Notes on NCSq

Taichi Uemura

1. Implementation

1.1. Notation

For a cell c, we write $\big((l(c), b(c)), (r(c), t(c))\big)$ for the bounding rectangle of c. Let $w = r - l$ be the width and $h = t - b$ the height.

1.2. Automatic stretching

Given a matrix of cells $\left(c_{ij}\right)_{1 \leq i \leq m, 1 \leq j \leq n}$, we want to arrange them satisfying the following:

(1) cells do not overlap each other;

(2) arrows are sufficiently long to match their labels.

To achieve this, we have to determine the boundary $\left(b_i, t_i\right)$ of i-th row and the boundary $\left(l_j, r_j\right)$ of j-th column. These boundaries must satisfies the following:

(1) if c_{ij} is an object, then $b_i \leq b\left(c_{ij}\right)$, $t\left(c_{ij}\right) \leq t_i$, $l_j \leq l\left(c_{ij}\right)$, and $r\left(c_{ij}\right) \leq r_j$;

(2) if c_{ij} is a horizontal arrow, then $b_i \leq b\left(c_{ij}\right)$ and $t\left(c_{ij}\right) \leq t_i$;

(3) if c_{ij} is a vertical arrow, then $l_j \leq l\left(c_{ij}\right)$ and $r\left(c_{ij}\right) \leq r_j$;

(4) if c_{ij} is a horizontal arrow with source $c_{i\sigma}$ and target $c_{i\tau}$ $(1 \leq \sigma < j < \tau \leq n)$, then

$$w\left(c_{ij}\right) \leq \left(r_\sigma - r\left(c_{i\sigma}\right)\right) + \left(l\left(c_{i\tau}\right) - l_\tau\right) + \sum_{\sigma < j' < \tau} w_{ij'}$$

(5) if c_{ij} is a vertical arrow with source $c_{\sigma j}$ and target $c_{\tau j}$ $(1 \leq \sigma < i < \tau \leq m)$, then

$$h\left(c_{ij}\right) \leq \left(b\left(c_{\sigma j}\right) - b_\sigma\right) + \left(t_\tau - t\left(c_{\tau j}\right)\right) + \sum_{\sigma < i' < \tau} h_{i'j}$$

Observe that horizontal adjustment and vertical adjustment are independent of each other, so we may first determine the boundary (b_i, t_i) of i-th row satisfying the following:

(1) if c_{ij} is either an object or a horizontal arrow, then $b_i \leq b(c_{ij})$ and $t(c_{ij}) \leq t_i$;

(2) if c_{ij} is a vertical arrow with source $c_{\sigma j}$ and target $c_{\tau j}$, then

$$h(c_{ij}) \leq \left(b(c_{\sigma j}) - b_\sigma\right) + \left(t_\tau - t(c_{\tau j})\right) + \sum_{\sigma<i'<\tau} h_{i'j}$$

and then determine the boundary (l_j, r_j) of j-th column satisfying the dual conditions. The first condition is immediately solved:

$$b_i \leq \min\left\{b(c_{ij}) \;\middle|\; c_{ij} \text{ is either an object or a horizontal arrow}\right\}$$

$$t_i \geq \max\left\{t(c_{ij}) \;\middle|\; c_{ij} \text{ is either an object or a horizontal arrow}\right\}$$

Let $b_i^{(1)}$ and $t_i^{(1)}$ be the right sides of these inequalities respectively. Put $y_i = b_i^{(1)} - b_i$ and $x_i = t_i - t_i^{(1)}$. Then the second condition is equivalent to that, if c_{ij} is a vertical arrow with source $c_{\sigma j}$ and target $c_{\tau j}$, then

$$d_{ij} \leq y_\sigma + x_\tau + \sum_{\sigma<i'<\tau} x_{i'} + y_{i'} \tag{A_{ij}}$$

where

$$d_{ij} = h(c_{ij}) - \left(\left(b(c_{\sigma j}) - b_\sigma^{(1)}\right) + \left(t_\tau^{(1)} - t(c_{\tau j})\right) + \sum_{\sigma<i'<\tau} h_{i'j}^{(1)}\right)$$

is a constant. Therefore, the goal becomes to find $x_1, y_1, \cdots, x_m, y_m \geq 0$ satisfying (A_{ij}) for every vertical arrow c_{ij} with source $c_{\sigma j}$ and target $c_{\tau j}$.

Clearly, sufficiently large (x_i, y_i)'s satisfy inequations (A_{ij})'s, but it will be better if we choose (x_i, y_i)'s as small as possible. It will be much better if (x_i, y_i)'s are balanced. This will be achieved by minimizing the sum of squares

$$x_1^2 + y_1^2 + \cdots + x_m^2 + y_m^2$$

Thus, vertical arrow stretching is reduced to the following constrained least-squares problem:

$$
\begin{aligned}
\text{minimize} \quad & x_1^2 + y_1^2 + \cdots + x_m^2 + y_m^2 \\
\text{subject to} \quad & d_{i_k j_k} \le y_{\sigma_k} + x_{\tau_k} + \sum_{\sigma_k < i' < \tau_k} x_{i'} + y_{i'} \text{ for each } k = 1, \cdots, p \\
& x_i, y_i \ge 0 \text{ for each } i = 1, \cdots, m
\end{aligned}
$$

where $c_{i_1 j_1}, \cdots, c_{i_p j_p}$ is the list of vertical arrows and $c_{\sigma_{i_k} j_k}$ and $c_{\tau_{i_k} j_k}$ are the source and the target, respectively, of $c_{i_k j_k}$.

1.2.1. Active-set methods

Since the number of variables will not be so large ($2m$ where m is the number of rows of the given diagram), active-set methods will be preferable to solve this least-squares problem. We refer the reader to [1] for details of active-set methods.

We first write the problem in a standard form. Let $m' = 2m$, $z_i = x_{i'}$ if $i = 2i'$ and $z_i = y_{i'}$ if $i = 2i' + 1$. Let $\left(A'\right)^T = \left(a_{ki}\right)_{k,i}$ be the $p \times m'$ matrix defined by $a_{ki} = 1$ if $2\sigma_k + 1 \le i \le 2\tau_k$ and $a_{ki} = 0$ otherwise. We set $d = \left(d_{i_k j_k}\right)_{1 \le k \le p}$ and $p' = p + m$. Let A^T be the $p' \times m'$-matrix $\begin{bmatrix} \left(A'\right)^T \\ I \end{bmatrix}$, and b be the p'-vector $\begin{bmatrix} d \\ 0 \end{bmatrix}$. Then the problem is formulated as follows:

$$
\begin{aligned}
\text{minimize} \quad & \frac{1}{2} z^T z \\
\text{subject to} \quad & A^T z \ge b
\end{aligned}
$$

An active-set method is an iterative method for solving minimization problems by guessing what the active set

$$
\mathcal{A}(z^*) = \left\{k \in \{1, \cdots, p'\} \;\middle|\; a_k z^* = b_k\right\}
$$

at the optimal point is. At each step, we are given a current point z and a subset $\mathcal{W} \subset \mathcal{A}(z)$ called a working set. The current point z is required to be a feasible point, that is, to satisfy $A^T z \ge b$. The working set $\mathcal{W}$ is required to satisfy that the vectors $\left\{a_k \;\middle|\; k \in \mathcal{W}\right\}$ are linearly independent.

We first find a direction ζ to which the objective function $\dfrac{1}{2}z^T z$ is decreasing. Such a ζ is found by solving the equality-constrained problem in ζ

$$\begin{aligned} &\text{minimize} \quad \frac{1}{2}\zeta^T\zeta + z^T\zeta \\ &\text{subject to} \quad A_{\mathcal{W}}^T\zeta = 0 \end{aligned}$$

where $A_{\mathcal{W}}^T$ denotes the submatrix of A^T consisting of the k-th rows for $k \in \mathcal{W}$. It is known that one can find the optimizer for such an equality-constrained problem by solving the KKT condition

$$\begin{bmatrix} I & -A_{\mathcal{W}} \\ A_{\mathcal{W}}^T & 0 \end{bmatrix}\begin{bmatrix} \zeta \\ \lambda \end{bmatrix} = \begin{bmatrix} -z \\ 0 \end{bmatrix}$$

in ζ and new variables $\lambda = (\lambda_k)_{k\in\mathcal{W}}$. This is equivalent to solving $\left(A_{\mathcal{W}}^T A_{\mathcal{W}}\right)\lambda = A_{\mathcal{W}}^T z$ first ($A_{\mathcal{W}}^T A_{\mathcal{W}}$ is positive definite since a_k's for $k \in \mathcal{W}$ are linearly independent) and then obtaining $\zeta = A_{\mathcal{W}}\lambda - z$.

Suppose that ζ is 0. Then we check if $\lambda_k \geq 0$ for all $k \in \mathcal{W}$. If so, then we are done, and z is the optimal solution. If not, choose $k \in \mathcal{W}$ minimizing λ_k, and proceed the next step with $z \leftarrow z$ and $\mathcal{W} \leftarrow \mathcal{W} \setminus \{k\}$. Of course, the new working set $\mathcal{W} \setminus \{k\}$ satisfies the linear-independence condition.

Suppose that ζ is not 0. In this case, we want to choose a step-length $\alpha \in [0,1]$ as large as possible keeping $z + \alpha\zeta$ feasible. We set

$$\alpha = \min\left(1, \min_{k\notin\mathcal{W}, a_k^T\zeta<0} \frac{b_k - a_k^T z}{a_k^T\zeta}\right)$$

The indices k minimizing α, if exist, are called the blocking constraints. We then proceed the next step with $z \leftarrow z + \alpha\zeta$ and $\mathcal{W} \leftarrow \mathcal{W} \cup \{k\}$ if k is the first blocking constraint and $\mathcal{W} \leftarrow \mathcal{W}$ if there are no blocking constraints. For a blocking constraint k, the new working set $\mathcal{W} \cup \{k\}$ satisfies the linear-independence condition because $a_k^T\zeta < 0$ while $a_{k'}^T\zeta = 0$ for $k' \in \mathcal{W}$.

1.2.2. Updating decomposition

The most expensive part in the active-set method is to solve the equation

$$\left(A_{\mathcal{W}}^T A_{\mathcal{W}}\right)\lambda = A_{\mathcal{W}}^T z$$

This equation can be solved using a QR decomposition of $A_{\mathcal{W}}$

$$A_{\mathcal{W}} = Q_{\mathcal{W}}\begin{bmatrix} R_{\mathcal{W}} \\ 0 \end{bmatrix}$$

where $Q_{\mathcal{W}}$ is an orthogonal matrix and $R_{\mathcal{W}}$ is an upper triangular matrix. Indeed, we have

$$A_{\mathcal{W}}^T A_{\mathcal{W}} = \begin{bmatrix} R_{\mathcal{W}}^T & 0 \end{bmatrix} Q_{\mathcal{W}}^T Q_{\mathcal{W}} \begin{bmatrix} R_{\mathcal{W}} \\ 0 \end{bmatrix} = R_{\mathcal{W}}^T R_{\mathcal{W}}$$

and then

$$\left(R_{\mathcal{W}}^T R_{\mathcal{W}}\right)\lambda = A_{\mathcal{W}}^T z$$

is easily solved because $R_{\mathcal{W}}$ is an upper triangular matrix.

To calculate a QR decomposition $\left(Q_{\mathcal{W}}, R_{\mathcal{W}}\right)$ efficiently, we note that the matrix $A_{\mathcal{W}}$ at a step only differs in one column from the previous step: one column is inserted or deleted. In either case, updating a QR decomposition is more efficient than recalculating. We refer the reader to [2] for details.

Let $\mathcal{W}'$ be the working set in the previous step and suppose that the current working set is $\mathcal{W} = \mathcal{W}' \setminus \{k\}$. Let R be the matrix obtained from $R_{\mathcal{W}'}$ by removing the k-th column. We have $A_{\mathcal{W}} = Q_{\mathcal{W}'}\begin{bmatrix} R \\ 0 \end{bmatrix}$, but R need not be upper triangular. Let $\left(Q, R_{\mathcal{W}}\right)$ be a QR decomposition of R and let $Q_{\mathcal{W}} = Q_{\mathcal{W}'}\operatorname{diag}(Q, I)$. Then $\left(Q_{\mathcal{W}}, R_{\mathcal{W}}\right)$ is a QR decomposition of $A_{\mathcal{W}}$. For a QR decomposition of R, we note that R is almost upper triangular: it is of the form

$$
\begin{bmatrix}
* & \cdots & * & \cdots & * & \cdots \\
\vdots & \ddots & \vdots & & \vdots & \\
0 & \cdots & * & * & * & \\
0 & \cdots & 0 & * & * & \\
0 & \cdots & 0 & * & * & \\
0 & \cdots & 0 & 0 & * & \cdots \\
\vdots & & & & \vdots & \ddots
\end{bmatrix}
$$

Then we obtain a QR decomposition by repeated plain rotation.

Suppose that the current working set is $\mathcal{W} = \mathcal{W}' \cup \{k\}$. We set $A_{\mathcal{W}} = \begin{bmatrix} A_{\mathcal{W}'} & a_k \end{bmatrix}$. Split $Q_{\mathcal{W}'} = \begin{bmatrix} Q_1 & Q_2 \end{bmatrix}$ with Q_1 the first $\mathcal{W}'$ columns. Find a Householder transformation H and a scalar ρ such that $H\left(Q_2^T a_k\right) = \rho \mathbf{e}_1$. Then $R_{\mathcal{W}} = \begin{bmatrix} R_{\mathcal{W}'} & Q_1^T a_k \\ 0 & \rho \end{bmatrix}$ and $Q_{\mathcal{W}} = Q_{\mathcal{W}'} \operatorname{diag}(I, H)$. A Householder transformation is a matrix of the form $H = I - uu^T$ such that $\|u\|_2 = \sqrt{2}$. For a given vector x, one can find a Householder transformation H such that $Hx = \|x\|_2 \mathbf{e}_1$ or $Hx = -\|x\|_2 \mathbf{e}_1$.

[1] Jorge Nocedal and Stephen J. Wright, "Numerical Optimization," ser. Springer Series in Operations Research and Financial Engineering. Springer, New York, NY, 2006.

[2] G. W. Stewart, "Matrix Algorithms: Volume 1: Basic Decompositions," Philadelphia, PA: Society for Industrial and Applied Mathematics (SIAM), 1998.