Evolutionary Computation Part 2

CS454 AI-Based Software Engineering

Shin Yoo

Crossover Operators

- Offsprings inherit genes from their parents, but not in identical forms.
- Think Mendelian recombination of alleles; since we don’t have alleles, we actually recombine the whole genotype.

Crossover Operators

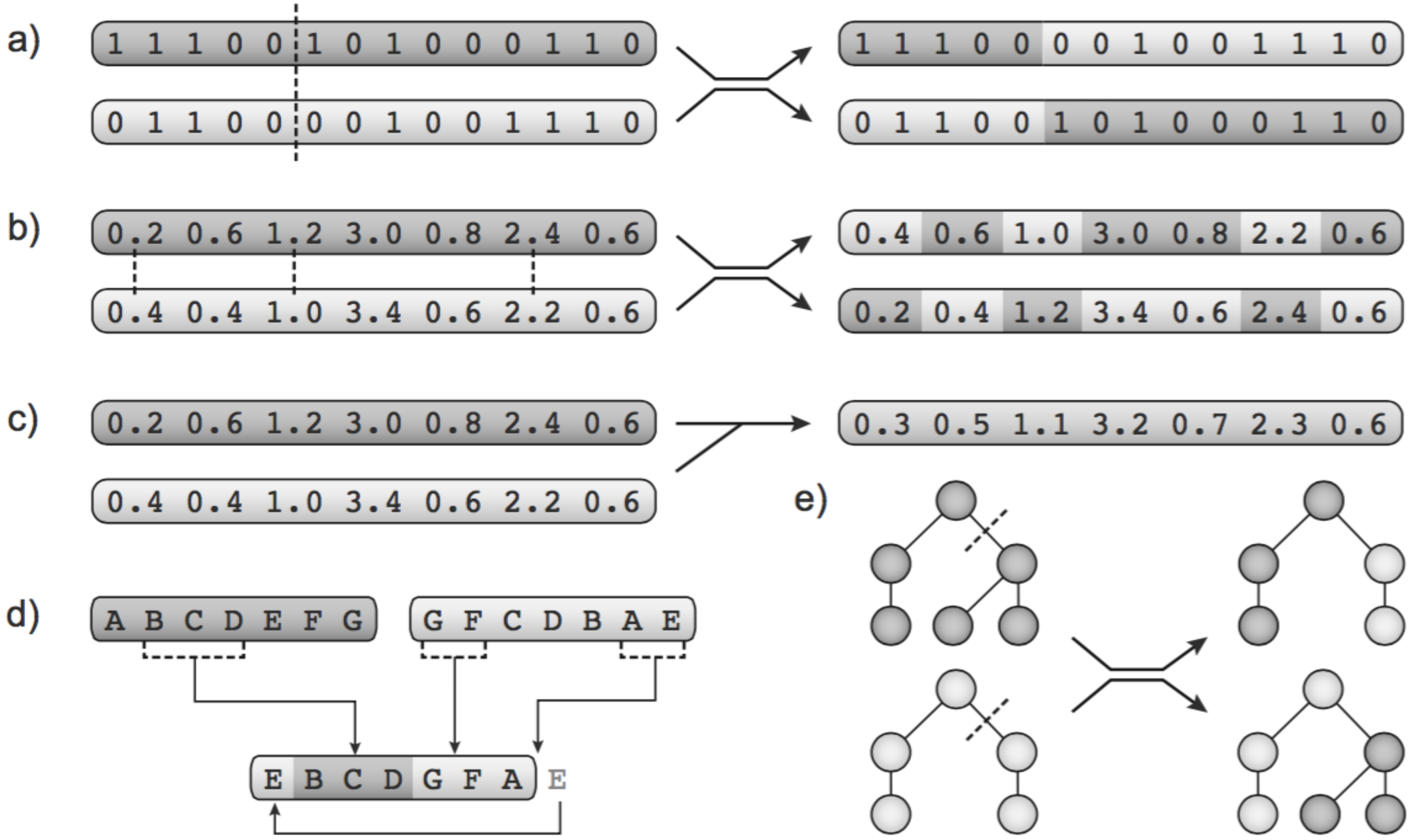


Figure 1.11 Examples of crossover operators. *a*) one-point; *b*) uniform; *c*) arithmetic; *d*) for sequences; *e*) for trees.

(from "Bio-inspired Artificial Intelligence: Theories, Methods, and Technologies"
by Dario Floreano and Claudio Mattiussi)

Mutation Operators

- This is, usually, the **only** way **new genetic material** is introduced into the population; without mutation, all we do is recombining the initial population (which was randomly generated).

Mutation Operator

- Small, local modifications to genotypes:
    - single bit-flip
    - adding/subtracting small amount to integers
    - swapping two elements in permutations
    - replacing one node in a tree with a different, compatible type

Generational Selection

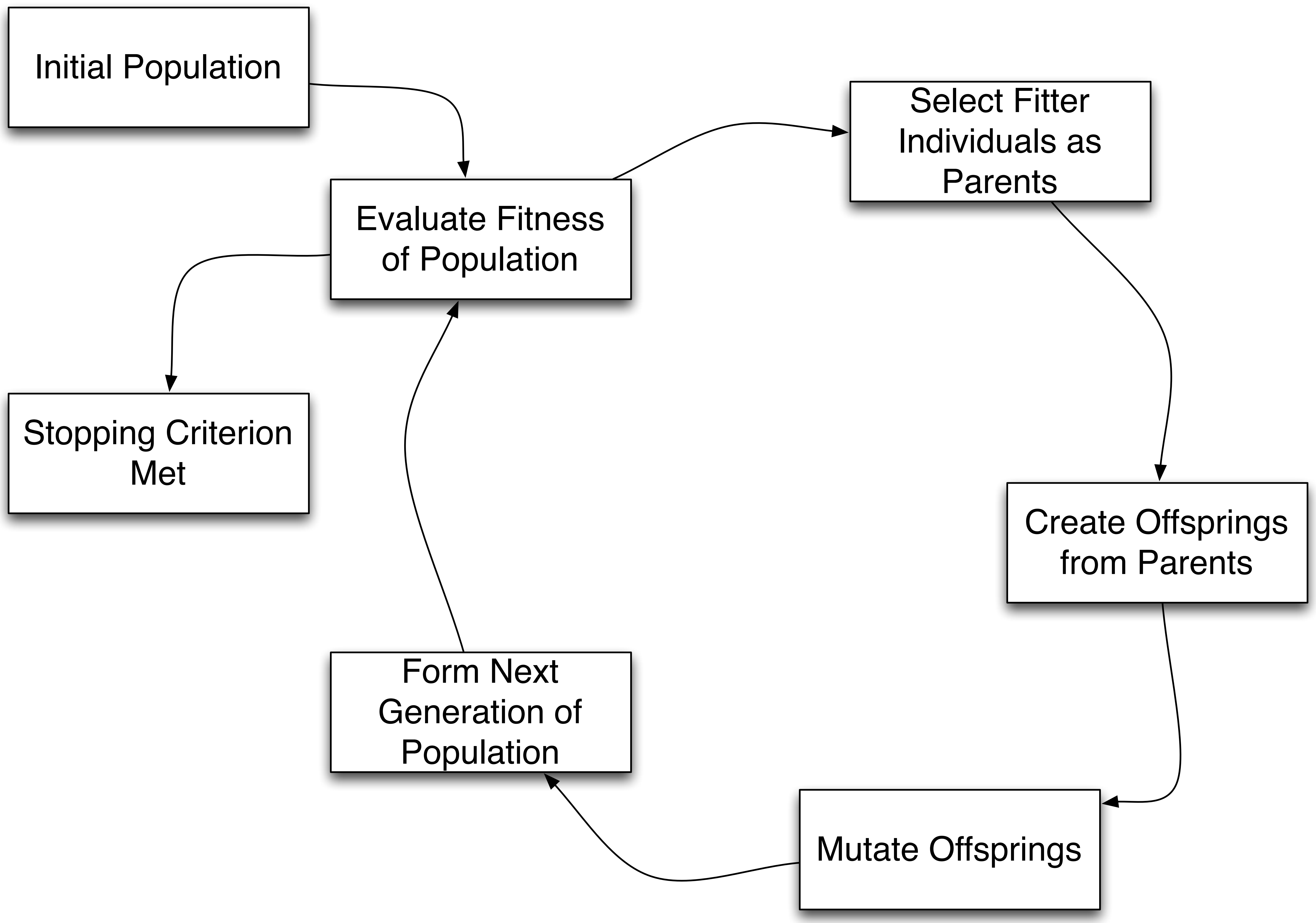
- Generational Replacement: the offsprings become the new current population (no parent survives)
- Elitism: maintain M best individuals from the parents' generation (reasons: noisy fitness, too strong mutations, too complicated search space…)
- Gradual Replacement: replace M worst individuals from the parents' generation with M best individuals from the offsprings.

Stopping Criterion

- Deciding one can be hard: these are stochastic algorithms, and you don’t know what the global optimum is.
- In reality, one of the following two:
  - Fixed number of fitness evaluations, or
  - When a good enough solution has been found

Parameters

- One weakness of GAs: many parameters to tune, no fixed guideline.
  - Population Size
  - Crossover Rate (usually high, we do want to crossover)
  - Mutation rate (usually low: e.g. `1/N` for 1 bit flip for each bit of length `N` bit string)
  - Elitism: the proportion of parent generation to preserve


Initial Population
Evaluate Fitness of Population
Select Fitter Individuals as Parents
Create Offsprings from Parents
Mutate Offsprings
Form Next Generation of Population
Stopping Criterion Met

Why (or when) does it work?

- Not much theoretic foundation.
- Schema Theory (John Holland, 1975): given genotypes of `k` symbols with length `l`, the schemata set is $\{s_0, \ldots, s_k, *\}$ where `*` means "don't care". There are $(k+1)^l$ schemas.
  - Intuitively, schemas can be thought of as non-consecutive building blocks to the solution.
  - Holland mathematically proved that selective reproduction allows exponentially increasing number of samples of schemas with better-than-average fitness, and exponentially decreasing number of schemas with lower-than-average fitness.

Schema Theorem

- Schema: a Hyperplane in the search space
  - For example, “11###”, where # is a “don’t care” symbol
- Instance: a concrete solution; a schema can include multiple instances
  - 11### contains 2^3 solution instances, e.g., 11111, 11001, …

Schema Theorem

- Fitness of a schema: the mean of all instances of the given schema
- Naturally, global optimization is to find the schema that has 1) the highest fitness value, and 2) zero “don’t care” symbol
- Holland argues that analysis of GA behavior becomes easier if we discuss in terms of groups of individuals (i.e., schema), rather than individual solutions

Features of a Schema

- Order: number of positions the schema that do not have the don’t care symbol
  - Given H = 1##0#1#0, $o(H) = 4$
- Defining length: distance between the outermost defined positions, which equals the number of possible crossover points
  - Given H = 1##0#1#0, $\delta(H) = 7$

Scheme Theorem

- We assume the following:
  - Fitness Proportional Selection
  - Single-point Crossover
  - Bitwise Mutation
  - Generational Survivor Selection (i.e., combine parent/offspring and select based on fitness ranking)

Schema Theorem

$$\mathbb{E}(m(H, t+1)) \geq m(H, t)\frac{f(H)}{a_t}(1 - \frac{\delta(H)}{l-1}p_c)(1 - p_m)^{o(H)}$$

- $m(H, t)$: number of instances of schema H at generation t
- a_t: average fitness of population at t
- l: length of chromosomes
- $o(H)$: order of H, $\delta(H)$: defining length of H
- p_c: crossover rate ,p_m: mutation rate

Case Study: Search-Based Software Testing

- Traditionally, GAs have been very popular with researchers: it appears fancy :)
- Is it really grounded on facts?
- Harman and McMinn (2007) compared the performance of HC and GA for automated test data generation for branch coverage for C programs.
  - `bibclean`, `eurocheck`, `gimp`, `space`, `spice`, `tiff`

Comparison of HC and GA

- There are branches that can only be covered by HC and GA respectively.
- Branches easier for GA: `bibclean`, especially in functions `check_ISBN()` and `check_ISSN()`.
- Why?

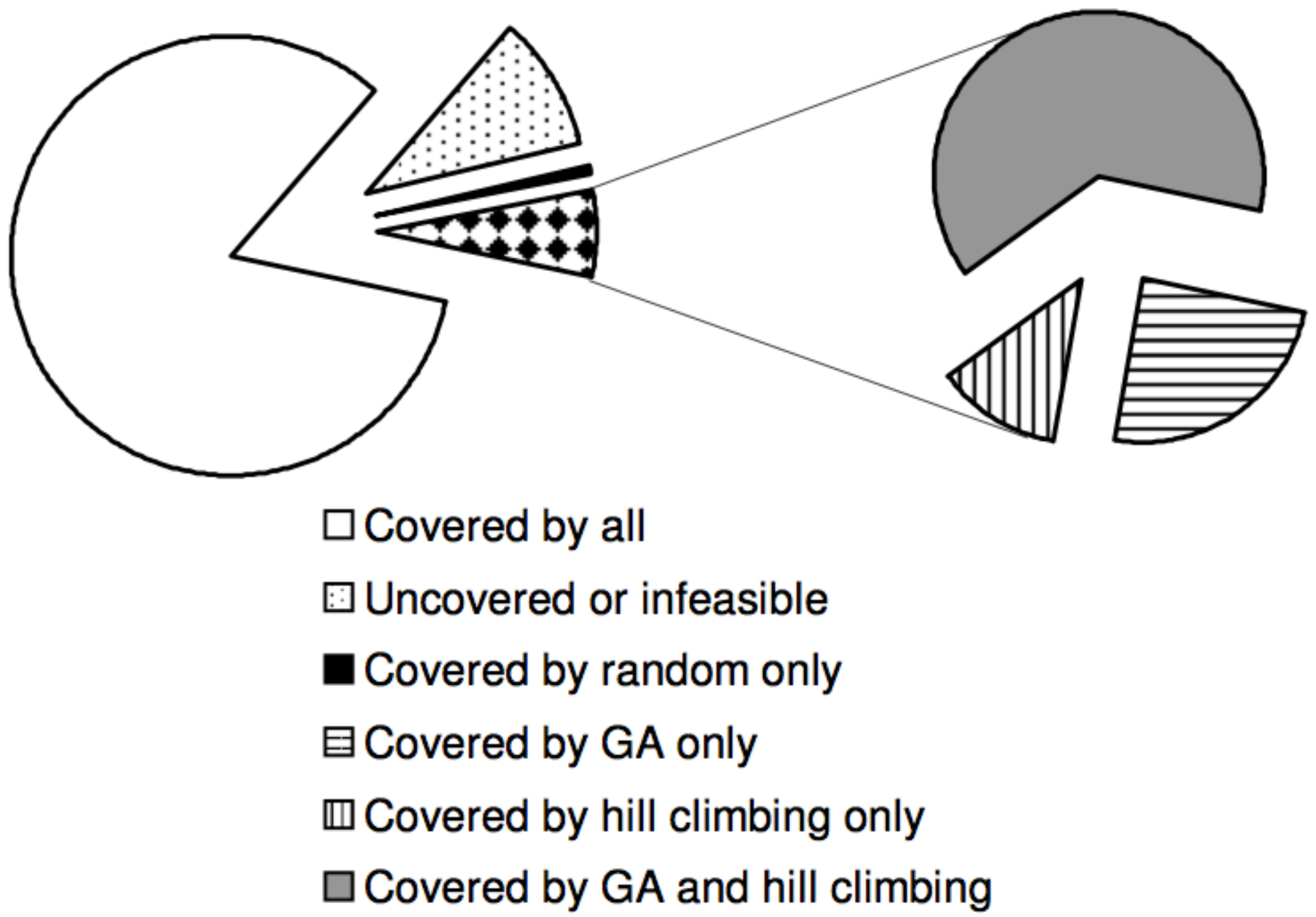


Figure 6: Pie chart showing proportions of branches covered by the different search methods

Proportion of branches covered by different algorithms (Harman & McMinn, 2007)

Schema Theory in Work

- “…Registration group identifiers have primarily been allocated within the **978 prefix element**. The single-digit group identifiers within the **978** prefix element are: 0 or 1 for English-speaking countries; 2 for French-speaking countries; 3 for German-speaking countries; 4 for Japan; 5 for Russian-speaking countries; and 7 for People's Republic of China. An example 5-digit group identifier is **99936**, for Bhutan. The allocated group IDs are: **0–5**, **600–621**, **7**, **80–94**, **950–989**, **9926–9989**, and **99901–99976**.” (from Wikipedia entry for ISBN)

Schema Theory in Work

- Once a small schema is formed (e.g., `9*`), it can be used as a building block for a larger schema (e.g. `99*`). Crossover allows assembly of different building blocks.

- This is also called **Building Block Hypothesis**: GAs work best for problems with building block structure in their solutions.

Computational Complexity

- !!?@#?!@#!!??!?
- Can only be considered in relation to a specific problem; often, analysis is done to problems with well defined structure, using probabilistic approach.

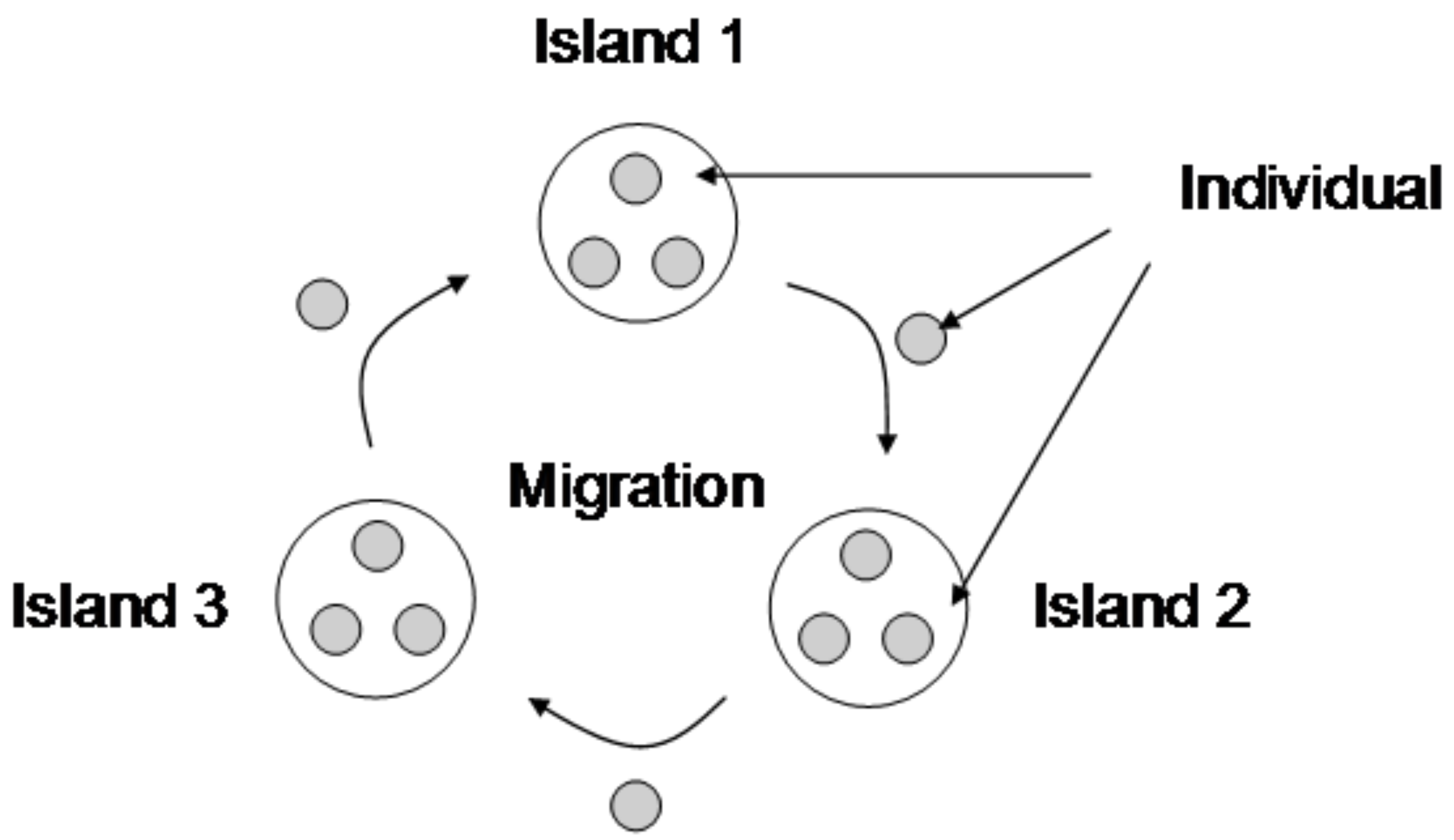
Population Diversity

- Just like biodiversity, population diversity is important for GA. Even solutions with worst fitness may still contain valuable schemas.
- Various auxiliary mechanisms have been developed to preserve and promote population diversity.

Island GA

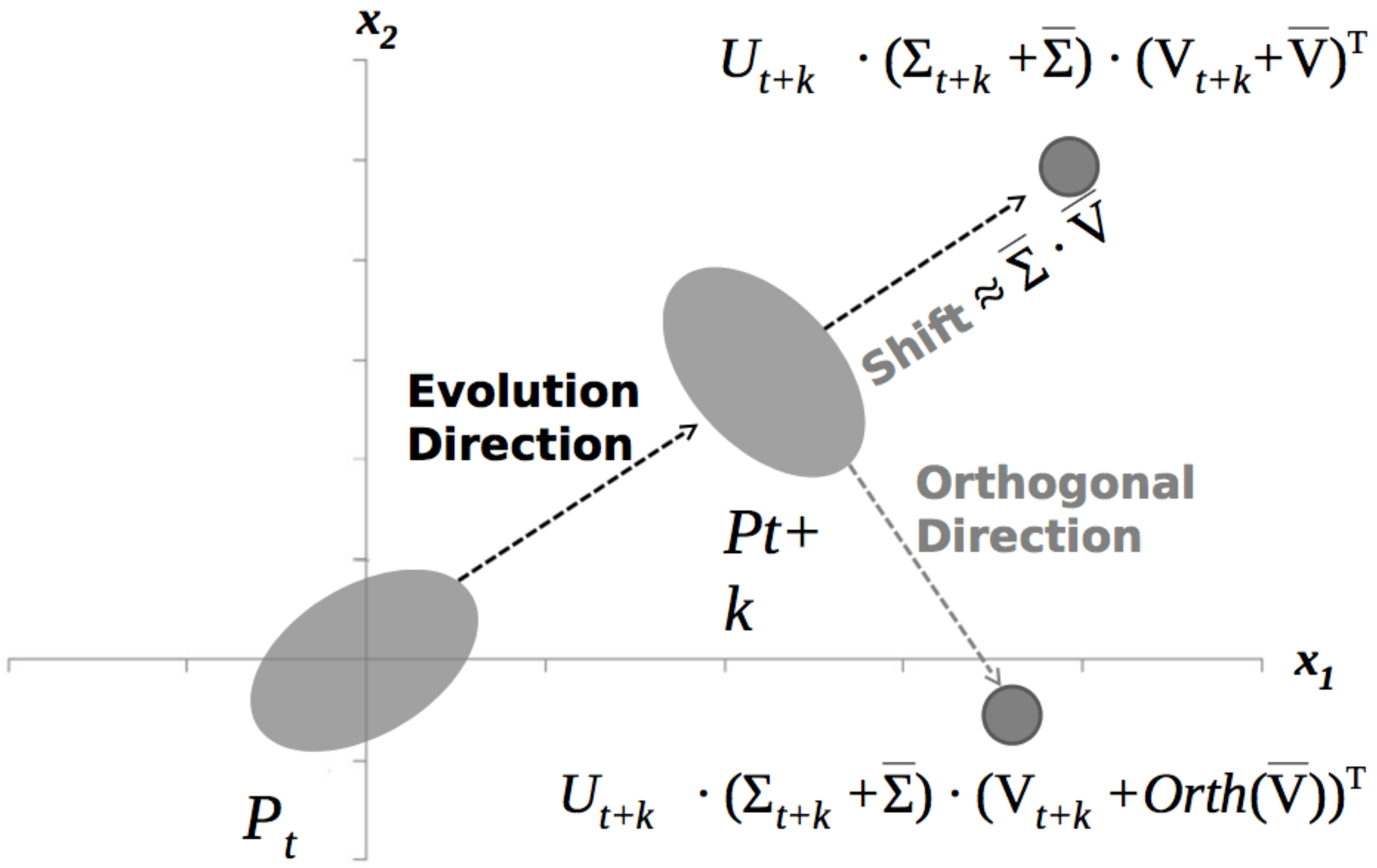
- Let multiple populations evolve in separation; every now and then, move individuals between islands.

- The Island Model Genetic Algorithm: On Separability, Population Size and Convergence, *Darrell Whitley, Soraya Rana, Robert B. Heckendorn,* Journal of Computing and Information Technology, Vol. 7 (1999), pp. 33-47



Orthogonal Exploration

- Determine the direction of evolution; forcefully replace worst solutions with generated solutions that explore orthogonal direction.
- Orthogonal exploration of the search space in evolutionary test case generation, *F. M. Kifetew, A. Panichella, A. De Lucia, R. Oliveto, and P. Tonella,* in Proceedings of the 2013 International Symposium on Software Testing and Analysis, ISSTA 2013



Real Applications

- GA is a **BIG** toolbox, full of specialised operators, representation, and other assorted tricks.
- Just like any other AI technique, the more domain knowledge you have, the better your optimisation will be.

Summary

- Understand the framework of Darwinian evolution.
- Optimisation using evolution works, based on:
  - Selection pressure
  - Schema theory (one possible explanation)
- Understand various genetic operators.
- Understand the importance of population diversity.