Chapter 9: Genetic Algorithms

Computational Intelligence: Second Edition

Contents

Compact Overview

- First proposed by Fraser in 1957
- Later by Bremermann in 1962 and Reed et al in 1967
- Popularized by Holland in 1975
- Genetic algorithms (GA) focuss on genetic evolution
- Main driving operators:
  - Selection, to model survival of the fittest
  - Recombination, to model reproduction
  - Mutation, to introduce new genetic material

Components

Proposed by Holland, having the following characteristics:

- A bitstring representation was used
- Proportional selection was used to select parents
- One-point cross-over
- Uniform mutation

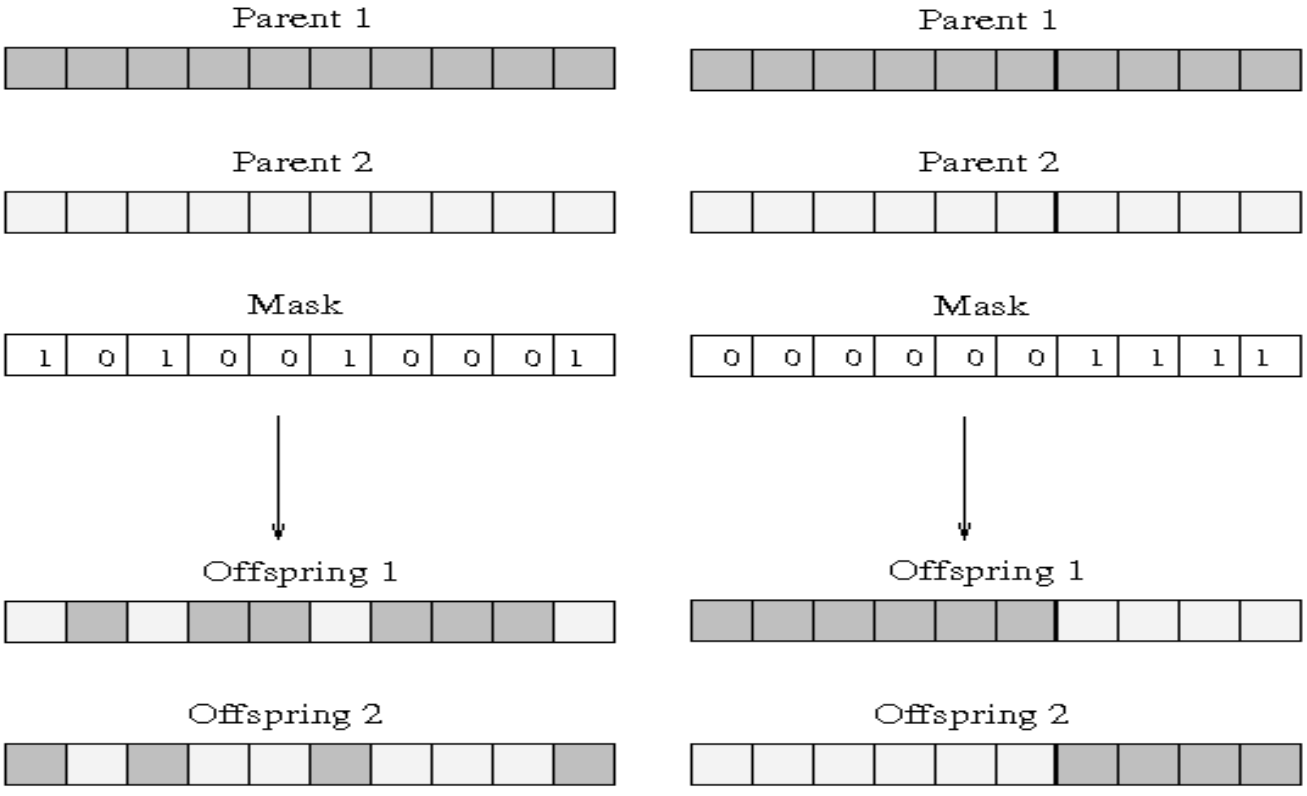
About Crossover

- To produce offspring by recombining genetic material from parents
- Main categories of operators:
  - **asexual** – an offspring is generated from one parent
  - **sexual** – two parents are used to produce one or two offspring
  - **multi-recombination** – more than two parents are use to produce one or more offspring
- Selection of parents:
  - Use any selection operator
  - Crossover occurs at a crossover probability, p_c

Selection Issues

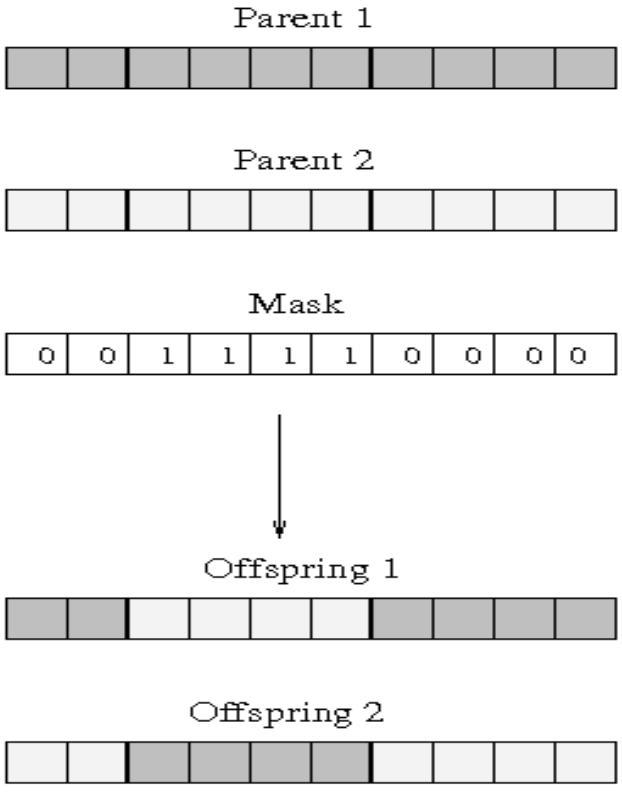
- Consider the following about selection:
  - The same individual may be selected as both parents
  - Same individual may take part in producing more than one offspring
- Recombination with replacement:
  - If one offspring is generated, it may replace the worst parent
  - Restriction can be imposed that offspring must be better
  - Boltzmann selection can be used to decide if offspring should replace the parent
  - Offspring may replace the worst individual in the population

Binary Representations: Figure 9.1



(a) Uniform Crossover (b) One-point Crossover

Binary Representations: Figure 9.1 (cont)



(c) Two-point Crossover

Generic Algorithm for Bitstring Crossovers: Algorithm 9.1

Let $\tilde{\mathbf{x}}_1(t) = \mathbf{x}_1(t)$ and $\tilde{\mathbf{x}}_2(t) = \mathbf{x}_2(t)$;
if $U(0,1) \leq p_c$ **then**
    Compute the binary mask, $\mathbf{m}(t)$;
    for $j = 1, \ldots, n_x$ **do**
        if $m_j = 1$ **then**
            //swap the bits
            $\tilde{\mathbf{x}}_{1j}(t) = \mathbf{x}_{2j}(t)$;
            $\tilde{\mathbf{x}}_{2j}(t) = \mathbf{x}_{1j}(t)$;
        end
    end
end

One-Point/Two-Point Crossover Mask: Algorithm 9.2, 9.3

Algorithm 1: One-Point Crossover Mask Calculation

Select the crossover point, $\xi \sim U(1, n_x - 1)$;
Initialize the mask: $m_j(t) = 0$, for all $j = 1, \ldots, n_x$;
for $j = \xi + 1$ *to* n_x **do**
    $m_j(t) = 1$;
end

Algorithm 2: Two-Point Crossover Mask Calculation

Select the two crossover points, $\xi_1, \xi_2 \sim U(1, n_x)$;
Initialize the mask: $m_j(t) = 0$, for all $j = 1, \ldots, n_x$;
for $j = \xi_1 + 1$ *to* ξ_2 **do**
    $m_j(t) = 1$;
end

One-Point/Two-Point Crossover Mask: Algorithm 9.4

Algorithm 3: Uniform Crossover Mask Calculation

Initialize the mask: $m_j(t) = 0$, for all $j = 1, \ldots, n_x$;
for $j = 1$ *to* n_x **do**
    if $U(0,1) \leq p_x$ **then**
        $m_j(t) = 1$;
    end
end

Multi-Parent Crossover

- Select n_p parent vectors
- Majority mating generates one offspring using

$$\tilde{\mathbf{x}}_{ij}(t) = \begin{cases} 0 & \text{if } n'_\mu \geq n_\mu/2,\ l = 1, \ldots, n_\mu \\ 1 & \text{otherwise} \end{cases}$$

  n'_μ is the number of parents with $x_{lj}(t) = 0$
- Multi-parent n-point crossover:
  - Select $n_\mu - 1$ identical crossover points in n_μ parents
  - One offspring is generated by selecting one segment from each parent

Crossover Hillclimbing

- Select two parents
- Continue to produce offspring until an offspring is better than a parent
- Crossover continues with these offspring as the parents
- If better offspring can not be found within a time limit, the worst parent is replaced with a randomly selected parent.

Floating-point Representation

- Discrete recombination:
  - One-point, two-point, n-point, uniform crossover
- Intermediate recombination – blends components across selected parents
- Linear recombination:
  - Select two parents, $\mathbf{x}_1(t)$ and $\mathbf{x}_2(t)$
  - Generate three offspring: $(\mathbf{x}_1(t) + \mathbf{x}_2(t)), (1.5\mathbf{x}_1(t) - 0.5\mathbf{x}_2(t))$ and $(-0.5\mathbf{x}_1(t) + 1.5\mathbf{x}_2(t))$
  - Select the best two as the offspring

Floating-point Representation (cont)

- Directional heuristic crossover:

$$\tilde{x}_{ij}(t) = U(0,1)(\mathbf{x}_{2j}(t) - \mathbf{x}_{1j}(t)) + \mathbf{x}_{2j}(t)$$

  subject to the constraint that parent $\mathbf{x}_2(t)$ cannot be worse than parent $\mathbf{x}_1(t)$
- Arithmetic crossover:
  - Multi-parent recombination
  - One offspring generated using

$$\tilde{x}_{ij}(t) = \sum_{l=1}^{n_\mu} \gamma_l x_{lj}(t), \quad \sum_{l=1}^{n_\mu} \gamma_l = 1$$

  - For $n_\mu = 2$, then

$$\tilde{x}_{ij}(t) = (1-\gamma)x_{1j}(t) + \gamma x_{2j}(t),\ \gamma \in [0,1]$$

Floating-point Representation (cont)

- Blend crossover (BLX-α):

$$\tilde{x}_{ij}(t) = (1 - \gamma_j)x_{1j}(t) + \gamma_j x_{2j}(t)$$

  with $\gamma_j = (1 + 2\alpha)U(0, 1) - \alpha$

- The BLX-α operator randomly picks, for each component, a random value in the range

$$[x_{1j}(t) - \alpha(x_{2j}(t) - x_{1j}(t)), x_{2j}(t) + \alpha(x_{2j}(t) - x_{1j}(t))]$$

- BLX-α assumes that $x_{1j}(t) < x_{2j}(t)$
- BLX-α explores more than arithmetic crossover

Floating-point Representation (cont)

- Geometrical crossover

$$\tilde{x}_{ij}(t) = (x_{1j}x_{2j})^{0.5}$$

- Generalized to multi-parent recombination

$$\tilde{x}_{ij}(t) = (x_{1j}^{\alpha_1} x_{2j}^{\alpha_2} \dots x_{n_\mu j}^{\alpha_{n_\mu}})$$

n_μ is the number of parents, and $\sum_{l=1}^{n_\mu} \alpha_l = 1$

Floating-point Representation (cont)

- Simulated binary crossover (SBX):

$$\begin{aligned}\tilde{x}_{1j}(t) &= 0.5[(1+\gamma_j)x_{1j}(t) + (1-\gamma_j)x_{2j}(t)] \\ \tilde{x}_{2j}(t) &= 0.5[(1-\gamma_j)x_{1j}(t) + (1+\gamma_j)x_{2j}(t)]\end{aligned}$$

where

$$\gamma_j = \begin{cases} (2r_j)^{\frac{1}{\eta+1}} & \text{if } r_j \leq 0.5 \\ \left(\frac{1}{2(1-r_j)}\right)^{\frac{1}{\eta+1}} & \text{otherwise} \end{cases} \quad (1)$$

$r_j \sim U(0,1)$, and $\eta > 0$ is the distribution index

- Generates offspring symmetrically about the parents
- No bias to any parent
- Large values of $\eta \rightarrow$ higher probability that offspring will be created near the parents
- Small η values, offspring will be more distant from the parents

Floating-point Representation (cont)

- Multi-parent crossover:
  - Intensify the explorative capabilities compared to two-parent operators
  - By aggregating information from multiple parents, more disruption is achieved
  - The resemblance between offspring and parents are on average smaller compared to two-parent operators
- Unimodal distributed (UNDX) operator
  - Two or more offspring are generated using three parents
  - Offspring are created from an ellipsoidal probability distribution
  - One of the axes is formed along the line that connects two of the parents
  - The extent of the orthogonal direction is determined from the perpendicular distance of the third parent from the axis

Floating-point Representation (cont)

- Generalized UNDX
  - Generalized to work with any number of parents, with $3 \leq n_\mu \leq n_s$
  - $n_\mu - 1$ parents are randomly selected
  - Calculate center of mass, $\overline{\mathbf{x}}(t)$, where

$$\overline{x}_j(t) = \sum_{l=1}^{n_\mu - 1} x_{lj}(t)$$

  - From the mean, $n_\mu - 1$ direction vectors,

$$\mathbf{d}_l(t) = \mathbf{x}_l(t) - \overline{\mathbf{x}}(t)$$

    are computed

Floating-point Representation (cont)

- Generalized UNDX (cont)
  - Compute the direction cosines

$$\mathbf{e}_l(t) = \mathbf{d}_l(t)/|\mathbf{d}_l(t)|$$

    where $|\mathbf{d}_l(t)|$ is the length of the vector
  - Select a random parent with index n_μ
  - Let $\mathbf{x}_{n_\mu}(t) - \overline{\mathbf{x}}(t)$ be the vector orthogonal to all $\mathbf{e}_l(t)$
  - Let $\delta = |\mathbf{x}_{n_\mu}(t) - \overline{\mathbf{x}}(t)|$
  - Let $\mathbf{e}_l(t), l = n_\mu, \ldots, n_s$ be the orthonormal basis of the subspace orthogonal to the subspace spanned by the direction cosines, $\mathbf{e}_l(t),\ l = 1, \ldots, n_\mu - 1$

Floating-point Representation (cont)

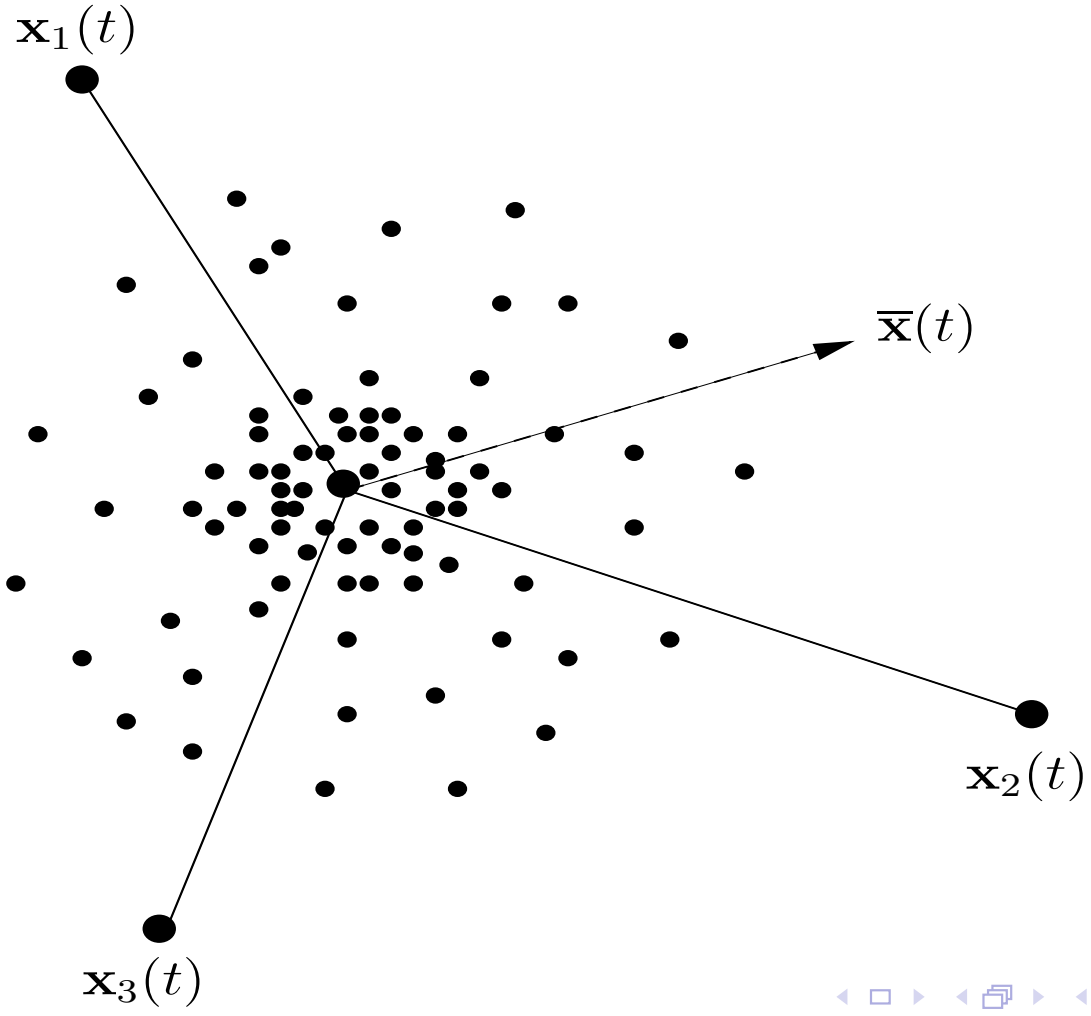
- Generalized UNDX (cont)
  - Offspring are then generated using

$$\tilde{\mathbf{x}}_i(t) = \bar{\mathbf{x}}(t) + \sum_{l=1}^{n_\mu - 1} N(0, \sigma_1^2)|\mathbf{d}_l|\mathbf{e}_l + \sum_{l=n_\mu}^{n_s} N(0, \sigma_2^2)\delta\mathbf{e}_l(t)$$

$$\sigma_1 = \frac{1}{\sqrt{n_\mu - 2}} \text{ and } \sigma_2 = \frac{0.35}{\sqrt{n_s - n_\mu - 2}}$$

  - Any number of offspring can be created, sampled around the center of mass of the selected parents
  - A higher probability is assigned to create offspring near the center rather than near the parents

Floating-point Representation (cont): Figure 9.2 (a)

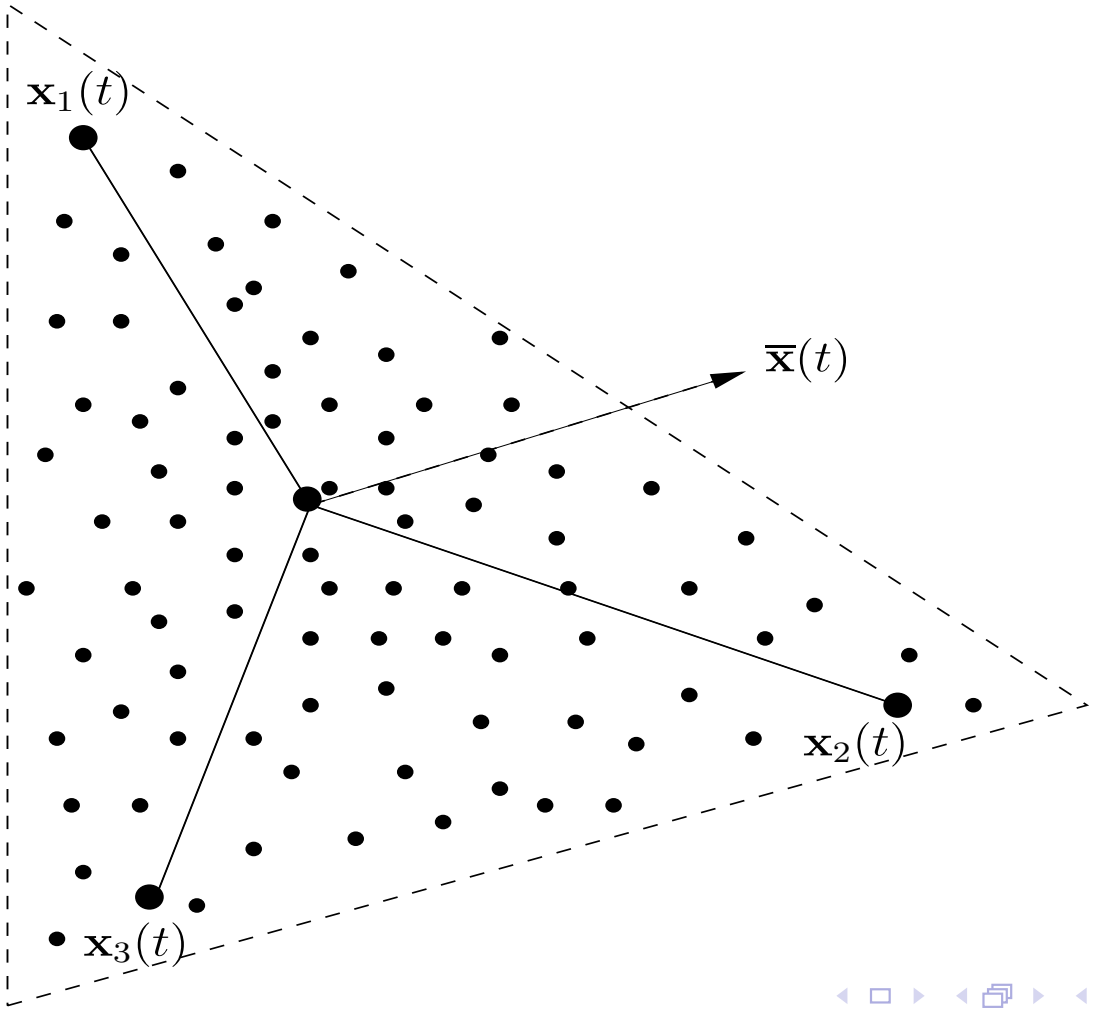


Floating-point Representation (cont)

- Simplex crossover (SPX):
  - Center of mass approach to recombination
  - Select $n_\mu > 2$ parents
  - Find the best parent, $\mathbf{x}_1(t)$
  - Find the worst parent, $\mathbf{x}_2(t)$
  - Compute center of mass, excluding worst parent
  - Generate one offspring,

$$\tilde{\mathbf{x}}(t) = \overline{\mathbf{x}}(t) + (\mathbf{x}_1(t) - \mathbf{x}_2(t))$$

Floating-point Representation (cont): Figure 9.2 (b)



Floating-point Representation (cont)

- Parent-centric operator (PCX):
  - Offspring are generated around selected parents
  - Select n_μ parents
  - Compute center of mass, $\overline{\mathbf{x}}(t)$
  - For each offspring, select a random parent, $\mathbf{x}_i(t)$
  - Compute direction vector

$$\mathbf{d}_i(t) = \mathbf{x}_i(t) - \overline{\mathbf{x}}(t)$$

  - From the other $n_\mu - 1$ parents, compute perpendicular distances, δ_l to the line $\mathbf{d}_i(t)$
  - Calculate the average over these distances

$$\overline{\delta} = \frac{\sum_{l=1, l \neq i}^{n_\mu} \delta_l}{n_\mu - 1}$$

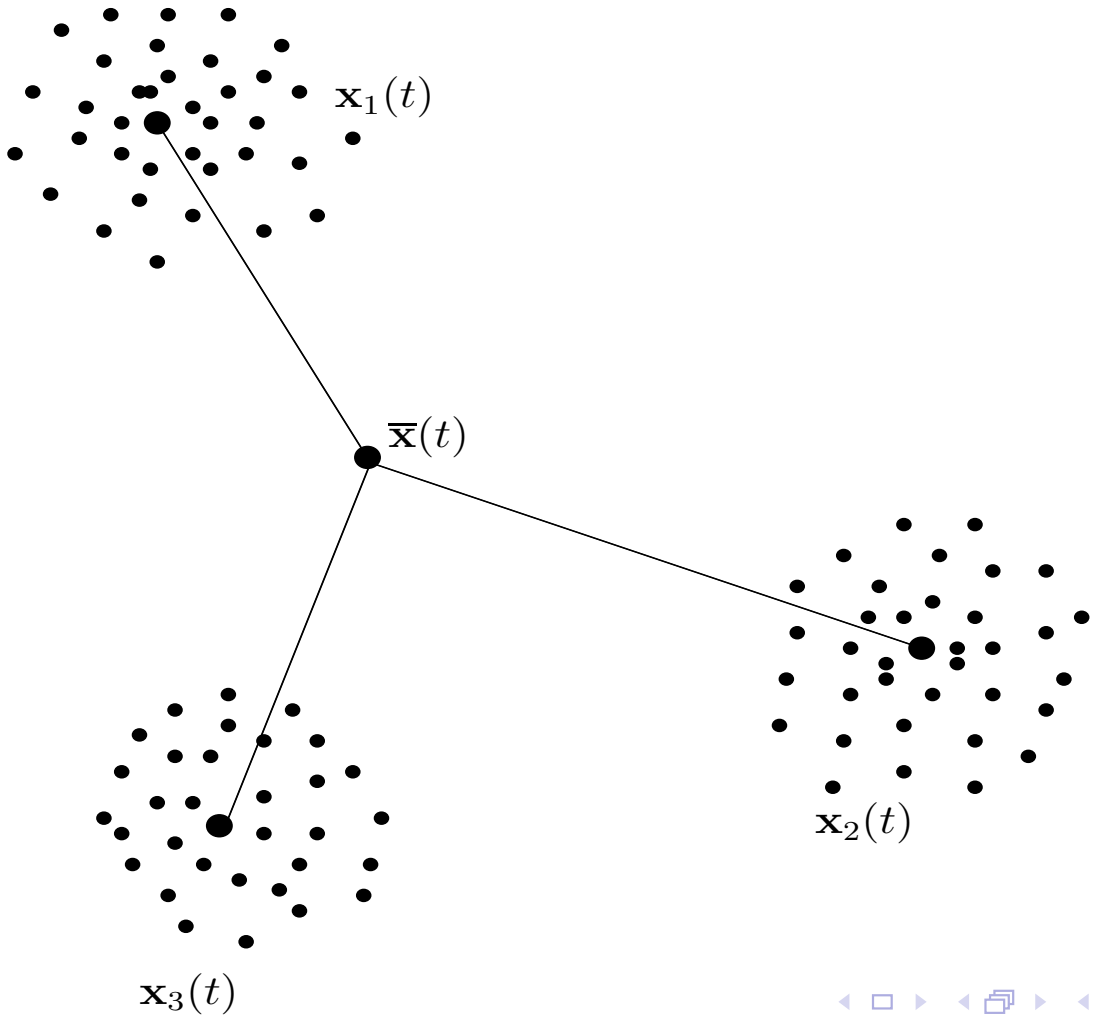
Floating-point Representation (cont)

- Parent-centric operator (cont):
  - Offspring is generated using

$$\tilde{\mathbf{x}}_i(t) = \mathbf{x}_i(t) + N(0, \sigma_1^2)|\mathbf{d}_i(t)| + \sum_{l=1, i \neq l}^{n_\mu} N(0, \sigma_2^2)\overline{\delta}\mathbf{e}_l(t)$$

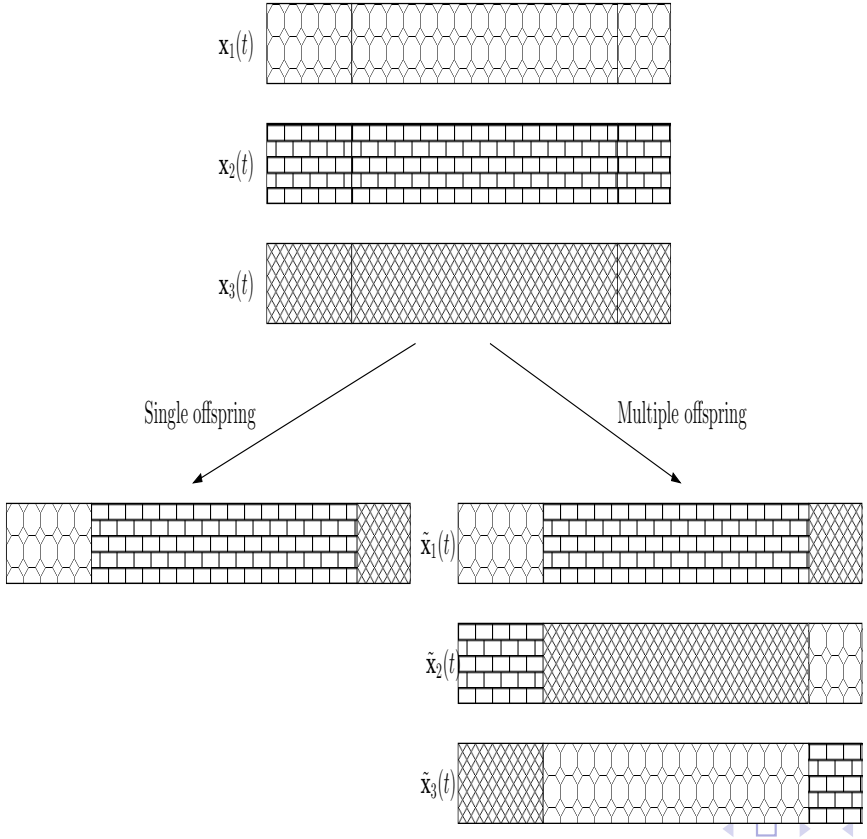
$\mathbf{e}_l(t)$ are the $n_\mu - 1$ orthonormal bases that span the subspace perpendicular to $\mathbf{d}_i(t)$

Floating-point Representation (cont): Figure 9.2 (b)



Floating-point Representation (cont)

- Diagonal crossover (fig 9.3):
  - A generalization of *n*-point crossover for more than two parents

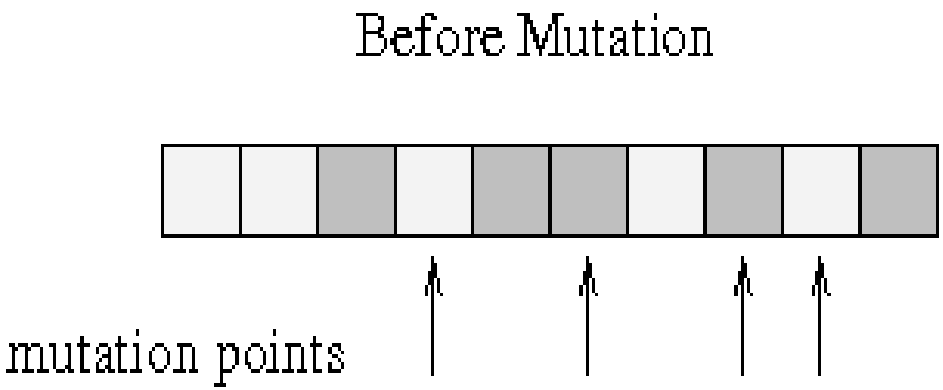


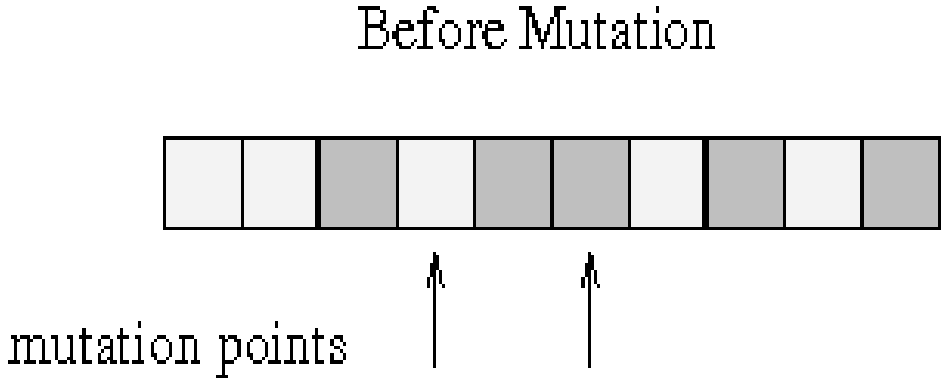
About Mutation

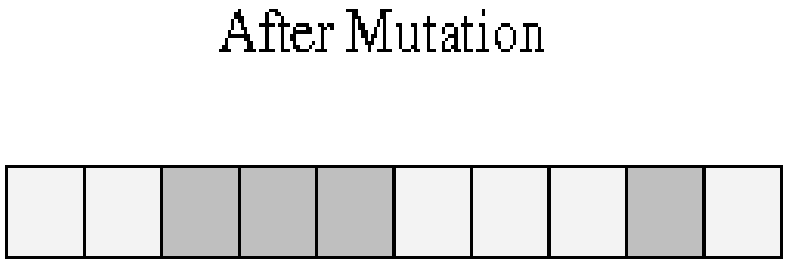
- The objective of mutation is to introduce new genetic material into an existing individual
- Adds diversity to the genetic characteristics of the population
- Ensures that the full range of allele is accessible for each gene
- Applied at a mutation probability, p_m
- Given that each gene is mutated at probability p_m, the probability that an individual will be mutated is given by

$$Prob(\tilde{\mathbf{x}}_i(t) \text{ is mutated}) = 1 - (1 - p_m)^{n_x}$$

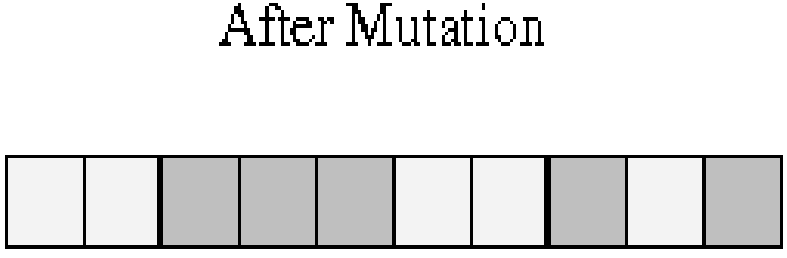
Binary Representations (fig 9.4)







(d) Random Mutate



(e) Inorder Mutate

Binary Representations: Algorithms 9.6, 9.7

Algorithm 4: Uniform/Random Mutation

for $j = 1, \ldots, n_x$ **do**
  if $U(0,1) \leq p_m$ **then**
    $x'_{ij}(t) = \neg \tilde{x}_{ij}(t)$;
  end
end

Algorithm 5: Inorder Mutation

Select mutation points, $\xi_1, \xi_2 \sim U(1, \ldots, n_x)$;
for $j = \xi_1, \ldots, \xi_2$ **do**
  if $U(0,1) \leq p_m$ **then**
    $x'_{ij}(t) = \neg \tilde{x}_{ij}(t)$;
  end
end

Floating-Point Representations

- One of the first proposals, using uniform mutation:

$$x'_{ij}(t) = \begin{cases} \tilde{x}_{ij}(t) + \Delta(t, x_{max,j} - \tilde{x}_{ij}(t)) & \text{if a random digit is 0} \\ \tilde{x}_{ij}(t) + \Delta(t, \tilde{x}_{ij}(t) - x_{min,j}(t)) & \text{if a random digit is 1} \end{cases}$$

  $\Delta(t, x)$ returns random values from the range $[0, x]$

- See the operators in chapters 11 and 12.

Macromutation – Headless Chicken

- Offspring is created by recombining a parent individual with a randomly generated individual
- Any crossover operator is used to perform the recombination
- No concept of inheritence
- Increase in diversity

Effects of Control Parameters

- Population size
- Mutation rate, p_m
- Crossover rate, m_c
- Dynamic mutation rates:
  - Mutation rate decreases exponentially with generation number

$$p_m(t) = \frac{1}{240} + \frac{0.11375}{2^t}$$

  - Mutation rate per bit, where n_b is the most significant bit

$$p_m(j) = \frac{0.3528}{2^{j-1}}$$

Effects of Control Parameters (cont)

- Large initial mutation rate favors exploration
- As mutation rate decreases, exploitation is favored
- For floating-point representations, mutation step size influences performance:
  - Large mutational step size – large jumps in search space, better exploration
  - Initially large step sizes, with step sizes reduced over time
  - Let step size be proportional to fitness of individual
  - The worse an individual, the larger the mutational step size
- Best values of parameters, including choice of evolutionary operators, are problem-dependent

Generation Gap Methods

- Select process can be described by two steps:
  - Parent selection
  - A replacement strategy that decides if offspring will replace parents, and which parents to replace
- Based on replacement strategy used, two main classes of GAs are identified:
  - Generational genetic algortihms (GGA)
    - Replacement strategy replaces all parents with their offspring after all offspring have been created and mutated
    - No overlap between populations of different generations
  - Steady state genetic algorithms (SSGA)
    - Immediately after an offspring is created and mutated, a replacement strategy is executed
    - Some overlap exists between populations of different generations

Generation Gap Methods: Replacement Strategies

- The amount of overlap between the current and new populations is referred to as the generation gap
- Replacement strategies for SSGA:
  - **Replace worst**, where the offspring replaces the worst individual of the current population
  - **Replace random**, where the offspring replaces a randomly selected individual of the current population
  - **Kill tournament**, where the individual to be replaced is selected using tournement selection to select the worst individual
  - **Replace oldest**, where a first-in-first-out strategy is followed by replacing the oldest individual of the current population

Generation Gap Methods: Replacement Strategies

- Replacement strategies for SSGA (cont):
  - **Conservative selection**combines a first-in-first-out replacement strategy with a modified deterministic binary tournament selection, where one individual is always the oldest individual
  - **Elitist** strategies, where the best individual is excluded from selection
  - **Parent-offspring** competition, where a selection strategy is used to decide if an offspring replaces one of its own parents

Interactive Evolution

- For some application, for example, evolving
  - art,
  - music,
  - animation,

  it is not possible to define an absolute fitness function
- For such applications, subjective judgment is needed, based on human intuition, aesthetical values or taste
- Interaction of a human evaluator as the "fitness function" is therefor required
- Interactive evolution (IE) involves a human user online into the selection and variation processes

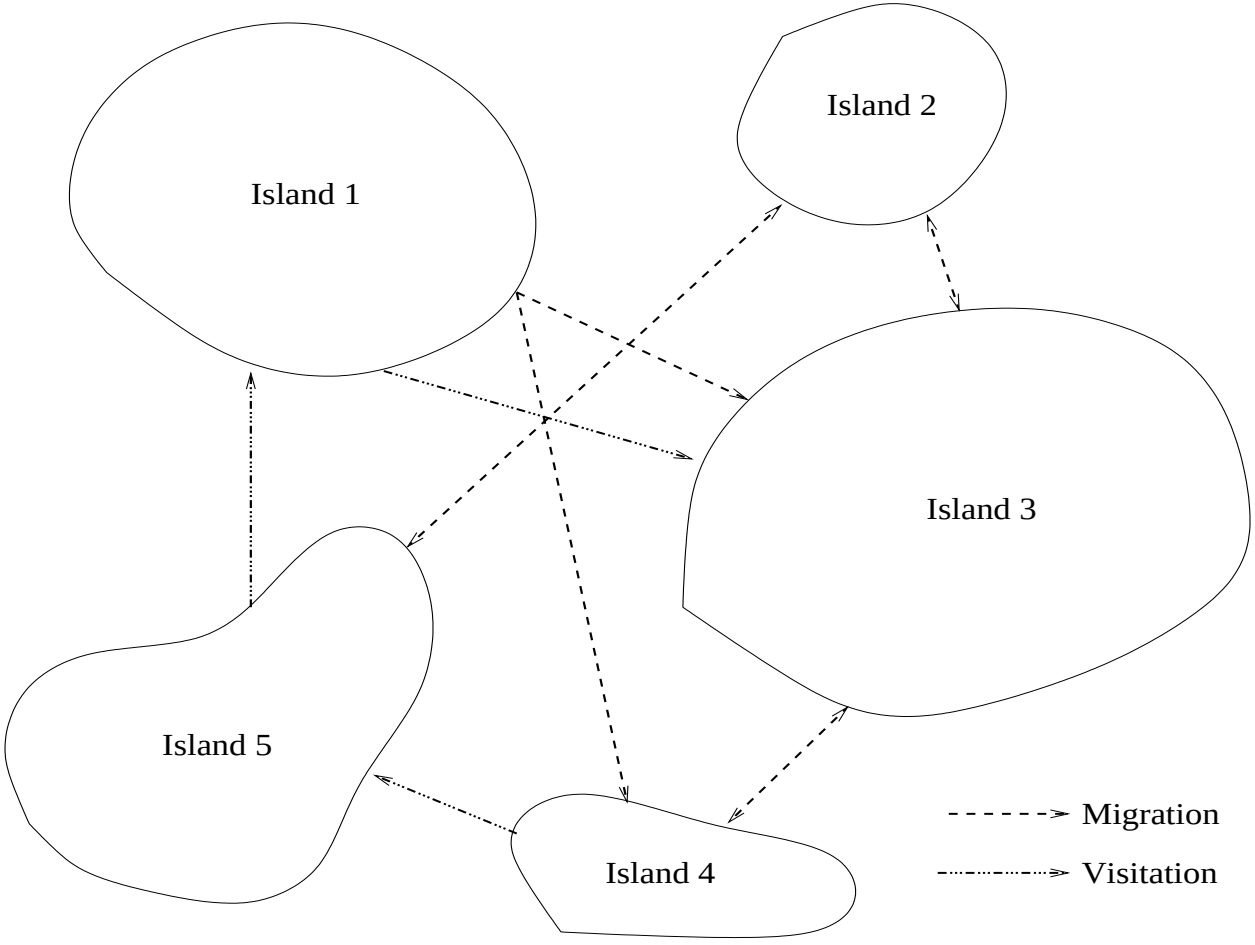
Interactive Evolution: Algorithm 9.9

Set the generation counter, $t = 0$;
Initialize the control parameters;
Create and initialize the population, $\mathcal{C}(0)$, of n_s individuals;
while *stopping condition(s) not true* **do**
    Determine reproduction operators, either automatically or via interaction;
    Select parents via interaction;
    Perform crossover to produce offspring;
    Mutate offspring;
    Select new population via interaction;
end

Interactive Evolution (cont)

- Interactive selection step:
  - Requires that the phenotype of individuals be generated from the genotype
  - Phenotype is then visualized
  - Based on the visual representations of candidate solutions, the user selects those individuals that will take part in reproduction
  - User also selects individuals that will survive to the next generation

Island Genetic Algorithms: Figure 9.5



Island Genetic Algorithms (cont)

- Migration policies specify:
  - A communications topology, which determines the migration paths between islands
  - A migration rate, which determines the frequency of migration
    - If migration occurs too early, the number of good building blocks in the migrants may be too small to have any influence at their destinations
    - Usually, migration occurs when each population has converged
    - After exchange of individuals, all populations are restarted
  - A selection machanism, to decide which individuals will migrate
  - A replacement strategy, to decide which individual of the destination island will be replaced

Island Genetic Algorithms (cont)

- Based on the selection and replacement strategies, island GAs can be grouped into two classes of algorithms:
  - Static island GAs
    - A topology is used to determine migration paths
  - Dynamic island GAs
    - Migration decisions are made probabilistically
- Static island GAs:
  - Deterministic selection and replacement strategies
  - A good migrant replaces a bad individual
  - A good migrant replaces a randomly selected individual
  - A randomly selected migrant replaces a bad individual
  - A randomly selected migrant replaces a randomly selected individual

Island Genetic Algorithms (cont)

- Migrant selection
  - Use any fitness-proportional selection method
  - Elitism to move the best individual to another population
  - Tournament selection
    - tournament sice of two
    - best individual will migrate
    - worst individual will be replaced by a migrant from neighboring population

Island Genetic Algorithms (cont)

- Dynamic island GAs:
  - Migration occurs at a probability
  - Destination island is probabilistically selected
  - Destination islands may use an acceptance strategy
    - An immigrant is probabilistically accepted if its fitness is better than the average fitness of the island (using, e.g. Boltzmann selection)
- Initialization of subpopulations:
  - Random initialization, where there will be overlaps in search space
  - Populations to cover different parts of the search space
  - For MOO, one population per subobjective
- Cooperative coevolutionary GA – similar to the cooperative PSO