Set cover/partitioning problem (II): crew scheduling

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

1. Motivating example: doctor scheduling problem
2. Crew scheduling problem
3. Airline crew scheduling problem

Motivating example: A&E doctor scheduling

- A hospital A&E department needs to keep doctors on call, so that a qualified individual is available to perform every medical procedure that might be required
- For each of several doctors available for on-call duty, the additional salary they need to be paid, and which procedures they can perform, is known.
- The goal to choose doctors so that each procedure is covered, at a minimum cost.
- **Question**: how are you going to solve this problem?

	Doctor 1	Doctor 2	Doctor 3	Doctor 4	Doctor 5	Doctor 6
Procedure 1	✓			✓		
Procedure 2	✓				✓	
Procedure 3		✓	✓			
Procedure 4	✓					✓
Procedure 5		✓	✓			✓
Procedure 6		✓				✓
Salary (k)	180	160	50	30	30	70

Staff/Crew scheduling problem

- Staff/Crew scheduling: assigning a group of workers (a crew) to a set of tasks
    - The staff (crews) are typically interchangeable
    - In some cases different crews possess different characteristics that affect which subsets of tasks they can complete
- **Main goal**: to cover all tasks while seeking to minimize labour costs, and a wide variety of constraints imposed by safety regulations and labour negotiations.
- Crew scheduling problem be formulated as a **set cover problem** or set partitioning problem (explained later)
- Crew scheduling research focuses on a particular application, rather than the general case
- Many applications in transportation, e.g., bus and rail transit, truck and rail freight transport, and freight and passenger air transportation

Airline crew scheduling problem

- Airline crew scheduling problem: why it is important?
  - Airlines typically have a fixed schedule that changes **at most** monthly – A true planning/scheduling problem
  - They provide a context for examining many of the elements common to all crew scheduling problems such as safety regulations
  - Airline industry is highly regulated: more constraints $\longrightarrow$ more difficult to solve
  - Airline crews receive substantially higher salaries than equivalent personnel in other modes of transportation; the savings associated with an improved airline crew schedule can be quite significant

Airline crew scheduling problem: a toy problem

- An airline has $m = 5$ scheduled flight-legs connecting 6 cities per week in its current service
- A **flight-leg** is a single flight flown by a single crew e.g. London - Sheffield
- Each flight leg must be flown.
- For example:

Flight-leg	Origin	Destination
1	Newcastle	Bath
2	Sheffield	Edinburgh
3	Plymouth	London
4	Edinburgh	Newcastle
5	London	Sheffield

Airline crew scheduling problem: a toy problem

- Round-trip rotation (pairing): a sequence of flight legs for a crew that begin and end at individual base locations
- Usually last for 2-5 days, must conform to all applicable regulations, work rules, restrictions and other factors
- Let $S_j (j = 1, \cdots, n)$ be all (in our example, $n = 7$) feasible round-trip rotations, each rotation is associated with a cost c_j

j	Round-trip Rotations S_j	cost c_j
1	Sheffield – Edinburgh – Newcastle – Sheffield	560
2	Sheffield – Edinburgh – London – Sheffield	335
3	Plymouth – London – Plymouth	420
4	Plymouth – London – Sheffield – Plymouth	470
5	Newcastle – Bath – Sheffield – Newcastle	545
6	Sheffield – Newcastle – Bath – London– Sheffield	660
7	Newcastle – Bath – Edinburgh – Newcastle	490

Airline crew scheduling problem: a toy problem

- Let $x_j \in \{0,1\},\ j = 1, \cdots, n$ be the decision variable
  - $x_j = 1$: Rotations S_j is selected
  - $x_j = 0$: otherwise
- The objective: to find the optimal collection of rotations with minimal costs such that each flight leg is covered by exactly one rotation.
- The objective function:

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

- Constraints: **Each flight leg is covered by exactly one rotation.**
- **Question**: Why each flight is cover by **exactly one rotation**?

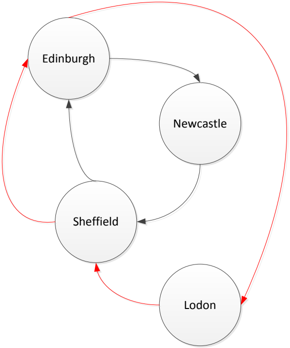
The constraints

- Constraints: **Each flight leg is covered by exactly one rotation.**
- Why exactly one rotation?
    - A flight leg must be covered
    - Flight leg is covered by no more than one rotations
        - One rotation is executed by one crew using one single flight (plane)
        - More than one rotations, e.g., two means one crew executes the flight and the other crew travels on the same flight as passengers – extra costs, e.g., tickets/hotel costs that NOT included in the rotation costs

The constraints

- Example: flight leg 2 (Sheffield to Edinburgh) is covered by rotations 1 and 2:

j	Round-trip Rotations S_j	cost c_j
1	Sheffield – Edinburgh – Newcastle – Sheffield	560
2	Sheffield – Edinburgh – London – Sheffield	335



The constraints

- **Constraints**: **Each flight leg is covered by exactly one rotation.**
- **Representation**: We can construct a $m \times n$ constraint matrix $\mathbf{a}$ to represent the constraints, where $a_{ij} = 1$ indicates flight-leg i is covered by rotation j; $a_{ij} = 0$ otherwise. For example, we construct a 5×7 constraint matrix $\mathbf{a}$

Table: Constraint Matrix $\mathbf{a}$

Flight-leg	Round-trip Rotation S_j						
	1	2	3	4	5	6	7
1 Newcastle to Bath	0	0	0	0	1	1	1
2 Sheffield to Edinburgh	1	1	0	0	0	0	0
3 Plymouth to London	0	0	1	1	0	0	0
4 Edinburgh to Newcastle	1	0	0	0	0	0	1
5 London to Sheffield	0	1	0	1	0	1	0

The constraints

- Constraints: **Each flight leg is covered by exactly one rotation.**
- The airline crew scheduling problem can be written as:

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

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

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

Set Cover vs Set Partitioning

- Key difference between airline crew scheduling problem and Set Cover problem:
    - **Set Cover Problem**: the constraints are inequality constraints:
    $$\sum_{j=1}^{n} a_{ij}x_j \geq 1,\ i = 1, \cdots, m$$
    - airline crew scheduling problem: the constraints are equality constraints:
    $$\sum_{j=1}^{n} a_{ij}x_j = 1,\ i = 1, \cdots, m$$
    - Crew scheduling problem $\in$ **Set Partitioning Problem**
    - But you can cast a set partitioning problem into a set cover problem by converting the equality constraints into inequality constraints:
    $$1 - \sum_{j=1}^{n} a_{ij}x_j \leq \epsilon,\ i, \cdots, m$$
    where ϵ is a small number

How to solve it: Stochastic local search algorithm

- The greedy algorithm you have implemented is a local search algorithm.
  - Good at exploitation: capable to find local optimum
  - Not good at exploration: gets stuck into local optimum
- Stochastic local search algorithm: escape from local optima by introducing randomness
- We will implement an algorithm in Balas and Ho [1980]

E. Balas and A. Ho. Set covering algorithms using cutting plans, heuristics and subgradient optimisation. A computational study. Mathematical Programming Study. 12:37-60, 1980

Set covering problem: stochastic local search algorithm

The stochastic local search for set cover problem

Let I represents which rows have been covered
Let $\mathcal{F}$ represents the solutions, i.e., columns have been selected by the algorithm
Initialise $I = \varnothing$
Initialise $\mathcal{F} = \varnothing$
while $|I| < m$ **do**
    Select a currently uncovered row j uniformly at random
    Select a column i that cover j with minimum cost
    Include column i as part of the solution : $\mathcal{F} \cup i$;
    Set I to include the rows covered by column i
end while

Code Example 1: Stochastic local search for the toy problem

- Run Example1.m to solve the toy problem (page 6)
- Open StochasticSetCover.m and I will explain

Code Example 2: a real-world Boening 727 crew scheduling problem

- Open the B727 air crew scheduling problem (b727.dat), which has 157 rotations and 29 flight-legs
- Use ReadInData.m to read the data into Matlab
- Run Example2.m to solve the B727 air crew scheduling problem (b727.dat). The best known solution is 94400 (see this paper)

Note: There are multiple real-world air crew scheduling problems at this web page here. You can even try some very large-scale ones, the results from the stochastic local search is not too far way from the best known results.

Let's solve the problem using GA

- Binary genetic algorithm:
  - Representation:

column j	1	2	3	4	$\cdots$	$n-1$	n
S_j	0	0	1	1	0	1	0

  - Constraint handling using penalty function:

$$h_i(\mathbf{x}) = 0, \; i = 1, \cdots, m$$

?

  - Note: each row (flight-leg) has a constraint, so in total there are m constraints
  - The new objective (fitness) function:

$$f'(\mathbf{x}) = f(\mathbf{x}) + \lambda \sum_{i=1}^{m} h_i(\mathbf{x})$$

where $h_i(\mathbf{x})$ is the equality penalty function.

Let's solve the problem using GA

- Let's take a look at the equation

$$\sum_{j=1}^{n} a_{ij} x_j = 1,\ i = 1, \cdots, m,$$

- Note: each row (flight-leg) has a constraint, so in total there are m (number of legs) constraints (rows)
- We can write this as a matrix form:

$$\sum_{j=1}^{n} a_{ij} x_j = \mathbf{ax} = 1,\ i = 1, \cdots, m,$$

where $\mathbf{x}$ is a binary vector $\mathbf{x} = [x_1, x_2, \cdots, x_n]^{\mathrm{T}}$

Let's solve the problem using GA

- For each solution $\mathbf{x}$, it must satisfy :

$$\mathbf{ax} - 1 = 0,$$

where $\mathbf{x}$ is a binary vector $\mathbf{x} = [x_1, x_2, \cdots, x_n]^{\mathrm{T}}$

- The constraint violations are::

$$\mathbf{h} = (\mathbf{ax} - 1)^2$$

, where $\mathbf{h}$ is is a positive vector $\mathbf{h} = [h_1, h_2, \cdots, h_m]^{\mathrm{T}}$

- The total degree of violation is:

$$\sum_{i=1}^{m} h_i$$

Exercise: solving crew scheduling problem using BGA

- Use your Binary GA to solve the problem:
  - Complete the variation operators if you have not finished
  - Following my stochastic local search example to write a fitness function
  - Add penalty function: you can use the simplest static penalty function with a penalty coefficient in the range of $[10000, 100000]$