8 Optimization Theory

8.4 The out-of-kilter algorithm (9 units)

The out-of-kilter algorithm is described in the IIA course, Algorithms and Networks, or in the reference at the end of this project description. Note that this project requires more extensive programming than many others; you should only attempt it if you are happy writing and debugging programs.

1 The Algorithm

The out-of-kilter algorithm is a general purpose algorithm that can be used to minimise the total cost of a constrained flow in an oriented network. Suppose the network has n nodes and m oriented arcs. We write $j \sim (i, i')$ if arc j has initial node i and terminal node i'. Let $x(j)$ be the flow along arc j (from node i to node i'). Define $c^-(j)$ and $c^+(j)$ to be the lower and upper capacity bounds on the flow in arc j. For the purposes of this project all the lower and upper capacities may be taken to be finite (though the general form of the algorithm can cope with the case where either or both are infinite on some or all arcs). Constants $d(j)$ give the cost per unit flow along arc j. The problem we wish to solve is

$$\text{minimise} \qquad \sum_{j=1}^{m} d(j)\, x(j)$$

$$\text{subject to} \qquad \sum_{\{j:j\sim(i,i')\}} x(j) \;-\; \sum_{\{j:j\sim(i',i)\}} x(j) = 0 \qquad \text{for each} \quad i = 1, \ldots, n \qquad (1)$$

$$\text{and} \qquad c^-(j) \leqslant x(j) \leqslant c^+(j) \qquad \text{for each} \quad j = 1, \ldots, m\,. \qquad (2)$$

If a flow $\mathbf{x}$ satisfies (1) then the flow is conserved at each node and we call the flow a *circulation*. If the flow $\mathbf{x}$ satisfies (2) we say it is feasible. Our problem is therefore to find the minimal cost feasible circulation for the problem. (The algorithm can easily be generalised to cope with problems with a specified non-zero divergence, but such problems can also be converted to minimal cost circulation problems in a standard way by adding an extra node and extra arcs to the network.)

The algorithm uses dual variables $y(i)$, $i = 1, \ldots, n$. One such variable is associated with each node, and it is usual to refer to $y(i)$ as the *potential* on node i. When $j \sim (i, i')$ the quantity $v(j) = y(i') - y(i)$ is known as the *tension* in the arc j. The algorithm can be summarised as follows.

1. Let $y(i) = 0$, $i = 1, 2, \ldots, n$ and choose $\mathbf{x}$ to be any initial circulation (feasible or not).

2. Check to see if all arcs are *in kilter*, meaning

$$\begin{aligned} x(j) &= c^-(j) && \text{if} \quad d(j) - v(j) > 0 \\ x(j) &= c^+(j) && \text{if} \quad d(j) - v(j) < 0 \\ c^-(j) \leqslant x(j) &\leqslant c^+(j) && \text{if} \quad d(j) - v(j) = 0\,. \end{aligned}$$

3. If so, **stop**, flow is optimal. If not, choose an arc j that is out-of-kilter and try to alter the flow on this arc to bring it closer to kilter, the distance from kilter being defined as the absolute value of the minimal change in flow $x(j)$ needed to bring the arc into kilter.

To do this, for example to increase the flow in arc $j \sim (i, i')$ you must look for a circuit including j around which you can increase the flow *without moving any arc further from kilter*. So, label node i' and successively label any nodes to which you can legally pump flow in this way. If you end up labelling node i, you have found a circuit, so put as much flow as possible around it. Now return to Step 2.

4. If you failed to find a circuit, you have labelled a cut, $[S, S^c]$ with $i' \in S$; $i \in S^c$. Alter the tension across this cut by changing $y(k) \to y(k) + \beta$ for all $k \in S$, where β is chosen so that no arc becomes further from kilter but either arc j comes into kilter, or it becomes possible to label at least one more node in S^c when you return to step 3. (If this is impossible the network is infeasible and there is no feasible circulation.) Now return to Step 2.

Programming Task: Write a program which will read in an initial circulation and then apply this algorithm to find the minimal cost feasible flow. You may assume that all constants and variables only take integer values. Your program should be able to print out the current flow and node numbers (potentials) each time it returns to Step 2. Since you will only be using the program to solve small problems you do *not* need to attempt to write an optimally efficient code; you should concentrate on getting one which works reliably.

Your write-up should include a brief description of how you represent the network in your program, and an indication of how you expect the running time of your algorithm to depend on n and m. You should describe briefly how you carry out the labelling procedure in step 3. Which parts of your program would warrant further attention if it was to be used for solving very large problems and running times became an important issue?

Question 1 Explain briefly how your program deals with the case of an infeasible network in step 4, and outline a proof that the conditions which cause your program to terminate do indeed imply that the network is infeasible.

Question 2 Outline a proof that your algorithm will always terminate (either with an optimal flow or with an indication that the problem is infeasible) in a finite number of steps.

Note: you are not required to give a complete description of the algorithm in your write-up.

2 Tests

Include in your write-up a proof that when all arcs are in kilter the flow is optimal. Test your program on three small examples (which you should make up yourself), one of each of the following possibilities: (a) the initial flow is a feasible but non-optimal circulation; (b) the initial flow is a non-feasible circulation (but the network does allow a feasible circulation); (c) the initial flow is a circulation but the network does not allow a feasible circulation (i.e., the problem is *infeasible*). In each case your output, which should be sufficiently detailed for you to check that the algorithm is performing as expected, should be included (and should not take up much more than one page).

3 Assignment problem

Question 3 Three aircraft can each attack all of three targets with independent hitting probabilities given by

$$\begin{array}{c} \\ \text{Aircraft} \end{array} \begin{array}{c} \text{Target} \\ \begin{pmatrix} 0.2 & 0.3 & 0.6 \\ 0.4 & 0.5 & 0.4 \\ 0.2 & 0.6 & 0.1 \end{pmatrix} \end{array}$$

You wish to assign the targets, one to each aircraft, in such a way that all three targets are covered and the expected number of targets hit is maximal. Construct a network, labelling each arc with its capacity constraints $[c^-(j), c^+(j)]$ and cost per unit flow, $d(j)$, which can be used to solve this problem. Use your program to find the optimal allocation and the expected number of targets hit. Explain why you expect the solution to your network problem to be a solution to the allocation problem (i.e., $x(j) = 0$ or 1). You should start the program 1) with zero initial flow, and 2) with a flow corresponding to allocation 1–1, 2–2, 3–3. Your write-up should contain the network you constructed with the optimal flow, along with the corresponding aircraft-target allocation, the expected number of targets hit, and the output from each run of your program. Also solve the assignment problem

$$\begin{array}{c} \\ \text{Aircraft} \end{array} \begin{array}{c} \text{Target} \\ \begin{pmatrix} 0.39 & 0.65 & 0.69 & 0.66 & 0.45 \\ 0.64 & 0.84 & 0.24 & 0.90 & 0.22 \\ 0.49 & 0.50 & 0.67 & 0.31 & 0.45 \\ 0.48 & 0.65 & 0.55 & 0.23 & 0.50 \\ 0.29 & 0.34 & 0.30 & 0.34 & 0.18 \end{pmatrix} \end{array}$$

in this case including only the final output (flow, potentials and cost) from your program.

Question 4 If you were considering problems of this type where the number of targets and aircraft, N, is large, how would the complexity of your algorithm, as a function of N, compare with solving the problem by trying all possible assignments?

Reference

Linear Programming & Network Flows (2nd Ed). M.S. Bazaraa, J.J. Jarvis & H.D. Sherali. Wiley 1990.

