Intro to Evolutionary Computing

David Howard
Research Scientist – Robotics and Autonomous Systems
CPS Program | Data61 | CSIRO

www.csiro.au

October 2018

Slides adapted from a guest lecture by a CSIRO colleague
david.howard@data61.csiro.au

CSIRO Autonomous Systems

CSIRO is Australia's largest science/research agency.
AS lab facts: ~100 people, mainly in Brisbane and Sydney.
Highly collaborative, strong ties to industry.

Lab focus is Field Robotics: engineering, control, vision, planning…

My role: apply evolutionary algorithms to increase the long term autonomy of real-world robots doing useful tasks.

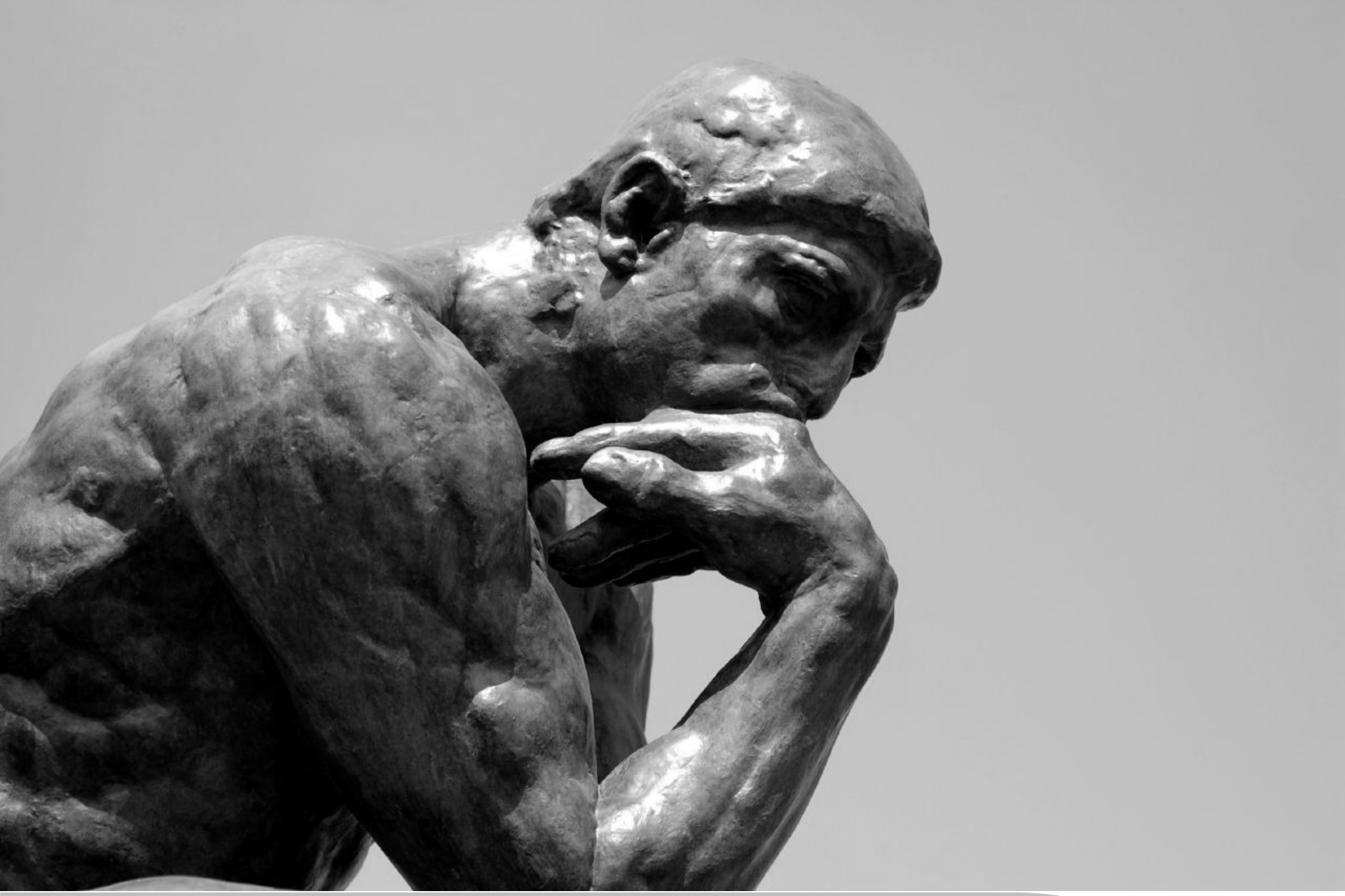
MOTIVATION

3.8 to 2.5 billion years ago

Mantis Shrimp

Zombie ant fungus

Tardigrade



Evolution in the real world

1. A creature is defined by a genome
2. Genome encodes traits (blonde hair, brown eyes, running speed, etc.) to appear in a phenotype
3. Genotype *expresses* a phenotype
4. The environment creates a selection pressure which gives us “survival of the fittest”
5. Highly fit creatures become parents and can pass on beneficial traits to children
6. Transfer is messy = random variations

The benefits of evolution

Creativity
Unsupervised
Complexity
Adaptive
Specialised
High performing
Robust

Incrementalism

Cooperation

Altruism

Evolution lets us study the effects that natural processes may have, and can harness them if beneficial!

Creatures vs robots

Co-evolution

History of Evolutionary Algorithms

Evolutionary Computing: the Origins

- Historical perspective
- Biological inspiration:
  - Darwinian evolution theory (simplified!)
  - Genetics (simplified!)
- Motivation for EC

Historical perspective (1/3)

- 1948, Turing:
  proposes “genetical or evolutionary search”
- 1962, Bremermann:
  optimization through evolution and recombination
- 1964, Rechenberg:
  introduces evolution strategies
- 1965, L. Fogel, Owens and Walsh:
  introduce evolutionary programming
- 1975, Holland:
  introduces genetic algorithms
- 1992, Koza:
  introduces genetic programming

Historical perspective (2/3)

- 1985: first international conference (ICGA)
- 1990: first international conference in Europe (PPSN)
- 1993: first scientific EC journal (MIT Press)
- 1997: launch of European EC Research Network EvoNet

Historical perspective (3/3)

EC in the early 21st Century:

- 4 major EC conferences, about 10 small related ones
- Many scientific core EC journals (ECJ, TEC, GPEM, EI)
- 1000+ EC-related papers published last year(estimate)
- uncountable (meaning: many) applications
- uncountable (meaning: ?) consultancy and R&D firms
- part of many university curricula

Darwinian Evolution (1/3): Survival of the fittest

- All environments have finite resources

  (i.e., can only support a limited number of individuals)
- Life forms have basic instinct/lifecycles geared towards reproduction
- Therefore some kind of selection is inevitable
- Those individuals that compete for the resources most effectively have increased chance of reproduction
- Note: fitness in natural evolution is a derived, secondary measure, i.e., we (humans) assign a high fitness to individuals with many offspring

Darwinian Evolution (2/3): Diversity drives change

- Phenotypic traits:
  - Behaviour / physical differences that affect response to environment
  - Partly determined by inheritance, partly by factors during development (nature vs. nurture)
  - Unique to each individual, partly as a result of random changes
- If phenotypic traits:
  - Lead to higher chances of reproduction
  - Can be inherited

  then they will tend to increase in subsequent generations, leading to new combinations of traits …

Darwinian Evolution (3/3): Summary

- Population consists of diverse set of individuals
- Combinations of traits that are better adapted tend to increase representation in population

Individuals are “units of selection”

- Variations occur through random changes yielding constant source of diversity, coupled with selection means that:

Population is the “unit of evolution”

- Note the absence of “guiding force”

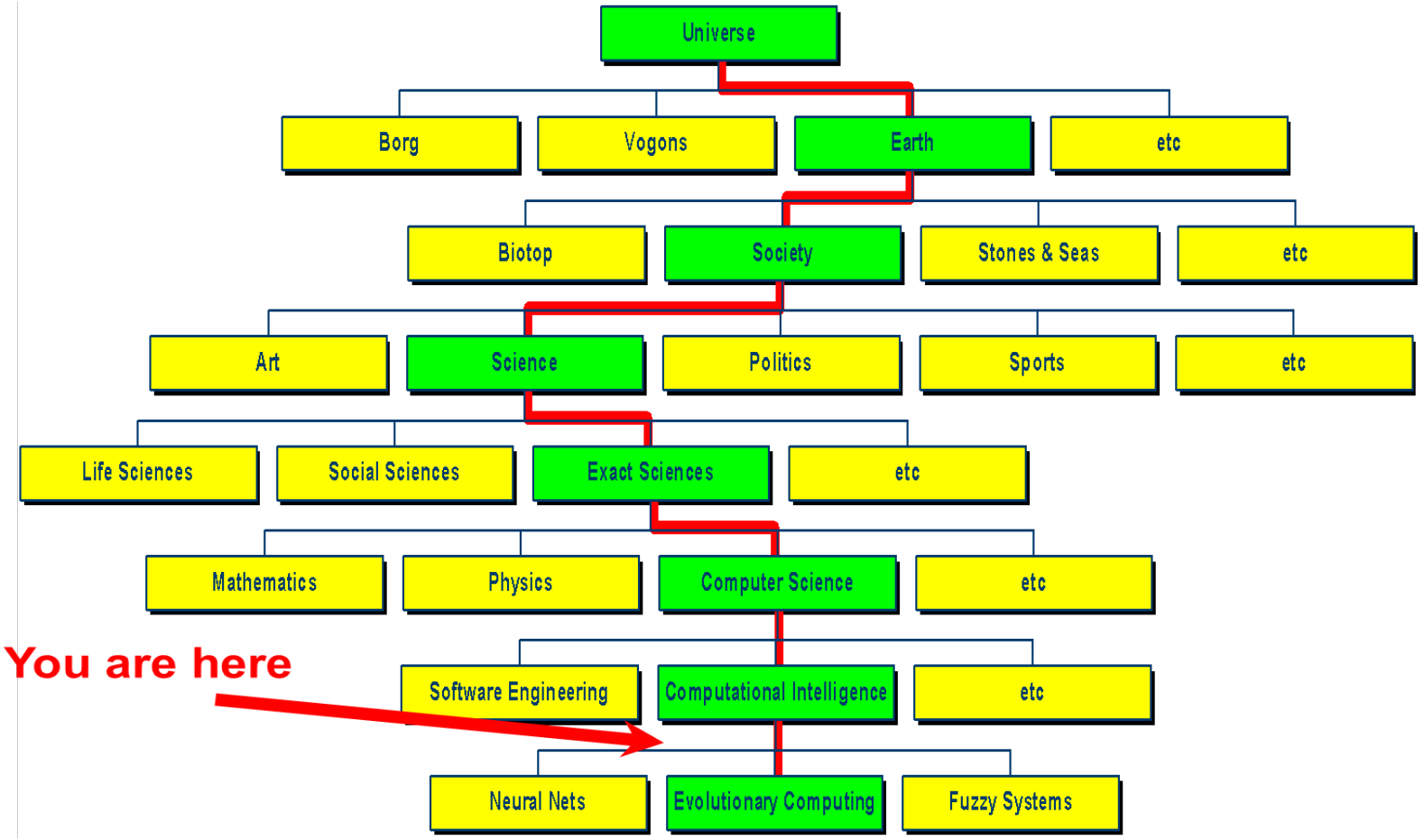
Quote (D. Dennett)

"If you have variation, heredity, and selection, then you must get evolution"

Quote (John Maynard Smith)

"So far, we have been able to study only one evolving system and we cannot wait for interstellar flight to provide us with a second. If we want to discover generalizations about evolving systems, we have to look at artificial ones."

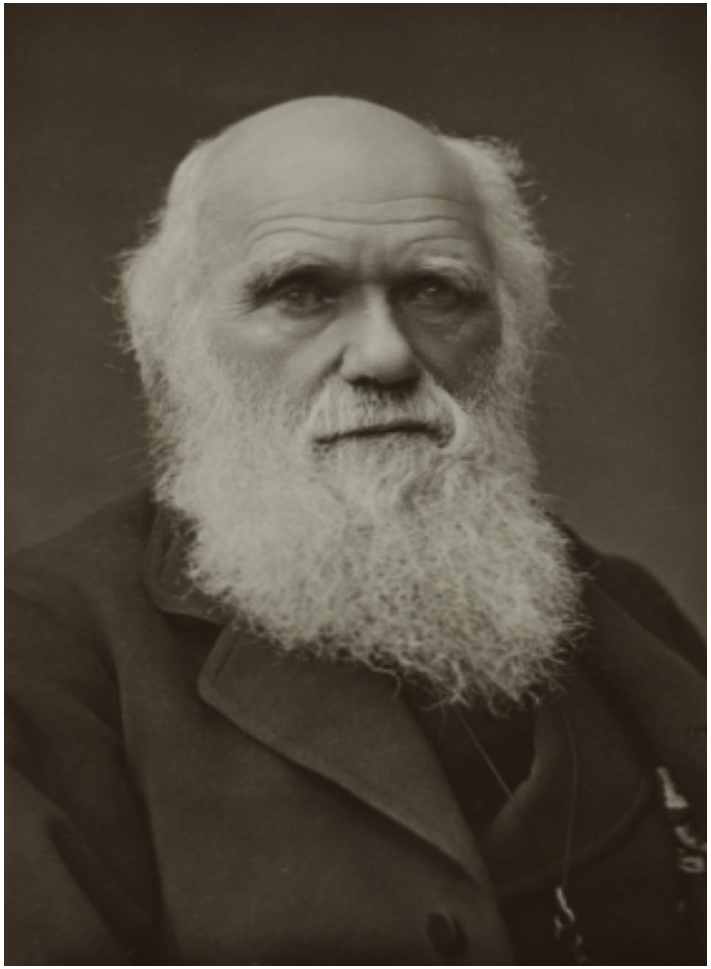
Positioning of EC (1/2)

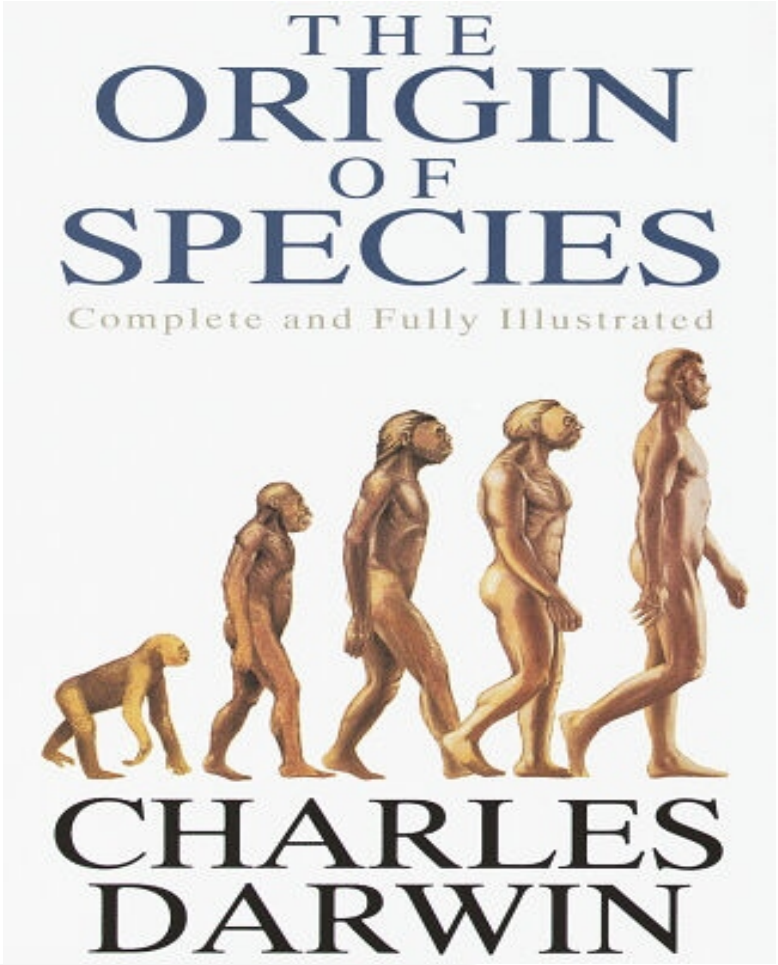


Positioning of EC (2/2)

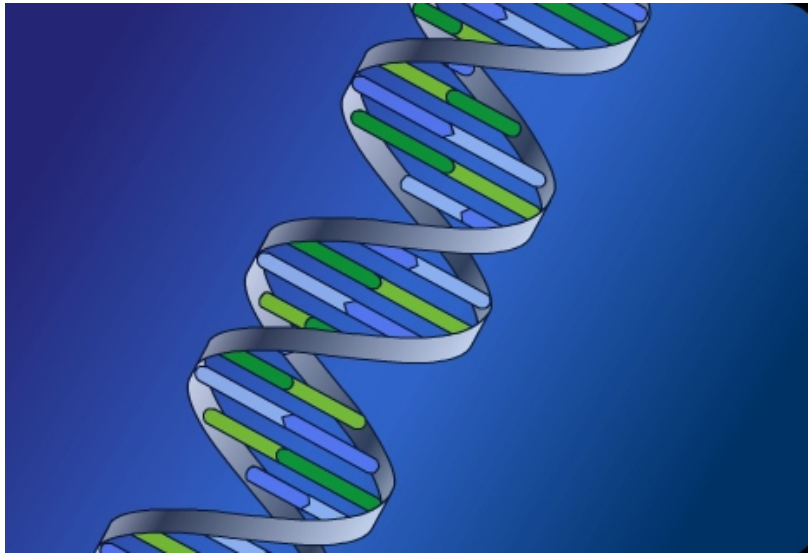
- EC is part of computer science
- EC is not part of life sciences/biology
- Biology delivered inspiration and terminology
- EC can be applied in biological research

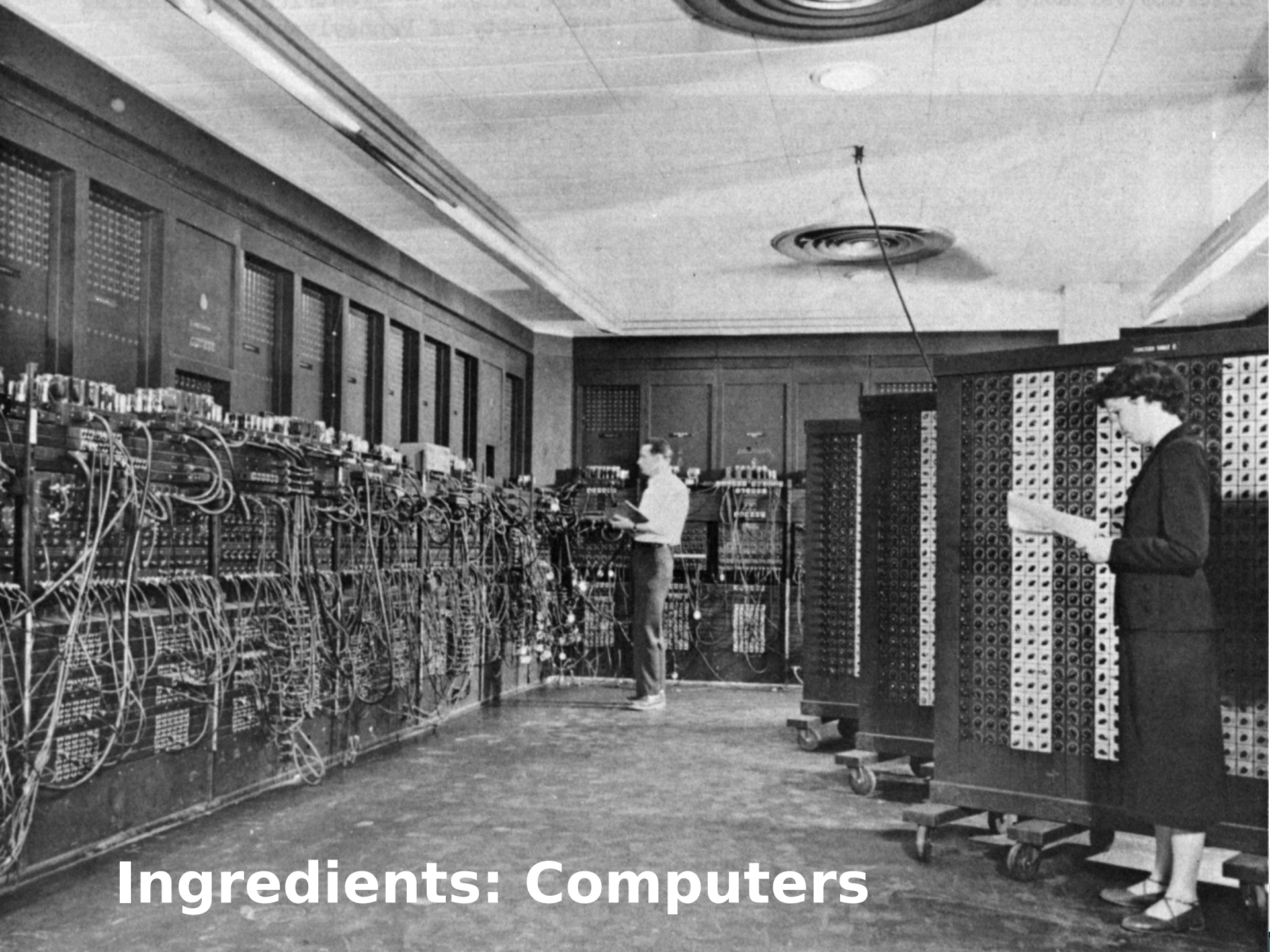
Ingredients: Evolution





Ingredients: Genetics

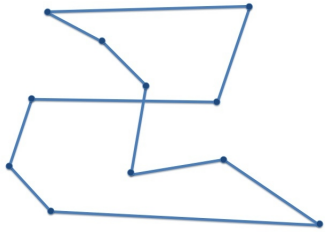




Ingredients: Computers

The EC Metaphor (1/2)

Individuals



Natural selection

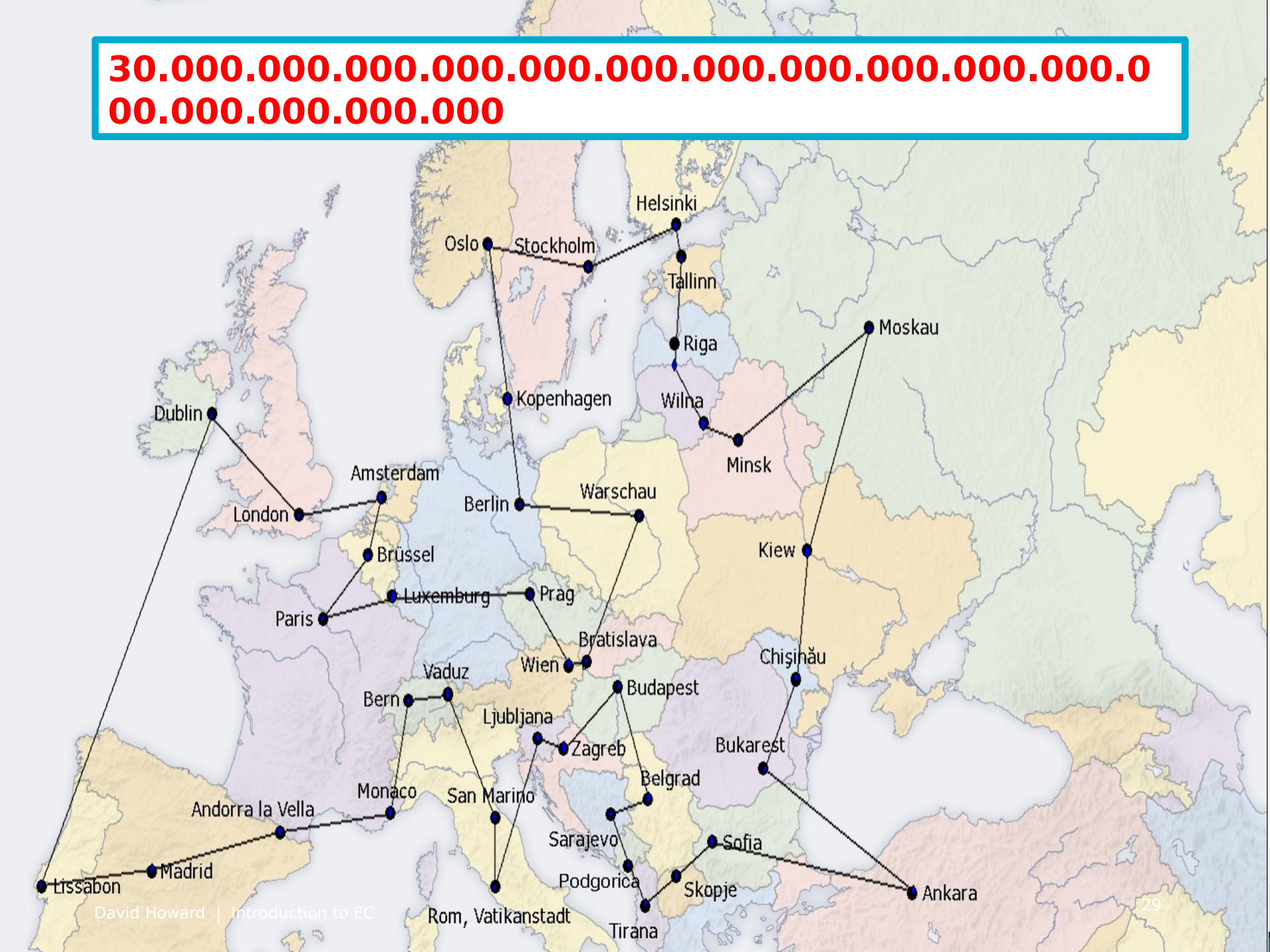
Reproduction

digital sex

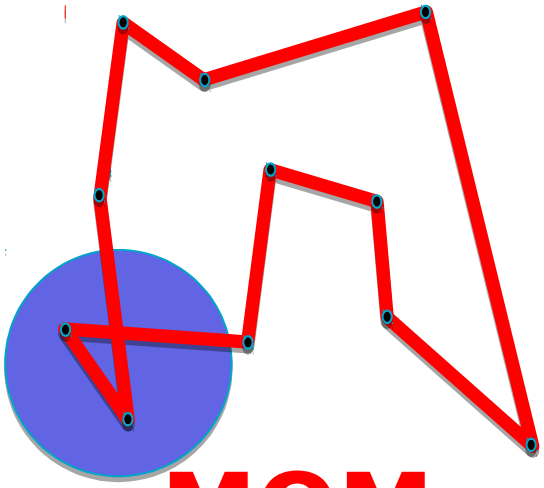
TSP example :

- Problem:
  - Given n cities
  - Find a complete tour with minimal length
- Encoding?

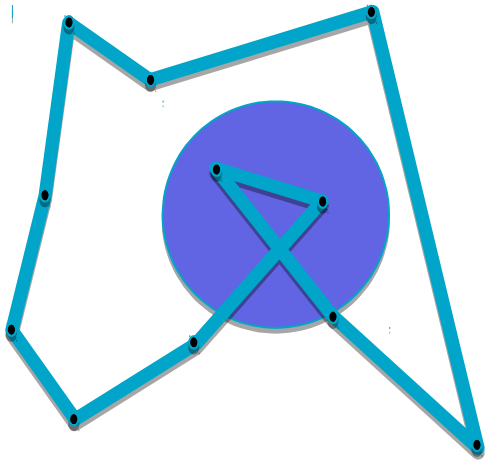
30.000.000.000.000.000.000.000.000.000.000.000.000.000.000.000



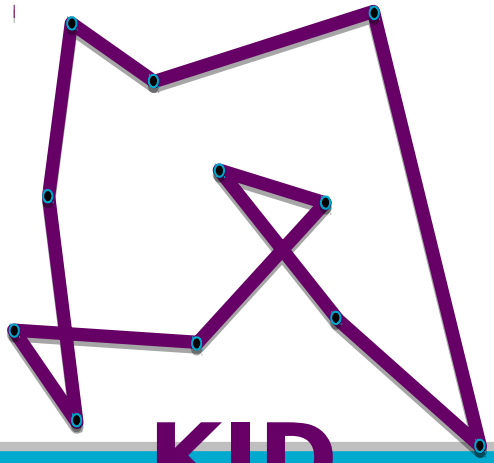
The EC Metaphor (2/2)



MOM



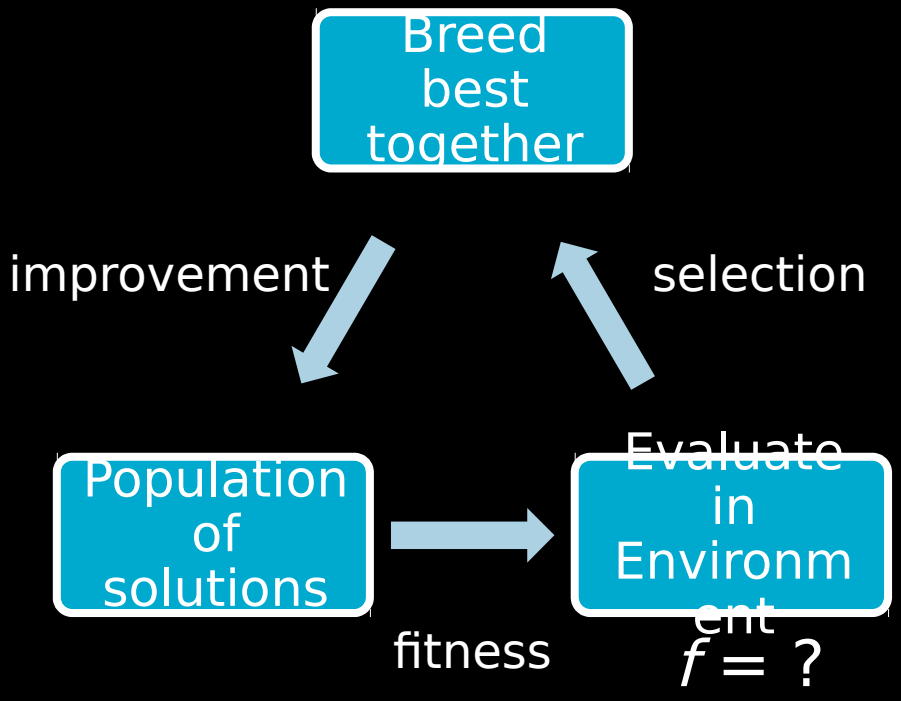
DAD



KID

Evolutionary algorithms 101

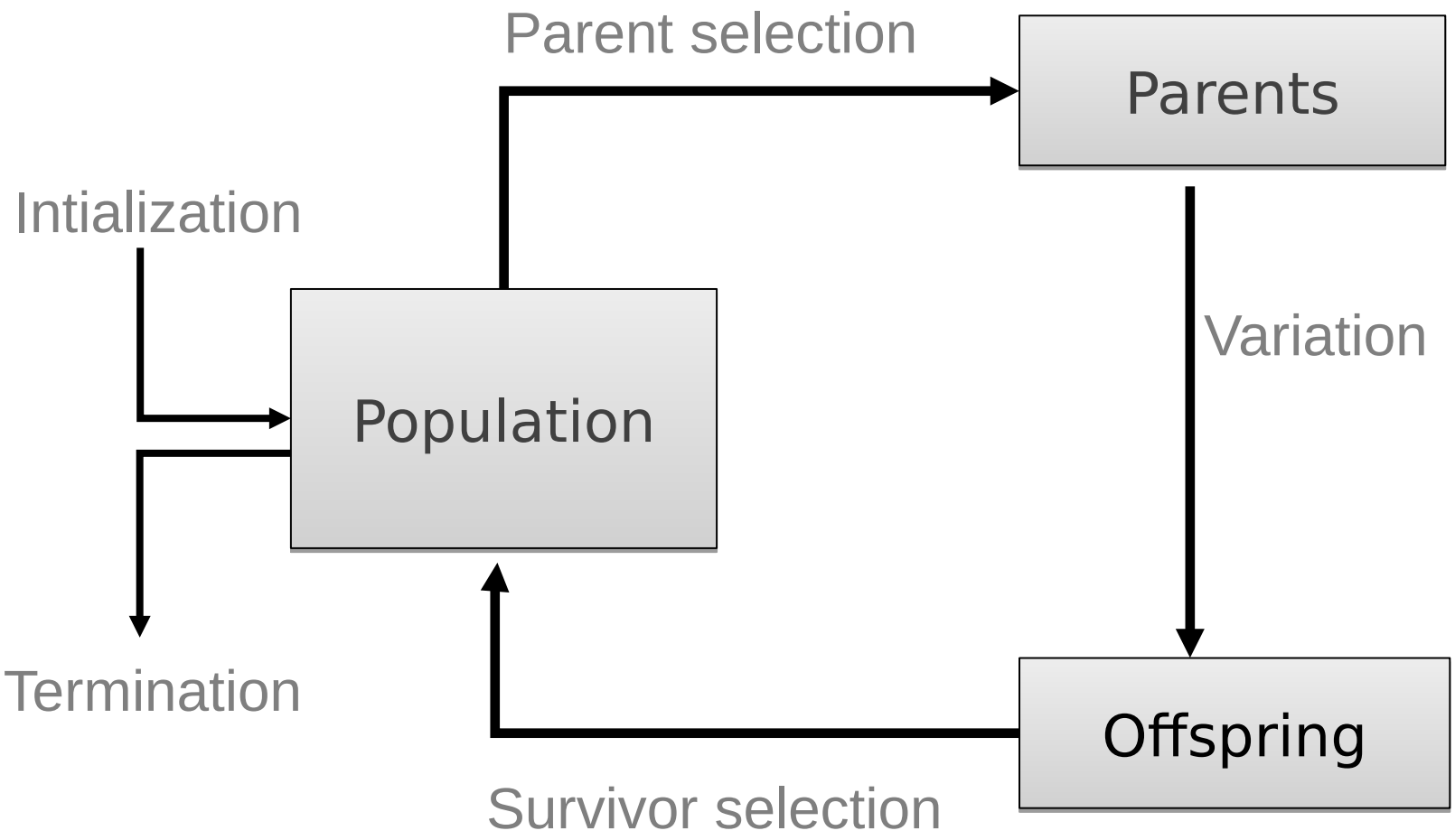
Evolutionary algorithms 101



What is an Evolutionary Algorithm?

- Scheme of an EA
- Main EA components:
  - Representation / evaluation / population
  - Parent selection / survivor selection
  - Recombination / mutation
- Typical EA behaviour

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”

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**

Natural evolution vs. EC

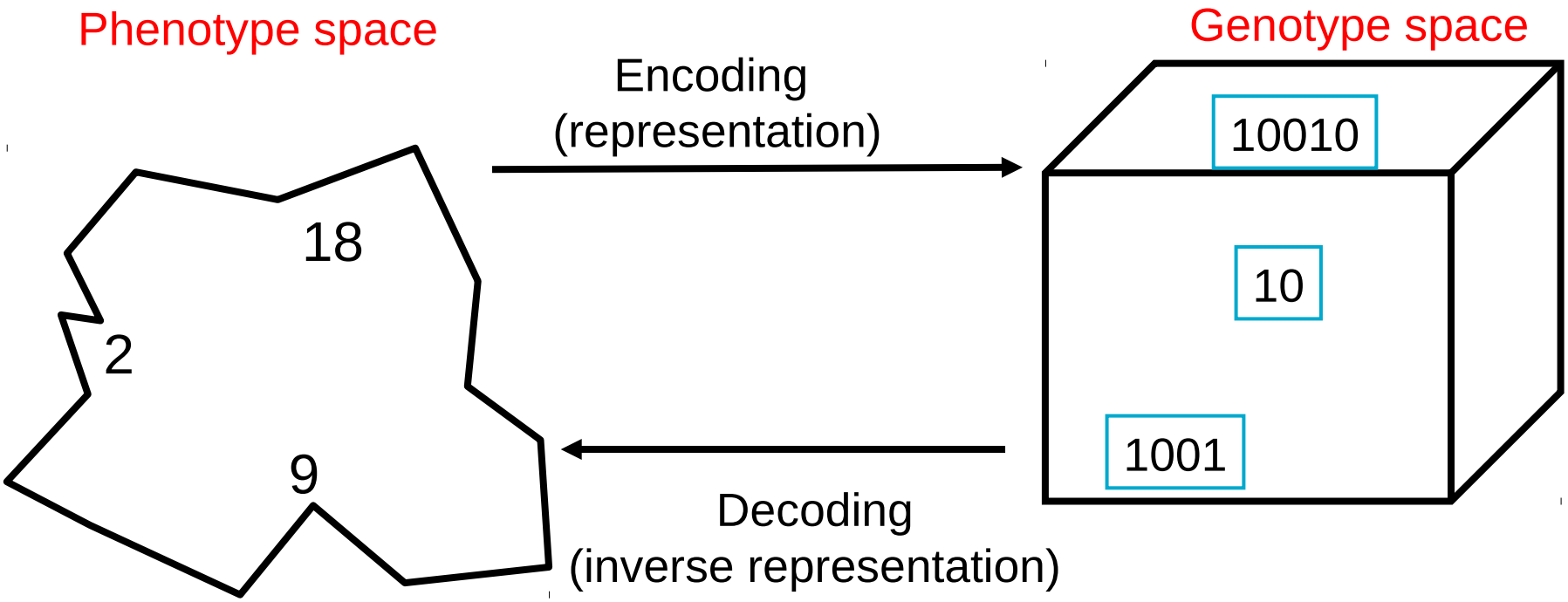
- **Fitness**
  - Natural: observed, 2ndary, “in the eye of the observer” (many offspring fit)
  - EC: predefined, primary (fit many offspring)
- **Execution**
  - Natural: decentralized, asynchronous birth & death, no formal generations
  - EC: centralized, synchronized birth & death, generation cycles
- **Genotype-phenotype mapping**
  - Natural: extremely complex, influenced by environment
  - EC: simple mathematical mapping or (parameterized) procedure
- **Population**
  - Natural: divided in species, spatial niches, population size can vary
  - EC: panmictic (all individuals are potential partners), population size is constant

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
  - 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)

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 (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
- 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
- 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 fitnesses / phenotypes / genotypes present (note: not the same thing)

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
  - even worst in current population usually has non-zero probability of being selected
- This *stochastic* nature can aid escape from local optima

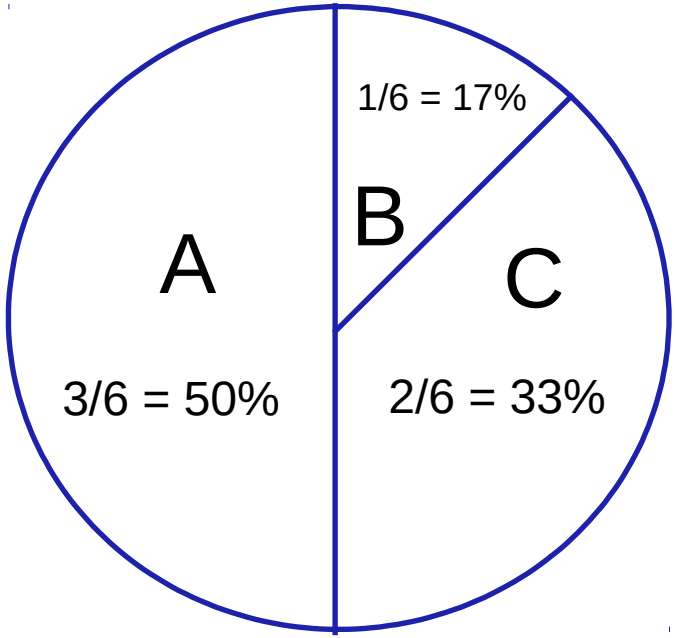
Main EA components: Selection mechanism (2/3)

Example: roulette wheel selection

fitness(A) = 3

fitness(B) = 1

fitness(C) = 2



In principle, any selection mechanism can be used for 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

Main EA components: Variation operators

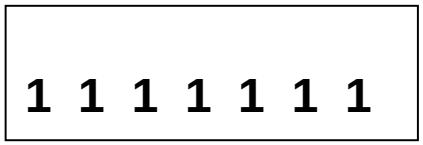
- Role: to generate new candidate solutions
- Usually divided into two types according to their arity (number of inputs):
  - Arity 1 : mutation operators
  - Arity >1 : recombination operators
  - Arity = 2 typically called crossover
  - Arity > 2 is formally possible, seldom used in EC
- There has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

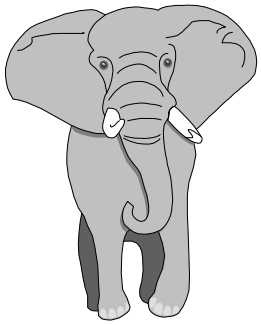
Main EA components: Mutation (1/2)

- Role: causes small, random variance
- 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
  - Evolutionary Programming for continuous variables – only search operator
  - Genetic Programming – hardly used

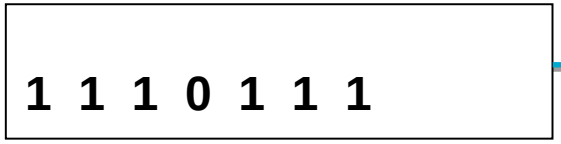
Main EA components: Mutation (2/2)

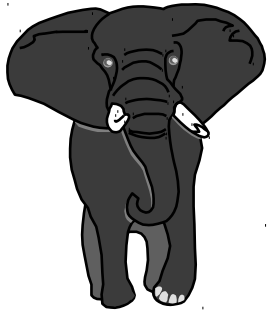
before





after

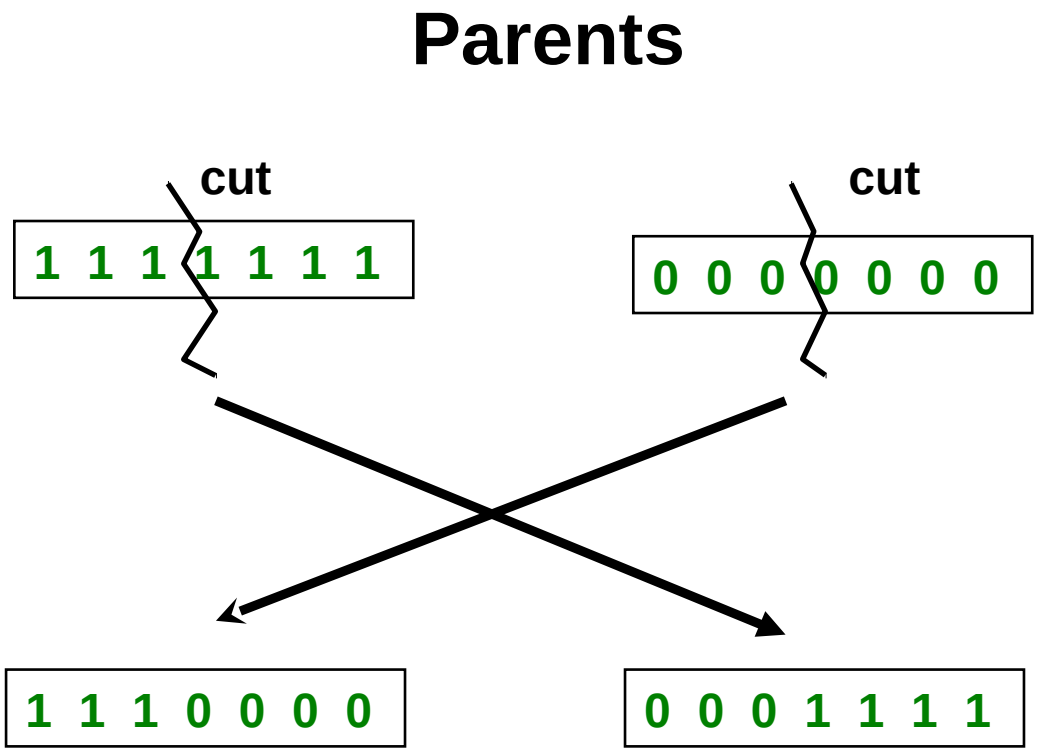


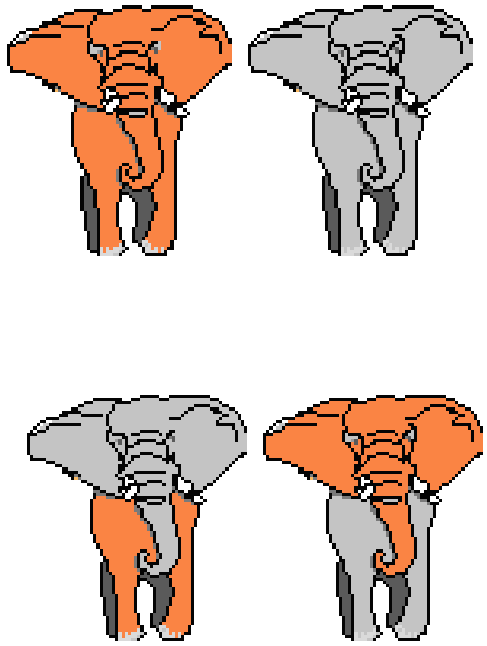


Main EA components: Recombination (1/2)

- 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: Initialisation / Termination

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

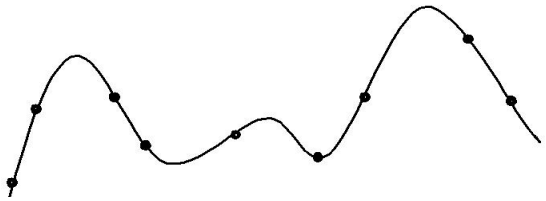
- Termination condition checked every generation
  - Reaching some (known/hoped for) fitness
  - Reaching some maximum allowed number of generations
  - Reaching some minimum level of diversity
  - Reaching some specified number of generations without fitness improvement

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
  - 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

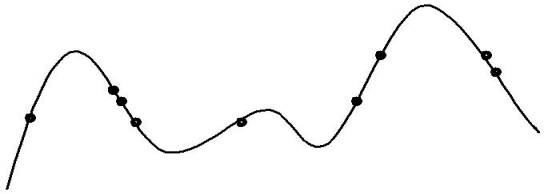
Typical EA behaviour: Stages

Stages in optimising on a 1-dimensional fitness landscape



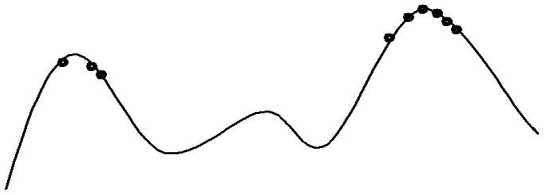
Early stage:

quasi-random population distribution



Mid-stage:

population arranged around/on hills



Late stage:

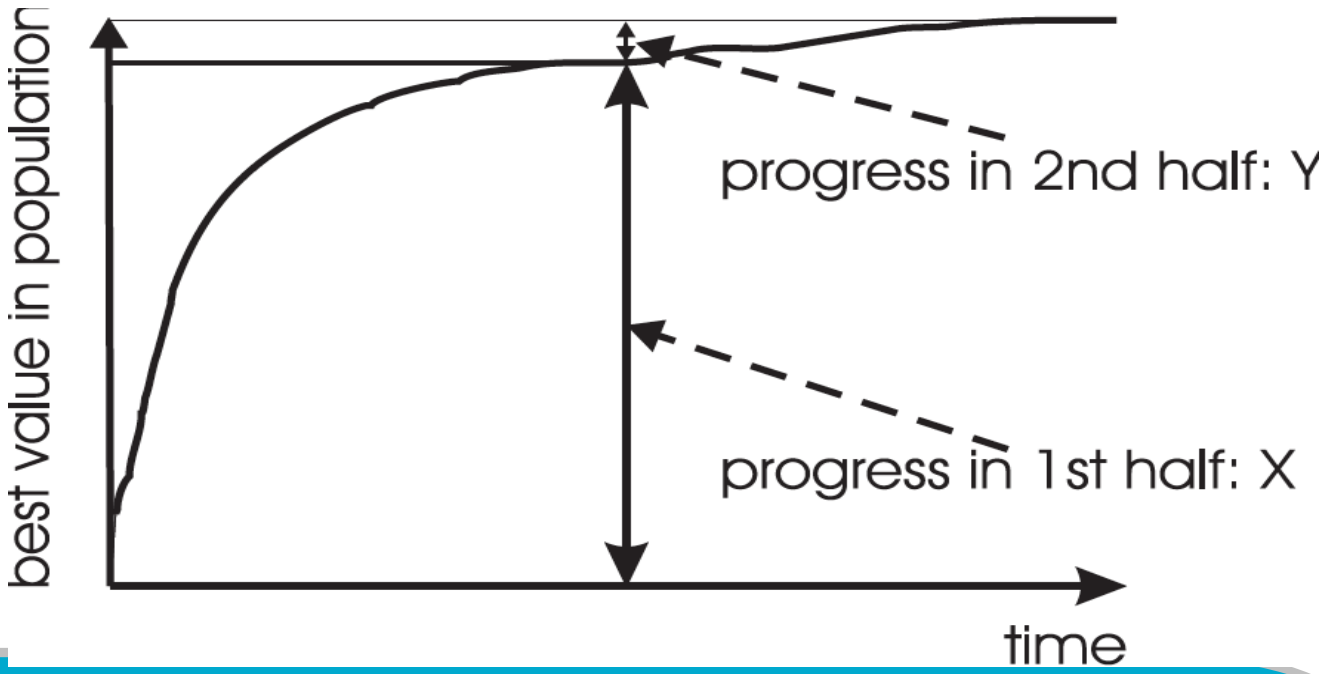
population concentrated on high hills

Typical EA behaviour:
Typical run: progression of fitness

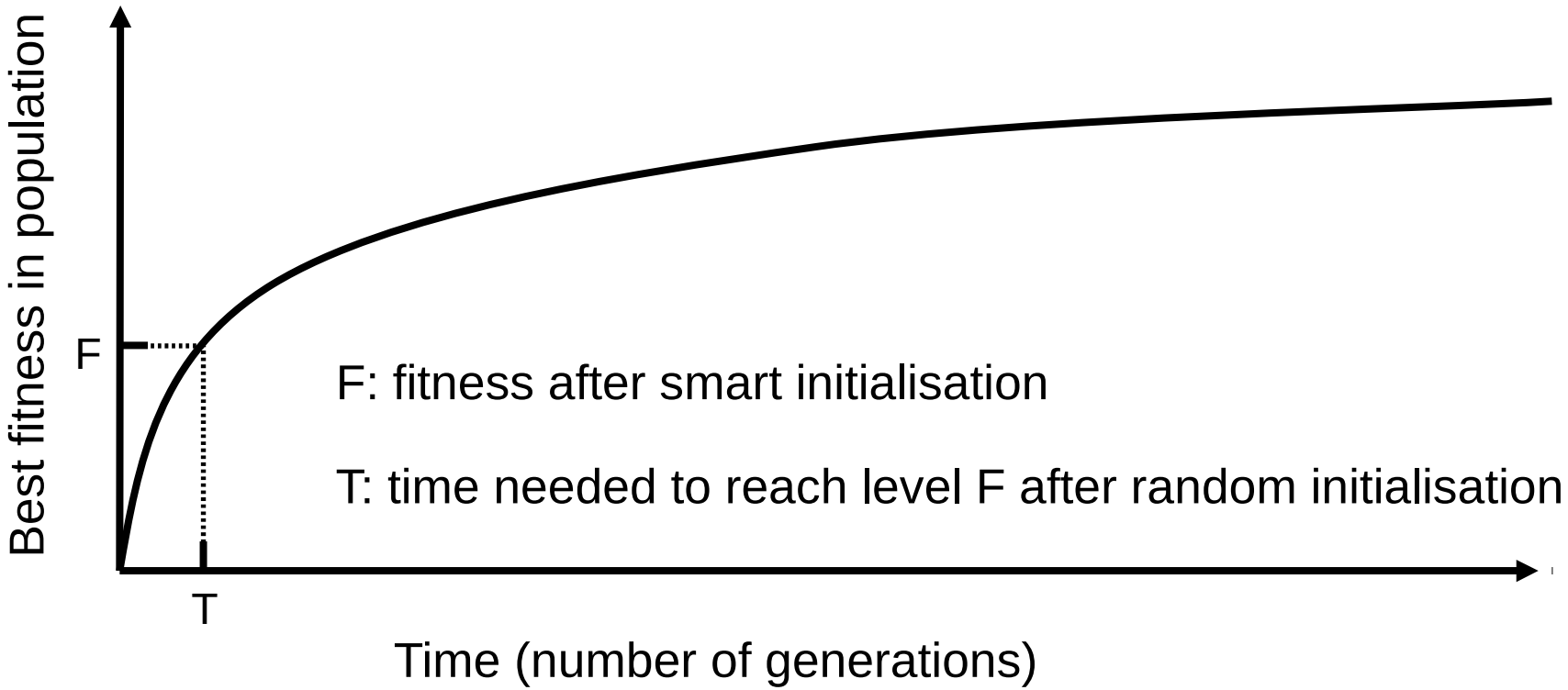
Typical run of an EA shows so-called “anytime behavior”

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?

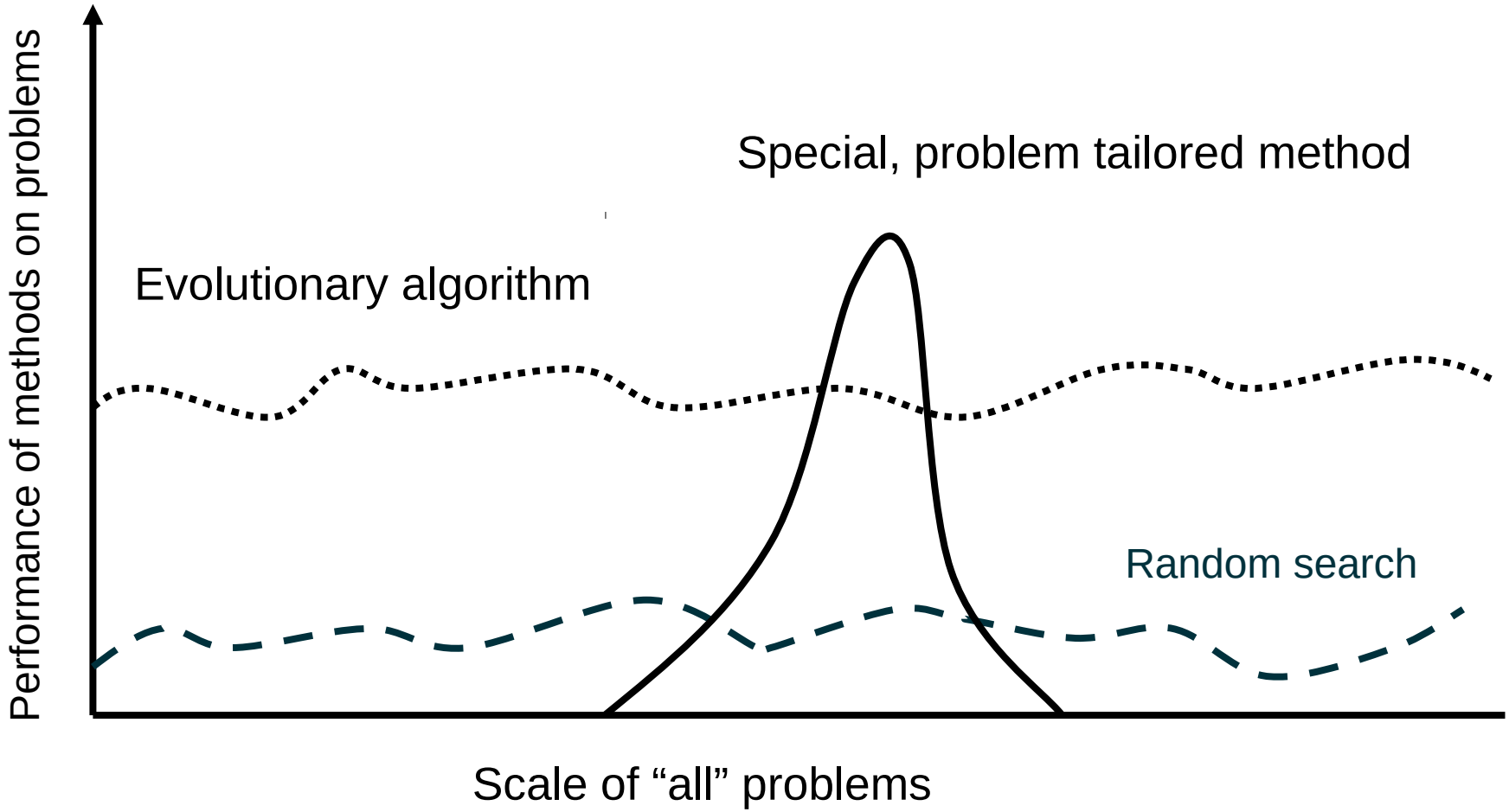


- Answer: it depends.
  - Possibly good, if good solutions/methods exist.
  - Care is needed, we don't want to bias/constrain!

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,
  - 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 behaviour:
EAs as problem solvers: Goldberg view (1989)



Important points:

- There are different ways to visualise an EA scheme
- Variation operators push towards novelty and selection operators push towards quality
- Selection operators are independent from problem but variation operators need to suit need to match the representation
- Selection operators act on the population level and variation operators on the individual level

Representations and search

Representation, Mutation, and Recombination

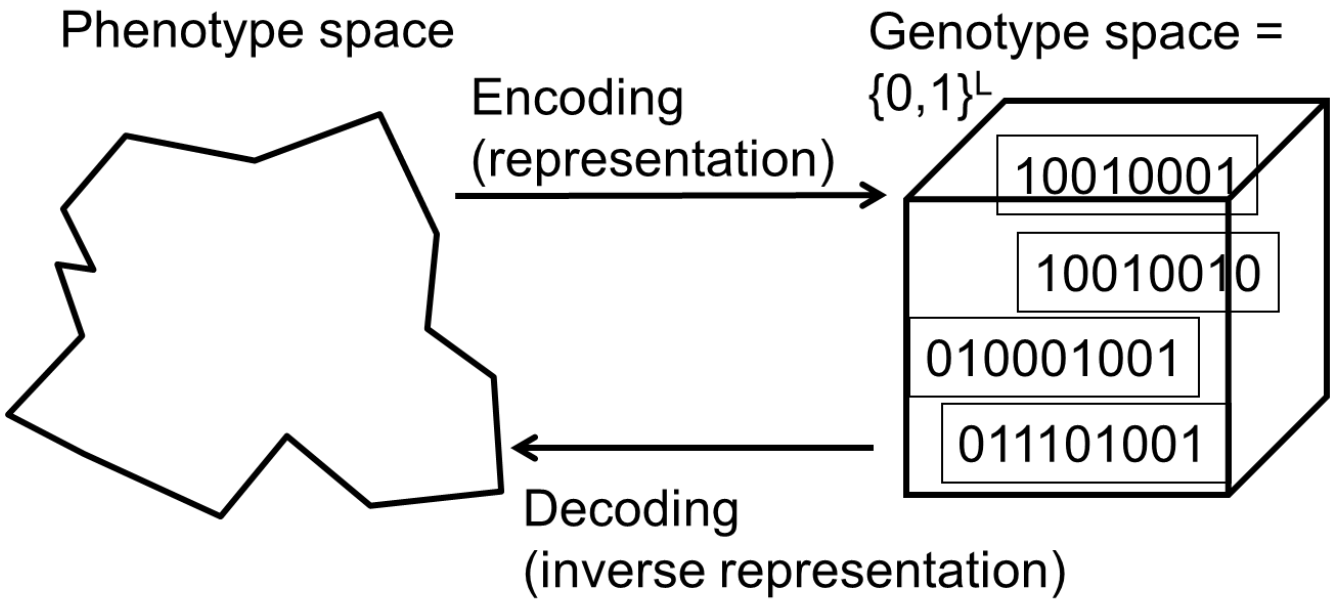
- Role of representation and variation operators
- Most common representation of genomes:
  - Binary
  - Integer
  - Real-Valued or Floating-Point
  - Permutation
  - Tree

Role of representation and variation operators

- First stage of building an EA and most difficult one: choose *right* representation for the problem
- Variation operators: mutation and crossover
- Type of variation operators needed depends on chosen representation

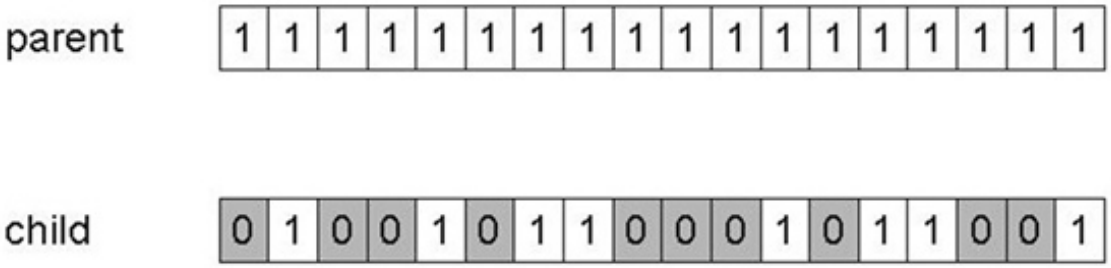
Binary Representation

- One of the earliest representations
- Genotype consists of a string of binary digits



Binary Representation: Mutation

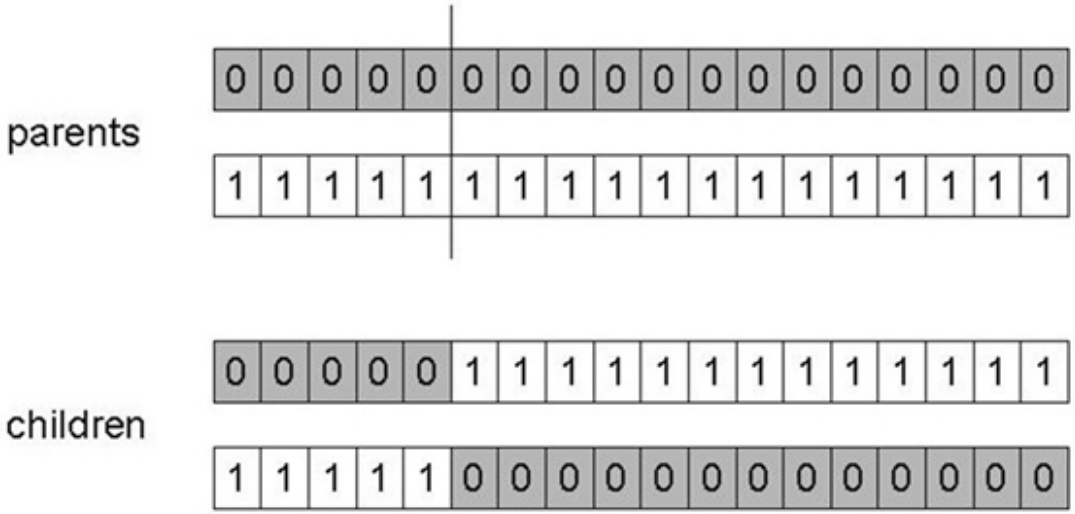
- Alter each gene independently with a probability p_m
- p_m is called the mutation rate
  - Typically between 1/pop_size and 1/ chromosome_length



- Mutation can cause variable effect (use gray coding)

Binary Representation: 1-point crossover

- Choose a random point on the two parents
- Split parents at this crossover point
- Create children by exchanging tails
- P_c typically in range (0.6, 0.9)

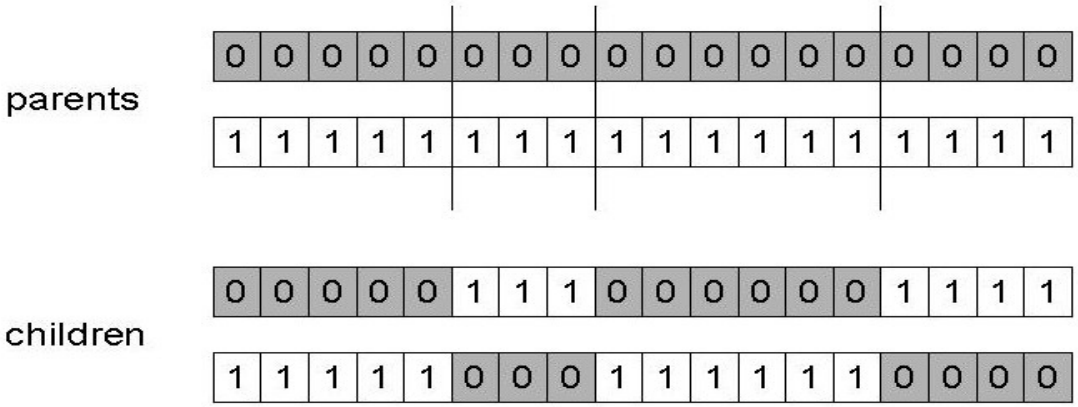


Binary Representation: Alternative Crossover Operators

- Why do we need other crossover(s)?
- Performance with 1-point crossover depends on the order that variables occur in the representation
  - More likely to keep together genes that are near each other
  - Can never keep together genes from opposite ends of string
  - This is known as Positional Bias
  - Can be exploited if we know about the structure of our problem, but this is not usually the case

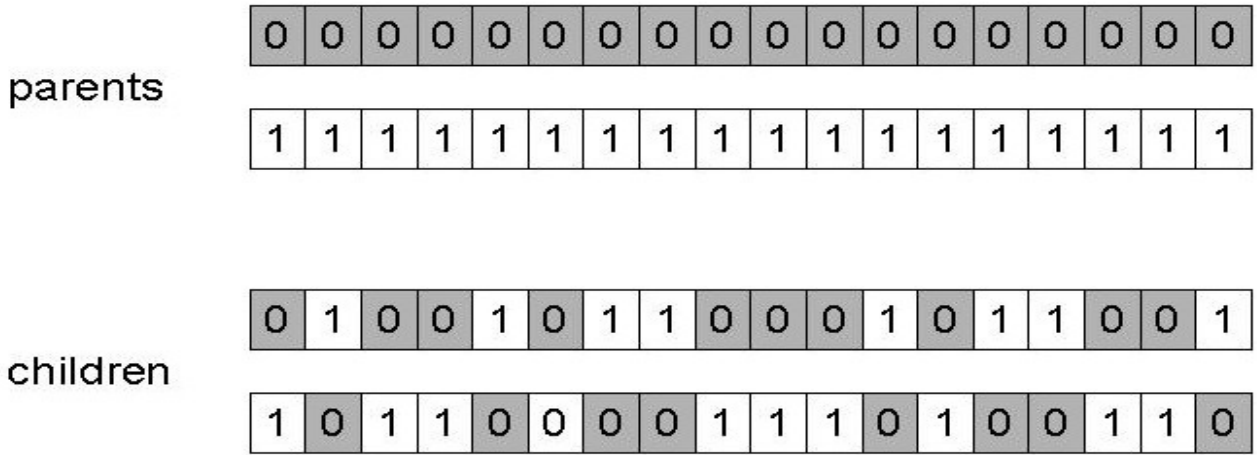
Binary Representation: n-point crossover

- Choose n random crossover points
- Split along those points
- Glue parts, alternating between parents
- Generalisation of 1-point (still some positional bias)



Binary Representation: Uniform crossover

- Assign 'heads' to one parent, 'tails' to the other
- Flip a coin for each gene of the first child
- Make an inverse copy of the gene for the second child
- Inheritance is independent of position



Binary Representation: Crossover OR mutation? (1/3)

- Decade long debate: which one is better / necessary / main-background

- Answer (at least, rather wide agreement):
  - it depends on the problem, but
  - in general, it is good to have both
  - both have another role
  - mutation-only-EA is possible, xover-only-EA would not work

Binary Representation: Crossover OR mutation? (2/3)

Exploration: Discovering promising areas in the search space, i.e. gaining information on the problem

Exploitation: Optimising within a promising area, i.e. using information

There is co-operation AND competition between them

- Crossover is explorative, it makes a *big* jump to an area somewhere “in between” two (parent) areas
- Mutation is exploitative, it creates random *small* diversions, thereby staying near (in the area of) the parent

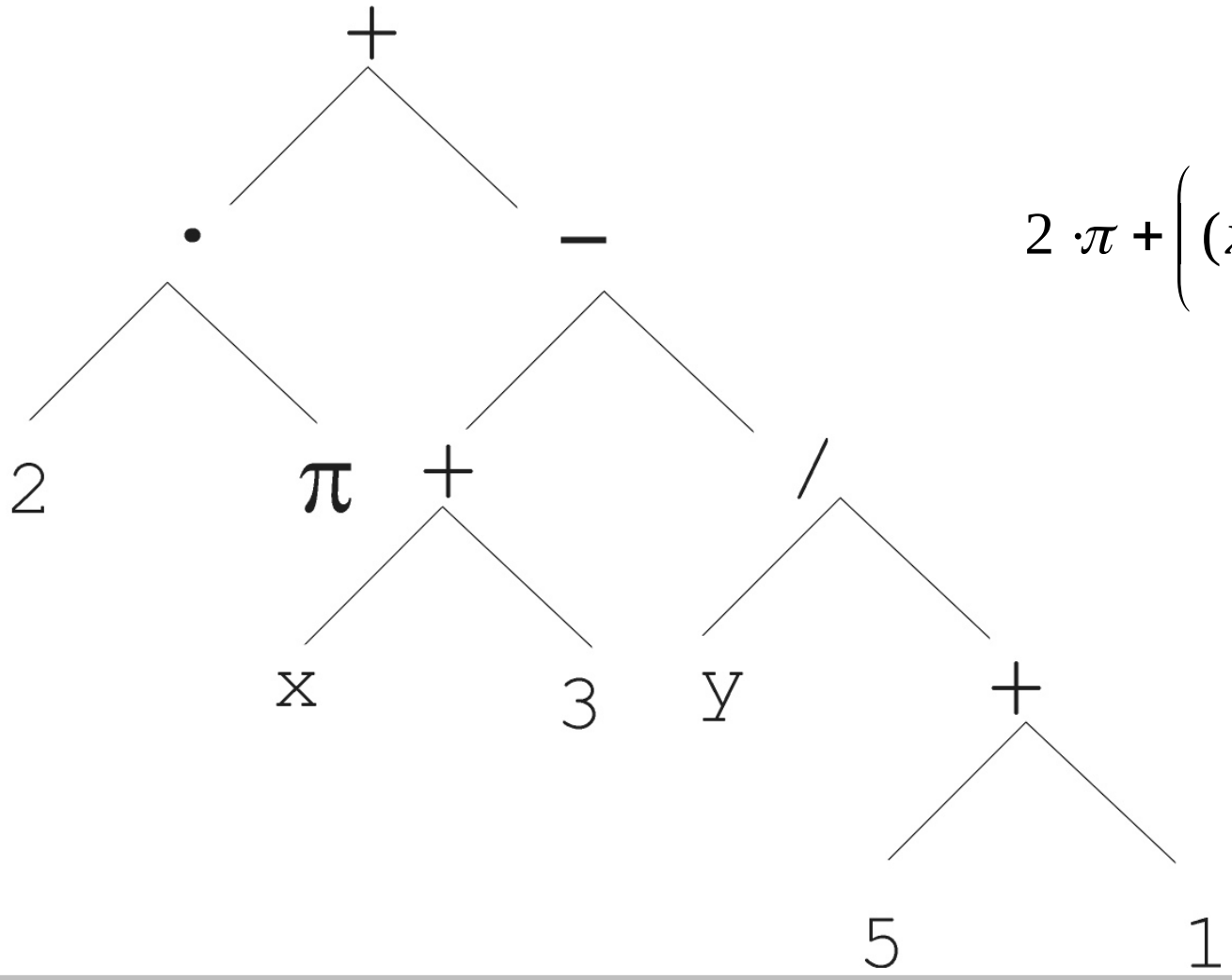
Binary Representation: Crossover OR mutation? (3/3)

- Only crossover can combine information from two parents
- Only mutation can introduce new information (alleles)
- Crossover does not change the allele frequencies of the population (thought experiment: 50% 0's on first bit in the population, ?% after performing n crossovers)
- To hit the optimum you often need a 'lucky' mutation

Integer & real representations

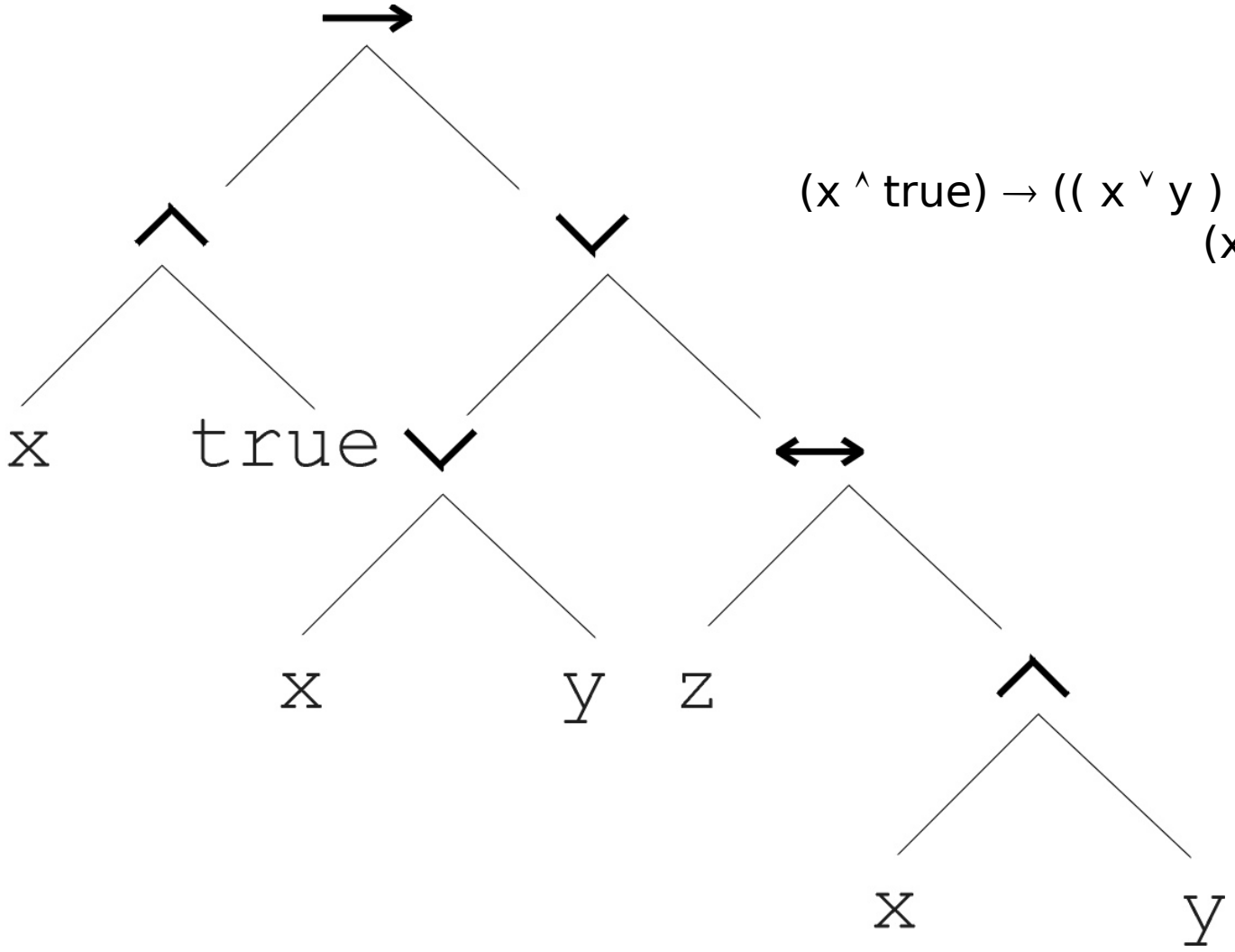
- Nowadays it is generally accepted that it is better to encode numerical variables directly (integers, floating point variables)
- Some problems naturally have integer variables, e.g. image processing parameters
- Others take categorical values from a fixed set e.g. {blue, green, yellow, pink}
- Many real-world problems are real valued
- N-point / uniform crossover operators work
- Mutation must search a range, rather than bit-flipping (integer selection, upper-lower, centre-spread, etc…)
- Same recombination as for binary representation

Tree Representation



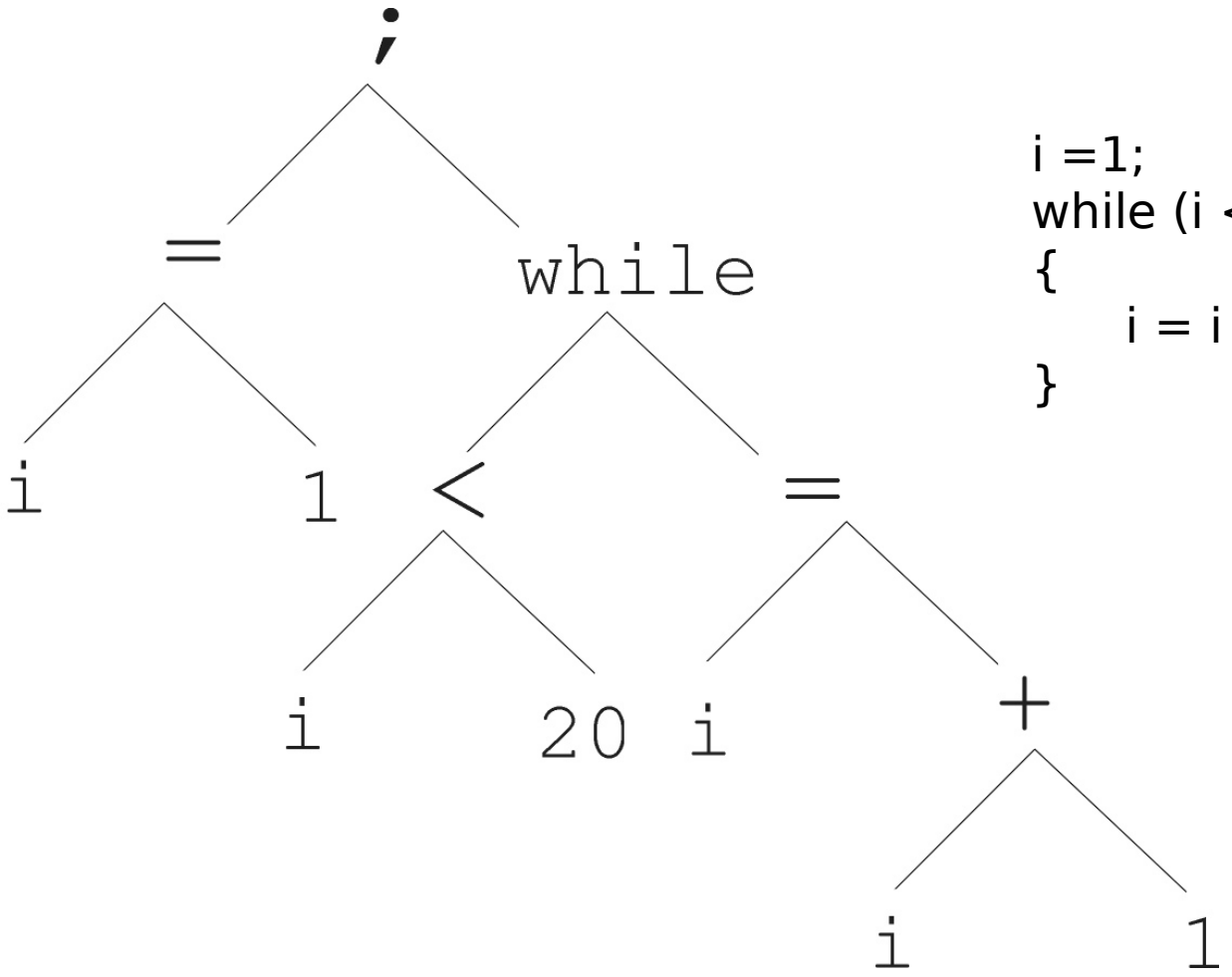
$$2 \cdot \pi + \left((x+3) - \frac{y}{5+1} \right)$$

Tree Representation



$(x \wedge true) \rightarrow ((x \vee y) \vee (z \leftrightarrow (x \wedge y)))$

Tree Representation




```
i =1;
while (i < 20)
{
    i = i +1
}
```

Tree Representation

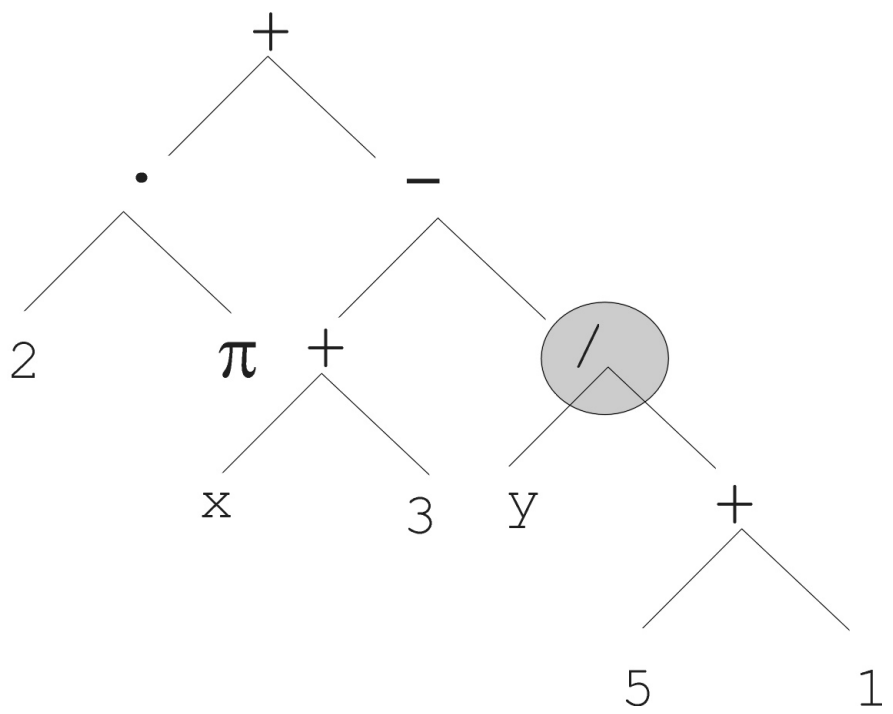
- In GA, ES, EP chromosomes are linear structures (bit strings, integer string, real-valued vectors, permutations)
- Tree shaped chromosomes are non-linear structures
- In GA, ES, EP the size of the chromosomes is fixed
- Trees in GP may vary in depth and width

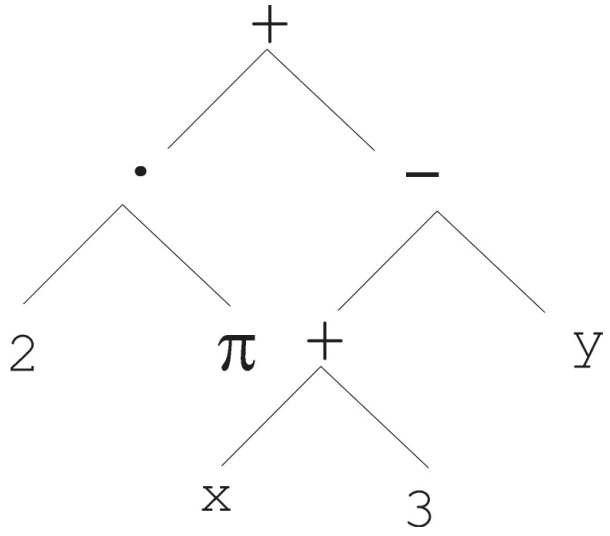
Tree Representation

- Symbolic expressions can be defined by
  - Terminal set T
  - Function set F (with the arities of function symbols)
- Adopting the following general recursive definition:
  - Every $t \in T$ is a correct expression
  - $f(e_1, \ldots, e_n)$ is a correct expression if $f \in F$, arity(f)=n and $e_1, \ldots, e_n$ are correct expressions
  - There are no other forms of correct expressions

Tree Representation: Mutation (1/2)

- Most common mutation: replace randomly chosen subtree by randomly generated tree





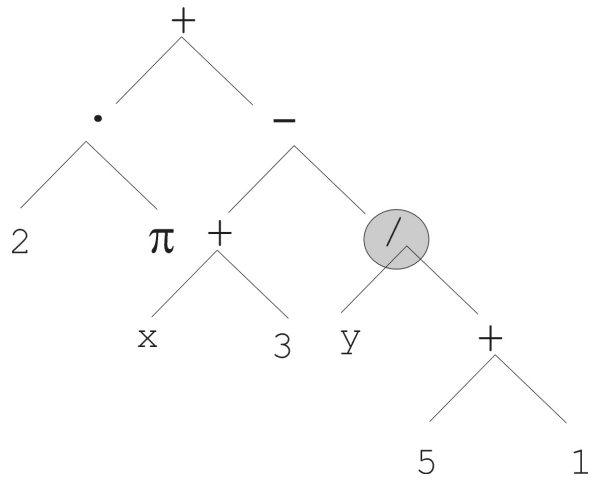
Tree Representation: Mutation (2/2)

- Mutation has two parameters:
  - Probability p_m to choose mutation
  - Probability to chose an internal point as the root of the subtree to be replaced
- The size of the child can exceed the size of the parent

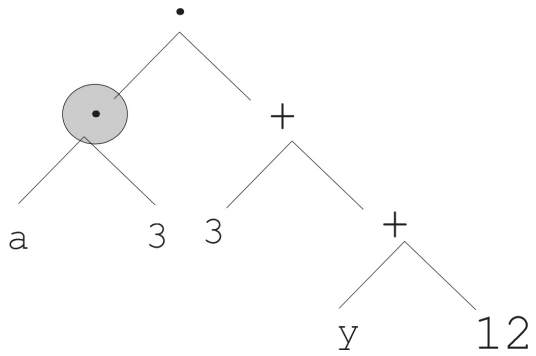
Tree Representation: Recombination (1/2)

- Most common recombination: exchange two randomly chosen subtrees among the parents
- Recombination has two parameters:
  - Probability p_c to choose recombination
  - Probability to chose an internal point within each parent as crossover point
- The size of offspring can exceed that of the parents

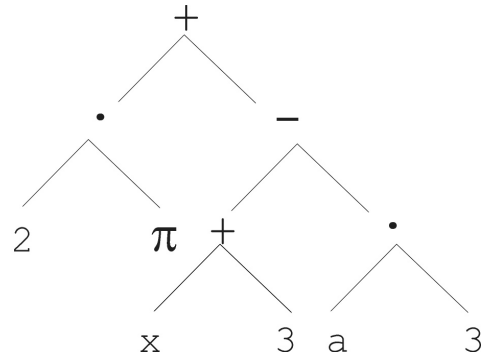
Tree Representation: Recombination (2/2)



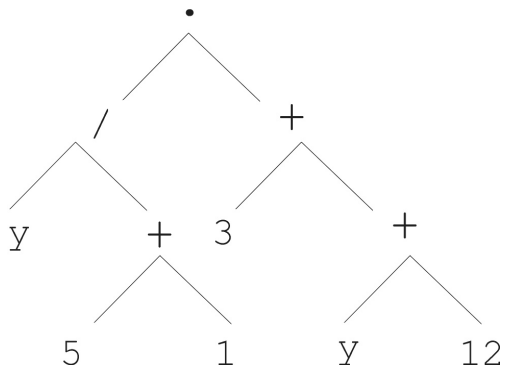
Parent 1



Parent 2



Child 1

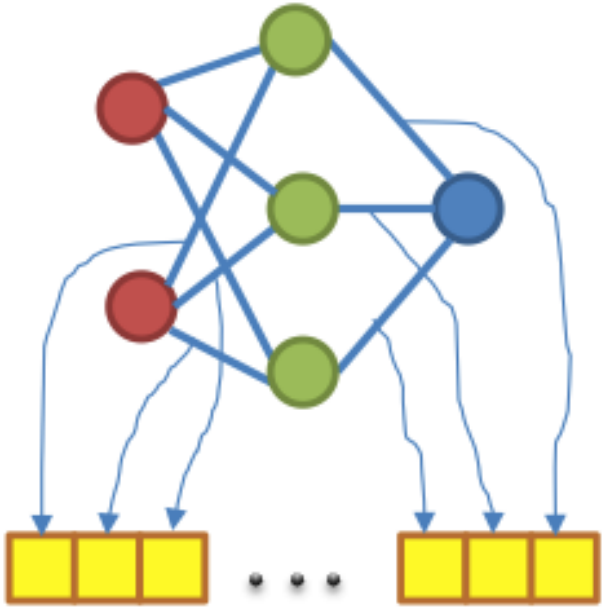


Child 2

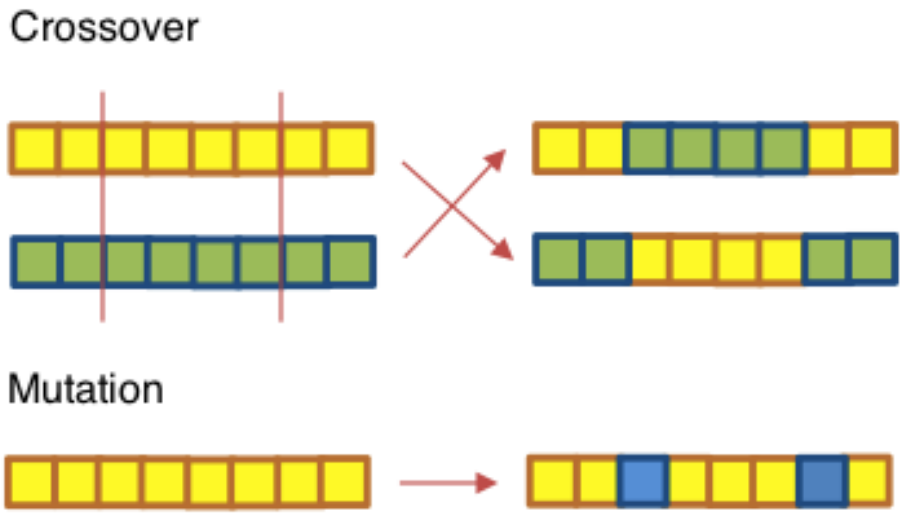
Important points

- Representation is key when designing an EA.
- A handful of data structures is enough to represent many (all?) problems.
- For any given problem there can be more suitable representations. Some are better than others.
- Self-adaptive mutation is a powerful mechanism. But remember to change the sigma first!

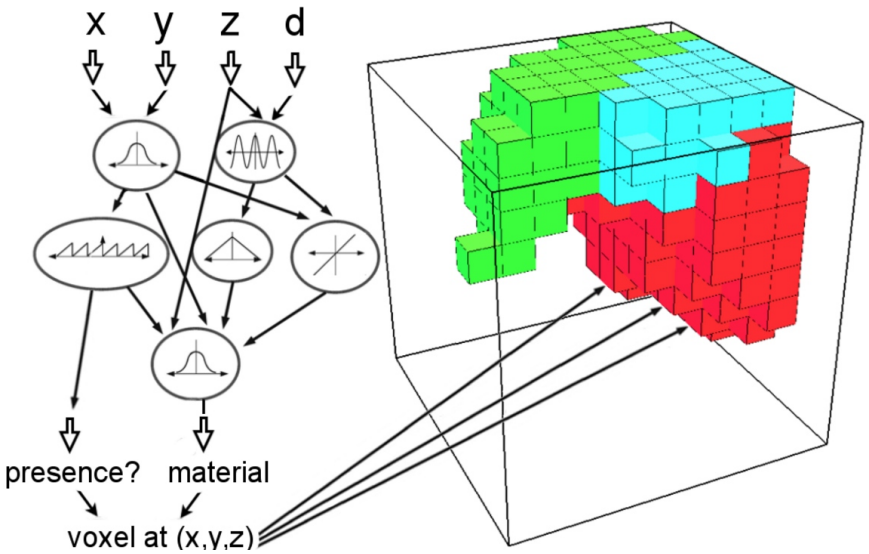
Advanced encodings

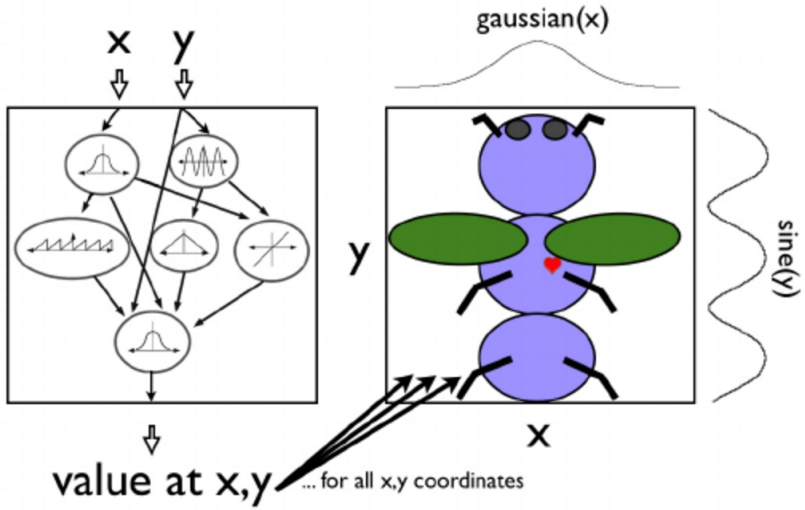


Transforming neural network into linear chromosome



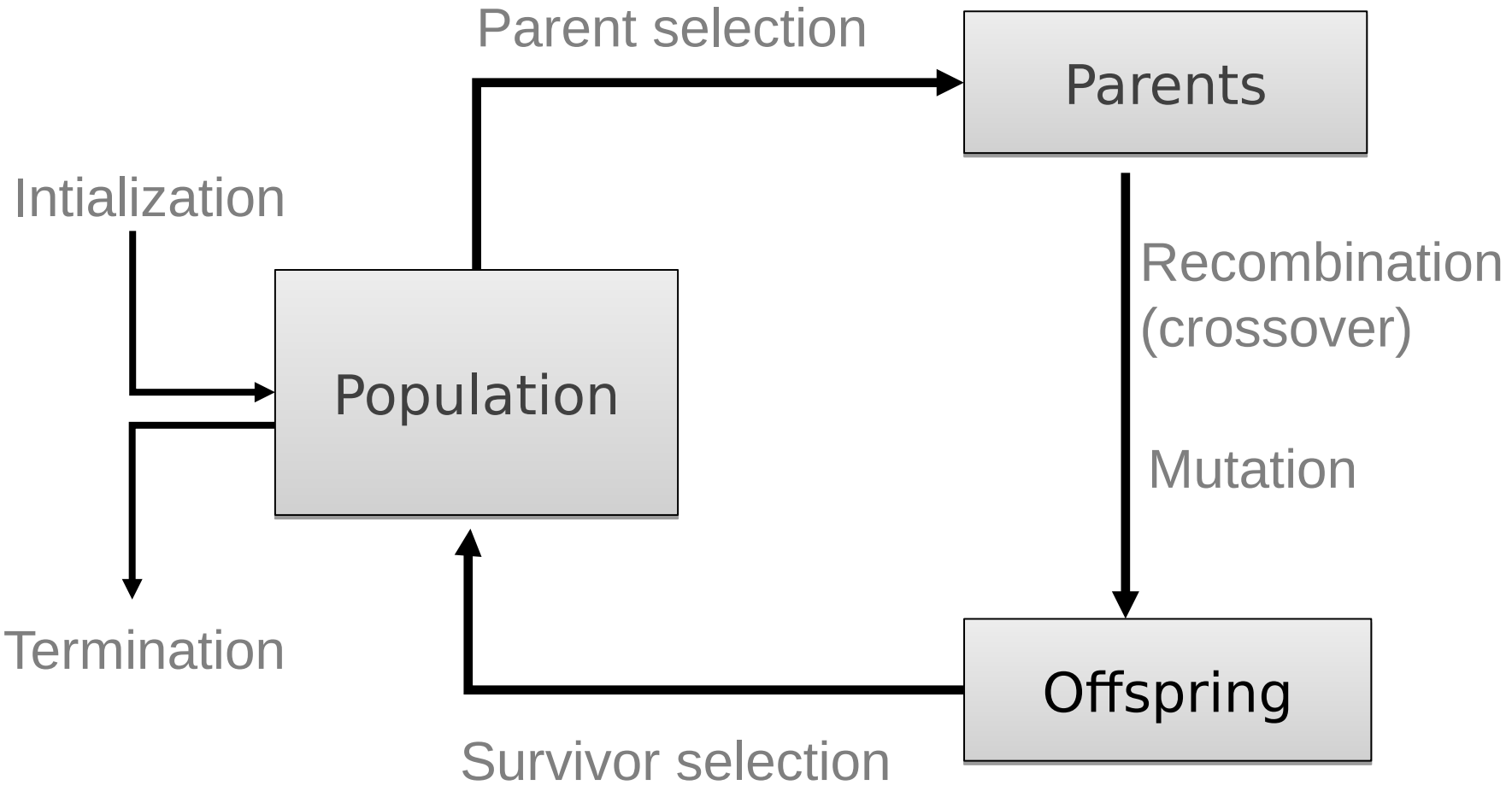
Compositional pattern-producing networks





Fitness, selection, population management

Scheme of an EA: General scheme of EAs



How to design a fitness function

- What might good fitness functions be for:

- **EASY**:
- Classification of a series of characters

- **HARDER**:
- Exploration of an environment? (reward a high level behaviour, not easily defined)
- Stable hover of a helicopter (compound weighting)

- **IMPOSSIBLE** (?):
- Human behaviour?

Population Management Models: Introduction

- Two different population management models exist:
  - Generational model
    - each individual survives for exactly one generation
    - the entire set of parents is replaced by the offspring
  - Steady-state model
    - a few offspring are generated per generation
    - a few members of the population are replaced
- Generation Gap
  - The proportion of the population replaced

Population Management Models: Fitness based competition

- Selection can occur in two places:
  - Selection from current generation to take part in mating (**parent selection**)
  - Selection from parents + offspring to go into next generation (**survivor selection**)

- Selection operators work on whole individuals
  - i.e. they are representation-independent !

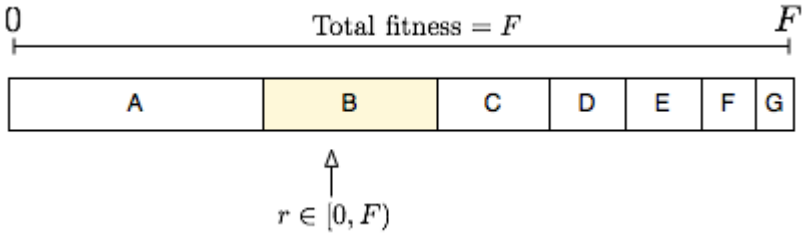
- Distinction between selection
  - Operators: define selection probabilities
  - Algorithms: define how probabilities are implemented

Parent Selection: Fitness-Proportionate Selection

- Probability for individual i to be selected for mating in a population size μ with FPS is

$$P_{FPS}(i) = f_i \Big/ \sum_{j=1}^{\mu} f_j$$

- Problems include
  1. One highly fit member can rapidly take over if rest of population is much less fit: Premature Convergence
  2. At the end of runs when fitnesses are similar, loss of selection pressure
  3. Highly susceptible to function transposition



FPS is also known as *roulette wheel selection*

Parent Selection: Fitness-Proportionate Selection 2

individual	Fitness for f	Sel. Prob. for f	Fitness for f+10	Sel. Prob. for f + 10	Fitness for f+100	Sel. Prob. for f + 100
A	1	0.1	11	0.275	101	0.326
B	4	0.4	14	0.35	104	0.335
C	5	0.5	15	0.375	105	0.339
SUM	10	1	40	1	310	1

Parent Selection: Rank-based Selection

- Attempt to remove problems of FPS by basing selection probabilities on *relative* rather than *absolute* fitness
- Rank population according to fitness and then base selection probabilities on rank
- In a population of size μ the fittest has rank μ-*1* and worst rank 0
- This imposes a sorting overhead on the algorithm, but this is usually negligible compared to the fitness evaluation time

Rank-based Selection: Linear Ranking

$$P_{lin\text{-}rank}(i) = \frac{(2 - s)}{\mu} + \frac{2i(s - 1)}{\mu(\mu - 1)}$$

- Parameterised by factor *s:* $1 < s \leq 2$
  - measures advantage of best individual
- Simple 3 member example

Individual	Fitness	Rank	P_{selFP}	P_{selLR} $(s = 2)$	P_{selLR} $(s = 1.5)$
A	1	0	0.1	0	0.167
B	4	1	0.4	0.33	0.33
C	5	2	0.5	0.67	0.5
Sum	10		1.0	1.0	1.0

Parent Selection: Tournament Selection (1/2)

- All methods above rely on global population statistics
  - Could be a bottleneck especially on structured or very large populations
  - Relies on external fitness function which might not exist: e.g. evolving game players
- Idea for a procedure using only local fitness information:
  - Pick *k* members at random then select the best of these
  - Repeat to select more individuals

Parent Selection: Tournament Selection (2/2)

- Probability of selecting *i* will depend on:
  - Rank of *i*
  - Size of sample *k*
    - higher *k* increases selection pressure
  - Whether contestants are picked with replacement
    - Picking without replacement increases selection pressure
  - Whether fittest contestant always wins (deterministic) or this happens with probability *p*

Survivor Selection

- Select μ individuals to form the next generation from μ parents and λ offspring
- The parent selection mechanisms can also be used for selecting survivors
- Survivor selection can be divided into two approaches:
  - Age-based selection
    - Fitness is not taken into account
    - In SSGA can implement as “delete-random” (not recommended) or as first-in-first-out (a.k.a. delete-oldest)
  - Fitness-based selection

Fitness-based selection (1/2)

- Elitism
  - Always keep at least one copy of the fittest solution so far
  - Widely used in both population models (GGA, SSGA)
  - Also with keep best N individuals or keep best x %
- Delete worst (a.k.a. GENITOR)
  - From Whitley's original Steady-State algorithm (he also used linear ranking for parent selection)
  - Rapid takeover: use with large populations or "no duplicates" policy
- Round-robin tournament
  - P: μ parents, O: λ offspring
  - Pairwise competitions in round-robin format:
    - Each solution x from P ∪ O is evaluated against q other randomly chosen solutions
    - For each comparison, a "win" is assigned if x is better than its opponent
    - The μ solutions with the greatest number of wins are retained to be parents of the next generation
  - Parameter q allows tuning selection pressure
  - Typically $q = 10$

Fitness-based replacement (2/2)

- (μ,λ)-selection a.k.a. "comma strategy"
  - based on the set of children only ($\lambda > \mu$)
  - choose best μ
- $(\mu+\lambda)$-selection a.k.a. "plus strategy"
  - based on the set of parents and children
  - choose best μ

- Often (μ,λ)-selection is preferred for:
  - Better in leaving local optima
  - Better in following moving optima
  - Using the + strategy bad σ values can survive in $\langle x,\sigma \rangle$ too long if their host x is very fit
- $\lambda \approx 7 \cdot \mu$ is a traditionally good setting (decreasing over the last couple of years, $\lambda \approx 3 \cdot \mu$ seems more popular lately)

Selection Pressure

- Takeover time τ^* is a measure to quantify the selection pressure
- The number of generations it takes until the application of selection completely fills the population with copies of the best individual
- Goldberg and Deb showed:

$$\tau^* = \frac{\ln \lambda}{\ln(\lambda / \mu)}$$

- Specific values for
  - an evolution strategy with $\mu = 15$ and $\lambda = 100$: $\tau^* \approx 2$
  - a GA with and proportional selection: $\tau^* = 460$

Summary

Evaluate population of individuals vs. some fitness function.

Select some parents, copy them, mutate and crossover, put them in the population

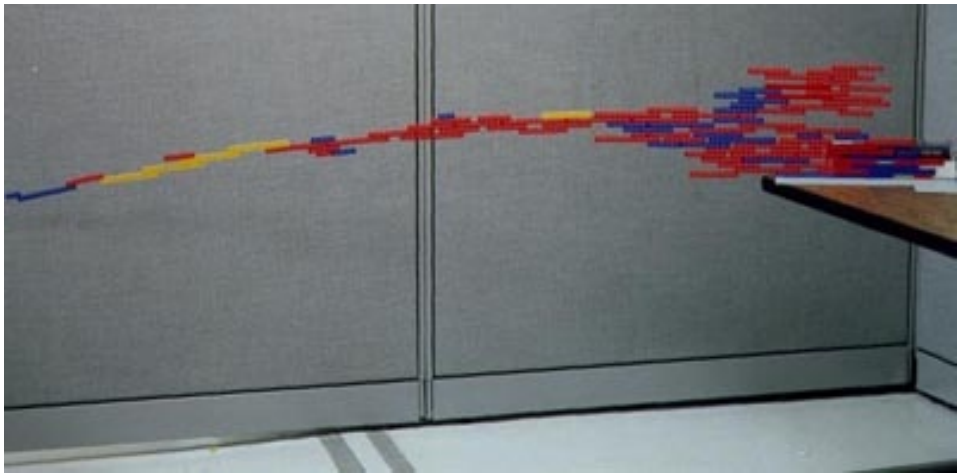
Children are a bit like parents, randomly better or worse but some will eventually be better!

Delete unfit individuals

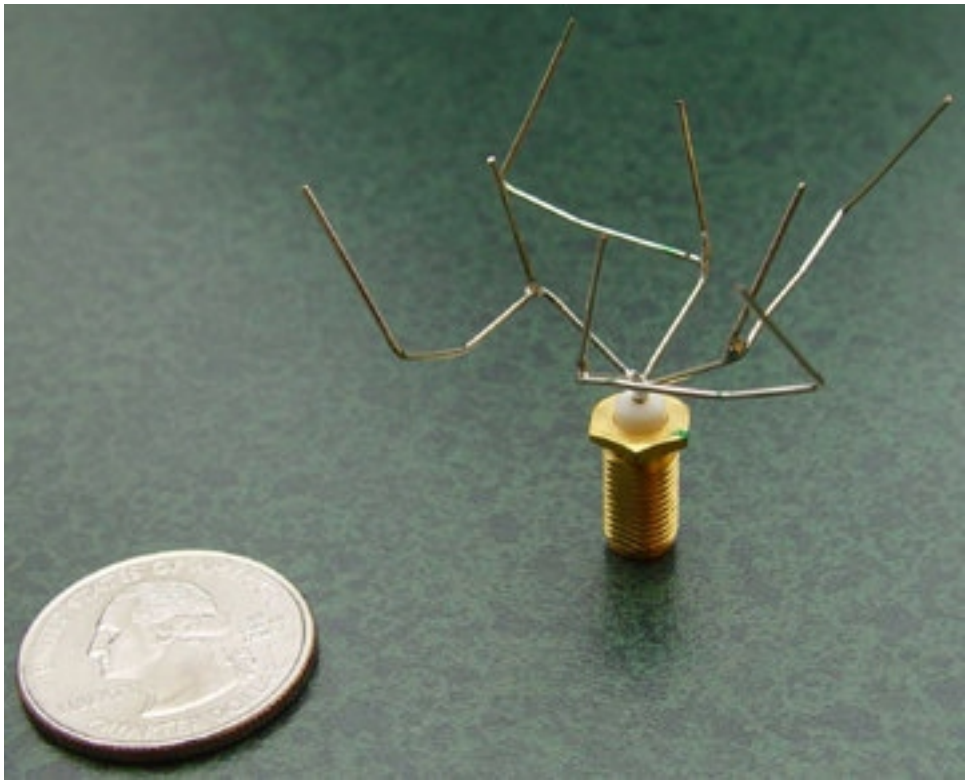
Population gradually improves!

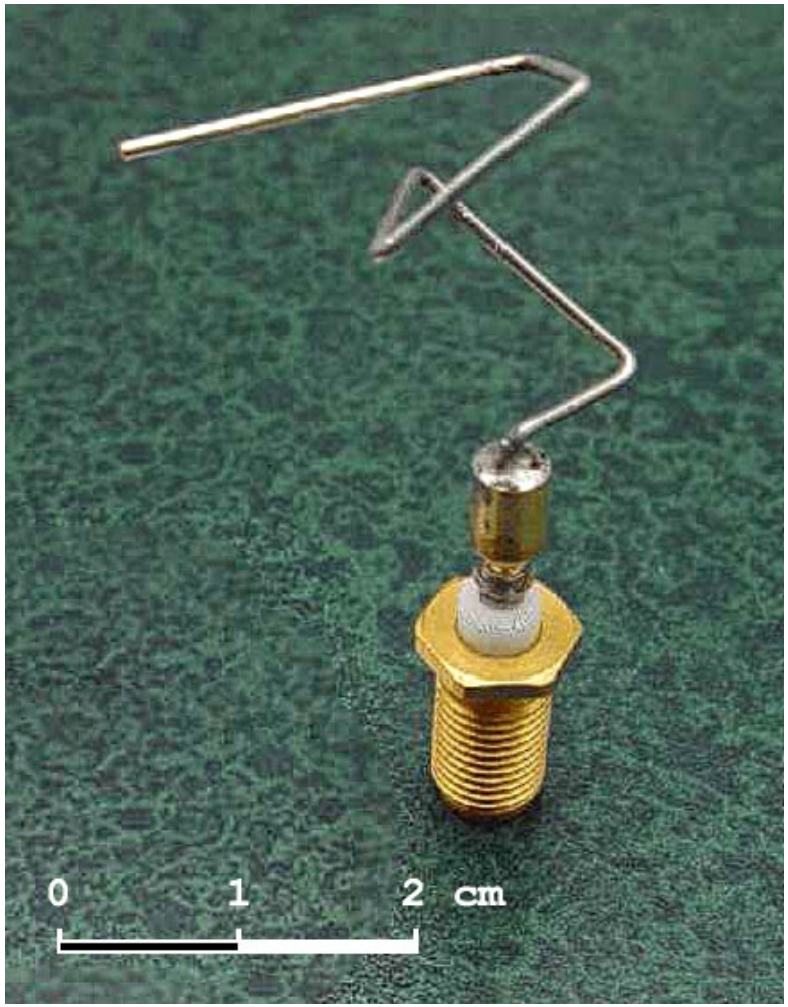
Applications of Evolutionary Algorithms

Creative designs



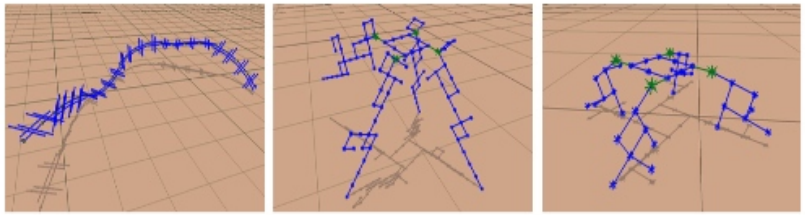
Interactive evolutionary art




0
1
2 cm

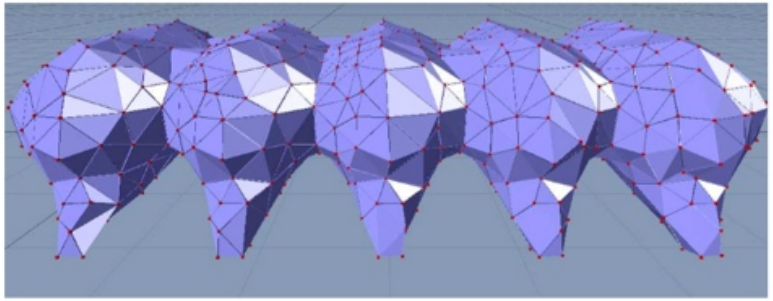
A

B

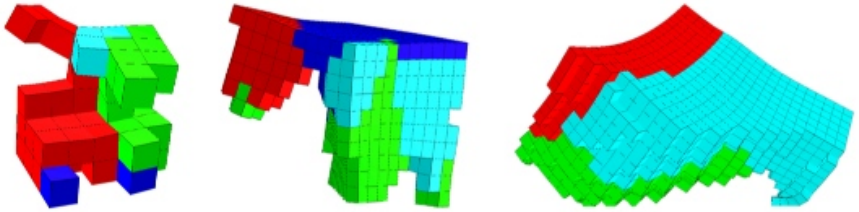


C

D



E



F