INFORMS Transactions on Education

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

A Tokyo Elevator Puzzle

Martin J. Chlond,

A Tokyo Elevator Puzzle

Martin J.Chlond
Lancashire Business School
University of Central Lancashire
Preston, PR1 2HE UK
mchlond@uclan.ac.uk

1. Introduction

The following "Elevator puzzle" is taken from "The Tokyo Puzzles" by Kojon Fujimura, first published in Japanese in 1969 and subsequently translated into English (Fujimura, 1979).

Figure 1 represents an eight-storey building with three elevators. The shaded cells represent those floors at which a particular elevator calls. Notice that all elevators call at both top and bottom floors and four floors in between. It is possible, given this configuration, to travel between any two floors by riding a single elevator. The question is, if each lift called at three floors in between top and bottom, how many lifts would be required such that any two floors are connected by a single elevator?

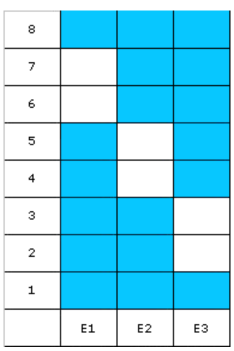

Figure 1: Eight-storey building with three elevators, calling at the shaded floors.

OR/MS practitioners will recognize this puzzle as a set-covering problem (Wolsey, 1998). Briefly, a set-covering problem is a binary integer program where the objective function coefficients and right-hand-sides all have the value 1, all the constraints are of the "greater than or equal to" type and the left-hand-side matrix consists exclusively of 1's and 0's.

2. Formulation

Given the condition that all lifts must call at the top and bottom floors we may ignore these for the purposes of formulation. We will use the notation $E\{n, r\}$ to represent an elevator problem with n floors, excluding top and bottom, where each elevator visits r floors in between top and bottom. We then define a matrix $\boldsymbol{C}$ with m rows (where $m = {}_nC_r$) and n columns and each row consists of a particular combination of r floors visited, specifically, $c_{i,j} = 1$ if the combination i involves a visit to floor j. $\boldsymbol{C}$ contains a complete list of the combinations of r visits. Finally, we define sets $M = \{1, ..., m\}$ and $N = \{1, ..., n\}$ and binary decision variables $x_i = 1$ if combination i is adopted.

We wish to minimize the number of elevators (combinations) required subject to the condition that there is an elevator available for each pair of floors. This may be achieved as follows.

An OPL model of this formulation is here[1] and a dataset to solve the problem for six floors and three visits per elevator is here[2]. The solution generated, with top and bottom floors added back, is shown in Figure 2.

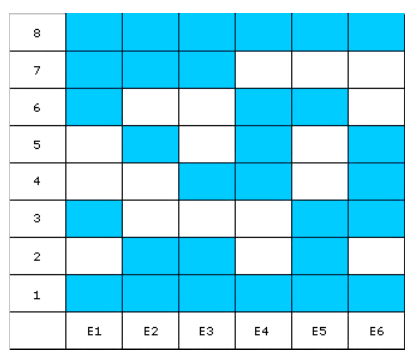

Figure 2: Solution for E{6, 3}.

(1) http://ite.pubs.informs.org/Vol6No3/Chlond/elevator.php
(2) http://ite.pubs.informs.org/Vol6No3/Chlond/data1.php

An Excel spreadsheet with Solver instructions for this puzzle instance is here[3].

A brief discussion of this type of puzzle is given by Martin Gardner (1973) and he asks for a solution for ten floors (including top and bottom) with elevators each calling at four floors in between. The dataset for the original problem was created manually but for this larger problem it is more convenient to write a function to automate the process of listing all possible combinations of floors. This is left as a programming exercise for the interested reader.

A dataset to solve Gardner's problem is >here[4] and a solution is shown in Figure 3.

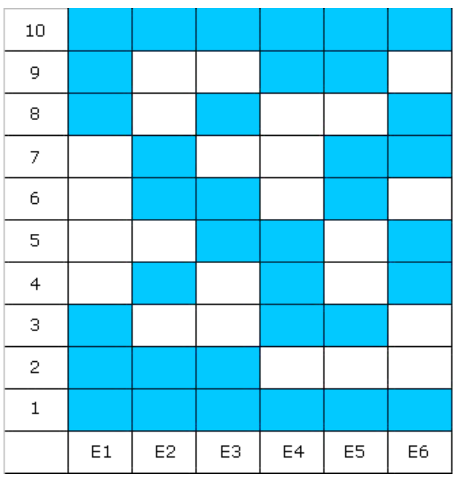

Figure 3: Solution for E{8, 4}.

References

Fujimura, K. (1979), *The Tokyo Puzzles*, Frederick Muller Limited, London

Gardner, M. (1973), "Elevator Puzzles," *Scientific American* Vol. 228, No. 2, pp. 106-109

Wolsey, L.A. (1998), *Integer Programming*, John Wiley and Sons

(3) http://ite.pubs.informs.org/Vol6No3/Chlond/elevator.xls
(4) http://ite.pubs.informs.org/Vol6No3/Chlond/data2.php

Appendix

elevator.mod

```
/*
 Model name   : elevator.mod
 Description  : solves Elevator puzzles
 Source       : The Tokyo Puzzles - Kobon Fujimura
 Date written : 7/8/06
 Written by   : Martin Chlond, Lancashire Business School
 Email        : mchlond@uclan.ac.uk
*/

int   m = ...;
int   n = ...;

range  M = 1..m;
range  N = 1..n;

float  c[M,N] = ...;
dvar boolean x[M];

minimize sum(i in M) x[i];

subject to {

  /* each pair of floors connected by at least one elevator */
  forall(j in N,k in N) sum(i in M) c[i,j]*c[i,k]*x[i] >= 1;

}

execute {

  /* format and display output */
  for(j in N) {
    for(i in M) {
      if(x[i]==1) {
        write(c[i][j]);
        write(" ");
      }
    }
    writeln();
  }
}
```

Data Set 1

```
/*
Data set for elevator puzzle
6 floors, 3 visits (excluding top and bottom)
*/

m = 20;
n = 6;

c = [
[1, 1, 1, 0, 0, 0],
[1, 1, 0, 1, 0, 0],
[1, 1, 0, 0, 1, 0],
[1, 1, 0, 0, 0, 1],
[1, 0, 1, 1, 0, 0],
[1, 0, 1, 0, 1, 0],
[1, 0, 1, 0, 0, 1],
[1, 0, 0, 1, 1, 0],
[1, 0, 0, 1, 0, 1],
[1, 0, 0, 0, 1, 1],
[0, 1, 1, 1, 0, 0],
[0, 1, 1, 0, 1, 0],
[0, 1, 1, 0, 0, 1],
[0, 1, 0, 1, 1, 0],
[0, 1, 0, 1, 0, 1],
[0, 1, 0, 0, 1, 1],
[0, 0, 1, 1, 1, 0],
[0, 0, 1, 1, 0, 1],
[0, 0, 1, 0, 1, 1],
[0, 0, 0, 1, 1, 1]];
```

Data Set 2

```
/*
Data set for elevator puzzle
8 floors, 4 visits (excluding top and bottom)
*/
m = 70;
n = 8;

c = [
[ 0 , 0 , 0 , 0 , 1 , 1 , 1 , 1 ],
[ 0 , 0 , 0 , 1 , 0 , 1 , 1 , 1 ],
[ 0 , 0 , 1 , 0 , 0 , 1 , 1 , 1 ],
[ 0 , 1 , 0 , 0 , 0 , 1 , 1 , 1 ],
[ 1 , 0 , 0 , 0 , 0 , 1 , 1 , 1 ],
[ 0 , 0 , 0 , 1 , 1 , 0 , 1 , 1 ],
[ 0 , 0 , 1 , 0 , 1 , 0 , 1 , 1 ],
[ 0 , 1 , 0 , 0 , 1 , 0 , 1 , 1 ],
[ 1 , 0 , 0 , 0 , 1 , 0 , 1 , 1 ],
[ 0 , 0 , 1 , 1 , 0 , 0 , 1 , 1 ],
[ 0 , 1 , 0 , 1 , 0 , 0 , 1 , 1 ],
[ 1 , 0 , 0 , 1 , 0 , 0 , 1 , 1 ],
[ 0 , 1 , 1 , 0 , 0 , 0 , 1 , 1 ],
[ 1 , 0 , 1 , 0 , 0 , 0 , 1 , 1 ],
[ 1 , 1 , 0 , 0 , 0 , 0 , 1 , 1 ],
[ 0 , 0 , 0 , 1 , 1 , 1 , 0 , 1 ],
[ 0 , 0 , 1 , 0 , 1 , 1 , 0 , 1 ],
[ 0 , 1 , 0 , 0 , 1 , 1 , 0 , 1 ],
[ 1 , 0 , 0 , 0 , 1 , 1 , 0 , 1 ],
[ 0 , 0 , 1 , 1 , 0 , 1 , 0 , 1 ],
[ 0 , 1 , 0 , 1 , 0 , 1 , 0 , 1 ],
[ 1 , 0 , 0 , 1 , 0 , 1 , 0 , 1 ],
[ 0 , 1 , 1 , 0 , 0 , 1 , 0 , 1 ],
[ 1 , 0 , 1 , 0 , 0 , 1 , 0 , 1 ],
[ 1 , 1 , 0 , 0 , 0 , 1 , 0 , 1 ],
[ 0 , 0 , 1 , 1 , 1 , 0 , 0 , 1 ],
[ 0 , 1 , 0 , 1 , 1 , 0 , 0 , 1 ],
[ 1 , 0 , 0 , 1 , 1 , 0 , 0 , 1 ],
[ 0 , 1 , 1 , 0 , 1 , 0 , 0 , 1 ],
[ 1 , 0 , 1 , 0 , 1 , 0 , 0 , 1 ],
[ 1 , 1 , 0 , 0 , 1 , 0 , 0 , 1 ],
[ 0 , 1 , 1 , 1 , 0 , 0 , 0 , 1 ],
[ 1 , 0 , 1 , 1 , 0 , 0 , 0 , 1 ],
[ 1 , 1 , 0 , 1 , 0 , 0 , 0 , 1 ],
[ 1 , 1 , 1 , 0 , 0 , 0 , 0 , 1 ],
[ 0 , 0 , 0 , 1 , 1 , 1 , 1 , 0 ],
[ 0 , 0 , 1 , 0 , 1 , 1 , 1 , 0 ],
[ 0 , 1 , 0 , 0 , 1 , 1 , 1 , 0 ],
[ 1 , 0 , 0 , 0 , 1 , 1 , 1 , 0 ],
[ 0 , 0 , 1 , 1 , 0 , 1 , 1 , 0 ],
[ 0 , 1 , 0 , 1 , 0 , 1 , 1 , 0 ],
[ 1 , 0 , 0 , 1 , 0 , 1 , 1 , 0 ],
[ 0 , 1 , 1 , 0 , 0 , 1 , 1 , 0 ],
[ 1 , 0 , 1 , 0 , 0 , 1 , 1 , 0 ],
[ 1 , 1 , 0 , 0 , 0 , 1 , 1 , 0 ],
[ 0 , 0 , 1 , 1 , 1 , 0 , 1 , 0 ],
[ 0 , 1 , 0 , 1 , 1 , 0 , 1 , 0 ],
[ 1 , 0 , 0 , 1 , 1 , 0 , 1 , 0 ],
[ 0 , 1 , 1 , 0 , 1 , 0 , 1 , 0 ],
[ 1 , 0 , 1 , 0 , 1 , 0 , 1 , 0 ],
[ 1 , 1 , 0 , 0 , 1 , 0 , 1 , 0 ],
[ 0 , 1 , 1 , 1 , 0 , 0 , 1 , 0 ],
[ 1 , 0 , 1 , 1 , 0 , 0 , 1 , 0 ],
[ 1 , 1 , 0 , 1 , 0 , 0 , 1 , 0 ],
[ 1 , 1 , 1 , 0 , 0 , 0 , 1 , 0 ],
[ 0 , 0 , 1 , 1 , 1 , 1 , 0 , 0 ],
[ 0 , 1 , 0 , 1 , 1 , 1 , 0 , 0 ],
[ 1 , 0 , 0 , 1 , 1 , 1 , 0 , 0 ],
[ 0 , 1 , 1 , 0 , 1 , 1 , 0 , 0 ],
[ 1 , 0 , 1 , 0 , 1 , 1 , 0 , 0 ],
[ 1 , 1 , 0 , 0 , 1 , 1 , 0 , 0 ],
[ 0 , 1 , 1 , 1 , 0 , 1 , 0 , 0 ],
[ 1 , 0 , 1 , 1 , 0 , 1 , 0 , 0 ],
[ 1 , 1 , 0 , 1 , 0 , 1 , 0 , 0 ],
[ 1 , 1 , 1 , 0 , 0 , 1 , 0 , 0 ],
[ 0 , 1 , 1 , 1 , 1 , 0 , 0 , 0 ],
[ 1 , 0 , 1 , 1 , 1 , 0 , 0 , 0 ],
[ 1 , 1 , 0 , 1 , 1 , 0 , 0 , 0 ],
[ 1 , 1 , 1 , 0 , 1 , 0 , 0 , 0 ],
[ 1 , 1 , 1 , 1 , 0 , 0 , 0 , 0 ]];
```