04. Schema Theorem

Wednesday, September 17, 2025 4:46 PM

What is a Genetic Algorithm (GA)?

A GA is like **nature-inspired problem solving**.

- We have a **population** of solutions (like different creatures).
- Each solution is represented as a **string of bits** (like DNA). Example: 101100.
- Good solutions "survive" and "mate" (crossover), sometimes "mutate" (random changes), and over generations, the population evolves towards better solutions.

What is a Schema?

A **Schema** is just a **pattern** inside those bit-strings that we want to keep track of.

Example:
Suppose solutions are 6-bit strings like:

- 101100
- 111101
- 100101

A **schema** is written with 0, 1, and * (where * means "don't care").

- Schema 1*1*** means: any string that starts with 1, has 1 at 3rd position, and we don't care about other positions.
  - 101100 fits this schema.
  - 111101 also fits.
  - 100101 ✗ does not fit.

So schema = a template or building block.

Why Schema is Important?

Schemas represent **useful patterns** in solutions.

- Like in evolution, certain "genes" (patterns of DNA) are good and survive across generations.
- In GA, we want to know: *How many strings with this schema will survive and grow in the next generation?*

This leads us to the **Schema Theorem**.

What is the Schema Theorem?

The Schema Theorem tells us **how schemas grow or shrink from one generation to the next**.

It says: Short, low-order, above-average schemas increase exponentially in the population.

Short schema = small distance between fixed positions.

- Example: 1*0*** is short (fixed positions close together).
- Easier to survive crossover.

Low-order schema = schema with fewer fixed positions (like only 2 bits fixed, others are *).

- Example: 1***** (order = 1).
- Less chance of being broken by mutation.

Above-average schema = schema that belongs to solutions with fitness above average.

- If individuals following schema are doing well (good fitness), they spread more.

How Does it Work (with Example)?

Imagine population size = 6, strings of length 6.
Population:

101100 → fitness = 6
111101 → fitness = 5
100101 → fitness = 3
101101 → fitness = 4
110100 → fitness = 2
111100 → fitness = 7

Average fitness = (6+5+3+4+2+7) / 6 = 27/6 = 4.5

Now take schema 1*1***.

- Which strings match?
  - 101100 (fitness 6) ☑
  - 111101 (fitness 5) ☑
  - 101101 (fitness 4) ☑
  - 111100 (fitness 7) ☑

So, 4 individuals match. Their **average fitness** = (6+5+4+7)/4 = 22/4 = 5.5

Schema fitness = 5.5, which is **greater than population average (4.5)**. So, this schema is **above-average**.

What happens next?

- **Selection**: Since schema's average fitness is higher, more offspring from these strings will be selected → schema grows.
- **Crossover**: If crossover point doesn't cut between important bits, schema survives.
- **Mutation**: If mutation doesn't flip the fixed bits, schema survives.

Thus, schema theorem says:

Good building blocks (short, low-order, above-average schemas) spread exponentially over generations.

Derivation of Schema Theorem

Defining Length δ(H):

The distance (number of positions) between the **first fixed position** and the **last fixed position** in a schema.

- Fixed position = where schema specifies 0 or 1 (not a *).
- Example: schema 1***1
  - First fixed bit at position 1
  - Last fixed bit at position 5
  - δ(H) = 5 − 1 = 4

Examples from slide

- 1****
  - Only fixed at position 1.
  - δ(H) = 0 (because first and last fixed positions are the same).
- 11***
  - Fixed at positions 1 and 2.
  - δ(H) = 2 − 1 = 1
- 1***1
  - Fixed at positions 1 and 5.
  - δ(H) = 5 − 1 = 4

Why important?

Crossover can break schemas. The **longer δ(H)**, the more likely crossover cuts inside and destroys the pattern. So **shorter schemas are safer**.

Order o(H):

The number of fixed positions (0s or 1s) in the schema.

Examples

- 1**** → only one fixed bit → o(H) = 1
- 0***** → o(H) = 1
- 11*** → two fixed bits → o(H) = 2
- 1***1 → two fixed bits → o(H) = 2

Why important?

Mutation can flip fixed bits.

- More fixed bits (higher order) → greater chance mutation destroys schema.
  So **low-order schemas are safer**.

- 11*** → two fixed bits → o(H) = 2
- 1***1 → two fixed bits → o(H) = 2

- "Our black box problem has 2^l possible solutions."
- But there are even more schemas than solutions!

Explanation

If chromosome length = l:

- Number of possible strings (solutions) = 2^l.
- But number of possible schemas = 3^l.
  - Because each position can be 0, 1, or *.

Example with length 3:

- Strings = $2^3 = 8$.
- Schemas = $3^3 = 27$.

Important idea

- Every individual string represents **many schemas at once**.
- Example: string 101 belongs to schemas 1*1, 10*, ***, 101, etc.
- So when we evolve strings, we're actually evolving **a huge collection of schemas simultaneously**.

- GA doesn't just evolve individuals.
- It evolves **good schemas** (building blocks).

Key idea

- **Goal:** Find good solutions (individuals).
- **How GA works:** By **promoting good schemas** (those with above-average fitness).
- **How to help GA:** Don't break them too much with crossover/mutation.

Example:
If schema 11*** produces high-fitness solutions, GA tries to increase the number of individuals matching it.

Survival of a Schema

- Schema $\boldsymbol{H}$ is a template over $\{\mathbf{0}, \mathbf{1}, *\}$.
- At time $\boldsymbol{t}$, we have $\boldsymbol{m(H,t)}$ copies in the population.

- Example: schema 00*.
- **Question: how many will survive to the next generation?**
  - That's $\boldsymbol{m(H,t+1)}$.

The central question: If we have **some strings matching a schema** at generation $\boldsymbol{t}$, how many will we expect in generation $\boldsymbol{t+1}$?

This is exactly what the Schema Theorem answers.

Remember:

- Selection Favors individuals with **higher fitness**.
- If schema H is present in high-fitness individuals, then more copies of H will appear in the next generation.

Example

Say schema = 1****.

- If most of the strings starting with 1 have high fitness, then 1**** gets more copies.
- If they're low fitness, then fewer copies survive.

So **selection is the main driver of schema growth**.

Selection $(f(i)/\bar{f})$

$\boldsymbol{f(i)}$ = fitness of individual, $\bar{\boldsymbol{f}}$ = faverage fitness of the population

- If an individual has **fitness above average**, it has a **higher chance of being selected**.
- If an individual has **fitness below average**, it has a **lower chance of being selected**.

So, **selection pressure** pushes schemas in good individuals forward.

Reproduction (Schema Growth Equation)

$$\boldsymbol{m(H,t+1) = m(H,t).\frac{f(H)}{\bar{f}}}$$

- $\boldsymbol{m(H,t)}$ = number of individuals matching schema $\boldsymbol{H}$ in current population.
- $\boldsymbol{f(H)}$ = average fitness of those individuals.
- $\bar{\boldsymbol{f}}$ = average fitness of the entire population.

- If schema $\boldsymbol{H}$ has **above-average fitness** $(\boldsymbol{f(H) > \bar{f}})$, then: $\frac{f(H)}{\bar{f}} > 1$ → schema grows (more copies next gen).
- If schema $\boldsymbol{H}$ has **below-average fitness**, then: $\frac{f(H)}{\bar{f}} < 1$ → schema shrinks (fewer copies next gen).

Example:

- Schema H average fitness = 8
- Population average = 5
- Ratio = 8/5 = **1.6 > 1**
- This means schema H produces **1.6 times as many offspring** (in expectation) → it **grows**.

Example:

- Schema H average fitness = 3
- Population average = 5
- Ratio = 3/5 = **0.6 < 1**
- Means schema H gets only 60% as many offspring as before → it **declines**.

Example

Population size = 6.

Schema = `1****`.

- Suppose at generation t: 3 individuals match it → $m(H,t) = 3$.
- Average fitness of those 3 = $f(H) = 8$.
- Average population fitness = $\bar{f} = 5$.

Expected number next generation:

$$m(H,t+1) = 3 \cdot \frac{8}{5} = 4.8$$

On average, schema 1**** grows from 3 to ~5 individuals!

we have **selection & reproduction only**. But we're ignoring the fact that **crossover and mutation can destroy schemas**.

let's see how **crossover** affects schemas.

Crossover

- Example string $A = 1101000$.
- Two example schemas:
  - $H_1 = *1*****0$ (Order = 2, Defining length = 5)
  - $H_2 = ***10**$ (Order = 2, Defining length = 1)

- When two parents crossover, a **cut point** is chosen (like cutting DNA).
- If the cut falls **between fixed bits** of a schema, the schema gets **destroyed**.

The longer the **defining length δ(H)**, the more places crossover can break it.

- H_1: δ = 5 → large gap → more risk of being broken.
- H_2: δ = 1 → short gap → safer.

Crossover Survival Probability

Crossover Survival Probability

$$p_s \geq 1 - p_c \cdot \frac{\delta(H)}{l-1}$$

Where:

- p_s = probability schema **survives** crossover.
- p_c = probability crossover happens.
- $\delta(H)$ = defining length.
- l = chromosome length.

- If crossover doesn't happen → schema survives.
- If crossover happens but cut is **outside defining length** → schema survives.
- Only if cut falls **inside defining length** → schema breaks.

So **shorter schemas** have higher survival chance.

Example

Chromosome length $l = 6$.
Crossover probability $p_c = 0.6$.

Schema $H = 1 * * * 1$.

- δ(H) = 4 (positions 1 and 5 fixed).
- Survival probability:

$$p_s \geq 1 - 0.6 \cdot \frac{4}{5} = 1 - 0.48 = 0.52$$

Only ~52% chance to survive crossover each time!

Now compare with short schema `11***`:

- δ(H) = 1

$$p_s \geq 1 - 0.6 \cdot \frac{1}{5} = 0.88$$

Much safer — 88% chance of survival.

- **Short schemas** survive crossover better.
- That's why Schema Theorem Favors **short, low-order schemas**.

Reproduction and Crossover

$$m(H, t+1) \geq m(H,t).\frac{f(H)}{\bar{f}} .[1 - p_c \cdot \frac{\delta(H)}{l-1}]$$

- $m(H,t)$: current number of schema copies
- $\frac{f(H)}{\bar{f}}$: selection effect, Probability of being chosen
- $[1 - p_c \cdot \frac{\delta(H)}{l-1}]$: crossover survival factor

This means **after reproduction + crossover**, the expected number of schema copies depends on:

1. How good schema's average fitness is compared to population.
2. How likely it is to survive crossover (shorter δ = safer).

So far, no mutation yet.

Mutation

What mutation does

- In a Genetic Algorithm, **mutation** means flipping a bit:
  - 0 → 1
  - 1 → 0
- Each bit flips with probability p_m.
- Example: if $p_m = 0.01$, each bit has a 1% chance of flipping.

Schema has fixed bits

A schema has some **fixed positions** (0 or 1), and some "don't care" (∗*).

- Only fixed positions matter for schema survival.
- If a fixed bit gets flipped, the schema is destroyed.

Probability of survival for one bit

- Probability that **a single bit is NOT flipped** = $1 - p_m$.
- Example: $p_m = 0.01$, then survival = 0.99 (99% safe).

Probability of survival for multiple bits

- Schema has $o(H)$ fixed bits.
- For schema to survive, **all of them must survive mutation**.
- Since mutation acts independently on each bit, multiply probabilities.

$$P(\text{schema survives}) = (1 - p_m)^{o(H)}$$

One bit

Schema with **1 fixed bit** (e.g., `1****`).

- Probability this bit does not mutate = $1 - p_m$.

So survival probability = $(1 - p_m)^1$.

Two bits

Schema with **2 fixed bits** (e.g., `1***0`).

- First bit survives = $1 - p_m$.
- Second bit survives = $1 - p_m$.

Multiply (because both must survive, independent events):

$$(1 - p_m) \times (1 - p_m) = (1 - p_m)^2$$

Three bits

Schema with **3 fixed bits** (e.g., `101**`).

- Bit 1 survives = $1 - p_m$.
- Bit 2 survives = $1 - p_m$.
- Bit 3 survives = $1 - p_m$.

Multiply:

$$(1 - p_m) \times (1 - p_m) \times (1 - p_m) = (1 - p_m)^3$$

Approximation for small p_m

If p_m is very small (like 0.01), we can use approximation:

$$(1 - p_m)^{o(H)} \approx 1 - o(H) \cdot p_m$$

Why?
This comes from the **binomial expansion** of $(1 - p_m)^{o(H)}$.
For small p_m, higher powers of p_m (like p_m^2) are tiny, so we ignore them.

Example:

- $o(H) = 2, p_m = 0.01$
- Exact: $(0.99)^2 = 0.9801$
- Approx: $1 - 2 \cdot 0.01 = 0.98$

Pretty close!

Because of the **binomial expansion**:

$$(1 - p_m)^{o(H)} = 1 - o(H)p_m + (\text{tiny terms involving } p_m^2, p_m^3, \dots)$$

$$1 - o(H).p_m$$

Example

Schema = `1***1`

$1 - o(H).p_m$

Example

Schema = `1***1`

- Order = 2 (two fixed bits).
- Mutation probability $p_m = 0.01$.

Survival probability:

$$(1 - 0.01)^2 = 0.99^2 = 0.9801 \approx 98\%$$

Low-order schema survives mutation better.

If order = 5:

$$(1 - 0.01)^5 = 0.951 \approx 95\%$$

Higher order schema more likely destroyed.

- Bit 2 survives = $1 - p_m$.
- Bit 3 survives = $1 - p_m$.

Multiply:

$$(1 - p_m) \times (1 - p_m) \times (1 - p_m) = (1 - p_m)^3$$

General case (o(H) bits)

If a schema has $o(H)$ fixed bits, and each one independently survives with probability $1 - p_m$, then:

$$\text{Survival probability} = (1 - p_m)^{o(H)}$$

- **Selection** pushes schemas up if above average.
- **Crossover** destroys schemas with probability depending on δ(H).
- **Mutation** destroys schemas with probability depending on o(H).

Put them together = Schema Theorem.

The Schema Theorem

$$m(H, t+1) \geq m(H, t).\frac{f(H)}{\bar{f}} .[1 - p_c .\frac{\delta(H)}{l-1} - o(H).p_m]$$

- $m(H, t)$: current number of schema copies
- $\frac{f(H)}{\bar{f}}$: selection effect, Probability of being chosen
- $\frac{\delta(H)}{l-1}$: crossover survival factor, Probability of crossover destroying schema
- $o(H).p_m$: Probability of mutation destroying schema

Schema survival is like a battle between **growth by selection** and **destruction by crossover & mutation**.

- If a schema is **short, low-order, and above-average fitness**, it will **grow exponentially** across generations.
- If it's long or has too many fixed bits, it will likely be destroyed.

Example

Suppose:

- Chromosome length $l = 6$
- Schema $H = 1 * * * 1$
  - Order $o(H) = 2$
  - Defining length $\delta(H) = 4$
- Population: $m(H, t) = 3$ copies right now
- Average fitness of schema: $f(H) = 8$
- Population average fitness: $\bar{f} = 5$
- $p_c = 0.6, p_m = 0.01$

Now:

$$m(H, t+1) \geq 3 \cdot \frac{8}{5} \cdot \left[1 - 0.6 \cdot \frac{4}{5} - 2 \cdot 0.01\right]$$

Step by step:

- $3 \cdot \frac{8}{5} = 4.8$
- Inside bracket: $1 - (0.6 \cdot 0.8) - 0.02 = 1 - 0.48 - 0.02 = 0.50$
- Multiply: $4.8 \times 0.50 = 2.4$

Expected number of schema copies next generation ≥ 2.4.

So even though schema had above-average fitness, its **long defining length** and fixed bits make it fragile → survival just about steady, not explosive growth.

The Schema Theorem is not just math — it gives **guidance**:

1. **Selection operators**
   - Should reward above-average schemas → that's why we use proportional selection, tournament selection, etc.
2. **Crossover operators**
   - If crossover often breaks good schemas, GA struggles.
   - That's why some crossovers (1-point, 2-point, uniform) are chosen carefully.
3. **Mutation operators**
   - Mutation should add diversity but not destroy too many good schemas.
   - That's why mutation rates are usually small (like 1%).

Main lesson: **GA works by protecting and multiplying useful patterns (schemas)**.

The Schema Theorem is a **theoretical guideline**, not a guarantee.

1. **Exponential growth claim (wrong)**
   - In practice, good schemas can grow, but not always exponentially, because crossover/mutation break them, and randomness affects small populations.
2. **Guaranteed convergence claim (wrong)**
   - GA may converge, but not always to the optimal solution. Sometimes it gets stuck in local optima.
3. **No Free Lunch**
   - For some problems, GA works great.
   - For others, it doesn't.
   - There's no universal guarantee.