2017-2018 Güz Yarıyılı
Balikesir Universitesi, Endustri Muhendisligi Bolumu

EMM4131
Popülasyon Temelli Algoritmalar
(Population-based Algorithms)

2

Introduction to Meta-heuristics and Evolutionary Algorithms

Yrd. Doç. Dr. İbrahim KÜÇÜKKOÇ

Web: http://ikucukkoc.baun.edu.tr

Email: ikucukkoc@balikesir.edu.tr

Optimization problems

- The many decision-making problems can be often expressed as a constrained **optimization problem** with some decision variables that are restricted by a set of constraints.
- Types of constrained optimization problems:
  - **Combinatorial problems**: When the decision variables are discrete
  - **Continuous problems**: When the decision variables are continuous
  - **Mixed problems**

Combinatorial Problems

Examples of real-world combinatorial optimization problems include:

- Assembly-line balancing problems
- Vehicle routing and scheduling problems
- Facility location problems
- Facility layout design problems
- Job sequencing and machine scheduling problems
- Manpower planning problems
- Production planning and distribution

– etc.

Combinatorial Problems

- Combinatorial optimization problems are often easy to state but very difficult to solve.
- Many of the problems arising in applications are NP-hard, that is, it is strongly believed that they cannot be solved to optimality within polynomially bounded computation time.
- Two classes of algorithms are available for the solution of combinatorial optimization problems:
  – Exact algorithms
  – Approximate algorithms

Exact algorithms

- **Exact algorithms** are guaranteed to find the **optimal solution** and to prove its optimality for every finite size instance of a combinatorial optimization problem within an **instancedependent dependent run time**.
- In the case of NP-hard problems, in the worst case, **exponential time** to find the optimum.
- For most NP-hard problems the performance of exact algorithms **is not satisfactory**.
- If optimal solutions cannot be efficiently obtained in practice, the only possibility is to trade optimality for efficiency.
- **Approximate algorithms**, often also called **heuristic methods** or simply **heuristics**, seek to obtain good, that is, near-optimal solutions at relatively low computational cost without being able to guarantee the optimality of solutions.

Metaheuristics

Metaheuristics

- A disadvantage of heuristic methods is that they:
  - – either generate only a very limited number of different solutions, or
  - – they stop at poor quality local optima, which is the case for iterative improvement methods.
- **Metaheuristics** have been proposed which try to bypass these problems.
- Metaheuristics apply to solve the problems known as of **difficult optimization**
- Available from the 1980s

Metaheuristics

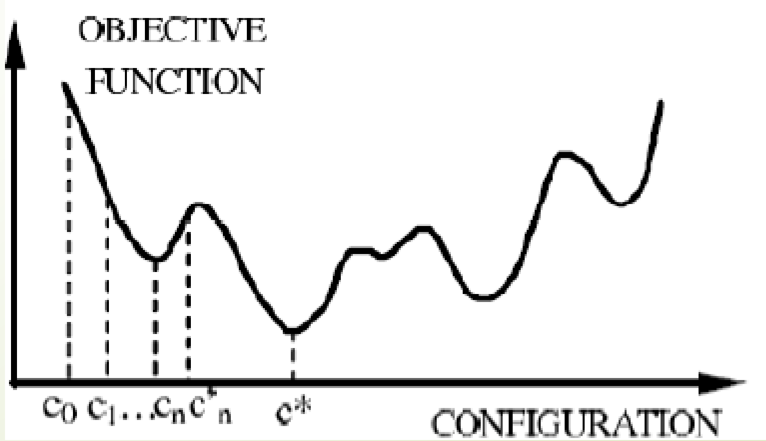
- **Definition**:

  – A **metaheuristic** is a set of algorithmic concepts that can be used to define heuristic methods applicable to **a wide set of** different problems.

  – A **metaheuristic** can be seen as a **general-purpose heuristic method** toward promising regions of the search space containing high-quality solutions.

  – A metaheuristic is a general algorithmic framework which can be applied to different optimization problems with relatively **few modifications** to make them adapted to a specific problem.

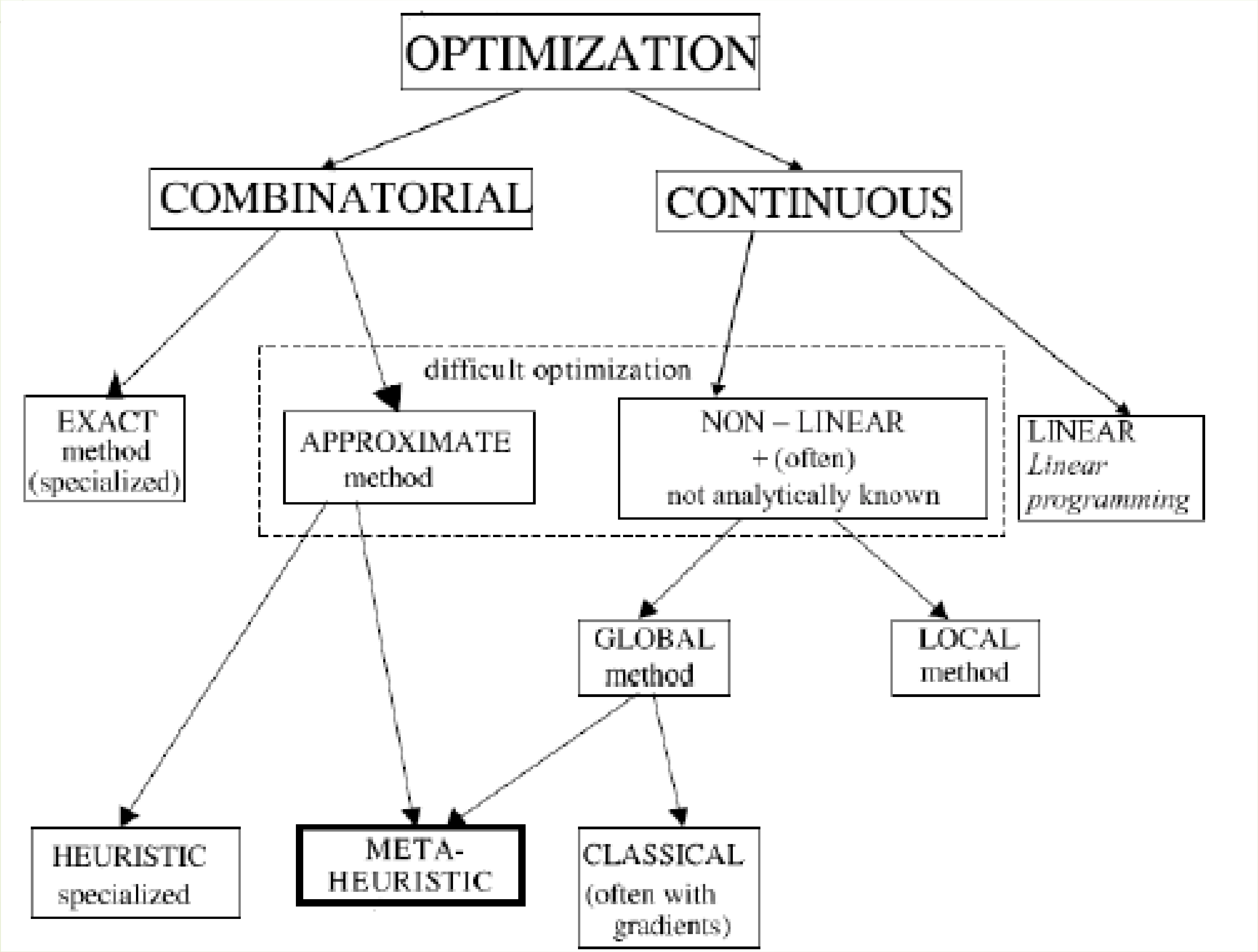
- **Metaheuristics have capability to be extracted from a local minimum.**



Metaheuristics

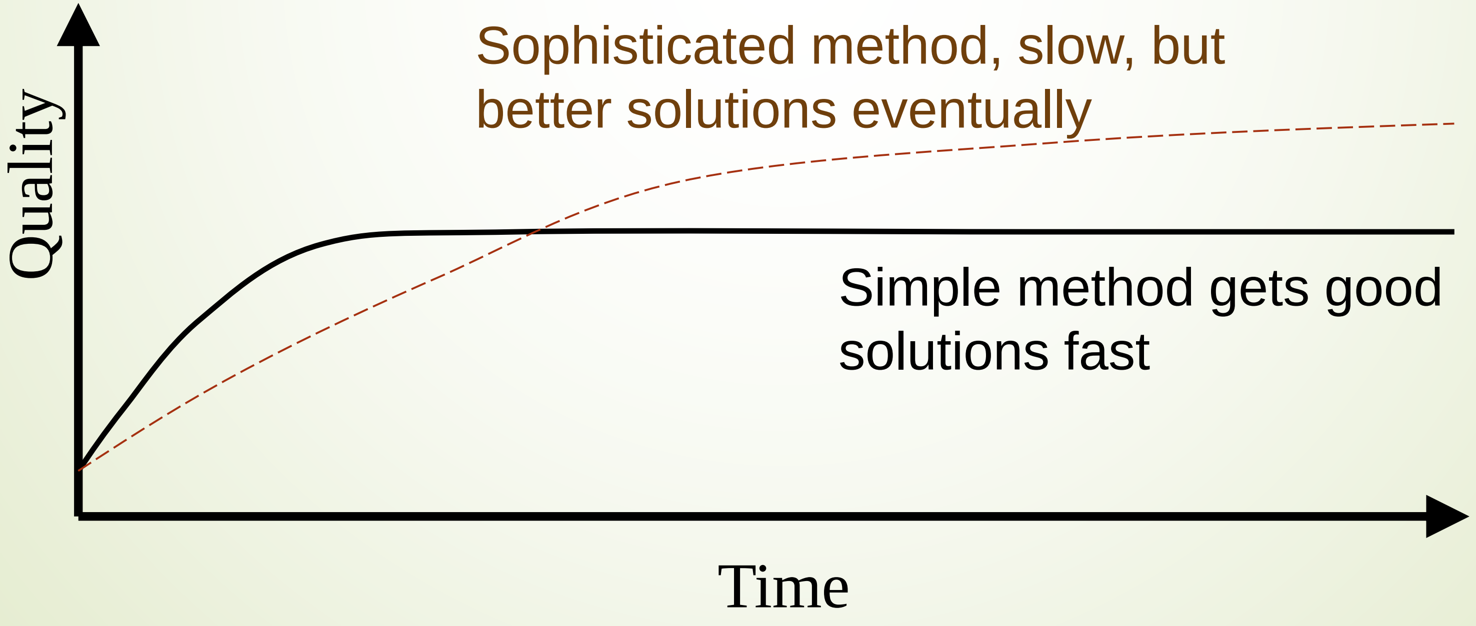
- The metaheuristics are from now on regularly employed in all the sectors of engineering,
- Examples of metaheuristics algorithms:
  – The evolutionary algorithms
  – The tabu search method
  – The ant colony optimization
  – The simulated annealing method
  – etc.

Metaheuristics



Typical Performance of Approximate Methods

- Evolutionary Algorithms turn out to be the most successful and generally useful approximate algorithms around. They often take a long time though – it's worth getting used to the following curve which tends to apply across the board.



What is an Evolutionary Algorithm?

The Main Evolutionary Computing Metaphor

EVOLUTION		**PROBLEM SOLVING**
Environment	$\longleftrightarrow$	Problem
Individual	$\longleftrightarrow$	Candidate Solution
Fitness	$\longleftrightarrow$	Quality

Fitness $\rightarrow$ chances for survival and reproduction

Quality $\rightarrow$ chance for seeding new solutions

Brief History 1: the ancestors

- 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

Survival of the fittest

- All environments have finite resources

  (i.e., can only support a limited number of individuals)
- Lifeforms 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

Population-Individual

- 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"

Natural Genetics

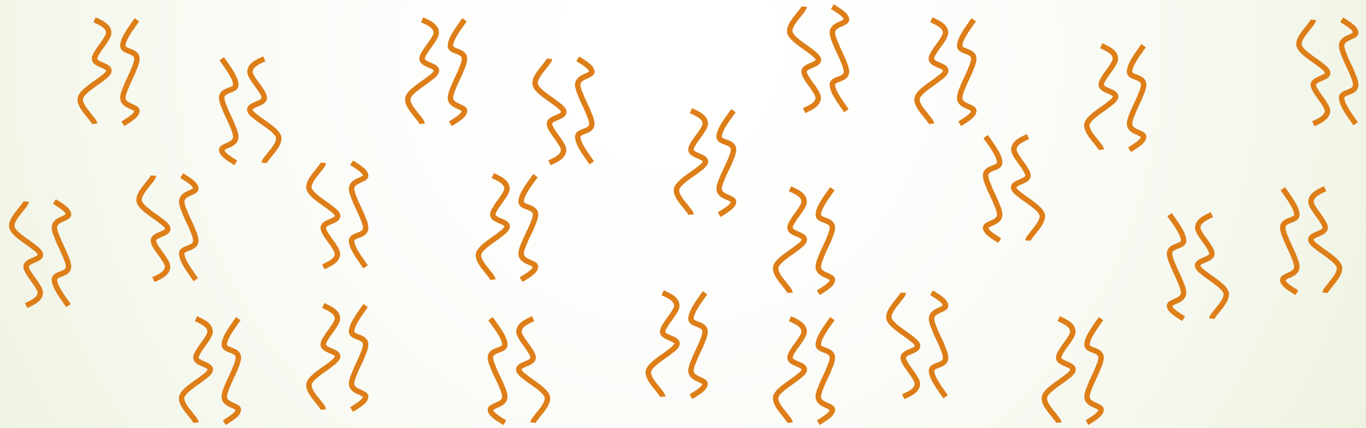
- The information required to build a living organism is coded in the DNA of that organism
- Genotype (DNA inside) determines phenotype
- Genes $\rightarrow$ phenotypic traits is a complex mapping
  - One gene may affect many traits (pleiotropy)
  - Many genes may affect one trait (polygeny)
- Small changes in the genotype lead to small changes in the organism (e.g., height, hair colour)

Genes and the Genome

- Genes are encoded in strands of DNA called chromosomes
- In most cells, there are two copies of each chromosome (diploidy)
- The complete genetic material in an individual's genotype is called the Genome
- Within a species, most of the genetic material is the same

Example: 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:

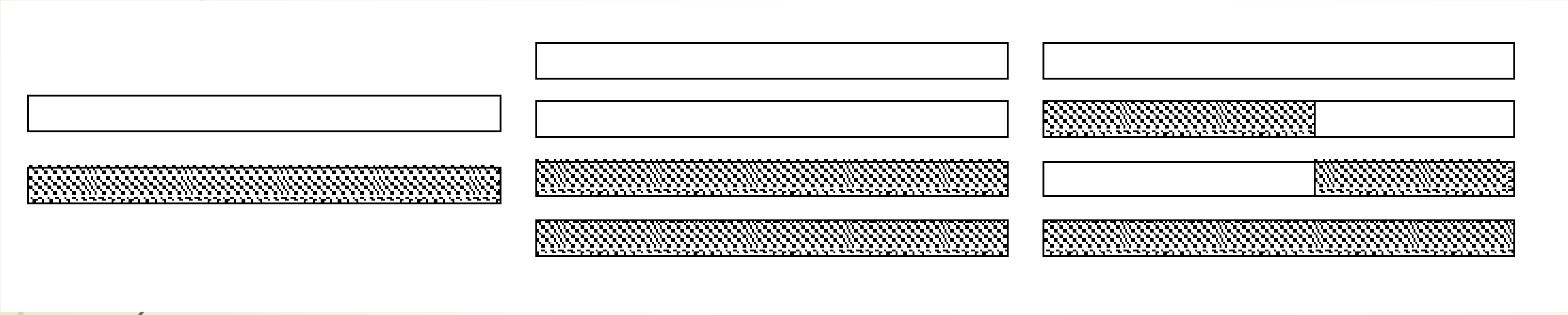


Reproductive Cells

- Gametes (sperm and egg cells) contain 23 individual chromosomes rather than 23 pairs
- Cells with only one copy of each chromosome are called Haploid
- Gametes are formed by a special form of cell splitting called meiosis
- During meiosis the pairs of chromosome undergo an operation called *crossing-over*

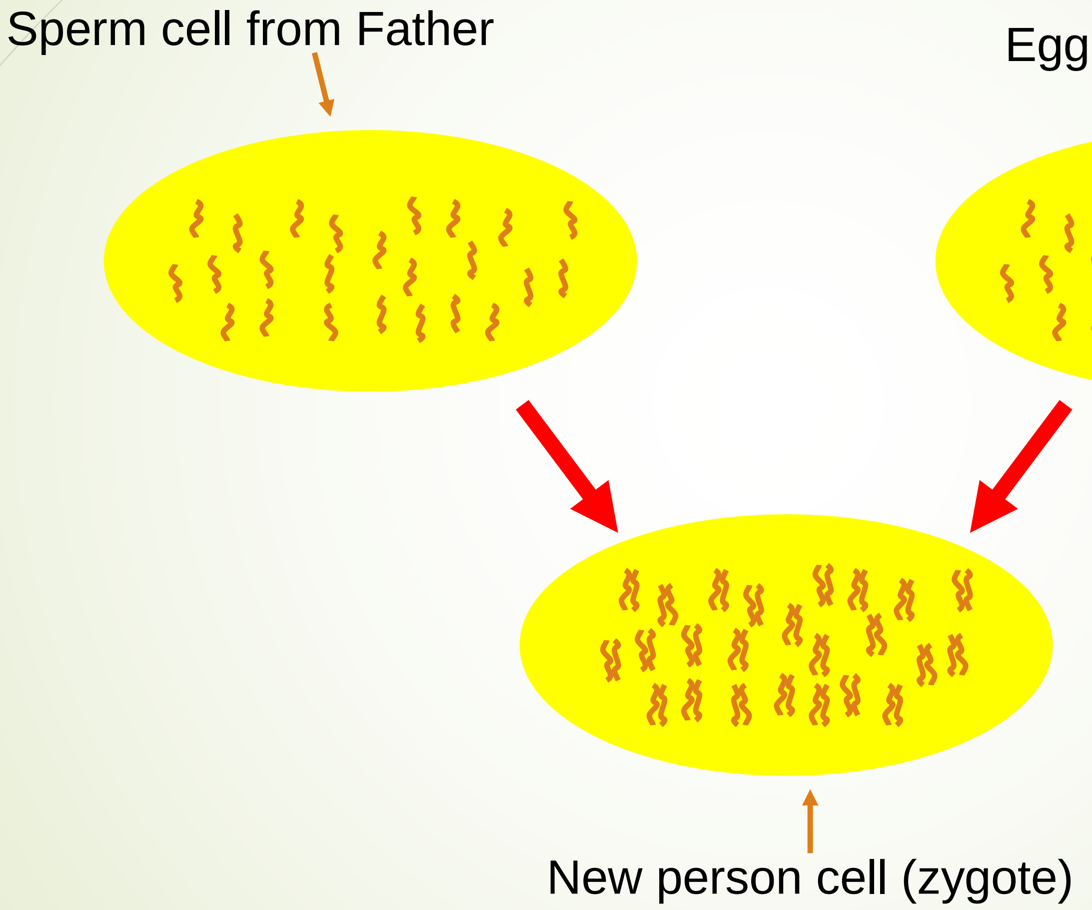
Crossing-over during meiosis

- Chromosome pairs align and duplicate
- 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



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

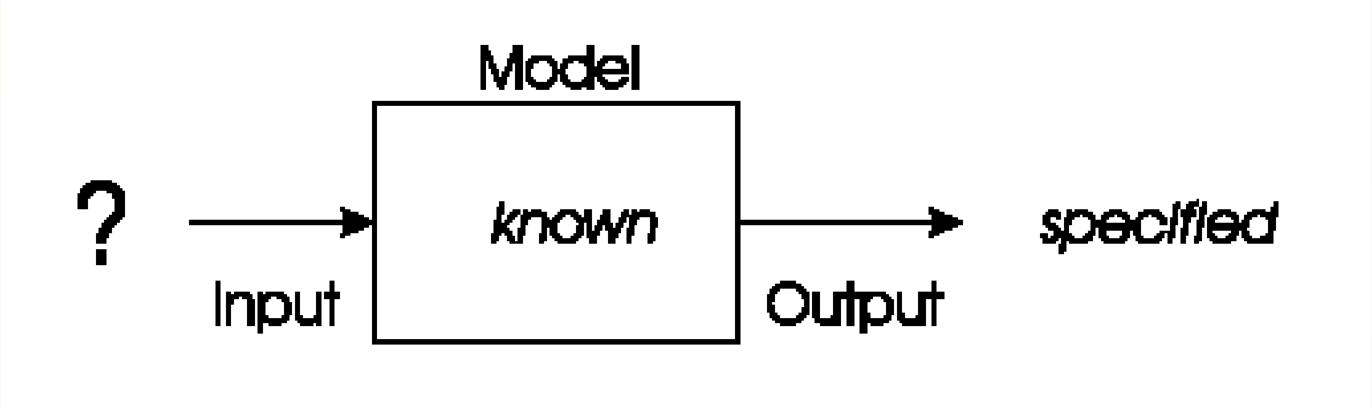
- Nature has always served as a source of inspiration for engineers and scientists
- The best problem solver known in nature is:
  - the (human) brain that created "the wheel, New York, wars and so on" (after Douglas Adams' Hitch-Hikers Guide)
  - the evolution mechanism that created the human brain (after Darwin's Origin of Species)
- Answer 1 → neurocomputing
- Answer 2 → evolutionary computing

Motivations for EC: 2

- Developing, analyzing, applying problem solving methods a.k.a. algorithms is a central theme in mathematics and computer science
- Time for thorough problem analysis decreases
- Complexity of problems to be solved increases
- Consequence:
  Robust problem solving technology needed

Problem type 1 : Optimisation

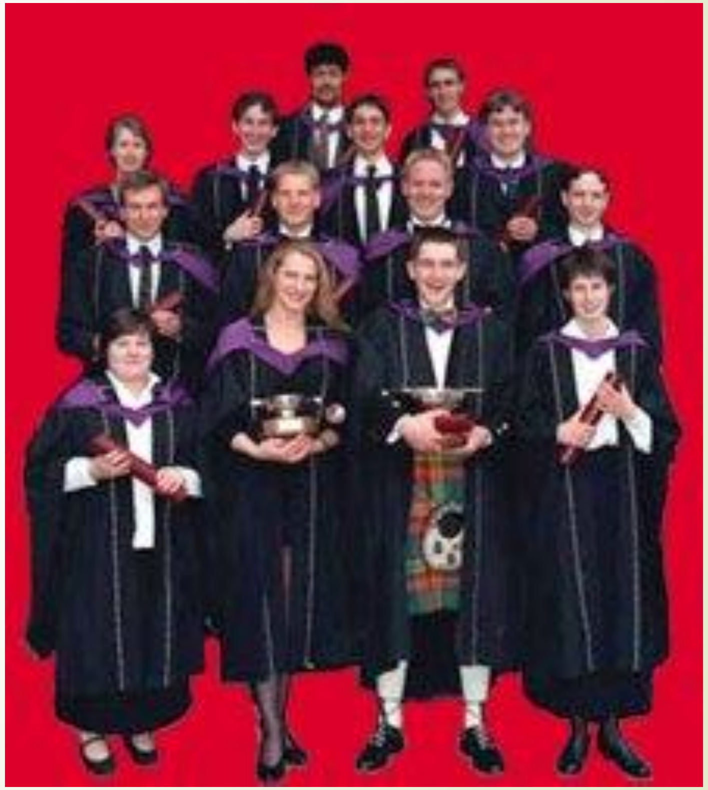
- We have a model of our system and seek inputs that give us a specified goal.

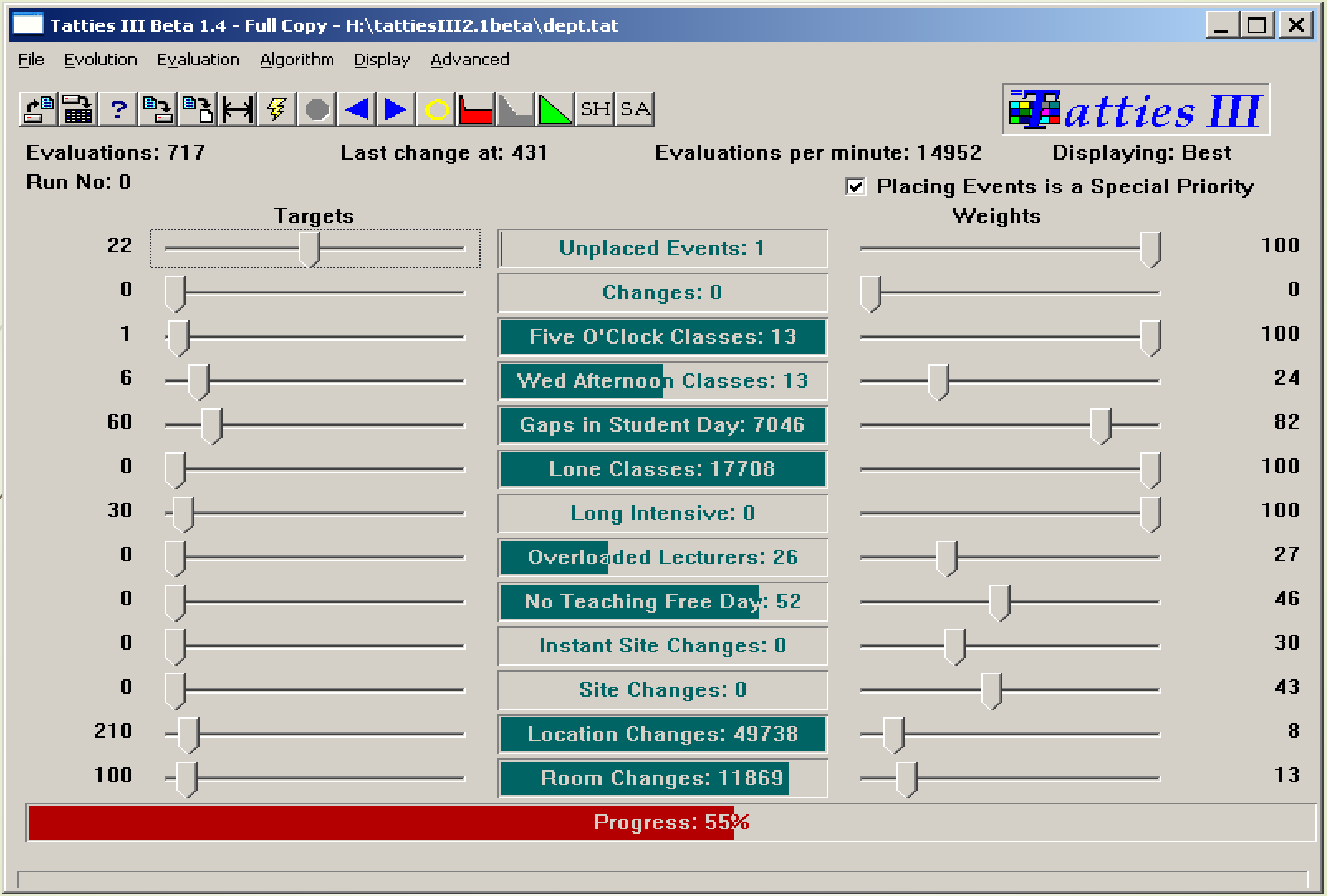


- e.g.
- time tables for university, call center, or hospital
- design specifications, etc etc

Optimisation example 1: University timetabling

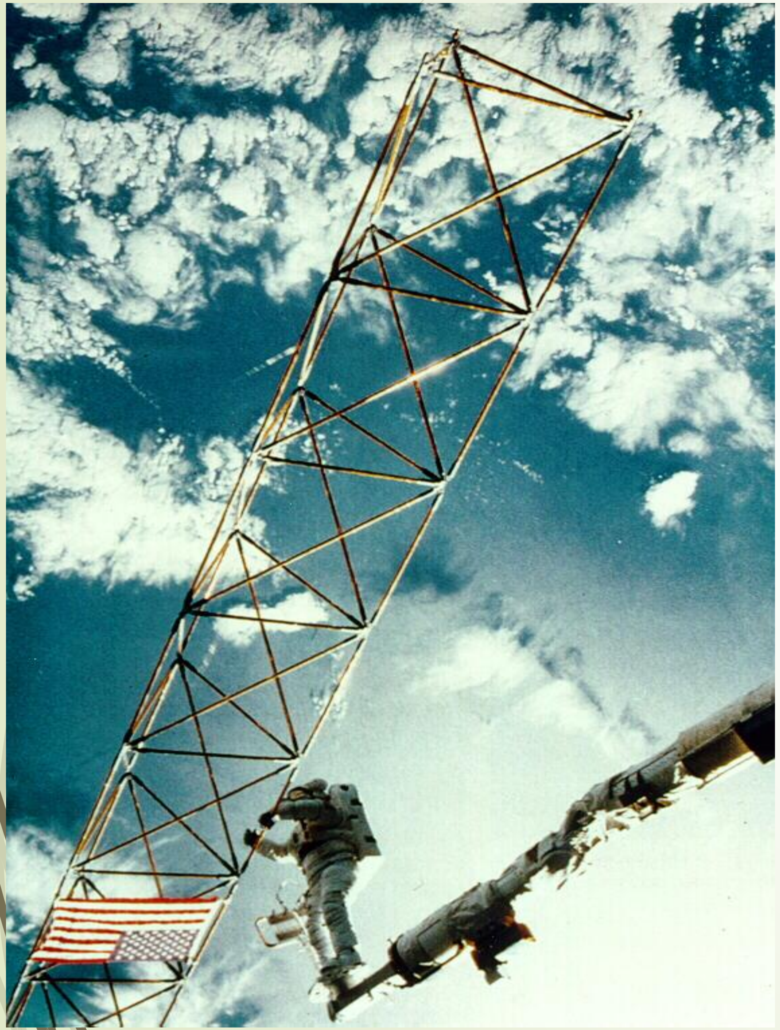
- Enormously big search space
- Timetables must be *good*
- “Good” is defined by a number of competing criteria
- Timetables must be feasible
- Vast majority of search space is infeasible





Tatties III Beta 1.4 - Full Copy - H:\tattiesIII2.1beta\dept.tat
File Evolution Evaluation Algorithm Display Advanced
SH SA
Tatties III
Evaluations: 717
Last change at: 431
Evaluations per minute: 14952
Displaying: Best
Run No: 0
Placing Events is a Special Priority
Targets
Weights
22 Unplaced Events: 1 100
0 Changes: 0 0
1 Five O'Clock Classes: 13 100
6 Wed Afternoon Classes: 13 24
60 Gaps in Student Day: 7046 82
0 Lone Classes: 17708 100
30 Long Intensive: 0 100
0 Overloaded Lecturers: 26 27
0 No Teaching Free Day: 52 46
0 Instant Site Changes: 0 30
0 Site Changes: 0 43
210 Location Changes: 49738 8
100 Room Changes: 11869 13
Progress: 55%

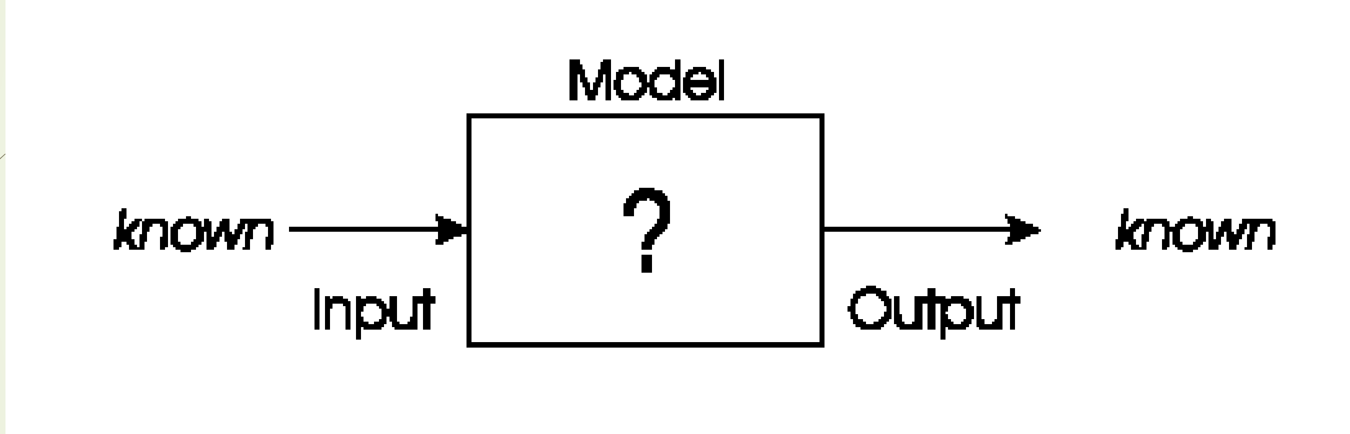
Optimisation example 2: Satellite structure



- Optimised satellite designs for NASA to 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



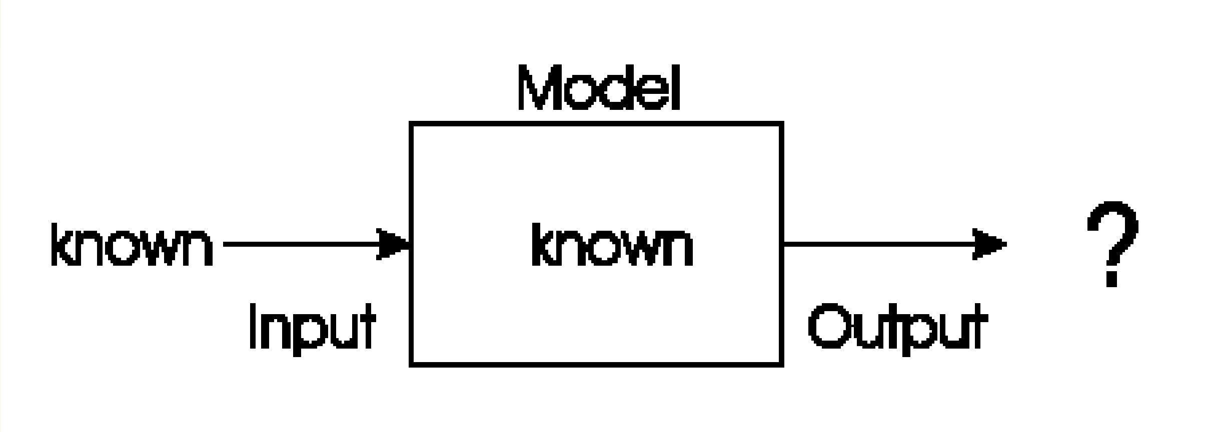
- Evolutionary machine learning

Modelling example: loan applicant creditibility

- British bank evolved creditability model to predict loan paying behavior of new applicants
- Evolving: prediction models
- Fitness: model accuracy on 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

Demonstration: magic square

- Given a 10x10 grid with a small 3x3 square in it
- Problem: arrange the numbers 1-100 on the grid such that
  - all horizontal, vertical, diagonal sums are equal (505)
  - a small 3x3 square forms a solution for 1-9
- Evolutionary approach to solving this puzzle:
  - Creating random begin arrangement
  - Making N mutants of given arrangement
  - Keeping the mutant (child) with the least error
  - Stopping when error is zero

Demonstration: magic square

- Software by M. Herdy, TU Berlin
- Interesting parameters:
  - Step1: small mutation, slow & hits the optimum
  - Step10: large mutation, fast & misses (“jumps over” optimum)
  - Mstep: mutation step size modified on-line, fast & hits optimum
- Start: double-click on icon below
- Exit: click on TUBerlin logo (top-right)

Application

References

- J. Dreo A. Petrowski, P. Siarry E. Taillard, **Metaheuristics for Hard Optimization**, Springer-Verlag, 2006.
- R.J. Moraga, G.W. DePuy, G.E. Whitehouse, **Metaheuristics: A Solution Methodology for Optimization Problems**, Handbook of Industrial and Systems Engineering, A.B. Badiru (Ed.), 2006.
- M. Yaghini, **What is a Metaheuristic?,** http://webpages.iust.ac.ir/yaghini/Courses/AOR_872/What%20is%20a%20Metaheuristic.pdf, Accessed 18 Sept 2017.