Plan for Today

- Evolutionary Algorithms (EA)
  - Optimization Basics
  - EA Basics

---

Optimization Basics

Given:
- **objective function** $f: X \to \mathbb{R}$
- **feasible region** $X$ (= nonempty set)

**Objective:** find solution with **minimal** or **maximal** value!

**Optimization problem:**
find $x^* \in X$ such that $f(x^*) = \min \{ f(x) : x \in X \}$

**Note:**
$max \{ f(x) : x \in X \} = -min \{ -f(x) : x \in X \}$
**Optimization Basics**

1. **Local Solution**
   
   - A local solution $x^* \in X$ is a solution where:
     
     $$\forall x \in N(x^*): f(x^*) \leq f(x)$$
     
     If $x^*$ is a local solution, then $f(x^*)$ is a local optimum / minimum.

   - Neighborhood of $x^*$ is a bounded subset of $X$.

   - Example: $X = \mathbb{R}^n$, $N(x^*) = \{ x \in X : \| x - x^* \|_2 \leq \varepsilon \}$

2. **Remark**
   
   - Every global solution / optimum is also a local solution / optimum.
   - The reverse is not necessarily true.

   - Example: $f: [a,b] \rightarrow \mathbb{R}$, global solution at $x^*$

**What makes optimization difficult?**

- Local optima (is it a global optimum or not?)
- Constraints (ill-shaped feasible region)
- Non-smoothness (weak causality)
- Discontinuities (no gradients)
- Lack of knowledge about problem (black / gray box optimization)

**Evolutionary Algorithm Basics**

**Idea:**

- Using biological evolution as a metaphor and pool of inspiration.

- Interpretation: biological evolution as an iterative method of improvement.

**Representation:**

- Feasible solution $x \in X = S_1 \times \ldots \times S_n$ = chromosome of individual.
- Multiset of feasible solutions = population: multiset of individuals.

**Objective Function:**

- $f: X \rightarrow \mathbb{R}$ = fitness function.
- Often: $X = \mathbb{R}^n$, $X = \mathbb{B}^n = \{0,1\}^n$, $X = \mathbb{P}_n = \{\pi : \pi$ is permutation of $\{1,2,\ldots,n\}\}$.

**Also:**

- Combinations like $X = \mathbb{R}^n \times \mathbb{B}^p \times \mathbb{P}_q$ or non-cartesian sets.

- Structure of feasible region / search space defines representation of individual.

**When using which optimization method?**

- **Mathematical Algorithms**
  - Problem explicitly specified.
  - Problem-specific solver available.
  - Problem well understood.
  - Resources for designing algorithm affordable.
  - Solution with proven quality required.

- **Randomized Search Heuristics**
  - Problem given by black / gray box.
  - No problem-specific solver available.
  - Problem poorly understood.
  - Insufficient resources for designing algorithm.
  - Solution with satisfactory quality sufficient.

- **Evolutionary Algorithms**

$$\Rightarrow$$ don't apply EAs

$$\Rightarrow$$ EAs worth a try
Evolutionary Algorithm Basics

Specific example: (1+1)-EA in $\mathbb{R}^n$ for minimizing some $f$: $\mathbb{R}^n \to \mathbb{R}$

population size = 1, number of offspring = 1, selects best from 1+1 individuals

1. initialize $X(0) \in \mathbb{R}^n$ uniformly at random, set $t = 0$
2. evaluate $f(X(0))$
3. select parent: $Y = X(0)$
4. variation: add random vector: $Y = Y + Z$, e.g. $Z \sim \mathcal{N}(0, I_n)$
5. evaluate $f(Y)$
6. selection: if $f(Y) \leq f(X(0))$ then $X(t+1) = Y$ else $X(t+1) = X(t)$
7. if not stopping then $t = t+1$, continue at (3)

Evolutionary Algorithm Basics

Selection

(a) select parents that generate offspring → selection for reproduction
(b) select individuals that proceed to next generation → selection for survival

necessary requirements:
- selection steps must not favor worse individuals
- one selection step may be neutral (e.g. select uniformly at random)
- at least one selection step must favor better individuals

typically: selection only based on fitness values $f(x)$ of individuals
seldom: additionally based on individuals’ chromosomes $x$ (→ maintain diversity)
Evolutionary Algorithm Basics

### Selection methods

Population \(P = (x_1, x_2, ..., x_\mu)\) with \(\mu\) individuals

**Two approaches:**
1. repeatedly select individuals from population with replacement
2. rank individuals somehow and choose those with best ranks (no replacement)

**Uniform / Neutral selection**
Choose index \(i\) with probability \(1/\mu\)

**Fitness-proportional selection**
Choose index \(i\) with probability \(s_i = \frac{f(x_i)}{\sum_{x \in P} f(x)}\)

- **Problems:** \(f(x) > 0\) for all \(x \in X\) required \(\Rightarrow g(x) = \exp(f(x)) > 0\)
- But already sensitive to additive shifts \(g(x) = f(x) + c\)
- Almost deterministic if large differences, almost uniform if small differences

**Rank-proportional selection**
Order individuals according to their fitness values
Assign ranks
Fitness-proportional selection based on ranks
\(\Rightarrow\) avoids all problems of fitness-proportional selection
But: best individual has only small selection advantage (can be lost!)

**K-ary tournament selection**
Draw \(k\) individuals uniformly at random (typically with replacement) from \(P\)
Choose individual with best fitness (break ties at random)
\(\Rightarrow\) has all advantages of rank-based selection and
Probability that best individual does not survive:
\[
\left(1 - \frac{1}{\mu}\right)^k \mu \approx e^{-k}
\]

### Selection methods: Elitism

**Elitist selection:** best parent is not replaced by worse individual.

- **Intrinsic elitism:** method selects from parent and offspring, best survives with probability 1
- **Forced elitism:** if best individual has not survived then re-injection into population, i.e., replace worst selected individual by previously best parent

<table>
<thead>
<tr>
<th>Method</th>
<th>(P{ select \ best } ) from parents &amp; offspring</th>
<th>Intrinsic elitism</th>
</tr>
</thead>
<tbody>
<tr>
<td>Neutral</td>
<td>&lt; 1</td>
<td>no</td>
</tr>
<tr>
<td>Fitness proportionate</td>
<td>&lt; 1</td>
<td>no</td>
</tr>
<tr>
<td>Rank proportionate</td>
<td>&lt; 1</td>
<td>no</td>
</tr>
<tr>
<td>K-ary tournament</td>
<td>&lt; 1</td>
<td>no</td>
</tr>
<tr>
<td>((\mu + \lambda))</td>
<td>= 1</td>
<td>yes</td>
</tr>
<tr>
<td>((\mu, \lambda))</td>
<td>= 1</td>
<td>no</td>
</tr>
</tbody>
</table>

Evolutionary Algorithm Basics

### Selection methods

Population \(P = (x_1, x_2, ..., x_\mu)\) with \(\mu\) individuals

**Rank-proportional selection**
Order individuals according to their fitness values
Assign ranks
Fitness-proportional selection based on ranks
\(\Rightarrow\) avoids all problems of fitness-proportional selection
But: best individual has only small selection advantage (can be lost!)

**K-ary tournament selection**
Draw \(k\) individuals uniformly at random (typically with replacement) from \(P\)
Choose individual with best fitness (break ties at random)
\(\Rightarrow\) has all advantages of rank-based selection and
Probability that best individual does not survive:
\[
\left(1 - \frac{1}{\mu}\right)^k \mu \approx e^{-k}
\]

Evolutionary Algorithm Basics

### Selection methods without replacement

Population \(P = (x_1, x_2, ..., x_\mu)\) with \(\mu\) parents and
Population \(Q = (y_1, y_2, ..., y_\lambda)\) with \(\lambda\) offspring

- \((\mu, \lambda)\)-selection or truncation selection on offspring or comma-selection
  Rank \(\lambda\) offspring according to their fitness
  Select \(\mu\) offspring with best ranks
  \(\Rightarrow\) best individual may get lost, \(\lambda \geq \mu\) required

- \((\mu + \lambda)\)-selection or truncation selection on parents + offspring or plus-selection
  Merge \(\lambda\) offspring and \(\mu\) parents
  Rank them according to their fitness
  Select \(\mu\) individuals with best ranks
  \(\Rightarrow\) best individual survives for sure

Evolutionary Algorithm Basics

### Selection methods

Population \(P = (x_1, x_2, ..., x_\mu)\) with \(\mu\) individuals

- **Rank-proportional selection**
  Order individuals according to their fitness values
  Assign ranks
  Fitness-proportional selection based on ranks
  \(\Rightarrow\) avoids all problems of fitness-proportional selection
  But: best individual has only small selection advantage (can be lost!)

- **K-ary tournament selection**
  Draw \(k\) individuals uniformly at random (typically with replacement) from \(P\)
  Choose individual with best fitness (break ties at random)
  \(\Rightarrow\) has all advantages of rank-based selection and
  Probability that best individual does not survive:
  \[
  \left(1 - \frac{1}{\mu}\right)^k \mu \approx e^{-k}
  \]

Evolutionary Algorithm Basics

### Selection methods: Elitism

**Elitist selection:** best parent is not replaced by worse individual.

- **Intrinsic elitism:** method selects from parent and offspring, best survives with probability 1
- **Forced elitism:** if best individual has not survived then re-injection into population, i.e., replace worst selected individual by previously best parent

<table>
<thead>
<tr>
<th>Method</th>
<th>(P{ select \ best } ) from parents &amp; offspring</th>
<th>Intrinsic elitism</th>
</tr>
</thead>
<tbody>
<tr>
<td>Neutral</td>
<td>&lt; 1</td>
<td>no</td>
</tr>
<tr>
<td>Fitness proportionate</td>
<td>&lt; 1</td>
<td>no</td>
</tr>
<tr>
<td>Rank proportionate</td>
<td>&lt; 1</td>
<td>no</td>
</tr>
<tr>
<td>K-ary tournament</td>
<td>&lt; 1</td>
<td>no</td>
</tr>
<tr>
<td>((\mu + \lambda))</td>
<td>= 1</td>
<td>yes</td>
</tr>
<tr>
<td>((\mu, \lambda))</td>
<td>= 1</td>
<td>no</td>
</tr>
</tbody>
</table>

Evolutionary Algorithm Basics

### Selection methods without replacement

Population \(P = (x_1, x_2, ..., x_\mu)\) with \(\mu\) parents and
Population \(Q = (y_1, y_2, ..., y_\lambda)\) with \(\lambda\) offspring

- \((\mu, \lambda)\)-selection or truncation selection on offspring or comma-selection
  Rank \(\lambda\) offspring according to their fitness
  Select \(\mu\) offspring with best ranks
  \(\Rightarrow\) best individual may get lost, \(\lambda \geq \mu\) required

- \((\mu + \lambda)\)-selection or truncation selection on parents + offspring or plus-selection
  Merge \(\lambda\) offspring and \(\mu\) parents
  Rank them according to their fitness
  Select \(\mu\) individuals with best ranks
  \(\Rightarrow\) best individual survives for sure
Evolutionary Algorithm Basics

Variation operators: depend on representation

- mutation → alters a single individual
- recombination → creates single offspring from two or more parents

may be applied
- exclusively (either recombination or mutation) chosen in advance
- exclusively (either recombination or mutation) in probabilistic manner
- sequentially (typically, recombination before mutation); for each offspring
- sequentially (typically, recombination before mutation) with some probability

Evolutionary Algorithm Basics

Variation in \( \mathbb{B}^n \)

- Recombination (two parents)
  - a) 1-point crossover → draw cut-point \( k \in \{1, \ldots, n-1\} \) uniformly at random; choose first \( k \) bits from 1st parent, choose last \( n-k \) bits from 2nd parent
  - b) K-point crossover → draw \( K \) distinct cut-points uniformly at random; choose bits \( 1 \) to \( k_1 \) from 1st parent, choose bits \( k_1+1 \) to \( k_2 \) from 2nd parent, choose bits \( k_2+1 \) to \( k_3 \) from 1st parent, and so forth ...
  - c) uniform crossover → for each index \( i \): choose bit \( i \) with equal probability from 1st or 2nd parent

- Recombination (multiparent: \( \rho = \text{#parents} \))
  - a) diagonal crossover (\( 2 < \rho < n \)) → choose \( \rho - 1 \) distinct cut points, select chunks from diagonals

\[
\begin{align*}
\text{AAAAA} & \rightarrow \text{BBBBB} \\
\text{CCCCC} & \rightarrow \text{DDDDD}
\end{align*}
\]

\[
\begin{align*}
\text{AAAAAAAAAA} & \rightarrow \text{BBBBCCDDDD} \\
\text{BBBBBEBBBB} & \rightarrow \text{BCCCDAAAA} \\
\text{CCCCCCCCCC} & \rightarrow \text{CDDDAABBBB} \\
\text{DDDDDDDDDD} & \rightarrow \text{DAAAABCCCC}
\end{align*}
\]

- b) gene pool crossover (\( \rho > 2 \)) → for each gene: choose donating parent uniformly at random

Evolutionary Algorithm Basics

Mutation

a) local → choose index \( k \in \{1, \ldots, n\} \) uniformly at random, flip bit \( k \), i.e., \( x_k = 1 - x_k \)

b) global → for each index \( k \in \{1, \ldots, n\} \): flip bit \( k \) with probability \( p_m \in (0, 1) \)

c) “nonlocal” → choose \( K \) indices at random and flip bits with these indices

d) inversion → choose start index \( k_s \) and end index \( k_e \) at random, invert order of bits between start and end index

Evolutionary Algorithm Basics

Individuals ∈ \( \{0, 1\}^n \)

Mutation

a) local

\[
\begin{pmatrix}
1 & 0 & 1 & 0 & 1 \\
0 & 1 & 0 & 1 & 0
\end{pmatrix}
\rightarrow
\begin{pmatrix}
1 & 0 & 0 & 1 & 0 \\
1 & 1 & 1 & 1 & 1
\end{pmatrix}
\]

b) global

\[
\begin{pmatrix}
1 & 0 & 1 & 0 & 1 \\
0 & 1 & 0 & 1 & 0
\end{pmatrix}
\rightarrow
\begin{pmatrix}
1 & 0 & 0 & 1 & 0 \\
1 & 1 & 1 & 1 & 1
\end{pmatrix}
\]

c) “nonlocal”

\[
\begin{pmatrix}
1 & 0 & 1 & 0 & 1 \\
0 & 1 & 0 & 1 & 0
\end{pmatrix}
\rightarrow
\begin{pmatrix}
1 & 0 & 0 & 1 & 0 \\
1 & 1 & 1 & 1 & 1
\end{pmatrix}
\]

d) inversion

\[
\begin{pmatrix}
1 & 0 & 1 & 0 & 1 \\
0 & 1 & 0 & 1 & 0
\end{pmatrix}
\rightarrow
\begin{pmatrix}
1 & 0 & 0 & 1 & 0 \\
1 & 1 & 1 & 1 & 1
\end{pmatrix}
\]
Evolutionary Algorithm Basics

Variation in $\mathbb{P}_n$

- **Mutation**
  - **local** $\rightarrow$ 2-swap / 1-translocation
    - $5 \ 3 \ 2 \ 4 \ 1$ $\rightarrow$ $5 \ 3 \ 2 \ 4 \ 1$
    - $5 \ 4 \ 3 \ 1$ $\rightarrow$ $5 \ 2 \ 4 \ 3 \ 1$
  - **global** $\rightarrow$ draw number $K$ of 2-swaps, apply 2-swaps $K$ times
    - $K$ is positive random variable; its distribution may be uniform, binomial, geometrical, …; $E[K]$ and $V[K]$ may control mutation strength

- **Recombination (two parents)**
  - a) order-based crossover (OB)
  - b) partially mapped crossover (PMX)
  - c) cycle crossover (CX)

- **Recombination (multiparent)**
  - a) $xx$ crossover
  - b) $xx$ crossover
  - c) $xx$ crossover

Evolutionary Algorithm Basics

Variation in $\mathbb{R}_n$

- **Mutation**
  - additive: $Y = X + Z$ ($Z$: $n$-dimensional random vector)
  - **local** $\rightarrow Z$ with bounded support
    - $f_Z(x) = \frac{4}{3} (1 - x^2) \cdot 1_{[-1,1]}(x)$
  - **nonlocal** $\rightarrow Z$ with unbounded support
    - $f_Z(x) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{x^2}{2}\right)$

**Definition**

Let $f_Z: \mathbb{R}^n \rightarrow \mathbb{R}^+$ be p.d.f. of r.v. $Z$. The set $\{ x \in \mathbb{R}^n : f_Z(x) > 0 \}$ is termed the **support** of $Z$. Most frequently used!
Evolutionary Algorithm Basics

Variation in $\mathbb{R}^n$

- Recombination (two parents)
  a) all crossover variants adapted from $\mathbb{R}^n$
  b) intermediate recombination
    \[ z = \xi \cdot x + (1 - \xi) \cdot y \] with $\xi \in [0, 1]$
  c) intermediate (per dimension)
    \[ \forall i : z_i = \xi_i \cdot x_i + (1 - \xi_i) \cdot y_i \] with $\xi_i \in [0, 1]$
  d) discrete
    \[ \forall i : z_i = B_i \cdot x_i + (1 - B_i) \cdot y_i \] with $B_i \sim B(1, \frac{1}{2})$
  e) simulated binary crossover (SBX)

Variation in $\mathbb{R}^n$

- Recombination (multiparent), $\rho \geq 3$ parents
  a) intermediate recombination
    \[ z = \sum_{k=1}^{\rho} \xi^{(k)} \cdot x^{(k)}_i \] where $\sum_{k=1}^{\rho} \xi^{(k)} = 1$ and $\xi^{(k)} \geq 0$
    (all points in convex hull)
  b) intermediate (per dimension)
    \[ \forall i : z_i = \sum_{k=1}^{\rho} \xi^{(k)} \cdot x^{(k)}_i \]
    \[ \forall i : z_i \in \left[ \min_k \{x^{(k)}_i\}, \max_k \{x^{(k)}_i\} \right] \]

Theorem

Let $f : \mathbb{R}^n \rightarrow \mathbb{R}$ be a strictly quasiconvex function. If $f(x) = f(y)$ for some $x \neq y$ then every offspring generated by intermediate recombination is better than its parents.

Proof:

\[ f \text{ strictly quasiconvex} \Rightarrow f(\xi \cdot x + (1 - \xi) \cdot y) < \max\{ f(x), f(y) \} \text{ for } 0 < \xi < 1 \]

since $f(x) = f(y) \Rightarrow \max\{ f(x), f(y) \} = \min\{ f(x), f(y) \}$

\[ \Rightarrow f(\xi \cdot x + (1 - \xi) \cdot y) < \min\{ f(x), f(y) \} \text{ for } 0 < \xi < 1 \]

Theorem

Let $f : \mathbb{R}^n \rightarrow \mathbb{R}$ be a differentiable function and $f(x) < f(y)$ for some $x \neq y$. If $(y - x)' \cdot \nabla f(x) < 0$ then there is a positive probability that an offspring generated by intermediate recombination is better than both parents.

Proof:

If $d' \cdot \nabla f(x) < 0$ then $d \in \mathbb{R}^n$ is a direction of descent, i.e.

\[ \exists \delta > 0 : \forall s \in (0, \delta] : f(x + s \cdot d) < f(x). \]

Here: $d = y - x$ such that $P\{ f(\xi x + (1 - \xi) y) < f(x) \} \geq \delta > 0$. ■

sublevel set $S_{\alpha} = \{ x \in \mathbb{R}^n : f(x) < \alpha \}$
Evolutionary Algorithms: Historical Notes

Idea emerged independently several times: about late 1950s / early 1960s.
Three branches / "schools" still active today.

- **Evolutionary Programming (EP):**
  Pioneers: Lawrence Fogel, Alvin Owen, Michael Walsh (New York, USA).
  Original goal: Generate intelligent behavior through simulated evolution.
  Later (~1990s) specialized to optimization in $\mathbb{R}^n$ by David B. Fogel.

- **Genetic Algorithms (GA):**
  Pioneer: John Holland (Ann Arbor, MI, USA).
  Original goal: Analysis of adaptive behavior.
  Applied to optimization tasks by PhD students (Kenneth de Jong, 1975; et al.).

- **Evolution Strategies (ES):**
  Pioneers: Ingo Rechenberg, Hans-Paul Schwefel, Peter Bienert (Berlin, Germany).
  Original goal: Optimization of complex systems.
  Approach: Viewing variation/selection as improvement strategy. First in $\mathbb{Z}^n$, then $\mathbb{R}^n$.

"Offspring" from GA branch:

- **Genetic Programming (GP):**
  Pioneers: Michael Lynn Cramer 1985, then: John Koza (Stanford, USA).
  Original goal: Evolve programs (parse trees) that must accomplish certain task.
  Approach: GA mechanism transferred to parse trees.
  Later: Programs as successive statements $\rightarrow$ Linear GP (e.g. Wolfgang Banzhaf)

Already beginning early 1990s:
Borders between EP, GA, ES, GP begin to blur ...

$\Rightarrow$ common term **Evolutionary Algorithm** embracing all kind of approaches
$\Rightarrow$ broadly accepted name for the field: **Evolutionary Computation**

Scientific journals: **Evolutionary Computation** (MIT Press) since 1993,
*IEEE Transactions on Evolutionary Computation* since 1997,
several more specialized journals started since then.