合肥學院
HEFEI UNIVERSITY

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

website

Introduction

- Evolutionary Algorithm for numerical optimization

Introduction

- Evolutionary Algorithm for numerical optimization
- Search space: vectors of real numbers, i.e., $\mathbb{G} = \mathbb{R}^n$

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]

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

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]...)

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

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

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

Section Outline

Population Treatment

- Evolution Strategys use truncation selection – we alreay discussed that in the GA lecture

Population Treatment

- Evolution Strategys use truncation selection – we alreay discussed that in the GA lecture
- Two population size-related parameters

Population Treatment

- Evolution Strategys use truncation selection – we alreay discussed that in the GA lecture
- Two population size-related parameters:
  1. λ number of offsprings

Population Treatment

- Evolution Strategys use truncation selection – we alreay discussed that in the GA lecture
- Two population size-related parameters:
  1. λ number of offsprings
  2. μ size of the mating pool

Population Treatment

- Evolution Strategys use truncation selection – we alreay discussed that in the GA lecture
- Two population size-related parameters:
  1. λ number of offsprings
  2. μ size of the mating pool
- $(\mu + \lambda)$-algorithm

Population Treatment

- Evolution Strategys use truncation selection – we alreay discussed that in the GA lecture
- Two population size-related parameters:
  1. λ number of offsprings
  2. μ size of the mating pool
- $(\mu + \lambda)$-algorithm:
  - λ offsprings are created from μ parents

Population Treatment

- Evolution Strategys use truncation selection – we alreay discussed that in the GA lecture
- Two population size-related parameters:
  1. λ number of offsprings
  2. μ size of the mating pool
- $(\mu + \lambda)$-algorithm:
  - λ offsprings are created from μ parents
  - the set of λ offsprings and μ parents forms the new population of size $ps = \mu + \lambda$

Population Treatment

- Evolution Strategys use truncation selection – we alreay discussed that in the GA lecture
- Two population size-related parameters:
  1. λ number of offsprings
  2. μ size of the mating pool
- $(\mu + \lambda)$-algorithm:
  - λ offsprings are created from μ parents
  - the set of λ offsprings and μ parents forms the new population of size $ps = \mu + \lambda$
  - only the μ fittest among these ps individuals survive

Population Treatment

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

Population Treatment

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

Population Treatment

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

Population Treatment

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

Population Treatment

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

Population Treatment

- $(\mu + \lambda)$-algorithm
- (μ, λ)-algorithm

Population Treatment

- $(\mu + \lambda)$-algorithm
- (μ, λ)-algorithm
- $(1 + 1)$-ES: ?

Population Treatment

- $(\mu + \lambda)$-algorithm
- (μ, λ)-algorithm
- $(1 + 1)$-ES: Hill Climbing

Population Treatment

- $(\mu + \lambda)$-algorithm
- (μ, λ)-algorithm
- $(1 + 1)$-ES: Hill Climbing
- $(1, 1)$-ES: ?

Population Treatment

- $(\mu + \lambda)$-algorithm
- (μ, λ)-algorithm
- $(1 + 1)$-ES: Hill Climbing
- $(1, 1)$-ES: Random Walk

Population Treatment

- $(\mu + \lambda)$-algorithm
- (μ, λ)-algorithm
- $(1 + 1)$-ES: Hill Climbing
- $(1, 1)$-ES: Random Walk
- $(\infty \overset{+}{,} *)$-ES: ?

Population Treatment

- $(\mu + \lambda)$-algorithm
- (μ, λ)-algorithm
- $(1 + 1)$-ES: Hill Climbing
- $(1, 1)$-ES: Random Walk
- $(\infty \overset{+}{,} *)$-ES: Random Sampling

Population Treatment

- $(\mu + \lambda)$-algorithm
- (μ, λ)-algorithm
- $(1 + 1)$-ES: Hill Climbing
- $(1, 1)$-ES: Random Walk
- $(\infty \overset{+}{,} *)$-ES: Random Sampling
- $(\mu/\rho + \lambda)$

Population Treatment

- $(\mu + \lambda)$-algorithm
- (μ, λ)-algorithm
- $(1 + 1)$-ES: Hill Climbing
- $(1, 1)$-ES: Random Walk
- $(\infty \overset{+}{,} *)$-ES: Random Sampling
- $(\mu/\rho + \lambda)$:
  - $(\mu + \lambda)$ Evolution Strategy with ρ-ary reproduction operation

Population Treatment

- $(\mu + \lambda)$-algorithm
- (μ, λ)-algorithm
- $(1 + 1)$-ES: Hill Climbing
- $(1, 1)$-ES: Random Walk
- $(\infty \overset{+}{,} *)$-ES: Random Sampling
- $(\mu/\rho + \lambda)$:
  - $(\mu + \lambda)$ Evolution Strategy with ρ-ary reproduction operation
  - $\rho =$ number of parents per offspring

Population Treatment

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

Population Treatment

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

Population Treatment

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

Population Treatment

- $(\mu + \lambda)$-algorithm
- (μ, λ)-algorithm
- $(1 + 1)$-ES: Hill Climbing
- $(1, 1)$-ES: Random Walk
- $(\infty \overset{+}{,} *)$-ES: Random Sampling
- $(\mu/\rho + \lambda)$:
  - $(\mu + \lambda)$ Evolution Strategy with ρ-ary reproduction operation
  - $\rho =$ number of parents per offspring
  - Default: $\rho = 1$ (mutation only); $\rho = 2$ (crossover)
- $(\mu/\rho, \lambda)$:
  - (μ, λ) Evolution Strategy with ρ-ary reproduction operation
  - from the λ offspring, only the μ best ones survive

Selection

- Only the μ fittest individuals survive, in both $(\mu + \lambda)$ and (μ, λ) ES

Selection

- Only the μ fittest individuals survive, in both $(\mu + \lambda)$ and (μ, λ) ES
- Simple, deterministic selection algorithm: $\mathrm{truncationSelection}$

Selection

- Only the μ fittest individuals survive, in both $(\mu + \lambda)$ and (μ, λ) ES
- Simple, deterministic selection algorithm: $\mathrm{truncationSelection}$
- Already discussed in the GA lecture

Truncation Selection

matePool $\longleftarrow$ truncationSelection(μ, pop)

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

Truncation Selection

Listing: The Truncation Selection Algorithm

```
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

Mutation

- Mutation and selection are the main operations that drive the ES

Mutation

- Mutation and selection are the main operations that drive the ES
- Most common ES: $(1+1)$, i.e., there is no recombination

Mutation

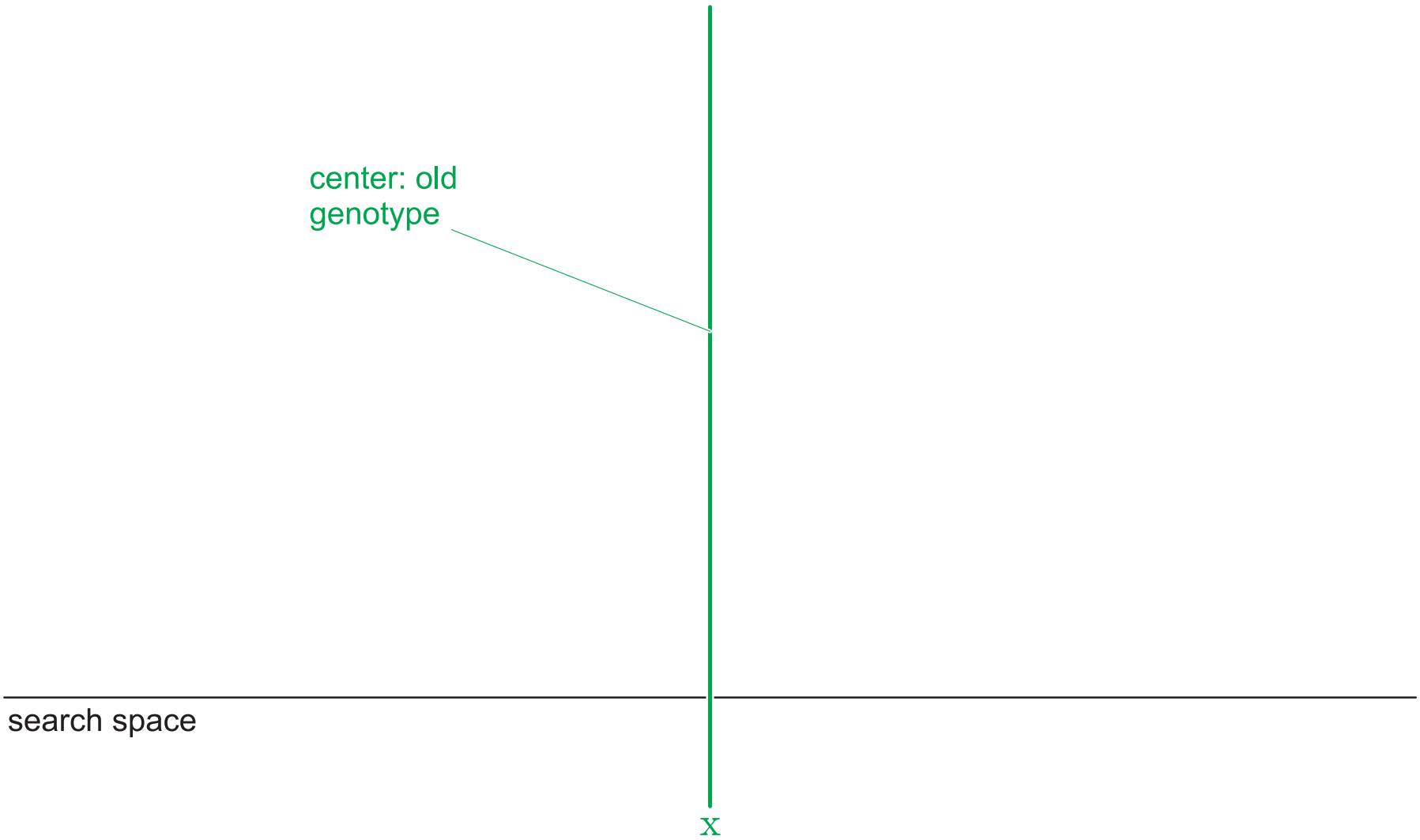
- Mutation and selection are the main operations that drive the ES
- Most common ES: $(1+1)$, i.e., there is no recombination
- For simplicity, assume that search space $\mathbb{G}$ and solution space $\mathbb{X}$ are the same (i.e., we do not need a GPM)

Mutation

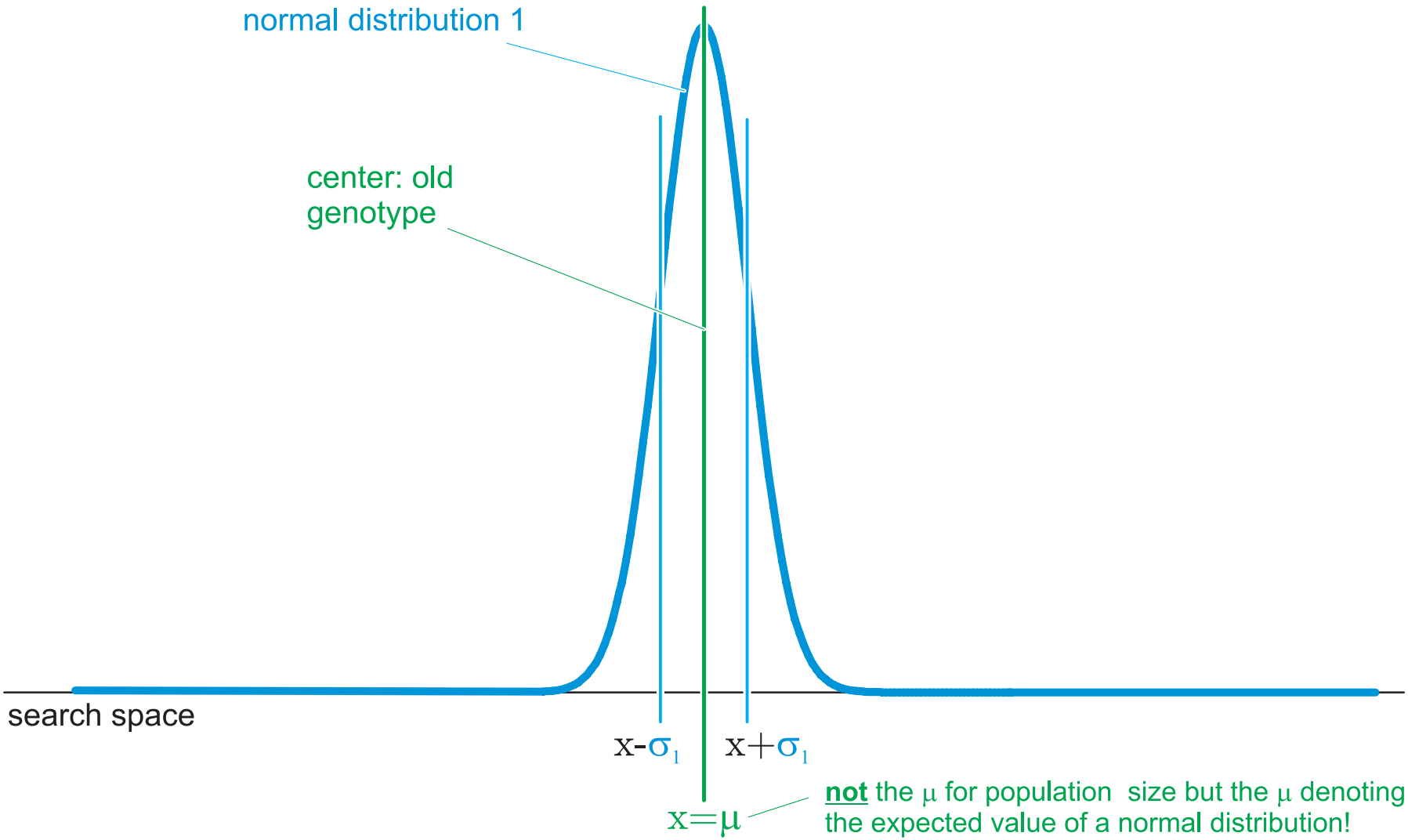
- Mutation and selection are the main operations that drive the ES
- Most common ES: $(1+1)$, i.e., there is no recombination
- For simplicity, assume that search space $\mathbb{G}$ and solution space $\mathbb{X}$ are the same (i.e., we do not need a GPM)
- Let us assume the search space is the real numbers, i.e., $\mathbb{X} = \mathbb{G} = \mathbb{R}$.

Mutation

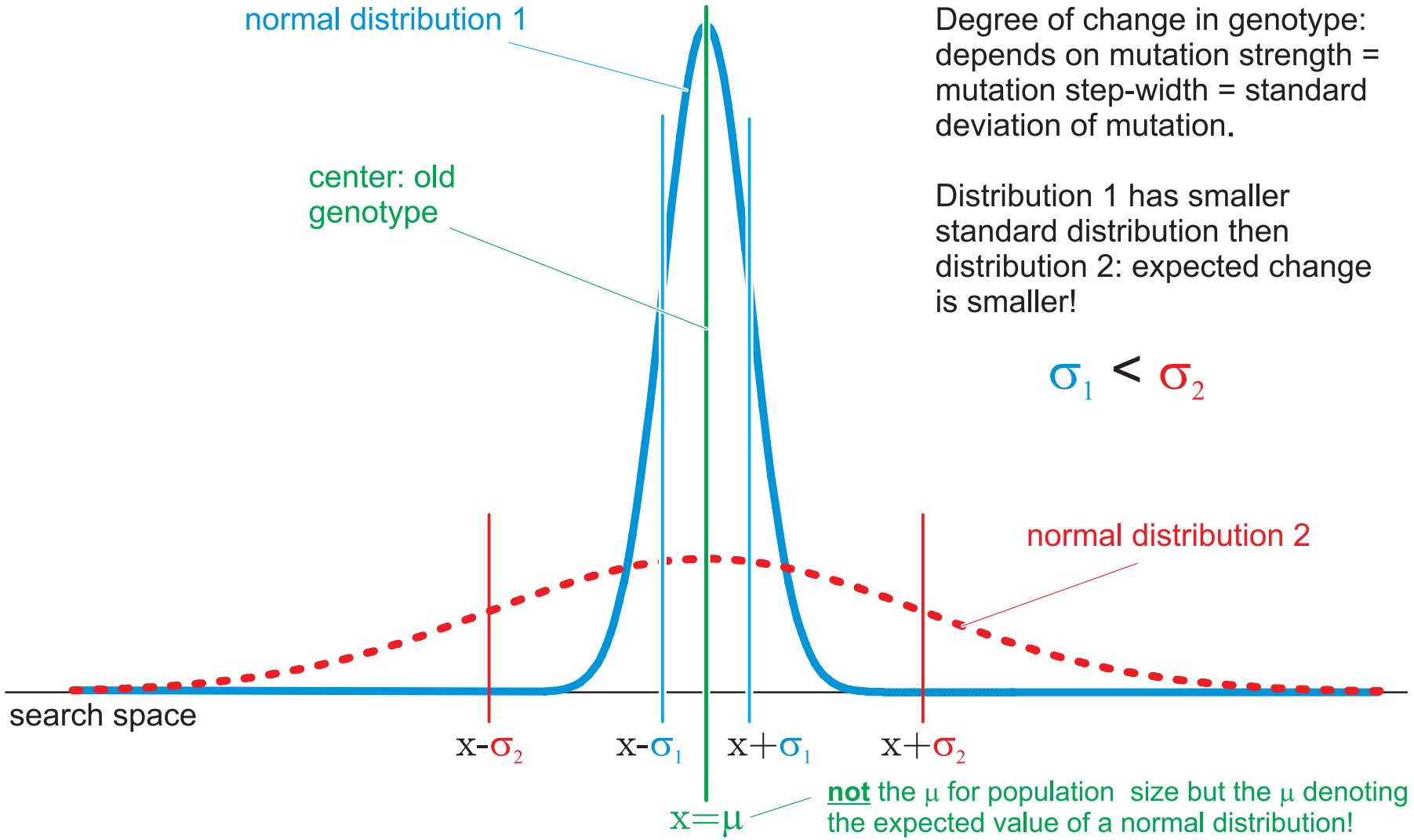
Mutation



Mutation


normal distribution 1
center: old genotype
search space
x-σ1
x+σ1
x=μ
not the μ for population size but the μ denoting the expected value of a normal distribution!

Mutation



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!

$$\sigma_1 < \sigma_2$$

Mutation

- 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 $\mathbb{G}$ and solution space $\mathbb{X}$ are the same (i.e., we do not need a GPM)
- Let us assume the search space is the real numbers, i.e., $\mathbb{X} = \mathbb{G} = \mathbb{R}$.
- 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 σ of normal distribution as step length

Objective Function

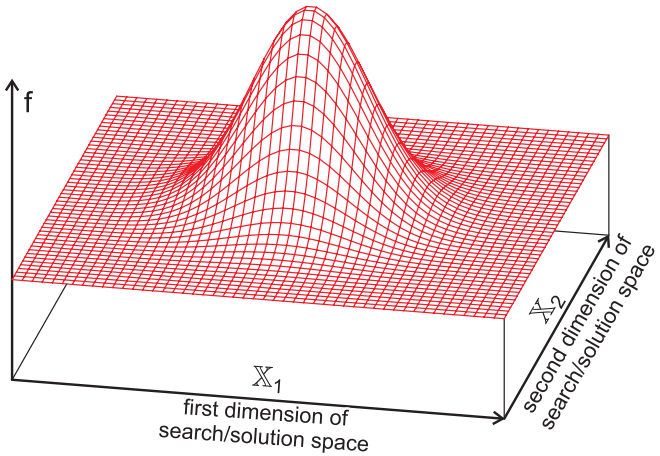
- The objective function is usually n-dimensional and may have different characteristics

Objective Function

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

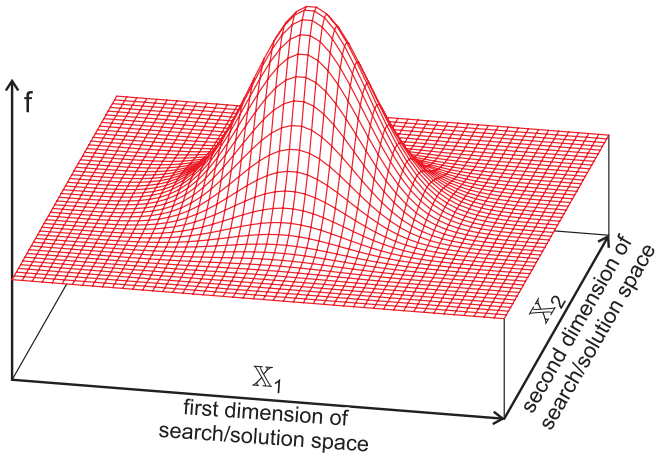
Objective Function

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



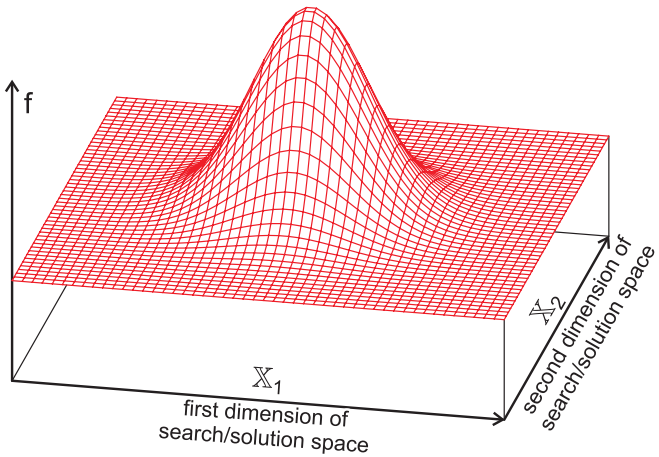
Objective Function

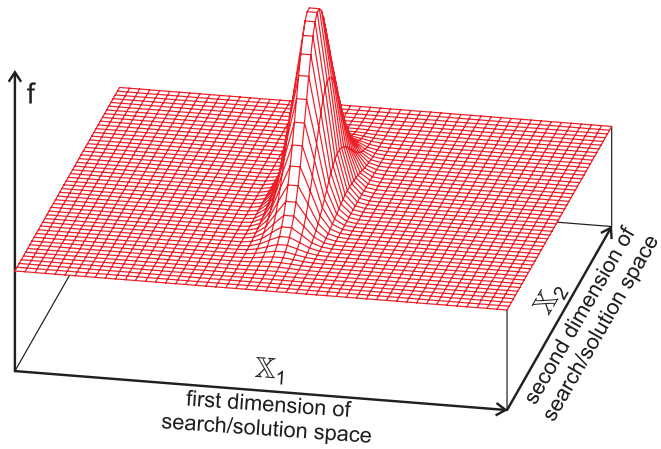
- The objective function is usually n-dimensional and may have different characteristics:
  - It could be nice: symmetric and axis-parallel
  - Maybe it is ill-defined (values alone one axis have much higher impact) and axis-parallel



Objective Function

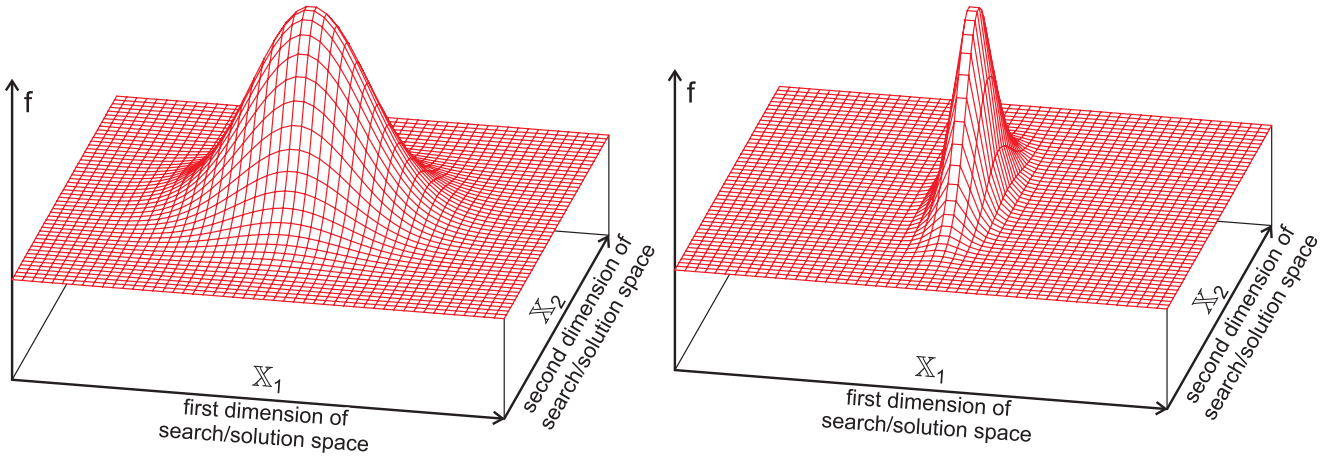
- The objective function is usually n-dimensional and may have different characteristics:
  - It could be nice: symmetric and axis-parallel
  - Maybe it is ill-defined (values alone one axis have much higher impact) and axis-parallel





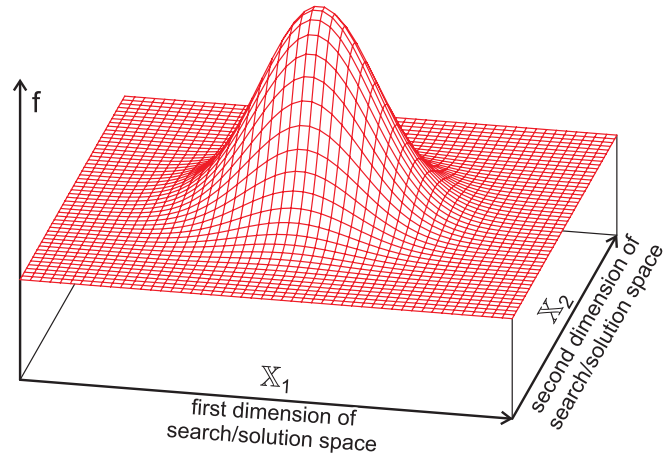
Objective Function

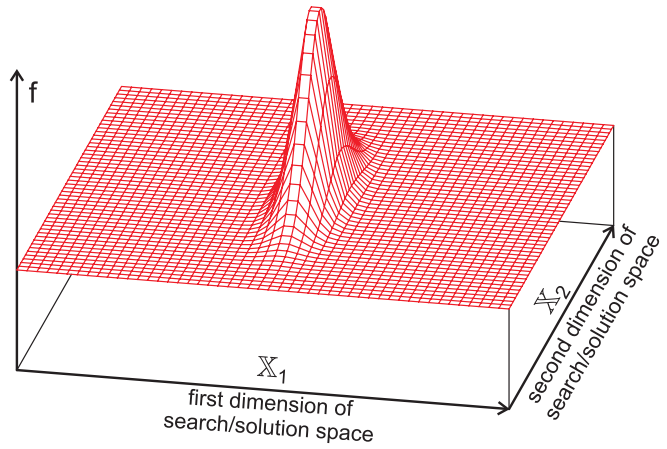
- The objective function is usually n-dimensional and may have different characteristics:
  - It could be nice: symmetric and axis-parallel
  - Maybe it is ill-defined (values alone one axis have much higher impact) and axis-parallel
  - Maybe it is both ill-defined and not axis-parallel

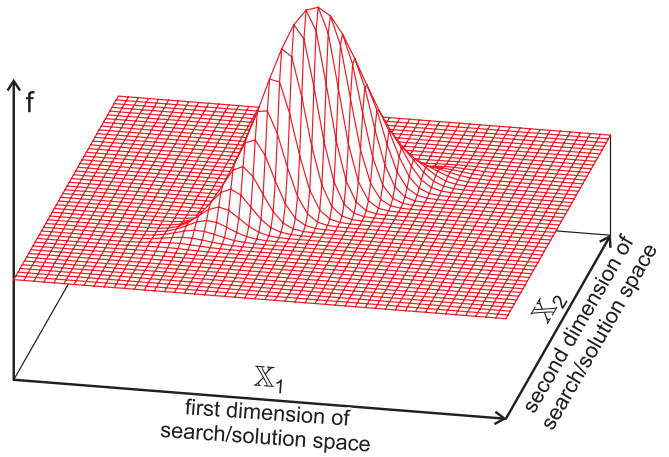


Objective Function

- 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

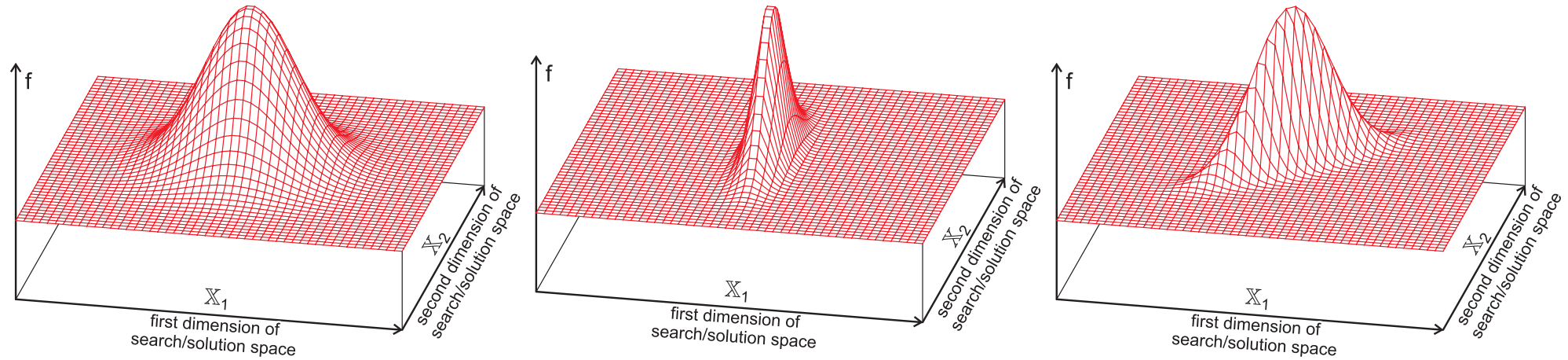






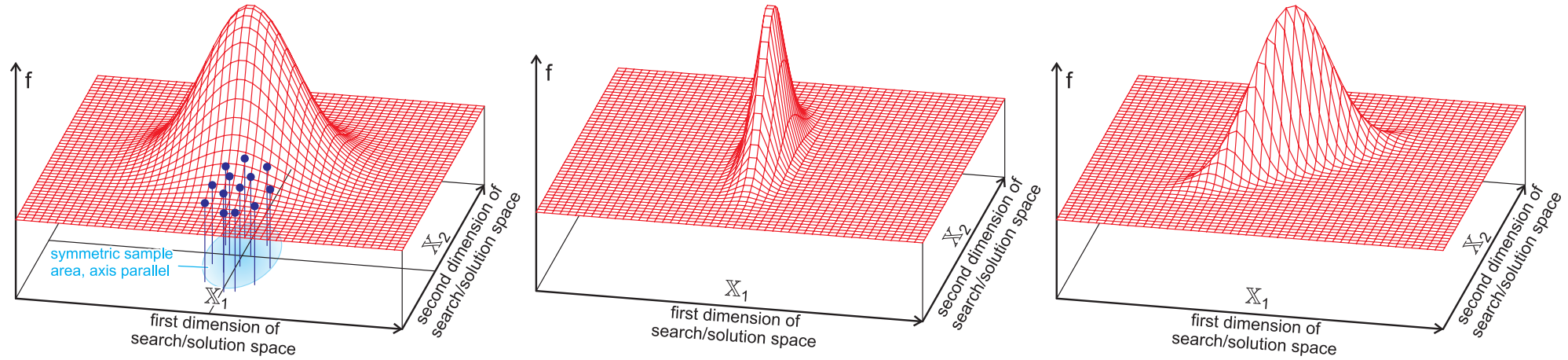
Objective Function

- 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!



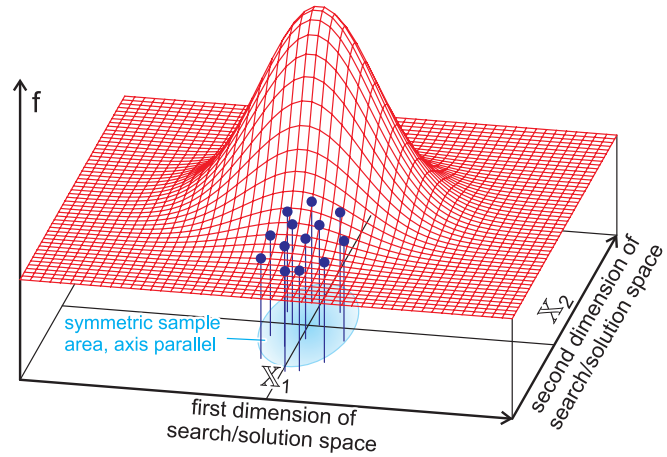
Objective Function

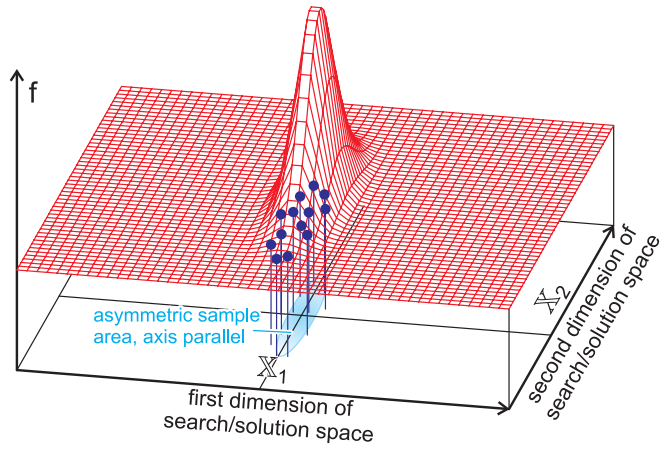
- The objective function is usually n-dimensional and may have different characteristics:
  - It could be nice: symmetric and axis-parallel
  - Maybe it is ill-defined (values alone one axis have much higher impact) and axis-parallel
  - Maybe it is both ill-defined and not axis-parallel
- We want to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

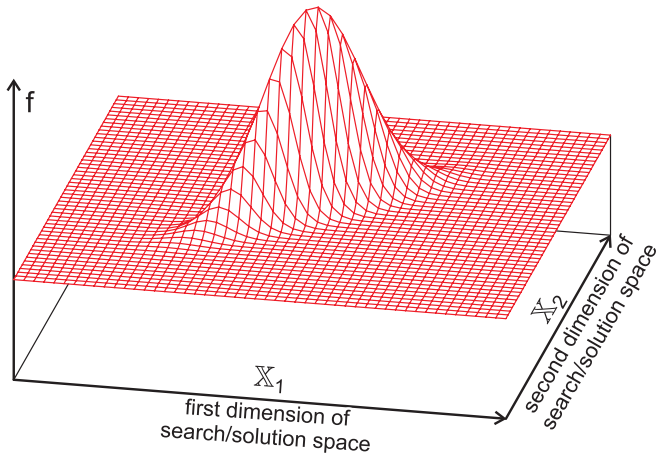


Objective Function

- 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!

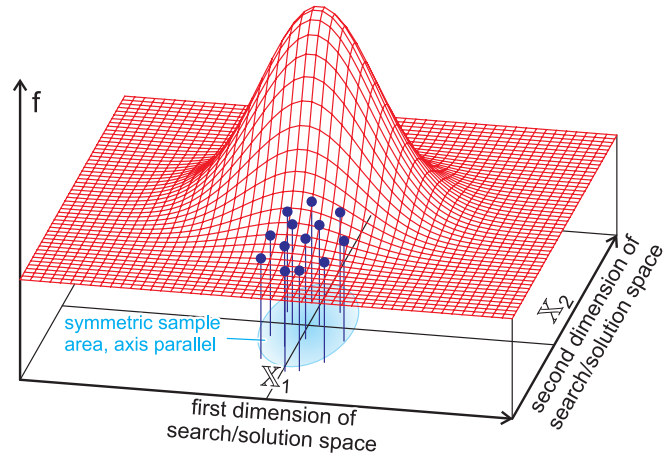


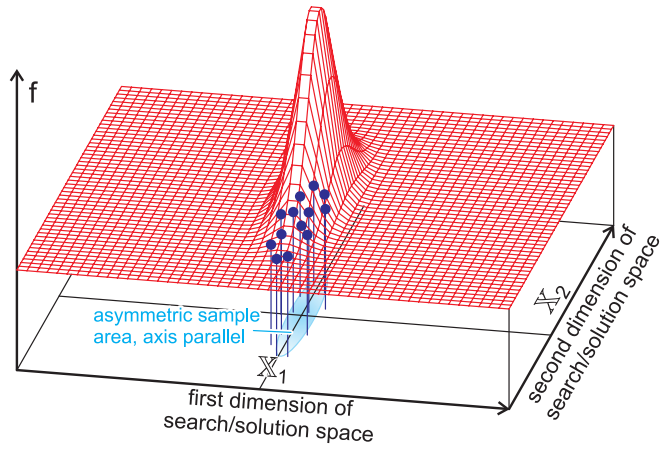


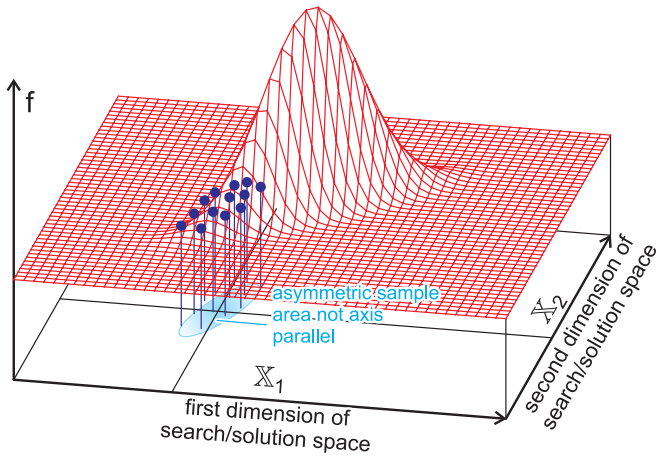


Objective Function

- 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!

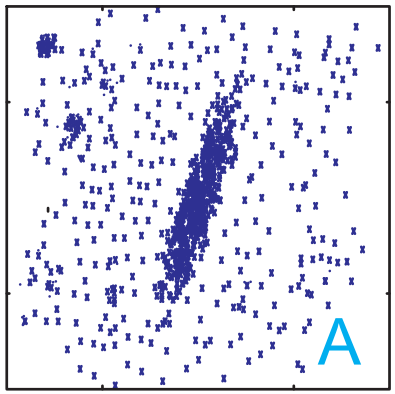






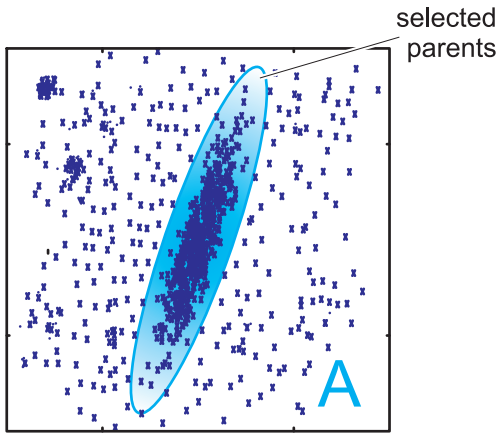
Mutation Operations B-C

- Let's look at the structure of a given population in a 2D-search space $\mathbb{X} \subseteq \mathbb{R}^2$



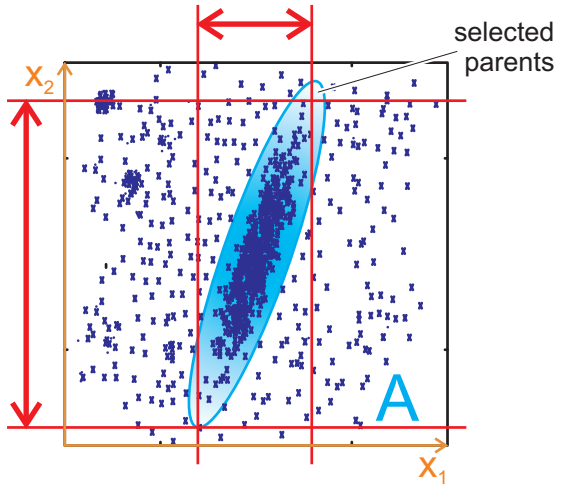
Mutation Operations B-C

- Let's look at the structure of a given population in a 2D-search space $\mathbb{X} \subseteq \mathbb{R}^2$
- Assume that the individuals in the blue sphere are selected, as they are the best ones for some reason



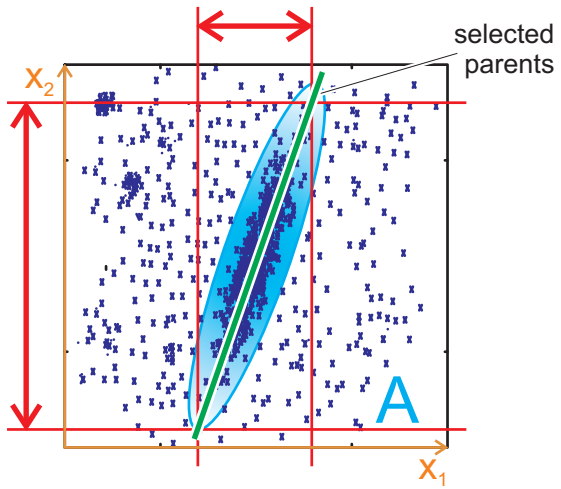
Mutation Operations B-C

- Let's look at the structure of a given population in a 2D-search space $\mathbb{X} \subseteq \mathbb{R}^2$
- 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



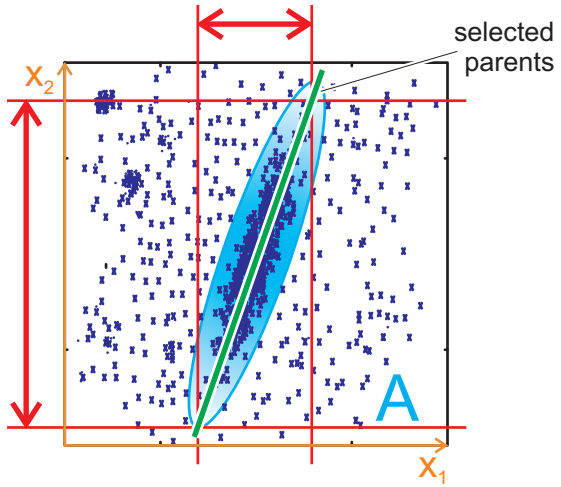
Mutation Operations B-C

- Let's look at the structure of a given population in a 2D-search space $\mathbb{X} \subseteq \mathbb{R}^2$
- 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



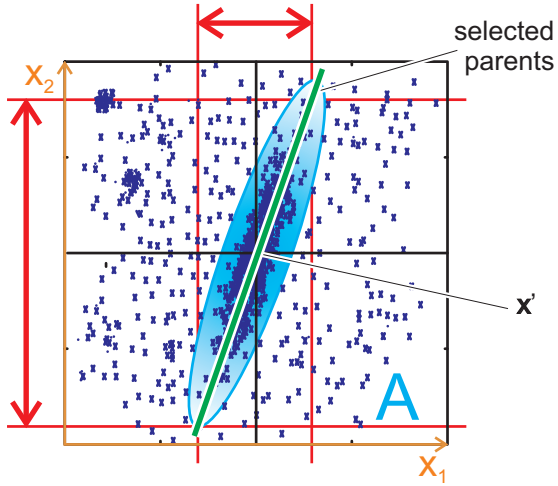
Mutation Operations B-C

- 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



Mutation Operations B-C

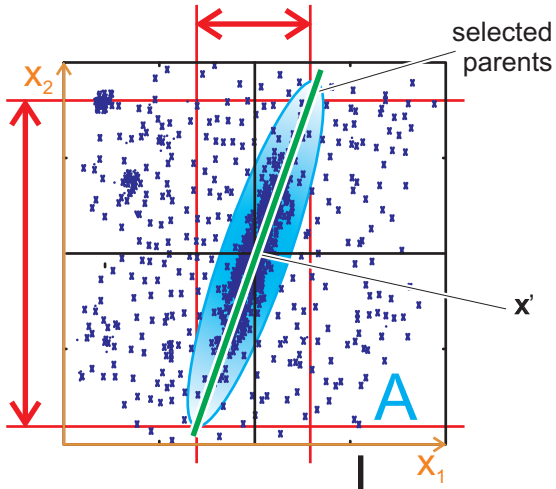
- Let's look at the structure of a given population in a 2D-search space $\mathbb{X} \subseteq \mathbb{R}^2$
- It would be cool if the (normal) distribution of possible offspring could also have such features
- Say we want to mutate the solution $\vec{x'}$ in the middle and create a *cloud* of n offspring points from it...



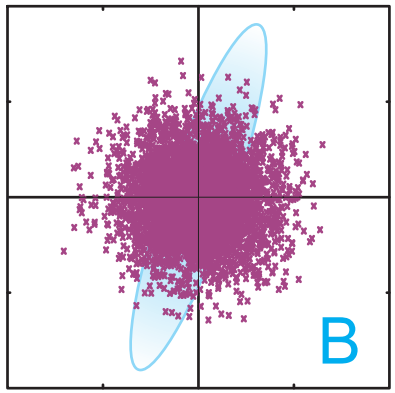
Mutation Operations B-C

- Let's look at the structure of a given population in a 2D-search space $\mathbb{X} \subseteq \mathbb{R}^2$
- Say we want to mutate the solution $\vec{x'}$ in the middle and create a *cloud* of n offspring points from it...

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



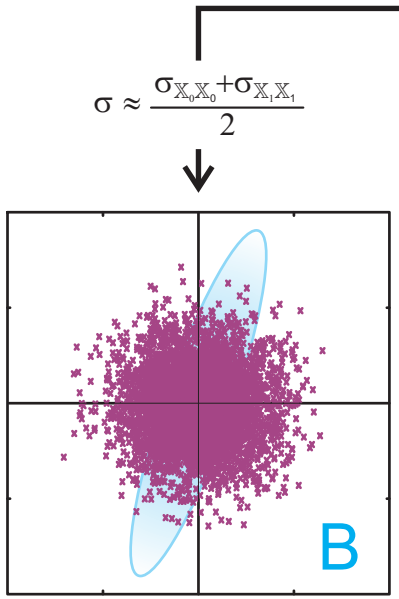
$$\sigma \approx \frac{\sigma_{\mathbb{X}_0\mathbb{X}_0} + \sigma_{\mathbb{X}_1\mathbb{X}_1}}{2}$$

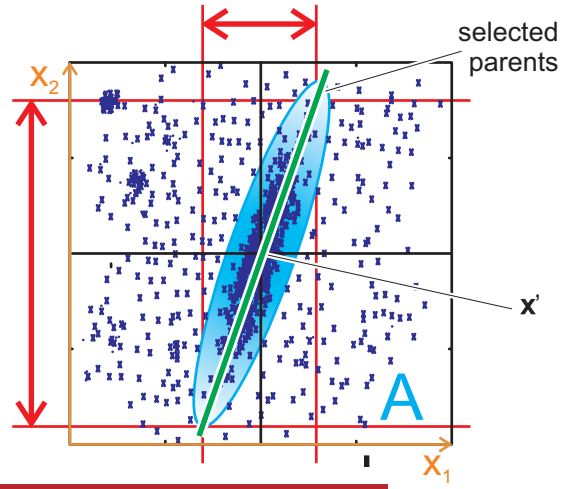


Mutation Operations B-C

- Let's look at the structure of a given population in a 2D-search space $\mathbb{X} \subseteq \mathbb{R}^2$
- Say we want to mutate the solution $\vec{x'}$ in the middle and create a *cloud* of n offspring points from it...

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



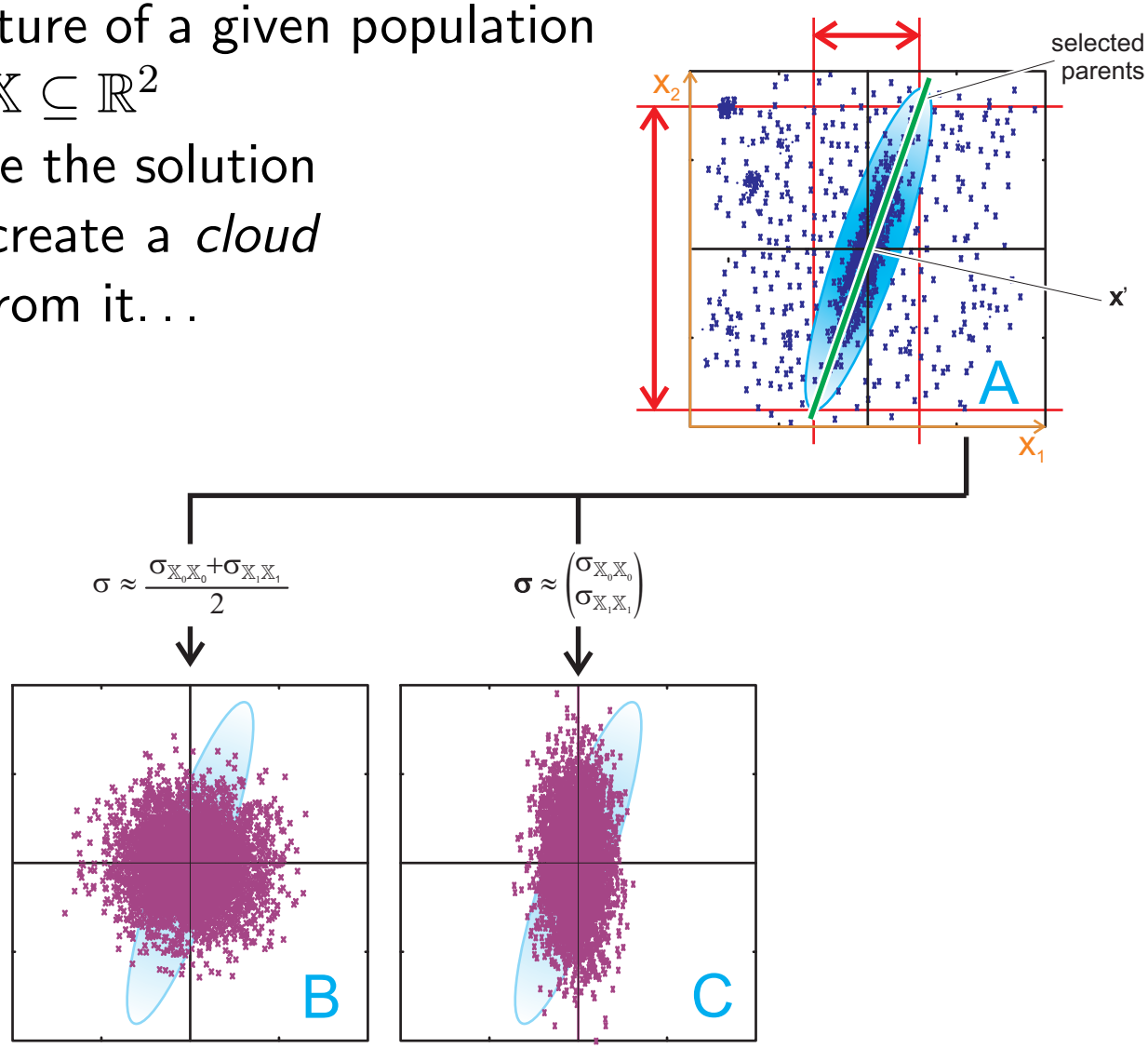


Many solutions generated outside the interesting range.

Mutation Operations B-C

- Let's look at the structure of a given population in a 2D-search space $\mathbb{X} \subseteq \mathbb{R}^2$
- Say we want to mutate the solution $\vec{x'}$ in the middle and create a *cloud* of n offspring points from it...

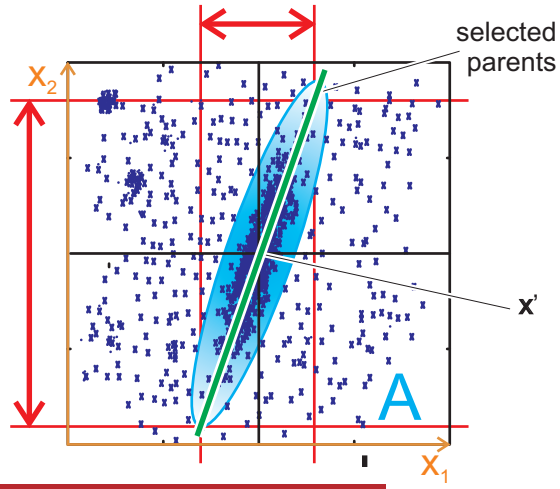
C if we mutate each dimension with a separate standard deviation (i.e., use a vector σ), we can get an elliptic cloud of points (the iso-probability lines form ellipses)



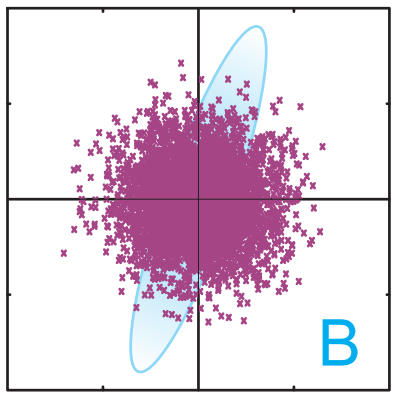
Mutation Operations B-C

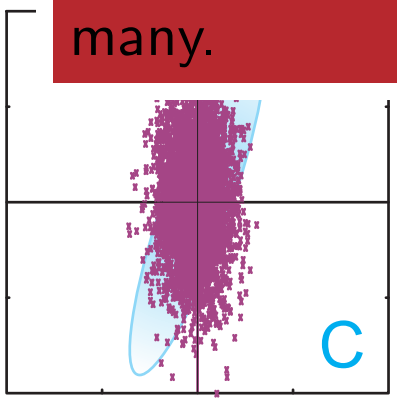
- Let's look at the structure of a given population in a 2D-search space $\mathbb{X} \subseteq \mathbb{R}^2$
- Say we want to mutate the solution $\vec{x'}$ in the middle and create a *cloud* of n offspring points from it...

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



$$\sigma \approx \frac{\sigma_{\mathbb{X}_0\mathbb{X}_0} + \sigma_{\mathbb{X}_1\mathbb{X}_1}}{2}$$



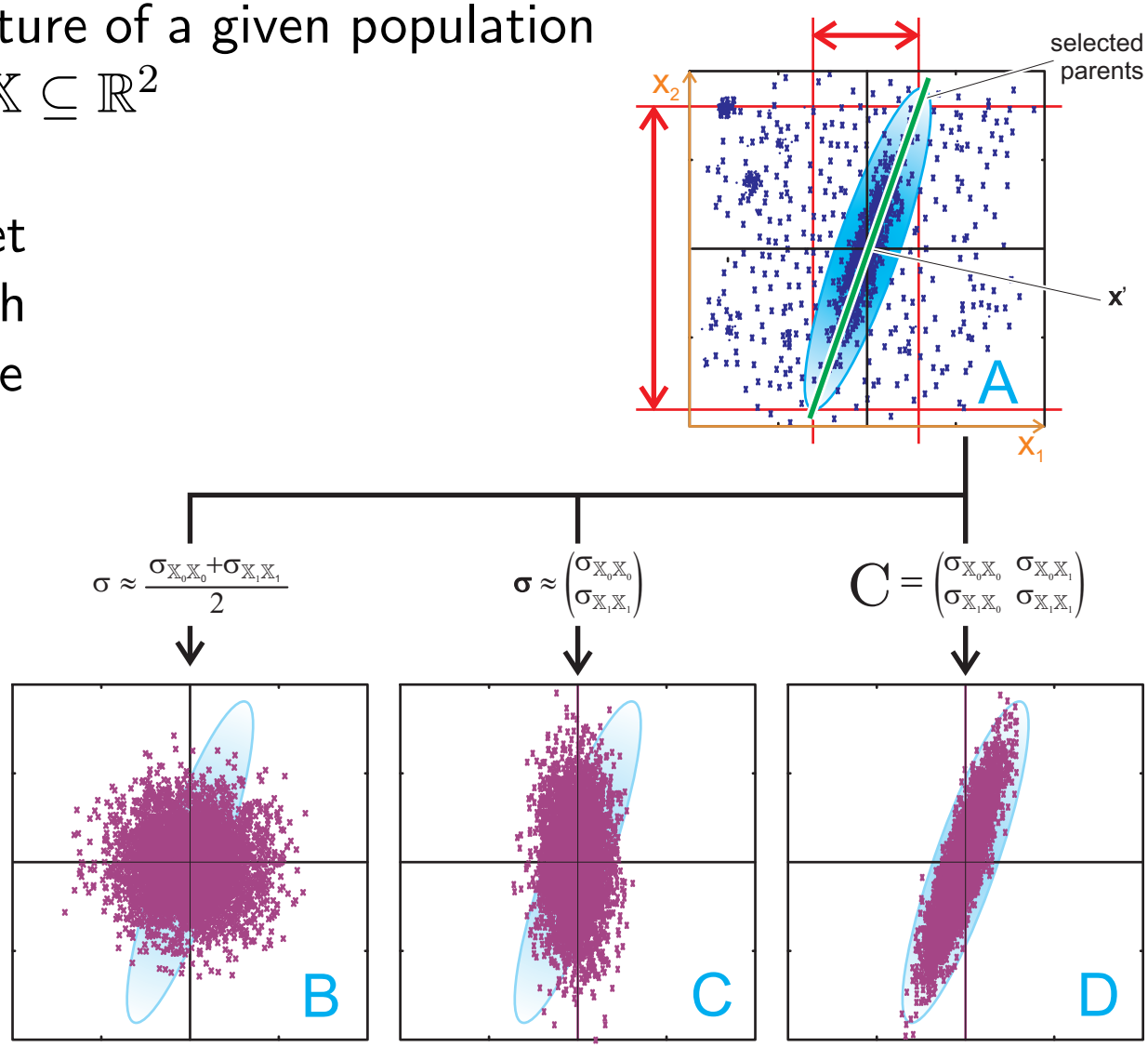


Fewer solutions generated outside the interesting range, but still many.

Mutation Operations B-C

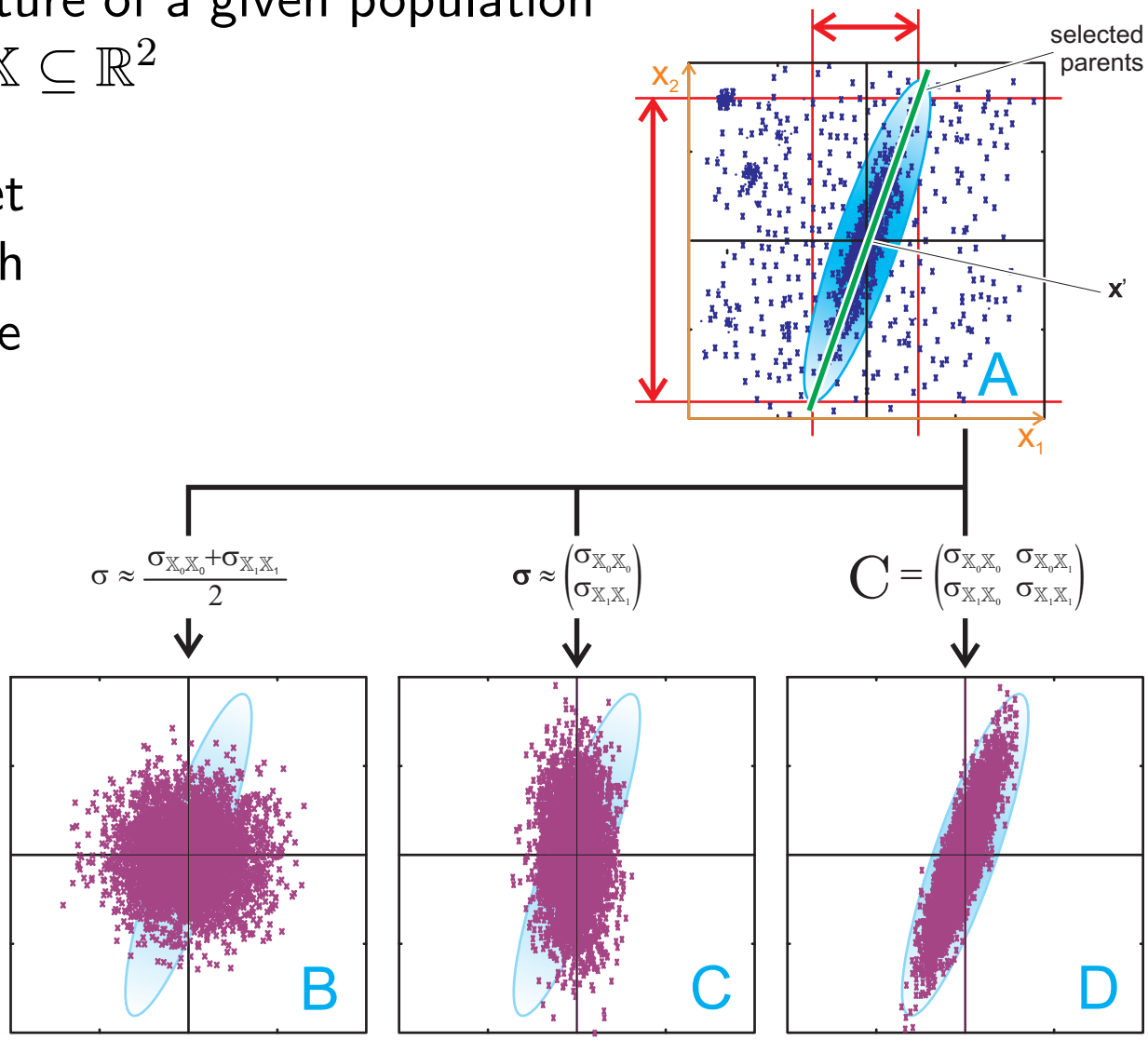
- Let's look at the structure of a given population in a 2D-search space $\mathbb{X} \subseteq \mathbb{R}^2$

D with full covariance matrices $\mathbf{C}$, we can get a cloud of points which is shaped similar to the selection, i.e., we has the shape of a rotated ellipse



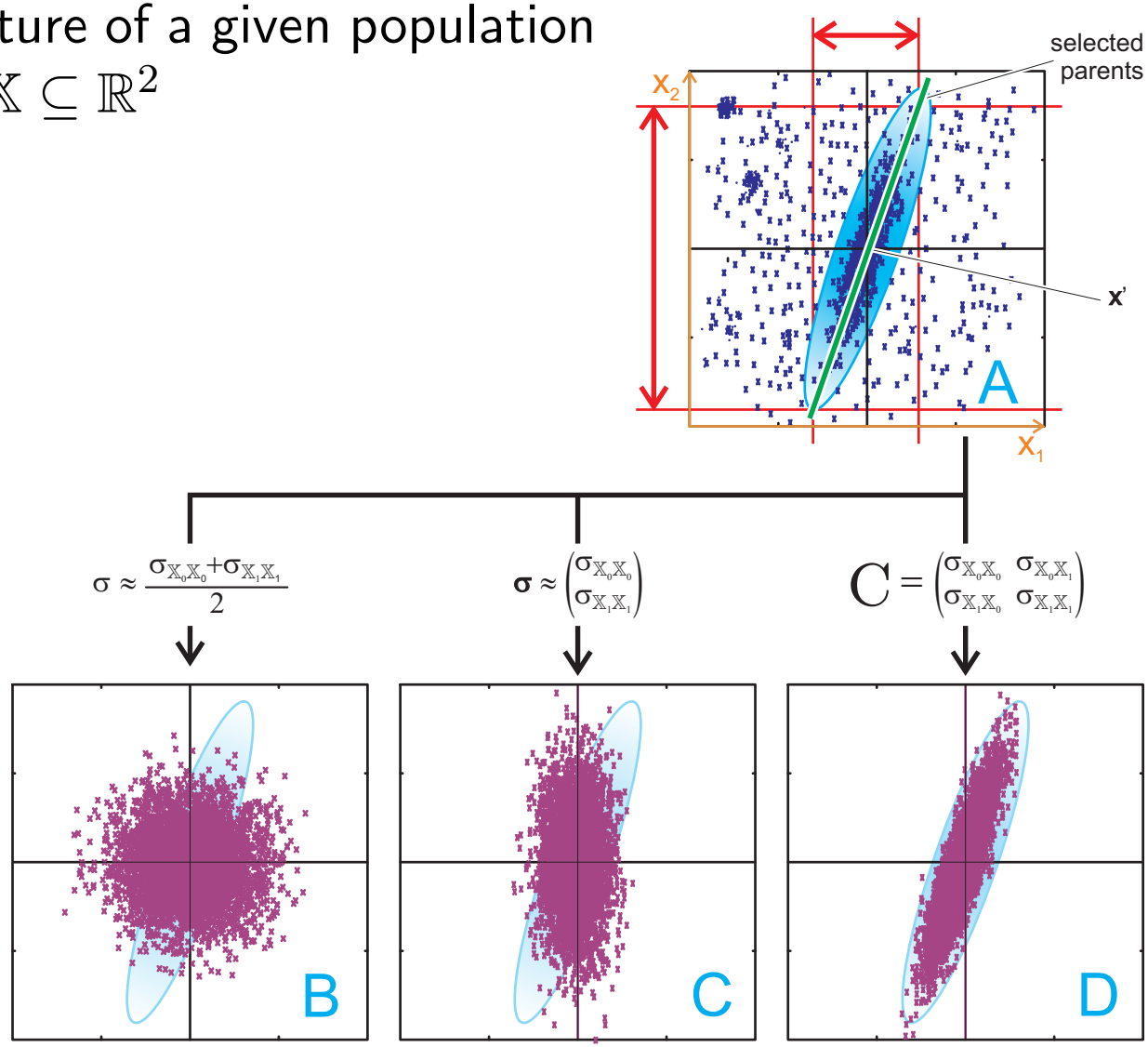
Mutation Operations B-C

- Let's look at the structure of a given population in a 2D-search space $\mathbb{X} \subseteq \mathbb{R}^2$
- **D** with full covariance matrices $\mathbf{C}$, we can get a cloud of points which is shaped similar to the selection, i.e., we has the shape of a rotated ellipse
- Of course, **D** is more complicated to implement than **C**, which is more complicated to implement than **B**



Mutation Operations B-C

- Let's look at the structure of a given population in a 2D-search space $\mathbb{X} \subseteq \mathbb{R}^2$
- 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 (σ, $\mathbf{C}$), and slower calculations



Mutation Operation B

- Single-valued standard deviation as step-width for mutation $p.w = \sigma$

$\vec{x} \longleftarrow \text{mutationES}_{w=\sigma}(\sigma, \vec{x}')$

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
  for $i \longleftarrow 0$ **up to** $n-1$ **do**
    $vecx_i \longleftarrow \vec{x}'_i + \sigma\{\text{Gaussian random number}\}$
  return $\vec{x}$

Mutation Operation C

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

$\vec{x} \longleftarrow \text{mutationES}_{w=\vec{\sigma}}(\vec{\sigma}, \vec{x}')$

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 $i \longleftarrow 0$ **up to** $n-1$ **do**
    $\vec{x}_i \longleftarrow \vec{x}'_i + \vec{\sigma}_i\{\text{Gaussian random number}\}$
  return $\vec{x}$

Mutation Operator B + C

Listing: Mutation Operator B + C

```
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
  }
}
```

Mutation Operation D

- Rotation matrix $p.w = \mathbf{M}$

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

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

begin
- **for** $i \longleftarrow 0$ **up to** $n-1$ **do**
  - $\vec{t}_i \longleftarrow$ {Gaussian random number}
- $\vec{x} \longleftarrow \vec{x}'$
- // $\vec{x} \longleftarrow \vec{x} + \mathbf{M}\vec{t}$
- **for** $i \longleftarrow 0$ **up to** $n-1$ **do**
  - **for** $j \longleftarrow 0$ **up to** $n-1$ **do**
    - $\vec{x}_i \longleftarrow \vec{x}_i + \mathbf{M}_{i,j} * \vec{t}_j$
- **return** $\vec{x}$

- $\mathbf{M}$ 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

Mutation Operation D

- Rotation matrix $p.w = \mathbf{M}$

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

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

begin
  for $i \longleftarrow 0$ **up to** $n-1$ **do**
    $\vec{t}_i \longleftarrow$ {Gaussian random number}
  $\vec{x} \longleftarrow \vec{x}'$
  // $\vec{x} \longleftarrow \vec{x} + \mathbf{M}\vec{t}$
  for $i \longleftarrow 0$ **up to** $n-1$ **do**
    for $j \longleftarrow 0$ **up to** $n-1$ **do**
      $\vec{x}_i \longleftarrow \vec{x}_i + \mathbf{M}_{i,j} * \vec{t}_j$
  return $\vec{x}$

- $\mathbf{M}$ basically is the Eigen-Vector matrix of the covariance matrix $\mathbf{M}$.

Mutation Operation D

- Rotation matrix $p.w = \mathbf{M}$

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

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

begin
  for $i \longleftarrow 0$ **up to** $n-1$ **do**
    $\vec{t}_i \longleftarrow$ {Gaussian random number}
  $\vec{x} \longleftarrow \vec{x}'$
  // $\vec{x} \longleftarrow \vec{x} + \mathbf{M}\vec{t}$
  for $i \longleftarrow 0$ **up to** $n-1$ **do**
    for $j \longleftarrow 0$ **up to** $n-1$ **do**
      $\vec{x}_i \longleftarrow \vec{x}_i + \mathbf{M}_{i,j} * \vec{t}_j$
  return $\vec{x}$

- $\mathbf{M}$ basically is the Eigen-Vector matrix of the covariance matrix $\mathbf{M}$.
- More information on sampling multi-dimensional normal distributions based on covariance/Eigen-Vector matrices: [12, 13]

Section Outline

Self-Adaptation

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

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

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

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*

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...

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)

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) or exogenous (maintained for whole population)

Section Outline

Self-Adaptation: Single Strategy Parameter

- Assume case B, a single standard deviation as step-width for the mutation normal distributed

Self-Adaptation: Single Strategy Parameter

- 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]

Self-Adaptation: Single Strategy Parameter

- 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

Self-Adaptation: Single Strategy Parameter

- 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:

Self-Adaptation: Single Strategy Parameter

- 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

Self-Adaptation: Single Strategy Parameter

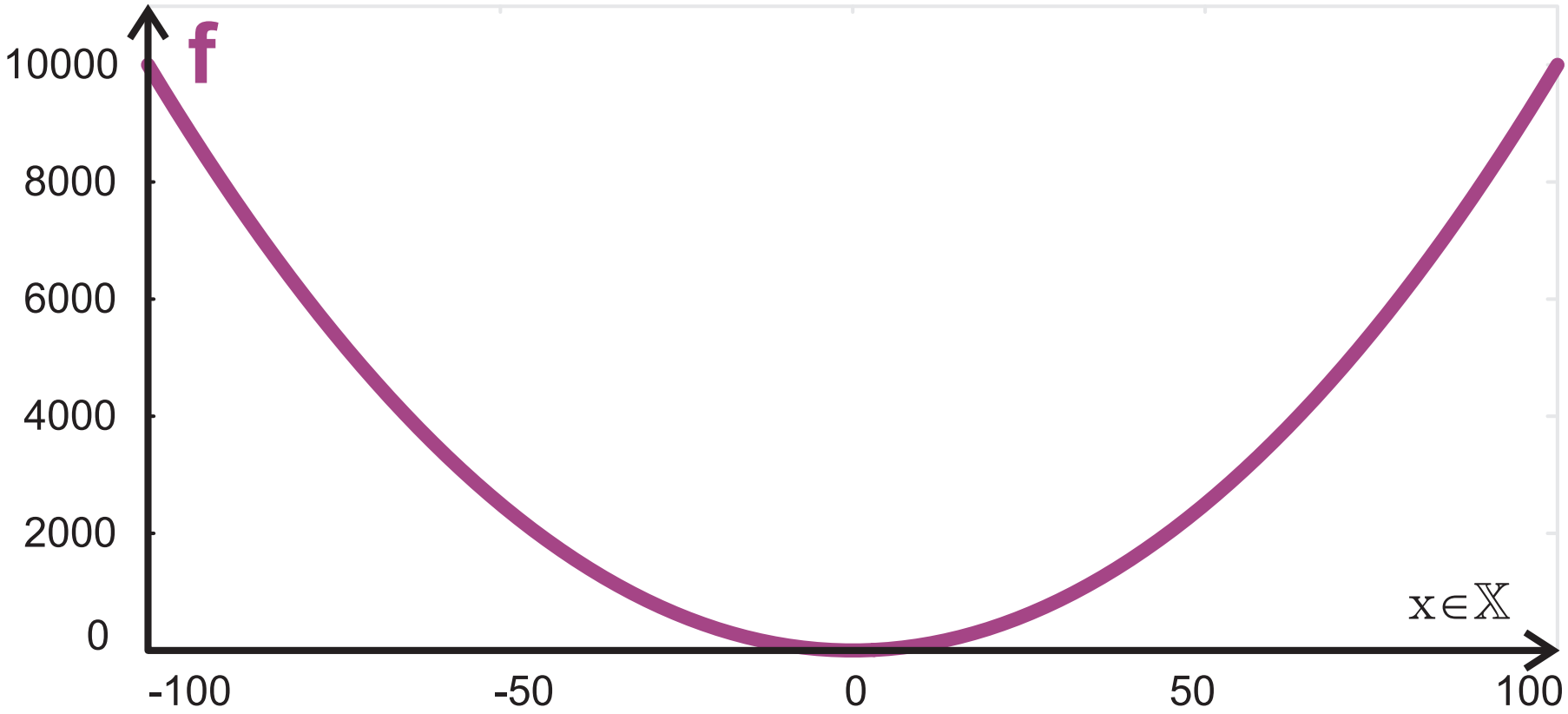
- Assume case B, a single standard deviation as step-width for the mutation normal distributed
- Idea: Adapt this step-width according to success of search [2, 14]
- In basic algorithm: $(1+1)$-ES
- Two key parameters are measured:
  1. success probability $P(S)$ of the mutation operation
  2. progress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

Self-Adaptation: Single Strategy Parameter

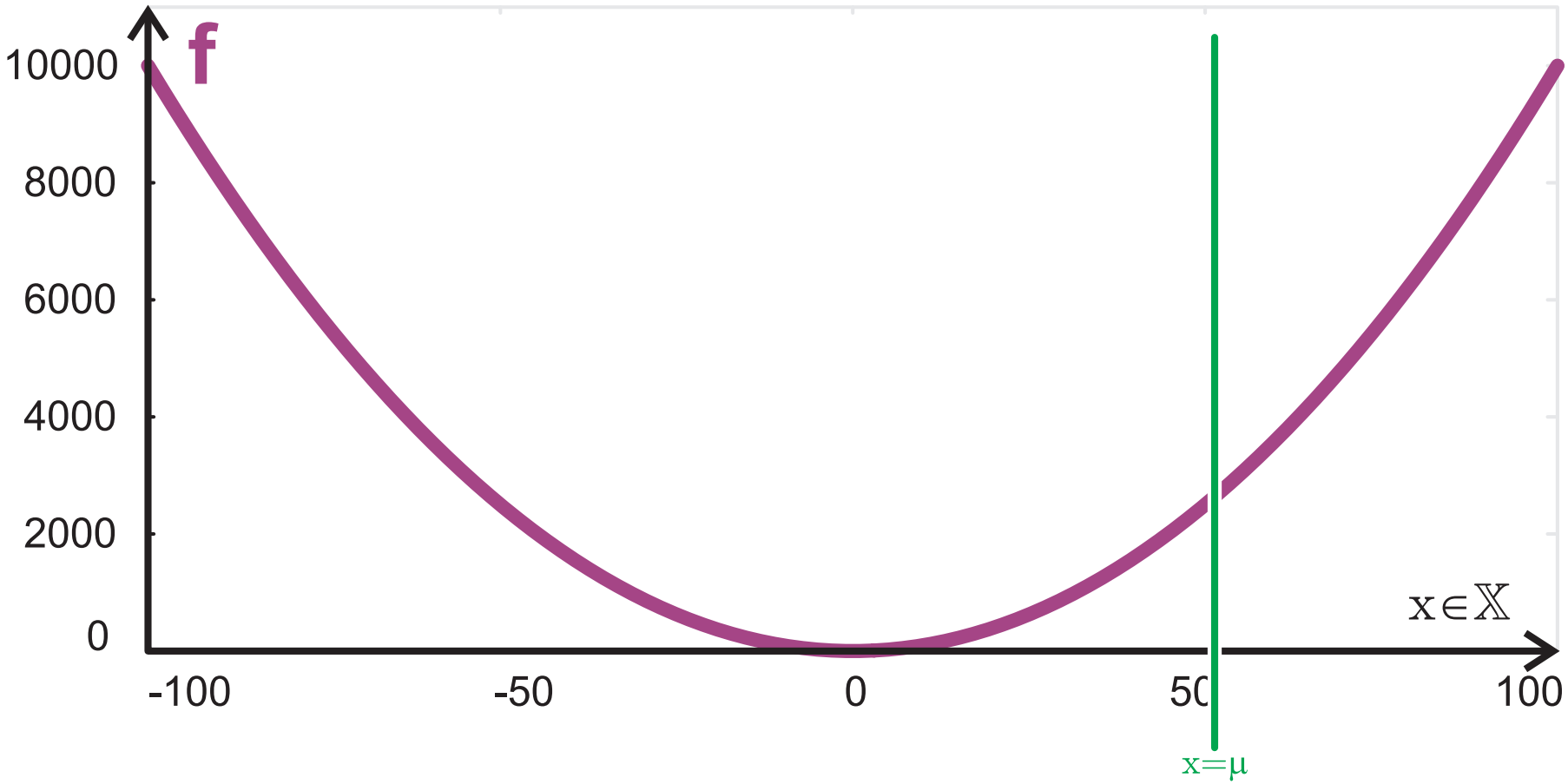
- Assume case B, a single standard deviation as step-width for the mutation normal distributed
- Idea: Adapt this step-width according to success of search [2, 14]
- In basic algorithm: $(1+1)$-ES
- Two key parameters are measured:
  1. success probability $P(S)$ of the mutation operation
  2. progress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

$$f(\vec{x}) = \sum_{i=1}^{n} \vec{x}_i^2 \text{ with } \mathbb{G} = \mathbb{X} \subseteq \mathbb{R}^n \tag{1}$$

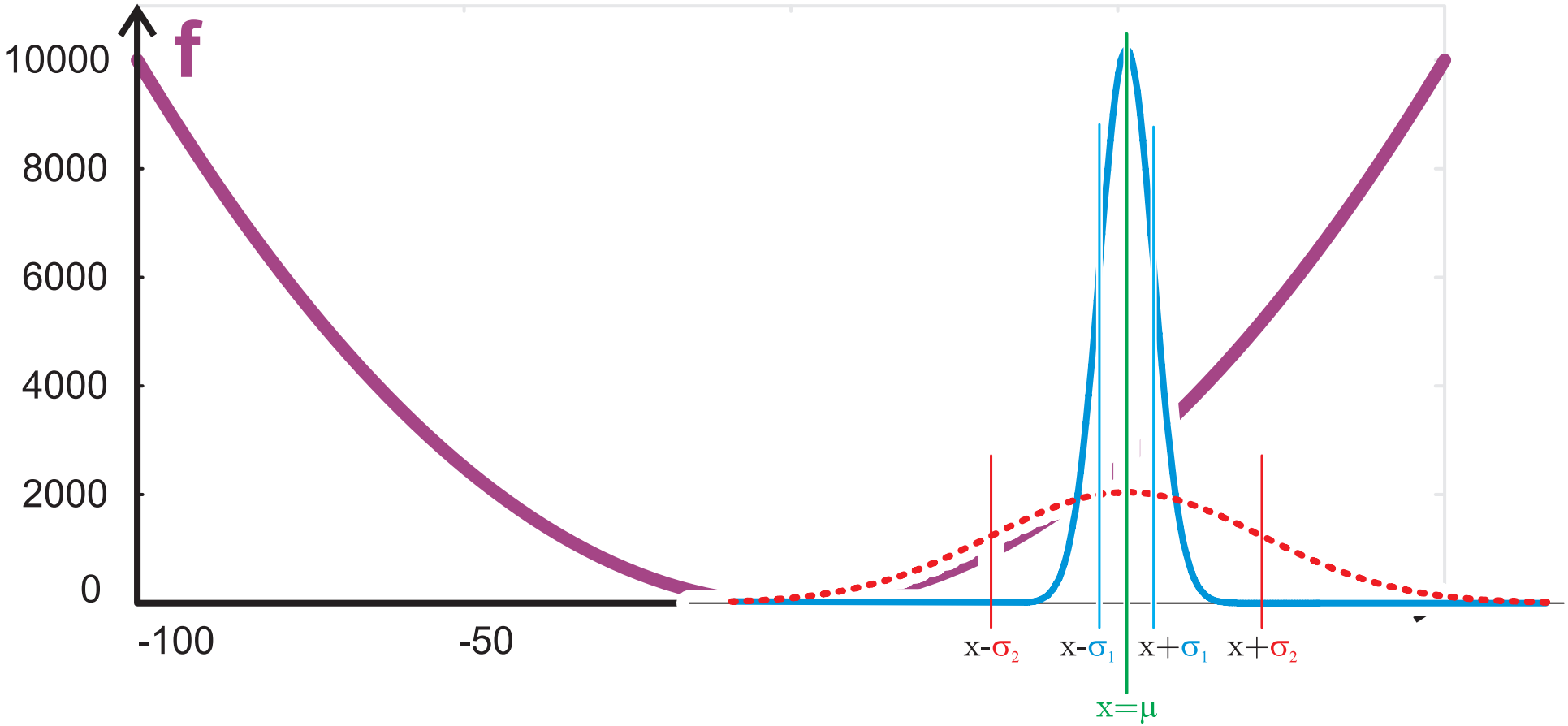
Self-Adaptation on Sphere Function



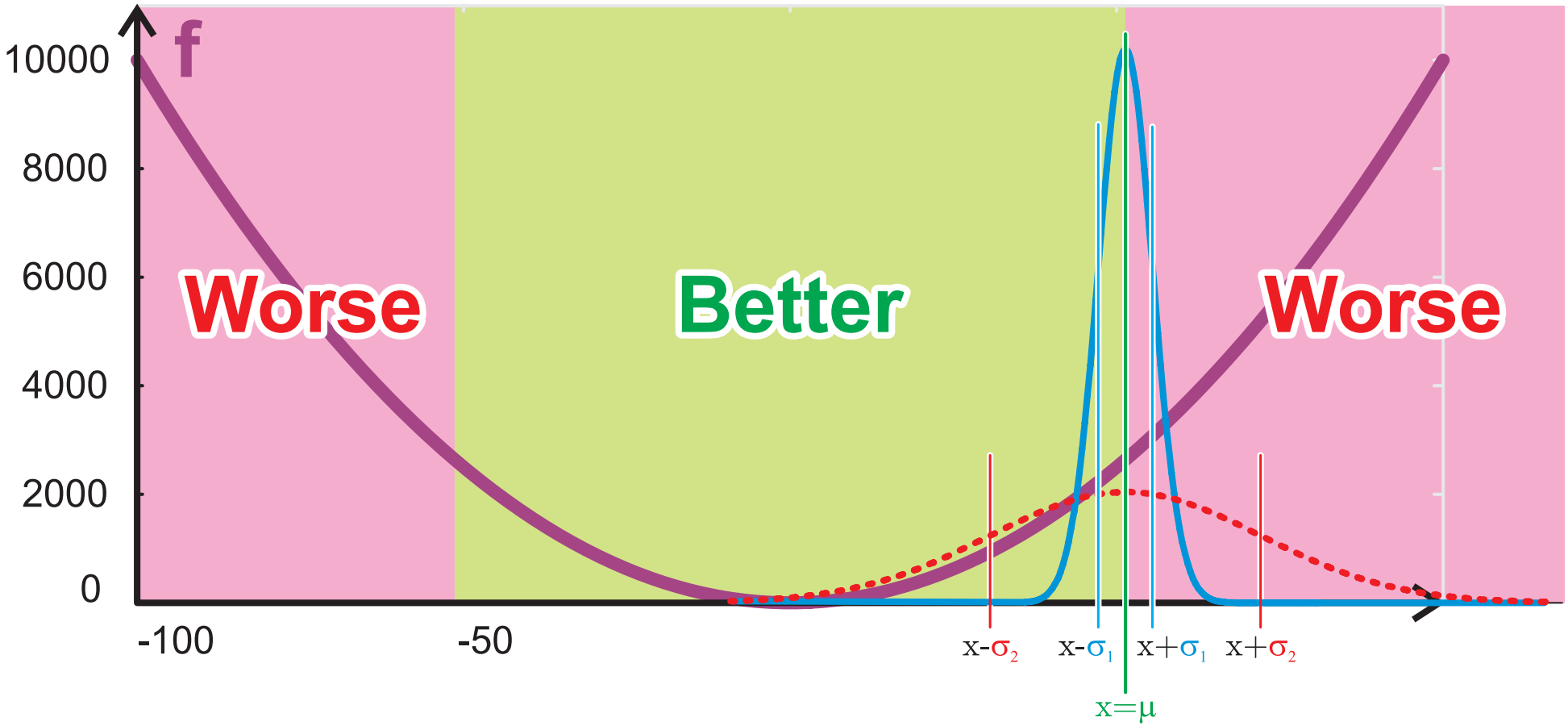
Self-Adaptation on Sphere Function



Self-Adaptation on Sphere Function



Self-Adaptation on Sphere Function



Self-Adaptation on Sphere Function

- For very small standard deviations σ:

Self-Adaptation on Sphere Function

- For very small standard deviations σ:

$$\lim_{\sigma \to 0} P(S) = 0.5 \tag{2}$$

$$\lim_{\sigma \to 0} \varphi = 0 \tag{3}$$

Self-Adaptation on Sphere Function

- For very small standard deviations σ:

$$\lim_{\sigma \to 0} P(S) = 0.5 \tag{2}$$

$$\lim_{\sigma \to 0} \varphi = 0 \tag{3}$$

- For very large standard deviations σ:

Self-Adaptation on Sphere Function

- For very small standard deviations σ:

$$\lim_{\sigma \to 0} P(S) = 0.5 \tag{2}$$

$$\lim_{\sigma \to 0} \varphi = 0 \tag{3}$$

- For very large standard deviations σ:

$$\lim_{\sigma \to +\infty} P(S) = 0 \tag{4}$$

$$\lim_{\sigma \to +\infty} \varphi = 0 \tag{5}$$

Self-Adaptation on Sphere Function

- For very small standard deviations σ:

$$\lim_{\sigma \to 0} P(S) = 0.5 \tag{2}$$

$$\lim_{\sigma \to 0} \varphi = 0 \tag{3}$$

- For very large standard deviations σ:

$$\lim_{\sigma \to +\infty} P(S) = 0 \tag{4}$$

$$\lim_{\sigma \to +\infty} \varphi = 0 \tag{5}$$

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

The $1/5$th Rule

Definition ($1/5$th Rule)

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

The $1/5$th Rule

Definition ($1/5$th Rule)

In order to obtain nearly optimal (local) performance of the $(1+1)$-ES with isotropic mutation, tune the mutation strength σ 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 σ from $\lim_{\sigma \to 0} P(S) = 0.5$ to $\lim_{\sigma \to +\infty} P(S) = 0$

Definition ($1/5$th Rule)

In order to obtain nearly optimal (local) performance of the $(1+1)$-ES with isotropic mutation, tune the mutation strength σ 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 σ 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

The $1/5$th Rule

Definition ($1/5$th Rule)

In order to obtain nearly optimal (local) performance of the $(1+1)$-ES with isotropic mutation, tune the mutation strength σ 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 σ 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 σ must be reduced

$1/5$th Rule-based $(1+1)$ Evolution Strategy

$p_{best} \longleftarrow (1+1)\ \mathrm{ES}_{\frac{1}{5}}(f, L, a, \sigma_0)$

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

- $(1+1)$-ES = self-adaptive hill climber

1/5th Rule-based $(1+1)$ Evolution Strategy

$p_{best} \longleftarrow (1{+}1)\ \mathrm{ES}_{\frac{1}{5}}(f, L, a, \sigma_0)$

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

- $(1+1)$-ES = self-adaptive hill climber
- Initialize iteration counter t, success counter s, and mutation strength σ

$1/5$th Rule-based $(1+1)$ Evolution Strategy

$p_{best} \longleftarrow (1{+}1)\ \mathrm{ES}_{\frac{1}{5}}(f, L, a, \sigma_0)$

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

- $(1+1)$-ES = self-adaptive hill climber
- Create initial point in search space $p_{best}.g$, map it to candidate solution $p_{best}.x$, compute its objective value $p_{best}.y = f(p_{best}.x)$, and store it in "current" individual p_{new}

$1/5$th Rule-based $(1+1)$ Evolution Strategy

$p_{best} \longleftarrow (1{+}1)\ \mathrm{ES}_{\frac{1}{5}}(f, L, a, \sigma_0)$

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

- $(1+1)$-ES = self-adaptive hill climber
- map point in search space $p_{new}.g$ of current individual to candidate solution $p_{new}.x$ and compute its objective value $p_{new}.y = f(p_{new}.x)$

$1/5$th Rule-based $(1+1)$ Evolution Strategy

$p_{best} \longleftarrow (1+1)\ \mathrm{ES}_{\frac{1}{5}}(f, L, a, \sigma_0)$

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

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

$1/5$th Rule-based $(1+1)$ Evolution Strategy

$p_{best} \longleftarrow (1+1)\ \mathrm{ES}_{\frac{1}{5}}(f, L, a, \sigma_0)$

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

- $(1+1)$-ES = self-adaptive hill climber
- If current individual p_{new} is better then best known individual p_{best},
- store it in p_{best} and

1/5th Rule-based $(1+1)$ Evolution Strategy

$p_{best} \longleftarrow (1{+}1)\ \mathrm{ES}_{\frac{1}{5}}(f, L, a, \sigma_0)$

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

- $(1+1)$-ES = self-adaptive hill climber
- If current individual p_{new} is better then best known individual p_{best},
- store it in p_{best} and
- increase success counter s by one.

$1/5$th Rule-based $(1+1)$ Evolution Strategy

$p_{best} \longleftarrow (1{+}1)\ \mathrm{ES}_{\frac{1}{5}}(f, L, a, \sigma_0)$

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

- $(1+1)$-ES = self-adaptive hill climber
- After every L iterations, it is time for self-adaptation.

$1/5$th Rule-based $(1+1)$ Evolution Strategy

$p_{best} \longleftarrow (1{+}1)\ \mathrm{ES}_{\frac{1}{5}}(f, L, a, \sigma_0)$

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

$1/5$th Rule-based $(1+1)$ Evolution Strategy

$p_{best} \longleftarrow (1{+}1)\ \mathrm{ES}_{\frac{1}{5}}(f, L, a, \sigma_0)$

begin
- $t \longleftarrow 1$
- $s \longleftarrow 0$
- $\sigma \longleftarrow \sigma_0$
- $p_{best}.g \longleftarrow$ create starting point
- $p_{best}.x \longleftarrow \mathrm{gpm}(p_{best}.g)$
- $p_{best}.y \longleftarrow f(p_{best}.x)$
- $p_{new} \longleftarrow p_{best}$
- **while** $\neg shouldTerminate$ **do**
  - $p_{new}.x \longleftarrow \mathrm{gpm}(p_{new}.g)$
  - $p_{new}.y \longleftarrow f(p_{new}.x)$
  - **if** $p_{new}.y < p_{best}.y$ **then**
    - $p_{best} \longleftarrow p_{new}$
    - $s \longleftarrow s + 1$
  - **if** $(t \bmod L) = 0$ **then**
    - $P(S) \longleftarrow \frac{s}{L}$
    - **if** $P(S) < 0.2$ **then** $\sigma \longleftarrow \sigma * a$
    - **else if** $P(S) > 0.2$ **then** $\sigma \longleftarrow \sigma / a$
    - $s \longleftarrow 0$
  - $p_{new}.g \longleftarrow \mathrm{mutationES}_{\sigma}(\sigma, p_{new})$
  - $t \longleftarrow t + 1$
- **return** p_{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 less than $0.2\ldots$

1/5th Rule-based $(1+1)$ Evolution Strategy

$p_{best} \longleftarrow (1{+}1)\ \mathrm{ES}_{\frac{1}{5}}(f, L, a, \sigma_0)$

begin
- $t \longleftarrow 1$
- $s \longleftarrow 0$
- $\sigma \longleftarrow \sigma_0$
- $p_{best}.g \longleftarrow$ create starting point
- $p_{best}.x \longleftarrow \mathrm{gpm}(p_{best}.g)$
- $p_{best}.y \longleftarrow f(p_{best}.x)$
- $p_{new} \longleftarrow p_{best}$
- **while** $\neg shouldTerminate$ **do**
  - $p_{new}.x \longleftarrow \mathrm{gpm}(p_{new}.g)$
  - $p_{new}.y \longleftarrow f(p_{new}.x)$
  - **if** $p_{new}.y < p_{best}.y$ **then**
    - $p_{best} \longleftarrow p_{new}$
    - $s \longleftarrow s + 1$
  - **if** $(t \bmod L) = 0$ **then**
    - $P(S) \longleftarrow \frac{s}{L}$
    - **if** $P(S) < 0.2$ **then** $\sigma \longleftarrow \sigma * a$
    - **else if** $P(S) > 0.2$ **then** $\sigma \longleftarrow \sigma / a$
    - $s \longleftarrow 0$
  - $p_{new}.g \longleftarrow \mathrm{mutationES}_{\sigma}(\sigma, p_{new})$
  - $t \longleftarrow t + 1$
- **return** p_{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 less than 0.2, decrease σ by multiplying it with constant $a \in [0, 1]$.

$1/5$th Rule-based $(1+1)$ Evolution Strategy

$p_{best} \longleftarrow (1{+}1)\ \mathrm{ES}_{\frac{1}{5}}(f, L, a, \sigma_0)$

begin
- $t \longleftarrow 1$
- $s \longleftarrow 0$
- $\sigma \longleftarrow \sigma_0$
- $p_{best}.g \longleftarrow$ create starting point
- $p_{best}.x \longleftarrow \mathrm{gpm}(p_{best}.g)$
- $p_{best}.y \longleftarrow f(p_{best}.x)$
- $p_{new} \longleftarrow p_{best}$
- **while** $\neg shouldTerminate$ **do**
  - $p_{new}.x \longleftarrow \mathrm{gpm}(p_{new}.g)$
  - $p_{new}.y \longleftarrow f(p_{new}.x)$
  - **if** $p_{new}.y < p_{best}.y$ **then**
    - $p_{best} \longleftarrow p_{new}$
    - $s \longleftarrow s + 1$
  - **if** $(t \bmod L) = 0$ **then**
    - $P(S) \longleftarrow \frac{s}{L}$
    - **if** $P(S) < 0.2$ **then** $\sigma \longleftarrow \sigma * a$
    - **else if** $P(S) > 0.2$ **then** $\sigma \longleftarrow \sigma / a$
    - $s \longleftarrow 0$
  - $p_{new}.g \longleftarrow \mathrm{mutationES}_{\sigma}(\sigma, p_{new})$
  - $t \longleftarrow t + 1$
- **return** p_{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 less than 0.2, decrease σ by multiplying it with constant $a \in [0, 1]$.
- if larger than $0.2\ldots$

$1/5$th Rule-based $(1+1)$ Evolution Strategy

$p_{best} \longleftarrow (1{+}1)\ \mathrm{ES}_{\frac{1}{5}}(f, L, a, \sigma_0)$

begin
- $t \longleftarrow 1$
- $s \longleftarrow 0$
- $\sigma \longleftarrow \sigma_0$
- $p_{best}.g \longleftarrow$ create starting point
- $p_{best}.x \longleftarrow \mathrm{gpm}(p_{best}.g)$
- $p_{best}.y \longleftarrow f(p_{best}.x)$
- $p_{new} \longleftarrow p_{best}$
- **while** $\neg shouldTerminate$ **do**
  - $p_{new}.x \longleftarrow \mathrm{gpm}(p_{new}.g)$
  - $p_{new}.y \longleftarrow f(p_{new}.x)$
  - **if** $p_{new}.y < p_{best}.y$ **then**
    - $p_{best} \longleftarrow p_{new}$
    - $s \longleftarrow s + 1$
  - **if** $(t \bmod L) = 0$ **then**
    - $P(S) \longleftarrow \frac{s}{L}$
    - **if** $P(S) < 0.2$ **then** $\sigma \longleftarrow \sigma * a$
    - **else if** $P(S) > 0.2$ **then** $\sigma \longleftarrow \sigma / a$
    - $s \longleftarrow 0$
  - $p_{new}.g \longleftarrow \mathrm{mutationES}_{\sigma}(\sigma, p_{new})$
  - $t \longleftarrow t + 1$
- **return** p_{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 σ by dividing it by constant $a \in [0, 1]$.

$1/5$th Rule-based $(1+1)$ Evolution Strategy

$p_{best} \longleftarrow (1{+}1)\ \mathrm{ES}_{\frac{1}{5}}(f, L, a, \sigma_0)$

begin
- $t \longleftarrow 1$
- $s \longleftarrow 0$
- $\sigma \longleftarrow \sigma_0$
- $p_{best}.g \longleftarrow$ create starting point
- $p_{best}.x \longleftarrow \mathrm{gpm}(p_{best}.g)$
- $p_{best}.y \longleftarrow f(p_{best}.x)$
- $p_{new} \longleftarrow p_{best}$
- **while** $\neg shouldTerminate$ **do**
  - $p_{new}.x \longleftarrow \mathrm{gpm}(p_{new}.g)$
  - $p_{new}.y \longleftarrow f(p_{new}.x)$
  - **if** $p_{new}.y < p_{best}.y$ **then**
    - $p_{best} \longleftarrow p_{new}$
    - $s \longleftarrow s + 1$
  - **if** $(t \bmod L) = 0$ **then**
    - $P(S) \longleftarrow \frac{s}{L}$
    - **if** $P(S) < 0.2$ **then** $\sigma \longleftarrow \sigma * a$
    - **else if** $P(S) > 0.2$ **then** $\sigma \longleftarrow \sigma / a$
    - $s \longleftarrow 0$
  - $p_{new}.g \longleftarrow \mathrm{mutationES}_{\sigma}(\sigma, p_{new})$
  - $t \longleftarrow t + 1$
- **return** p_{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 σ by dividing it by constant $a \in [0, 1]$.
- Finally, reset success counter s to 0.

$1/5$th Rule-based $(1+1)$ Evolution Strategy

$p_{best} \longleftarrow (1+1)\ \mathrm{ES}_{\frac{1}{5}}(f, L, a, \sigma_0)$

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

- $(1+1)$-ES = self-adaptive hill climber
- Perform mutation by using step-width σ

$1/5$th Rule-based $(1+1)$ Evolution Strategy

$p_{best} \longleftarrow (1+1)\ \mathrm{ES}_{\frac{1}{5}}(f, L, a, \sigma_0)$

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

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

$1/5$th Rule-based $(1+1)$ Evolution Strategy

$p_{best} \longleftarrow (1{+}1)\ \mathrm{ES}_{\frac{1}{5}}(f, L, a, \sigma_0)$

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

- $(1+1)$-ES = self-adaptive hill climber
- Return best candidate solution discovered.

(1+1) ES with 1/5th rule

Listing: (1+1) ES with 1/5th rule

```java
public class ES1P1<X> extends OptimizationAlgorithm<double[], X> {
  public Individual<double[], X> solve(final IObjectiveFunction<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
  }
}
```

$1/5$th Rule: Advantages and Drawbacks

- The $1/5^{th}$ rule has **advantages**

$1/5$th Rule: Advantages and Drawbacks

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

$1/5$th Rule: Advantages and Drawbacks

- The $1/5^{th}$ rule has **advantages**:
  - Many mutations are successful $\longrightarrow$ we can make larger steps and progress faster
  - Many mutations are unsuccessful $\longrightarrow$ we make smaller steps so that we can approach the optimum instead of jumping over it

$1/5$th Rule: Advantages and Drawbacks

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

$1/5$th Rule: Advantages and Drawbacks

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

$1/5$th Rule: Advantages and Drawbacks

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

$1/5$th Rule: Advantages and Drawbacks

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

$1/5$th Rule: Advantages and Drawbacks

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

$1/5$th Rule: Advantages and Drawbacks

- The $1/5^{th}$ rule has **advantages**:
  - Many mutations are successful $\longrightarrow$ we can make larger steps and progress faster
  - Many mutations are unsuccessful $\longrightarrow$ 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 σ: cannot model different step widths for different dimensions or dependencies between dimensions

$1/5$th Rule: Advantages and Drawbacks

- The $1/5^{th}$ rule has **advantages**:
  - Many mutations are successful $\longrightarrow$ we can make larger steps and progress faster
  - Many mutations are unsuccessful $\longrightarrow$ 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 σ: cannot model different step widths for different dimensions or dependencies between dimensions
  - Only a single parameter σ: cannot easily be extended for population-based methods

Section Outline

Endogeneous Parameters

- Instead of having one central exogeneous[1] (set of) strategy parameter(s)...

[1] ≡ outside 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!

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

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[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

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
- Information $p.w$ undergoes reproduction, similar to genotypes $p.g \in \mathbb{G}$

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
- 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.

Endogeneous Adaptation

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

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

- Evolution Strategies follow same basic pattern as GAs

Endogeneous Adaptation

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

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

- 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

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

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

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

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

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

Endogeneous Adaptation

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

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

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

Endogeneous Adaptation

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

```
begin
    t ⟵ 1
    pop ⟵ random initialization
    while ¬shouldTerminate do
        perform genotype-phenotype mapping
        compute objective values (possibly update best-so-far solution p_best)
        if t > 1 then
            if strategy = (μ/ρ, λ) then
                matePool ⟵ truncationSelection(μ, pop)
            else
                matePool ⟵ truncationSelection(μ, pop ∪ matePool)
        else
            matePool ⟵ pop
        for i ⟵ 1 up to λ do
            parents ⟵ choose ρ parents from matePool
            p_new.g ⟵ recombinationES(p.g ∀p ∈ parents)
            p_new.w ⟵ infoRecombinationES(p.w ∀p ∈ parents)
            p_new.w ⟵ infoMutationES(p_new.w)
            p_new.g ⟵ mutationES(p_new.g, p_new.w)
            population[i] ⟵ p_new
        t ⟵ t + 1
    return best solution x̃ discovered
```

- Evolution Strategies follow same basic pattern as GAs
- Perform survival selection
  - in (μ, λ)-ESs, select only from the current population

Endogeneous Adaptation

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

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

- Evolution Strategies follow same basic pattern as GAs
- Perform survival selection
  - in (μ, λ)-ESs, select only from the current population
  - in $(\mu + \lambda)$-ESs, select from the current population and the mating pool

Endogeneous Adaptation

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

```
begin
    t ⟵ 1
    pop ⟵ random initialization
    while ¬shouldTerminate do
        perform genotype-phenotype mapping
        compute objective values (possibly update best-so-far solution p_best)
        if t > 1 then
            if strategy = (μ/ρ, λ) then
                matePool ⟵ truncationSelection(μ, pop)
            else
                matePool ⟵ truncationSelection(μ, pop ∪ matePool)
        else
            matePool ⟵ pop
        for i ⟵ 1 up to λ do
            parents ⟵ choose ρ parents from matePool
            p_new.g ⟵ recombinationES(p.g ∀p ∈ parents)
            p_new.w ⟵ infoRecombinationES(p.w ∀p ∈ parents)
            p_new.w ⟵ infoMutationES(p_new.w)
            p_new.g ⟵ mutationES(p_new.g, p_new.w)
            population[i] ⟵ p_new
        t ⟵ t + 1
    return best solution x̃ discovered
```

- Evolution Strategies follow same basic pattern as GAs
- Perform survival selection
  - in (μ, λ)-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.

Endogeneous Adaptation

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

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

- Evolution Strategies follow same basic pattern as GAs
- For each offspring individual that we want to create...

Endogeneous Adaptation

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

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

Endogeneous Adaptation

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

```
begin
    t ⟵ 1
    pop ⟵ random initialization
    while ¬shouldTerminate do
        perform genotype-phenotype mapping
        compute objective values (possibly update best-so-far solution p_best)
        if t > 1 then
            if strategy = (μ/ρ, λ) then
                matePool ⟵ truncationSelection(μ, pop)
            else
                matePool ⟵ truncationSelection(μ, pop ∪ matePool)
        else
            matePool ⟵ pop
        for i ⟵ 1 up to λ do
            parents ⟵ choose ρ parents from matePool
            p_new.g ⟵ recombinationES(p.g ∀p ∈ parents)
            p_new.w ⟵ infoRecombinationES(p.w ∀p ∈ parents)
            p_new.w ⟵ infoMutationES(p_new.w)
            p_new.g ⟵ mutationES(p_new.g, p_new.w)
            population[i] ⟵ p_new
        t ⟵ t + 1
    return best solution x̃ discovered
```

- Evolution Strategies follow same basic pattern as GAs
- For each offspring individual that we want to create. . .
  - . . . first select ρ parents,
  - (re)combine the parental genotypes,

Endogeneous Adaptation

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

```
begin
    t ⟵ 1
    pop ⟵ random initialization
    while ¬shouldTerminate do
        perform genotype-phenotype mapping
        compute objective values (possibly update best-so-far solution p_best)
        if t > 1 then
            if strategy = (μ/ρ, λ) then
                matePool ⟵ truncationSelection(μ, pop)
            else
                matePool ⟵ truncationSelection(μ, pop ∪ matePool)
        else
            matePool ⟵ pop
        for i ⟵ 1 up to λ do
            parents ⟵ choose ρ parents from matePool
            p_new.g ⟵ recombinationES(p.g ∀p ∈ parents)
            p_new.w ⟵ infoRecombinationES(p.w ∀p ∈ parents)
            p_new.w ⟵ infoMutationES(p_new.w)
            p_new.g ⟵ mutationES(p_new.g, p_new.w)
            population[i] ⟵ p_new
        t ⟵ t + 1
    return best solution x̃ discovered
```

- Evolution Strategies follow same basic pattern as GAs
- For each offspring individual that we want to create...
  - ...first select ρ parents,
  - (re)combine the parental genotypes,
  - and then (re)combine the endogenous information.

Endogeneous Adaptation

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

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

Endogeneous Adaptation

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

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

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

begin
- $t \longleftarrow 1$
- pop $\longleftarrow$ random initialization
- **while** $\neg shouldTerminate$ **do**
  - perform genotype-phenotype mapping
  - compute objective values (possibly update best-so-far solution p_{best})
  - **if** $t > 1$ **then**
    - **if** strategy $= (\mu/\rho, \lambda)$ **then**
      - matePool $\longleftarrow$ truncationSelection(μ, pop)
    - **else**
      - matePool $\longleftarrow$ truncationSelection$(\mu, \text{pop} \cup \text{matePool})$
  - **else**
    - matePool $\longleftarrow$ pop
  - **for** $i \longleftarrow 1$ **up to** λ **do**
    - parents $\longleftarrow$ choose ρ parents from matePool
    - $p_{new}.g \longleftarrow \text{recombinationES}(p.g \forall p \in \text{parents})$
    - $p_{new}.w \longleftarrow \text{infoRecombinationES}(p.w \forall p \in \text{parents})$
    - $p_{new}.w \longleftarrow \text{infoMutationES}(p_{new}.w)$
    - $p_{new}.g \longleftarrow \text{mutationES}(p_{new}.g, p_{new}.w)$
    - $population[i] \longleftarrow p_{new}$
  - $t \longleftarrow t + 1$
- **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
  - Good settings survive along with the candidate solutions that they have created

Endogeneous Adaptation

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

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

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

Endogeneous Adaptation

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

```
begin
    t ⟵ 1
    pop ⟵ random initialization
    while ¬shouldTerminate do
        perform genotype-phenotype mapping
        compute objective values (possibly update best-so-far solution p_best)
        if t > 1 then
            if strategy = (μ/ρ, λ) then
                matePool ⟵ truncationSelection(μ, pop)
            else
                matePool ⟵ truncationSelection(μ, pop ∪ matePool)
        else
            matePool ⟵ pop
        for i ⟵ 1 up to λ do
            parents ⟵ choose ρ parents from matePool
            p_new.g ⟵ recombinationES(p.g ∀p ∈ parents)
            p_new.w ⟵ infoRecombinationES(p.w ∀p ∈ parents)
            p_new.w ⟵ infoMutationES(p_new.w)
            p_new.g ⟵ mutationES(p_new.g, p_new.w)
            population[i] ⟵ p_new
        t ⟵ t + 1
    return best solution x̃ 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

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

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

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

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

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

Endogeneous Adaptation

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

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

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

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

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

Section Outline

[A] Discrete Recombination

- Extension of uniform crossover to ρ real vectors

[A] Discrete Recombination

- Extension of uniform crossover to ρ real vectors
- if $\rho = 2$, returns corner of hyper-cube created by parents

[A] Discrete Recombination

- Extension of uniform crossover to ρ real vectors
- if $\rho = 2$, returns corner of hyper-cube created by parents

$\vec{g}' \longleftarrow$ recombinationDiscrete(parents)

Input: parents: the list of ρ parent individuals
Data: i: a counter variable
Data: p: a parent individual
Output: $\vec{g}'$: the offspring of the parents

begin
  for $i \longleftarrow 0$ **up to** $n-1$ **do**
    $p \longleftarrow$ parents[{randomly from $0..\rho-1$}]
    $\vec{g}'_i \longleftarrow p.g_i$
  return $\vec{g}'$

Discrete Recombination

Listing: Discrete Recombination for 2 Parents

```
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;
  }
}
```

[B] Intermediate Recombination

- Extension of weighted average crossover to ρ real vectors

[B] Intermediate Recombination

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

[B] Intermediate Recombination

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

$\vec{g}' \longleftarrow$ recombinationIntermediate(parents)

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

begin
  for $i \longleftarrow 0$ **up to** $n-1$ **do**
    $s \longleftarrow 0$
    for $j \longleftarrow 0$ **up to** $\rho - 1$ **do**
      $p \longleftarrow \text{parents}[j]$
      $s \longleftarrow s + p.g_i$
    $\vec{g}'_i \longleftarrow \frac{s}{\rho}$
  return $\vec{g}'$

Intermediate Recombination

Listing: Intermediate Recombination for 2 Parents

```
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;
  }
}
```

Section Outline

Reproduction of Endogenous Parameters

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

Reproduction of Endogenous Parameters

- 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

- 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

Reproduction of Endogenous Parameters

- 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...

Reproduction of Endogenous Parameters

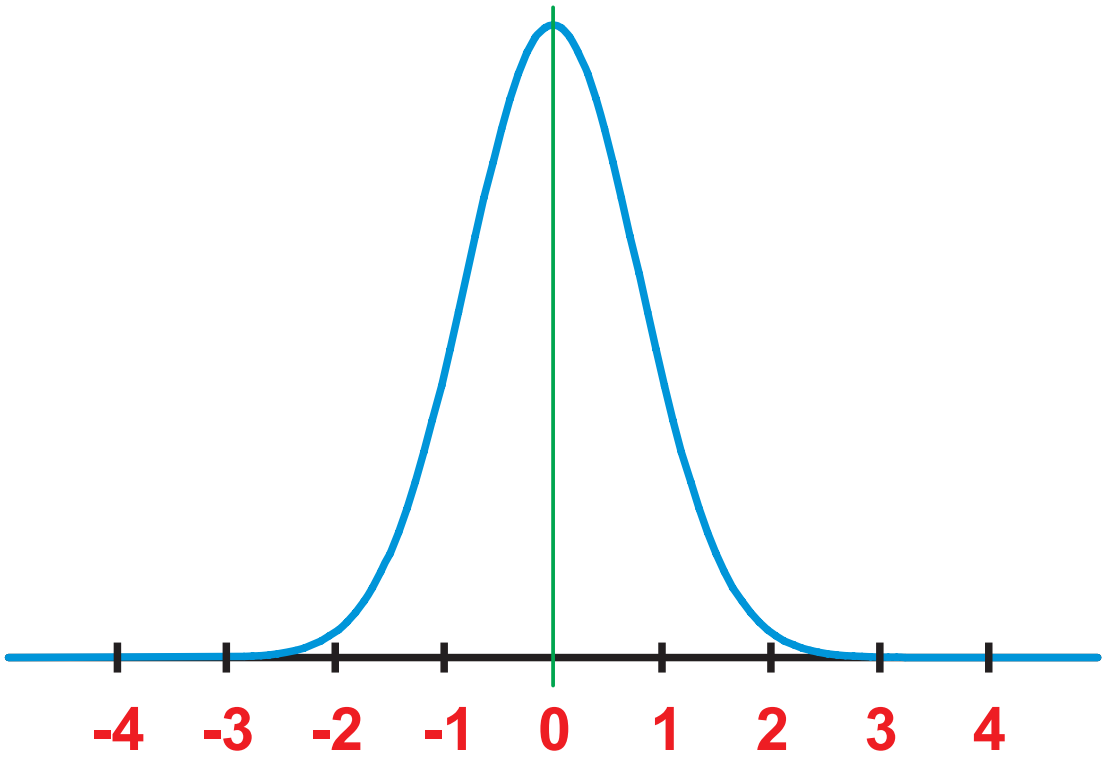
- Recombination of strategy parameters $p.w$: intermediate crossover
- Mutation is different from mutation of a genotype: we mutate a mutation strength
- Mutation means to mutate the degree with which a candidate solution should be changed
- Here, not the absolute value (1, 7, 1.5, etc.) is interesting...
- ...but more the scale, i.e., 1, 10, 0.1, 10000, 0.001, ...

Reproduction of Endogenous Parameters

- Recombination of strategy parameters $p.w$: intermediate crossover
- So we are concerned about the scale

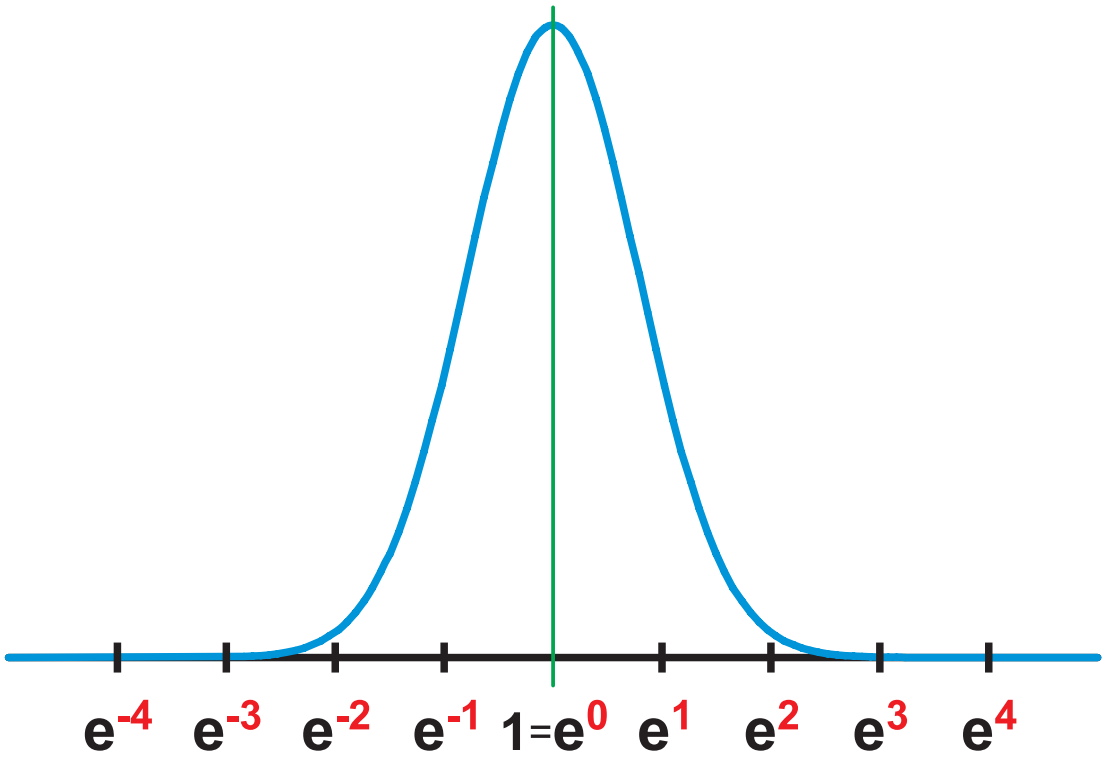
Reproduction of Endogenous Parameters

- 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

$\vec{\sigma} \longleftarrow$ 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
- $\nu \longleftarrow e^{\tau_0\{\text{Gaussian random number}\}}$
- $\vec{\sigma} \longleftarrow \vec{0}$
- **for** $i \longleftarrow 0$ **up to** $n-1$ **do**
  - $\vec{\sigma}_i \longleftarrow \nu * e^{\tau\{\text{Gaussian random number}\}} * \vec{\sigma}'_i$
- **return** $\vec{\sigma}$

- Mutation of strategy parameters: apply lognormal mutation

Reproduction of Endogenous Parameters

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

$\vec{\sigma} \longleftarrow$ 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
- $\nu \longleftarrow e^{\tau_0 \{\text{Gaussian random number}\}}$
- $\vec{\sigma} \longleftarrow \vec{0}$
- **for** $i \longleftarrow 0$ **up to** $n-1$ **do**
  - $\vec{\sigma}_i \longleftarrow \nu * e^{\tau \{\text{Gaussian random number}\}} * \vec{\sigma}'_i$
- **return** $\vec{\sigma}$

- Mutation of strategy parameters: apply lognormal mutation with

$$\tau_0 = \frac{c}{\sqrt{2n}} \tag{6}$$

Reproduction of Endogenous Parameters

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

$\vec{\sigma} \longleftarrow$ 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
  $\nu \longleftarrow e^{\tau_0\{\text{Gaussian random number}\}}$
  $\vec{\sigma} \longleftarrow \vec{0}$
  for $i \longleftarrow 0$ **up to** $n-1$ **do**
    $\vec{\sigma}_i \longleftarrow \nu * e^{\tau\{\text{Gaussian random number}\}} * \vec{\sigma}'_i$
  return $\vec{\sigma}$

- Mutation of strategy parameters: apply lognormal mutation with

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

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

Reproduction of Endogenous Parameters

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

$\vec{\sigma} \longleftarrow$ 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
  $\nu \longleftarrow e^{\tau_0\{\text{Gaussian random number}\}}$
  $\vec{\sigma} \longleftarrow \vec{0}$
  for $i \longleftarrow 0$ **up to** $n-1$ **do**
    $\vec{\sigma}_i \longleftarrow \nu * e^{\tau\{\text{Gaussian random number}\}} * \vec{\sigma}'_i$
  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

$\vec{\sigma} \longleftarrow$ 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
- $\nu \longleftarrow e^{\tau_0\{\text{Gaussian random number}\}}$
- $\vec{\sigma} \longleftarrow \vec{0}$
- **for** $i \longleftarrow 0$ **up to** $n-1$ **do**
  - $\vec{\sigma}_i \longleftarrow \nu * e^{\tau\{\text{Gaussian random number}\}} * \vec{\sigma}'_i$
- **return** $\vec{\sigma}$

- Mutation of strategy parameters: apply lognormal mutation with

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

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

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

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

Section Outline

CMA-ES

- Covariance Matrix Adaptation Evolution Strategy (CMA-ES) by Hansen et al. [16–23] in the mid-1990s

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]

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

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)

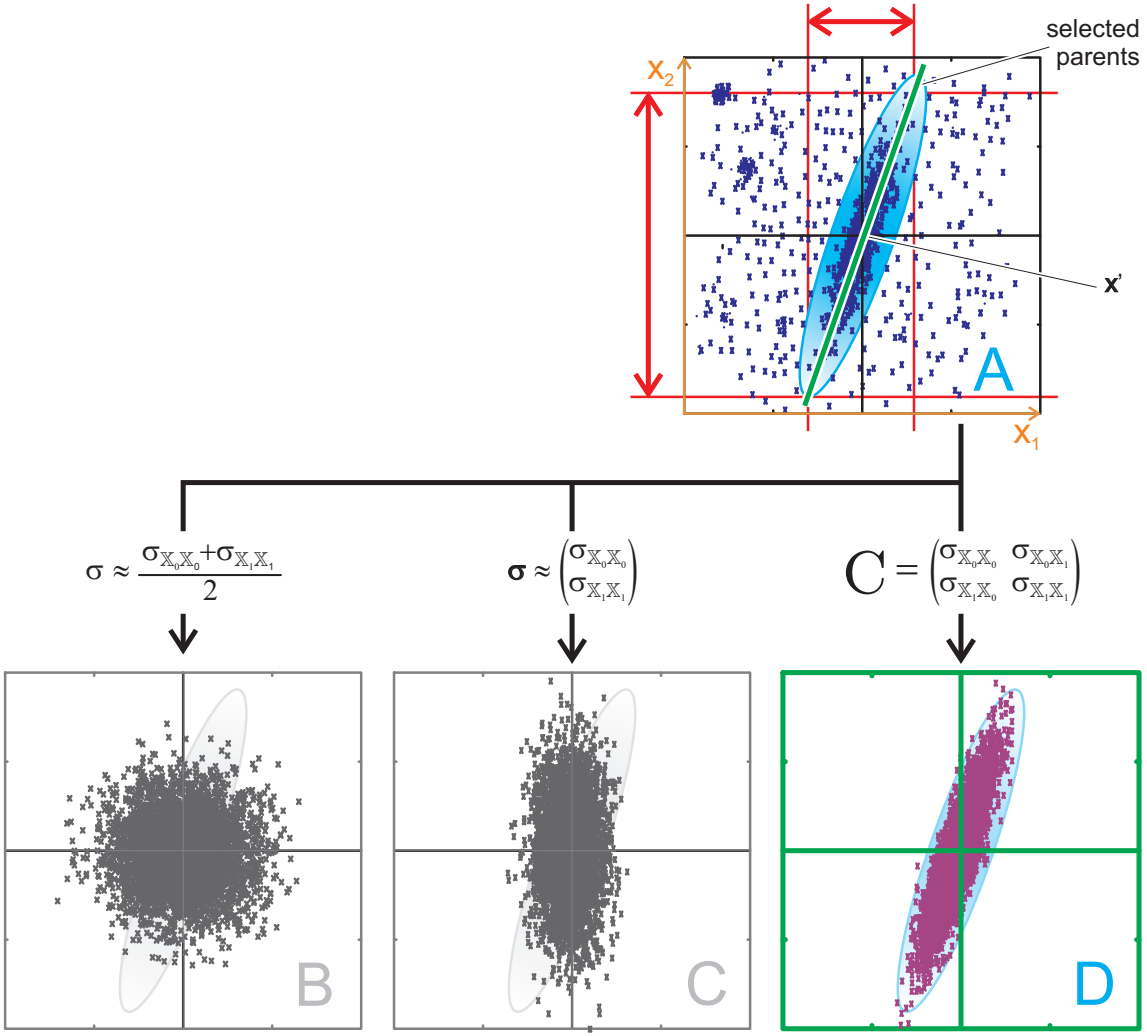
CMA-ES: Principles

- The new offspring population of each generation is sampled from a multi-variate normal distribution [25]

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

CMA-ES: Principles



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

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

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 μ)

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 μ)
- Both are exogenous parameters maintained centrally and updated by the algorithm

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 μ)
- Both are exogenous parameters maintained centrally and updated by the algorithm
  - The center μ 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 μ)
- Both are exogenous parameters maintained centrally and updated by the algorithm
  - The center μ 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 parets

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 μ)
- Both are exogenous parameters maintained centrally and updated by the algorithm
  - The center μ 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 parets
- New population is sampled from normal distribution, parent individuals are discarted (no traditional mutation/crossover)

CMA-ES: Invariances

- Using co-variance information allows to express relationships between variables

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")

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]

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]

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)

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), such as
    - rotation

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), such as
    - rotation
    - reflection, and

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), 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!)

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
- We want this! (Remember: Roulette-Wheel Selection versus Truncation Selection)

CMA-ES: Material

- Algorithm itself is a bit more complicated and will not be discussed here

CMA-ES: Material

- 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

CMA-ES: Material

- 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.,

CMA-ES: Material

- 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 [28]: `http://www.marmakoide.org/content/code/eskit.html`

CMA-ES: Material

- 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 [28]: `http://www.marmakoide.org/content/code/eskit.html`
  - Official source code page: `http://www.lri.fr/~hansen/cmaes_inmatlab.html`

CMA-ES: Material

- 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 [28]: `http://www.marmakoide.org/content/code/eskit.html`
  - Official source code page: `http://www.lri.fr/~hansen/cmaes_inmatlab.html`
- Much literature [16–23, 25] and comprehensive benchmark results [20, 22–24]

CMA-ES: Material

- 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 [28]: `http://www.marmakoide.org/content/code/eskit.html`
  - Official source code page: `http://www.lri.fr/~hansen/cmaes_inmatlab.html`
- Much literature [16–23, 25] and comprehensive benchmark results [20, 22–24]
- A dedicated website: `http://www.lri.fr/~hansen/cmaesintro.html`

CMA-ES: Material

- 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 [28]: `http://www.marmakoide.org/content/code/eskit.html`
  - Official source code page: `http://www.lri.fr/~hansen/cmaes_inmatlab.html`
- Much literature [16–23, 25] and comprehensive benchmark results [20, 22–24]
- A dedicated website: `http://www.lri.fr/~hansen/cmaesintro.html`
- An own Wikipedia [29] page: `https://en.wikipedia.org/wiki/CMA-ES`

CMA-ES: Material

- 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 [28]: `http://www.marmakoide.org/content/code/eskit.html`
  - Official source code page: `http://www.lri.fr/~hansen/cmaes_inmatlab.html`
- Much literature [16–23, 25] and comprehensive benchmark results [20, 22–24]
- A dedicated website: `http://www.lri.fr/~hansen/cmaesintro.html`
- An own Wikipedia [29] page: `https://en.wikipedia.org/wiki/CMA-ES`
- A tutorial: [25]

CMA-ES: Material

- 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 [28]: `http://www.marmakoide.org/content/code/eskit.html`
  - Official source code page: `http://www.lri.fr/~hansen/cmaes_inmatlab.html`
- Much literature [16–23, 25] and comprehensive benchmark results [20, 22–24]
- A dedicated website: `http://www.lri.fr/~hansen/cmaesintro.html`
- An own Wikipedia [29] page: `https://en.wikipedia.org/wiki/CMA-ES`
- A tutorial: [25]
- When doing numerical optimization, this is the way to go!

Section Outline

Summary

- Evolution Strategies are Evolutionary Algorithms for (usually) $\mathbb{G} \subseteq \mathbb{R}^n$

Summary

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

Summary

- 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

Summary

- 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

Summary

- 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

Summary

- 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., $1/5$th rule

Summary

- 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., $1/5$th rule
- Endogenous method: information encoded in individuals

Summary

- 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., $1/5$th rule
- Endogenous method: information encoded in individuals
- Intermediate crossover, Dominant crossover

Summary

- 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., $1/5$th rule
- Endogenous method: information encoded in individuals
- Intermediate crossover, Dominant crossover
- Log-Normal Parameter Mutation

Summary

- 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., $1/5$th 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

Caspar David Friedrich, “Der Wanderer über dem Nebelmeer”, 1818
http://en.wikipedia.org/wiki/Wanderer_above_the_Sea_of_Fog

Bibliography

Bibliography I

1. Ingo Rechenberg. *Cybernetic Solution Path of an Experimental Problem*. Farnborough, Hampshire, UK: Royal Aircraft Establishment, August 1965. Library Translation 1122.
2. Ingo Rechenberg. *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. PhD thesis, Berlin, Germany: Technische Universität Berlin, 1971. URL `http://books.google.de/books?id=QcNNGQAACAAJ`.
3. Ingo Rechenberg. *Evolutionsstrategie '94*, volume 1 of *Werkstatt Bionik und Evolutionstechnik*. Bad Cannstadt, Stuttgart, Baden-Württemberg, Germany: Frommann-Holzboog Verlag, 1994. ISBN 3-7728-1642-8 and 978-3-772-81642-0. URL `http://books.google.de/books?id=savAAAACAAJ`.
4. Hans-Paul Schwefel. Kybernetische evolution als strategie der exprimentellen forschung in der strömungstechnik. Master's thesis, Berlin, Germany: Technische Universität Berlin, 1965.
5. Hans-Paul Schwefel. Experimentelle optimierung einer zweiphasendüse teil i. Technical Report 35, Berlin, Germany: AEG Research Institute, 1968. Project MHD–Staustrahlrohr 11.034/68.
6. Hans-Paul Schwefel. *Evolutionsstrategie und numerische Optimierung*. PhD thesis, Berlin, Germany: Technische Universität Berlin, Institut für Meß- und Regelungstechnik, Institut für Biologie und Anthropologie, 1975.
7. John Henry Holland. Outline for a logical theory of adaptive systems. *Journal of the Association for Computing Machinery (JACM)*, 9(3):297–314. doi: 10.1145/321127.321128.
8. John Henry Holland. Adaptive plans optimal for payoff-only environments. In *Proceedings of the Second Hawaii International Conference on System Sciences (HICSS'69)*, pages 917–920, Honolulu, HI, USA: University of Hawaii at Manoa, January 22–24, 1969. Amsterdam, The Netherlands: North-Holland Scientific Publishers Ltd. DTIC Accession Number: AD0688839.
9. John Henry Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. Ann Arbor, MI, USA: University of Michigan Press, 1975. ISBN 0-472-08460-7 and 978-0-472-08460-9. URL `http://books.google.de/books?id=JE5RAAAAMAAJ`.
10. John Henry Holland. Nonlinear environments permitting efficient adaptation. In Julius T. Tou, editor, *Proceedings of the Symposium on Computer and Information Sciences II*, pages 147–164, Columbus, OH, USA, August 22–24, 1966. London, New York: Academic Press.
11. Kenneth Alan De Jong. *An Analysis of the Behavior of a Class of Genetic Adaptive Systems*. PhD thesis, Ann Arbor, MI, USA: University of Michigan, August 1975. URL `http://cs.gmu.edu/~eclab/kdj_thesis.html`.
12. Stefan Niemczyk and Thomas Weise. A general framework for multi-model estimation of distribution algorithms. Technical report, Kassel, Hesse, Germany: University of Kassel, Fachbereich 16: Elektrotechnik/Informatik, Distributed Systems Group, March 10, 2010.

Bibliography II

13. Thomas Weise, Stefan Niemczyk, Raymond Chiong, and Mingxu Wan. A framework for multi-model edas with model recombination. In *Proceedings of the 4th European Event on Bio-Inspired Algorithms for Continuous Parameter Optimisation (EvoNUM'11), Applications of Evolutionary Computation – Proceedings of EvoApplications 2011: EvoCOMPLEX, EvoGAMES, EvoIASP, EvoINTELLIGENCE, EvoNUM, and EvoSTOC, Part 1 (EvoAPPLICATIONS'11)*, volume 6624 of *Lecture Notes in Computer Science (LNCS)*, pages 304–313, Torino, Italy, April 27–29, 2011. Berlin, Germany: Springer-Verlag GmbH. doi: 10.1007/978-3-642-20525-5_31.
14. Hans-Georg Beyer. *The Theory of Evolution Strategies*. Natural Computing Series. New York, NY, USA: Springer New York, May 27, 2001. ISBN 3-540-67297-4 and 978-3-540-67297-5. URL http://books.google.de/books?id=8tbInLufkTMC.
15. Hans-Georg Beyer and Hans-Paul Schwefel. Evolution strategies – a comprehensive introduction. *Natural Computing: An International Journal*, 1(1):3–52, March 2002. doi: 10.1023/A:1015059928466. URL http://www.cs.bham.ac.uk/~pxt/NIL/es.pdf.
16. Nikolaus Hansen, Andreas Ostermeier, and Andreas Gawelczyk. On the adaptation of arbitrary normal mutation distributions in evolution strategies: The generating set adaptation. In Larry J. Eshelman, editor, *Proceedings of the Sixth International Conference on Genetic Algorithms (ICGA'95)*, pages 57–64, Pittsburgh, PA, USA: University of Pittsburgh, July 15–19, 1995. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.29.9321.
17. Nikolaus Hansen and Andreas Ostermeier. Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation. In Keisoku Jidō and Seigyo Gakkai, editors, *Proceedings of IEEE International Conference on Evolutionary Computation (CEC'96)*, pages 312–317, Nagoya, Aichi, Japan: Nagoya University, Symposium & Toyoda Auditorium, May 20–22, 1996. Los Alamitos, CA, USA: IEEE Computer Society Press. URL http://www.lri.fr/~hansen/CMAES.pdf.
18. Nikolaus Hansen and Andreas Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2):159–195, 2001. URL http://www.bionik.tu-berlin.de/user/niko/cmaartic.pdf.
19. Nikolaus Hansen, Sibylle D. Müller, and Petros Koumoutsakos. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (cma-es). *Evolutionary Computation*, 11(1):1–18, Spring 2003. doi: 10.1162/106365603321828970. URL http://mitpress.mit.edu/journals/pdf/evco_11_1_1_0.pdf.

Bibliography III

20. Nikolaus Hansen and Stefan Kern. Evaluating the cma evolution strategy on multimodal test functions. In Xin Yao, Edmund K. Burke, José Antonio Lozano, Jim Smith, Juan Julián Merelo-Guervós, John A. Bullinaria, Jonathan E. Rowe, Peter Tiňo, Ata Kabán, and Hans-Paul Schwefel, editors, *Proceedings of the 8th International Conference on Parallel Problem Solving from Nature (PPSN VIII)*, volume 3242/2004 of *Lecture Notes in Computer Science (LNCS)*, pages 282–291, Birmingham, UK, September 18–22, 2008. Berlin, Germany: Springer-Verlag GmbH. doi: 10.1007/978-3-540-30217-9_29. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.69.163.
21. Nikolaus Hansen. The cma evolution strategy: A comparing review. In José Antonio Lozano, Pedro Larrañaga, Iñaki Inza, and Endika Bengoetxea, editors, *Towards a New Evolutionary Computation – Advances on Estimation of Distribution Algorithms*, volume 192/2006 of *Studies in Fuzziness and Soft Computing*, pages 75–102. Berlin, Germany: Springer-Verlag GmbH, 2006. URL http://www.lri.fr/~hansen/hansenedacomparing.pdf.
22. Anne Auger and Nikolaus Hansen. A restart cma evolution strategy with increasing population size. In David Wolfe Corne, Zbigniew Michalewicz, Robert Ian McKay, Ágoston E. Eiben, David B. Fogel, Carlos M. Fonseca, Günther R. Raidl, Kay Chen Tan, and Ali M. S. Zalzala, editors, *Proceedings of the IEEE Congress on Evolutionary Computation (CEC'05)*, pages 1769–1776, Edinburgh, Scotland, UK, September 2–5, 2005. Piscataway, NJ, USA: IEEE Computer Society. doi: 10.1109/CEC.2005.1554902. URL http://www.lri.fr/~hansen/cec2005ipopcmaes.pdf.
23. Anne Auger and Nikolaus Hansen. Performance evaluation of an advanced local search evolutionary algorithm. In David Wolfe Corne, Zbigniew Michalewicz, Robert Ian McKay, Ágoston E. Eiben, David B. Fogel, Carlos M. Fonseca, Günther R. Raidl, Kay Chen Tan, and Ali M. S. Zalzala, editors, *Proceedings of the IEEE Congress on Evolutionary Computation (CEC'05)*, volume 2, pages 1777–1784, Edinburgh, Scotland, UK, September 2–5, 2005. Piscataway, NJ, USA: IEEE Computer Society. doi: 10.1109/CEC.2005.1554903. URL http://www.lri.fr/~hansen/cec2005localcmaes.pdf.

Bibliography IV

24. Nikolaus Hansen. Benchmarking a bi-population cma-es on the bbob-2009 function testbed. In Franz Rothlauf, Günther R. Raidl, Anna Isabel Esparcia-Alcázar, Ying-Ping Chen, Gabriela Ochoa, Ender Ozcan, Marc Schoenauer, Anne Auger, Hans-Georg Beyer, Nikolaus Hansen, Steffen Finck, Raymond Ros, L. Darrell Whitley, Garnett Wilson, Simon Harding, William Benjamin Langdon, Man Leung Wong, Laurence D. Merkle, Frank W. Moore, Sevan G. Ficici, William Rand, Rick L. Riolo, Nawwaf Kharma, William R. Buckley, Julian Francis Miller, Kenneth Owen Stanley, Jaume Bacardit i Peñarroya, Will N. Browne, Jan Drugowitsch, Nicola Beume, Mike Preuß, Stephen L. Smith, Stefano Cagnoni, Alexandru Floares, Aaron Baughman, Steven Matt Gustafson, Maarten Keijzer, Arthur Kordon, and Clare Bates Congdon, editors, *Proceedings of the 11th Annual Conference – Companion on Genetic and Evolutionary Computation Conference (GECCO'09)*, pages 2389–2395, Montréal, QC, Canada: Delta Centre-Ville Hotel, July 8–12, 2009. New York, NY, USA: Association for Computing Machinery (ACM). doi: 10.1145/1570256.1570333. URL http://hal.archives-ouvertes.fr/inria-00382093/en.
25. Nikolaus Hansen. *The CMA Evolution Strategy: A Tutorial*. Orsay, France: Université Paris Sud, Institut National de Recherche en Informatique et en Automatique (INRIA) Futurs, Équipe TAO, June 28, 2011. URL http://www.lri.fr/~hansen/cmatutorial.pdf.
26. Nikolaus Hansen, Raymond Ros, Nikolas Mauny, Marc Schoenauer, and Anne Auger. Impacts of invariance in search: When cma-es and pso face ill-conditioned and non-separable problems. 11(8):5755–5769, December 2011. doi: 10.1016/j.asoc.2011.03.001. URL hal.inria.fr/inria-00583669/PDF/hansen2011impacts.pdf. INRIA Report inria-00583669, version 1, 2011-04-06.
27. Nikolaus Hansen. The cma evolution strategy, February 13, 2012. URL http://www.lri.fr/~hansen/cmaesintro.html.
28. Alexandre Devert. *ESKit*. Oslo, Østlandet, Norway: Gitorious.org, 2012. URL http://www.marmakoide.org/content/code/eskit.html.
29. Wikipedia – the free encyclopedia, 2009. URL http://en.wikipedia.org/.