Algorithms and their Applications CS2004 (2020-2021)

Dr Mahir Arzoky

16.1 The Travelling Salesperson Problem

Labs and CW assessments

- All class tests (Task #1) should be submitted by **16/02/2021 at 11:00AM**
- Task #2 assessment brief will be released towards the end of the week
- Check module breakdown on Blackboard!

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
  - Heuristic Search
  - Hill Climbing and Simulated Annealing
  - Parameter Optimisation (Applications)
  - Evolutionary Computation
  - Swarm Intelligence

Introduction

- In this lecture we are going to look in detail at a specific problem
    - **The Travelling Salesperson Problem**

The Travelling Salesperson Problem

- The Travelling Salesperson Problem (TSP) is an extremely well studied problem
  - Within optimisation (circa 1930s)
  - Within Heuristic Search
  - It is very simple to define
- Solving the TSP will allow us to solve a number of real world problem
  - I.e. You show that a problem can be decomposed into the TSP…

Solving the TSP

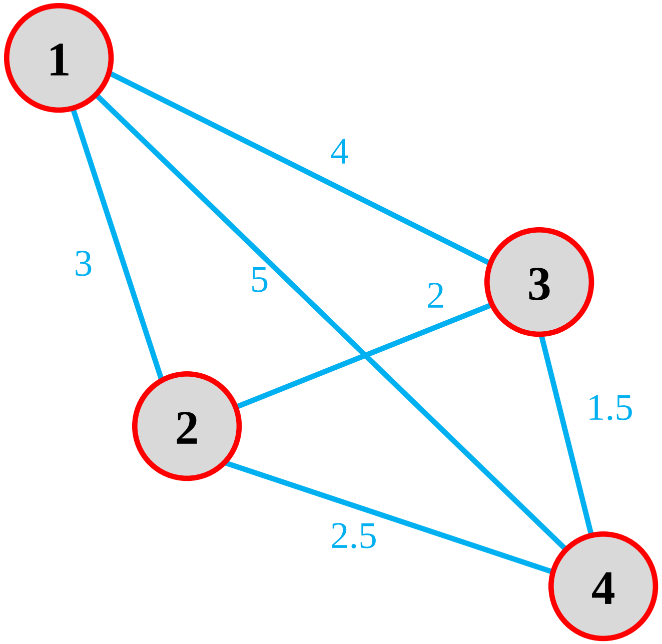
- The Travelling Salesperson Problem is well known to be **NP-Hard**
- This means there is no direct algorithmic or mathematical way to derive a solution in polynomial time
  - The search space (as we will see later) is $O(n!)$
  - One of the worst that we have seen to date…
- We need approximate or Heuristic search methods to find a solution (or partial solution)
  - Hill Climbing
  - Simulated Annealing
  - Etc...

TSP Definition – Part 1

- A sales person has to visit n cities as part of their job
- The aim is to start off at one of the cities, visit each city exactly once and then to arrive back at the starting city
  - This is called a **tour**
- The objective is to find a tour where the sum of the total distance travelled is a minimum
  - I.e. This is analogous to the sales person trying to minimise their petrol costs (or driving time…)

TSP Definition – Part 2

- In order to solve the TSP we need to know how "far" each city is from each other
- We will use the notation $d(i,j)$ to represent the distance from city i to city j
- Note:
  - $d(i,j)=d(j,i)$, i.e. the distance from city i to city j is the same as from city j to city i
  - $d(i,i) = 0$, i.e.: a city is zero distance from itself



$d(1,2)=3$

$d(1,3)=4$

$d(1,4)=5$

$d(2,3)=2$

$d(2,4)=2.5$

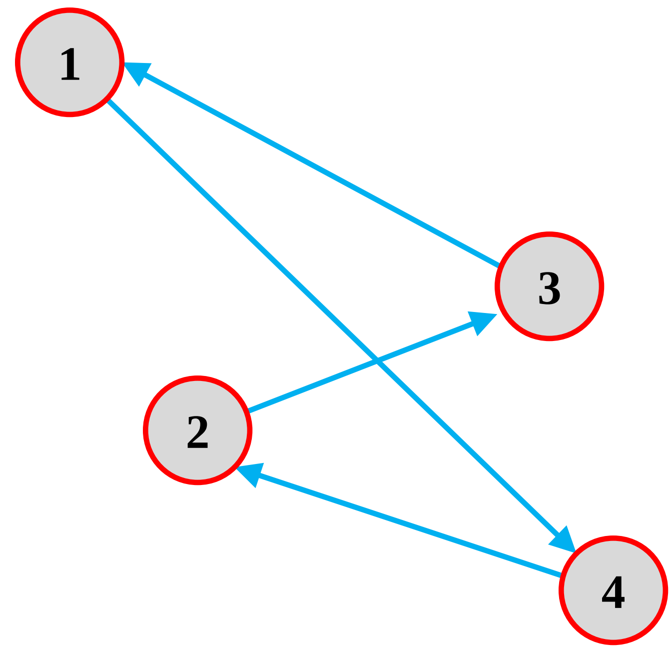
$d(3,4)=1.5$

TSP Representation – Part 1

- A natural way to represent a solution to the TSP problem is to represent a **tour** as a **permutation** of the integers $1,2,\ldots,n$
  - Starting at the left-hand side of the permutation and visiting the cities in the specified order, moving from left to right
  - This will ensure that we do not visit any city twice
- A **permutation** is defined as a shuffling of a set of objects
- If we have n objects then there are $n!$ possible permutations $4! = 24$, $10! = 3{,}628{,}800$, $100! \approx 9.333\times10^{157}$
  - Number of atoms in the known Universe $\approx 10^{80}$
  - Number of seconds the Universe it thought to have existed for $\approx 4.3\times10^{17}$
  - If every atom was a computer that could evaluate a trillion (10^{12}) tours per second then a size 100 problem would take $\approx 10^{48}$ universes…

TSP Representation – Part 2

- In our previous example a possible tour might be represented as 1,4,2,3
- This would mean that we start at city 1, visit cities 4, then 2, then 3 and then **back** to city 1



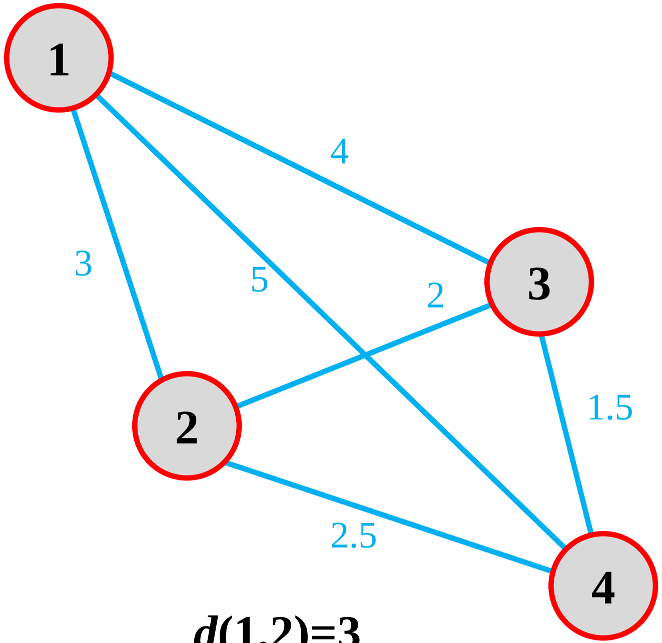
Scoring a TSP Tour – Part 1

- If we are dealing with n cities then we will define a tour, T, as a permutation of the integers $1,\ldots,n$
  - We will define t_i as the ith value of T
  - I.e. the ith city we visit
  - Therefore t_1 is the start and end city
- We will define $f(T)$ as the total cost/distanced travelled carrying out that tour

Scoring a TSP Tour – Part 2

- $f(T)$ is defined as follows:

$$f(T) = \left(\sum_{i=1}^{n-1} d(t_i, t_{i+1}) \right) + d(t_n, t_1)$$



Thus if T={1,4,2,3} then

$f(T)=d(t_1,t_2)+d(t_2,t_3)+d(t_3,t_4)+d(t_4,t_1)$

$f(T)= d(1,4)+d(4,2)+d(2,3)+d(3,1)$

$f(T)=5+2.5+2+4=13.5$

If T={1,2,3,4} then

$f(T)= d(1,2)+d(2,3)+d(3,4)+d(4,1)$

$f(T)=3+2+1.5+5=11.5$

$d(1,2)=3$
$d(1,3)=4$
$d(1,4)=5$
$d(2,3)=2$
$d(2,4)=2.5$
$d(3,4)=1.5$

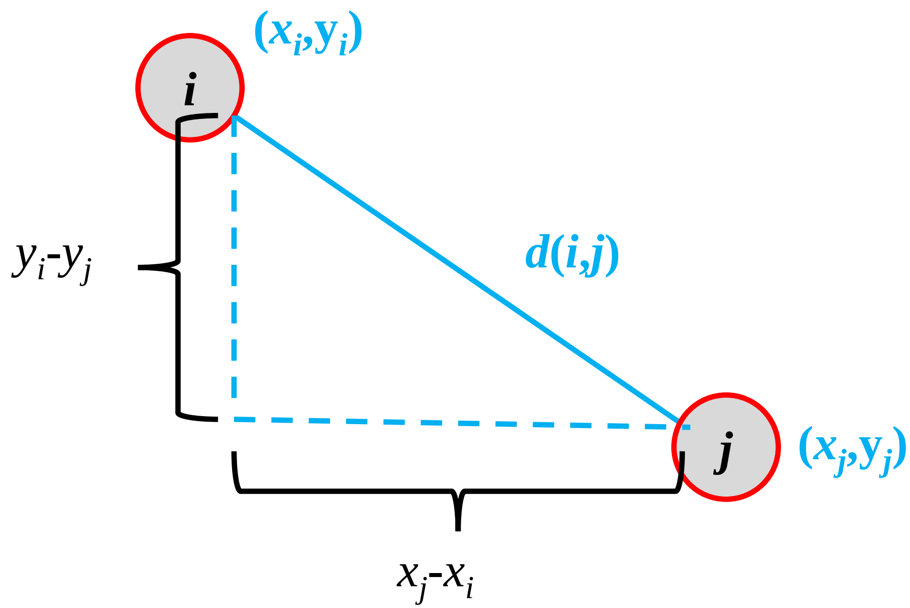
Solving the TSP

- We will now look at solving the TSP using a **Hill Climbing** algorithm
- We will need the following:
  - The number of cities and the distances between each pair
  - A representation and random starting point
  - A fitness function
  - A small change operator

Cities and Distances – Part 1

- If we are provided with the number of cities (n) and a matrix containing all of the $d(i,j)$ then we can move onto the next step
- However we might just be given the coordinates or map location of each city
- If this is the case then we need to compute each pair of distances
- We will assume that (x_i,y_i) is the location of city i on a map

Cities and Distances – Part 2



We can compute each $d(i,j)$ using **Pythagoras's** theorem to get the **Euclidean** distance between city i and j

$$d(i, j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$$

Representation and Starting Point

- Representation will be a vector/array of integers containing a permutation of $\{1,\ldots,n\}$
  - How do we generate a random permutation?
  - There are many ways to do this...

```
Algorithm 1. RandPerm(N)
Input: Number of cities N
1) Let P = list of length N, (|P|=N) where p_i=i
2) Let T = an empty list
3) While |P| > 0
4)     Let i = UI(1,|P|)
5)     Add p_i to the end of T
6)     Delete the ith element (p_i) from P
7) End While
Output: Random tour T
```

Fitness Function

- We will use the scoring function we discussed previously
- This is the length of the tour
  - Distance travelled
- We will try and minimise this value
- More details can be found in the laboratory worksheet

A Small Change

- We must make sure that our small change is valid
  - If we make a change to T it must still be a permutation
- One such operator is the **Swap** operator
- We choose two random element of T, t_i and t_j where $i \neq j$ and then let $t_i = t_j$ and $t_j = t_i$

```
Algorithm 2. Swap(T)
Input: A tour (permutation) of size N
1) Let i=j=0
2) While i=j
3)     Let i = UI(1,|T|)
4)     Let j = UI(1,|T|)
5) End While
6) Let temp = t_i, Let t_i = t_j, Let t_j = temp
Output: Changed tour T
```

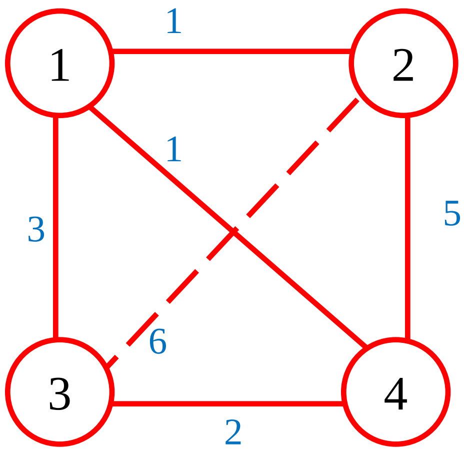
Quality of a TSP Solution

- It would be very useful if we knew whether a TSP solution was any good or not
- With the **Scales** and **OneMax** problems we had an idea of what we were aiming for
  - E.g. A fitness of 0 (Scales) or n (OneMax)
- Does such a limit exist for a given instance of the TSP?
  - There are many lower estimates or lower bounds for the TSP
  - Most of them are very, very complex [mathematically]
  - We will use one that is based upon **Minimum Spanning Trees**
    - Not the most accurate however – but one of the “easiest” to understand

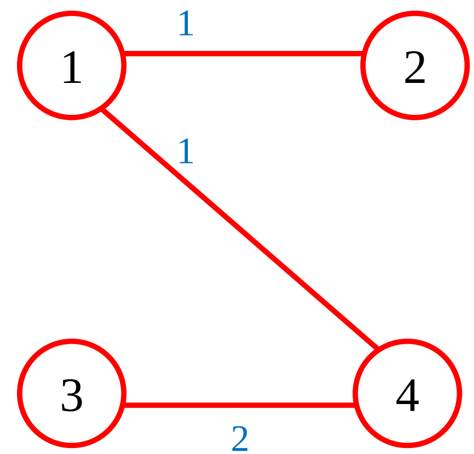
Recap: Minimum Spanning Trees

- A spanning tree is a sub-graph that is also a tree (no cycles) that contains all of the nodes of the super-graph
    - With minimum number of edges
- A graph may have many spanning trees
- The **cost** of a spanning tree is the sum of all of the edge weights
- A **minimum spanning tree** (MST - there may be many) is the spanning tree with the minimum cost

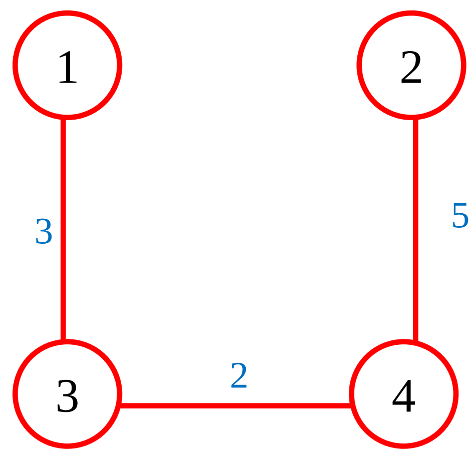
Minimum Spanning Tree Example



Graph of cities



Minimum Spanning Tree, Cost = 4



Spanning Tree, Cost = 10

TSP and MST – Part 1

- We can use the length of the minimum spanning tree as an absolute lower limit on the fitness of a TSP solution
- The MST may/will not be a valid TSP solution
- The minimum value of a TSP solution can never be lower than the length of the corresponding MST

TSP and MST – Part 2

- Let D be a matrix such that $d_{ij}=d(i,j)$
  - I.e. The distance between city i and j
- Let $TSP(D)$ be a solution to the Travelling Salesman Problem applied to the distances in D
- Let $MST(D)$ be the cost (length) of the Minimum Spanning Tree of D
- Then we have: $\boldsymbol{f(TSP(D)) \geq MST(D)}$
- Hence we can define the efficiency as:

$$\frac{MST(D)}{f\left(TSP(D)\right)} \times 100\% \leq 100\%$$

TSP Applications

- Vehicle routing
  - Snow ploughs
  - Parcel delivery
  - Etc..
- Often you can mathematically transform a problem into the travelling salesman problem
  - Computer wiring
  - Manufacturing cell layout
  - Scheduling (e.g. orders)
  - DNA sequencing

Next Lecture

- Next week is ASK week – there is no lecture
- We will look at Bin Packing and Data Clustering for our next lecture

Next Laboratory

- ❑ The laboratory will involve running and comparing algorithms for solving the TSP
  - ❑ This worksheet is a little more complex than the others and thus might take a bit more time
  - ❑ Make sure you come to the laboratories!