Optimization, Project 1

Matthew Grasinger

2015/10/14

1 Problem Setup

1.1 Cost Function

The unknowns in the river crossing problem are the number of trips for each particular type of trip across the river. These trips are organized into forward trips,

$$\mathbf{f} = \begin{bmatrix} f_{\alpha\alpha} & f_{\alpha\beta} & f_{\alpha\gamma} & f_{\alpha\delta} & f_{\beta\beta} & f_{\beta\gamma} & f_{\beta\delta} & f_{\gamma\gamma} & f_{\gamma\delta} & f_{\delta\delta} \end{bmatrix}^T \tag{1}$$

where $f_{\alpha\alpha}$ is the number of forward trips that α takes itself, $f_{\alpha\beta}$ is the number of forward trips that α and β take together, $f_{\alpha\gamma}$ is the number of forward trips that α and γ take together, and so on and so forth. And backward trips,

$$\mathbf{b} = \begin{bmatrix} b_\alpha & b_\beta & b_\gamma & b_\delta \end{bmatrix}^T \tag{2}$$

where b_α is the number of backward trips that α takes, b_β is the number of backward trips that β and takes, b_γ is the number of backward trips that γ takes, and b_δ is the number of backward trips that δ takes. Note that, there is no need to consider backward trips with multiple rowers because it is obvious no such trips will be part of the optimal solution.

In order to express the river crossing problem in standard form, the unknowns must be organized into a single vector of unknowns. This is expressed formally in the following equation:

$$\mathbf{x} = \begin{bmatrix} \mathbf{f} \\ \mathbf{b} \end{bmatrix} \tag{3}$$

where $\mathbf{x}$ is a 14 element column vector representing the unknowns in the problem. The cost associated with any trip is equivalent to the slower rower's time needed to cross the stream. The cost vector is given by the following equation:

$$\mathbf{c} = \begin{bmatrix} 1 \\ 2 \\ 5 \\ 10 \\ 2 \\ 5 \\ 10 \\ 5 \\ 10 \\ 10 \\ 1 \\ 2 \\ 5 \\ 10 \end{bmatrix} \tag{4}$$

The goal of the linear program then is

$$\min \mathbf{c}^T \mathbf{x} \tag{5}$$

1.2 Constraints

Two of the more straight forward constraints of the problem can be expressed in words as, (1) the number of trips for any particular type of trip must be positive, i.e. we cannot have a negative number of trips, and (2) everyone must end up on the opposite side of the river. These constraints are given formally as, $\mathbf{x} \geq \mathbf{0}$ and,

$$\begin{aligned} f_{\alpha\alpha} + f_{\alpha\beta} + f_{\alpha\gamma} + f_{\alpha\delta} - b_\alpha &= 1 \\ f_{\alpha\beta} + f_{\beta\beta} + f_{\beta\gamma} + f_{\beta\delta} - b_\beta &= 1 \\ f_{\alpha\gamma} + f_{\beta\gamma} + f_{\gamma\gamma} + f_{\gamma\delta} - b_\gamma &= 1 \\ f_{\alpha\delta} + f_{\beta\delta} + f_{\gamma\delta} + f_{\delta\delta} - b_\delta &= 1 \end{aligned} \tag{6}$$

Two further assumptions are made involving the number of forward trips and the number of backward trips. Some thought shows that the optimal solutions will consist of three forward trips and two backward trips. One can arrive at the conclusion by considering the facts that (1) only two people can travel across the river in the canoe at a time, and (2) someone must travel back across the river with the canoe to bring more people across, i.e. the canoe cannot take itself back across the river. These two constraints are express formally as,

$$f_{\alpha\alpha} + f_{\alpha\beta} + f_{\alpha\gamma} + f_{\alpha\delta} + f_{\beta\beta} + f_{\beta\gamma} + f_{\beta\delta} + f_{\gamma\gamma} + f_{\gamma\delta} + f_{\delta\delta} = 3 \tag{7}$$

$$b_\alpha + b_\beta + b_\gamma + b_\delta = 2$$

In order to express these constraints in standard form, they must be expressed in the form of $\mathbf{Ax} = \mathbf{b}$. $\mathbf{x}$ was defined in Equation 3. $\mathbf{A}$ and $\mathbf{b}$ are expressed formally as,

$$\mathbf{A} = \left[\begin{array}{cccccccccc} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & & & & & & \\ \hline 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & & & & & & \\ \hline 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & -1 & 0 & & & & & & \\ \hline 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & -1 & & & & & & \\ \hline 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & & & & & & \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & & & & & & \end{array}\right] \tag{8}$$

$$\mathbf{b} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 3 \\ 2 \end{bmatrix}$$

2 MATLAB Code

The following matlab script was written and ran in order to solve the river crossing problem:

```
% change algorithm to simplex method
options = optimoptions('linprog','Algorithm','simplex');

% faa, fab, fag, fad, fbb, fbg, fbd, fgg, fgd, fdd, ba, bb, bg, bd
%   1,   2,   3,   4,   5,   6,   7,   8,   9,  10, 11, 12, 13, 14

f = [1; 2; 5; 10; 2; 5; 10; 5; 10; 10; 1; 2; 5; 10];

Aeq = [
1   1   1   1   1   1   1   1   1   1   0   0   0   0;
1   1   1   1   0   0   0   0   0   0   -1  0   0   0;
0   1   0   0   1   1   1   0   0   0   0   -1  0   0;
0   0   1   0   0   1   0   1   1   0   0   0   -1  0;
0   0   0   1   0   0   1   0   1   1   0   0   0   -1;
0   0   0   0   0   0   0   0   0   0   1   1   1   1
];

beq = [3   1   1   1   1   2];

num_vars = size(Aeq, 2);
LB = zeros(num_vars);
UB(1:num_vars) = Inf;

[x, fval, exitflag, output] = linprog(f, [], [], Aeq, beq, LB, UB);

% faa, fab, fag, fad, fbb, fbg, fbd, fgg, fgd, fdd, ba, bb, bg, bd
%   1,   2,   3,   4,   5,   6,   7,   8,   9,  10, 11, 12, 13, 14
if (x(1) > 1e-6)
    fprintf('%.0f = the number of forward trips alpha makes alone\n', x(1));
end
if (x(2) > 1e-6)
    fprintf('%.0f = the number of forward trips alpha makes with beta\n', x(2));
```

```
end
if (x(3) > 1e-6)
    fprintf('%.0f = the number of forward trips alpha makes with gamma\n', x(3));
end
if (x(4) > 1e-6)
    fprintf('%.0f = the number of forward trips alpha makes with delta\n', x(4));
end
if (x(5) > 1e-6)
    fprintf('%.0f = the number of forward trips beta makes alone\n', x(5));
end
if (x(6) > 1e-6)
    fprintf('%.0f = the number of forward trips beta makes with gamma\n', x(6));
end
if (x(7) > 1e-6)
    fprintf('%.0f = the number of forward trips beta makes with delta\n', x(7));
end
if (x(8) > 1e-6)
    fprintf('%.0f = the number of forward trips gamma makes alone\n', x(8));
end
if (x(9) > 1e-6)
    fprintf('%.0f = the number of forward trips gamma makes with delta\n', x(9));
end
if (x(10) > 1e-6)
    fprintf('%.0f = the number of forward trips delta makes alone\n', x(10));
end
if (x(11) > 1e-6)
    fprintf('%.0f = the number of backward trips alpha makes\n', x(11));
end
if (x(12) > 1e-6)
    fprintf('%.0f = the number of backward trips beta makes\n', x(12));
end
if (x(13) > 1e-6)
    fprintf('%.0f = the number of backward trips gamma makes\n', x(13));
end
if (x(14) > 1e-6)
    fprintf('%.0f = the number of backward trips delta makes\n', x(14));
end

fprintf('The total time taken for everyone to cross = %.0f\n', f'*x);
```

3 Results

The output of the script was:

```
2 = the number of forward trips alpha makes with beta
1 = the number of forward trips gamma makes with delta
1 = the number of backward trips alpha makes
1 = the number of backward trips beta makes
The total time taken for everyone to cross = 17
```

Although the linear program, as set up, has no concept of the order in which the sequence of events happen, it can be seen that the constraints were sufficient to produce an answer that can be ordered in a logically consistent sequence of events. The optimal solution can be organized into two different sequences, both of which are an equivalent cost. The first possible sequence is,

1. α and β make a forward trip together.
2. α makes a backward trip.
3. γ and δ make a forward trip together.

4. β makes a backward trip.
5. α and β make a forward trip together.

The second possible sequence is,

1. α and β make a forward trip together.
2. β makes a backward trip.
3. γ and δ make a forward trip together.
4. α makes a backward trip.
5. α and β make a forward trip together.