Distributed convex optimization: a consensus-based Newton-Raphson approach

Damiano Varagnolo

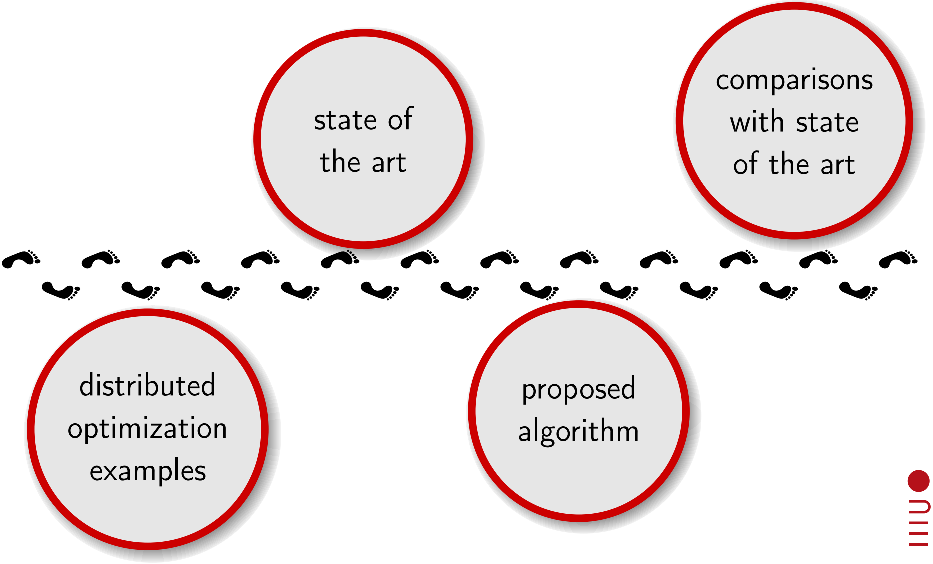
joint work with A. Cenedese, G. Pillonetto, L. Schenato, F. Zanella

Department of Information Engineering - University of Padova

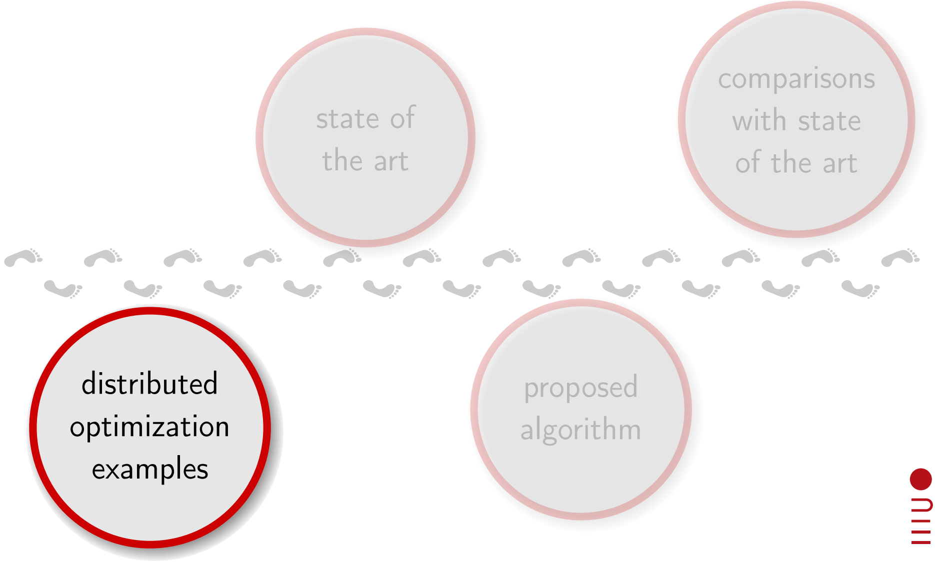
July 5th, 2011

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

This talk


state of the art
comparisons with state of the art
distributed optimization examples
proposed algorithm

This talk


distributed optimization examples
state of the art
proposed algorithm
comparisons with state of the art

Distributed optimization and its importance

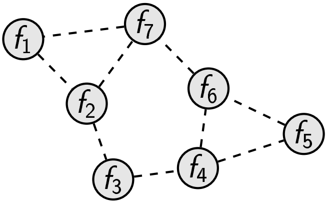
Problem formulation

$$\begin{aligned} &\text{minimize} && f(x) = \textstyle\sum_{i=1}^{N} f_i(x) \\ &\text{subject to} && g(x) \leq 0 \\ & && x \in \mathcal{X} \end{aligned}$$

under ***convexity assumptions***

Multi-agents scenario

Networked system where neighbors cooperate to find the optimum



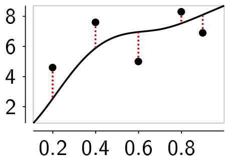
Distribution optimization - Example 1

Regression in sensor networks

(e.g. when estimation = optimization of a cost function)

Residuals minimization

$$\begin{array}{ll} \min_\theta & \sum_{i=1}^N \phi(y_i - \hat{y}_i) \\ \text{s.t.} & \hat{y}_i = \theta^T x_i \end{array}$$

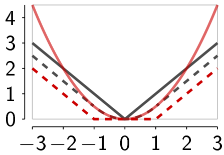


$\phi(r) = |r|^2$ (least squares)

$\phi(r) = |r|$ (least abs. deviations)

$$\phi(r) = \begin{cases} 0 & \text{if } |r| < 1 \\ |r| - 1 & \text{otherwise} \end{cases}$$ (Vapnik)

$$\phi(r) = \begin{cases} |r|^2 & \text{if } |r| < 1 \\ 2(|r| - 1) & \text{otherwise} \end{cases}$$ (Huber)



Distribution optimization - Example 2

Classification in sensor networks

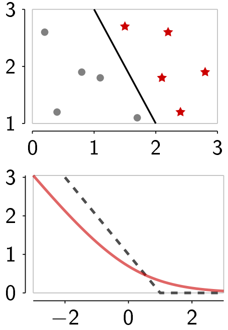
(e.g. when classification = optimization of a cost function)

Support Vector Machine Classification

$$\min_{w,w_0} \sum_{i=1}^{N} \left[1 - y_i \left(w x_i + w_0\right)\right]_+ + \lambda \left\|w\right\|^2$$

$\downarrow$ (smooth approximation)

$$\min_{w,w_0} \sum_{i=1}^{N} \log \left[1 + e^{-y_i (w x_i + w_0)}\right] + \lambda \left\|w\right\|^2$$



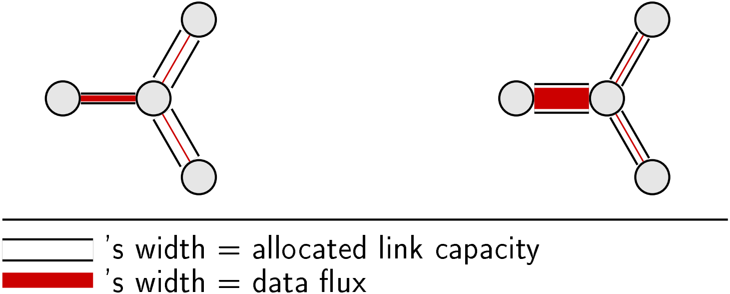
Distribution optimization - Example 3

Resource allocation in wireless systems

(e.g. when optimal allocation = optimization of a cost function)

Links capacity allocation [Johansson 2008]

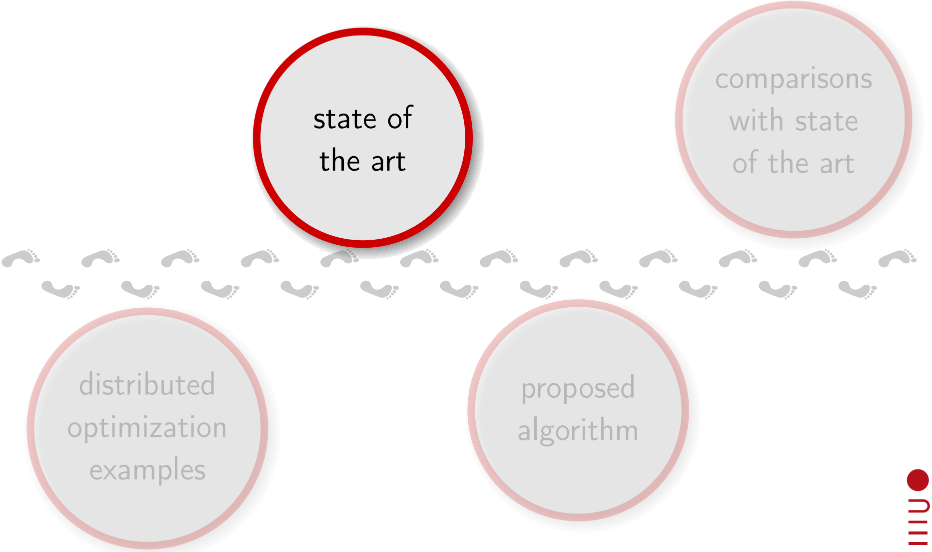
suboptimal allocation | optimal allocation



's width = allocated link capacity

's width = data flux

Update


state of the art
comparisons with state of the art
distributed optimization examples
proposed algorithm

State of the art

Distributed optimization methods: 3 main categories

- Primal decompositions methods
  (e.g. distributed subgradients)
- Dual decompositions methods
  (e.g. alternating direction method of multipliers)
- Heuristic methods
  (e.g. swarm optimization, genetic algorithms)

Primal decomposition methods (centralized)

Subgradient methods [Kiwiel 2004]

$$x_{k+1} = \mathcal{P}_{\mathcal{X}} \left[x_k + \alpha_k \cdot g\left(x_k\right) \right]$$

with

- $g\left(x_k\right) :=$ subgradient of $f(\cdot)$ at x_k
- $\alpha_k :=$ stepsize
- $\mathcal{P}_{\mathcal{X}}\left[\cdot\right] :=$ projector on $\mathcal{X}$

Convergence properties

- α_k typically needs to be diminishing for non-smooth f
- $g(\cdot)$ may be required to be bounded
- ...

Primal decomposition methods (distributed)

Distributed subgradient methods [Nedić Ozdaglar 2009]

$$x_i(k+1) = \mathcal{P}_{\mathcal{X}} \left[\sum_{j=1}^{N} a_{ij}(k) x_j(k) + \alpha_i(k) g_i\big(x_i(k)\big) \right]$$

with

- $\sum_{j=1}^{N} a_{ij}(k) x_j(k) :=$ aver. consensus step on local estimates $x_j(k)$
- $g_i\big(x_i(k)\big) :=$ local subgradient of local cost $f_i(\cdot)$ at $x_i(k)$
- $\alpha_i(k) :=$ local stepsize

Convergence properties [Nedić Ozdaglar (2007)]

E.g., for *bounded subgradients* and $\alpha_i(k) = \alpha$ then

$$\lim_{k \to +\infty} \inf f\big(x_i(k)\big) = f^* + \text{small constant}$$

Dual decomposition methods (centralized)

Method of Multipliers [Bertsekas 1982]

Primal reformulation:

$$\begin{array}{ll} \text{minimize} & f(x) + \frac{\rho}{2} \|Ax - b\|_2^2 \\ \text{subject to} & Ax = b \end{array}$$

yelds to

1. $x_{k+1} = \arg\min_x \left(f(x) + \lambda_k^T (Ax - b) + \frac{\rho}{2} \|Ax - b\|_2^2 \right)$
2. $y_{k+1} = y_k + \rho(Ax_k - b)$

Convergence properties

- convergence to the optimum under mild assumptions (milder than for original dual ascent [Boyd et al. (2010)])

Dual decomposition methods (distributed)

Alternating Direction Method of Multipliers [Bertsekas Tsitsiklis 1997]

$$\begin{array}{ll} \text{minimize} & f_1(x_1) + f_2(x_2) \\ \text{subject to} & A_1 x_1 + A_2 x_2 - b = 0 \end{array}$$

Augmented Lagrangian:

$$\begin{aligned} L_\rho(x_1, x_2, \lambda) := \; & f_1(x_1) + f_2(x_2) \\ & + \lambda^T (A_1 x_1 + A_2 x_2 - b) \\ & + \frac{\rho}{2} \|A_1 x_1 + A_2 x_2 - b\|_2^2 \end{aligned}$$

Algorithm

1. $x_1(k+1) = \arg\min_{x_1} L_\rho\big(x_1, x_2(k), \lambda(k)\big)$
2. $x_2(k+1) = \arg\min_{x_2} L_\rho\big(x_1(k+1), x_2, \lambda(k)\big)$
3. $\lambda(k+1) = \lambda(k) + \rho\,(A_1 x_1 + A_2 x_2 - b)$

Dual decomposition methods (distributed)

Second-order dual descents [Bertsekas (2011)]

idea: use Newton-like procedures in the dual ascent step

$\downarrow$

usually involve graphs' Laplacian

Algorithms

Strategies to estimate Laplacians:

(usually based on matrices splitting strategies)

- [Zargham et al. (2011)] → Taylor expansions of Hessians
- [Jadbabaie et al. (2009)] → consensus-based iterative scheme

Drawbacks of the considered algorithms

Primal based strategies

- may be slow
- may not converge to the optimum

Dual based strategies

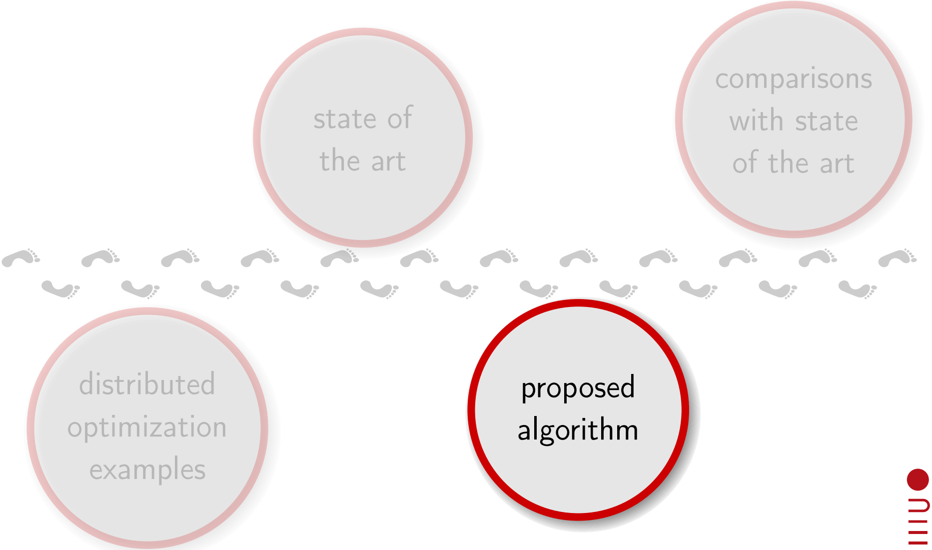
- may be computationally expensive
- require topological knowledge
- implementation to handle time-varying graphs, time delays, etc. may require effort

Motivations for our method

The algorithm that we want:

1. easy to be implemented
2. with small computational requirements
3. does not require synchronization or topology knowledge
4. assured to converge to global optimum
5. inheriting good properties of standard consensus
   convergence proofs, robustness, ...

Update


state of the art
comparisons with state of the art
distributed optimization examples
proposed algorithm

Our position in literature

How the proposed algorithm relates to other techniques?

- primal decomposition method
- unconstrained convex optimization
- uses second-order approximations
- strong assumptions on the cost functions
  (all other algorithms can work under our hypotheses)

our contribute: better convergence speed for primal methods

Illustrative example: quadratic local cost functions

Derivation of the algorithm - step 1 on 3

Simplified scalar scenario

$$f_i(x) = \frac{1}{2} a_i \left(x - b_i\right)^2 + c_i \qquad\qquad a_i > 0$$

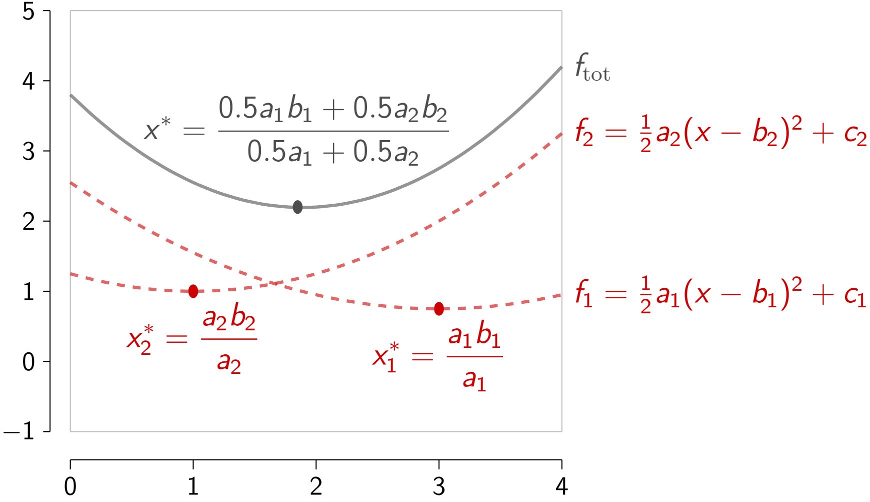
Corresponding solution

$$x^* = \frac{\sum_{i=1}^{N} a_i b_i}{\sum_{i=1}^{N} a_i} = \frac{\frac{1}{N}\sum_{i=1}^{N} a_i b_i}{\frac{1}{N}\sum_{i=1}^{N} a_i}$$

i.e. ***parallel of 2 average consensi!***

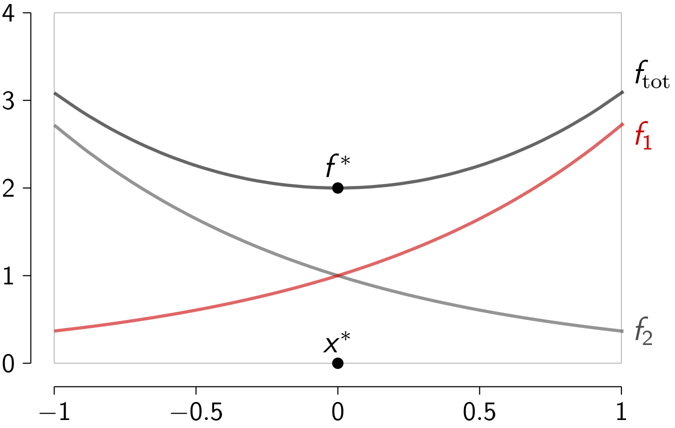
Illustrative example: quadratic local cost functions

Derivation of the algorithm - step 1 on 3 - graphical interpretation



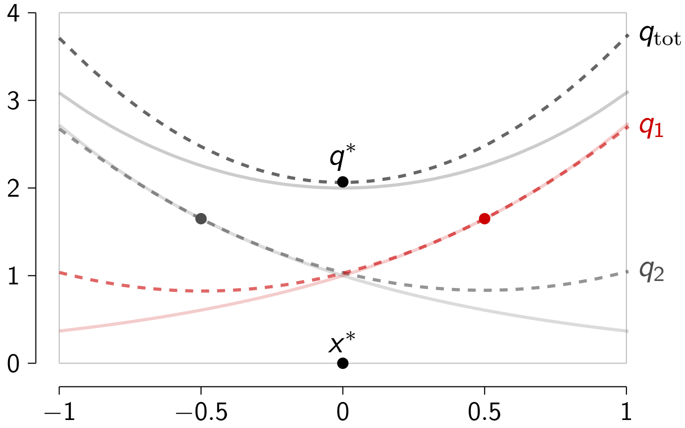
And for generic convex local cost functions?

Derivation of the algorithm - step 1 on 3



And for generic convex local cost functions?

Derivation of the algorithm - step 1 on 3



The initial idea

Derivation of the algorithm - step 2 on 3

For quadratics ...

$$x^* = \frac{\frac{1}{N}\sum_{i=1}^{N} a_i b_i}{\frac{1}{N}\sum_{i=1}^{N} a_i}$$

with

- $a_i b_i = f_i''(x_i)x_i - f_i'(x_i)$
- $a_i = f_i''(x_i)$

Does it imply that, for generic functions, ...??

$$x^* = \frac{\frac{1}{N}\sum_{i=1}^{N} \left(f_i''(x_i)x_i - f_i'(x_i)\right)}{\frac{1}{N}\sum_{i=1}^{N} f_i''(x_i)}$$

The initial idea

Derivation of the algorithm - step 2 on 3

The algorithm with the previous intuitions

1. initialize as before:
   - $y_i(0) := f_i''(x_i(0))x_i(0) - f_i'(x_i(0))$
   - $z_i(0) := f_i''(x_i(0))$
2. run 2 ***average consensi*** (P doubly stochastic):
   - $y_i(k+1) = Py_i(k)$
   - $z_i(k+1) = Pz_i(k)$
3. locally compute $x_i(k) = \dfrac{y_i(k)}{z_i(k)}$

remember: for quadratics $x^* = \dfrac{\frac{1}{N}\sum_{i=1}^{N}\left(f_i''(x_i)x_i - f_i'(x_i)\right)}{\frac{1}{N}\sum_{i=1}^{N} f_i''(x_i)}$

The initial idea

Derivation of the algorithm - step 3 on 3

Is this the algorithm?

$$\begin{cases} y_i(k+1) & = P y_i(k) \\ z_i(k+1) & = P z_i(k) \\ x_i(k) & = \dfrac{y_i(k)}{z_i(k)} \end{cases} \qquad \begin{cases} y_i(0) := f_i''(x_i(0))x_i(0) - f_i'(x_i(0)) \\ z_i(0) := f_i''(x_i(0)) \end{cases}$$

No, ***must provide 2 little modifications***:

- x_i changes $\Rightarrow$ must track the changing $f_i'(x_i)$ and $f_i''(x_i)$
- $x_i(k) = \dfrac{y_i(k)}{z_i(k)}$ too aggressive!! Should make it milder

The complete algorithm

Definitions

- $g_i(x_i(k)) = f_i''(x_i(k))x_i(k) - f_i'(x_i(k))$
- $h_i(x_i(k)) = f_i''(x_i(k))$
- bold font = vectorization

Main procedure

$$\begin{cases} \mathbf{y}(k+1) = P^M\left[\mathbf{y}(k) + \mathbf{g}(\mathbf{x}(k)) - \mathbf{g}(\mathbf{x}(k-1))\right] \\ \mathbf{z}(k+1) = P^M\left[\mathbf{z}(k) + \mathbf{h}(\mathbf{x}(k)) - \mathbf{h}(\mathbf{x}(k-1))\right] \\ \mathbf{x}(k+1) = (1-\varepsilon)\mathbf{x}(k) + \varepsilon \dfrac{\mathbf{y}(k+1)}{\mathbf{z}(k+1)} \end{cases}$$

Initialization

$$\mathbf{x}(0) = \mathbf{y}(0) = \mathbf{z}(0) = \mathbf{g}(\mathbf{x}(-1)) = \mathbf{h}(\mathbf{x}(-1)) = \mathbf{0}$$

Convergence property

Hypotheses

- $f_i \in \mathcal{C}^2(\mathbb{R})$
- f_i' and f_i'' bounded
- f_i strictly convex
- $x^* \neq \pm\infty$
- ***null initial conditions***

Thesis

- there is a positive $\bar{\varepsilon}$ s.t. if $\varepsilon < \bar{\varepsilon}$ then, exponentially,

$$\lim_{k \to +\infty} \mathbf{x}(k) = x^* \mathbb{1}$$

Robustness property

Additional hypothesis

- not-null initial conditions, but s.t.

$$\alpha := \mathbb{1}^T \left(\mathbf{v}(0) - \mathbf{y}(0)\right)$$

$$\beta := \mathbb{1}^T \left(\mathbf{w}(0) - \mathbf{z}(0)\right)$$

Thesis

- there are positive $\bar{\varepsilon}, \bar{\alpha}, \bar{\beta}$ s.t. if $\varepsilon < \bar{\varepsilon}, \alpha < \bar{\alpha}, \beta < \bar{\beta}$ then, exponentially,

$$\lim_{k \to +\infty} \mathbf{x}(k) = \phi(\alpha, \beta)\mathbb{1}$$

where $\phi(\alpha, \beta)$ is continuous and $\phi(0, 0) = x^*$

Sketch of the proof

importance of the proof:
gives insights on key properties

1. transform the algorithm in a continuous-time system
2. recognize the existence of a two-time scales dynamical system
3. analyze separately fast and slow dynamics
   (standard singular perturbation model analysis approach [Khalil (2002)])

1) transformation in a continuous-time system

$$\begin{cases} \mathbf{y}(k+1) &= P^M\left[\mathbf{y}(k) + \mathbf{g}(\mathbf{x}(k)) - \mathbf{g}(\mathbf{x}(k-1))\right] \\ \mathbf{z}(k+1) &= P^M\left[\mathbf{z}(k) + \mathbf{h}(\mathbf{x}(k)) - \mathbf{h}(\mathbf{x}(k-1))\right] \\ \mathbf{x}(k+1) &= (1-\varepsilon)\mathbf{x}(k) + \varepsilon\frac{\mathbf{y}(k+1)}{\mathbf{z}(k+1)} \end{cases}$$

1) transformation in a continuous-time system

$$\begin{cases} \mathbf{y}(k+1) &= P^M\left[\mathbf{y}(k) + \mathbf{g}(\mathbf{x}(k)) - \mathbf{g}(\mathbf{x}(k-1))\right] \\ \mathbf{z}(k+1) &= P^M\left[\mathbf{z}(k) + \mathbf{h}(\mathbf{x}(k)) - \mathbf{h}(\mathbf{x}(k-1))\right] \\ \mathbf{x}(k+1) &= (1-\varepsilon)\mathbf{x}(k) + \varepsilon\frac{\mathbf{y}(k+1)}{\mathbf{z}(k+1)} \end{cases}$$

$$\downarrow \qquad M = 1 \quad P := I - K$$

$$\begin{cases} \varepsilon\dot{\mathbf{v}}(t) &= -\mathbf{v}(t) + \mathbf{g}\left(\mathbf{x}(t)\right) \\ \varepsilon\dot{\mathbf{w}}(t) &= -\mathbf{w}(t) + \mathbf{h}\left(\mathbf{x}(t)\right) \\ \varepsilon\dot{\mathbf{y}}(t) &= -K\mathbf{y}(t) + (I-K)\left[\mathbf{g}\left(\mathbf{x}(t)\right) - \mathbf{v}(t)\right] \\ \varepsilon\dot{\mathbf{z}}(t) &= -K\mathbf{z}(t) + (I-K)\left[\mathbf{h}\left(\mathbf{x}(t)\right) - \mathbf{w}(t)\right] \\ \dot{\mathbf{x}}(t) &= -\mathbf{x}(t) + \frac{\mathbf{y}(t)}{\mathbf{z}(t)} \end{cases}$$

2) two-time scales dynamical system

$$
\begin{cases}
\varepsilon \dot{\mathbf{v}}(t) &= -\mathbf{v}(t) + \mathbf{g}\left(\mathbf{x}(t)\right) \\
\varepsilon \dot{\mathbf{w}}(t) &= -\mathbf{w}(t) + \mathbf{h}\left(\mathbf{x}(t)\right) \\
\varepsilon \dot{\mathbf{y}}(t) &= -K\mathbf{y}(t) + (I - K)\left[\mathbf{g}\left(\mathbf{x}(t)\right) - \mathbf{v}(t)\right] \\
\varepsilon \dot{\mathbf{z}}(t) &= -K\mathbf{z}(t) + (I - K)\left[\mathbf{h}\left(\mathbf{x}(t)\right) - \mathbf{w}(t)\right] \\
\hline
\dot{\mathbf{x}}(t) &= -\mathbf{x}(t) + \dfrac{\mathbf{y}(t)}{\mathbf{z}(t)}
\end{cases}
$$

Intuition: if ε is sufficiently small ...

- first subsystem is much faster than second one
- first subsystem is globally exponentially stable

3) analysis of the fast dynamics ($\varepsilon \to 0$)

$$\begin{cases} \varepsilon \dot{\mathbf{v}}(t) = -\mathbf{v}(t) + \mathbf{g}\left(\mathbf{x}(t)\right) \\ \varepsilon \dot{\mathbf{w}}(t) = -\mathbf{w}(t) + \mathbf{h}\left(\mathbf{x}(t)\right) \\ \varepsilon \dot{\mathbf{y}}(t) = -K\mathbf{y}(t) + (I - K)\left[\mathbf{g}\left(\mathbf{x}(t)\right) - \mathbf{v}(t)\right] \\ \varepsilon \dot{\mathbf{z}}(t) = -K\mathbf{z}(t) + (I - K)\left[\mathbf{h}\left(\mathbf{x}(t)\right) - \mathbf{w}(t)\right] \end{cases}$$

$$\Downarrow$$

$$\begin{cases} \mathbb{1}^T \dot{\mathbf{y}}(t) = \mathbb{1}^T \dot{\mathbf{v}}(t) \\ \mathbb{1}^T \dot{\mathbf{z}}(t) = \mathbb{1}^T \dot{\mathbf{w}}(t) \end{cases} \Longrightarrow \begin{cases} \mathbf{v}(t) \to \mathbf{g}\left(\mathbf{x}(t)\right) \\ \mathbf{w}(t) \to \mathbf{h}\left(\mathbf{x}(t)\right) \\ \mathbf{y}(t) \to \left[\frac{1}{N}\sum_{i=1}^N g_i\left(x_i(t)\right)\right]\mathbb{1} \\ \mathbf{z}(t) \to \left[\frac{1}{N}\sum_{i=1}^N h_i\left(x_i(t)\right)\right]\mathbb{1} \end{cases}$$

3) analysis of the fast dynamics ($\varepsilon \to 0$)

The tracking mechanism

$$\begin{cases} \mathbf{y}(t) \to \left[\dfrac{1}{N} \displaystyle\sum_{i=1}^{N} g_i \left(x_i(t) \right) \right] \mathbb{1} \\ \mathbf{z}(t) \to \left[\dfrac{1}{N} \displaystyle\sum_{i=1}^{N} h_i \left(x_i(t) \right) \right] \mathbb{1} \end{cases}$$

means

$$\begin{cases} \mathbf{y}(t) \to \left[\dfrac{1}{N} \displaystyle\sum_{i=1}^{N} f_i'' \left(x_i(t) \right) x_i(t) - f_i' \left(x_i(t) \right) \right] \mathbb{1} \\ \mathbf{z}(t) \to \left[\dfrac{1}{N} \displaystyle\sum_{i=1}^{N} f_i'' \left(x_i(t) \right) \right] \mathbb{1} \end{cases}$$

And the slow dynamics? ($\varepsilon \to 0$)

$$\dot{\mathbf{x}}(t) = -\mathbf{x}(t) + \frac{\mathbf{y}(\mathbf{x}(t))}{\mathbf{z}(\mathbf{x}(t))}$$

$$\dot{\mathbf{x}}(t) \approx -\mathbf{x}(t) + \frac{\left[\frac{1}{N}\sum_{i=1}^{N} f_i''\left(x_i(t)\right) x_i(t) - f_i'\left(x_i(t)\right)\right]\mathbb{1}}{\left[\frac{1}{N}\sum_{i=1}^{N} f_i''\left(x_i(t)\right)\right]\mathbb{1}}$$

$$\dot{x}_{\mathrm{ave}} \approx -\frac{f'_{\mathrm{global}}\left(x_{\mathrm{ave}}\right)}{f''_{\mathrm{global}}\left(x_{\mathrm{ave}}\right)}$$

i.e. a ***continuous-time Newton-Raphson strategy***

Recap

1. intuition: $\frac{\frac{1}{N}\sum_{i=1}^{N} f_i''(x_i)\,x_i - f_i'(x_i)}{\frac{1}{N}\sum_{i=1}^{N} f_i''(x_i)}$ and x^* are related

2. construct a system s.t.
   - $y_i(k) \rightarrow \frac{1}{N}\sum_{i=1}^{N} f_i''(x_i(k))\,x_i(k) - f_i'(x_i(k))$
   - $z_i(k) \rightarrow \frac{1}{N}\sum_{i=1}^{N} f_i''(x_i(k))$

3. discover that
   - $x_i(k) \rightarrow x_{\text{ave}}(k)$
   - $x_{\text{ave}}(k)$ evolution driven by a Newton-Raphson algorithm

Properties

Good qualities

- easy to be implemented
- "small" computational requirements
- inherits good qualities of consensus:
  - small topological knowledge requirements
  - robust to numerical error and communication noise

Bad qualities

Up to now, requires strong assumptions:

- $f_i \in \mathcal{C}^2(\mathbb{R})$
- f_i strictly convex
- f_i' and f_i'' bounded

On the necessity of the requirements

$f_i \in \mathcal{C}^2$

needed for the existence of f_i' and f_i''

f_i strictly convex

needed to assure $\dfrac{\frac{1}{N}\sum_{i=1}^{N} f_i''(x_i)\, x_i - f_i'(x_i)}{\frac{1}{N}\sum_{i=1}^{N} f_i''(x_i)} \neq \mathrm{NaN}$

f_i' and f_i'' bounded

assure non-pathological Newton-Raphson evolutions

Update

state of the art

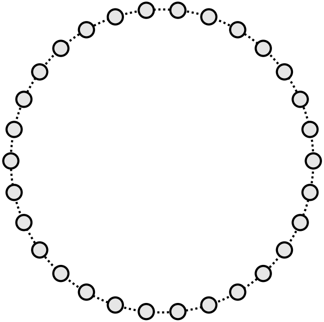
comparisons with state of the art

distributed optimization examples

proposed algorithm

Experiments description

- circulant graph, $N = 30$
- $P = \begin{bmatrix} 0.5 & 0.25 & & & 0.25 \\ 0.25 & 0.5 & 0.25 & & \\ & \ddots & \ddots & \ddots & \\ & & 0.25 & 0.5 & 0.25 \\ 0.25 & & & 0.25 & 0.5 \end{bmatrix}$
- $f_i =$ sum of exponentials

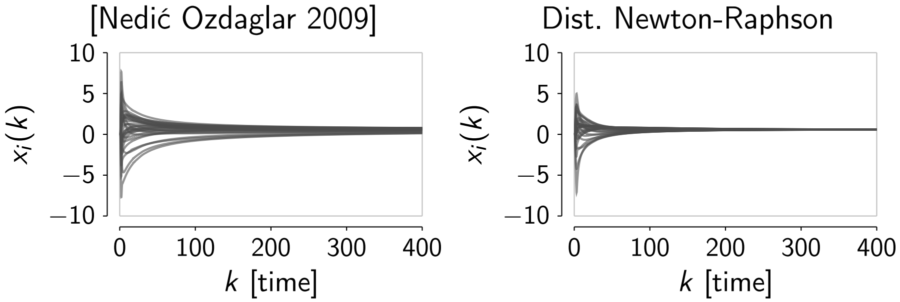


Comparisons with a Distributed Subgradient

Algorithm from [Nedić Ozdaglar 2009]

1. $\mathbf{x}^{(c)}(k) = P\mathbf{x}(k)$ (consensus step)
2. $x_i(k+1) = x_i^{(c)}(k) - \frac{\rho}{k} f_i'\left(x_i^{(c)}(k)\right)$ (local gradient descent)

Numerical comparison

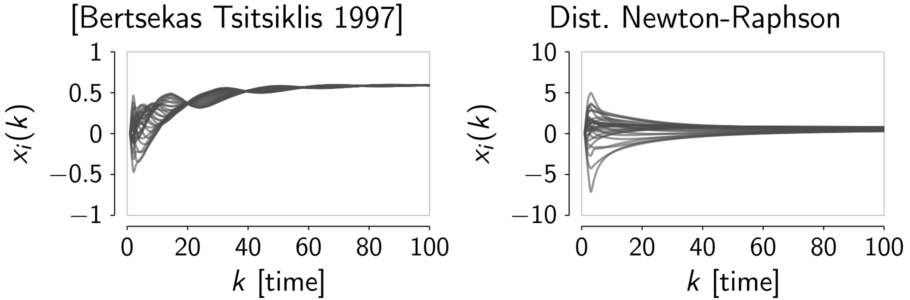


Comparisons with an ADMM (first-order)

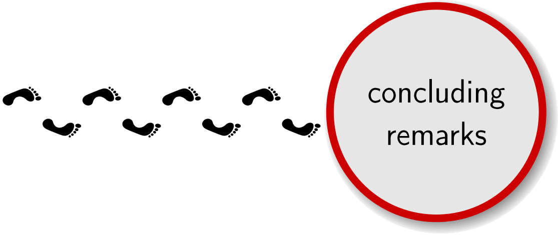
Algorithm from [Bertsekas Tsitsiklis 1997]

$$L_\rho := \sum_i \Big[f_i(x_i) + y_i^{(\ell)}(x_i - z_{i-1}) + y_i^{(c)}(x_i - z_i) + y_i^{(r)}(x_i - z_{i+1}) + \frac{\delta}{2}|x_i - z_{i-1}|^2 + \frac{\delta}{2}|x_i - z_i|^2 + \frac{\delta}{2}|x_i - z_{i+1}|^2 \Big]$$

Numerical comparison



Update



Conclusions and future works

The algorithm we proposed ...

- is a distributed Newton-Raphson strategy (+)
- requires minimal network topology knowledge (+)
- requires minimal agents synchronization (+)
- extremely simple to be implemented (+)
- converges to global optimum under convexity and smoothness assumptions (+ / -)
- numerically faster than subgradients (+) but slower than ADMM (-)

Conclusions and future works

Currently working on (or already performed)

- extension to multi-dimensional problems
- extension to modified Newton strategies
- analytical characterization of the convergence speed for quadratic functions and specific graphs
  (with comparisons to other methods)

Conclusions and future works

Plans for the future

- relax the assumptions
  (strict convexity, $\mathcal{C}^2$, ...)
- find automatic stepsizes tuning strategies
- propose quasi-Newton strategies

K. C. Kiwiel (2004)
Convergence of approximate and incremental subgradient methods for convex optimization
SIAM Journal on Optimization

D. P. Bertsekas (1982)
Constrained Optimization and Lagrange Multiplier Methods
Academic Press

D. P. Bertsekas and J. N. Tsitsiklis (1997)
Parallel and Distributed Computation: Numerical Methods
Athena Scientific

A. Nedić and A. Ozdaglar (2009)
Distributed subgradient methods for multi-agent optimization
IEEE Transactions on Automatic Control

B. Johansson (2008)
On Distributed Optimization in Networked Systems
Ph.D. Thesis, KTH

A. Nedić and A. Ozdaglar (2007)
On the Rate of Convergence of Distributed Subgradient Methods for Multi-agent Optimization
CDC

S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein (2010)
Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers
Foundations and Trends in Machine Learning

M. Zargham, A. Ribeiro, A. Ozdaglar, A. Jadbabaie (2011)
Accelerated Dual Descent for Network Optimization
ACC

A. Jadbabaie, A. Ozdaglar, M. Zargham (2009)
A Distributed Newton Method For Network Optimization
CDC

D. P. Bertsekas (2011)
Centralized and Distributed Newton Methods for Network Optimization and Extensions
Technical Report LIDS 2866

H. K. Khalil (2002)
Nonlinear Systems
Prentice Hall

Distributed convex optimization: a consensus-based Newton-Raphson approach

Damiano Varagnolo

joint work with A. Cenedese, G. Pillonetto, L. Schenato, F. Zanella

Department of Information Engineering - University of Padova

July 5th, 2011

varagnolo@dei.unipd.it
www.dei.unipd.it/∼varagnolo/
google: damiano varagnolo