An Introduction to Metaheuristic Algorithms and the Problems they (try to) Solve

Rhyd Lewis

Cardiff School of Mathematics / Cardiff Business School

- LewisR9@cf.ac.uk
- http://www.cf.ac.uk/maths/contactsandpeople/profiles/lewisr9.html

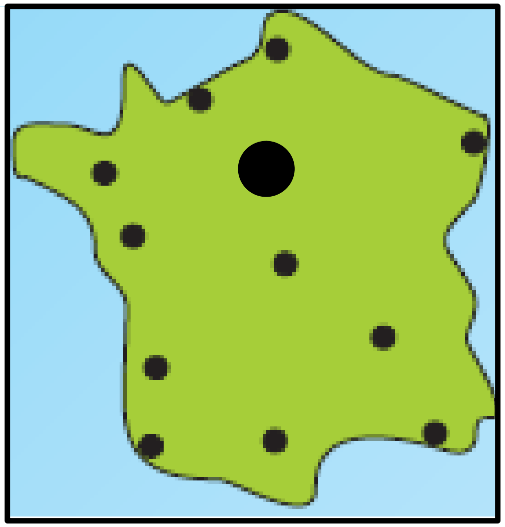
Talk Summary

- An introduction to **intractable** problems
- Solving intractable problems with **metaheuristic** algorithms
- Software Demonstration
- **Case Study:** The Second International Timetabling Competition `www.cs.qub.ac.uk/itc2007/`
- **Conclusions:** Advantages and limitations of metaheuristic algorithms

The Travelling Salesman Problem (TSP)

- A classic combinatorial optimisation problem
- Given *n* cities on a map, find the shortest route that visits all cities once, and starts and ends at the same city.

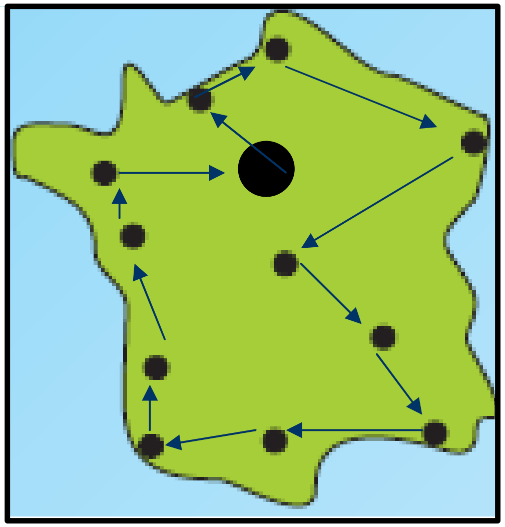
Starting and ending in Paris, which route allows us to visit all major cities with the least amount of travelling?
(We assume straight-line distances between cities for now)



The Travelling Salesman Problem (TSP)

- A classic combinatorial optimisation problem
- Given n cities on a map, find the shortest route that visits all cities once, and starts and ends at the same city.

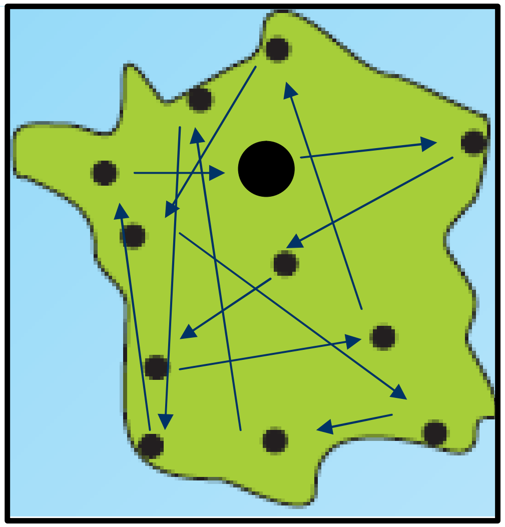
Starting and ending in Paris, which route allows us to visit all major cities with the least amount of travelling?
(We assume straight-line distances between cities for now)



The Travelling Salesman Problem (TSP)

- A classic combinatorial optimisation problem
- Given n cities on a map, find the shortest route that visits all cities once, and starts and ends at the same city.

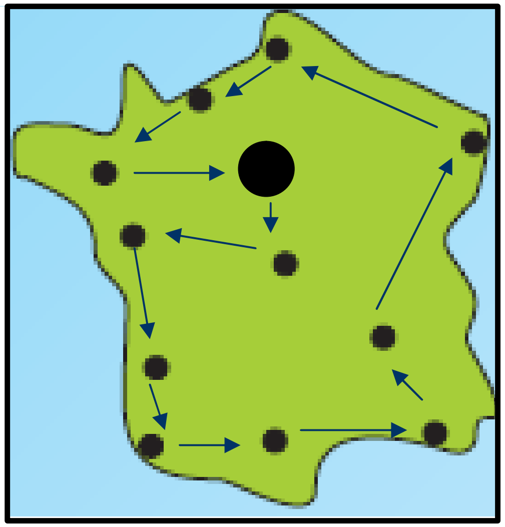
Starting and ending in Paris, which route allows us to visit all major cities with the least amount of travelling?
(We assume straight-line distances between cities for now)



The Travelling Salesman Problem (TSP)

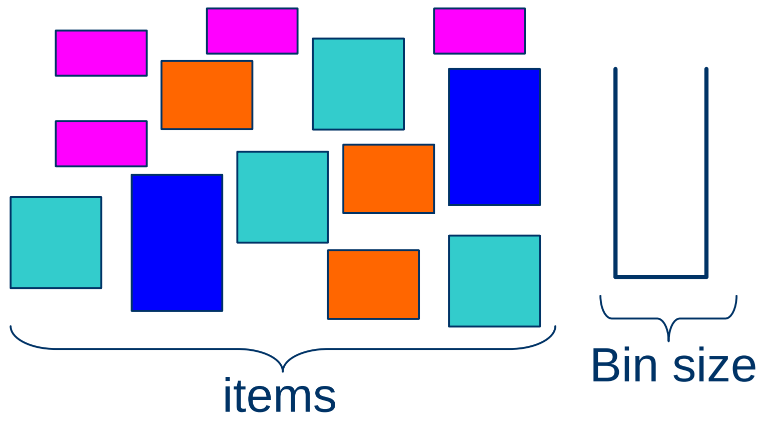
- A classic combinatorial optimisation problem
- Given n cities on a map, find the shortest route that visits all cities once, and starts and ends at the same city.

Starting and ending in Paris, which route allows us to visit all major cities with the least amount of travelling?
(We assume straight-line distances between cities for now)



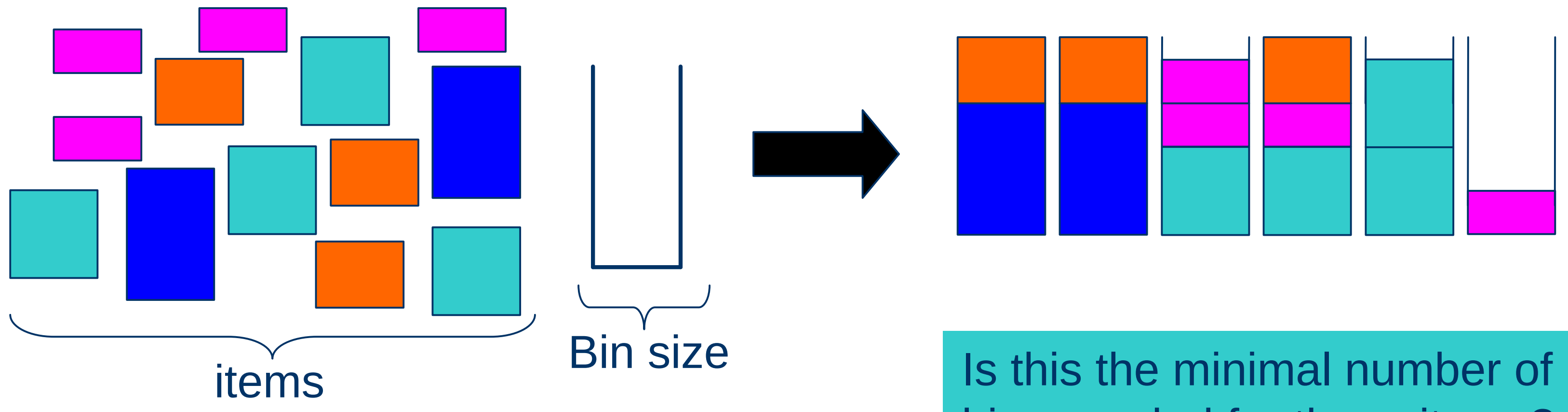
The 1-Dimensional Bin Packing Problem

Given *n* items of different (1D) sizes, and given some fixed-capacity bins, pack the items into a **minimum** number of bins



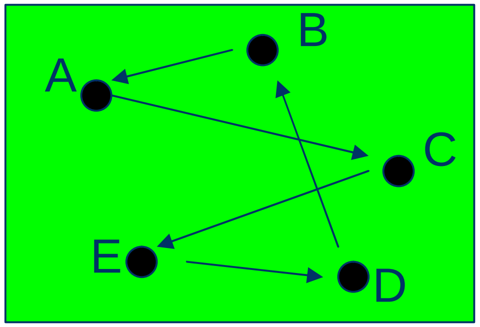
The 1-Dimensional Bin Packing Problem

Given n items of different (1D) sizes, and given some fixed-capacity bins, pack the items into a **minimum** number of bins



Is this the minimal number of bins needed for these items?

TSP Growth Rates



- A route around a map can be represented as a permutation of the n cites:
  - E.g. For 5 cities, **[B, A, C, E, D]** means “start at city B, then go to city A, then city C, then E, then D, and return to city B”
- Given n cities, there is a total of $n!$ permutations (where $n! = n \times (n-1) \times (n-2) \times \ldots \times 2 \times 1$)
- Some permutations represent the same routes – there are actually $\frac{1}{2}(n-1)!$ different routes in total.

An Inconvenient Truth … (in layman's terms)

- The only algorithm that will **guarantee** to return the provably optimal solution to **any** instance of the TSP needs to check the majority of – if not all – possible routes.
- However, the number of routes grows exponentially, quickly making the problem intractable:

Number of routes for different *n*'s

Cities (*n*)	5	10	50	100	1000
Routes [$\frac{1}{2}(n-1)!$]	12	181,440	3.04×10^{62}	4.67×10^{155}	**Lots!!!**

Other Intractable problems

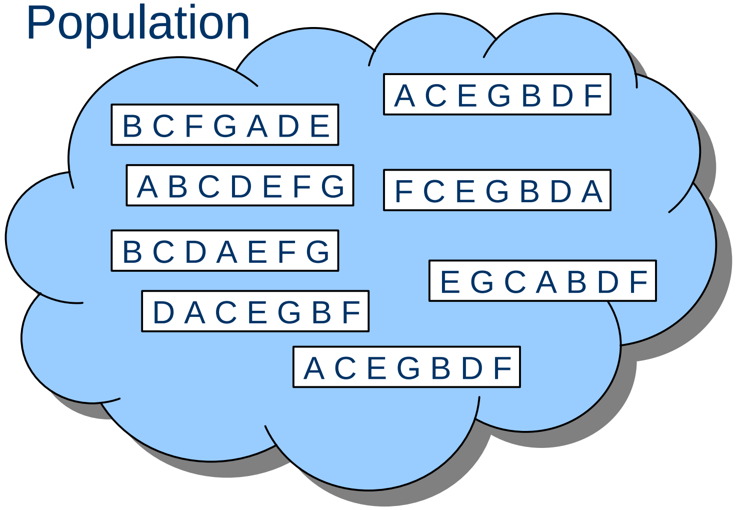
- Intractable problems arise in many areas:
  - Packing Problems
  - Games (Sudoku, *Tetris*, Minesweeper)
  - Vehicle routing problems
  - Scheduling and Timetabling Problems (see later)
  - Graph theoretic problems, and so on.
- To tackle them, we might:
  - Attempt to avoid or redefine the problem
  - Use some brute-force algorithm and limit ourselves only to small instances
  - Use **approximation algorithms** that will hopefully give us a solution that is “good enough” for practical purposes

Solving Intractable Problems using Metaheuristics

- A metaheuristic is a *general algorithmic framework* for addressing intractable problems
- They are often (though not necessarily) inspired by processes occurring in nature, e.g.
  - **Darwinian Natural Selection**
  - Annealing
  - Collective behaviour of ants
- Others merely provide neat ways of exploring the huge search spaces in efficient and effective ways.
- Typically, metaheuristics are approximation algorithms – they cannot always produce provably optimal solutions, but they do have the potential to produce good solutions in short amounts of time (if used appropriately).

Example: An “Evolutionary Algorithm” for the TSP

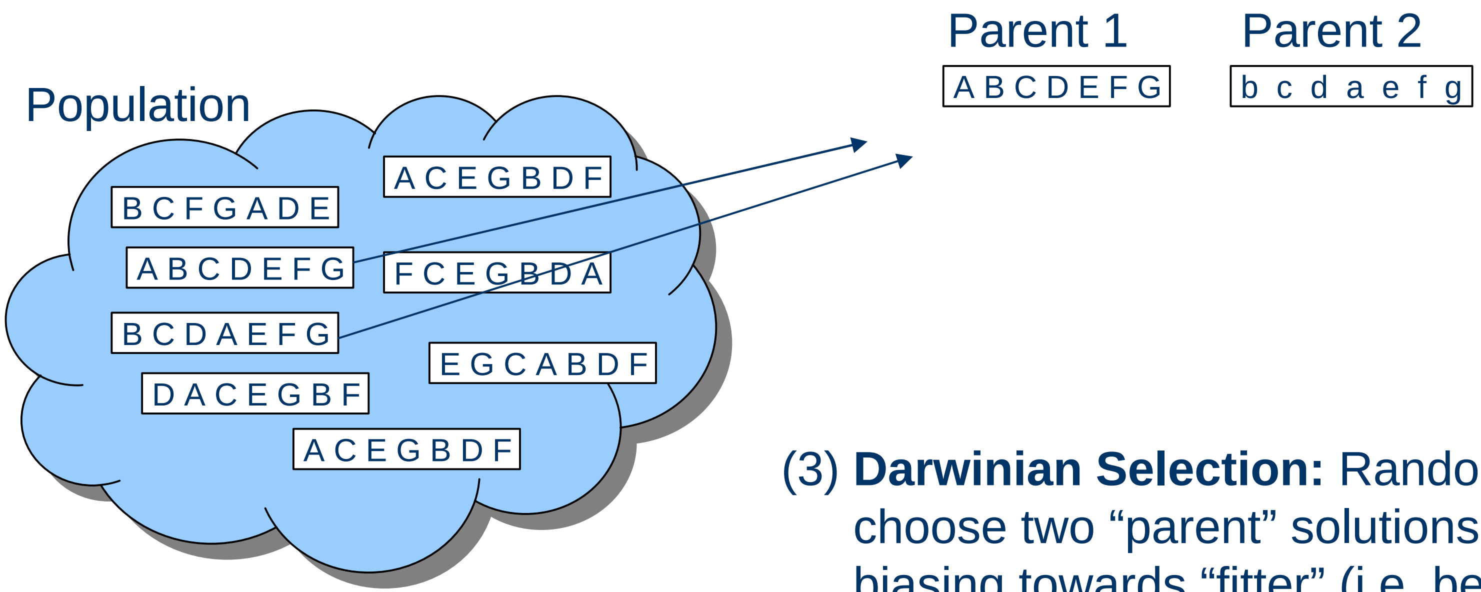
Population



(1) Randomly produce an “**initial population”** of valid candidate solutions.

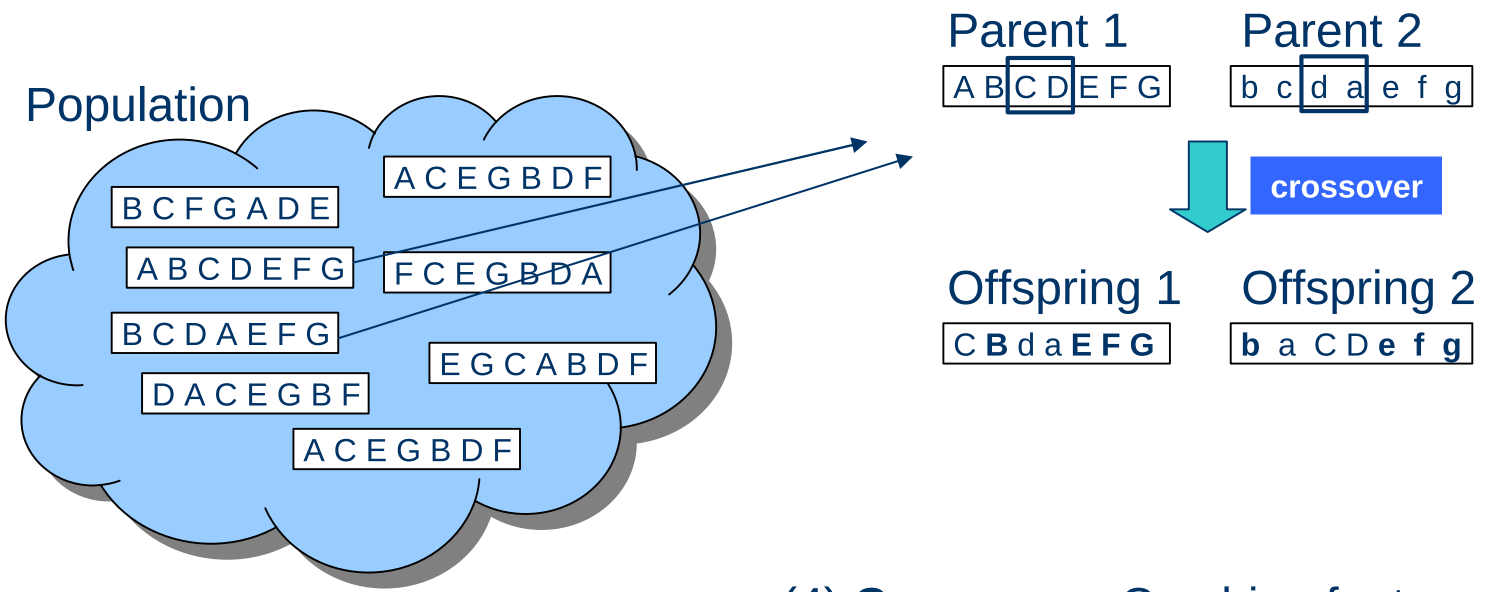
(2) Calculate the cost **(fitness)** of each member of the population

Example: An “Evolutionary Algorithm” for the TSP



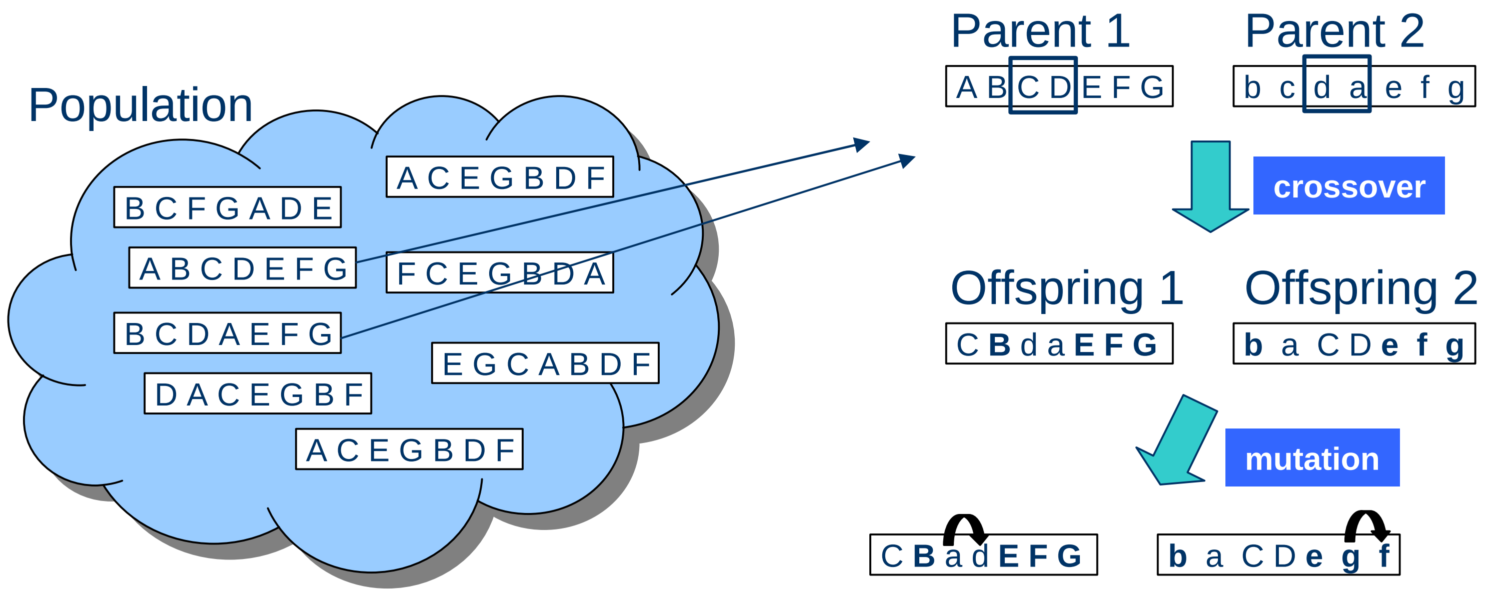
(3) **Darwinian Selection:** Randomly choose two “parent” solutions, biasing towards “fitter” (i.e. better quality solutions)

Example: An “Evolutionary Algorithm” for the TSP



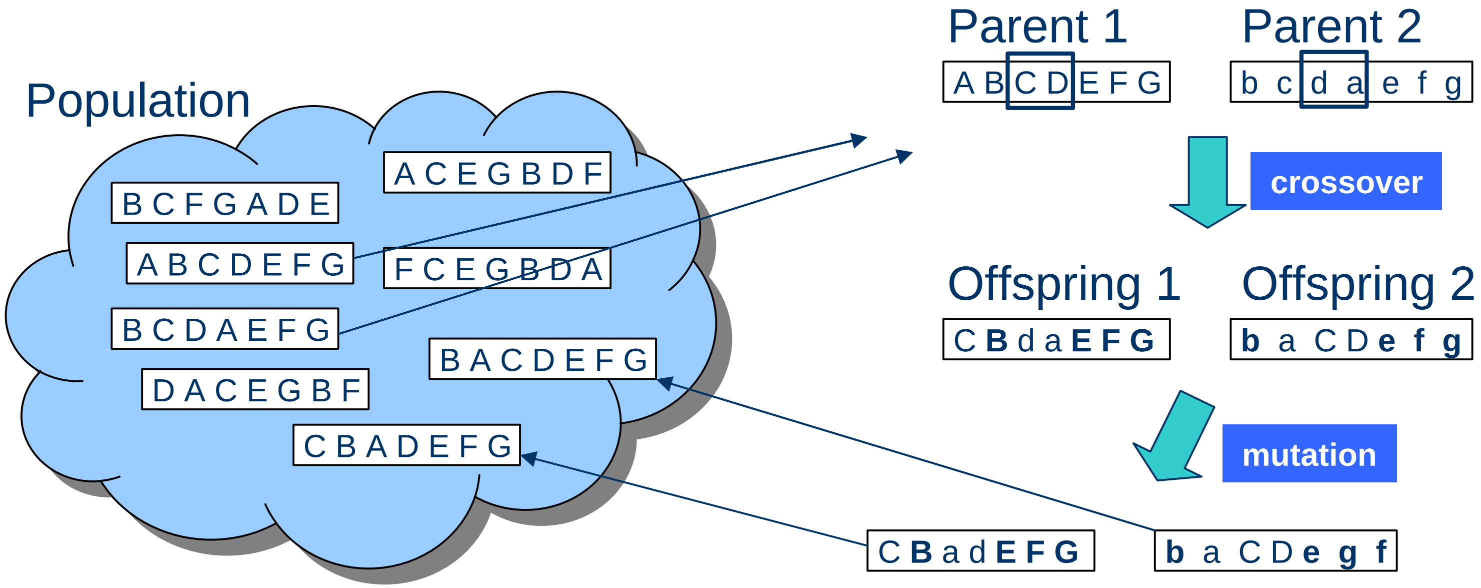
(4) **Crossover:** Combine features of the two parents to form two new “offspring” solutions

Example: An “Evolutionary Algorithm” for the TSP



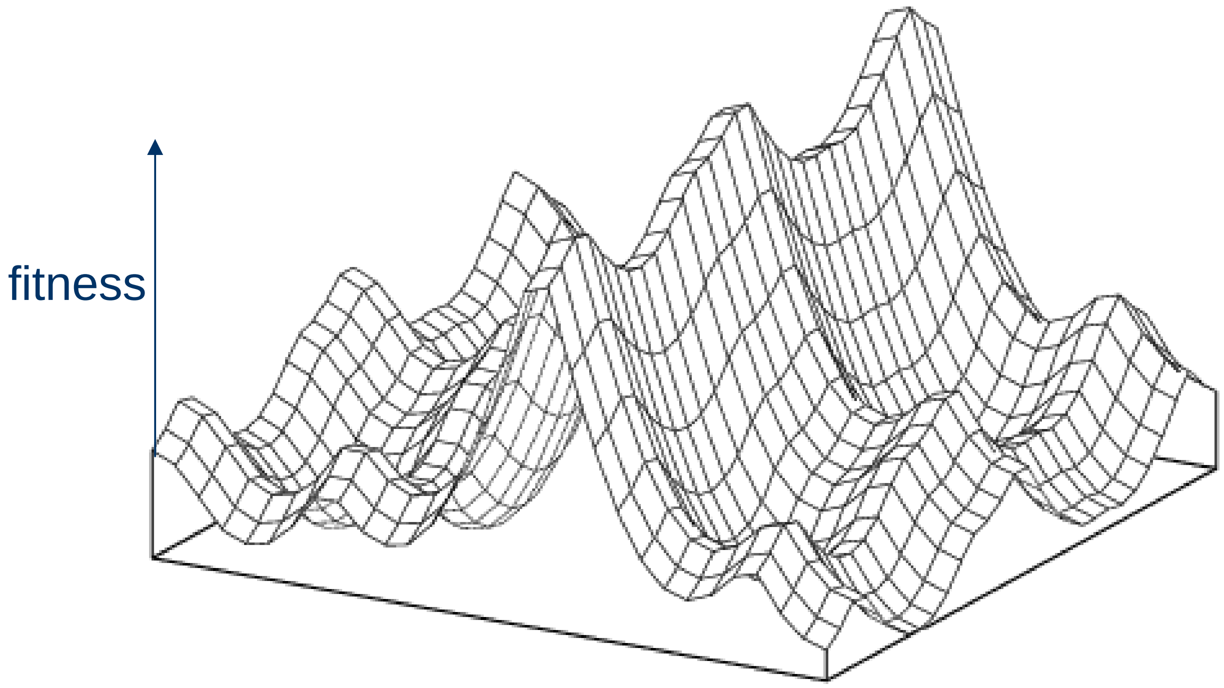
(5) **Mutation:** Make a small number of random alterations to the offspring

Example: An "Evolutionary Algorithm" for the TSP

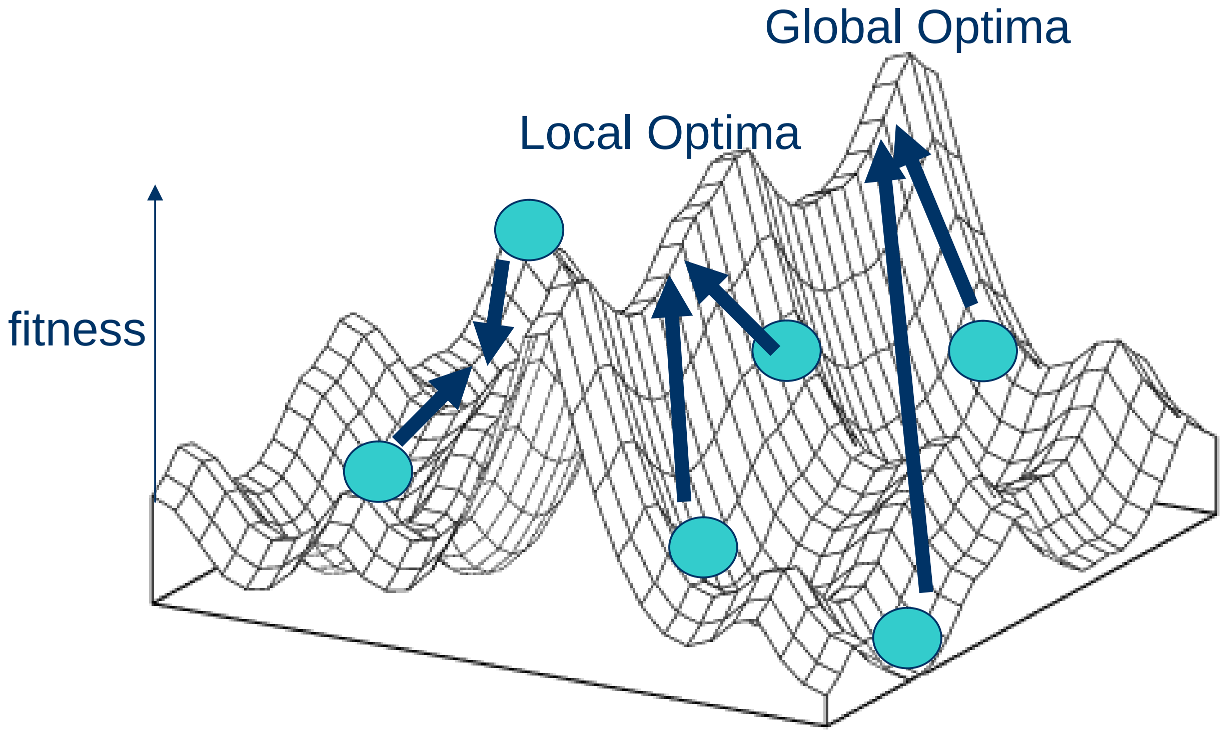


(6) **Replacement:** Reinsert the new offspring back into the population, and go back to Step (3)

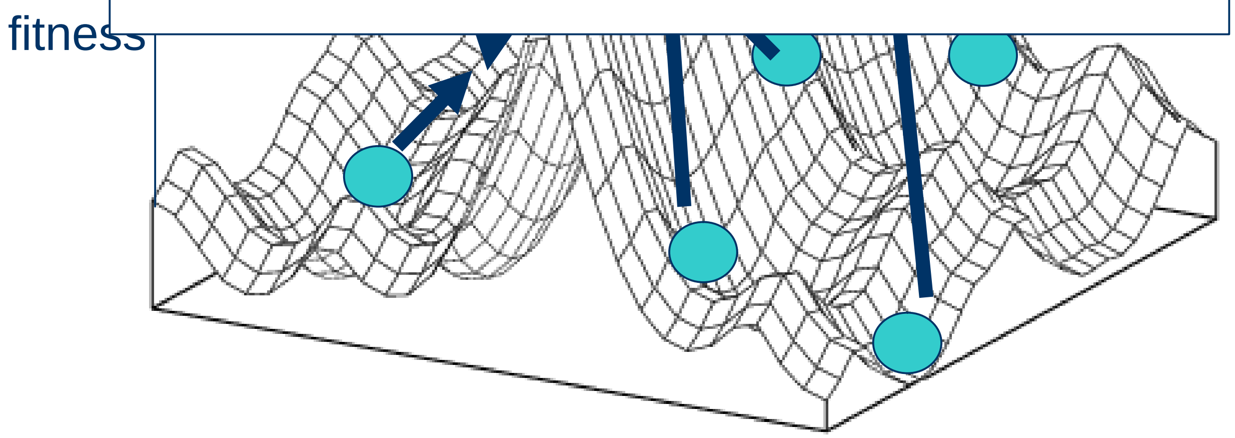
Fitness Landscapes of Combinatorial Optimisation Problems



Fitness Landscapes of Combinatorial Optimisation Problems



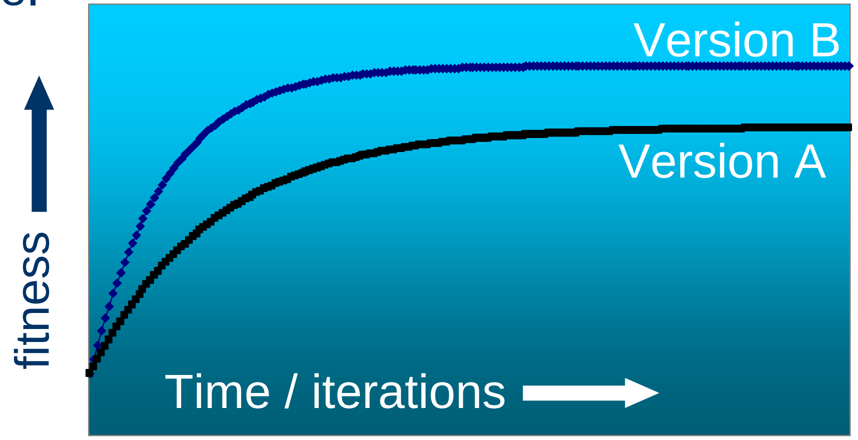
Fitness Landscapes of Combinatorial Optimisation Problems



A demonstration…

Typical behaviour of an Evolutionary algorithm

- As the population evolves, the quality of the solutions in the population tends to increase.



- Typically, the performance of the EA will be affected by choice of:
  - Parameter Settings (Population size, mutation rate, etc.)
  - Types of operators, population policies used etc.

  Unfortunately, this is somewhat of a “black art”

Ant Colony Optimisation (in one slide)

- Another important metaheuristic

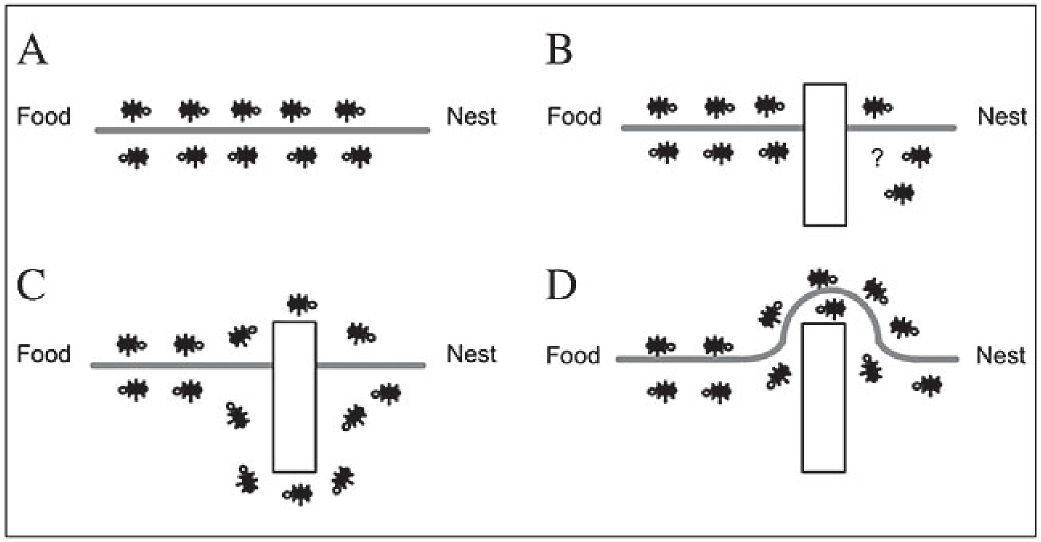


Figure 2. A. Ants in a pheromone trail between nest and food; **B.** an obstacle interrupts the trail; **C.** ants find two paths to go around the obstacle; **D.** a new pheromone trail is formed along the shorter path.

- Like evolutionary algorithms, ACO is applicable to a wide range of problems

Metaheuristics: Research Questions

- Research questions include:
  - What quality of solutions can we expect from our algorithm?
  - How fast is the algorithm?
  - How do the solutions / run-times compare to other methods?
  - How robust/reliable is the algorithm?
  - Is the algorithm more reliable with certain types of problem instances (e.g. those of a certain size)?
- Such questions are usually answered empirically
  - Cardiff's **CONDOR** pool comes in very useful here ☺

Case Study: University Timetabling

Hi Dave - Professor Jones says that he can teach Monday mornings, but he'll need to finish early on Wednesdays, and will need three free hours on Fridays to walk his dog.

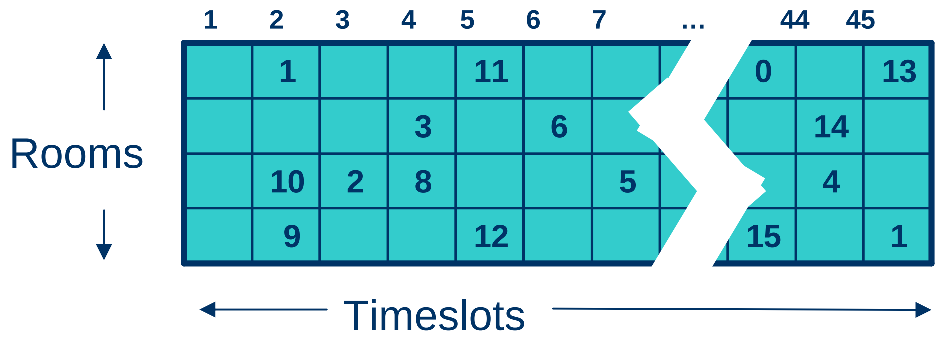
- A problem common to all universities
- Assign "events" to timeslots and rooms while obeying various constraints
- Typically constraints for this problem are idiosyncratic (every university is different)
- Research in the field typically disconnected

Case Study: The International Timetabling Competition *ITC2007*

- **`www.cs.qub.ac.uk/itc2007/`**
- Run between August ’07 and January ’08
- **Idea:** Design the best algorithm for a number of benchmark problem instances.
- Three competition tracks – exam timetabling, curriculum-based timetabling, and post enrolment-based timetabling
- Any type of algorithm was permitted, including commercial software.
- A strict run time limit imposed (approx 5 min. depending on machine and platform)
- Performance judged on solution quality *at the time limit*
- Algorithms were ranked against one another, and performance was verified on the organisers computers

Case Study: The International Timetabling Competition *ITC2007*

- Example: Post enrolment-based course timetabling (track 2)



- Assign each event to a room and timeslot such that:
  - No student or room is double-booked
  - Precedence constraints are obeyed
  - All events occur in *suitable* rooms
- Soft Constraints are also considered, such as:
  - Students should not have to sit three lectures in a row
  - Students should not have a lecture in the 5pm timeslot
  - Students should not have just one lecture in a day

Case Study: The International Timetabling Competition *ITC2007*

- Over 40 entrants from across the globe
- All finalists in each track used metaheuristic-based approaches

Results of Track 2

Rank	Entrants	Affiliation
(1)	Hadrien Cambazard, Emmanue Hebrard, Barry O'Sullivan, and Alexandre Papadopoulos	Cork Constraint Computation Centre, **Ireland**
(2)	Mitsunori Atsuta, Koji Nonobe, and Toshihide Ibaraki	Kwansei-Gakuin University, **Japan**
(3)	Marco Chiarandini, Chris Fawcett, and Holger Hoos	University of Southern **Denmark**
(4)	Clemens Nothegger, Alfred Mayer, Andreas Chwatal, and Gunther Raidl	Vienna University of Technology, **Austria**
(5)	Tomas Müller	Purdue University, **USA**

Summary + Conclusions

- Metaheuristics an effective tool in our armoury against intractable problems
- General algorithmic frameworks applicable to a wide range of problem types.
- However:
  - There is no “one-size fits all” policy, different approaches *seem* to work well with different problems.
  - Development times are often high
  - Theoretical studies are difficult. Algorithm design is often considered an art, and analysis is usually empirical
  - Difficult to state bounds on solution quality

Summary + Conclusions

- Metaheuristics an effective tool in our armoury against intractable problems
- [illegible]
- [illegible]
  - considered an art, and analysis is usually empirical
  - Difficult to state bounds on solution quality

Thanks for listening!

- TSP Demo Program by Konstantin Boukreev

`www.codeproject.com/KB/recipes/tspapp.aspx`

- Speaker, Rhyd Lewis

`www.cardiff.ac.uk/maths/contactsandpeople/profiles/lewisr9.html`