When “Network Flow” Meets “Convex Optimization”

@luk036

2021-11-10

当“网络流”遇上“凸优化”

@luk036

2021-11-10

“Network Flow” says:

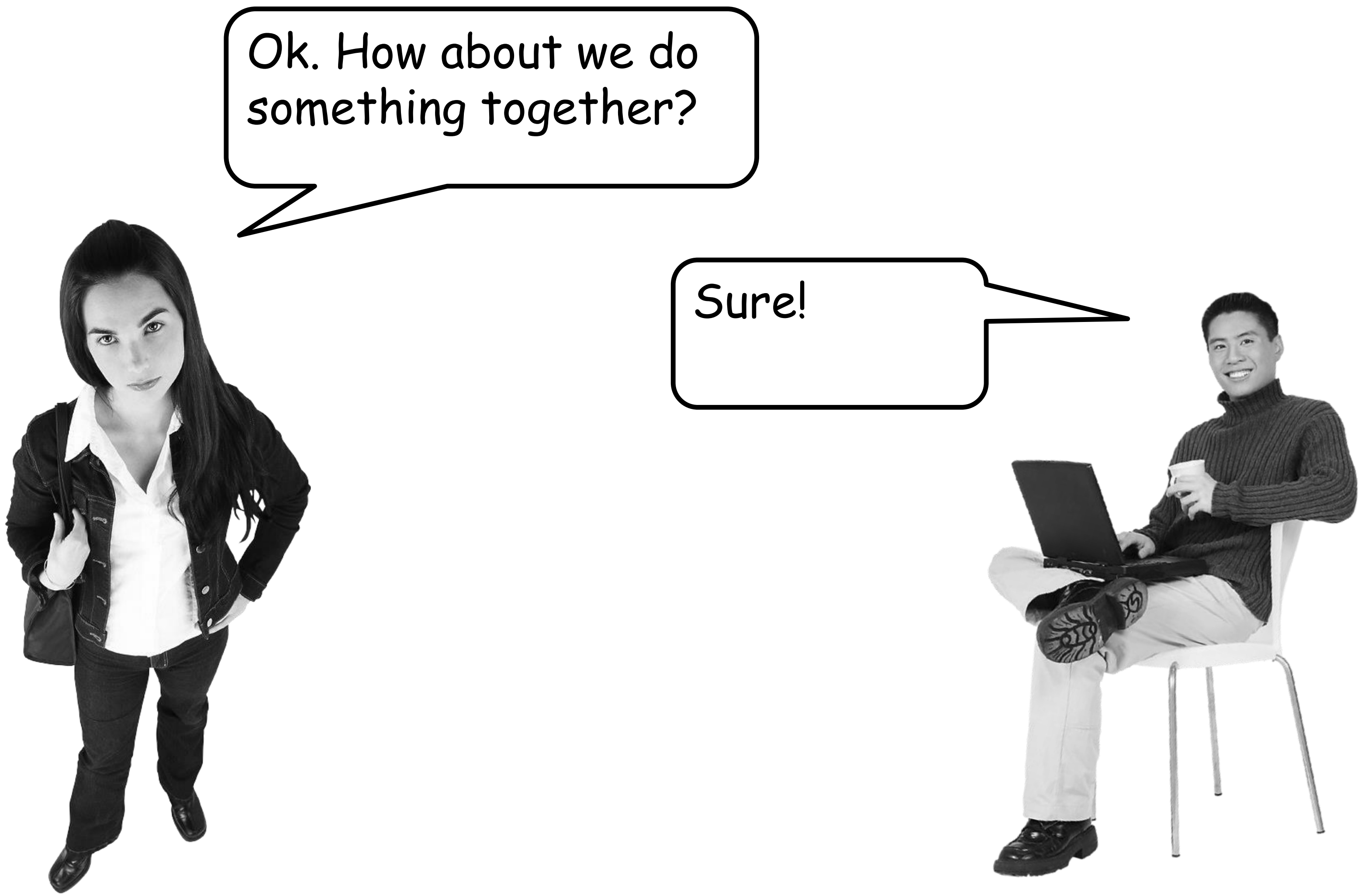
$$
\begin{array}{ll}
\text{minimize} & f(x) \\
\text{subject to} & c^{-} \leq x \leq c^{+} \\
 & A^{\mathsf{T}} x = b,\ b(V) = 0
\end{array}
$$

“Convex Optimization” says:

Well, maybe I can do something to it if f(x) is convex or quasi-convex.

minimize	$f(x)$
subject to	$c^{-} \leq x \leq c^{+}$
	$A^{\top} x = b,\ b(V) = 0$

Network Flow + Convex Optimization



PARAMETRIC POTENTIAL PROBLEM

Parametric potential problems

Consider:

$$\begin{aligned} \max \quad & g(\beta), \\ \text{s.t.} \quad & y \leq d(\beta), \\ & Au = y, \end{aligned}$$

where $g(\beta)$ and $d(\beta)$ are concave.

Note: the parametric flow problems can be defined in a similar way.

Network flow says:

- For fixed β, the problem is feasible precisely when there exists no negative cycle.
- Negative cycle detection can be done efficiently using the Bellman-Ford-like methods
- If a negative cycle C is found, then $\sum_{(i,j)\in C} d_{ij}\,(\beta) < 0$

Convex Optimization says:

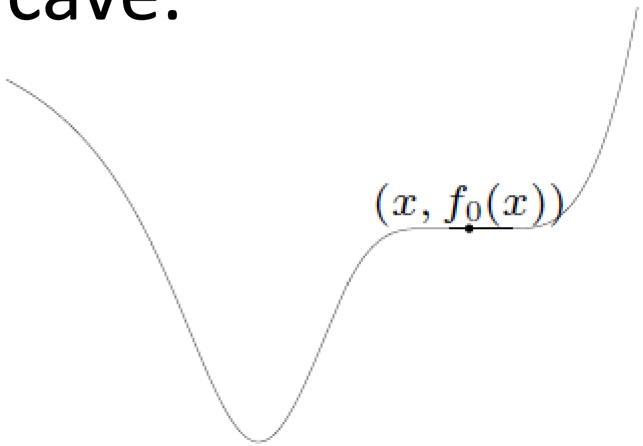
- If both sub-gradients of $g(\beta)$ and $d(\beta)$ are known, then the *bisection method* can be used for solving the problem efficiently.
- Also, for multi-parameter problems, the *ellipsoid method* can be used.

Quasi-convex Minimization

Consider:

$$\begin{aligned} \max \quad & f(\beta), \\ \text{s.t.} \quad & y \le d(\beta), \\ & Au = y, \end{aligned}$$

where $f(\beta)$ is *quasi-convex* and $d(\beta)$ are concave.



Examples of Quasi-Convex Functions

- $\sqrt{|y|}$ is quasi-convex on $\mathbb{R}$
- $\log(y)$ is quasi-linear on $\mathbb{R}_{++}$
- $f(y_1, y_2) = y_1 y_2$ is quasi-concave on $\mathbb{R}_{++}^2$
- Linear-fractional function:
  - $f(x) = (a^{\mathrm{T}}x + b)/(c^{\mathrm{T}}x + d)$
  - dom $f = \{x \mid c^{\mathrm{T}}x + d > 0\}$
- Distance ratio function:
  - $f(x) = \| x - a \|_2 / \| x - b \|_2$
  - dom $f = \{x \mid \| x - a \|_2 \leq \| x - b \|_2\}$

Convex Optimization says:

- If f is quasi-convex, there exists a family of functions ϕ_t such that:
  - $\phi_t(\beta)$ is convex w.r.t. β
  - $\phi_t(\beta)$ is non-increasing w.r.t. t
  - t-sublevel set of f is 0-sublevel set of ϕ_t, i.e.

$$f(\beta) \leq t \Leftrightarrow \phi_t(\beta) \leq 0$$

Convex Optimization says:

- For example:

  $f(\beta) = p(\beta)/q(\beta)$

  with p convex, q concave,

  $p(\beta) \geq 0$, $q(\beta) > 0$ on dom f,

  can take $\phi_t(\beta) = p(\beta) - t \cdot q(\beta)$

Convex Optimization says:

Consider a feasibility problem:

$$
\begin{aligned}
\text{find} \quad & \beta, \\
\text{s.t.} \quad & \phi_t(\beta) \leq 0, \\
& y \leq d(\beta), Au = y,
\end{aligned}
$$

- If feasible, then $t \geq p^*$;
- If infeasible, then $t < p^*$.
- Binary search on t can be used for obtaining p^*.

Quasi-convex Network Problem

- Again, the feasibility problem can be solved efficiently by the bisection method or the ellipsoid method, together with the negative cycle detection technique.

- Q. Any EDA's applications ???

Monotonic Minimization

- Consider the following problem:

$$\begin{aligned} \min \quad & \max_{ij} f_{ij}(y_{ij}), \\ \text{s.t.} \quad & Au = y, \end{aligned}$$

where $f_{ij}(y_{ij})$ is non-decreasing.

- The problem can be recast as:

$$\begin{aligned} \max \quad & \beta, \\ \text{s.t.} \quad & y \leq f_{ij}^{-1}(\beta), \\ & Au = y, \end{aligned}$$

where $f_{ij}^{-1}(\beta)$ is non-deceasing w.r.t. β.

E.g. Yield-driven Optimization

- Consider the following problem:

$$\begin{aligned} \max \quad & \min_{ij} \Pr(y_{ij} \leq \tilde{d}_{ij}) \\ \text{s.t.} \quad & Au = y, \end{aligned}$$

where $\tilde{d}_{ij}$ is a random variables.

- Equivalent to the problem:

$$\begin{aligned} \max \quad & \beta, \\ \text{s.t.} \quad & y \leq \Pr(y_{ij} \leq \tilde{d}_{ij}), \\ & Au = y, \end{aligned}$$

where $f_{ij}^{-1}(\beta)$ is non-deceasing w.r.t. β.

E.g. Yield-driven Optimization

- Let $F_{ij}(x)$ is the cdf of $\boldsymbol{d}_{ij}$.
- Then:

$$\beta \leq \Pr(y_{ij} \leq \boldsymbol{d}_{ij})$$

$$=> \beta \leq 1 - F_{ij}(y_{ij})$$

$$=> y_{ij} \leq F_{ij}^{-1}(1 - \beta)$$

- The problem becomes:

maximum β

subject to $y_{ij} \leq F_{ij}^{-1}(1 - \beta)$,

$A\,u = y$

E.g. Yield-driven Optimization

- If $\boldsymbol{d}_{ij}$ is a Gaussian random variable with mean d_{ij} and variance s_{ij}.
- Then the problem further reduces to:

maximum	β
subject to	$y \leq d - s\,\beta$,
	$A\,u = y$

Network flow says:

- Monotonic problem can be solved efficiently using cycle-cancelling methods such as Howard's algorithm.

MIN-COST FLOW PROBLEM

Min-Cost Flow Problem (linear)

- Consider:

$$\begin{aligned} \min \quad & d^{\mathrm{T}} x, \\ \text{s.t.} \quad & c^- \le x \le c^+, \\ & A^{\mathrm{T}} x = b, b(V) = 0 \end{aligned}$$

- Some c^+ could be $+\infty$, some c^- could be $-\infty$
- A^{T} is the incidence matrix of a network G.

- Conventional (integration):

$$\int_{S} d\,\tilde{\omega} = \oint_{\partial S} \tilde{\omega}$$

- Discrete (pairing):

$$[\tau, A\omega] = [A^{T}\tau, \omega]$$

- where

$$\tau_i = \begin{cases} 1 & \text{if } e_i \in S\,, \\ 0 & \text{otherwise}\,. \end{cases}$$

Conventional Algorithms

- Augmented path based
  - Start from an infeasible solution
  - Inject minimal flow into the augmented path while maintaining infeasibility in each iteration
  - Stop when there is no flow to inject into the path
- Cycle cancelling based
  - Start from a feasible solution x_0
  - find a better sol'n $x_1 = x_0 + \alpha\Delta x$, where α is positive.

General Descent Method

1. **Input**: a starting $x \in \mathrm{dom}\, f$
2. **Output**: x^*
3. **repeat**
    1. Determine a descent direction Δx.
    2. Line search. Choose a step size $\alpha > 0$.
    3. Update. $x := x + \alpha\, \Delta x$
4. **until** a stopping criterion is satisfied;

Some Common Descent Directions

- For convex problems, the search direction must satisfy $\nabla f(x)^T \Delta x < 0$.
- Gradient descent:
  - $\Delta x = -\nabla f(x)^T$
- Steepest descent:
  - Δx_{nsd} = argmin{ $\nabla f(x)^T v \mid \|v\| = 1$ }.
  - $\Delta x_{sd} = \|\nabla f(x)\| \Delta x_{nsd}$ (un-normalized)
- Newton's method:
  - $\Delta x = -\nabla^2 f(x)^{-1} \nabla f(x)$

Network flow says:

- Here, there is a better way to choose Δx!
- Let $x_1 = x_0 + \alpha\Delta x$, then we have:

min $d^T x_0 + \alpha d^T \Delta x$ => $d^T \Delta x < 0$

s. t. $c^- - x_0 \leq \alpha\Delta x \leq c^+ - x_0$ => residual graph G_x

$A^T \Delta x = 0$ => Δx is a cycle!

- In other words, choose Δx to be a negative cycle with cost d!
  - Simple negative cycle, or
  - Minimum mean cycle

Network flow says:

- Step size is limited by the capacity constraints:
  - $\alpha_1 = \min_{ij} \{c^+ - x_0\}$, for $\Delta x_{ij} > 0$
  - $\alpha_2 = \min_{ij} \{x_0 - c^-\}$, for $\Delta x_{ij} < 0$
  - $\alpha_{lin} = \min\{\alpha_1, \alpha_2\}$
- If $\alpha_{lin} = +\infty$, the problem is unbounded.

Network flow says:

- An initial feasible solution can be obtained by a similar construction of residual graph and cost vector.
- The LEMON package implements this cycle cancelling algorithm.

Min-Cost Flow Convex Problem

- Problem Formulation:

$$\min \quad f(x) \quad \text{<----- convex}$$

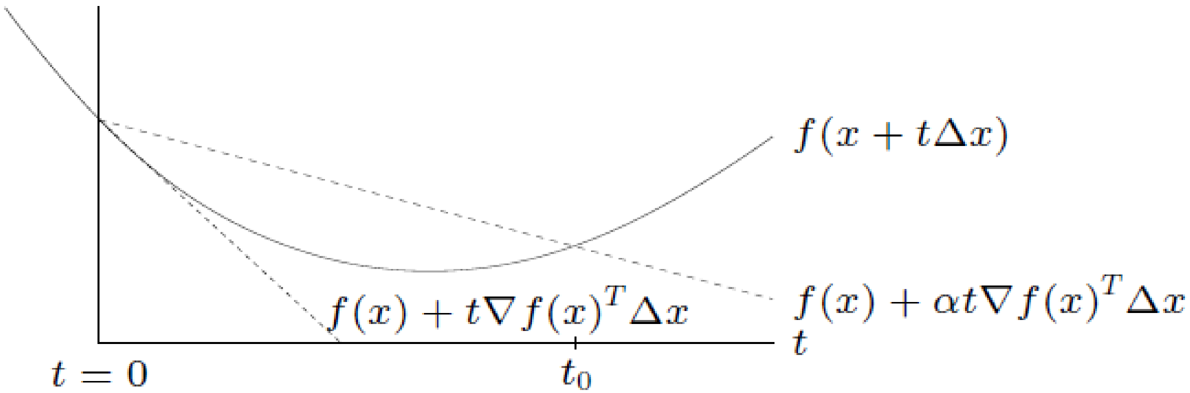
$$\text{s. t.} \quad 0 \leq x \leq c$$

$$A^{T} x = b,\ b(V) = 0$$

Common Line Search Types

- Exact line search: $t = \mathrm{argmin}_{t>0} f(x + t\Delta x)$
- Backtracking line search (with parameters $\alpha \in (0,1/2)$, $\beta \in (0,1)$)
  - starting at $t = 1$, repeat $t := \beta t$ until

    $f(x + t\Delta x) < f(x) + \alpha t \nabla f(x)^T \Delta x$
  - graphical interpretation: backtrack until $t \leq t_0$



Network flow says:

- The Step size is further limited by:
  - $\alpha_{cvx} = \min\{\alpha_{lin}, t\}$
- In each iteration, choose Δx as a negative cycle of G_x, with cost $\nabla f(x)$ such that $\nabla f(x)^T \Delta x < 0$

Quasi-convex Minimization

- Problem Formulation:

$$\begin{aligned} \min \quad & f(x) \quad \text{<---- quasi-convex} \\ \text{s. t.} \quad & 0 \le x \le c \\ & A^{\mathrm{T}} x = b,\ b(V) = 0 \end{aligned}$$

- The problem can be recast as:

$$\begin{aligned} \min \quad & t \\ \text{s. t.} \quad & f(x) \le t, \\ & 0 \le x \le c \\ & A^{\mathrm{T}} x = b,\ b(V) = 0 \end{aligned}$$

Convex Optimization says:

- Consider a convex feasibility problem:

$$\begin{aligned} \text{find} \quad & x \\ \text{s. t.} \quad & \phi_t(x) \le 0, \qquad\qquad (2) \\ & 0 \le x \le c \\ & A^\top x = b,\ b(V) = 0 \end{aligned}$$

  – if feasible, we can conclude that $t \ge p^*$;
  – if infeasible, $t \le p^*$
- Binary search on t can be used for obtaining p^*.

Network flow says:

- Choose Δx to be a negative cycle of G_x with cost $\nabla \phi_t(x)$
- If no negative cycle is found, and $\phi_t(x) > 0$, we conclude that problem (2) is infeasible.
- Iterate until x becomes feasible, i.e. $\phi_t(x) \leq 0$.

E.g. Linear-Fractional Cost

- Problem Formulation:

$$\begin{aligned} \min \quad & (e^{T}x + f) / (g^{T}x + h) \\ \text{s. t.} \quad & 0 \leq x \leq c \\ & A^{T} x = b,\ b(V) = 0 \end{aligned}$$

- The problem can be recast as:

$$\begin{aligned} \min \quad & t \\ \text{s. t.} \quad & (e^{T}x + f) - t(g^{T}x + h) \leq 0 \\ & 0 \leq x \leq c \\ & A^{T} x = b,\ b(V) = 0 \end{aligned}$$

Convex Optimization says:

- Consider a convex feasibility problem:

find $\quad x$

s. t. $\quad (e - t*g)^{\mathsf{T}}x + (f - t*h) \leq 0,$

$0 \leq x \leq c$

$A^{\mathsf{T}} x = b, \; b(V) = 0$

  – if feasible, we conclude that $t \geq p^*$;

  – if infeasible, $t \leq p^*$

- Binary search on t can be used for obtaining p^*.

Network flow says:

- Choose Δx to be a negative cycle of G_x, with cost $(e - t{*}g)$, i.e. $(e - t{*}g)^{\mathrm{T}}\Delta x < 0$
- If no negative cycle is found, and $(e - t{*}g)^{\mathrm{T}}x_0 + (f - t{*}h) > 0$, we conclude that the problem is infeasible.
- Iterate until $(e - t{*}g)^{\mathrm{T}}x_0 + (f - t{*}h) \leq 0$.

E.g. Statistical Optimization

- Consider the problem:

  min $\Pr(\mathbf{d}^T x > \alpha)$ <--- quasi-convex

  s. t. $0 \leq x \leq c$

  $A^T x = b, b(V) = 0$

  - $\mathbf{d}$ is random vector with mean d and covariance Σ.
  - hence, $\mathbf{d}^T x$ is a random variable with mean $d^T x$ and variance $x^T \Sigma x$.

Statistical Optimization

- The problem can be recast as:

$$\begin{aligned} \min \quad & t \\ \text{s. t.} \quad & \Pr(\mathbf{d}^T x > \alpha) \le t \\ & 0 \le x \le c \\ & A^T x = b,\ b(V) = 0 \end{aligned}$$

- Note:

$$\Pr(\mathbf{d}^T x > \alpha) \le t$$

$$\Rightarrow d^T x + F^{-1}(1-t) \|\Sigma^{1/2} x\|_2 \le \alpha$$

(convex quadratic constraint w.r.t x)

Recall that the gradient of
$d^T x + F^{-1}(1-t)||\Sigma^{½} x||_2$ is
$d + F^{-1}(1-t)(||\Sigma^{½} x||_2)^{-1}\Sigma x$

Problem w/ additional Constraints

- Problem Formulation:

$$\min \quad f(x)$$

$$\text{s. t.} \quad 0 \leq x \leq c$$

$$A^{\mathrm{T}} x = b,\ b(V) = 0$$

$$s^{\mathrm{T}} x \leq \gamma$$ <----- added

E.g. Yield-driven Delay Padding

- Consider the following problem:

$$\begin{array}{ll} \text{maximum} & \gamma\,\beta - c^{T}p \\ \text{subject to} & \beta \le \Pr(y_{ij} \le \mathbf{d}_{ij} + p_{ij}), \\ & A\,u = y,\ p \ge 0 \end{array}$$

- p: delay padding
- γ: weight (determined by a trade-off curve of yield and buffer cost)
- $\mathbf{d}_{ij}$: Gaussian random variable with mean d_{ij} and variance s_{ij}.

E.g. Yield-driven Delay Padding

- The problem is equivalent to:

  maximum $\gamma\,\beta - c^{\mathsf{T}}p$

  subject to $y \le d - \beta\, s + p,$

  $A\,u = y,\ p \ge 0$

- or its dual:

  minimize $d^{\mathsf{T}}x$

  subject to $0 \le x \le c$

  $A^{\mathsf{T}}x = 0,$

  $s^{\mathsf{T}}x \le \gamma$

Considering Barrier Method

- Approximation via logarithmic barrier:

$$\min \quad f(x) + (1/t)\,\phi(x)$$
$$\text{s. t.} \quad 0 \le x \le c$$
$$A^{\mathsf{T}} x = b,\ b(V) = 0$$

- where $\phi(x) = -\log\,(\gamma - s^{\mathsf{T}} x)$
- Approximation improves as $t \to \infty$
- Here, $\nabla\phi(x) = s \,/\, (\gamma - s^{\mathsf{T}} x)$

Barrier Method

Input: a feasible x, $t := t^{(0)}$, $\mu > 1$, tolerance ǫ > 0

Output: x^*

repeat

1. Centering step. Compute $x^*(t)$ by minimizing $t\,f + \phi$
2. Update. $x := x^*(t)$.
3. Increase t. $t := \mu t$.

until $1/t <$ ǫ;

- ☞ Note: Centering is usually done by Newton's method in general.

$$
\begin{array}{lll}
\min & d^T x_0 + \alpha d^T p & \Rightarrow d^T < 0 \\
\text{s.t.} & -x_0 \leq \alpha p \leq c - x_0 & \Rightarrow \text{residual graph} \\
 & A^T p = 0 & \Rightarrow p \text{ is a cycle!}
\end{array}
$$

Network flow says:

- In the centering step, instead of using the Newton descent direction, we can replace it with a negative cycle on the residual graph.

Not yet finish.
Any good
suggestion?