Metaheuristic Optimization
12. Evolution Strategies

Thomas Weise · 汤卫思
tweise@hfuu.edu.cn · http://iao.hfuu.edu.cn

Hefei University, South Campus 2
Faculty of Computer Science and Technology
Institute of Applied Optimization
230601 Shushan District, Hefei, Anhui, China
Econ. & Tech. Devel. Zone, Jinxiu Dadao 99

合肥学院 南艳湖校区/南2区
计算机科学与技术系
应用优化研究所
中国 安徽省 合肥市 蜀山区 230601
经济技术开发区 锦绣大道99号
Outline

1. Population Treatment
2. Mutation
3. Self-Adaptation
4. The 1/5th Rule
5. Endogeneous Adaptation
6. Recombination
7. Parameter Reproduction
8. CMA-ES
Evolutionary Algorithm for numerical optimization
• Evolutionary Algorithm for numerical optimization
• Search space: vectors of real numbers, i.e., $\mathbb{G} = \mathbb{R}^n$
Evolutionary Algorithm for numerical optimization

Search space: vectors of real numbers, i.e., $\mathbb{G} = \mathbb{R}^n$

Developed by Rechenberg \cite{1-3} and Schwefel \cite{4-6}
Evolutionary Algorithm for numerical optimization

- Search space: vectors of real numbers, i.e., $\mathbb{G} = \mathbb{R}^n$
- Developed by Rechenberg$^{[1–3]}$ and Schwefel$^{[4–6]}$ in Dortmund (Germany) in the 1960s
Introduction

- Evolutionary Algorithm for numerical optimization
- Search space: vectors of real numbers, i.e., $\mathbb{G} = \mathbb{R}^n$
- Developed by Rechenberg [1–3] and Schwefel [4–6] in Dortmund (Germany) in the 1960s (before most of Holland’s works on Genetic Algorithms [7–10] and 10 years before Genetic Algorithms were used to solve mathematical functions by De Jong [11]... )
• Evolutionary Algorithm for numerical optimization
• Search space: vectors of real numbers, i.e., $G = \mathbb{R}^n$
• Developed by Rechenberg [1–3] and Schwefel [4–6] in Dortmund (Germany) in the 1960s (before most of Holland’s works on Genetic Algorithms [7–10] and 10 years before Genetic Algorithms were used to solve mathematical functions by De Jong [11]...)
• Different population treatments
Introduction

- Evolutionary Algorithm for numerical optimization
- Search space: vectors of real numbers, i.e., $\mathbb{G} = \mathbb{R}^n$
- Developed by Rechenberg\textsuperscript{[1–3]} and Schwefel\textsuperscript{[4–6]} in Dortmund (Germany) in the 1960s (before most of Holland’s works on Genetic Algorithms\textsuperscript{[7–10]} and 10 years before Genetic Algorithms were used to solve mathematical functions by De Jong\textsuperscript{[11]}...)
- Different population treatments
- Mutation as main search operation
Introduction

- Evolutionary Algorithm for numerical optimization
- Search space: vectors of real numbers, i.e., $\mathbb{G} = \mathbb{R}^n$
- Developed by Rechenberg [1–3] and Schwefel [4–6] in Dortmund (Germany) in the 1960s (before most of Holland’s works on Genetic Algorithms [7–10] and 10 years before Genetic Algorithms were used to solve mathematical functions by De Jong [11]…)

- Different population treatments
- Mutation as main search operation
- Idea: Self-adaptation of search – search operations automatically fine-tuned according to progress of search
Introduction

- Evolutionary Algorithm for numerical optimization
- Search space: vectors of real numbers, i.e., $\mathbb{G} = \mathbb{R}^n$
- Developed by Rechenberg $^{[1-3]}$ and Schwefel $^{[4-6]}$ in Dortmund (Germany) in the 1960s (before most of Holland’s works on Genetic Algorithms $^{[7-10]}$ and 10 years before Genetic Algorithms were used to solve mathematical functions by De Jong $^{[11]}$ . . .)
- Different population treatments
- Mutation as main search operation
- Idea: Self-adaptation of search – search operations automatically fine-tuned according to progress of search
- Endogenous and exogenous strategy parameters
• Evolution Strategies use truncation selection – we already discussed that in the GA lecture
• Evolution Strategies use truncation selection – we already discussed that in the GA lecture
• Two population size-related parameters
• Evolution Strategies use truncation selection – we already discussed that in the GA lecture
• Two population size-related parameters:
  1. $\lambda$ number of offsprings
• Evolution Strategies use truncation selection – we already discussed that in the GA lecture
• Two population size-related parameters:
  1. $\lambda$ number of offsprings
  2. $\mu$ size of the mating pool
- Evolution Strategies use truncation selection – we already discussed that in the GA lecture
- Two population size-related parameters:
  1. $\lambda$ number of offsprings
  2. $\mu$ size of the mating pool
- $(\mu + \lambda)$-algorithm
• Evolution Strategys use truncation selection – we already discussed that in the GA lecture

• Two population size-related parameters:
  1. $\lambda$ number of offsprings
  2. $\mu$ size of the mating pool

• $(\mu + \lambda)$-algorithm:
  • $\lambda$ offsprings are created from $\mu$ parents
• Evolution Strategys use truncation selection – we already discussed that in the GA lecture

• Two population size-related parameters:
  1. \( \lambda \) number of offsprings
  2. \( \mu \) size of the mating pool

• \((\mu + \lambda)\)-algorithm:
  • \( \lambda \) offsprings are created from \( \mu \) parents
  • the set of \( \lambda \) offsprings and \( \mu \) parents forms the new population of size
    \[ ps = \mu + \lambda \]
• Evolution Strategies use truncation selection – we already discussed that in the GA lecture

• Two population size-related parameters:
  1. $\lambda$ number of offsprings
  2. $\mu$ size of the mating pool

• $(\mu + \lambda)$-algorithm:
  • $\lambda$ offsprings are created from $\mu$ parents
  • the set of $\lambda$ offsprings and $\mu$ parents forms the new population of size $ps = \mu + \lambda$
  • only the $\mu$ fittest among these $ps$ individuals survive
• Evolution Strategies use truncation selection – we already discussed that in the GA lecture

• Two population size-related parameters:
  1. \( \lambda \) number of offsprings
  2. \( \mu \) size of the mating pool

• \((\mu + \lambda)\)-algorithm:
  • \( \lambda \) offsprings are created from \( \mu \) parents
  • the set of \( \lambda \) offsprings and \( \mu \) parents forms the new population of size \( ps = \mu + \lambda \)
  • only the \( \mu \) fittest among these \( ps \) individuals survive
  • parents may survive: preservative/steady-state method
• Evolution Strategies use truncation selection – we already discussed that in the GA lecture

• Two population size-related parameters:
  1. $\lambda$ number of offsprings
  2. $\mu$ size of the mating pool

• $(\mu + \lambda)$-algorithm:
  • $\lambda$ offsprings are created from $\mu$ parents
  • the set of $\lambda$ offsprings and $\mu$ parents forms the new population of size $ps = \mu + \lambda$
  • only the $\mu$ fittest among these $ps$ individuals survive
  • parents may survive: preservative/steady-state method

• $(\mu, \lambda)$-algorithm
Population Treatment

- Evolution Strategys use truncation selection – we already discussed that in the GA lecture
- Two population size-related parameters:
  1. $\lambda$ number of offsprings
  2. $\mu$ size of the mating pool
- $(\mu + \lambda)$-algorithm:
  - $\lambda$ offsprings are created from $\mu$ parents
  - the set of $\lambda$ offsprings and $\mu$ parents forms the new population of size $ps = \mu + \lambda$
  - only the $\mu$ fittest among these $ps$ individuals survive
  - parents may survive: preservative/steady-state method
- $(\mu, \lambda)$-algorithm:
  - $\lambda \geq \mu$ offspring are created from $\mu$ parents
Population Treatment

- Evolution Strategys use truncation selection – we already discussed that in the GA lecture.
- Two population size-related parameters:
  1. $\lambda$ number of offsprings
  2. $\mu$ size of the mating pool
- $(\mu + \lambda)$-algorithm:
  - $\lambda$ offsprings are created from $\mu$ parents
  - the set of $\lambda$ offsprings and $\mu$ parents forms the new population of size $ps = \mu + \lambda$
  - only the $\mu$ fittest among these $ps$ individuals survive
  - parents may survive: preservative/steady-state method
- $(\mu, \lambda)$-algorithm:
  - $\lambda \geq \mu$ offspring are created from $\mu$ parents
  - from the $\lambda$ offspring, only the $\mu$ best ones survive
• Evolution Strategies use truncation selection – we already discussed that in the GA lecture

• Two population size-related parameters:
  1. $\lambda$ number of offsprings
  2. $\mu$ size of the mating pool

• $(\mu + \lambda)$-algorithm:
  • $\lambda$ offsprings are created from $\mu$ parents
  • the set of $\lambda$ offsprings and $\mu$ parents forms the new population of size $ps = \mu + \lambda$
  • only the $\mu$ fittest among these $ps$ individuals survive
  • parents may survive: preservative/steady-state method

• $(\mu, \lambda)$-algorithm:
  • $\lambda \geq \mu$ offspring are created from $\mu$ parents
  • from the $\lambda$ offspring, only the $\mu$ best ones survive
  • all parents are discarded: extinctive/generational EA
• (µ + λ)-algorithm
• (µ, λ)-algorithm
- $(\mu + \lambda)$-algorithm
- $(\mu, \lambda)$-algorithm
- $(1 + 1)$-ES: ?
• $(\mu + \lambda)$-algorithm
• $(\mu, \lambda)$-algorithm
• $(1 + 1)$-ES: Hill Climbing
• $(\mu + \lambda)$-algorithm
• $(\mu, \lambda)$-algorithm
• $(1 + 1)$-ES: Hill Climbing
• $(1, 1)$-ES: ?
Population Treatment

- $(\mu + \lambda)$-algorithm
- $(\mu, \lambda)$-algorithm
- $(1 + 1)$-ES: Hill Climbing
- $(1, 1)$-ES: Random Walk
• $(\mu + \lambda)$-algorithm
• $(\mu, \lambda)$-algorithm
• $(1 + 1)$-ES: Hill Climbing
• $(1, 1)$-ES: Random Walk
• $(\infty \uparrow \ast)$-ES: ?
Population Treatment

- $(\mu + \lambda)$-algorithm
- $(\mu, \lambda)$-algorithm
- $(1 + 1)$-ES: Hill Climbing
- $(1, 1)$-ES: Random Walk
- $(\infty, \ast)$-ES: Random Sampling
• $(\mu + \lambda)$-algorithm
• $(\mu, \lambda)$-algorithm
• $(1 + 1)$-ES: Hill Climbing
• $(1, 1)$-ES: Random Walk
• $(\infty \dagger *)$-ES: Random Sampling
• $(\mu/\rho + \lambda)$
(μ + λ)-algorithm
(μ, λ)-algorithm
(1 + 1)-ES: Hill Climbing
(1, 1)-ES: Random Walk
(∞ + *)-ES: Random Sampling
(μ/ρ + λ):
  (μ + λ) Evolution Strategy with ρ-ary reproduction operation
• $(\mu + \lambda)$-algorithm
• $(\mu, \lambda)$-algorithm
• $(1 + 1)$-ES: Hill Climbing
• $(1, 1)$-ES: Random Walk
• $(\infty \uparrow *)$-ES: Random Sampling
• $(\mu/\rho + \lambda)$:
  • $(\mu + \lambda)$ Evolution Strategy with $\rho$-ary reproduction operation
  • $\rho =$ number of parents per offspring
• ($\mu + \lambda$)-algorithm
• ($\mu, \lambda$)-algorithm
• (1 + 1)-ES: Hill Climbing
• (1, 1)-ES: Random Walk
• ($\infty \dagger \ast$)-ES: Random Sampling
• ($\mu/\rho + \lambda$):
  • ($\mu + \lambda$) Evolution Strategy with $\rho$-ary reproduction operation
  • $\rho$ = number of parents per offspring
  • Default: $\rho = 1$ (mutation only); $\rho = 2$ (crossover)
Population Treatment

- $(\mu + \lambda)$-algorithm
- $(\mu, \lambda)$-algorithm
- $(1 + 1)$-ES: Hill Climbing
- $(1, 1)$-ES: Random Walk
- $(\infty \div *)$-ES: Random Sampling
- $(\mu/\rho + \lambda)$:
  - $(\mu + \lambda)$ Evolution Strategy with $\rho$-ary reproduction operation
  - $\rho =$ number of parents per offspring
  - Default: $\rho = 1$ (mutation only); $\rho = 2$ (crossover)
- $(\mu/\rho, \lambda)$
- $(\mu + \lambda)$-algorithm
- $(\mu, \lambda)$-algorithm
- $(1 + 1)$-ES: Hill Climbing
- $(1, 1)$-ES: Random Walk
- $(\infty \uparrow *)$-ES: Random Sampling
- $(\mu/\rho + \lambda)$:
  - $(\mu + \lambda)$ Evolution Strategy with $\rho$-ary reproduction operation
  - $\rho = \text{number of parents per offspring}$
  - Default: $\rho = 1$ (mutation only); $\rho = 2$ (crossover)
- $(\mu/\rho, \lambda)$:
  - $(\mu, \lambda)$ Evolution Strategy with $\rho$-ary reproduction operation
- $(\mu + \lambda)$-algorithm
- $(\mu, \lambda)$-algorithm
- $(1 + 1)$-ES: Hill Climbing
- $(1, 1)$-ES: Random Walk
- $(\infty + *)$-ES: Random Sampling
- $(\mu/\rho + \lambda)$:
  - $(\mu + \lambda)$ Evolution Strategy with $\rho$-ary reproduction operation
  - $\rho$ = number of parents per offspring
  - Default: $\rho = 1$ (mutation only); $\rho = 2$ (crossover)
- $(\mu/\rho, \lambda)$:
  - $(\mu, \lambda)$ Evolution Strategy with $\rho$-ary reproduction operation
  - from the $\lambda$ offspring, only the $\mu$ best ones survive
• Only the $\mu$ fittest individuals survive, in both $(\mu + \lambda)$ and $(\mu, \lambda)$ ES
• Only the $\mu$ fittest individuals survive, in both $(\mu + \lambda)$ and $(\mu, \lambda)$ ES
• Simple, deterministic selection algorithm: truncationSelection
• Only the $\mu$ fittest individuals survive, in both $(\mu + \lambda)$ and $(\mu, \lambda)$ ES
• Simple, deterministic selection algorithm: truncationSelection
• Already discussed in the GA lecture
Truncation Selection

\[
\text{matePool} \leftarrow \text{truncationSelection}(\mu, \text{pop})
\]

**Input:** pop: the list of individuals to select from (length $\lambda$ or $\mu + \lambda$)

**Input:** $\mu$: the number of individuals to be placed into the mating pool matePool

**Output:** matePool: the survivors of the truncation which now form the mating pool

begin
  sort the pop according to fitness (best first)
  return first $\mu$ individuals from pop
Listing: The Truncation Selection Algorithm

```java
public class TruncationSelection implements ISelectionAlgorithm {
    public void select(final Individual<?, ?>[] pop, final Individual<?, ?>[] mate, final Random r) {
        Arrays.sort(pop);
        System.arraycopy(pop, 0, mate, 0, mate.length);
    }
}
```
Section Outline

1. Population Treatment
2. Mutation
3. Self-Adaptation
4. The 1/5th Rule
5. Endogeneous Adaptation
6. Recombination
7. Parameter Reproduction
8. CMA-ES
• Mutation and selection are the main operations that drive the ES
• Mutation and selection are the main operations that drive the ES
• Most common ES: \((1 + 1)\), i.e., there is no recombination
• Mutation and selection are the main operations that drive the ES
• Most common ES: \((1 + 1)\), i.e., there is no recombination
• For simplicity, assume that search space \(G\) and solution space \(X\) are the same (i.e., we do not need a GPM)
Mutation and selection are the main operations that drive the ES

Most common ES: $(1 + 1)$, i.e., there is no recombination

For simplicity, assume that search space $G$ and solution space $X$ are the same (i.e., we do not need a GPM)

Let us assume the search space is the real numbers, i.e., $X = G = \mathbb{R}$. 
center: old genotype
Mutation

normal distribution 1

center: old genotype

x - σ₁ x + σ₁

x = μ

not the μ for population size but the μ denoting the expected value of a normal distribution!
Degree of change in genotype: depends on mutation strength = mutation step-width = standard deviation of mutation.

Distribution 1 has smaller standard distribution then distribution 2: expected change is smaller!

$\sigma_1 < \sigma_2$

Not the $\mu$ for population size but the $\mu$ denoting the expected value of a normal distribution!
Mutation and selection are the main operations that drive the ES

Most common ES: \((1 + 1)\), i.e., there is no recombination

For simplicity, assume that search space \(G\) and solution space \(X\) are the same (i.e., we do not need a GPM)

Let us assume the search space is the real numbers, i.e., \(X = G = \mathbb{R}\).

ESes mutate an real value \(x \in \mathbb{R}\) by replacing it with a new sample from a normal distribution with \(\mu = x\)

Parameter of the mutation operator: standard deviation \(\sigma\) of normal distribution as step length
• The objective function is usually $n$-dimensional and may have different characteristics
The objective function is usually $n$-dimensional and may have different characteristics:

- It could be nice: symmetric and axis-parallel
The objective function is usually $n$-dimensional and may have different characteristics:

- It could be nice: symmetric and axis-parallel
The objective function is usually \( n \)-dimensional and may have different characteristics:

- It could be nice: symmetric and axis-parallel
- Maybe it is ill-defined (values alone one axis have much higher impact) and axis-parallel
The objective function is usually $n$-dimensional and may have different characteristics:

- It could be nice: symmetric and axis-parallel
- Maybe it is ill-defined (values alone one axis have much higher impact) and axis-parallel
The objective function is usually $n$-dimensional and may have different characteristics:

- It could be nice: symmetric and axis-parallel
- Maybe it is ill-defined (values along one axis have much higher impact) and axis-parallel
- Maybe it is both ill-defined and not axis-parallel
The objective function is usually $n$-dimensional and may have different characteristics:

- It could be nice: symmetric and axis-parallel
- Maybe it is ill-defined (values alone one axis have much higher impact) and axis-parallel
- Maybe it is both ill-defined and not axis-parallel
The objective function is usually \( n \)-dimensional and may have different characteristics:

- It could be nice: symmetric and axis-parallel
- Maybe it is ill-defined (values alone one axis have much higher impact) and axis-parallel
- Maybe it is both ill-defined and not axis-parallel

We want to be able to create new candidate solutions in a way that can “follow” these shapes!
The objective function is usually $n$-dimensional and may have different characteristics:

- It could be nice: symmetric and axis-parallel
- Maybe it is ill-defined (values alone one axis have much higher impact) and axis-parallel
- Maybe it is both ill-defined and not axis-parallel

We want to be able to create new candidate solutions in a way that can “follow” these shapes!
Objective Function

- The objective function is usually \( n \)-dimensional and may have different characteristics:
  - It could be nice: symmetric and axis-parallel
  - Maybe it is ill-defined (values alone one axis have much higher impact) and axis-parallel
  - Maybe it is both ill-defined and not axis-parallel
- We want to be able to create new candidate solutions in a way that can “follow” these shapes!
The objective function is usually $n$-dimensional and may have different characteristics:

- It could be nice: symmetric and axis-parallel
- Maybe it is ill-defined (values alone one axis have much higher impact) and axis-parallel
- Maybe it is both ill-defined and not axis-parallel

We want to be able to create new candidate solutions in a way that can "follow" these shapes!
Let's look at the structure of a given population in a 2D-search space $\mathbb{X} \subseteq \mathbb{R}^2$.
• Let's look at the structure of a given population in a 2D-search space $X \subseteq \mathbb{R}^2$
• Assume that the individuals in the blue sphere are selected, as they are the best ones for some reason
Let's look at the structure of a given population in a 2D-search space $X \subseteq \mathbb{R}^2$.

Assume that the individuals in the blue sphere are selected, as they are the best ones for some reason.

The interesting range on the $x_1$-axis seems to be small, whereas the interesting range on the $x_2$-axis is rather large.
Let's look at the structure of a given population in a 2D-search space $X \subseteq \mathbb{R}^2$.

The interesting range on the $x_1$-axis seems to be small, whereas the interesting range on the $x_2$-axis is rather large.

Also, there is a correlation between the two dimensions: selected solutions with larger $x_1$ values also tend to have larger $x_2$ values and vice versa.
• Let’s look at the structure of a given population in a 2D-search space $X \subseteq \mathbb{R}^2$
• It would be cool if the (normal) distribution of possible offspring could also have such features
• Let's look at the structure of a given population in a 2D-search space $\mathbb{X} \subseteq \mathbb{R}^2$
• It would be cool if the (normal) distribution of possible offspring could also have such features
• Say we want to mutate the solution $\vec{x'}$ in the middle and create a *cloud* of $n$ offspring points from it...
Let's look at the structure of a given population in a 2D-search space $X \subseteq \mathbb{R}^2$.

Say we want to mutate the solution $\vec{x}'$ in the middle and create a cloud of $n$ offspring points from it.

If we mutate each dimension with the same standard deviation, we get a round cloud of points (the iso-probability lines form circles).
Let’s look at the structure of a given population in a 2D-search space \( \mathbf{X} \subseteq \mathbb{R}^2 \).

Say we want to mutate the solution \( \mathbf{x}' \) in the middle and create a *cloud* of \( n \) offspring points from it...

If we mutate each dimension with the *same* standard deviation, we get a round cloud of points (the iso-probability lines form *circles*).

Many solutions generated outside the interesting range.
• Let’s look at the structure of a given population in a 2D-search space \( X \subseteq \mathbb{R}^2 \)

• Say we want to mutate the solution \( \vec{x}' \) in the middle and create a *cloud* of \( n \) offspring points from it...

C if we mutate each dimension with a separate standard deviation (i.e., use a vector \( \sigma \)), we can get an elliptic cloud of points (the iso-probability lines form *ellipses*).
• Let's look at the structure of a given population in a 2D-search space $X \subseteq \mathbb{R}^2$

• Say we want to mutate the solution $\vec{x}'$ in the middle and create a cloud of $n$ offspring points from it...

C if we mutate each dimension with a separate standard deviation (i.e., use a vector $\sigma$), we can get an elliptic cloud of points (the iso-probability lines form ellipses)

\[ \sigma \approx \frac{\sigma_{xx_1} + \sigma_{xx_2}}{2} \]

Fewer solutions generated outside the interesting range, but still many.
Let's look at the structure of a given population in a 2D-search space $\mathbf{X} \subseteq \mathbb{R}^2$ with full covariance matrices $\mathbf{C}$, we can get a cloud of points which is shaped similar to the selection, i.e., we have the shape of a rotated ellipse.
Let’s look at the structure of a given population in a 2D-search space \( \mathbb{X} \subseteq \mathbb{R}^2 \) with full covariance matrices \( \mathbf{C} \), we can get a cloud of points which is shaped similar to the selection, i.e., we have the shape of a rotated ellipse.

Of course, \( \mathbf{D} \) is more complicated to implement than \( \mathbf{C} \), which is more complicated to implement than \( \mathbf{B} \).
Let's look at the structure of a given population in a 2D-search space $X \subseteq \mathbb{R}^2$.

Of course, D is more complicated to implement than C, which is more complicated to implement than B.

Also: for C (and even more so for D), we need more data ($\sigma$, C), and slower calculations.
• Single-valued standard deviation as step-width for mutation $p.w = \sigma$

\[ \vec{x} \leftarrow \text{mutationES}_{w=\sigma}(\sigma, \vec{x}') \]

**Input:** $\vec{x}' \in \mathbb{R}^n$: the input vector  
**Input:** $\sigma \in \mathbb{R}$: the standard deviation of the mutation  
**Data:** $i$: a counter variable  
**Output:** $\vec{x} \in \mathbb{R}^n$: the mutated version of $x'$

**begin**

\[
\text{for } i \leftarrow 0 \text{ up to } n - 1 \text{ do } \\
\quad \text{vecx}_i \leftarrow \vec{x}'_i + \sigma \{\text{Gaussian random number}\} \\
\text{return } \vec{x}
\]

**end**
- Vector of standard deviations as step-width for mutation $p.w = \bar{\sigma}$

\[ \vec{x} \leftarrow \text{mutationES}_{w=\bar{\sigma}}(\vec{\sigma}, \vec{x}') \]

**Input:** $\vec{x}' \in \mathbb{R}^n$: the input vector  
**Input:** $\vec{\sigma} \in \mathbb{R}^n$: the standard deviation vector of the mutation  
**Data:** $i$: a counter variable  
**Output:** $\vec{x} \in \mathbb{R}^n$: the mutated version of $\vec{x}'$

begin  
for $i \leftarrow 0$ up to $n - 1$ do  
  $\vec{x}_i \leftarrow \vec{x}'_i + \sigma_i$ \{Gaussian random number\} 
return $\vec{x}$
public class RnESUnaryNormal extends Rn implements IUnarySearchOperation<double[]> {
    public double[] mutate(final double[] genotype, final double[] sigma, final Random r) {
        double d;
        double[] g = genotype.clone(); // copy the original vector

        for (int i = g.length; (--i) >= 0;) {
            do {
                // create a value close to that gene by using the step length parameter
                d = (g[i] + (r.nextGaussian() * sigma[i % sigma.length]));
            } while (((d < this.min) || (d > this.max))); // make sure that value is OK

            g[i] = d; // store value into copied genotype
        }

        return g; // return the modified copy of the original genotype
    }
}
• Rotation matrix \( p.w = M \)

\[
\mathbf{x} \leftarrow \text{mutationES}_{w=M}(\mathbf{M}, \mathbf{x}')
\]

**Input:** \( \mathbf{x}' \in \mathbb{R}^n \): the input vector  
**Input:** \( \mathbf{M} \in \mathbb{R}^{n \times n} \): an (orthogonal) rotation matrix  
**Data:** \( i, j \): a counter variable  
**Data:** \( \mathbf{t} \): a temporary vector  
**Output:** \( \mathbf{x} \in \mathbb{R}^n \): the mutated version of \( \mathbf{x}' \)

begin  
for \( i \leftarrow 0 \) up to \( n - 1 \) do  
  \( \mathbf{t}_i \leftarrow \) {Gaussian random number}  
  \( \mathbf{x} \leftarrow \mathbf{x}' \)  
  // \( \mathbf{x} \leftarrow \mathbf{x} + \mathbf{M}\mathbf{t} \)  
for \( i \leftarrow 0 \) up to \( n - 1 \) do  
  for \( j \leftarrow 0 \) up to \( n - 1 \) do  
    \( \mathbf{x}_i \leftarrow \mathbf{x}_i + \mathbf{M}_{i,j} \cdot \mathbf{t}_j \)  
return \( \mathbf{x} \)

• \( \mathbf{M} \) does not directly represent a standard deviation, but can be computed from the covariance matrix\(^1\) of an \( n \)-dimensional normal distribution

---

\(^1\)\( \mathbf{M} \) if measured, \( \mathbf{M} \) if theoretical
• Rotation matrix $p.w = M$

$\vec{x} \leftarrow \text{mutationES}_{w=M}(\vec{x}', \vec{x}')$

**Input:**  
- $\vec{x}' \in \mathbb{R}^n$: the input vector  
- $M \in \mathbb{R}^{n \times n}$: an (orthogonal) rotation matrix

**Data:**  
- $i, j$: a counter variable  
- $\vec{t}$: a temporary vector

**Output:**  
- $\vec{x} \in \mathbb{R}^n$: the mutated version of $\vec{x}'$

begin
  for $i \leftarrow 0$ up to $n - 1$ do
    $\vec{t}_i \leftarrow \{$Gaussian random number$\}$
  end

  $\vec{x} \leftarrow \vec{x}'$
  // $\vec{x} \leftarrow \vec{x} + M\vec{t}'$

  for $i \leftarrow 0$ up to $n - 1$ do
    for $j \leftarrow 0$ up to $n - 1$ do
      $\vec{x}_i \leftarrow \vec{x}_i + M_{i,j} \ast \vec{t}_j$
    end
  end

return $\vec{x}$

• $M$ basically is the Eigen-Vector matrix of the covariance matrix $M$. 
• Rotation matrix $p.w = M$

$$\vec{x} \leftarrow \text{mutationES}_{w=M}(M, \vec{x}')$$

**Input:** $\vec{x}' \in \mathbb{R}^n$: the input vector

**Input:** $M \in \mathbb{R}^{n \times n}$: an (orthogonal) rotation matrix

**Data:** $i, j$: a counter variable

**Data:** $\vec{t}$: a temporary vector

**Output:** $\vec{x} \in \mathbb{R}^n$: the mutated version of $\vec{x}'$

begin
  for $i \leftarrow 0$ up to $n - 1$ do
    $\vec{t}_i \leftarrow \{\text{Gaussian random number}\}$
    $\vec{x} \leftarrow \vec{x}'$
  
  // $\vec{x} \leftarrow \vec{x} + M\vec{t}$
  for $i \leftarrow 0$ up to $n - 1$ do
    for $j \leftarrow 0$ up to $n - 1$ do
      $\vec{x}_i \leftarrow \vec{x}_i + M_{i,j} \times \vec{t}_j$
  
  return $\vec{x}$

• $M$ basically is the Eigen-Vector matrix of the covariance matrix $M$.

• More information on sampling multi-dimensional normal distributions based on covariance/Eigen-Vector matrices: $^{[12, 13]}$
1. Population Treatment
2. Mutation
3. Self-Adaptation
4. The 1/5th Rule
5. Endogeneous Adaptation
6. Recombination
7. Parameter Reproduction
8. CMA-ES
• Evolution Strategies are self-adaptive: the parameter (step width) of the mutation operator changes over time
• Evolution Strategies are self-adaptive: the parameter (step width) of the mutation operator changes over time
• Mutation step size – remember the simple hill climbing for real-valued optimization:
• Evolution Strategies are self-adaptive: the parameter (step width) of the mutation operator changes over time

• Mutation step size – remember the simple hill climbing for real-valued optimization:
  • Large step size: high initial speed of improvement, later slow improvement
• Evolution Strategies are self-adaptive: the parameter (step width) of the mutation operator changes over time

• Mutation step size – remember the simple hill climbing for real-valued optimization:
  • **Large step size**: high initial speed of improvement, later slow improvement
  • **Small step size**: initially low speed, high speed for some time, then lower speed again
Self-Adaptation

• Evolution Strategies are self-adaptive: the parameter (step width) of the mutation operator changes over time.

• Mutation step size – remember the simple hill climbing for real-valued optimization:
  • **Large step size**: high initial speed of improvement, later slow improvement
  • **Small step size**: initially low speed, high speed for some time, then lower speed again
  • This is the exploration versus exploitation dilemma we talked about in Lesson 11: *Difficulties in Optimization*
Evolution Strategies are self-adaptive: the parameter (step width) of the mutation operator changes over time.

- **Mutation step size** – remember the simple hill climbing for real-valued optimization:
  - **Large step size**: high initial speed of improvement, later slow improvement
  - **Small step size**: initially low speed, high speed for some time, then lower speed again

This is the exploration versus exploitation dilemma we talked about in Lesson 11: *Difficulties in Optimization*

- Instead of choosing a fixed step-size with its drawbacks...
Self-Adaptation

- Evolution Strategies are self-adaptive: the parameter (step width) of the mutation operator changes over time.
- Mutation step size – remember the simple hill climbing for real-valued optimization:
  - **Large step size**: high initial speed of improvement, later slow improvement.
  - **Small step size**: initially low speed, high speed for some time, then lower speed again.
  - This is the exploration versus exploitation dilemma we talked about in Lesson 11: *Difficulties in Optimization*.
- Instead of choosing a fixed step-size with its drawbacks... Let the optimization algorithm adapt the step size over time.
Self-Adaptation

- Evolution Strategies are self-adaptive: the parameter (step width) of the mutation operator changes over time
- Mutation step size – remember the simple hill climbing for real-valued optimization:
  - Large step size: high initial speed of improvement, later slow improvement
  - Small step size: initially low speed, high speed for some time, then lower speed again
  - This is the exploration versus exploitation dilemma we talked about in Lesson 11: *Difficulties in Optimization*

- Instead of choosing a fixed step-size with its drawbacks... Let the optimization algorithm adapt the step size over time
- Can either be endogeneous (encoded in individuals and evolve with them)
Evolution Strategies are self-adaptive: the parameter (step width) of the mutation operator changes over time.

Mutation step size – remember the simple hill climbing for real-valued optimization:

- **Large step size**: high initial speed of improvement, later slow improvement
- **Small step size**: initially low speed, high speed for some time, then lower speed again
- This is the exploration versus exploitation dilemma we talked about in Lesson 11: *Difficulties in Optimization*

Instead of choosing a fixed step-size with its drawbacks... Let the optimization algorithm adapt the step size over time.

Can either be **endogeneous** (encoded in individuals and evolve with them) or **exogenous** (maintained for whole population)
Section Outline

1. Population Treatment
2. Mutation
3. Self-Adaptation
4. The 1/5th Rule
5. Endogeneous Adaptation
6. Recombination
7. Parameter Reproduction
8. CMA-ES
• Assume case B, a single standard deviation as step-width for the mutation normal distributed
• Assume case B, a single standard deviation as step-width for the mutation normal distributed
• Idea: Adapt this step-width according to success of search \[2, 14\]
- Assume case B, a single standard deviation as step-width for the mutation normal distributed
- Idea: Adapt this step-width according to success of search [2, 14]
- In basic algorithm: (1 + 1)-ES
• Assume case B, a single standard deviation as step-width for the mutation normal distributed
• Idea: Adapt this step-width according to success of search \[2, 14\]
• In basic algorithm: \((1 + 1)\)-ES
• Two key parameters are measured:
Assume case B, a single standard deviation as step-width for the mutation normal distributed

Idea: Adapt this step-width according to success of search \cite{2, 14}

In basic algorithm: \((1 + 1)\)-ES

Two key parameters are measured:

1. success probability \(P(S)\) of the mutation operation
• Assume case B, a single standard deviation as step-width for the mutation normal distributed

• Idea: Adapt this step-width according to success of search [2, 14]

• In basic algorithm: (1 + 1)-ES

• Two key parameters are measured:
  1. success probability $P(S)$ of the mutation operation
  2. progress rate $\varphi$, i.e., the expected distance gain towards the optimum
• Assume case B, a single standard deviation as step-width for the mutation normal distributed
• Idea: Adapt this step-width according to success of search \cite{2, 14}
• In basic algorithm: (1 + 1)-ES
• Two key parameters are measured:
  1. success probability $P(S)$ of the mutation operation
  2. progress rate $\varphi$, i.e., the expected distance gain towards the optimum
• Example: Sphere function

$$f(\vec{x}') = \sum_{i=1}^{n} \vec{x}_i^2 \text{ with } G = X \subseteq \mathbb{R}^n$$ (1)
Self-Adaptation on Sphere Function
Self-Adaptation on Sphere Function
Self-Adaptation on Sphere Function
• For very small standard deviations $\sigma$: 
• For very small standard deviations $\sigma$:

\[
\lim_{\sigma \to 0} P(S) = 0.5 \quad (2)
\]

\[
\lim_{\sigma \to 0} \varphi = 0 \quad (3)
\]
• For very small standard deviations $\sigma$:

\[
\lim_{\sigma \to 0} P(S) = 0.5 \quad (2)
\]
\[
\lim_{\sigma \to 0} \varphi = 0 \quad (3)
\]

• For very large standard deviations $\sigma$:
• For very small standard deviations $\sigma$:

$$\lim_{\sigma \to 0} P(S') = 0.5$$  \hspace{1cm} (2)

$$\lim_{\sigma \to 0} \varphi = 0$$  \hspace{1cm} (3)

• For very large standard deviations $\sigma$:

$$\lim_{\sigma \to +\infty} P(S') = 0$$  \hspace{1cm} (4)

$$\lim_{\sigma \to +\infty} \varphi = 0$$  \hspace{1cm} (5)
Self-Adaptation on Sphere Function

- For very small standard deviations $\sigma$:

$$\lim_{\sigma \to 0} P(S') = 0.5$$  \hspace{1cm} (2)

$$\lim_{\sigma \to 0} \varphi = 0$$  \hspace{1cm} (3)

- For very large standard deviations $\sigma$:

$$\lim_{\sigma \to +\infty} P(S') = 0$$  \hspace{1cm} (4)

$$\lim_{\sigma \to +\infty} \varphi = 0$$  \hspace{1cm} (5)

- In between the two extreme cases (for $0 < \sigma < +\infty$) lies an area where $\varphi > 0$ and $0 < P(S') < 0.5$. 

**Definition (1/5th Rule)**

In order to obtain nearly optimal (local) performance of the \((1 + 1)\)-ES with isotropic mutation, tune the mutation strength \(\sigma\) in such a way that the success rate \(P(S)\) (estimated based on past operator applications) is about \(1/5\)\(^{[15]}\).
In order to obtain nearly optimal (local) performance of the \((1 + 1)\)-ES with isotropic mutation, tune the mutation strength \(\sigma\) in such a way that the success rate \(P(S)\) (estimated based on past operator applications) is about \(1/5\) \textsuperscript{[15]}. 

- \(P(S)\) is monotonously decreasing with rising \(\sigma\) from \(\lim_{\sigma \to 0} P(S) = 0.5\) to \(\lim_{\sigma \to +\infty} P(S) = 0\)
In order to obtain nearly optimal (local) performance of the \((1 + 1)\)-ES with isotropic mutation, tune the mutation strength \(\sigma\) in such a way that the success rate \(P(S)\) (estimated based on past operator applications) is about \(1/5^{[15]}\).

- \(P(S)\) is monotonously decreasing with rising \(\sigma\) from \(\lim_{\sigma \to 0} P(S) = 0.5\) to \(\lim_{\sigma \to +\infty} P(S) = 0\)
  - if success probability \(P(S) > 0.2\), increase the mutation strength \(\sigma \Rightarrow\) faster progress towards optimum
In order to obtain nearly optimal (local) performance of the (1 + 1)-ES with isotropic mutation, tune the mutation strength \( \sigma \) in such a way that the success rate \( P(S) \) (estimated based on past operator applications) is about \( \frac{1}{5} \) \[^{[15]}\].

- \( P(S) \) is monotonously decreasing with rising \( \sigma \) from \( \lim_{\sigma \to 0} P(S) = 0.5 \) to \( \lim_{\sigma \to +\infty} P(S) = 0 \)
  1. if success probability \( P(S) > 0.2 \), increase the mutation strength \( \sigma \) \( \Rightarrow \) faster progress towards optimum
  2. if the fraction of accepted mutations falls below 0.2, step size is too large and \( \sigma \) must be reduced
(1 + 1)-ES = self-adaptive hill climber
1/5th Rule-based $(1 + 1)$ Evolution Strategy

$p_{\text{best}} \leftarrow (1+1) \ ES_{\frac{1}{5}}(f, L, a, \sigma_0)$

begin
\[
\begin{align*}
t &\leftarrow 1 \\
 s &\leftarrow 0 \\
 \sigma &\leftarrow \sigma_0 \\
p_{\text{best.}g} &\leftarrow \text{create starting point} \\
p_{\text{best.x}} &\leftarrow \text{gpm}(p_{\text{best.g}}) \\
p_{\text{best.y}} &\leftarrow f(p_{\text{best.x}}) \\
p_{\text{new}} &\leftarrow p_{\text{best}} \\
\text{while} & \neg \text{shouldTerminate} \text{ do} \\
\quad p_{\text{new.x}} &\leftarrow \text{gpm}(p_{\text{new.g}}) \\
\quad p_{\text{new.y}} &\leftarrow f(p_{\text{new.x}}) \\
\quad \text{if} & \ p_{\text{new.y}} < p_{\text{best.y}} \text{ then} \\
\quad & \quad p_{\text{best}} \leftarrow p_{\text{new}} \\
\quad & \quad s \leftarrow s + 1 \\
\quad \text{if} & \ (t \mod L) = 0 \text{ then} \\
\quad & \quad P(S) \leftarrow \frac{s}{L} \\
\quad & \quad \text{if} \ P(S) < 0.2 \text{ then} \sigma \leftarrow \sigma \ast a \\
\quad & \quad \text{else if} \ P(S) > 0.2 \text{ then} \sigma \leftarrow \sigma / a \\
\quad & \quad s \leftarrow 0 \\
\quad p_{\text{new.g}} &\leftarrow \text{mutationES}_{\sigma}(\sigma, p_{\text{new}}) \\
\quad t &\leftarrow t + 1
\end{align*}
\]
return $p_{\text{best}}$

- $(1 + 1)$-ES = self-adaptive hill climber
- Initialize iteration counter $t$, success counter $s$, and mutation strength $\sigma$
\(p_{\text{best}} \leftarrow (1+1) \text{ ES}_\frac{1}{5}(f, L, a, \sigma_0)\)

begin
\[
\begin{align*}
t &\leftarrow 1 \\
s &\leftarrow 0 \\
\sigma &\leftarrow \sigma_0 \\
p_{\text{best}}.g &\leftarrow \text{create starting point} \\
p_{\text{best}}.x &\leftarrow \text{gpm}(p_{\text{best}}.g) \\
p_{\text{best}}.y &\leftarrow f(p_{\text{best}}.x) \\
p_{\text{new}} &\leftarrow p_{\text{best}} \\
\text{while} \ \neg \text{shouldTerminate} \ \text{do} & \\
\quad p_{\text{new}}.x &\leftarrow \text{gpm}(p_{\text{new}}.g) \\
\quad p_{\text{new}}.y &\leftarrow f(p_{\text{new}}.x) \\
\quad \text{if} \ p_{\text{new}}.y < p_{\text{best}}.y \ \text{then} & \\
\quad \quad p_{\text{best}} &\leftarrow p_{\text{new}} \\
\quad \quad s &\leftarrow s + 1 \\
\quad \text{if} \ (t \mod L) = 0 \ \text{then} & \\
\quad \quad P(S) &\leftarrow \frac{s}{L} \\
\quad \quad \text{if} \ P(S) < 0.2 \ \text{then} \ \sigma &\leftarrow \sigma \ast a \\
\quad \quad \text{else if} \ P(S) > 0.2 \ \text{then} \ \sigma &\leftarrow \sigma / a \\
\quad \quad s &\leftarrow 0 \\
\quad p_{\text{new}}.g &\leftarrow \text{mutationES}_{\sigma}(p_{\text{new}}) \\
\quad t &\leftarrow t + 1 \\
\end{align*}
\]

return \(p_{\text{best}}\)

- \((1 + 1)\)-ES = self-adaptive hill climber
- Create initial point in search space \(p_{\text{best}}.g\), map it to candidate solution \(p_{\text{best}}.x\), compute its objective value \(p_{\text{best}}.y = f(p_{\text{best}}.x)\), and store it in “current” individual \(p_{\text{new}}\)
1/5th Rule-based \((1 + 1)\) Evolution Strategy

\[ p_{\text{best}} \leftarrow (1+1) \text{ ES}_{\frac{1}{5}}(f, L, a, \sigma_0) \]

\begin{algorithm}
\begin{algorithmic}
\State \texttt{begin}
\State \hspace{1em} t \leftarrow 1
\State \hspace{1em} s \leftarrow 0
\State \hspace{1em} \sigma \leftarrow \sigma_0
\State \hspace{1em} p_{\text{best}.g} \leftarrow \text{create starting point}
\State \hspace{1em} p_{\text{best}.x} \leftarrow \text{gpm}(p_{\text{best}.g})
\State \hspace{1em} p_{\text{best}.y} \leftarrow f(p_{\text{best}.x})
\State \hspace{1em} p_{\text{new}} \leftarrow p_{\text{best}}
\While{\neg \text{shouldTerminate}}
\State \hspace{2em} p_{\text{new}.x} \leftarrow \text{gpm}(p_{\text{new}.g})
\State \hspace{2em} p_{\text{new}.y} \leftarrow f(p_{\text{new}.x})
\If{p_{\text{new}.y} < p_{\text{best}.y}}
\State \hspace{3em} p_{\text{best}} \leftarrow p_{\text{new}}
\State \hspace{3em} s \leftarrow s + 1
\EndIf
\If{(t \mod L) = 0}
\State \hspace{2em} P(S) \leftarrow \frac{s}{L}
\If{P(S) < 0.2}
\State \hspace{3em} \sigma \leftarrow \sigma \times a
\ElseIf{P(S) > 0.2}
\State \hspace{3em} \sigma \leftarrow \sigma / a
\EndIf
\endIf
\State \hspace{2em} s \leftarrow 0
\State \hspace{2em} p_{\text{new}.g} \leftarrow \text{mutationES}_\sigma(\sigma, p_{\text{new}})
\State \hspace{2em} t \leftarrow t + 1
\EndWhile
\Return p_{\text{best}}
\end{algorithmic}
\texttt{end}
\end{algorithm}

- \((1 + 1)\)-ES = self-adaptive hill climber
- map point in search space \(p_{\text{new}.g}\) of current individual to candidate solution \(p_{\text{new}.x}\) and compute its objective value \(p_{\text{new}.y} = f(p_{\text{new}.x})\)
$p_{\text{best}} \leftarrow (1+1)\ ES_{\frac{1}{5}}(f, L, a, \sigma_0)$

begin
  $t \leftarrow 1$
  $s \leftarrow 0$
  $\sigma \leftarrow \sigma_0$
  $p_{\text{best}}.g \leftarrow \text{create starting point}$
  $p_{\text{best}}.x \leftarrow \text{gpm}(p_{\text{best}}.g)$
  $p_{\text{best}}.y \leftarrow f(p_{\text{best}}.x)$
  $p_{\text{new}} \leftarrow p_{\text{best}}$
while $\neg \text{shouldTerminate}$ do
  $p_{\text{new}}.x \leftarrow \text{gpm}(p_{\text{new}}.g)$
  $p_{\text{new}}.y \leftarrow f(p_{\text{new}}.x)$
  if $p_{\text{new}}.y < p_{\text{best}}.y$ then
    $p_{\text{best}} \leftarrow p_{\text{new}}$
    $s \leftarrow s + 1$
  if $(t \mod L) = 0$ then
    $P(S) \leftarrow \frac{s}{L}$
    if $P(S) < 0.2$ then $\sigma \leftarrow \sigma \ast a$
  else if $P(S) > 0.2$ then $\sigma \leftarrow \sigma / a$
  $s \leftarrow 0$
  $p_{\text{new}}.g \leftarrow \text{mutationES}_\sigma(\sigma, p_{\text{new}})$
  $t \leftarrow t + 1$
return $p_{\text{best}}$

- $(1+1)$-ES = self-adaptive hill climber
- If current individual $p_{\text{new}}$ is better then best known individual $p_{\text{best}}$. 

Metaheuristic Optimization Thomas Weise 25/51
\( p_{\text{best}} \leftarrow (1+1) \text{ ES}_{\frac{1}{5}} (f, L, a, \sigma_0) \)

begin
\hspace{1em} t \leftarrow 1
\hspace{1em} s \leftarrow 0
\hspace{1em} \sigma \leftarrow \sigma_0
\hspace{1em} p_{\text{best}.g} \leftarrow \text{create starting point}
\hspace{1em} p_{\text{best}.x} \leftarrow \text{gpm}(p_{\text{best}.g})
\hspace{1em} p_{\text{best}.y} \leftarrow f(p_{\text{best}.x})
\hspace{1em} p_{\text{new}} \leftarrow p_{\text{best}}
while \neg \text{shouldTerminate} do
\hspace{2em} p_{\text{new}.x} \leftarrow \text{gpm}(p_{\text{new}.g})
\hspace{2em} p_{\text{new}.y} \leftarrow f(p_{\text{new}.x})
\hspace{2em} \text{if } p_{\text{new}.y} < p_{\text{best}.y} \text{ then}
\hspace{3em} p_{\text{best}} \leftarrow p_{\text{new}}
\hspace{3em} s \leftarrow s + 1
\hspace{2em} \text{if } (t \bmod L) = 0 \text{ then}
\hspace{3em} P(S) \leftarrow \frac{s}{L}
\hspace{4em} \text{if } P(S) < 0.2 \text{ then } \sigma \leftarrow \sigma \ast a
\hspace{4em} \text{else if } P(S) > 0.2 \text{ then } \sigma \leftarrow \sigma / a
\hspace{3em} \text{if } P(S) > 0.2 \text{ then } \sigma \leftarrow \sigma / a
\hspace{2em} s \leftarrow 0
\hspace{2em} p_{\text{new}.g} \leftarrow \text{mutationES}_{\sigma}(\sigma, p_{\text{new}})
\hspace{2em} t \leftarrow t + 1
return p_{\text{best}}

- \( (1+1) \text{-ES} = \text{self-adaptive hill climber} \)
- If current individual \( p_{\text{new}} \) is better then best known individual \( p_{\text{best}} \),
- store it in \( p_{\text{best}} \) and
1/5th Rule-based \((1 + 1)\) Evolution Strategy

\[ p_{\text{best}} \leftarrow (1+1) \text{ES}_{\frac{1}{5}}(f, L, a, \sigma_0) \]

\textbf{begin}
\[ t \leftarrow 1 \]
\[ s \leftarrow 0 \]
\[ \sigma \leftarrow \sigma_0 \]
\[ p_{\text{best}}.g \leftarrow \text{create starting point} \]
\[ p_{\text{best}}.x \leftarrow \text{gpm}(p_{\text{best}}.g) \]
\[ p_{\text{best}}.y \leftarrow f(p_{\text{best}}.x) \]
\[ p_{\text{new}} \leftarrow p_{\text{best}} \]
\textbf{while} \( \neg \text{shouldTerminate} \) \textbf{do}
\[ p_{\text{new}}.x \leftarrow \text{gpm}(p_{\text{new}}.g) \]
\[ p_{\text{new}}.y \leftarrow f(p_{\text{new}}.x) \]
\[ \textbf{if} \ p_{\text{new}}.y < p_{\text{best}}.y \ \textbf{then} \]
\[ \quad p_{\text{best}} \leftarrow p_{\text{new}} \]
\[ \quad s \leftarrow s + 1 \]
\[ \textbf{if} \ (t \mod L) = 0 \ \textbf{then} \]
\[ \quad P(S) \leftarrow \frac{s}{L} \]
\[ \quad \textbf{if} \ P(S) < 0.2 \ \textbf{then} \ \sigma \leftarrow \sigma \ast a \]
\[ \quad \textbf{else if} \ P(S) > 0.2 \ \textbf{then} \ \sigma \leftarrow \sigma / a \]
\[ \quad s \leftarrow 0 \]
\[ p_{\text{new}}.g \leftarrow \text{mutationES}_\sigma(\sigma, p_{\text{new}}) \]
\[ t \leftarrow t + 1 \]
\textbf{return} \( p_{\text{best}} \)

- \((1 + 1)\)-ES = self-adaptive hill climber
- If current individual \( p_{\text{new}} \) is better then best known individual \( p_{\text{best}} \), store it in \( p_{\text{best}} \) and 
- increase success counter \( s \) by one.
### 1/5th Rule-based \((1 + 1)\) Evolution Strategy

\[
p_{\text{best}} \leftarrow (1+1) \text{ES} \frac{1}{5}(f, L, a, \sigma_0)
\]

**Metaheuristic Optimization**

- **\((1 + 1)\)-ES** = self-adaptive hill climber
- After every \(L\) iterations, it is time for self-adaptation.
$p_{\text{best}} \leftarrow (1+1)\ ES_{\frac{1}{5}}(f, L, a, \sigma_0)$

begin
\begin{align*}
t &\leftarrow 1 \\
s &\leftarrow 0 \\
\sigma &\leftarrow \sigma_0 \\
p_{\text{best}}.g &\leftarrow \text{create starting point} \\
p_{\text{best}}.x &\leftarrow \text{gpm}(p_{\text{best}}.g) \\
p_{\text{best}}.y &\leftarrow f(p_{\text{best}}.x) \\
p_{\text{new}} &\leftarrow p_{\text{best}} \\
\text{while } \neg \text{shouldTerminate} \text{ do} \\
p_{\text{new}}.x &\leftarrow \text{gpm}(p_{\text{new}}.g) \\
p_{\text{new}}.y &\leftarrow f(p_{\text{new}}.x) \\
\text{if } p_{\text{new}}.y < p_{\text{best}}.y \text{ then} \\
\quad p_{\text{best}} &\leftarrow p_{\text{new}} \\
\quad s &\leftarrow s + 1 \\
\text{if } (t \mod L) = 0 \text{ then} \\
\quad P(S) &\leftarrow \frac{s}{L} \\
\quad \text{if } P(S) < 0.2 \text{ then } \sigma &\leftarrow \sigma \ast a \\
\quad \text{else if } P(S) > 0.2 \text{ then } \sigma &\leftarrow \sigma / a \\
\quad s &\leftarrow 0 \\
p_{\text{new}}.g &\leftarrow \text{mutationES}_\sigma(\sigma, p_{\text{new}}) \\
t &\leftarrow t + 1 \\
\text{return } p_{\text{best}}
\end{align*}

- $(1 + 1)$-ES = self-adaptive hill climber
- After every $L$ iterations, it is time for self-adaptation.
- Compute the achieved success rate $P(S)$ during the last $L$ steps.
1/5th Rule-based \((1 + 1)\) Evolution Strategy

\[
p_{\text{best}} \leftarrow (1+1) \text{ ES}_{\frac{1}{5}}(f, L, a, \sigma_0)
\]

\begin{algorithm}
\begin{algorithmic}
\STATE \begin{align*}
& t \leftarrow 1 \\
& s \leftarrow 0 \\
& \sigma \leftarrow \sigma_0 \\
& p_{\text{best}}.g \leftarrow \text{create starting point} \\
& p_{\text{best}}.x \leftarrow \text{gpm}(p_{\text{best}}.g) \\
& p_{\text{best}}.y \leftarrow f(p_{\text{best}}.x) \\
& p_{\text{new}} \leftarrow p_{\text{best}} \\
\end{align*}
\STATE \WHILE {\neg \text{shouldTerminate}}
\STATE \begin{align*}
& p_{\text{new}}.x \leftarrow \text{gpm}(p_{\text{new}}.g) \\
& p_{\text{new}}.y \leftarrow f(p_{\text{new}}.x) \\
& \textbf{if} \; p_{\text{new}}.y < p_{\text{best}}.y \; \textbf{then} \\
& \begin{align*}
& p_{\text{best}} \leftarrow p_{\text{new}} \\
& s \leftarrow s + 1 \\
\end{align*}
\STATE \textbf{if} \; (t \mod L) = 0 \; \textbf{then} \\
\STATE \begin{align*}
& P(S) \leftarrow \frac{s}{L} \\
& \textbf{if} \; P(S) < 0.2 \; \textbf{then} \; \sigma \leftarrow \sigma \times a \\
& \textbf{else if} \; P(S) > 0.2 \; \textbf{then} \; \sigma \leftarrow \sigma / a \\
& \end{align*}
\STATE \begin{align*}
& s \leftarrow 0 \\
& p_{\text{new}}.g \leftarrow \text{mutationES}_\sigma(\sigma, p_{\text{new}}) \\
& t \leftarrow t + 1 \\
\end{align*}
\STATE \textbf{return} \; p_{\text{best}}
\end{algorithmic}
\end{algorithm}

- \((1 + 1)\)-ES = self-adaptive hill climber
- After every \(L\) iterations, it is time for self-adaptation.
- Compute the achieved success rate \(P(S)\) during the last \(L\) steps.
- if less than 0.2...
\( p_{\text{best}} \leftarrow \text{(1+1) ES}_{\frac{1}{5}}(f, L, a, \sigma_0) \)

\begin{algorithm}
\begin{algorithmic}
\State \( t \leftarrow 1 \)
\State \( s \leftarrow 0 \)
\State \( \sigma \leftarrow \sigma_0 \)
\State \( p_{\text{best}}.g \leftarrow \text{create starting point} \)
\State \( p_{\text{best}}.x \leftarrow \text{gpm}(p_{\text{best}}.g) \)
\State \( p_{\text{best}}.y \leftarrow f(p_{\text{best}}.x) \)
\State \( p_{\text{new}} \leftarrow p_{\text{best}} \)
\While{\neg \text{shouldTerminate}}
\State \( p_{\text{new}}.x \leftarrow \text{gpm}(p_{\text{new}}.g) \)
\State \( p_{\text{new}}.y \leftarrow f(p_{\text{new}}.x) \)
\If{\( p_{\text{new}}.y < p_{\text{best}}.y \)}
\State \( p_{\text{best}} \leftarrow p_{\text{new}} \)
\State \( s \leftarrow s + 1 \)
\EndIf
\If{(\( t \mod L \) = 0)}
\State \( P(S) \leftarrow \frac{s}{L} \)
\If{\( P(S) < 0.2 \)}
\State \( \sigma \leftarrow \sigma * a \)
\ElseIf{\( P(S) > 0.2 \)}
\State \( \sigma \leftarrow \sigma / a \)
\EndIf
\State \( s \leftarrow 0 \)
\State \( p_{\text{new}}.g \leftarrow \text{mutationES}_\sigma(\sigma, p_{\text{new}}) \)
\State \( t \leftarrow t + 1 \)
\EndIf
\EndWhile
\State \text{return} \( p_{\text{best}} \)
\end{algorithmic}
\end{algorithm}

- \((1 + 1)\text{-ES} = \text{self-adaptive hill climber}\)
- After every \( L \) iterations, it is time for self-adaptation.
- Compute the achieved success rate \( P(S) \) during the last \( L \) steps.
- if less than 0.2, decrease \( \sigma \) by multiplying it with constant \( a \in [0, 1] \).
1/5th Rule-based \((1 + 1)\) Evolution Strategy

\[ \text{\(p_{\text{best}} \leftarrow (1+1)\ \text{ES}_{\frac{1}{5}}(f, L, a, \sigma_0)\)} \]

\begin{align*}
&\text{begin} \\
&t \leftarrow 1 \\
&s \leftarrow 0 \\
&\sigma \leftarrow \sigma_0 \\
&p_{\text{best}}.g \leftarrow \text{create starting point} \\
&p_{\text{best}}.x \leftarrow \text{gpm}(p_{\text{best}}.g) \\
&p_{\text{best}}.y \leftarrow f(p_{\text{best}}.x) \\
&p_{\text{new}} \leftarrow p_{\text{best}} \\
\text{while} \neg \text{shouldTerminate} \text{ do} \\
&\quad p_{\text{new}}.x \leftarrow \text{gpm}(p_{\text{new}}.g) \\
&\quad p_{\text{new}}.y \leftarrow f(p_{\text{new}}.x) \\
&\quad \text{if } p_{\text{new}}.y < p_{\text{best}}.y \text{ then} \\
&\quad\quad p_{\text{best}} \leftarrow p_{\text{new}} \\
&\quad\quad s \leftarrow s + 1 \\
&\quad \text{if } (t \mod L) = 0 \text{ then} \\
&\quad\quad P(S) \leftarrow \frac{s}{L} \\
&\quad\quad \text{if } P(S) < 0.2 \text{ then } \sigma \leftarrow \sigma \ast a \\
&\quad\quad \text{else if } P(S) > 0.2 \text{ then } \sigma \leftarrow \sigma / a \\
&\quad\quad s \leftarrow 0 \\
&\quad p_{\text{new}}.g \leftarrow \text{mutationES}_\sigma(\sigma, p_{\text{new}}) \\
&t \leftarrow t + 1 \\
\text{return } p_{\text{best}}
\end{align*}

- \((1 + 1)\)-ES = self-adaptive hill climber
- After every \(L\) iterations, it is time for self-adaptation.
- Compute the achieved success rate \(P(S)\) during the last \(L\) steps.
- if less than 0.2, decrease \(\sigma\) by multiplying it with constant \(a \in [0, 1]\).
- if larger than 0.2...
1/5th Rule-based \( (1 + 1) \) Evolution Strategy

\[ p_{\text{best}} \leftarrow (1+1) \text{ ES}_{\frac{1}{5}}(f, L, a, \sigma_0) \]

begin
\[ t \leftarrow 1 \]
\[ s \leftarrow 0 \]
\[ \sigma \leftarrow \sigma_0 \]
\[ p_{\text{best}}.g \leftarrow \text{create starting point} \]
\[ p_{\text{best}}.x \leftarrow \text{gpm}(p_{\text{best}}.g) \]
\[ p_{\text{best}}.y \leftarrow f(p_{\text{best}}.x) \]
\[ p_{\text{new}} \leftarrow p_{\text{best}} \]
while \(!\text{shouldTerminate}\) do
\[ p_{\text{new}}.x \leftarrow \text{gpm}(p_{\text{new}}.g) \]
\[ p_{\text{new}}.y \leftarrow f(p_{\text{new}}.x) \]
if \( p_{\text{new}}.y < p_{\text{best}}.y \) then
\[ p_{\text{best}} \leftarrow p_{\text{new}} \]
\[ s \leftarrow s + 1 \]
if \((t \bmod L) = 0\) then
\[ P(S) \leftarrow \frac{s}{L} \]
if \(P(S) < 0.2\) then \( \sigma \leftarrow \sigma \ast a \)
else if \(P(S) > 0.2\) then \( \sigma \leftarrow \sigma / a \)
\[ s \leftarrow 0 \]
\[ p_{\text{new}}.g \leftarrow \text{mutationES}_\sigma(\sigma, p_{\text{new}}) \]
\[ t \leftarrow t + 1 \]
end while
return \( p_{\text{best}} \)

- \((1 + 1)\)-ES = self-adaptive hill climber
- After every \( L \) iterations, it is time for self-adaptation.
- Compute the achieved success rate \( P(S) \) during the last \( L \) steps.
- if larger than 0.2, increase \( \sigma \) by dividing it by constant \( a \in [0, 1] \).
Rule-based (1 + 1) Evolution Strategy

\[ p_{\text{best}} \leftarrow (1+1) \text{ES}_{\frac{1}{5}}(f, L, a, \sigma_0) \]

begin
  \[ t \leftarrow 1 \]
  \[ s \leftarrow 0 \]
  \[ \sigma \leftarrow \sigma_0 \]
  \[ p_{\text{best}}.g \leftarrow \text{create starting point} \]
  \[ p_{\text{best}}.x \leftarrow \text{gpm}(p_{\text{best}}.g) \]
  \[ p_{\text{best}}.y \leftarrow f(p_{\text{best}}.x) \]
  \[ p_{\text{new}} \leftarrow p_{\text{best}} \]
  while \( \neg \text{shouldTerminate} \) do
    \[ p_{\text{new}}.x \leftarrow \text{gpm}(p_{\text{new}}.g) \]
    \[ p_{\text{new}}.y \leftarrow f(p_{\text{new}}.x) \]
    if \( p_{\text{new}}.y < p_{\text{best}}.y \) then
      \[ p_{\text{best}} \leftarrow p_{\text{new}} \]
      \[ s \leftarrow s + 1 \]
    if \( (t \mod L) = 0 \) then
      \[ P(S) \leftarrow \frac{s}{L} \]
      if \( P(S) < 0.2 \) then \( \sigma \leftarrow \sigma * a \)
      else if \( P(S) > 0.2 \) then \( \sigma \leftarrow \sigma / a \)
      \[ s \leftarrow 0 \]
      \[ p_{\text{new}}.g \leftarrow \text{mutationES}_\sigma(\sigma, p_{\text{new}}) \]
    \[ t \leftarrow t + 1 \]
  return \( p_{\text{best}} \)

- (1 + 1)-ES = self-adaptive hill climber
- After every \( L \) iterations, it is time for self-adaptation.
- Compute the achieved success rate \( P(S) \) during the last \( L \) steps.
- if larger than 0.2, increase \( \sigma \) by dividing it by constant \( a \in [0, 1] \).
- Finally, reset success counter \( s \) to 0.
\( p_{\text{best}} \leftarrow (1+1) \text{ ES}_{\frac{1}{5}}(f, L, a, \sigma_0) \)

begin
\[
\begin{align*}
& t \leftarrow 1 \\
& s \leftarrow 0 \\
& \sigma \leftarrow \sigma_0 \\
& p_{\text{best}.g} \leftarrow \text{create starting point} \\
& p_{\text{best}.x} \leftarrow \text{gpm}(p_{\text{best}.g}) \\
& p_{\text{best}.y} \leftarrow f(p_{\text{best}.x}) \\
& p_{\text{new}} \leftarrow p_{\text{best}} \\
\end{align*}
\]
while \( \neg \text{shouldTerminate} \) do
\[
\begin{align*}
& p_{\text{new}.x} \leftarrow \text{gpm}(p_{\text{new}.g}) \\
& p_{\text{new}.y} \leftarrow f(p_{\text{new}.x}) \\
& \text{if } p_{\text{new}.y} < p_{\text{best}.y} \text{ then} \\
& \quad p_{\text{best}} \leftarrow p_{\text{new}} \\
& \quad s \leftarrow s + 1 \\
\end{align*}
\]
if \( (t \mod L) = 0 \) then
\[
\begin{align*}
& P(S) \leftarrow \frac{s}{L} \\
& \text{if } P(S) < 0.2 \text{ then } \sigma \leftarrow \sigma \times a \\
& \text{else if } P(S) > 0.2 \text{ then } \sigma \leftarrow \sigma / a \\
& s \leftarrow 0 \\
& p_{\text{new}.g} \leftarrow \text{mutationES}_\sigma(\sigma, p_{\text{new}}) \\
& t \leftarrow t + 1 \\
\end{align*}
\]
return \( p_{\text{best}} \)

- \((1+1)-\text{ES} = \text{self-adaptive hill climber}\)
- Perform mutation by using step-width \( \sigma \)
1/5th Rule-based \((1 + 1)\) Evolution Strategy

\[ p_{\text{best}} \leftarrow (1+1) \ ES_{\frac{1}{5}}(f, L, a, \sigma_0) \]

\[
\begin{align*}
\text{begin} & \\
& t \leftarrow 1 \\
& s \leftarrow 0 \\
& \sigma \leftarrow \sigma_0 \\
& p_{\text{best}}.g \leftarrow \text{create starting point} \\
& p_{\text{best}}.x \leftarrow gpm(p_{\text{best}}.g) \\
& p_{\text{best}}.y \leftarrow f(p_{\text{best}}.x) \\
& p_{\text{new}} \leftarrow p_{\text{best}} \\
\text{while } \neg \text{shouldTerminate do} & \\
& p_{\text{new}}.x \leftarrow gpm(p_{\text{new}}.g) \\
& p_{\text{new}}.y \leftarrow f(p_{\text{new}}.x) \\
& \text{if } p_{\text{new}}.y < p_{\text{best}}.y \text{ then} & \\
& & p_{\text{best}} \leftarrow p_{\text{new}} \\
& & s \leftarrow s + 1 \\
& \text{if } (t \mod L) = 0 \text{ then} & \\
& & P(S) \leftarrow \frac{s}{L} \\
& & \text{if } P(S) < 0.2 \text{ then } \sigma \leftarrow \sigma * a \\
& & \text{else if } P(S) > 0.2 \text{ then } \sigma \leftarrow \sigma / a \\
& & s \leftarrow 0 \\
& p_{\text{new}}.g \leftarrow \text{mutationES}_\sigma(\sigma, p_{\text{new}}) & \\
& t \leftarrow t + 1 & \\
\text{return } p_{\text{best}} 
\end{align*}
\]

- \((1 + 1)\)-ES = self-adaptive hill climber
- Increase iteration counter \(t\)
(1 + 1)-ES = self-adaptive hill climber

Return best candidate solution discovered.
public class ES1P1<X> extends OptimizationAlgorithm<double[], X> {
    public Individual<double[], X> solve(final IOobjectiveFunction<X> f) {
        Individual<double[], X> pstar = new Individual<>(); // best individual
        Individual<double[], X> pnew = new Individual<>(); // "new" individual
        RnESUnaryNormal esUnary = ((RnESUnaryNormal) (this.unary));

        double[] sigma = new double[] { this.sigma0};
        int s = 0;
        int t = 1; // init success and iteration counter

        pstar.g = this.nullary.create(this.random); // create first genotype
        pstar.x = this.gpm.gpm(pstar.g); // get phenotype for that genotype
        pstar.v = f.compute(pstar.x); // evaluate: how good is the phenotype?

        while (!this.termination.shouldTerminate()) { // until we should finish...
            pnew.g = esUnary.mutate(pstar.g, sigma, this.random); // mutate using sigma
            pnew.x = this.gpm.gpm(pnew.g); // get phenotype for that genotype
            pnew.v = f.compute(pnew.x); // evaluate the new phenotype

            if (pnew.v <= pstar.v) { // if the new individual is better...
                pstar.assign(pnew); // it becomes the new best individual
                s++; // count the success
            }

            if ((t % this.L) == 0) { // is it time to update sigma?
                double Ps = ((((double) s) / ((double) (this.L)))); // need floating point div!
                if (Ps < 0.2d) {
                    sigma[0] *= this.a; // not enough success: decrease sigma
                } else {
                    if (Ps > 0.2d) {
                        sigma[0] /= this.a; // too successful: increase sigma
                    }
                }
                s = 0; // reset success counter
            }

            t++; // count iteration
        }
        return pstar; // return the best individual that we have discovered
    }
}
The $1/5^{th}$ rule has advantages.
The $1/5^{th}$ rule has **advantages**:

- Many mutations are successful $\rightarrow$ we can make larger steps and progress faster
The $1/5^{th}$ rule has **advantages**:

- Many mutations are successful $\rightarrow$ we can make larger steps and progress faster
- Many mutations are unsuccessful $\rightarrow$ we make smaller steps so that we can approach the optimum instead of jumping over it
• The 1/5\textsuperscript{th} rule has \textbf{advantages}:
  • Many mutations are successful \(\longrightarrow\) we can make larger steps and progress faster
  • Many mutations are unsuccessful \(\longrightarrow\) we can make smaller steps so that we can approach the optimum instead of jumping over it
  • Balance between exploration and exploitation
The $1/5^{th}$ rule has advantages:

- Many mutations are successful $\rightarrow$ we can make larger steps and progress faster
- Many mutations are unsuccessful $\rightarrow$ we can make smaller steps so that we can approach the optimum instead of jumping over it
- Balance between exploration and exploitation
- Sound theoretical foundation
The 1/5\textsuperscript{th} rule has advantages:

- Many mutations are successful $\rightarrow$ we can make larger steps and progress faster
- Many mutations are unsuccessful $\rightarrow$ we make smaller steps so that we can approach the optimum instead of jumping over it
- Balance between exploration and exploitation
- Sound theoretical foundation

The 1/5\textsuperscript{th} rule has drawbacks
The 1/5\textsuperscript{th} rule has \textbf{advantages}:

- Many mutations are successful $\rightarrow$ we can make larger steps and progress faster
- Many mutations are unsuccessful $\rightarrow$ we make smaller steps so that we can approach the optimum instead of jumping over it
- Balance between exploration and exploitation
- Sound theoretical foundation

The 1/5\textsuperscript{th} rule has \textbf{drawbacks}:

- Can easily lead to premature convergence: no improvement $\Rightarrow$ smaller steps
The $\frac{1}{5}^{th}$ rule has **advantages**:

- Many mutations are successful $\implies$ we can make larger steps and progress faster
- Many mutations are unsuccessful $\implies$ we make smaller steps so that we can approach the optimum instead of jumping over it
- Balance between exploration and exploitation
- Sound theoretical foundation

The $\frac{1}{5}^{th}$ rule has **drawbacks**:

- Can easily lead to premature convergence: no improvement $\Rightarrow$ smaller steps
- intended for $(1 + 1)$-ES: makes no use of population
• The 1/5\textsuperscript{th} rule has advantages:
  • Many mutations are successful $\rightarrow$ we can make larger steps and progress faster
  • Many mutations are unsuccessful $\rightarrow$ we make smaller steps so that we can approach the optimum instead of jumping over it
  • Balance between exploration and exploitation
  • Sound theoretical foundation

• The 1/5\textsuperscript{th} rule has drawbacks:
  • Can easily lead to premature convergence: no improvement $\Rightarrow$ smaller steps
  • intended for (1 + 1)-ES: makes no use of population
  • Only a single parameter $\sigma$: cannot model different step widths for different dimensions or dependencies between dimensions
The $1/5^{th}$ rule has **advantages**:

- Many mutations are successful $\rightarrow$ we can make larger steps and progress faster
- Many mutations are unsuccessful $\rightarrow$ we make smaller steps so that we can approach the optimum instead of jumping over it
- Balance between exploration and exploitation
- Sound theoretical foundation

The $1/5^{th}$ rule has **drawbacks**:

- Can easily lead to premature convergence: no improvement $\Rightarrow$ smaller steps
- intended for (1 + 1)-ES: makes no use of population
- Only a single parameter $\sigma$: cannot model different step widths for different dimensions or dependencies between dimensions
- Only a single parameter $\sigma$: cannot easily be extended for population-based methods
Section Outline

1. Population Treatment
2. Mutation
3. Self-Adaptation
4. The 1/5th Rule
5. Endogeneous Adaptation
6. Recombination
7. Parameter Reproduction
8. CMA-ES
• Instead of having one central \textit{exogeneous} \textsuperscript{1} (set of) strategy parameter(s)...

\textsuperscript{1} \equiv \textit{outside of the genes}
• Instead of having one central *exogeneous* (set of) strategy parameter(s)...
• create one set of parameters of each individual in the Evolution Strategy!
• Instead of having one central **exogeneous** (set of) strategy parameter(s)...

• create one set of parameters of each individual in the Evolution Strategy!

• i.e., the individual records extended with information $p.w$ encoding the strategy parameters
• Instead of having one central **exogeneous** (set of) strategy parameter(s)...

• create one set of parameters of each individual in the Evolution Strategy!

• i.e., the individual records extended with information $p.w$ encoding the strategy parameters

• Strategy parameters are now similar to (or inside) the genotype/genes, i.e., are **endogeneous**$^1$

---

$^1 \equiv$ inside of the genes
Endogeneous Parameters

• Instead of having one central exogeneous (set of) strategy parameter(s)...
• create one set of parameters of each individual in the Evolution Strategy!
• i.e., the individual records extended with information $p.w$ encoding the strategy parameters
• Strategy parameters are now similar to (or inside) the genotype/genes, i.e., are endogeneous
• $p.w$ could be step-width for mutation if applied to individual $p$
• Instead of having one central **exogeneous** (set of) strategy parameter(s)...
• create one set of parameters of each individual in the Evolution Strategy!
• i.e.,, the individual records extended with information $p.w$ encoding the strategy parameters
• Strategy parameters are now similar to (or inside) the genotype/genes, i.e., are **endogeneous**
• $p.w$ could be step-width for mutation if applied to individual $p$
• Information $p.w$ undergoes reproduction, similar to genotypes $p.g \in \mathbb{G}$
• Instead of having one central *exogeneous* (set of) strategy parameter(s)...
• create one set of parameters of each individual in the Evolution Strategy!
• i.e., , the individual records extended with information $p.w$ encoding the strategy parameters
• Strategy parameters are now similar to (or inside) the genotype/genes, i.e., are *endogeneous*
• $p.w$ could be step-width for mutation if applied to individual $p$
• Information $p.w$ undergoes reproduction, similar to genotypes $p.g \in \mathbb{G}$
• As it is subject to selection, good strategy parameters will be discovered in the same way in which good candidate solutions are discovered.
\[ p_{\text{best}} \leftarrow \text{generalES}(f, \mu, \lambda, \rho) \]

\begin{itemize}
  \item Evolution Strategies follow same basic pattern as GAs
\end{itemize}

begin
  \[ t \leftarrow 1 \]
  \[ \text{pop} \leftarrow \text{random initialization} \]
  \textbf{while} \( \neg \text{shouldTerminate} \) \textbf{do}
    \text{perform genotype-phenotype mapping}
    \text{compute objective values (possibly update best-so-far solution \( p_{\text{best}} \))}
    \textbf{if} \( t > 1 \) \textbf{then}
      \textbf{if} \( \text{strategy} = (\mu/\rho, \lambda) \) \textbf{then}
        \[ \text{matePool} \leftarrow \text{truncationSelection}(\mu, \text{pop}) \]
      \textbf{else}
        \[ \text{matePool} \leftarrow \text{truncationSelection}(\mu, \text{pop} \cup \text{matePool}) \]
    \textbf{else}
      \[ \text{matePool} \leftarrow \text{pop} \]
    \textbf{for} \( i \leftarrow 1 \) \textbf{up to} \( \lambda \) \textbf{do}
      \[ \text{parents} \leftarrow \text{choose} \ \rho \ \text{parents from} \ \text{matePool} \]
      \[ p_{\text{new}.g} \leftarrow \text{recombinationES}(p.g \forall p \in \text{parents}) \]
      \[ p_{\text{new}.w} \leftarrow \text{infoRecombinationES}(p.w \forall p \in \text{parents}) \]
      \[ p_{\text{new}.w} \leftarrow \text{infoMutationES}(p_{\text{new}.w}) \]
      \[ p_{\text{new}.g} \leftarrow \text{mutationES}(p_{\text{new}.g}, p_{\text{new}.w}) \]
      \[ \text{population}[i] \leftarrow p_{\text{new}} \]
    \[ t \leftarrow t + 1 \]
\textbf{return} best solution \( \tilde{x} \) discovered
Endogeneous Adaptation

$p_{best} \leftarrow \text{generalES}(f, \mu, \lambda, \rho)$

\begin{align*}
\text{begin} \\
\quad t &\leftarrow 1 \\
\quad \text{pop} &\leftarrow \text{random initialization} \\
\text{while } &\neg \text{shouldTerminate} \text{ do} \\
\quad &\text{perform genotype-phenotype mapping} \\
\quad &\text{compute objective values (possibly update best-so-far solution } p_{best}) \\
\quad &\text{if } t > 1 \text{ then} \\
\quad &\quad \text{if strategy} = (\mu/\rho, \lambda) \text{ then} \\
\quad &\quad \quad \text{matePool} \leftarrow \text{truncationSelection}(\mu, \text{pop}) \\
\quad &\quad \text{else} \\
\quad &\quad \quad \text{matePool} \leftarrow \text{truncationSelection}(\mu, \text{pop} \cup \text{matePool}) \\
\quad &\text{else} \\
\quad &\quad \text{matePool} \leftarrow \text{pop} \\
\quad &\text{for } i \leftarrow 1 \text{ up to } \lambda \text{ do} \\
\quad &\quad \text{parents} \leftarrow \text{choose } \rho \text{ parents from matePool} \\
\quad &\quad p_{new.g} \leftarrow \text{recombinationES}(p.g \forall p \in \text{parents}) \\
\quad &\quad p_{new.w} \leftarrow \text{infoRecombinationES}(p.w \forall p \in \text{parents}) \\
\quad &\quad p_{new.w} \leftarrow \text{infoMutationES}(p_{new.w}) \\
\quad &\quad p_{new.g} \leftarrow \text{mutationES}(p_{new.g}, p_{new.w}) \\
\quad &\quad \text{population}[i] \leftarrow p_{new} \\
\quad &\quad t \leftarrow t + 1 \\
\text{return} &\text{best solution } \tilde{x} \text{ discovered}
\end{align*}

- Evolution Strategies follow same basic pattern as GAs
- Start with creating a random initial population, where each individual $p$ has a (random) point in search space $p.g$ (genotype) and (random) information $p.w$ component
Endogeneous Adaptation

\[ p_{\text{best}} \leftarrow \text{generalES}(f, \mu, \lambda, \rho) \]

begin
\[
t \leftarrow 1 \\
pop \leftarrow \text{random initialization} \\
\text{while } \neg \text{shouldTerminate do} \\
\text{perform genotype-phenotype mapping} \\
\text{compute objective values (possibly update best-so-far solution } p_{\text{best}}) \\
\text{if } t > 1 \text{ then} \\
\quad \text{if strategy} = (\mu/\rho, \lambda) \text{ then} \\
\quad \quad \text{matePool} \leftarrow \text{truncationSelection}(\mu, \text{pop}) \\
\quad \quad \text{else} \\
\quad \quad \quad \text{matePool} \leftarrow \text{truncationSelection}(\mu, \text{pop} \cup \text{matePool}) \\
\quad \text{else} \\
\quad \quad \text{matePool} \leftarrow \text{pop} \\
\text{for } i \leftarrow 1 \text{ up to } \lambda \text{ do} \\
\quad \text{parents} \leftarrow \text{choose } \rho \text{ parents from matePool} \\
\quad p_{\text{new}.g} \leftarrow \text{recombinationES}(p.g \forall p \in \text{parents}) \\
\quad p_{\text{new}.w} \leftarrow \text{infoRecombinationES}(p.w \forall p \in \text{parents}) \\
\quad p_{\text{new}.w} \leftarrow \text{infoMutationES}(p_{\text{new}.w}) \\
\quad p_{\text{new}.g} \leftarrow \text{mutationES}(p_{\text{new}.g}, p_{\text{new}.w}) \\
\quad \text{population}[i] \leftarrow p_{\text{new}} \\
\quad t \leftarrow t + 1 \\
\text{return best solution } \tilde{x} \text{ discovered} \]

- Evolution Strategies follow same basic pattern as GAs
- Perform the genotype-phenotype mapping, i.e., translate the genotypes \( p.g \) to the corresponding candidate solutions \( p.x \)
Endogeneous Adaptation

\[ p_{\text{best}} \leftarrow \text{generalES}(f, \mu, \lambda, \rho) \]

\begin{algorithm}
\begin{algorithmic}
\State \( t \leftarrow 1 \)
\State \( \text{pop} \leftarrow \text{random initialization} \)
\While {\neg \text{shouldTerminate}}
\State perform genotype-phenotype mapping
\State compute objective values (possibly update best-so-far solution \( p_{\text{best}} \))
\If {\( t > 1 \)}
\If {strategy = \((\mu/\rho, \lambda)\)}
\State \( \text{matePool} \leftarrow \text{truncationSelection}(\mu, \text{pop}) \)
\Else
\State \( \text{matePool} \leftarrow \text{truncationSelection}(\mu, \text{pop} \cup \text{matePool}) \)
\EndIf
\Else
\State \( \text{matePool} \leftarrow \text{pop} \)
\EndIf
\For {\( i \leftarrow 1 \) up to \( \lambda \)}
\State \( \text{parents} \leftarrow \text{choose} \ \rho \ \text{parents from} \ \text{matePool} \)
\State \( p_{\text{new}.g} \leftarrow \text{recombinationES}(p.g \forall p \in \text{parents}) \)
\State \( p_{\text{new}.w} \leftarrow \text{infoRecombinationES}(p.w \forall p \in \text{parents}) \)
\State \( p_{\text{new}.w} \leftarrow \text{infoMutationES}(p_{\text{new}.w}) \)
\State \( p_{\text{new}.g} \leftarrow \text{mutationES}(p_{\text{new}.g}, p_{\text{new}.w}) \)
\State \( \text{population}[i] \leftarrow p_{\text{new}} \)
\EndFor
\State \( t \leftarrow t + 1 \)
\EndWhile
\Return \text{best solution} \bar{x} \text{ discovered}
\end{algorithmic}
\end{algorithm}

- Evolution Strategies follow same basic pattern as GAs
- Compute the objective values (update best-so-far solution \( p_{\text{best}} \) if a new, better one is discovered)
Endogeneous Adaptation

$p_{\text{best}} \leftarrow \text{generalES}(f, \mu, \lambda, \rho)$

\[ \text{begin} \]
\[ t \leftarrow 1 \]
\[ \text{pop} \leftarrow \text{random initialization} \]
\[ \text{while } \neg \text{shouldTerminate do} \]
\[ \quad \text{perform genotype-phenotype mapping} \]
\[ \quad \text{compute objective values (possibly update best-so-far solution } p_{\text{best}}) \]
\[ \quad \text{if } t > 1 \text{ then} \]
\[ \quad \quad \text{if } \text{strategy} = (\mu/\rho, \lambda) \text{ then} \]
\[ \quad \quad \quad \text{matePool } \leftarrow \text{truncationSelection}(\mu, \text{pop}) \]
\[ \quad \quad \text{else} \]
\[ \quad \quad \quad \text{matePool } \leftarrow \text{truncationSelection}(\mu, \text{pop} \cup \text{matePool}) \]
\[ \quad \text{else} \]
\[ \quad \quad \text{matePool } \leftarrow \text{pop} \]
\[ \text{for } i \leftarrow 1 \text{ up to } \lambda \text{ do} \]
\[ \quad \text{parents } \leftarrow \text{choose } \rho \text{ parents from matePool} \]
\[ \quad p_{\text{new}.g} \leftarrow \text{recombinationES}(p.g \forall p \in \text{parents}) \]
\[ \quad p_{\text{new}.w} \leftarrow \text{infoRecombinationES}(p.w \forall p \in \text{parents}) \]
\[ \quad p_{\text{new}.w} \leftarrow \text{infoMutationES}(p_{\text{new}.w}) \]
\[ \quad p_{\text{new}.g} \leftarrow \text{mutationES}(p_{\text{new}.g, p_{\text{new}.w}}) \]
\[ \quad \text{population}[i] \leftarrow p_{\text{new}} \]
\[ \quad t \leftarrow t + 1 \]
\[ \text{return best solution } \tilde{x} \text{ discovered} \]

- Evolution Strategies follow same basic pattern as GAs
- Perform survival selection
Endogeneous Adaptation

\[ p_{\text{best}} \leftarrow \text{generalES}(f, \mu, \lambda, \rho) \]

begin

\[ t \leftarrow 1 \]
\[ \text{pop} \leftarrow \text{random initialization} \]

while \( \neg \text{shouldTerminate} \) do

perform genotype-phenotype mapping
compute objective values (possibly update best-so-far solution \( p_{\text{best}} \))

if \( t > 1 \) then

if \( \text{strategy} = (\mu/\rho, \lambda) \) then

\[ \text{matePool} \leftarrow \text{truncationSelection}(\mu, \text{pop}) \]

else

\[ \text{matePool} \leftarrow \text{truncationSelection}(\mu, \text{pop} \cup \text{matePool}) \]

else

\[ \text{matePool} \leftarrow \text{pop} \]

for \( i \leftarrow 1 \) up to \( \lambda \) do

\[ \text{parents} \leftarrow \text{choose } \rho \text{ parents from } \text{matePool} \]
\[ p_{\text{new}.g} \leftarrow \text{recombinationES}(p.g \forall p \in \text{parents}) \]
\[ p_{\text{new}.w} \leftarrow \text{infoRecombinationES}(p.w \forall p \in \text{parents}) \]
\[ p_{\text{new}.w} \leftarrow \text{infoMutationES}(p_{\text{new}.w}) \]
\[ p_{\text{new}.g} \leftarrow \text{mutationES}(p_{\text{new}.g}, p_{\text{new}.w}) \]

\[ \text{population}[i] \leftarrow p_{\text{new}} \]

\[ t \leftarrow t + 1 \]

return best solution \( \tilde{x} \) discovered

- Evolution Strategies follow same basic pattern as GAs
- Perform survival selection
  - in \((\mu, \lambda)\)-ESs, select only from the current population
\( p_{\text{best}} \leftarrow \text{generalES}(f, \mu, \lambda, \rho) \)

\[
\begin{align*}
&\text{begin} \\
&t \leftarrow 1 \\
&\text{pop} \leftarrow \text{random initialization} \\
\text{while} \ \neg \text{shouldTerminate} \ \text{do} \\
&\text{perform genotype-phenotype mapping} \\
&\text{compute objective values (possibly update best-so-far solution } p_{\text{best}}) \\
&\text{if } t > 1 \text{ then} \\
&\quad \text{if } \text{strategy} = (\mu/\rho, \lambda) \text{ then} \\
&\quad \quad \text{matePool} \leftarrow \text{truncationSelection}(\mu, \text{pop}) \\
&\quad \text{else} \\
&\quad \quad \text{matePool} \leftarrow \text{truncationSelection}(\mu, \text{pop} \cup \text{matePool}) \\
&\text{else} \\
&\quad \text{matePool} \leftarrow \text{pop} \\
&\text{for } i \leftarrow 1 \text{ up to } \lambda \text{ do} \\
&\quad \text{parents} \leftarrow \text{choose } \rho \text{ parents from matePool} \\
&\quad p_{\text{new}.g} \leftarrow \text{recombinationES}(p.g \forall p \in \text{parents}) \\
&\quad p_{\text{new}.w} \leftarrow \text{infoRecombinationES}(p.w \forall p \in \text{parents}) \\
&\quad p_{\text{new}.w} \leftarrow \text{infoMutationES}(p_{\text{new}.w}) \\
&\quad p_{\text{new}.g} \leftarrow \text{mutationES}(p_{\text{new}.g, p_{\text{new}.w})} \\
&\quad \text{population}[i] \leftarrow p_{\text{new}} \\
&t \leftarrow t + 1 \\
&\text{return best solution } \tilde{x} \text{ discovered}
\end{align*}
\]

- Evolution Strategies follow same basic pattern as GAs
- Perform survival selection
  - in \((\mu, \lambda)\)-ESs, select only from the current population
  - in \((\mu + \lambda)\)-ESs, select from the current population and the mating pool
\( p_{\text{best}} \leftarrow \text{generalES}(f, \mu, \lambda, \rho) \)

begin
\[ t \leftarrow 1 \]
\[ \text{pop} \leftarrow \text{random initialization} \]
while \( \neg \text{shouldTerminate} \) do
\[ \text{perform genotype-phenotype mapping} \]
\[ \text{compute objective values (possibly update best-so-far solution } p_{\text{best}}) \]
if \( t > 1 \) then
\[ \text{if strategy} = (\mu/\rho, \lambda) \text{ then} \]
\[ \text{matePool} \leftarrow \text{truncationSelection}(\mu, \text{pop}) \]
else
\[ \text{matePool} \leftarrow \text{truncationSelection}(\mu, \text{pop} \cup \text{matePool}) \]
else
\[ \text{matePool} \leftarrow \text{pop} \]
for \( i \leftarrow 1 \) up to \( \lambda \) do
\[ \text{parents} \leftarrow \text{choose } \rho \text{ parents from matePool} \]
\[ p_{\text{new}.g} \leftarrow \text{recombinationES}(p.g \forall p \in \text{parents}) \]
\[ p_{\text{new}.w} \leftarrow \text{infoRecombinationES}(p.w \forall p \in \text{parents}) \]
\[ p_{\text{new}.w} \leftarrow \text{infoMutationES}(p_{\text{new}.w}) \]
\[ p_{\text{new}.g} \leftarrow \text{mutationES}(p_{\text{new}.g, p_{\text{new}.w}}) \]
\[ \text{population}[i] \leftarrow p_{\text{new}} \]
\[ t \leftarrow t + 1 \]
return best solution \( \tilde{x} \) discovered

- Evolution Strategies follow same basic pattern as GAs
- Perform survival selection
  - in \((\mu, \lambda)\)-ESs, select only from the current population
  - in \((\mu + \lambda)\)-ESs, select from the current population and the mating pool
- Only in first iteration: no selection.
Evolution Strategies follow same basic pattern as GAs

For each offspring individual that we want to create...
Endogeneous Adaptation

\[ p_{\text{best}} \leftarrow \text{generalES}(f, \mu, \lambda, \rho) \]

begin
\[
t \leftarrow 1
\]
\[
\text{pop} \leftarrow \text{random initialization}
\]
while \( \neg \text{shouldTerminate} \) do
\[
\text{perform genotype-phenotype mapping}
\]
\[
\text{compute objective values (possibly update best-so-far solution } p_{\text{best}} \text{)}
\]
if \( t > 1 \) then
\[
\text{if strategy} = (\mu/\rho, \lambda) \text{ then}
\]
\[
\text{matePool} \leftarrow \text{truncationSelection}(\mu, \text{pop})
\]
else
\[
\text{matePool} \leftarrow \text{truncationSelection}(\mu, \text{pop} \cup \text{matePool})
\]
else
\[
\text{matePool} \leftarrow \text{pop}
\]
\for i \leftarrow 1 \text{ up to } \lambda \do
\[
\text{parents} \leftarrow \text{choose } \rho \text{ parents from matePool}
\]
\[
p_{\text{new}.g} \leftarrow \text{recombinationES}(p.g \forall p \in \text{parents})
\]
\[
p_{\text{new}.w} \leftarrow \text{infoRecombinationES}(p.w \forall p \in \text{parents})
\]
\[
p_{\text{new}.w} \leftarrow \text{infoMutationES}(p_{\text{new}.w})
\]
\[
p_{\text{new}.g} \leftarrow \text{mutationES}(p_{\text{new}.g}, p_{\text{new}.w})
\]
\[
\text{population}[i] \leftarrow p_{\text{new}}
\]
\[
t \leftarrow t + 1
\]
return best solution \( \tilde{x} \) discovered

- Evolution Strategies follow same basic pattern as GAs
- For each offspring individual that we want to create...
  - ...first select \( \rho \) parents,
Metaheuristic Optimization
Evolution Strategies follow the same basic pattern as GAs. For each offspring individual that we want to create...

- ...first select $\rho$ parents,
- (re)combine the parental genotypes,
- and then (re)combine the endogenous information.
\[ p_{\text{best}} \leftarrow \text{generalES}(f, \mu, \lambda, \rho) \]

begin
\[
t \leftarrow 1 \\
\text{pop} \leftarrow \text{random initialization} \\
\text{while } \neg \text{shouldTerminate do} \\
\quad \text{perform genotype-phenotype mapping} \\
\quad \text{compute objective values (possibly update best-so-far solution } p_{\text{best}}) \\
\quad \text{if } t > 1 \text{ then} \\
\quad \quad \text{if strategy} = (\mu/\rho, \lambda) \text{ then} \\
\quad \quad \quad \text{matePool} \leftarrow \text{truncationSelection}(\mu, \text{pop}) \\
\quad \quad \text{else} \\
\quad \quad \quad \text{matePool} \leftarrow \text{truncationSelection}(\mu, \text{pop} \cup \text{matePool}) \\
\quad \text{else} \\
\quad \quad \text{matePool} \leftarrow \text{pop} \\
\text{for } i \leftarrow 1 \text{ up to } \lambda \text{ do} \\
\quad \text{parents} \leftarrow \text{choose} \ \rho \ \text{parents from matePool} \\
\quad p_{\text{new}.g} \leftarrow \text{recombinationES}(p.g \forall p \in \text{parents}) \\
\quad p_{\text{new}.w} \leftarrow \text{infoRecombinationES}(p.w \forall p \in \text{parents}) \\
\quad p_{\text{new}.w} \leftarrow \text{infoMutationES}(p_{\text{new}.w}) \\
\quad p_{\text{new}.g} \leftarrow \text{mutationES}(p_{\text{new}.g, p_{\text{new}.w}) \\
\quad \text{population}[i] \leftarrow p_{\text{new}} \\
\text{t} \leftarrow t + 1 \\
\text{return best solution } \tilde{x} \text{ discovered} \]

- Evolution Strategies follow same basic pattern as GAs
- For each offspring individual that we want to create...
  - One endogeneous set of parameters (for mutation operation) exists per individual
\[ p_{\text{best}} \leftarrow \text{generalES}(f, \mu, \lambda, \rho) \]

```
begin
  t ← 1
  pop ← random initialization
  while ¬shouldTerminate do
    perform genotype-phenotype mapping
    compute objective values (possibly update best-so-far solution \( p_{\text{best}} \))
    if \( t > 1 \) then
      if strategy = (\( \mu/\rho \), \( \lambda \)) then
        matePool ← truncationSelection(\( \mu \), pop)
      else
        matePool ← truncationSelection(\( \mu \), pop \( \cup \) matePool)
    else
      matePool ← pop
    for \( i ← 1 \) up to \( \lambda \) do
      parents ← choose \( \rho \) parents from matePool
      \( p_{\text{new.g}} \) ← recombinationES(\( p.g \forall p \in \text{parents} \))
      \( p_{\text{new.w}} \) ← infoRecombinationES(\( p.w \forall p \in \text{parents} \))
      \( p_{\text{new.w}} \) ← infoMutationES(\( p_{\text{new.w}} \))
      \( p_{\text{new.g}} \) ← mutationES(\( p_{\text{new.g}}, p_{\text{new.w}} \))
      \( \text{population}[i] \leftarrow p_{\text{new}} \)
    t ← t + 1
  return best solution \( \tilde{x} \) discovered
```

- Evolution Strategies follow same basic pattern as GAs
- For each offspring individual that we want to create...
  - One endogeneous set of parameters (for mutation operation) exists per individual
  - Parameter setting of algorithm evolves along with the solutions
Endogeneous Adaptation

$p_{\text{best}} \leftarrow \text{generalES}(f, \mu, \lambda, \rho)$

\begin{verbatim}
begin
  t ← 1
  pop ← random initialization
  while ¬shouldTerminate do
    perform genotype-phenotype mapping
    compute objective values (possibly update best-so-far solution $p_{\text{best}}$)
    if $t > 1$ then
      if strategy = $(\mu/\rho, \lambda)$ then
        matePool ← truncationSelection($\mu$, pop)
      else
        matePool ← truncationSelection($\mu$, pop $\cup$ matePool)
    else
      matePool ← pop
    for $i ← 1$ up to $\lambda$ do
      parents ← choose $\rho$ parents from matePool
      $p_{\text{new}.g} ← \text{recombinationES}(p.g \forall p \in \text{parents})$
      $p_{\text{new}.w} ← \text{infoRecombinationES}(p.w \forall p \in \text{parents})$
      $p_{\text{new}.w} ← \text{infoMutationES}(p_{\text{new}.w})$
      $p_{\text{new}.g} ← \text{mutationES}(p_{\text{new}.g}, p_{\text{new}.w})$
      population[$i$] ← $p_{\text{new}}$
    t ← $t + 1$
return best solution $\tilde{x}$ discovered
\end{verbatim}

- Evolution Strategies follow same basic pattern as GAs
- For each offspring individual that we want to create...
  - One endogeneous set of parameters (for mutation operation) exists per individual
  - Parameter setting of algorithm evolves along with the solutions
  - Good settings survive along with the candidate solutions that they have created
Endogeneous Adaptation

\[ p_{\text{best}} \leftarrow \text{generalES}(f, \mu, \lambda, \rho) \]

begin
  \[ t \leftarrow 1 \]
  \[ \text{pop} \leftarrow \text{random initialization} \]
  \[ \text{while} \ \neg \text{shouldTerminate} \ \text{do} \]
  \hline
    perform genotype-phenotype mapping
    compute objective values (possibly update best-so-far solution \( p_{\text{best}} \))
  \hline
  if \( t > 1 \) then
    if strategy = (\( \mu/\rho, \lambda \)) then
      \[ \text{matePool} \leftarrow \text{truncationSelection}(\mu, \text{pop}) \]
    else
      \[ \text{matePool} \leftarrow \text{truncationSelection}(\mu, \text{pop} \cup \text{matePool}) \]
  else
    \[ \text{matePool} \leftarrow \text{pop} \]
  \hline
  for \( i \leftarrow 1 \) up to \( \lambda \) do
    \[ \text{parents} \leftarrow \text{choose} \ \rho \ \text{parents} \ \text{from} \ \text{matePool} \]
    \[ p_{\text{new}.g} \leftarrow \text{recombinationES}(p.g \forall p \in \text{parents}) \]
    \[ p_{\text{new}.w} \leftarrow \text{infoRecombinationES}(p.w \forall p \in \text{parents}) \]
    \[ p_{\text{new}.w} \leftarrow \text{infoMutationES}(p_{\text{new}.w}) \]
    \[ p_{\text{new}.g} \leftarrow \text{mutationES}(p_{\text{new}.g, p_{\text{new}.w}}) \]
    \[ \text{population}[i] \leftarrow p_{\text{new}} \]
  \hline
  \[ t \leftarrow t + 1 \]
\hline
return best solution \( \tilde{x} \) discovered

- Evolution Strategies follow same basic pattern as GAs
- Mutate the endogenous information (parameter settings for genotype mutation)
Endogeneous Adaptation

\[ p_{\text{best}} \leftarrow \text{generalES}(f, \mu, \lambda, \rho) \]

begin
\[ t \leftarrow 1 \]
\[ \text{pop} \leftarrow \text{random initialization} \]
while \( \neg \text{shouldTerminate} \) do
    perform genotype-phenotype mapping
    compute objective values (possibly update best-so-far solution \( p_{\text{best}} \))
    if \( t > 1 \) then
        if strategy = (\( \mu/\rho \), \( \lambda \)) then
            \[ \text{matePool} \leftarrow \text{truncationSelection}(\mu, \text{pop}) \]
        else
            \[ \text{matePool} \leftarrow \text{truncationSelection}(\mu, \text{pop} \cup \text{matePool}) \]
    else
        \[ \text{matePool} \leftarrow \text{pop} \]
    for \( i \leftarrow 1 \) up to \( \lambda \) do
        parents \leftarrow \text{choose} \( \rho \) parents from \( \text{matePool} \)
        \[ p_{\text{new}.g} \leftarrow \text{recombinationES}(p.g \forall p \in \text{parents}) \]
        \[ p_{\text{new}.w} \leftarrow \text{infoRecombinationES}(p.w \forall p \in \text{parents}) \]
        \[ p_{\text{new}.w} \leftarrow \text{infoMutationES}(p_{\text{new}.w}) \]
        \[ p_{\text{new}.g} \leftarrow \text{mutationES}(p_{\text{new}.g}, p_{\text{new}.w}) \]
        \[ \text{population}[i] \leftarrow p_{\text{new}} \]
    \[ t \leftarrow t + 1 \]
return best solution \( \tilde{x} \) discovered

- Evolution Strategies follow same basic pattern as GAs
- Mutate the endogenous information (parameter settings for genotype mutation)
- Endogenous information is mutated before applying it in the mutation operation: only the values that actually influenced the fitness are in \( w \)
Endogeneous Adaptation

\[ p_{\text{best}} \leftarrow \text{generalES}(f, \mu, \lambda, \rho) \]

begin
\[ t \leftarrow 1 \]
\[ \text{pop} \leftarrow \text{random initialization} \]
while \( \neg \text{shouldTerminate} \) do
\[ \text{perform genotype-phenotype mapping} \]
\[ \text{compute objective values (possibly update best-so-far solution } p_{\text{best}}) \]
\[ \text{if } t > 1 \text{ then} \]
\[ \text{if strategy} = (\mu/\rho, \lambda) \text{ then} \]
\[ \text{matePool} \leftarrow \text{truncationSelection}(\mu, \text{pop}) \]
\[ \text{else} \]
\[ \text{matePool} \leftarrow \text{truncationSelection}(\mu, \text{pop} \cup \text{matePool}) \]
\[ \text{else} \]
\[ \text{matePool} \leftarrow \text{pop} \]
for \( i \leftarrow 1 \) up to \( \lambda \) do
\[ \text{parents} \leftarrow \text{choose } \rho \text{ parents from matePool} \]
\[ p_{\text{new}.g} \leftarrow \text{recombinationES}(p.g \forall p \in \text{parents}) \]
\[ p_{\text{new}.w} \leftarrow \text{infoRecombinationES}(p.w \forall p \in \text{parents}) \]
\[ p_{\text{new}.w} \leftarrow \text{infoMutationES}(p_{\text{new}.w}) \]
\[ p_{\text{new}.g} \leftarrow \text{mutationES}(p_{\text{new}.g}, p_{\text{new}.w}) \]
\[ \text{population}[i] \leftarrow p_{\text{new}} \]
\[ t \leftarrow t + 1 \]
return best solution \( \tilde{x} \) discovered

- Evolution Strategies follow same basic pattern as GAs
- Use the mutated endogenous information as parameter for the mutation operator (e.g., as step length) in order to mutate the newly created genotype
Endogeneous Adaptation

\[ p_{\text{best}} \leftarrow \text{generalES}(f, \mu, \lambda, \rho) \]

begin
\[
\begin{align*}
    t & \leftarrow 1 \\
    \text{pop} & \leftarrow \text{random initialization} \\
    \text{while } & \neg \text{shouldTerminate do} \\
    & \text{perform genotype-phenotype mapping} \\
    & \text{compute objective values (possibly update best-so-far solution } p_{\text{best}}) \\
    \text{if } t & > 1 \text{ then} \\
    & \text{if strategy } = (\mu/\rho, \lambda) \text{ then} \\
    & \quad \text{matePool } \leftarrow \text{truncationSelection}(\mu, \text{pop}) \\
    & \quad \text{else} \\
    & \quad \text{matePool } \leftarrow \text{truncationSelection}(\mu, \text{pop} \cup \text{matePool}) \\
    \text{else} & \\
    & \quad \text{matePool } \leftarrow \text{pop} \\
    \text{for } i & \leftarrow 1 \text{ up to } \lambda \text{ do} \\
    & \quad \text{parents } \leftarrow \text{choose } \rho \text{ parents from matePool} \\
    & \quad p_{\text{new}.g} \leftarrow \text{recombinationES}(p.g \forall p \in \text{parents}) \\
    & \quad p_{\text{new}.w} \leftarrow \text{infoRecombinationES}(p.w \forall p \in \text{parents}) \\
    & \quad p_{\text{new}.w} \leftarrow \text{infoMutationES}(p_{\text{new}.w}) \\
    & \quad p_{\text{new}.g} \leftarrow \text{mutationES}(p_{\text{new}.g}, p_{\text{new}.w}) \\
    & \quad \text{population}[i] \leftarrow p_{\text{new}} \\
    & \quad t \leftarrow t + 1 \\
\end{align*}
\]

return best solution \( \tilde{x} \) discovered

- Evolution Strategies follow same basic pattern as GAs
- Put the new individuals into the population
Endogeneous Adaptation

\[
p_{\text{best}} \leftarrow \text{generalES}(f, \mu, \lambda, \rho)
\]

begin
\[t \leftarrow 1\]
\[\text{pop} \leftarrow \text{random initialization}\]
while \(\neg \text{shouldTerminate}\) do
perform genotype-phenotype mapping
compute objective values (possibly update best-so-far solution \(p_{\text{best}}\))
if \(t > 1\) then
  if \text{strategy} = (\mu/\rho, \lambda) then
    \[\text{matePool} \leftarrow \text{truncationSelection}(\mu, \text{pop})\]
  else
    \[\text{matePool} \leftarrow \text{truncationSelection}(\mu, \text{pop} \cup \text{matePool})\]
else
  \[\text{matePool} \leftarrow \text{pop}\]
for \(i \leftarrow 1\) up to \(\lambda\) do
  \[\text{parents} \leftarrow \text{choose} \ \rho\ \text{parents from matePool}\]
  \[p_{\text{new}.g} \leftarrow \text{recombinationES}(p.g \forall p \in \text{parents})\]
  \[p_{\text{new}.w} \leftarrow \text{infoRecombinationES}(p.w \forall p \in \text{parents})\]
  \[p_{\text{new}.w} \leftarrow \text{infoMutationES}(p_{\text{new}.w})\]
  \[p_{\text{new}.g} \leftarrow \text{mutationES}(p_{\text{new}.g}, p_{\text{new}.w})\]
  \[\text{population}[i] \leftarrow p_{\text{new}}\]
\[t \leftarrow t + 1\]
return best solution \(\tilde{x}\) discovered

- Evolution Strategies follow same basic pattern as GAs
- Put the new individuals into the population...
- ...and start the next cycle
Endogeneous Adaptation

\[ p_{\text{best}} \leftarrow \text{generalES}(f, \mu, \lambda, \rho) \]

- Evolution Strategies follow same basic pattern as GAs
- Return the best candidate solutions that were discovered
Section Outline

1 Population Treatment
2 Mutation
3 Self-Adaptation
4 The 1/5th Rule
5 Endogeneous Adaptation
6 Recombination
7 Parameter Reproduction
8 CMA-ES

Metaheuristic Optimization

Thomas Weise
• Extension of uniform crossover to $\rho$ real vectors
• Extension of uniform crossover to \( \rho \) real vectors
• if \( \rho = 2 \), returns corner of hyper-cube created by parents
• Extension of uniform crossover to \( \rho \) real vectors
• if \( \rho = 2 \), returns corner of hyper-cube created by parents

\[
\vec{g}' \leftarrow \text{recombinationDiscrete(parents)}
\]

**Input:** parents: the list of \( \rho \) parent individuals
**Data:** \( i \): a counter variable
**Data:** \( p \): a parent individual
**Output:** \( \vec{g}' \): the offspring of the parents

begin
for \( i \leftarrow 0 \) up to \( n - 1 \) do
\[
\begin{align*}
    p & \leftarrow \text{parents[\{randomly from 0..\rho - 1\}]} \\
    \vec{g}'_i & \leftarrow p \cdot g_i
\end{align*}
\]
return \( \vec{g}' \)
public class RnBinaryDiscrete extends Rn implements IBinarySearchOperation<double[][]> {

    public double[] recombine(final double[] parent1, final double[] parent2, final Random r) {

        double[] res = parent1.clone();

        for (int i = parent2.length; (--i) >= 0;) {
            if (r.nextBoolean()) {
                res[i] = parent2[i];
            }
        }

        return res;
    }
}
• Extension of weighted average crossover to $\rho$ real vectors
• Extension of weighted average crossover to $\rho$ real vectors
• Returns a point inside of hyper-cube defined by the parents
Intermediate Recombination

- Extension of weighted average crossover to $\rho$ real vectors
- Returns a point inside of hyper-cube defined by the parents

\[
\vec{g}' \leftarrow \text{recombinationIntermediate}(\text{parents})
\]

**Input:** parents: the list of $\rho$ parent individuals  
**Data:** $p$: a parent individual  
**Output:** $\vec{g}'$: the offspring of the parents

begin  
for $i \leftarrow 0$ up to $n - 1$ do  
    $s \leftarrow 0$  
    for $j \leftarrow 0$ up to $\rho - 1$ do  
        $p \leftarrow \text{parents}[j]$  
        $s \leftarrow s + p.g_i$  
        $\vec{g}'_i \leftarrow \frac{s}{\rho}$  
return $\vec{g}'$
```java
public class RnBinaryIntermediate extends Rn implements IBinarySearchOperation<double[]>
{

    public double[] recombine(final double[] parent1, final double[] parent2, final Random r) {

        double[] res = new double[parent1.length];

        for (int i = parent2.length; (--i) >= 0;) {
            res[i] = (0.5d * (parent1[i] + parent2[i]));
        }

        return res;
    }

}
```
1 Population Treatment
2 Mutation
3 Self-Adaptation
4 The 1/5th Rule
5 Endogeneous Adaptation
6 Recombination
7 Parameter Reproduction
8 CMA-ES
• Recombination of strategy parameters $p.w$: intermediate crossover
Recombination of strategy parameters $p.w$: intermediate crossover

Mutation is different from mutation of a genotype: we mutate a mutation strength
• Reproduction of endogenous parameters $p.w$: intermediate crossover
• Mutation is different from mutation of a genotype: we mutate a mutation strength
• Mutation means to mutate the degree with which a candidate solution should be changed
• Recombination of strategy parameters $p.w$: intermediate crossover
• Mutation is different from mutation of a genotype: we mutate a mutation strength
• Mutation means to mutate the degree with which a candidate solution should be changed
• Here, not the absolute value (1, 7, 1.5, etc.) is interesting...
Recombination of strategy parameters $p_w$: intermediate crossover

Mutation is different from mutation of a genotype: we mutate a mutation strength

Mutation means to mutate the degree with which a candidate solution should be changed

Here, not the absolute value (1, 7, 1.5, etc.) is interesting...

...but more the scale, i.e., 1, 10, 0.1, 10000, 0.001, ...
• Recombination of strategy parameters $p.w$: intermediate crossover
• So we are concerned about the scale
• Recombination of strategy parameters $p.w$: intermediate crossover
• So we are concerned about the scale
• Normal distribution produces values of approximately same scale around its center
Reproduction of Endogenous Parameters

- Recombination of strategy parameters $p.w$: intermediate crossover
- So we are concerned about the scale
- Log-Normal distribution produces values of difference scale
Reproduction of Endogenous Parameters

- Recombination of strategy parameters $p.w$: intermediate crossover
- Mutation of strategy parameters: apply lognormal mutation
Reproduction of Endogenous Parameters

- Recombination of strategy parameters $p.w$: intermediate crossover

- Mutation of strategy parameters: apply lognormal mutation

$$\vec{\sigma} \leftarrow \text{infoMutationES}(\vec{\sigma}')$$

**Input:** $\vec{\sigma}' \in \mathbb{R}^n$: the old mutation strength vector

**Data:** $i$: a counter variable

**Output:** $\vec{\sigma} \in \mathbb{R}^n$: the new mutation strength vector

begin

\[ \nu \leftarrow e^{\tau_0}\{\text{Gaussian random number}\} \]
\[ \vec{\sigma} \leftarrow \vec{0} \]

for $i \leftarrow 0$ up to $n - 1$ do

\[ \vec{\sigma}_i \leftarrow \nu \ast e^{\tau}\{\text{Gaussian random number}\} \ast \vec{\sigma}'_i \]

return $\vec{\sigma}$
Reproduction of Endogenous Parameters

- Recombination of strategy parameters $p.w$: intermediate crossover

$\tilde{\sigma} \leftarrow \text{infoMutationES}(\tilde{\sigma}')$

**Input:** $\tilde{\sigma}' \in \mathbb{R}^n$: the old mutation strength vector

**Data:** $i$: a counter variable

**Output:** $\tilde{\sigma} \in \mathbb{R}^n$: the new mutation strength vector

begin

$\nu \leftarrow e^{\tau_0}$ \{Gaussian random number\}

$\tilde{\sigma} \leftarrow 0$

for $i \leftarrow 0$ up to $n - 1$ do

$\tilde{\sigma}_i \leftarrow$

$\nu \ast e^{\tau}$ \{Gaussian random number\} $\ast \tilde{\sigma}'_i$

return $\tilde{\sigma}$

- Mutation of strategy parameters: apply lognormal mutation with

$$\tau_0 = \frac{c}{\sqrt{2n}} \quad (6)$$
Reproduction of Endogenous Parameters

- Recombination of strategy parameters \( p.w \): intermediate crossover

\[
\bar{\sigma} \leftarrow \text{infoMutationES}(\bar{\sigma}')
\]

**Input:** \( \bar{\sigma}' \in \mathbb{R}^n \): the old mutation strength vector

**Data:** \( i \): a counter variable

**Output:** \( \bar{\sigma} \in \mathbb{R}^n \): the new mutation strength vector

**begin**

\[
\begin{align*}
\nu & \leftarrow e^{\tau_0}\{\text{Gaussian random number}\} \\
\bar{\sigma} & \leftarrow \mathbf{0} \\
\text{for } i & \leftarrow 0 \text{ up to } n - 1 \text{ do} \\
& \quad \bar{\sigma}_i \leftarrow \nu \cdot e^{\tau}\{\text{Gaussian random number}\} \cdot \bar{\sigma}'_i \\
\text{return } \bar{\sigma}
\end{align*}
\]

- Mutation of strategy parameters: apply lognormal mutation with

\[
\begin{align*}
\tau_0 &= \frac{c}{\sqrt{2n}} \\
\tau &= \frac{c}{\sqrt{2\sqrt{n}}}
\end{align*}
\]
Reproduction of Endogenous Parameters

- Recombination of strategy parameters $p.w$: intermediate crossover

$\vec{\sigma} \leftarrow \text{infoMutationES}(\vec{\sigma}')$

**Input:** $\vec{\sigma}' \in \mathbb{R}^n$: the old mutation strength vector

**Data:** $i$: a counter variable

**Output:** $\vec{\sigma} \in \mathbb{R}^n$: the new mutation strength vector

**begin**

\[
\nu \leftarrow e^{\tau_0} \{\text{Gaussian random number}\}
\]

\[
\vec{\sigma} \leftarrow \vec{0}
\]

for $i \leftarrow 0 \; \text{up to} \; n - 1$ do

\[
\vec{\sigma}_i \leftarrow \nu \cdot e^{\tau} \{\text{Gaussian random number}\} \cdot \vec{\sigma}'_i
\]

end

return $\vec{\sigma}$

**Mutation of strategy parameters:** apply lognormal mutation with

\[
\tau_0 = \frac{c}{\sqrt{2n}} \quad (6)
\]

\[
\tau = \frac{c}{\sqrt{2\sqrt{n}}} \quad (7)
\]

\[
c = 1 \quad (8)
\]
Reproduction of Endogenous Parameters

- Recombination of strategy parameters $p.w$: intermediate crossover

$$\bar{\sigma} \leftarrow \text{infoMutationES}(\bar{\sigma}')$$

**Input:** $\bar{\sigma}' \in \mathbb{R}^n$: the old mutation strength vector  
**Data:** $i$: a counter variable  
**Output:** $\bar{\sigma} \in \mathbb{R}^n$: the new mutation strength vector

```
begin
  \nu \leftarrow e^{\tau_0}\{\text{Gaussian random number}\}
  \bar{\sigma} \leftarrow 0
  \text{for } i \leftarrow 0 \text{ up to } n - 1 \text{ do}
    \bar{\sigma}_i \leftarrow \nu \cdot e^{\tau}\{\text{Gaussian random number}\} \cdot \bar{\sigma}'_i
  \text{return } \bar{\sigma}
```

- Mutation of strategy parameters: apply lognormal mutation with

$$\tau_0 = \frac{c}{\sqrt{2n}} \quad (6)$$

$$\tau = \frac{c}{\sqrt{2\sqrt{n}}} \quad (7)$$

$$c = 1 \quad (8)$$

$$n \equiv \text{dimension} \quad (9)$$
Section Outline

1. Population Treatment
2. Mutation
3. Self-Adaptation
4. The 1/5th Rule
5. Endogeneous Adaptation
6. Recombination
7. Parameter Reproduction
8. CMA-ES
• Covariance Matrix Adaptation Evolution Strategy (CMA-ES) by Hansen et al. [16–23] in the mid-1990s
Covariance Matrix Adaptation Evolution Strategy (CMA-ES) by Hansen et al. \[16–23\] in the mid-1990s

Extremely powerful optimization method for continuous domains \(\mathbb{R}^n\) \([20, 22–24]\)
- Covariance Matrix Adaptation Evolution Strategy (CMA-ES) by Hansen et al. [16–23] in the mid-1990s
- Extremely powerful optimization method for continuous domains ($\mathbb{R}^n$) [20, 22–24]
- Works well on rugged landscapes with discontinuities, sharp bends or ridges, noise, local optima, outliers
CMA-ES

- Covariance Matrix Adaptation Evolution Strategy (CMA-ES) by Hansen et al. [16–23] in the mid-1990s
- Extremely powerful optimization method for continuous domains ($\mathbb{R}^n$) [20, 22–24]
- Works well on rugged landscapes with discontinuities, sharp bends or ridges, noise, local optima, outliers . . . if a landscape is uni-modal and continuous, one would not need a metaheuristic method anyway)
The new offspring population of each generation is sampled from a multi-variate normal distribution \cite{25}
The new offspring population of each generation is sampled from a multi-variate normal distribution\textsuperscript{[25]} similar to the method D introduced for mutation.
CMA-ES: Principles

\[ \sigma \approx \frac{\sigma_{x,x_1} + \sigma_{x,x_2}}{2} \]

\[ \sigma \approx \frac{\sigma_{x,x_1}}{\sigma_{x,x_1}} \]

\[ C = \begin{pmatrix} \sigma_{x,x_0} & \sigma_{x,x_1} \\ \sigma_{x,x_1} & \sigma_{x,x_2} \end{pmatrix} \]

\[ \begin{array}{c} \text{selected parents} \\ A \end{array} \]

\[ \begin{array}{c} x_1' \\ B \end{array} \]

\[ \begin{array}{c} x_2' \\ C \end{array} \]

\[ \begin{array}{c} x' \\ D \end{array} \]
The new offspring population of each generation is sampled from a multi-variate normal distribution \cite{25} similar to the method \textbf{D} introduced for mutation which allows to represent second-order relationships, i.e., dependencies between variables.
The new offspring population of each generation is sampled from a multi-variate normal distribution\(^{[25]}\) similar to the method D introduced for mutation which allows to represent second-order relationships, i.e., dependencies between variables.

Parents selected via Truncation selection.
The new offspring population of each generation is sampled from a multi-variate normal distribution \cite{25} similar to the method D introduced for mutation which allows to represent second-order relationships, i.e., dependencies between variables.

- Parents selected via Truncation selection
- Of course, we need a matrix $C$ (step-width) of co-variances for the normal distribution and a center (mean $\mu$)
The new offspring population of each generation is sampled from a multi-variate normal distribution \cite{25} similar to the method \textbf{D} introduced for mutation which allows to represent second-order relationships, i.e., dependencies between variables.

- Parents selected via Truncation selection
- Of course, we need a matrix $C$ (step-width) of co-variances for the normal distribution and a center (mean $\mu$)
- Both are exogenous parameters maintained centrally and updated by the algorithm
The new offspring population of each generation is sampled from a multi-variate normal distribution \cite{25} similar to the method introduced for mutation which allows to represent second-order relationships, i.e., dependencies between variables.

- Parents selected via Truncation selection
- Of course, we need a matrix $C$ (step-width) of co-variances for the normal distribution and a center (mean $\mu$)
- Both are exogenous parameters maintained centrally and updated by the algorithm
  - The center $\mu$ of the distribution is set to the weighted average of the selected parent points
CMA-ES: Principles

- The new offspring population of each generation is sampled from a multi-variate normal distribution \[^{25}\] similar to the method \(D\) introduced for mutation which allows to represent second-order relationships, i.e., dependencies between variables.
- Parents selected via Truncation selection.
- Of course, we need a matrix \(C\) (step-width) of co-variances for the normal distribution and a center (mean \(\mu\)).
- Both are exogenous parameters maintained centrally and updated by the algorithm:
  - The center \(\mu\) of the distribution is set to the weighted average of the selected parent points.
  - Covariance matrix is *updated* in each iteration with information from the selected parents.
CMA-ES: Principles

- The new offspring population of each generation is sampled from a multi-variate normal distribution\(^{[25]}\) similar to the method\(D\) introduced for mutation which allows to represent second-order relationships, i.e., dependencies between variables
- Parents selected via Truncation selection
- Of course, we need a matrix\(C\) (step-width) of co-variances for the normal distribution and a center (mean\(\mu\))
- Both are exogenous parameters maintained centrally and updated by the algorithm
  - The center\(\mu\) of the distribution is set to the weighted average of the selected parent points
  - Covariance matrix is\(updated\) in each iteration with information from the selected parents
- New population is sampled from normal distribution, parent individuals are discarded (no traditional mutation/crossover)
• Using co-variance information allows to express relationships between variables
Using co-variance information allows to express relationships between variables (e.g., “in good solutions, large values of $x_1$ require large values of $x_2$”)
Using co-variance information allows to express relationships between variables (e.g., “in good solutions, large values of $x_1$ require large values of $x_2$”)

This leads to rotation invariance: if the objective function is rotated in the search space, CMA-ES will give the same result! [26, 27]
• Using co-variance information allows to express relationships between variables (e.g., “in good solutions, large values of $x_1$ require large values of $x_2$”)

• This leads to rotation invariance: if the objective function is rotated in the search space, CMA-ES will give the same result! [26, 27]

• Generally, CMA-ES is invariant to [27]
• Using co-variance information allows to express relationships between variables (e.g., “in good solutions, large values of $x_1$ require large values of $x_2$”)

• This leads to rotation invariance: if the objective function is rotated in the search space, CMA-ES will give the same result! [26, 27]

• Generally, CMA-ES is invariant to [27]:
  • angle preserving (rigid) transformations of the search space (if the start points are also transformed)
• Using co-variance information allows to express relationships between variables (e.g., “in good solutions, large values of $x_1$ require large values of $x_2$”)

• This leads to rotation invariance: if the objective function is rotated in the search space, CMA-ES will give the same result! [26, 27]

• Generally, CMA-ES is invariant to [27]:
  • angle preserving (rigid) transformations of the search space (if the start points are also transformed), such as
    • rotation
• Using co-variance information allows to express relationships between variables (e.g., “in good solutions, large values of $x_1$ require large values of $x_2$”)

• This leads to rotation invariance: if the objective function is rotated in the search space, CMA-ES will give the same result! [26, 27]

• Generally, CMA-ES is invariant to [27]:
  • angle preserving (rigid) transformations of the search space (if the start points are also transformed), such as
    • rotation
    • reflection, and
• Using co-variance information allows to express relationships between variables (e.g., “in good solutions, large values of $x_1$ require large values of $x_2$”)

• This leads to rotation invariance: if the objective function is rotated in the search space, CMA-ES will give the same result! [26, 27]

• Generally, CMA-ES is invariant to [27]:
  • angle preserving (rigid) transformations of the search space (if the start points are also transformed), such as
    • rotation
    • reflection, and
    • translation
CMA-ES: Invariances

- Using co-variance information allows to express relationships between variables (e.g., “in good solutions, large values of $x_1$ require large values of $x_2$”)
- This leads to rotation invariance: if the objective function is rotated in the search space, CMA-ES will give the same result! [26, 27]
- Generally, CMA-ES is invariant to [27]:
  - angle preserving (rigid) transformations of the search space (if the start points are also transformed)
  - order preserving (i.e., strictly monotonic) transformations of the objective function value
CMA-ES: Invariances

- Using co-variance information allows to express relationships between variables (e.g., “in good solutions, large values of $x_1$ require large values of $x_2$”)

- This leads to rotation invariance: if the objective function is rotated in the search space, CMA-ES will give the same result! [26, 27]

- Generally, CMA-ES is invariant to [27]:
  - angle preserving (rigid) transformations of the search space (if the start points are also transformed)
  - order preserving (i.e., strictly monotonic) transformations of the objective function value (i.e., $||x||^2$ and $3||x||^{0.2} - 100$ lead to same result!)
Using co-variance information allows to express relationships between variables (e.g., “in good solutions, large values of $x_1$ require large values of $x_2$”)

This leads to rotation invariance: if the objective function is rotated in the search space, CMA-ES will give the same result! [26, 27]

Generally, CMA-ES is invariant to [27]:
- angle preserving (rigid) transformations of the search space (if the start points are also transformed)
- order preserving (i.e., strictly monotonic) transformations of the objective function value

We want this! (Remember: Roulette-Wheel Selection versus Truncation Selection)
- Algorithm itself is a bit more complicated and will not be discussed here
• Algorithm itself is a bit more complicated and will not be discussed here
• But: There exist many excellent, ready-to-use, industrial-strength, open-source implementations
• Algorithm itself is a bit more complicated and will not be discussed here
• But: There exist many excellent, ready-to-use, industrial-strength, open-source implementations, e.g.,
• Algorithm itself is a bit more complicated and will not be discussed here
• But: There exist many excellent, ready-to-use, industrial-strength, open-source implementations, e.g.,
  • ESKit\textsuperscript{[28]}: http://www.marmakoide.org/content/code/eskit.html
- Algorithm itself is a bit more complicated and will not be discussed here
- But: There exist many excellent, ready-to-use, industrial-strength, open-source implementations, e.g.,
  - ESKit\textsuperscript{[28]}: http://www.marmakoide.org/content/code/eskit.html
  - Official source code page:
    http://www.lri.fr/~hansen/cmaes_inmatlab.html
- Algorithm itself is a bit more complicated and will not be discussed here
- But: There exist many excellent, ready-to-use, industrial-strength, open-source implementations, e.g.,
  - ESKit\textsuperscript{[28]}: http://www.marmakoide.org/content/code/eskit.html
- Much literature\textsuperscript{[16–23, 25]} and comprehensive benchmark results\textsuperscript{[20, 22–24]}
• Algorithm itself is a bit more complicated and will not be discussed here
• But: There exist many excellent, ready-to-use, industrial-strength, open-source implementations, e.g.,
  • ESKit\textsuperscript{[28]}: \url{http://www.marmakoide.org/content/code/eskit.html}
  • Official source code page:
    \url{http://www.lri.fr/~hansen/cmaes_inmatlab.html}
• Much literature\textsuperscript{[16–23, 25]} and comprehensive benchmark results\textsuperscript{[20, 22–24]}
• A dedicated website: \url{http://www.lri.fr/~hansen/cmaesintro.html}
• Algorithm itself is a bit more complicated and will not be discussed here

• But: There exist many excellent, ready-to-use, industrial-strength, open-source implementations, e.g.,
  • ESKit\textsuperscript{[28]}: http://www.marmakoide.org/content/code/eskit.html
  • Official source code page:
    http://www.lri.fr/~hansen/cmaes_inmatlab.html

• Much literature\textsuperscript{[16–23, 25]} and comprehensive benchmark results\textsuperscript{[20, 22–24]}

• A dedicated website: http://www.lri.fr/~hansen/cmaesintro.html

• An own Wikipedia\textsuperscript{[29]} page: https://en.wikipedia.org/wiki/CMA-ES
Algorithm itself is a bit more complicated and will not be discussed here.

But: There exist many excellent, ready-to-use, industrial-strength, open-source implementations, e.g.,

- ESKit\textsuperscript{[28]}: \url{http://www.marmakoide.org/content/code/eskit.html}
- Official source code page: \url{http://www.lri.fr/~hansen/cmaes_inmatlab.html}

- Much literature\textsuperscript{[16–23, 25]} and comprehensive benchmark results\textsuperscript{[20, 22–24]}
- A dedicated website: \url{http://www.lri.fr/~hansen/cmaesintro.html}
- A tutorial: \textsuperscript{[25]}
Algorithm itself is a bit more complicated and will not be discussed here.

But: There exist many excellent, ready-to-use, industrial-strength, open-source implementations, e.g.,

- ESKit \cite{eskit}: http://www.marmakoide.org/content/code/eskit.html

- Much literature \cite{16-23, 25} and comprehensive benchmark results \cite{20, 22-24}
- A dedicated website: http://www.lri.fr/~hansen/cmaesintro.html
- An own Wikipedia \cite{29} page: https://en.wikipedia.org/wiki/CMA-ES
- A tutorial: \cite{25}

When doing numerical optimization, this is the way to go!
Section Outline

1. Population Treatment
2. Mutation
3. Self-Adaptation
4. The 1/5th Rule
5. Endogeneous Adaptation
6. Recombination
7. Parameter Reproduction
8. CMA-ES

Metaheuristic Optimization

Thomas Weise
• Evolution Strategies are Evolutionary Algorithms for (usually) \( \mathbb{G} \subseteq \mathbb{R}^n \)
• Evolution Strategies are Evolutionary Algorithms for (usually) \( G \subseteq \mathbb{R}^n \)
• Population handling: \((\mu/\rho + \lambda)\) or \((\mu/\rho, \lambda)\)
• Evolution Strategies are Evolutionary Algorithms for (usually) $\mathcal{G} \subseteq \mathbb{R}^n$

• Population handling: $(\mu/\rho + \lambda)$ or $(\mu/\rho, \lambda)$

• Mutation step strength: single value, vector, or matrix
• Evolution Strategies are Evolutionary Algorithms for (usually) \( G \subseteq \mathbb{R}^n \)
• Population handling: \( (\mu/\rho + \lambda) \) or \( (\mu/\rho, \lambda) \)
• Mutation step strength: single value, vector, or matrix
• Self-adaptive: adapt to the current phase of the search
- Evolution Strategies are Evolutionary Algorithms for (usually) \( G \subseteq \mathbb{R}^n \)
- Population handling: \( (\mu/\rho + \lambda) \) or \( (\mu/\rho, \lambda) \)
- Mutation step strength: single value, vector, or matrix
- Self-adaptive: adapt to the current phase of the search
- Search operator strength is “learned” during optimization
Summary

- Evolution Strategies are Evolutionary Algorithms for (usually) $G \subseteq \mathbb{R}^n$
- Population handling: $({\mu}/\rho + \lambda)$ or $({\mu}/\rho, \lambda)$
- Mutation step strength: single value, vector, or matrix
- Self-adaptive: adapt to the current phase of the search
- Search operator strength is “learned” during optimization
- Exogeneous method: central parameters, based on success, e.g., 1/5th rule
• Evolution Strategies are Evolutionary Algorithms for (usually) $\mathbb{G} \subseteq \mathbb{R}^n$
• Population handling: $(\mu/\rho + \lambda)$ or $(\mu/\rho, \lambda)$
• Mutation step strength: single value, vector, or matrix
• Self-adaptive: adapt to the current phase of the search
• Search operator strength is “learned” during optimization
• Exogeneous method: central parameters, based on success, e.g., 1/5th rule
• Endogenous method: information encoded in individuals
• Evolution Strategies are Evolutionary Algorithms for (usually) $\mathbb{G} \subseteq \mathbb{R}^n$
• Population handling: $(\mu/\rho + \lambda)$ or $(\mu/\rho, \lambda)$
• Mutation step strength: single value, vector, or matrix
• Self-adaptive: adapt to the current phase of the search
• Search operator strength is “learned” during optimization
• Exogeneous method: central parameters, based on success, e.g., 1/5th rule
• Endogenous method: information encoded in individuals
• Intermediate crossover, Dominant crossover
Evolution Strategies are Evolutionary Algorithms for (usually) $G \subseteq \mathbb{R}^n$

Population handling: $(\mu/\rho + \lambda)$ or $(\mu/\rho, \lambda)$

Mutation step strength: single value, vector, or matrix

Self-adaptive: adapt to the current phase of the search

Search operator strength is “learned” during optimization

Exogeneous method: central parameters, based on success, e.g., 1/5th rule

Endogenous method: information encoded in individuals

Intermediate crossover, Dominant crossover

Log-Normal Parameter Mutation
• Evolution Strategies are Evolutionary Algorithms for (usually) $G \subseteq \mathbb{R}^n$
• Population handling: $(\mu/\rho + \lambda)$ or $(\mu/\rho, \lambda)$
• Mutation step strength: single value, vector, or matrix
• Self-adaptive: adapt to the current phase of the search
• Search operator strength is “learned” during optimization
• Exogeneous method: central parameters, based on success, e.g., 1/5th rule
• Endogenous method: information encoded in individuals
• Intermediate crossover, Dominant crossover
• Log-Normal Parameter Mutation
• CMA-ES: Powerful Tool!!!
谢谢

Thank you

Thomas Weise [汤卫思]
tweise@hfuu.edu.cn
http://iao.hfuu.edu.cn

Hefei University, South Campus 2
Institute of Applied Optimization
Shushan District, Hefei, Anhui,
China


