# Evolutionary Computing

Chapter 3 – What is an Evolutionary Algorithm?

# Common model of evolutionary processes

- Population of individuals
- Individuals have a fitness
- Reproduction / variation operators
  - mutation
  - recombination (a.k.a. crossover)
- Selection towards higher fitness
  - “survival of the fittest” and
  - “mating of the fittest”
- The fitness of the population increases over time

# Two pillars of evolution

There are two competing forces

**Increasing** population **diversity** by variation

- mutation
- recombination

Push towards **novelty**

**Decreasing** population **diversity** by selection

- of parents
- of survivors

Push towards **quality**

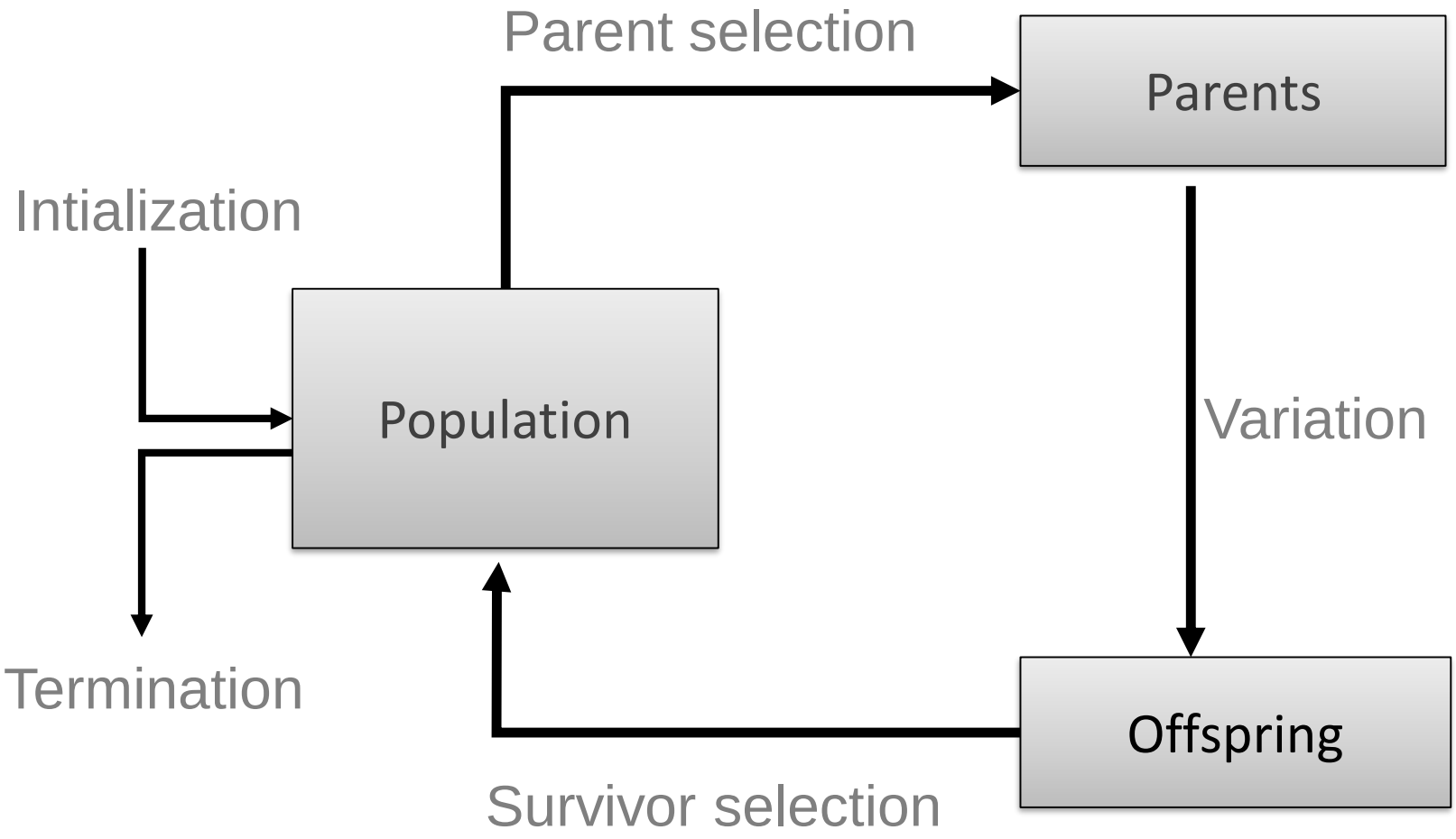
- Selection operators act on population level
- Variation operators act on individual level

# What is an Evolutionary Algorithm?

- Scheme of an EA
- Main EA components:
  - Representation / evaluation / population
  - Parent selection / survivor selection
  - Mutation / recombination
- Excercise: eight-queens problem
- EA behaviour
- EAs in context

# General scheme of EAs

arrows are the operations



# EA scheme in pseudo-code

```
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
```

# Main EA components: Representation

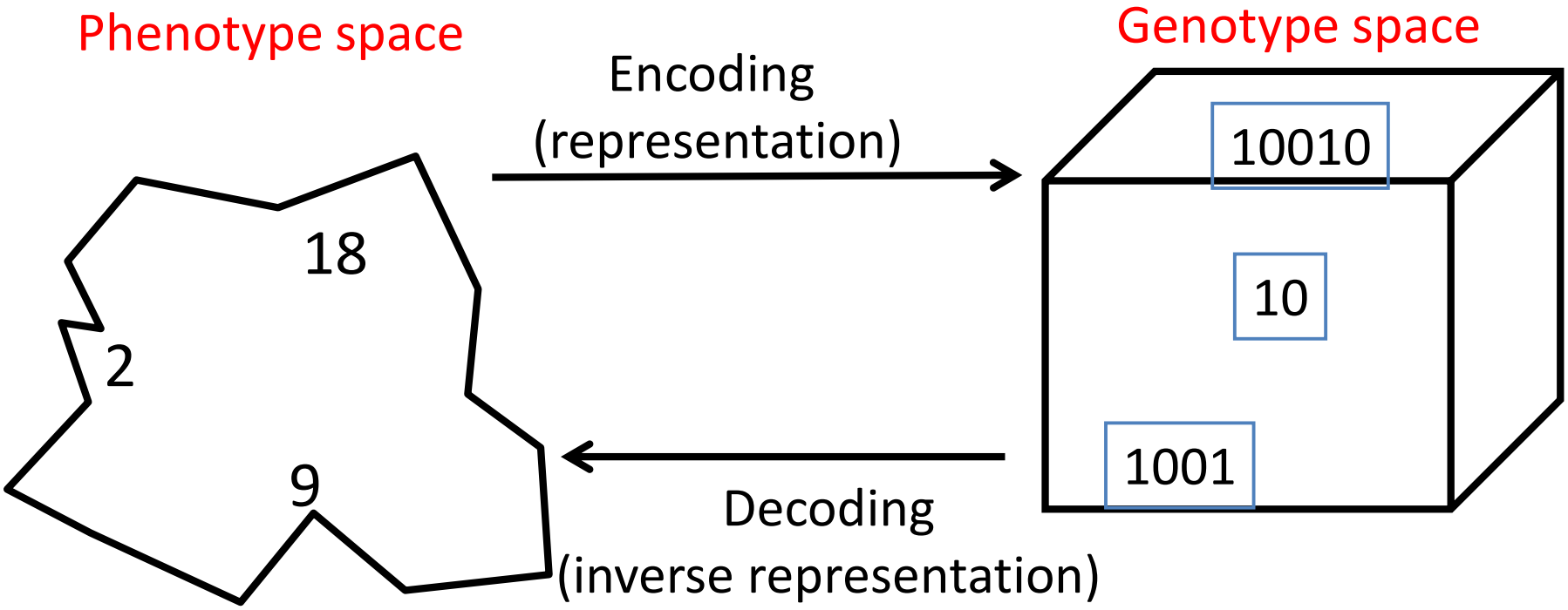
- Role: provides code for candidate solutions that can be manipulated by variation operators
- Leads to two levels of existence
  - phenotype: object in original problem context, the outside
  - genotype: code to denote that object, the inside (chromosome, "digital DNA")
- Implies two mappings:
  - Encoding : phenotype → genotype (not necessarily one to one)
  - Decoding : genotype → phenotype (must be one to one)
- Chromosomes contain genes, which are in (usually fixed) positions called loci (sing. locus) and have a value (allele)

# Some terminology

- Variable vs. value
- Gene vs. allele
- Example:
  - x vs. 3.14
  - color vs. red

# Main EA components: Representation

Example: represent integer values by their binary code




In order to find the global optimum, every feasible solution must be represented in genotype space

# Main EA components: Evaluation / fitness function

Role:

- Represents the task to solve, the requirements to adapt to (can be seen as "the environment")
- Enables selection (provides basis for comparison)
- e.g., some phenotypic traits are advantageous, desirable, e.g. big ears cool better, these traits are rewarded by more offspring that will expectedly carry the same trait

- A.k.a. *quality* function or *objective* function
- Assigns a single real-valued fitness to each phenotype which forms the basis for selection
  - So the more discrimination (different values) the better
- Typically we talk about fitness being maximised
  - Some problems may be best posed as minimisation problems, but conversion is trivial

# Main EA components: Population

- Role: holds the candidate solutions of the problem as individuals (genotypes)
- Formally, a population is a multiset of individuals, i.e. the same element might occur multiple times
- **Population is the basic unit of evolution, i.e., the population is evolving, not the individuals**
- Selection operators act on population level
- Variation operators act on individual level

# Main EA components: Population

- Some sophisticated EAs also assert a spatial structure on the population e.g., a grid
- Selection operators usually take whole population into account i.e., reproductive probabilities are *relative* to current generation
- Diversity of a population refers to the number of different fitness values / phenotypes / genotypes present (note: not the same thing)

# Main EA components: Selection

Role:

- Identifies individuals
  - to become parents
  - to survive
- Pushes population towards higher fitness
- Usually probabilistic / stochastic
  - high quality solutions more likely to be selected than low quality
  - but not guaranteed
  - even worst in current population usually has non-zero probability of being selected
- This *stochastic* nature can help escape from local optima

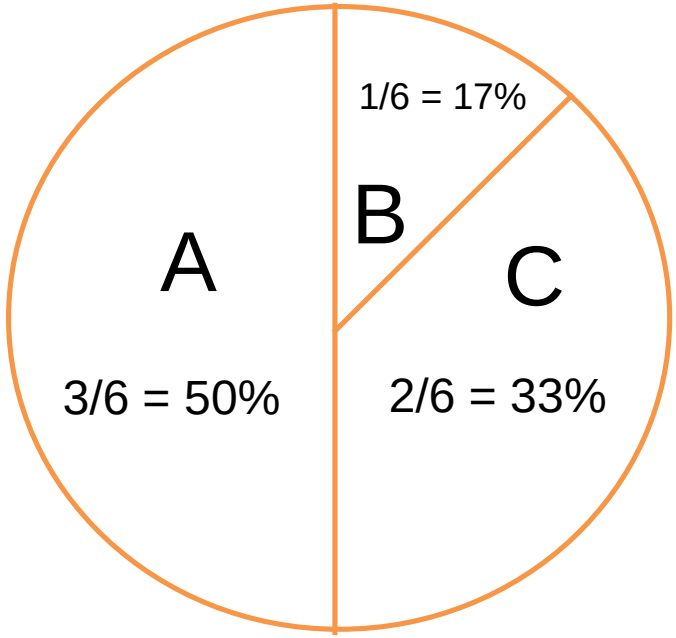
# Main EA components: Selection

Example: roulette wheel selection

fitness(A) = 3

fitness(B) = 1

fitness(C) = 2




In principle, any selection mechanism can be used for parent selection as well as for survivor selection

# Main EA components: Selection

- Survivor selection a.k.a. ***replacement***
- Most EAs use fixed population size so need a way of going from parents + offspring to next generation
- Often deterministic (while parent selection is usually stochastic)
  - Fitness based : e.g., rank parents + offspring and take best
  - Age based: make as many offspring as parents and delete all parents
- Sometimes a combination of stochastic and deterministic, e.g., elitism (the best *n* individuals always survive) is a deterministic rule

# Main EA components: Variation operators

- Role: 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
  - Arity > 2 multi-parent reproduction, is possible, seldom used in EC
- There has been much debate about relative importance of recombination and mutation
  - Nowadays most EAs use both – pragmatic attitude is advisable
  - **Variation operators must match the given representation**

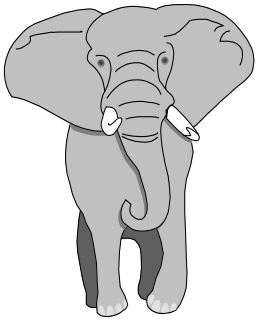
# Main EA components: Mutation

- Role: causes small, random variations
- Acts on one genotype and delivers another
- Element of randomness is essential and differentiates it from other heuristic operators
- Importance ascribed depends on representation and historical dialect:
  - Binary GAs – background operator responsible for preserving and introducing diversity
  - EP – only search operator
  - GP – hardly used
- May guarantee connectedness of search space and hence convergence proofs (see later in Theory chapter)

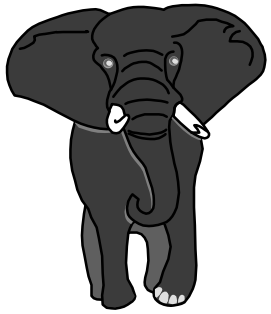
# Main EA components: Mutation

GENOTYPE PHENOTYPE

**before** 1 1 1 1 1 1 1



**after** 1 1 1 0 1 1 1



# Main EA components: Recombination

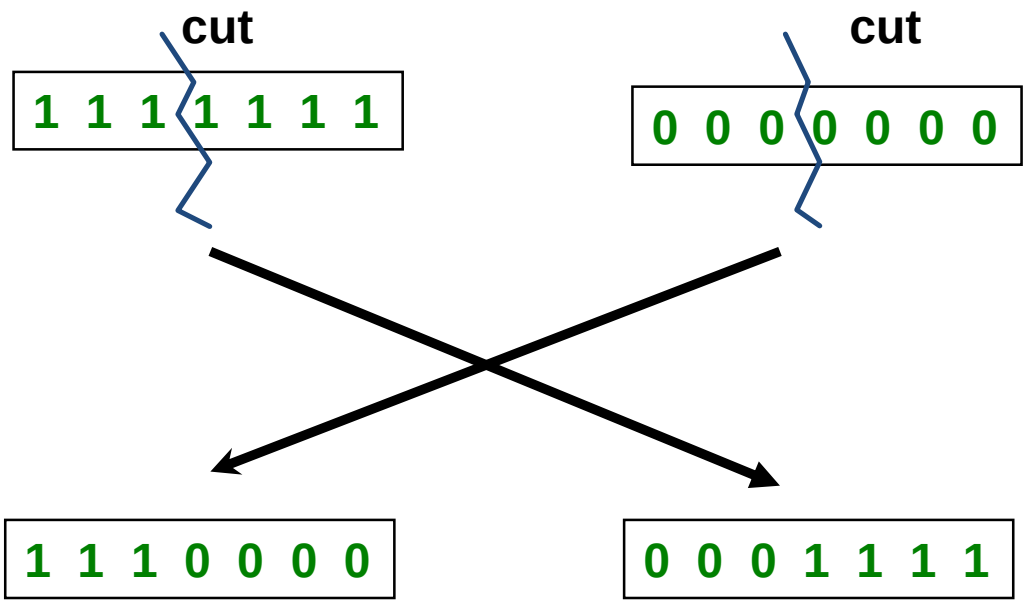
- Role: merges information from parents into offspring
- Choice of what information to merge is stochastic
- Most offspring may be worse, or the same as the parents
- Hope is that some are better by combining elements of genotypes that lead to good traits
- Principle has been used for millennia by breeders of plants and livestock

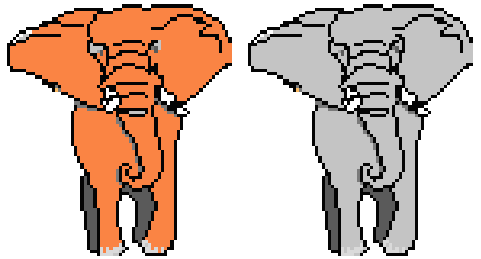
# Main EA components: Recombination

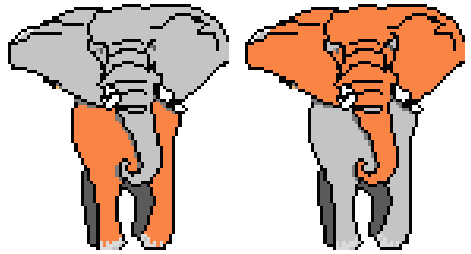
GENOTYPE

PHENOTYPE

**Parents**





**Offspring**

# Main EA components: Initialisation / Termination

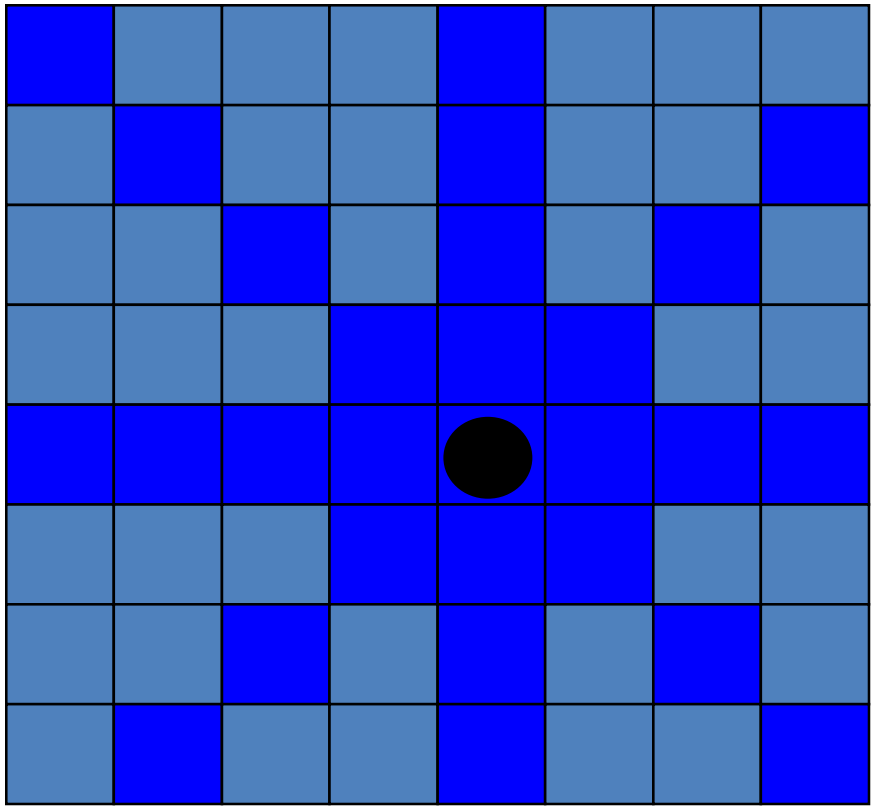
- Initialisation usually done at random,
  - Need to ensure even spread and mixture of possible alleles (values)
  - Can include existing solutions, or use problem-specific heuristics, to "seed" the population
- Termination condition checked every generation
  - Reaching some (known/hoped for) fitness level
  - Reaching some maximum number of generations
  - Reaching some minimum level of diversity
  - Reaching some specified number of generations without fitness improvement (stagnation)

# Different types of EAs

- Historically different flavours of EAs have been associated with different data types to represent solutions
  - Binary strings : Genetic Algorithms
  - Real-valued vectors : Evolution Strategies
  - Finite State Machines: Evolutionary Programming
  - LISP trees: Genetic Programming
- These historical differences are largely irrelevant, best strategy
  - choose representation to suit problem
  - choose variation operators to suit representation
- Selection operators only use fitness and so are independent of representation

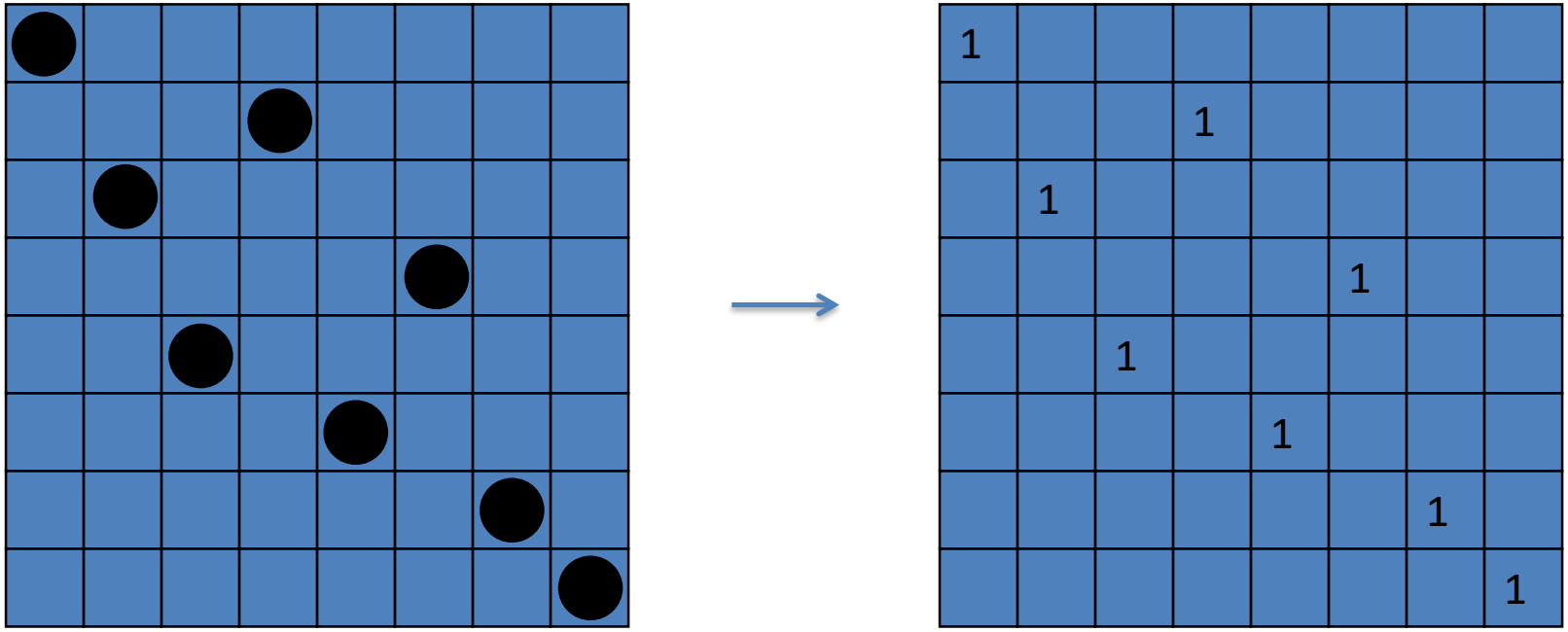
# Exercise:
# solving the 8-queens problem

# The 8-queens problem

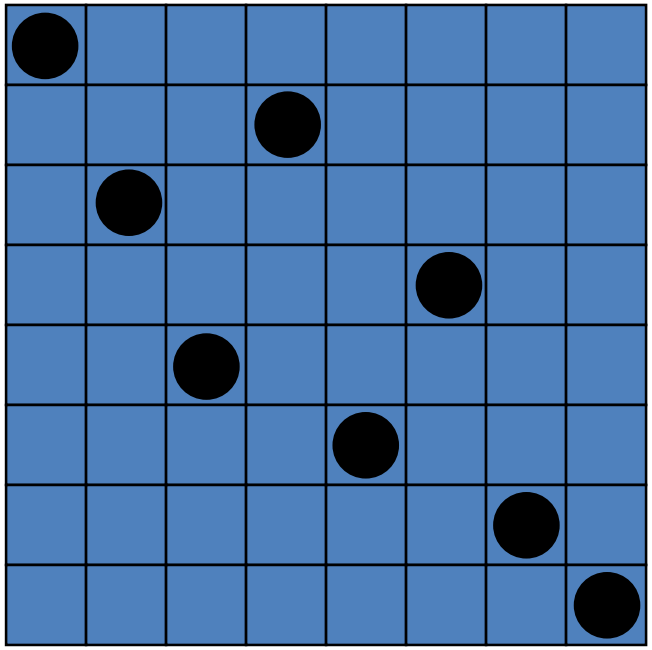


Place 8 queens on an 8x8 chessboard in such a way that they cannot check each other.

# The 8-queens problem: Representation



# The 8-queens problem: Representation



- Table of 8 x 8
- Bit-string of length 64

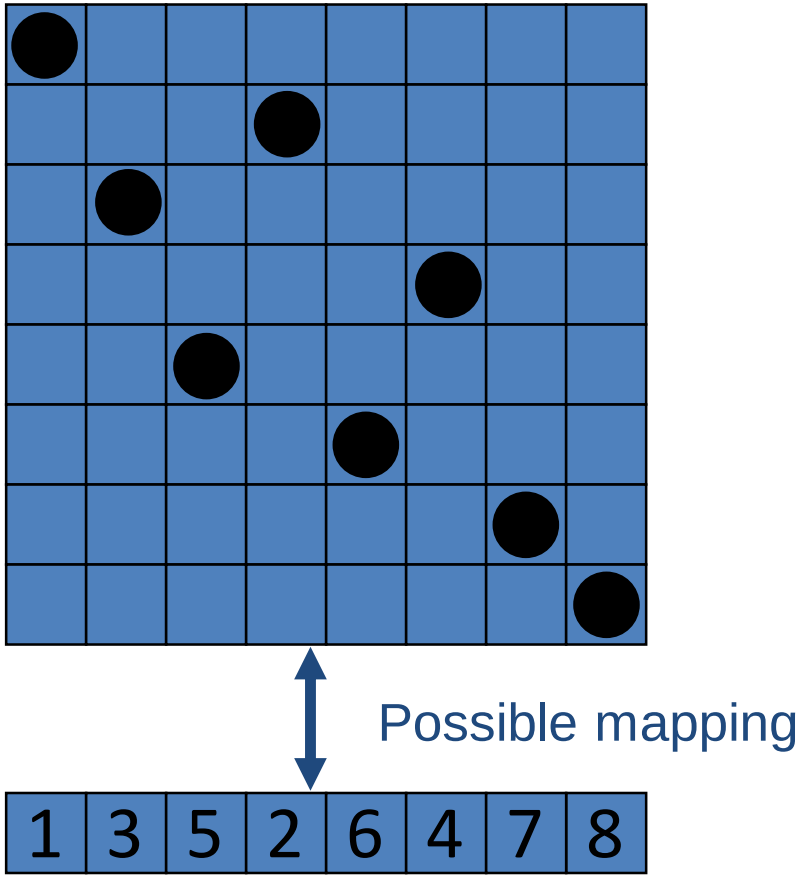
But:

- More than 8 queens is an infeasible solution
- You already know that there can't be more than two queens in one row or column

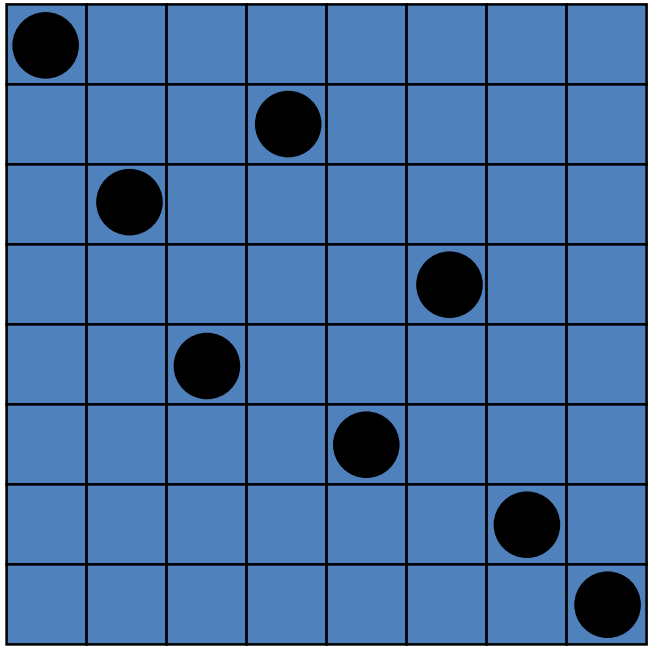
# The 8-queens problem: Representation

Phenotype:
a board configuration

Genotype:
a permutation of
the numbers 1–8



# The 8-queens problem: Representation



| 1 | 3 | 5 | 2 | 6 | 4 | 7 | 8 |
|---|---|---|---|---|---|---|---|

- Reduction of the dimension of the search space from 64 to 8
- Removal of many poor solutions

# The 8-queens problem: Representation

Different variation operators are suitable for different representations.

# The 8-queens problem: Fitness evaluation

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

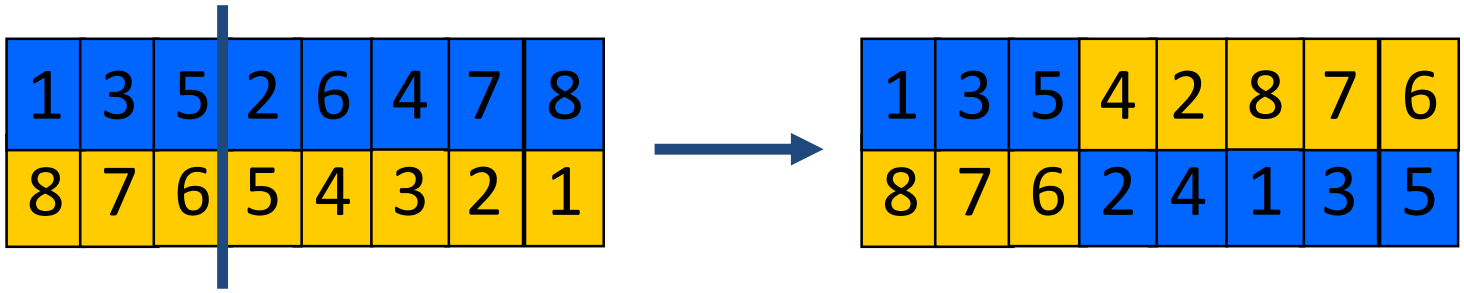
# The 8-queens problem: Mutation

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 randomly and take best two to undergo crossover
- Survivor selection (replacement)
  - When inserting a new child into the population, choose an existing member to replace, e.g,
  - the worst individual
  - the best individual that is worse than this child:
    - sorting the whole population by decreasing fitness
    - enumerating this list from high to low
    - replacing the first with a fitness lower than the given child

# The 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 |

Note that this is ***only one possible*** set of choices of operators and parameter values.

# Typical EA behaviour: Stages

Stages in optimising on a 1-dimensional fitness landscape

Early stage:

quasi-random population distribution

Mid-stage:

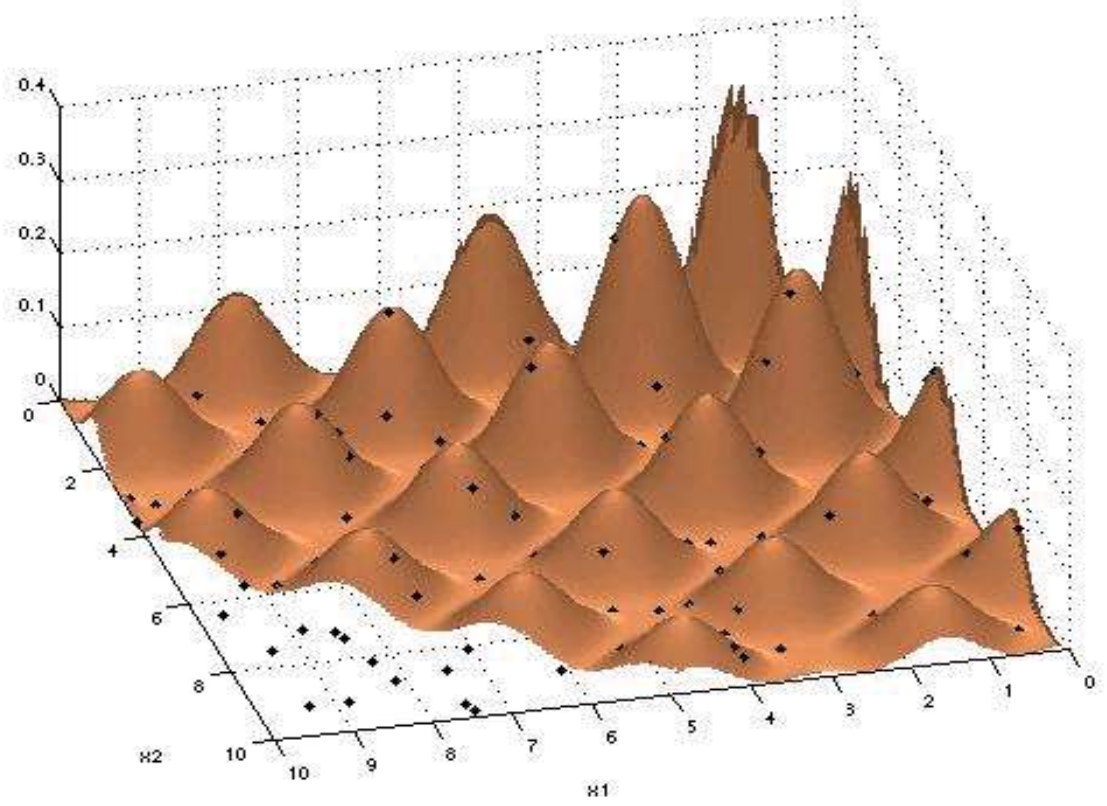
population arranged around/on hills

Late stage:

population concentrated on high hills

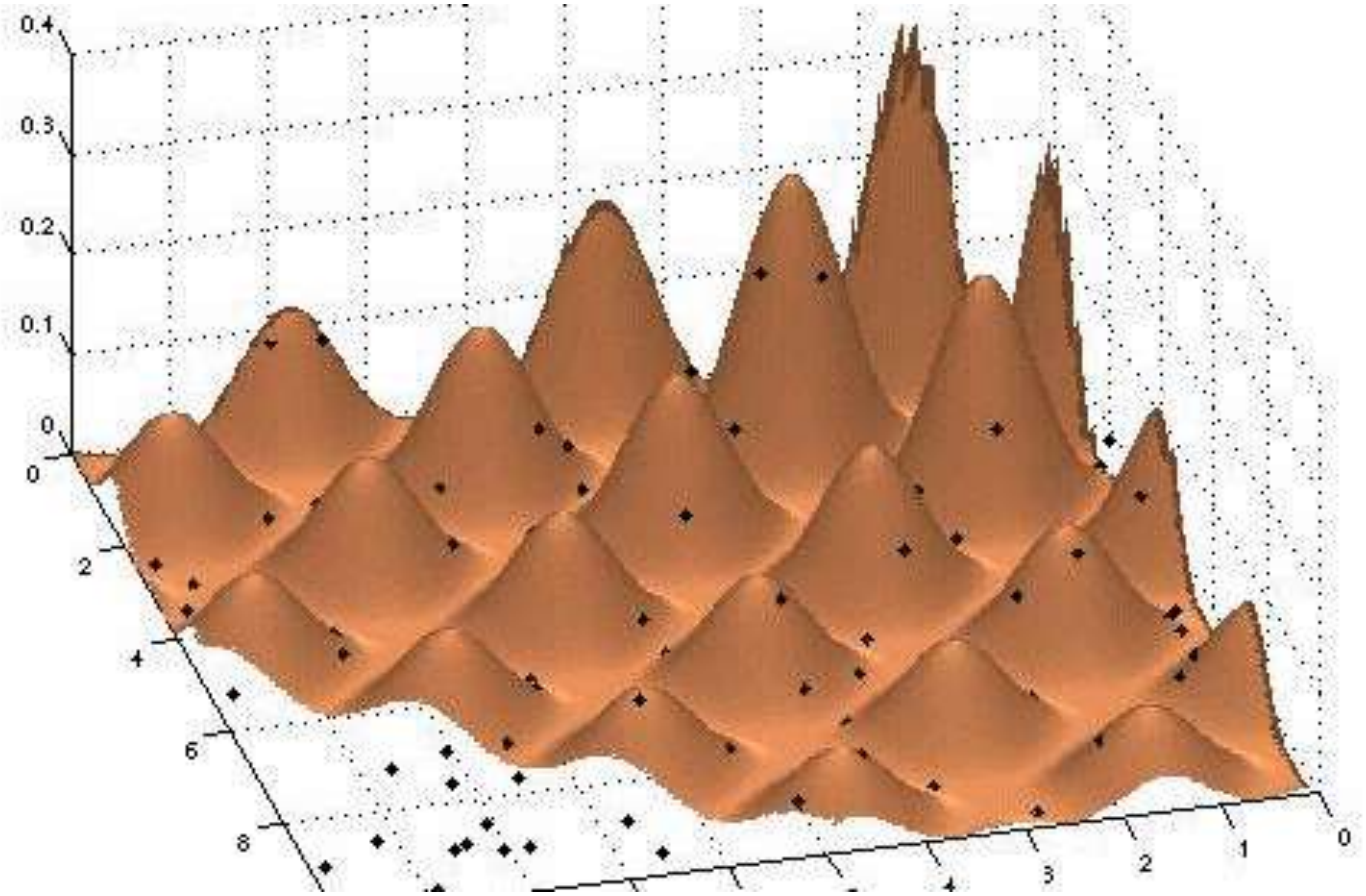
# Working of an EA demo

Searching a fitness landscape without “niching”

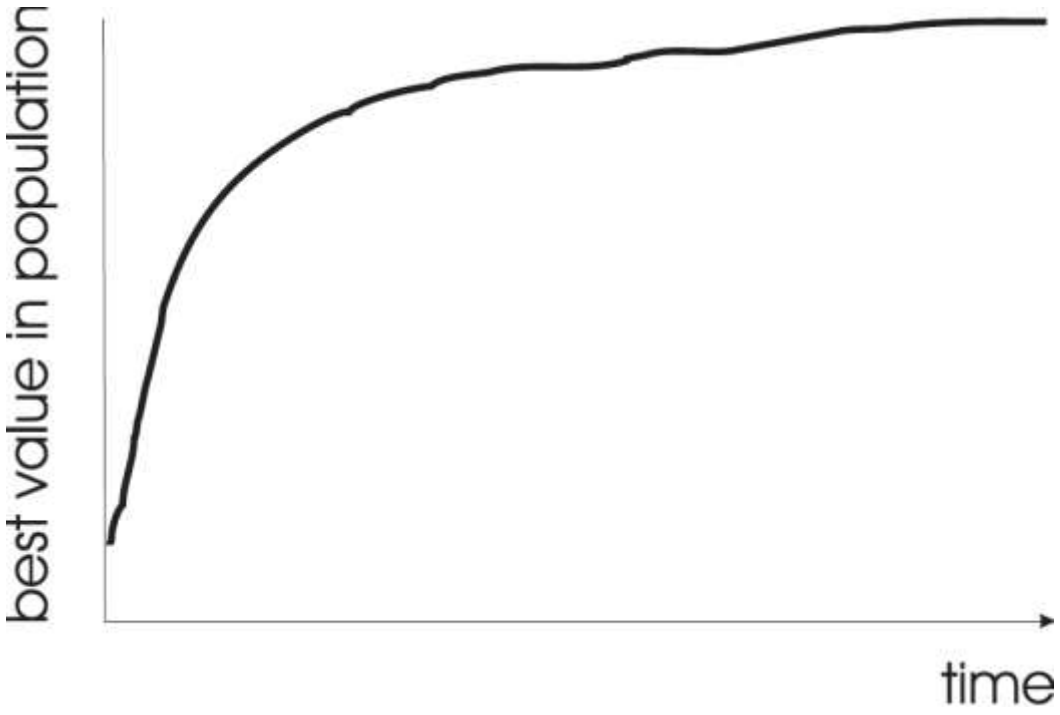


# Working of an EA demo

Searching a fitness landscape with “niching”

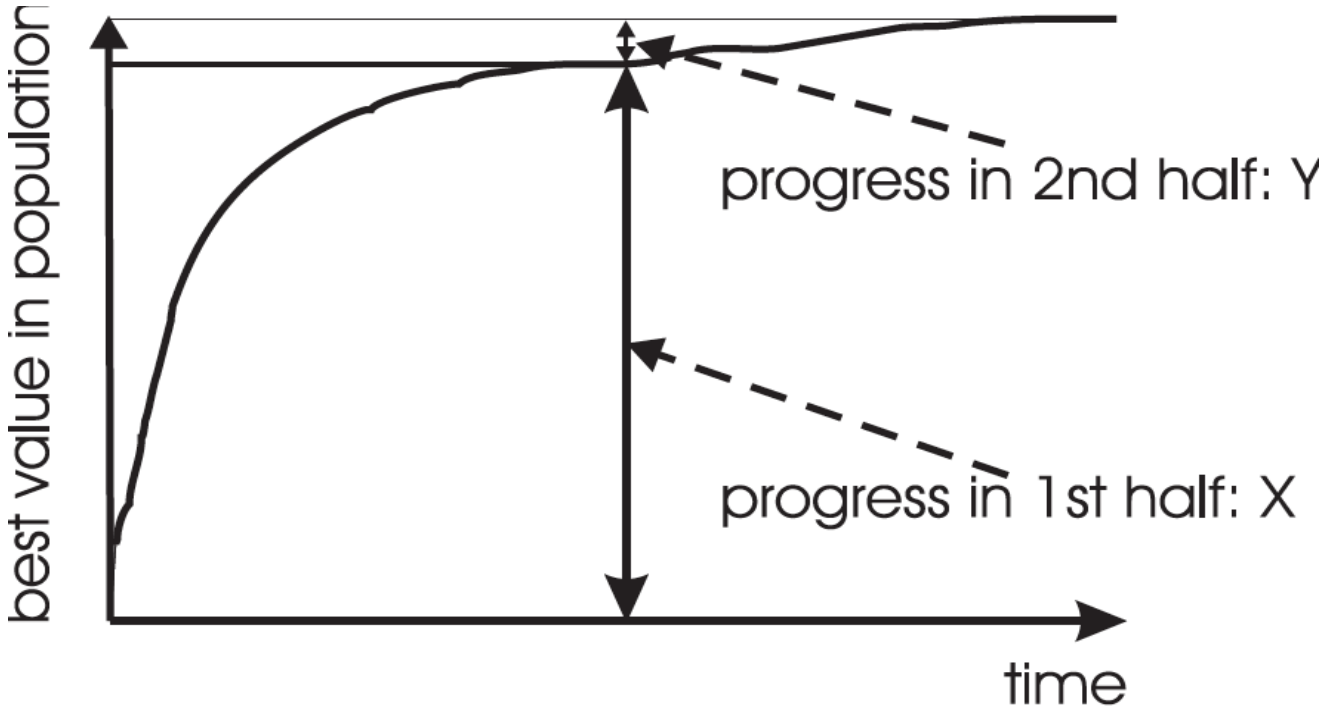


# Typical EA behaviour: progression of fitness



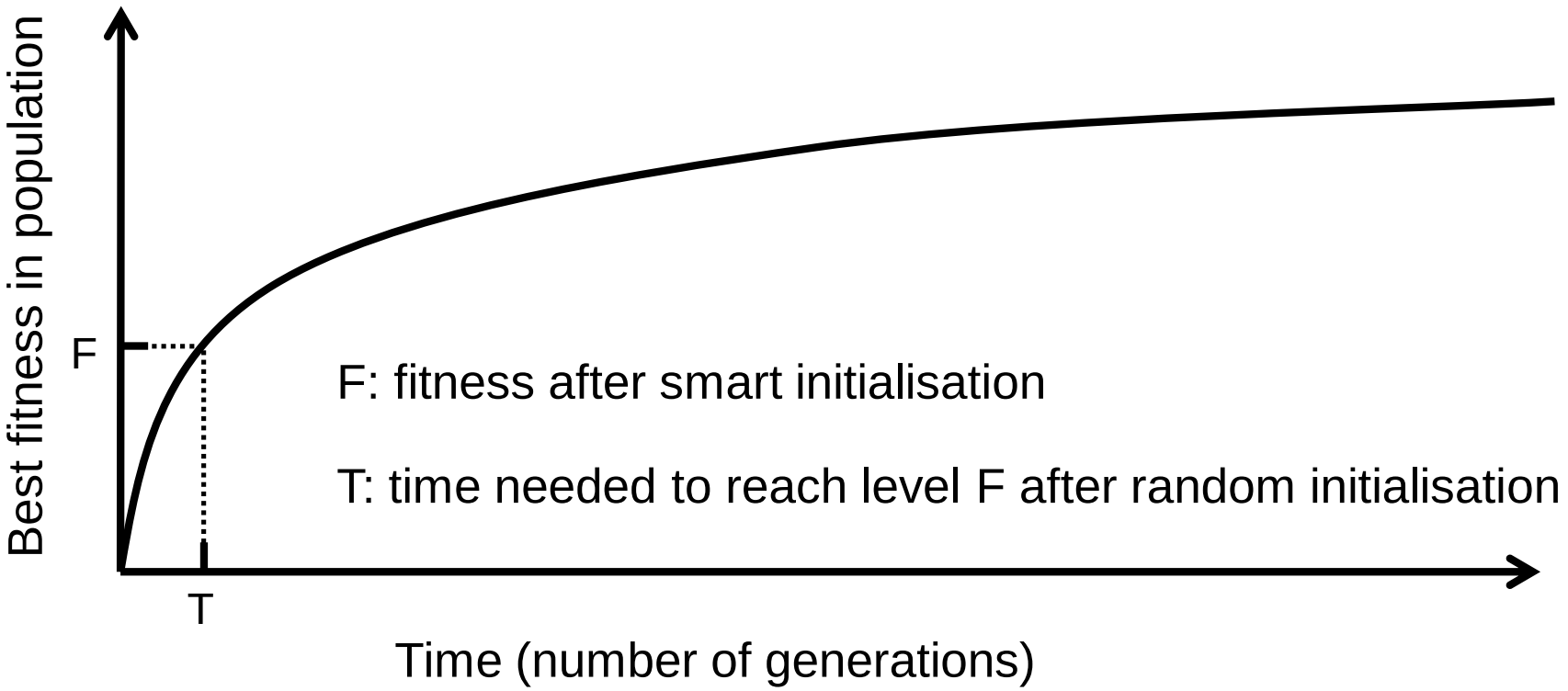

Typical run of an EA shows so-called “anytime behavior”

# Are long runs beneficial?




- Answer:
  - It depends on how much you want the last bit of progress
  - May be better to do more short runs

# Is it worth expending effort on smart initialisation?

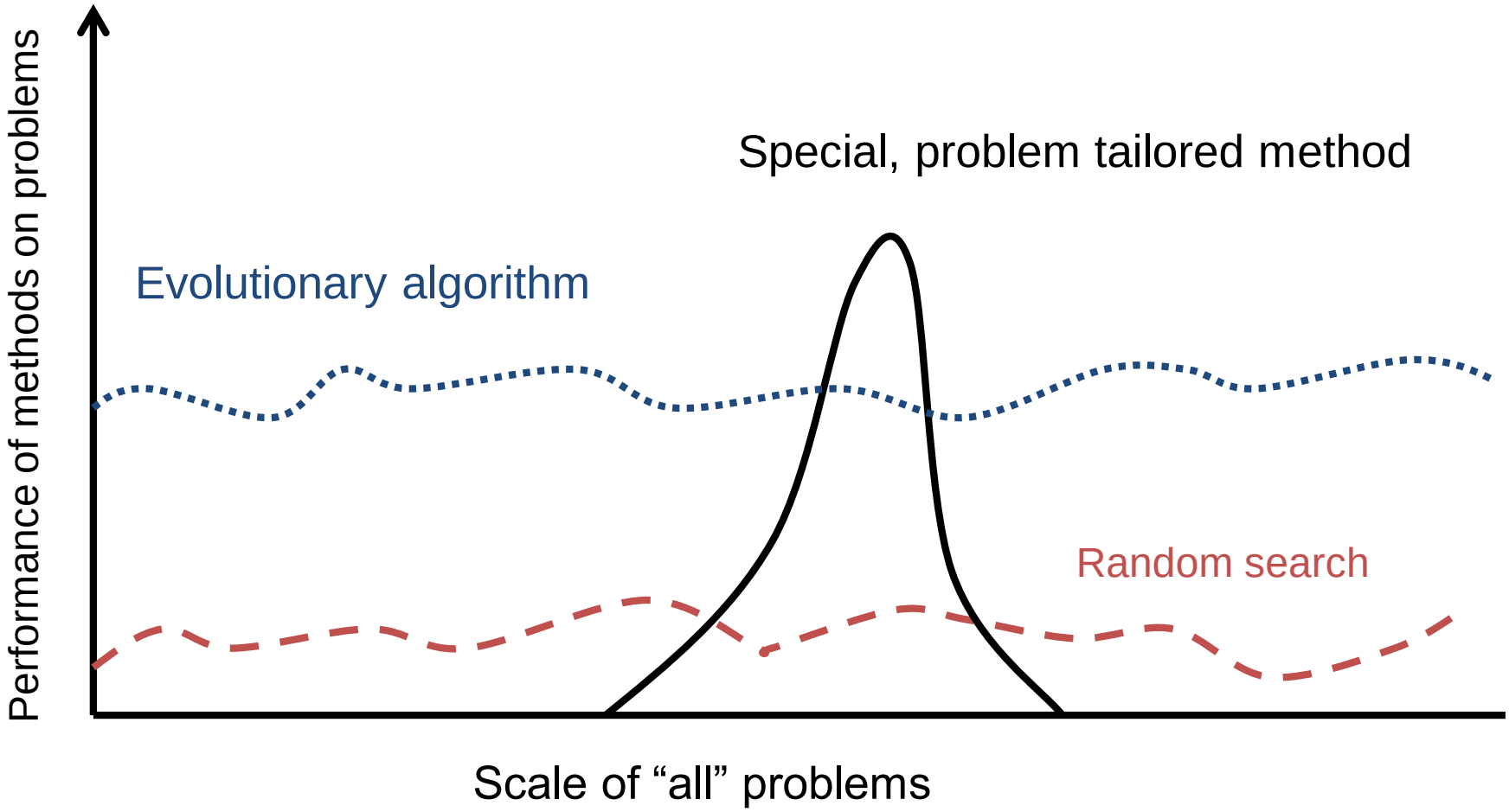



- Answer: it depends.
  - Possibly good, if good solutions/methods exist.
  - May not be necessary, EA will catch up quickly
  - Care is needed, see chapter/lecture on hybridisation.

# Evolutionary Algorithms in context

- EAs fall into the category of generate-and-test algorithms, a.k.a. trial-and-error algorithms
- They are stochastic, population-based search algorithms
- For most problems a problem-specific tool may:
  - perform better than a generic search algorithm on most instances,
  - have limited utility,
  - not do well on all instances
- EAs are meant to provide robust tool that shows:
  - evenly good performance
  - over a range of problems and instances

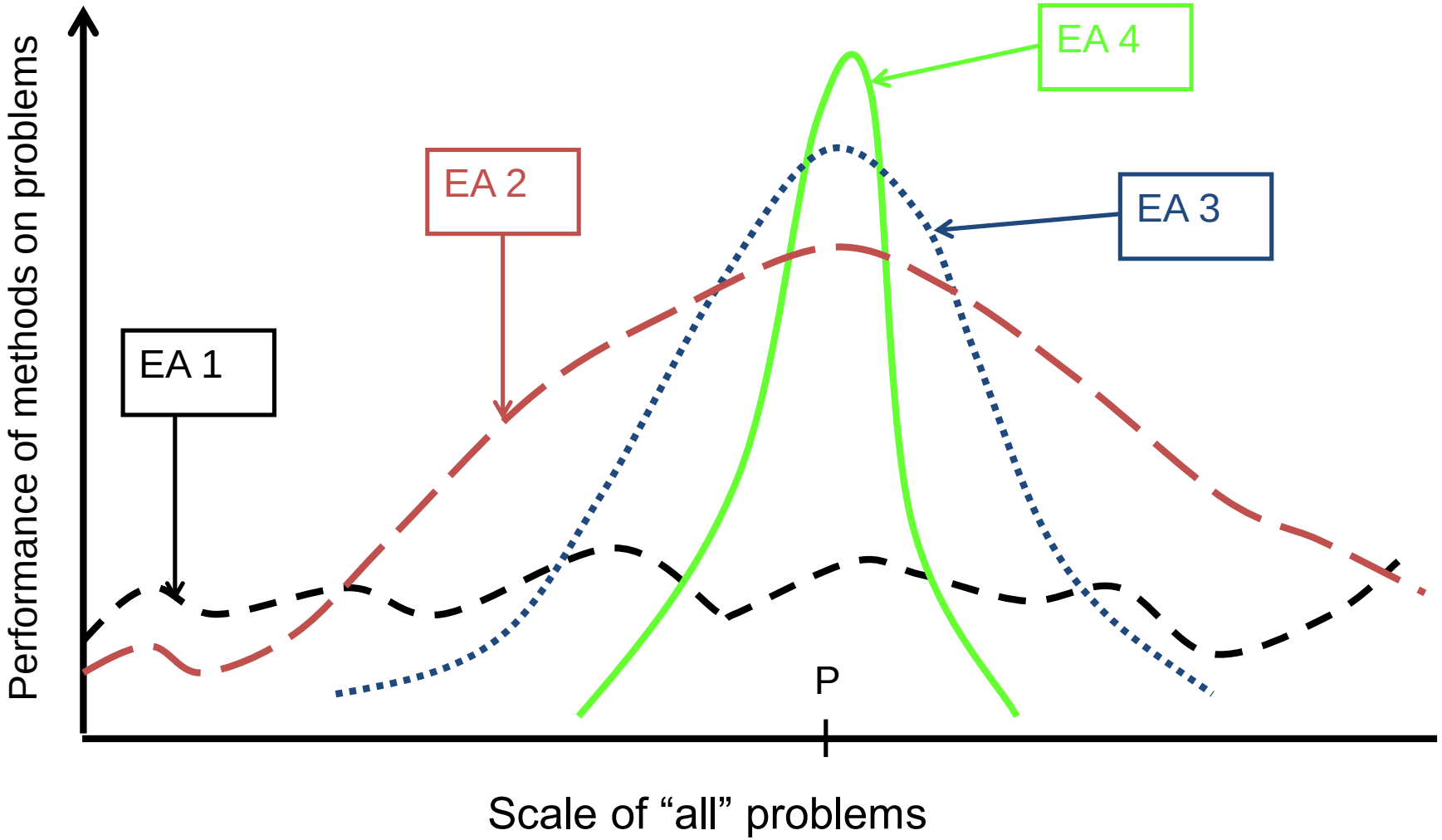
# EAs as problem solvers: Goldberg view (1989)



# EAs and domain knowledge

- Trend in the 90's: adding problem specific knowledge to EAs
  - (special variation operators, repair, etc)
- Result: EA performance curve "deformation":
  - better on problems of the given type
  - worse on problems different from given type
  - amount of added knowledge is variable
- Theory suggests the search for an "all-purpose" algorithm may be fruitless

# EAs as problem solvers: Michalewicz view (1996)



# EC and global optimisation

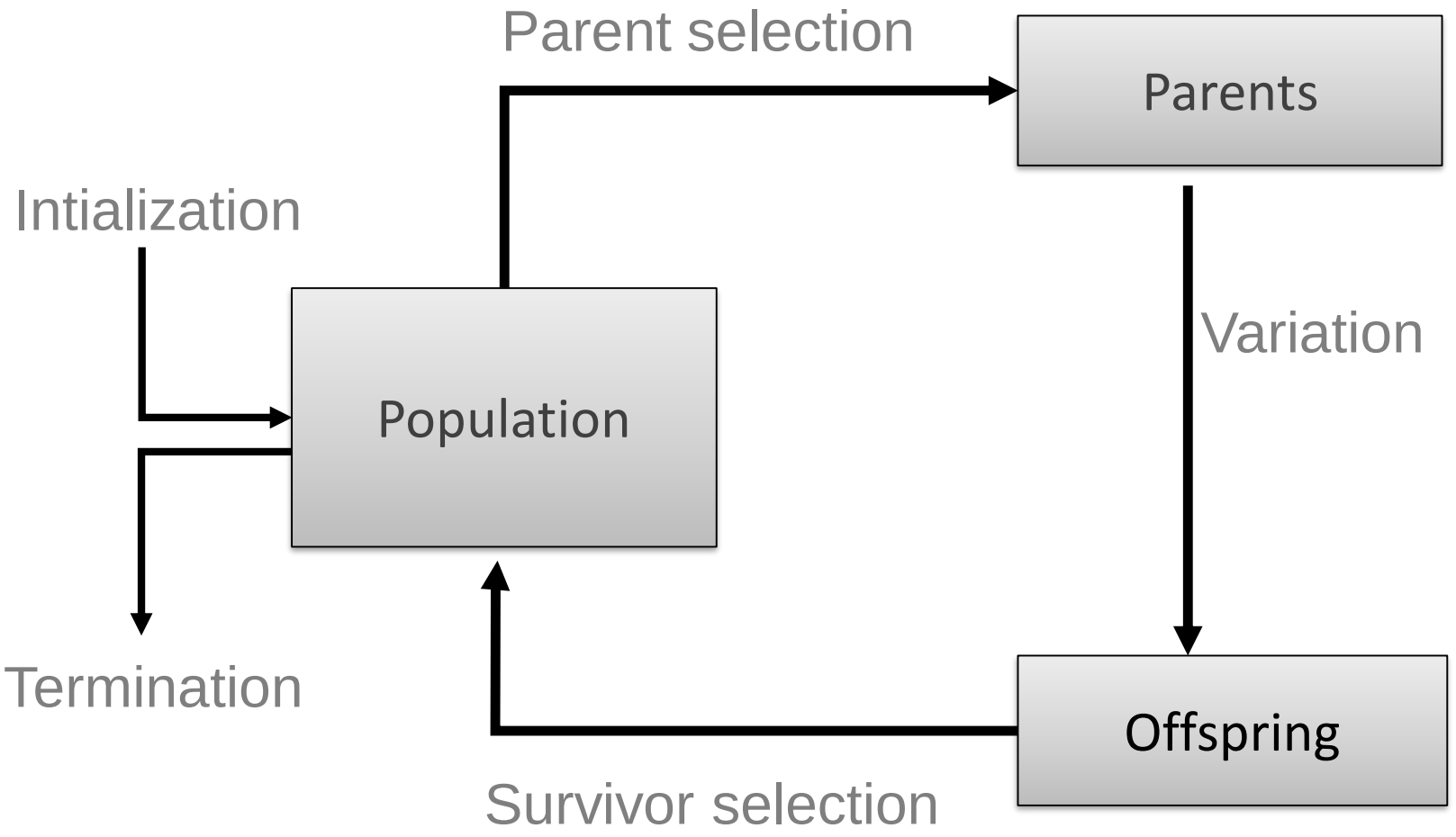
- Global optimisation: search for finding best solution $x^*$ out of some fixed set $S$
- Deterministic approaches
  - e.g. box decomposition (branch and bound etc)
  - Guarantee to find $x^*$ ,
  - May have bounds on runtime, usually super-polynomial
- Heuristic approaches (generate and test)
  - rules for deciding which $x \in S$ to generate next
  - no guarantees that best solutions found are globally optimal
  - no bounds on runtime

# EC and neighbourhood search

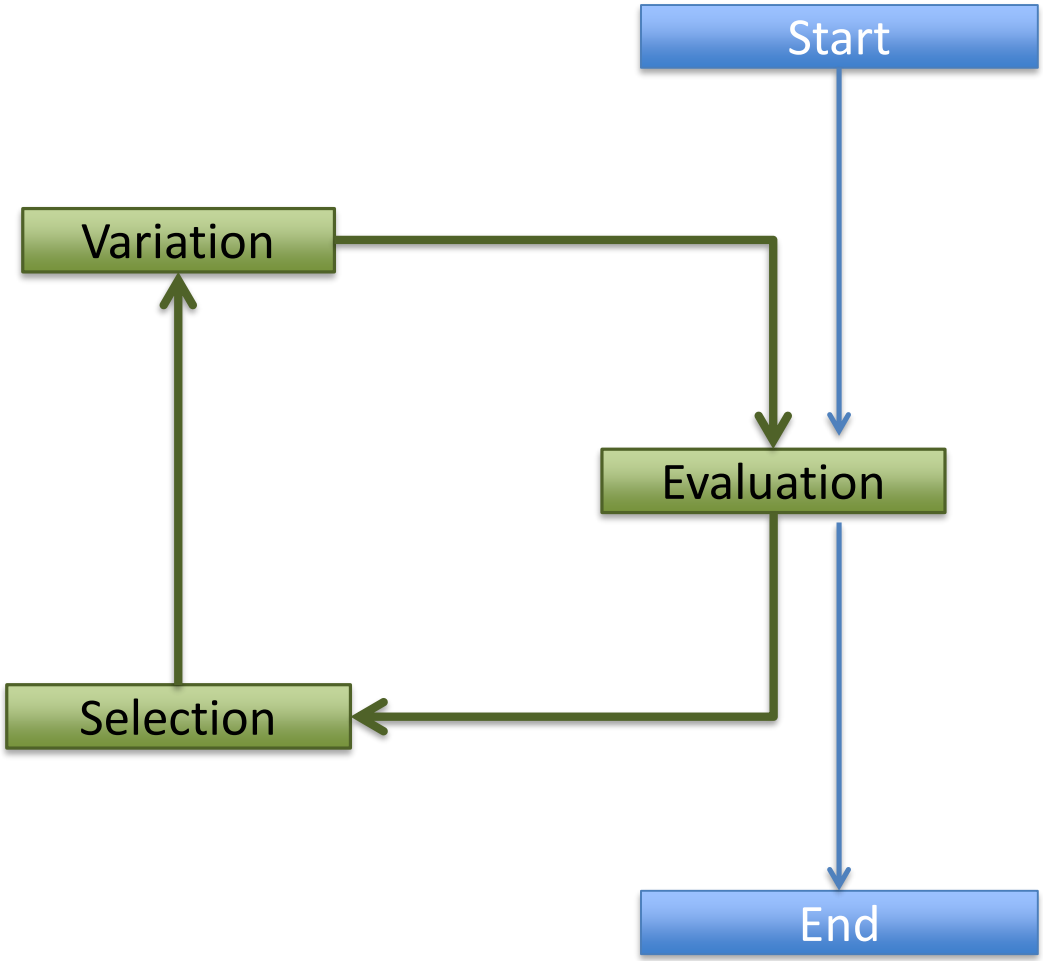
- Many heuristics impose a neighbourhood structure on *S*
- Such heuristics may guarantee that best point found is *locally optimal* e.g. Hill-Climbers:
  - But problems often exhibit many local optima
  - Often very quick to identify good solutions
- EAs are distinguished by (the combination of) the following:
  - Use of population
  - Use of multiple, stochastic search operators
  - Variation operators with arity >1, thus combining information of more candidate solutions
  - Stochastic selection

Question: what is a neighbourhood in an EA?

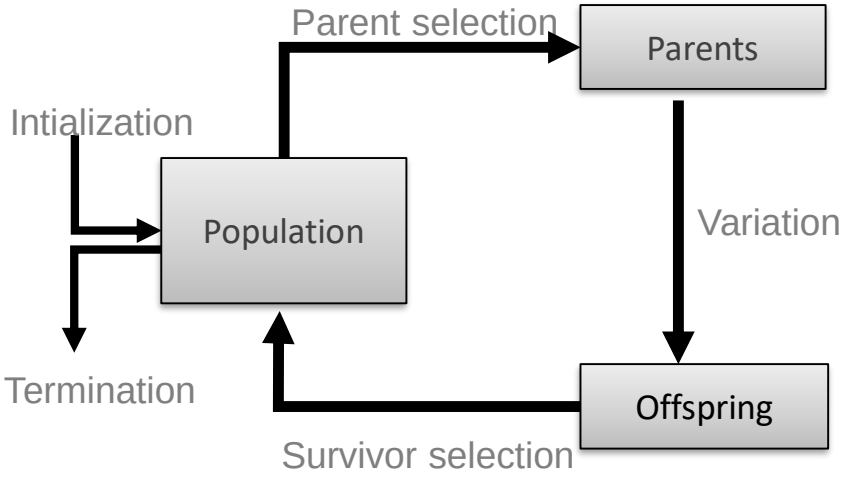
# General scheme of EAs – version 1

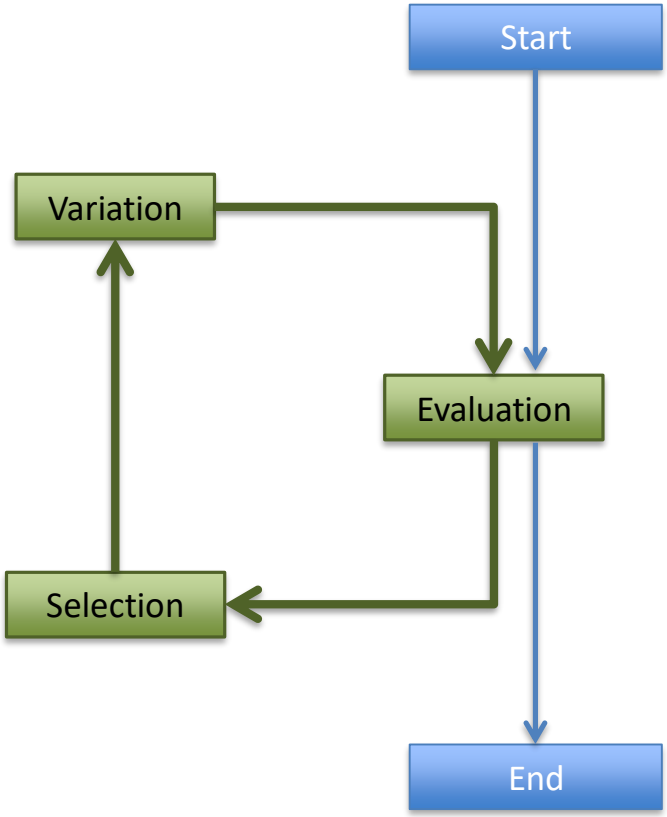


# General scheme of EAs – version 2



# Q: What is the difference?



# Important points:

- There are different diagrams to represent an EA scheme
- Variation operators push towards novelty and selection operators push towards quality
- Variation operators need to match the representation, selection operators are independent from the representation (hence, from the problem at hand)
- Selection operators act on the population level and variation operators on the individual level