Algorithms and Heuristics
(Assignment guidance)

Dr Kamal Bentahar

School of Computing, Electronics and Mathematics
Coventry University

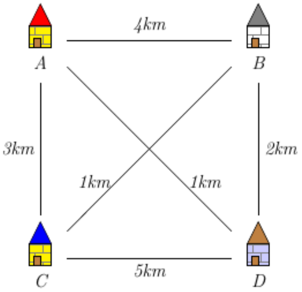
TSP
Examples
Optimization problems
Strategies
Exact methods
Exhaustive search
DP
Time-Space trade-off
Approximation methods
Greedy search
Meta-heuristics
Metaheuristics
II
GRASP
SA
Tabu
GA
ACO

Travelling Salesman Problem

- One of the most famous problems in CS.
- Given a **list of cities** and the **distances between each pair of cities**, what is the shortest possible route that visits each city and returns to the origin city?
- **NP-hard** problem!

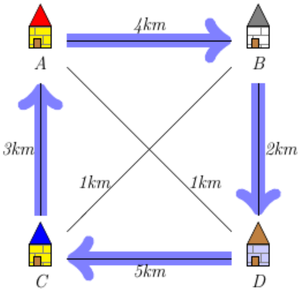
  (This will be explained in a later lectures.)

Travelling Salesman Problem



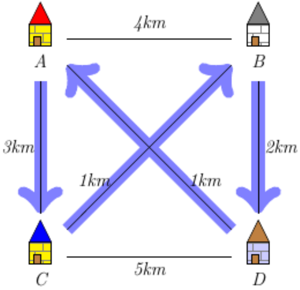
Shortest tour?

Travelling Salesman Problem



$$4 + 2 + 5 + 3 = 14$$

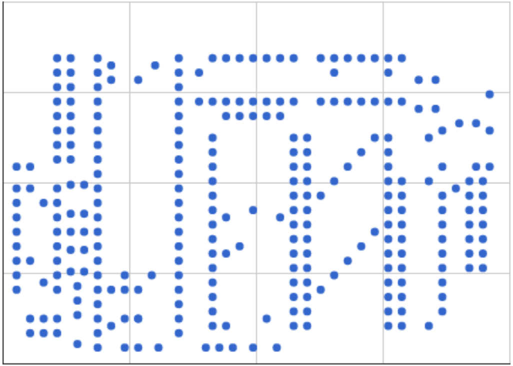
Travelling Salesman Problem



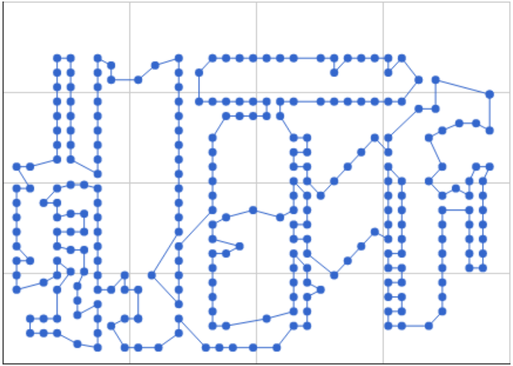
$$3 + 1 + 2 + 1 = 7$$

TSP
Examples
Optimization problems
Strategies
Exact methods
Exhaustive search
DP
Time-Space trade-off
Approximation methods
Greedy search
Meta-heuristics
Metaheuristics
II
GRASP
SA
Tabu
GA
ACO

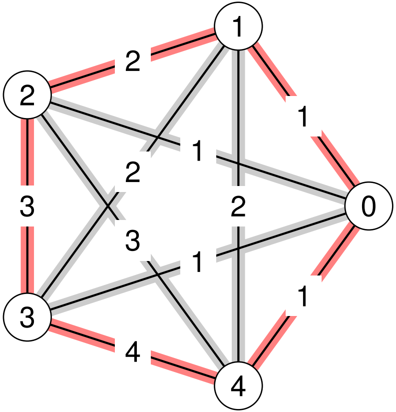
Travelling Salesman Problem



Travelling Salesman Problem



Concepts



Let G be a complete weighted graph with n vertices...

- **Complete**: the graph is undirected, has no self-loops, and each node is connected to all the other vertices.
- **Weighted**: the edges have a weight (a positive integer).
- **Cycle**: a path that visits every vertex once, and goes back to the start point.
- **Total cost of the cycle**: sum of the edge weights of the cycle.

TSP
Examples
Optimization problems
Strategies
Exact methods
Exhaustive search
DP
Time-Space trade-off
Approximation methods
Greedy search
Meta-heuristics
Metaheuristics
II
GRASP
SA
Tabu
GA
ACO

Formal definition of the problem

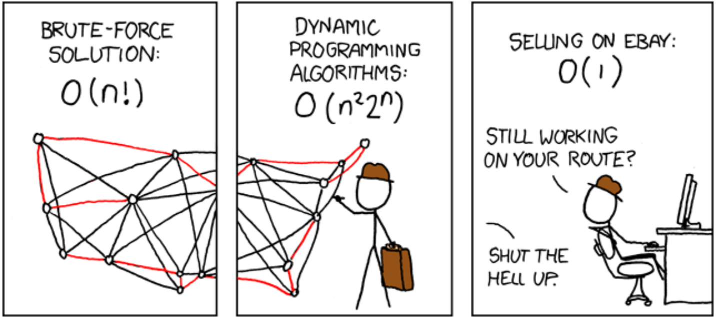
Given G as above, the versions of the TSP are defined as follows:

- **Decisional TSP (D-TSP)**: Given a total cost k, decide if G is has a cycle of length $\leq k$.
- **Search TSP**: Given a total cost k, search for a cycle of length $\leq k$ in G. (If found then return it, otherwise say that there is no such cycle.)
- **Optimization TSP**: Given G, find a cycle of minimal total cost.

Travelling Salesman Problem – what is the issue?

Number of cities n	Number of paths $(n-1)!/2$
3	1
4	3
5	12
6	60
7	360
8	$2,520$
9	$20,160$
10	$181,440$
15	$43,589,145,600$
20	6.082×10^{16}
71	5.989×10^{99}

Travelling Salesman Problem – what is the issue?



Optimization problems

A decision problem has a *True* or *False* answer, whereas an “optimization problem” involves finding an **extremum** of a function of several parameters.

Optimization Problems

Maximize or **minimize** a given function (over its *domain* of definition).

Useful strategies for tackling **NP-hard** problems

1. Tractable special cases which can be solved quickly.
2. Exact methods
   - Exhaustive search.
   - Possibly better exponential time algorithms, e.g. Dynamic Programming.
3. Approximation methods – fast, but not always correct.
   - Greedy search
   - Meta-heuristics

Exact Methods: Exhaustive search

- General problem-solving method
- Always finds solution if it exists
- Usually expensive - tends to grow exponentially or worse

Exhaustive search

```
1: for for all possible candidates do
2:     if candidate satisfies the problem's conditions then
3:         return candidate
4:     end if
5: end for
6: return no solution
```


TSP
Examples
Optimization problems
Strategies
Exact methods
Exhaustive search
DP
Time-Space trade-off
Approximation methods
Greedy search
Meta-heuristics
Metaheuristics
II
GRASP
SA
Tabu
GA
ACO

Exact Methods: Dynamic Programming

- Build solution by first solving smaller problem instances
- Suitable when the problem has:
  - overlapping sub-problems
  - and optimal sub-structure making global optima a function of local optima.

Dynamic Programming

1: Charachterize structure of optimal solution.
2: Recursively define value of optimal solution.
3: Compute in a bottom-up manner - store intermediate results in a table.

Time-Space trade off

Dynamic Programming vs Exhaustive Search

- Exhaustive search tends to require less space but more time.
- Dynamic programming: space complexity can be big (table size).

Heuristic/Meta-heuristic methods

- Give up on exactness, but hope for near optimal solution, in “reasonable” time.
- May be the only feasible way to obtain near optimal solutions at relatively low computational cost.
- Two main approaches:
  1. **Construction methods** work on partial solutions, trying to extend them in the best possible way to complete problem solutions.
  2. **Local search methods** move in the search space of complete solutions.

TSP
Examples
Optimization problems
Strategies
Exact methods
Exhaustive search
DP
Time-Space trade-off
Approximation methods
Greedy search
Meta-heuristics
Metaheuristics
II
GRASP
SA
Tabu
GA
ACO

Heuristic/Meta-heuristic methods

- Give up on exactness, but hope for near optimal solution, in “reasonable” time.
- May be the only feasible way to obtain near optimal solutions at relatively low computational cost.
- Two main approaches:
  1. **Construction methods** work on partial solutions, trying to extend them in the best possible way to complete problem solutions.
  2. **Local search methods** move in the search space of complete solutions.

When is it best to use (meta-)heuristics to solve optimization problems?
When the problem is NP-Hard, otherwise solve exactly.

Approximation methods: Greedy search

Build solution to a problem in an incremental way, starting with an empty initial solution and iteratively adding appropriate solution components (without backtracking) until a complete solution is built.

Algorithmic skeleton of the greedy construction heuristic

$s \leftarrow$ empty solution
while s is not a complete solution **do**
    $e \leftarrow$ solution component with the best heuristic estimate
    updates s by adding the component e
end while
return s

At each iteration, a component that maximizes the immediate gain is selected. (Decisions best in the short term without considering long term consequences)

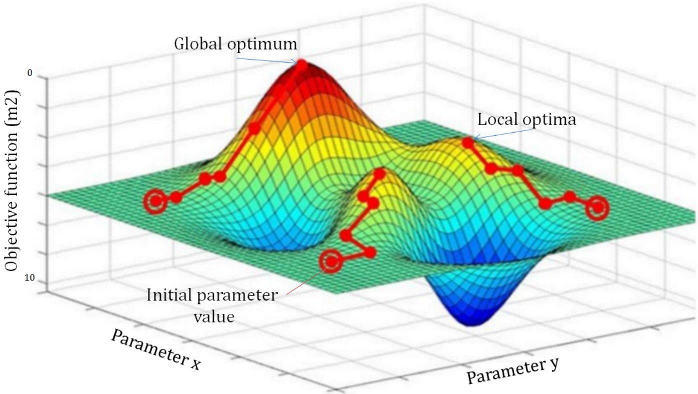
It can be quite efficient.

Meta-heuristics

1. Multi-starts
2. GRASP
3. Tabu Search
4. Iterative improvement (Local search)
5. Simulated annealing (Probabilities for worsening moves)
6. Tabu search (Adaptive memory)
7. Genetic Algorithms
8. Ant Colony Optimization
9. ...

Local Search - Neighbourhoods and Optima

Each solution candidate has a **neighbourhood** of solutions which can be reached by making small changes.



Local search may get stuck in a local optimum.

Local Search – Strategies

- **Best fit**: search the whole neighbourhood and then move to the best neighbour solution.
- **First fit**: search neighbourhood; move to the first improving solution found.
- **Random first fit**: pick random solutions from the neighbourhood; move to the first one found.
- **Candidate list strategies**: reduce the number of possible choices at each step: only search a subset of the neighbourhood solutions.
- **Multi starts**: restart every time the algorithm gets stuck (random changes, ruling out previous choices).

0) Iterative Improvement

- Search a “neighbourhood” of a solution for an improvement.
- Move to improved solution and search its neighbourhood.
- Keep going until you find no more improvements.

Can use with initial solution from greedy or randomly generated.

Try to minimize objective function f using local search

```
determine an initial candidate solution s.        ▷ e.g. through greedy search
while s is not a local optimum do
    choose a neighbour s′ of s such that f(s′) < f(s)
    s ← s′
end while
return s
```


TSP
Examples
Optimization problems
Strategies
Exact methods
Exhaustive search
DP
Time-Space trade-off
Approximation methods
Greedy search
Meta-heuristics
Metaheuristics
II
GRASP
SA
Tabu
GA
ACO

1) Greedy Randomized Adaptive Search Procedure (GRASP)

$s \leftarrow$ empty solution
while termination criterion is not satisfied **do**
    generate candidate solution s' using a randomized greedy search
    perform a local search on s'
    if s' is better than s then $s \leftarrow s'$
end while
return s

2) Simulated Annealing

Effective approach modelled on the cooling of molten materials.
We have a variable T called temperature, which decreases, simulating cooling.
Probabilities are based on the Boltzmann distribution.

determine initial candidate solution s
set initial temperature T according to annealing schedule
while termination condition not satisfied **do**
    probabilistically choose a neighbour s' of s
    if s' satisfies probabilistic acceptance criterion **then**
        $s \leftarrow s'$
    end if
    update T according to annealing schedule
end while
return s

3) Tabu search

An alternative to the randomized approach is the memory-based approach

- Solutions consist of many components
- After removing a component from a solution, we mark it as tabu (forbidden) for some number of iterations
- The number of iterations is called the tabu tenure
- The neighbourhood is then restricted to use non-tabu components

```
determine initial candidate solution s
while termination condition not satisfied do
    determine set N of non-tabu neighbours of s
    choose a best improving solution s′ in N
    update tabu attributes based on s′
end while
return s
```

4) Genetic Algorithms

So far we have looked at **trajectory approaches**, where we keep only one current solution and make progressive modifications to it.
Population based approaches use more than one solution at a time and make progressive changes to that population:

- Genetic/evolutionary algorithms
- Swarm intelligence (Ant Colony Optimisation, etc.)

```
determine initial population p
while termination criterion not satisfied do
    generate set p_r of new candidates by recombination
    generate set p_m of new candidates from p and p_r by mutation
    select new population p from candidates in p, p_r, p_m
end while
```

5) Ant Colony Optimisation

```
Set parameters and initialize pheromone trails.
while termination criterion not satisfied do
    Construct Ant Solutions
    Apply Local Search                                  ▷ Optional
    Update Pheromones
end while
```