Networked and Distributed Control Systems

Tamás Keviczky

Delft Center for Systems and Control
Delft University of Technology
Delft, The Netherlands

Lecture 6
2020

Aim of Lectures 6-9

“To present some of the key techniques for decomposition and distributed optimization in a coherent and comprehensible manner and their potential use in control”

Focus on understanding, and not on detailed proofs

- each lecture could be a full-semester course
- you will have to work with the reference material yourself!

Focus on fundamentals

- many techniques date back to 60’s-80’s
- but some are very recent, and research frontier is not far away

References at the end of the slides (will be included in handouts).

General Problem Formulation

Convex optimization problems

$$\begin{aligned} \underset{\theta}{\text{minimize}} \quad & f(\theta) \\ \text{subject to} \quad & \theta \in \Theta \end{aligned}$$

- $f : \mathbb{R}^M \to \mathbb{R}$ is a convex function, in general non-differentiable.
- $\Theta \subset \mathbb{R}^M$ is closed and convex

General Problem Formulation

Convex optimization problems

$$\begin{aligned} \underset{\theta}{\text{minimize}} \quad & f(\theta) \\ \text{subject to} \quad & \theta \in \Theta \end{aligned}$$

- $f : \mathbb{R}^M \to \mathbb{R}$ is a convex function, in general non-differentiable.
- $\Theta \subset \mathbb{R}^M$ is closed and convex
- Typical off-line solutions include: gradient descent, fixed-point iterations, Newton methods, interior point methods, etc.

General Problem Formulation

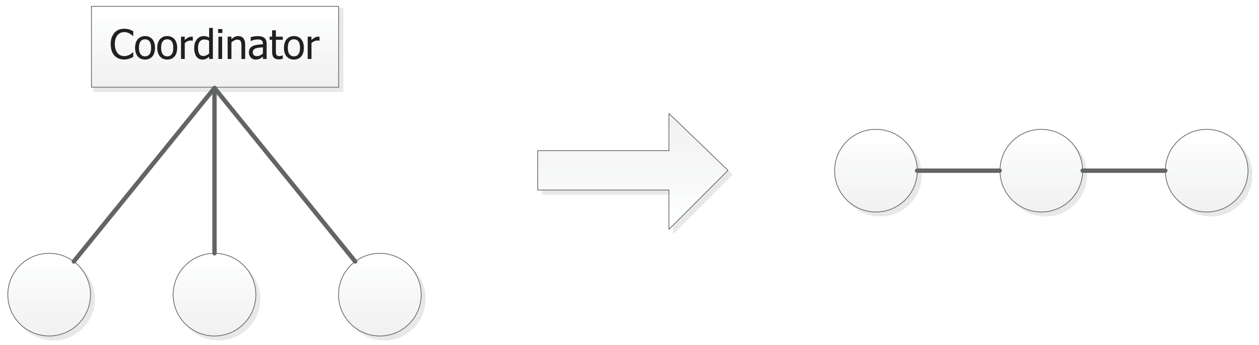
Convex optimization problems

$$\begin{aligned} \underset{\theta}{\text{minimize}} \quad & f(\theta) \\ \text{subject to} \quad & \theta \in \Theta \end{aligned}$$

- $f : \mathbb{R}^M \to \mathbb{R}$ is a convex function, in general non-differentiable.
- $\Theta \subset \mathbb{R}^M$ is closed and convex
- Typical off-line solutions include: gradient descent, fixed-point iterations, Newton methods, interior point methods, etc.
- Motivating problems with $f(\theta) = \sum_{i=1}^{N} f^i(\theta)$ include
    - resource allocation in computer networks
    - estimation in sensor networks
    - finite-time, optimal rendezvous of multiple dynamical agents

Distributed Optimization

- Distributed control and coordination in networks of systems
- Information and processing power is distributed among agents (cooperative and parallel solution)
- Global objective through local computations and interaction
- Traditional rationale is to improve scalability of solvers (huge data sets)
- Nowadays it is envisioned as a uniform approach to distributed decision-making
- Focus on methods without a central coordinator

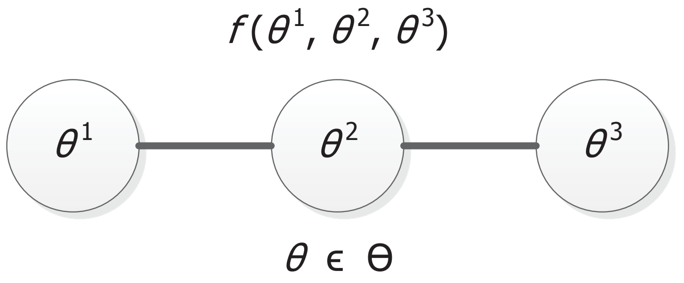


Examples of Networked Decision-Making Problems

- Distributed agreement (consensus)
- Distributed estimation (e.g. MHE)
- Distributed control (e.g. MPC)
- Resource allocation
- Distributed detection
- Communication protocol design (Internet congestion control, scheduling and power control in wireless systems)
- Power and water distribution
- Vehicle coordination and planning
- Sensor, social, and data networks
- and many more...

Typical Challenges in Distributed Optimization

$$\begin{aligned} \underset{\theta}{\text{minimize}} \quad & f(\theta^1, \theta^2, \theta^3) \\ \text{subject to} \quad & \theta^i \in \Theta \end{aligned}$$



In the networked, distributed setting one node is responsible for each variable.

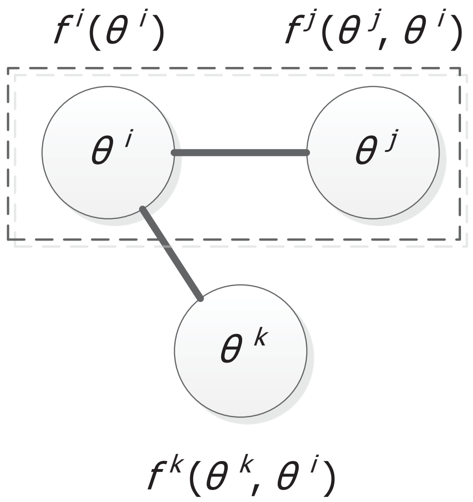
Basic gradient method

$$\theta_{k+1} = \mathcal{P}_\Theta \left[\theta_k - \alpha_k \nabla f(\theta_k)\right]$$

Issues include evaluating the gradient, computing the projection, setting the step-length.

Characteristics of Distributed Optimization Problems

- Information constraints
  - what do the nodes know?
  - what can they decide?
  - how do decisions interact?



- Decision variables (local vs global)
- Objective functions (separable vs coupled, private vs public)
- Constraints (individual vs coupled)
- Execution (synchronous vs asynchronous)
- Constraint satisfaction or agreement (asymptotically vs every iteration)

Outline of Lectures 6-8

- Basics of convex optimization
  - Using first-order methods
  - Brief duality theory (see on extra slides)
- Distributed optimization and decomposition methods
  - Basic techniques and ideas, proofs in references
- A brief lecture on ADMM (see on extra slides)
- Incremental subgradient methods
  - Example: finite-time optimal rendezvous
- Relaxing communication constraints
  - Basic consensus models and results (see Lecture 3)
  - Combined consensus/subgradient methods
- Summary

Convex Optimization Using First-Order Methods

The aim is to understand

- Properties and analysis techniques for basic gradient method
- The interplay between problem structure and convergence rate guarantees
- How we can deal with non-smoothness, noise and constraints

Convex Optimization Using First-Order Methods

The aim is to understand

- Properties and analysis techniques for basic gradient method
- The interplay between problem structure and convergence rate guarantees
- How we can deal with non-smoothness, noise and constraints

Convex optimization

- Minimize convex function subject to convex constraints
- Local minima global, strong and useful theory

Convex Optimization Using First-Order Methods

The aim is to understand

- Properties and analysis techniques for basic gradient method
- The interplay between problem structure and convergence rate guarantees
- How we can deal with non-smoothness, noise and constraints

Convex optimization

- Minimize convex function subject to convex constraints
- Local minima global, strong and useful theory

First-order methods

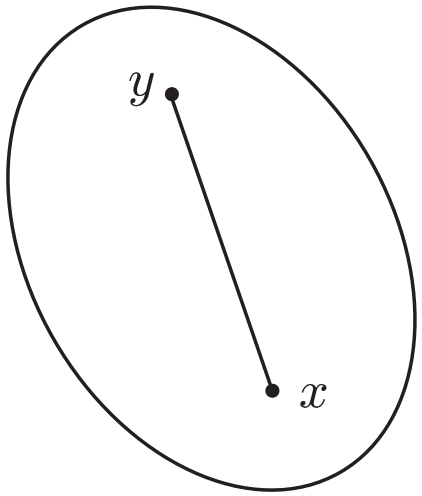
- Only use function and gradient evaluations (i.e., no Hessians)
- Easy to analyze, implement and distribute, yet competitive

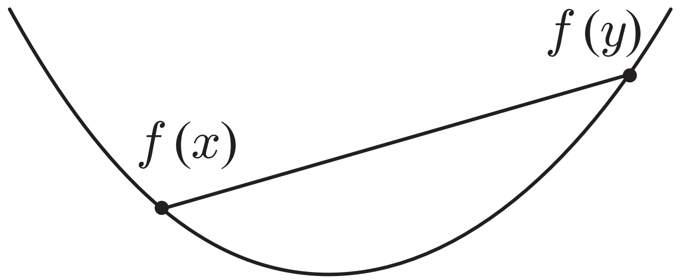
Basic Definitions

- Convexity
- Affine lower bounds from convexity
- Strong convexity - quadratic lower bound
- Lipschitz-continuous gradient - quadratic upper bound
- Condition number - impacts performance of first-order methods

Convergence rate definitions

Convex Sets and Functions

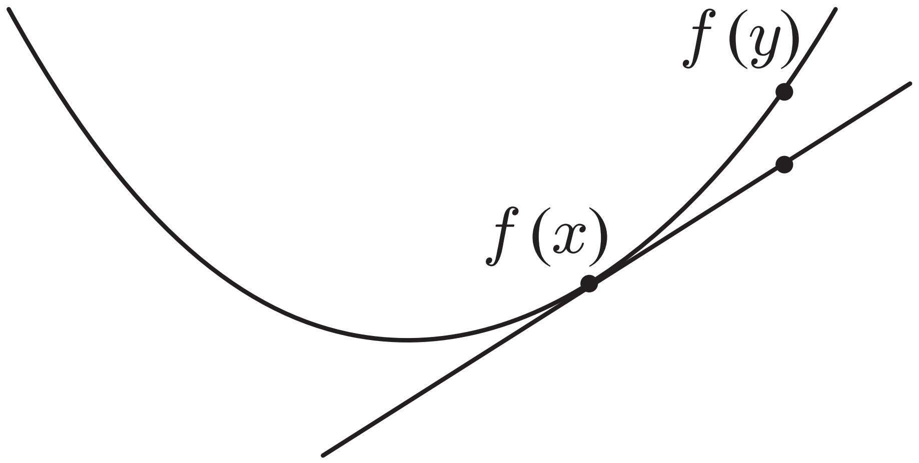




$$\alpha x + (1-\alpha) y \in X$$
$$\alpha \in [0,1]$$

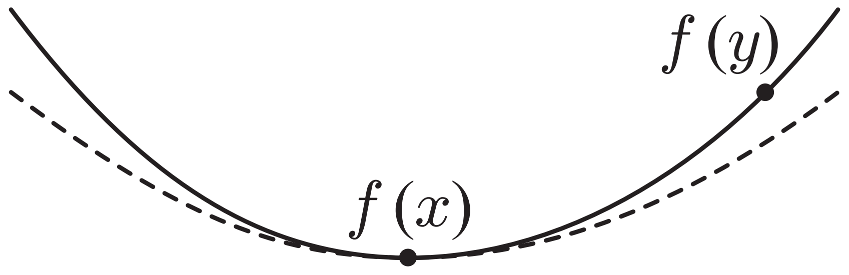
$$\alpha f(x) + (1-\alpha) f(y) \geq f(\alpha x + (1-\alpha) y)$$
$$\alpha \in [0,1]$$

Affine Lower Bounds from Convexity



$$f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle$$

Strong Convexity - Quadratic Lower Bound

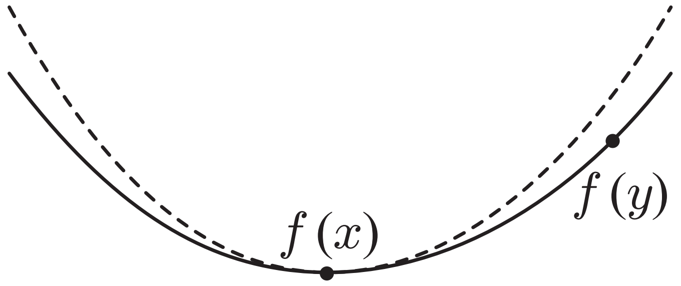


$$f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle + \frac{c}{2}\|y - x\|^2$$

Lipschitz-Continuous Gradient - Quadratic Upper Bound

Lipschitz-continuous gradient:

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|$$

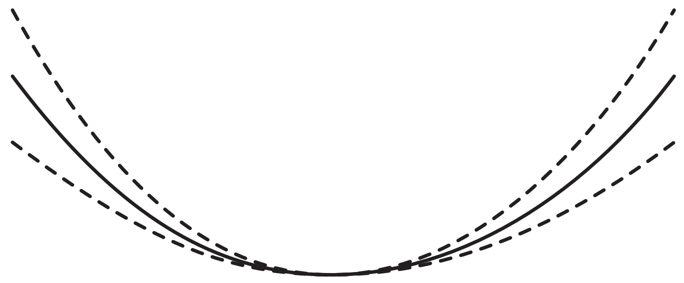


Yields a quadratic upper bound:

$$f(y) \leq f(x) + \langle \nabla f(x), y - x \rangle + \frac{L}{2}\|y - x\|^2$$

Condition Number

Strongly convex functions with Lipschitz gradient are bounded from above and below by quadratic functions.



The condition number $\kappa = L/c$ impacts performance of first-order methods.

(This property holds globally only for a limited function class.)

Convergence Rate Definitions

Typical choices for the error sequence:

$$e_i = f(X_i) - f^*, \quad e_i = \inf_{X^* \in \mathbb{X}^*} \|X_i - X^*\|, \quad e_i = \min_{0 \leq k \leq i} \|\nabla f(X_k)\|$$

Convergence Rate Definitions

Typical choices for the error sequence:

$$e_i = f(X_i) - f^*, \quad e_i = \inf_{X^* \in \mathbb{X}^*} \|X_i - X^*\|, \quad e_i = \min_{0 \le k \le i} \|\nabla f(X_k)\|$$

Linear convergence rate:

$$\boxed{\limsup_{i \to \infty} \frac{e_{i+1}}{e_i} \le q, \quad q \in (0,1)}$$

Also called geometric convergence (e.g., gradient and fast gradient methods).

Convergence Rate Definitions

Typical choices for the error sequence:

$$e_i = f(X_i) - f^*, \quad e_i = \inf_{X^* \in \mathbb{X}^*} \|X_i - X^*\|, \quad e_i = \min_{0 \le k \le i} \|\nabla f(X_k)\|$$

Linear convergence rate:

$$\boxed{\limsup_{i \to \infty} \frac{e_{i+1}}{e_i} \le q, \quad q \in (0,1)}$$

Also called geometric convergence (e.g., gradient and fast gradient methods).

Sublinear convergence rate:

$$\boxed{\limsup_{i \to \infty} \frac{e_{i+1}}{e_i} \ge 1}$$

That is, if it does not converge linearly, e.g., $e_i \le \frac{K}{(i+1)^p}$ with $p = \frac{1}{2}, 1, 2$.
Includes examples of subgradient, gradient, and fast gradient methods.

Convergence Rate Definitions

Superlinear convergence rate:

$$\boxed{\limsup_{i\to\infty} \frac{e_{i+1}}{e_i} = 0}$$

If it converges linearly with any convergence ratio $q \in (0, 1)$.

Example: Quasi-Newton methods (locally).

Convergence Rate Definitions

Superlinear convergence rate:

$$\boxed{\limsup_{i\to\infty} \frac{e_{i+1}}{e_i} = 0}$$

If it converges linearly with any convergence ratio $q \in (0,1)$.

Example: Quasi-Newton methods (locally).

Quadratic convergence rate:

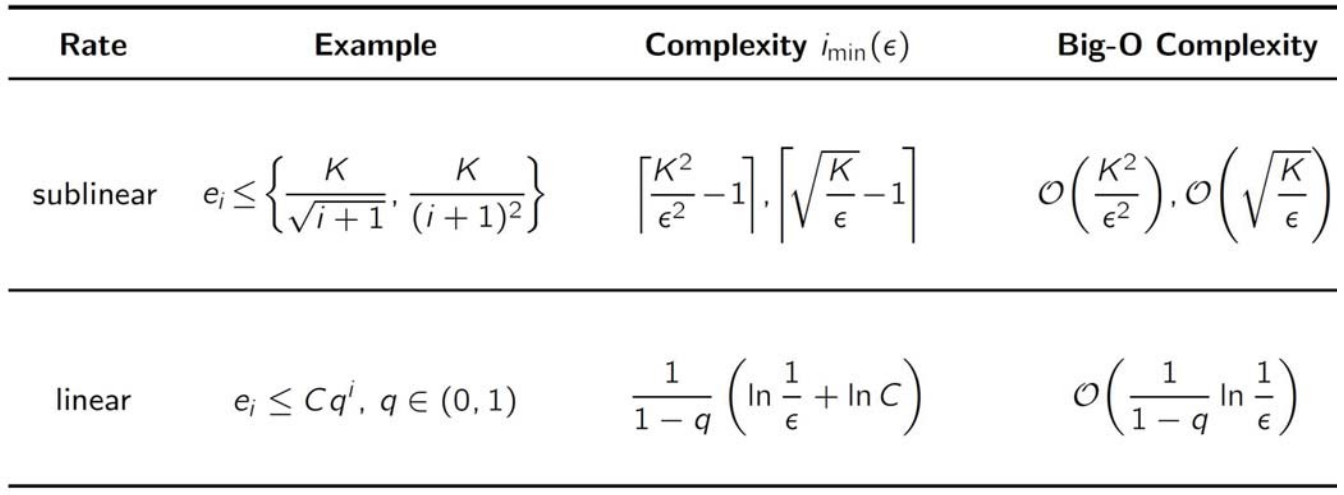
$$\boxed{e_{i+1} \leq M e_i^2}$$

Example: Newton's method (locally).

Complexity Definitions

A different measure for expressing the speed of convergence of a method.

Lower bound on the number of iterations such that $e_i \le \epsilon$.

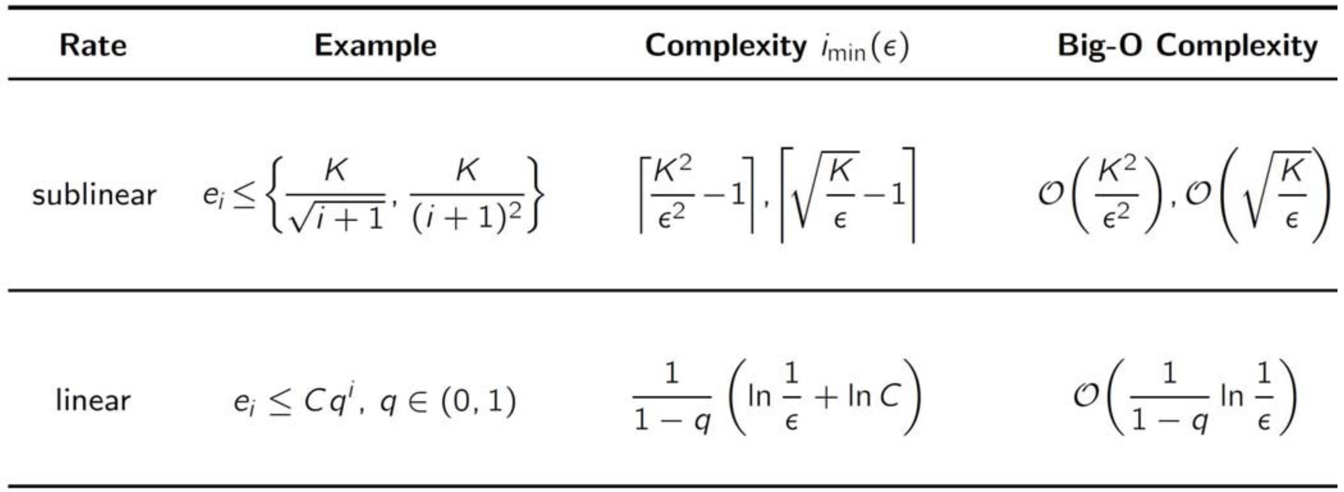


Rate	Example	Complexity $i_{\min}(\epsilon)$	Big-O Complexity
sublinear	$e_i \le \left\{\frac{K}{\sqrt{i+1}}, \frac{K}{(i+1)^2}\right\}$	$\left\lceil \frac{K^2}{\epsilon^2} - 1 \right\rceil, \left\lceil \sqrt{\frac{K}{\epsilon}} - 1 \right\rceil$	$\mathcal{O}\left(\frac{K^2}{\epsilon^2}\right), \mathcal{O}\left(\sqrt{\frac{K}{\epsilon}}\right)$
linear	$e_i \le Cq^i,\ q \in (0,1)$	$\frac{1}{1-q}\left(\ln \frac{1}{\epsilon} + \ln C\right)$	$\mathcal{O}\left(\frac{1}{1-q} \ln \frac{1}{\epsilon}\right)$

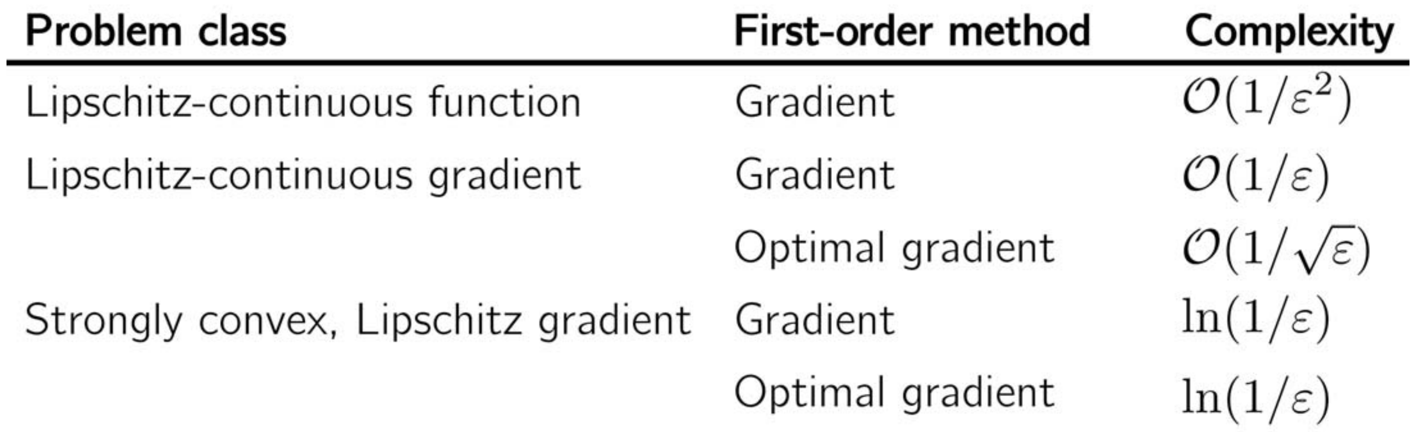
Complexity Definitions

A different measure for expressing the speed of convergence of a method.

Lower bound on the number of iterations such that $e_i \leq \epsilon$.



Rate	Example	Complexity $i_{\min}(\epsilon)$	Big-O Complexity
sublinear	$e_i \leq \left\{ \frac{K}{\sqrt{i+1}}, \frac{K}{(i+1)^2} \right\}$	$\left\lceil \frac{K^2}{\epsilon^2} - 1 \right\rceil, \left\lceil \sqrt{\frac{K}{\epsilon}} - 1 \right\rceil$	$\mathcal{O}\left(\frac{K^2}{\epsilon^2}\right), \mathcal{O}\left(\sqrt{\frac{K}{\epsilon}}\right)$
linear	$e_i \leq Cq^i,\ q \in (0,1)$	$\frac{1}{1-q}\left(\ln \frac{1}{\epsilon} + \ln C\right)$	$\mathcal{O}\left(\frac{1}{1-q} \ln \frac{1}{\epsilon}\right)$



Problem class	First-order method	Complexity
Lipschitz-continuous function	Gradient	$\mathcal{O}(1/\varepsilon^2)$
Lipschitz-continuous gradient	Gradient	$\mathcal{O}(1/\varepsilon)$
	Optimal gradient	$\mathcal{O}(1/\sqrt{\varepsilon})$
Strongly convex, Lipschitz gradient	Gradient	$\ln(1/\varepsilon)$
	Optimal gradient	$\ln(1/\varepsilon)$

Gradient Descent

Distributed optimization techniques often build on gradient-descent (easy to implement, analyze, yet competitive):

Basic gradient method

$$\theta_{k+1} = \mathcal{P}_{\Theta}\left[\theta_k - \alpha_k \nabla f(\theta_k)\right]$$

In off-line techniques, step-length is often decided by line-search.

If f is strictly convex, this converges for sufficiently small constant α, and the convergence rate depends on the Hessian of f.

The basic gradient method

Basic gradient method

$$x(t+1) = x(t) - \alpha(t)\nabla f(x(t))$$

A **descent** method (for small enough step-size $\alpha(t)$).

Convergence proof.

$$\begin{aligned}\|x(t+1) - x^\star\|_2^2 &= \|x(t) - x^\star\|_2^2 - 2\alpha(t)\langle \nabla f(x(t)), x(t) - x^\star\rangle + \alpha(t)^2\|\nabla f(x(t))\|_2^2 \\ &\leq \|x(t) - x^\star\|_2^2 - 2\alpha(t)\left(f(x(t)) - f^\star\right) + \alpha(t)^2\|\nabla f(x(t))\|_2^2\end{aligned}$$

Where the inequality follows from convexity of f

Gradient method convergence proof

Applying recursively, we find

$$\|x(T) - x^\star\|_2^2 \leq \|x(0) - x^\star\|_2^2 - 2\sum_{t=0}^{T-1} \alpha(t)(f(x(t)) - f^\star) + \sum_{t=0}^{T-1} \alpha^2(t)\|\nabla f(x(t))\|_2^2$$

Since gradient method is descent, and norms are non-negative

$$2(f(x(T)) - f^\star)\sum_{t=0}^{T-1} \alpha(t) \leq \|x(0) - x^\star\|_2^2 + \sum_{t=0}^{T-1} \alpha^2(t)\|\nabla f(x(t))\|_2^2$$

Hence, with $R_0 = \|x(0) - x^\star\|$

$$f(x(T)) - f^\star \leq \frac{R_0^2 + \sum_{t=0}^{T-1} \alpha^2(t)\|\nabla f(x(t))\|_2^2}{2\sum_{t=0}^{T-1} \alpha(t)}$$

Further assumptions needed to guarantee convergence!

Gradient method discussion

If we assume that f is Lipschitz, *i.e.* $\|\nabla f(x(t))\| \leq L_f$

$$f(x(T)) - f^\star) \leq \frac{R_0^2 + L_f^2 \sum_{t=0}^{T-1} \alpha^2(t)}{2\sum_{t=0}^{T-1} \alpha(t)}$$

Then,

- For fixed step-size $\alpha(t) = \alpha$

$$\lim_{T\to\infty} f(x(T)) \leq f^\star + \frac{\alpha L_f^2}{2}$$

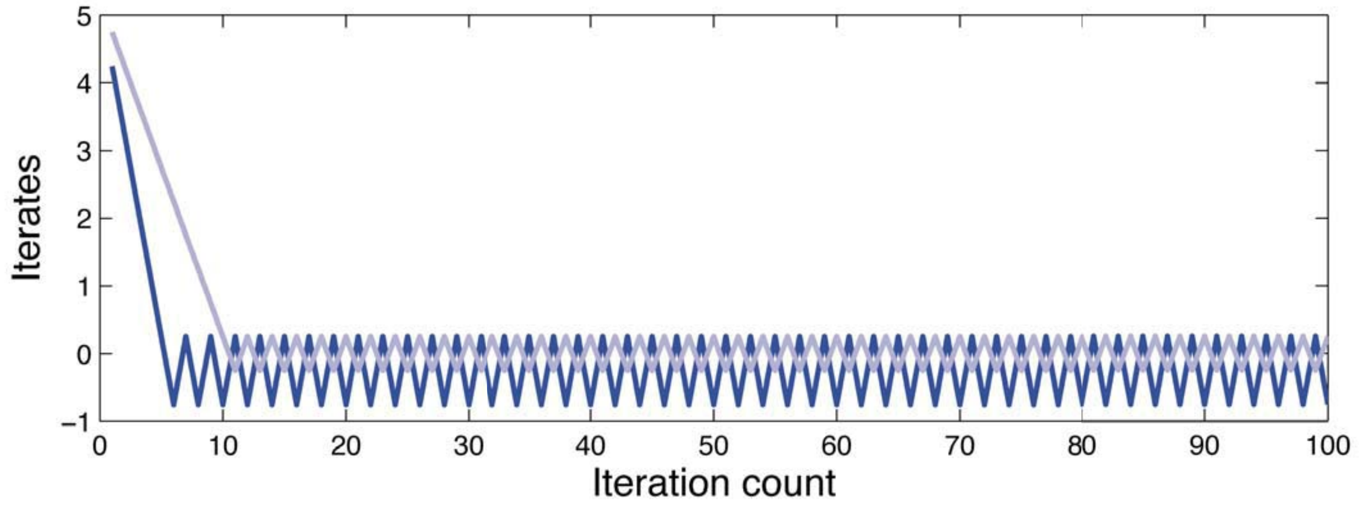
- For diminishing stepsizes $\sum_{t=0}^{\infty} \alpha^2(t) < \infty, \ \sum_{t=0}^{\infty} \alpha(t) = \infty$

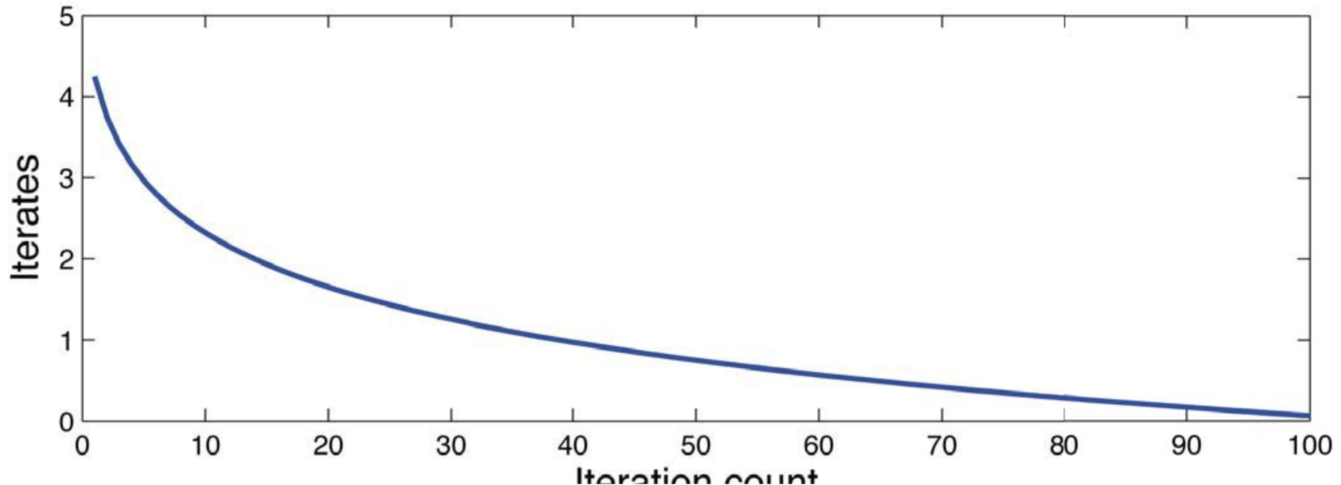
$$\lim_{T\to\infty} f(x(T)) = f^\star$$

- Accuracy ε can be obtained in $\left(R_0 L_f\right)^2 / \varepsilon^2$ steps

Example

Smaller residual error for smaller stepsize, convergence for diminishing





Strongly convex functions with Lipschitz gradient

As in the basic gradient method proof

$$\|x(t+1) - x^\star\|_2^2 = \|x(t) - x^\star\|_2^2 - 2\alpha(t)\langle \nabla f(x(t)), x(t) - x^\star \rangle + \alpha^2(t)\|\nabla f(x(t))\|_2^2$$

For strongly convex functions with Lipschitz-continuous gradient, it holds

$$\langle \nabla f(x(t)), x(t) - x^\star \rangle \geq \frac{cL}{c+L}\|x(t) - x^\star\|_2^2 + \frac{1}{c+L}\|\nabla f(x(t))\|^2$$

so

$$\|x(t+1) - x^\star\|_2^2 \leq \left(1 + \frac{2\alpha(t)cL}{c+L}\right)\|x(t) - x^\star\|_2^2 + \alpha(t)\left(\alpha(t) - \frac{2}{c+L}\right)\|\nabla f(x(t))\|_2^2$$

Hence, if $\alpha(t) \leq 2/(c+L)$ we obtain **linear convergence** rate

$$\|x(t+1) - x^\star\|_2^2 \leq \left(1 - \frac{2cL}{c+L}\alpha(t)\right)\|x(t) - x^\star\|_2^2$$

Order-optimal methods

The basic gradient method is **not** the optimal first-order method.

– optimal first-order methods typically use memory, e.g.

$$x(t+1) = y(t) - L^{-1}\nabla f(y(t))$$
$$y(t+1) = x(t+1) + \frac{1-\sqrt{\kappa}}{1+\sqrt{\kappa}}(x(t+1) - x(t))$$

Particularly useful when f is convex and has Lipschitz-continuous gradient

– from $\mathcal{O}(1/\varepsilon)$ to $\mathcal{O}(1/\sqrt{\varepsilon})$

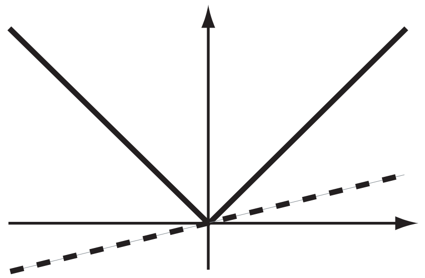
– achieves optimal rate (same as basic gradient) also in other cases

– not always fastest first-order method

Gradient methods: limits of performance

Problem class	**First-order method**	**Complexity**	**e=1%**
Lipschitz-continuous function	Gradient	$\mathcal{O}(1/\varepsilon^2)$	10,000
Lipschitz-continuous gradient	Gradient	$\mathcal{O}(1/\varepsilon)$	100
	Optimal gradient	$\mathcal{O}(1/\sqrt{\varepsilon})$	10
Strongly convex, Lipschitz gradient	Gradient	$\ln(1/\varepsilon)$	2.3
	Optimal gradient	$\ln(1/\varepsilon)$	

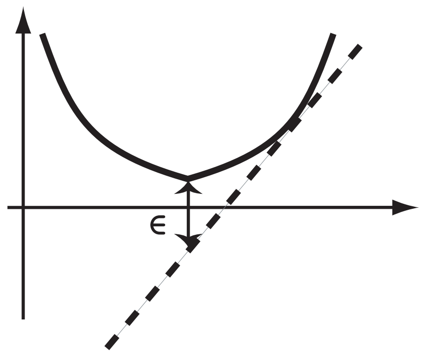
Subgradient and ϵ-subgradient



Definition

g is a subgradient for f at θ_0 if $g \in \partial f(\theta_0)$.

$$\partial f(\theta_0) = \left\{ g \in \mathbb{R}^M | f(\theta) \geq f(\theta_0) + g^\intercal(\theta - \theta_0), \forall \theta \in \mathbb{R}^M \right\}.$$



Definition

g is an ϵ-subgradient for f at θ_0 if $g \in \partial_\epsilon f(\theta_0)$.

$$\partial_\epsilon f(\theta_0) = \left\{ g \in \mathbb{R}^M | f(\theta) \geq f(\theta_0) + g^\intercal(\theta - \theta_0) - \epsilon, \forall \theta \in \mathbb{R}^M \right\}.$$

Subgradients are affine global underestimators and coincide with gradient if f is smooth.

Non-Smooth Version of Gradient Descent (Subgradient Methods)

If a convex optimization problem includes a non-differentiable $f(\cdot)$

$$\begin{aligned} \underset{\theta}{\text{minimize}} \quad & f(\theta) \\ \text{subject to} \quad & \theta \in \Theta \end{aligned}$$

then we can solve it with

Projected subgradient method

$$\theta_{k+1} = \mathcal{P}_\Theta \left[\theta_k - \alpha_k g_k\right]$$

where g_k is a subgradient of f at θ_k.

The method converges if g is bounded and $\sum_k^\infty \alpha_k^2 < \infty, \sum_k^\infty \alpha_k = \infty$.

The subgradient method

As the gradient method, but using subgradients instead

$$x(t+1) = x(t) - \alpha(t)s(t), \quad s(t) \in \partial f(x(t))$$

Not a descent method.

Hence, cannot bound $\sum_{t=0}^{T} \alpha(t)(f(x(t)) - f^\star)$ as before . Rather, we find

$$\inf_t f(x(t)) \leq f^\star + \frac{R_0^2 + \sum_{t=0}^{T} \alpha^2(t)\|s(t)\|_2^2}{2\sum_{t=0}^{T} \alpha(t)}$$

If subgradients are bounded, then same conclusions as for gradient method. (step-size, convergence rates, ...)

Averages behave better...

The running averages of iterates

$$\overline{x}(t) = \frac{1}{t}\sum_{k=0}^{t} x(k)$$

are often better-behaved than iterates themselves.

Specifically, if subgradients are bounded $\|s_x\| \leq L$, then averages satisfy

$$f(\overline{x}(T)) \leq f^\star + \frac{\sqrt{2}R_0 L}{\sqrt{T}}$$

(note how “inf” is gone)

Gradient method for constrained optimization

Constrained minimization problem

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & x \in X \end{array}$$

If projections onto X are easy to compute, can use **projected gradient**

$$x(t+1) = P_X\{x(t) - \alpha(t)\nabla f(x(t))\}$$

Same convergence proof as before, since projections are non-expansive

$$\|P_X\{x\} - P_X\{y\}\|^2 \leq \|x - y\|^2$$

Beyond the basic methods

Smooth optimization of non-smooth functions

- epsilon-optimal solution to non-smooth problem requires many iterations
- often better to smooth function and apply order-optimal method

Exploiting structure

- when problem is smooth problem + easily-solvable non-smooth
- many current applications in compressed sensing, sparse optimization

...

Summary

First-order methods for convex optimization:

- Gradient method: convergence proof and convergence rate estimates
- Order-optimal gradient methods: more states, but still only gradient information
- Easy to implement, strong performance for certain problem classes

Non-smooth optimization:

- Subgradient method
- Not a descent method, averaging gives better properties

References

- R.T. Rockafellar, Convex Analysis, Princeton University Press, 1996.
- D.P. Bertsekas, Nonlinear Programming: 2nd Edition, Athena Scientific, 1999.
- S. Boyd and L. Vandenberghe, Convex Optimization, Cambridge University Press. http://www.stanford.edu/ boyd/cvxbook/
- D.P. Bertsekas, A. Nedic, and A.E. Ozdaglar, Convex Analysis and Optimization, Athena Scientific, 2003.
- M. Johansson, Distributed optimization - intensive course, KTH Stockholm. http://www.ee.kth.se/∼mikaelj/dopt/

Networked and Distributed Control Systems

Tamás Keviczky

Delft Center for Systems and Control
Delft University of Technology
Delft, The Netherlands

Lecture 7
2020

Outline of Lectures 6-8

- Basics of convex optimization
  - Using first-order methods
  - Brief duality theory (see on extra slides)
- Distributed optimization and decomposition methods
  - Basic techniques and ideas, proofs in references
- A brief lecture on ADMM (see on extra slides)
- Incremental subgradient methods
  - Example: finite-time optimal rendezvous
- Relaxing communication constraints
  - Basic consensus models and results (see Lecture 3)
  - Combined consensus/subgradient methods
- Summary

Brief Duality Theory (also on extra slides)

Objective function and constraint set are assumed convex, thus local minima are also global.

For each primal problem, there is an associated dual

$$\begin{aligned} &\underset{\theta}{\text{minimize}} \quad f(\theta) && \underset{\lambda}{\text{maximize}} \quad d(\lambda^1, \cdots, \lambda^m) \\ &\text{subject to} \quad h^i(\theta) \leq 0 && \text{subject to} \quad \lambda^i \geq 0 \end{aligned}$$

Dual problem is always convex, and the dual function $d(\lambda)$ underestimates $f(\theta^*)$.

Recipe: introduce Lagrange multipliers (λ^i) and form Lagrangian

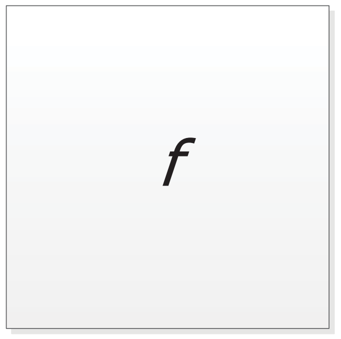
$$L(\theta, \lambda) = f(\theta) + \sum_i \lambda^i h^i(\theta)$$

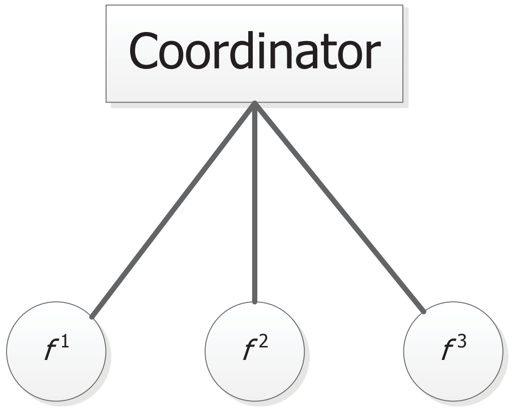
The dual function is obtained by minimizing over the primal variables (θ):

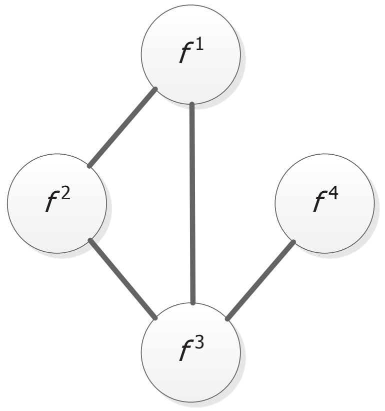
$$d(\lambda) = \inf_{\theta} \left[f(\theta) + \sum_i \lambda^i h^i(\theta) \right]$$

The dual function is often non-smooth, and has structure which the primal problem does not.

Model for Distributed Optimization







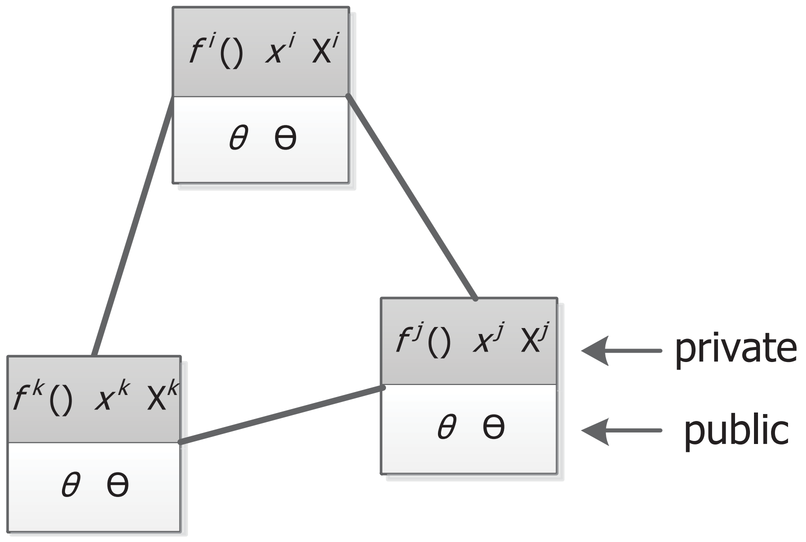
$$\begin{aligned} \underset{\theta}{\text{minimize}} \quad & f(\theta) \\ \text{subject to} \quad & \theta \in \Theta \end{aligned}$$

$$\begin{aligned} \underset{\theta}{\text{minimize}} \quad & \sum_{i=1}^{N} f^i(\theta) \\ \text{subject to} \quad & \theta \in \Theta \end{aligned}$$

$$\begin{aligned} \underset{\theta}{\text{minimize}} \quad & \sum_{i=1}^{N} f^i(\theta) \\ \text{subject to} \quad & \theta \in \Theta \\ & \mathcal{G} = (\mathcal{V}, \mathcal{E}) \end{aligned}$$

Model for Distributed Optimization

$$\begin{aligned} \underset{x,\theta}{\text{minimize}} \quad & \sum_{i=1}^{N} f^i(x^i, \theta^i, y^i) \\ \text{subject to} \quad & y^i = h^i(x, \theta) \\ & (x^i, \theta^i, y^i) \in C^i \end{aligned}$$

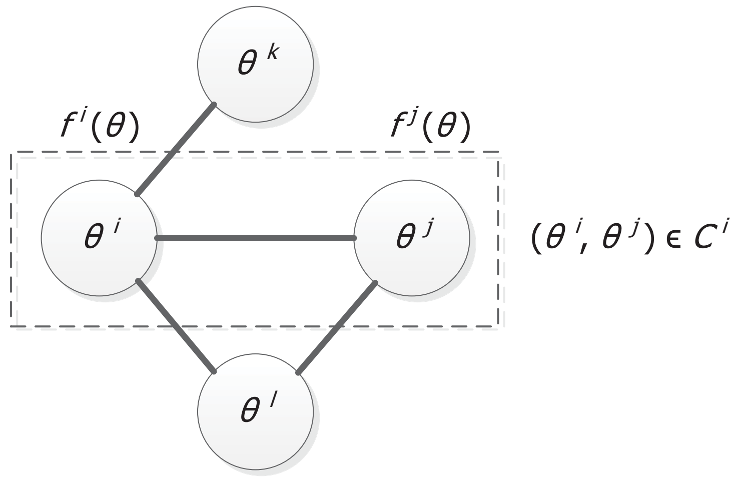


Information exchange is allowed only between connected nodes (graph constraint).

θ^i: complicating variables under control of node i
x^i: local variables of node i
y^i: information observable to node i

A Simplified Model

$$\begin{aligned} \underset{x,\theta}{\text{minimize}} \quad & \sum_{i=1}^{N} f^i(x^i, \theta^i) \\ \text{subject to} \quad & (\theta^i, \theta^j) \in C^i \\ & \mathcal{G} = (\mathcal{V}, \mathcal{E}) \end{aligned}$$



The graph is assumed to be static, without losses or delays, and the node clocks synchronized.

Distributed Optimization Through Decompositions

The basic idea of a decomposition is to decompose the original large problem (coupling constraints or objective function) into subproblems which are then coordinated by a master problem by means of some kind of signaling (master problem).

Primal decomposition methods

- based on decomposing the original primal problem
- various names:
    - partitioning of variables
    - decomposition by right-hand side allocation
    - decomposition with respect to variables

Dual decomposition methods

- based on decomposing the dual of the problem
- various names:
    - Lagrangian relaxation of the coupling constraints
    - decomposition with respect to constraints

Distributed Optimization Through Decompositions

Remarks

- A given decomposition method may lead to more than one distributed algorithm.
- Different distributed algorithms may be developed based on the same decomposition, depending on the choice of descent direction (e.g. gradient), the ordering of variable updates (e.g. Jacobi or Gauss-Seidel), and the timescale of nested loops.

Interpretation of decomposition methods

- Primal decomposition methods

  Master problem allocates amount of resources each subproblem can use (competing demands).

- Dual decomposition methods

  Master problem sets the pricing strategy (price of resources for each subproblem which decides amount based on price).

- The master problem can often be solved distributively through message passing (local or global, implicit or explicit).
- Penalty-function-based algorithms

  Distributed algorithms obtained by moving the coupled constraints to the augmented objective function in the primal problem through a penalty term.

Primal Decomposition

- Appropriate when the problem has a *coupling variable* such that, when fixed to some value, the rest of the optimization problem decouples into several subproblems.

Coupling through variables

$$\begin{aligned} \underset{y,\{\theta^i\}}{\text{minimize}} \quad & \sum_i f^i(\theta^i) \\ \text{subject to} \quad & \theta^i \in \Theta^i, \forall i \\ & A^i\theta^i \leq y, \forall i \\ & y \in \mathcal{Y} \end{aligned}$$

- If variable y were fixed, then the problem would decouple.
- This suggests separating the problem into two levels of optimization.

Primal Decomposition

At the *lower level*

We have the subproblems, one for each i over θ^i, with fixed y:

$$\begin{aligned} \underset{\theta^i}{\text{minimize}} \quad & f^i(\theta^i) \\ \text{subject to} \quad & \theta^i \in \Theta^i, \\ & A^i\theta^i \leq y \end{aligned}$$

At the *higher level*

We have the master problem, updating the coupling variable y:

$$\begin{aligned} \underset{y}{\text{minimize}} \quad & \sum_i f^{i*}(y) \\ \text{subject to} \quad & y \in \mathcal{Y} \end{aligned}$$

where $f^{i*}(y)$ is the optimal objective value of the i-th subproblem for a given y.

Dual Decomposition

- Appropriate when the problem has a *coupling constraint*, which if relaxed, the optimization problem decouples into several subproblems.

Coupling through constraints

$$\begin{aligned} \underset{\{\theta^i\}}{\text{minimize}} \quad & \sum_i f^i(\theta^i) \\ \text{subject to} \quad & \theta^i \in \Theta^i, \forall i \\ & \sum_i h^i(\theta^i) \leq c \end{aligned}$$

- If the last constraint was absent, then the problem would decouple.
- This suggests relaxing the coupling constraint such that the problem separates into two levels of optimization.

Dual Decomposition

At the *lower level*

We have the subproblems, one for each i over θ^i

$$\begin{aligned}\underset{\theta^i}{\text{minimize}} \quad & f^i(\theta^i) + \lambda^\mathsf{T} h^i(\theta^i)\\ \text{subject to} \quad & \theta^i \in \Theta^i\end{aligned}$$

At the *higher level*

We have the master dual problem, updating the dual variable λ:

$$\begin{aligned}\underset{\lambda}{\text{maximize}} \quad & d(\lambda) = \sum_i d^i(\lambda) - \lambda^\mathsf{T} c\\ \text{subject to} \quad & \lambda \geq 0\end{aligned}$$

where $d(\lambda)$ is the dual function obtained as the maximum value of the Lagrangian for a given λ.

This approach will only give appropriate results if strong duality holds.

Multi-Level Primal and Dual Decompositions

Primal and dual decompositions can be applied *recursively* to a problem to obtain smaller and smaller subproblems.

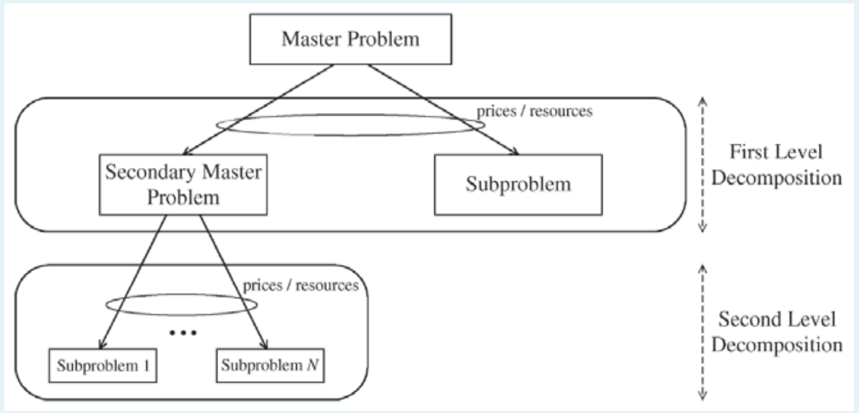
Combined problem

$$\begin{aligned} \underset{y,\{\theta^i\}}{\text{minimize}} \quad & \sum_i f^i(\theta^i, y) \\ \text{subject to} \quad & \theta^i \in \Theta^i, \forall i \\ & \sum_i h^i(\theta^i) \le c \\ & A^i\theta^i \le y, \forall i, y \in \mathcal{Y} \end{aligned}$$

Decoupling possible by

- primal decomposition (with respect to y)
- dual decomposition (with respect to $\sum_i h^i(\theta^i) \le c$)

Two-level decomposition



Using both leads to two-level optimization decomposition

- master primal problem
- secondary master dual problem
- subproblems

Alternative approaches by changing sequence of decompositions.

Example for Use of Primal Decomposition

$$\underset{x,\theta}{\text{minimize}} \quad f^i(x^i,\theta) + f^j(x^j,\theta)$$

Eliminate primal variables by

$$\phi^i(\theta) = \inf_{x^i} f^i(x^i,\theta), \quad \phi^j(\theta) = \inf_{x^j} f^j(x^j,\theta)$$

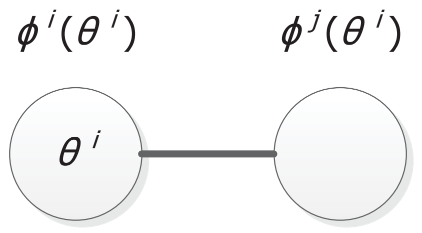
and consider

$$\underset{\theta}{\text{minimize}} \quad \phi^i(\theta) + \phi^j(\theta)$$

Which of the two nodes should take care of the variable θ if there is no coordinator?

Example for Use of Primal Decomposition (cont'd)

Let's say node i takes responsibility for θ:



Then a computational procedure for (minimize $\phi^i(\theta) + \phi^j(\theta)$) can be the following:

- Node i fixes θ^i, announces to node j
- Node j returns subgradient g^j of ϕ^j at θ^i
- Node i updates θ^i according to

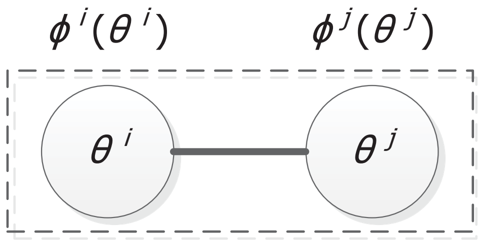
$$\theta^i_{k+1} = \mathcal{P}_\Theta \left[\theta^i_k - \alpha \left(g^i(\theta^i) + g^j(\theta^i)\right)\right]$$

This leads to asymptotic convergence.

Example for Use of Dual Decomposition

Introduce local variables, enforce consistency:

$$
\begin{aligned}
\underset{\theta}{\text{minimize}} \quad & \phi^i(\theta^i) + \phi^j(\theta^j) \\
\text{subject to} \quad & \theta^i = \theta^j
\end{aligned}
$$



The Lagrangian of the problem is

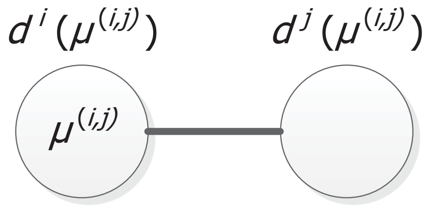
$$
\begin{aligned}
L(x, \mu^{(i,j)}) &= \phi^i(\theta^i) + \phi^j(\theta^j) + \mu^{(i,j)}(\theta^i - \theta^j) \\
&= \phi^i(\theta^i) + \mu^{(i,j)}\theta^i + \phi^j(\theta^j) - \mu^{(i,j)}\theta^j
\end{aligned}
$$

which leads to a separable dual function

$$
\begin{aligned}
d(\mu^{(i,j)}) &= \inf_{\theta^i} \left[\phi^i(\theta^i) + \mu^{(i,j)}\theta^i\right. \; + \inf_{\theta^j} \left[\phi^j(\theta^j) - \mu^{(i,j)}\theta^j\right. \\
&:= d^i(\mu^{(i,j)}) + d^j(\mu^{(i,j)})
\end{aligned}
$$

Example for Use of Dual Decomposition (cont'd)

Some node needs to take care of the dual variable



Computationally similar to primal decomposition:

- Node i fixes $\mu^{(i,j)}$, announces to node j
- Node j returns subgradient $\theta^{j*}(\mu^{(i,j)})$ of d^j at $\mu^{(i,j)}$
- Node i updates Lagrange multiplier according to

$$\mu_{k+1}^{(i,j)} = \mu_k^{(i,j)} + \alpha \quad \theta^{i*}(\mu_{\mathrm{k}}^{(i,j)}) - \theta^{j*}(\mu_{\mathrm{k}}^{(i,j)})$$

Dual variable converges to optimum asymptotically.

An Alternative Method: Fixed-Point Iterations

Use explicit characterization of optimal solution (from optimality conditions)

$$\underset{\theta}{\text{minimize}}\, \phi^i(\theta^i, \theta^j) + \phi^j(\theta^j, \theta^i)$$

First-order optimality conditions:

$$\frac{d}{d\theta^i}\left[\phi^i(\cdot) + \phi^j(\cdot)\right] := \gamma^i(\theta^i, \theta^j) = 0$$
$$\frac{d}{d\theta^j}\left[\phi^j(\cdot) + \phi^i(\cdot)\right] := \gamma^j(\theta^j, \theta^i) = 0$$

Solve via fixed-point iterations:

$$\theta^i_{k+1} = \theta^i_k - \gamma^i(\theta^i_k, \theta^j_k)$$
$$\theta^j_{k+1} = \theta^j_k - \gamma^j(\theta^j_k, \theta^i_k)$$

Assumed that γ^i and γ^j can be evaluated using local information.

Example with Complicating Constraints

In many cases it is the constraints that complicate matters (e.g. limited total resources):

$$\begin{aligned} \underset{\theta}{\text{minimize}} \quad & f^i(\theta^i) + f^j(\theta^j) \\ \text{subject to} \quad & \theta^i + \theta^j \leq 1 \end{aligned}$$

Nodes need to coordinate to fulfill the constraints. Some typical questions include:

- Asymptotically vs in every iteration?
- Hard vs soft constraints?

Similar techniques can be used as for coupled objectives.

Example with Complicating Constraints (Dual Decomposition)

$$\begin{aligned} \underset{\theta}{\text{minimize}} \quad & f^i(\theta^i) + f^j(\theta^j) \\ \text{subject to} \quad & \theta^i + \theta^j \leq 1 \end{aligned}$$

Similarly to the case of coupled objectives, introduce multiplier for constraint:

$$L(\theta, \lambda^{(i,j)}) = f^i(\theta^i) + f^j(\theta^j) + \lambda^{(i,j)}(\theta^i + \theta^j - 1)$$

Dual function is separable

$$d(\lambda^{(i,j)}) = \inf_{\theta^i} \left[f^i(\theta^i) + \lambda^{(i,j)}\theta^i \right. + \inf_{\theta^j} \left[f^j(\theta^j) - \lambda^{(i,j)}\theta^j \right. - \lambda^{(i,j)}$$

Example with Complicating Constraints (Dual Decomposition)

Computational procedure: one node takes care of dual variable update

- Node i fixes $\lambda^{(i,j)}$, announces to node j
- Node j returns subgradient of d^j at $\lambda^{(i,j)}$
- Node i updates Lagrange multiplier according to

$$\lambda_{k+1}^{(i,j)} = \mathcal{P}_{\mathbb{R}^+}\left[\lambda_k^{(i,j)} + \alpha\left(\theta^i + \theta^j - 1\right)\right]$$

Asymptotic convergence. Constraints satisfied only in the limit.

Example with Complicating Constraints (Primal Decomposition)

$$\begin{aligned} \underset{\theta}{\text{minimize}} \quad & f^i(\theta^i) + f^j(\theta^j) \\ \text{subject to} \quad & \theta^i + \theta^j \leq 1 \end{aligned}$$

Rewrite the problem as

$$\begin{aligned} \underset{\theta}{\text{minimize}} \quad & f^i(\theta^i) + f^j(\theta^j) \\ \text{subject to} \quad & \theta^i \leq r^i \\ & \theta^j \leq 1 - r^i \end{aligned}$$

and consider as a problem in r^i, i.e. $\text{minimize}_{r^i}\, \phi^i(r^i) + \phi^j(r^i)$ with

$$\begin{aligned} \phi^i(r^i) &= \inf_{\theta^i} f^i(\theta^i) \quad : \quad \theta^i \leq r^i \\ \phi^j(r^i) &= \inf_{\theta^j} f^j(\theta^j) \quad : \quad \theta^j \leq 1 - r^i \end{aligned}$$

Procedure as before. Constraint is *always* satisfied!

Example with Complicating Constraints (Penalty functions)

$$\begin{aligned} \underset{\theta}{\text{minimize}} \quad & f^i(\theta^i) + f^j(\theta^j) \\ \text{subject to} \quad & \theta^i + \theta^j \leq 1 \end{aligned}$$

Replace constraint by penalty function $\rho(\cdot)$ and consider

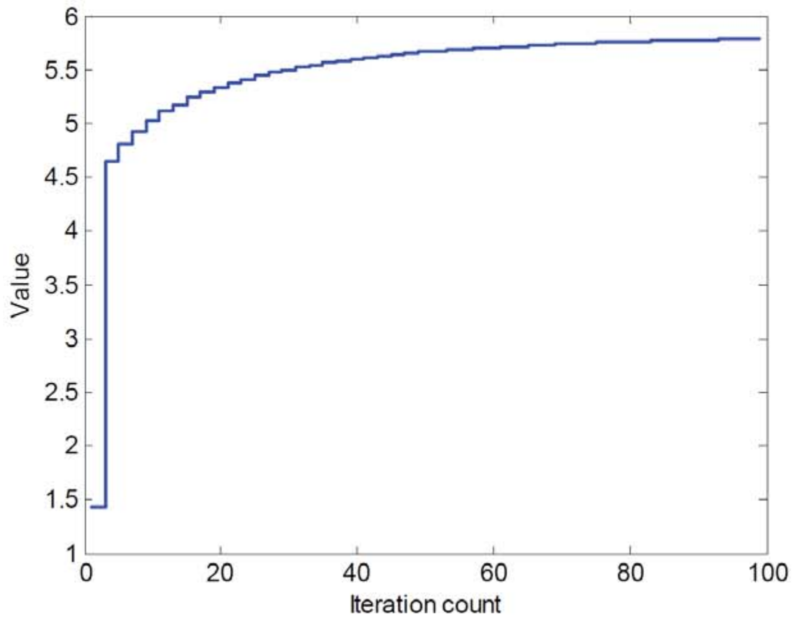
$$\underset{\theta}{\text{minimize}}\, f^i(\theta^i) + f^j(\theta^j) + \kappa\rho(\theta^i + \theta^j - 1)$$

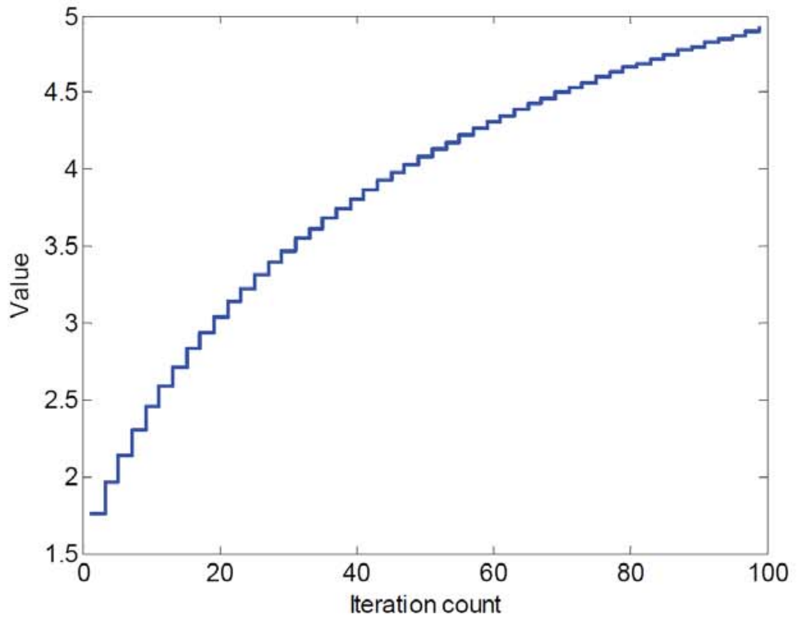
where $\rho(z) \to \infty$ as $z \to 0^-$, e.g. $\rho(z) = -\log(-z)$.
This modified problem can be solved by e.g. gradient descent.
Some issues include:

- approximate technique (optimality - consistency tradeoff)
- penalty function not separable (only if couplings are local)

Numerical Example with Complicating Constraints

$$\begin{array}{ll} \underset{x}{\text{minimize}} & 1/x_1 + 2/x_2 \\ \text{subject to} & x_1 \geq 0, x_2 \geq 0 \\ & x_1 + x_2 \leq 1 \end{array}$$

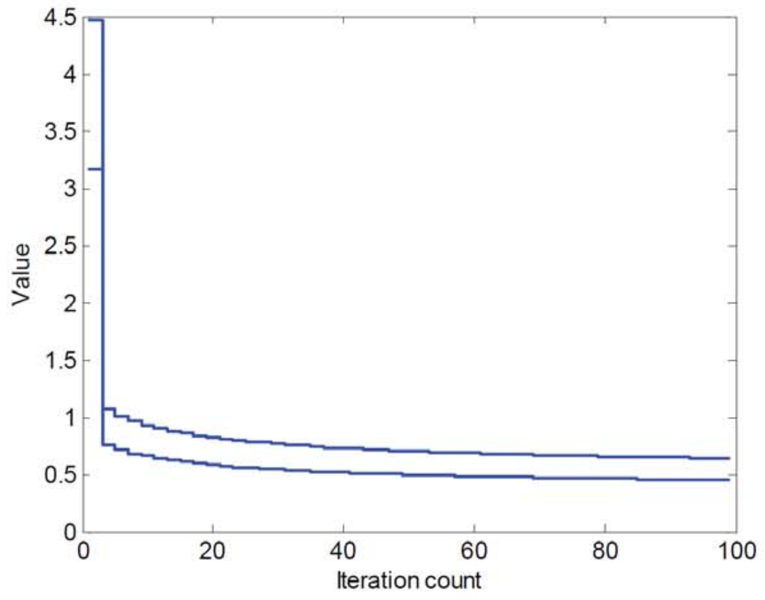


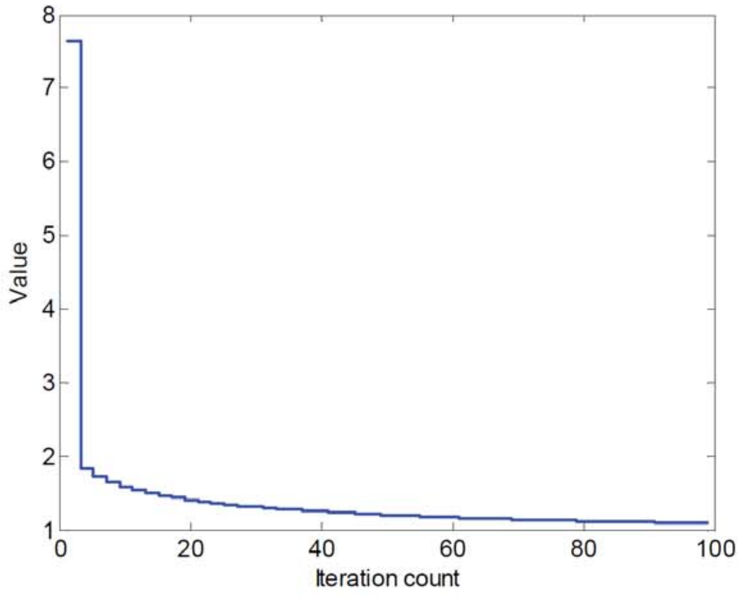


Dual objective (left) and dual variable (right) look all right.

Numerical Example with Complicating Constraints

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & 1/x_1 + 2/x_2 \\ \text{subject to} \quad & x_1 \geq 0, x_2 \geq 0 \\ & x_1 + x_2 \leq 1 \end{aligned}$$

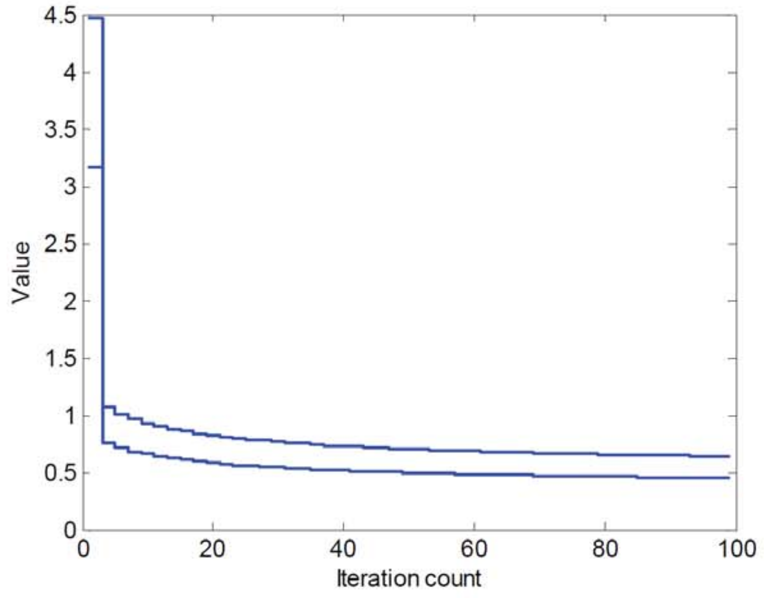


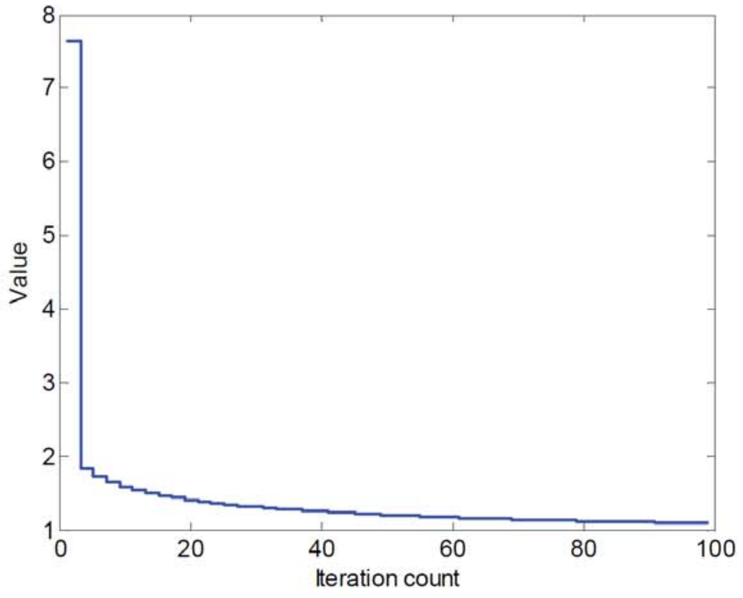


Primal variables (left) and their sum (right) indicate some issues. What do you think?

Numerical Example with Complicating Constraints

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & 1/x_1 + 2/x_2 \\ \text{subject to} \quad & x_1 \geq 0, x_2 \geq 0 \\ & x_1 + x_2 \leq 1 \end{aligned}$$





Primal variables (left) and their sum (right) indicate some issues.
What do you think?
Primal feasibility is non-trivial if problem is not strictly convex.

A Brief Lecture on ADMM

See on extra slides...

Summary

- Motivation for distributed optimization
    - From parallel computing to distributed decision-making
- A model for networked optimization
    - Local and global variables and constraints (possibly coupled)
    - Local objective functions (possible coupled)
    - Graph models information and communication constraints
- The basic tools and techniques
    - Complicating variables (objectives), coupling constraints
    - Primal/dual decomposition techniques

Acknowledgements

- M. Johansson
- M. Fazel

References

- R.T. Rockafellar, Convex Analysis, Princeton University Press, 1996.
- D.P. Bertsekas, Nonlinear Programming: 2nd Edition, Athena Scientific, 1999.
- S. Boyd and L. Vandenberghe, Convex Optimization, Cambridge University Press. http://www.stanford.edu/ boyd/cvxbook/
- D.P. Bertsekas and J.N. Tsitsiklis, Parallel and Distributed Computation: Numerical Methods, Athena Scientific, 1997. http://athenasc.com/pdcbook.html
- D.P. Bertsekas, A. Nedic, and A.E. Ozdaglar, Convex Analysis and Optimization, Athena Scientific, 2003.

References

- M. Johansson, Distributed optimization - intensive course, KTH Stockholm. http://www.ee.kth.se/∼mikaelj/dopt/
- B. Johansson, On Distributed Optimization in Networked Systems, PhD Thesis, KTH Stockholm, 2008.

Brief duality theory

Unconstrained minimization of a convex, differentiable $f : \mathbb{R}^n \to \mathbb{R}$,

$$\text{minimize} \quad f(x)$$

Minimizing sequence: $x^{(k)},\ k \to \infty$

$$f(x^{(k)}) \to f^*$$

Optimality condition: $\nabla f(x^*) = 0$

If $\nabla^2 f(x) \geq mI$, then a stopping criterion is

$$f(x) - f^* \leq \frac{1}{2m} \left\| \nabla f(x) \right\|^2$$

[Boyd – Vandenberghe, 2004]

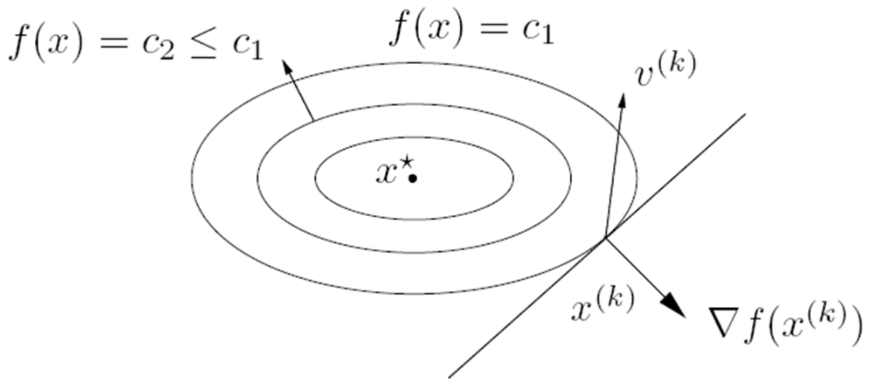
Descent methods

```
**given** starting point $x \in \mathbf{dom}\, f$
**repeat**
    1. *Compute a search direction* $v$
    2. *Line search.* Choose step size $t > 0$
    3. *Update.* $x := x + tv$
**until** stopping criterion is satisfied
```

descent method: $f(x^{(k+1)}) < f(x^{(k)})$

since f convex, v must be a **descent direction:**

$$\nabla f(x^{(k)})^T v^{(k)} < 0$$



Descent methods

choose v (descent direction)

- $v^{(k)} = -\nabla f(x^{(k)})$ (gradient)
- $v^{(k)} = -H^{(k)}\nabla f(x^{(k)})$, $H^{(k)} = H^{(k)T} \succ 0$
- $v^{(k)} = -\nabla^2 f(x^{(k)})^{-1}\nabla f(x^{(k)})$ (Newton's)

choose t (step size)

- exact line search: $t = \mathrm{argmin}_{s>0} f(x + sv)$
- backtracking line search (picks step size that reduces f 'enough')
- constant step size

Main idea extends to nondifferentiable f using subgradients.

The Lagrangian and Lagrange dual function

standard form optimization problem

$$\begin{array}{ll} \text{minimize} & f_0(x) \\ \text{subject to} & f_i(x) \le 0, \ i = 1, \dots, m \end{array}$$

- optimal value $p^\star$, domain D
- called **primal problem** (in context of duality)

(for now) we **don't** assume convexity

Lagrangian $L : \mathbf{R}^{n+m} \to \mathbf{R}$

$$L(x, \lambda) = f_0(x) + \lambda_1 f_1(x) + \cdots + \lambda_m f_m(x)$$

- λ_i called *Lagrange multipliers* or *dual variables*
- objective is *augmented* with weighted sum of constraint functions

(Lagrange) dual function $g : \mathbf{R}^m \to \mathbf{R} \cup \{-\infty\}$

$$\begin{aligned} g(\lambda) &= \inf_x L(x, \lambda) \\ &= \inf_x \left(f_0(x) + \lambda_1 f_1(x) + \cdots + \lambda_m f_m(x)\right) \end{aligned}$$

- minimum of augmented cost as function of weights
- can be $-\infty$ for some λ
- g is concave (even if f_i not convex!)

example: LP

$$\begin{array}{ll} \text{minimize} & c^T x \\ \text{subject to} & a_i^T x - b_i \le 0, \ i = 1, \dots, m \end{array}$$

$$\begin{aligned} L(x, \lambda) &= c^T x + \sum_{i=1}^m \lambda_i (a_i^T x - b_i) \\ &= -b^T \lambda + (A^T \lambda + c)^T x \end{aligned}$$

hence $g(\lambda) = \begin{cases} -b^T \lambda & \text{if } A^T \lambda + c = 0 \\ -\infty & \text{otherwise} \end{cases}$

Lower bound property and Lagrange dual problem

if $\lambda \succeq 0$ and x is primal feasible, then

$$g(\lambda) \leq f_0(x)$$

proof: if $f_i(x) \leq 0$ and $\lambda_i \geq 0$,

$$\begin{aligned} f_0(x) &\geq f_0(x) + \sum_i \lambda_i f_i(x) \\ &\geq \inf_z \left(f_0(z) + \sum_i \lambda_i f_i(z) \right) \\ &= g(\lambda) \end{aligned}$$

$f_0(x) - g(\lambda)$ is called the **duality gap** of (primal feasible) x and $\lambda \succeq 0$

minimize over primal feasible x to get, for any $\lambda \succeq 0$,

$$g(\lambda) \leq p^\star$$

$\lambda \in \mathbf{R}^m$ is **dual feasible** if $\lambda \succeq 0$ and $g(\lambda) > -\infty$

dual feasible points yield lower bounds on optimal value!

let's find **best** lower bound on $p^\star$:

$$\begin{array}{ll} \text{maximize} & g(\lambda) \\ \text{subject to} & \lambda \succeq 0 \end{array}$$

- called **(Lagrange) dual problem** (associated with primal problem)
- always a convex problem, even if primal isn't!
- optimal value denoted $d^\star$
- we always have $d^\star \leq p^\star$ (called *weak duality*)
- $p^\star - d^\star$ is *optimal duality gap*

Strong duality and constraint qualifications

for convex problems, we (usually) have *strong duality:*

$$d^\star = p^\star$$

(strong duality does not hold, in general, for nonconvex problems)

when strong duality holds, dual optimal $\lambda^\star$ serves as *certificate of optimality* for primal optimal point $x^\star$

many conditions or **constraint qualifications** guarantee strong duality for convex problems

Slater's condition: if primal problem is strictly feasible (and convex), *i.e.*, there exists $x \in \textbf{relint}\, D$ with

$$f_i(x) < 0, \; i = 1, \ldots, m$$

then we have $p^\star = d^\star$

A brief lecture on ADMM

(SC42100 course)

Tamás Keviczky

Delft Center for Systems and Control
Delft University of Technology
The Netherlands

2020

Outline

- Brief review of duality theory
- Dual decomposition and ascent methods
- Method of multipliers
- ADMM
- Some examples
- Open problems and references

Main sources:

- Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers (Boyd, Parikh, Chu, Peleato, Eckstein)
- EE364b course at Stanford University

Duality Theory in Optimization

Convex optimization problem (equality constrained for simplicity):

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & Ax = b \end{array}$$

- Lagrangian: $L(x, y) = f(x) + y^T(Ax - b)$
- Dual function: $g(y) = \inf_x L(x, y)$
- Dual problem: maximize $g(y) \quad \rightarrow \quad y^\star$
- Primal optimizer: $x^\star = \arg\min_x L(x, y^\star)$

Dual Ascent Method

The gradient method for the dual problem is

$$y^{k+1} = y^k + \alpha^k \nabla g(y^k),$$

where $\nabla g(y^k) = A\tilde{x} - b$ with $\tilde{x} = \arg\min_x L(x, y^k)$.

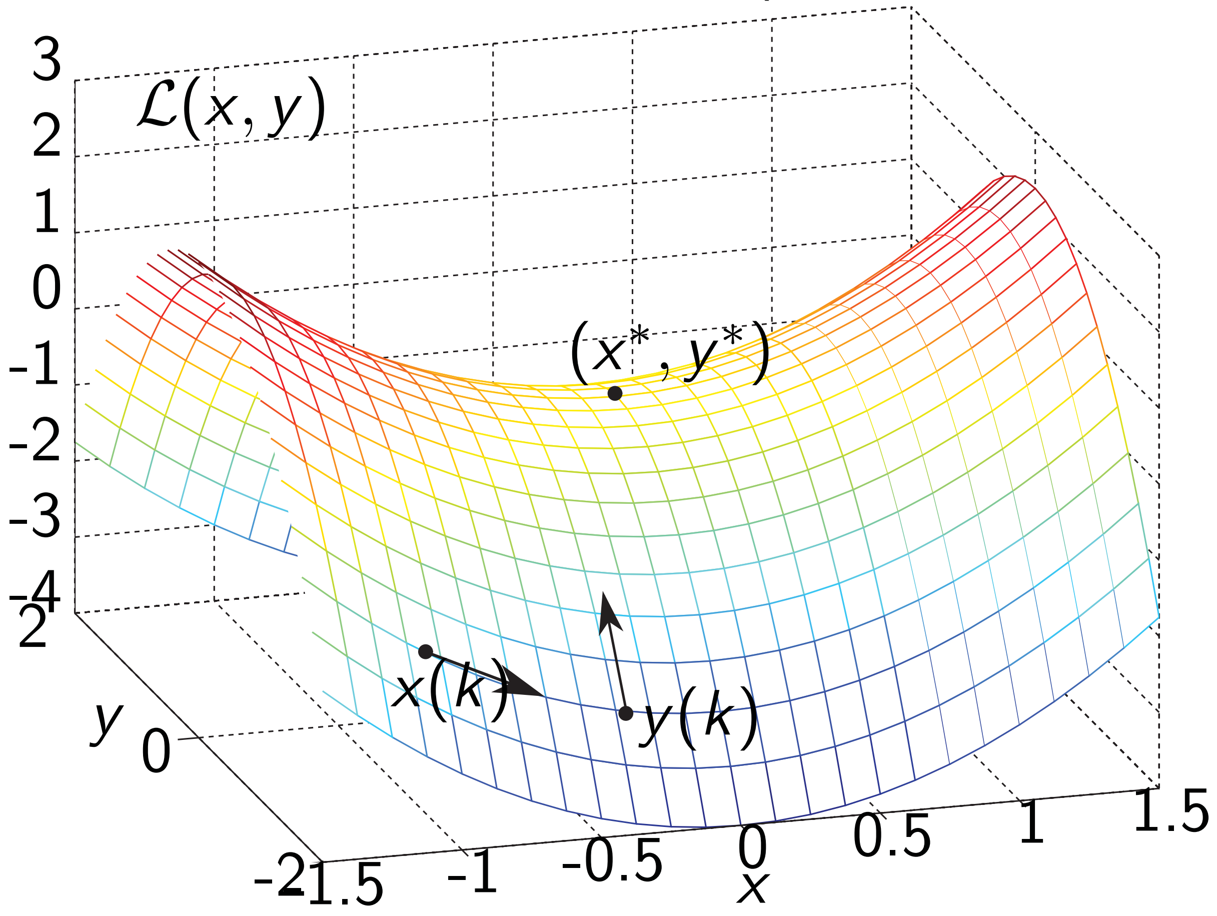
This leads to the following dual ascent method:

$$\begin{aligned} x^{k+1} &:= \arg\min_x L(x, y^k) && \text{x-minimization} \\ y^{k+1} &:= y^k + \alpha^k (Ax^{k+1} - b) && \text{dual update} \end{aligned}$$

- Simple to implement, yet often slow
- Strong assumptions for convergence (strict convexity, finite f, etc.)
- Significant speedup using Nesterov's fast gradient method

Illustration of Dual Ascent

A function with a saddle point (possible Lagrangian)



Dual Decomposition

If $f(x)$ is a separable function

$$f(x) = f_1(x_1) + \cdots + f_N(x_N), \quad x = (x_1, \cdots, x_N),$$

then the Lagrangian is separable in x:

$$L(x, y) = L_1(x_1, y) + \cdots + L_N(x_N, y) - y^T b,$$

$$L_i(x_i, y) = f_i(x_i) + y^T A_i x_i$$

Thus the x-minimization step in dual ascent splits into N separate minimizations that can be computed in parallel:

$$x_i^{k+1} := \arg\min_{x_i} L_i(x_i, y^k)$$

Dual Decomposition Implementation

Dual decomposition can be used to solve large-scale problems by

- solving in parallel iterative subproblems
- coordinating them via dual variable update

$$
\begin{aligned}
x^{k+1} &:= \arg\min_{x} L_i(x_i, y^k) \qquad i = 1, \ldots, N \\
y^{k+1} &:= y^k + \alpha^k \left(\sum_{i=1}^{N} A_i x_i^{k+1} - b \right)
\end{aligned}
$$

Main steps for implementation:

- broadcast y^k dual variable
- update x_i in parallel
- collect $A_i x_i^{k+1}$

Needs many assumptions for convergence, which is often quite slow.

Method of Multipliers

The main aim is to robustify the dual ascent approach by using an **augmented Lagrangian**:

$$L_\rho(x,y) = f(x) + y^T(Ax - b) + \frac{\rho}{2}\|Ax - b\|_2^2$$

The method of multipliers then replaces the Lagrangian and the dual update step size in the standard dual descent:

$$\begin{aligned} x^{k+1} &:= \arg\min_x L_\rho(x, y^k) \\ y^{k+1} &:= y^k + \rho\left(Ax_i^{k+1} - b\right) \end{aligned}$$

Dual Update Step in Method of Multipliers

Based on the first step, x^{k+1} minimizes $L_\rho(x, y^k)$ thus

$$\begin{aligned} 0 &= \nabla_x L_\rho(x^{k+1}, y^k) \\ &= \nabla_x f(x^{k+1}) + A^T \left(y^k + \rho(Ax^{k+1} - b)\right) \\ &= \nabla_x f(x^{k+1}) + A^T(y^{k+1}) \end{aligned}$$

Recall that for differentiable f the optimality conditions (primal and dual feasibility) are

$$Ax^\star - b = 0, \qquad \nabla f(x^\star) + A^T y^\star = 0,$$

therefore the dual update $y^{k+1} = y^k + \rho(x^{k+1} - b)$ makes (x^{k+1}, y^{k+1}) dual feasible, however primal feasibility is only achieved in the limit: $Ax^{k+1} - b \quad \rightarrow \quad 0.$

Properties of the Method of Multipliers

- Converges under much more relaxed conditions compared to dual decomposition (nondifferentiable, non-strictly convex, infinite f, etc.)
- Quadratic penalty on constraint violation makes the augmented Lagrangian non-separable (the x-update cannot be decomposed)

The solution: **Alternating Direction Method of Multipliers**

- Good robustness of method of multipliers
- Supports decomposition
- Can be considered as either "robust dual decomposition" or "decomposable method of multipliers"

Alternating Direction Method of Multipliers

Assume two sets of variables with a separable objective function where f and g are convex:

$$\begin{array}{ll} \text{minimize} & f(x) + g(z) \\ \text{subject to} & Ax + Bz = c \end{array}$$

and the following augmented Lagrangian

$$L_\rho(x, z, y) = f(x) + g(z) + y^T(Ax + Bz - c) + \frac{\rho}{2}\|Ax + Bz - c\|_2^2.$$

Then the ADMM method consists of the following steps:

$$\begin{array}{ll} x^{k+1} := \arg\min_x L_\rho(x, z^k, y^k) & x\text{-minimization} \\ z^{k+1} := \arg\min_z L_\rho(x^{k+1}, z, y^k) & z\text{-minimization} \\ y^{k+1} := y^k + \rho(Ax^{k+1} + Bz^{k+1} - c) & \text{dual update} \end{array}$$

Alternating Direction Method of Multipliers

ADMM performs a Gauss-Seidel iteration over the two variables. The method reduces to the method of multipliers if we minimized over x and z jointly. Since we minimize over x with z fixed (and vice versa) the problem is split into two.

Recall:

$$\text{(Jacobi)} \quad x_i^{k+1} = \arg\min_{x_i} f(x_1^k, \ldots, x_{i-1}^k, x_i, x_{i+1}^k, \ldots, x_N^k)$$

$$\text{(Gauss-Seidel)} \quad x_i^{k+1} = \arg\min_{x_i} f(x_1^{k+1}, \ldots, x_{i-1}^{k+1}, x_i, x_{i+1}^k, \ldots, x_N^k)$$

Recent extension to asynchronous setting using randomized Gauss-Seidel of Douglas-Rachford splitting (CDC'13).

ADMM and Optimality Conditions

Recall the optimality conditions for differentiable f and g:

- Primal feasibility: $Ax + Bz - c = 0$
- Dual feasibility: $\nabla f(x) + A^T y = 0, \quad \nabla g(z) + B^T y = 0$

(For non-differentiable case, zero should be in the subdifferential.)
Based on the second step, z^{k+1} minimizes $L_\rho(x^{k+1}, z, y^k)$ thus

$$\begin{aligned} 0 &= \nabla g(z^{k+1}) + B^T y^k + \rho B^T (Ax^{k+1} + Bz^{k+1} - c) \\ &= \nabla g(z^{k+1}) + B^T y^{k+1}, \end{aligned}$$

therefore with the ADMM dual update the triple $(x^{k+1}, z^{k+1}, y^{k+1})$ satisfies the second dual feasibility condition, however primal feasibility and first dual feasibility is only achieved as $k \to \infty$.
The primal residual $r^{k+1} = Ax^{k+1} + Bz^{k+1} - c$ and the dual residual $s^{k+1} = \rho A^T B(z^{k+1} - z^k)$ can be bounded with some desired accuracy to get stopping criteria.

ADMM with Scaled Dual Variables

ADMM can also be expressed in a scaled dual form, where the dual variables are scaled $\left(u^k = \frac{1}{\rho} y^k\right)$ by combining the linear and quadratic terms in the augmented Lagrangian (completion of squares).

$$\begin{aligned} L_\rho(x,z,y) &= f(x) + g(z) + y^T(Ax + Bz - c) + \frac{\rho}{2}\|Ax + Bz - c\|_2^2 \\ &= f(x) + g(z) + \frac{\rho}{2}\|Ax + Bz - c + u\|_2^2 + \text{const.} \end{aligned}$$

$$\begin{aligned} x^{k+1} &:= \arg\min_x \left(f(x) + \frac{\rho}{2}\|Ax + Bz^k - c + u^k\|_2^2 \right) \\ z^{k+1} &:= \arg\min_z \left(g(z) + \frac{\rho}{2}\|Ax^{k+1} + Bz - c + u^k\|_2^2 \right) \\ u^{k+1} &:= u^k + (Ax^{k+1} + Bz^{k+1} - c) \end{aligned}$$

ADMM Properties

- Very few assumptions are needed for convergence
  (f, g convex, closed, proper, L_0 has a saddle point)
- Primal feasibility and optimality are achieved asymptotically
- Good results in a few tens of iterations, although slow to converge to very high accuracy: worst case complexity is $\mathcal{O}(1/\epsilon^2)$

ADMM and its variants are considered to be very robust algorithms for

- Large-scale optimization
  (machine learning, statistics, dynamic networks)
- Distributed optimization
  (cooperative computing agents with message passing)

ADMM Properties

ADMM can either give simple single-processor algorithms that can be competitive with state-of-the-art complex solvers, or it can be used to coordinate many processors, each solving a substantial problem on its own, in order to solve a very large problem.

ADMM is a meta-algorithm that coordinates existing solvers to solve problems of arbitrary size. The update steps can be implemented using an analytical solution, Newton's method, Conjugate Gradient, first-order methods, custom methods.

ADMM Related Algorithms

ADMM is the same as, or closely related to, many methods with other names, such as

- Operator splitting methods
  (Douglas, Peaceman, Rachford, Lions, Mercier, 1950s, 1979)
- Proximal point algorithm (Rockafellar 1976)
- Dykstra's alternating projections algorithm (1983)
- Spingarn's method of partial inverses (1985)
- Rockafellar-Wets progressive hedging (1991)
- Proximal methods (Rockafellar, many others, 1976 to present)
- Bregman iterative methods (2008present)

Most of these are special cases of the proximal point algorithm.

Special Patterns / Structure in Problem Data

Notice that the x-update step requires minimizing $f(x) + \frac{\rho}{2}\|Ax - v\|_2^2$ with $v = Bz^k - c + u^k$, which remains constant during this update step. A similar observation holds for the z-update. This often leads to several special cases, that allow simplification of the update by exploiting structure, e.g.:

- Suppose f is block-separable

$$f(x) = f_1(x_1) + \cdots + f_N(x_N), \qquad x = (x_1, \ldots, x_N)$$

  and A^TA is block diagonal. Then the x-update splits up into N parallel updates of x_i.

- When $A = I$, the x-update becomes the so-called proximal operator $\mathbf{prox}_{f,\rho}(v)$. When f is the indicator function of the convex set $\mathcal{C}$, then the x-update becomes the projection of v onto $\mathcal{C}$.

ADMM for Constrained Convex Optimization

Consider the generic problem

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & x \in \mathcal{C} \end{array}$$

In order to translate this to ADMM form, take g as the indicator function of $\mathcal{C}$:

$$\begin{array}{ll} \text{minimize} & f(x) + g(z) \\ \text{subject to} & x - z = 0 \end{array}$$

The ADMM algorithm then becomes

$$\begin{aligned} x^{k+1} &:= \arg\min_x \left(f(x) + \frac{\rho}{2} \|x - z^k + u^k\|_2^2 \right) \\ z^{k+1} &:= \Pi_{\mathcal{C}} \left(x^{k+1} + u^k \right) \\ u^{k+1} &:= u^k + x^{k+1} - z^{k+1} \end{aligned}$$

ADMM for Consensus Optimization

The problem to be solved contains N objective terms:

$$\text{minimize} \quad \sum_{i=1}^{N} f_i(x)$$

This can be expressed in ADMM form as:

$$\begin{array}{ll} \text{minimize} & \sum_{i=1}^{N} f_i(x_i) \\ \text{subject to} & x_i - z = 0 \end{array}$$

where

- the x_i are local variables
- z is the global variable
- $x_i - z = 0$ are consistency or consensus constraints
- we can add regularization using an extra $g(z)$ term

ADMM for Consensus Optimization

The augmented Lagrangian becomes:

$$L_\rho(x, z, y) = \sum_{i=1}^{N} \left(f_i(x_i) + y_i^T (x_i - z) + \frac{\rho}{2} \|x_i - z\|_2^2 \right)$$

and the ADMM algorithm can be written as

$$\begin{aligned}
x_i^{k+1} &:= \operatorname*{arg\,min}_{x_i} \left(f_i(x_i) + y_i^{kT} (x_i - z^k) + \frac{\rho}{2} \|x_i - z^k\|_2^2 \right) \\
z^{k+1} &:= \frac{1}{N} \sum_{i=1}^{N} \left(x_i^{k+1} + \frac{1}{\rho} y_i^k \right) \\
y_i^{k+1} &:= y_i^k + \rho(x_i^{k+1} - z^{k+1})
\end{aligned}$$

When regularization is used, the averaging in the z-update is followed by $\mathbf{prox}_{g,\rho}$.

ADMM for Consensus Optimization

This algorithm can be simplified further. By averaging the y-updates and substituting the result to the z-update, we can write the ADMM as

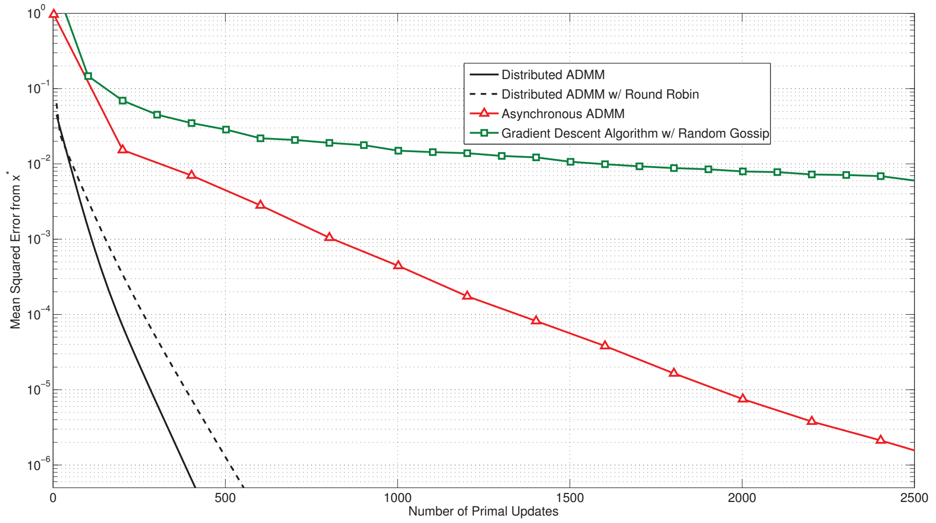
$$\begin{aligned} x_i^{k+1} &:= \arg\min_{x_i} \left(f_i(x_i) + y_i^{kT}(x_i - \bar{x}^k) + \frac{\rho}{2}\|x_i - \bar{x}^k\|_2^2 \right) \\ y_i^{k+1} &:= y_i^k + \rho(x_i^{k+1} - \bar{x}^{k+1}) \end{aligned}$$

where $\bar{x}^k = \frac{1}{N}\sum_{i=1}^N x_i^k$. The dual variables are separately updated to drive the variables into consensus, and quadratic regularization helps pull the variables toward their average value while still attempting to minimize each local f_i.

We can interpret consensus ADMM as a method for solving problems where objective and constraints are distributed across multiple processors. Each one has to handle its own objective and constraint term, plus a quadratic term (updated in each iteration). The linear parts of the quadratic terms are updated in such a way that the variables converge to a common value (the solution of the full problem).

Open Problems

- How to choose ρ in the augmented Lagrangian?
- How to scale equality constraints?
- How to use Nesterov's accelerated gradient methods within ADMM?
- Convergence rate needs better characterizations
  - without strong convexity assumption of objective
  - when extending to multiple sets of variables
  - accelerated and asynchronous versions, etc.



Some References

Augmented Lagrangian algorithm

- D.P. Bertsekas, Constrained Optimization and Lagrange Multiplier Methods (1982)

Alternating direction method of multipliers and related algorithms

- S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein, Distributed optimization and statistical learning via the alternating direction method of multipliers (2010)
- D. Goldfarb, S. Ma, K. Scheinberg, Fast alternating linearization methods for minimizing the sum of two convex functions, arxiv.org/abs/0912.4571 (2010)
- T. Goldstein and S. Osher, The split Bregman method for L1-regularized problems, SIAM J. Imag. Sciences (2009)
- M. Hong and Z-Q. Luo, On the linear convergence of the alternating direction method of multipliers, arxiv.org/abs/1208.3922v3 (2013)
- F. Iutzeler, P. Bianchi, P. Ciblat and W. Hachem, Asynchronous distributed optimization using a randomized alternating direction method of multipliers, CDC (2013)

Some References

Proximal point algorithm and fast proximal point algorithm

- O. Güler, On the convergence of the proximal point algorithm for convex minimization, SIAM J. Control and Optimization (1991)
- O. Güler, New proximal point algorithms for convex minimization, SIOPT (1992)
- O. Güler, Augmented Lagrangian algorithm for linear programming, JOTA (1992)
- P.L. Combettes and J-C. Pesquet, Proximal Splitting Methods in Signal Processing, arxiv.org/abs/0912.3522v4 (2010)

Proximal Gradient Methods

- Consider the problem

$$\text{minimize} \quad f(x)$$

with $f(x)$ continuously differentiable.

- Recast in the equivalent form

$$\begin{array}{ll} \text{minimize} & f(x) + \frac{1}{2c}\|x-y\|_2^2 \\ \text{subject to} & x = y \end{array}$$

- For $y = x^{k-1}$:

$$x^k = \arg\min_x \left\{ f(x) + \frac{1}{2c}\|x - x^{k-1}\|_2^2 \right\} \Rightarrow$$

$$x^k = x^{k-1} - c\nabla f(x^{k-1})$$

and x^k is the **proximal map** associated with $f(x)$.

Proximal Gradient Methods

- Reduces to normal gradient iteration for $f(x)$ linear.
- Can handle constraints and non-differentiability.
- Has convergence rate $\mathcal{O}(\frac{1}{k})$ at best.

Accelerated version: Apply proximal map at point y^k - linear combination of $\{x^{k-1}, x^{k-2}\}$ (FISTA, Beck, Teboulle 2009).

- Similar methods were proposed earlier by Nesterov (1983, 1988).
- Best achievable convergence rate is attained at $\mathcal{O}(\frac{1}{k^2})$.

Challenge:

- Could the ADMM iterations be written as proximal gradient iterations?
- If so, could we use the accelerated techniques in ADMM?

Networked and Distributed Control Systems

Tamás Keviczky

Delft Center for Systems and Control
Delft University of Technology
Delft, The Netherlands

Lecture 8
2020

Outline of Lectures 6-8

- Basics of convex optimization
  - Using first-order methods
  - Brief duality theory (see on extra slides)
- Distributed optimization and decomposition methods
  - Basic techniques and ideas, proofs in references
- A brief lecture on ADMM (see on extra slides)
- Incremental subgradient methods
  - Example: finite-time optimal rendezvous
- Relaxing communication constraints
  - Basic consensus models and results (see Lecture 3)
  - Combined consensus/subgradient methods
- Summary

A Brief Lecture on ADMM

See on extra slides...

Decentralized Computation for Subgradient Methods

So far we have seen examples for these types of problems with non-differentiable $f(\cdot)$ arising from decomposition techniques:

$$\begin{aligned} \underset{\theta}{\text{minimize}} \quad & f(\theta) = \sum_{i=1}^{N} f^i(\theta) \\ \text{subject to} \quad & \theta \in \Theta \end{aligned}$$

We know they can be solved by the subgradient method:

Subgradient iteration

$$\theta_{k+1} = \mathcal{P}_\Theta \left[\theta_k - \alpha_k \sum_{i=1}^{N} g_k^i \right]$$

where g_k^i is a subgradient of f^i at θ_k:

$$f^i(\theta) \geq f^i(\theta_k) + (g_k^i)^\intercal(\theta - \theta_k)), \quad \forall \theta \in \mathbb{R}^M.$$

Incremental Subgradient Methods

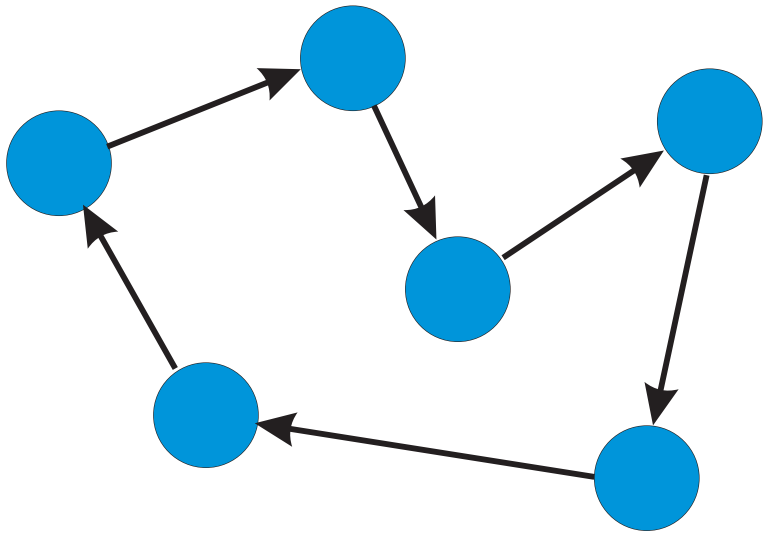
Implementing master problem (signaling, negotiation, etc.) in a decentralized way.

- Assumptions
    - optimal parameter value exists
    - norm of all subgradients are bounded
    - square summable stepsize

Incremental Subgradient Methods

Implementing master problem (signaling, negotiation, etc.) in a decentralized way.

- Assumptions
  - optimal parameter value exists
  - norm of all subgradients are bounded
  - square summable stepsize
- Operation
  - circulate a parameter estimate through the network (cyclic or random sequence)
  - make small adjustments to the estimate at each node based on its local data



Incremental Subgradient Methods

Implementing master problem (signaling, negotiation, etc.) in a decentralized way.

- Assumptions
  - optimal parameter value exists
  - norm of all subgradients are bounded
  - square summable stepsize
- Operation
  - circulate a parameter estimate through the network (cyclic or random sequence)
  - make small adjustments to the estimate at each node based on its local data
- Advantages
  - simple to implement
  - applicable to a general class of problems
  - analyzable rate of convergence (required number of cycles)

[Nedic - Bertsekas]

Incremental Subgradient Methods

- Incremental version updates θ in a sequence of N steps (subiterations):

Incremental subgradient iterations

$$\vartheta_k^0 = \theta_k, \qquad \text{(initialize)}$$

$$\begin{aligned} \vartheta_k^i &= \mathcal{P}_\Theta\left[\vartheta_k^{i-1} - \alpha_k g_k^i\right], \\ g_k^i &\in \partial f^i(\vartheta_k^{i-1}), \quad i = 1, \ldots, N, \end{aligned} \qquad \text{(iterate)}$$

$$\theta_{k+1} = \vartheta_k^N, \qquad \text{(update)}$$

- Convergence can be faster than standard method.
- Calculations can be performed in a distributed way through sequential updates to θ.

Example: Finite-Time Optimal Rendezvous (FTOR)

- Consider N dynamic agents

$$\begin{aligned} x^i_{t+1} &= A^i x^i_t + B^i u^i_t, \\ y^i_t &= C^i x^i_t \end{aligned}$$

- Polyhedral constraints

$$x^i_t \in \mathcal{X}^i, \quad u^i_t \in \mathcal{U}^i, \quad t \geq 0$$

Finite-time rendezvous

$$\begin{aligned} y^i_{T+k} &= \theta, \quad \forall k \geq 0, \quad i = 1, \ldots, N, \\ u^i_{T+k} &= u^i_T, \quad \forall k \geq 0, \quad i = 1, \ldots, N \end{aligned}$$

FTOR Problem Formulation

- Each agent has nontrivial, constrained LTI dynamics
- The rendezvous point $\theta \in \Theta$ is not fixed a priori, but chosen optimally

Cost function associated with the *i*-th system

$$\begin{aligned} V^i\left(x_k^i, u_k^i, \theta\right) = &\left(x_k^i - x_e^i(\theta)\right)^\top Q^i\left(x_k^i - x_e^i(\theta)\right) \\ &+ \left(u_k^i - u_e^i(\theta)\right)^\top R^i\left(u_k^i - u_e^i(\theta)\right) \end{aligned}$$

FTOR Problem Formulation

Optimization problem

$$
\begin{aligned}
\min_{U,\theta} \quad & \sum_{i=1}^{N} \sum_{k=0}^{T-1} V^i\left(x_k^i, u_k^i, \theta\right) \\
\text{subject to} \quad & x_{k+1}^i = A^i x_k^i + B^i u_k^i, \\
& y_k^i = C^i x_k^i, \\
& x_k^i \in \mathcal{X}^i, \quad k = 1, \ldots, T, \\
& u_k^i \in \mathcal{U}^i, \quad k = 0, \ldots, T-1, \\
& y_T^i = \theta, x_T^i = x_e^i(\theta), \\
& x_0^i = x^i(0), \quad i = 1, \ldots, N, \\
& \theta \in \Theta
\end{aligned}
$$

Primal Decomposition of FTOR

- Eliminate control inputs $u^i = k^i(x^i, \theta)$

$$f^i(x^i, \theta) = \min_{U^i} \sum_{k=0}^{T-1} V^i\left(x_k^i, u_k^i, \theta\right)$$
$$\text{subject to} \;\; \text{constraints}, \quad k = 1, \ldots, T-1$$

- Express the optimization problem as

$$f^*(x) = \min_{\theta} \sum_{i=1}^{N} f^i(x^i, \theta)$$
$$\text{subject to} \;\; \theta \in \Theta$$

- This belongs to general problem class defined earlier

[Johansson et al, 2006]

Incremental Subgradient Solution of FTOR

Using primal decomposition, the FTOR can be expressed as

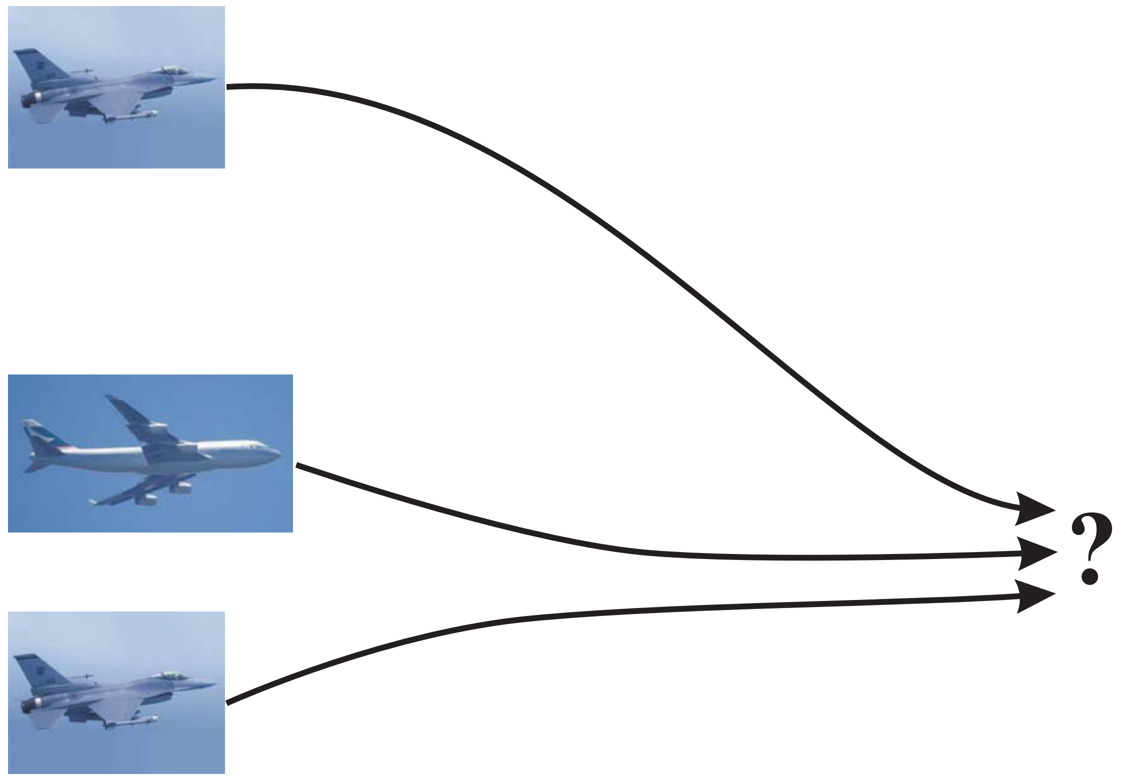
$$f^*(x) = \min_{\theta} \sum_{i=1}^{N} f^i(x^i, \theta)$$
$$\text{subject to} \quad \theta \in \Theta$$

Solution using cyclic incremental subgradient method

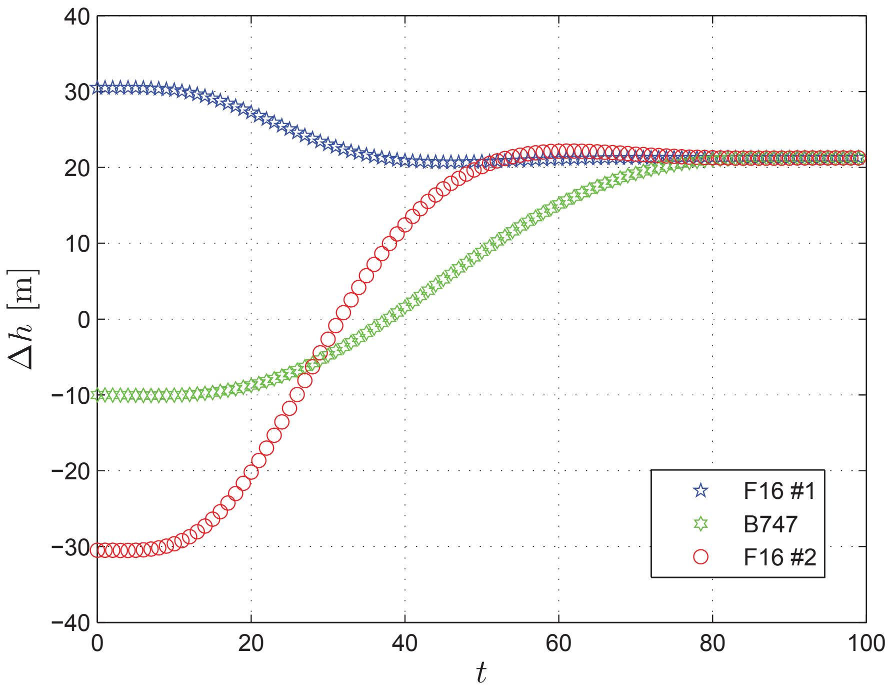
```
1. Initialize $\theta_0$ and $\alpha_0$, set $k = 0$, $h = 1$
2. $\alpha_h = \frac{\alpha_0}{h}$
3. for $i = 1$ to $N$ do
    (a) Compute a subgradient $\lambda_k^i$ for $f^i(\theta_k)$
    (b) $\theta_{k+1} = \mathcal{P}_\Theta \left[\theta_k - \alpha_h \lambda_k^i\right]$
    (c) $k = k + 1$
4. $h = h + 1$ go to step 2.
```

Numerical Example - Aerial Refueling Scenario

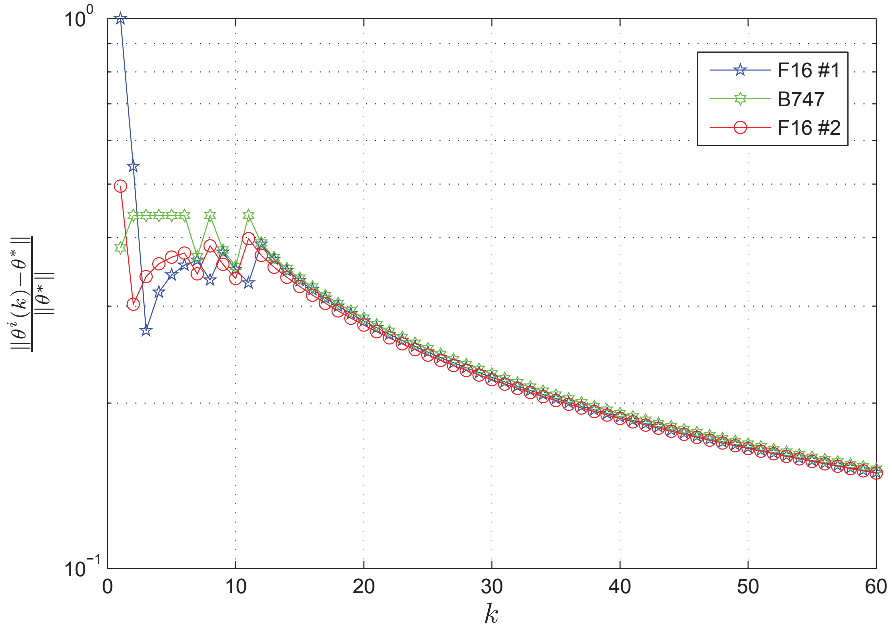
- Three aircraft need to rendezvous for refueling
- The rendezvous variable is altitude



Numerical Example - Aerial Refueling Scenario



Centralized solution

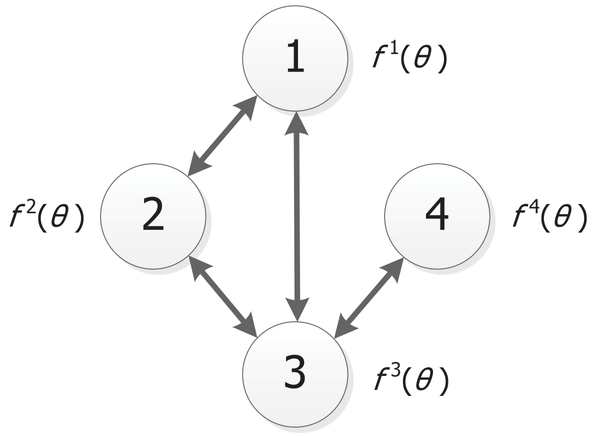


Convergence of local estimates

Generalized Problem Formulation

$$\underset{\theta}{\text{minimize}} \quad f(\theta) = \sum_{i=1}^{N} f^i(\theta)$$

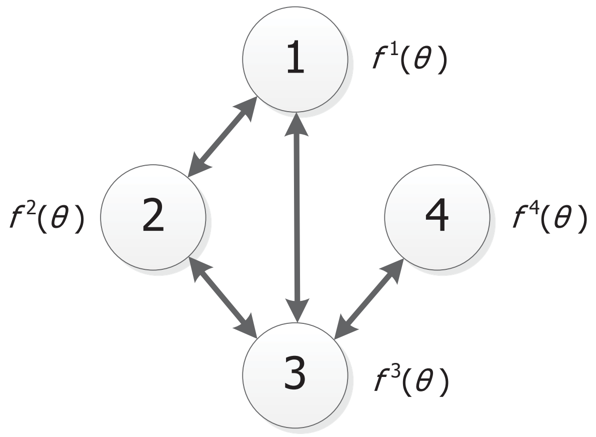
$$\text{subject to} \quad \theta \in \Theta,$$



- $f^i \; : \; \mathbb{R}^M \to \mathbb{R}$ nondifferentiable convex functions.
- $\Theta \subseteq \mathbb{R}^M$ nonempty, closed, and convex set.
- Computations should be performed in a distributed fashion.

Generalized Problem Formulation

$$\begin{aligned} \underset{\theta}{\text{minimize}} \quad & f(\theta) = \sum_{i=1}^{N} f^i(\theta) \\ \text{subject to} \quad & \theta \in \Theta, \end{aligned}$$



- $f^i : \mathbb{R}^M \to \mathbb{R}$ nondifferentiable convex functions.
- $\Theta \subseteq \mathbb{R}^M$ nonempty, closed, and convex set.
- Computations should be performed in a distributed fashion.
- Information exchange is only allowed through edges of an N-node undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$.

Generalization of Decomposition Methods to Several Nodes

- Primal decomposition: Needs a coordinator, hard to implement in this form
- Dual decomposition: Needs one consistency price per edge
- Incremental subgradient: Needs to pass token in cycle
- Towards a More Flexible Communication Scheme:
  - Avoid sequential (cyclic or random) incremental subgradient update scheme.
  - Networked incremental subgradient using random walk.
  - Combine subgradient updates with local consensus iterations and rely on convergence properties of consensus protocols.

Networked Incremental Subgradients

Incremental subgradient methods are not peer-to-peer, they need to pass an estimate in a ring, or to a random neighbor.

It is possible to devise an incremental subgradient method that only relies on peer-to-peer communication.

The estimate needs to make an unbiased random walk on the network, which can be done in a distributed fashion, and convergence can be shown in a stochastic sense.

Consensus/Subgradient Methods

Key trick for distributing dual decomposition:

$$\begin{array}{ll} \underset{\theta}{\text{minimize}} & \sum_{i=1}^{N} f^i(\theta) \\ \text{subject to} & \theta \in \Theta \end{array} \quad \Rightarrow \quad \begin{array}{ll} \underset{\theta}{\text{minimize}} & \sum_{i=1}^{N} f^i(\theta^i) \\ \text{subject to} & \theta^i \in \Theta \\ & \theta^i = \theta^j, \forall (i,j) \in \mathcal{E} \end{array}$$

With dual decomposition we relax consistency requirements between local versions of the decision variable.
An alternative idea is to “neglect and project”:

- Each node has local view of global decision variables
- Updates in direction of negative subgradient
- Coordinates with neighbors to achieve consistency (using consensus iterations)

Consensus in Multi-Agent Systems

Consensus

Agreement regarding a quantity of interest that depends on the state of all agents.

Consensus algorithm

Interaction rule specifying information exchange between an agent and it's neighbors in a network.
Objective:
Local variables of each agent converge to the same value.

Consensus in Multi-Agent Systems

Consensus

Agreement regarding a quantity of interest that depends on the state of all agents.

Consensus algorithm

Interaction rule specifying information exchange between an agent and it's neighbors in a network.
Objective:
Local variables of each agent converge to the same value.

- Consider a set of $N = \{1, \ldots, n\}$ agents that try to reach an agreement on a common scalar value (consensus).

Basic Consensus Models and Results

See Lecture 3 slides...

Consensus Example

- Assume agents have dynamics $\dot{x}_i = u_i$. Exchanging information based on graph Laplacian L:

$$\dot{x}_i = \sum_{j \in \mathcal{N}_i} (x_j(t) - x_i(t)) \quad \Rightarrow \quad \dot{x} = -Lx$$

- This is a distributed consensus algorithm. If the graph is undirected and connected, it converges to the average of the initial states.
- The consensus process here is nothing else, but gradient descent for

$$\phi(x) = \frac{1}{2} x^\mathsf{T} L x$$
$$\dot{x} = -\nabla \phi(x)$$

Combined Consensus/Subgradient Scheme

- Goal is to use agreement protocols to relax communication constraints in distributed optimization schemes.

Modified subgradient iterations

$$\theta^i_{k+1} = \mathcal{P}_\Theta \left[\sum_{j=1}^{N} [W^\varphi]_{ij} \left(\theta^j_k - \alpha_k g^j(\theta^j_k) \right) \right]$$

with $W = I - \varepsilon L(\mathcal{G})$ Perron matrix corresponding to the communication graph.

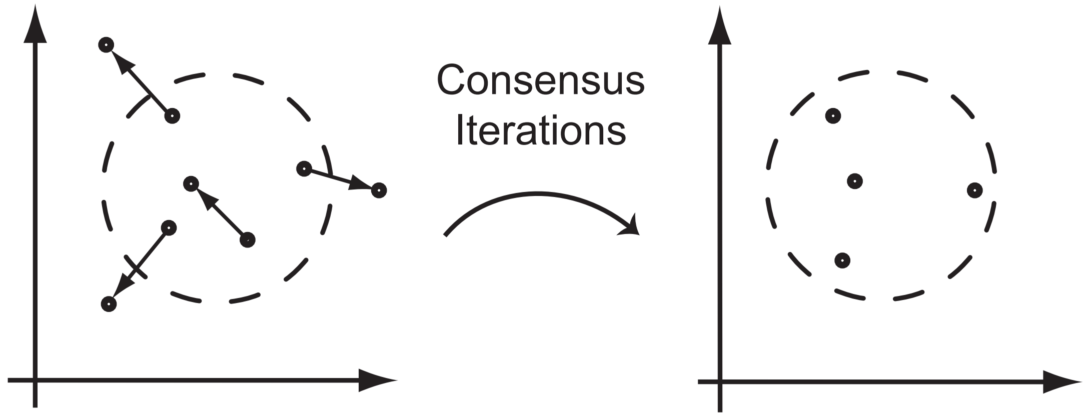
[Johansson et al., 2008]

Main Iterations of the Algorithm

1. Perform local subgradient update on local variable x^i.
2. Do φ consensus iterations with neighbors.
   (Can be interpreted as enforcing approximate equality constraints with neighboring variables.)
3. Repeat.

Convergence Analysis

- Establish a lower bound on the number of consensus steps φ that will ensure that the local variables will remain in a ball of radius β of their average, from one iteration to the next.
- Use approximate subgradient at the average value to account for different local subgradients.



Convergence Result

Theorem

Under appropriate assumptions, the sequence $\{\theta_k^1, \dots, \theta_k^N\}_{k=0}^{\infty}$ generated by the combined SG/consensus update with φ consensus iterations and $\left\|\theta_0^i - \bar{\theta}_0\right\| \leq \beta$, converges to a neighborhood of the optimal point:

$$\liminf_{k \to \infty} f(\theta_k^i) \leq f^\star + \frac{\alpha N C^2}{2} + 3NC\beta, \ \forall i = 1, \dots, N.$$

Remarks

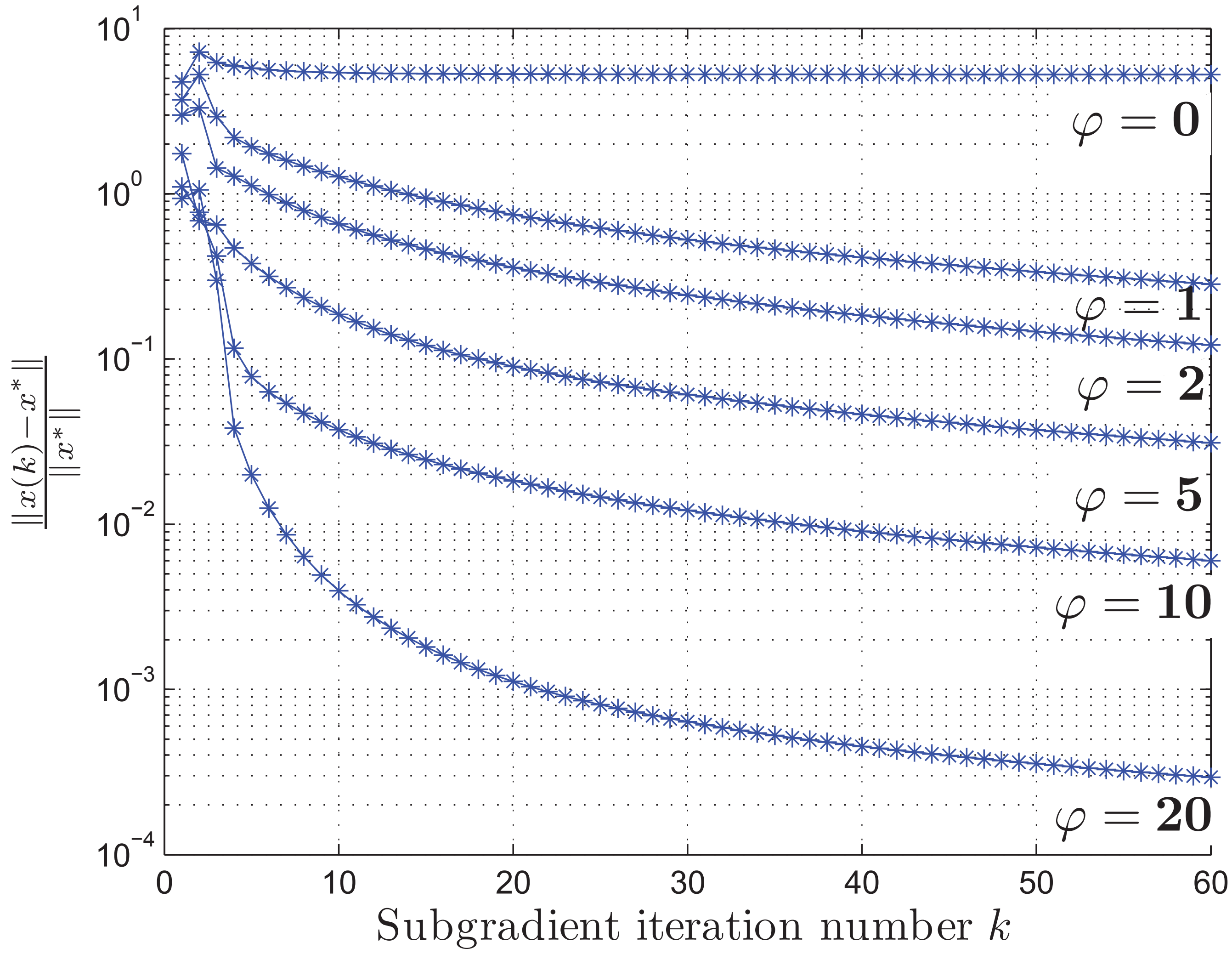
- We can get $\liminf_{k\to\infty} f(\theta_k^i)$ to be arbitrarily close to $f^\star$, by choosing the constants α and β arbitrarily small.
- Note that the number of required consensus negotiations, φ, to reach a fixed β is fixed, i.e., independent of k.

Illustration - Numerical Example

- Finite-time optimal rendezvous problem with three double integrator agents.
- Consensus matrix:

$$W = \begin{pmatrix} 0.75 & 0.25 & 0 \\ 0.25 & 0.5 & 0.25 \\ 0 & 0.25 & 0.75 \end{pmatrix}$$

Illustration - Numerical Example



Summary

- Motivation for distributed optimization
  - From parallel computing to distributed decision-making
- A model for networked optimization
  - Local and global variables and constraints (possibly coupled)
  - Local objective functions (possible coupled)
  - Graph models information and communication constraints
- The basic tools and techniques
  - Complicating variables (objectives), coupling constraints
  - Primal/dual decomposition techniques
  - ADMM
  - Example for extensions using combined incremental subgradient/consensus scheme

Other Related Topics and Extensions

- More advanced network models
  - Asynchronous communication
  - Communication losses and delays, quantization
  - Stochastically time-varying network parameters
  - Time-varying topology
- Stability and performance analysis with advanced network models
  - Lyapunov function construction
  - Continuous-time analysis
  - Convergence-rate analysis
- Coordination cost
  - Communication overhead vs convergence rate
  - Non-cooperative case
- Communication protocol issues
  - Beyond nearest neighbor communications
  - Gossiping vs tree-based aggregation
- Non-convex optimization, tricks for uncovering convexity, structure

Acknowledgements

- M. Johansson
- B. Johansson
- K.H. Johansson
- M. Cao

References

- R.T. Rockafellar, Convex Analysis, Princeton University Press, 1996.
- D.P. Bertsekas, Nonlinear Programming: 2nd Edition, Athena Scientific, 1999.
- S. Boyd and L. Vandenberghe, Convex Optimization, Cambridge University Press. http://www.stanford.edu/ boyd/cvxbook/
- D.P. Bertsekas and J.N. Tsitsiklis, Parallel and Distributed Computation: Numerical Methods, Athena Scientific, 1997. http://athenasc.com/pdcbook.html
- D.P. Bertsekas, A. Nedic, and A.E. Ozdaglar, Convex Analysis and Optimization, Athena Scientific, 2003.

References

- M. Johansson, Distributed optimization - intensive course, KTH Stockholm. http://www.ee.kth.se/∼mikaelj/dopt/
- B. Johansson, On Distributed Optimization in Networked Systems, PhD Thesis, KTH Stockholm, 2008.
- B. Johansson, T. Keviczky, M. Johansson, and K.H. Johansson, "Subgradient Methods and Consensus Algorithms for Solving Convex Optimization Problems," 47th IEEE Conference on Decision and Control, 2008, pp. 4185–4190.

Networked and Distributed Control Systems

Tamás Keviczky

Delft Center for Systems and Control
Delft University of Technology
Delft, The Netherlands

Lecture 9
2020

Lecture Outline

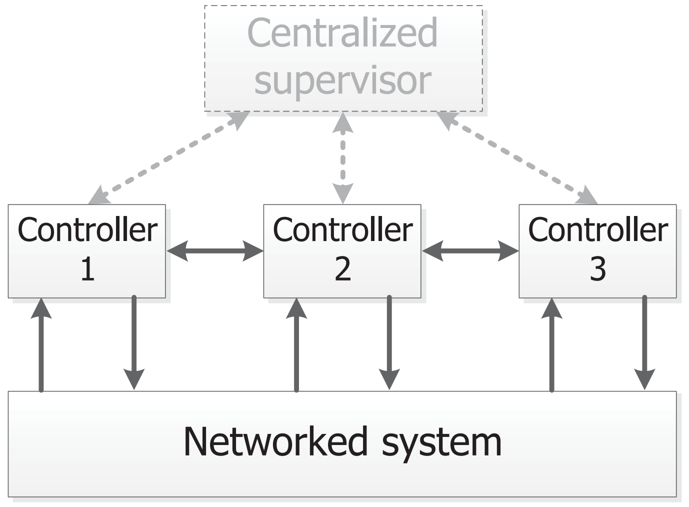
- Centralized vs distributed control
- Online optimization-based multi-agent coordination
  - Bottom-up approach, examples, DMPC
  - Top-down approach
- Typical problem classes for top-down approach
- Distributed model predictive consensus example
- Desiderata for decomposition and coordination mechanisms
- Summary

Centralized vs Distributed Control

- Centralized control for large-scale networked systems:
  - global model needed
  - computational complexity not scalable
  - maintenance can be hard
  - single controller may mean increased vulnerability
  - mature theory
- Distributed control for large-scale networked systems:
  - local models
  - parallelizable computation of limited-size control problems
  - local measurements
  - maintenance/upgrades could be localized
  - global properties harder to ensure/analyze
    (stability, constraint fulfillment, performance)

Typical Distributed / Hierarchical Architecture

- Measure/estimate local states
- Compute control actions locally
- Exchange decisions with neighbors, possibly iterating on local computations
- Apply the current command input to local actuators
- Possibly interact with upper level of decision making (hierarchical control)



Main questions

- How to guarantee global closed-loop stability?
- How to ensure feasibility of global constraints?
- Loss of performance w.r.t. centralized control?

Online Optimization-Based Multi-Agent Coordination

Motivation, key aspects:

- Distributed control and estimation in networks of systems
- Information and processing power is distributed among cooperating agents
- Global objective through local computations and interaction
- Design is local (on-line) as opposed to global (off-line)

Bottom-up approach

Formulate simple local optimization problems that lead to desired global behavior.

Top-down approach

Formulate global optimization problem and solve by decomposition, distributed optimization.

Bottom-Up Approach

1. Formulate an optimization problem for each agent that reflects the global objectives.

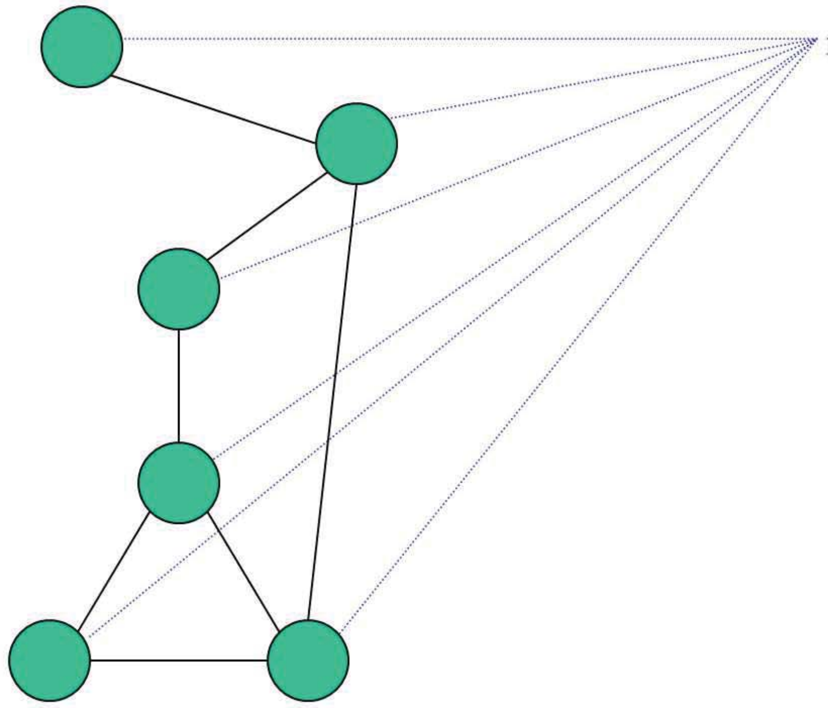
Bottom-Up Approach

1. Formulate an optimization problem for each agent that reflects the global objectives.
2. Incorporate local knowledge, exchange information locally. (cost function, models, constraints, neighboring states or measurements)

Bottom-Up Approach

1. Formulate an optimization problem for each agent that reflects the global objectives.
2. Incorporate local knowledge, exchange information locally. (cost function, models, constraints, neighboring states or measurements)
3. Solve and implement repeatedly. (e.g. receding horizon implementation)

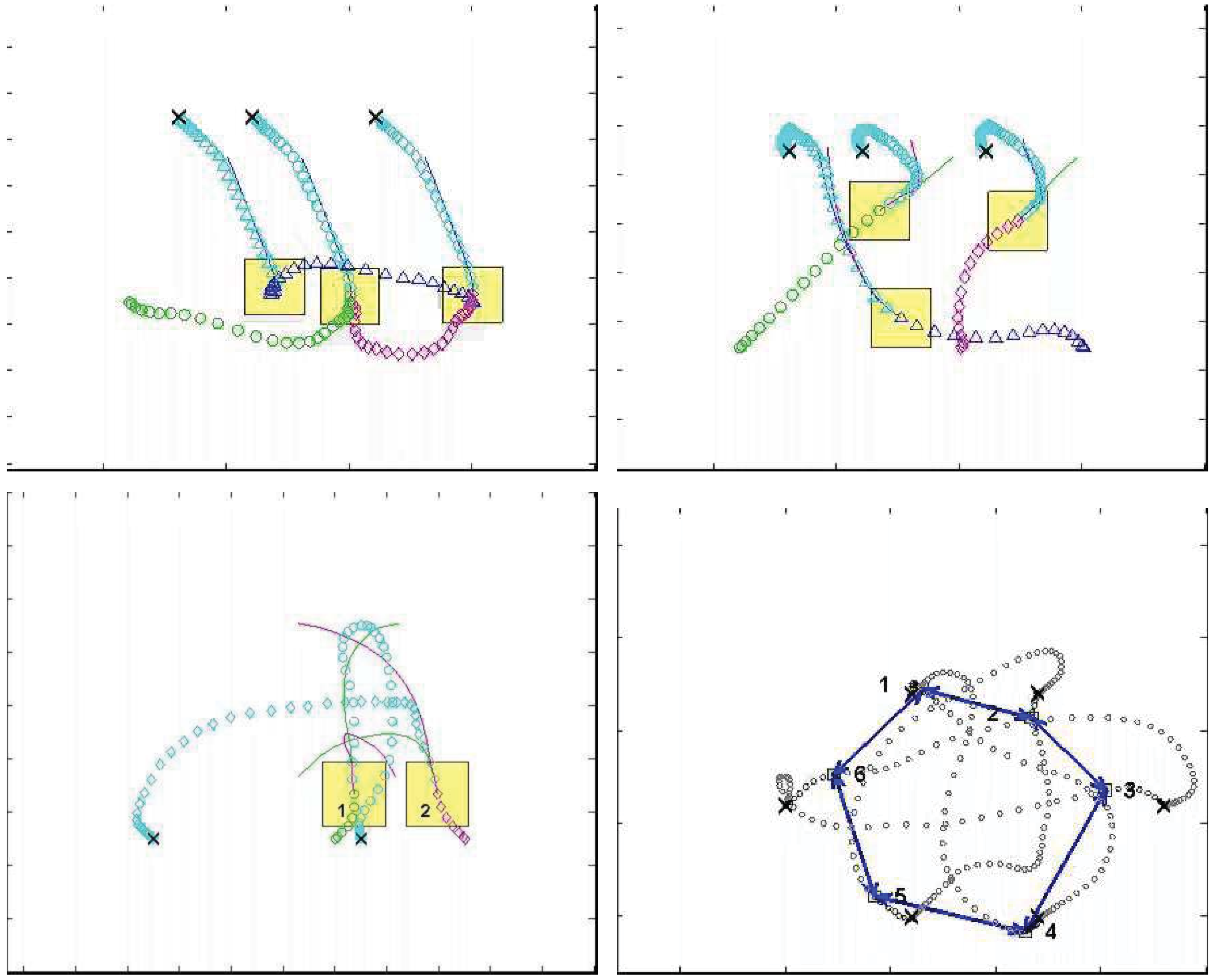
Bottom-Up Approach (Example MPC Formulation)



$$\min_{\tilde{U}_t^i} \sum_{k=0}^{N-1} \overbrace{l^i(x^i_{k,t}, u^i_{k,t}, \tilde{x}^i_{k,t}, \tilde{u}^i_{k,t}) + l^i_N(x^i_{N,t}, \tilde{x}^i_{N,t})}^{\text{Local performance index}}$$

$$\text{subj. to} \left\{ \begin{array}{l}
\left. \begin{array}{l}
x^i_{k+1,t} = f^i(x^i_{k,t}, u^i_{k,t}), \;\; k \geq 0 \\
x^i_{k,t} \in \mathcal{X}^i, \;\; u^i_{k,t} \in \mathcal{U}^i, \\
\qquad k = 1, \ldots, N-1
\end{array} \right\} \begin{array}{l}\text{Own model} \\ \text{and constraints}\end{array} \\
\left. \begin{array}{l}
x^j_{k+1,t} = f^j(x^j_{k,t}, u^j_{k,t}), \\
\qquad (j,i) \in \mathcal{A}, \;\; k \geq 0 \\
x^j_{k,t} \in \mathcal{X}^j, \;\; u^j_{k,t} \in \mathcal{U}^j, \\
\qquad (j,i) \in \mathcal{A}, \\
\qquad k = 1, \ldots, N-1
\end{array} \right\} \begin{array}{l}\text{Neighbors' model} \\ \text{and constraints}\end{array} \\
\left. \begin{array}{l}
g^{i,j}(x^i_{k,t}, u^i_{k,t}, x^j_{k,t}, u^j_{k,t}) \leq 0, \\
\qquad (i,j) \in \mathcal{A}, \\
\qquad k = 1, \ldots, N-1
\end{array} \right\} \begin{array}{l}\text{Interaction constr.} \\ \text{with neighbors,}\end{array} \\
\left. \begin{array}{l}
g^{q,r}(x^q_{k,t}, u^q_{k,t}, x^r_{k,t}, u^r_{k,t}) \leq 0, \\
\qquad (q,i) \in \mathcal{A}, \;\; (r,i) \in \mathcal{A}, \\
\qquad k = 1, \ldots, N-1
\end{array} \right\} \text{between neighbors} \\
x^i_{N,t} \in \mathcal{X}^i_f, \;\; x^j_{N,t} \in \mathcal{X}^j_f, (i,j) \in \mathcal{A} \\
x^i_{0,t} = x^i_t, \;\; \tilde{x}^i_{0,t} = \tilde{x}^i_t,
\end{array} \right.$$

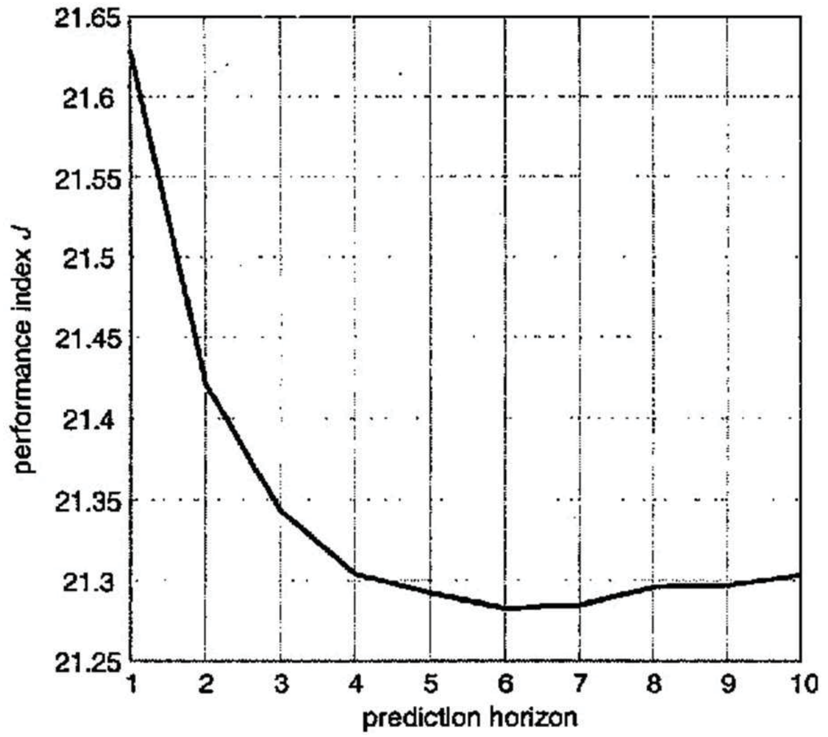
Bottom-Up Approach (Examples 1)



[Keviczky et al, 2006], [Borrelli et al, 2005]

Interesting Observations

- Distributed MPC approach provides feasible solutions. (In many cases comparable to the centralized one.)
- Horizon length selection is less trivial in the distributed case. (Trade-off curve is different from classical MPC, increasing horizon length can lead to infeasibility.)
- Influencing quality of solutions and feasibility by changing weight on cost terms.



[Jia - Krogh, 2001]

Bottom-Up Approach (Example 2)

Model

Low-level

- Highly nonlinear, constrained model
- Aero look-up table with wind tunnel data
- Inputs are vane deflections and throttle

High-Level

- MIMO constrained, piecewise linear models
- Inputs are north, east, altitude references

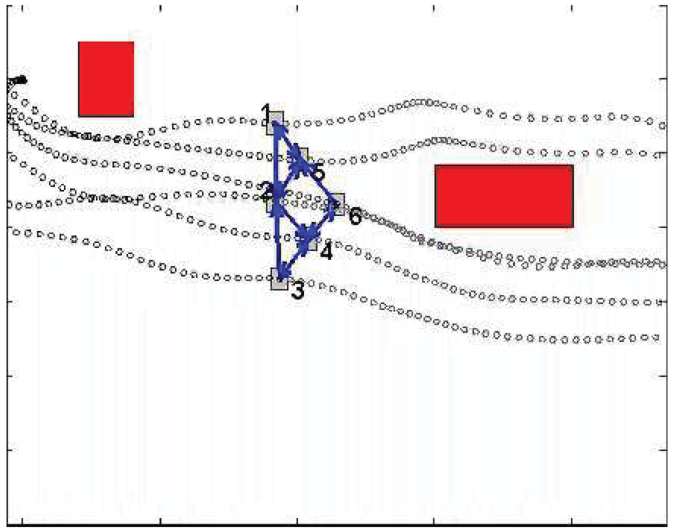
Constraints

- Speed and acceleration constraints
- Collision avoidance constraints (non-convex)

Bottom-Up Approach (Example 2)

Objectives and challenges

- Autonomous arrangement, large and tight formations
- Complex maneuvering challenges
- Perform obstacle avoidance
- Maintain relative positions
- Maintain safe distances
- Achieve a desired target condition (e.g. position, common velocity)
- Changing information exchange topology



[Keviczky et al, 2008]

Distributed MPC Bottom-Up Approaches

Many different approaches exist (assumptions, models, constraints)

subsystem models	constraints	intersample iterations	broadcast prediction	state constraints	stability constraints	authors
coupled	local inputs	no	no	no	none	Alessio, Barcelli, Bemporad
coupled	local inputs	yes	no	no	none	Venkat, Rawlings, Wright
coupled	local inputs	yes	yes	no	none	Mercangöz, Doyle
decoupled	local inputs	no	yes	yes	compatibility	Dunbar, Murray
decoupled	local states and inputs	no	no	yes	none	Keviczky, Borrelli, Balas
coupled	local states	no	yes	yes	contractive	Jia, Krogh
coupled, nonlinear	local inputs	no	no	no	contractive	Magni, Scattolini

You could easily think up a version (at least the formulation) for your problem...

- Typical stability/feasibility issues due to prediction mismatch between subsystems.
- Local MPC stability does not guarantee global stability.
- Typical Lyapunov function for stability analysis: sum of local value functions.

Typical Issues

These practical results are attractive, yet...

- decomposition is mostly ad-hoc

Typical Issues

These practical results are attractive, yet...

- decomposition is mostly ad-hoc
- no a priori guarantees on performance

Typical Issues

These practical results are attractive, yet...

- decomposition is mostly ad-hoc
- no a priori guarantees on performance
- analysis is performed a posteriori

Typical Issues

These practical results are attractive, yet...

- decomposition is mostly ad-hoc
- no a priori guarantees on performance
- analysis is performed a posteriori
- design assumptions often very conservative

Typical Issues

These practical results are attractive, yet...

- decomposition is mostly ad-hoc
- no a priori guarantees on performance
- analysis is performed a posteriori
- design assumptions often very conservative

Alternative approach

Development of systematic design tools to distribute and coordinate the global optimization problem among agents, instead of distributing the problem formulation.

Top-Down Approach

Joint optimization over subsystems for increased performance (compared to bottom-up approaches).

Typical decomposition methods for distributed optimization:

- Primal decomposition (lecture 7)
- Dual decomposition (lecture 7) (Lagrangian relaxation)
- Penalty function method (lecture 7)
- Proximal point method (Augmented Lagrangian)
- Auxiliary problem principle

From Distributed Control/Estimation to Distributed Optimization

Estimation and control problems related to complex networked systems can be posed as coupled optimization problems.

Interaction gives rise to coupling in cost or constraints with a specific algebraic structure
(sparse matrices, exploited in numerical algorithms).

Centralized, parallel, or sparse solvers. Various control topologies.
[Scattolini, 2011]

Distributed optimization algorithms come into the picture when communication is not all-to-all, but along the edges of a graph.

Typical tradeoffs between: convergence rate/speed, amount of required communication, distributed computation architecture.

Typical Problem Classes

1. Decoupled cost, common decision variable

$$\begin{aligned}\underset{\theta}{\text{minimize}} \quad & \sum_{i=1}^{N} f^i(\theta) \\ \text{subject to} \quad & \theta \in \Theta\end{aligned}$$

2. Decoupled cost, coupled (but sparse) constraints

$$\begin{aligned}\underset{\theta^i,\dots,\theta^N}{\text{minimize}} \quad & \sum_{i=1}^{N} f^i(\theta^i) \\ \text{subject to} \quad & \theta^i \in \Theta^i, \quad h^i(\theta^j \mid j \in \mathcal{N}^i) = 0, \quad \forall i\end{aligned}$$

3. Coupled cost, decoupled constraints

$$\begin{aligned}\underset{\theta^i,\dots,\theta^N}{\text{minimize}} \quad & f(\theta^i, \dots, \theta^N) \\ \text{subject to} \quad & \theta^i \in \Theta^i\end{aligned}$$

Typical Problem Classes

1. Decoupled cost, common decision variable

$$\begin{aligned} \underset{\theta}{\text{minimize}} \quad & \sum_{i=1}^{N} f^i(\theta) \\ \text{subject to} \quad & \theta \in \Theta \end{aligned}$$

- Example: Moving Horizon Estimation (MHE)
- Each sensor measures some process variables, computes local estimate, exchanges with neighbors.
- Goal: all sensors asymptotically reach a reliable estimate of the overall system state.

Distributed Moving Horizon Estimation

Each node i in a sensor network solves a MHE problem and computes an estimate of the state, based on *neighboring* information:

$$
\begin{aligned}
\widehat{\phi}^*_{T,i} = \min_{\overline{x}_i, \{w_i(k)\}_{k=T-N}^{T-1}} \quad & \widehat{\phi}_{T,i}(\overline{x}_i; \{w_i(k)\}_{k=T-N}^{T-1}) \\
\text{subject to} \quad & x_i(k+1) = Ax_i(k) + w_i(k), \\
& y_i(k) = C_i x_i(k) + v_i(k) \\
& x_i(k) \in \mathcal{X}, \quad w_i(k) \in \mathcal{W}, \quad \forall k
\end{aligned}
$$

with objective function

$$
\begin{aligned}
\widehat{\phi}_{T,i}(\overline{x}_i; \{w_i(k)\}_{k=T-N}^{T-1}) = \sum_{k=T-N}^{T-1} \Big(v_i^\top(k) R^{-1} v_i(k) + w_i^\top(k) Q^{-1} w_i(k) \Big) + \\
+ (\overline{x}_i - \hat{x}_{i,c}(T-N))^\top \Pi_{T-N,i}^{-1} (\overline{x}_i - \hat{x}_{i,c}(T-N)) + \widehat{\phi}^*_{T-N,i}
\end{aligned}
$$

Distributed Moving Horizon Estimation

- The arrival cost embeds a *consensus term*

$$\hat{x}_{i,c}(T-N) = \sum_{j \in \mathcal{N}_i} k_{ij} \hat{x}_j(T-N)$$

  that is a weighted sum of the neighbors' estimates.
- The matrices $\Pi_{T-N,i}$ are calculated also using consensus:

$$\Pi_{T-N,i} = 2 \sum_{j \in \mathcal{N}_i} k_{ij}^2 \tilde{\Pi}_{T-N,j}$$

  where each matrix $\tilde{\Pi}_{T-N,i}$ is defined by a recursive Riccati equation.
- Sufficient conditions for convergence can be derived for interleaved consensus & moving horizon scheme.

[Farina et al, 2010]

Typical Problem Classes

1. Decoupled cost, common decision variable
   - Example: Moving Horizon Estimation (MHE)

2. Decoupled cost, coupled (but sparse) constraints
   - Example: Distributed finite-time optimal control problem with interacting subsystem dynamics (for MPC)
   - Can be solved e.g. by primal subgradient, dual subgradient, dual fast gradient, dual interior-point algorithms. [Necoara et al, 2011]

3. Coupled cost, decoupled constraints
   - Example: Cooperative control problem for dynamically decoupled systems
   - Can be solved in parallel with Jacobi or Gauss-Seidel algorithm:

$$\text{(Jacobi)} \quad \theta^i_{k+1} = \arg\min_{\theta^i \in \Theta^i} f(\theta^1_k, \ldots, \theta^{i-1}_k, \theta^i, \theta^{i+1}_k, \ldots, \theta^N_k)$$

$$\text{(Gauss-Seidel)} \quad \theta^i_{k+1} = \arg\min_{\theta^i \in \Theta^i} f(\theta^1_{k+1}, \ldots, \theta^{i-1}_{k+1}, \theta^i, \theta^{i+1}_k, \ldots, \theta^N_k)$$

Typical Problem Classes

1. Decoupled cost, common decision variable
2. Decoupled cost, coupled (but sparse) constraints
3. Coupled cost, decoupled constraints

We can go from 3. $\rightarrow$ 2. by introducing auxiliary variables and moving the coupling terms from cost to constraints.

Going from 2. $\rightarrow$ 3. is also possible, by moving the coupling constraints into the cost.

Recall Example: Finite-Time Optimal Rendezvous (FTOR)

- Consider N dynamic agents

$$\begin{aligned} x_{t+1}^i &= A^i x_t^i + B^i u_t^i, \\ y_t^i &= C^i x_t^i \end{aligned}$$

- Polyhedral constraints

$$x_t^i \in \mathcal{X}^i, \quad u_t^i \in \mathcal{U}^i, \quad t \geq 0$$

Finite-time rendezvous

$$\begin{aligned} y_{T+k}^i &= \theta, \quad \forall k \geq 0, \quad i = 1, \ldots, N, \\ u_{T+k}^i &= u_T^i, \quad \forall k \geq 0, \quad i = 1, \ldots, N \end{aligned}$$

FTOR Problem Formulation

- Each agent has nontrivial, constrained LTI dynamics
- The rendezvous point $\theta \in \Theta$ is not fixed a priori, but chosen optimally

Cost function associated with the *i*-th system

$$\begin{aligned} V^i\left(x_k^i, u_k^i, \theta\right) = {} & \left(x_k^i - x_e^i(\theta)\right)^\mathsf{T} Q^i \left(x_k^i - x_e^i(\theta)\right) \\ & + \left(u_k^i - u_e^i(\theta)\right)^\mathsf{T} R^i \left(u_k^i - u_e^i(\theta)\right) \end{aligned}$$

FTOR Problem Formulation

Optimization problem

$$
\begin{aligned}
\min_{U_t,\theta_t} \quad & \sum_{i=1}^{N} \sum_{k=0}^{T-1} V^i\left(x^i_{k,t}, u^i_{k,t}, \theta_t\right) \\
\text{subject to} \quad & x^i_{k+1,t} = A^i x^i_{k,t} + B^i u^i_{k,t}, \\
& y^i_{k,t} = C^i x^i_{k,t}, \\
& x^i_{k,t} \in \mathcal{X}^i, \quad k = 1, \ldots, T, \\
& u^i_{k,t} \in \mathcal{U}^i, \quad k = 0, \ldots, T-1, \\
& y^i_{T,t} = \theta_t, x^i_{T,t} = x^i_e(\theta_t), \\
& x^i_{0,t} = x^i_t, \quad i = 1, \ldots, N, \\
& \theta_t \in \Theta
\end{aligned}
$$

Primal Decomposition of FTOR

- Eliminate control inputs $u_t^i = k^i(x_t^i, \theta_t)$

$$f^i(x_t^i, \theta_t) = \min_{U_t^i} \sum_{k=0}^{T-1} V^i\left(x_{k,t}^i, u_{k,t}^i, \theta_t\right)$$
$$\text{subject to} \ \ \text{constraints}, \quad k = 1, \ldots, T-1$$

- Express the optimization problem as

$$f^*(x_t) = \min_{\theta_t} \sum_{i=1}^{N} f^i(x_t^i, \theta_t)$$
$$\text{subject to} \ \ \theta_t \in \Theta$$

- This belongs to general problem class 1. defined earlier

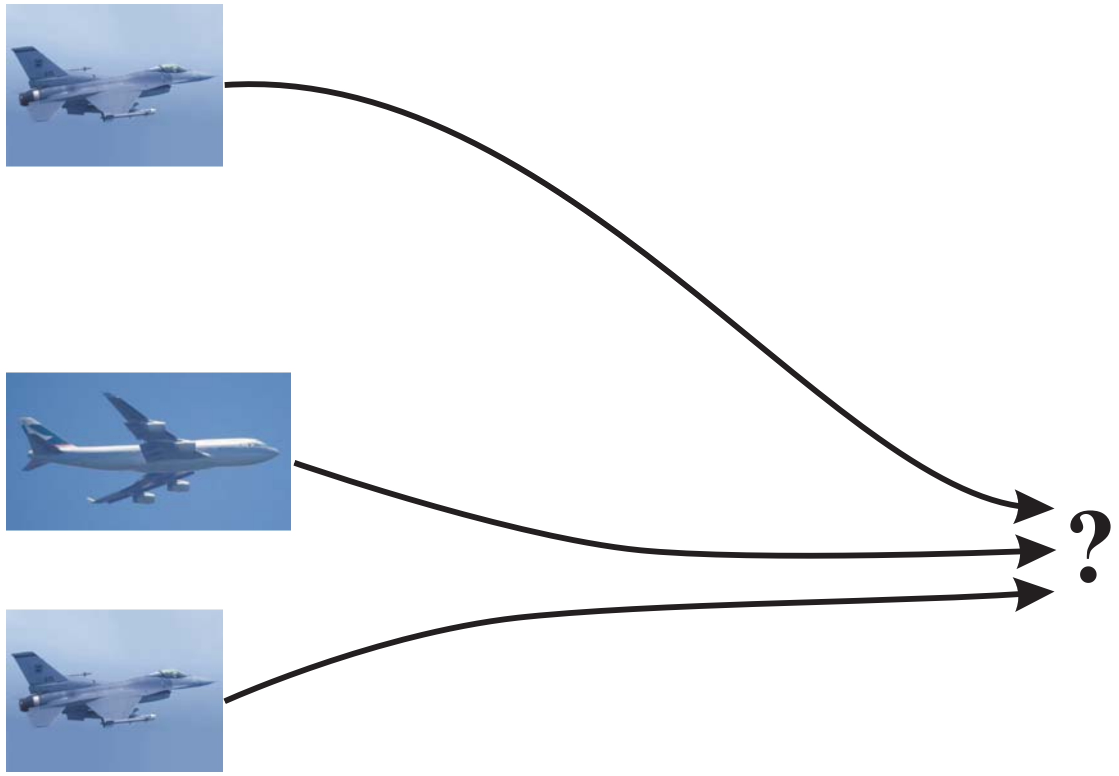
[Johansson et al, 2006]

Receding Horizon Implementation of FTOR

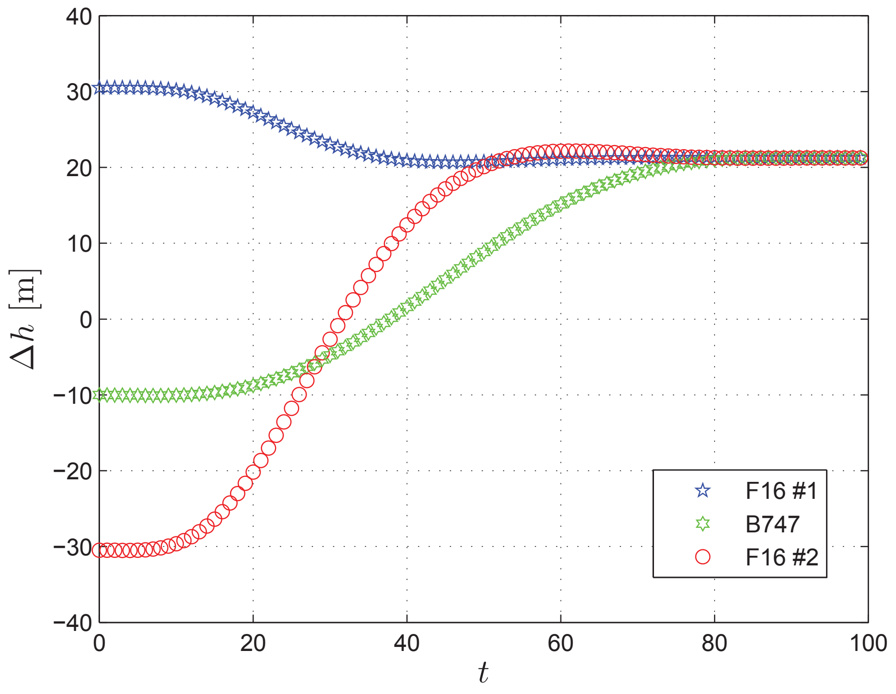
- Measure current state x_t^i
- Formulate Finite-Time Optimal Rendezvous Problem
- Solve using incremental subgradient method (distributed computations, sequential updates)
- Implement local solution corresponding to local rendezvous point

Numerical Example - Aerial Refueling Scenario

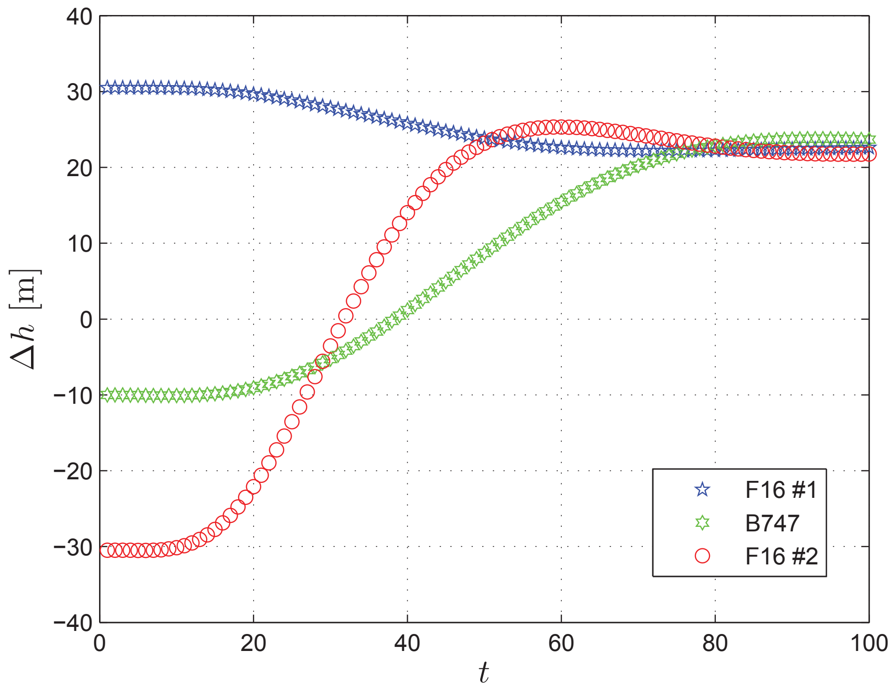
- Three aircraft need to rendezvous for refueling
- The rendezvous variable is altitude



Numerical Example - Aerial Refueling Scenario

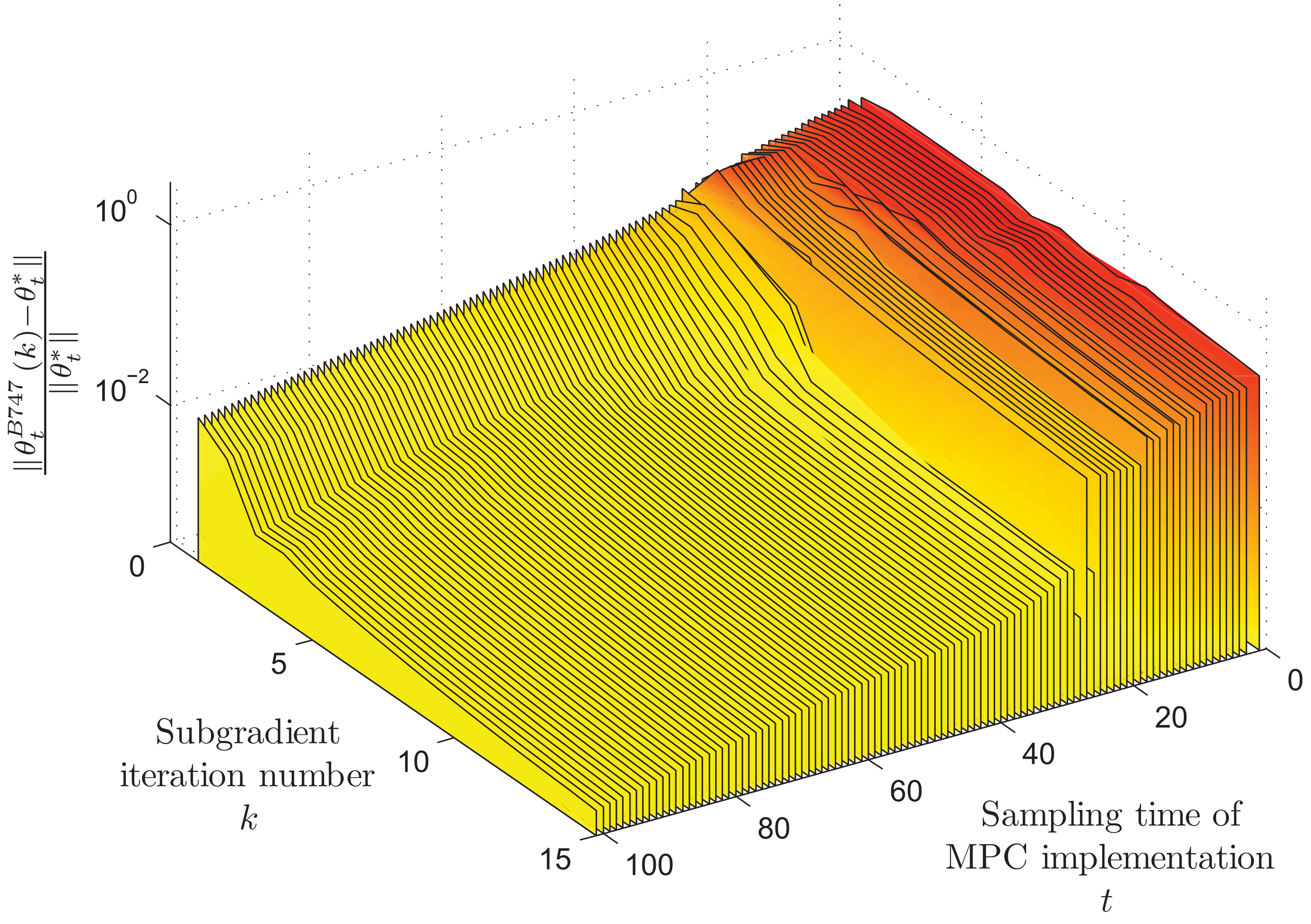


Centralized solution



MPC implementation with 15 subgradient iterations

Numerical Example - Aerial Refueling Scenario



[Keviczky-Johansson, 2008]

Important Questions

What happens if the negotiations get interrupted?

1. How can we still guarantee stability?
    - additional constraints
    - global cost function (cooperation)
    - using results on suboptimal MPC
2. How can we still guarantee feasibility?
    - ensuring feasibility of each iterate
    - challenging with coupling constraints (dual decomposition)

Decomposition and Coordination Mechanisms

Desiderata, Research Topics

What would need to be addressed for an efficient/implementable online optimization-based method in a distributed setting?

Decomposition and Coordination Mechanisms

Desiderata, Research Topics

What would need to be addressed for an efficient/implementable online optimization-based method in a distributed setting?

- Arbitrary connected (time-varying, delayed) comm. graph.

Decomposition and Coordination Mechanisms

Desiderata, Research Topics

What would need to be addressed for an efficient/implementable online optimization-based method in a distributed setting?

- Arbitrary connected (time-varying, delayed) comm. graph.
- Finite number of coordinating iterations (a priori tunable). (Rate analysis for real-time implementation.)

Decomposition and Coordination Mechanisms

Desiderata, Research Topics

What would need to be addressed for an efficient/implementable online optimization-based method in a distributed setting?

- Arbitrary connected (time-varying, delayed) comm. graph.
- Finite number of coordinating iterations (a priori tunable). (Rate analysis for real-time implementation.)
- Pre-specified distance from optimality. (Bounded suboptimality.)

Decomposition and Coordination Mechanisms

Desiderata, Research Topics

What would need to be addressed for an efficient/implementable online optimization-based method in a distributed setting?

- Arbitrary connected (time-varying, delayed) comm. graph.
- Finite number of coordinating iterations (a priori tunable). (Rate analysis for real-time implementation.)
- Pre-specified distance from optimality. (Bounded suboptimality.)
- Approximate primal solutions from dual decomposition. (Primal feasibility of each iterate.)

Decomposition and Coordination Mechanisms

Desiderata, Research Topics

What would need to be addressed for an efficient/implementable online optimization-based method in a distributed setting?

- Arbitrary connected (time-varying, delayed) comm. graph.
- Finite number of coordinating iterations (a priori tunable). (Rate analysis for real-time implementation.)
- Pre-specified distance from optimality. (Bounded suboptimality.)
- Approximate primal solutions from dual decomposition. (Primal feasibility of each iterate.)
- Distributed, on-line testing of termination criteria. (With a priori complexity bound.)

Decomposition and Coordination Mechanisms

Desiderata, Research Topics

What would need to be addressed for an efficient/implementable online optimization-based method in a distributed setting?

- Arbitrary connected (time-varying, delayed) comm. graph.
- Finite number of coordinating iterations (a priori tunable). (Rate analysis for real-time implementation.)
- Pre-specified distance from optimality. (Bounded suboptimality.)
- Approximate primal solutions from dual decomposition. (Primal feasibility of each iterate.)
- Distributed, on-line testing of termination criteria. (With a priori complexity bound.)
- Heterogeneous subsystems and constraints.

Decomposition and Coordination Mechanisms

Desiderata, Research Topics

What would need to be addressed for an efficient/implementable online optimization-based method in a distributed setting?

- Arbitrary connected (time-varying, delayed) comm. graph.
- Finite number of coordinating iterations (a priori tunable). (Rate analysis for real-time implementation.)
- Pre-specified distance from optimality. (Bounded suboptimality.)
- Approximate primal solutions from dual decomposition. (Primal feasibility of each iterate.)
- Distributed, on-line testing of termination criteria. (With a priori complexity bound.)
- Heterogeneous subsystems and constraints.
- Non-convex coupling constraints.

Summary

- Centralized vs distributed control aspects
- Bottom-up and top-down approaches for online optimization-based multi-agent coordination (including simulation examples, DMPC)
- Typical problem classes for top-down approach and relationship to distributed optimization / decomposition methods
- Distributed model predictive consensus example
- Specific features are needed for applicability in real-time receding horizon control and estimation.

Acknowledgements

- K.H. Johansson
- I. Necoara
- A. Bemporad
- J.B. Rawlings

References

- I. Necoara, V. Nedelcu, and I. Dumitrache, "Parallel and distributed optimization methods for estimation and control in networks," Journal of Process Control, 2011, Vol. 21, No. 5, pp. 756–766.
- J.B. Rawlings and D.Q. Mayne, Model Predictive Control: Theory and Design, Nob Hill Publishing, 2009.
- T. Keviczky, F. Borrelli, K.C. Fregene, D. Godbole, and G.J. Balas, "Decentralized Receding Horizon Control and Coordination of Autonomous Vehicle Formations," IEEE Transactions on Control Systems Technology, 2008, Vol. 16, No. 1, pp. 19–33.
- T. Keviczky, F. Borrelli and G.J. Balas, "Decentralized Receding Horizon Control for Large Scale Dynamically Decoupled Systems," Automatica, 2006, Vol. 42, No. 12, pp. 2105–2115.
- F. Borrelli, T. Keviczky, G.J. Balas, G. Stewart, K. Fregene and D. Godbole, "Hybrid Decentralized Control of Large Scale Systems," Hybrid Systems: Computation and Control, in ser. Lecture Notes in Computer Science, Springer, March 2005, Vol. 3414, pp. 168–183.
- M. Farina, G. Ferrari-Trecate, and R. Scattolini, "Distributed moving horizon estimation for linear constrained systems," IEEE Transactions on Automatic Control, 2010, Vol. 55, No. 11, pp. 2462–2475.

References

- A.N. Venkat, J.B. Rawlings, and S.J. Wright, “Stability and optimality of distributed model predictive control, ” CDC-ECC’05, Seville, Spain.
- B.T. Stewart, A.N. Venkat, J.B. Rawlings, S.J. Wright, and G. Pannocchia, “Cooperative distributed model predictive control,” System & Control Letters, 2010, Vol. 59, pp. 460–469.
- D. Jia and B. Krogh, “Distributed Model Predictive Control,” In Proc. American Control Conference, Arlington, VA, 2001, pp. 2767–2772.
- M. Mercangöz and F.J. Doyle III, “Distributed model predictive control of an experimental four-tank system,” Journal of Process Control, 2007, Vol. 17, No. 3, pp. 297-308.
- W.B. Dunbar and R.M. Murray, “Distributed receding horizon control with application to multi-vehicle formation stabilization,” Automatica, 2006, Vol. 42, No. 4, pp. 549–558.
- L. Magni and R. Scattolini, “Stabilizing decentralized model predictive control of nonlinear systems,” Automatica, 2006, Vol. 42, No. 7, pp. 1231–1236.
- R. Scattolini, “Architectures for distributed and hierarchical model predictive control – a review,” Journal of Process Control, 2009, Vol. 19, pp. 723–731.
- A. Alessio, D. Barcelli, and A. Bemporad, “Decentralized model predictive control of dynamically coupled linear systems,” Journal of Process Control, 2011, Vol. 21, No. 5, pp. 705-714.