Set cover problem and algorithms (I)

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

Some motivating examples: Facility Location Optimisation

Example modified from Gurobi

- A large supermarket chain wants to expend its business in Northern England in the UK
  - The locations of the new supermarkets have been decided
  - We need to build warehouses to supply these new supermarkets
  - A set of good candidate locations for the warehouses have been determined
  - Each warehouse will supply some supermarkets with reasonable transport cost
  - There will be the different building costs associating with warehouses in different locations
- **Problem**: how to chose warehouses locations to minimise the building cost but cover all new supermarkets?
- This is called Facility Location problem

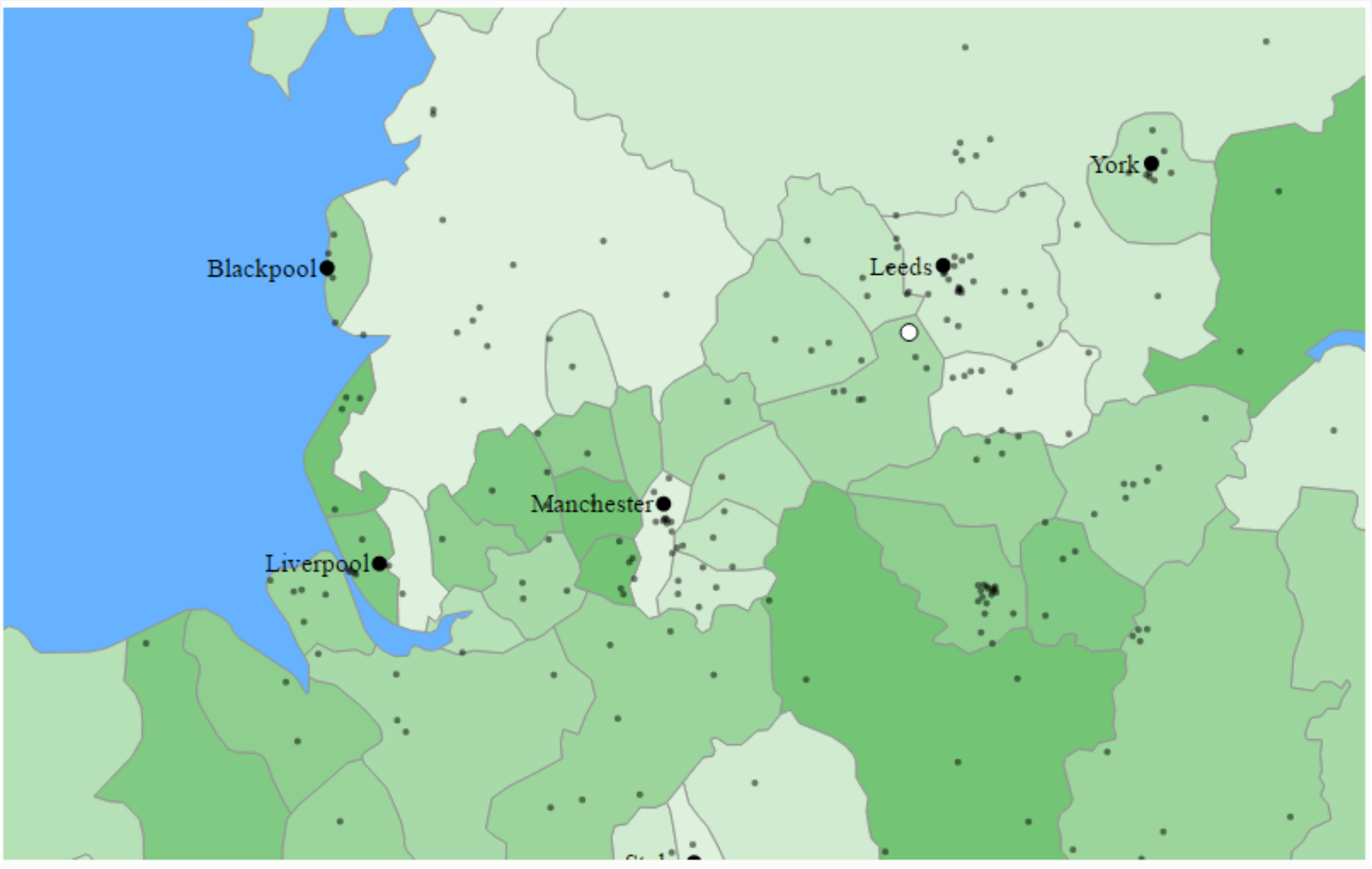

Figure: Picture from GeoLytix

Facility Location Optimisation

- Given
  - a set of m new supermarkets
  - a set of candidate warehouse locations, e.g., $S = \{S_1, S_2, \cdots, S_n\}$, of which each location (S_j) covers a subset of the supermarkets and has an associated cost c_j.
- Find a set of warehouse locations from S with minimum cost but covers all m supermarkets.
- How to solve the problem?

Set cover problem

- Given
  - a universe U of m elements
  - a collection of subsets of U say $S = \{S_1, S_2, \cdots, S_n\}$ where every subset S_j has an associated cost c_j.
- Find a minimum cost subcollection of S that covers all elements of U.

Set cover problem

Why set cover problem is important?

- One of Karp's 21 NP-complete problems
- Many problems can be formulated as set cover problem:
  - Facility location problems
    - The supermarket warehouse planning problem
    - Fire station planning: how to build the minimum number of fire stations and ensure that at least one fire station is near 15 minutes of each town/village
  - Product planning: how to plan a set of products, of each satisfies some requirements of a population of customers, so that the cost of for production is minimise but can cover all requirements of customers?
  - **Scheduling problem**:
    - **Airline, railway, bus crew scheduling problem**
    - Staff scheduling problem

Set cover problem: a simple example

- Given
  - a universe U of m elements u_i, e.g., $U = \{1, 2, 3, 4, 5\}$
  - Note: u_i can be anything, e.g., numbers, supermarkets, flight-legs etc.
  - a collection of subsets of U, e.g., $S = \{S_1, S_2, S_3\}$ with associated cost c_j:
    - $S_1 = \{1, 3, 4\}$, $c_1 = 5$
    - $S_2 = \{2, 5\}$, $c_2 = 10$
    - $S_3 = \{1, 2, 3, 4\}$, $c_3 = 3$
- Find a minimum cost subcollection $\mathcal{F}$ of S that covers all elements of U.
- Only 3 possible solutions to cover U, which can be easily solved.

Set cover algorithm: intuition

- How to design algorithms to solve the problem?

Set cover algorithm: intuition

- The problem: Find a **minimum** cost subcollection of $\mathcal{F}$ that covers **all** elements of U.
- **Intuition**: iteratively select subsets, each time select a subset that covers maximum uncovered elements but with minimum cost

Set cover problem: a simple greedy algorithm

The greedy algorithm for set cover problem

Let X represents set of uncovered elements
Let $\mathcal{F}$ represents the set cover picked by the algorithm
Initialise $X = U$
Initialise $\mathcal{F} = \varnothing$
while $X \neq \varnothing$ **do**
    Calculate cost effectiveness of each subset: $e_j = \frac{c_j}{|S_j \cap X|}$, $j = 1, \cdots, n$
    Find the set S_i in $\{S_1, S_2, ...S_n\}$ with minimum e_i,
    $\mathcal{F} = \mathcal{F} \cup S_i$; $X = X \setminus S_i$
end while

Implementation: Greedy algorithm (Set version)

Aim: we will implement a naive greedy algorithm for set cover problem using set operators in Matlab.

Code example 1: Matlab set operations

- **Preparation (10 mins)**: Please read Set Operations
- **Code example 1 (5 mins)**:

```
U = [1, 2, 3, 4, 5];
S1 = [1, 3, 4];
S2 = [2, 5];
S3 = [1, 2, 3, 4];
intersect(S1,U)
intersect(U, S1)
setdiff(U, S1)
setdiff(S1, U)
union(S1, S2)
union(S1, U)
```

Cell arrays

- In order to store sets (arrays) of different sizes, we can use cell array.
- A more general and power data structure in Matlab
- The same cell array can hold elements of different types:
  - numeric matrices of different sizes;
  - character arrays of different sizes
  - other cells;
  - structs;
  - objects.

Code example 2: Cell arrays operations

- Constructing a cell array
  ```
  S = {S1 S2 S3};
  ```
- **Note**: A n-by-m cell array is made up of $n \times m$, 1-by-1 cell arrays,
- Indexing cell array: Two ways to index into and assign into a cell array:
  - () brackets: access or assign **cells**;
  - **Code example 3:** `S(1,1)`
  - {} brackets: access or assign **the data within those cells**.
  - **Code example 4:** `S{1,1}`

Exercise 1: Greedy set cover algorithm (Try at home)

- **Homework**: Implement a greedy set cover algorithm
  - Based on the template of the greedy algorithm (GreedySetCoveringSO_outline.m), implement the algorithm following the pseudo-code in page 11.
  - Solve: Given a universe U of 13 elements: $U = \{1, 2, \cdots, 13\}$, and a collection of subsets of U, i.e., $S = \{S_1, S_2, S_3, S_4, S_5\}$ with unit cost, i.e., $c_j = 1$ $(j = 1, \cdots, 5)$:
    - $S_1 = \{1, 2\}$
    - $S_2 = \{2, 3, 4, 5\}$
    - $S_3 = \{6, 7, 8, 9, 10, 11, 12, 13\}$
    - $S_4 = \{1, 3, 5, 7, 9, 11, 13\}$
    - $S_5 = \{2, 4, 6, 8, 10, 12, 13\}$

    find a minimum cost subcollection of S that covers all elements of U
  - Work out what is the optimal solution by yourself, then check whether the solution found by your greedy algorithm is optimal? Use debug to understand how the greedy algorithm selects subcollections.

Problems with the greedy set cover algorithm

- **Representation problem**: set representation – set operations are time consuming – Cannot handle large scale problems.
- **Search performance problem**: Cannot find the optimal solution: $\{S_4, S_5\}$

Solutions to the problems

- **Representation problem:** set operations are time consuming – Cannot handle large scale problems.
  - **Solution**: Using the integer programming formulation.
- **Search performance problem:** Greedy algorithm will be trapped by local optimal solutions and cannot find the global optimal solution: $\{S_4, S_5\}$
  - **Solution**: Introduce randomness to make it a stochastic local search algorithm
  - We will implement:
    - Binary Genetic Algorithm (BGA)

Implementation: BGA for set cover problem

Aim: we will implement GA to solve integer programming formulation of set cover problem in Matlab

Set cover problem: Integer programming formulation

- Let $x_j \in \{0, 1\},\ j = 1, \cdots, n$ be the decision variables
  - $x_j = 1$: subset S_j is selected
  - $x_j = 0$: otherwise
- Let $c_j,\ j = 1, \cdots, n$ be the costs of the subsets $S_j,\ j = 1, \cdots, n$
- Let $\mathbf{a}$ which is $m \times n$ matrix to represent S , where $a_{ij} = 1$ indicates element i in U is covered by subset S_j; $a_{ij} = 0$ otherwise.

Set cover problem: integer programming formulation

- **Example**: given a universe U of $m = 5$ elements u_i: $U = \{a, b, c, d, e\}$ and a collection of $n = 3$ subsets of U, i.e., $S = \{S_1, S_2, S_3\}$
  - $S_1 = \{a, c, d\}$, $c_1 = 5$
  - $S_2 = \{b, e\}$, $c_2 = 10$
  - $S_3 = \{a, b, c, d\}$, $c_3 = 3$
- 3 subsets – 3 decision variables: x_1, x_2 and x_3
- We construct a 5×3 matrix $\mathbf{a}$:

u_i	S		
	S_1	S_2	S_3
a	1	0	1
b	0	1	1
c	1	0	1
d	1	0	1
e	0	1	0

In essence, we use the decision variables x_i to select columns

Set cover problem: integer programming formulation

- The integer programming formulation of the set cover problem can be written as:

$$\text{minimise} \sum_{j=1}^{n} c_j x_j \tag{1}$$

$$\text{subject to} \sum_{j=1}^{n} a_{ij} x_j \geq 1, \qquad i = 1, \cdots, m, \tag{2}$$

$$x_j \in \{0, 1\}, \qquad j = 1, \cdots, n, \tag{3}$$

- Explanation:
  - The first constraint ensures the solution covers every element of the universe
  - Essence: Sum up the values of each row to ensure each sum is no less than 1 $\rightarrow$ Ensure each element u_i, $(i = 1, \cdots, m)$ in U is at least covered by one selected column

Code examples 4

- **IP formation of the above example (10 mins)**:

```
a = [1 0 1; 0 1 1; 1 0 1; 1 0 1; 0 1 0]
c=[5 10 3]
x = [0 1 1]
TotalCost = x*c'
ElementCoveredTimes = a*x'
Violations = (ElementCoveredTimes - 1)<0
```

- **Note**: In essence, the decision variables can be seen as the indices of selected columns of $\mathbf{a}$:

```
 ColumnCost = c(x==1)  %The costs of selected column
```

Exercise 3: solving set cover problem using BGA

Exercise 3 (30 mins):

- Write a function (objective function) with:
  - **Inputs**: a solution (a vector of binary decision variable $x_j \in \{0, 1\},\ j = 1, \cdots, n$), costs ($c_j,\ j = 1, \cdots, n$) and $\mathbf{a}$ (a $m \times n$ matrix)
  - **Outputs**: fitness, total cost (equation (1)) and the sum of violation degrees of constraints. Use penalty coefficient $r = 10000$ to calculate fitness.
- Use the problems in Code example 4 to test your function.
- Plug in this objective function into the Binary Genetic Algorithm you have implemented last week to solve the problem in Exercise 1 (page 16).
  - **NOTE**: You need to construct $\mathbf{a}$ and c. Also this time we are dealing with a minimisation problem while last time the project selection problem was a maximisation problem.