Artificial Life
Summer 2025

Evolutionary Algorithms 3

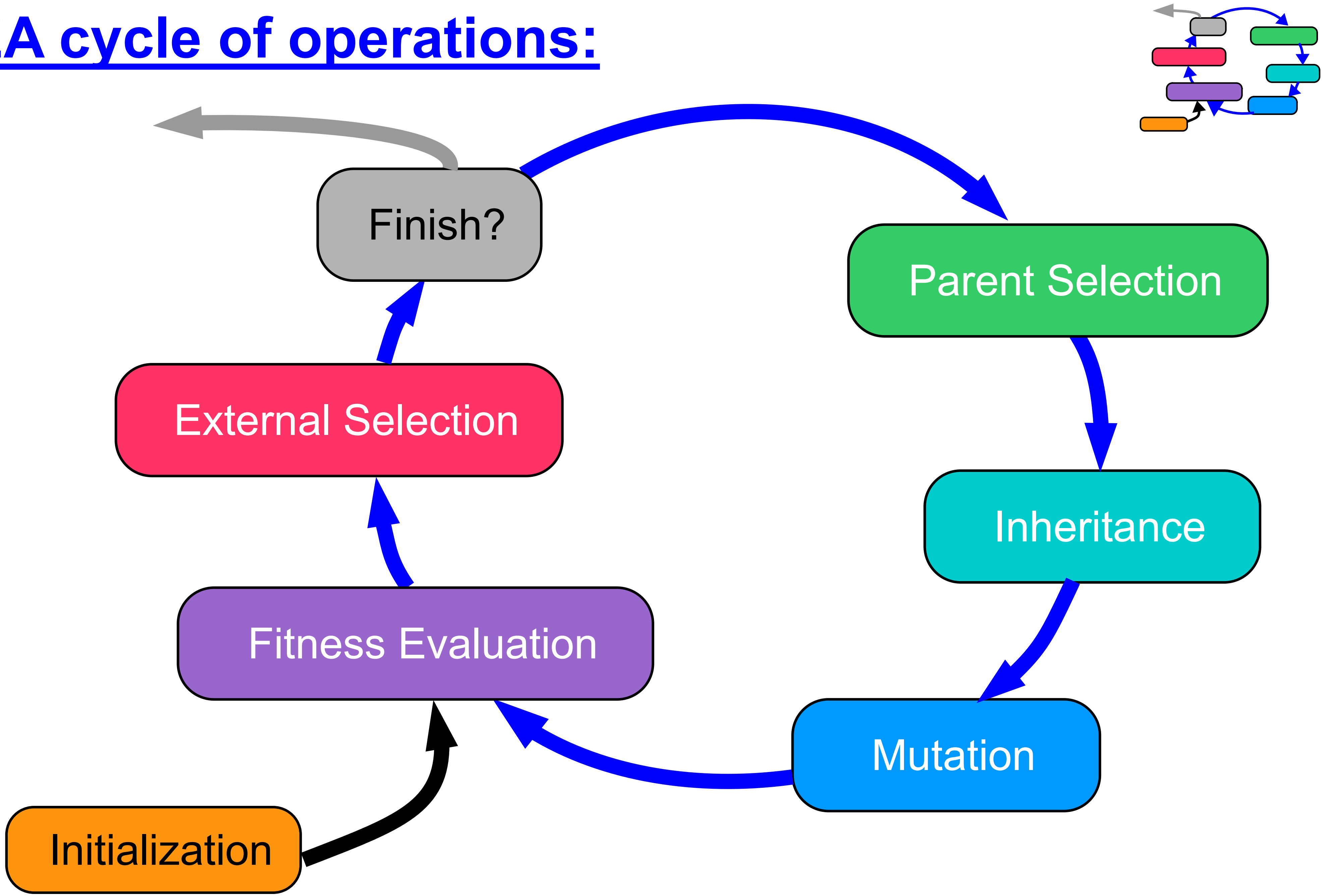
Master Computer Science [MA-INF 4201]
Mon 14:15 – 15:45, HSZ, HS-2

Dr. Nils Goerke, Autonomous Intelligent Systems,
Department of Computer Science, University of Bonn

Overview:

- Genome structure
- Example: 8 queens
- Super-individuals
- External-selection and parent-selection combined
- Probabilistic parent-selection
  - Wheel of fortune
  - Softmax selection
  - Tournament selection
- Genetic programming
- Co-evolution

EA cycle of operations:



Overview:

- **Genome structure**
- Super-individuals
- External-selection and parent-selection combined
- Probabilistic parent-selection
  - Wheel of fortune
  - Softmax selection
  - Tournament selection
- Genetic programming
- Co-evolution

EA: Genome Structure

Naive structuring of the genome:

“Normal” structuring of the genome:

Sophisticated structuring of the genome:

EA:

Genome Structure

Naive structuring of the genome:
Pro: easy to implement, only few knowledge necessary.
Con: large search space, a lot of local maxima possible,
may be hard or impossible to find a good solution.

"Normal" structuring of the genome:

Sophisticated structuring of the genome:

EA: Genome Structure

Naive structuring of the genome:
Pro: easy to implement, only few knowledge necessary.
Con: large search space, a lot of local maxima possible, may be hard or impossible to find a good solution.

"Normal" structuring of the genome:
Pro: still easy to implement, some knowledge necessary, wide variety to implement inheritance and mutation.
Con: still a lot of bad or illegal genomes possible.

Sophisticated structuring of the genome:

EA: Genome Structure

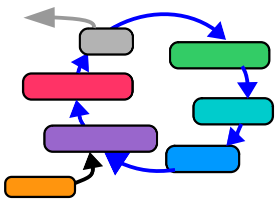
Naive structuring of the genome:
Pro: easy to implement, only few knowledge necessary.
Con: large search space, a lot of local maxima possible, may be hard or impossible to find a good solution.

“Normal” structuring of the genome:
Pro: still easy to implement, some knowledge necessary, wide variety to implement inheritance and mutation.
Con: still a lot of bad or illegal genomes possible.

Sophisticated structuring of the genome:
Pro: only legal, or good genomes are to be investigated.
Con: profound knowledge about the process and of the kind of possible solutions is required, can become computational expensive to implement inheritance and mutation.

EA:

Example: TSP



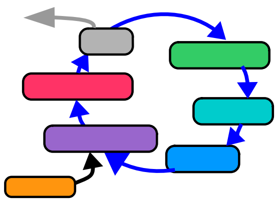
Objective:

find the shortest route visiting each point from a set of given points (cities) exactly once (Traveling Salesman Problem).

Genome:

Fitness Function:

EA: Example: TSP



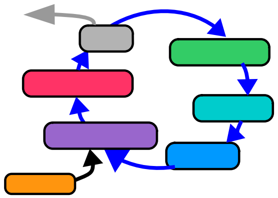
Objective:
find the shortest route visiting each point from a set of given points (cities) exactly once (Traveling Salesman Problem).

Genome:
an ordered set of points (cities, C_i), sequence of cities to be visited, containing each city exactly once.

g = $\{ C_3, C_{17}, C_i, C_{42}, \ldots \}$

Fitness Function:

EA: Example: TSP



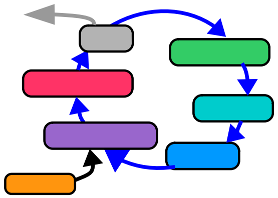
Objective:
find the shortest route visiting each point from a set of given points (cities) exactly once (Traveling Salesman Problem).

Genome:
an ordered set of points (cities, C_i), sequence of cities to be visited, containing each city exactly once.
$\mathbf{g} = \{ C_3, C_{17}, C_i, C_{42}, \ldots\}$

Fitness Function:
$\mathbf{f(g)}$ is the sum of all distances $\mathbf{d(C_i,C_{i+1})}$ between subsequent cities in the list (genome). Of course all distances $\mathbf{d(C_i,C_j)}$ between the cities i and j is required; either by a distance matrix or by a calculation from the coordinates.

EA:

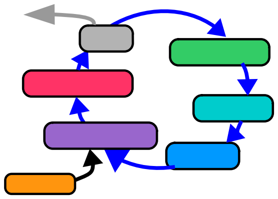
Example: TSP



Easy Inheritance:

Mutation:

EA: Example: TSP

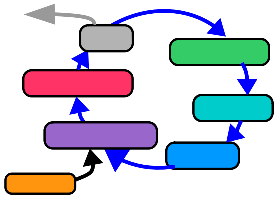


Easy Inheritance:

Since a classical cross over will generate illegal genomes the k=1, copy operator, omitting recombination is O.K.

Mutation:

EA: Example: TSP



Easy Inheritance:

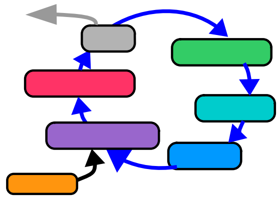
Since a classical cross over will generate illegal genomes the k=1, copy operator, omitting recombination is O.K.

Mutation:

exchange 2 cities within the sequence (genome),
or even a complete subsequence,
or invert the order of a subsequence within the genome.

EA:

Example: TSP



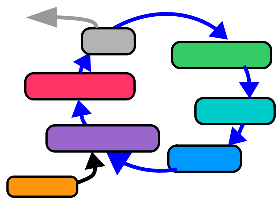
Easy Inheritance:
Since a classical cross over will generate illegal genomes the k=1, copy operator, omitting recombination is O.K.

Mutation:
exchange 2 cities within the sequence (genome),
or even a complete subsequence,
or invert the order of a subsequence within the genome.

g **=** { … , London, Rio, Moscow, New York, Paris, Tokyo, LA, ... }

EA:

Example: TSP



Easy Inheritance:

Since a classical cross over will generate illegal genomes the k=1, copy operator, omitting recombination is O.K.

Mutation:

exchange 2 cities within the sequence (genome),
or even a complete subsequence,
or invert the order of a subsequence within the genome.

g **=** { … , London, Rio, Moscow, New York, Paris, Tokyo, LA, ... }

g' **=** { … , London, Paris, Moscow, New York, Rio, Tokyo, LA, ... }

EA: Example: TSP

Easy Inheritance:

Since a classical cross over will generate illegal genomes the k=1, copy operator, omitting recombination is O.K.

Mutation:

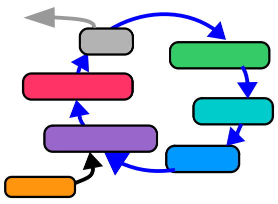
exchange 2 cities within the sequence (genome),
or even a complete subsequence,
or invert the order of a subsequence within the genome.

g = { … , London, Rio, Moscow, New York, Paris, Tokyo, LA, ... }

g' = { … , London, Paris, Moscow, New York, Rio, Tokyo, LA, ... }

g'' = { … , London, Tokyo, Rio, Moscow, New York, Paris, LA, ... }

EA: Example: TSP



Easy Inheritance:

Since a classical cross over will generate illegal genomes the k=1, copy operator, omitting recombination is O.K.

Mutation:

exchange 2 cities within the sequence (genome),
or even a complete subsequence,
or invert the order of a subsequence within the genome.

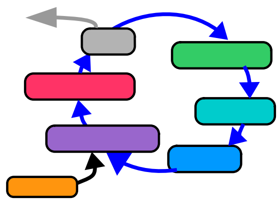
g = { … , London, Rio, Moscow, New York, Paris, Tokyo, LA, ... }

g' = { … , London, Paris, Moscow, New York, Rio, Tokyo, LA, ... }

g'' = { … , London, Tokyo, Rio, Moscow, New York, Paris, LA, ... }

g''' = { … , London, Paris, New York, Moscow, Rio, Tokyo, LA, ... }

EA: Example: TSP



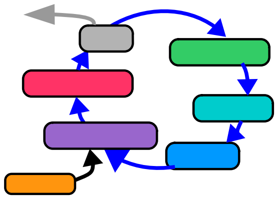
Sophisticated Inheritance:

A more sophisticated inheritance operator would have to guarantee that the offspring still represent legal genomes,
 - routes that visit each city exactly once - .
This may cause high computational effort,
which might be larger than the potential benefit.

Sophisticated Mutation:

EA:

Example: TSP



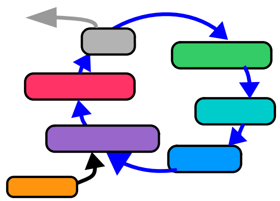
Sophisticated Inheritance:

A more sophisticated inheritance operator would have to guarantee that the offspring still represent legal genomes,
 - routes that visit each city exactly once - .
This may cause high computational effort,
which might be larger than the potential benefit.

Sophisticated Mutation:

a more sophisticated mutation operator might take the fitness value f(g) of the genome g into account, to make only those changes to the genome that have a high probability for a fitness increase.

EA: Example: TSP



Sophisticated Inheritance:

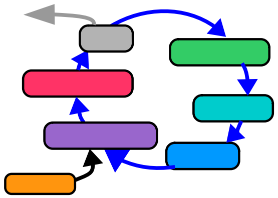
A more sophisticated inheritance operator would have to guarantee that the offspring still represent legal genomes,
- routes that visit each city exactly once - .
This may cause high computational effort,
which might be larger than the potential benefit.

Sophisticated Mutation:

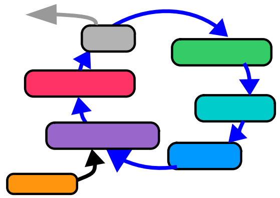
a more sophisticated mutation operator might take the fitness value f(g) of the genome g into account, to make only those changes to the genome that have a high probability for a fitness increase.
How do you know?
Either by theory (if available), or by a heuristics.

EA:

Example: 8 Queens



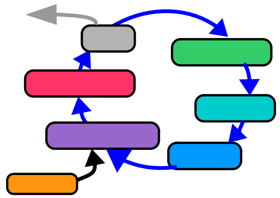
EA: Example: 8 Queens



The 8 queens puzzle with an evolutionary algorithm:

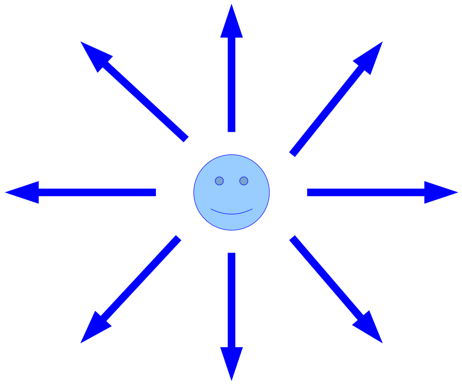
The task of the 8 queens puzzle is to place 8 queens on a chess board (8x8) so that they can not reach each other.

EA: Example: 8 Queens

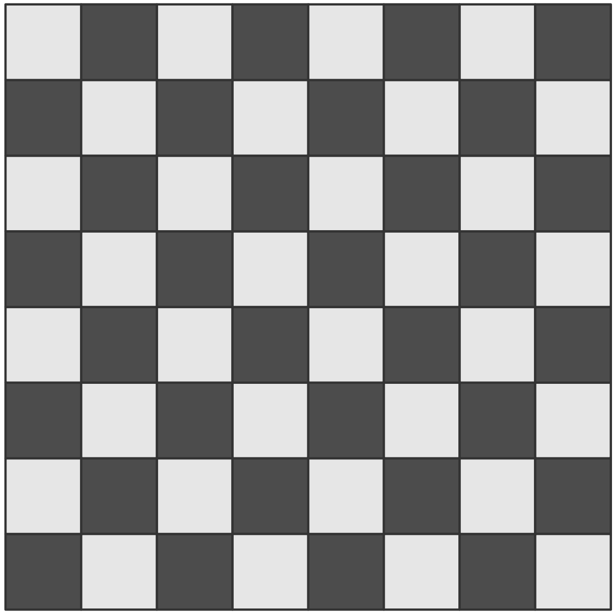


The 8 queens puzzle with an evolutionary algorithm:

The task of the 8 queens puzzle is to place 8 queens on a chess board (8x8) so that they can not reach each other.



allowed moves for a queen

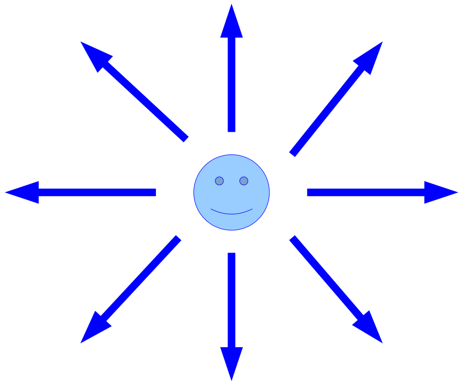


chess board with 8x8 = 64 positions

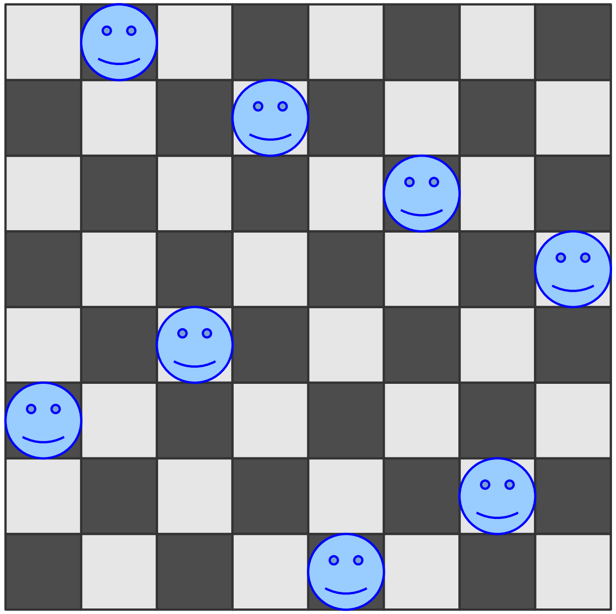
EA: Example: 8 Queens

The 8 queens puzzle with an evolutionary algorithm:

The task of the 8 queens puzzle is to place 8 queens on a chess board (8x8) so that they can not reach each other.



allowed moves for a queen



One of the **92** solutions

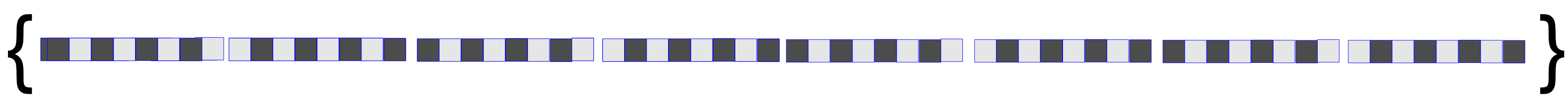
chess board with 8x8 = 64 positions

EA: Example: 8 Queens

Very naive implementation of the genome:

a 64 bit binary vector; queen is 1, no queen is 0;
more than 8 queens are possible.

This very naive implementation is generating an extremely large search space with 2^64 possible genomes. This is beyond any computing power to be investigated in total.

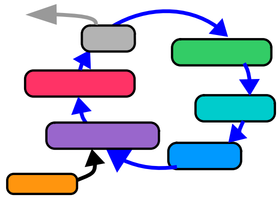


Semi naive implementation of the genome:

a 64 bit binary vector; queen is 1, no queen is 0;
exactly 8 queens == 8 bits set are possible.

Still the resulting search space is very large.

EA: Example: 8 Queens



Normal/Sophisticated implementation of the genome:

8 rows of 8 bit binary vectors; queen is 1, no queen is 0; each row contains exactly one queen.

This is reducing the search space to 8^8 = 16777216 possibilities, (which can be managed by brute-force).

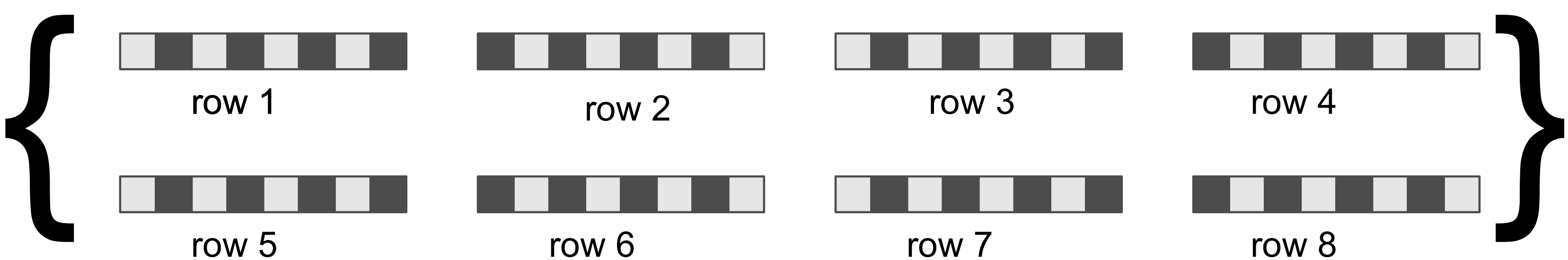
EA: Example: 8 Queens

Normal/Sophisticated implementation of the genome:

8 rows of 8 bit binary vectors; queen is 1, no queen is 0;
each row contains exactly one queen.

This is reducing the search space to $8^8 = 16777216$ possibilities, (which can be managed by brute-force).

Genome =



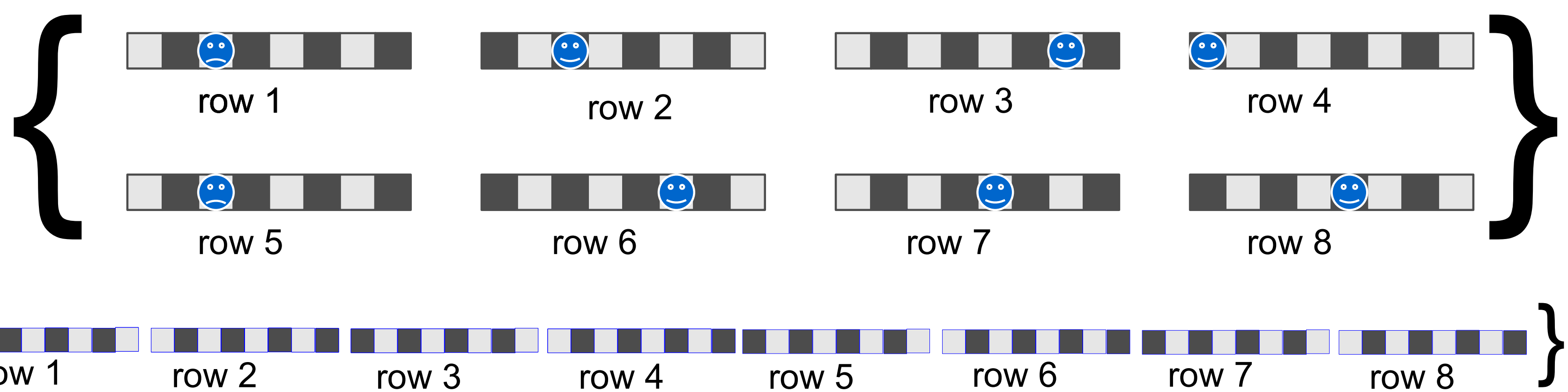
EA: Example: 8 Queens

Normal/Sophisticated implementation of the genome:

8 rows of 8 bit binary vectors; queen is 1, no queen is 0;
each row contains exactly one queen.

This is reducing the search space to $8^8 = 16777216$ possibilities, (which can be managed by brute-force).

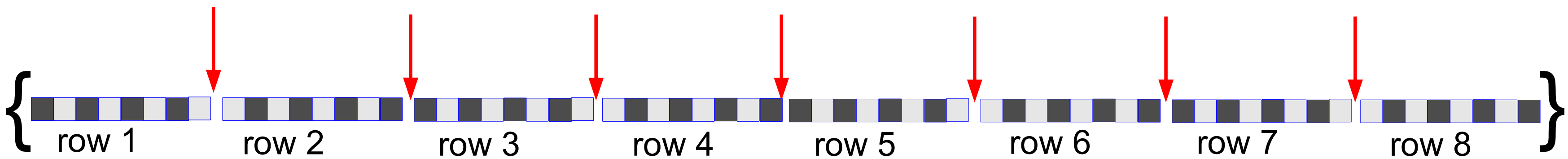
Genome =



EA: Example: 8 Queens

Inheritance, recombination, 1 point cross over
cross-over points only between the rows

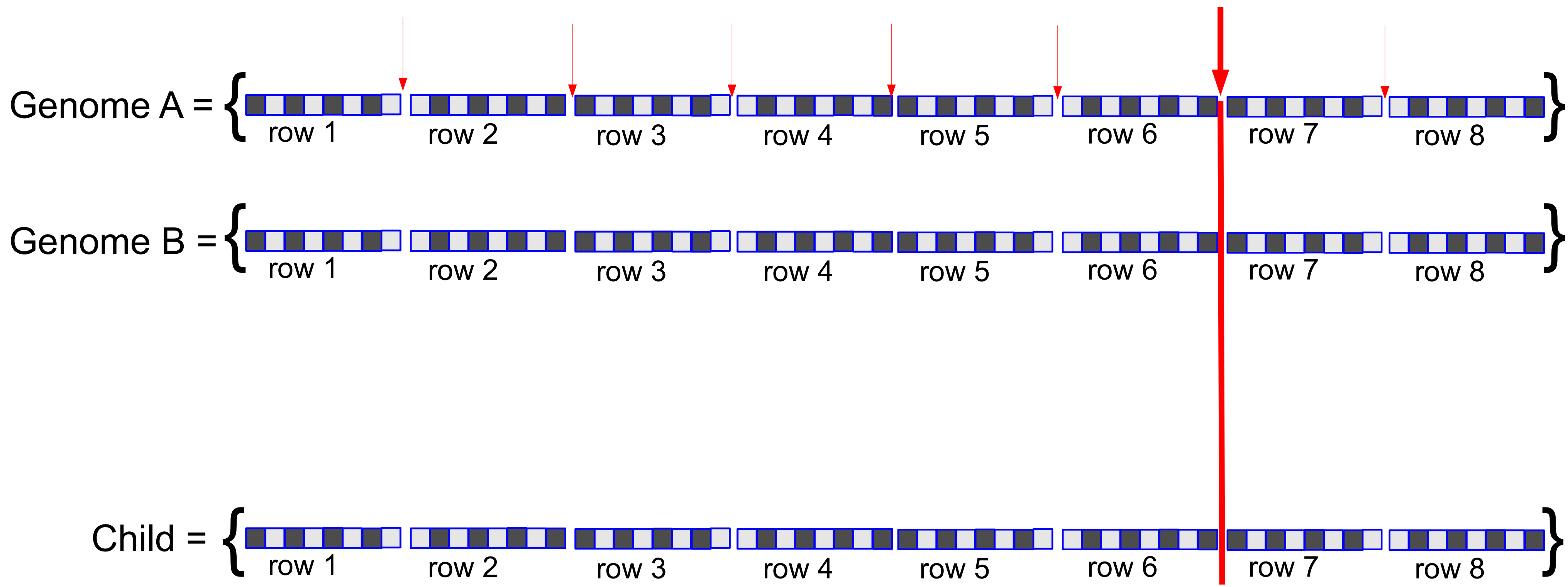
Genome A =



EA:

Example: 8 Queens

Inheritance, recombination, 1 point cross over
cross-over points only between the rows

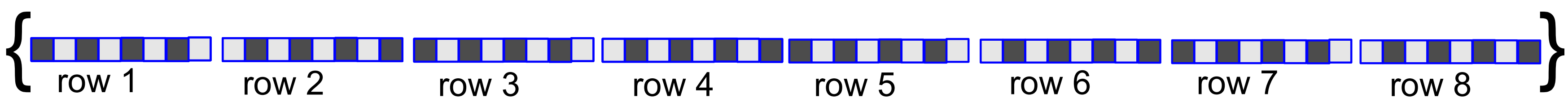


EA:

Example: 8 Queens

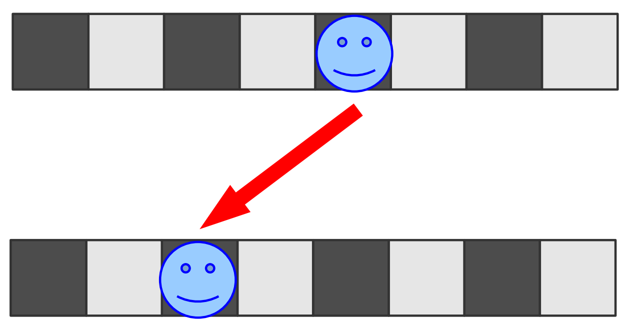
Mutation only the position of the queen within the row is altered

Genome A =



Chose a random row **r** (1, …, 8):
and pick a new random position (1, …, 8) for the queen

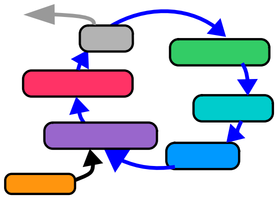
before mutation row **3**



after mutation row **3**

EA:

Example: 8 Queens



A more sophisticated implementation of the genome:

If you have an even more sophisticated idea for structuring the genome, don't forget to shape the inheritance and mutation operators and the fitness function accordingly.

Go ahead,

feel free to do experiments

EA:

Example: 8 Queens

The objective, is to find a placement of the 8 queens so that they can't reach each other.

The fitness function for the EA should reflect this:

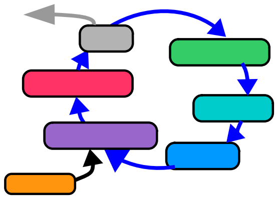
a large value of **f** if the placement is o.k. **I**
a small value of **f** if the placement is not o.k. **O**

A fitness function that yields a binary value (**O**,**I**) is not a good idea for an evolutionary algorithm.

The resulting fitness surface is flat (**O**), with only a few isolated peeks (**I**).

The value of this kind of fitness function is not reflecting, that a genome can be close to a possible solution.

EA: Example: 8 Queens



The objective, is to find a placement of the 8 queens so that they can't reach each other.

An appropriate fitness function for an EA must be shaped accordingly;

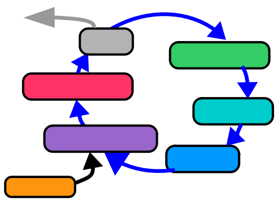
In addition, the fitness function **f** value should implement, that genomes **g,** close to an optimal solution should yield a larger fitness value **f(g)** than those far away.

The optimal would be, to have a fitness function **f(g)** that is proportional to the distance between the genome **g** to an optimal genome **g***.

Unfortunately this is not possible in general.

EA:

Example: 8 Queens



The objective, is to find a placement of the 8 queens so that they can't reach each other.

A proposal for a fitness function for the 8 queens problem:

Each possibilty that one queen can attack another queen is counted as **-1**
The fitness value **f(g)** is the sum of all attack possibilities.

This yields a graded response as required,
with a maximal value **f(g*) = 0.0** when no attack is possible.

Caution:
This fitness function is only appropriate if the number of queens is fixed to 8 *(0 queens => 0 attack possibilities)*

EA:

Example:

Implementing a task using an Evolutionary Algorithm, a series of structuring decision will be necessary.

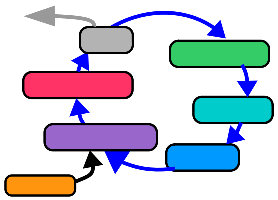
The sequence below is not common theory, but just my personal choice of doing it.

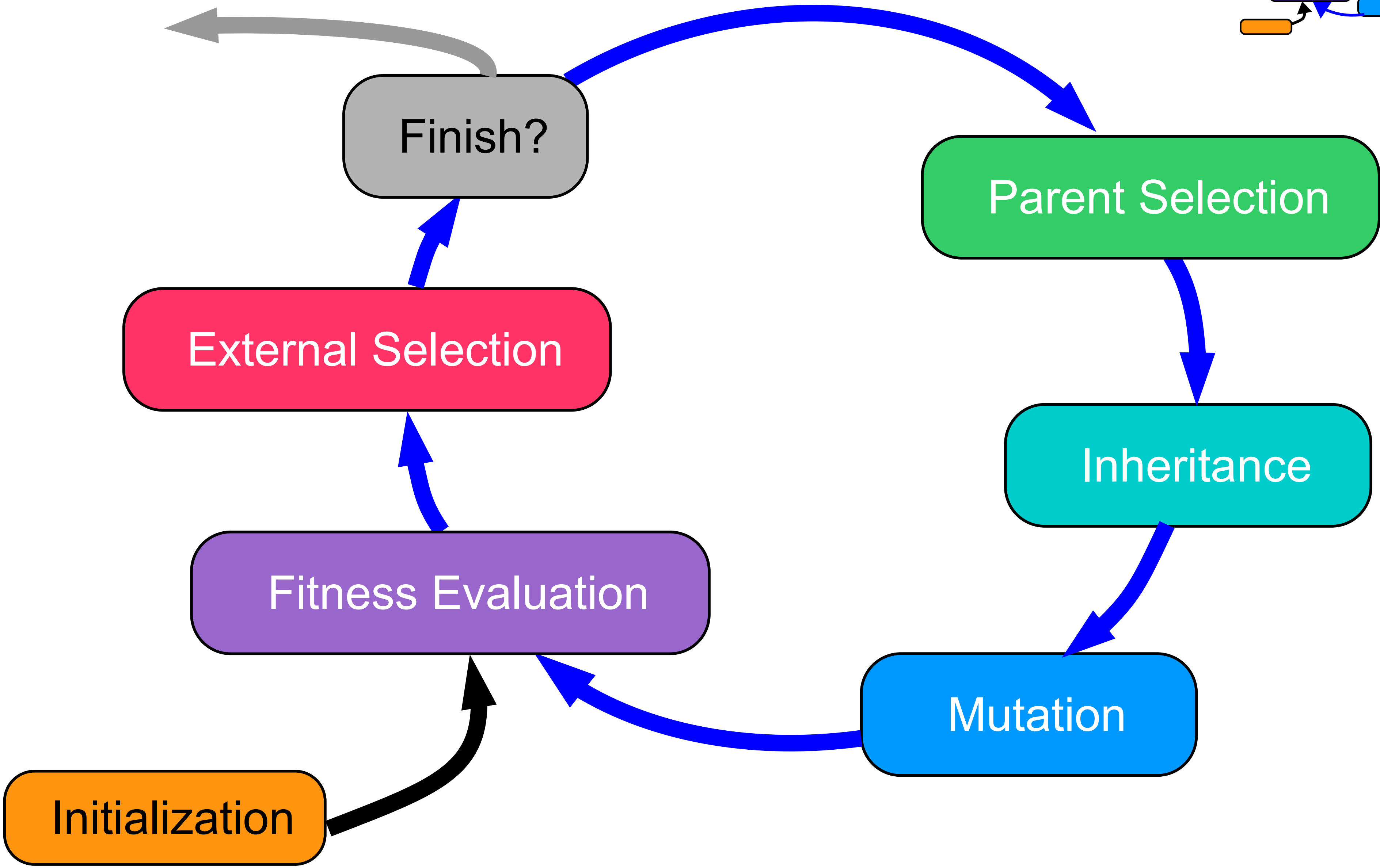
Objective:	specify the objective
Genome:	structure the genome
Fitness Function:	define an appropriate fitness function
Inheritance:	layout the inheritance process
Mutation:	layout the mutation process
Selection Strategy:	specify the selection strategy

Overview:

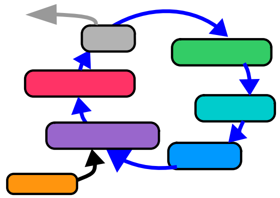
- Genome structure
- **Super-individuals**
- External-selection and parent-selection combined
- Probabilistic parent-selection
  - Wheel of fortune
  - Softmax selection
  - Tournament selection
- Genetic programming
- Co-evolution

EA cycle of operations:



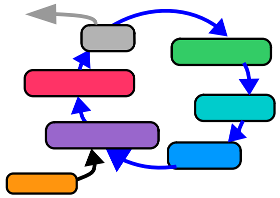


EA: Super-Individual



Sometimes, the combination of selection and inheritance is producing a so called: **Super-Individual.**

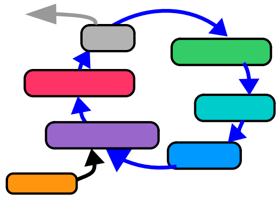
EA: Super-Individual



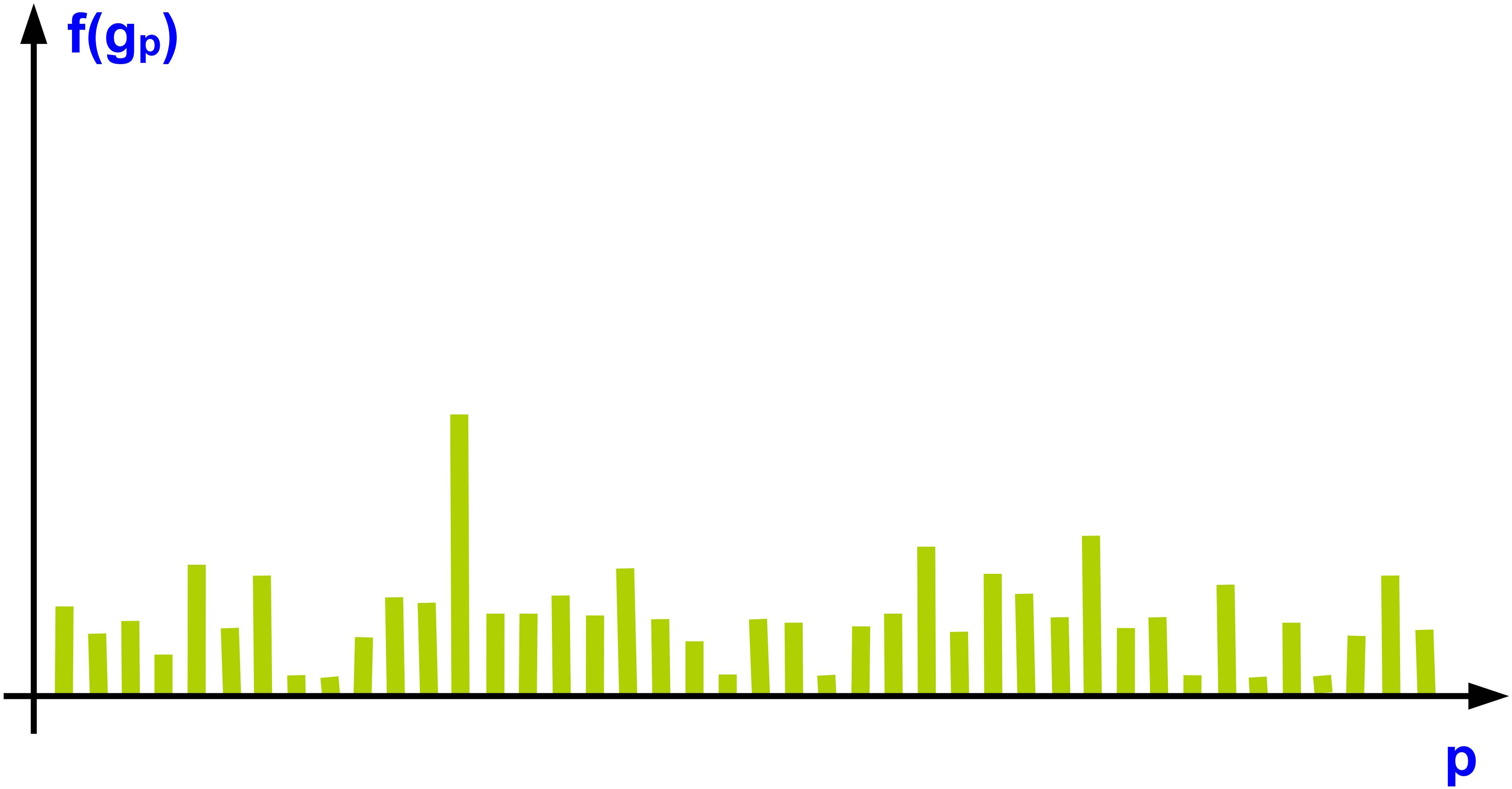
Sometimes, the combination of selection and inheritance is producing a so called: **Super-Individual.**

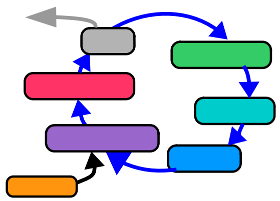
The fitness values throughout the population will typically vary.

EA: Super-Individual



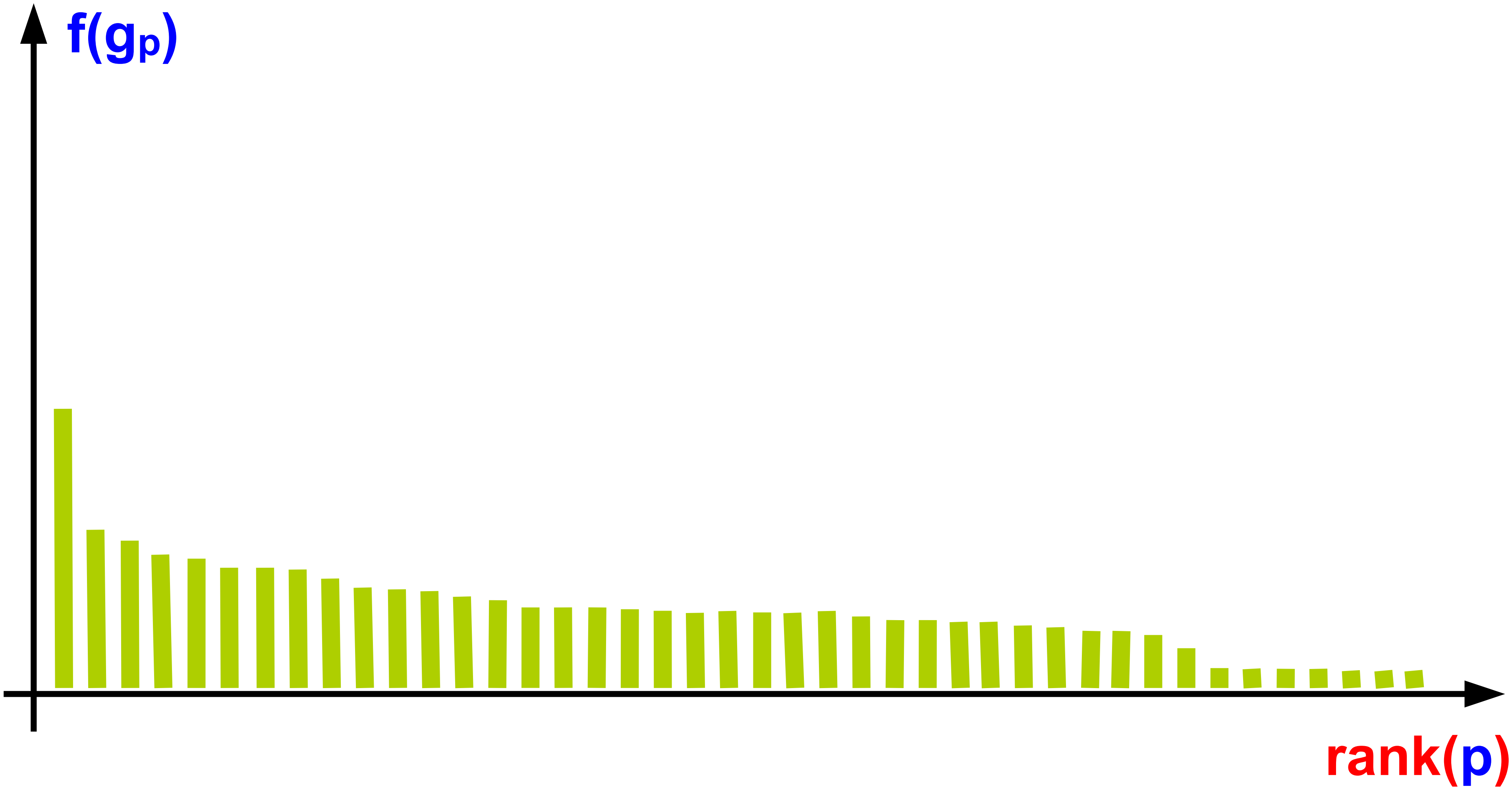
Fitness values will vary within the population.

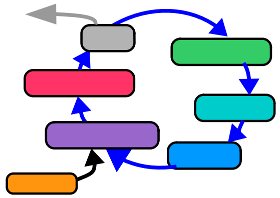




EA: Super-Individual

Sorted fitness values.

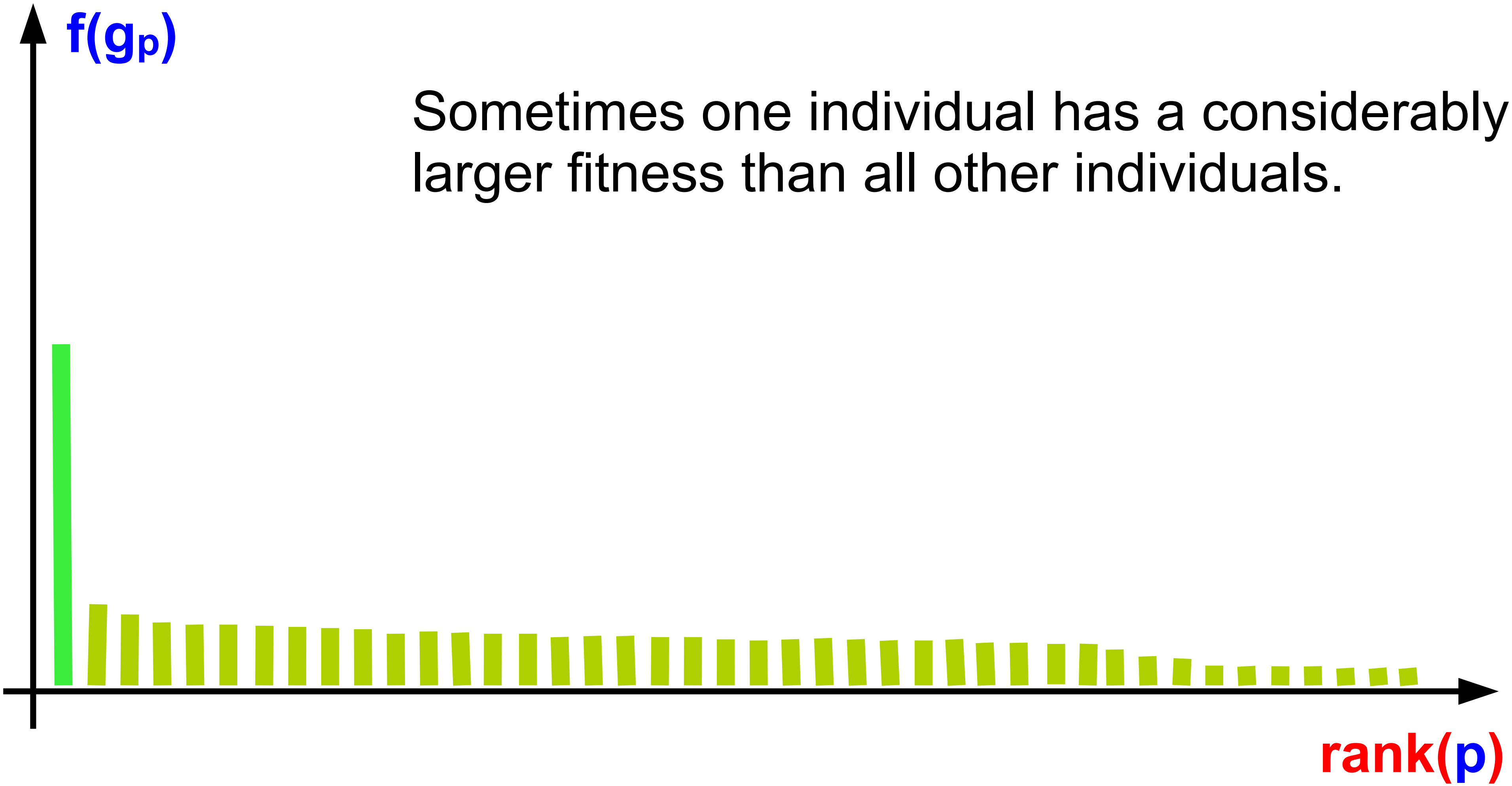




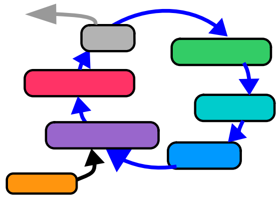
EA: Super-Individual

Sorted fitness values.

Sometimes one individual has a considerably larger fitness than all other individuals.



EA: Super-Individual

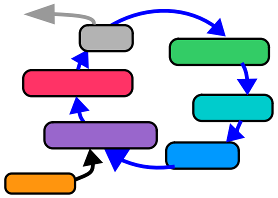


Sometimes, the combination of selection and inheritance is producing a so called: **Super-Individual.**

The fitness values throughout the population will typically vary.

In the case, that one individual has a considerably larger fitness than all other individuals, it can happen that this individual is selected more often
(e.g. fitness proportional selection, μ **<<** λ , …).

EA: Super-Individual



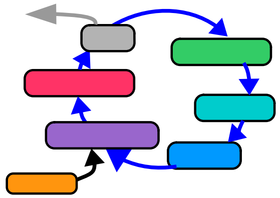
Sometimes, the combination of selection and inheritance is producing a so called: **Super-Individual.**

The fitness values throughout the population will typically vary.

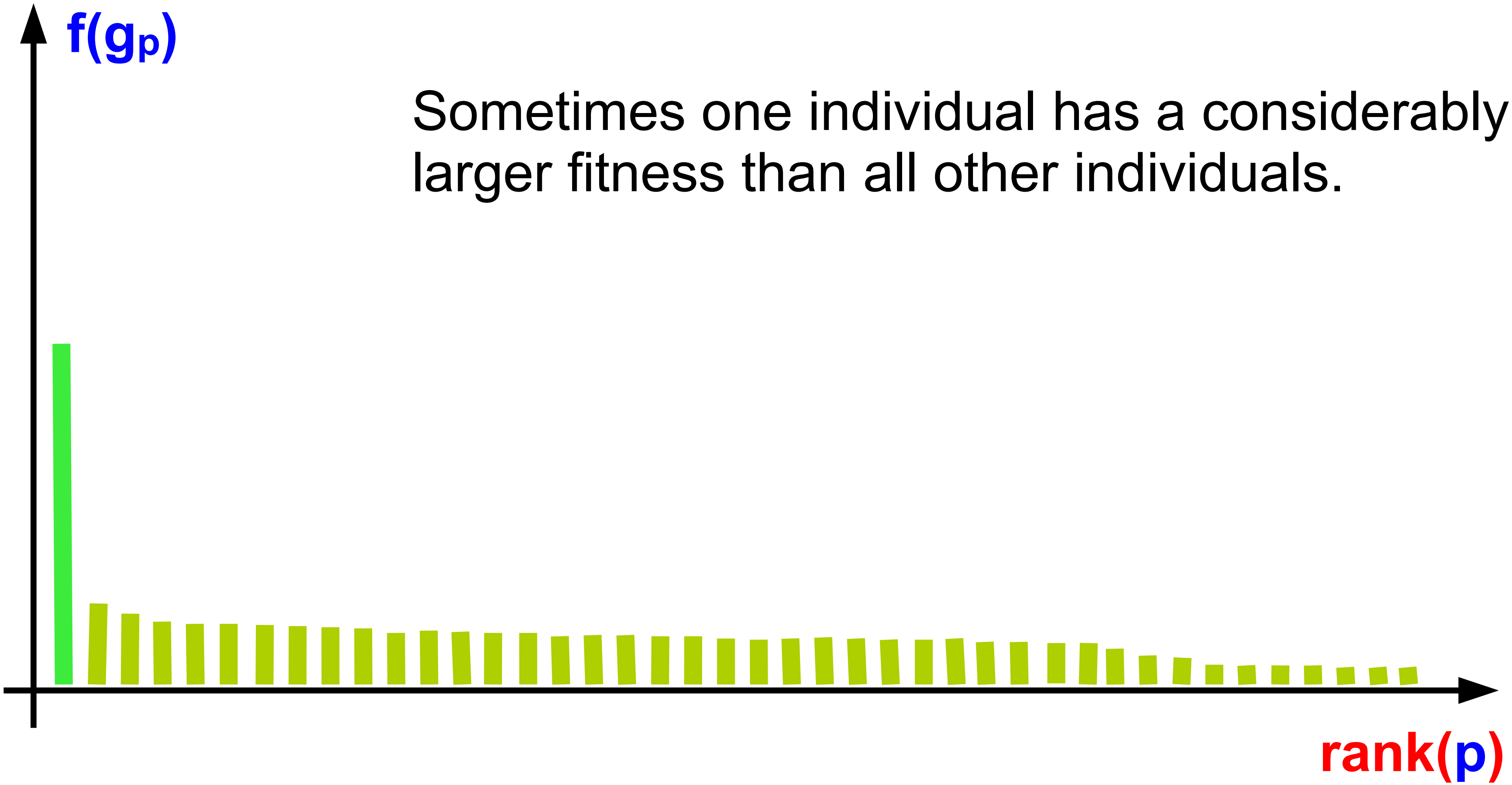
In the case, that one individual has a considerably larger fitness than all other individuals, it can happen that this individual is selected more often
(e.g. fitness proportional selection, μ **<<** λ , …).

Thus, it will produce more offspring than other individuals, increasing the number of individuals with a high fitness; which is explicitly intended by the principle of inheritance.

EA: Super-Individual

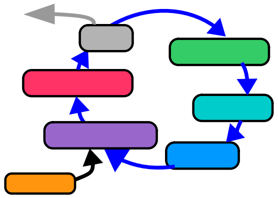


Sorted fitness values.

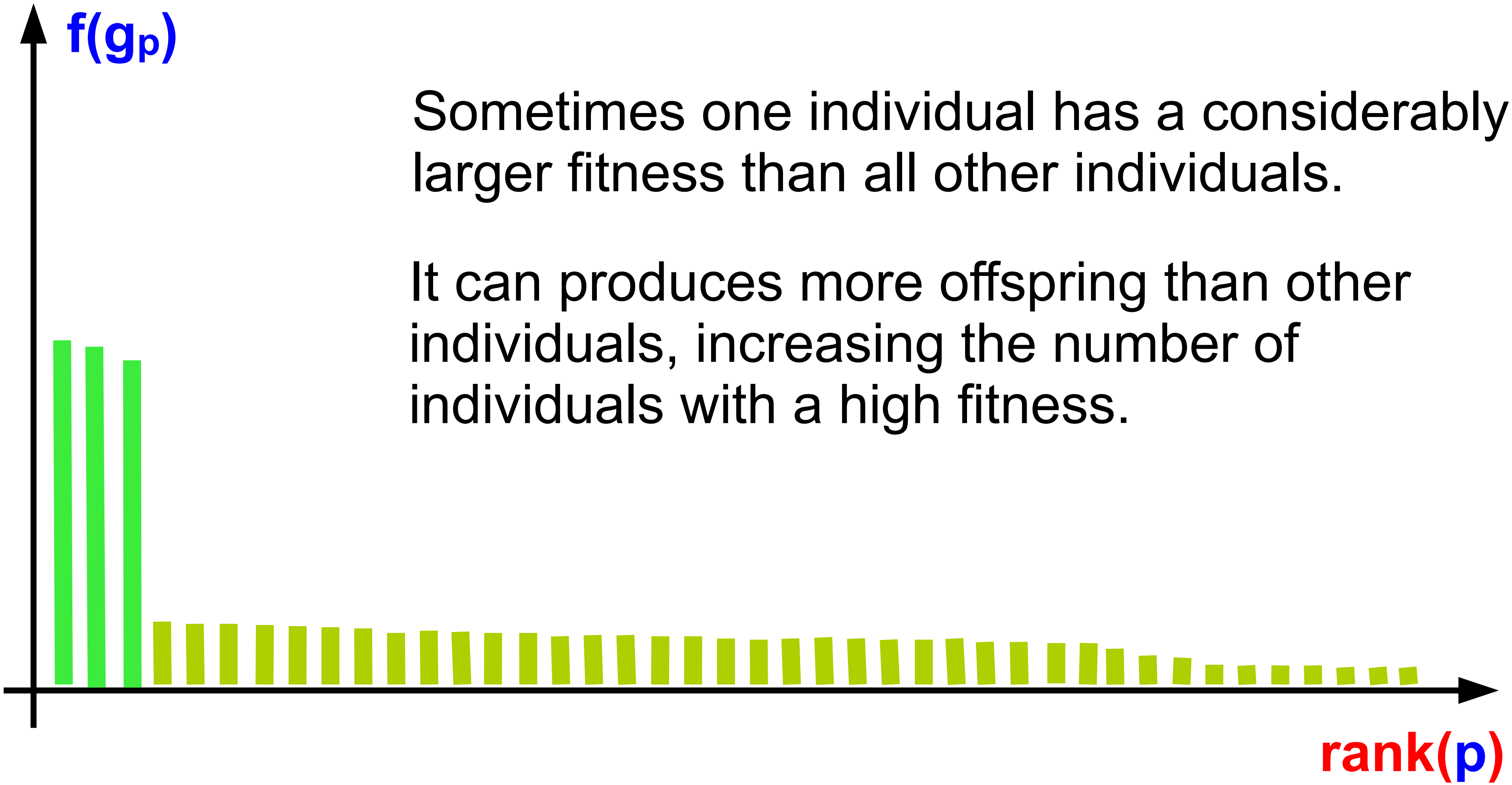


Sometimes one individual has a considerably larger fitness than all other individuals.

EA: Super-Individual

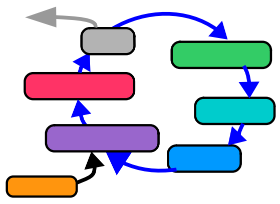


Sorted fitness values.



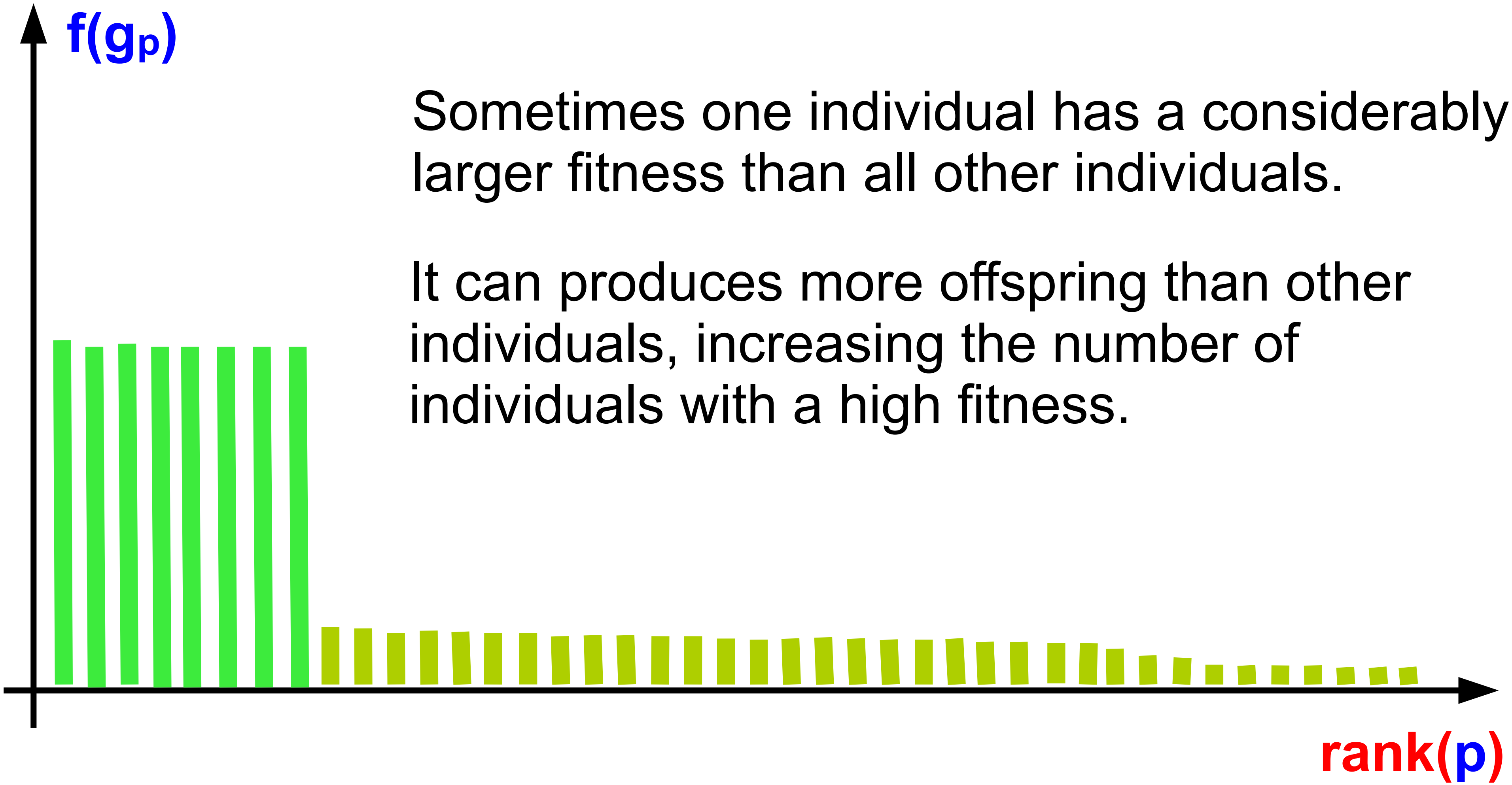
Sometimes one individual has a considerably larger fitness than all other individuals.

It can produces more offspring than other individuals, increasing the number of individuals with a high fitness.



EA: Super-Individual

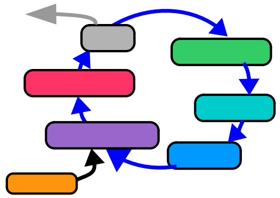
Sorted fitness values.



Sometimes one individual has a considerably larger fitness than all other individuals.

It can produces more offspring than other individuals, increasing the number of individuals with a high fitness.

EA: Super-Individual

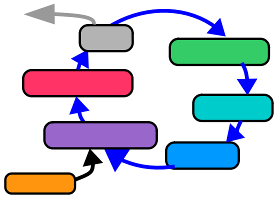


Sometimes, the combination of selection and inheritance is producing a so called: **Super-Individual.**

It can happen, that an individual with a rather high fitness is generating so much offspring, that they **dominate** the entire population.

Which means, that a large percentage of the population has identical (or almost identical) genomes.

EA: Super-Individual



Sometimes, the combination of selection and inheritance is producing a so called: **Super-Individual.**

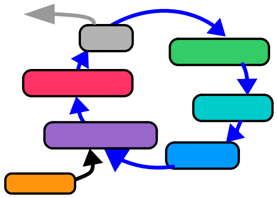
It can happen, that an individual with a rather high fitness is generating so much offspring, that they dominate the entire population.

Which means, that a large percentage of the population has identical (or almost identical) genomes.

On one hand, a **Super-Individual** with a large fitness is good for the overall performance of the population,

on the other hand, if a large percentage of the population has identical genomes the **diversity** throughout the population is lost.

EA: Super-Individual

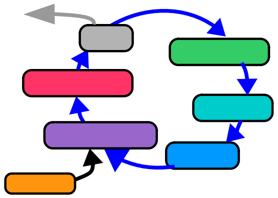


Generally speaking, the occurrence of a super-individual should be **avoided** to maintain the necessary diversity within the population (EAs are multi hypothesis approaches).

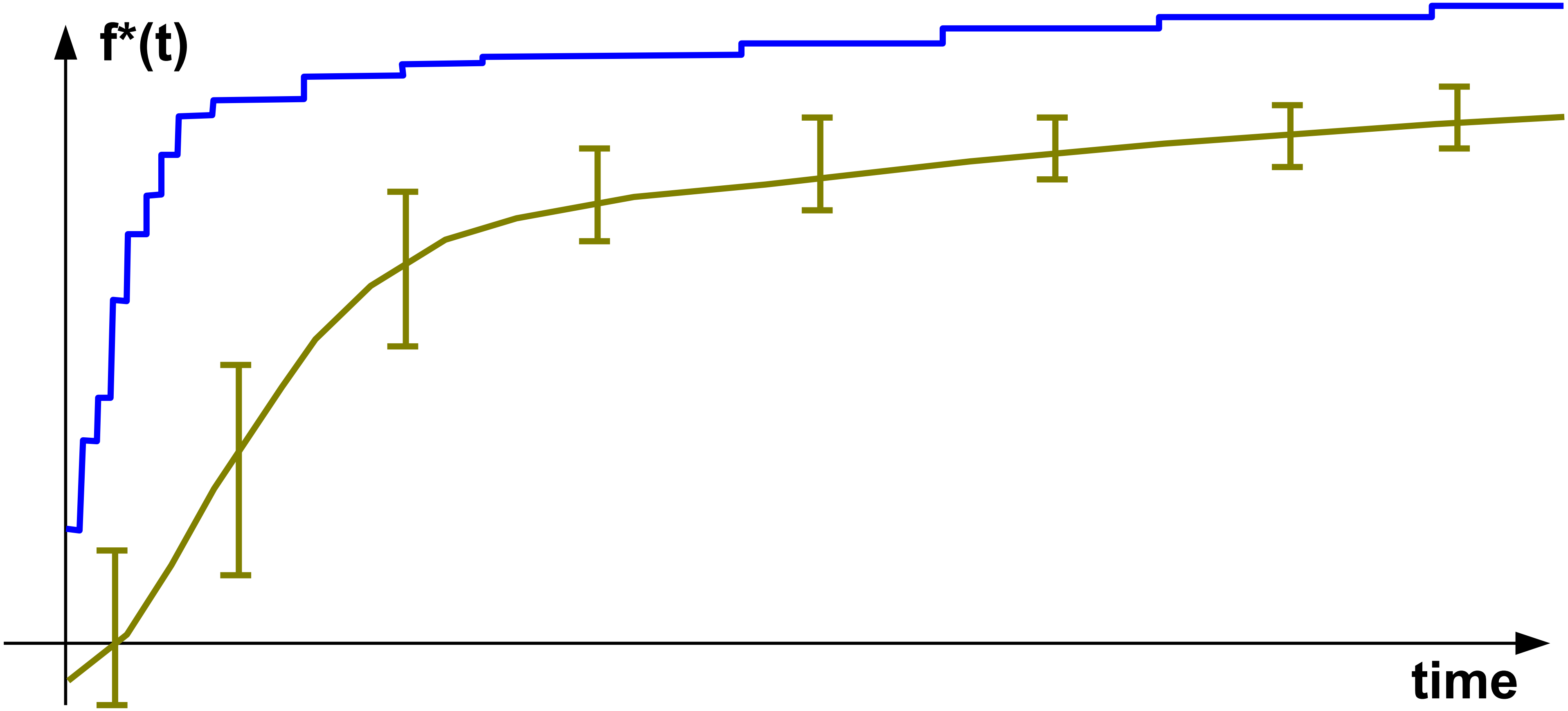
The detection of a super-individual can be really hard.
A hint for the occurrence is a shrinking variance of fitness values in the parent population.

A close investigation of the performance graph is helpful.

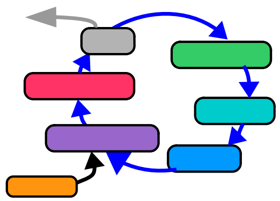
EA: Super-Individual



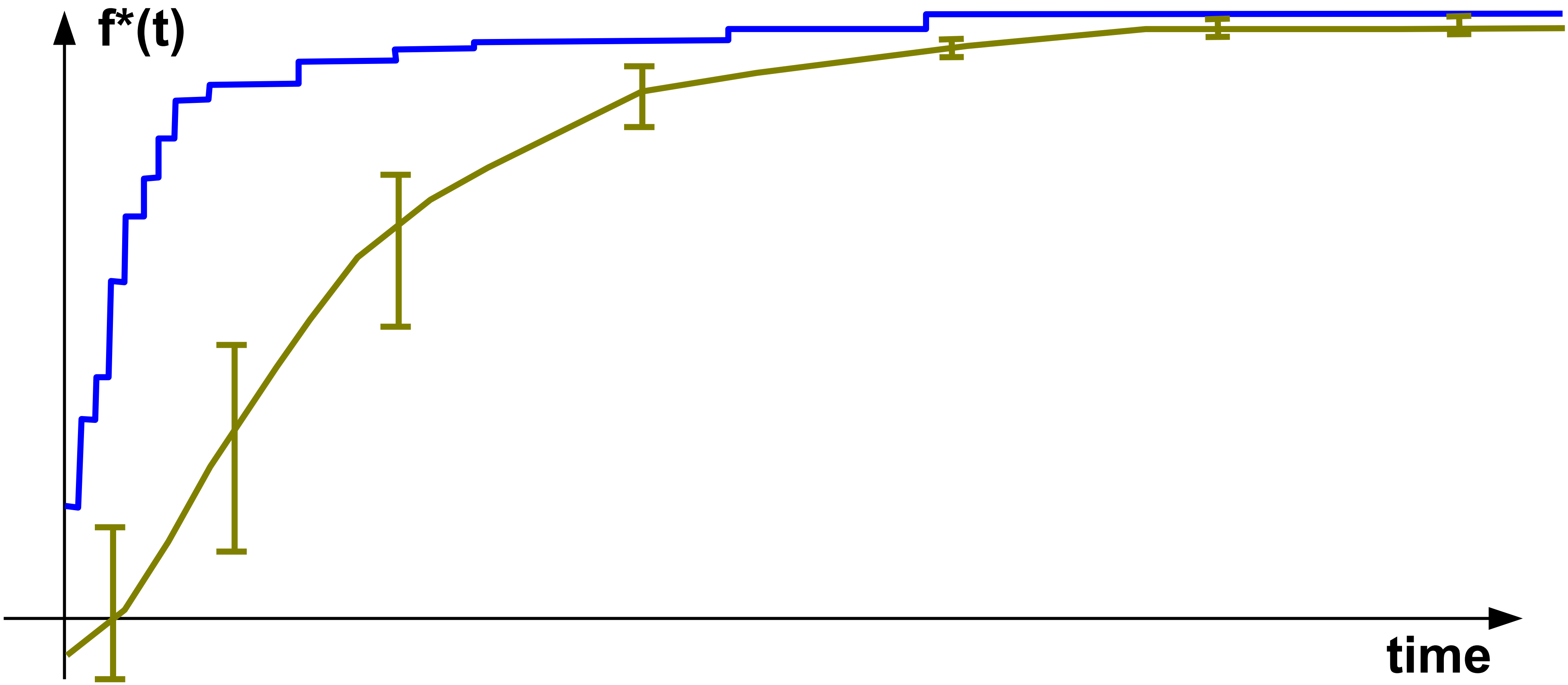
The performance graph is showing the development of the fitness f*(t) of the best individual and the average fitness in each generation with respect to time.



EA: Super-Individual



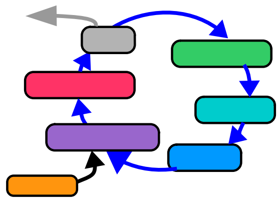
The performance graph might me an **indicator** for the occurrence of a super-individual dominating the population.

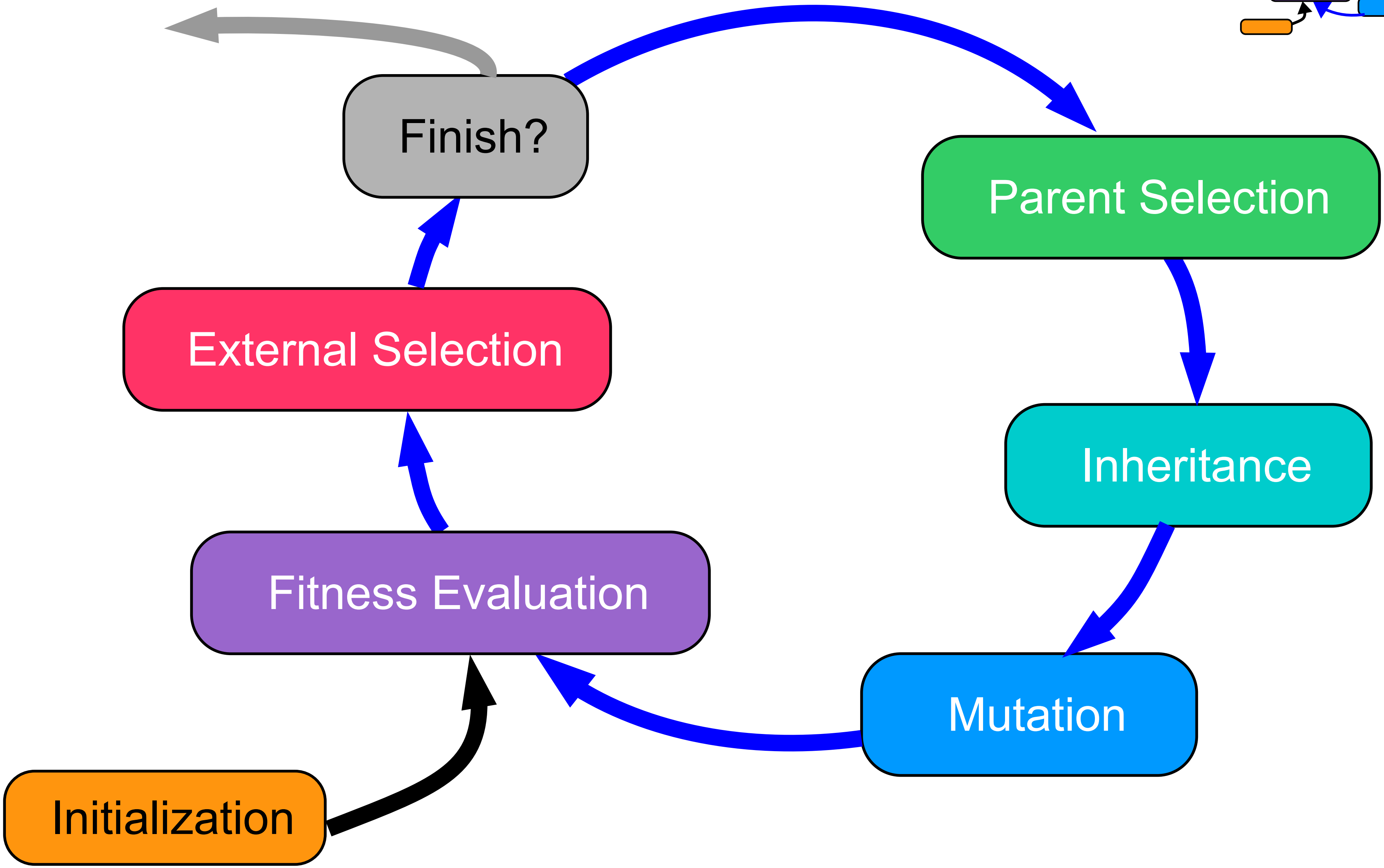


Overview

- Genome structure
- Example: 8 queens
- Super-individuals
- **External-selection and parent-selection combined**
- Probabilistic parent-selection
  - Wheel of fortune
  - Softmax selection
  - Tournament selection
- Genetic programming
- Co-evolution

EA cycle of operations:

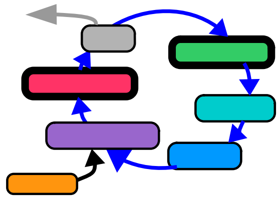




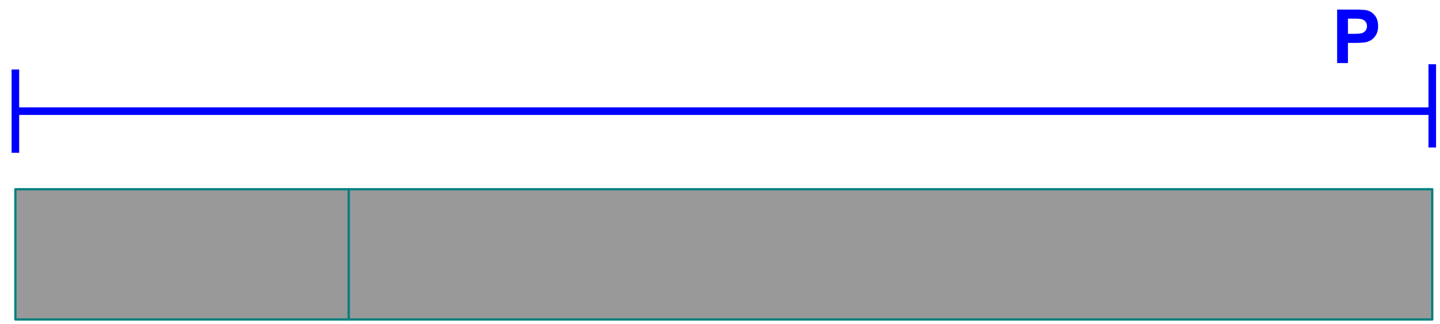
EA:

External Selection

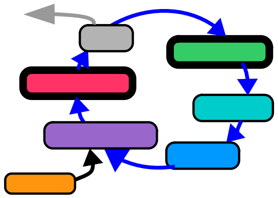
Parent Selection



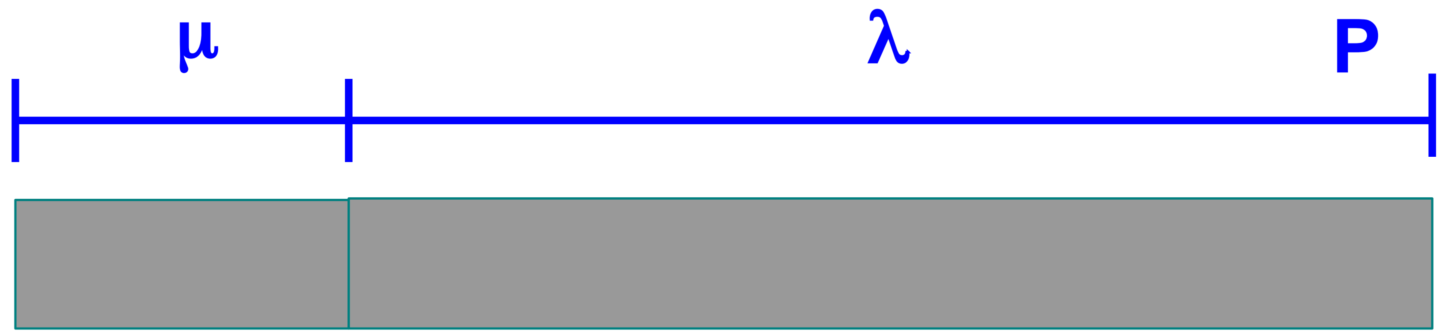
Have **P**, keep **μ**, generate **λ** offspring



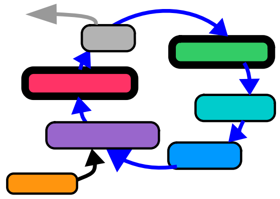
EA: External Selection Parent Selection



Have **P**, keep **μ**, generate **λ** offspring

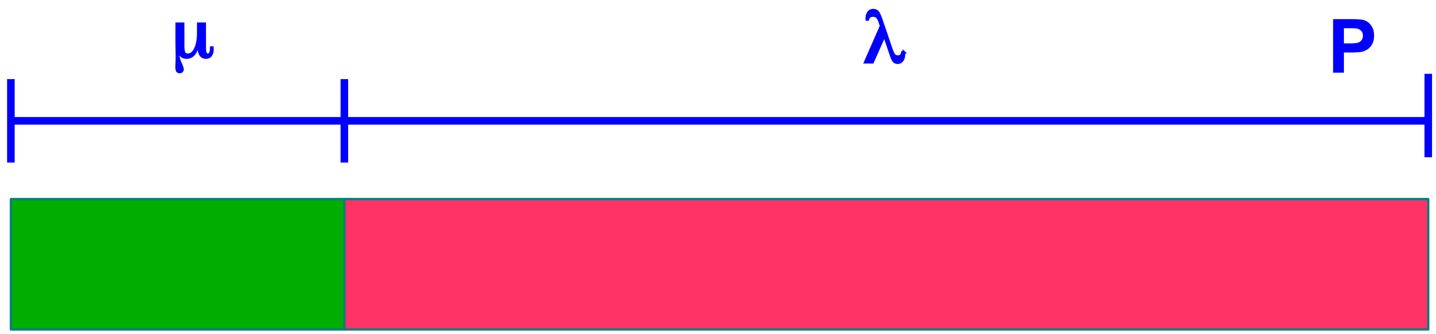


EA: External Selection Parent Selection

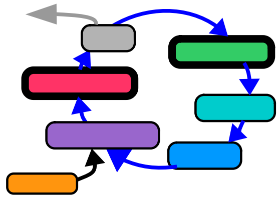


Have **P**, keep **μ**, generate **λ** offspring

external selection

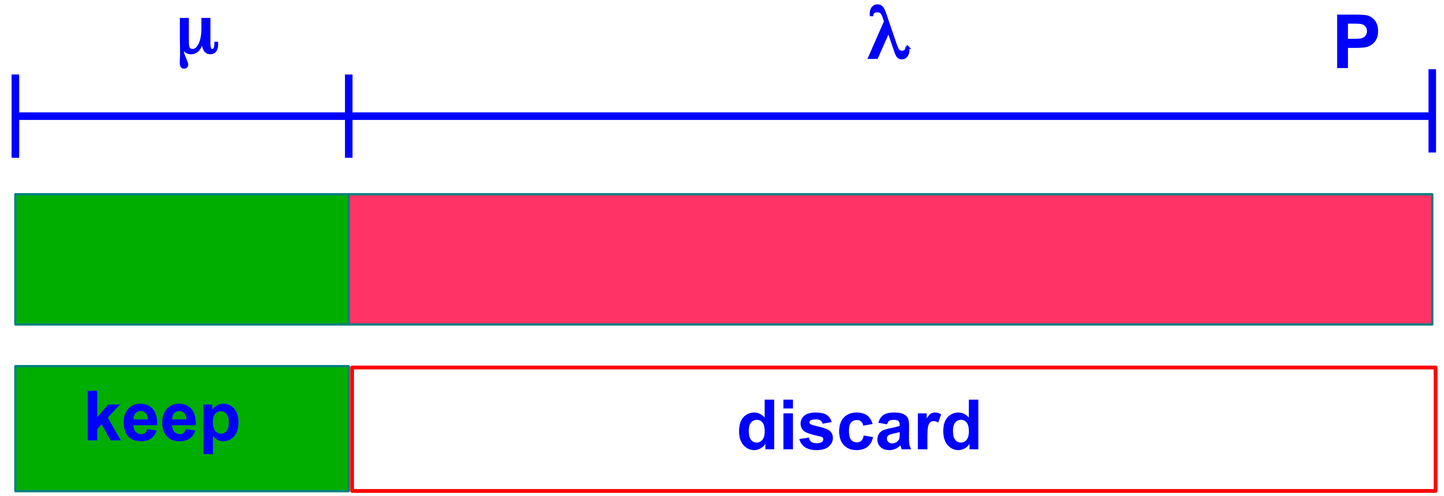


EA: External Selection Parent Selection



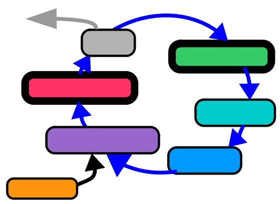
Have **P**, keep **μ**, generate **λ** offspring

external selection

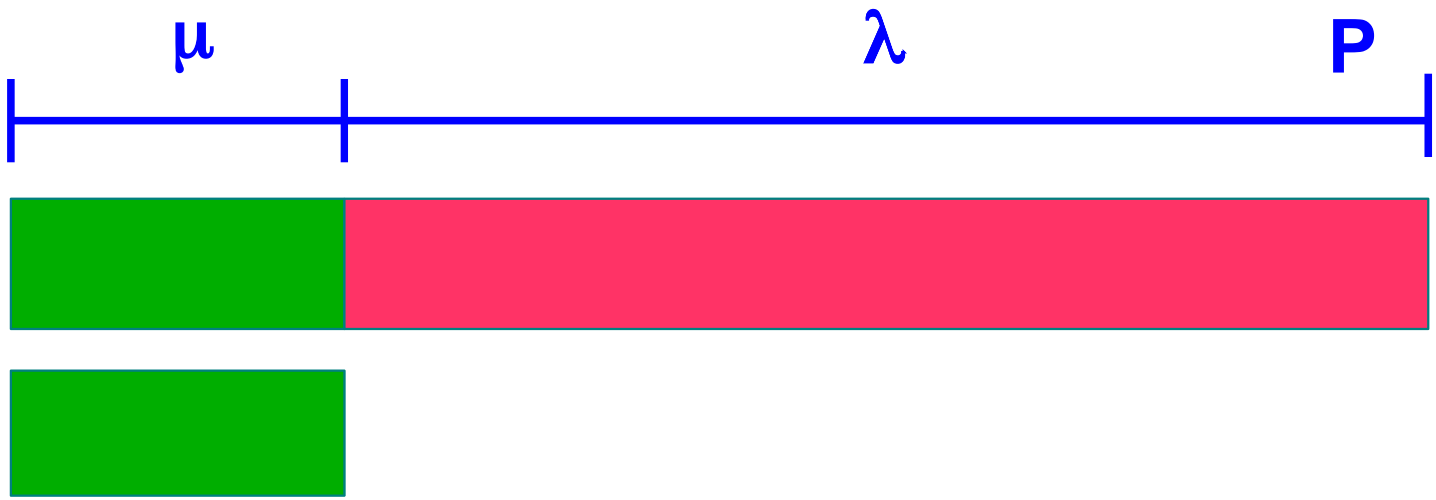


EA:

External Selection **Parent Selection**



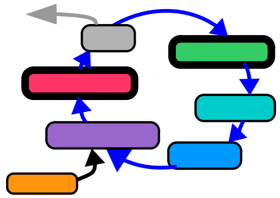
Have **P**, keep **μ**, generate **λ** offspring



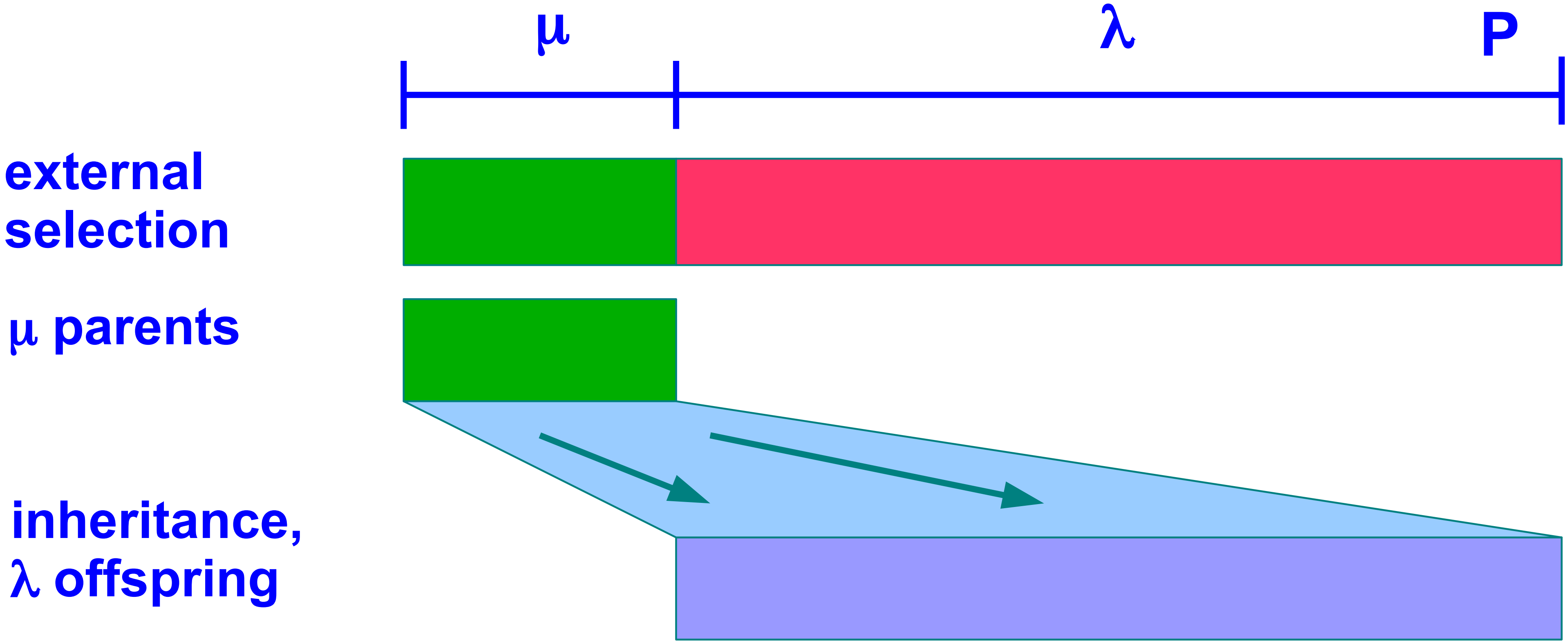
external selection

μ parents

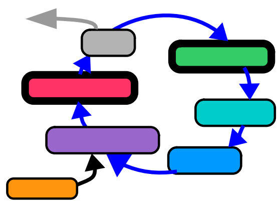
EA: External Selection / Parent Selection



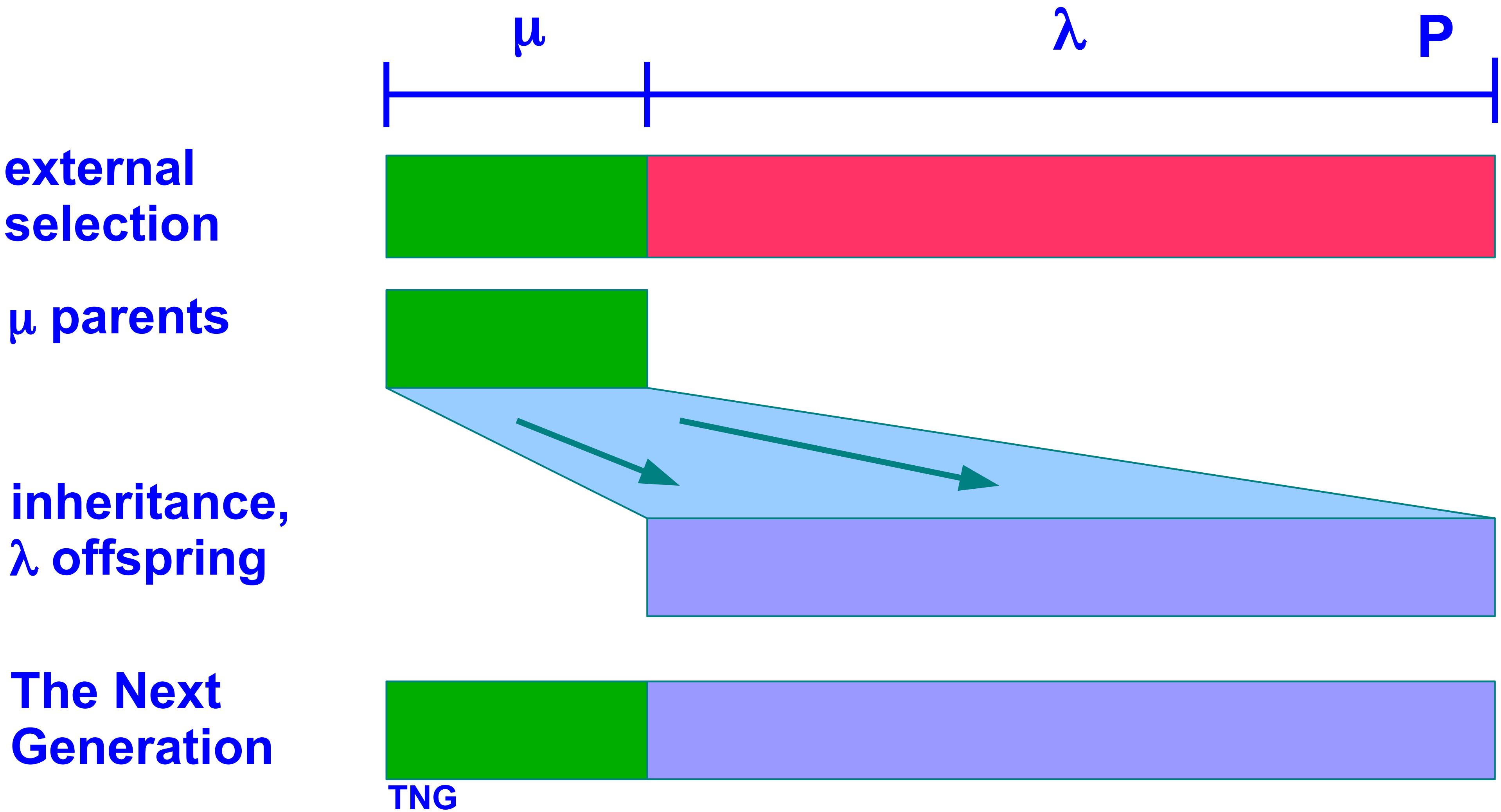
Have **P**, keep **μ**, generate **λ** offspring



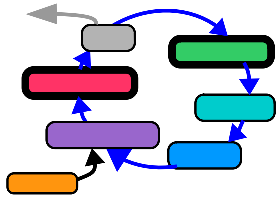
EA: External Selection | Parent Selection



Have **P**, keep **μ**, generate **λ** offspring



EA: External Selection Parent Selection



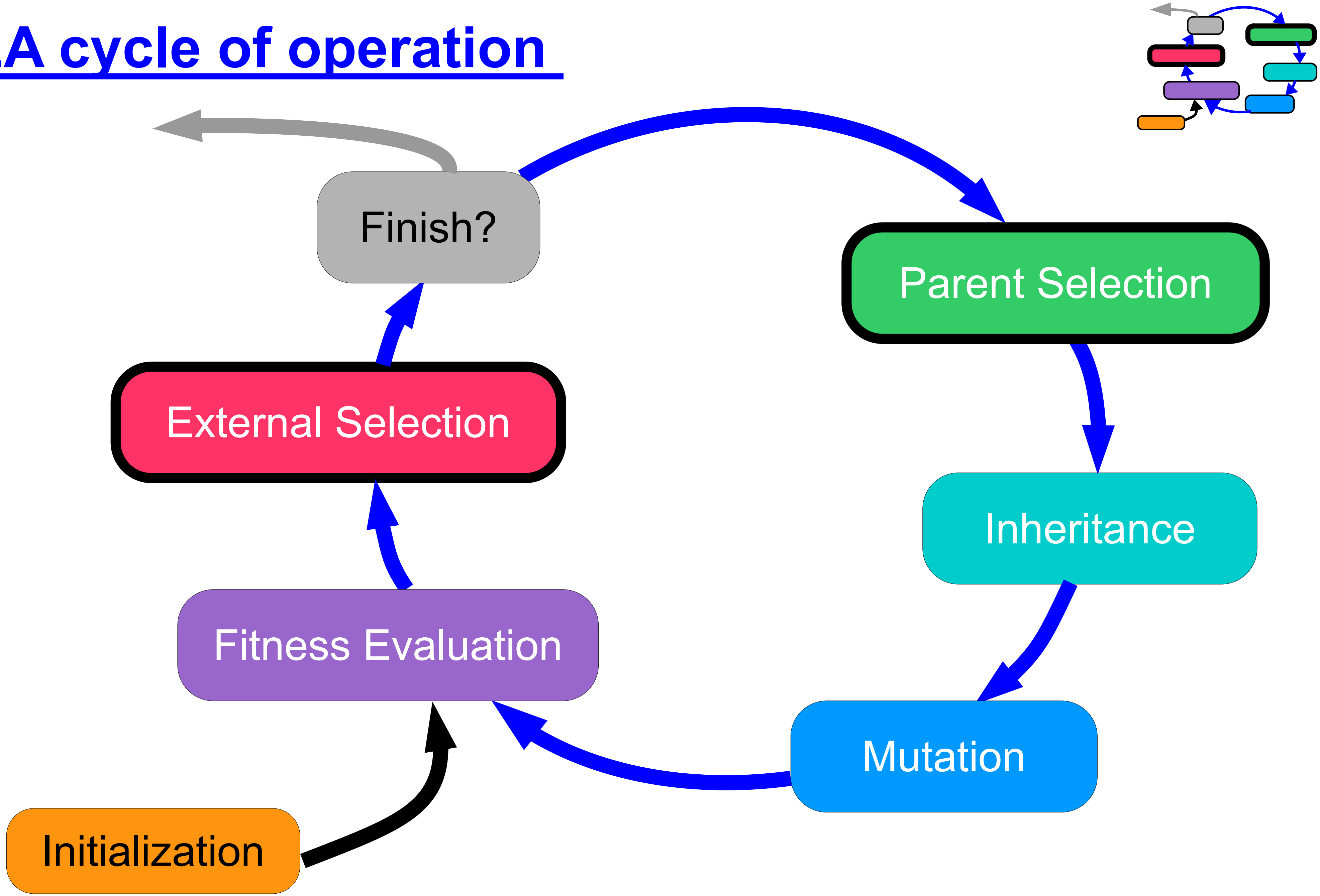
(1 + 1): $\mu=1$ one parent, $\lambda=1$ one child,
inheritance by copying, only mutation
rank based, deterministic external selection

(1 + λ): $\mu=1$ one parent, λ children, offspring,
inheritance by copying, only mutation
rank based, deterministic external selection

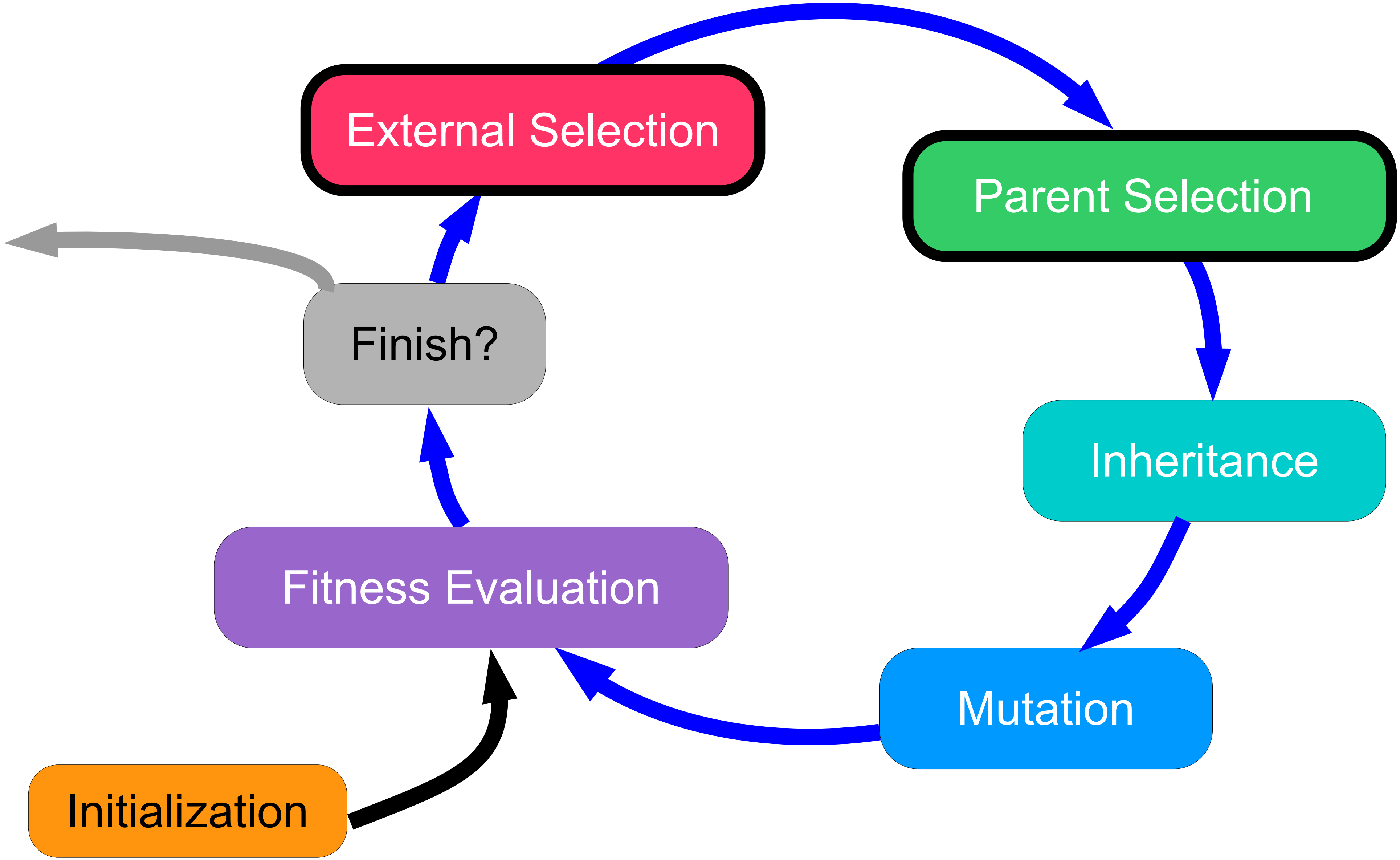
(μ + λ): μ parents, λ offspring, parents survive
recombination, mutation, external selection.

(μ , λ): μ parents, λ offspring,
recombination, mutation, external selection,
parents are discarded.

EA cycle of operation



EA cycle of operation



EA:

Parent Selection **External Selection**

An alternative scheme to operate the evolutionary algorithm is to **separate** parent selection from external selection.

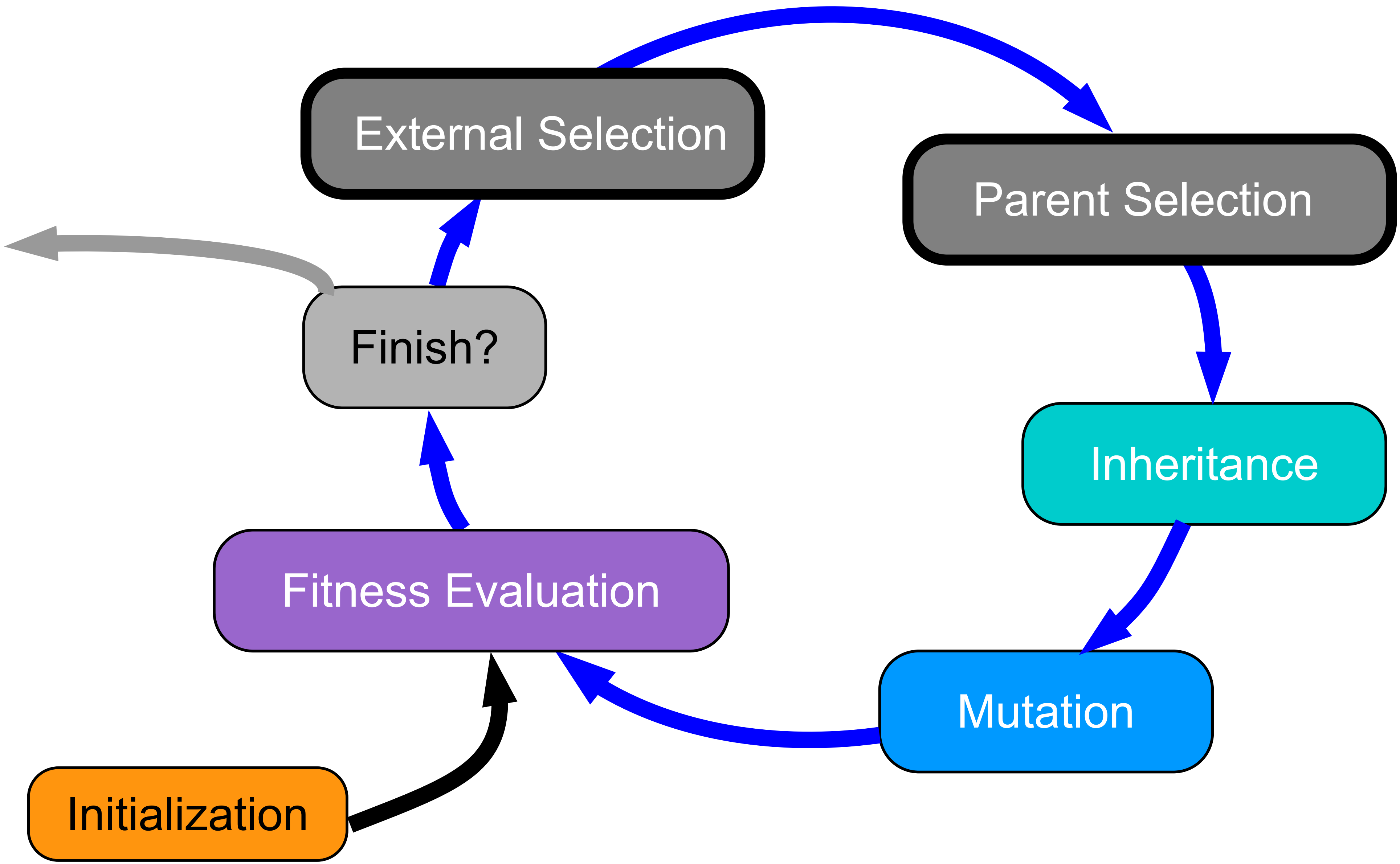
Then, within **external selection**, the μ individuals to survive, are selected from the complete population of P individuals, and λ individuals are discarded .

And, within **parent selection** the parents are selected from the complete population as well, building the pool of parents for subsequent inheritance.

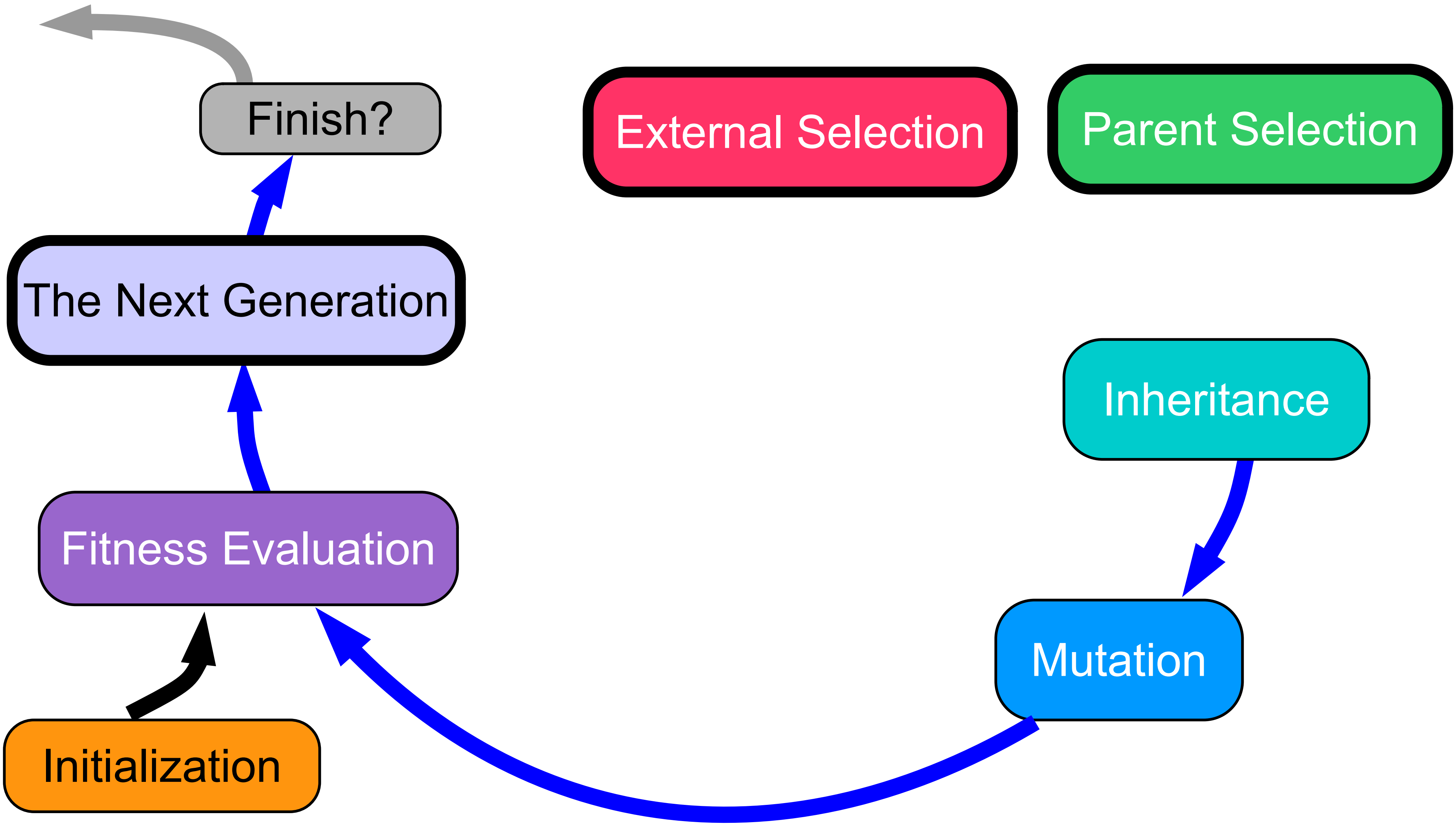
The **inheritance step** generates λ new individuals, from the pool of parents; the pool of parents is erased afterwards.

The Next Generation is build by combining the μ survivors with the λ offspring ($\mu + \lambda = P$).

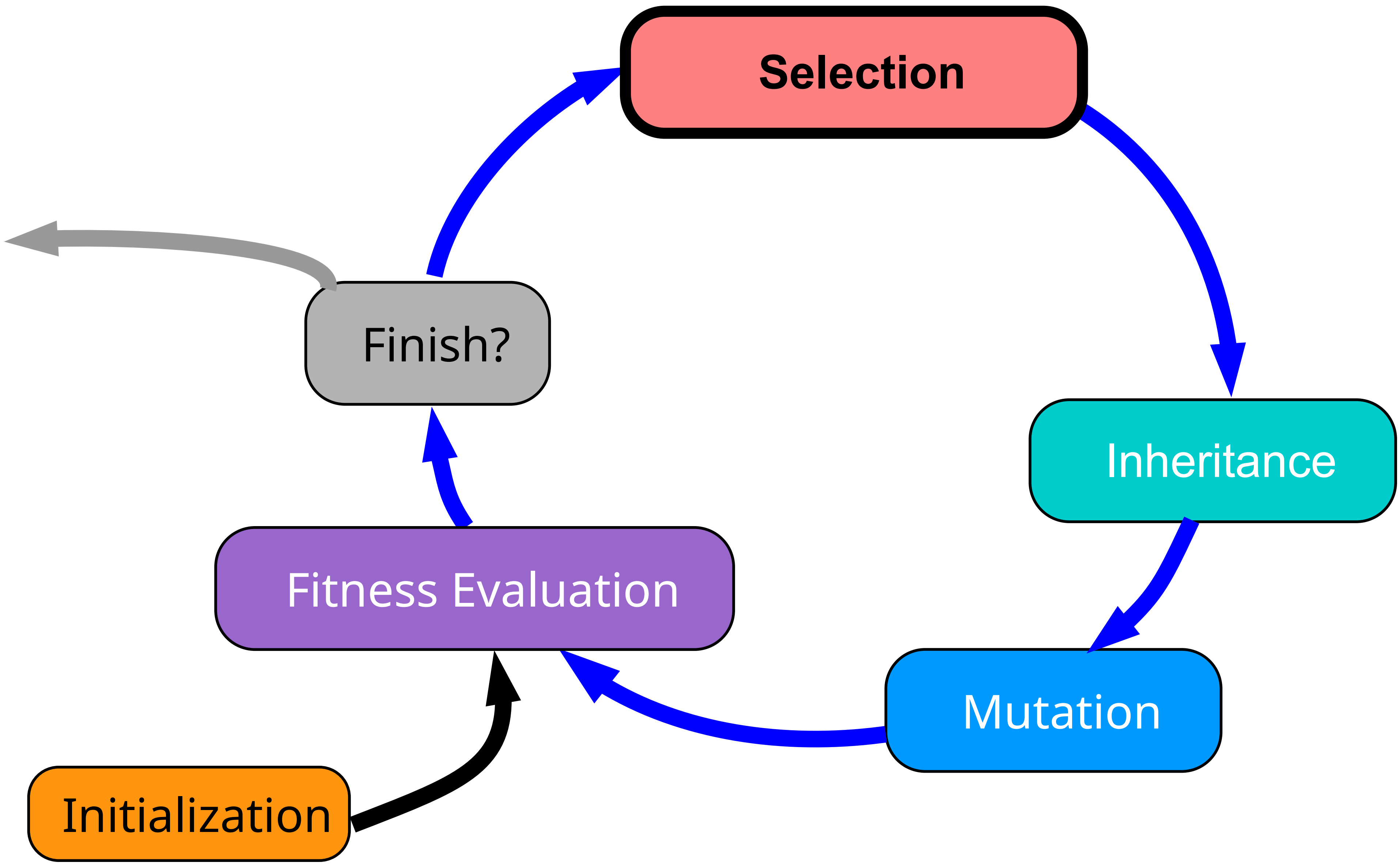
EA cycle of operation



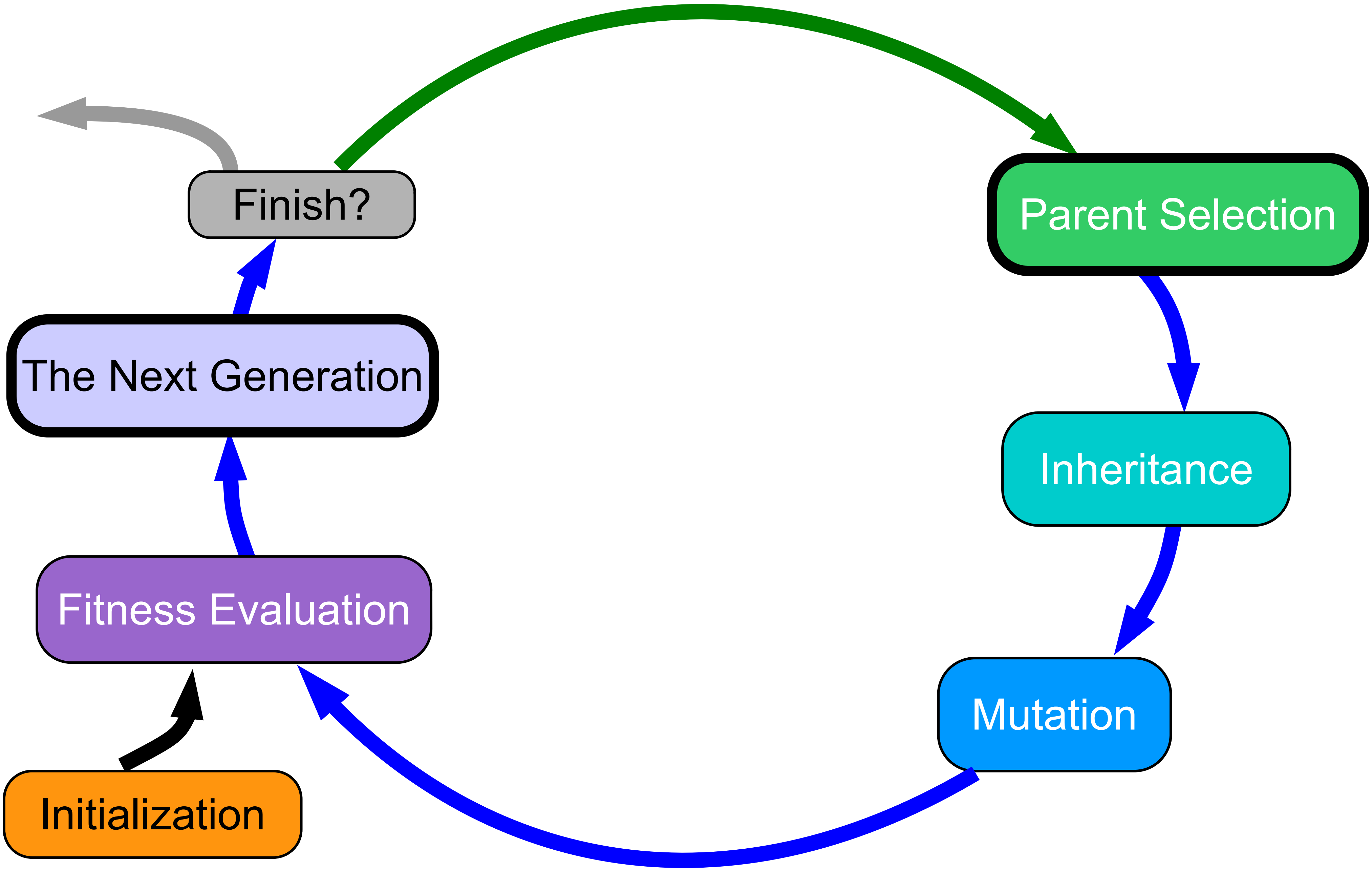
EA alternative cycle of operation



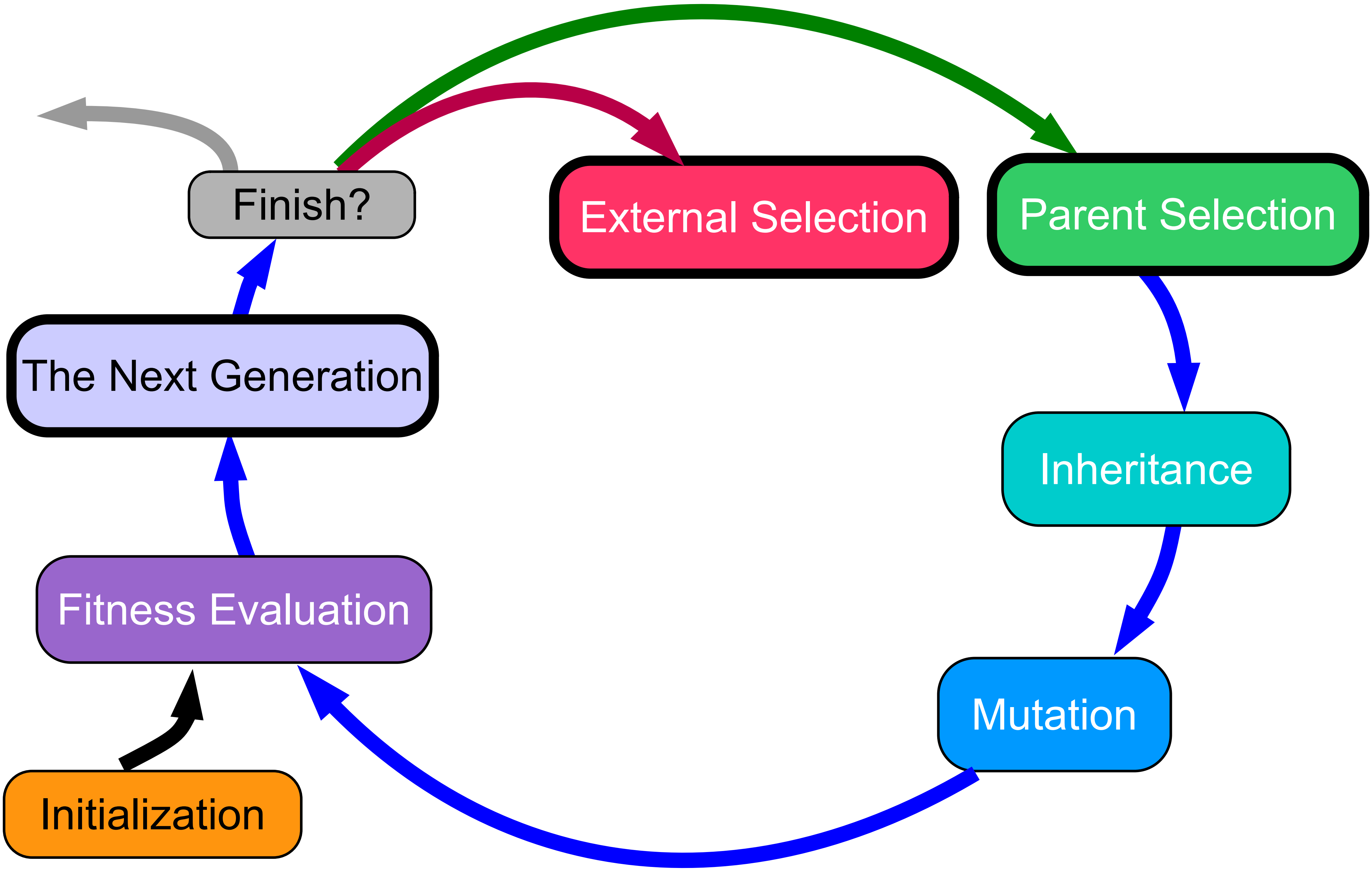
EA cycle of operation



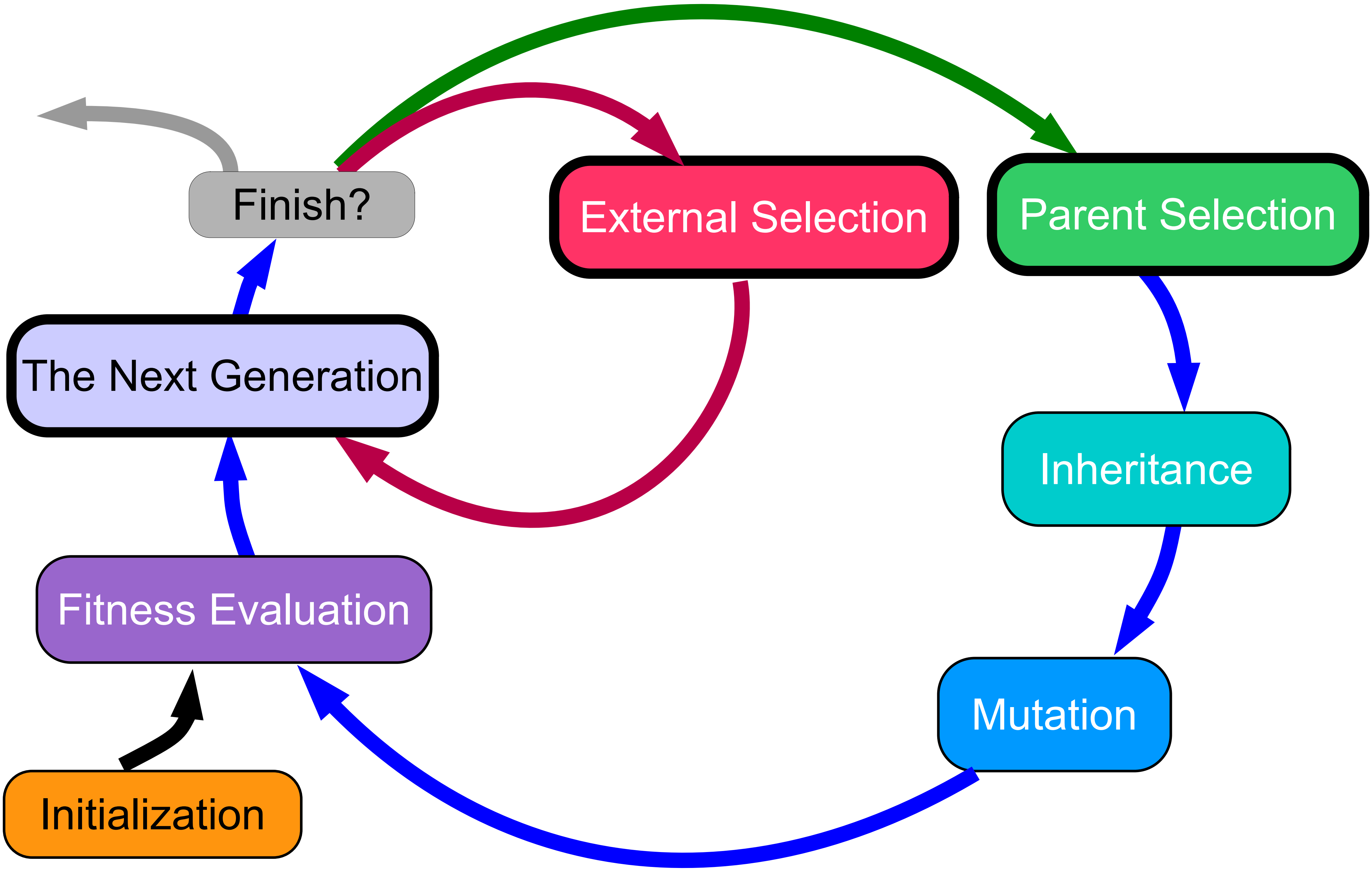
EA alternative cycle of operation



EA alternative cycle of operation



EA alternative cycle of operation



EA:

Parent Selection

External Selection

Have **P**, keep μ, generate λ offspring

P

EA:

Parent Selection **External Selection**

Have **P**, keep **μ**, generate **λ** offspring

P

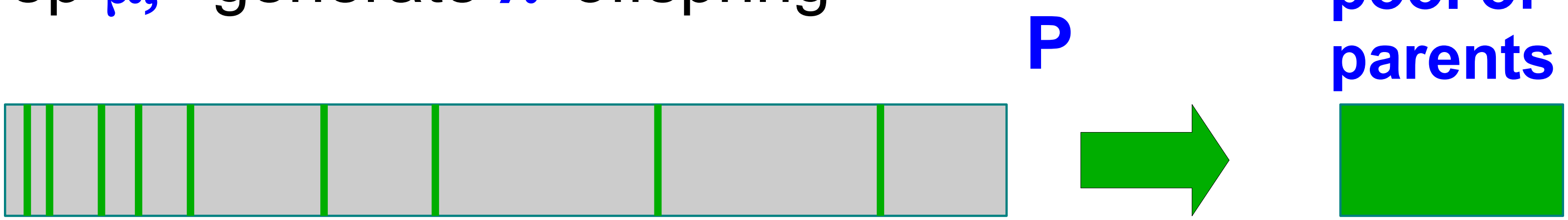
parent selection

EA:

Parent Selection **External Selection**

Have **P**, keep **μ**, generate **λ** offspring

parent selection



EA:

Parent Selection **External Selection**

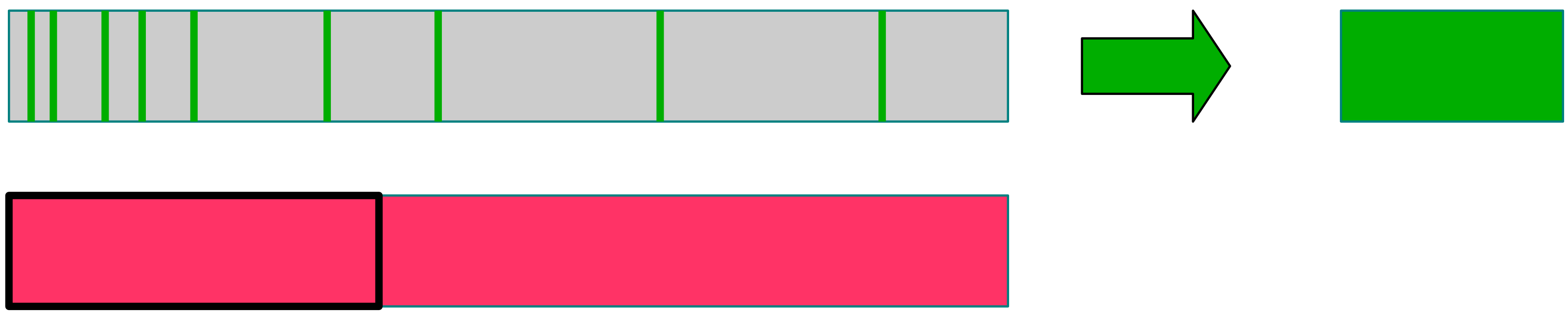
Have **P**, keep **μ**, generate **λ** offspring

P

pool of parents

parent selection

external selection



EA:

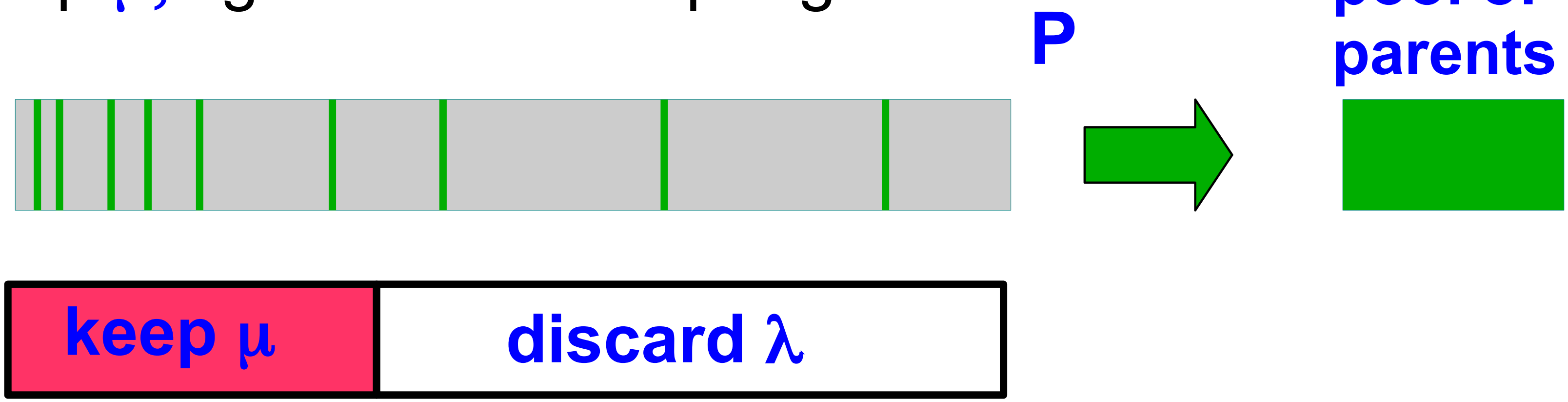
Parent Selection **External Selection**

Have **P,** keep **μ,** generate **λ** offspring

pool of parents

parent selection

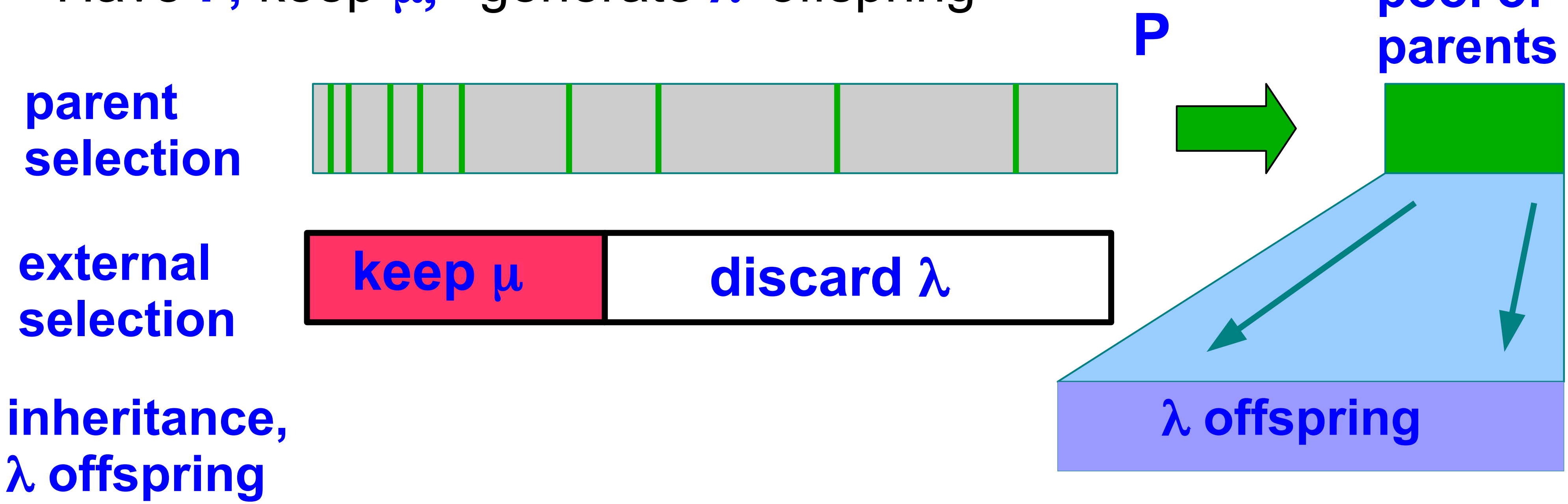
external selection



EA:

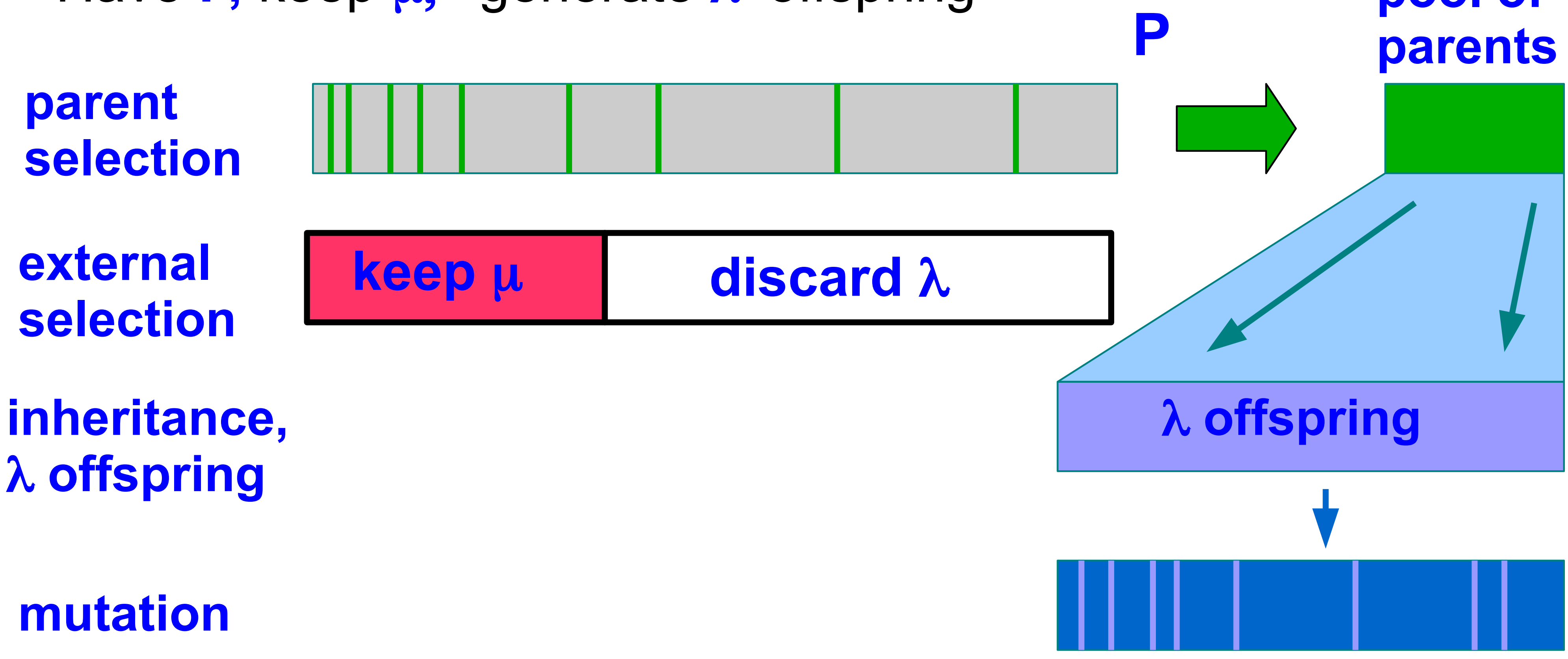
Parent Selection **External Selection**

Have **P**, keep **μ**, generate **λ** offspring



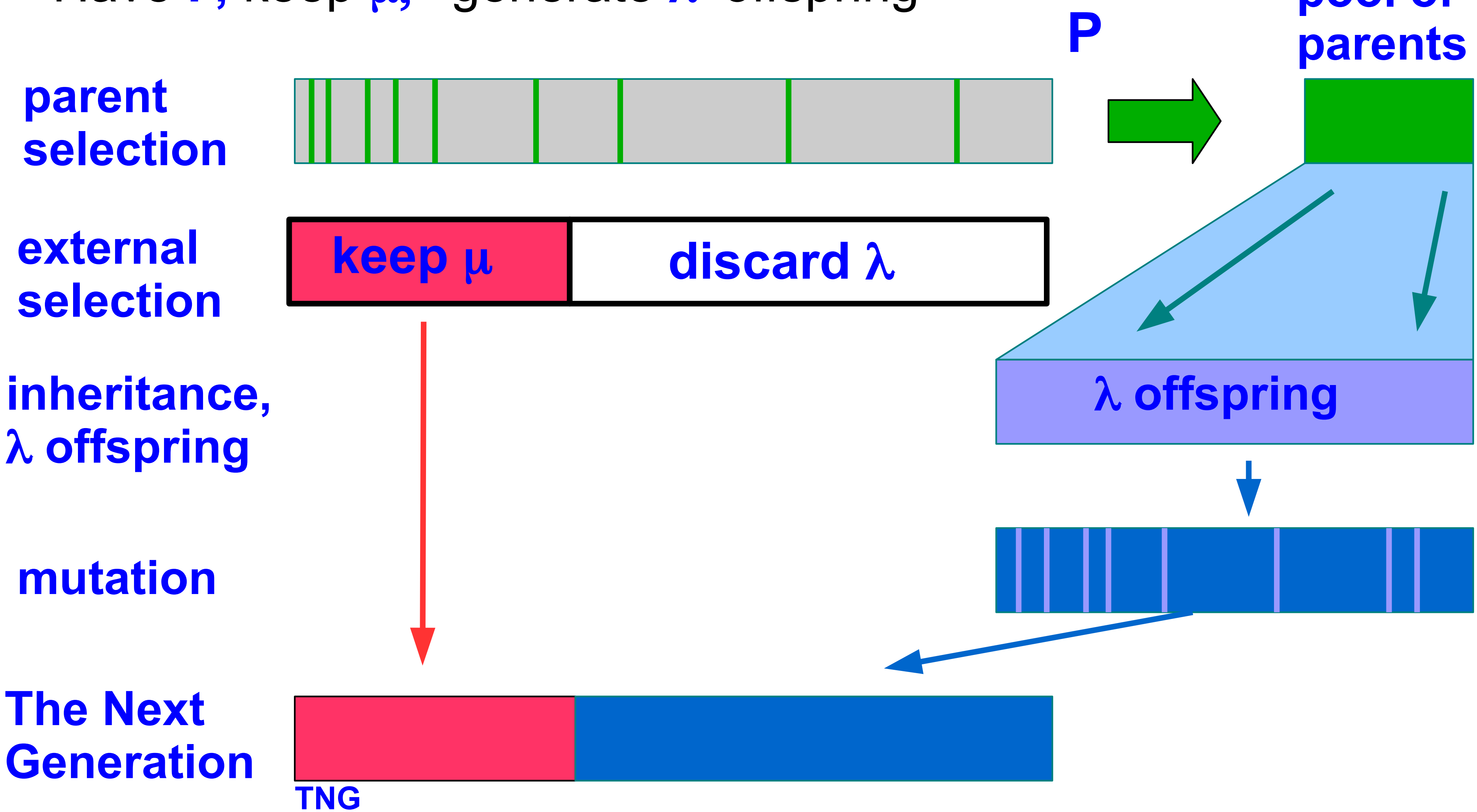
EA: Parent Selection External Selection

Have **P**, keep **μ**, generate **λ** offspring

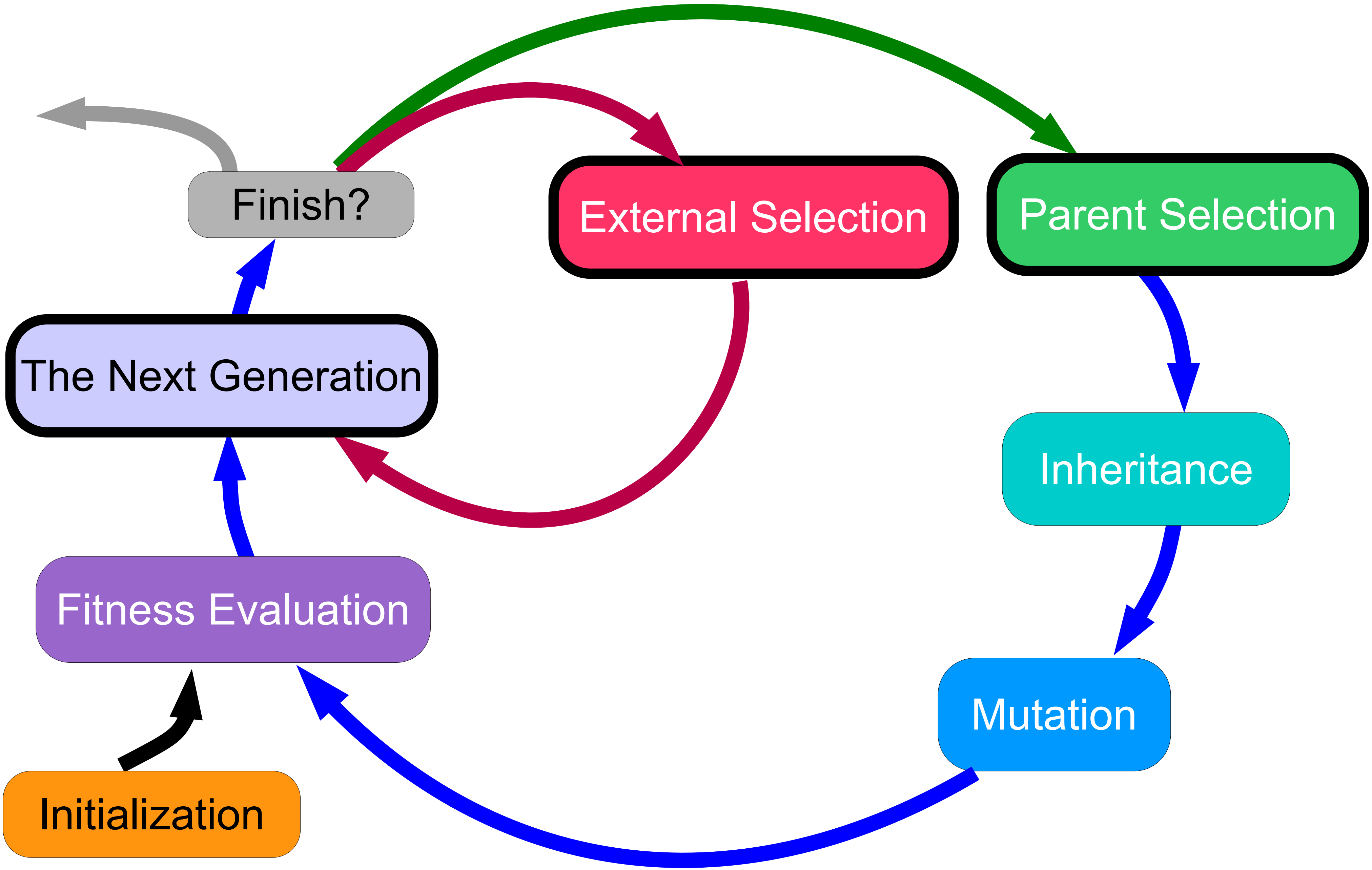


EA: Parent Selection / External Selection

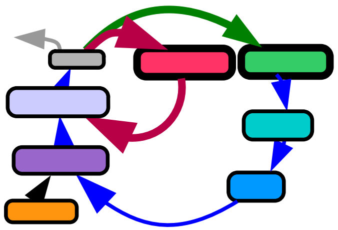
Have **P**, keep **μ**, generate **λ** offspring



EA alternative cycle of operation



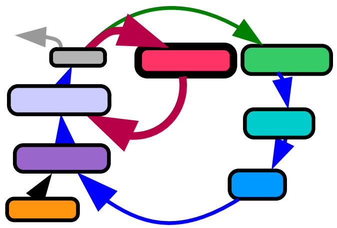
EA: Selection



Common strategies for the **selection** are:

- random choice
- schedule based, e.g. round robin
- fitness based elitism:
  - fitness proportional choice
  - rank proportional choice
- fitness based stochastic:
  - fitness proportionate, probabilistic choice
  - rank proportionate, probabilistic choice
- combinations of the above

EA: External Selection



Elitism is a way to design the selection, and inheritance steps to shape the next generation following the principle of **Exploitation**.

Using elitism a subset of the population is surviving (excluding mutation), and will be part of the next generation.

Typically the μ best individuals are chosen to survive.

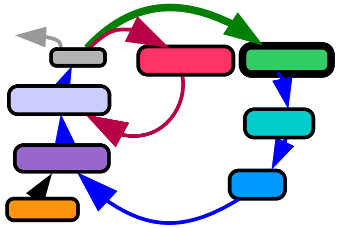
The **(** μ **+** λ **)** deterministic, rank dependent strategy, with taking the μ best individuals as parents is a common implementation of elitism.

But the pool of parents, and the elite of μ surviving individuals can be different.

Overview

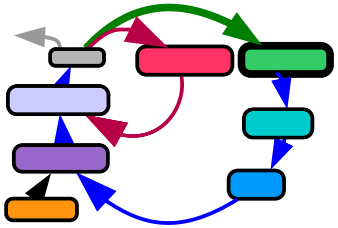
- Genome structure
- Example: 8 queens
- Super-individuals
- External-selection and parent-selection combined
- **Probabilistic parent-selection**
  - Wheel of fortune
  - Tournament selection
- Genetic programming
- Co-evolution

EA: Parent Selection



Within parent selection one has to try to obey the two principles of **Exploration** and **Exploitation** at the same time.

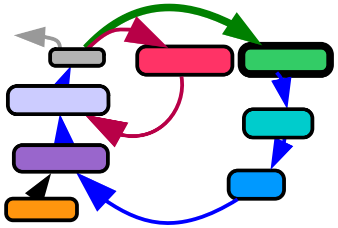
EA: Parent Selection



Within parent selection one has to try to obey the two principles of **Exploration** and **Exploitation** at the same time.

Of course, one is trying to use the acquired knowledge as good as possible, while maintaining the diversity within the population.

EA: Parent Selection

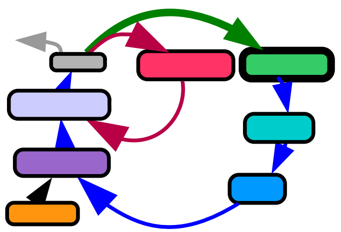


Within parent selection one has to try to obey the two principles of **Exploration** and **Exploitation** at the same time.

Of course, one is trying to use the acquired knowledge as good as possible, while maintaining the diversity within the population.

Therefore it is a good idea to build up the pool of parents with a lot of the best individuals and explicitly take some individuals with a lower performance.

EA: Parent Selection



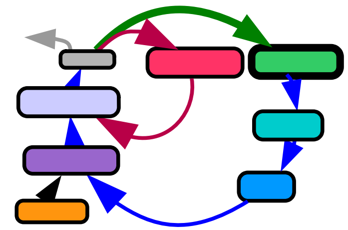
Within parent selection one has to try to obey the two principles of **Exploration** and **Exploitation** at the same time.

Of course, one is trying to use the acquired knowledge as good as possible, while maintaining the diversity within the population.

Therefore it is a good idea to build up the pool of parents with a lot of the best individuals and explicitly take some individuals with a lower performance.

A common way to implement this is to make the probability of being chosen for the pool of parents depending on the achieved fitness **f(g)**:

EA: Parent Selection



Within parent selection one has to try to obey the two principles of **Exploration** and **Exploitation** at the same time.

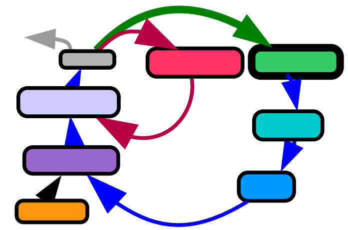
Of course, one is trying to use the acquired knowledge as good as possible, while maintaining the diversity within the population.

Therefore it is a good idea to build up the pool of parents with a lot of the best individuals and explicitly take some individuals with a lower performance.

A common way to implement this is to make the probability of being chosen for the pool of parents depending on the achieved fitness **f(g)**:

fitness proportional or **rank proportional**

EA: Parent Selection



Within parent selection one has to try to obey the two principles of **Exploration** and **Exploitation** at the same time.

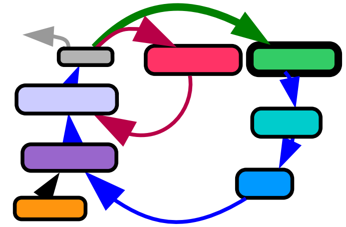
Of course, one is trying to use the acquired knowledge as good as possible, while maintaining the diversity within the population.

Therefore it is a good idea to build up the pool of parents with a lot of the best individuals and explicitly take some individuals with a lower performance.

A common way to implement this is to make the probability of being chosen for the pool of parents depending on the achieved fitness **f(g)**:

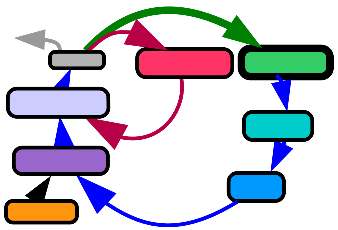
~~fitness proportional~~ or **rank proportional**

EA: Parent Selection



It showed to be feasible to make up the set of parents with those individuals, that have shown a good fitness **f(g)**, and explicitly include additional individuals that have not performed as good.

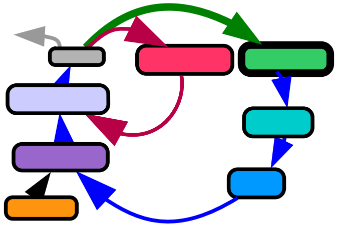
EA: Parent Selection



It showed to be feasible to make up the set of parents with those individuals, that have shown a good fitness **f(g)**, and explicitly include additional individuals that have not performed as good.

A **probabilistic parent selection**, depending on the reached fitness values **f(g)** is a common way to implement this.

EA: Parent Selection



It showed to be feasible to make up the set of parents with those individuals, that have shown a good fitness **f(g)**, and explicitly include additional individuals that have not performed as good.

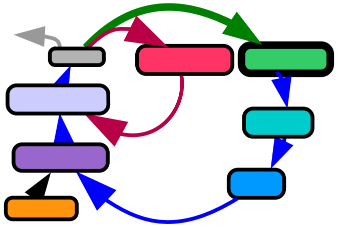
A **probabilistic parent selection**, depending on the reached fitness values **f(g)** is a common way to implement this.

Common probabilistic, fitness based parent selection:
- **Wheel of Fortune, (Roulette-Wheel Selection)**
- Boltzmann Selection, Softmax-Selection
- Tournament selection

Overview

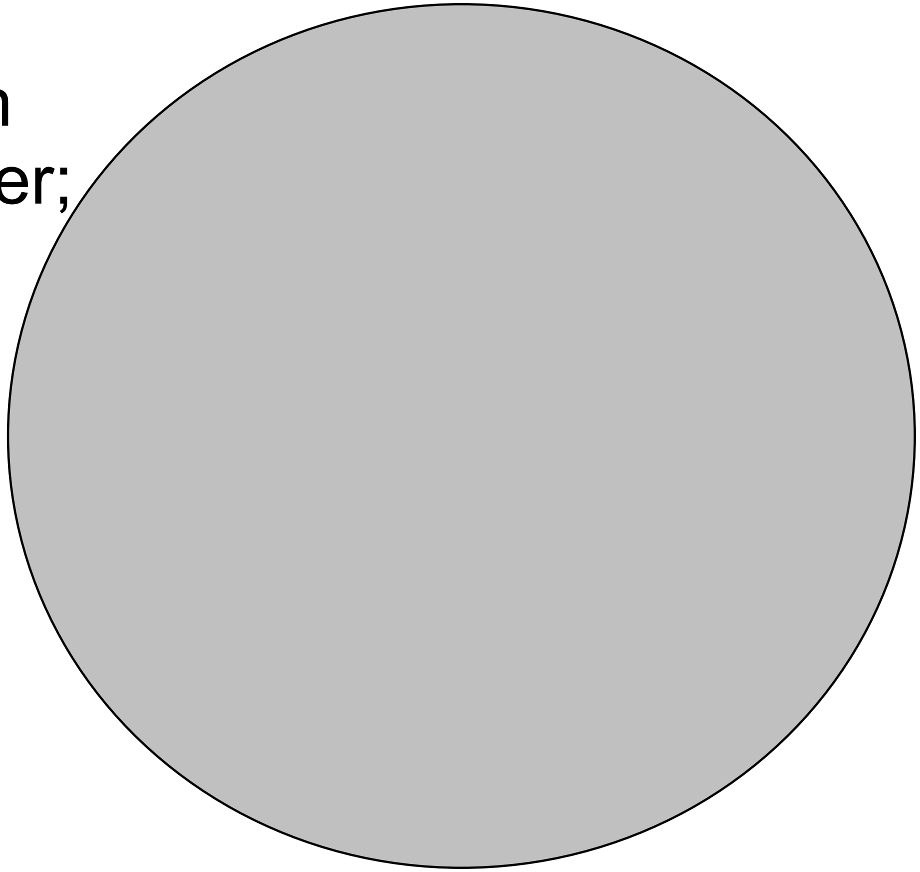
- Genome structure
- Example: 8 queens
- Super-individuals
- External-selection and parent-selection combined
- **Probabilistic parent-selection**
  - **Wheel of fortune**
  - Softmax selection
  - Tournament selection
- Genetic programming
- Co-evolution

EA: Parent Selection

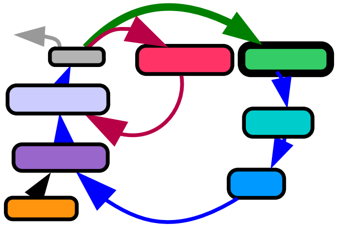


A widely used way to implement a probabilistic, rank proportionate parent selection is the **wheel of fortune**,

Where the probability **ω** to be chosen
is larger, when the fitness **f(g)** is better;
is larger, when the rank **r(f(g))**
of the individual is smaller.

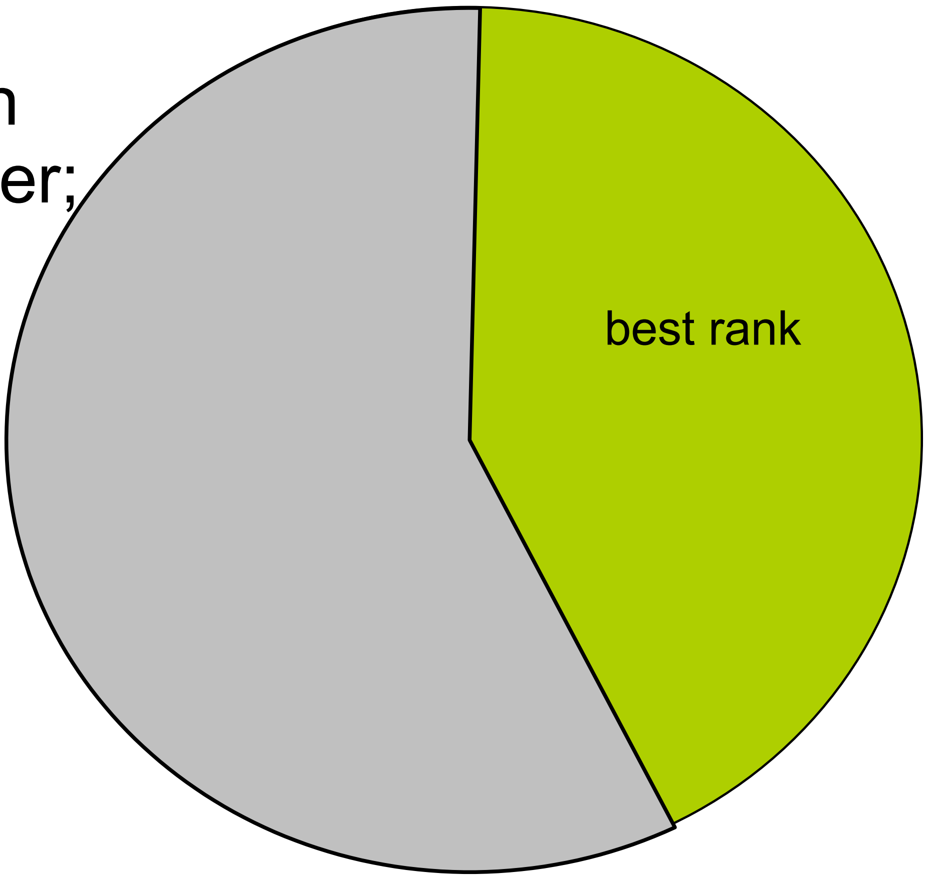


EA: Parent Selection

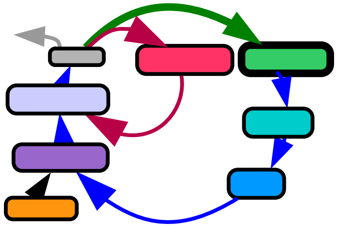


A widely used way to implement a probabilistic, rank proportionate parent selection is the **wheel of fortune**,

Where the probability ω to be chosen is larger, when the fitness **f(g)** is better; is larger, when the rank **r(f(g))** of the individual is smaller.

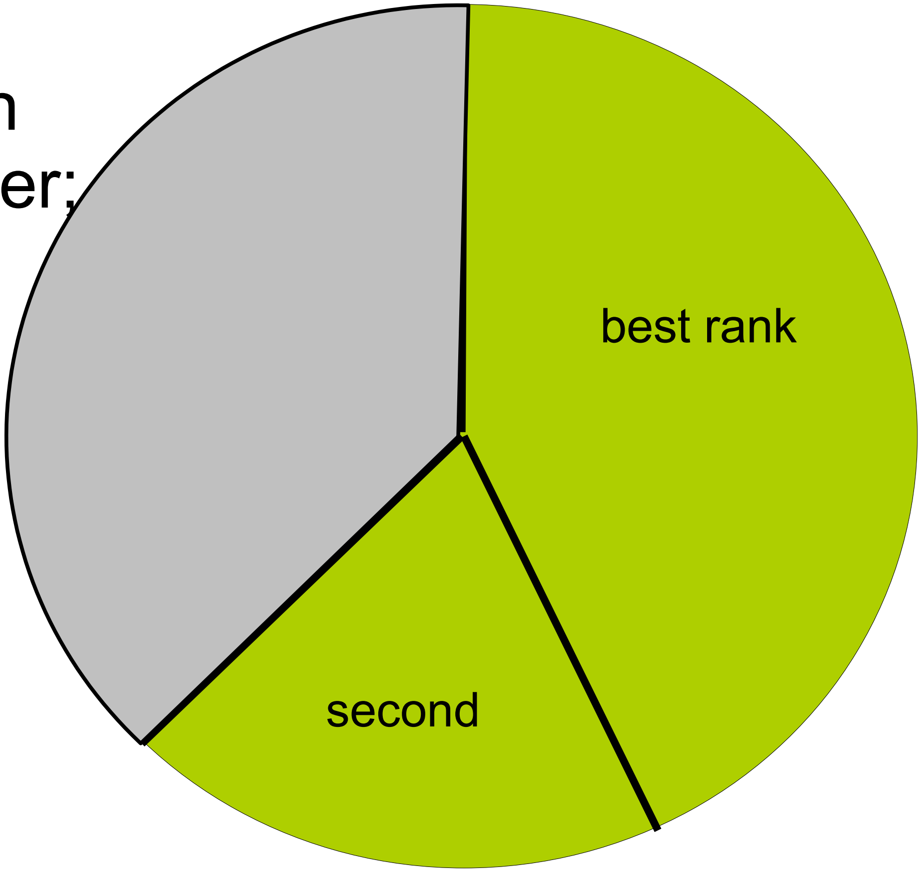


EA: Parent Selection

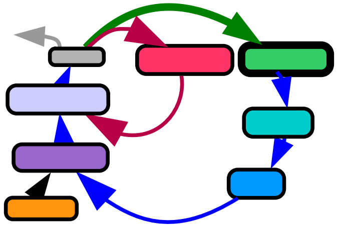


A widely used way to implement a probabilistic, rank proportionate parent selection is the **wheel of fortune**,

Where the probability ω to be chosen is larger, when the fitness **f(g)** is better; is larger, when the rank **r(f(g))** of the individual is smaller.

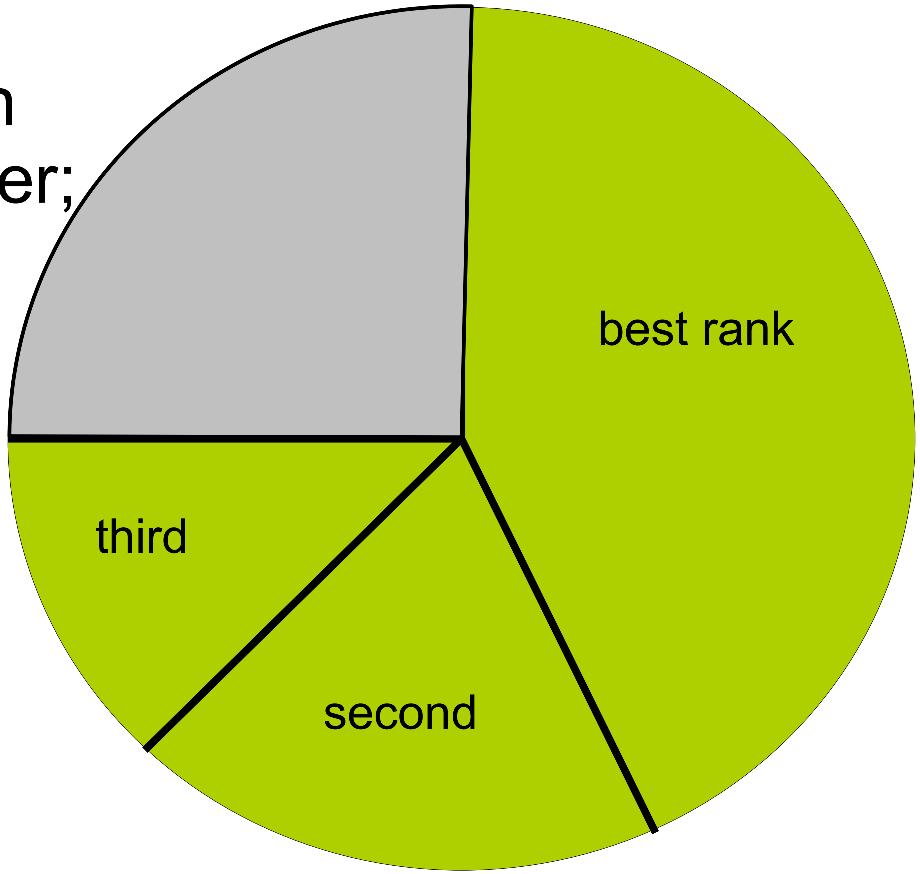


EA: Parent Selection

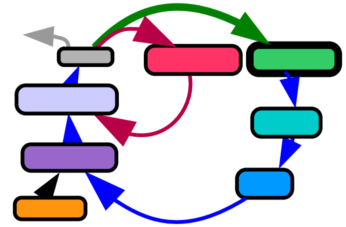


A widely used way to implement a probabilistic, rank proportionate parent selection is the **wheel of fortune**,

Where the probability ω to be chosen is larger, when the fitness $f(g)$ is better; is larger, when the rank $r(f(g))$ of the individual is smaller.

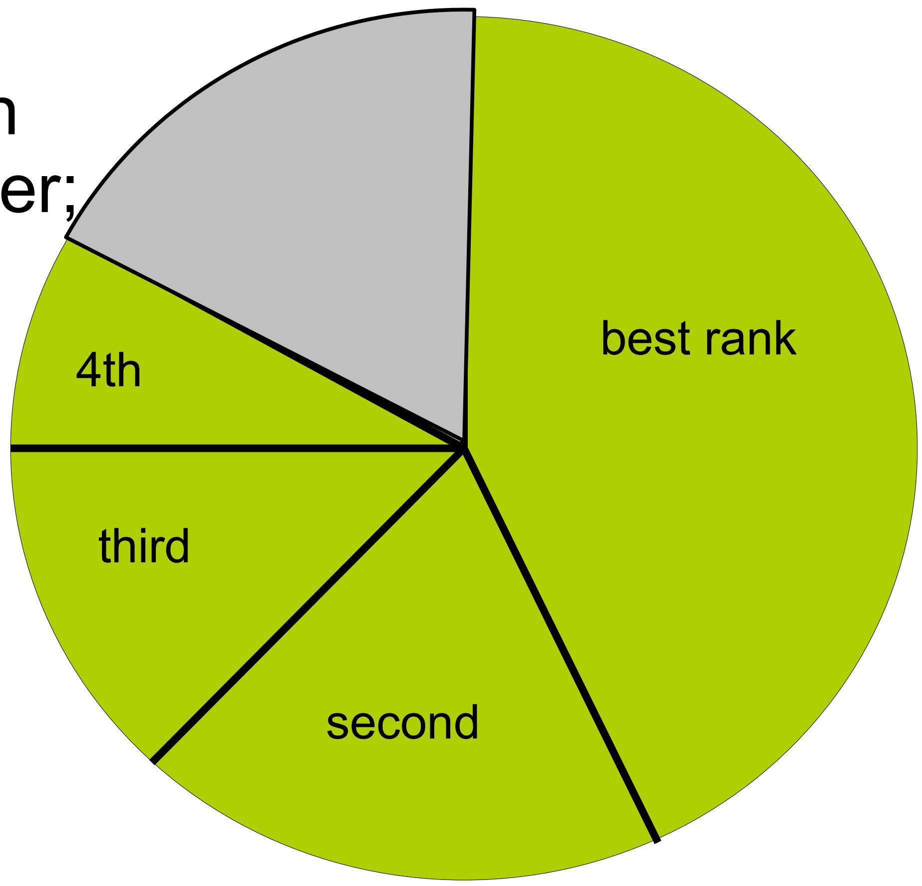


EA: Parent Selection

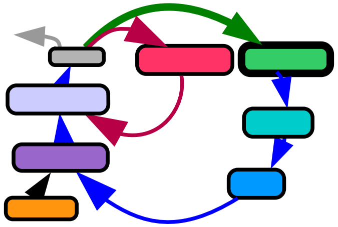


A widely used way to implement a probabilistic, rank proportionate parent selection is the **wheel of fortune**,

Where the probability ω to be chosen is larger, when the fitness **f(g)** is better; is larger, when the rank **r(f(g))** of the individual is smaller.

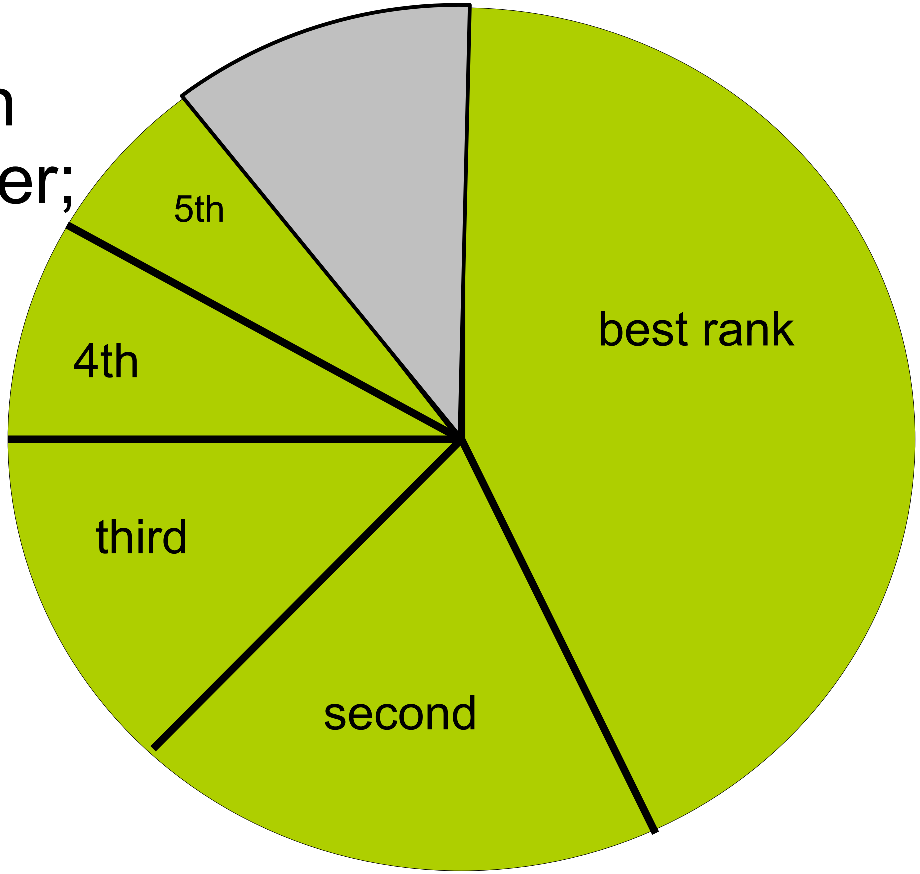


EA: Parent Selection

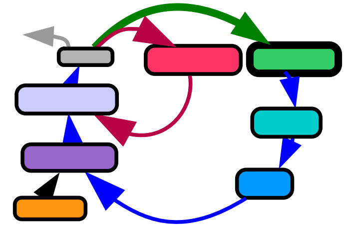


A widely used way to implement a probabilistic, rank proportionate parent selection is the **wheel of fortune**,

Where the probability ω to be chosen is larger, when the fitness **f(g)** is better; is larger, when the rank **r(f(g))** of the individual is smaller.

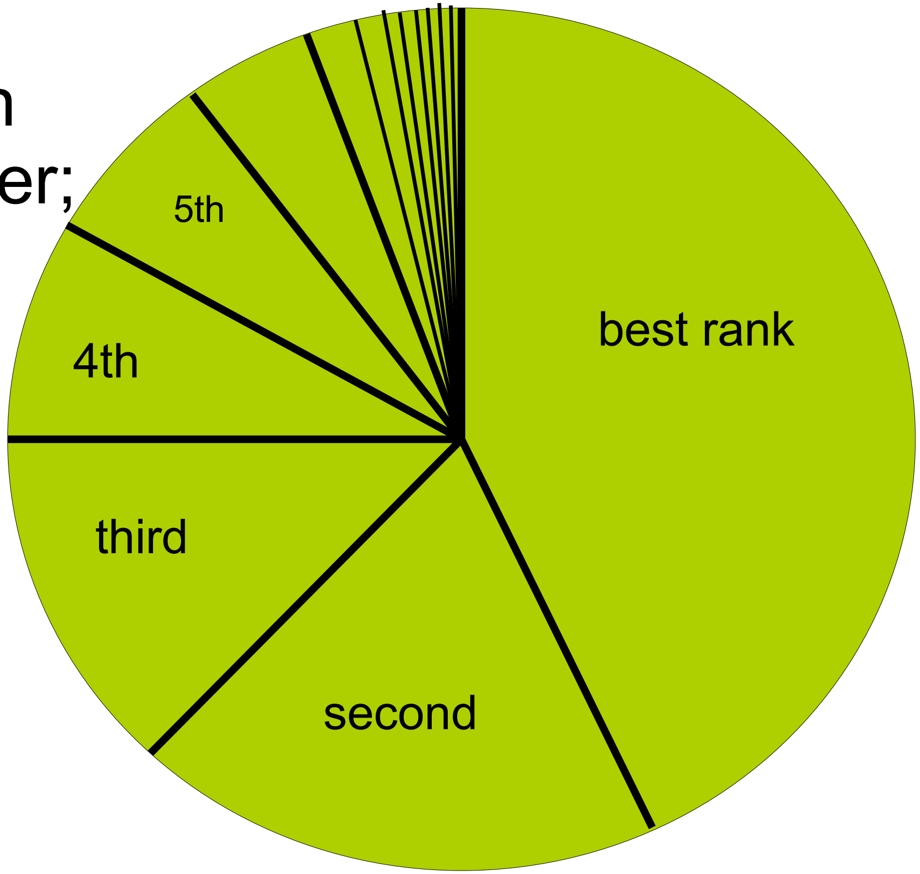


EA: Parent Selection

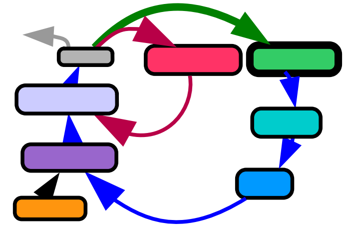


A widely used way to implement a probabilistic, rank proportionate parent selection is the **wheel of fortune**,

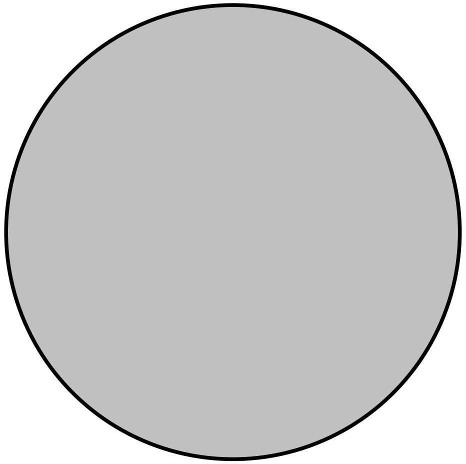
Where the probability ω to be chosen is larger, when the fitness **f(g)** is better; is larger, when the rank **r(f(g))** of the individual is smaller.



EA: Parent Selection

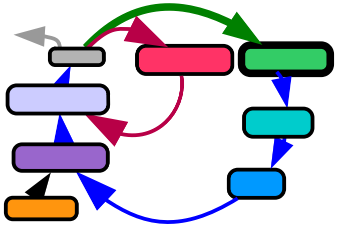


„**Wheel of Fortune**“,.
The chance, or the probability ω to be chosen is larger with smaller fitness dependent rank **r(f(g))** of the individual.



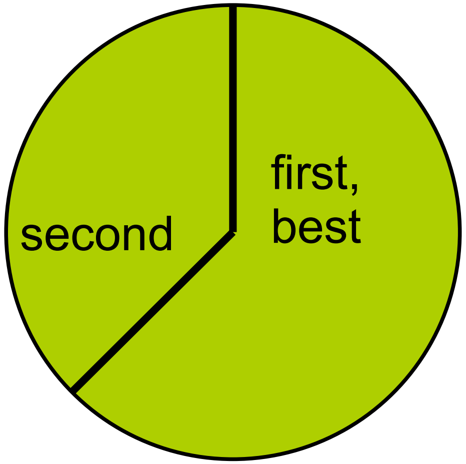
P=2:

EA: Parent Selection



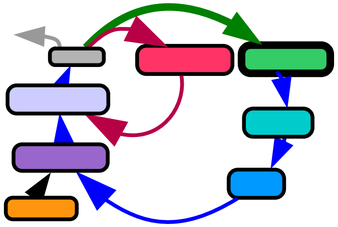
„**Wheel of Fortune**“,.

The chance, or the probability ω to be chosen is larger with smaller fitness dependent rank **r(f(g))** of the individual.



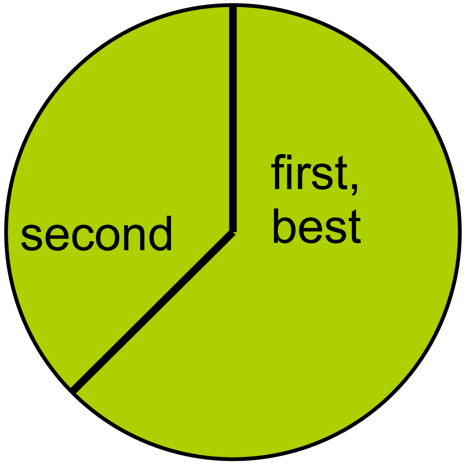
P=2:

EA: Parent Selection

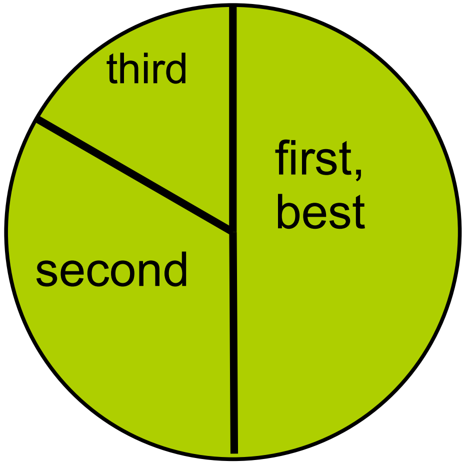


„**Wheel of Fortune**“,.

The chance, or the probability ω to be chosen is larger with smaller fitness dependent rank **r(f(g))** of the individual.

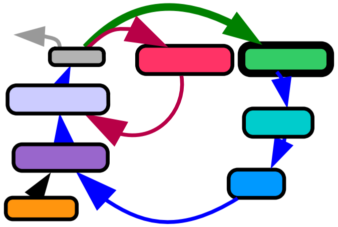


P=2:

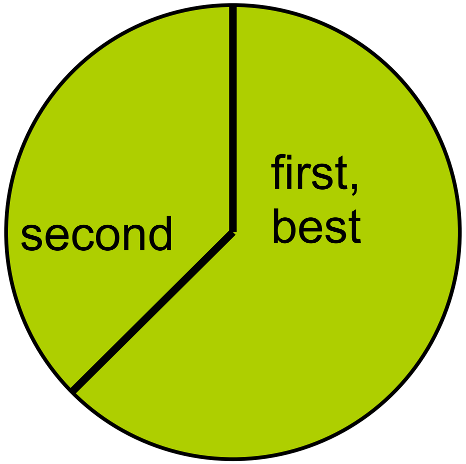


P=3:

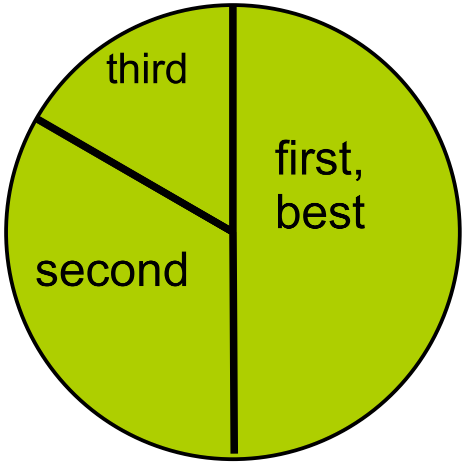
EA: Parent Selection



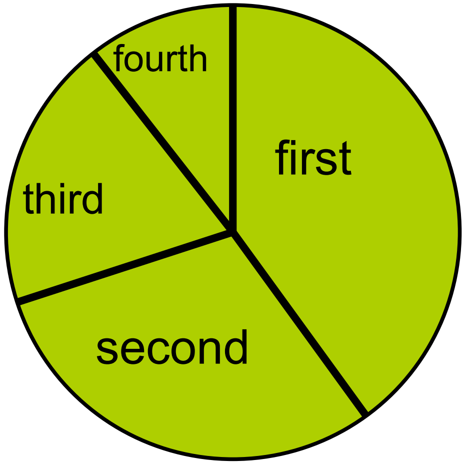
„**Wheel of Fortune**",.
The chance, or the probability ω to be chosen is larger with smaller fitness dependent rank $r(f(g))$ of the individual.



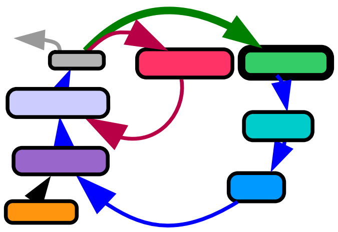
P=2:



P=3:

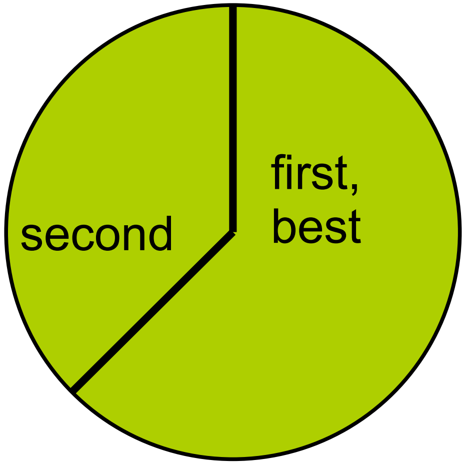


P=4:

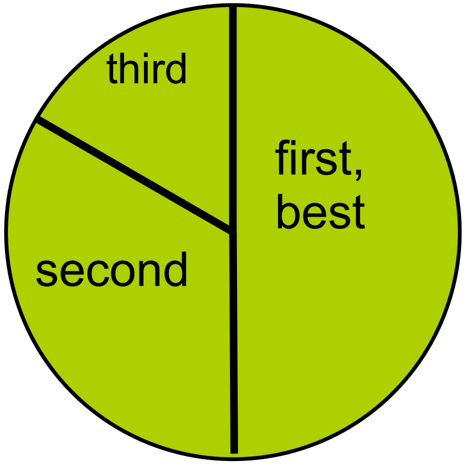


EA: Parent Selection

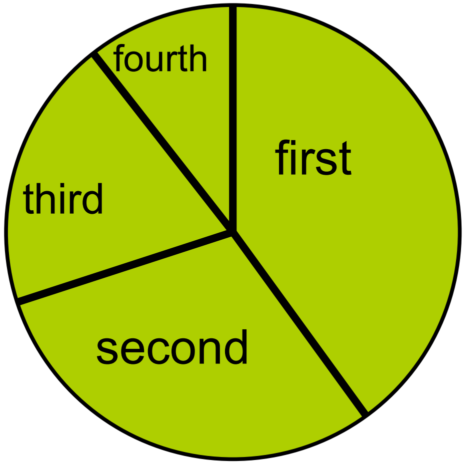
„**Wheel of Fortune**“,.
The chance, or the probability ω to be chosen is larger with smaller fitness dependent rank **r(f(g))** of the individual.



P=2:
2/3 + 1/3 = 1.0



P=3:
3/6 + 2/6 + 1/6 = 1.0

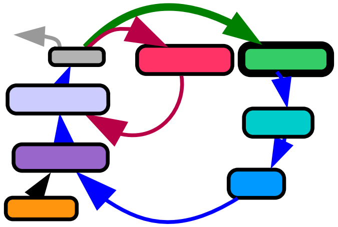


P=4:
4/10 + 3/10 + 2/10 + 1/10 = 1.0

Overview

- Genome structure
- Example: 8 queens
- Super-individuals
- External-selection and parent-selection combined
- **Probabilistic parent-selection**
  - Wheel of fortune
  - **Softmax selection**
  - Tournament selection
- Genetic programming
- Co-evolution

EA: Parent Selection

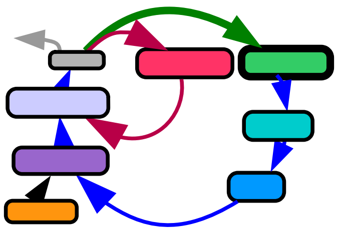


It showed to be feasible to make up the set of parents with those individuals, that have shown a good fitness **f(g)**, and explicitly include additional individuals that have not performed as good.

A **probabilistic parent selection**, depending on the reached fitness values **f(g)** is a common way to implement this.

Common probabilistic, fitness based parent selection:
- Wheel of Fortune, (Roulette-Wheel Selection)
- Boltzmann Selection, **Softmax-Selection**
- Tournament selection

EA: Parent Selection



Another common way to implement the parent selection, is using the **softmax function** for probabilistic selection.

The parents are taken randomly from the population. The probability ω_p to be taken is determined by the fitness $f(p)$ of the individual p.

$$\omega_p = \frac{e^{f(p)/\tau}}{\sum_{q=1}^{P} e^{f(q)/\tau}}$$

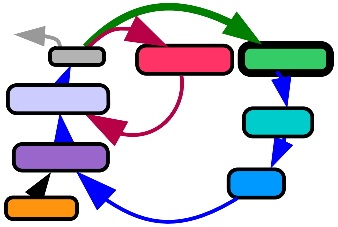
The positive parameter τ is called a temperature. Large value of τ generates a (nearly) equiprobable distribution.

Easy to implement, easy to control the selection pressure.

Overview

- Genome structure
- Example: 8 queens
- Super-individuals
- External-selection and parent-selection combined
- **Probabilistic parent-selection**
  - Wheel of fortune
  - Softmax selection
  - **Tournament selection**
- Genetic programming
- Co-evolution

EA: Parent Selection

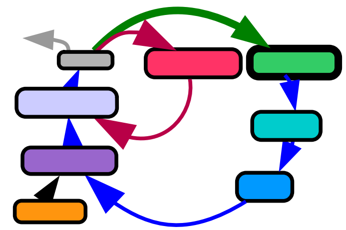


It showed to be feasible to make up the set of parents with those individuals, that have shown a good fitness **f(g)**, and explicitly include additional individuals that have not performed as good.

A **probabilistic parent selection**, depending on the reached fitness values **f(g)** is a common way to implement this.

Common probabilistic, fitness based parent selection:
- Wheel of Fortune, (Roulette-Wheel Selection)
- Boltzmann Selection, Softmax-Selection
- **Tournament selection**

EA: Parent Selection



Another common way to shape the parent selection, is called **tournament selection**.

Some individuals are taken randomly from the population, and compared pairwise (randomly chosen) in a form of **tournament**, the winner of each tournament is now eligible for the pool of parents.

To further increase the fitness of the pool of parents, the winning individuals can now undergo further stages of tournaments among each other.

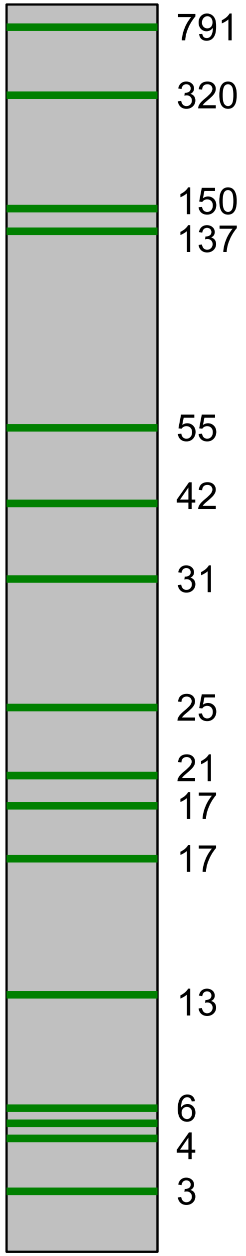
Easy to implement, easy to control the selection pressure.

EA: Parent Selection

Tournament selection

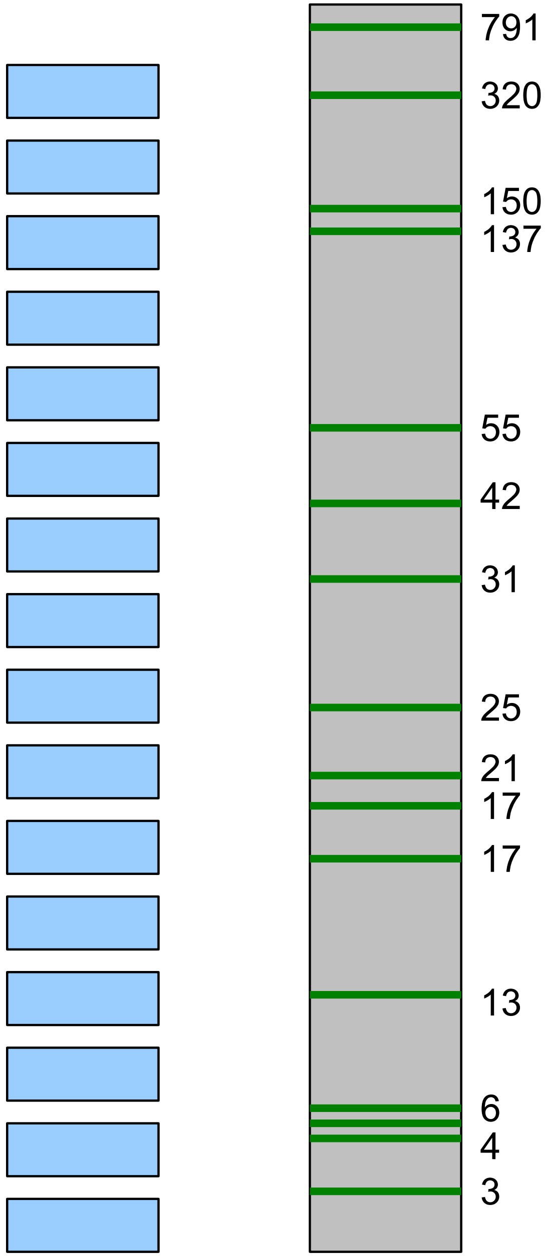
EA: Parent Selection

Tournament selection



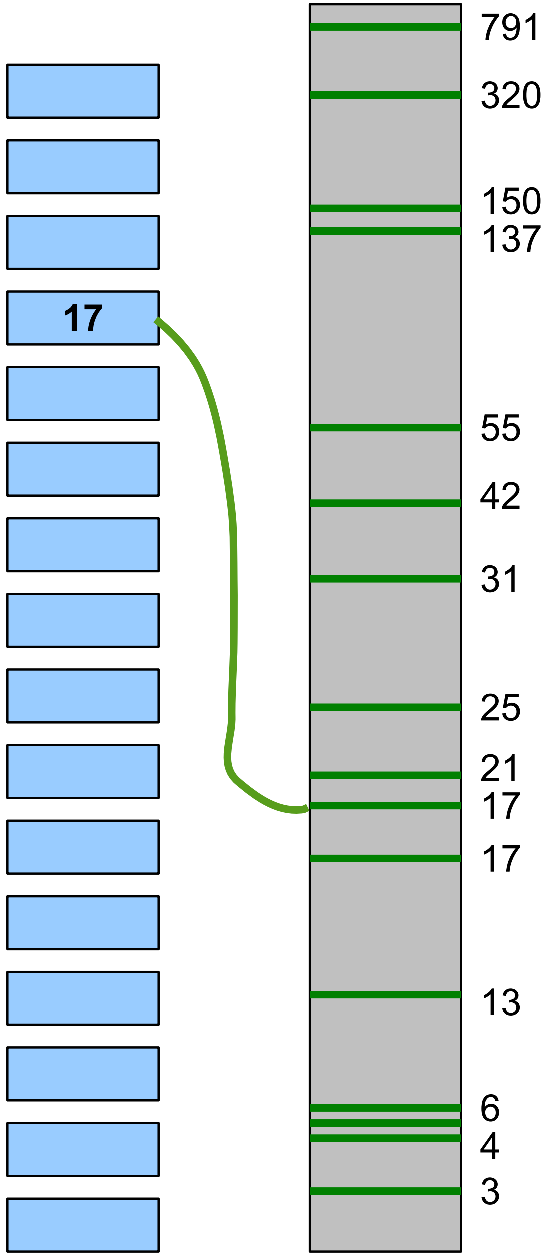
EA: Parent Selection

Tournament selection



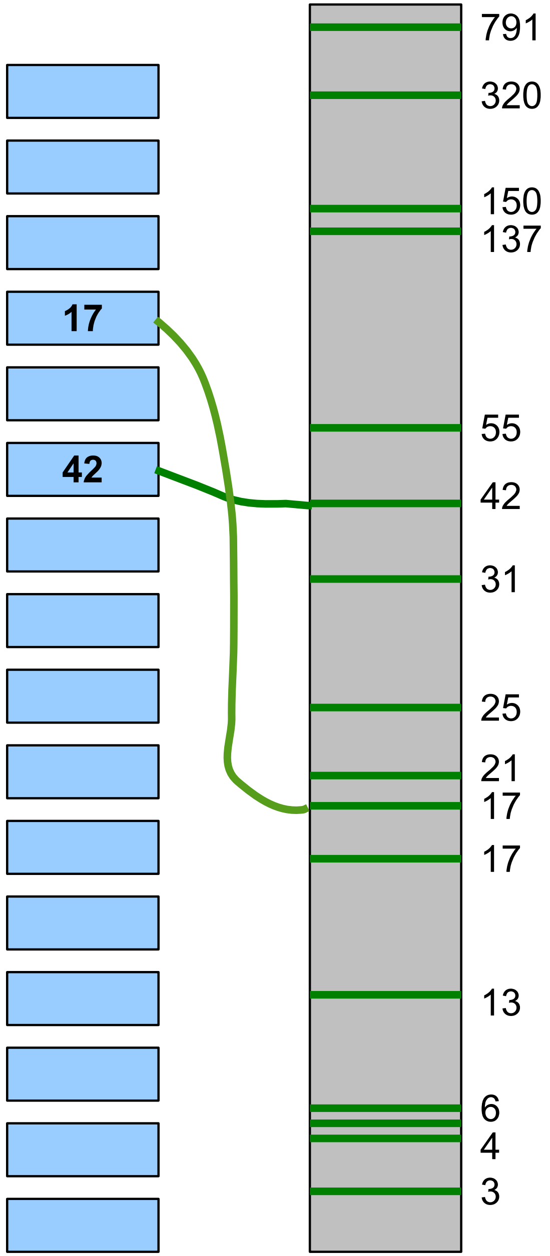
EA: Parent Selection

Tournament selection



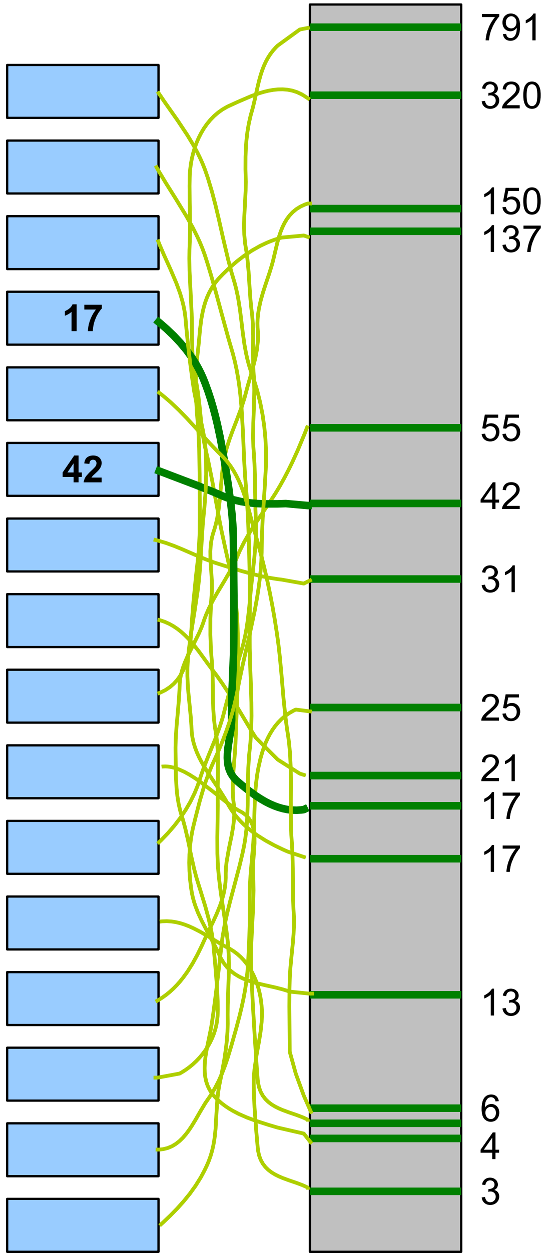
EA: Parent Selection

Tournament selection



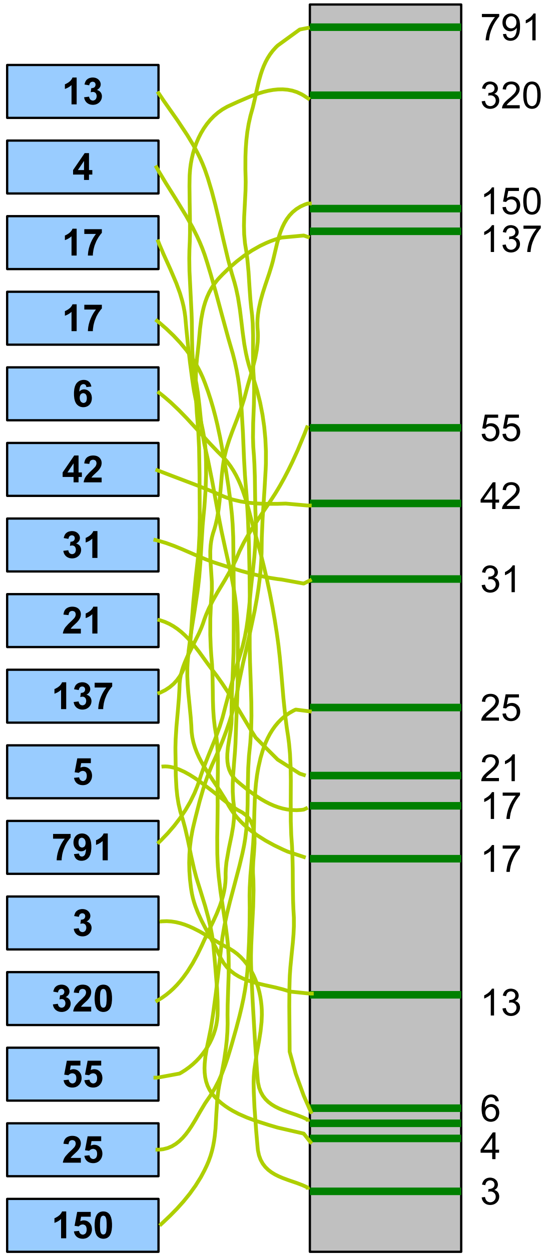
EA: Parent Selection

Tournament selection



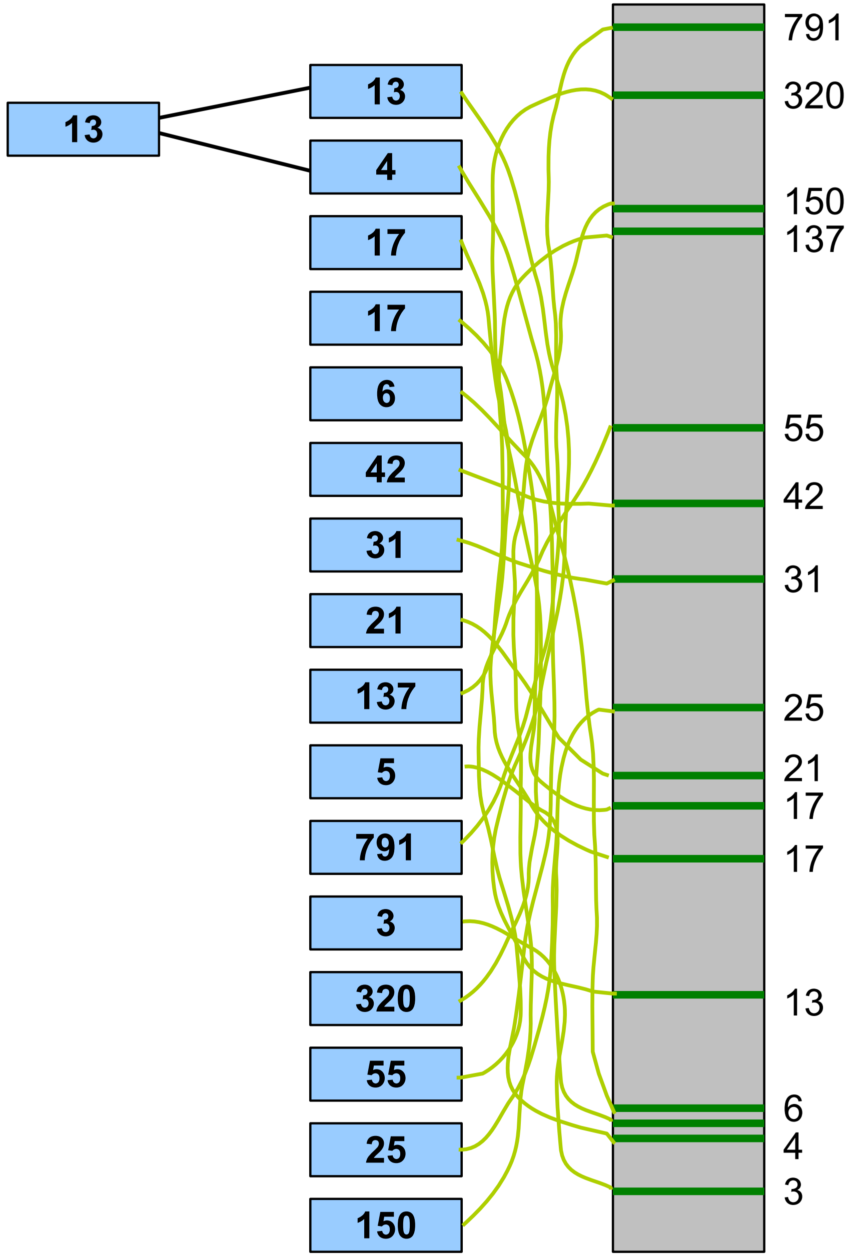
EA: Parent Selection

Tournament selection



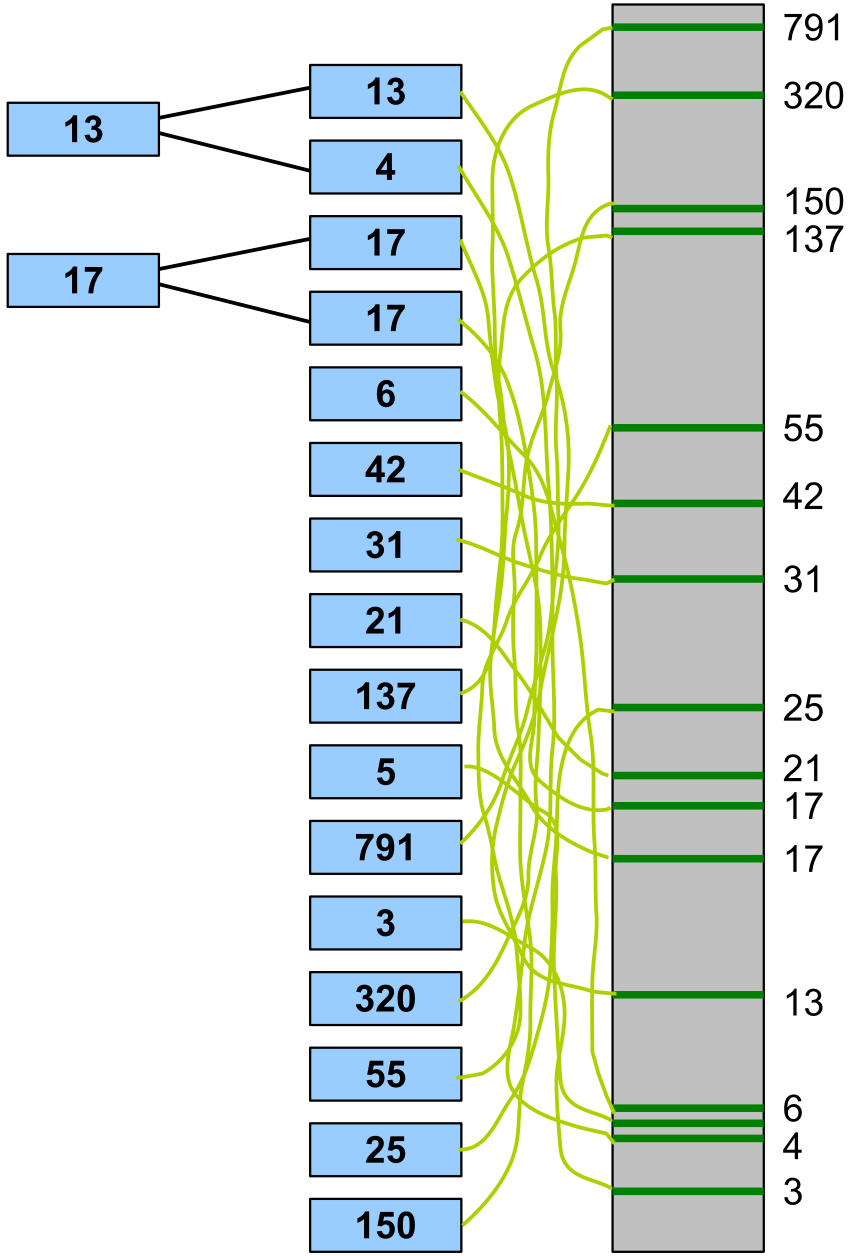
EA: Parent Selection

Tournament selection



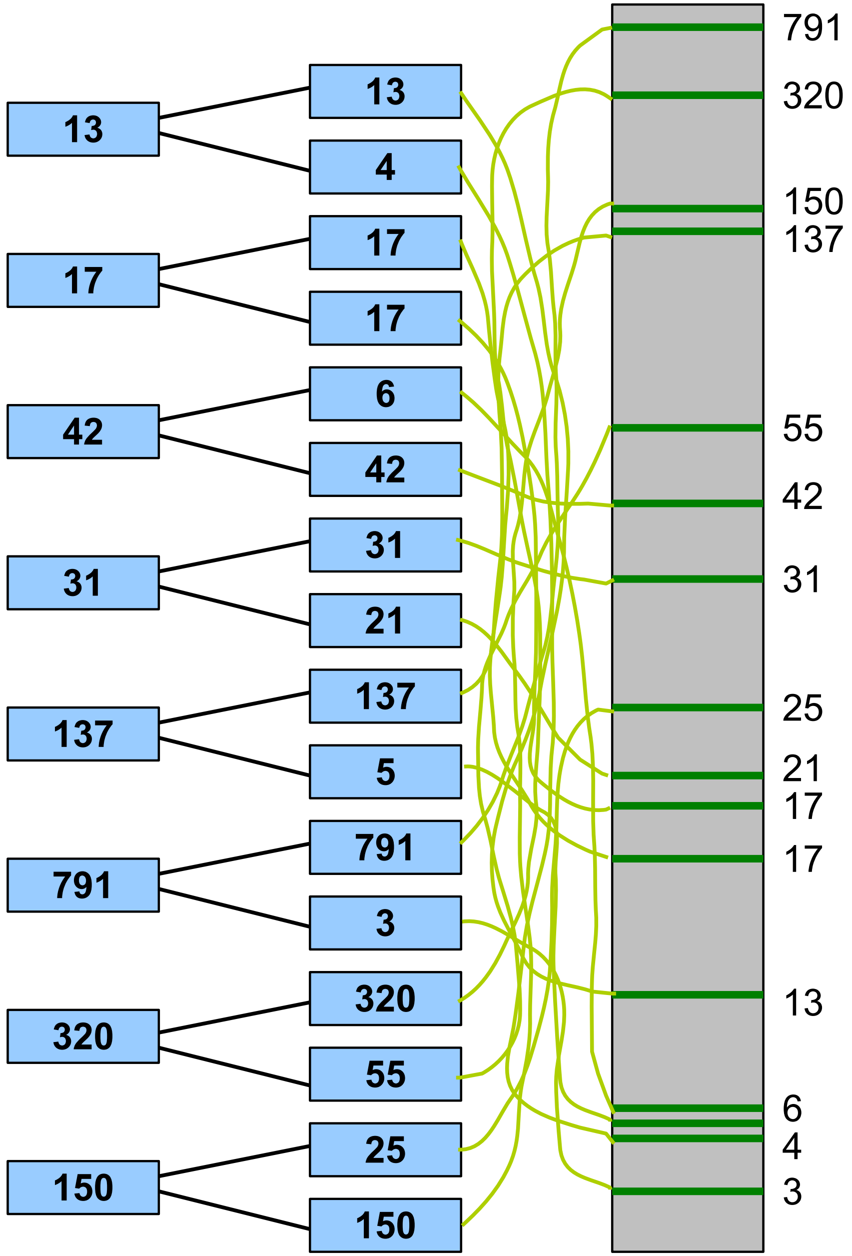
EA: Parent Selection

Tournament selection



EA: Parent Selection

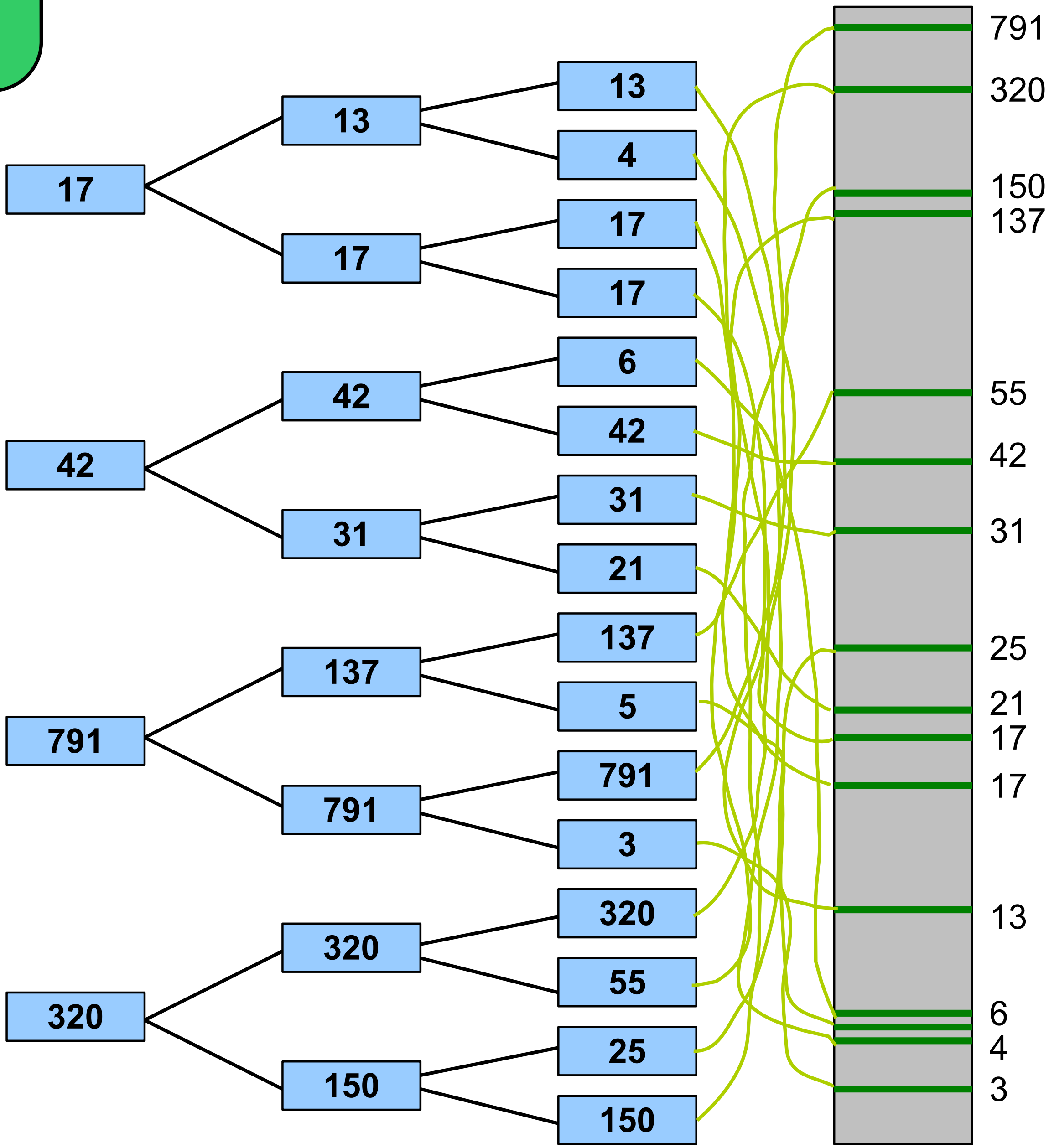
Tournament selection



EA:

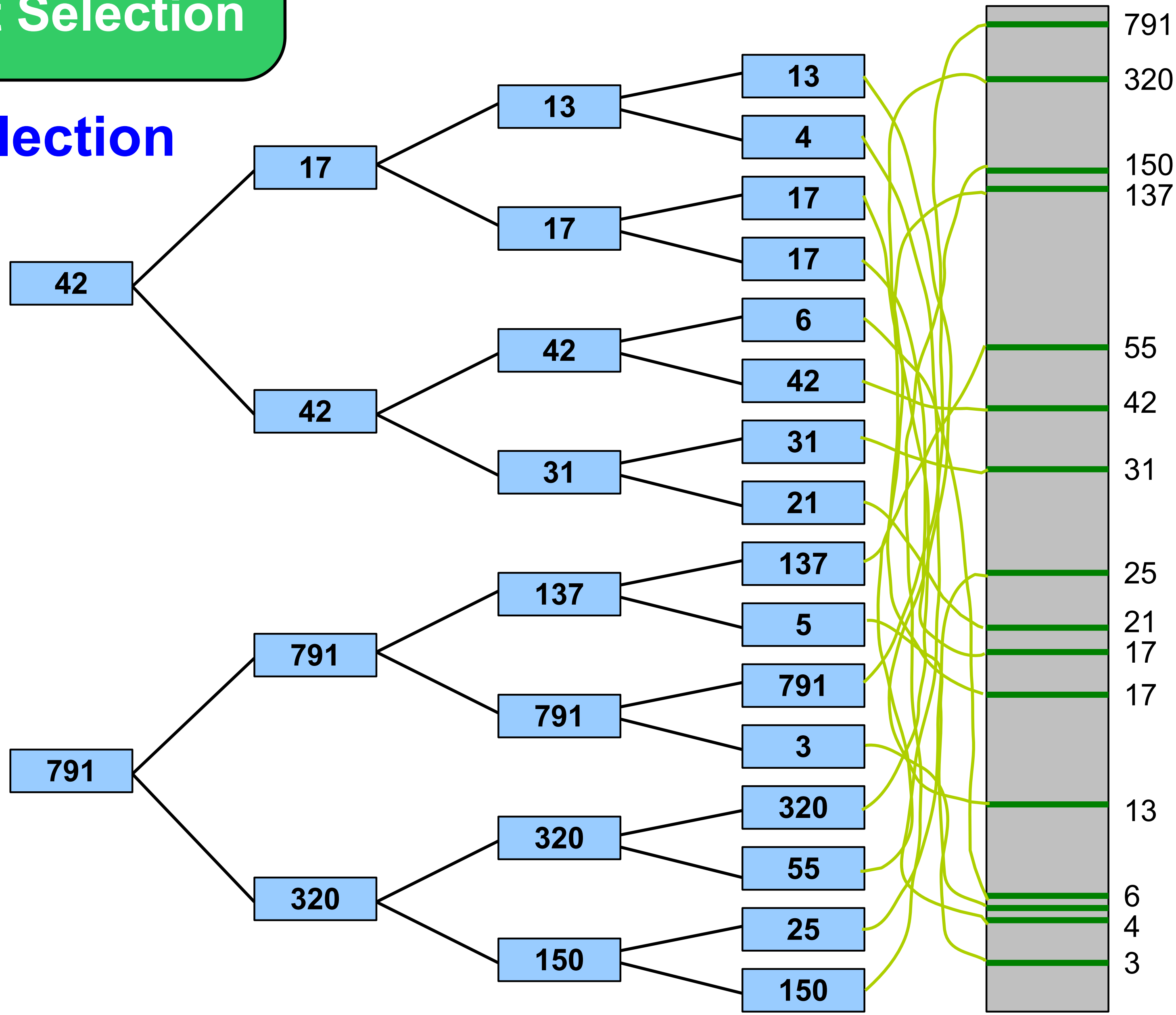
Parent Selection

Tournament selection



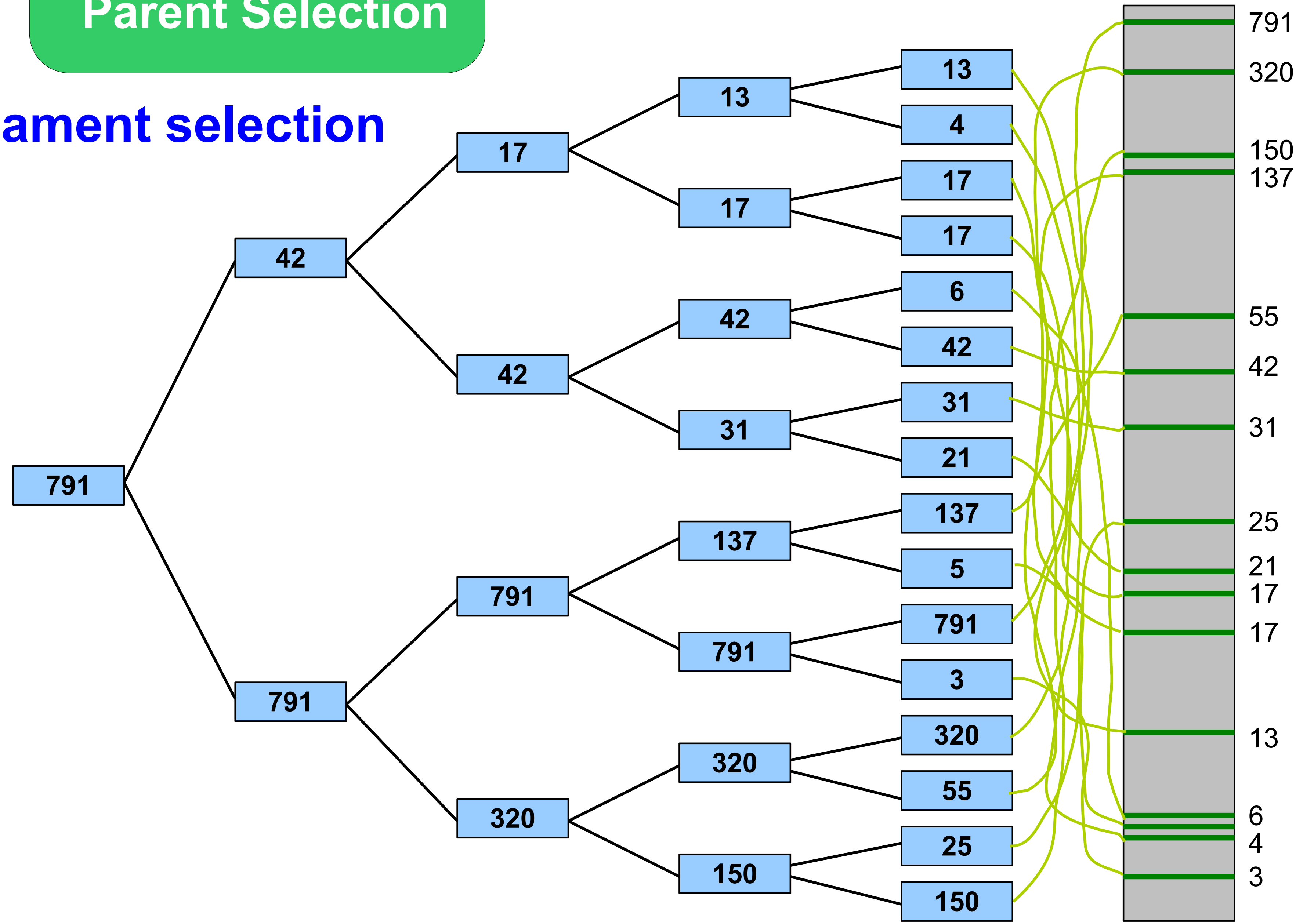
EA: Parent Selection

Tournament selection



EA: Parent Selection

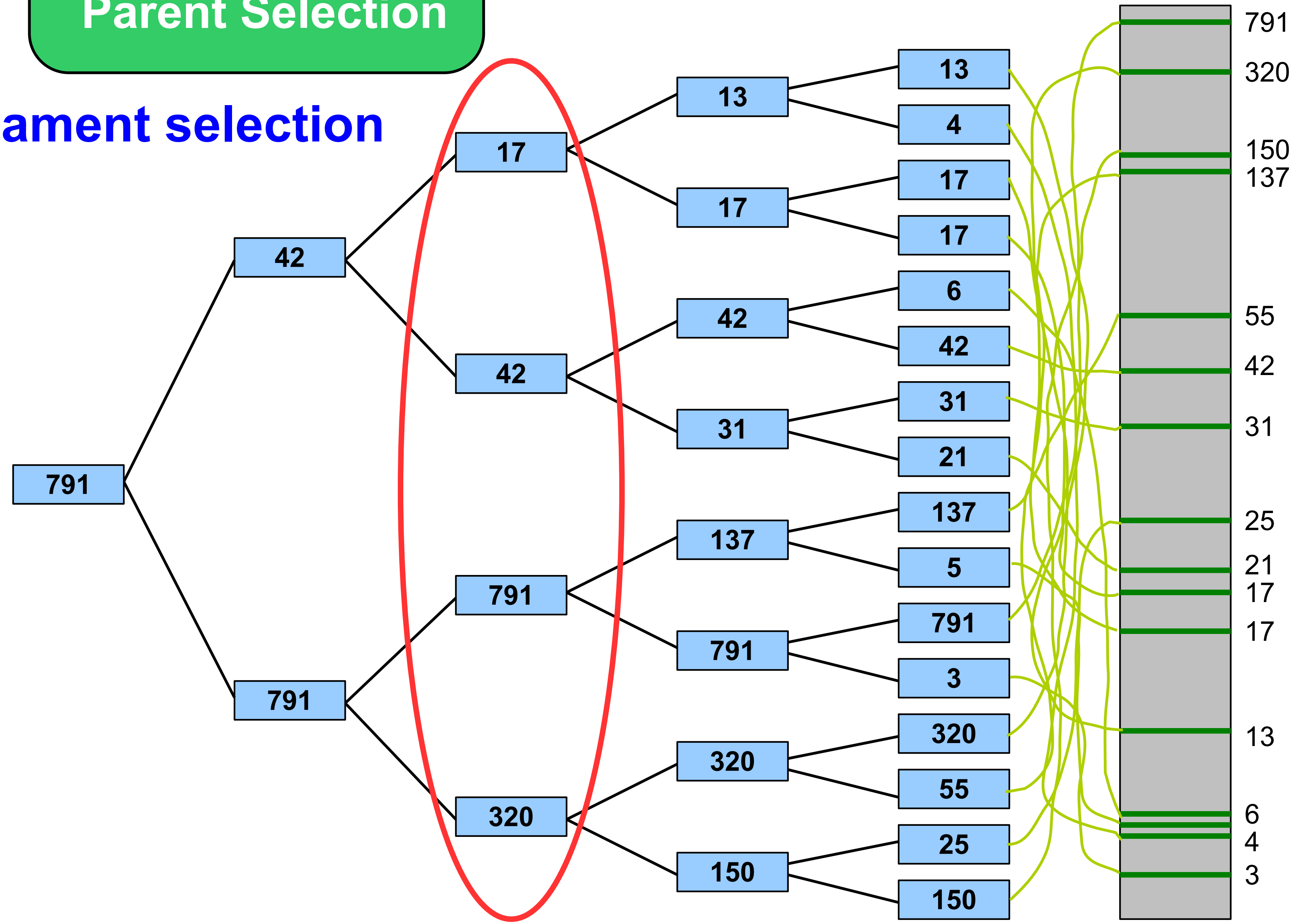
Tournament selection



EA:

Parent Selection

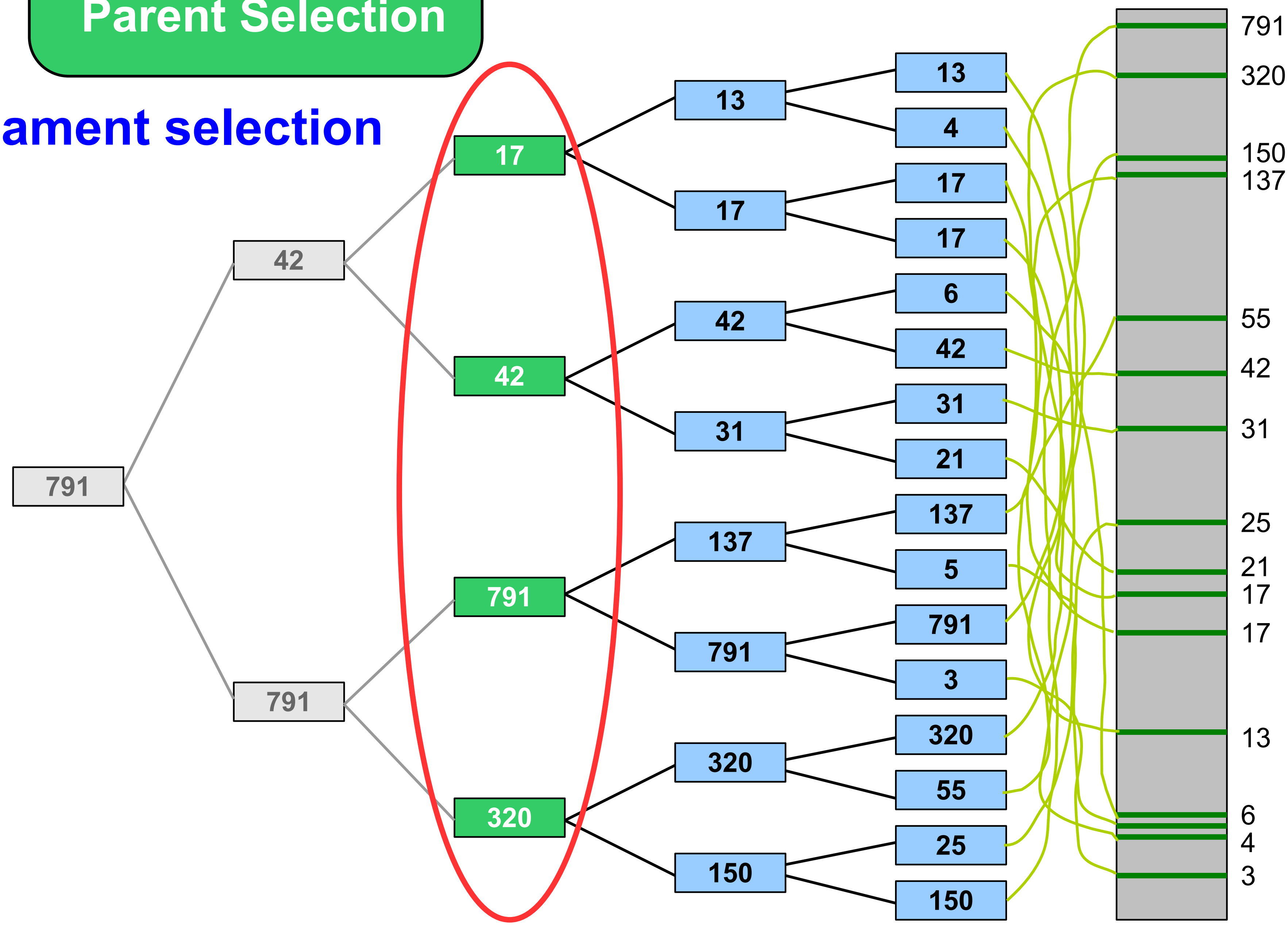
Tournament selection



EA:

Parent Selection

Tournament selection



EA: Parent Selection

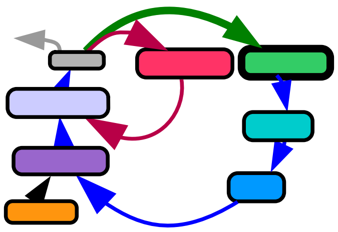
Another common way to shape the parent selection, is called **tournament selection**.

Some individuals are taken randomly from the population, and compared pairwise (randomly chosen) in a form of **tournament**, the winner of each tournament is now eligible for the pool of parents.

To further increase the fitness of the pool of parents, the winning individuals can now undergo further stages of tournaments among each other.

Easy to implement, easy to control the selection pressure.

EA: Parent Selection



It showed to be feasible to make up the set of parents with those individuals, that have shown a good fitness **f(g)**, and explicitly include additional individuals that have not performed as good.

A **probabilistic parent selection**, depending on the reached fitness values **f(g)** is a common way to implement this.

Common probabilistic, fitness based parent selection:
- Wheel of Fortune, (Roulette-Wheel Selection)
- Boltzmann Selection, Softmax-Selection
- Tournament selection

Overview

- Genome structure
- Example: 8 queens
- Super-individuals
- External-selection and parent-selection combined
- Probabilistic parent-selection
  - Wheel of fortune
  - Softmax selection
  - Tournament selection
- **Genetic programming**
- Co-evolution

Historic Remarks, Different Approaches

Evolutionary Computation (EC)

Swarm Behavior / Swarm Intelligence
- Ant Algorithm
- Ant Colony Optimization
- Particle Swarm Optimization

Evolutionary Algorithms (EA)
- Genetic Algorithms (GA)
- **Genetic Programming (GP)**
- Evolutionary Strategies (ES)
- **Evolutionary Programming (EP)**

Historic Remarks, Different Approaches

Evolutionary Computation (EC)

Evolutionary Algorithms (EA)

- Genetic Algorithms (GA)
  John Henry Holland (1975)
- Evolutionary Strategies (ES)
  Ingo Rechenberg (1965), Hans-Paul Schwefel (1970)
- **Evolutionary Programming (EP)**
  Lawrence J. Fogel (1964)
- **Genetic Programming (GP)**
  N.A.Barricelli (1954), R.M.Friedberg (1958)

EA: Genetic & Evolutionary Programming

EA: Genetic & Evolutionary Programming

Although proposed as an alternative approach to “normal computer programming”, the subject of **Genetic or Evolutionary Programming** refers to EA applications where the genome is regarded as a sequence of commands (a program).

EA: Genetic & Evolutionary Programming

Although proposed as an alternative approach to "normal computer programming", the subject of **Genetic or Evolutionary Programming** refers to EA applications where the genome is regarded as a sequence of commands (a program).

The fitness **f(g)** of the genome **g** is the quality the program achieves within the given application.

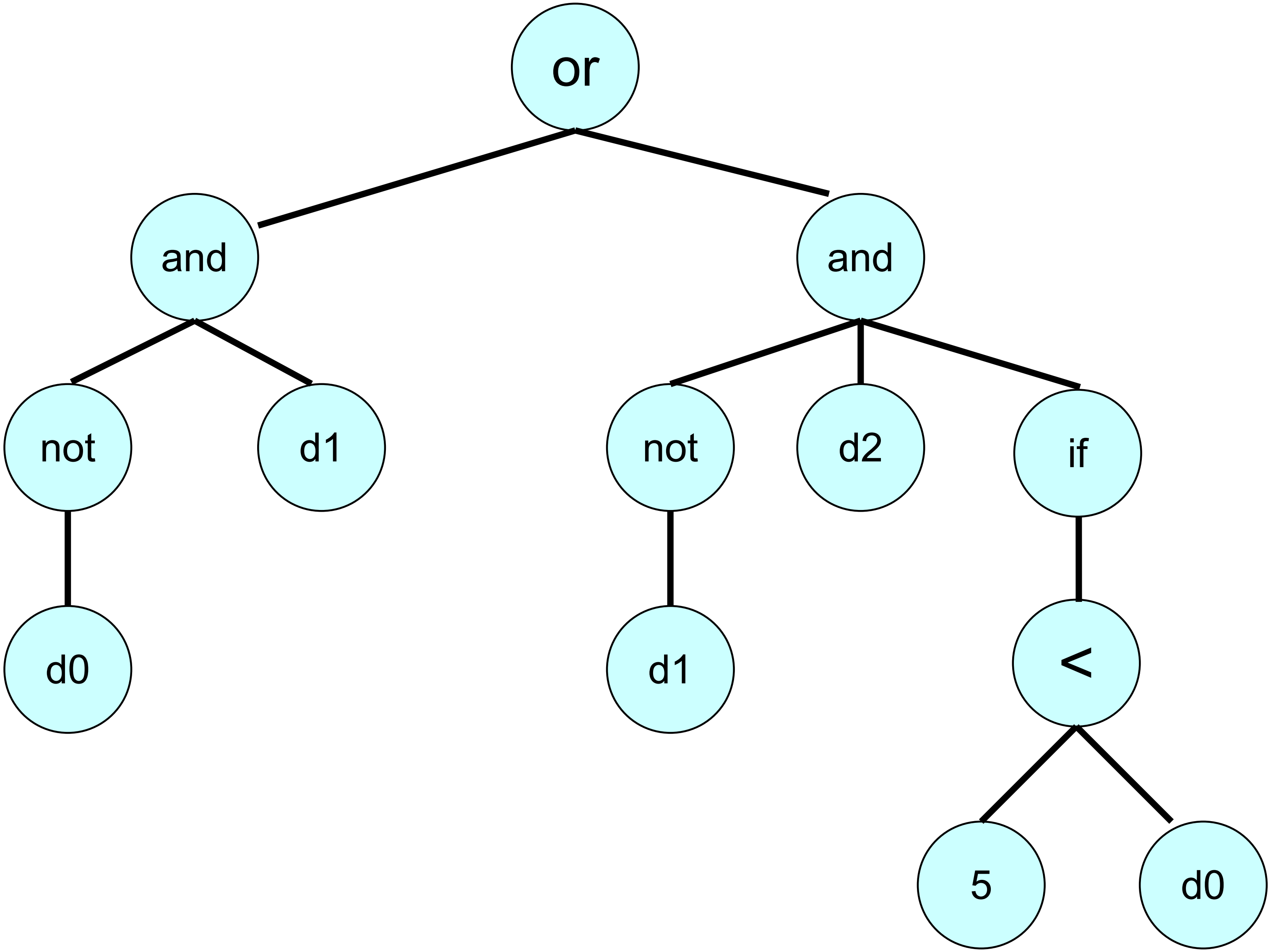
EA: Genetic & Evolutionary Programming

Although proposed as an alternative approach to “normal computer programming”, the subject of **Genetic or Evolutionary Programming** refers to EA applications where the genome is regarded as a sequence of commands (a program).

The fitness **f(g)** of the genome **g** is the quality the program achieves within the given application.

The basis for genetic programming is typically not the source code level, but a genome that is representing a formal description of the code (syntax graph, prefix notation of functions, …).

EA: Genetic & Evolutionary Programming



not d0 AND d1 OR not d1 AND d2 AND if(5<d0)

Overview

- Genome structure
- Example: 8 queens
- Super-individuals
- External-selection and parent-selection combined
- Probabilistic parent-selection
  - Wheel of fortune
  - Softmax selection
  - Tournament selection
- Genetic programming
- **Co-evolution**

EA:

Co Evolution

EA: Co Evolution

Within a normal evolutionary algorithm a fitness function **$f(g)$** is used to evaluate the performance of all P genomes **g_i** from the population.

EA: Co Evolution

Within a normal evolutionary algorithm a fitness function **f(g)** is used to evaluate the performance of all P genomes g_i from the population.

Lets take a second population with **Q** individuals and a second fitness function **h(g).**

EA: Co Evolution

Within a normal evolutionary algorithm a fitness function **f(g)** is used to evaluate the performance of all P genomes g_i from the population.

Lets take a second population with **Q** individuals and a second fitness function **h(g).**

Now these two populations are interconnected via their fitness functions **f** and **h** , where one population is creating the fitness for the other population, and vice versa.

EA: Co Evolution

Within a normal evolutionary algorithm a fitness function **f(g)** is used to evaluate the performance of all P genomes g_i from the population.

Lets take a second population with **Q** individuals and a second fitness function **h(g).**

Now these two populations are interconnected via their fitness functions **f** and **h** , where one population is creating the fitness for the other population, and vice versa.

The population is not evaluated against a static fitness function, but against a changing other population.
The performance of one population is the fitness for the others population.

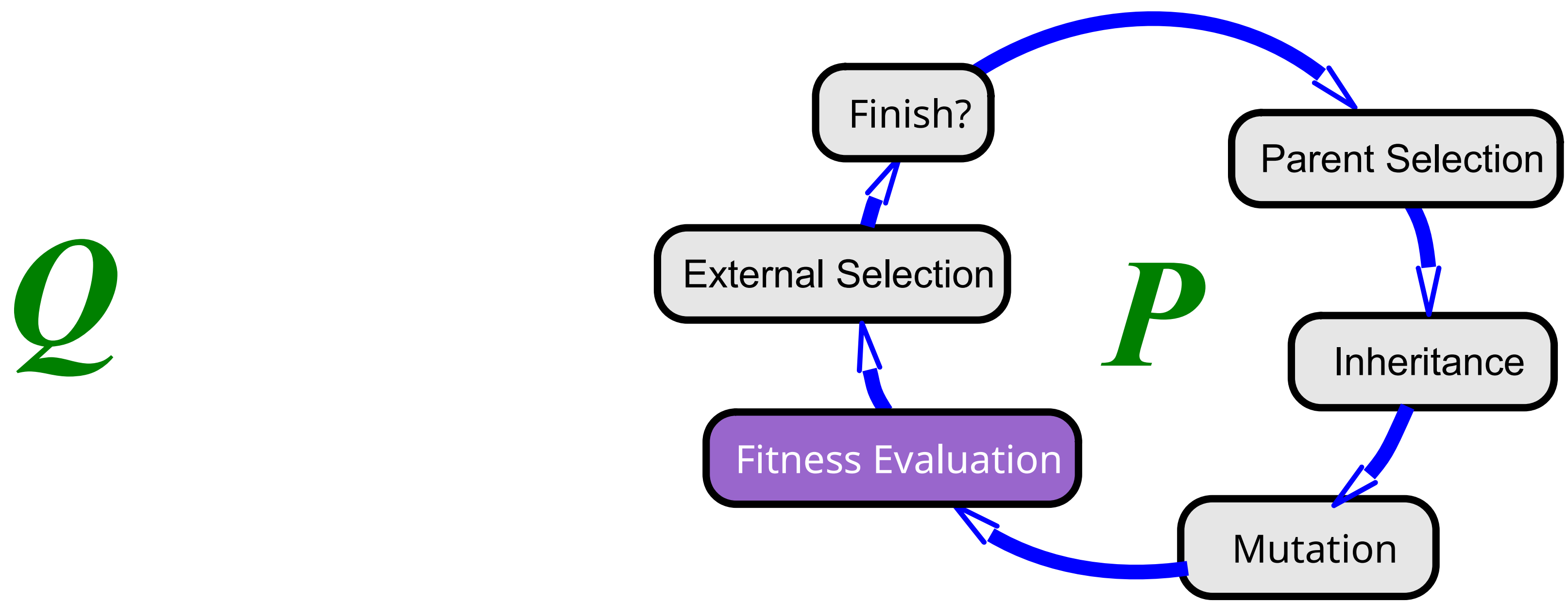
EA: Co Evolution

Two populations are interconnected via their fitness functions where one population is creating the fitness for the other population, and vice versa.
The performance of one population is the fitness for the other population.

Q P

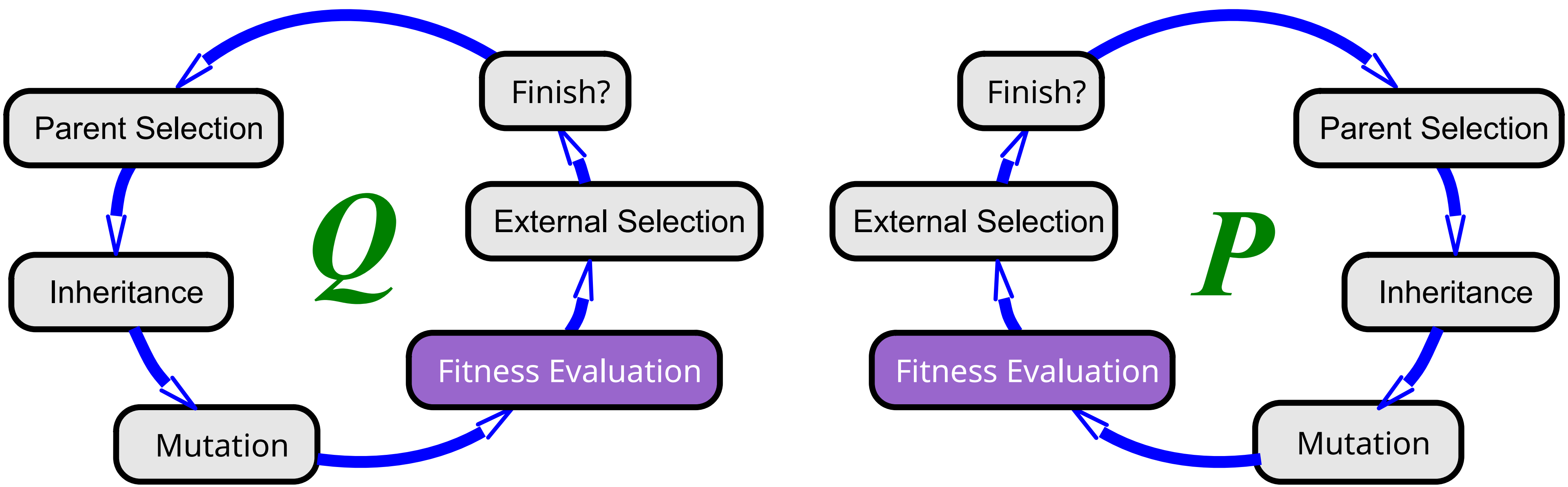
EA: Co Evolution

Two populations are interconnected via their fitness functions where one population is creating the fitness for the other population, and vice versa.
The performance of one population is the fitness for the other population.



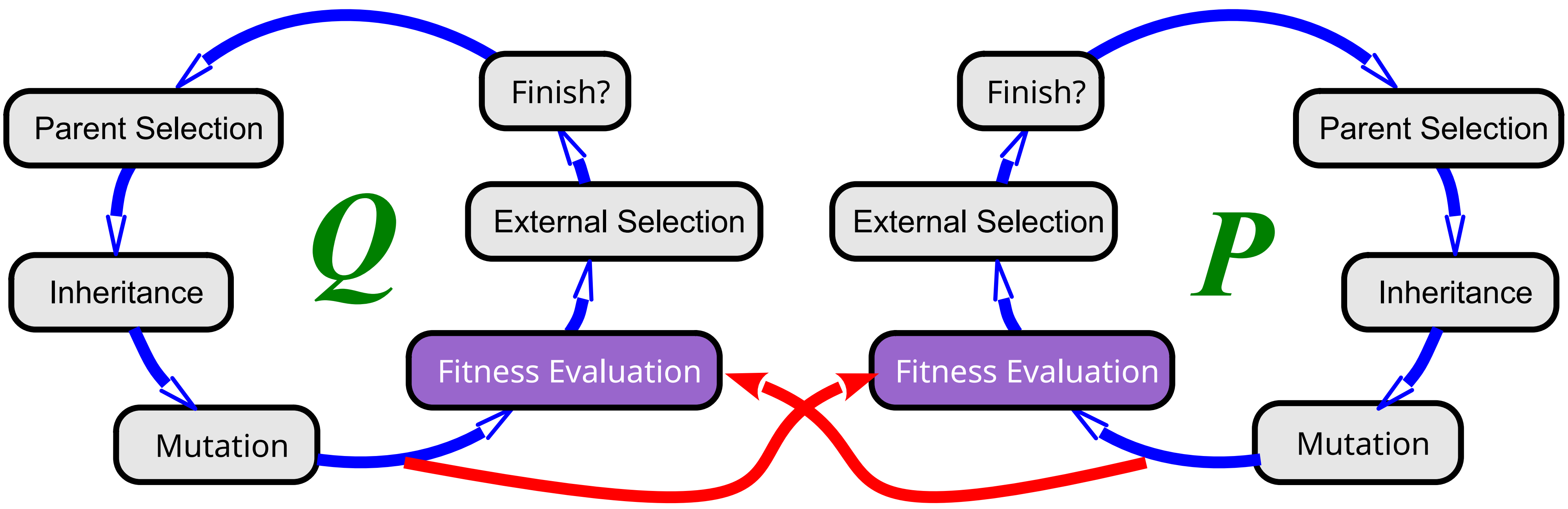
EA: Co Evolution

Two populations are interconnected via their fitness functions where one population is creating the fitness for the other population, and vice versa.
The performance of one population is the fitness for the other population.



EA: Co Evolution

Two populations are interconnected via their fitness functions where one population is creating the fitness for the other population, and vice versa.
The performance of one population is the fitness for the other population.



EA: Co Evolution

Co evolution can be used to simulate the development of two species interacting:

e.g. a predator-prey system
or two agents, or two players in games, ...

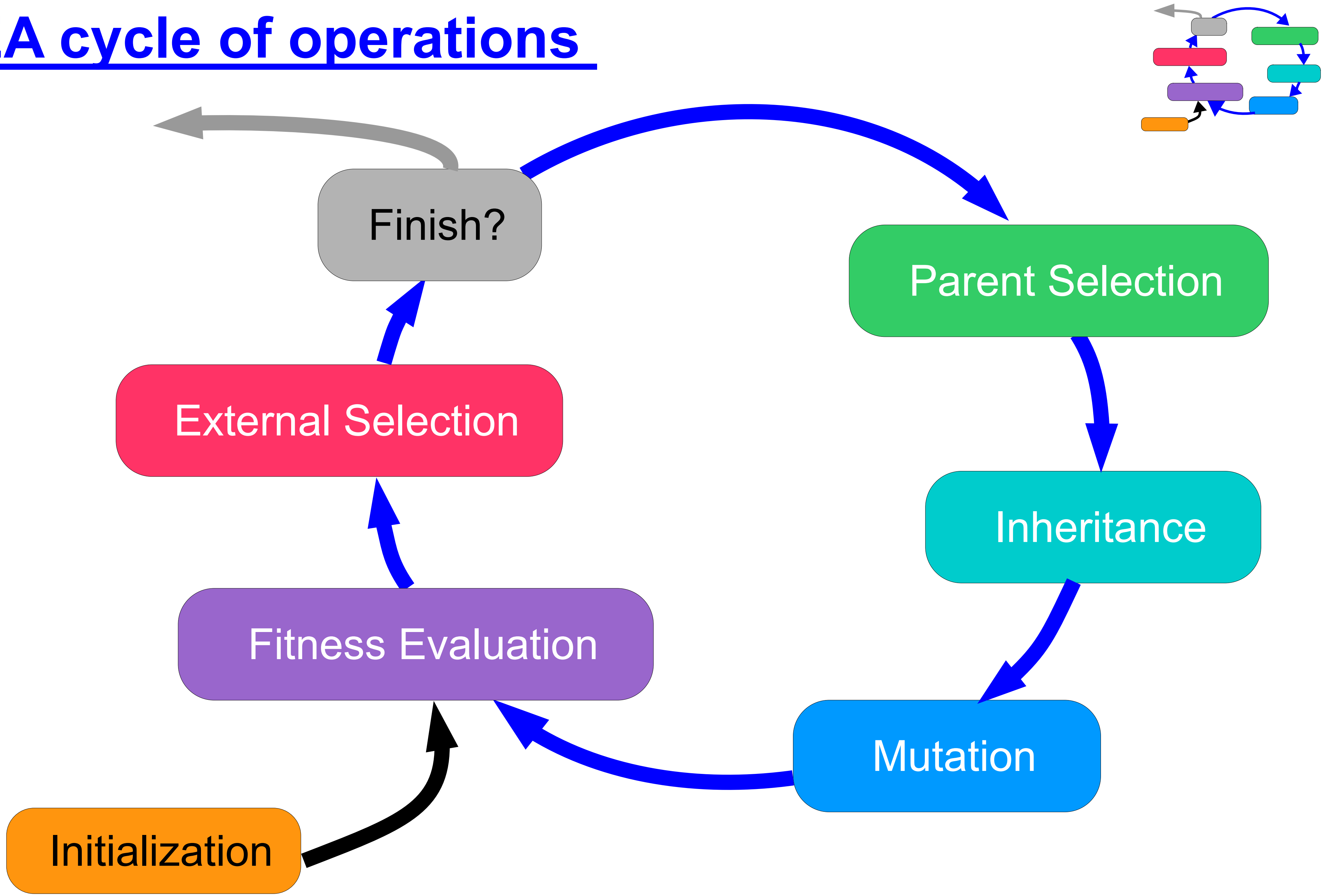
Since both fitness functions, are not static but changing over time, a final objective is sometimes hard to determine.

The achieved results in co evolution for one population is only use-able in interaction with the other population.

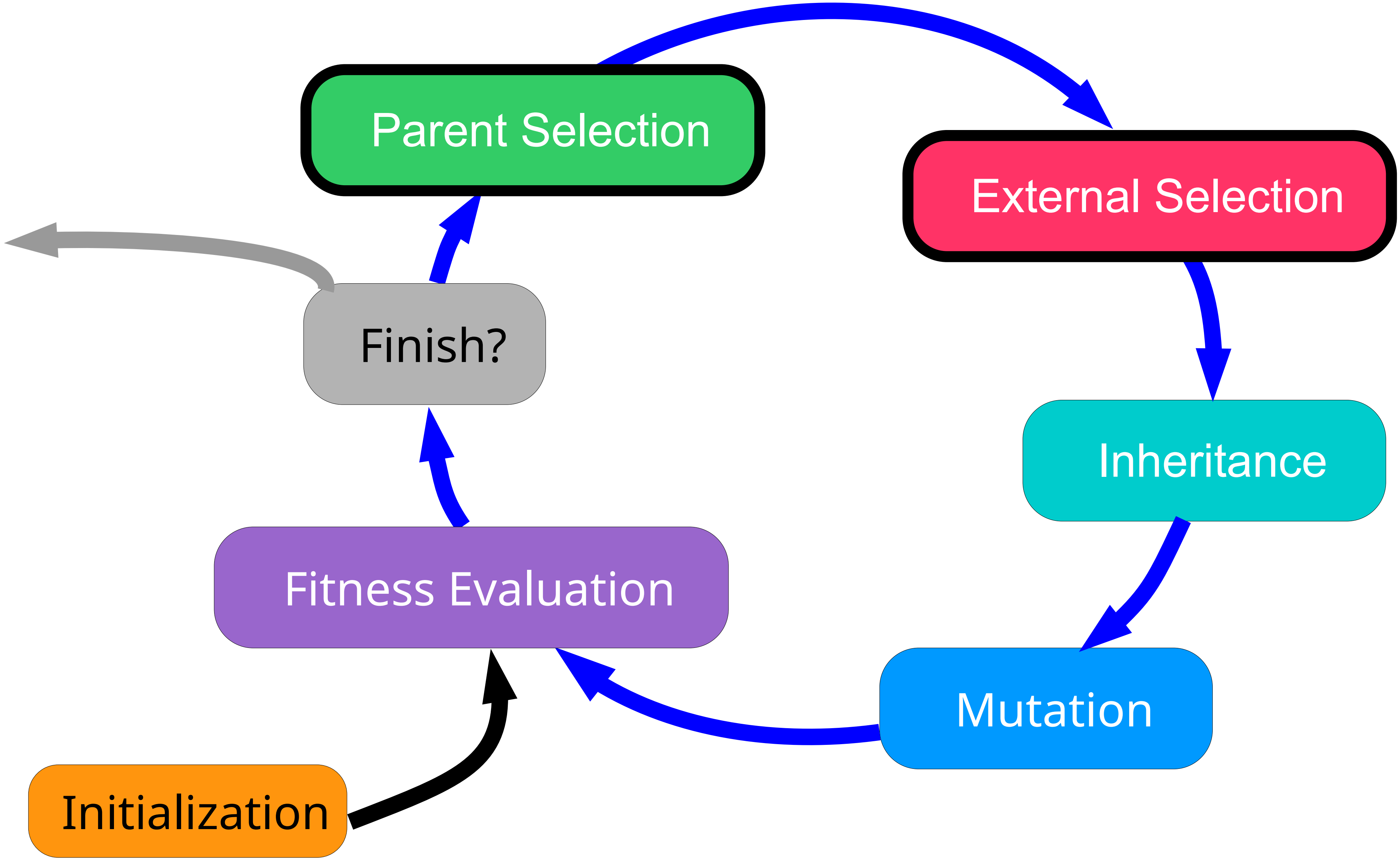
Overview

- Genome structure
- Example: 8 queens
- Super-individuals
- External-selection and parent-selection combined
- Probabilistic parent-selection
  - Wheel of fortune
  - Softmax selection
  - Tournament selection
- Genetic programming
- Co-evolution

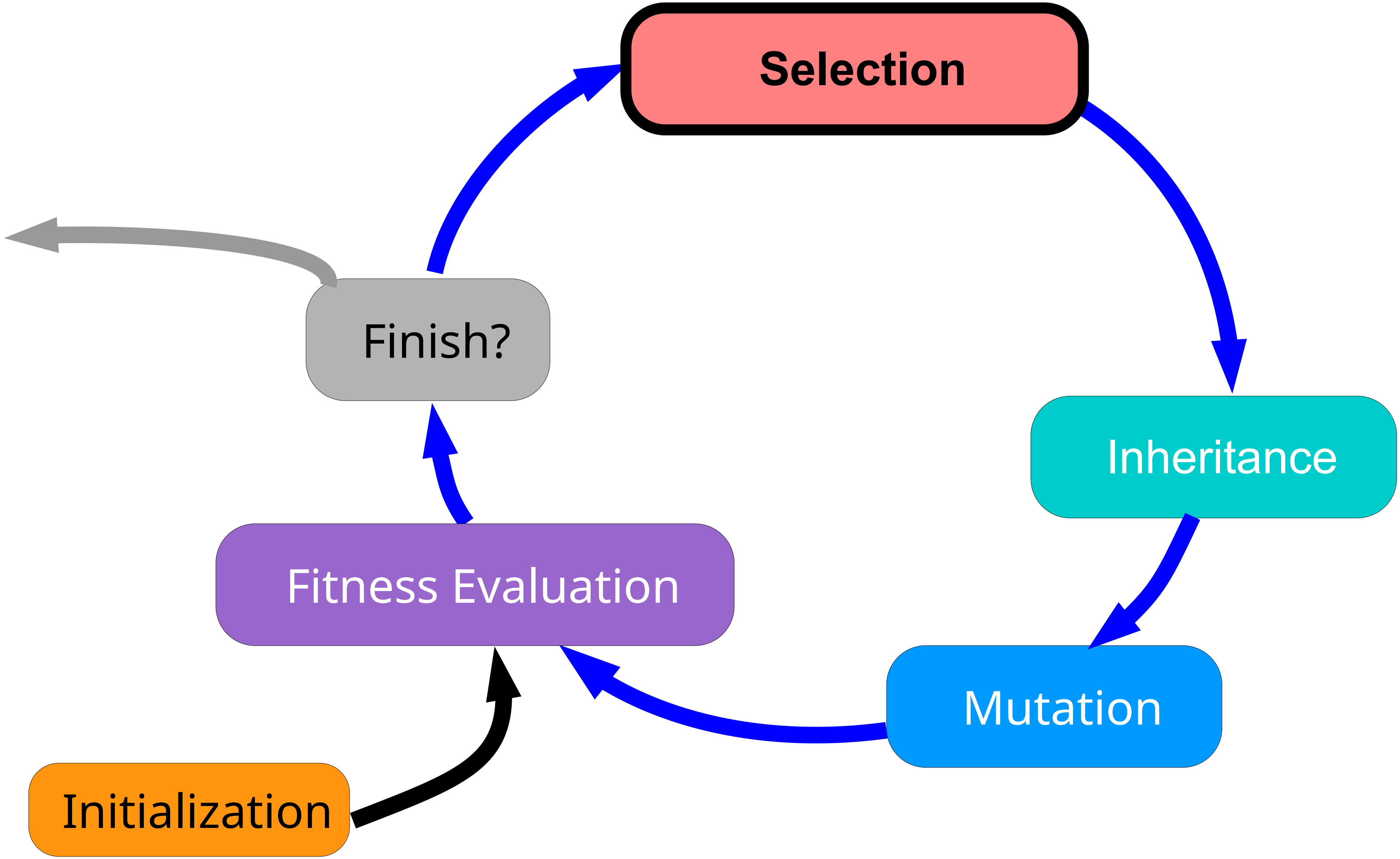
EA cycle of operations



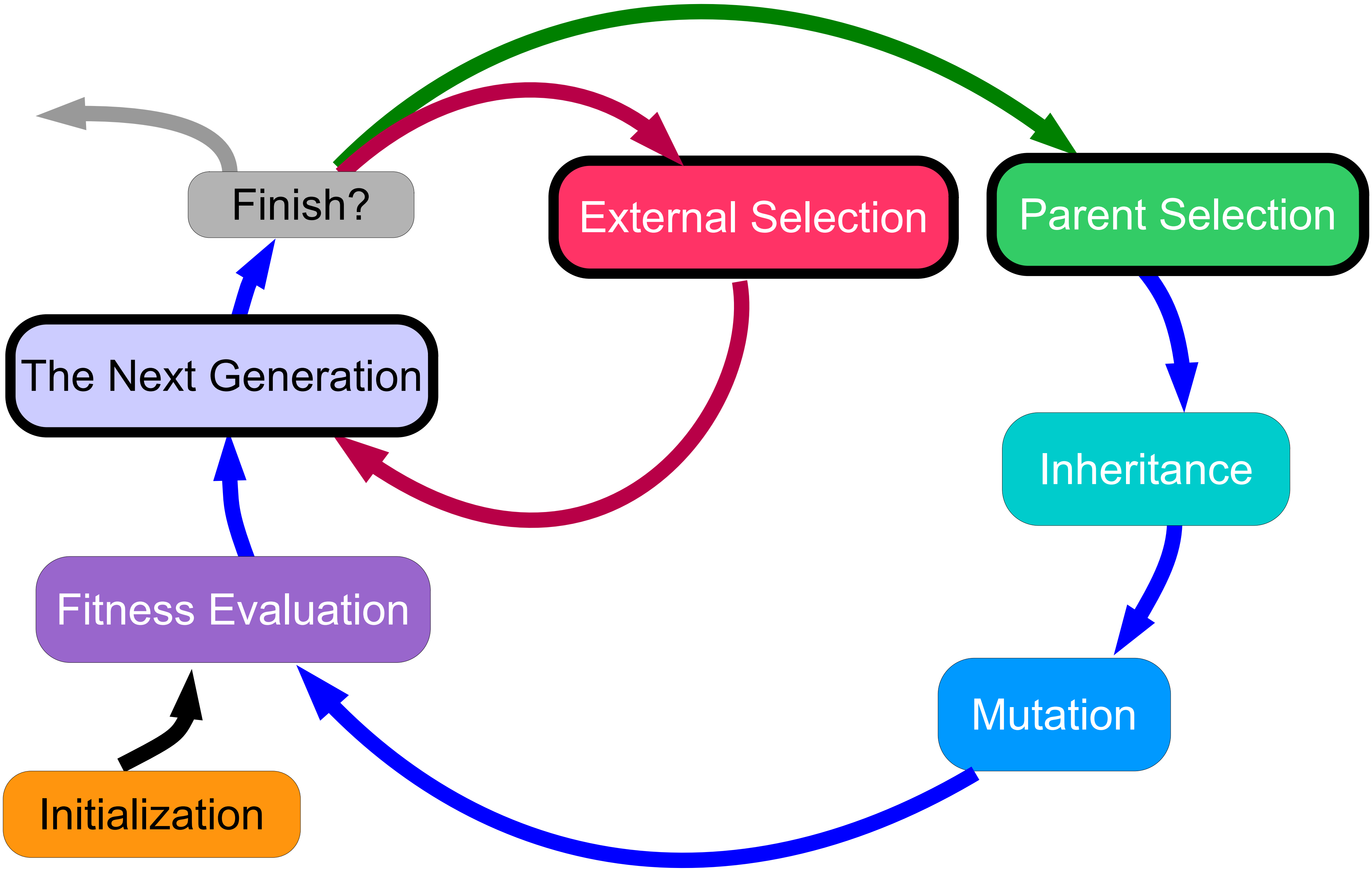
EA cycle of operation



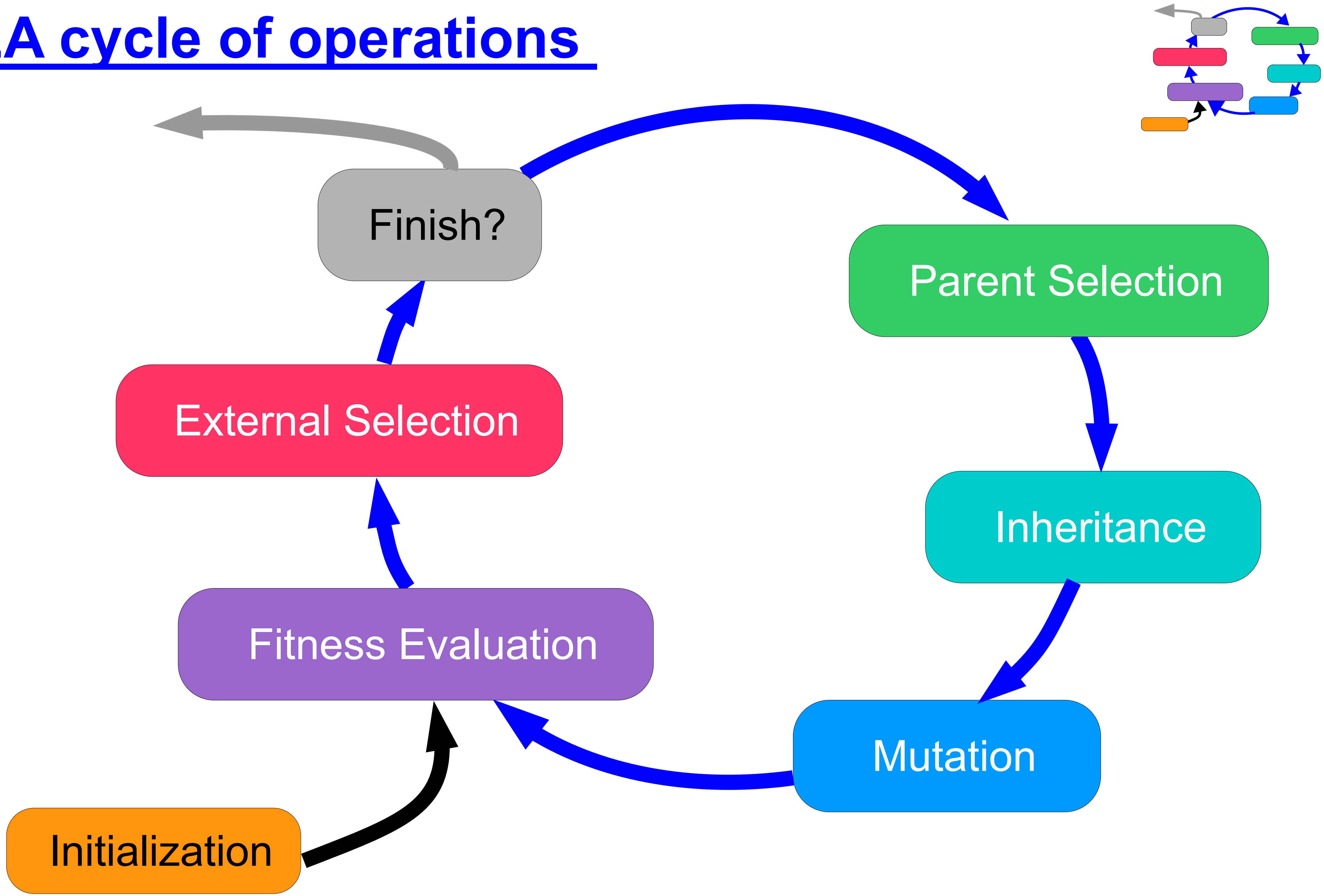
EA cycle of operation



EA alternative cycle of operation



EA cycle of operations



Some important dates

Wed 14.5.25: **Dies Academicus** , special talks, no regular teaching

Thu 29.5.25 : **Ascension Day** , no lectures, no exercises, ...

Sun 8.6. - 9.6.25 : **Pentecost, Whitsun, Pfingsten**, Public holiday
Tue 10.6.25 – Fri 13.6.25 : Excursion week, no lectures, exercises, …

Thu 19.6.25 : **Feast of Corpus Christi**, no lectures, exercises, …

Programming Assignment PA-E

Write a Python Programm, that implements an evolutionary algorithm to **maximize the length** of a route going **twice** through a given set of *N* points (cities) in 2-dimensions.

Starting, and ending point are open to be determined by the algorithm; each point (city) must be visited **exactly twice**.

The *N* points $X_n = (x_1, x_2)_n$ shall be read in from the text-file
Positions_PA-E.txt

This list are the positions of European Capitals w.r.t. a map:
http://upload.wikimedia.org/wikipedia/commons/6/64/Europe_capitals_map_de.png

There is an additional new file: Positions2_PA-E.txt
augmented with the names of the European Capitals.

Programming Assignment PA-E

This list are the positions of European Capitals w.r.t. a map:
http://upload.wikimedia.org/wikipedia/commons/6/64/Europe_capitals_map_de.png

There is an additional new file: Positions2_PA-E.txt
augmented with the names of the European Capitals.

```
#Artificial_Life_Lecture_SS25_Programming_Assignment_PA_E_List_of_46_Positions_of_European_Capitals_in_2D
#Positions_are_according_to_map_http://upload.wikimedia.org/wikipedia/commons/6/64/Europe_capitals_map_de.png
 1    446   621      #Amsterdam
 2    328   913      #Andorra_la_Vella
 3   1062 966   #Ankara
 4    857  1071 #Athens
 5    755   884      #Belgrade
 6    606   627      #Berlin
 7    477   798      #Bern
 8    684   764      #Bratislava
 9    430   671      #Brussels
10    889   857      #Bucharest
11    720   786      #Budapest
12    927   780      #Chisinau
13    592   535      #Copenhagen
14    249   548      #Dublin
15    788   383      #Helsinki
16    940   667      #Kiew
17     51   960      #Lisbon
18    627   831      #Ljubljana
19    …..   …..      ………...
```

http://upload.wikimedia.org/wikipedia/commons/6/64/Europe_capitals_map_de.png

Artificial Life
Summer 2025

Evolutionary Algorithms 3

Master Computer Science [MA-INF 4201]
Mon 14:15 – 15:45, HSZ, HS-2

Dr. Nils Goerke, Autonomous Intelligent Systems,
Department of Computer Science, University of Bonn

Artificial Life
Summer 2025

Evolutionary Algorithms 3

Thank you for your patience

Dr. Nils Goerke, Autonomous Intelligent Systems,
Department of Computer Science, University of Bonn