EARL conference

Using Linear Programming for Route Planning and Job Scheduling

5th September 2024
Brighton, UK

vodafone

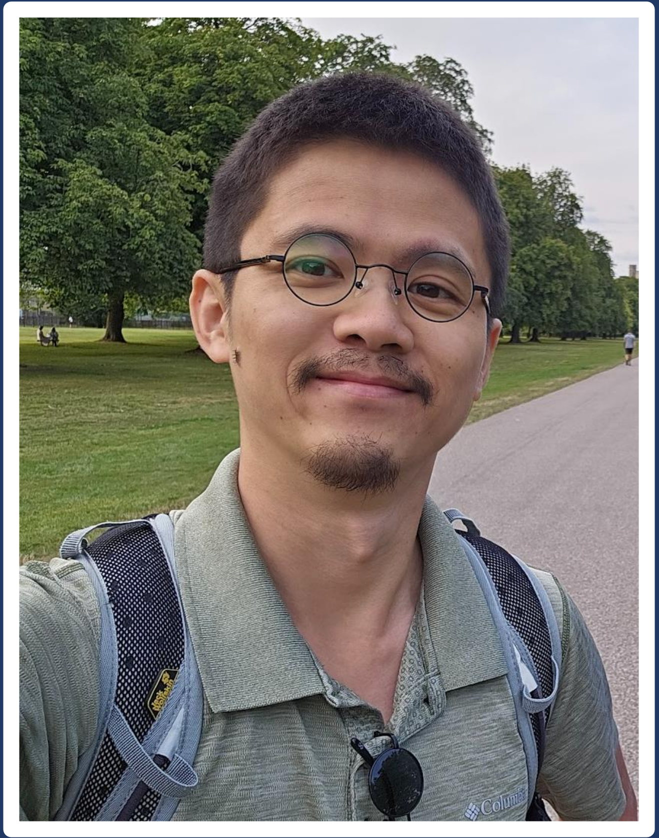
Timothy Wong CStat CEng MBCS

Senior Data Scientist, Vodafone

timothywong731.github.io

Who am I

Timothy (Tim) is a professional Data Scientist with over ten years experience in big data, machine learning and analytics applications. He previously led the Data Science function at a large energy company in the UK. His experience spans across multiple sectors including energy, telecommunications, defence and national security. Tim is professionally qualified as a Chartered Statistician (CStat) as well as a Chartered Engineer (CEng).



I hosted talks at:

- EARL 2016, 2017 and 2019 (London)
- USER 2017 (Brussels), 2018 (Brisbane)
- ERUM 2018 (Budapest)
- … and more ☺

EARL conference

Resource Allocation

- There's a number of jobs requiring fulfilment
- There's a number of resources capable of fulfilling those jobs
- We need to allocate resources to jobs, <u>efficiently</u>!

<u>Legend</u>

- Job
- Resource
- Starting/finishing point

EARL conference

Optimisation Problem(s)

- We aim to fulfil as many jobs as possible
- Resources must start and end at the same location
- Certain jobs may have higher priority over the others
- Minimise travel distance, or time

EARL conference

Knapsack Problem

- Maximise value in the knapsack

Let
$I =$ total number of items
$v_i =$ *value of* i^{th} *item*
$w_i =$ *weight of* i^{th} *item*
$x_i =$ *allocation of* i^{th} *item*

$$Max. \quad z = \sum_{i=1}^{I} v_i x_i$$

$s.t.$

(1) $x_i \in \{0,1\} \quad \forall i = 1,2,3,\ldots,I$

(2) $\sum_{i=1}^{I} w_i x_i \leq 10 \quad \forall i = 1,2,3,\ldots,I$

Knapsack Problem

- Maximise value in the knapsack

Max 10kg

Maximise total value

Item can either be assigned (1) or not (0)

Sum of weight cannot exceed 10 kg

$$
\begin{aligned}
\textit{Let}\quad & I = && \text{total number of items} \\
& v_i = && \textit{value of } i^{th} \textit{item} \\
& w_i = && \textit{weight of } i^{th} \textit{ item} \\
& x_i = && \textit{allocation of } i^{th} \textit{ item}
\end{aligned}
$$

$$Max. \quad z = \sum_{i=1}^{I} v_i x_i$$

$s.t.$

$$(1) \quad x_i \in \{0,1\} \qquad \forall i = 1,2,3,\ldots,I$$

$$(2) \quad \sum_{i=1}^{I} w_i x_i \leq 10 \qquad \forall i = 1,2,3,\ldots,I$$

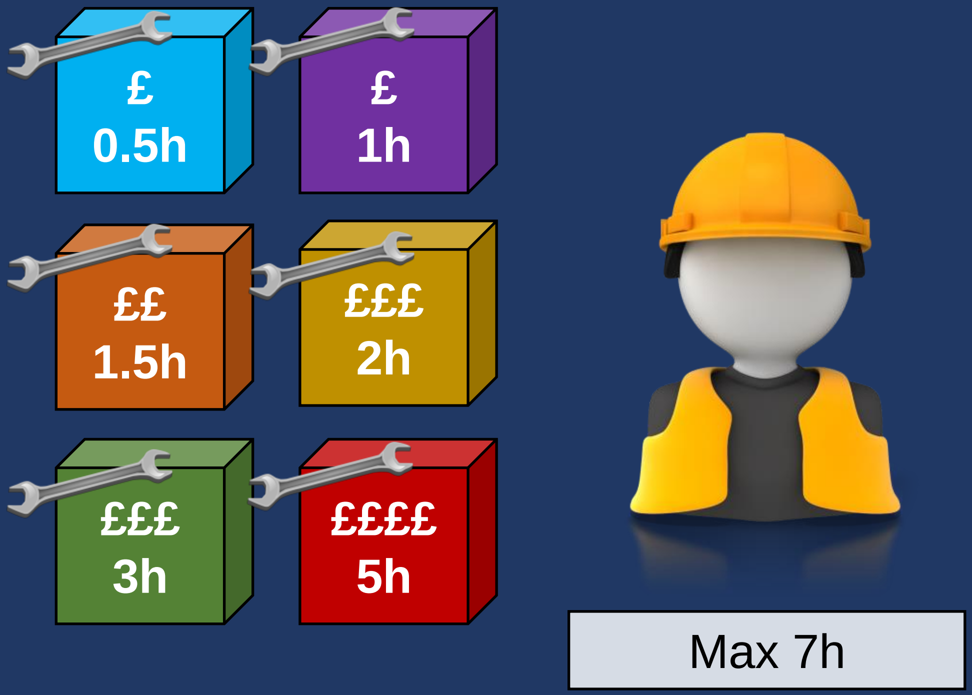
Knapsack Problem

```r
library(ROI.plugin.glpk)
library(ompr)
library(ompr.roi)
library(dplyr)

max_capacity <- 10
v <- c(5, 8, 10, 40, 60, 70)
w <- c(2, 3, 4, 5, 6, 8)
N <- length(v)

result <- MIPModel() |>
  add_variable(x[i], i = 1:N, type = "binary") |>
  set_objective(sum_over(v[i] * x[i], i = 1:N), "max") |>
  add_constraint(sum_over(w[i] * x[i], i = 1:N) <= max_capacity) |>
  solve_model(with_ROI(solver = "glpk"))

solution <- result |>
  get_solution(x[i])

x <- solution |>
  filter(value > 0) |>
  pull(i)

paste0("Items selected: ", paste0(x, collapse = ", "))
paste0("Total value: £", sum(v[x]))
paste0("Total weight: ", sum(w[x]), "kg")
```

ARL conference

Knapsack Problem

- Adapt this into our business context…



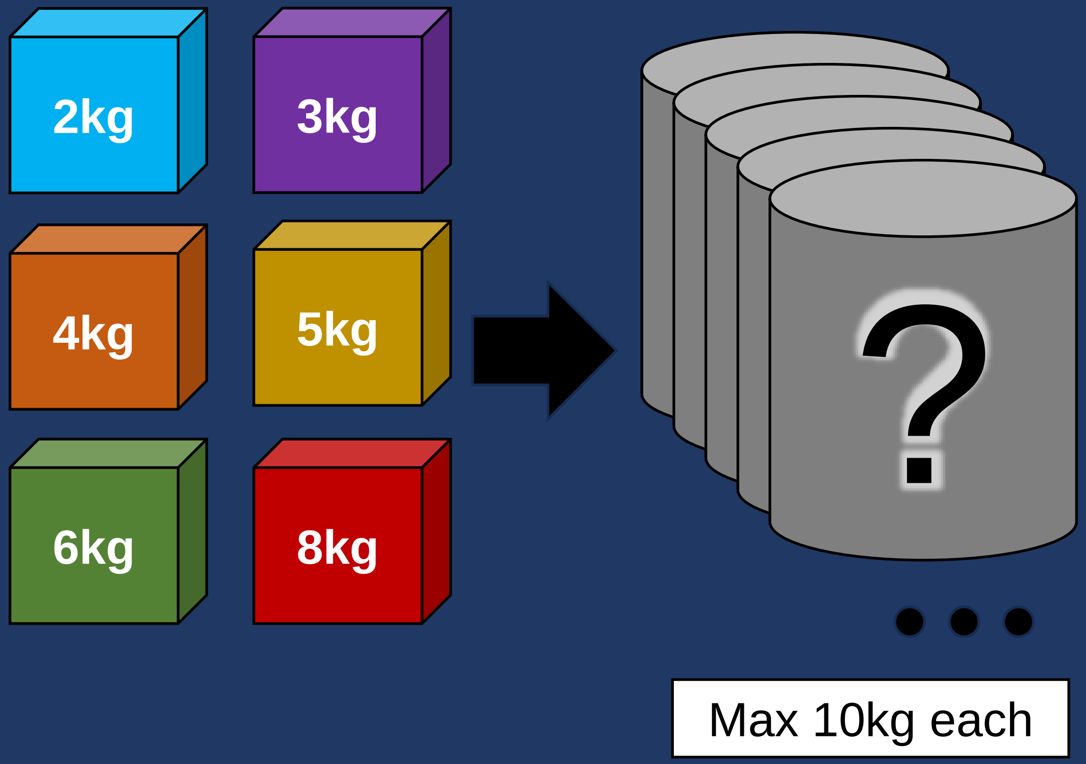
- Can allocate jobs to resource
- Can maximise efficiency

- Works only if there's only one resource
- Doesn't figure out the order of the jobs
- Doesn't address the starting / finishing point

EARL conference

Bin Packing Problem

- Pack items into least number of bins



Let

$I =$ total number of items

$J =$ maximum number of bins

$w_i =$ weight of i^{th} item

$x_{ij} =$ allocation of i^{th} item

$y_j =$ allocation of the j^{th} bin

Min.

$$z = \sum_{j=1}^{J} y_j$$

s.t.

(1) $x_{ij} \in \{0,1\} \quad \forall i = 1,2,3,\ldots,I \quad j = 1,2,3,\ldots,J$

(2) $y_j \in \{0,1\} \quad \forall j = 1,2,3,\ldots,J$

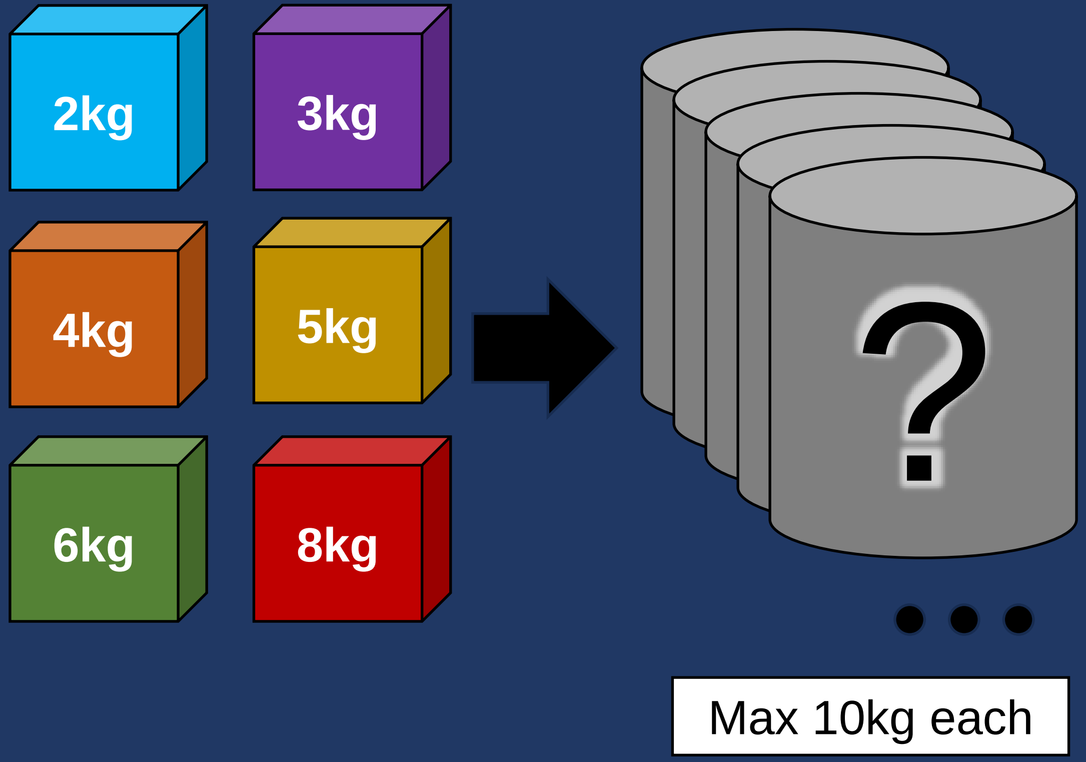
(3) $\sum_{i=1}^{I} w_i x_{ij} \le 10 y_j \quad \forall j = 1,2,3,\ldots,J$

(4) $\sum_{j=1}^{J} x_{ij} = 1 \quad \forall i = 1,2,3,\ldots,I$

EARL conference

Bin Packing Problem

- Pack items into least number of bins



Let

$$I = \text{total number of items}$$
$$J = \textit{maximum number of bins}$$
$$w_i = \textit{weight of } i^{th} \textit{ item}$$
$$x_{ij} = \textit{allocation of } i^{th} \textit{ item}$$
$$y_j = \textit{allocation of the } j^{th} \textit{ bin}$$

Minimise the number of bins used

$$Min. \quad z = \sum_{j=1}^{J} y_j$$

s.t.

Item can either be assigned (1) or not (0)

$$(1) \quad x_{ij} \in \{0,1\} \qquad \forall i = 1,2,3,\ldots,I \quad j = 1,2,3,\ldots,J$$

Bins can either be assigned (1) or not (0)

$$(2) \quad y_j \in \{0,1\} \qquad \forall j = 1,2,3,\ldots,J$$

Total weight of each bin cannot exceed 10 kg

$$(3) \quad \sum_{i=1}^{I} w_i x_{ij} \leq 10 y_j \qquad \forall j = 1,2,3,\ldots,J$$

All items must be assigned

$$(4) \quad \sum_{j=1}^{J} x_{ij} = 1 \qquad \forall i = 1,2,3,\ldots,I$$

EARL conference

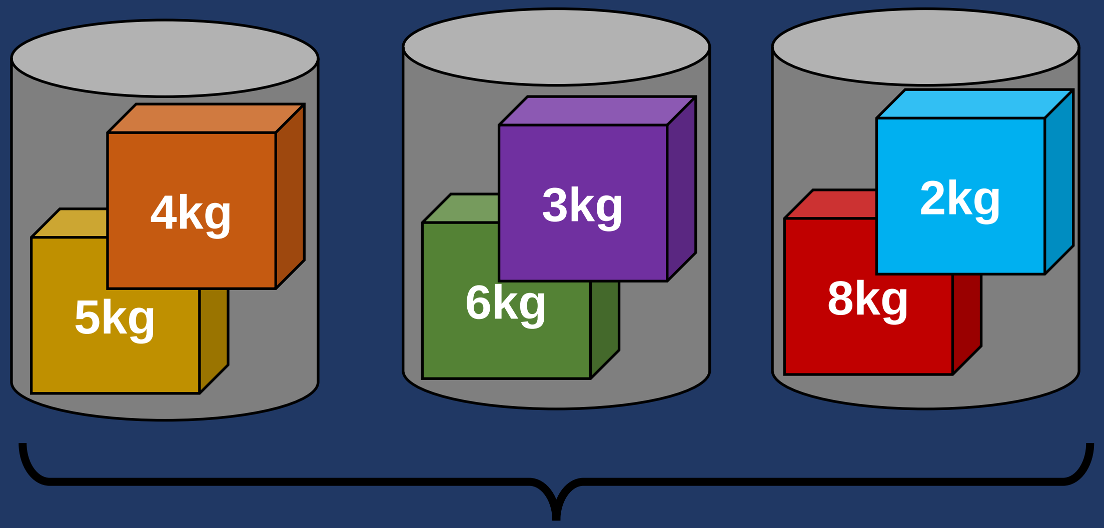
Bin Packing Problem

```r
library(ROI.plugin.glpk)
library(ompr)
library(ompr.roi)
library(dplyr)

max_capacity <- 10
w <- c(2, 3, 4, 5, 6, 8)
N <- length(v)

max_bins <- 4

result <- MIPModel() |>
  add_variable(y[j], j = 1:max_bins, type = "binary") |>
  add_variable(x[i, j], i = 1:N, j = 1:max_bins, type = "binary") |>
  set_objective(sum_over(y[j], j = 1:max_bins), "min") |>
  add_constraint(sum_over(w[i] * x[i, j], i = 1:N) <= y[j] * max_capacity,
                 j = 1:max_bins) |>
  add_constraint(sum_over(x[i, j], j = 1:max_bins) == 1, i = 1:N) |>
  solve_model(with_ROI(solver = "glpk", verbose = TRUE))

solution <- result |>
  get_solution(x[i, j])

solution |>
  filter(value > 0)

```

EARL conference

Bin Packing Problem

- Put this into context again…



Max 10kg each

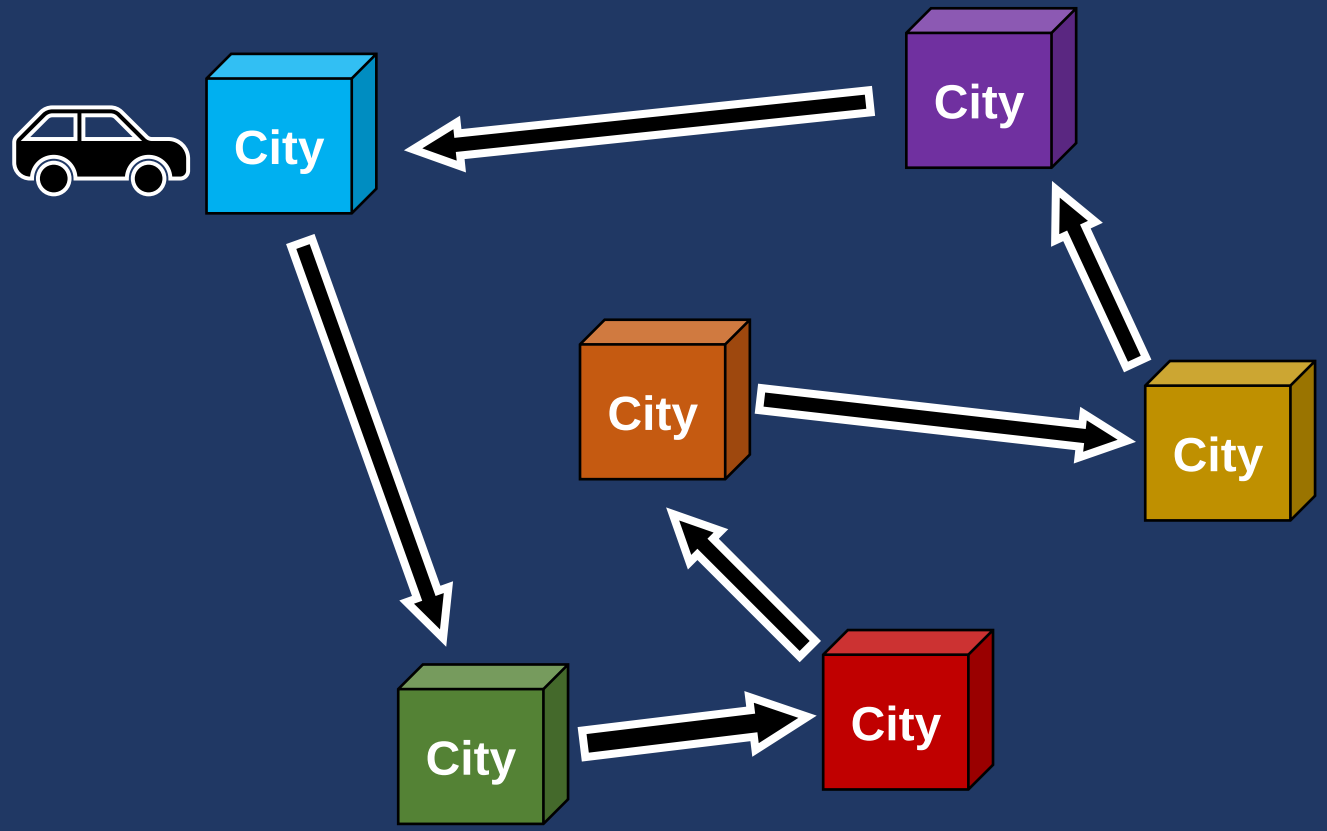
- Now it can handle multiple jobs and multiple resources!

- Still doesn't figure out the order of the jobs
- Doesn't address the starting / finishing point
- Doesn't handle value of the jobs

EARL conference

Travelling Salesman Problem (TSP)

- Find out the shortest path to visit each city exactly once and return to the original city



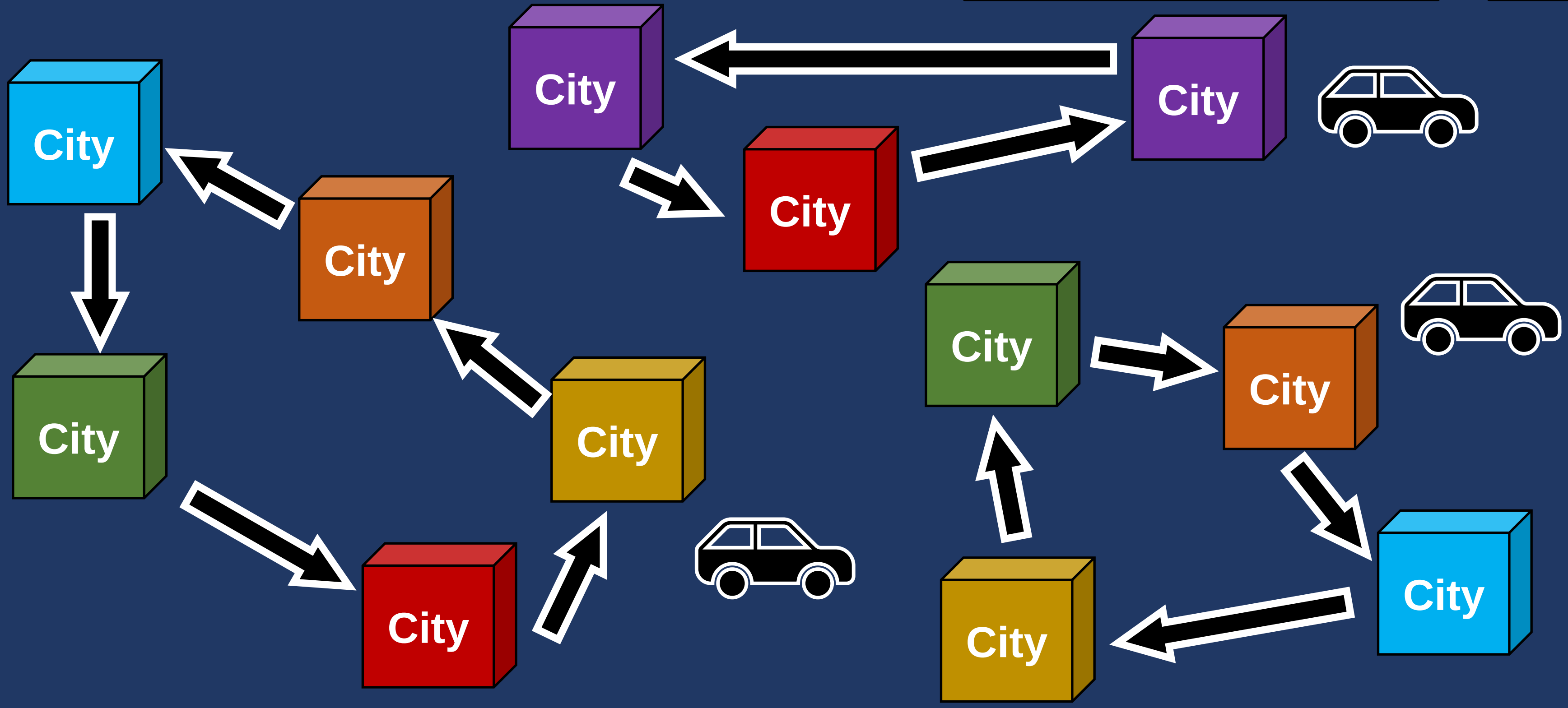
EARL conference

Multiple TSP

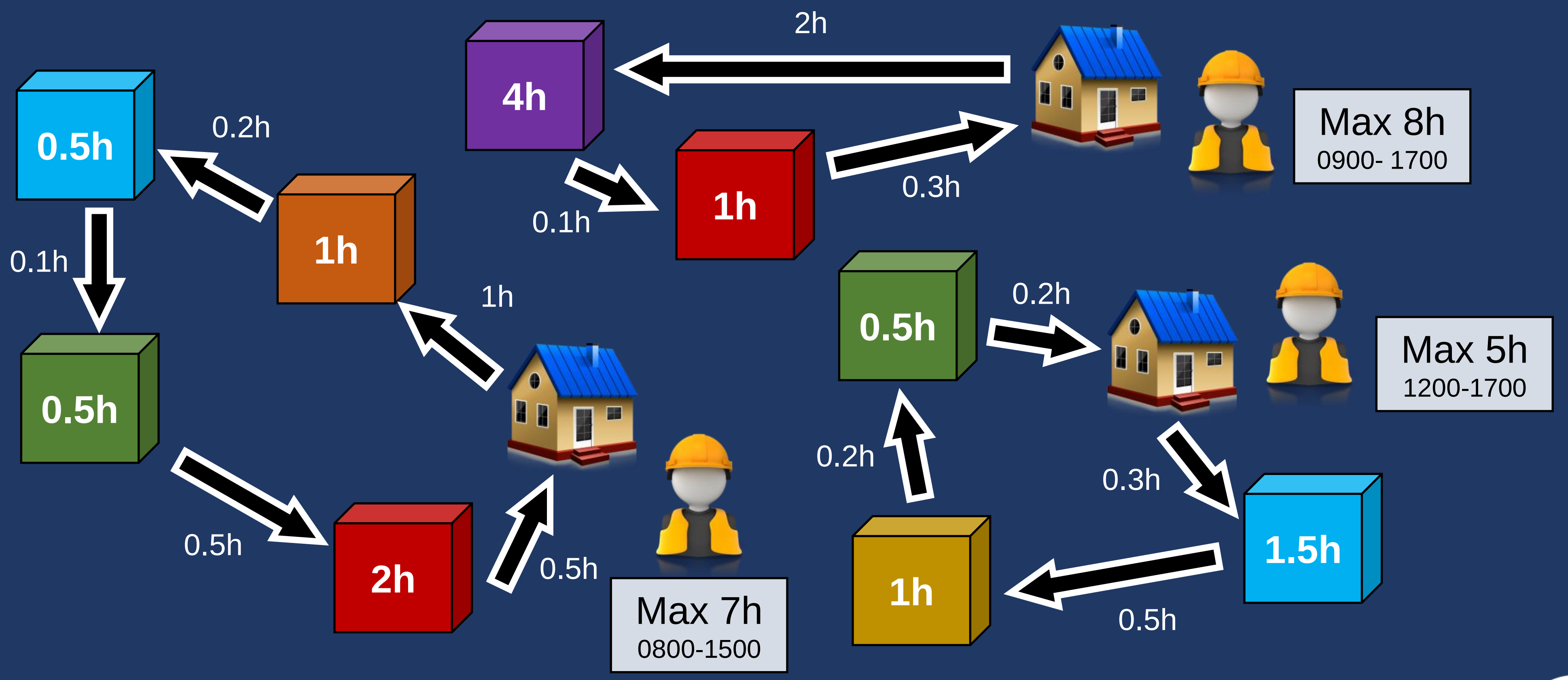
- Similar to TSP but with multiple salesmen.

- Order of jobs
- Completes a circuit

- Doesn’t take into account the duration of jobs / resource capacity
- Starting point not fixed



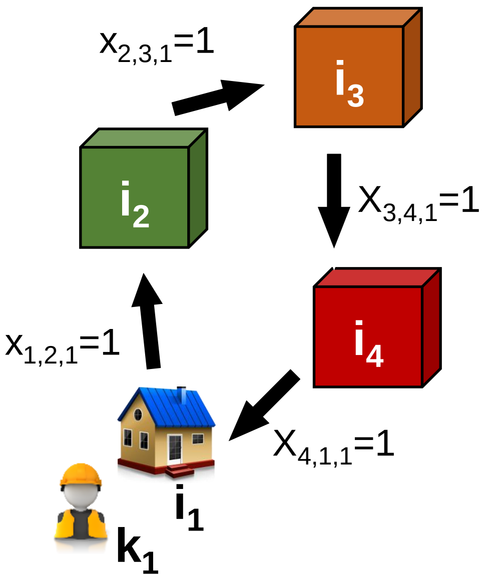
Job Scheduling and Route Optimisation



EARL conference

Job Scheduling and Route Optimisation

Let
- $I =$ No. of locations
- $K =$ No. of travelling agents
- $x =$ Allocation matrix
- $v =$ Travel costs matrix
- $d =$ Job costs vector
- $f =$ Offsets vector
- $c =$ Capacities vector
- $p =$ Depots vector



Min.

$$z = \sum_{i=1}^{I}\sum_{i'=1}^{I}\sum_{k=1}^{K} v_{ii'}x_{ii'k} + d_{i'}x_{ii'k}$$

s.t.

(1) $x_{ii'k} \in \{0,1\}$ $\quad \forall i = 1,2,3,\ldots,I;\ i' = 1,2,3,\ldots,I;\ k' = 1,2,3,\ldots,K$

(2) $\sum_{i=1}^{I}\sum_{i'=1}^{I} v_{ii'}x_{ii'k} + d_{i'}x_{ii'k} \leq f_k + c_k$ $\quad \forall k = 1,2,3,\ldots,K$

(3) $x_{iik} = 0$ $\quad \forall i = 1,2,3,\ldots,I;\ k = 1,2,3,\ldots,K$

(4) $\sum_{i'=1}^{I} x_{p_k i' k} = 1$ $\quad \forall k = 1,2,3,\ldots,K$

(5) $\sum_{i=1}^{I} x_{i p_k k} = 1$ $\quad \forall k = 1,2,3,\ldots,K$

(6) $\sum_{i'=1}^{I} x_{i'ik} = \sum_{i'=1}^{I} x_{ii'k}$ $\quad \forall i = 1,2,3,\ldots,I;\ k = 1,2,3,\ldots,K$

(7) $\sum_{i'=1}^{I}\sum_{k=1}^{K} x_{ii'k} = 1$ $\quad \forall i = 1,2,3,\ldots,I$

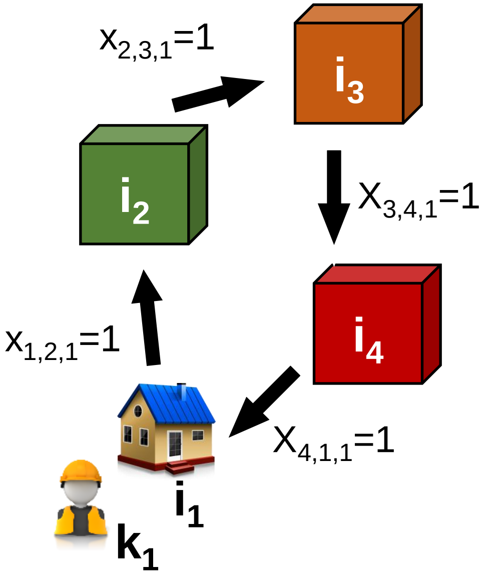
(8) $\sum_{i=1}^{I}\sum_{k=1}^{K} x_{ii'k} = 1$ $\quad \forall i' = 1,2,3,\ldots,I$

(9) $1 \leq u_{ik} \leq I$ $\quad \forall k = 1,2,3,\ldots,K;\ i = 1,2,3,\ldots,I;\ i \neq p_k$

(10) $\left.\begin{array}{l} u_{ik} \geq 2 \\ u_{ik} - u_{i'k} + 1 \leq (I-1)(1 - x_{ii'k}) \end{array}\right\}$ $\quad \forall i = 1,2,3,\ldots,I;\ k = 1,2,3,\ldots,K$

Job Scheduling and Route Optimisation

Let
I = No. of locations
K = No. of travelling agents
x = Allocation matrix
v = Travel costs matrix
d = Job costs vector
f = Offsets vector
c = Capacities vector
p = Depots vector



Min.

$$z = \sum_{i=1}^{I}\sum_{i'=1}^{I}\sum_{k=1}^{K} v_{ii'}x_{ii'k} + d_{i'}x_{ii'k}$$ ← Minimise total cost (eg. time)

s.t.

(1) $x_{ii'k} \in \{0,1\}$ $\quad \forall i = 1,2,3,\ldots,I;\ i' = 1,2,3,\ldots,I;\ k' = 1,2,3,\ldots,K$ ← Allocation vector

(2) $\sum_{i=1}^{I}\sum_{i'=1}^{I} v_{ii'}x_{ii'k} + d_{i'}x_{ii'k} \leq f_k + c_k$ $\quad \forall k = 1,2,3,\ldots,K$ ← Total travel and job costs must not exceed agent capacity

(3) $x_{iik} = 0$ $\quad \forall i = 1,2,3,\ldots,I;\ k = 1,2,3,\ldots,K$ ← Cannot revisit the same job

(4) $\sum_{i'=1}^{I} x_{p_k i'k} = 1$ $\quad \forall k = 1,2,3,\ldots,K$ ← Every worker must leave home

(5) $\sum_{i=1}^{I} x_{ip_k k} = 1$ $\quad \forall k = 1,2,3,\ldots,K$ ← Every worker must come back home

(6) $\sum_{i'=1}^{I} x_{i'ik} = \sum_{i'=1}^{I} x_{ii'k}$ $\quad \forall i = 1,2,3,\ldots,I;\ k = 1,2,3,\ldots,K$ ← Every worker must leave the job after attending it

(7) $\sum_{i'=1}^{I}\sum_{k=1}^{K} x_{ii'k} = 1$ $\quad \forall i = 1,2,3,\ldots,I$ ← Exactly one worker goes to each job

(8) $\sum_{i=1}^{I}\sum_{k=1}^{K} x_{ii'k} = 1$ $\quad \forall i' = 1,2,3,\ldots,I$ ← Exactly one worker leaves each job

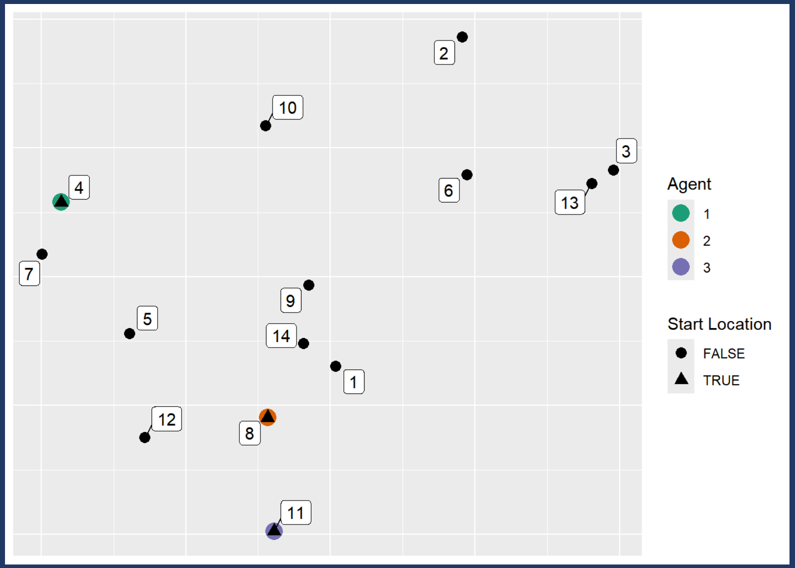
(9) $1 \leq u_{ik} \leq I$ $\quad \forall k = 1,2,3,\ldots,K;\ i = 1,2,3,\ldots,I;\ i \neq p_k$

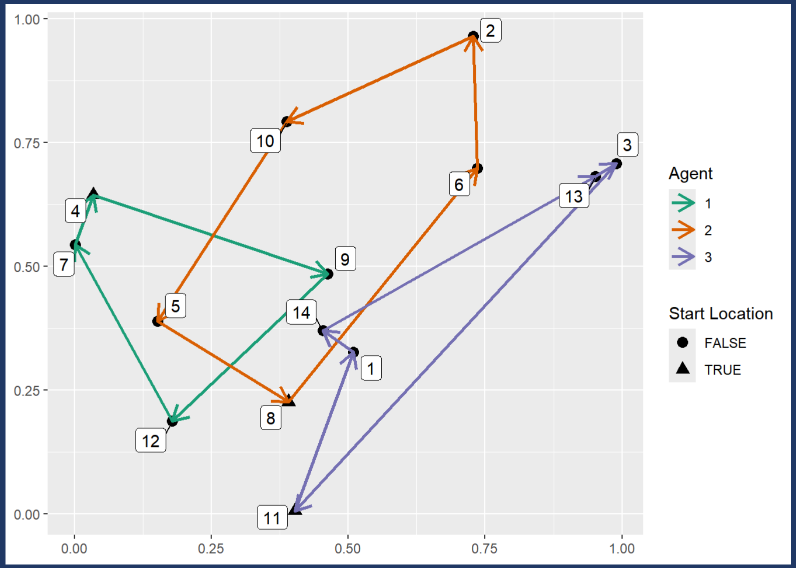
(10) $\left.\begin{array}{l} u_{ik} \geq 2 \\ u_{ik} - u_{i'k} + 1 \leq (I-1)(1 - x_{ii'k}) \end{array}\right\}$ $\quad \forall i = 1,2,3,\ldots,I;\ k = 1,2,3,\ldots,K$

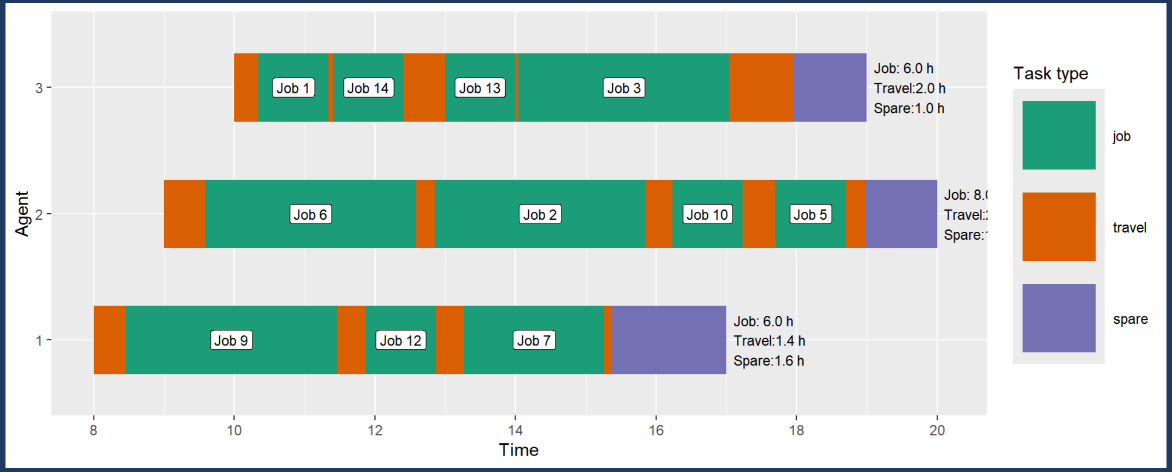
(9)–(10) ← Ensure no subtour

Notebook Example

https://timothywong731.github.io/scheduling/







conference

Q&A

Using Linear Programming for Route Planning and Job Scheduling

https://timothywong731.github.io/scheduling/

Today's notebook example

Timothy Wong CStat CEng MBCS

Senior Data Scientist

timothy.wong@hotmail.co.uk

linkedin.com/in/timothywong731

@timothywong731

timothywong731.github.io