What is an Evolutionary Algorithm?

Lecture # 5

Esmaeil Nourani

GA Quick Overview

- Developed: USA in the 1970's
- Early names: J. Holland, K. DeJong, D. Goldberg
- Typically applied to:
  - discrete optimization
- Attributed features:
  - not too fast
  - good heuristic for combinatorial problems
- Special Features:
  - Traditionally emphasizes combining information from good parents (crossover)
  - many variants, e.g., reproduction models, operators

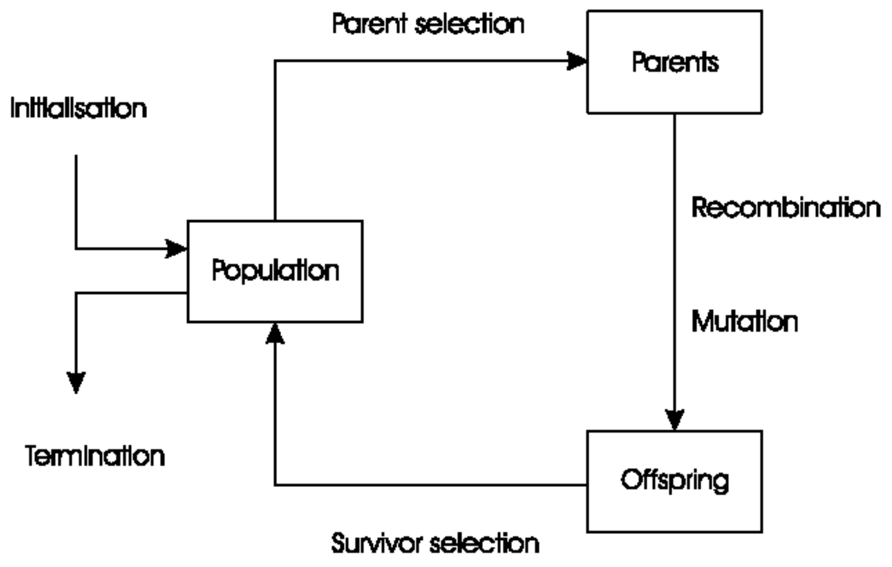
The Main Evolutionary Computing Metaphor

EVOLUTION		PROBLEM SOLVING
Environment	⟷	Problem
Individual	⟷	Candidate Solution
Fitness	⟷	Quality

Fitness $\rightarrow$ chances for survival and reproductio

Quality $\rightarrow$ chance for seeding new solutions

General Scheme of EAs



Pseudo-code for typical EA

```
BEGIN
  INITIALISE population with random candidate solutions;
  EVALUATE each candidate;
  REPEAT UNTIL ( TERMINATION CONDITION is satisfied ) DO
    1 SELECT parents;
    2 RECOMBINE pairs of parents;
    3 MUTATE the resulting offspring;
    4 EVALUATE new candidates;
    5 SELECT individuals for the next generation;
  OD
END
```

Key Concepts

- Representations
  - Candidate solutions (individuals)
- Evaluation (Fitness)   Function
  - Assigns a fitness value to each Candidate solutions which forms the basis for selection
  - Typically we talk about fitness being maximised
  - problems may be best posed as minimisation problems
- Population
  - Holds possible solutions

Parent Selection Mechanism

- Assigns variable probabilities of individuals acting as parents depending on their fitnesses
- Usually probabilistic
  - high quality solutions more likely to become parents than low quality but not guaranteed
  - even worst in current population usually has non-zero probability of becoming a parent
- This *stochastic* nature can aid escape from local optima

Variation Operators

- Role is to generate new candidate solutions
- Usually divided into two types according to their arity (number of inputs):
  - Arity 1 : mutation operators
  - Arity >1 : Recombination operators
  - Arity = 2 typically called crossover
- There has been much debate about relative importance of recombination and mutation
  - Nowadays most EAs use both
  - Choice of particular variation operators is representation dependant

Mutation

- Acts on one solution and delivers another
- Element of randomness is essential and differentiates it from other unary heuristic operators
- May guarantee connectedness of search space and hence convergence proofs

Recombination

- Merges information from parents into offspring
- Choice of what information to merge is stochastic
- Most offspring may be worse, or the same as the parents
- Hope is that some are better by combining elements of genotypes that lead to good traits

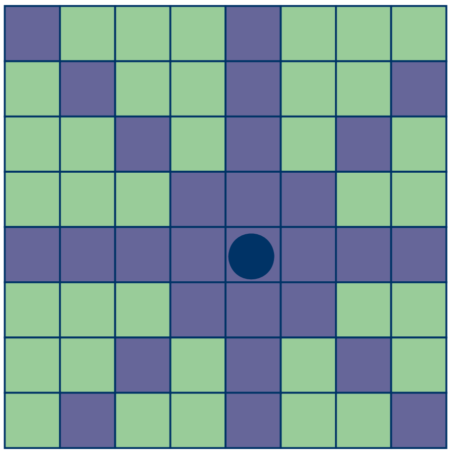
Survivor Selection

- ***replacement***
- Most EAs use fixed population size so need a way of going from (parents + offspring) to next generation
- Often deterministic
  - Fitness based : e.g., rank parents+offspring and take best
  - Age based: make as many offspring as parents and delete all parents
- Sometimes do combination (elitism)

Initialisation / Termination

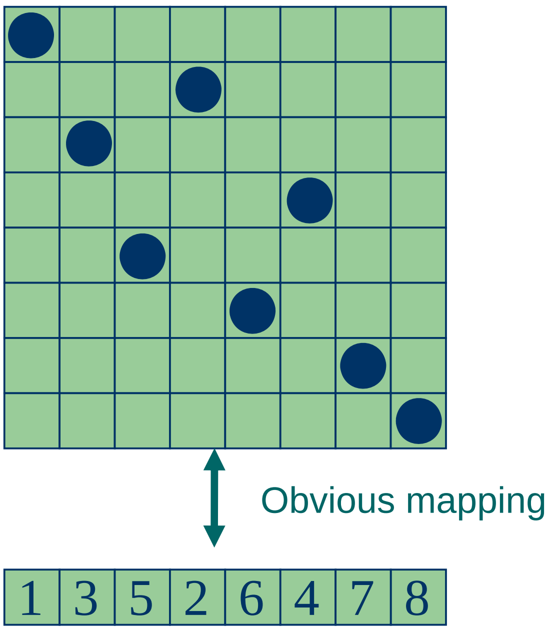
- Initialisation usually done at random,
- Termination condition checked every generation
  - Reaching some (known/hoped for) fitness
  - Reaching some maximum allowed number of generations
  - Reaching some minimum level of diversity
  - Reaching some specified number of generations without fitness improvement

Example: the 8 queens problem



Place 8 queens on an 8x8 chessboard in such a way that they cannot check each other

The 8 queens problem: representation



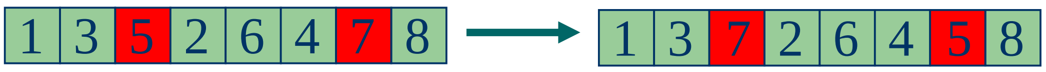
a permutation of the numbers 1 - 8

8 Queens Problem: Fitness evaluation

- Penalty of one queen:
  the number of queens she can check.
- Penalty of a configuration:
  the sum of the penalties of all queens.
- Note: penalty is to be minimized
- Fitness of a configuration:
  inverse penalty to be maximized

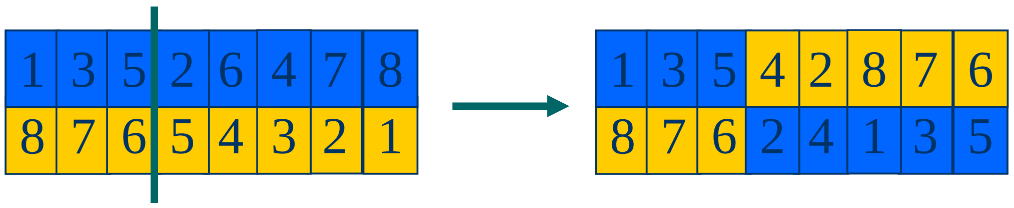
The 8 queens problem: Mutation

Small variation in one permutation, e.g.:
• swapping values of two randomly chosen positions,



The 8 queens problem: Recombination

Combining two permutations into two new permutations:
• choose random crossover point
• copy first parts into children
• create second part by inserting values from other parent:
  • in the order they appear there
  • beginning after crossover point
  • skipping values already in child



The 8 queens problem: Selection

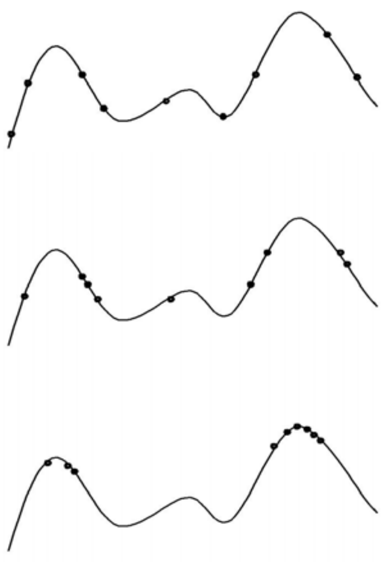
- Parent selection:
  - Pick 5 parents and take best two to undergo crossover
- Survivor selection (replacement)
  - When inserting a new child into the population, choose an existing member to replace by:
  - sorting the whole population by decreasing fitness
  - enumerating this list from high to low and replacing the first

8 Queens Problem: summary

Representation	Permutations
Recombination	"Cut-and-crossfill" crossover
Recombination probability	100%
Mutation	Swap
Mutation probability	80%
Parent selection	Best 2 out of random 5
Survival selection	Replace worst
Population size	100
Number of Offspring	2
Initialisation	Random
Termination condition	Solution or 10,000 fitness evaluation

Typical behaviour of an EA

Phases in optimising on a 1-dimensional fitness landscape



Early phase:

quasi-random population distribution

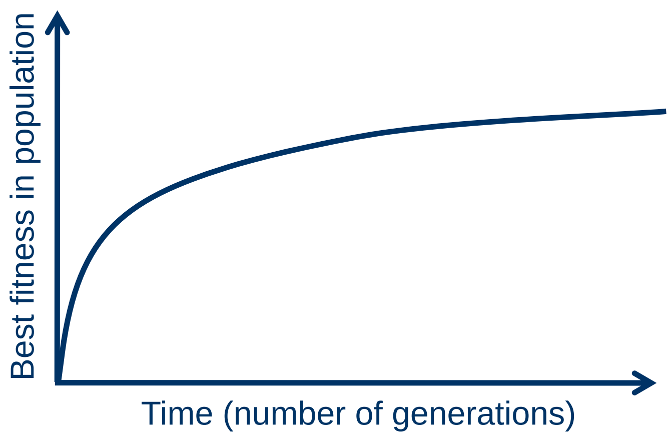
Mid-phase:

population arranged around/on hills

Late phase:

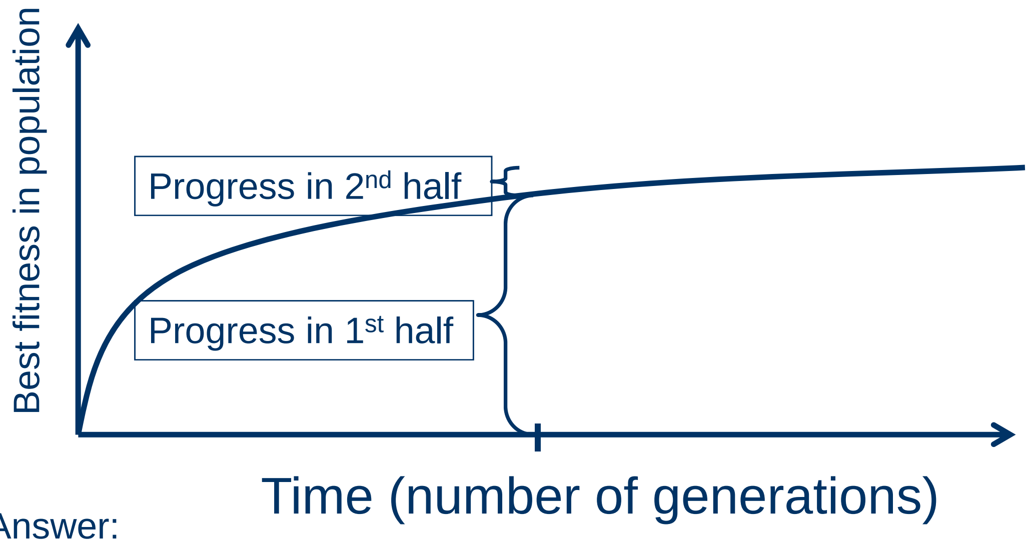
population concentrated on high hills

Typical run: progression of fitness



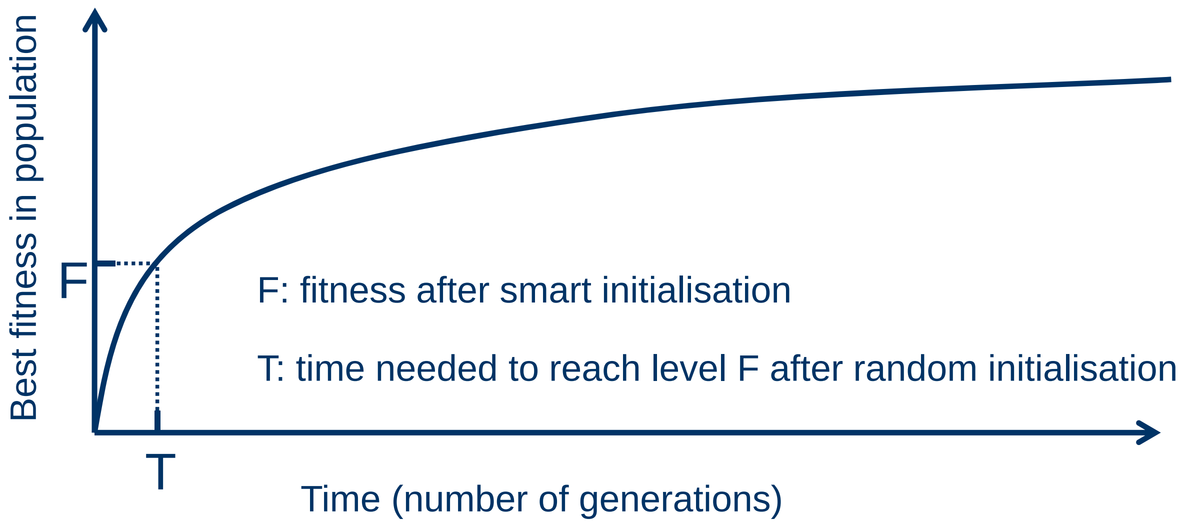
Typical run of an EA shows so-called “anytime behavior”

Are long runs beneficial?



- Answer:
  - it depends how much you want the last bit of progress
  - it may be better to do more shorter runs

Is it worth expending effort on smart initialisation?



- Answer : it depends:
  - possibly, if good solutions/methods exist.

Evolution strategies

ES quick overview

- Developed: Germany in the 1970's
- Early names: I. Rechenberg, H.-P. Schwefel
- Attributed features:
  - fast
  - good optimizer for real-valued optimisation

*E*volution Strategies (ES)

Among the simplest ES algorithms is the *(μ, λ) algorithm.*

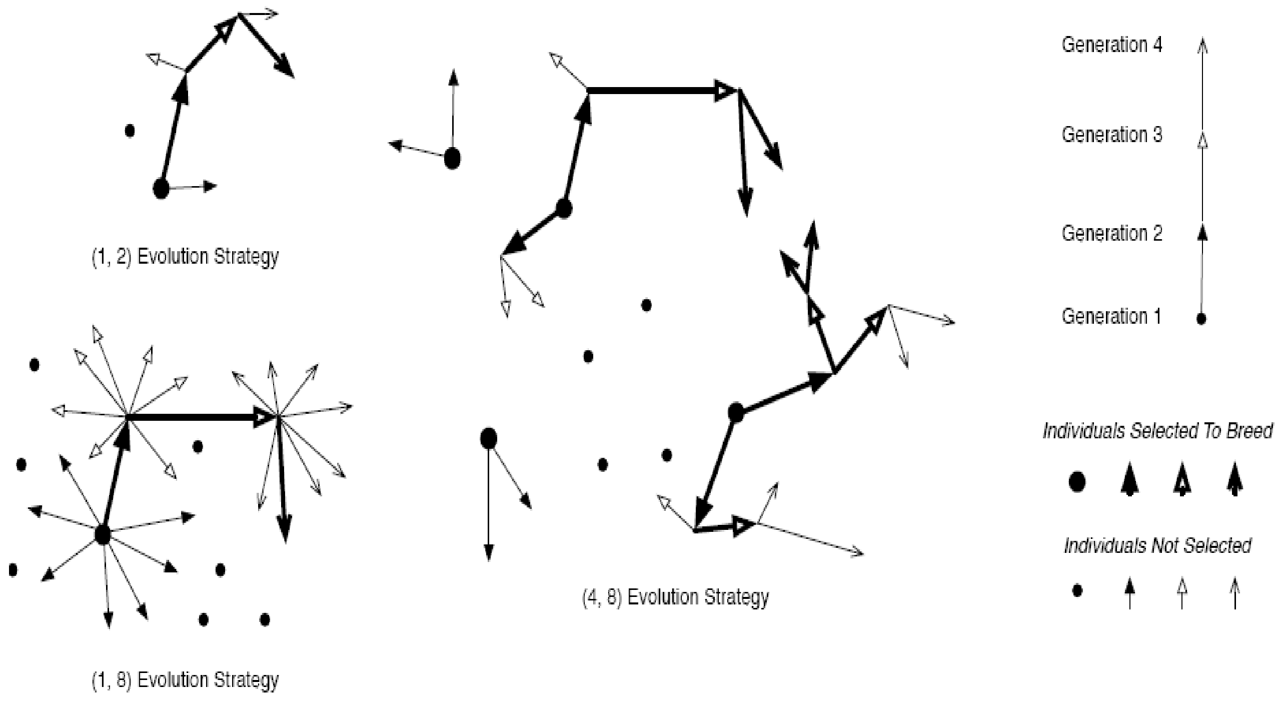
We begin with a population of λ (typically) *number of individuals, generated randomly. We then iterate as follows. First we assess* the fitness of all the individuals. Then we delete from the population all but the *μ fittest ones (this* is all there's to Truncation Selection)

(μ, λ) Evolution Strategies (ES)

- Each of the *μ fittest individuals gets to produce* **λ** */μ children* through an ordinary Mutation. All told we've created **λ** *new children. Our Join operation is simple:* the children just replace the parents, who are discarded. The iteration continues anew.

- In short, *μ is the number of parents which survive, and λ is the number of kids that the μ* parents make in total. Notice that **λ** *should be a multiple of μ. ES practitioners usually refer to* their algorithm by the choice of *μ and λ . For example, if μ = 5 and λ = 20, then we have a (5, 20)* Evolution Strategy.

(μ, λ) Evolution Strategies (ES)


(1, 2) Evolution Strategy
(1, 8) Evolution Strategy
(4, 8) Evolution Strategy
Generation 4
Generation 3
Generation 2
Generation 1
Individuals Selected To Breed
Individuals Not Selected

(μ, λ) Evolution Strategies (ES)

Algorithm 18 *The (μ, λ) Evolution Strategy*

```
$\mu \leftarrow$ number of parents selected
$\lambda \leftarrow$ number of children generated by the parents

$P \leftarrow \{\}$
for $\lambda$ times do                                          ▷ Build Initial Population
    $P \leftarrow P \cup \{$new random individual$\}$
$Best \leftarrow \square$
repeat
    for each individual $P_i \in P$ do
        AssessFitness($P_i$)
        if $Best = \square$ or Fitness($P_i$) $>$ Fitness($Best$) then
            $Best \leftarrow P_i$
    $Q \leftarrow$ the $\mu$ individuals in $P$ whose Fitness( ) are greatest    ▷ Truncation Selection
    $P \leftarrow \{\}$                        ▷ Join is done by just replacing $P$ with the children
    for each individual $Q_j \in Q$ do
        for $\lambda / \mu$ times do
            $P \leftarrow P \cup \{$Mutate(Copy($Q_j$))$\}$
until $Best$ is the ideal solution or we have run out of time
return $Best$
```

exploration versus exploitation

exploration versus exploitation.

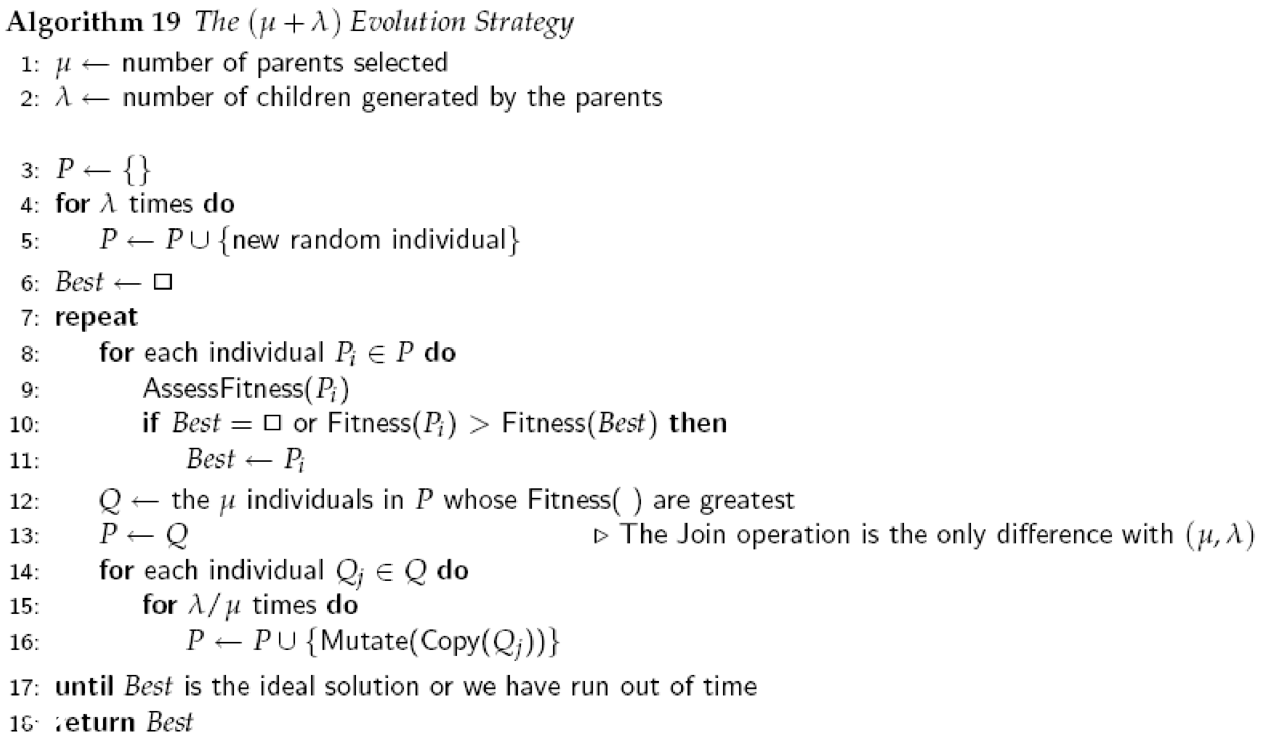
- The size of $\boldsymbol{\lambda}$. *This essentially controls the sample size for each population, and is basically* the same thing as the *n variable in Steepest-Ascent Hill Climbing With Replacement. At the* extreme, as *$\boldsymbol{\lambda}$ approaches* ***infinite****, the algorithm approaches exploration (random search).*

- The size of μ*. This controls how selective the algorithm is; low values of μ with respect to $\boldsymbol{\lambda}$* push the algorithm more towards exploitative search as only the best individuals survive.

Evolution Strategy ($\mu + \lambda$)

- The second Evolution Strategy algorithm is called ($\mu + \lambda$). *It differs from (μ, λ) in only one* respect: the Join operation. Recall that in (μ, λ) *the parents are simply replaced with the children in* the next generation. But in ($\mu + \lambda$), *the next generation consists of the μ parents plus the λ new* children. That is, the parents compete with the kids. Thus the next and all successive generations are $\mu + \lambda$ *in size.*

Evolution Strategy ($\mu + \lambda$)



Algorithm 19 *The ($\mu + \lambda$) Evolution Strategy*

```
$\mu \leftarrow$ number of parents selected
$\lambda \leftarrow$ number of children generated by the parents

$P \leftarrow \{\}$
for $\lambda$ times do
    $P \leftarrow P \cup \{$new random individual$\}$
$Best \leftarrow \square$
repeat
    for each individual $P_i \in P$ do
        AssessFitness($P_i$)
        if $Best = \square$ or Fitness($P_i$) $>$ Fitness($Best$) then
            $Best \leftarrow P_i$
    $Q \leftarrow$ the $\mu$ individuals in $P$ whose Fitness( ) are greatest
    $P \leftarrow Q$                    ▷ The Join operation is the only difference with $(\mu, \lambda)$
    for each individual $Q_j \in Q$ do
        for $\lambda / \mu$ times do
            $P \leftarrow P \cup \{$Mutate(Copy($Q_j$))$\}$
until $Best$ is the ideal solution or we have run out of time
return $Best$
```

Evolution Strategy ($\mu + \lambda$)

- Generally speaking, *($\mu + \lambda$) may be more exploitative than (μ, λ) because high-fit parents* persist to compete with the children. This has risks: a sufficiently fit parent may defeat other population members over and over again, eventually causing the entire population to prematurely converge to immediate descendants of that parent, at which point the whole population has been trapped in the local optimum surrounding the parent.

Ref

- Slides adapted from Advanced Algorithms course, presented by Dr. kourosh ziarati