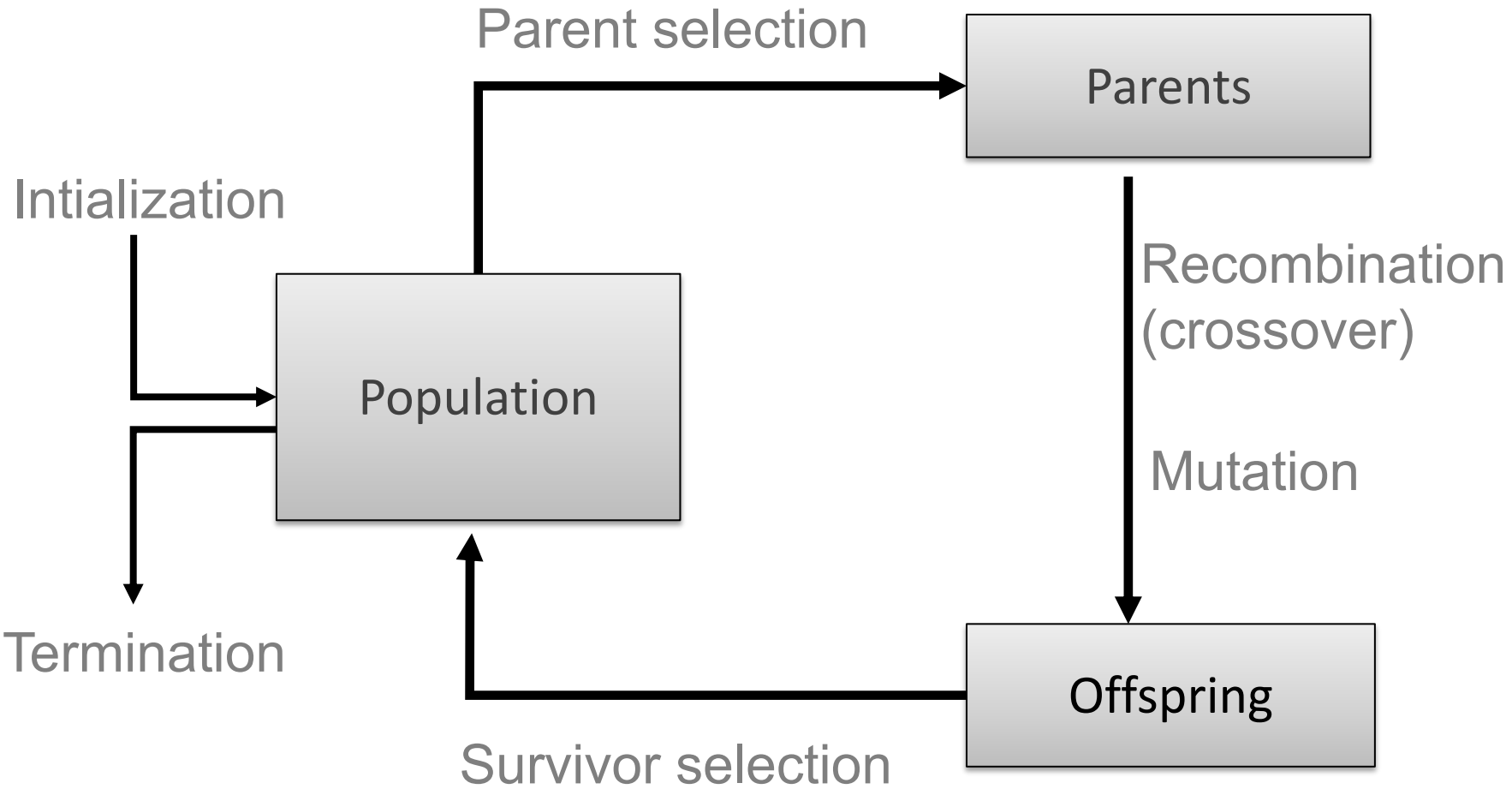
Evolutionary Computing

Chapter 3

Chapter 3:
What is an Evolutionary Algorithm?

- Scheme of an EA
- Main EA components:
  - Representation / evaluation / population
  - Parent selection / survivor selection
  - Recombination / mutation
- Examples: eight-queens problem
- Typical EA behaviour
- EAs and global optimisation

Scheme of an EA:
General scheme of EAs



Scheme of an EA:
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
```

Scheme of an EA: Common model of evolutionary processes

- Population of individuals
- Individuals have a fitness
- Variation operators: crossover, mutation
- Selection towards higher fitness
  - "survival of the fittest" and
  - "mating of the fittest"

Neo Darwinism:

Evolutionary progress towards higher life forms

=

Optimization according to some fitness-criterion

Scheme of an EA: Two pillars of evolution

There are two competing forces

Increasing population **diversity** by genetic operators

- mutation
- recombination

Push towards **novelty**

Decreasing population **diversity** by selection

- of parents
- of survivors

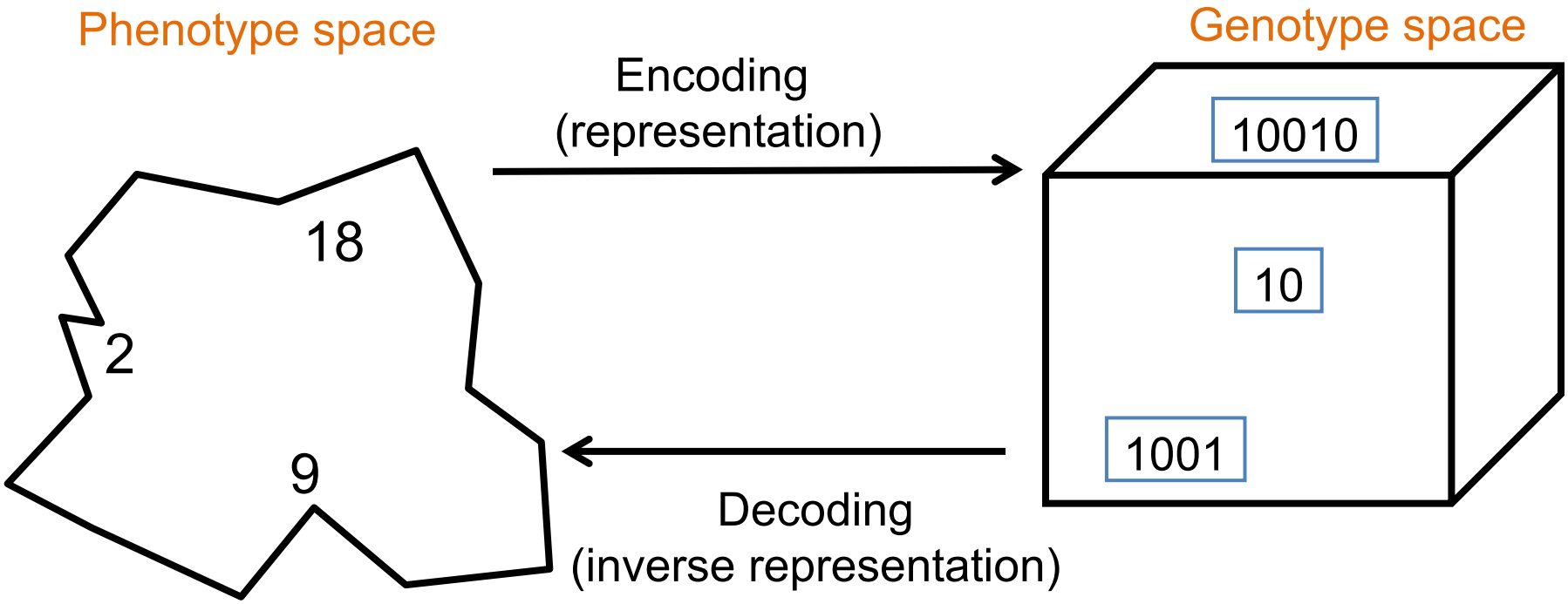
Push towards **quality**

Main EA components: Representation (1/2)

- 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
  - Decoding : genotype=> phenotype

Main EA components: Representation (2/2)

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 (1/2)

- Role: holds the candidate solutions of the problem as individuals
- Formally, a population is a multiset of individuals, i.e. repetitions are possible
- Population is the basic unit of evolution, i.e., the population is evolving, not the individuals
- Selection operators act on population
- Variation operators act on individual

Main EA components: Population (2/2)

- 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 present individuals

Main EA components: Selection mechanism (1/3)

Role:

- Identifies individuals
  - to become parents
  - to survive
- Pushes population towards higher fitness
- Usually probabilistic
  - high quality solutions more likely to be selected than low quality
  - but not guaranteed
- This *stochastic* nature can aid escape from local optima

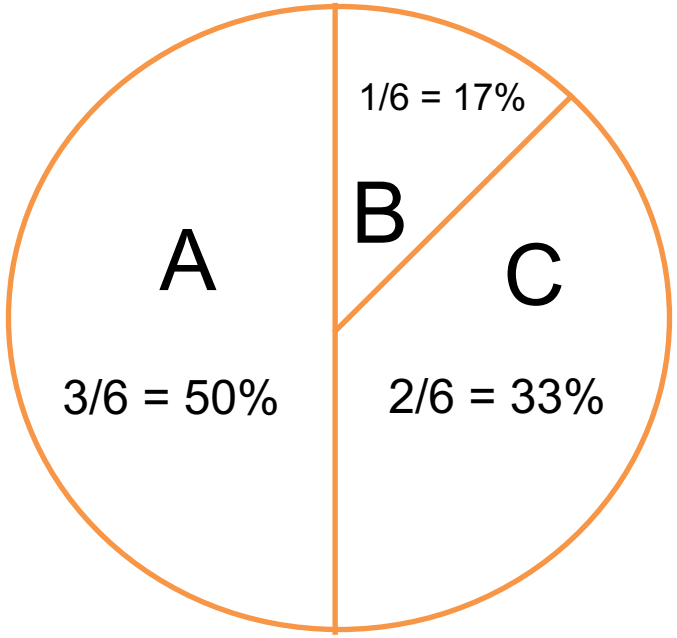
Main EA components: Selection mechanism (2/3)

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 mechanism (3/3)

- 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 (elitism)

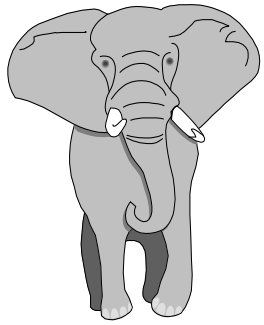
Main EA components: Variation operators

- 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 is formally possible, rarely used in EC
- There has been much debate about relative importance of recombination and mutation
  - Nowadays most EAs use both

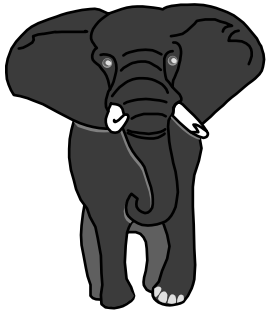
Main EA components: Mutation

- Role: causes small, random variance
- Acts on one genotype and delivers another
- Element of randomness is essential and differentiates it from other unary heuristic operators

before 1 1 1 1 1 1 1



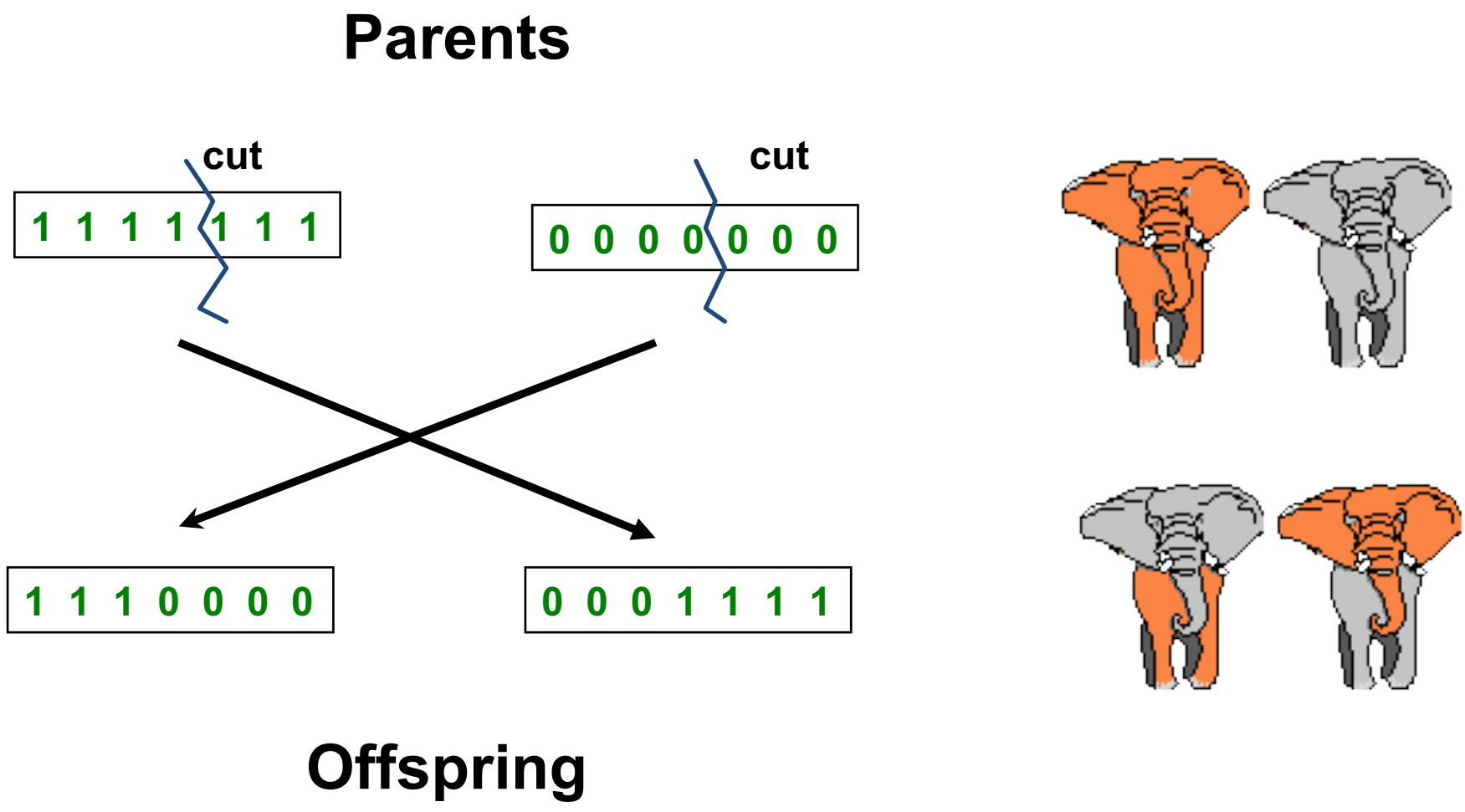
after 1 1 1 0 1 1 1



Main EA components: Recombination (1/2)

- 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 (2/2)



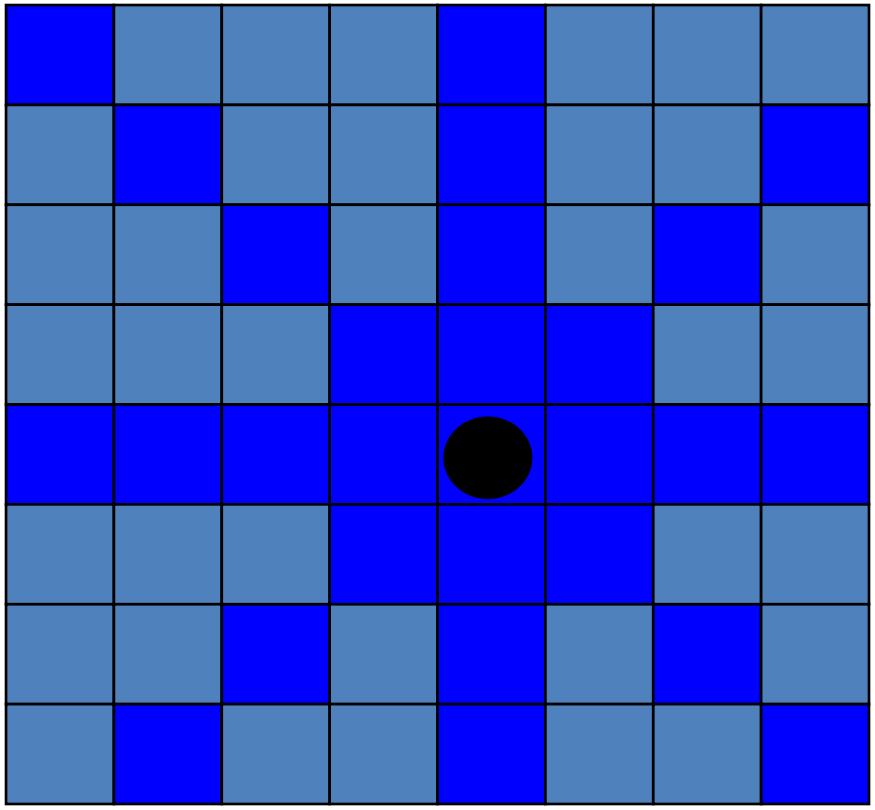
Main EA components: Initialization / Termination

- Initialisation usually done at random,
  - Need to ensure even spread and mixture of possible 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
  - Reaching some maximum allowed number of generations
  - Reaching some minimum level of diversity
  - Reaching some specified number of generations without fitness improvement

Main EA components: What are the 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 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

Example:
The 8-queens problem

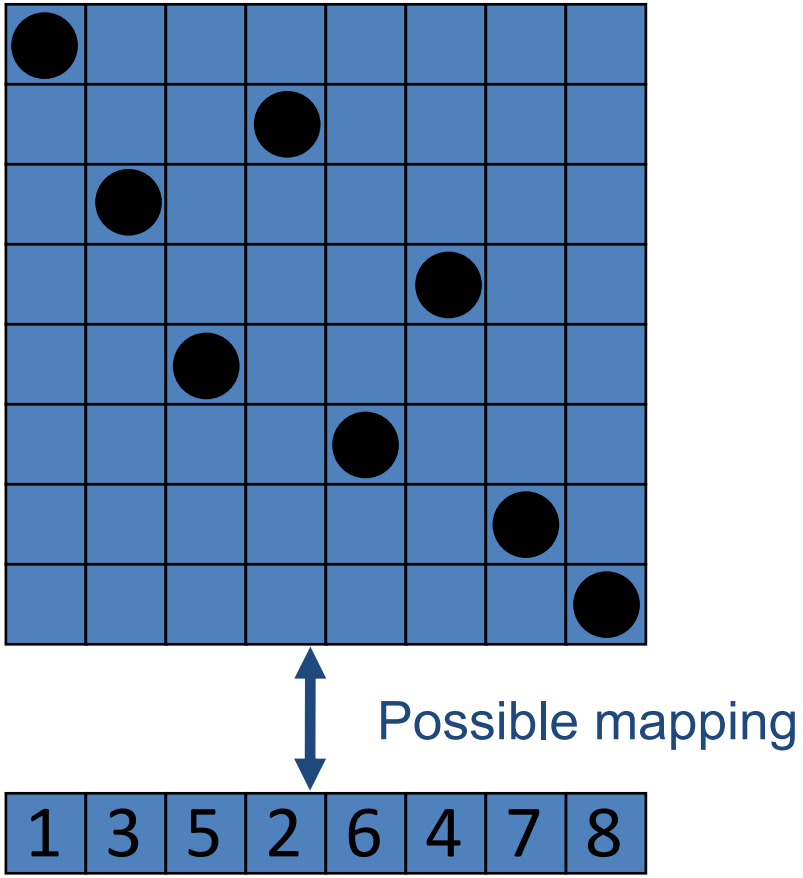


Place 8 queens on an 8x8 chessboard in such a way that they cannot check each other

The 8-queens problem: Representation

Phenotype:
a board configuration

Genotype:
a permutation of
the numbers 1–8



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

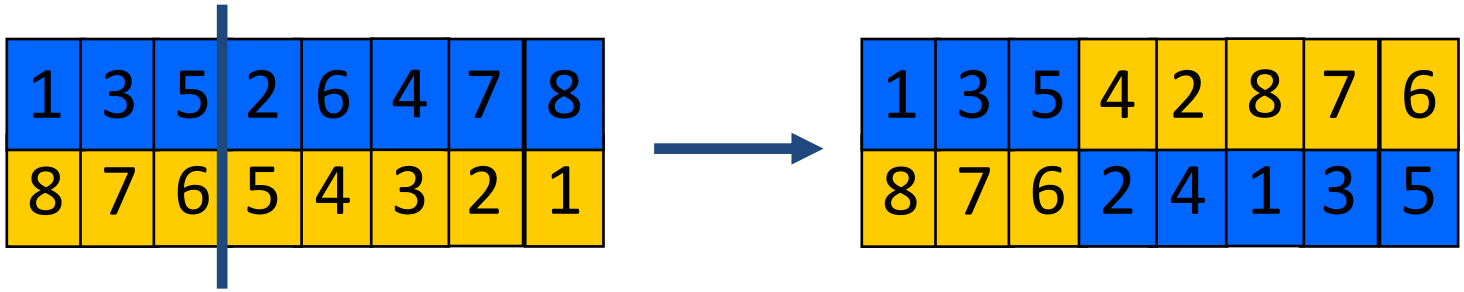
Small variation in one permutation, e.g.:

- 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 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
  - 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

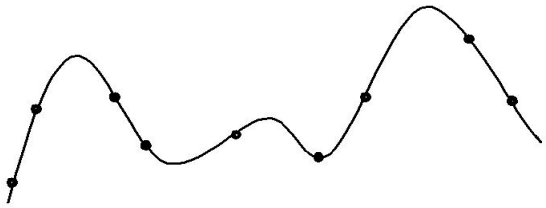
Note that is ***only one possible***
set of choices of operators and parameters

Atividade

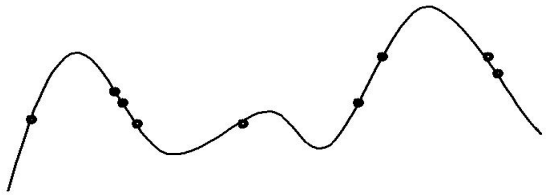
- Adapte sua implementação das 8-rainhas para ser solucionada pelo sistema evolutivo proposto no slide anterior.
- Entrega: 09/12/2020
- Submeter via SIGAA.

Typical EA behaviour: Stages

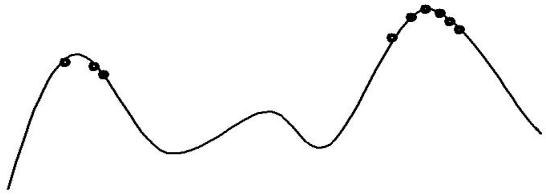
Stages in optimising on a 1-dimensional fitness landscape



Early stage:
quasi-random population distribution

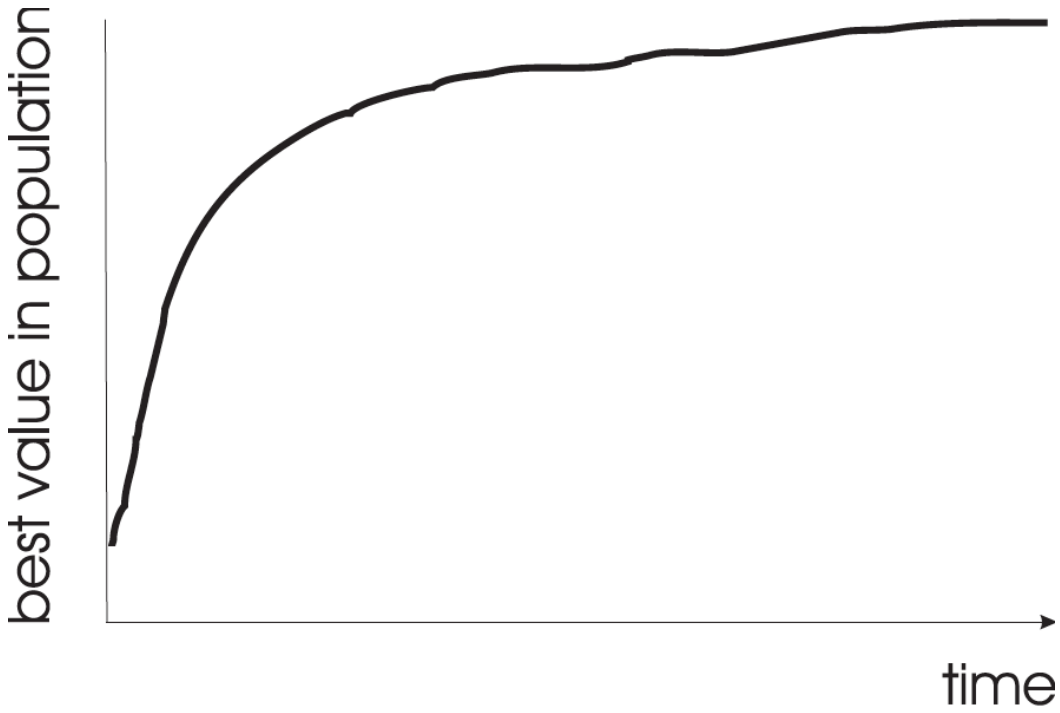


Mid-stage:
population arranged around/on hills



Late stage:
population concentrated on high hills

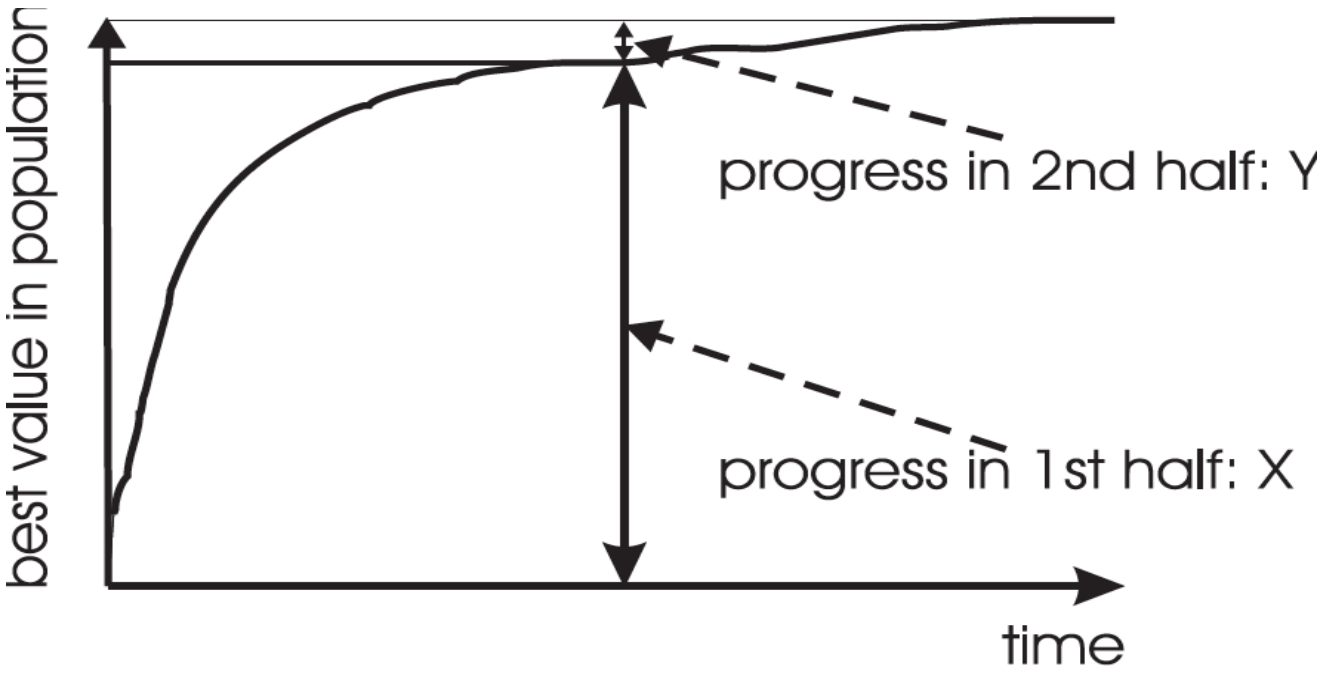
Typical EA behaviour:
Typical run: progression of fitness



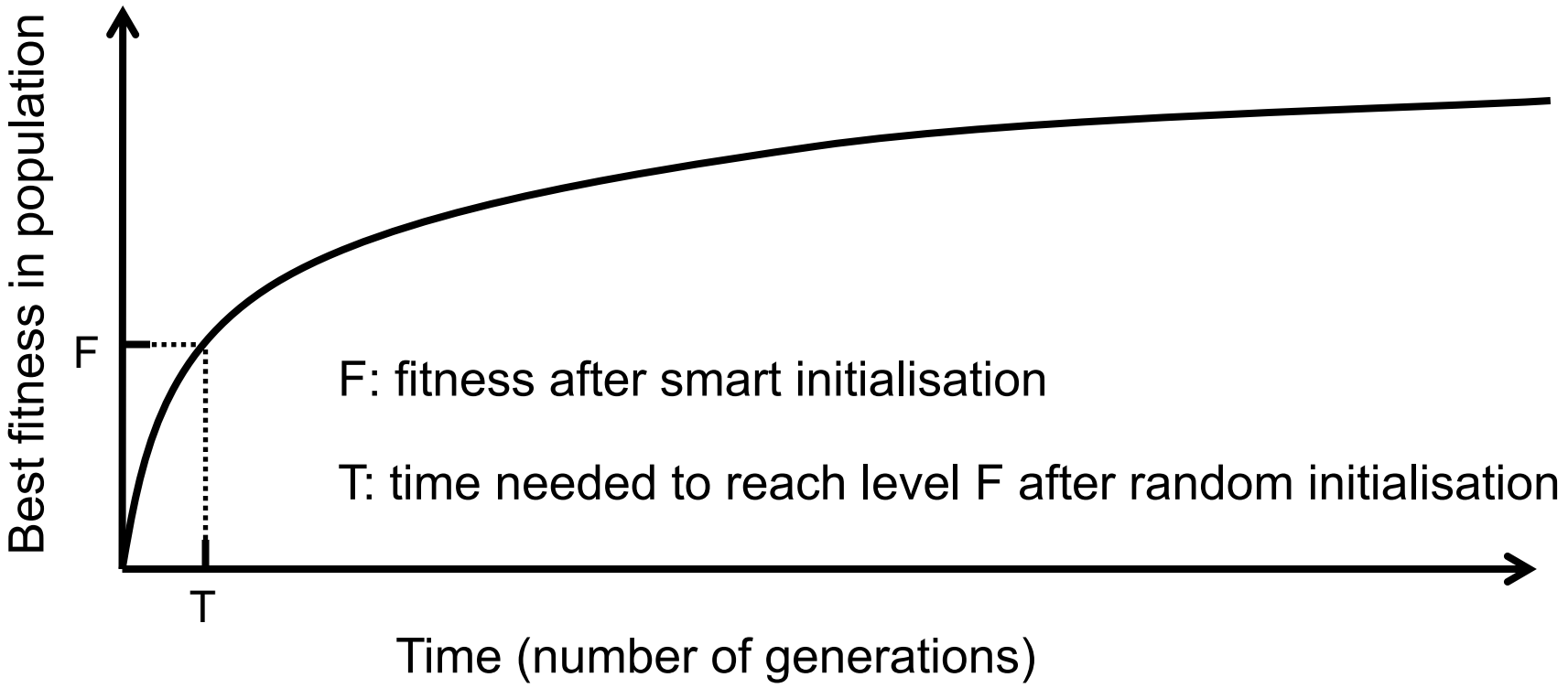
Typical run of an EA shows so-called “anytime behavior”

Typical EA behaviour: 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



Typical EA behaviour:
Is it worth expending effort on smart initialisation?

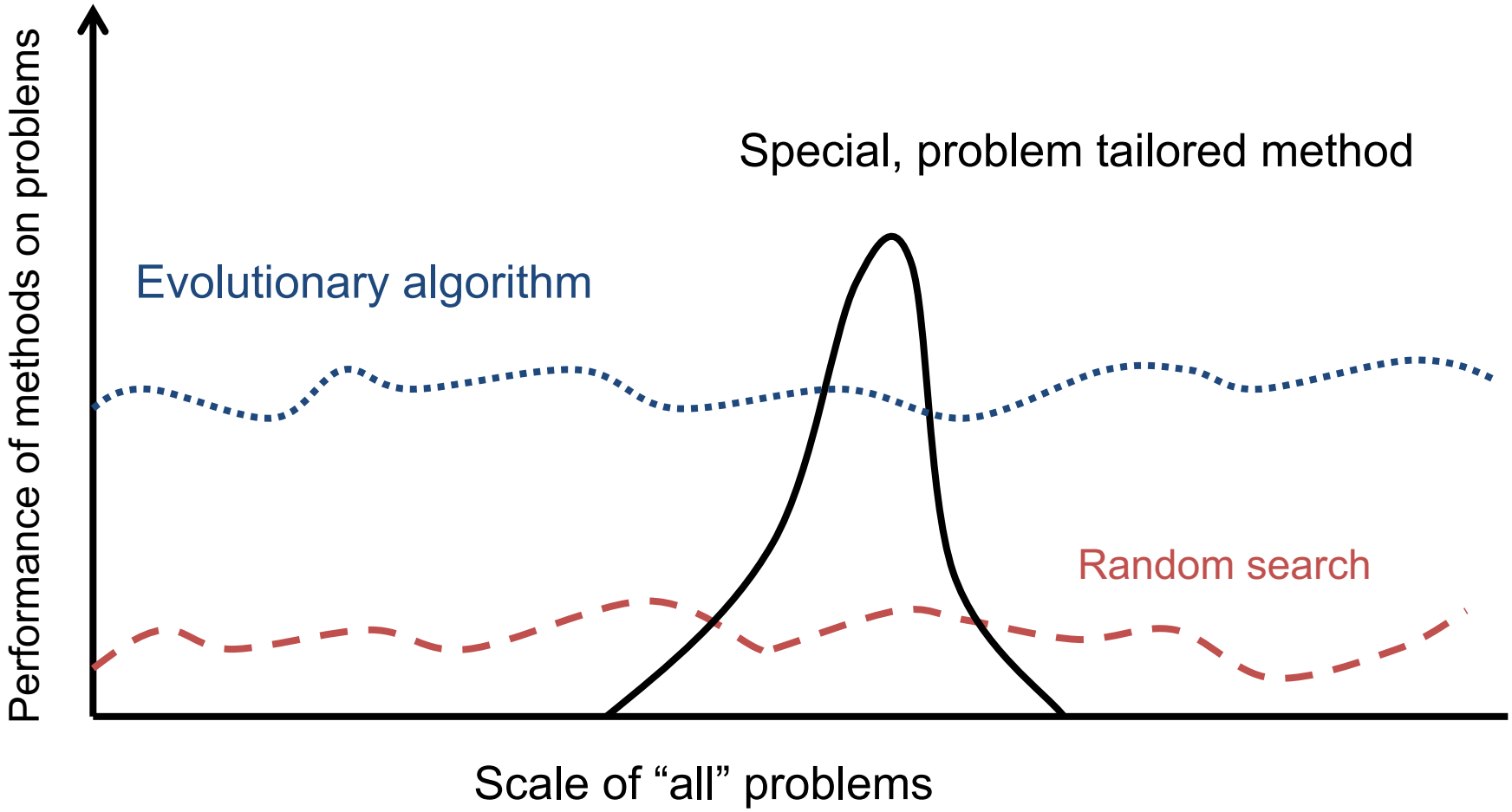


- Answer: it depends.
  - Possibly good, if good solutions/methods exist.
  - Care is needed.

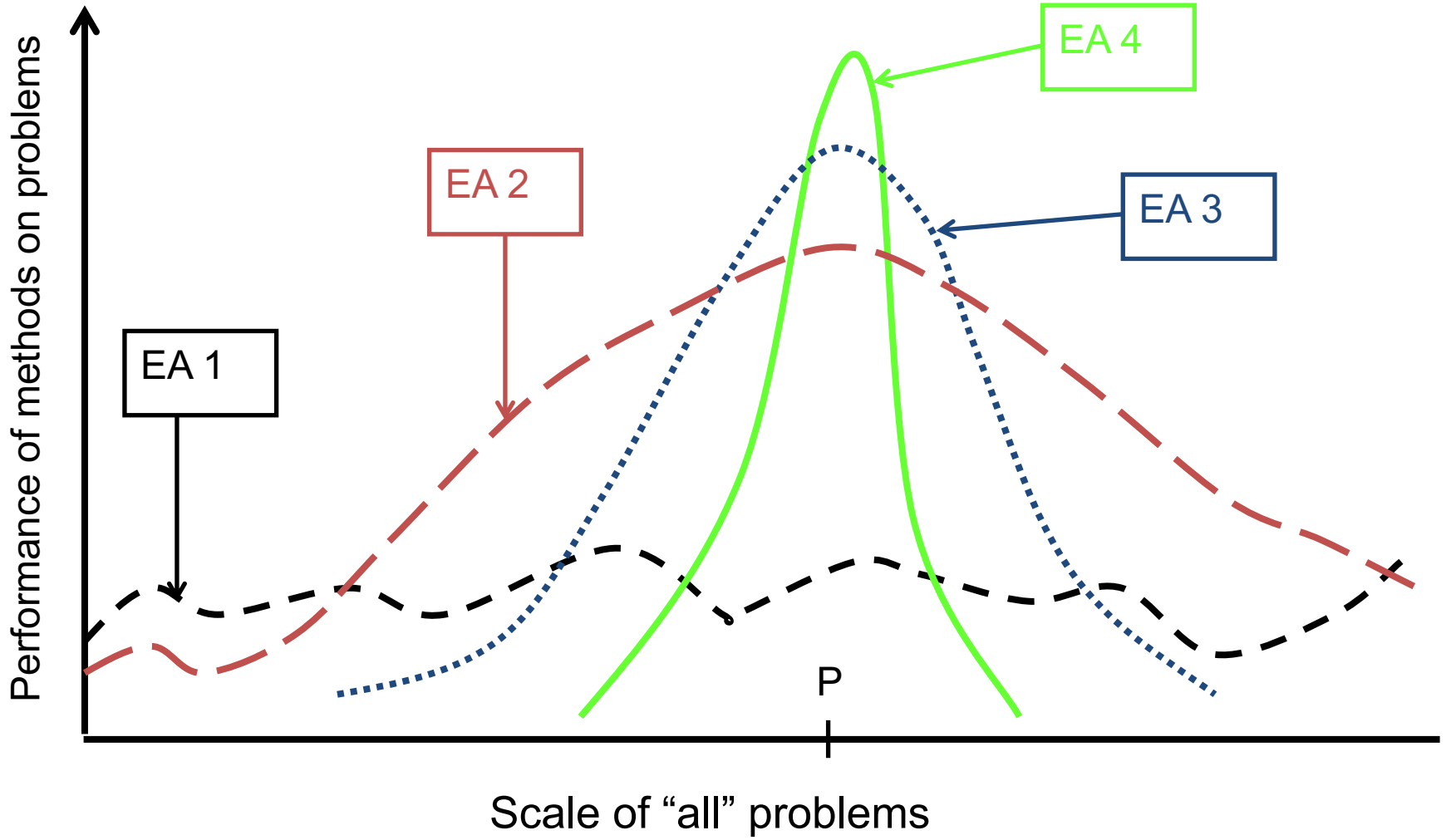
Typical EA behaviour: Evolutionary Algorithms in context

- There are many views on the use of EAs as robust problem solving tools
- For most problems a problem-specific tool may:
  - perform better than a generic search algorithm on most instances,
- Goal is to provide robust tools that provide:
  - evenly good performance
  - over a range of problems and instances

Typical EA behaviour:
EAs as problem solvers: Goldberg view (1989)



Typical EA behaviour:
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 limit 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 limit on runtime