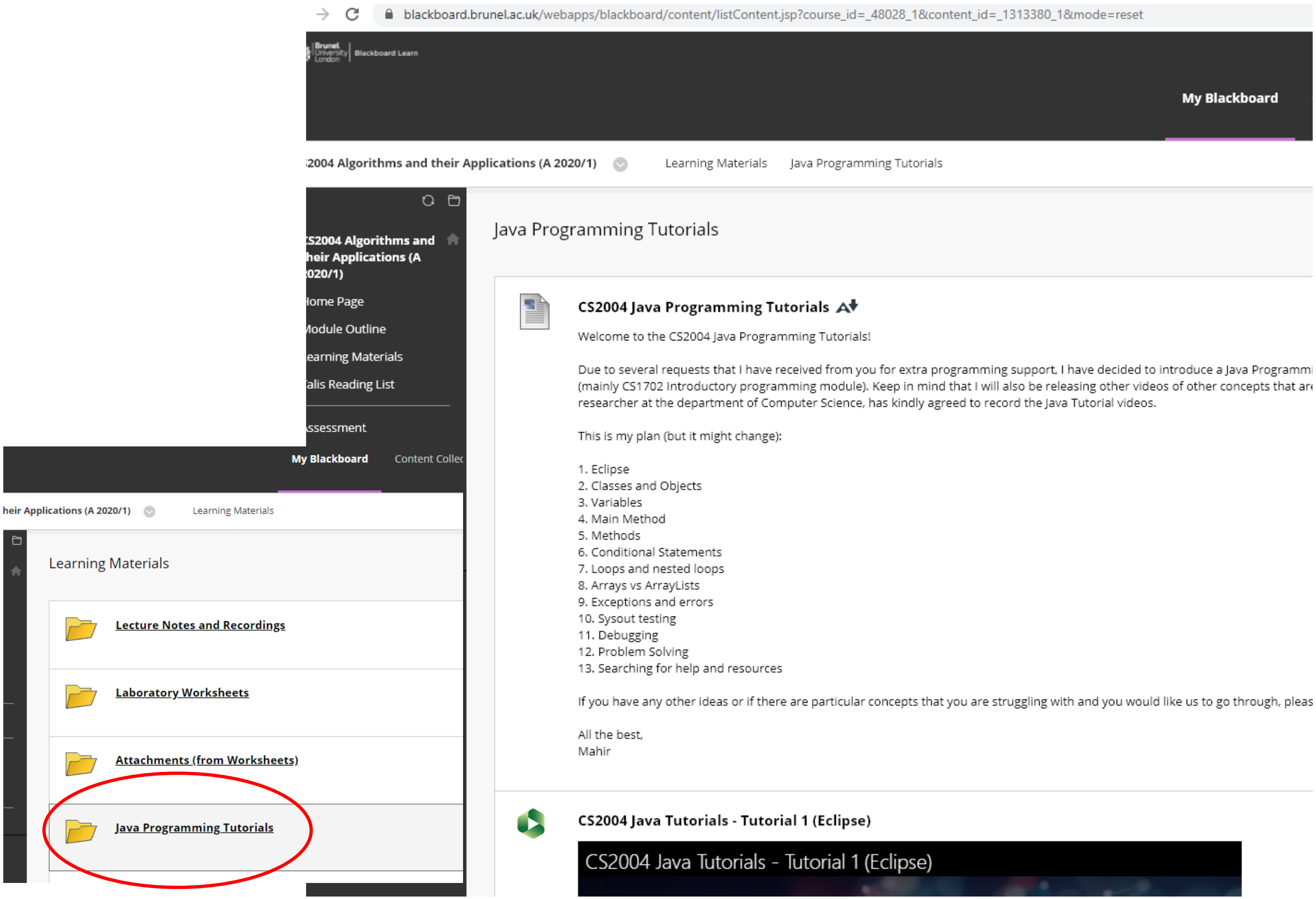
Algorithms and their Applications CS2004 (2020-2021)

Dr Mahir Arzoky

9.1 Search Algorithms, Representation, Fitness and Fitness Landscapes

Notices

Basic Java Programming Tutorials…



CodeRunner Tests So Far…

- Class Test CRI
  - Only 133 attempted the test
- Class Test CRII
  - 34 students attempted the test
- The four class tests' marks will be averaged to give the mark for Task #1
- All class tests must be passed to pass Task #1
- Task #1 weighs 30% of the coursework aspect of the module
- If you have not passed Task #1, you can still attempt Task #2 but you will be capped at D-grade

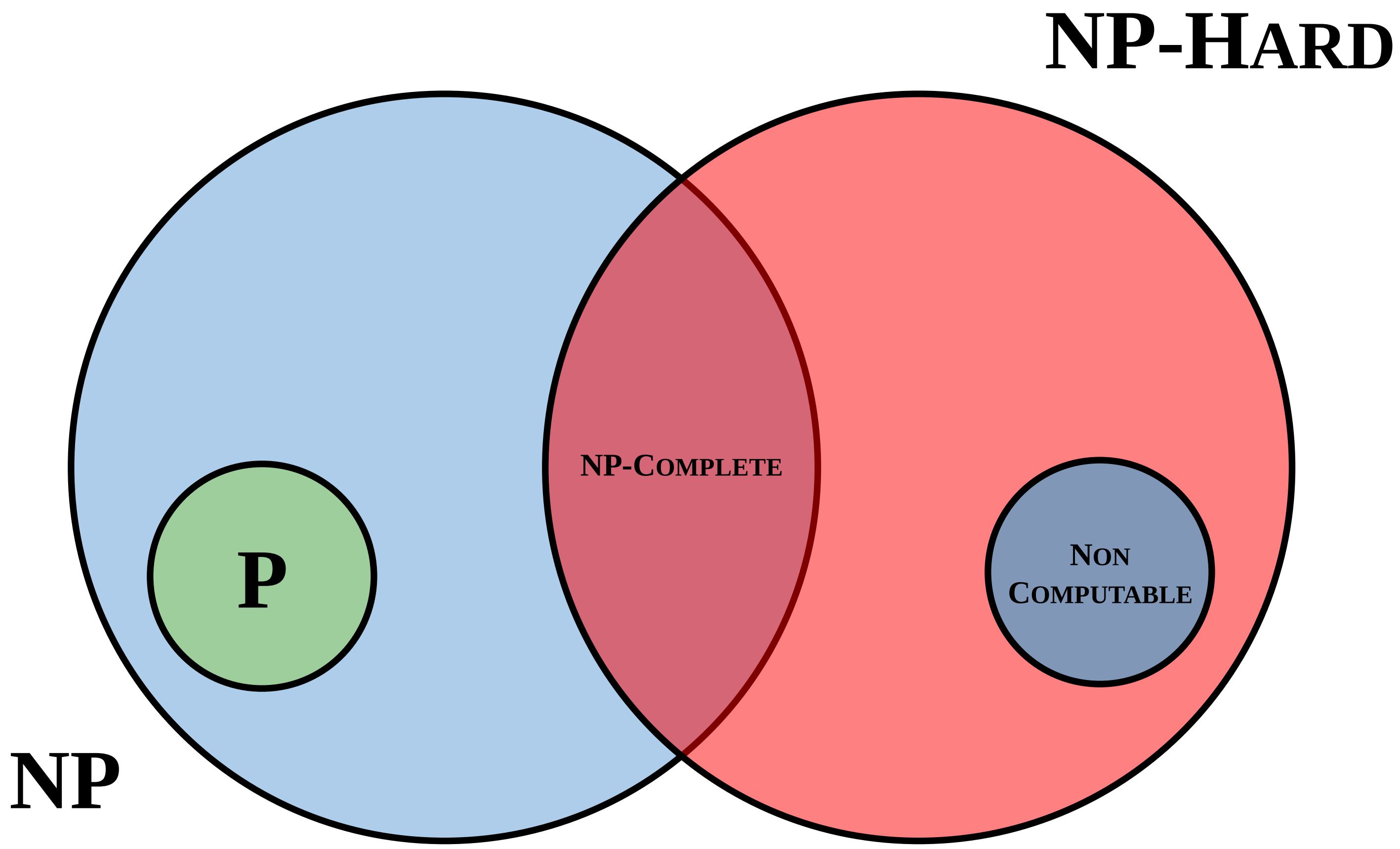
Previously On CS2004…

- So far we have looked at:
  - Concepts of Computation and Algorithms
  - Comparing algorithms
  - Some mathematical foundation
  - The Big-Oh notation
  - Computational Complexity
  - Data structures
  - Sorting Algorithms
  - Various graph traversal algorithms

Search and Fitness

- Within this lecture we are going to look in more detail about the concepts of search and fitness
- This starts the “second” half of the module where we look at:
  - Heuristics
  - Approximation algorithms
  - Natural (nature inspired) computation
  - Applications

The Classes of Algorithms - Recap



NP-Hard Problems - Recap

- It is a class of problems that can not be solved in the 'traditional' way
- Typically these problems:
  - Have a "best known" computational complexity in exponential time, e.g. $O(n) = 2^n$
  - E.g. $2^1=2$, $2^{10}=1024$, $2^{95}=$ 39,614,081,257,132,168,796,771,975,168
  - They have no direct analytical solution
- Thus, we may have to approximate a solution to them

Definition of a Search Problem

- For some problems we need to search for a solution from a (usually) very large number of possibilities
  - **Search problems**, i.e. searching for the answer in some **solution space**
- The solution spaces for many every day problems can be very, very large, too large to search exhaustively
- The search algorithm may not find the optimal solution to the problem, however, it will give a good solution in **reasonable time**
- We often need special search algorithms known as **meta-heuristics**

What is a Heuristic?

- A heuristic is a "rule of thumb" or some loose set of guidelines
  - That may find a solution (but not guaranteed)
  - E.g. getting out of a maze by keeping your hand against the maze wall
- In Artificial Intelligence these are used to improve the performance of methods, in our case, search methods
  - Expert knowledge
  - Common sense
  - We will look into specific methods later...

Search Problems

- Many problems can be thought of as searching through candidate solutions to find one that is optimal
  - Systematically search through potential solutions **without** considering all of them…
- We need some way to evaluate the fitness of the candidate solutions
  - Optimal = the fittest = the best etc…
- Also some sense of the adjacency (similarity) of solutions (or neighbourhood)

Fitness

- In order to search for a solution we must be able to compare potential solutions
- E.g. Is solution s_1 better than solution s_2?
  - Thus we have the concept of fitness
- We must derive a function (the **fitness / objective function**) that maps a solution to a value that rates how good the solution solves the problem in hand
- We either try and **maximise** or **minimise** the fitness
- A slight change in solution quality should result in a corresponding change in the fitness
  - Solution quality goes down → Fitness goes down (decreases)
  - Solution quality goes up → Fitness goes up (increases)

Fitness Landscapes – Part 1

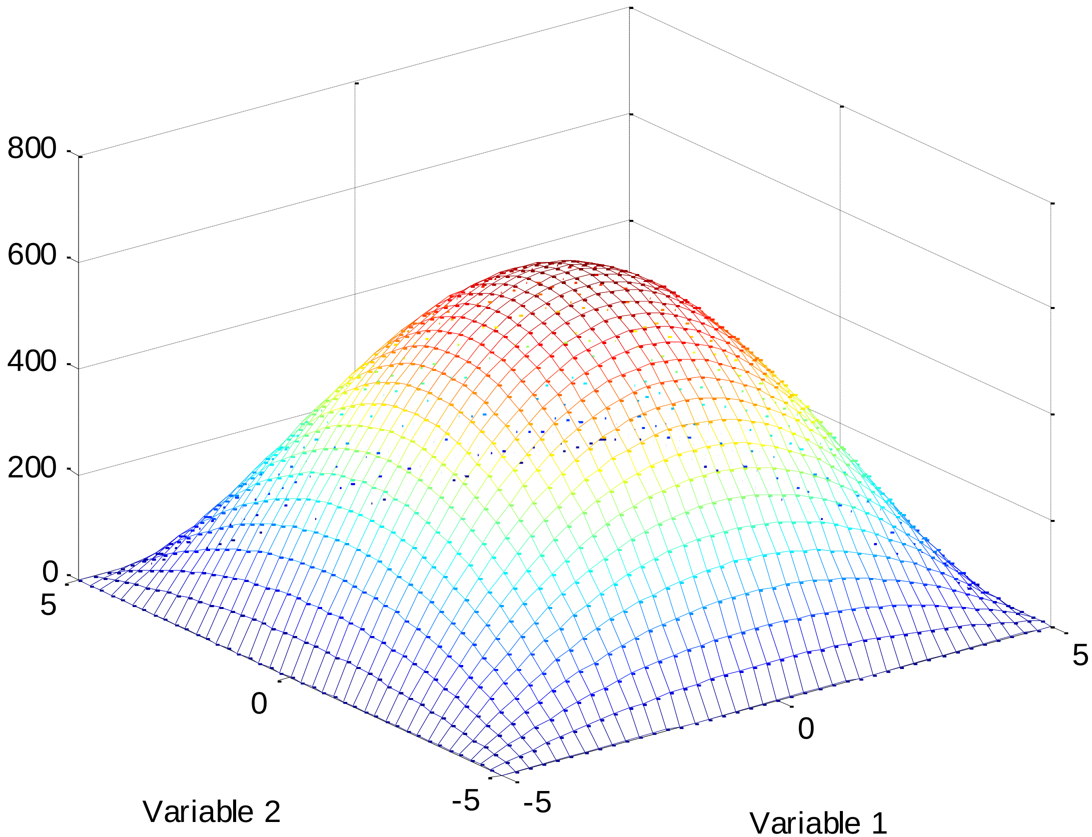
- It can be **helpful** to think of searching a landscape of solutions where:
  - The x and y co-ordinates represent a particular solution
  - Altitude (z axis) represents the fitness of that solution
- This leads to the analogy that we wish to search for or climb peaks (or descend…)
- Describing the nature of a landscape characterises (in an abstract way) classes of problems

Fitness Landscapes – Part 2

- The collection of **all possible solutions** can be considered as a high dimensional space, called the **search space** or **fitness landscape**
  - Each point in the search space represents one possible solution
- Often some concept of distance between solutions exists and some concept of how "good" each point in the search space is (fitness/objective function)
- Techniques exist to map a high dimensional space to a two dimensional space so we can plot the space/landscape (we will not be covering these!)

Fitness Landscapes – Part 3

- Smooth and regular spaces are easy to search

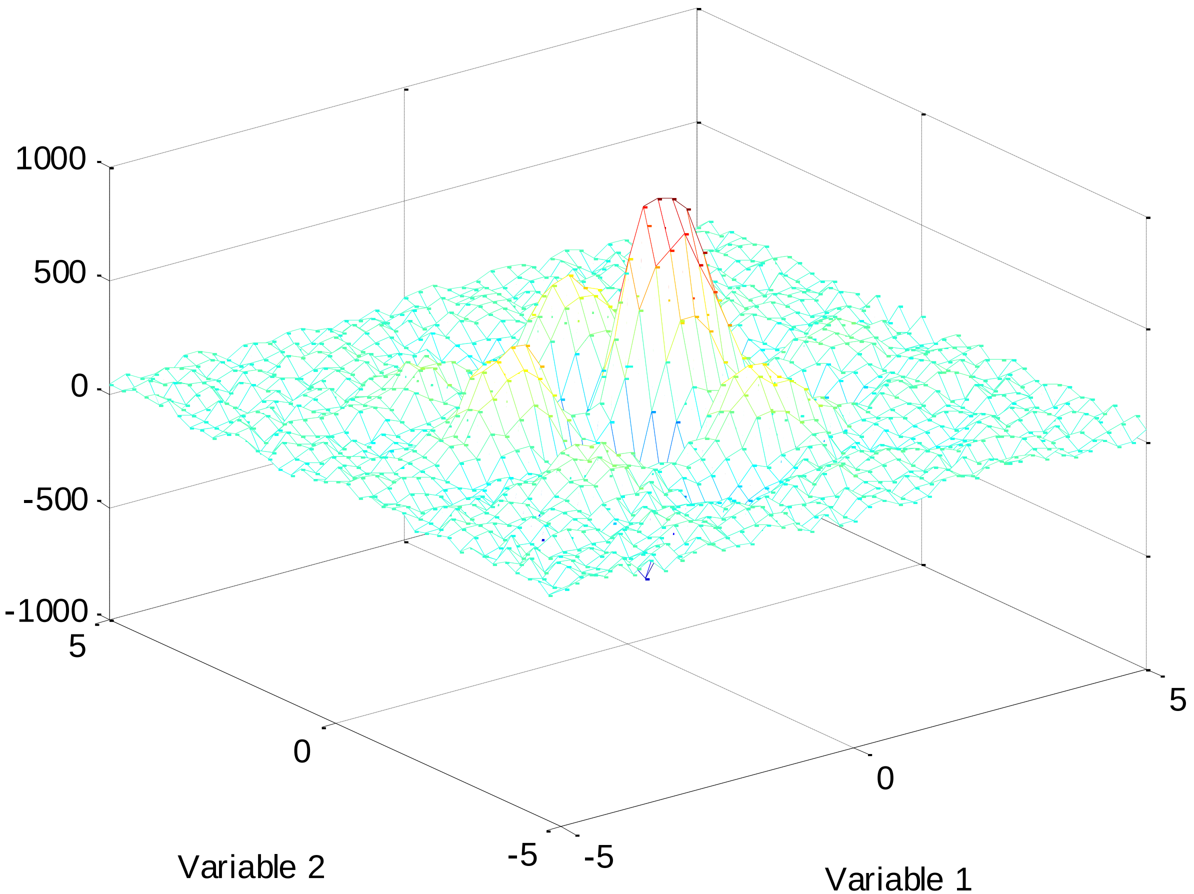


The x and y co-ordinates represent a particular solution

Altitude (z axis) represents the fitness of that solution

Fitness Landscapes – Part 4

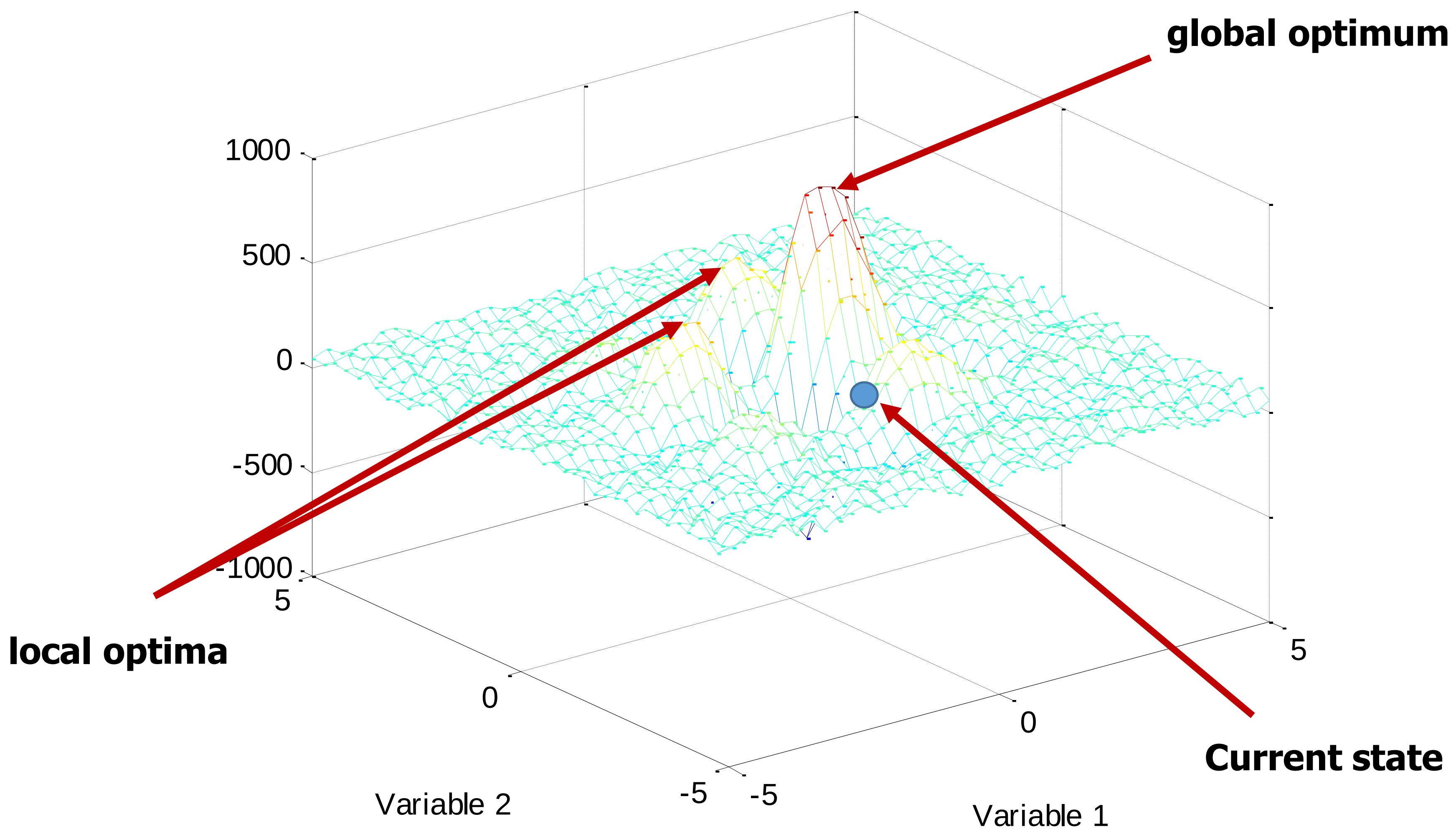
- Irregular spaces are difficult to search



Global and Local Optima – Part 1

- The **global optimum** is the point or points in the search space with the best objective function evaluation
- A **local optimum** is the point or points in a subset or section of the search space with the best objective function evaluation
- Note that the subset or section of the search space in question may contain the global optima
- Many search techniques can find local optima, but get "stuck" at them and cannot move on to find the global optima
  - E.g. Random Mutation Hill Climbing [we will cover next week…]

Global and Local Optima – Part 2



Representation

- When we are trying to solve a search based problem we need a way to represent a potential solution
- This is usually a mathematical and/or data structure based way of describing the solution to a problem
- A good representation:
  - Should be a one to one mapping
  - No redundancy
  - No ambiguities
  - All potential solutions should be represented

The Scales Problem

- We are going to spend some time looking at a particular problem in detail
- We are going to design the fitness function for this problem and use it in a number of worksheet exercises
- The aim is to see how a number of different heuristic search methods perform against this problem

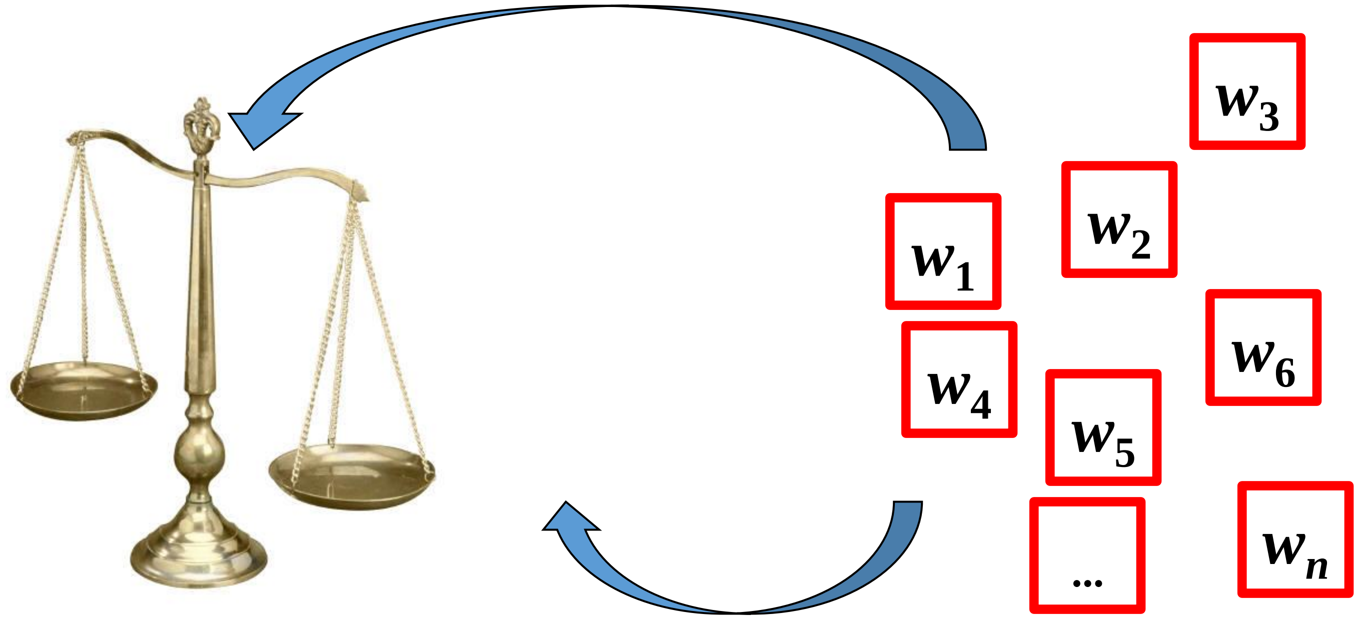
The Scales Problem

Given n objects of various weights,

split them into two equally heavy piles

(or as equal as possible)

Scales – Part 1



- We have n weights that are $W = [w_1, w_2, \ldots, w_n]$ in weight, $w_i > 0$
- We want to divide them into two equal in weight, or as equal as possible piles
- We are going to work with the general case, not a specific set of given weights, we want to solve the problem for **any** W

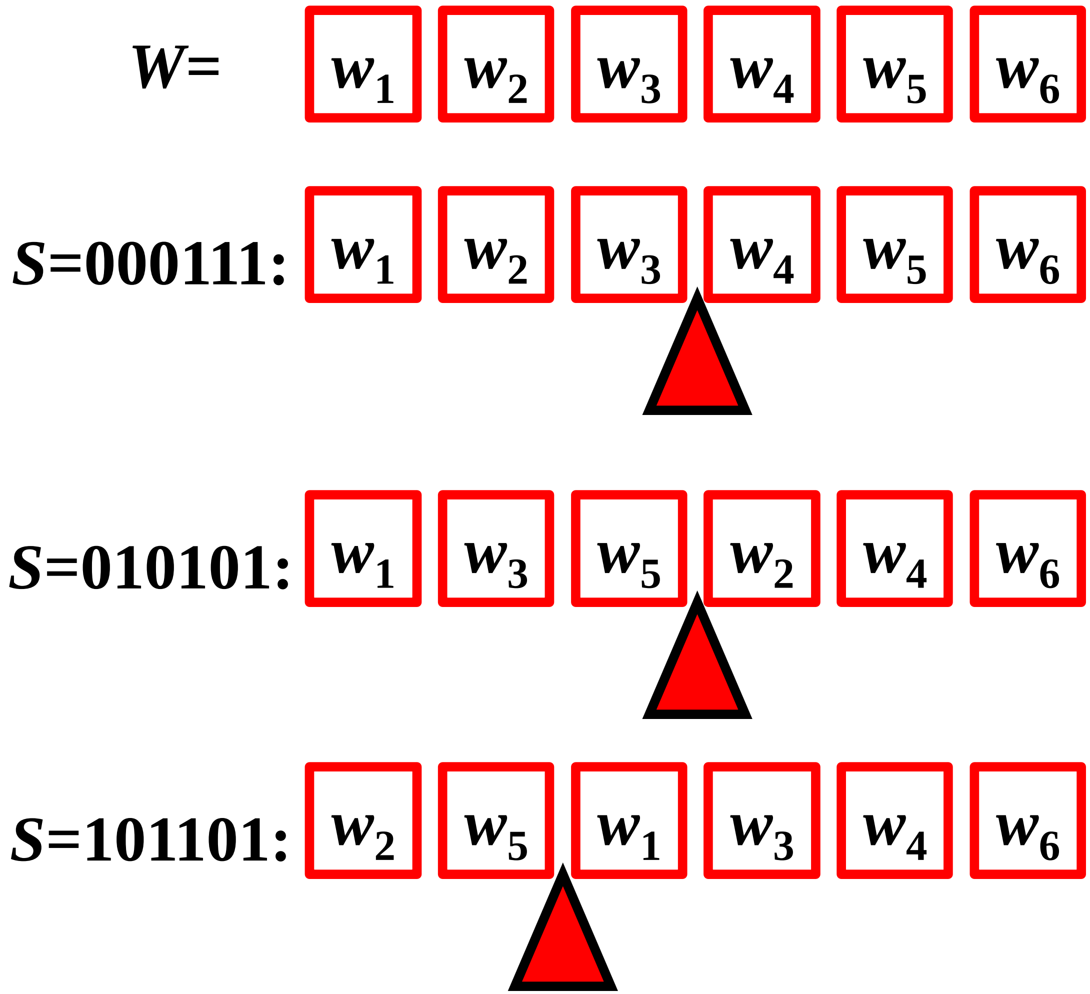
Scales – Part 2

- We first look to see if there is a simple or standard method to solve the problem
  - Do not reinvent the wheel!
  - For this example problem, there is not!
- We then need to:
  - Design a representation
  - Construct a fitness function
  - Apply a heuristic search method

Scales – Representation – Part 1

- Each weight (w_i) is either on the left hand side or the right hand side
- Given that we have n items into two piles/sets we could use a binary representation
- We represent a solution as an n length binary string (or array/vector/list…) where:
  - A zero (0) in position i means that weight i is on the left side of the scales
  - A one (1) in position i means that weight i is on the right side of the scales
  - This can represent all possible allocations
  - We will refer to this string as S and each bit as s_i
  - If s_i = 0 then weight i is on the left hand side pan/scale
  - If s_i = 1 then weight i is on the right hand side pan/scale

Scales – Representation – Part 2

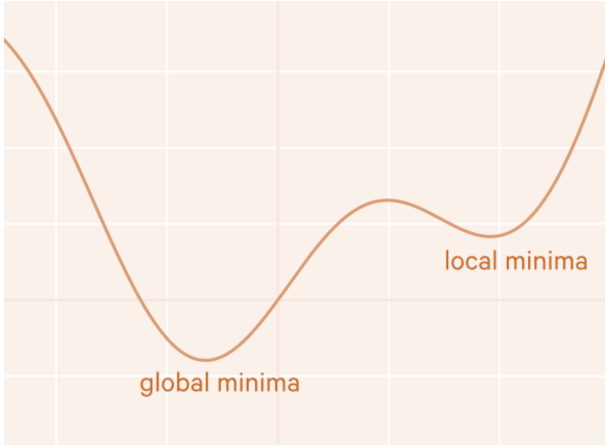


Scales – Fitness – Part 1

- We now need to design an appropriate fitness function
- This function should score how good a solution (a binary string) is at solving our problem
- What's the aim of the problem?
  - Equal balancing (or as near as possible)
- Balanced would be when the sum of the weights on the left hand side (LHS) of the scales equals the sum of the right hand side (RHS)

Scales – Fitness – Part 2

- The worse this difference is the worse our solution is at solving the problem
- What is the best fitness?
  - Zero (balanced)
- How about the worst fitness?
  - When all the weights are on either the left hand side or right hand side
- Thus, this is a **minimisation** problem
- Let L = Sum of LHS weights
- Let R = Sum of RHS weights
- Fitness = $|L-R|$
  - Why take the absolute value?



Scales – Fitness – Part 3

- So our fitness function will take two parameters
  - A potential solution – binary string of length n
  - A set of weights – a real vector/array of length n
- It will then return a real number
- Each weight w_i will either be added onto the left hand side L or the right hand side R
- So we can iterate through each weight adding it to L or R depending on what side of the scales the representation specifies the weight is on

The Fitness Function for Scales

```
Algorithm 1. ScalesFitness(S,W)
Input: S – a binary string of length n
       W – a real vector/array of weights of length n
1) Let L = 0
2) Let R = 0
3) For i = 1 to n
4)     If s_i = 0 then
5)         Let L = L + w_i
6)     Else
7)         Let R = R + w_i
8)     End If
9) End For
Output: |L-R| - the difference in weight between sides
```

Initialise L and R

Iterate through all weights

Check which side

Add to left side

Add to right side

Return difference

Scales Fitness – Example – Part 1

- So imagine we have five weights
- $W=\{1,2,3,4,10\}$
- $w_1=1$, $w_2=2$, $w_3=3$, $w_4=4$, $w_5=10$

- Some example fitness (F):
  - S=11111
    - $L=0$, $R=1+2+3+4+10=20$, $F=|0-20|=20$
  - S=10101
    - $L=2+4$, $R=1+3+10$, $F=|6-14|=8$

Scales Fitness – Example – Part 2

- $W=\{1,2,3,4,10\}$
- $w_1=1$, $w_2=2$, $w_3=3$, $w_4=4$, $w_5=10$
  - $S=01010$
    - $L=1+3+10=14$, $R=2+4=6$, $F=|14-6|=8$
  - $S=11110$
    - $L=10$, $R=1+2+3+4=10$, $F=|10-10|=0$

Scales – What Next?

- We have a representation
- We have a fitness function
- We now want to search through a number of possible S until we find the best one
  - I.e. When $F(S,W)=0$
  - (or as close to zero as possible)
- We therefore need to apply an appropriate heuristic search method
  - Random Mutation Hill Climbing, Stochastic Hill Climbing, Random Restart Hill Climbing, Simulated Annealing, Genetic Algorithms, Particle Swarm Optimisation, Tabu-Search, Iterated Local Search, Evolutionary Programming, etc...
  - We will cover all of these (and more)!!!

This Week's Laboratory

- You will be implementing the **Scales Fitness Function**
- This laboratory is **VERY** important, it is needed for several of future worksheets
- It is important to get this right!

Next Lecture

- We will be looking at some Heuristic search algorithms
- We will apply them to the **Scales** problem...