European Social Fund Prague & EU: We invests in your future.

Combinatorial optimisation
Seminar No. 7
Application of network flows

Zdeněk Bäumelt (baumezde@fel.cvut.cz)
Přemysl Šůcha (suchap@fel.cvut.cz)

April 4, 2013

1 Network flows

A number of practical problems dealing with combinatorial optimisation can be solved using network flows. To be able to imagine the problem the network is a graph where each edge is a pipe through which liquid flows. The goal is to find a flow (maximal, feasible, etc.) in the network on condition that circulation loss is ignored.

Definition 1.1 ***Flow***. *Let G is a directed graph. Network flow is such an evaluation of edges that each edge is weighted by real number $f : E(G) \to \mathbb{R}$ and each node satisfies* ***Kirchhoff's law*** *[1]*

$$\sum_{e \in E^+(v)} f(e) = \sum_{e \in E^-(v)} f(e) \tag{1}$$

where $E^+(v)$ is a set of leaving edges from the node v and $E^-(v)$ is a set of entering edges to the node v.

One of the most frequent problem dealing with network flows is the *Minimum Cost Flow Problem*. The problem is formulated as follows. Let (G, b, l, u, c) is a transport network where G is a directed graph, b is a vector of the source and sink nodes, matrices l and u determine lower and upper bounds of edges, and vector c provides the cost per unit of flow for each edge. The goal is to find the cheapest feasible flow such that for each node $v \in V(G)$ the following is satisfied.

$$\sum_{e \in E^+(v)} f(e) - \sum_{e \in E^-(v)} f(e) = b(v). \tag{2}$$

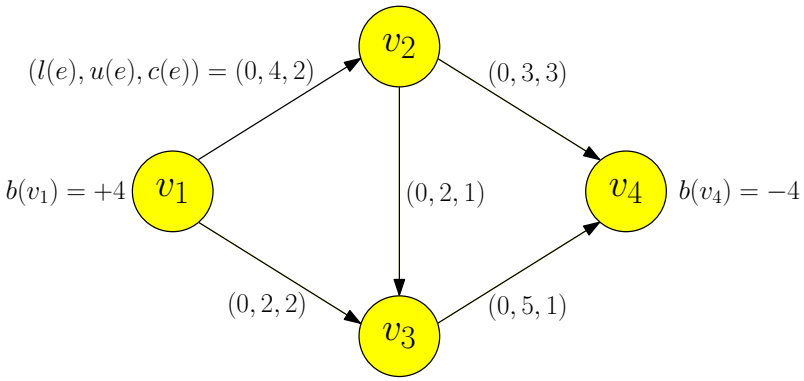


Figure 1: An example of the network.

2 Reconstruction of binary images using network flows

The aim of the exercise is to reconstruct a square binary image using projection data and network flows [2, 3], i.e. each pixel has assigned a black or white colour (0 = black, 1 = white) according to projections. This method, which is often used in medicine to reconstruct a three-dimensional image from the projections scanned by a X-ray machine, will be explained by way of example.

Let's have horizontal and vertical projections $sumR$ (denoted as $\mathcal{R}$) and $sumC$ (denoted as $\mathcal{C}$) respectively. The i-th element of vector $sumR$ is the sum of the pixel values in row i. In a similar way, the j-th element of vector $sumC$ is the sum of the pixel values in column j. The goal is to reconstruct a binary image that is 3 by 3 pixels in size (see Figure 2).

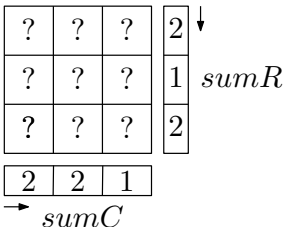


Figure 2: An example of the image which will be reconstructed.

In spite of using projections $\mathcal{R}$ and $\mathcal{C}$ the solution to the problem is ambiguous (see Figure 3). In case of having additional projections (e.g. diagonal projections) the number of solutions can be reduced, for example both images in Figure 3 have the same horizontal and vertical projections but the diagonal projections (i.e. $sumD$ and $sumA$) are different. However, not even four projections are sufficient to ensure an accurate reconstruction of an arbitrary image. On the other hand, the more projections you have the more precise reconstruction you get.

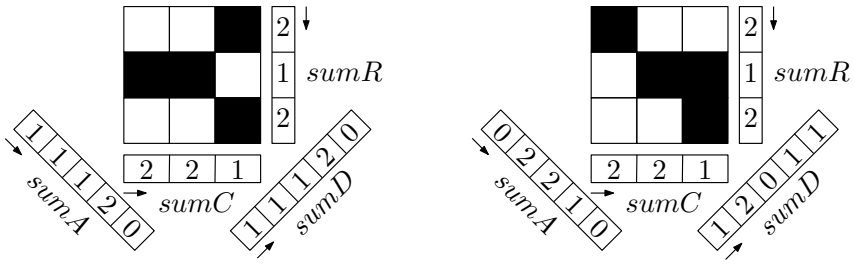


Figure 3: An illustration of the ambiguous solutions.

2.1 Transformation of the projections into network flows

Having given the horizontal and vertical projections a binary image can be reconstructed using network flows, more precisely the *Minimum Cost Flow Problem*. An example of such network graph and corresponding projections is shown in Figure 4. Nodes R_i and C_j correspond with the projections $\mathcal{R}$ and $\mathcal{C}$ respectively. Each edge can be likened to the pixel at position (R_i, C_j) and its maximal flow is limited to one since a reconstructed image is binary.

The above mentioned process can be used independently of an image size. Let $n_1 \times n_2$ is a size of an image then the corresponding graph G has n_1 source nodes R_i, and n_2 sink nodes C_j.

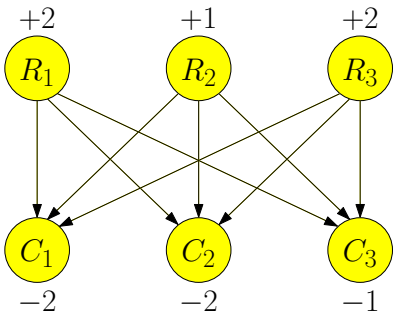


Figure 4: An example of network (G, b, l, u, c) that corresponds to projections $\mathcal{R}$ and $\mathcal{C}$.

2.2 The algorithm for reconstructing binary images

Data: Projections $\mathcal{R}$ and $\mathcal{C}$.
Result: Reconstructed image I.

I is $n_1 \times n_2$ zero matrix;
c_{prev} is $(n_1 + n_2) \times (n_1 + n_2)$ zero matrix;
create vector b and matrices l, u according to $\mathcal{R}, \mathcal{C}$;
for $ic = 1 : countOfIterations$ **do**
  $c = \text{fce}(c_{prev}, I)$;
  $c_{prev} = c$;
  create empty graph G;
  F is the solution of the Minimum Flow Cost Problem using network (G, b, l, u, c);
  transform F into image I;
  display image I;
end

Vector b is created by the following way.

```
>> b = [sumR -sumC]';
```

Matrices u, l, c are created according to the edges where each of them has assigned triple $\big(l(e), u(e), c(e)\big)$. If an image is binary then $\forall e \in E(G) : l(e) = 0, u(e) = 1, c(e) \in \mathbb{R}$. To get details about $c(e)$ calculation please refer to section 2.3. Having created all necessary vector and matrices the Minimum Flow Cost Problem can be solved by the TORSCHE `mincostflow` function.

```
% create empty graph
>> G = graph;
% generate b,l,u,c
% ...
% solve the minimal cost flow problem
>> F = G.mincostflow(c,l,u,b);
```

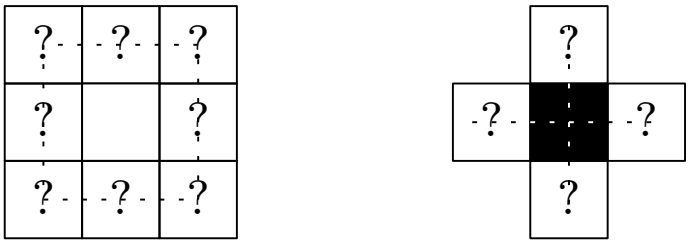
The output of the `mincostflow` function is matrix F which corresponds to the minimum cost feasible flows in network (G, b, l, u, c). Matrix F has to be transformed into a binary image by the following way. For each edge having a non-zero flow set its corresponding pixel to white otherwise leave it black. The algorithm is iterating until a stop criterion is reached, i.e. an image is stable or a given number of iterations is performed. To display image I use the following script.

```
>> subplot(..., ..., ...);
>> imagesc(logical(I));
>> colormap(gray);
>> axis off;
>> axis square;
```

2.3 Calculating the edge cost using knowledge of the image structure

Matrix c is one of the parameters of the `mincostflow` function. If there is no edge between two nodes then set its cost to 0[1]. The cost for each existing edge in graph G is calculated as follows.

1. Set the initial edge cost to zero, i.e. $c(e) = 0$.
2. If a 3×3 pixel neighbourhood can be selected in the previous image (see Figure 5) then continue to the next step otherwise go to step 7.
3. If the central pixel is white and another white pixel cannot be found in its Moore neighbourhood (8 surrounding pixels) then set $c(e) = 1$.
4. If the central pixel is white and exactly one pixel is white in its Moore neighbourhood then set $c(e) = 0.2$.
5. If the central pixel is white and exactly two pixels are white in its Moore neighbourhood then set $c(e) = 0.1$.
6. If the central pixel is black and there is at least one pair of the opposite white pixels in its von Neumann neighbourhood (4 surrounding pixels) then set $c(e)$ = -0.1.
7. The resulting edge cost is $c(e) = c(e) + 0.5 \cdot c_{prev}(e)$ where c_{prev} is the next-to-last edge cost.



(a) Moore neighbourhood (b) von Neumann neighbourhood

Figure 5: Types of neighbourhoods.

3 A seminar assignment

A seminar assignment: Using `sumR = [3,2,4,1]` and `sumC = [2,1,?,3]` projections calculate "?" value and sketch a network graph similar to Figure 4.

4 A homework assignment

A homework assignment: Using projections (vectors *sumR* and *sumC*) saved in file `projectionData.mat` reconstruct a square binary image that is 20 by 20 pixels in size. The Minimum Cost Flow Problem should be solved in a way how it is described in section 2. In case of having a correct implementation the final image should be retrieved in 62 iterations.

References

[1] J. Demel, *Grafy a jejich aplikace*. Academia, second ed., 2002.

[2] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall; United States Ed edition, 1993.

[3] K. J. Batenburg, "A network flow algorithm for reconstructing binary images from discrete x-rays," *J. Math. Imaging Vis.*, vol. 27, no. 2, pp. 175–191, 2007.

[1] The TORSCHE toolbox requires this value to be set to zero but it is more intuitive to set this value to ∞.