimum.



Putting It Together — One GA Generation (Trace)

Maximise $f(x) = x^2$, $x \in [0, 31]$ — from selection to next generation

Parents (mating pool)	Crossover site	Offspring	x	$f = x^2$	Note
01101 × 11000	$k = 4$	01100	12	144	
01101 × 11000	$k = 4$	11001	25	625	new best!
10011 × 11000	$k = 2$	10000	16	256	
10011 × 11000	$k = 2$	11011	27	729	new best!

Progress

Parent best was $f(24) = 576$. After one generation of crossover the offspring reach $f(27) = \mathbf{729}$ — the population has **improved**.

Then

A small mutation (e.g. flipping a low-order bit) perturbs one offspring, keeping diversity for the next round. The cycle repeats.

GA Architectures & Parameters

Architectures

- **Generational** GA: replace the whole population each generation.
- **Steady-state** GA: replace only a few individuals per step.
- **Elitist** GA: always carry the best forward.
- **Island / parallel** GA: sub-populations evolve separately and occasionally *migrate*.

Schema idea (Holland)

Short, low-order, above-average *schemata* (building blocks) receive exponentially more trials — the intuition behind why GAs work.

Typical control parameters

Parameter	Typical value
Population size N	20–200
Crossover rate p_c	0.6–0.9
Mutation rate p_m	0.001–0.05 ($\approx 1/L$)
Selection	tournament / roulette
Elitism	keep top 1–2
Generations	problem dependent

Note

No single setting is best for all problems (*No Free Lunch* theorem) — parameters are tuned per problem.

Quick Check — Round 1 (Genetics & GA)

[Q1]

Name the four GA operators in the order they are applied each generation.

[Q2]

A population has fitnesses $\{10, 30, 40, 20\}$. What is the roulette-wheel selection probability of the third individual?

[Q3]

Single-point crossover at $k = 3$ of 10110 and 01001 gives which two children?

[Q4]

What problem does mutation guard against, and what happens if p_m is too large?

Answers — Round 1

[A1]

Selection → **Crossover** → **Mutation** → (evaluate &) **replacement**. Encoding is done once, at initialisation.

[A2]

$\sum f = 10 + 30 + 40 + 20 = 100;\ \ p_3 = \dfrac{40}{100} =$ **0.40** (40%).

[A3]

Swap tails after position 3: `101` — `10` and `010` — `01` give `10101` and `01010`.

[A4]

It guards against **premature convergence** (loss of diversity). If p_m is too large the GA degenerates into a **random search** and loses good building blocks.

Single-Objective Optimisation with a GA — Case Study

Problem: the 0/1 Knapsack

Choose items to **maximise value** without exceeding a weight capacity W.

- **Encoding**: a bit per item (1=take).
- **Fitness**: total value, with a penalty if weight $> W$.
- **Operators**: uniform crossover, bit-flip mutation.

Why GA?

n items $\Rightarrow 2^n$ subsets. For $n = 50$ that is $\approx 10^{15}$ — exhaustive search is infeasible, but a GA finds a near-optimal pack quickly.

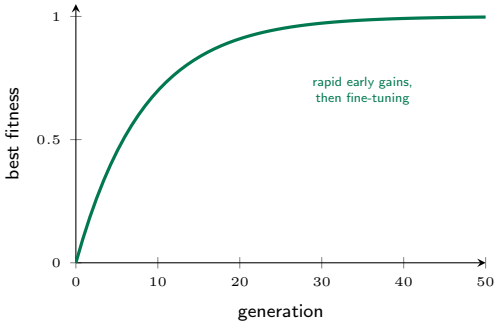


Figure: A typical GA convergence curve — best fitness rises quickly, then plateaus near the optimum.

Swarm Intelligence — The Idea

Definition

Swarm Intelligence is collective, intelligent behaviour that **emerges** from many *simple* agents following *simple local rules*, with no central controller.

Principles

- **Decentralised** — no leader.
- **Local interaction** — agents sense neighbours / environment.
- **Self-organisation** — global order from local rules.
- **Stigmergy** — indirect communication via the environment (e.g. pheromone).

Nature's examples

Ant trails, bird flocking, fish schooling, bee foraging — none has a boss, yet all solve hard collective problems.

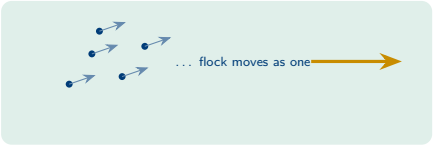


Figure: Simple agents + local rules ⇒ global behaviour.

Ant Colony Optimization (ACO)

Biological inspiration

Ants deposit **pheromone** on paths. Shorter paths are traversed faster, accumulate *more* pheromone, and attract *more* ants — a positive-feedback loop that converges on the shortest route. (Dorigo, 1992.)

Transition rule

Probability that an ant at node i moves to j:

$$p_{ij} = \frac{[\tau_{ij}]^{\alpha}\,[\eta_{ij}]^{\beta}}{\sum_{k\in\text{allowed}}[\tau_{ik}]^{\alpha}\,[\eta_{ik}]^{\beta}}$$

τ = pheromone, $\eta = 1/d$ = heuristic (visibility), α, β weight them.

Pheromone update

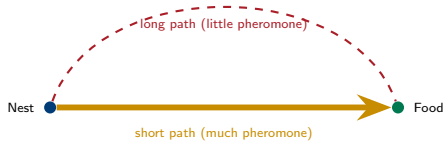


Figure: Positive feedback — the shorter path wins more pheromone and becomes dominant.

Great for

routing, TSP, scheduling, network optimisation — discrete/combinatorial problems.

Numerical Example — ACO Transition Probability

An ant at node i can go to A, B or C

With $\alpha = 1$, $\beta = 2$ and $\eta = 1/d$:

Edge	τ	d	$\eta = 1/d$
A	2.0	2.0	0.50
B	4.0	1.0	1.00
C	1.0	4.0	0.25

Compute $[\tau]^{\alpha}[\eta]^{\beta}$

$$A : 2.0^1 \cdot 0.50^2 = 2.0(0.25) = 0.500$$
$$B : 4.0^1 \cdot 1.00^2 = 4.0(1.00) = 4.000$$
$$C : 1.0^1 \cdot 0.25^2 = 1.0(0.0625) = 0.0625$$
$$\Sigma = 4.5625$$

Probabilities

$$p_A = \frac{0.500}{4.5625} = \mathbf{0.110}$$
$$p_B = \frac{4.000}{4.5625} = \mathbf{0.877}$$
$$p_C = \frac{0.0625}{4.5625} = \mathbf{0.014}$$

(Sum $= 1.000$.)

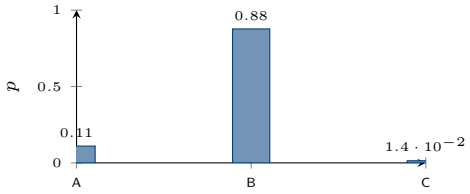


Figure: Edge B dominates — short distance *and* high pheromone.

Particle Swarm Optimization (PSO)

Biological inspiration

Models a **flock of birds** searching for food. Each *particle* is a candidate solution that flies through the search space, pulled toward its own best and the swarm's best. (Kennedy & Eberhart, 1995.)

Velocity update

$$v_i \leftarrow \underbrace{w\, v_i}_{\text{inertia}} + \underbrace{c_1 r_1 (p_i - x_i)}_{\text{cognitive}} + \underbrace{c_2 r_2 (g - x_i)}_{\text{social}}$$

Position update

$$x_i \leftarrow x_i + v_i$$

p_i = particle's personal best; g = global best; $r_1, r_2 \sim U(0,1)$; w = inertia; c_1, c_2 = acceleration coefficients

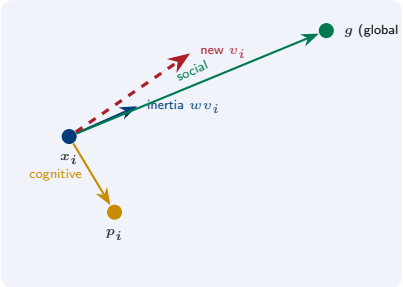


Figure: Each particle's motion blends inertia, its own best pull, and the swarm best pull.

Numerical Example — PSO Velocity Update

One 1-D particle

Given at the current step:

$$x_i = 2.0, \quad v_i = 1.0$$
$$p_i = 4.0, \quad g = 5.0$$
$$w = 0.7, \quad c_1 = c_2 = 1.5$$
$$r_1 = 0.6, \quad r_2 = 0.4$$

Step 1 — three components

$$\text{inertia} = 0.7 \times 1.0 = 0.70$$
$$\text{cognitive} = 1.5 \times 0.6 \times (4.0 - 2.0) = 1.80$$
$$\text{social} = 1.5 \times 0.4 \times (5.0 - 2.0) = 1.80$$

Step 2 — new velocity

$$v_i^{\text{new}} = 0.70 + 1.80 + 1.80 = \mathbf{4.30}$$

Step 3 — new position

$$x_i^{\text{new}} = x_i + v_i^{\text{new}} = 2.0 + 4.30 = \mathbf{6.30}$$

Interpretation

Both the personal best (4.0) and the global best (5.0) lie to the *right* of $x = 2$, so the particle accelerates rightward, overshooting slightly — velocity clamping (a $v_{\max}$) is often applied to keep steps bounded.

GA vs. ACO vs. PSO — Comparison

	Genetic Algorithm	ACO	PSO
Inspiration	Evolution / genetics	Ant foraging (pheromone)	Bird flocking
Encoding	Chromosomes (strings)	Paths on a graph	Position vectors
Memory	Population only	Pheromone trails	Personal + global best
Best for	Discrete & continuous	Discrete / combinatorial	Continuous
Operators	Selection, crossover, mutation	Trail update, evaporation	Velocity/position update
Key params	p_c, p_m, N	α, β, ρ	w, c_1, c_2

Common DNA

All three are **population-based**, **derivative-free**, **stochastic** global optimisers that balance exploration and exploitation — members of the soft computing family from Unit 1.

Applications of Evolutionary & Swarm Methods

Genetic Algorithms

- Travelling salesman, vehicle routing, scheduling
- Feature selection & hyper-parameter tuning in ML
- Engineering design, antenna & structural optimisation
- Game playing, neural-network evolution (neuro-evolution)

Ant Colony Optimization

- Network & packet routing, load balancing
- TSP and combinatorial optimisation
- Job-shop scheduling, assignment problems

Particle Swarm Optimization

- Continuous function optimisation
- Neural-network weight training
- Power-system economic dispatch, MPPT
- Controller (PID / fuzzy) parameter tuning

Hybrids & relatives

GA+PSO, memetic algorithms, differential evolution, artificial bee colony, firefly, cuckoo search — and **NSGA-II** for *multi-objective* problems (Unit 5).

Quick Check — Round 2 (ACO & PSO)

[Q1]

In ACO, what does *pheromone evaporation* (the $(1-\rho)$ term) achieve?

[Q2]

Write the PSO velocity update and name its three components.

[Q3]

A PSO particle has $x = 1,\ v = 2,\ p = 3,\ g = 6,\ w = 0.5,\ c_1 = c_2 = 2,\ r_1 = r_2 = 0.5$. Compute v^{new} and x^{new}.

[Q4] MCQ

Which algorithm uses *stigmergy* (indirect communication via the environment)?
(a) GA (b) ACO (c) PSO (d) None

Answers — Round 2

[A1]

Evaporation **forgets old / poor trails**, preventing unlimited pheromone build-up and **premature convergence** — it keeps the search exploring.

[A2]

$v_i \leftarrow wv_i + c_1 r_1 (p_i - x_i) + c_2 r_2 (g - x_i)$: **inertia** ($wv_i$), **cognitive** (toward personal best p_i), **social** (toward global best g).

[A3]

$v^{\text{new}} = 0.5(2) + 2(0.5)(3-1) + 2(0.5)(6-1) = 1 + 2 + 5 = \mathbf{8}$; $\quad x^{\text{new}} = 1 + 8 = \mathbf{9}$.

[A4]

(b) ACO — ants communicate indirectly by depositing and sensing pheromone in the environment.

Summary of Unit 4

What we covered

1. **Genetics & evolution** — variation, inheritance, selection $\rightarrow$ probabilistic search.
2. **GA framework** — initialise, evaluate, select, cross, mutate, repeat.
3. **Operators** — encoding, roulette/tournament selection, crossover, mutation; worked numerics on $f(x) = x^2$.
4. **Architectures** — generational, steady-state, elitist, island; schema idea.
5. **Single-objective optimisation** — knapsack / TSP case studies.
6. **Swarm intelligence** — decentralised, self-organising, stigmergy.
7. **ACO** — pheromone-guided transition rule; worked probability example.
8. **PSO** — inertia + cognitive + social velocity update; worked numeric.

Evolution and swarms give us robust, derivative-free global optimisers.

Practice Questions (Exam Oriented)

Short answer (2–5 marks)

1. Map five biological terms to their genetic-algorithm counterparts.
2. Explain exploration vs. exploitation and how GA operators provide each.
3. Write the roulette-wheel selection probability formula with an example.
4. State the ACO transition-probability formula and define each symbol.
5. Write the PSO velocity-update equation and label its three terms.

Long answer (8–10 marks) — include numerics

6. For $f(x) = x^2$ on 5-bit strings $\{13, 24, 8, 19\}$, compute selection probabilities and expected copies.
7. Perform single-point crossover and a bit-flip mutation on two given parents; report child fitnesses.
8. Explain the full GA cycle with a flow diagram and stopping criteria.
9. Given $\tau, \eta, \alpha, \beta$ for three edges, compute the ACO transition probabilities.
10. Given a PSO particle's state, compute the new velocity and position; interpret the result.

Think & Explore (Take-Home)

Numeric drill

For $f(x) = x^2$ on $x \in [0, 31]$:

- Start from a random 4-string population.
- Do one full generation by hand (selection, crossover, mutation).
- Report the best fitness before and after — did it improve?

Coding (self-learning)

Implement a GA with `DEAP`, ACO for a small TSP, and PSO with `pyswarms`; plot best-fitness vs. iteration for each.

Reading & reflection

- Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*.
- Dorigo & Stützle, *Ant Colony Optimization*.
- Kennedy & Eberhart, "Particle Swarm Optimization," 1995.
- Prepare for **Unit 5**: multi-objective optimisation and NSGA-II.

Reminder

Assessment-2 covers Units 3 & 4; Assignment 2 covers Units 3, 4 & 5 — handwritten, step-by-step, single PDF via the course page.

References

Textbooks (as per the course page)

1. D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley. **[Main text for Unit 4]**
2. J.-S. R. Jang, C.-T. Sun and E. Mizutani, *Neuro-Fuzzy and Soft Computing*, Prentice Hall.
3. G. J. Klir and B. Yuan, *Fuzzy Sets and Fuzzy Logic: Theory and Applications*, Prentice Hall.

Seminal papers

4. J. H. Holland, *Adaptation in Natural and Artificial Systems*, Univ. of Michigan Press, 1975.
5. M. Dorigo, V. Maniezzo, A. Colorni, "Ant System: optimization by a colony of cooperating agents," *IEEE Trans. SMC*, 1996.
6. J. Kennedy and R. Eberhart, "Particle Swarm Optimization," *Proc. IEEE ICNN*, 1995.

NPTEL: *Soft Computing*, IIT Kharagpur · Course page:
https://www.gcjana.in/courses/shardauniversity/2601/soft-computing/

Thank You

Any Questions?

"It is not the strongest of the species that survives,
but the one most responsive to change."
— in the spirit of Charles Darwin

Post your doubts on the course page →
https://www.gcjana.in/courses/shardauniversity/2601/soft-computing/#Post-doubts