Evolutionary Computing

Chapter 3

Chapter 3: What is an Evolutionary Algorithm?

- EC metaphor
- 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
- EC and neighbourhood search

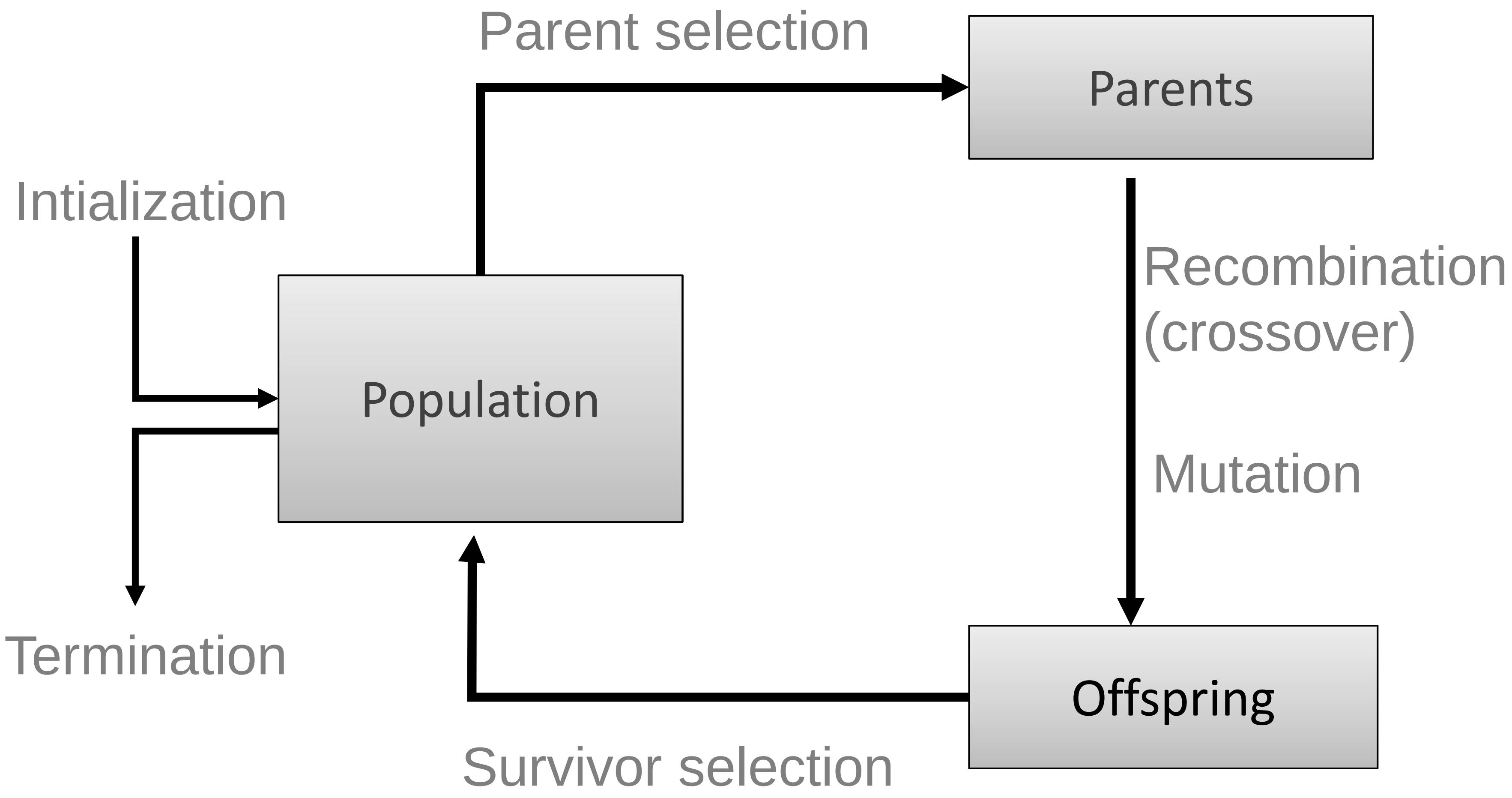
Recap of EC metaphor (1/2)

- A population of individuals exists in an environment with limited resources
- ***Competition*** for those resources causes selection of those ***fitter*** individuals that are better adapted to the environment
- These individuals act as seeds for the generation of new individuals through recombination and mutation
- The new individuals have their fitness evaluated and compete (possibly also with parents) for survival.
- Over time ***Natural selection*** causes a rise in the fitness of the population

Recap of EC metaphor (2/2)

- EAs fall into the category of “generate and test” algorithms
- They are stochastic, population-based algorithms
- Variation operators (recombination and mutation) create the necessary diversity and thereby facilitate novelty
- Selection reduces diversity and acts as a force pushing quality

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
(optimization on a fitness landscape)

Scheme of an EA:
Two pillars of evolution

There are two competing but complementary procedures:

Increasing population **diversity** by genetic operators

- mutation
- recombination

Push towards **novelty**

Decreasing population **diversity** by selection

- of parents
- of survivors

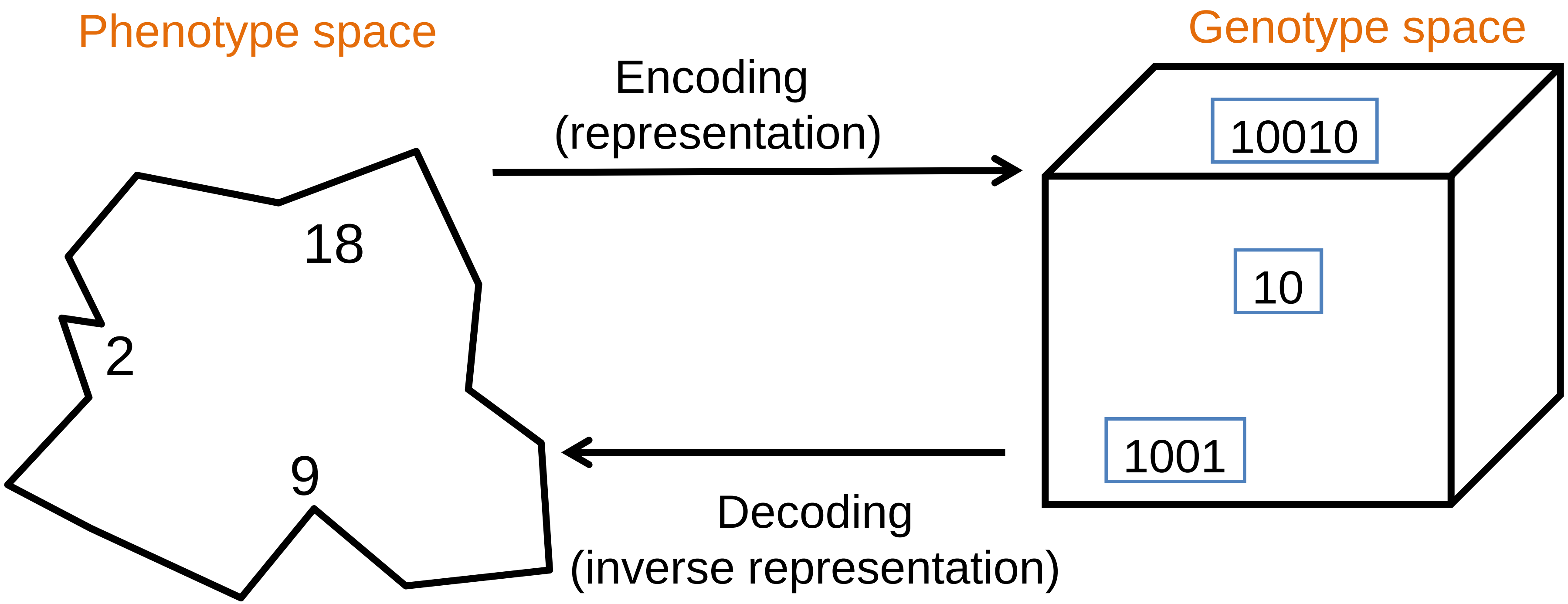
Push towards **quality**

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 (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)

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:
  - Establishes criteria to assess a solution, the requirements to be adapted (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 a problem as a group of individuals (genotypes).
- 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 themselves.
- Selection operators act on population level.
- Variation operators act on individual level.

Main EA components: Population (2/2)

- Some sophisticated EAs also assert a spatial structure on the population e.g., a grid:
  - The structure often constraints the interaction between individuals.
- Selection operators usually take whole population into account i.e., reproductive probabilities are *relative* to *current* generation:
  - Reproductive probability should consider individual fitness.
- Diversity of a population refers to the number of different fitnesses / phenotypes / genotypes present (note: not the same thing)

Main EA components: Selection mechanism (1/3)

Role:

- Identifies individuals:
  - to become parents;
  - to survive.
- Induces population towards higher fitness.
- Usually probabilistic
  - High quality solutions more likely to be selected than low quality;
  - even worst in current population usually has non-zero probability of being selected.
- This *stochastic* nature can aid escape from local optima.

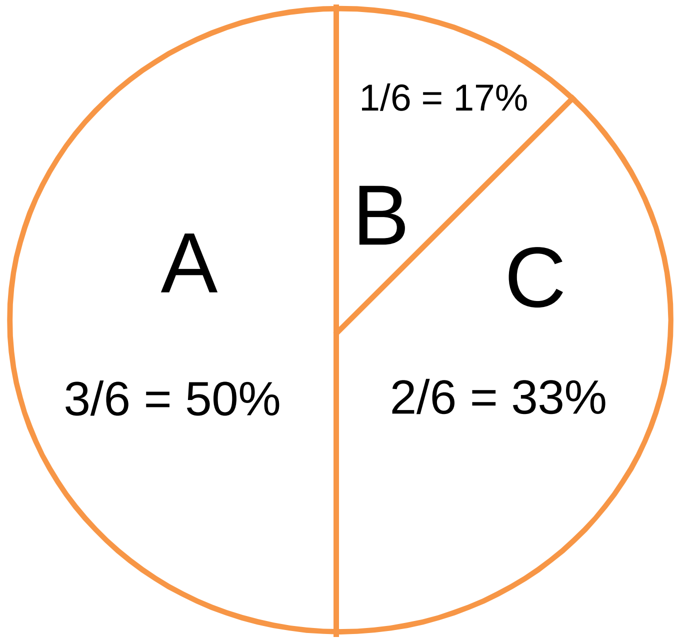
Main EA components: Selection mechanism (2/3)

Example: roulette wheel selection

fitness(A) = 3

fitness(B) = 1

fitness(C) = 2



In principle, any selection mechanism can be used for both parent selection and 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.
- It can be 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

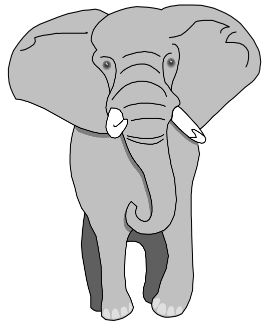
- Role: to generate new candidate solutions.
- Usually divided into two types according to their arity (number of inputs):
  - Arity 1 : mutation operators;
  - Arity >1 : recombination operators;
  - Arity = 2 typically called crossover;
  - Arity > 2 is formally possible, seldom used in EC.
- There has been much debate about relative importance of recombination and mutation:
  - Nowadays most EAs use both;
  - Variation operators must match the given representation.

Main EA components: Mutation (1/2)

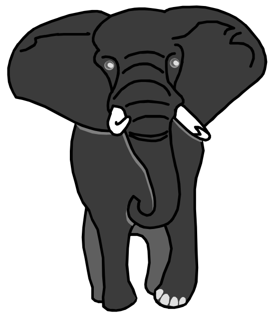
- Role: causes small and random variance.
- Acts on one genotype and delivers another.
- Element of randomness is essential and differentiates it from other unary heuristic operators.
- Importance ascribed depends on representation and historical dialect:
  - Binary GAs – background operator responsible for preserving and introducing diversity;
  - EP for FSM’s / continuous variables – only search operator;
  - GP – hardly used.
- May guarantee connectedness of search space and hence convergence proofs.

Main EA components:
Mutation (2/2)

before 1 1 1 1 1 1 1



after 1 1 1 0 1 1 1

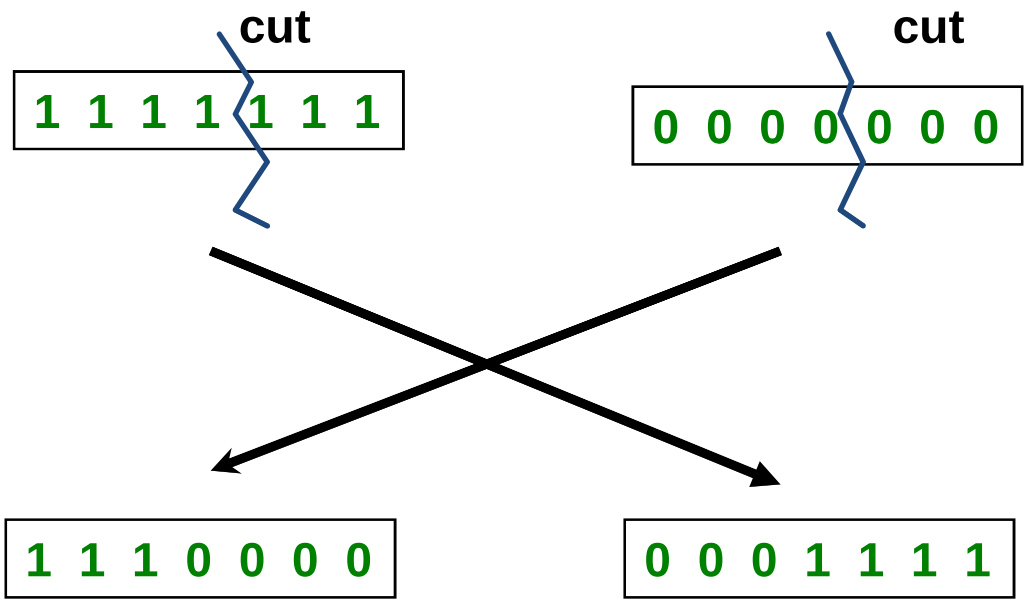


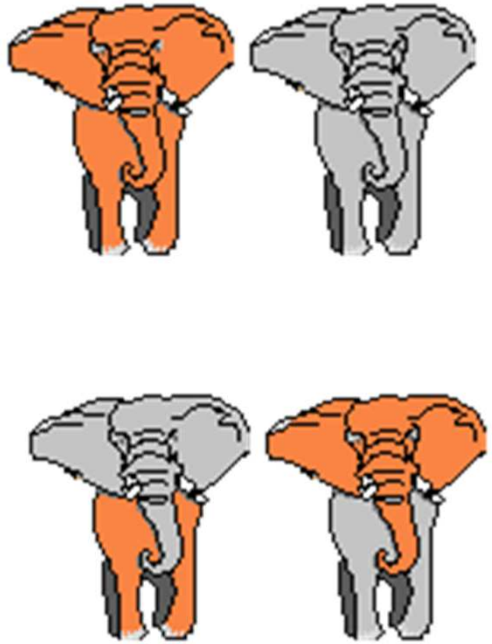
Main EA components: Recombination (1/2)

- Role: merges information from parents into offspring.
- Choice of what information to merge is stochastic.
- Most offspring may be less fit or equally fit than the parents.
- Hopefully, some of them can be fitter as a result of the combination of 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)

Parents





Offspring

Main EA components: Initialisation / Termination

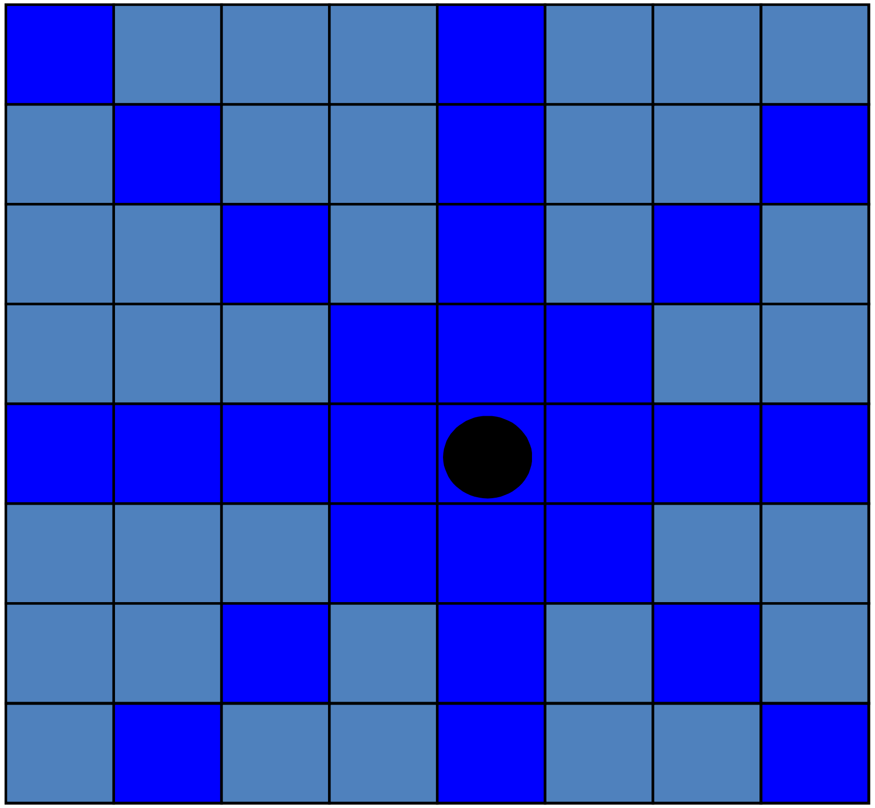
- Initialisation usually done at random:
  - Need to ensure even spread and combinations of possible allele values;
  - Can also include existing solutions, or use problem-specific heuristics, to “seed” the population.

- 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;
  - Combinations of the alternatives above.

Main EA components: What are the different types of EAs

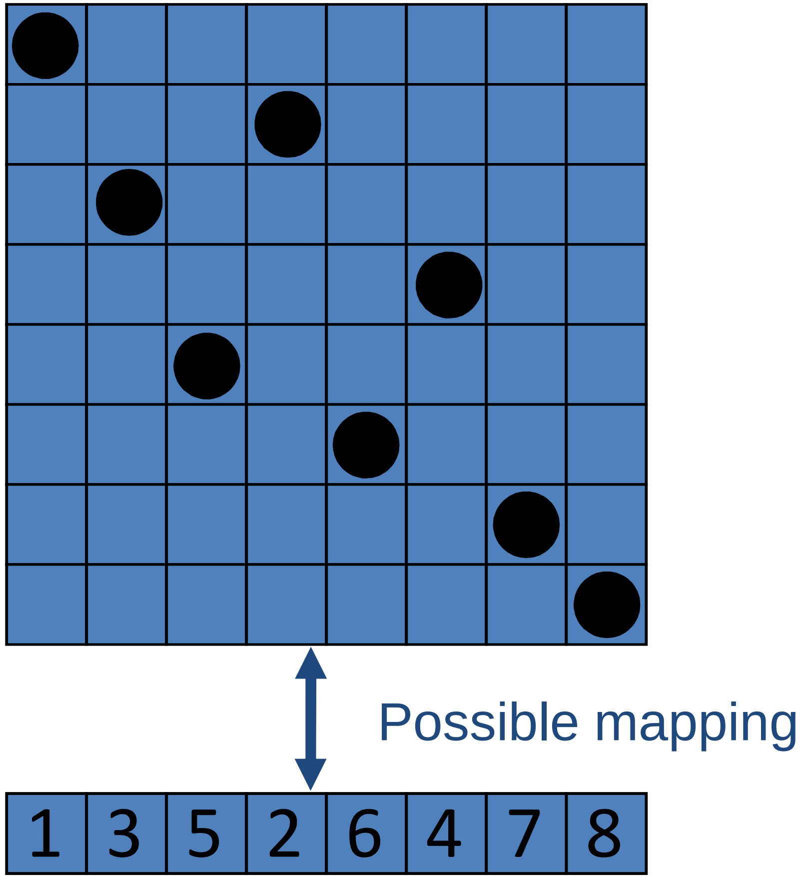
- Historically different flavours of EAs have been associated with different data types to represent solutions:
  - Binary strings : Genetic Algorithms;
  - Real-valued vectors : Evolution Strategies;
  - Finite state Machines: Evolutionary Programming;
  - LISP trees: Genetic Programming.
- These differences 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,

1	3	5	2	6	4	7	8

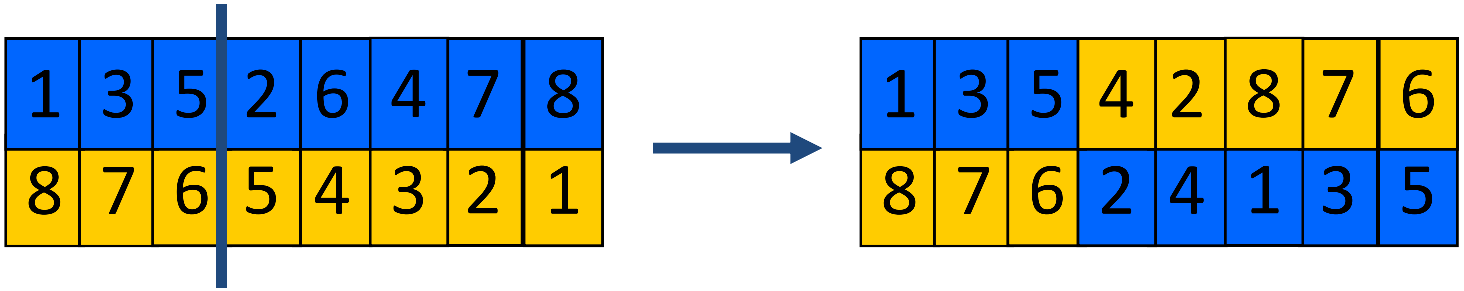
$\rightarrow$

1	3	7	2	6	4	5	8

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 (tournament):
  - 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.

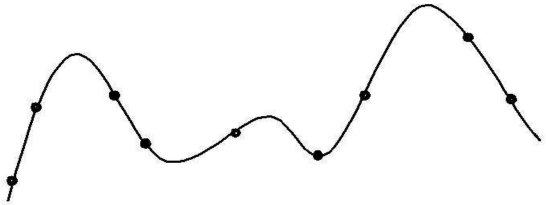
The 8-queens problem: Summary

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

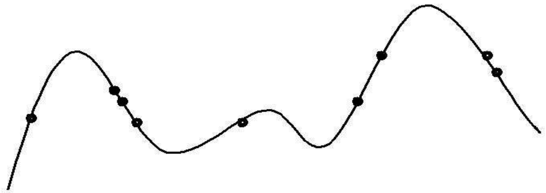
Typical EA behavior: Stages

Stages in optimizing on a 1-dimensional fitness landscape



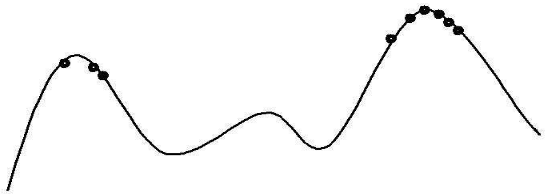
Early stage:

Population tends to be random distributed



Mid-stage:

Population tends to be arranged around/on hills

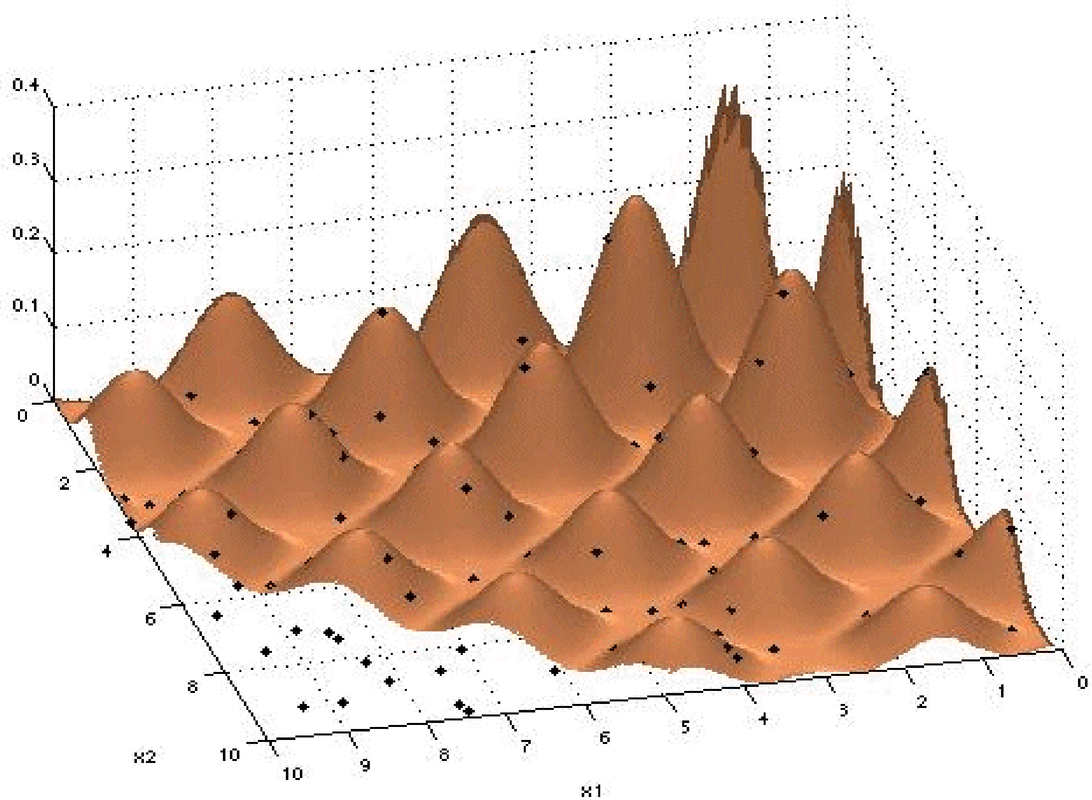


Late stage:

Population concentrated on high hills

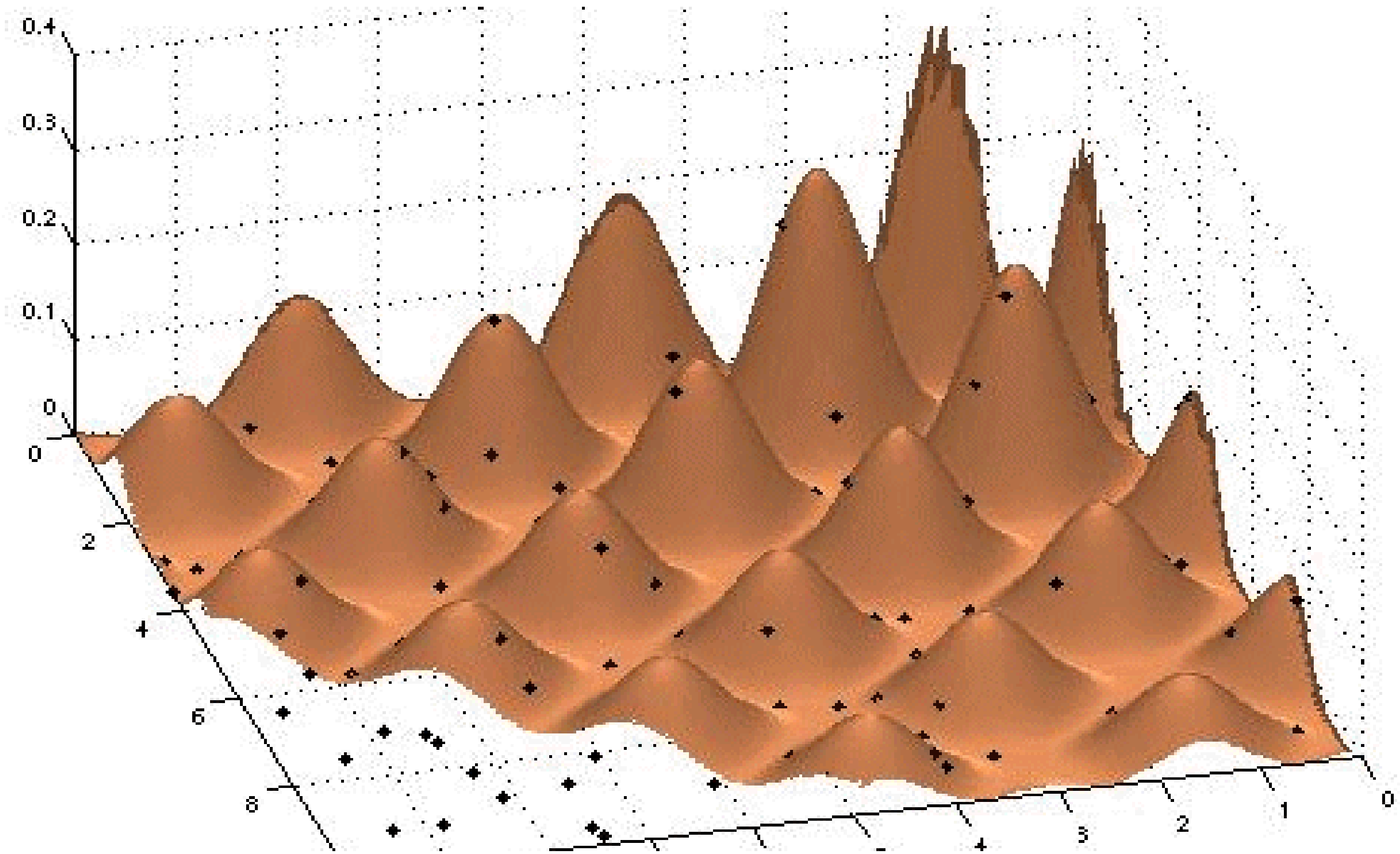
Typical EA behavior: Working of an EA demo (1/2)

Searching a fitness landscape without “niching”



Typical EA behaviour: Working of an EA demo (2/2)

Searching a fitness landscape with “niching”

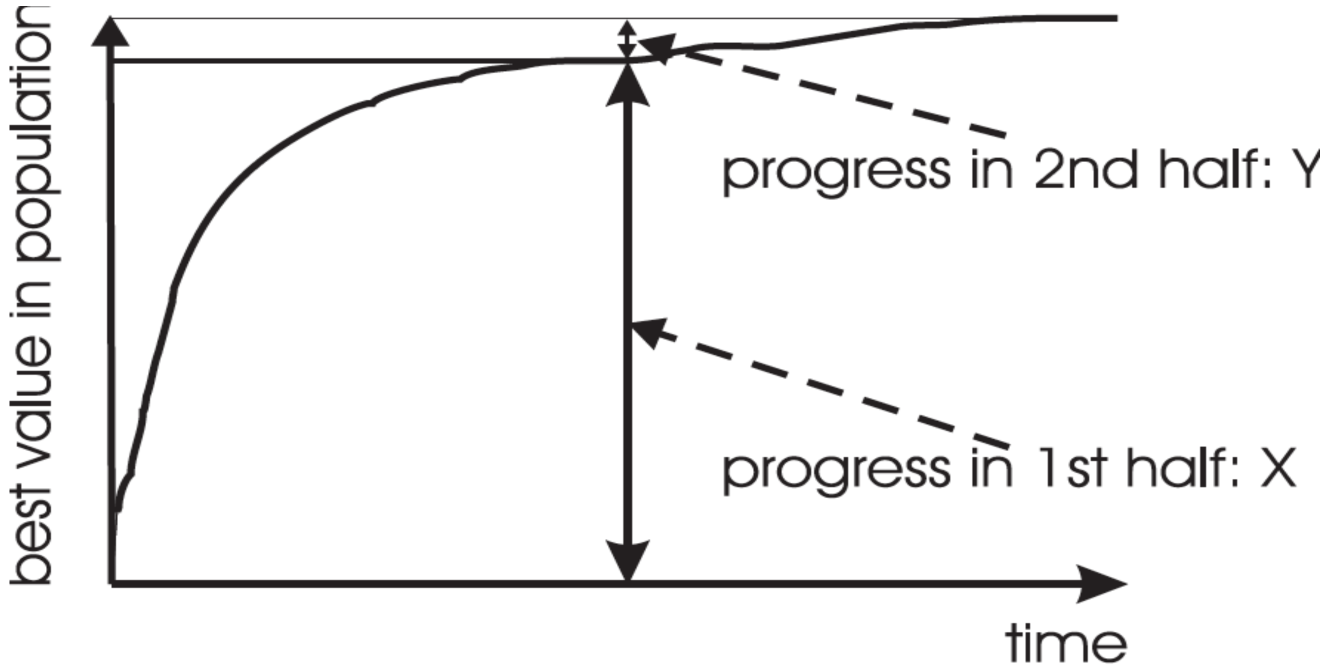


Typical EA behaviour:
Typical run: progression of fitness

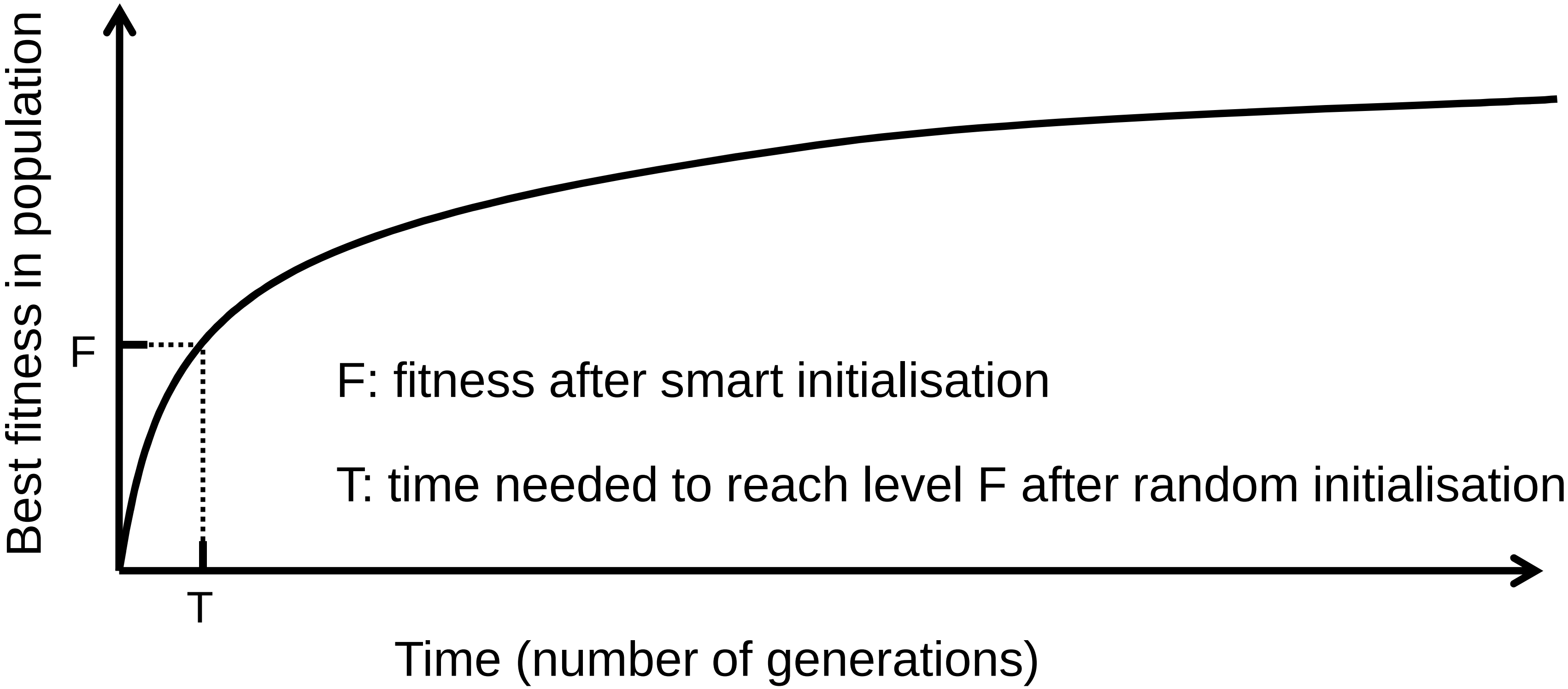
Typical run of an EA shows so-called “anytime behavior”

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?



F: fitness after smart initialisation

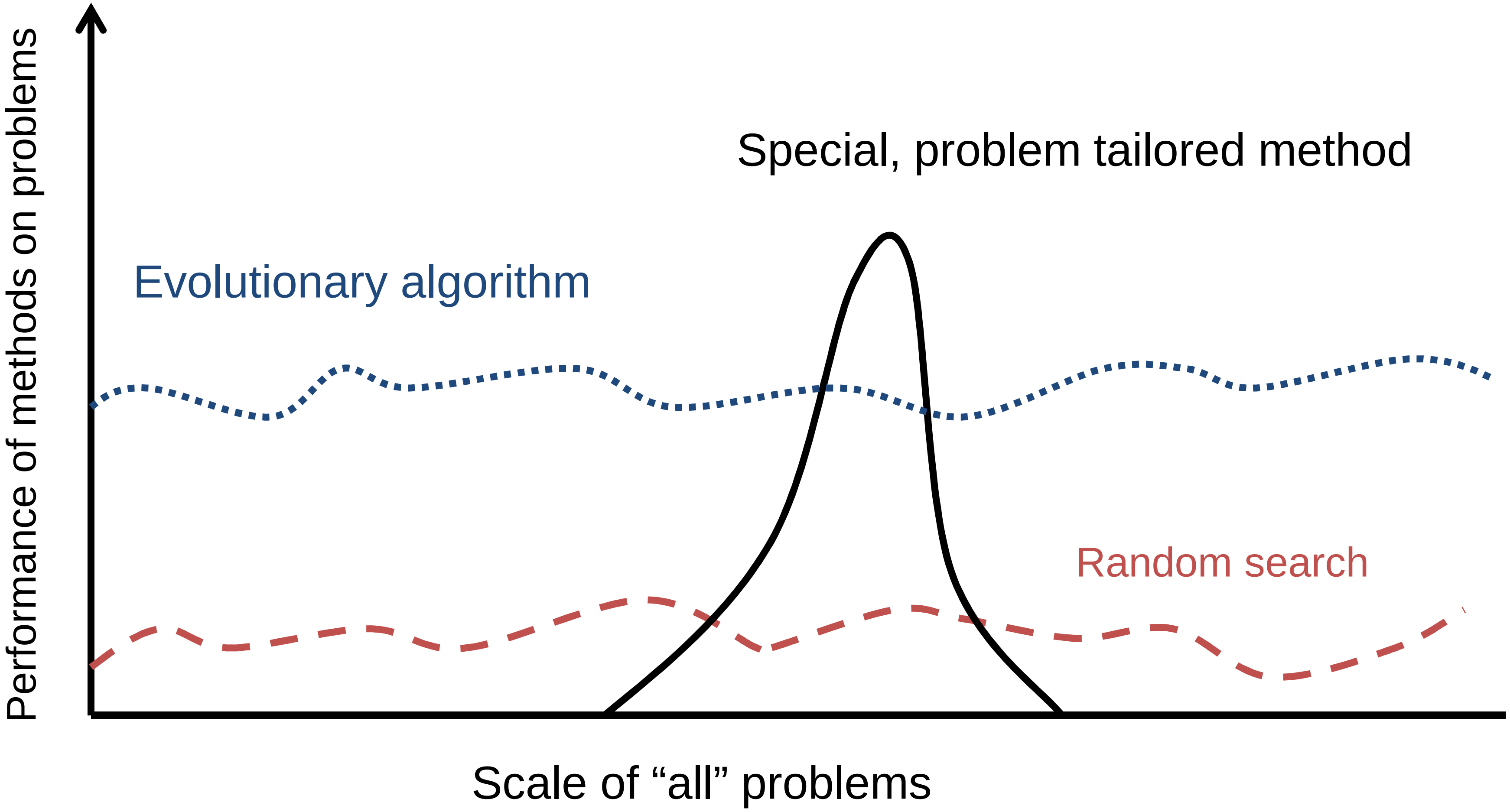
T: time needed to reach level F after random initialisation

- Answer: it depends.
  - Possibly good, if good solutions/methods exist.
  - Care is needed, see chapter/lecture on hybridisation.

Typical EA behavior: 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;
  - Have limited utility;
  - Not do well on all instances.
- Goal is to provide robust tools that provide:
  - Evenly good performance;
  - Over a range of problems and instances.

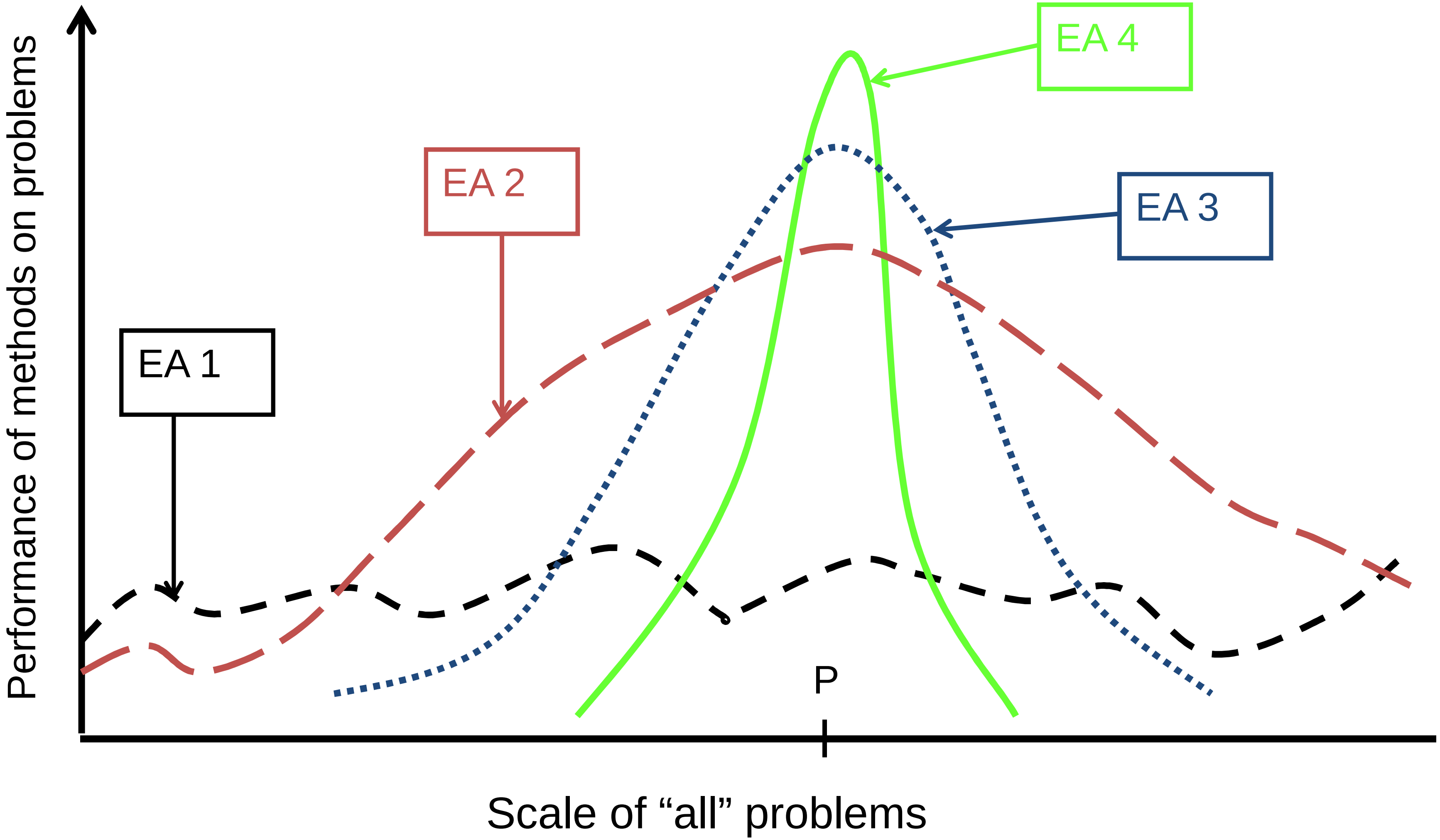
Typical EA behavior:
EAs as problem solvers: Goldberg view (1989)



Typical EA behaviour: EAs and domain knowledge

- Trend in the 90's:
  - Addition of problem specific knowledge to Eas, such as special variation operators, repair;
- 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.
- Recent theory suggests the search for an "all-purpose" algorithm may be fruitless.

Typical EA behavior:
EAs as problem solvers: Michalewicz view (1996)



Adapted from A.E. Eiben and J.E. Smith, Introduction to Evolutionary Computing 2014

EC and global optimisation

- Global Optimisation: search for finding best solution x^* out of some fixed set *S;*
- Deterministic approaches:
  - Such as 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.

- "I don't care if it works as long as it converges"

  vs.

- "I don't care if it converges as long as it works"

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:
  - Use of population;
  - Use of multiple, stochastic search operators;
  - Especially variation operators with arity >1;
  - Stochastic selection.
- Question: what is the neighbourhood in an EA?