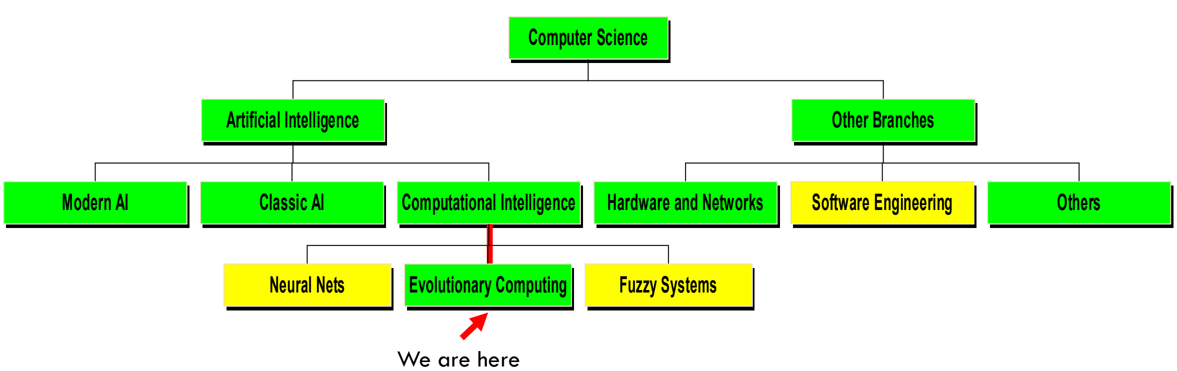
CHAPTER 2 – SECTION 1
EVOLUTIONARY COMPUTING

Siamak Sarmady, UUT – (Based on the slides from A.E. Eiben)

Locating Evolutionary Computing (and GA) within CS

- Genetic algorithm is from a family of algorithms called evolutionary algorithms.
- Evolutionary Algorithms themselves are considered as metaheuristic algorithms.
- Biology has been the inspiration and EC can sometimes be useful in biological research too.



Brief History

- 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

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

2003:

- 3 major and 10 smaller related ones
- 3 scientific core EC journals
- 750-1000 papers published just in 2003
- Many applications

Darwinian Evolution 1: Survival of the fittest

- All environments have finite resources (i.e., can only support a limited number of individuals)
- Creatures have a basic instinct that leads them towards reproduction
- In order to reproduce, some kind of selection of a mate is inevitable
- Those that compete better for the resources have better chance of reproduction

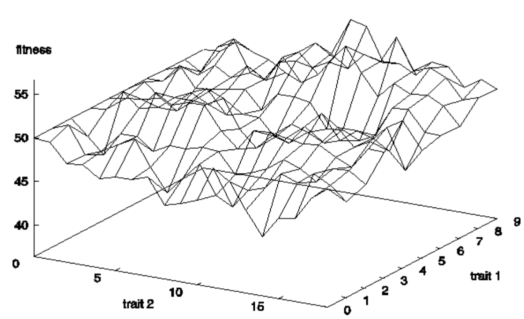
- **Note:** Humans assume, individuals that have many offspring (or chance of having better offspring) have a high fitness…

Darwinian Evolution 2: Evolution

- Population consists of diverse set of individuals
- Combinations of traits that are better, tend to increase in population (better traits propagate to the population using combination)
- Variations occur through random changes yielding constant source of diversity (evolution is caused by the random variations)

Adaptive landscape metaphor (Wright, 1932)

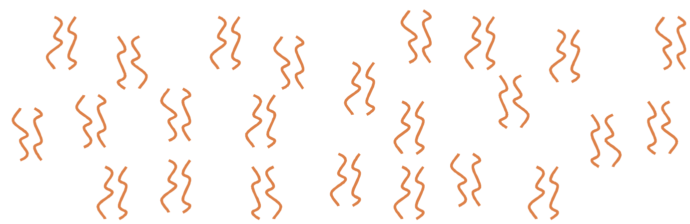
- We can consider that the population with n traits to be in a n+1-dimensional landscape with height corresponding to fitness…
- Being in a point (having some specific traits) determines the fitness of an individual.
- Each different individual (phenotype) represents a single point on the landscape
- Population is therefore a "cloud" of points, moving on the landscape over time as it evolves
- **Selection and combination:** "pushes" population up the landscape (distributes good features and therefore increases the fitness of the population)
- **Genetic drift:**
  - Random variations in features can cause the population "melt down" hills, thus crossing valleys and leaving local optima…



Example with two traits

Chromosomes in Homo Sapiens

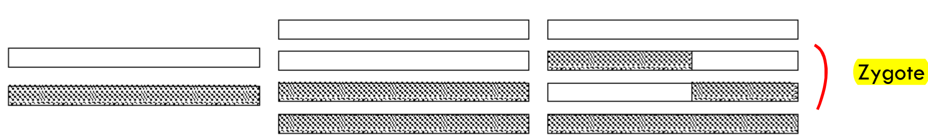
- Human DNA is organised into chromosomes
- Human body cells contains 23 pairs of chromosomes which together define the physical attributes of the individual:



- Gametes (the female and male cells in a combination) contain 23 individual chromosomes rather than 23 pairs
- Gametes are formed by cell splitting called meiosis
- During meiosis the pairs of chromosome undergo an operation called *crossing-over*

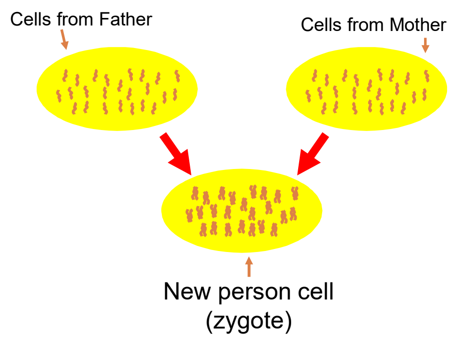
Crossing-over during meiosis

- Chromosome pairs align and duplicate
- **Combination:** Inner pairs link at a centromere and swap parts of themselves
- Outcome is one copy of maternal/paternal chromosome plus two entirely new combinations
- After crossing-over one of each pair goes into each gamete



Fertilisation

- What happens in nature



- New zygote rapidly divides etc creating many cells all with the same genetic contents
- Although all cells contain the same genes, depending on, for example where they are in the organism, they will behave differently
- This process of differential behaviour during development is called ontogenesis

Mutation

- Occasionally some of the genetic material changes very slightly during this process (replication error)
- This means that the child might have genetic material information not inherited from either parent
- This can be
  - **Catastrophic:** offspring in not viable (most likely)
  - **Neutral:** new feature not influences fitness
  - **Advantageous:** strong new feature occurs
- Redundancy in the genetic code forms a good way of error checking

Motivations for EC: 1

- **Inspirations:** nature has always served as a source of inspiration for engineers and scientists. We sometimes encounter very complex problems that we cannot solve in an acceptable time…

- **Best problem solvers in nature:**
  - The (human) brain that created "the wheel, New York, wars and so on" (after Douglas Adams' Hitch-Hikers Guide)
  - The evolution mechanism that created the human brain (after Darwin's Origin of Species)

- **What scientists built after those inspirations:**
  - Answer 1 → Neuro-computing (Neural Networks…)
  - Answer 2 → Evolutionary computing

The Evolutionary Computing Metaphor

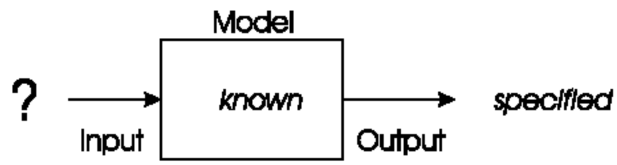
- So, scientists were inspired by Darwin's theory of evolution. They proposed several evolutionary (e.g. Genetic Algorithm) and many other nature inspired algorithms.

PROBLEM SOLVING	↔	**EVOLUTION**
Problem	↔	Environment
Candidate Solution	↔	Individual
Quality		Fitness

- **Solving a programming problem:** we have a problem, we produce a candidate solution and we measure its quality.
- **Evolution domain:** we have an environment and one or more individuals that evolve.We measure the fitness of individuals and stop the evolution when an acceptable fitness is reached.
  - Fitness → chances for survival and reproduction
  - Quality → chance for seeding new solutions

Problem type 1 : Optimisation

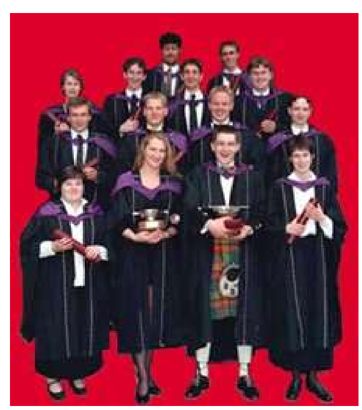
- We have a model of our system and seek inputs that give us a specified goal



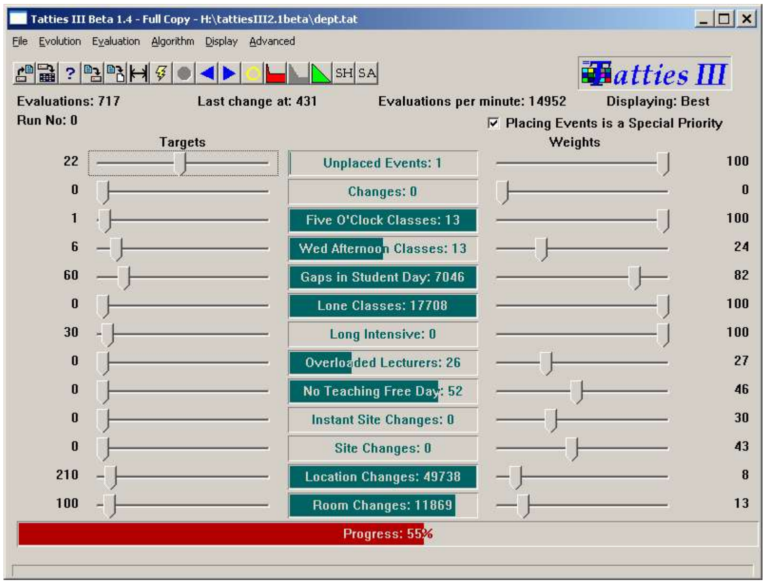
- Examples:
  - Time tables for university, call center, or hospital
  - Design specifications, …
- **Uni. Time Table:** What plan we use for the university (or the model of it) to provide our specified goals (i.e. plans that make students and lecturers happy)

Optimisation Problem Example 1: University timetabling

- Enormously big search space
- Timetables must be good
- “Good” is defined by a number of competing criteria
- Timetables must be feasible
- Vast majority of search space is infeasible

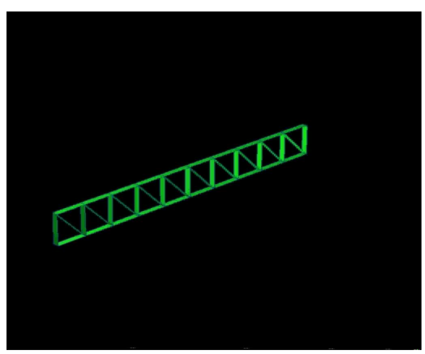


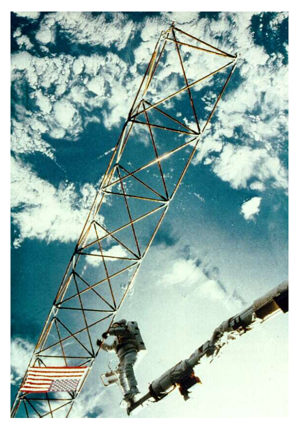
Optimisation Problem Example 1: University timetabling



Optimisation Problem Example 2: Satellite structure

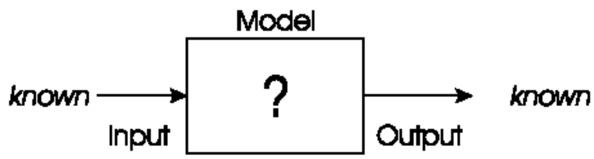
- Optimizing satellite designs for NASA that maximize vibration isolation
- **Evolving:** design structures
- **Fitness:** vibration resistance
- Evolutionary "creativity"





Problem types 2: Modelling

- We have corresponding sets of inputs & outputs and seek model that delivers correct output for every known input



- Evolutionary machine learning

- **Neural Networks:** we have inputs and outputs, we want a function (and its parameters) that map the inputs to output…

Modelling example: Loan applicant creditibility

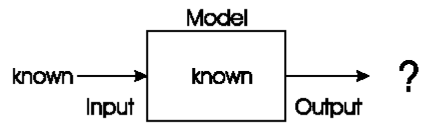
- Banks build creditability model to predict loan paying behavior of new applicants

- **Prediction model:** we can use linear, non-linear, statistical, neural networks or any type of model for prediction

- **Evolving:** prediction models (or their parameters)
- **Fitness:** model accuracy on the historical data

Problem type 3: Simulation

- We have a given model and wish to know the outputs that arise under different input conditions

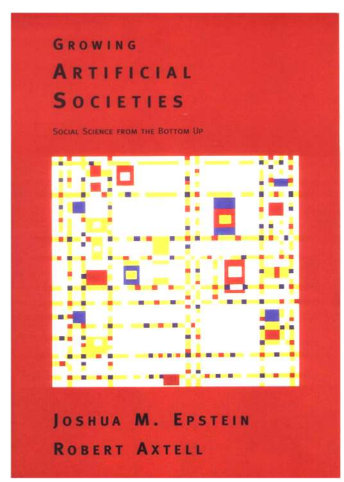


- Often used to answer "what-if" questions in evolving dynamic environments
  - e.g. Evolutionary economics, Artificial Life

- In this type of simulations, we produce different outputs and check whether it matches the input and model we have or not… When it matched, we take it as the result of simulation

Simulation example: evolving artificial societies

- Simulate trade, economic competition, …
- Then we can use models to find better strategies and policies
- Evolutionary economy



Simulation – Evolution of a Car

Evolutionary Algorithms

Evolutionary Computing – Intro 1

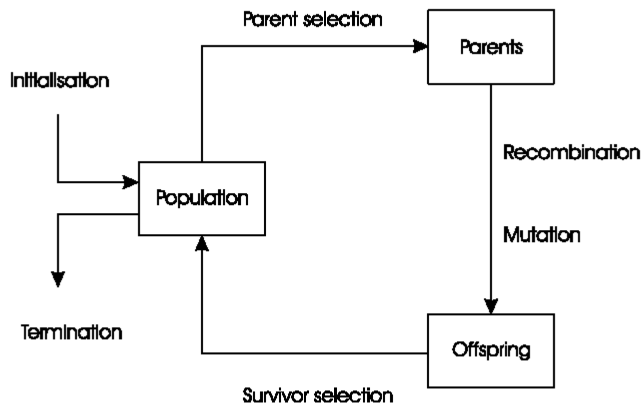
- EAs are built based on the inspiration from natural evolution.
- **Generate and test algorithms:** EAs fall into the category of "generate and test" algorithms
  - These are stochastic, population-based algorithms
- **Major operations:**
  - **Variation operators (recombination and mutation):** create the necessary diversity and thereby facilitate novelty
  - **Selection:** reduces diversity and acts as a force for raising the quality

Evolutionary Computing – Intro 2

- **Population of individuals:** exists in an environment with limited resources
- **Competition:** because of those limited resources, ***fitter*** individuals that are better adapted to the environment are selected for reproduction
- **Seeds for next generations:** selected individuals act as seeds for the generation of new individuals through recombination and mutation
- **Competition of the new generation:** new individuals have their fitness evaluated and compete (possibly also with parents) for survival (next round of selection).
- **Rise of fitness:** over time, natural selection causes a rise in the fitness of the population

General Scheme of EAs

- General flow in evolutionary algorithms is as follows (with some modifications):



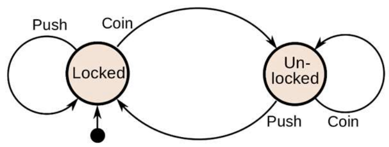
Pseudo-code for typical EA

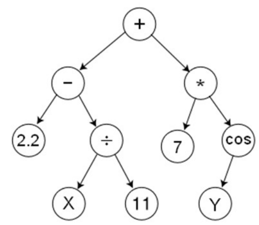
- Pseudo code of the operations:

```
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
```

What are the different types of EAs

- **Different flavours of EAs (different representations):**
  - **Genetic Algorithms:** Binary or integer strings
  - **Evolution Strategies:** Real-valued vectors, mutation strength encoded in individuals and optimized
  - **Evolutionary Programming:** finite state machine, program structure fixed, parameters evolve
  - **Genetic Programming:** LISP trees





$$\left(2.2 - \left(\frac{X}{11}\right)\right) + \left(7 * \cos(Y)\right)$$

- These differences are largely irrelevant, best strategy
  - choose suitable representation for the problem
  - choose suitable variation operators for the representation
- **Selection operators:** use fitness and so are independent of representation

Representation

- **Phenotype space (Real world):** candidate solutions (individuals)
- **Genotype space (Genes world):** candidate solutions (above) encoded in chromosomes
  - **Encoding :** phenotype → genotype (not necessarily one to one)
  - **Decoding :** genotype → phenotype (must be one to one)
- **Gene:** chromosomes contain genes, which are in (usually fixed) positions called loci (sing. locus) and have a value
- **Every feasible solution must be representable in genotype space**

Fitness (Evaluation) Function

- Represents the requirements that the population should adapt to.
  Alternative names:
  - Quality function
  - Objective function
- Assigns a real-valued fitness to each candidate solution
  - Forms the basis for selection
  - So the more discrimination (different values) the better
- **Optimization:** typically we talk about fitness being maximised
  - Some problems may be best posed as minimisation problems, but conversion is trivial

Population

- Holds possible or candidate solutions (representations of them)
- Usually fixed size genotypes
- **Sophisticated representations:** may use 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:** refers to the number of different fitnesses OR phenotypes Or genotypes present (note not the same thing) in the population

Parent Selection Mechanism

- **Parent selection probability:** usually variable probabilities are assigned to individuals depending on their fitness
  - High quality solutions more likely to become parents, but not guaranteed
  - Even worst in current population usually get small but non-zero probability of becoming a parent
- **Escape from local optima:** the *stochastic* nature can aid escape from local optima

Variation Operators

- **Role:** 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
- **Importance:** much debate about relative importance of recombination and mutation
  - Nowadays most EAs use both
  - Choice of particular variation operators is representation dependant

Mutation

- Acts on one genotype and delivers another
- Element of randomness is essential and differentiates it from other unary heuristic operators
- Importance depends on representation and design:
  - **Binary GAs:** preserves and introduces diversity
  - **EP for FSM's/ continuous variables:** only search operator
  - **GP:** hardly used

Recombination

- Transfers 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 live creatures (i.e. cows)

Survivor Selection (*replacement)*

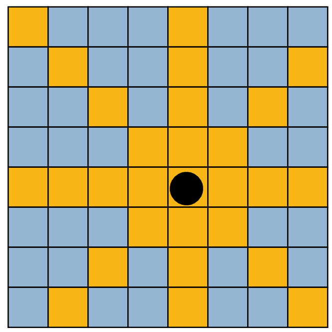
- **Replacement strategy:** Most EAs use fixed population size so need a way of going from (parents + offspring) to next generation
- Often deterministic but could be stochastic
  - **Fitness based:**
    - Rank "parents + offspring" and take best
    - Assign probabilities proportionate to fitness and then use stochastic selection
  - **Age based:**
    - Make as many offspring as parents and delete all parents
  - **Both:** sometimes do combination (elitism)

Initialisation / Termination

- Initialisation usually done at random,
  - Need to ensure even spread and mixture of possible gene 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

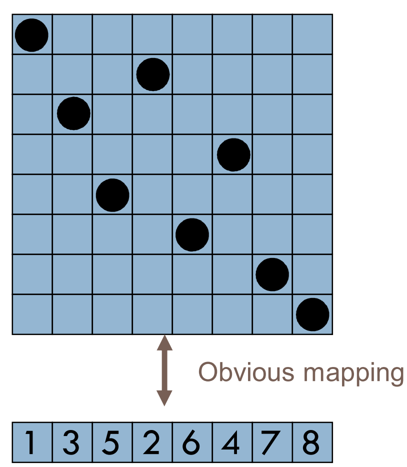
Example: the 8 queens problem

- **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



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

The 8 queens problem: Recombination

- **Recombination:** combining two (or more) 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

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

→

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

The 8 queens problem: Mutation

- **Mutation:** small variation in one permutation
  - **Example:** swapping values of two randomly chosen positions,

1	3	5	2	6	4	7	8

→

1	3	7	2	6	4	5	8

The 8 queens problem: Selection

- **Parent selection:**
  - Pick 5 random 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
    - Sort the whole population by decreasing fitness
    - Replace the first with a fitness lower than the given child

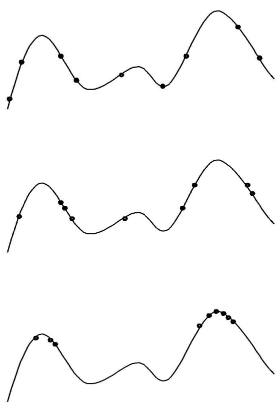
8 Queens Problem: summary

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

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

Typical behaviour of an EA

- Phases in optimizing on a 1-dimensional fitness landscape



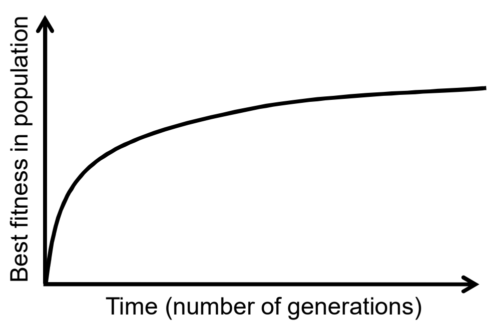
Early phase:
Quasi-random population distribution

Mid-phase:
Population arranged around/on hills

Late phase:
Population concentrated on high hills

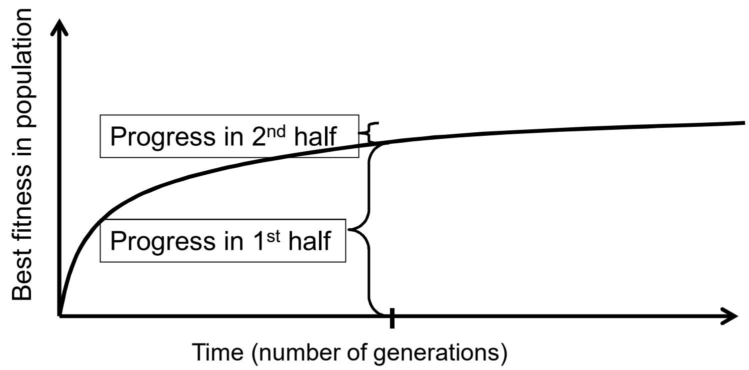
Typical run: progression of fitness

- Typical run of an EA shows so-called "anytime behavior"



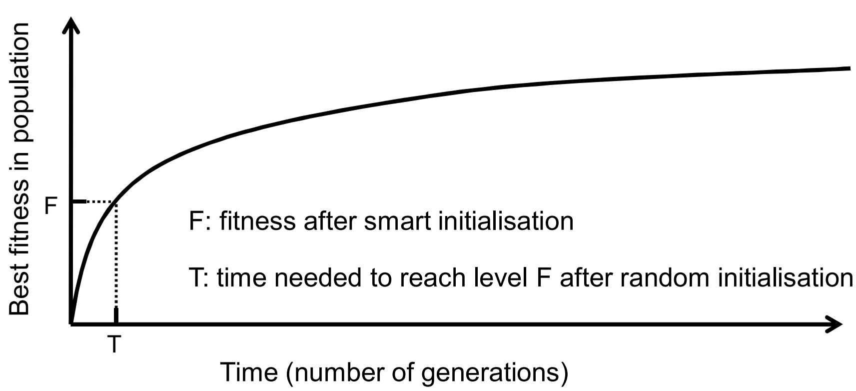
Are long runs beneficial?

- **Are long runs beneficial?**
  - It depends how much you want the last bit of progress
  - It may be better to do more shorter runs



Is it worth expending effort on smart initialisation?

- **Is it worth expending effort on smart initialization?**
  it depends
  - Possibly, if good solutions/methods exist.
  - Care is needed, see chapter on hybridization (don't eliminate diversity)

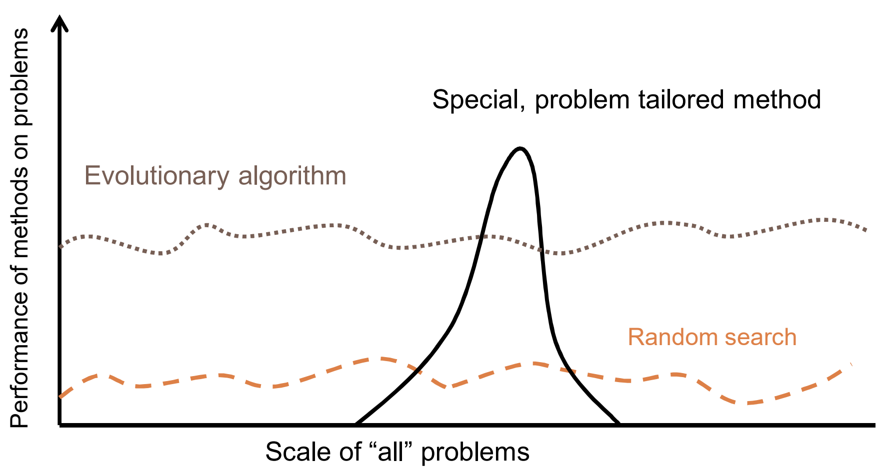


Evolutionary Algorithms in Context

- There are many views on the use of EAs as robust and generic 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

EAs as problem solvers: Goldberg's 1989 view

- Specialized methods work better on specific problems. EA generic methods work evenly good on all problems and instances.

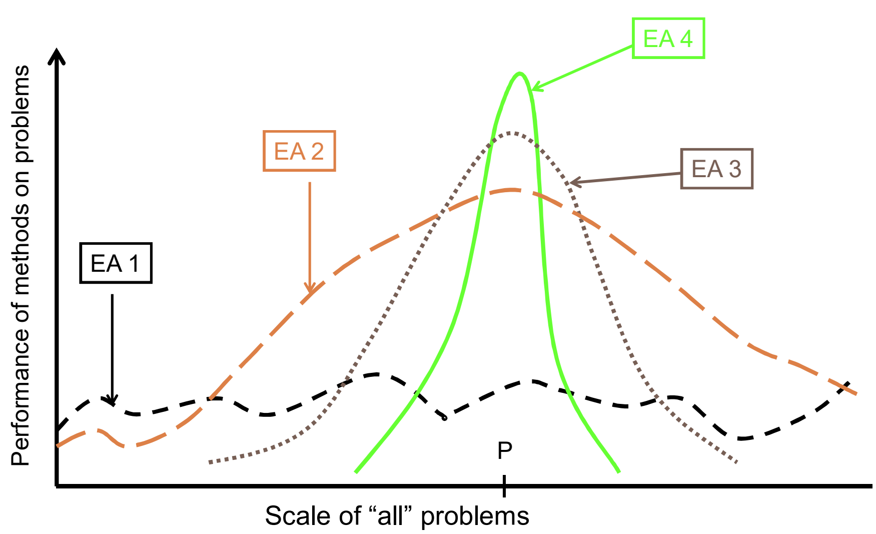


EAs and domain knowledge

- Trend in the 90's:
  - Adding problem specific knowledge to EAs (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

Michalewicz' 1996 view

- EA performance curve "deformation"



EC and Global Optimisation

- **Global Optimisation:** search for finding best solution x^* out of some fixed set S

- **Deterministic approaches:**
  - Branch and bound: DFS, BFS, …
  - Guarantee to find x^*, but may run in super-polynomial time

- **Heuristic Approaches (generate and test):**
  - Rules for deciding which $x \in S$ to generate next
  - No guarantees that best solutions found are globally optimal

EC vs. Neighbourhood Search Algorithms

- Some heuristics may only guarantee that best point found is locally optimal (e.g. Hill-Climbing algorithms)
  - **But** problems often exhibit many local optima
  - Are often very quick to identify good solutions

- **EAs are distinguished (from other heuristic) by:**
  - Use of population
  - Use of multiple stochastic search operators
  - Especially variation operators with arity >1
  - Stochastic selection