SSCNO:
A Fortran 90 Library for
Separable Strictly Convex
Network Optimization

Aleksandar Donev, Dr. Phillip Duxbury
Physics and Astronomy Department
Michigan State University

July 31, 2001
Convex separable network optimization:

Energy functional:

\[
\arg \min_x \left\{ f(x) = \sum_{e(i,j)} f(i,j) \left[ x(i,j) \right] \right\} \quad (1)
\]

Flow conservation:

\[
Ax = b \Rightarrow \sum_{e(i,j)} x(i,j) - \sum_{e(j,i)} x(j,i) = b_i \quad (2)
\]
x arc flows (primal variables)

λ nodal Lagrange multipliers (dual variables)

A node-arc incidence matrix of a graph G (constraint matrix)

f total cost function—separable and convex (strictly-convex and twice continuously differentiable)

\[ f_{(i,j)} \text{ arc elemental costs, with} \]
\[ g_{(i,j)}(x_{(i,j)}) = f'_{(i,j)}(x_{(i,j)}) \text{ and} \]
\[ H_{(i,j)}(x_{(i,j)}) = f''_{(i,j)}(x_{(i,j)}) \]

b nodal supplies-demands

t arc tensions, \( t_{(i,j)} = \lambda_i - \lambda_j \)
Why is this kind of problem central to materials science?

Many transport phenomena in disordered systems can be modeled as network optimization problems:

- **Current flow** in non-linear resistive circuits:
  - dissipation in granular superconducting materials
  - breakdown in varistors and dielectrics

- **Fluid flow** in channel networks:
  - flow through porous media
  - river formation (sometimes non-convex)
In these systems, the elemental cost functions are related to the **tension–flow characteristics** of the arcs:

\[ t = t(x), \text{ or equivalently } x = x(t) = t^{-1}(t) \]

\[ f(x) = \int_{\zeta=0}^{x} t(\zeta) d\zeta \quad (3) \]

- **voltage–current**: \( t(x) \equiv V(I) \)

- **pressure–flow**: \( t(x) \equiv P(Q) \)

\( V(I), P(Q) \) are monotonically increasing \( \Rightarrow \) the elemental cost functions are convex
Physical Review B, 2000:
In a network of linear resistors, this [finding the flow] reduces to solving a set of linear [Kirchoff’s] equations, for which many standard methods exist. For non-linear resistors, we are not aware of any general method...Due to its robustness and applicability to highly disordered systems with many variables we choose a simulated annealing algorithm to perform the minimization.

Why do physicists use simulated annealing to solve a linearly-constrained fully-separable strictly convex optimization problem?

- Strict-convexity goes unnoticed, or not-familiar with complexity theorems
- Lack of available public-domain software
- Lack of communication between computational science disciplines and applied mathematicians
Analogies between physics and applied mathematics:

<table>
<thead>
<tr>
<th>Physics</th>
<th>Applied math</th>
</tr>
</thead>
<tbody>
<tr>
<td>$I$</td>
<td>$x$</td>
</tr>
<tr>
<td>$U$</td>
<td>$\lambda$</td>
</tr>
<tr>
<td>$V$</td>
<td>$t$</td>
</tr>
<tr>
<td>$P$</td>
<td>$f$</td>
</tr>
<tr>
<td>$R$</td>
<td>$H$</td>
</tr>
<tr>
<td>$C = \frac{1}{R}$</td>
<td>$H^*$</td>
</tr>
</tbody>
</table>

Variables:

- Current $I$
- Potential $U$
- Voltage $V$
- Power $P$
- Resistance $R$
- Conductance $C = \frac{1}{R}$
- Flow $x$
- Multiplier $\lambda$
- Tension (gradient) $t$
- Cost $f$
- Hessian $H$
- Conjugate Hessian $H^*$

Optimality conditions:

- $\text{KKT}_1 : Ax = b$  
  First Kirchoff’s law
- $\text{KKT}_2 : t = A^T \lambda = \nabla f(x)$  
  Second Kirchoff’s law
Newton-based Algorithms in SSCN0

1. (Truncated) **Dual Newton Algorithm (TDN)**

   **Potential-based:** Solves the unconstrained dual problem using the truncated Newton algorithm

2. (Truncated) **Sequential Quadratic Programming Algorithm (TSQP)**

   **Flow-based:** Solves directly the primal problem by making successive quadratic approximations to the Lagrangian (cost function)
A physicist’s summary of TDN (and something for the rest of you):

1. Make initial guess for the node potentials $\lambda$

2. Calculate the arc voltages from the potentials:

   $$t = A^T \lambda$$

3. Use the voltage-current characteristics of the arcs to find the currents through the arcs from the voltages:

   $$x = x(t)$$

Or as mathematicians would say, form the conjugate convex function:

$$x^* = \arg \sup_x \left[ x^T t - f(x) \right] = g^{(-1)}(t)$$
Note: TSQP requires calculating the voltages from the currents:

\[ t = t(x) \]

This step uses the user-supplied elemental functions!

4. Find the excess currents at the nodes from the currents through the arcs:

\[ \Delta b = Ax - b \]

Or as mathematicians would say, find the gradient of the dual Lagrangian:

\[ \nabla q(\lambda) = Ax^* - b \]

5. If the excess flow is small enough, stop and return \( x \) and \( \lambda \)
6. Solve a random-resistor linear network using \(-\Delta b\) as a supply-demand and the conductances of the arcs at the current flows (voltages) to find the excess potentials at the nodes \(\Delta \lambda\), *possibly inexactl*y

Or mathematically, solve the Newton linear system:

\[
\begin{bmatrix}
AH^*A^T
\end{bmatrix} \Delta \lambda = -(Ax^* - b)
\]

where

\[
\mathcal{H} = \nabla^2 q(\lambda) = AH^*A^T
\]

is the conjugate Hessian, which has the structure of a generalized Laplacian (*conductance matrix*), and \(H^* = [H(x^*)]^{-1}\) are the conductances (inverse resistances) of the arcs.

This step is shared by TDN and TSQP and is the most time-consuming.
7. Update the potentials taking the largest step along $\Delta \lambda$ possible.

That is, perform a line search along $\Delta \lambda$:

$$ \alpha = \arg \min_q [\lambda + \alpha \Delta \lambda] $$

and update $\lambda \leftarrow \lambda + \alpha \Delta \lambda$

8. Go back to step 2
Details: Linear Solver

*Generalized Laplacians* are common to many disciplines. Our graphs are *very large, very sparse, near-neighbour grid lattices in 2 and 3D*, for which we implemented two methods:

**Cholesky Factorization** using the public-domain TAUCS factorization library and the SCOTCH graph partitioning and reordering library

**Preconditioned Conjugate Gradient** (allows truncation) with novel *support-graph preconditioners* based on maximal-weight spanning trees and graph partitioning.

Reference: *Support-Graph Preconditioners*, M. Bern, J. Gilbert, B. Hendrickson, N. Nguyen and S. Toledo

This is a promising field of research!
Details: Elemental Cost Functions

All of the physics is in the tension-flow arc characteristics. The ones interesting to physics are peculiar:

**Very ill-conditioned** Conductances vary over several orders of magnitude, with sharp transitions in the tension or flow behaviour.

**Non-smooth or not-strictly convex** We do not know the exact physical form of these characteristics.

Our approximations are usually piecewise-continuous, non-differentiable and not strictly monotonically increasing.

**Help needed in regularization approaches**—proximal point and smoothing algorithms
Examples of Potential-Flow Characteristics

1. Power-law costs (plastic fluids, river formation...)

\[ f_i(x_i) = R_i |x_i|^\alpha \]

By changing \( \alpha \) we cover a whole range of behaviour:

- \( \alpha > 1 \) strictly convex (flow has cycles)
- \( 0 < \alpha \leq 1 \) concave (spanning tree-like flows)
2. Cost functions with thresholds (superconductors, varistors, plastics, oil-recovery...)

These exhibit **macroscopic flow localization**

- **Critical flow** $x^0$:

  $$t_i(x_i) = R_i |x_i| \left[ 1 + \tanh \left( \frac{|x_i| - x_i^{(0)}}{\xi} \right) \right]$$

  Related to Max-Flow (Min-Cut): *Flow saturation (voltage drop onset) occurs on minimal cuts*

- **Critical tension** $t^0$:

  $$x_i(t_i) = R_i |t_i| \left[ 1 + \tanh \left( \frac{|t_i| - t_i^{(0)}}{\xi} \right) \right]$$

  Related to Min-Path: *Flow onset (voltage saturation) occurs on paths*
Progress and Future Directions:

1. **Algorithm:**

   Smooth strictly convex optimization is simple and applicable:

   • Improve linear solvers in 3D
   • Develop regularization smoothing and proximal techniques

2. **Implementation:**

   • Code written in *Fortran 90*, with interfaces to many public-domain libraries (CHACO, SCOTCH, TAUCS, DISLIN, HilbertSFC, etc.)
   • **Parallellization** is possible and necessary—billions of nodes needed for real applications (polish algorithm first!)
Something for everyone:

- **Applied mathematicians** — regularization and better truncation algorithms, and linear solvers

- **Computer scientists** — parallel implementation and more public-domain libraries

- **Physicists** — testing and feedback

Visit
http://computation.pa.msu.edu/NO/
for details and codes!