Theory Lecture 9

Optimisation

Learning Objectives

- Introduce the concept of optimisation
- Understand gradient descent
- Look at some optimisation heuristics

Solving by Optimisation

Example Problem: Maximum Cut

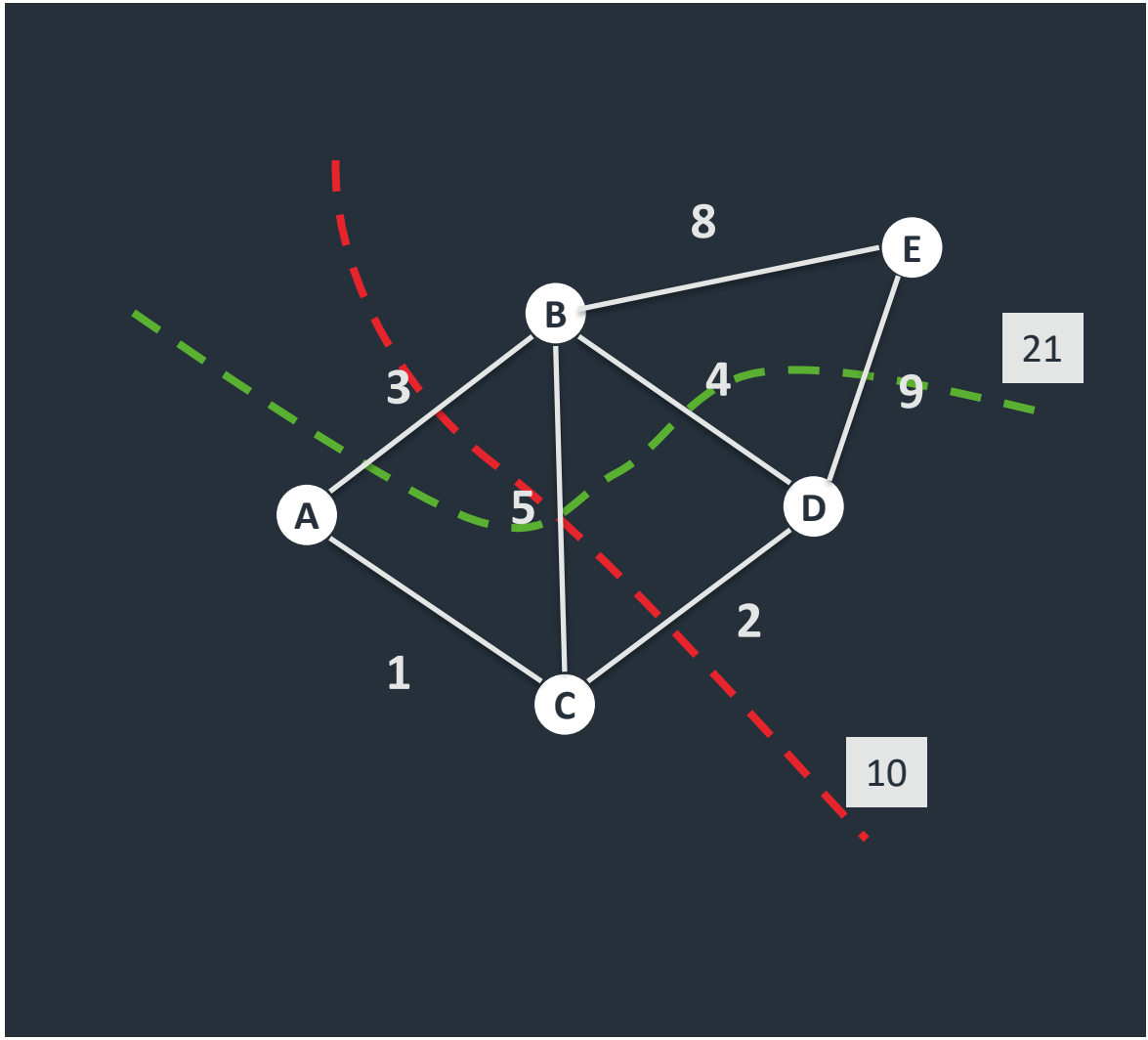
A graph cut is a partition of the vertices of a graph into two sets. **The value of the cut is the sum of the weights of cut edges.**

The maximum cut problem is finding the partition with the maximum sum of weights.

Each vertex must be in partition 1 or partition 2, so we can represent a possible solution by a set of binary variables, $b(i)$ **=0 if vertex** i **is in partition 1**, and $b(i) = 1$ **if it is in partition 2.**

There are 2^{n-1} possible solutions, too many to search for a large graph.

e.g., [00000] ~ [11111]; [11100] ~ [00011]



Optimisation

Many problems can be described in terms of an optimisation problem. We define a **score function**, which is either maximum or minimum at the solution to the problem. We then try to **find that maximum or minimum**.

Example: In max-cut, the score function is the weight of the cut and the problem is

$$\max_B S(B) = \max_B \sum_{b(i) \neq b(j)} W(i,j)$$

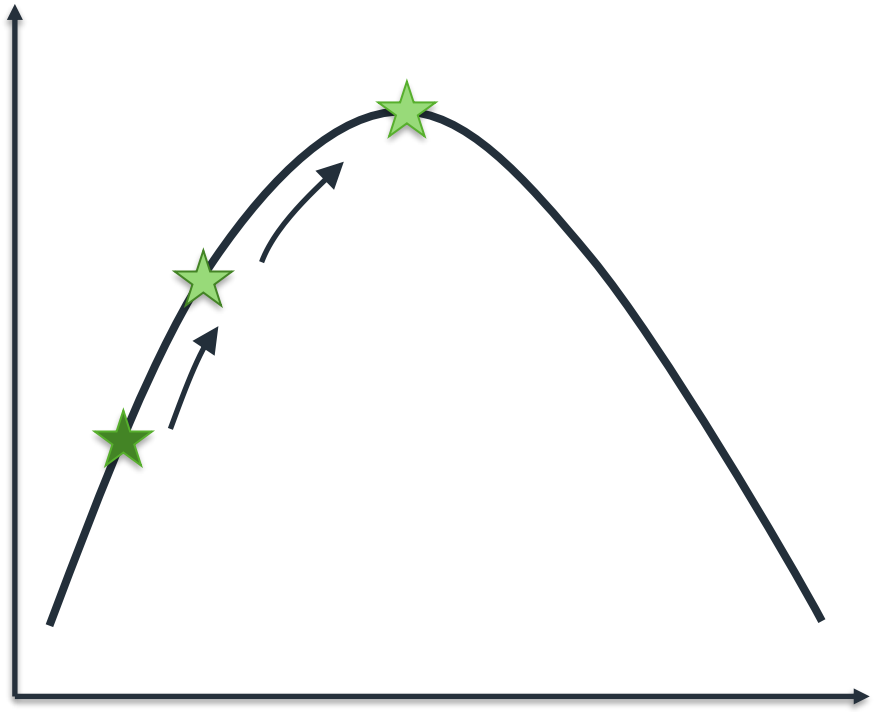
Example 2: In the n-queens problem from the last problem set, we could score the number of attacks on a queen. The solution is the minimum of this score (i.e. 0).

Optimisation(2)

Finding the minimum is as hard as solving the original problem. However, the score function tells us something about the quality of all PSs and we can use this to develop **optimisation heuristics**.

Consider the score function on the right.
Start at some random point.
Move in the direction of increasing gradient.
We will reach the maximum – the solution

This is **gradient ascent**.

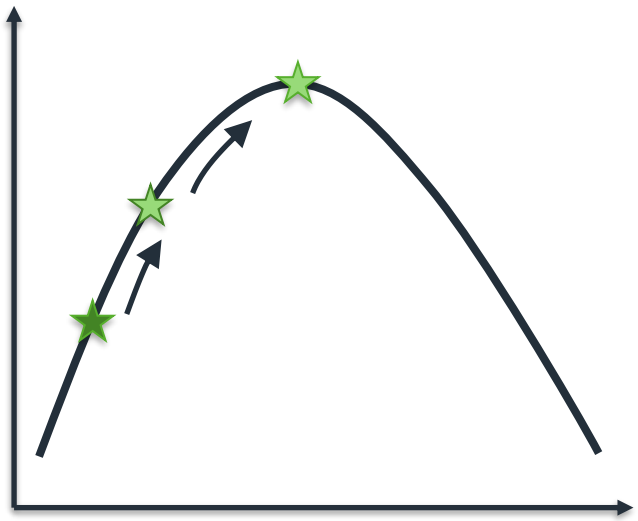


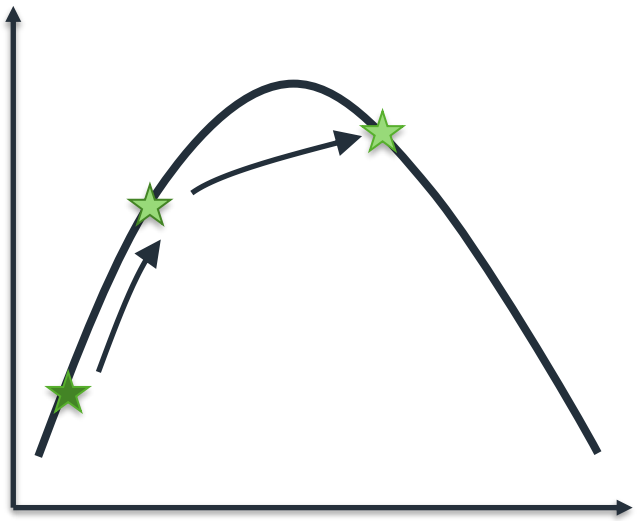
Gradient Ascent

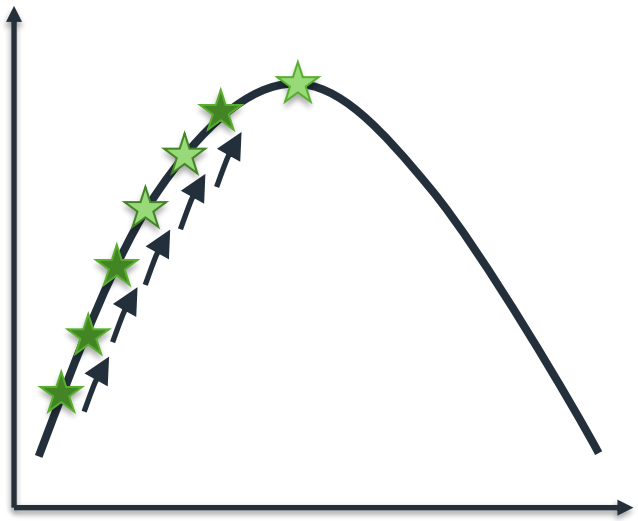
Gradient ascent is an example of a **local heuristic** in optimisation. We look in the local neighbourhood of the current PS to find a better PS (according to the score function).

A **move** is the operation which takes us from one configuration to a nearby one.

In the previous diagram, a move was one step to the left or right.







Gradient Ascent

Gradient ascent is an example of a **local heuristic** in optimisation. We look in the local neighbourhood of the current PS to find a better PS (according to the score function).

A **move** is the operation which takes us from one configuration to a nearby one.

In the previous diagram, a move was one step to the left or right.

Example: In the max-cut problem, a move could be swapping a vertex from one partition to the other.

We explore all allowed moves, and select the one with the best score.

Repeating this takes us to the maximum (or minimum).

Gradient Ascent (2)

Let $N(m) = \{m' | m \to m'\}$ be the set of all moves we can take from m. These are called the **neighbours of m**.

Then gradient ascent chooses the move

$$m^* = \arg \max_{m' \in N(m)} S(m')$$

i.e. the move from the set of allowed moves with the best score.

Example: max-cut

$$N(B) = B \cup \{(b_0, b_1, \dots, \neg b_i, \dots, b_{n-1}) \forall i\}$$

$$B^* = \arg \max_{B' \in N(B)} \sum_{b'(i) \neq b'(j)} W(i,j)$$

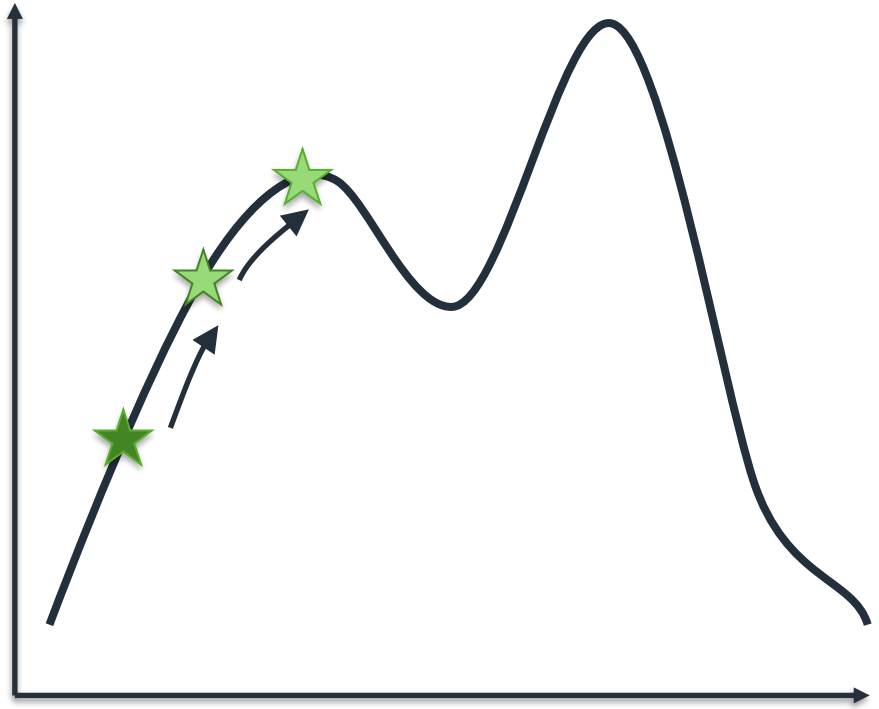
We explore n new configurations at each step.

Problems with Gradient Ascent

Gradient ascent is **a heuristic and does not guarantee to find the best solution**. There are a number of potential problems.

Local maximum – the method will find a maximum, but it may not be the highest maximum in the score function.

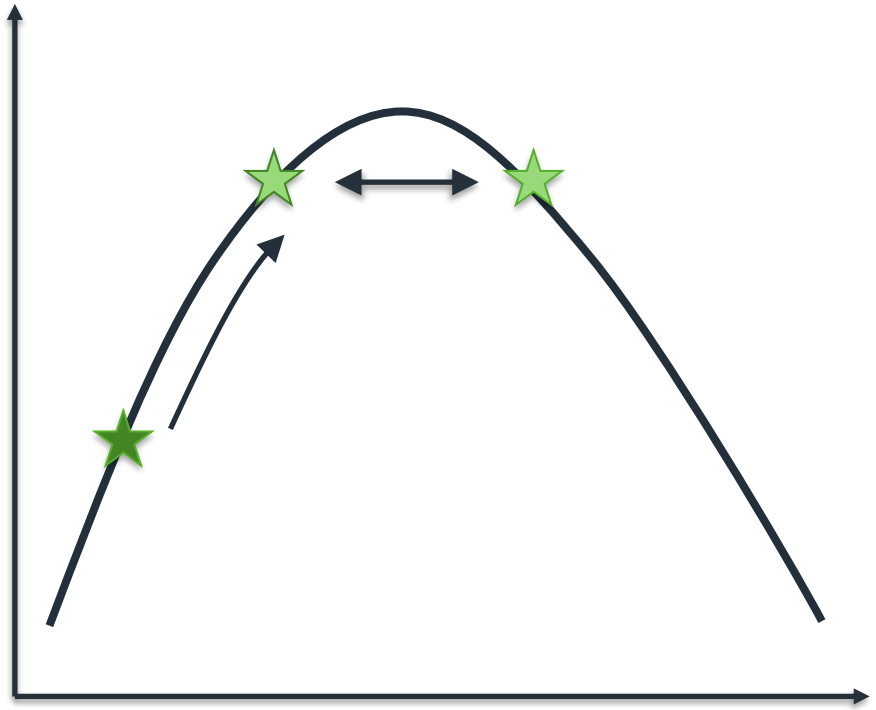
We do not know about the existence of a better solution elsewhere.



Problems with Gradient Ascent (2)

Gradient ascent is a heuristic and does not guarantee to find the best solution. There are a number of potential problems.

Overshoot – if the moves are too big, we may overshoot the maximum. Too small and it may take many steps to find the solution.



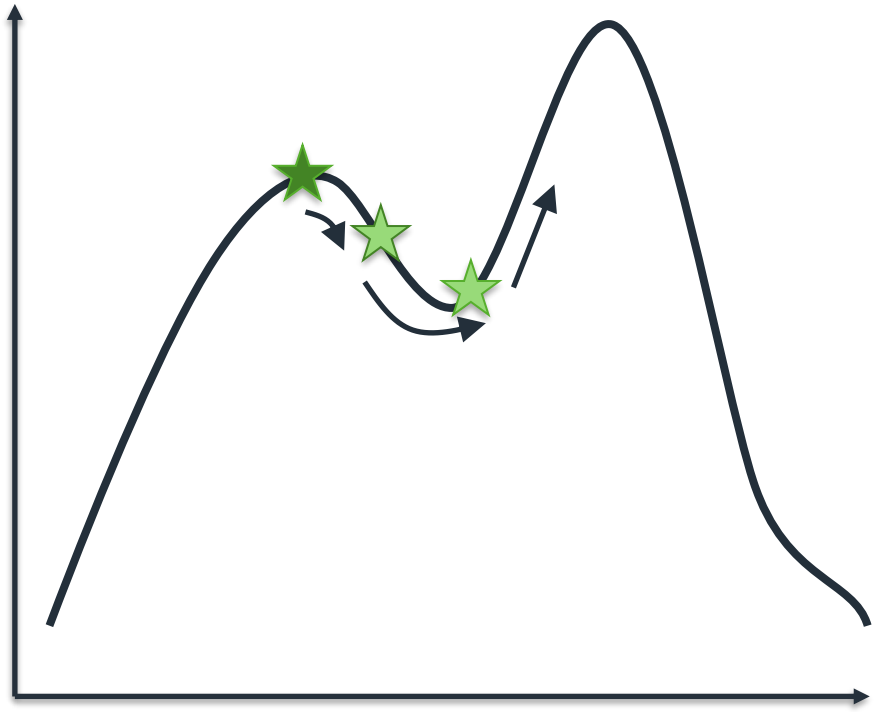
Simulated annealing

Simulated annealing is a heuristic for optimisation which adds a random element to moves, to help overcome the problem of local maxima.

Moves are allowed which **decrease the score**.

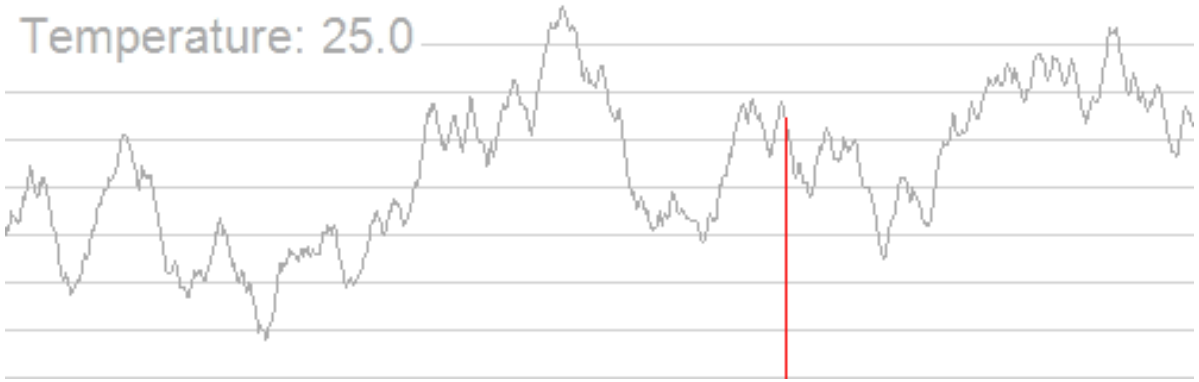
A sequence of bad moves can get out of the local maximum and find a better one elsewhere.

This process is similar to the physical cooling of metal, hence the name.



Simulated annealing

Simulated annealing is a heuristic for optimisation which adds a random element to moves, to help overcome the problem of local maxima.



Simulated Annealing (2)

Simulated Annealing

```
Initial random possible solution m
Initialise temperature T
while T>Tmin
   select random m′ from N(m)
   if S(m′)>S(m) then m=m′
   else if P(m′,m)>rand_uniform(0,1)
      m=m′
   endif
   reduce T
endwhile
```

Moves are chosen at random from the neighbours.

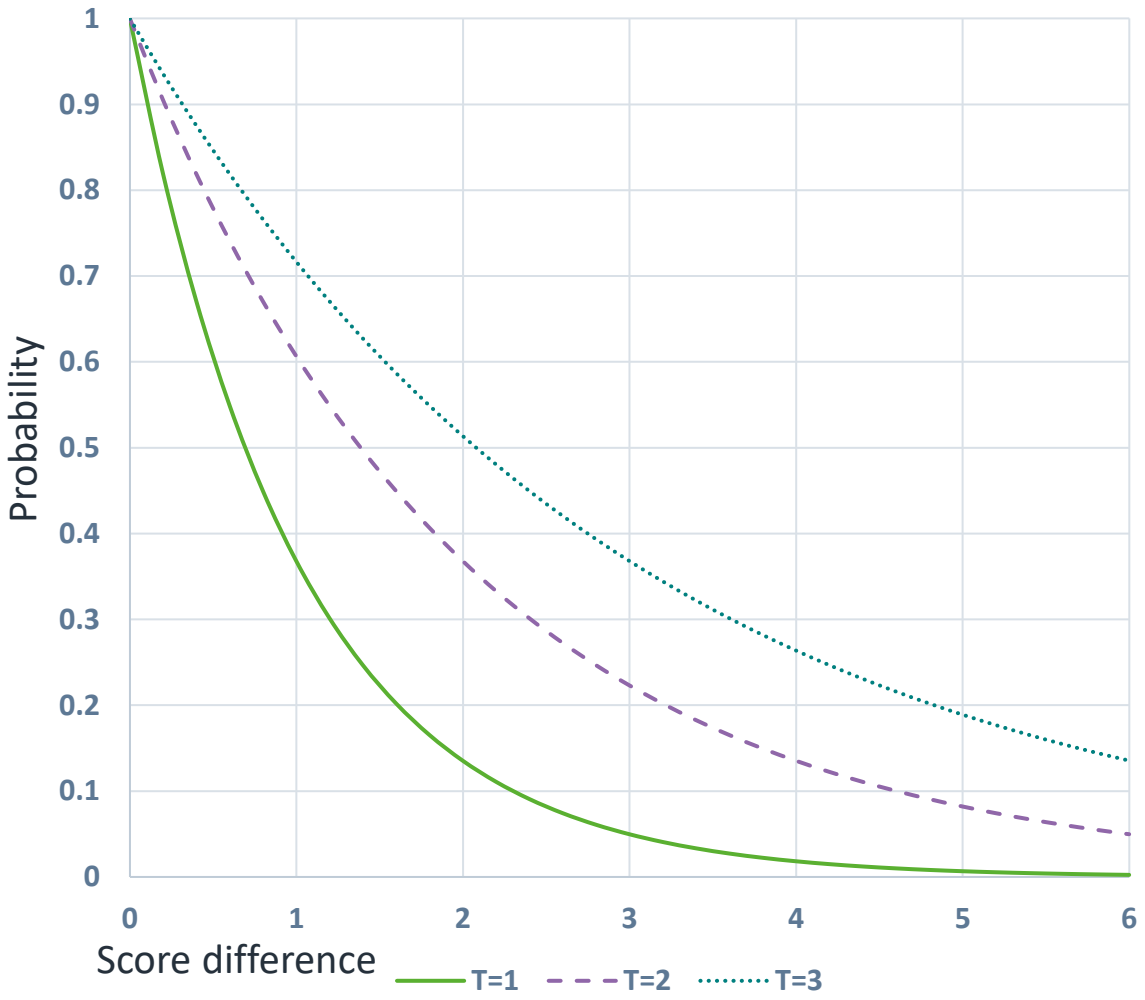
If the new PS is better, it is accepted.

If the new move is worse, it is accepted with a probability

$$P(m', m) = \exp\left(\frac{S(m') - S(m)}{T}\right)$$

We reduce *T* gradually throughout the algorithm.

Simulated annealing



The temperature is critical to simulated annealing.

- T should be the same scale as differences in scores
- T begins high, which readily allows moves which reduce the score
- T is gradually reduced. This penalises bad moves more strongly
- As $T \rightarrow 0$, score-decreasing moves are not allowed at all. This is then (random) gradient ascent

Problems with Simulated Annealing

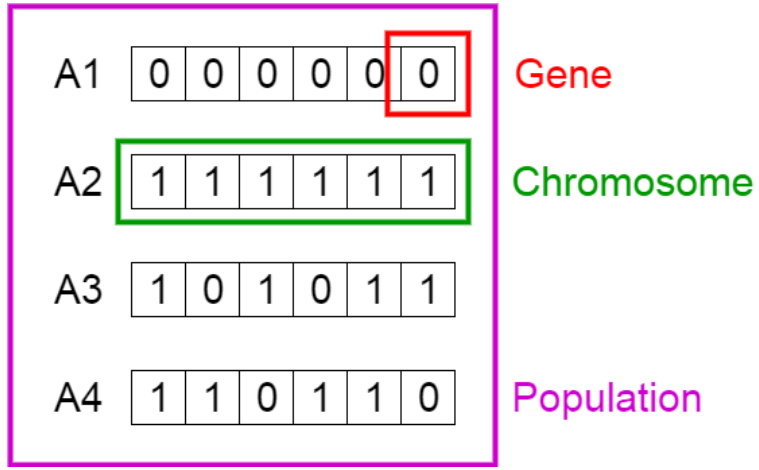
Simulated annealing can overcome the problem of local maxima, but

- The moves have a **random element**, so **slow** compared to gradient ascent
- The **temperature** must be set and controlled properly to avoid local maxima
- It is not clear before running SA how to **control the temperature properly**

Genetic Algorithm

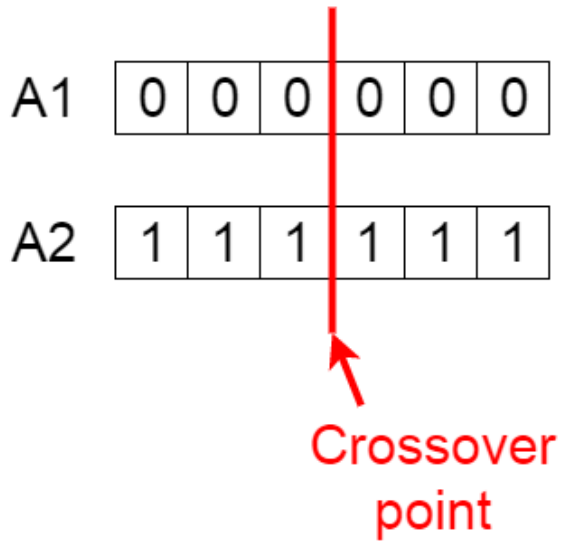
A genetic algorithm is a population-based optimisation algorithm. It maintains a set of possible solutions (the population) which are then optimised by the genetic algorithms

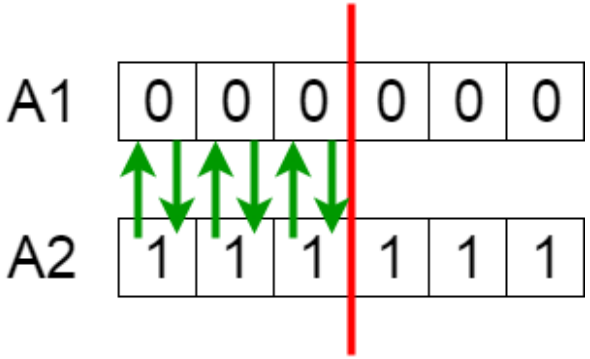
1. Initial random population
2. Calculate the fitness (=score) of each member of the population
3. **Select** a fraction of the fittest members as parents, discard the rest
4. Combine the PSs of two parents into a new offspring PS using **crossover**
5. Make a random move on some offspring (**mutation**)
6. Repeat at (2) until the fitness stops improving

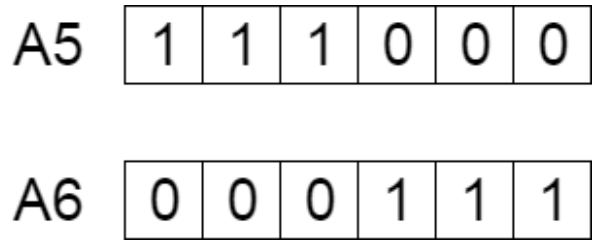


Genetic Algorithm (2)

Crossover combines part of the PS from each parent to create a new PS. This helps to explore the solutions between the current population. Care is needed to make sure the combination is still a PS.

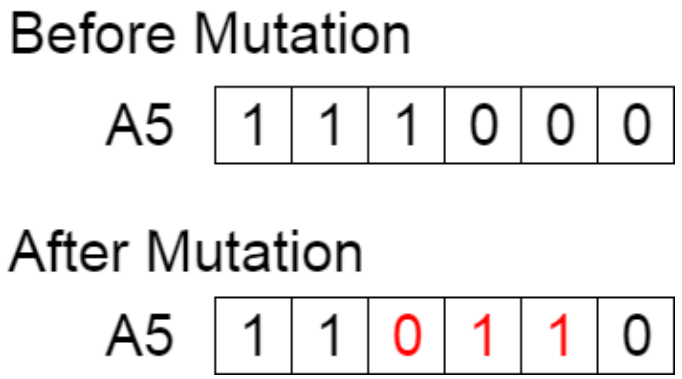






Genetic Algorithm (3)

Mutation is the equivalent of the score-decreasing step in simulated annealing. It allows the generation of new less fit solutions which may escape from a local maximum.



Since there are multiple possible solutions in the population, it can search a large part of the solution space at the same time. Like SA, it is computationally expensive.

Summary

Understand:

- Optimisation for problem solving
- The gradient ascent heuristic
- The problem of local optima
- Simulated Annealing
- Genetic Algorithm

Read

- Skiena, Sections 12.6, 12.9