Optimization Problems and Travelling Salesman Problem

Shan He

School for Computational Science
University of Birmingham

Module 06-27818 and 27819: Advanced Aspects of Nature-Inspired Search and Optimisation (Ext)

Outline of Topics

Motivating example

How to become rich and healthy?

A pathetic example

- Save money: minimise the cost of buying food
- Being alive: For each nutrient, e.g., Vitamin C, at least meet the minimum required level

Food	**Calcium**	**Vitamin C**	**Calories**	Price (GBP/100g)
Broccoli	47	89.2	53	0.381
Milk	276	0	120	0.1
Oranges	40	53.2	87	0.271

Exercise 1: Let's sovle it!

- Let's solve it by LibraOffice Calc Solver
- Download the zip file from Canvas
- Open DietProblem.ods
- Click Menu $\rightarrow$ Tools $\rightarrow$ Solver

Diet problem

- Diet problem:
  - Given: a set of available foods with cost and nutrition information
  - Sought: selected foods with minimum cost but also satisfy daily nutritional requirement
- Studied in the 1930s and 1940s.
- Motivated by USA Army's desire to minimize the cost of feeding soldiers while still providing a healthy diet
- Nobel Laureate George Stigler further formulated this as Stigler Diet problem
- Essentially an optimisation problem (linear programming problem).

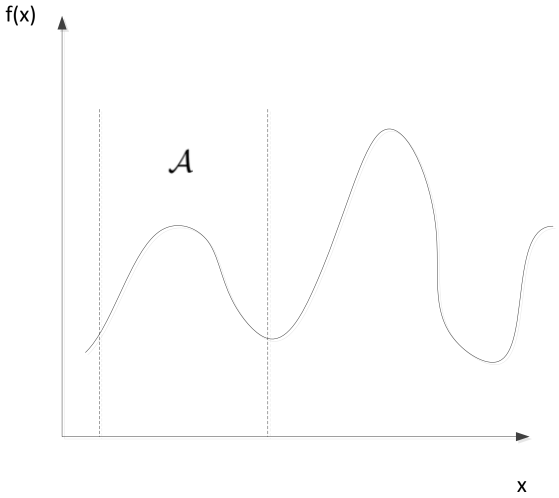
What is optimisation?

- Optimization: to find the best or optimal solution to a problem
- Examples are everywhere:
    - Diet problem
    - Portfolio optimisation: you have an investment portfolio, e.g., cash and shares, you know the potential return and financial risk of each asset. How to build the perfect investment portfolio to maximise the return while keep risk in control?
    - Engineering optimisation: you are required to design a product with some materials. How to design a product with minimal materials while keeping the quality?

Definition of optimisation

- Definition:
  - Given: a function $f(x) : \mathcal{A} \to \mathbb{R}$ from some set $\mathcal{A}$ to the real numbers
  - Sought: an element x^* in $\mathcal{A}$ such that $f(x^*) \leq f(x)$ ("minimisation") or $f(x^*) \geq f(x)$ ("maximisation") for all x in $\mathcal{A}$.
- Function $f(x)$ is called objective function, or cost function (minimisation), fitness function (maximisation in Evolutionary Computation)
- $\mathcal{A}$ is called feasible set, which is some subset of the Euclidean space specified by a set of constraints
- The domain $\mathcal{A}$ of $f(x)$ is called the search space, while the elements of $\mathcal{A}$, e.g., $x \in \mathcal{A}$ are called candidate solutions or feasible solutions.

Definition of optimisation



Categories of optimisation problems

- Depends on the nature of objective function:
  - Linear vs non-linear
  - Linear function:
    - Additivity: $f(x+y) = f(x) + f(y)$
    - Homogeneity: $f(\alpha x) = \alpha f(x)$ for all α
  - If it is non-linear:
    - Convex vs non-convex
  - Multi-objective vs single objective
  - Constrained vs non-constrained
- Depends on the nature of solutions:
  - Continuous vs Discrete (also known as a combinatorial optimization)

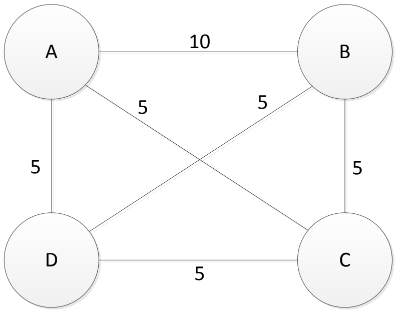
Algorithms for optimisation problems

- Mathematical programming algorithms, e.g., linear programming
- Search and enumeration algorithms
  - Brute force algorithms, enumerating all possible candidate solutions and check
  - Improved brute force algorithms, e.g., branch and bound algorithms
  - **Heuristic algorithms**
    - **Randomised algorithms** ✓
    - **Local search, e.g., greedy search**
    - **Stochastic local search algorithms**, e.g., genetic algorithms

Travelling salesman problem (TSP)

- Travelling salesman problem (TSP):
  - Given: a list of cities and the distances between each pair of them,
  - Sought: the shortest route that visits each city exactly once and returns to the origin city
- Mentioned in a travelling salesman handbook in 1832 and formally defined by the Irish mathematician William. R. Hamilton
- Many real-world problems can be formulated TSP problem: PCBs (Printed Circuit Boards) design, vehicle routing, even X-Ray crystallography (See this book)
- Conceptually simple but computationally difficult: best benchmark for development and evaluation of combinatorial optimisation algorithms

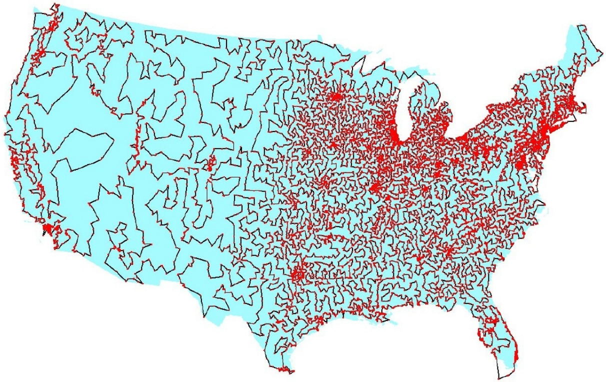
TSP example: 4 cities TSP



Optimal solution?

TSP example

Optimal solution for “usa13509” from TSPLIB. Cities with population at least 500 in the continental US (in 1995). In total 13509 cities.



Why heuristic algorithms for TSP?

- Difficult for other algorithms: shown to be NP-hard
  - Brute force algorithms: complexity is $O(n!)$, the factorial of the number of cities.
  - Improved brute force algorithms, e.g., branch and cut algorithms: $O(1.9999^n)$ [1]
    - Still time consuming, e.g., for usa13509, the running time required is $O(2^{13509})$
    - In fact, usa13509 was solved in 1998 by Rice University using two clusters of 44 CPUs, took approximately three months from start to finish (see News)
    - The largest instance of TSP problem is an instance in TSPLIB of 85,900 cities, taking over 136 CPU-years [2]!

[1] Woeginger, G.J. (2003), "Exact Algorithms for NP-Hard Problems: A Survey", Combinatorial Optimization Eureka, You Shrink! Lecture notes in computer science, vol. 2570, Springer, pp. 185207. [2] Applegate, D. L.; Bixby, R. M.; Chvtal, V.; Cook, W. J. (2006), The Traveling Salesman Problem, ISBN 0-691-12993-2.

Code example: Generating solutions for TSP

- Download the source code from Canvas
- `load('cities.mat')`
- Open the matrix ‘cities’, which is a 2×48 cities TSP problem
    - 48 USA state capital cities
    - The minimal tour has length 10628.
    - Check this page
- Open the optimal solution ‘att48_s.txt’ file and copy
- `optimalsolution =[` paste the solution `]`
- `inputcities = cities(:,optimalsolution)`
- `distance(inputcities)`
- Read help file about how to generate random permutation: `help randperm`

Exercise: Randomised search algoirthms for TSP

- Open ‘randomsearch_skeleton.m’ file
- Complete the code

Conclusion

- Optimisation problems can be very difficult
- Some of them are NP-hard: exact optimal solution of NP-problems requires (in the worst case) an exhaustive search.
- Optimisation is all about exploration and exploitation
- Randomness can facilitate exploration, but not exploitation (local search)
- I will introduce local search and stochastic local search algorithm, e.g., simulated annealing next week