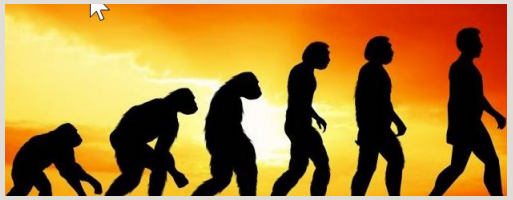
AI LAB 2025

AN INTRODUCTION TO ARTIFICIAL INTELLIGENCE

SHAY BUSHINSKY, SPRING 2025

LAB2: GENETIC PROCESSES

ADVANCED EVOLUTIONARY INSPIRED ALGORITHMS



IN THIS LECTURE

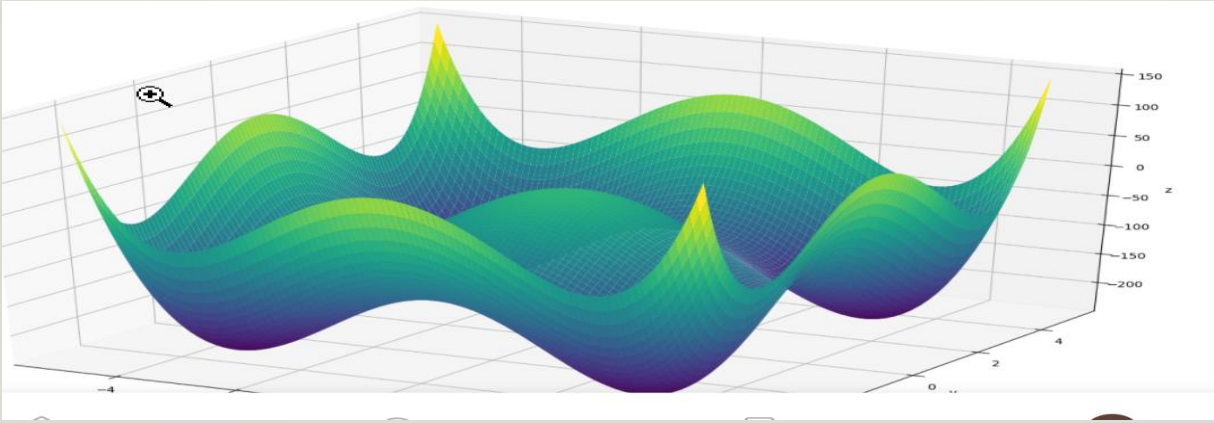
- Similarity Metrics
- Mutation Control
- Extending GA
  - Niching
  - Crowding
  - Speciation (Threshold & Clustering)
  - The Island Model, Random Immigrants
  - Exaptation

OPTIMIZATION ASPIRATION LIST…

1. Optimality (find Global Optima)
2. Completeness (find all solutions)
3. Fast Convergence
4. Reasonable Running Time

THE UNCONSTRUCTIVE AND DESTRUCTIVE CROSSOVER OPERATOR

- Diversity is crucial because crossing over a homogeneous population **does not produce new solutions.**
- Conversely, crossing over between good solutions **might be harmful** due to selection pressure when multiple peaks exist.



FIGHTING LOCAL OPTIMA

EVOLUTION CONTROL

LOCAL OPTIMA SIGNALS

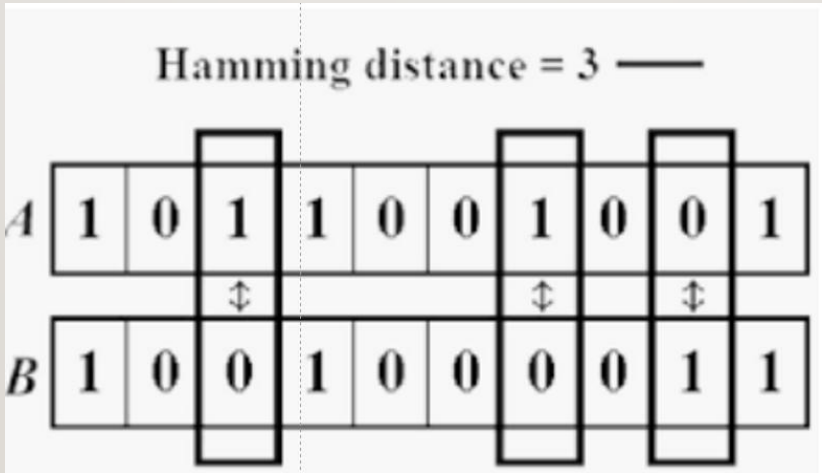
- Observing:
  1. Gene similarity
  2. Generation fitness average and standard deviation

MEASURING SIMILARITY

Quantifying Genetic Distance to Guide Diversity and Selection

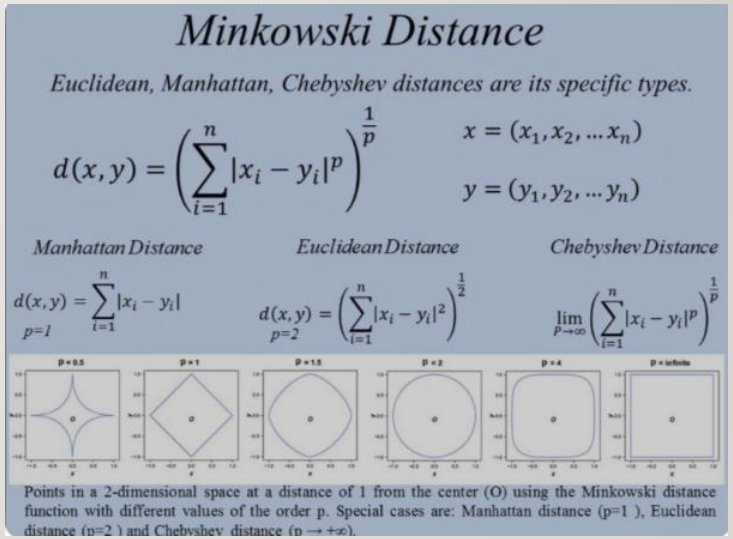
BINARY INSTANCES DISTANCE

- The similarity between two individuals is defined by the distance between them – denoted dij.
- E.g., the similarity between two binary strings can be defined by their **Hamming distance**.



DISTANCE BETWEEN NUMERIC INSTANCES

- For Numeric Representations:
- L1 – Manhattan (individual feature value matter)
- L2 – Euclidian (all features are equivalent)
- Lp – Minkowski p=∞ is Chebyshev Distance



EDIT DISTANCE - STRING SIMILARITY

- Measure how dissimilar two strings (e.g., words)
- Counts the minimum number of operations required to transform one string into the other.

- Typical Allowed Operations:
  1. **Insertion**: Adding a character.
  2. **Deletion**: Removing a character.
  3. **Substitution**: Replacing one character with another.

EDIT DISTANCE- EXAMPLE

```
>>> s = "Mannhaton"
>>> s = s[:2] + s[3:]          # deletion
>>> s
'Manhaton'
>>> s = s[:5] + "t" + s[5:]    # insertion
>>> s
'Manhatton'
>>> s = s[:7] + "a" + s[8:]    # substitution
>>> s
'Manhattan'
```

EDIT-DISTANCE APPLICATIONS

1. **Spell Checking**: Finding the closest word to a misspelled word.
2. **DNA Sequencing**: Comparing genetic sequences.
3. **Natural Language Processing**: Measuring text similarity.

TYPES OF EDIT DISTANCE

1. The Levenshtein Distance:

- Only allows insertion, deletion, and substitution.
- **Example**: The edit distance between "kitten" and "sitting" is 3 (substitute 'k' with 's', 'e' with 'i', and add 'g').

2. The Damerau-Levenshtein Distance:

- Adds the operation of **transposition** (swapping two adjacent characters).
- **Example**: The edit distance between "ca" and "abc" is 2 (insert 'b', transpose 'a' and 'c').

DIFFERENT MEASURE OF DISTANCE BETWEEN PERMUTATIONS

Distance Measure	Use Case	Example Application
Number of Swaps	Adjacent swap transformations	Sorting algorithms (e.g., bubble sort)
Kendall Tau Distance	Pairwise disagreements in order	Ranking systems (e.g., race order comparisons)
Spearman's Rho	Rank correlation (monotonic relationships)	Statistical rank correlation analysis
Hamming Distance	Positional differences	Error detection in coding theory
Cayley Distance	General transpositions in permutation groups	Group theory in algebra
Ulam Distance	Subsequence similarity	Genetic sequence comparison in bioinformatics
Damerau-Levenshtein Distance	Edit operations (insertions, deletions, substitutions, transpositions)	Text correction and string matching
Footrule Distance	Total displacement in rankings	Comparing linear rankings

KENDALL TAU DISTANCE BETWEEN PERMUTATIONS EXAMPLE

- D(0, 3, 1, 6, 2, 5, 4 and 1, 0, 3, 6, 4, 2, 5) = 4

  as the pairs: 0-1, 3-1, 2-4, 5-4 are in different order in the two permutations

  - all other pairs are in the same order

- To calculate the Kendall Tau distance, compare each pair of values from list 1 with the corresponding pair in list 2 and count the number of pairs that are in the opposite order.

BIN PACKING SIMILARITY (HUNGARIAN ALGORITHM)

- **Set Difference:**
  Measure bin similarity via symmetric difference.
- **Cost Matrix:**
  Build matrix of item differences between bins.
- **Hungarian Match:**
  Find optimal matching with minimal total cost.
- **Normalization:**
  Divide total cost by number of items.
- **Python Use:**
  scipy.optimize.linear_sum_assignment.

COMBATTING EARLY CONVERGENCE

1. **Mutation control**
2. *Selection techniques (covered)*
3. Punish Similarity
4. Diversify
5. Multi objective

TECHNIQUE 1: MUTATION CONTROL

MUTATION LEVELS

- **Population-Wide Mutation Control**:
  - Focuses on the entire population
  - Using adaptive mutation rates and diversity measures to maintain variability and avoid local minima.

- **Individual-Specific Mutation Control**:
  - Targets specific individuals
  - Adjusting mutation rates based on fitness, age, or specific traits to enhance exploration and prevent premature convergence.

POPULATION BASED MUTATION CONTROL

ADAPTING MUTATION RATES GLOBALLY TO BALANCE EXPLORATION AND CONVERGENCE

MUTATION CONTROL METHODS

1. **The Basic Mutation Operator Mode**:
   - Easy implementation but hard to control resultד

2. **The Non-Uniform Mutation Mode**:
   - Mutation probability is greater early in the evolution, with the evolution advancing, mutation probability is appropriately reduced

GENERATIONAL FITNESS ADAPTIVE CONTROL

- Decrease the mutation rate if generational fitness is too high.
- Increase the mutation rate if generational fitness is too low.

- Dynamic adjustments to mutation rate
- Can be achieved through **linear or non-linear** functions

ADAPTIVE LINEAR DECREASE FUNCTIONS

1. **Decrease linearly** from **max** to **min** over generations

$$f(t) = p_{\max} - \left(\frac{p_{\max} - p_{\min}}{T} \right) \cdot t$$

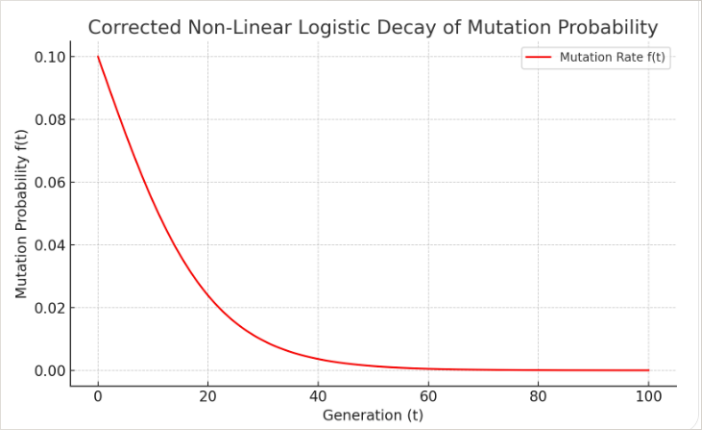
Where:

- f(t): value (e.g., mutation probability) at generation t
- pmax: initial (maximum) value
- pmin: final (minimum) value
- T: total number of generations
- t: current generation ($0 \leq t \leq T$)
- This formula decreases the value **linearly** from pmax to pmin as t increases.

ADAPTIVE NON-LINEAR DECREASE FUNCTIONS

2. **Non-Linear logistic decay (**r=rate of decay**)**

$$f(t) = \frac{2p_{\max}^2 e^{-rt}}{p_{\max} + p_{\max} e^{-rt}} = \frac{2p_{\max} e^{-rt}}{1 + e^{-rt}}$$



- Where:
- **f(t) mutation probability at generation t**
- **Pmax** max mutation probability
- **r** rate of decay
- f(t) decreases over time due to e^{rt}

- If f(t) falls bellow some minimum it is fixed to the minimum (lower bound)

MONITOR GENERATIONAL FITNESS

1. Average Fitness
2. Best Fitness
3. Fitness STD
4. Fitness Improvement

TRIGGERED HYPER MUTATION (THM)

- Increase the probability of mutation when the solution quality drops
  - Purpose: encourage diversity

CONDITIONS FOR TRIGGERING HYPERMUTATIONS

- Define criteria for triggering hyper mutation:
  - Set a **threshold** for **detecting stagnation**, such as

    1. **A minimum improvement** in the **best fitness** value **or**
    2. The **average fitness** value over a certain number of generations

TRIGGERED HYPER MUTATION FITNESS MONITORING

- Keep track of the best or average fitness value over the specified number of generations

- Check for stagnation:
  - If the improvement in fitness over the specified number of generations is below the defined threshold, trigger the hyper mutation

THM HYPER MUTATION SCOPE

- When hyper mutation is triggered, temporarily increase the mutation probability to a higher value:

1. For a predefined **number of generations** or
2. Until a significant improvement in the solution **quality** is observed

MUTATION RATE RESTORATION

- Restore the mutation probability to its original value
  - After the hyper mutation period is over, **or**
  - When a significant improvement in solution quality is achieved

INDIVIDUAL BASED MUTATION CONTROL

Tailoring Mutation Rates per Individual to Enhance Adaptive Search

FITNESS-BASED ADAPTIVE MUTATION CONTROL

- **High fitness individuals** have a **lower mutation** probability, while low fitness individuals have a higher mutation probability.

- This method can effectively **protect** the excellent individuals, but it can easily get trapped in local optima

AGE-BASED MUTATION CONTROL

- Implementing mutation strategies based on the "age" of individuals
  - age = number of generations they have survived.

- Older individuals might receive higher mutation rates to avoid premature convergence, while younger individuals might receive lower rates.

DEFINING RELATIVE FITNESS

- A measure of an individual's reproductive success compared to others in the population
- Expressed as a **ratio** of the individual's fitness to the average fitness in the population.
- **Normalization**: Ensures the fitness values are scaled appropriately for further calculations.

CALCULATING RELATIVE FITNESS

- Compute the relative fitness of individuals in a population and the selection pressure

1. **Compute <f>**: the population mean fitness
2. **Compute Rfi = fi / <f>:** the relative fitness
3. **Compute the normalizing constant**: SF=$\sum Rfi$
4. **Normalize relative fit**: Rfi-norm = Rfi / SF
5. **Measure the Selection Pressure:**

- s^2: sample variance
- Rf_i^{norm}: normalized fitness of the i-th sample
- $\langle Rf^{\text{norm}} \rangle$: mean of all normalized fitness values
- n: number of samples

Measure the Selection Pressure:

$$s^2 = \frac{1}{n-1} \sum_{i=1}^{n} (Rf_i^{\text{norm}} - \langle Rf^{\text{norm}} \rangle)^2$$

SELF-ADAPTIVE MUTATION CONTROL

- Change mutation probability based on the **relative fitness** of individuals in the population

$$p_{\mathrm{mut},i} = p_{\max} \cdot (1 - R f_i^{\mathrm{norm}})$$

  → Mutation probability **decreases as fitness increases**
  → Encourages **exploration for weak individuals**, **stability for strong ones**

- pmax: max mutation rate (e.g., 0.1–0.3)
- Optionally ensuring a lower bound:

$$p_{\mathrm{mut},i} = \max(p_{\min},\ p_{\max}(1 - R f_i^{\mathrm{norm}}))$$

- Depending on the relative fitness from the environment

SELF-ADAPTING MUTATION ALGORITHM

1. Evaluate the fitness of individuals in the population:

   Calculate an individual's **relative fitness** measure by:

   - Normalizing the fitness values OR
   - Ranking the individuals based on their fitness

2. Update the mutation probability for **every individual** based on their **relative fitness**

SELF-ADAPTIVE FITNESS

introduces **adaptive pressure** to **shift priorities over time** — allowing exploration, preserving diversity, and steering the search intelligently.

SELF-ADAPTIVE FITNESS FUNCTIONS

- Dynamically adjust selection pressure
- Promote diversity by shifting evaluation criteria
- Encourage exploration in stagnant populations
- Balance exploitation and exploration
- Adapt in real time to evolving needs

Self-Adaptive Fitness Function (Formula)

- Let $F(x,t)$ be the fitness of candidate x at time t
- $F(x,t) = \alpha(t) \cdot f(x) + [1 - \alpha(t)] \cdot g(x,t)$
- $f(x)$: static evaluation (traditional fitness)
- $g(x,t)$: adaptive term (e.g., diversity, novelty)
- $\alpha(t)$: dynamic weighting factor that evolves with t

WHAT CAN g(x,t) REPRESENT?

g(x,t)	Meaning	Use Case
Diversity Score	Inverse of similarity to others	Promote spread across search space
Novelty	How different an in individual's behavior is	Exploration in deceptive landscapes
Age-based reward	Based on how many generations it survived	Preserves long-surviving individuals
Entropy-based	From solution distribution entropy	Favors exploration when population is clustered
Sparsity metric	From feature/phenotype sparsity	Rewards underrepresented individuals
Rank-based scaling	Fitness rank rather than absolute value	Reduces domination by extreme elites

THE NOVELTY MEASURE

- Computes average distance to k-nearest neighbors
- Higher score indicates more unique solutions
- Encourages exploration of uncharted regions
- Mitigates clustering around local optima
- Enhances population diversity

NOVELTY-BASED REWARD

- **Motivation**:
  - Drives search toward **uncharted regions** of the solution space.
  - Avoids over-exploiting areas with many similar individuals.

$$g(x,t) = \frac{1}{k}\sum_{j=1}^{k} \mathrm{dist}(x, \mathrm{neighbor}_j)$$

- **Benefits**:
  - Encourages **solution diversity**
  - Avoids **local optima** by rewarding exploration
  - Useful in **deceptive or sparse fitness landscapes**
- **Best used when**: traditional fitness rewards similar solutions too strongly or lacks gradient

AGE-BASED REWARD

- **g(x,t) = age(x)**
- **Motivation**:
  - Older individuals are assumed to be **robust** or **resilient**.
  - They survived multiple selection phases, even if they aren't the absolute fittest.
- **Benefits**:
  - Protects **structurally useful but underperforming** individuals.
  - Helps preserve **diverse lineages**.
  - Prevents premature loss of **genetic material** that might become useful later.

ENTROPY-BASED REWARD

- **g(x,t)=H(Population)**
- **Motivation**:
- Low entropy = **population is clustered** (risk of convergence)
- High entropy = **diverse solutions** (healthy search)

Benefits:

- Encourages **exploration when the population is too similar**
- Dynamically boosts **diversity** under stagnation
- Complements novelty and age-based control for robust evolution

SPARSITY-BASED REWARD

- Measure **local density** (e.g., inverse of k-nearest neighbor density):

$$g(x,t) = \frac{1}{\text{density}(x)}$$

- **Benefits**:
  - Promotes **exploration of rarely visited regions**
  - Prevents **loss of niche behaviors** or rare feature combinations
  - Enhances **solution diversity** and avoids local convergence

RANK-BASED FITNESS SCALING

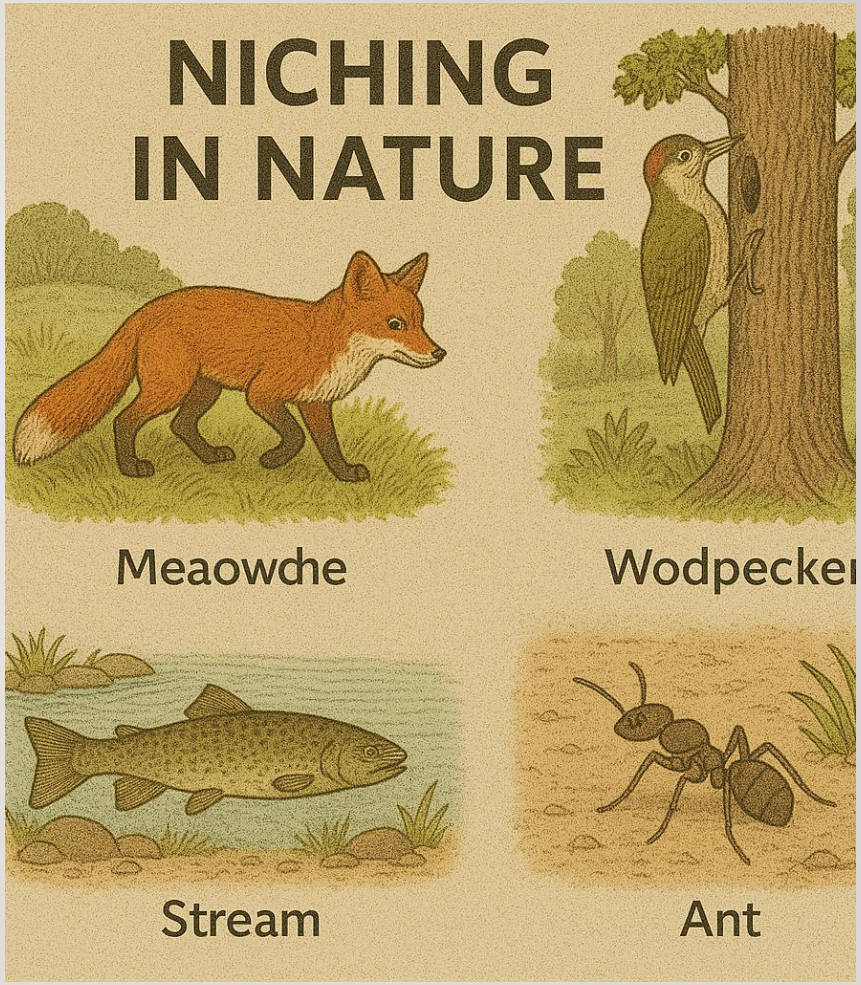
- **Definition**: Assigns fitness based on an individual's **rank** rather than raw fitness values.
- **Motivation**:
  - Prevents **extremely fit individuals** from dominating selection.
  - Maintains **selection pressure** without losing diversity.
- **How it's used**:
  - Rank individuals from best to worst (or vice versa)
  - Assign scaled fitness (e.g., linear, exponential, or sigmoid curve)
  - Use f′(x) in place of raw fitness in selection and adaptive models

$$f'(x) = \text{scale}(\text{rank}(x))$$

- **Benefits**:
  - Smooths out differences in performance
  - Preserves **weaker individuals** for exploration
  - Helps **maintain diversity** in early and late generations
- **Best used when**: fitness values vary wildly or contain sharp gradients

NICHING

Maintaining Diversity to Discover Multiple Optima



NICHING FOR MULTI SOLUTIONS

- **Niching** methods extend genetic algorithms to domains that require the location and maintenance of **multiple solutions:**

1. Classification and machine learning
2. Multimodal function optimization
3. **Multi-objective function optimization**
4. Simulation of complex and adaptive systems

NICHING & CROWDING

- **Penalize** individuals that are "close" to other individuals in the population
- Prevents the convergence of the whole population to just one of the peaks
- Two techniques are used:
  - 1. Fitness Sharing
  - 2. Crowding

THE SHARING METHOD

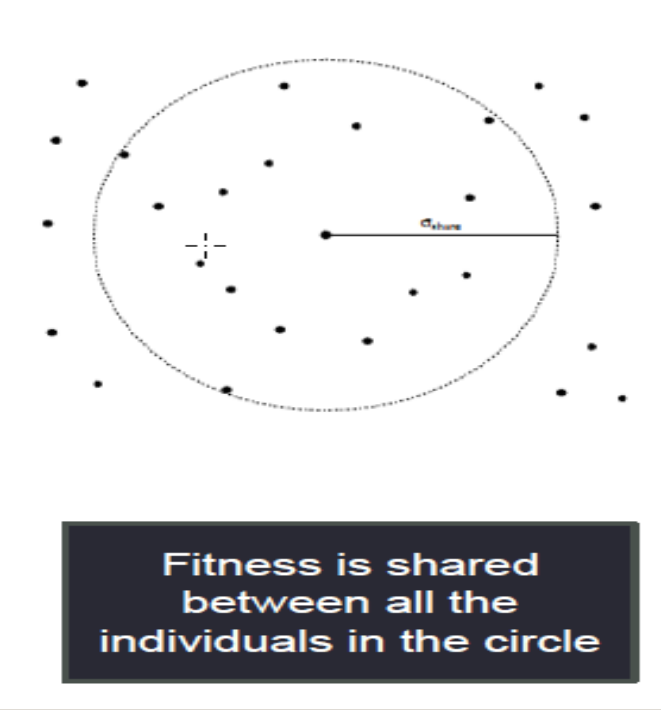
- Population is first divided into niches
- Shared fitness of any individual is computed only with respect to the individuals that are in its niche
- **Sharing Radius** defines the niche size
- Individuals within this radius will be regarded as **similar** and thus need to share fitness

FITNESS SHARING

- Transforms the **raw fitness** of an individual into the **shared one** (usually lower)
- **Motivation**: there is only a limited and fixed number of "resources" (i.e., fitness value) available at each niche
- Individuals occupying the same niche will have to share the resources

THE SIGMA-SHARING RADIUS PARAMETER

- Sharing radius, σ-share, defines the niche size



FITNESS SHARING PRINCIPLES

1. The more similar two genes are, the lower their individual fitness will become…

2. The more it shares, the more its fitness is scaled down

THE SHARING FUNCTION AND THE SHARED FITNESS FUNCTION

- Define a sharing function:

$$sh(d_{ij}) = \begin{cases} 1 - \left(\frac{d_{ij}}{\sigma_{share}}\right)^{\alpha}, & \text{if } d_{ij} < \sigma_{share}, \\ 0, & \text{otherwise}, \end{cases}$$

- Based on it, the shared fitness is defined as:

$$f_{share}(i) = \frac{f_{raw}(i)}{\sum_{j=1}^{\mu} sh(d_{ij})}$$

- **μ** denotes the population size
- **α** is a constant – usually equals 1
- **σ** niche radius is **fixed by the user** at some minimum distance between peaks

- The fitness of each member is scaled down based on its proximity to others in same niche

SHARING FUNCTION NOTES

- dij **is an entry in a similarity (or distance) matrix** for the entire population.
- It represents the **distance** (or dissimilarity) between individual i and individual j.

The full matrix D=[dij] is:

- **Symmetric** if the distance function is symmetric (e.g., Euclidean)
- **Square** with shape n×n, where n is the population size

1. If the **sh** function finds that dij is less than sigma-share, it returns a value in the range [0,1] that increases as dij decreases

FUNCTION ANALYSIS

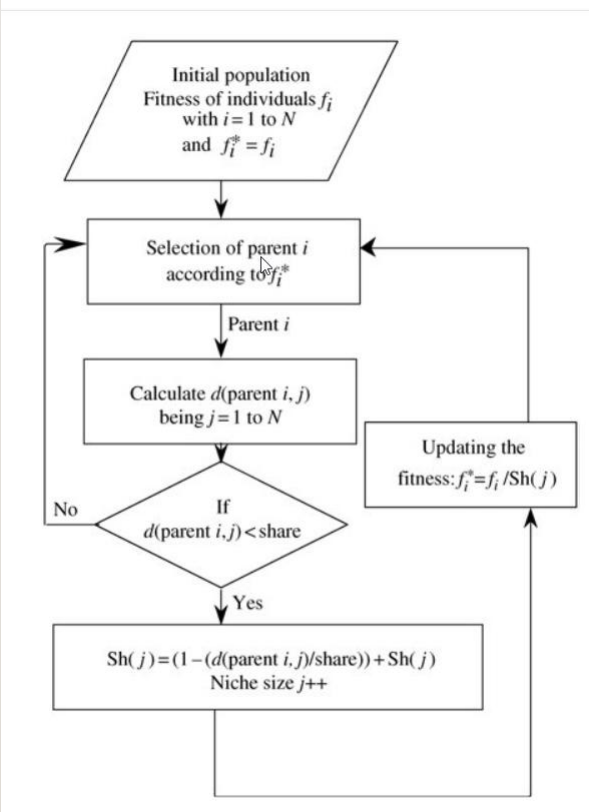
- sh(dij) returns a value in **[0,1]**.
- The **closer** two individuals are (smaller dij), the **larger** the sharing value.
- Specifically, When dij < σ-share:
  - dij→0 ⇒ sh(dij)→1
  - dij→σ-share ⇒ sh(dij)→0

So: **closer neighbors "share" more fitness**, reducing individual advantage when surrounded by similar solutions.

- **If dij ≥ σ-share**
  - sh(dij)=0→ they don't share any fitness
  - This means they are in **separate niches**

THE NICHE ALGORITHM FLOW

1. **Initialize Population**
   - Start with N individuals with fitness values f_i.
   - Set initial shared fitness $f_i^s = f_i$ for all.
2. **Select Parent i**
   - Choose parent i for reproduction based on **shared fitness** f_i^s, not raw fitness.
   - This helps reduce selection bias toward crowded regions.
3. **Distance Evaluation**
   - For each other individual j in the population, compute $d(\text{parent}_i, j)$, the **distance** between the parent and individual j.



THE NICHE ALGORITHM FLOW

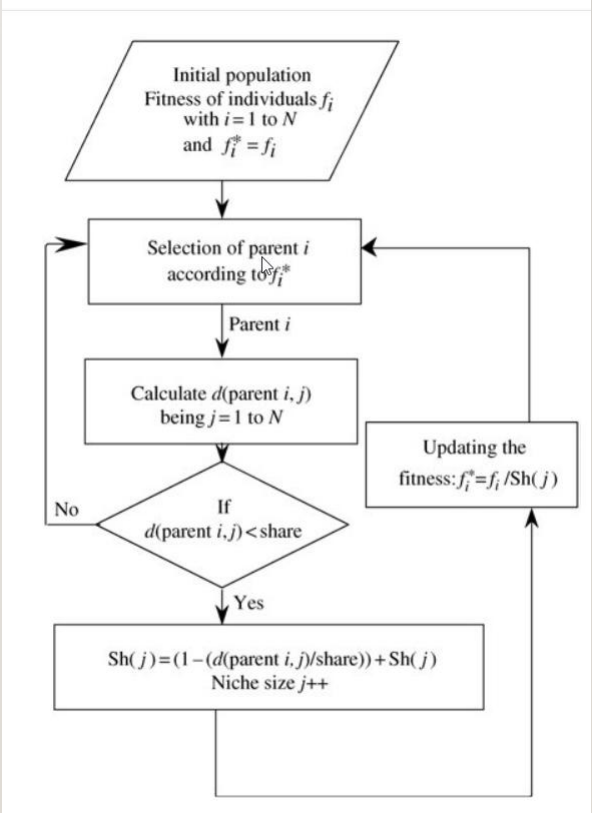
4. **Check Niche Radius**
   - If $d(\text{parent}_i, j) < \sigma_{\text{share}}$, they are considered in the same **niche**.
5. **Apply Sharing Function**
   - Compute contribution of individual j to niche count:

$$sh(j) \mathrel{+}= 1 - \left(\frac{d(\text{parent}_i, j)}{\sigma_{\text{share}}}\right)$$

   - Increment niche size counter.
6. **Update Shared Fitness**
   - Scale fitness of individual i using the total sharing from similar individuals:

$$f_i^s = \frac{f_i}{\sum_j sh(d_{ij})}$$

   - This reduces the fitness of individuals **surrounded by similar others**, discouraging overcrowding.
7. **Repeat for Next Individual**



THE SHARING EFFECT ON THE POPULATION

1. Discourage convergence to a single region of the fitness function:
   - The more individuals try to move in, the worse off they all are.

2. If the GA converges to a single local optimum somewhere, then the fitness of that optimum decreases
   - Due the increased competition within the niche.

IDEAL STEADY STATE AT LOCAL OPTIMA

- Eventually, another region of the fitness landscape becomes more attractive, and individuals **migrate** over there

- The idea is to reach a **steady state** -- a fixed point in the dynamics -- where an **appropriate representation of each niche is maintained**

NICHING EFFECT

- Reduce the effect of **genetic drift** resulting from the selection operator in the simple genetic algorithms

- They maintain population diversity and permit genetic algorithms to explore more search space to identify multiple peaks, whether optimal or otherwise

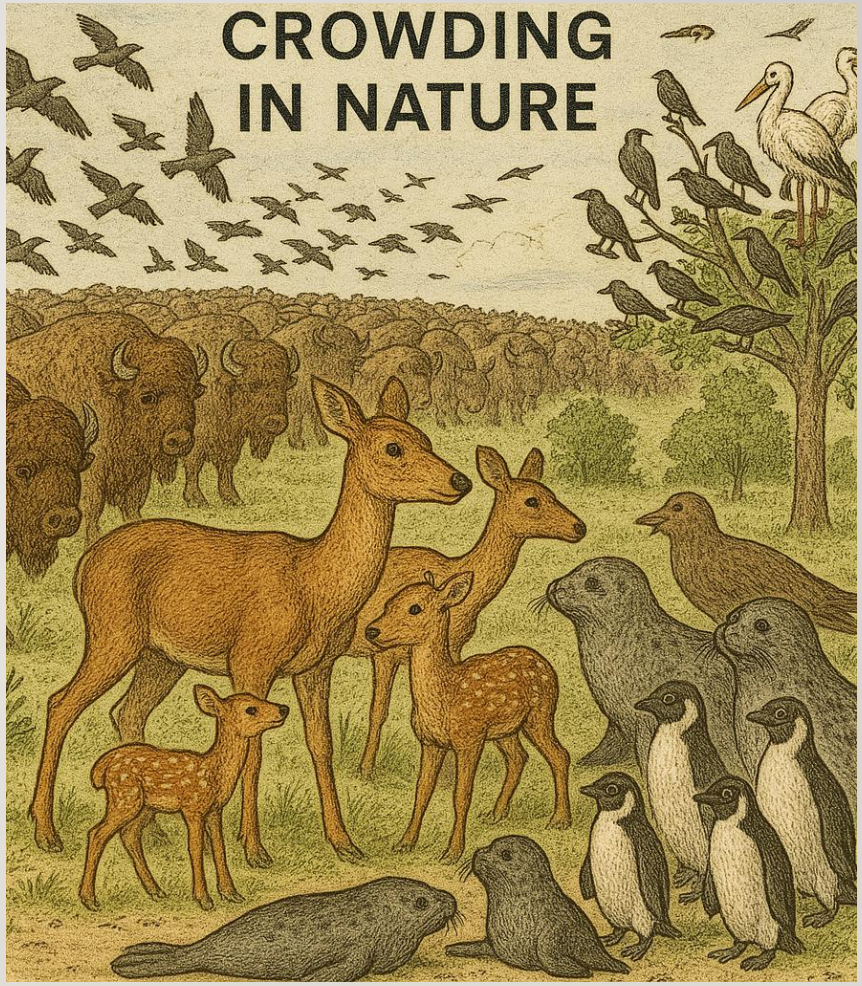
NICHING CRITIQUE

1. Sharing is hard because of the need to **manually set the niche radius,** and the algorithm is quite sensitive to this choice

2. It is also computation consuming

CROWDING

PRESERVING DIVERSITY BY LIMITING SIMILAR OFFSPRING REPLACEMENT



CROWDING IN NATURE

- Similar individuals in natural population, often of the same species, compete against each other for limited resources

- Dissimilar individuals tend to occupy different niches, they typically don't compete

CROWDING IN GA

- Crowding was first introduced by John De Jong
- A technique for preserving population diversity and preventing premature convergence

- **Fitness-Based Replacement**: Offspring replace parents based on improved fitness.

- In **Deterministic Crowding** not only the quality of potential solutions is important, but also their proximity to their parents

REPRODUCTION UNDER CROWDING

- New members of a species replace older members of that species, **not replacing members of other species**
- Crowding doesn't increase the diversity of the population, rather it strives to **maintain the pre-existing diversity**
- It's not directly influenced by fitness value

NON-DETERMINISTIC CROWDING

- **Selection**: Use traditional methods like tournament or roulette wheel selection.
- **Crossover**: Perform crossover operations to generate offspring.
- **Mutation**: Apply mutations to the offspring.
- **Replacement**: Compare offspring to parents or a subset of the population.
- Replace individuals based on calculated **probabilities of similarity and fitness**.

THE BOLTZMANN REPLACEMENT PROBABILITY

$$P(\text{replace P1 with O1}) = \frac{e^{-\Delta F/T}}{1 + e^{-\Delta F/T}} \quad \text{where } \Delta F = F_{\text{P1}} - F_{\text{O1}}$$

- $\Delta F_{\text{P1\_O1}}$: Fitness difference between parent P1 and offspring O1.
- T: Temperature parameter controlling randomness in the replacement process.

- The Role of Temperature (T):
  - If offspring O1 is **much better** than parent P1 → high replacement probability
  - If offspring is **worse**, it **might still replace** the parent if T is high (adds stochasticity)

THE KEY STOCHASTIC STEP

- **draw a random number** $r \in [0,1]$.
  If $r < P(\text{replace})$, the offspring replaces the parent.

- Provides **non-deterministic survival**.

Worse solutions may survive, allowing exploration.

- Controlled by T: higher T → more randomness (flatter sigmoid), lower T → sharper threshold.

NON-DETERMINISTIC CROWDING EXAMPLE

- Parent 1 (P1): Fitness = 50
- Parent 2 (P2): Fitness = 60

- Offspring 1 (O1): Fitness = 55 (generated by crossover and mutation between P1 and P2)
- Offspring 2 (O2): Fitness = 65 (generated by crossover and mutation between P1 and P2)

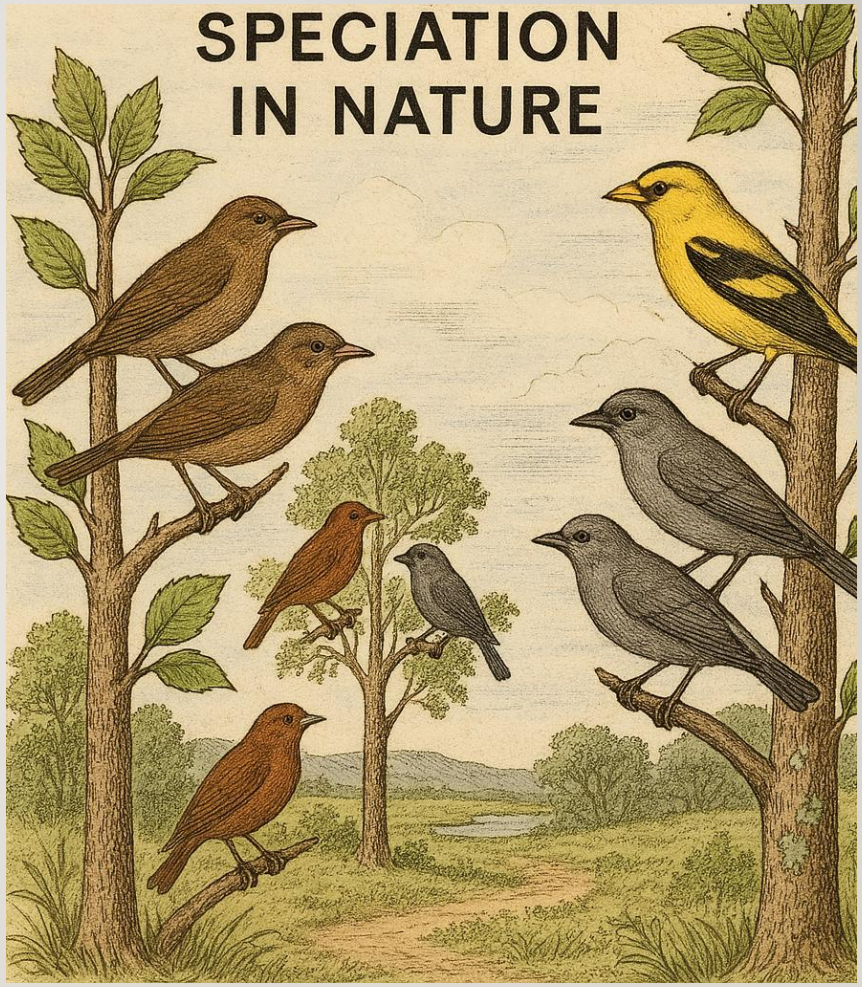
CALCULATE THE REPLACEMENT PROBABILITIES

- Calculate the replacement probability for each parent-offspring pair using the Boltzmann selection scheme:

- P(replace P1 with O1) = exp(-ΔF_P1_O1 / T) / (1 + exp(-ΔF_P1_O1 / T))
- P(replace P2 with O2) = exp(-ΔF_P2_O2 / T) / (1 + exp(-ΔF_P2_O2 / T))

SPECIATION

PROMOTING DIVERSITY BY GROUPING SIMILAR INDIVIDUALS INTO SPECIES



SPECIATION IN NATURE

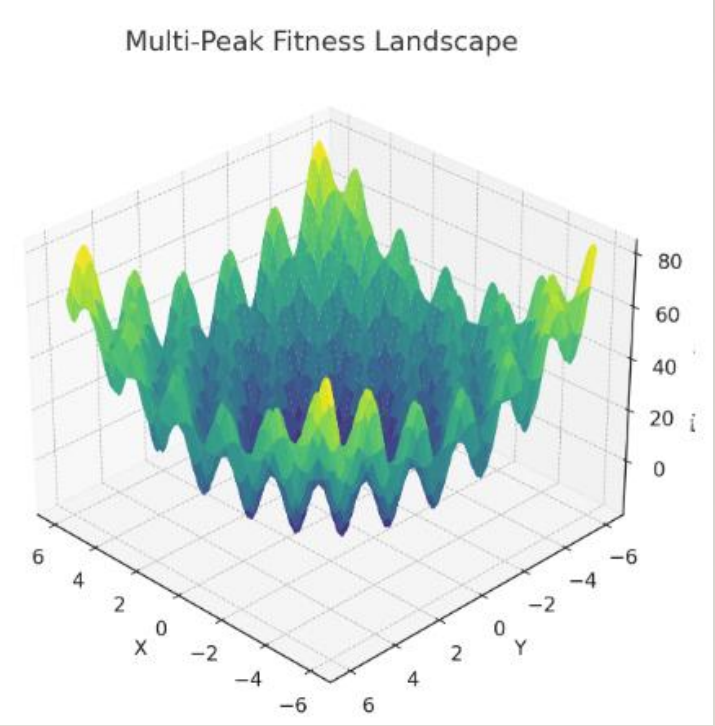
- The evolutionary process by which populations evolve to become distinct species.
- The formation of new and distinct species in the course of evolution.
- It occurs when a group within a species separates from other members of its species and develops its own unique characteristics.

SPECIATION IN GA

- Divides the population of candidates into a small number of species
- Each species will be a group of solutions that are allowed to crossover with each other

SPECIATION DYNAMICS

- Speciation is focused on converging to the actual peaks
- The *speciation* heuristic **penalizes crossover** between candidate solutions that are too similar:
  - If too similar, then crossover **may completely be forbidden**
- This encourages **population diversity** and helps prevent premature convergence to a less optimal solution



THRESHOLD SPECIATION

- Relies on similarity measurements between two genes

1. The **species count** is the desired number of species (common default=30)

2. The **species threshold** specifies the minimum similarity that genes must have to be the same species (adjusted during runtime)

SPECIES MEMBERSHIP

- Example:
- Genome 1: [2.0, 3.0, 5.0]
- Genome 2: [1.0, 2.0, 1.0]
- Euclidian Distance = 4.242641

- If the **Euclidean distance** is less than the speciation threshold => these two genomes would be considered as in the same species

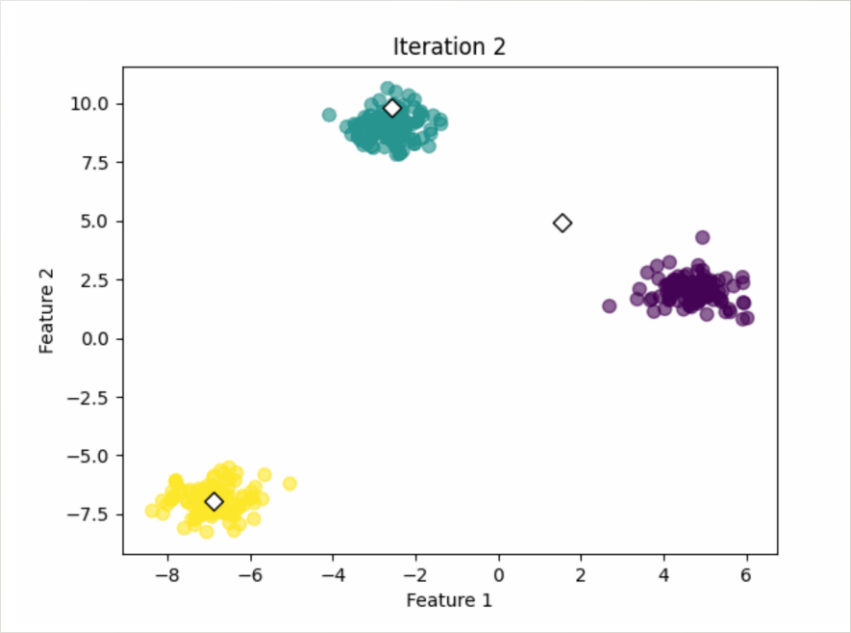
THRESHOLD CONTROL

- The threshold is **adjusted** to maintain the parameter species count at the end of generation:
  - If too many species => Increase threshold
  - If too few species => Decrease threshold

VARIANT: CLUSTERING SPECIATION

- Threshold speciation increases the effectiveness of crossover which can be a destructive operator

- Use **k-means** or **k-medoids** to cluster the population – **no need for the threshold parameter**



FINDING THE OPTIMAL K

- Start with 2 clusters
- Increase cluster count until max

- Choose optimal K by either one of the methods
  - The optimal Silhouette Score
  - The elbow method

THE SILHOUETTE COEFFICIENT

- Measure of how similar an object is to its own cluster (cohesion) compared to other clusters (separation)

- The silhouette ranges from −1 to +1
- A high value indicates that the object is well matched to its own cluster and poorly matched to neighboring clusters

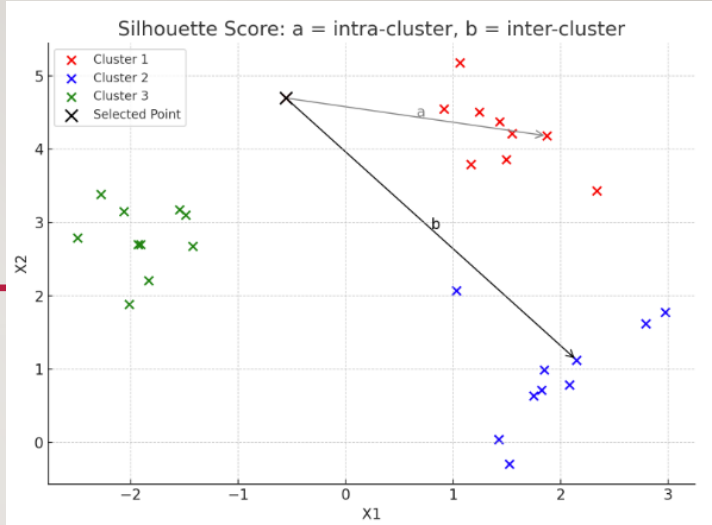
THE SILHOUETTE SCORE

- Silhouette Score = (b-a)/max(a,b)

- Where:
  - a = average intra-cluster distance i.e., the average distance between each point within a cluster
  - b = average inter-cluster distance i.e., the average distance between all clusters

INTERPRETATION

- **High score (close to 1)**: well-clustered, far from other clusters
- **Low score (close to 0)**: near the decision boundary between clusters
- **Negative score**: possibly misclassified



$$\text{Silhouette Score} = \frac{b - a}{\max(a, b)}$$

SILHOUETTE OPTIMAL K

- If most objects have a high value, then the clustering configuration is appropriate
- If many points have a low or negative value, then the clustering configuration may have too many or too few clusters
- The silhouette can be calculated with any distance metric, such as the Euclidean distance or the Manhattan distance

THE SILHOUETTE SCORE ALGORITHM

- For a data point i within cluster C:
- # **Calculate the average distance between i and all other points in C:**
- a(i) = (sum of distances between i and all other points in C) / (number of points in C - 1)
- # **Calculate the average distance between i and all points in the nearest neighboring cluster**:
- b(i) = min { (sum of distances between i and all points in cluster j) / (number of points in cluster j) } for all clusters j != C
- # **Calculate the silhouette score for i:**
- s(i) = (b(i) - a(i)) / max { a(i), b(i) }

SILHOUETTE ALGORITHM OVERALL SCORE

- # **The silhouette score for a cluster is then the average of all the silhouette scores for each data point within the cluster**:

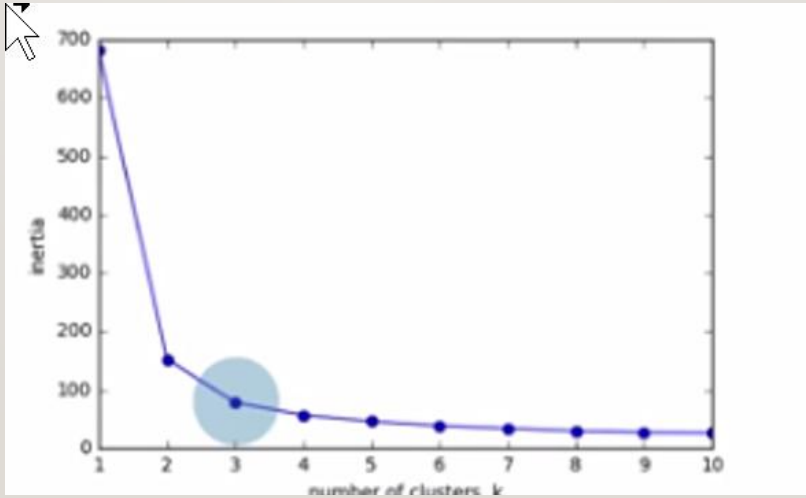
- s(C) = (sum of s(i) for all i in C) / (number of points in C)

- **#The overall silhouette score for the clustering solution is the average of the silhouette scores for all clusters**:

- s = (sum of s(C) for all clusters) / (number of clusters)

"THE ELBOW METHOD"

- K-means strives to minimizes the Inertia
- More clusters minimize the Inertia but overfits
- The sweet spot is the **minimum** number of clusters where the **Inertia flattens**:



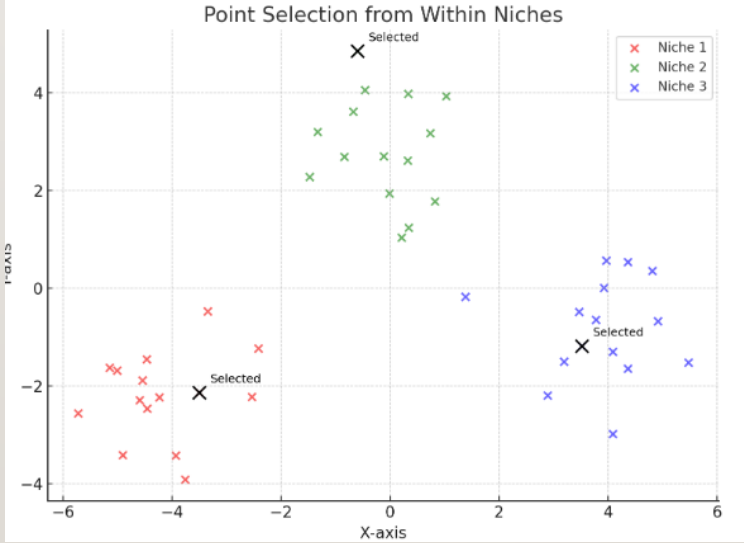
KEY NOTES

NICHES FORMATION

- In clustering speciation, like in niching and crowding, the niches evolve dynamically over time.
- In contrast, in threshold speciation, the niches tend to remain more or less constant.

PARENT SELECTION IN VARIANTS

- In both niching and crowding, methods such as RWS are applied to the population similar to a regular GA.
- However, in speciation, selection is done within species, preventing mixing between parents from different species.



THE ISLAND MODEL RANDOM IMMIGRANTS

THE ISLAND MODEL IN POPULATION GENETICS

- **Definition**: Populations divided into subpopulations (islands) with occasional migration.
- **Purpose**: Understand effects of gene flow, genetic drift, and selection in **subdivided populations.**
- **Implications**: Leads to genetic divergence; may contribute to speciation with low migration and accumulated differences.

THE ISLAND MODEL IN GA

- **Concept**: Population divided into subpopulations (islands).
- **Execution**: Each island runs its own version of the genetic algorithm.
- **Parallelism**: Typically, each island runs on a different processor.

THE ISLAND MODEL ADVANTAGES

- **Scalability**: Easily scalable across multiple processors, enhancing computational efficiency.
- **Robustness**: Reduces the likelihood of premature convergence by maintaining diverse subpopulations.
- **Adaptability**: Can test different **genetic algorithm variants** simultaneously.

IMMIGRATION POLICY PARAMETERS

1. No. of individuals undergoing migration
2. Frequency of migrations
3. Policy of selecting immigrants
4. Immigrant replacement policy
5. Topology of communication amongst islands
6. Nature of island communication (synchronous / asynchronous)

BASIC RANDOM IMMIGRANTS POLICY

- **Random Replacement**: Replace a fraction of the population with randomly generated individuals each generation.

- **Replacement Strategy**: Defines which individuals (random or worst) are replaced by immigrants.

ELITISM-BASED RANDOM IMMIGRATION POLICY

1. **Elite Retrieval**: Retrieve the elite from the previous generation.
2. **Immigrant Creation**: Use the elite as the base to create immigrants via mutation.
3. **Replacement**: Replace the worst individuals in the current population with these immigrants

COMMUNICATION TOPOLOGY

- Determine which islands communicate with each other and which not
- Communication is in the form of migration:
  - sending over individuals from one island to another
- Usually, islands that are neighboring can communicate

MIGRATION STRATEGY HYPERPARAMETERS SUMMARY

- **Migration Rate**: Frequency of migration events.
- **Migration Selection**: Criteria for selecting individuals to migrate (selection strategies).
- **Replacement Selection**: Criteria for selecting individuals to be replaced at the destination.

ISLAND INITIALIZATION APPROACHES

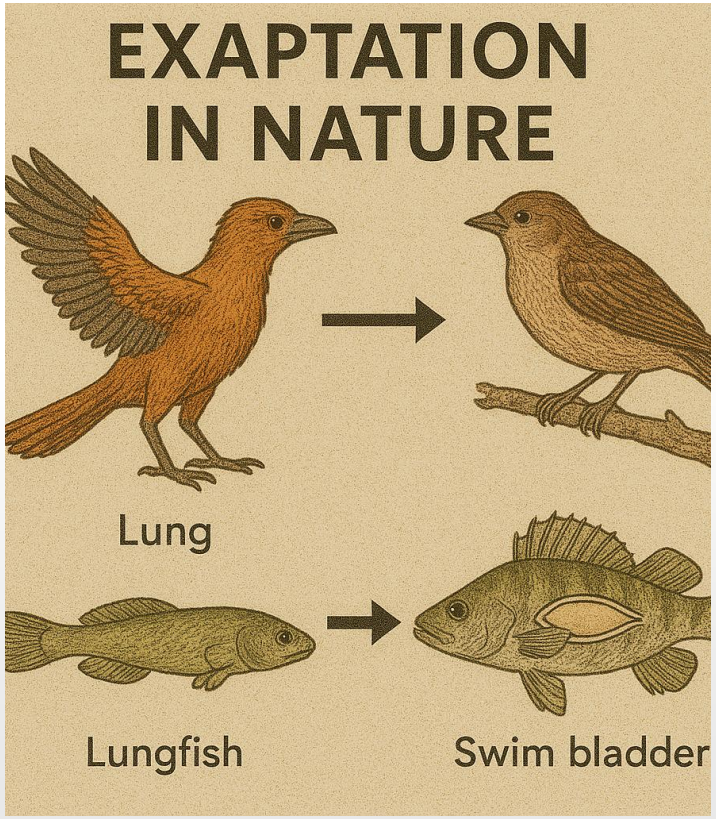
a) Set the <u>same</u> initial migration hyperparameters

b) Set <u>different</u> initial GA hypermeters e.g.

- Mutation Rate, Mutation Probability, and Crossover operators

- Islands could be **different Genetic Algorithm variants**

EXAPTATION - PREADAPTATION



EXAPTATION IN NATURE

- A process where a structure adapted for one function evolves to serve a new function.

- Bird feathers are a classic example:
  - initially they may have evolved for temperature regulation, but later were adapted for flight

PREADAPTATION IN GENETIC ALGORITHMS

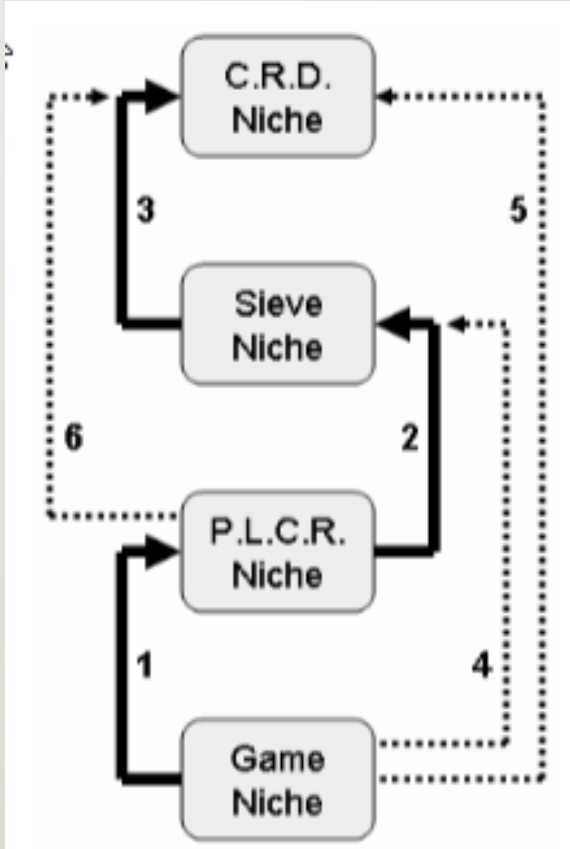
- **Seed population** using prior knowledge or near-optimal heuristics
- **Bootstrap with simpler fitness**, then shift to target objective
- **Progressive difficulty** to guide learning
- **Transfer & multitask evolution**: reuse evolved solutions across tasks
- Examples: clustering in TSP, human-designed sub-tours

EXAPTATION VIA NICHE MIGRATION

- Different niches are seeded – each with **different fitness function**
- Before next generation, check for potential migrant individuals to copy into another niche

EXAMPLE: COMPLEX SIX MIGRATION PATHS

- **Multiple Paths:** Indicates a complex migration strategy.
- **Niche Interaction**: Shows interaction and exchange between niches, promoting diversity.
- **Directionality**: Arrows indicate migration direction, affecting information flow.



EXAPTATION VIA MIGRATION

- An individual is chosen from source niche by size-two tournament selection
- Its fitness is evaluated by its niche fitness function
- **Viable** = same genetic representation
  - If **viable**: then migrated only if **better than least fit** in the destination
  - if it is **not viable** in the destination niche: No migration

THE END