

C20 Distributed Systems

Lecture 1

Kostas Margellos

University of Oxford



Logistics

- **Who:** Kostas Margellos, Control Group, IEB 50.16
contact : kostas.margellos@eng.ox.ac.uk
- **When:** 4 lectures,
weeks 5 & 6 – Thu, Fri @4pm
- **Where:** LR2
- **Other info :**
 - ▶ 2 example classes (week 7) : Wed 3-5pm (LR2) – Fri 9-11am (LR3)
 - ▶ Lecture slides & handwritten notes available on Canvas
 - ▶ Teaching style : Mix of slides and whiteboard !

References

- Bertsekas & Tsitsiklis (1989)
Parallel and distributed computation : Numerical methods
Athena Scientific (*some figures taken from Chapter 3*).
- Bertsekas (2015)
Convex optimization algorithms
Athena Scientific (*Chapter 5*).
- Facchinei, Scutari & Sagratella (2015)
Parallel selective algorithms for nonconvex big data optimization,
IEEE Transactions on Signal Processing, 63(7), 1874–1889.
- Nedich, Ozdaglar & Parrilo (2010)
Constrained consensus and optimization in multi-agent networks,
IEEE Transactions on Automatic Control, 55(4), 922–938.
- Margellos, Falsone, Garatti & Prandini (2018)
Distributed constrained optimization and consensus in uncertain networks via proximal minimization,
IEEE Transactions on Automatic Control, 63(5), 1372-1387.
- Falsone, Margellos, Garatti & Prandini (2018)
Distributed constrained optimization and consensus in uncertain networks via proximal minimization,
Automatica, 84(10), 149-158.

Motivation

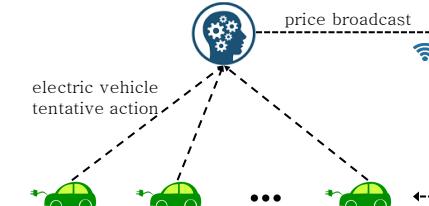
- Networks (Power, Social, etc.)
- ▶ Large scale infrastructures
- ▶ Multi-agent – Multiple interacting entities/users
- ▶ Heterogeneous – Different physical or technological constraints per agent ; different objectives per agent
- Challenge : Optimizing the performance of a network ...
 - ▶ Computation : Problem size too big !
 - ▶ Communication : Not all communication links at place ; link failures
 - ▶ Information privacy : Agents may not want to share information with everyone (e.g. facebook)

Why go decentralized/distributed?

- ① Scalable methodology
 - ▶ Communication :
 - Decentralized : With some central authority
 - Distributed : Only between neighbours
 - ▶ Computation : Only local ; in parallel for all agents
- ② Information privacy
 - ▶ Agents do not reveal information about their preferences (encoded by objective and constraint functions) to each other
- ③ Resilience to communication failures
- ④ Numerous applications
 - ▶ Wireless networks
 - ▶ Optimal power flow
 - ▶ Electric vehicle charging control
 - ▶ Energy management in building networks

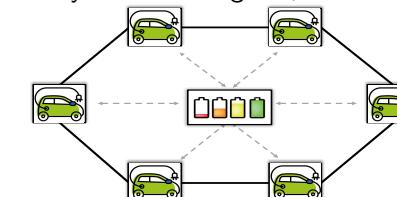
Decentralized vs. Distributed

- ① **Decentralized** : All agents with a central authority/coordinator



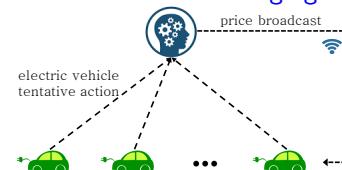
Decentralized vs. Centralized : Agents "broadcast" only tentative information **not** everything

- ② **Distributed** : Only with some agents, termed neighbours



Multi-agent problem classes

Motivating example : Electric vehicle charging



- Charging rate of each vehicle : x_i (in units of power)
- Electric vehicles are like batteries : X_i encodes limits on charging rate

Price depends on everybody's consumption

$$\text{minimize } \sum_i x_i^\top p(\sum_i x_i) \quad [\text{price function } p(\cdot)]$$

subject to : $x_i \in X_i$, for all i [limitations on the charging rate]

Multi-agent problem classes

Cost coupled problems

$$\begin{aligned} & \text{minimize } F(x_1, \dots, x_m) \\ & \text{subject to} \\ & \quad x_i \in X_i, \quad \forall i = 1, \dots, m \end{aligned}$$

- Agents have **separate decisions** : x_i for agent i
- Agents have **separate constraint sets** : X_i for agent i
- Agents aim at minimizing a **single objective function** F that couples their decisions

Can we transform one problem class to another?

From cost coupled to constraint coupled problems

$$\text{minimize } \gamma = \sum_i \frac{\gamma}{m}$$

subject to

$$x_i \in X_i, \forall i = 1, \dots, m$$

$$F(x_1, \dots, x_m) \leq \gamma$$

- Introduce an additional scalar epigraphic variable γ
- Move coupling to the constraints, i.e. $F(x_1, \dots, x_m) \leq \gamma$
- **Price to pay** : Coupling can **not** be split among several functions, each of them depending only on x_i , i.e. not in the form $\sum_i g_i(x_i) \leq 0$



Can we transform one problem class to another?

Yes, but ...

- We can transform from some problem classes to others
- Often those reformulations are useful
- However, they come with drawbacks :
 - may increase number of decision variables,
 - or lead to non-separable constraints,
 - or non-differentiable objective functions

So necessary to develop algorithms tailored to each problem class

Can we transform one problem class to another?

From decision coupled to cost coupled problems

$$\text{minimize } F(x_1, \dots, x_m) = \sum_i f_i(x) + I_{X_i}(x)$$

subject to : **no constraints**

- Lift the constraints in the objective function via characteristic functions, i.e., for each i ,

$$I_{X_i}(x) = \begin{cases} 0 & \text{if } x \in X_i; \\ +\infty & \text{otherwise.} \end{cases}$$

- New problem does not have any constraints
- **Price to pay** : The new objective function is **not** differentiable, even if each f_i is differentiable

Part I : Decentralized algorithms

Cost coupled problems

Cost coupled problems¹

$$\text{minimize } F(x_1, \dots, x_m)$$

subject to

$$x_i \in X_i, \forall i = 1, \dots, m$$

- Denote by x^* a minimizer of the cost coupled problem
- Denote by F^* its minimum value

1. Throughout we assume that all functions and sets are **convex**



Mathematical prelims : Lipschitz & Contraction mappings

- Let $T: X \rightarrow X$. We call T a **Lipschitz** mapping if there exists $\alpha > 0$ such that

$$\|T(x) - T(y)\| \leq \alpha \|x - y\|, \text{ for all } x, y \in X$$

- We call a Lipschitz mapping T **contraction** mapping if $\alpha \in [0, 1)$
- Parameter $\alpha \in [0, 1)$ is called the modulus of contraction of T
- We should always specify the norm

Convergence of contractive iterations

Assume T is a contraction with modulus $\alpha \in [0, 1)$ and X is a closed set.

- T has a unique fixed-point $T(x^*) = x^*$
- The Picard-Banach iteration $x(k+1) = T(x(k))$ converges to x^* geometrically, i.e.

$$\|x(k) - x^*\| \leq \alpha^k \|x(0) - x^*\|, \text{ for all } k \geq 0$$

The Jacobi algorithm

- Iterative algorithm

Initialize: Select (arbitrarily) $x_i(0) \in X_i$, for all $i = 1, \dots, m$

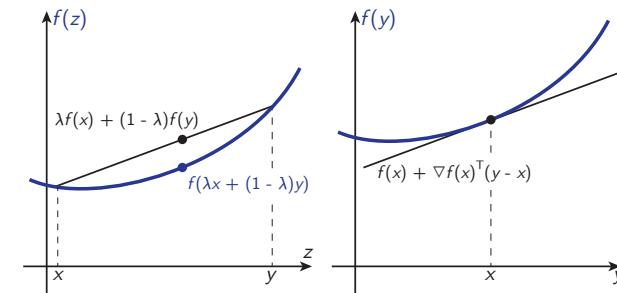
For each iteration $k = 1, \dots$

- Collect $x(k) = (x_1(k), \dots, x_m(k))$ from central authority
- Agents update their local decision in parallel, i.e. for all $i = 1, \dots, m$

$$x_i(k+1) = \arg \min_{x_i \in X_i} F(x_1(k), \dots, x_{i-1}(k), x_i, x_{i+1}(k), \dots, x_m(k))$$

end for

Mathematical prelims : Convexity vs strong convexity



- Strong convexity is “stronger” than convexity – uniqueness of optimum & lower bound on growth

$$f(y) \geq f(x) + \nabla f(x)^T (y - x) + \sigma \|y - x\|^2, \text{ where } \sigma > 0$$

- We can fit a quadratic function between the “true” function and its linear approximation
- For quadratic functions strong is the same with strict convexity

The Jacobi algorithm

- Agents coupled via a single objective function

$$\begin{aligned} &\text{minimize } F(x_1, \dots, x_m) \\ &\text{subject to : } x_i \in X_i, \forall i = 1, \dots, m \end{aligned}$$

- Collect $x(k) = (x_1(k), \dots, x_m(k))$ from central authority
- Agents update their local decision in parallel

$$x_i(k+1) = \arg \min_{x_i \in X_i} F(x_1(k), \dots, x_{i-1}(k), x_i, x_{i+1}(k), \dots, x_m(k))$$

- Block coordinate descent method ; agents act in **best response**
- Parallelizable method : Agent i uses the k -th updates of all agents

Jacobi algorithm : Convergence

Theorem : Convergence of Jacobi algorithm

If F is differentiable and there exists small enough γ such that

$$T(x) = x - \gamma \nabla F(x)$$

is a contraction mapping (modulus in $[0, 1]$), then there exists a minimizer x^* of the cost coupled problem such that

$$\lim_{k \rightarrow \infty} \|x(k) - x^*\| = 0$$

- Best response but a gradient step appears in convergence
- A sufficient condition for T to be a contractive map is F to be a strongly convex function
- Can we relax this condition ?

Regularized Jacobi algorithm : Convergence

Theorem : Convergence of regularized Jacobi algorithm

Assume that F is convex and ∇F is Lipschitz continuous with constant L . Assume also that

$$c > \frac{m-1}{2m-1} \sqrt{mL}$$

We then have that $\lim_{k \rightarrow \infty} \|F(x(k)) - F^*\| = 0$

- Algorithm converges in value, not necessarily in iterates, i.e. not necessarily $\lim_{k \rightarrow \infty} \|x(k) - x^*\| = 0$
- Penalty term c increases as $m \rightarrow \infty$
- The more agents the "slower" the overall process

The regularized Jacobi algorithm

- ① Collect $x(k) = (x_1(k), \dots, x_m(k))$ from central authority
- ② Agents update their local decision in parallel

$$x_i(k+1) = \arg \min_{x_i \in X_i} F(x_1(k), \dots, x_{i-1}(k), x_i, x_{i+1}(k), \dots, x_m(k)) + c \|x_i - x_i(k)\|_2^2$$

- Jacobi algorithm + regularization term
- Penalty term acts like "inertia" from previous tentative solution of agent i
- New objective function is strongly convex due to regularization

The Gauss-Seidel algorithm

- ① Collect $x(k) = (x_1(k+1), \dots, x_{i-1}(k+1), x_i(k), \dots, x_m(k))$
- ② Agent i updates

$$x_i(k+1) = \arg \min_{x_i \in X_i} F(x_1(k+1), \dots, x_{i-1}(k+1), x_i, x_{i+1}(k), \dots, x_m(k))$$

- Block coordinate descent method ; agents act in best response
- Sequential : Agent i uses the $(k+1)$ -th updates of preceding agents
- Similar convergence results with Jacobi algorithm : If F is strongly convex (strict convexity is sufficient) with respect to each individual argument, then $\lim_{k \rightarrow \infty} \|F(x(k)) - F^*\| = 0$

Summary

Decentralized algorithms for cost coupled problems

minimize $F(x_1, \dots, x_m)$
 subject to $x_i \in X_i, \forall i = 1, \dots, m$

- The Jacobi algorithm : parallel updates
 F differentiable and **strongly convex**
 - The regularized Jacobi algorithm : parallel updates
 F differentiable and just convex
 - The Gauss-Seidel algorithm : sequential updates
 F differentiable and **strongly convex** per agent's decision
 - ⇒ For quadratic functions $x^\top Qx$:
 - convex if $Q \succeq 0$; strongly convex if $Q > 0$
 - Strong convexity = strict convexity

Thank you for your attention!
Questions?

Contact at :
kostas.margellos@eng.ox.ac.uk

Michaelmas Term 2024

C20 Distributed Systems

November 9, 2024

25 / 26

C20 Distributed Systems

Lecture 2

Kostas Margellos

University of Oxford



Michaelmas Term 2024

C20 Distributed Systems

November 9, 2022

1 / 24

- The Jacobi algorithm : parallel updates
 F differentiable and **strongly convex**
 - The regularized Jacobi algorithm : parallel updates
 F differentiable and just convex
 - The Gauss-Seidel algorithm : sequential updates
 F differentiable and **strongly convex** per agent's decision
 - ⇒ For quadratic functions $x^T Qx$:
 - convex if $Q \succeq 0$; strongly convex if $Q > 0$
 - Strong convexity = strict convexity

Recap

Decentralized algorithms for cost coupled problems

minimize $F(x_1, \dots, x_m)$
 subject to $x_i \in X_i, \forall i = 1, \dots, m$

- The Jacobi algorithm : parallel updates
 F differentiable and **strongly convex**
- The regularized Jacobi algorithm : parallel updates
 F differentiable and just convex
- The Gauss-Seidel algorithm : sequential updates
 F differentiable and **strongly convex** per agent's decision
 - ⇒ For quadratic functions $x^T Qx$:
 - convex if $Q \succeq 0$; strongly convex if $Q > 0$
 - Strong convexity = strict convexity

Michaelmas Term 202

C20 Distributed Systems

November 9, 2024

2 / 24

Part I : Decentralized algorithms

Decision coupled problems

Decision coupled problems – The primal

$$\text{minimize } \sum_i f_i(x)$$

subject to

$$x \in X_i, \forall i = 1, \dots, m$$

Michaelmas Term 2024

C20 Distributed Systems

November 9, 2024

3 / 24

Part I : Decentralized algorithms

Decision coupled problems

- Decentralized solution roadmap

- The main algorithm for this is the [Alternating Direction Method of Multipliers \(ADMM\)](#)
- The predecessor of ADMM is the [Augmented Lagrangian](#) algorithm
- The Augmented Lagrangian is in turn based on the [Proximal algorithm](#)

Proximal \implies Augmented Lagrangian \implies ADMM

The proximal minimization algorithm

- Consider a differentiable F ¹. The following problems are equivalent

Standard minimization program

$$\begin{aligned} & \text{minimize } F(x) \\ & \text{subject to : } x \in X \end{aligned}$$

Proximal minimization program

$$\begin{aligned} & \text{minimize } F(x) + \frac{1}{2c} \|x - y\|^2 \\ & \text{subject to : } x \in X, y \in \mathbb{R}^n \end{aligned}$$

- The proximal problem has an objective function which is differentiable and strongly convex (for any fixed y)
- We can solve it iteratively via the Gauss-Seidel algorithm ; converges for any $c > 0$ (see Lecture 1)
- Alternate between minimizing x and y

1. The proximal algorithm converges even without the differentiability assumption ; see function at slide 8. We impose it here to show convergence via the Gauss-Seidel scheme.

Michaelmas Term 2024

C20 Distributed Systems

November 9, 2024

5 / 24

The proximal minimization algorithm

- The following problems are equivalent

Standard minimization program

$$\begin{aligned} & \text{minimize } F(x) \\ & \text{subject to : } x \in X \end{aligned}$$

Proximal minimization program

$$\begin{aligned} & \text{minimize } F(x) + \frac{1}{2c} \|x - y\|^2 \\ & \text{subject to : } x \in X, y \in \mathbb{R}^n \end{aligned}$$

Proximal algorithm :

- $x(k+1) = \arg \min_{x \in X} F(x) + \frac{1}{2c} \|x - y(k)\|^2$
 - $y(k+1) = x(k+1)$
- ... or
- $x(k+1) = \arg \min_{x \in X} F(x) + \frac{1}{2c} \|x - x(k)\|^2$

Michaelmas Term 2024

C20 Distributed Systems

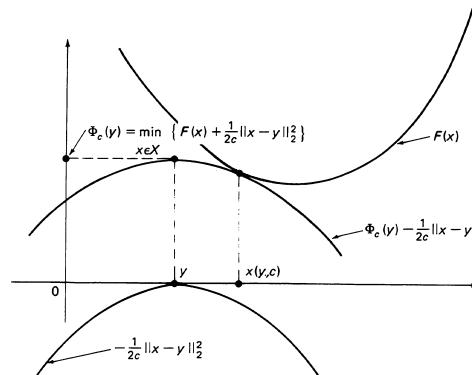
November 9, 2024

6 / 24

The proximal minimization algorithm

Geometric interpretation

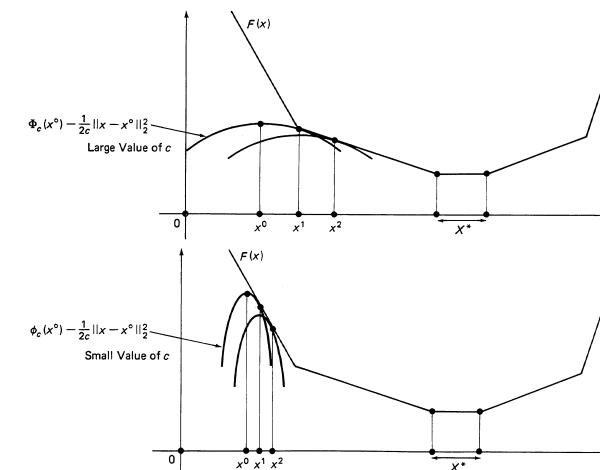
- Let $\Phi_c(y) = \min F(x) + \frac{1}{2c} \|x - y\|^2$ achieved at $x = x(y, c)$
 - Hence, $\Phi_c(y) = F(x(y, c)) + \frac{1}{2c} \|x(y, c) - y\|^2 \leq F(x) + \frac{1}{2c} \|x - y\|^2$
 $\Rightarrow \Phi_c(y) - \frac{1}{2c} \|x - y\|^2 \leq F(x)$, with equality at $x = x(y, c)$



The proximal minimization algorithm

Geometric interpretation

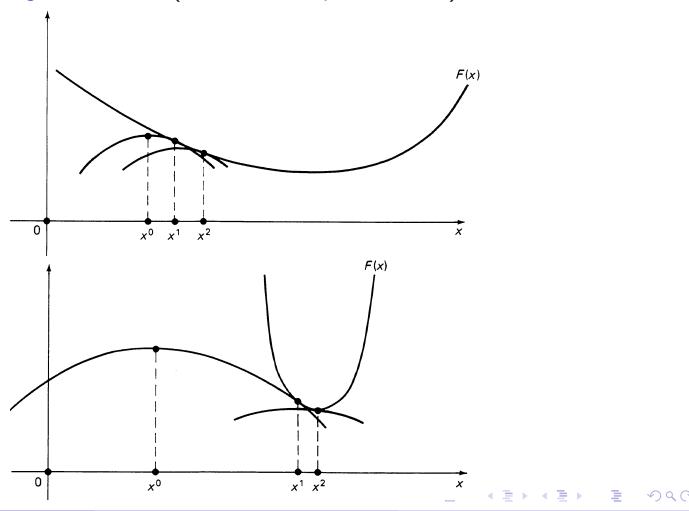
- Effect of **large** and **small** values of α



The proximal minimization algorithm

Geometric interpretation

- Effect of the **growth** of F (flat and steep functions)



The augmented Lagrangian algorithm

- Consider the following problems

Standard program

$$\begin{aligned} & \text{minimize}_{x \in X} F(x) \\ & \text{subject to : } Ax = b \end{aligned}$$

Augmented program

$$\begin{aligned} & \text{minimize}_{x \in X} F(x) + \frac{c}{2} \|Ax - b\|^2 \\ & \text{subject to : } Ax = b \end{aligned}$$

- Trivially equivalent problems : For any feasible x , the “proxy” term becomes zero
 - Resembles the structure of the proximal algorithm
 - $Ax = b$ models *complicating* constraints :
if $F(x) = \sum_i f_i(x_i)$ and $X = X_1 \times \dots \times X_m$, then $Ax = b$ models coupling among agents’ decisions

Proof (cont'd)

Step 3 :

- Substituting back the value of \bar{z}

$$\begin{aligned} y^* &= \arg \max_y \left\{ q(y) + \min_z \left((\lambda(k) - y)^\top z + \frac{c}{2} \|z\|^2 \right) \right\} \\ &= \arg \max_y \left\{ q(y) - \frac{1}{2c} \|y - \lambda(k)\|^2 \right\} \end{aligned}$$

- At the same time, due to the equality constraint in Step 1,
 $z(k+1) = Ax(k+1) - b$, hence

$$\lambda(k+1) = \lambda(k) + c(Ax(k+1) - b) \implies \lambda(k+1) = y^*$$

which in turn implies that

$$\lambda(k+1) = \arg \max_y q(y) - \frac{1}{2c} \|y - \lambda(k)\|^2$$

Back to decision coupled problems

Recall the equivalence between decision and constraint coupled problems

Decision coupled problem

$$\begin{aligned} &\text{minimize}_i f_i(x) \\ &\text{subject to : } x \in X_i, \forall i \end{aligned}$$

Constraint coupled problem

$$\begin{aligned} &\text{minimize}_i f_i(x_i) \\ &\text{subject to : } x_i \in X_i, \forall i \\ &\quad x_i = z, \forall i \end{aligned}$$

- We will show that this constraint coupled problem is in the form of

$$\begin{aligned} &\text{minimize}_{x \in X} F(x) \\ &\text{subject to : } Ax = b \end{aligned}$$

Proof (cont'd)

Step 4 : Putting everything together ...

- The augmented Lagrangian primal dual scheme

$$\begin{aligned} ① \quad x(k+1) &= \arg \min_{x \in X} F(x) + \lambda(k)^\top (Ax - b) + \frac{c}{2} \|Ax - b\|^2 \\ ② \quad \lambda(k+1) &= \lambda(k) + c(Ax(k+1) - b) \end{aligned}$$

... is equivalent to

$$① \quad \lambda(k+1) = \arg \max_y q(y) - \frac{1}{2c} \|y - \lambda(k)\|^2$$

- Proximal algorithm for the dual function $q(y)$!
- It converges for any c as $q(y)$ is the dual function thus always concave, i.e. $\lim_{k \rightarrow \infty} \|\lambda(k) - \lambda^*\| = 0$ for some optimal λ^*
- For the primal variables we can only show something slightly weaker : they asymptotically achieve the optimal value F^*

Decision coupled problems

Consider the following assignments :

- Decision vector

$$x \leftarrow (x_1, \dots, x_m, z)$$

- Constraint sets

$$X \leftarrow X_1 \times \dots \times X_m \times \mathbb{R}^n$$

- Objective function

$$F(x_1, \dots, x_m, z) \leftarrow \sum_i f_i(x_i)$$

- Matrices A and b :

$$Ax = b \Leftrightarrow \begin{bmatrix} -1 & 0 & \dots & 0 & 1 \\ 0 & -1 & \dots & 0 & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & -1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_m \\ z \end{bmatrix} = 0$$

- Dual variable : $\lambda \leftarrow (\lambda_1, \dots, \lambda_m)$

$$\lambda(k)^\top (Ax - b) = \sum_i \lambda_i^\top (z - x_i) \text{ and } \|Ax - b\|^2 = \sum_i \|z - x_i\|^2$$

Summary

Decision coupled problems

$$\begin{aligned} & \text{minimize } \sum_i f_i(x) \\ & \text{subject to} \\ & \quad x \in X_i, \forall i = 1, \dots, m \end{aligned}$$

Thank you for your attention !
Questions ?

Introduced three different algorithms

- Proximal minimization algorithm
- Augmented Lagrangian algorithm
- Augmented Lagrangian with **one** pass of the inner loop = ADMM

Proximal \implies Augmented Lagrangian \implies ADMM

C20 Distributed Systems Lecture 3

Kostas Margellos

University of Oxford



Contact at :

kostas.margellos@eng.ox.ac.uk

Recap

Decision coupled problems

$$\begin{aligned} & \text{minimize } \sum_i f_i(x) \\ & \text{subject to} \\ & \quad x \in X_i, \forall i = 1, \dots, m \end{aligned}$$

Introduced three different algorithms

- Proximal minimization algorithm
- Augmented Lagrangian algorithm
- Augmented Lagrangian with **one** pass of the inner loop = ADMM

Proximal \implies Augmented Lagrangian \implies ADMM

Recap : Augmented Lagrangian algorithm

Inner loop : Gauss-Seidel algorithm !

begin loop

- ➊ Primal update for \mathbf{z} information from central authority

$$\mathbf{z} = \frac{1}{m} \sum_i \mathbf{x}_i - \frac{1}{mc} \sum_i \lambda_i(k)$$

- ➋ Primal update for \mathbf{x}_i in parallel for all agents

$$\mathbf{x}_i = \arg \min_{\mathbf{x}_i \in X_i} f_i(\mathbf{x}_i) - \lambda_i(k)^T \mathbf{x}_i + \frac{c}{2} \|\mathbf{z} - \mathbf{x}_i\|^2$$

end loop

- ➌ Dual update in parallel for all agents

$$\lambda_i(k+1) = \lambda_i(k) + c(\mathbf{z}(k+1) - \mathbf{x}_i(k+1))$$

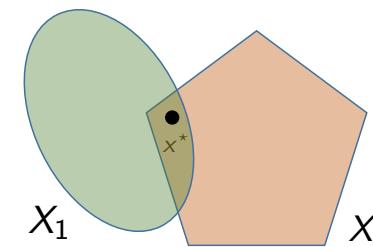
Example

Feasibility problem – part of Question 4, Example Paper

Find a point \mathbf{x}^* at the intersection (assumed to be non-empty) of two (possibly different) convex sets X_1 and X_2 , i.e.

$$\begin{aligned} & \text{minimize } 0 && [\text{any constant would work}] \\ & \text{subject to } \mathbf{x} \in X_1 \text{ and } \mathbf{x} \in X_2 \end{aligned}$$

Apply Augmented Lagrangian algorithm initializing at $\lambda_1(0) = \lambda_2(0) = 0$.



Example (cont'd)

- Decision coupled problem with 2 agents ; notice that $f_1(\mathbf{x}) = f_2(\mathbf{x}) = 0$
- Consider $k = 0$ and focus at the inner loop of the Augmented Lagrangian algorithm
- Recall that $\lambda_1(0) = \lambda_2(0) = 0$

Outer loop at $k = 0$; main steps of inner loop

$$➊ \quad \mathbf{z} = \frac{\mathbf{x}_1 + \mathbf{x}_2}{2} - \frac{\lambda_1(0) + \lambda_2(0)}{2c} = \frac{\mathbf{x}_1 + \mathbf{x}_2}{2}$$

$$\begin{aligned} ➋ \quad \mathbf{x}_1 &\leftarrow \arg \min_{\mathbf{x}_1 \in X_1} -\lambda_1(0)\mathbf{x}_1 + \frac{c}{2} \|\mathbf{z} - \mathbf{x}_1\|^2 = \arg \min_{\mathbf{x}_1 \in X_1} \frac{c}{2} \|\mathbf{z} - \mathbf{x}_1\|^2 \\ \mathbf{x}_2 &\leftarrow \arg \min_{\mathbf{x}_2 \in X_2} -\lambda_2(0)\mathbf{x}_2 + \frac{c}{2} \|\mathbf{z} - \mathbf{x}_2\|^2 = \arg \min_{\mathbf{x}_2 \in X_2} \frac{c}{2} \|\mathbf{z} - \mathbf{x}_2\|^2 \end{aligned}$$

- Second step exhibits a nice structure and geometric interpretation
- Solve the unconstrained quadratic program and project on the constraint set (X_1 and X_2 , respectively)

Example (cont'd)

- Denote by $\Pi_{X_i}[\mathbf{z}]$ the projection of \mathbf{z} on the set X_i
- Inner loop becomes then ...

Outer loop at $k = 0$; main steps of inner loop

$$➊ \quad \mathbf{z} = \frac{\mathbf{x}_1 + \mathbf{x}_2}{2}$$

$$➋ \quad \mathbf{x}_1 \leftarrow \arg \min_{\mathbf{x}_1 \in X_1} \frac{c}{2} \|\mathbf{z} - \mathbf{x}_1\|^2 = \Pi_{X_1}[\mathbf{z}]$$

$$\mathbf{x}_2 \leftarrow \arg \min_{\mathbf{x}_2 \in X_2} \frac{c}{2} \|\mathbf{z} - \mathbf{x}_2\|^2 = \Pi_{X_2}[\mathbf{z}]$$

- This is just the Gauss-Seidel to solve the problem

$$\text{minimize}_{\mathbf{z}, \mathbf{x}_1 \in X_1, \mathbf{x}_2 \in X_2} \frac{c}{2} \sum_{i=1,2} \|\mathbf{z} - \mathbf{x}_i\|^2$$

- Hence it converges to the minimum, which occurs when $\mathbf{x}_1 = \mathbf{x}_2 = \mathbf{z}$

Example (cont'd)

- Since upon convergence of the inner loop $x_1 = x_2 = z$, then the outer loop update becomes

$$\lambda_i(1) = \lambda_i(0) + c(z(1) - x_i(1)) = 0, \text{ for } i = 1, 2$$

- Similarly, $\lambda_i(k) = 0$ for all $k \geq 0$
- Effectively we only have one loop!

Simplified single-loop algorithm

① Averaging step : $z(k+1) = \frac{x_1(k)+x_2(k)}{2}$

② Parallel projections :

$$x_1(k+1) = \Pi_{X_1}[z(k+1)] \text{ and } x_2(k+1) = \Pi_{X_2}[z(k+1)]$$

For decision coupled problems ...

Augmented Lagrangian with one Gauss-Seidel pass = ADMM

① Primal update for z information from central authority

$$z(k+1) = \frac{1}{m} \sum_i x_i(k) - \frac{1}{mc} \sum_i \lambda_i(k)$$

② Primal update for x_i in parallel for all agents

$$x_i(k+1) = \arg \min_{x_i \in X_i} f_i(x_i) - \lambda_i(k)^T x_i + \frac{c}{2} \|z(k+1) - x_i\|^2$$

③ Dual update

$$\lambda_i(k+1) = \lambda_i(k) + c(z(k+1) - x_i(k+1))$$

Example (cont'd)

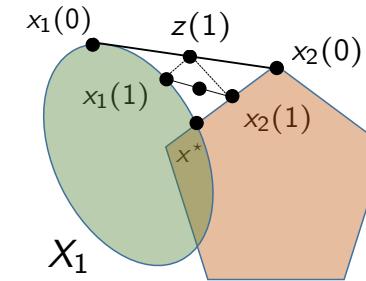
Simplified single-loop algorithm

① Averaging step : $z(k+1) = \frac{x_1(k)+x_2(k)}{2}$

② Parallel projections :

$$x_1(k+1) = \Pi_{X_1}[z(k+1)] \text{ and } x_2(k+1) = \Pi_{X_2}[z(k+1)]$$

Schematic illustration of the single-loop iterations



For decision coupled problems ...

Equivalent notation in line with ADMM literature (the roles of x and z are reversed) – only notational change !

① Primal update for x information from central authority

$$x(k+1) = \frac{1}{m} \sum_i z_i(k) - \frac{1}{mc} \sum_i \lambda_i(k)$$

② Primal update for z_i in parallel for all agents

$$z_i(k+1) = \arg \min_{z_i \in X_i} f_i(z_i) - \lambda_i(k)^T z_i + \frac{c}{2} \|x(k+1) - z_i\|^2$$

③ Dual update

$$\lambda_i(k+1) = \lambda_i(k) + c(x(k+1) - z_i(k+1))$$

The Alternating Direction Method of Multipliers (ADMM)

- ADMM even more general than decision coupled problems
- Splitting algorithm : decouples optimization across groups of variables

Group variables

$$\begin{aligned} & \text{minimize } F_1(\mathbf{x}) + F_2(\mathbf{Ax}) \\ & \text{subject to : } \mathbf{x} \in C_1, \quad \mathbf{Ax} \in C_2 \end{aligned}$$

Equivalent reformulation

$$\begin{aligned} & \text{minimize } F_1(\mathbf{x}) + F_2(\mathbf{z}) \\ & \text{subject to : } \mathbf{x} \in C_1, \quad \mathbf{z} \in C_2 \\ & \quad \mathbf{Ax} = \mathbf{z} \end{aligned}$$

Decision coupled problems as a special case again

Original problem

$$\begin{aligned} & \text{minimize } \sum_i f_i(x) \\ & \text{subject to : } x \in X_i, \quad \forall i \end{aligned}$$

Equivalent problem

$$\begin{aligned} & \text{minimize } \sum_i f_i(z_i) \\ & \text{subject to : } z_i \in X_i, \quad \forall i \\ & \quad z_i = x, \quad \forall i \end{aligned}$$

- Let z_1, \dots, z_m be copies of x
- Original (decision coupled problem) becomes equivalent to a constrained coupled one
- Show then that the equivalent problem is in the format for which ADMM is applicable

ADMM algorithm

Effectively Augmented Lagrangian with one Gauss-Seidel pass

$$\begin{aligned} ① \quad & \mathbf{x}(k+1) = \arg \min_{\mathbf{x} \in C_1} F_1(\mathbf{x}) + \lambda(k)^T \mathbf{A} \mathbf{x} + \frac{\epsilon}{2} \| \mathbf{A} \mathbf{x} - \mathbf{z}(k) \|^2 \\ ② \quad & \mathbf{z}(k+1) = \arg \min_{\mathbf{z} \in C_2} F_2(\mathbf{z}) - \lambda(k)^T \mathbf{z} + \frac{\epsilon}{2} \| \mathbf{A} \mathbf{x}(k+1) - \mathbf{z} \|^2 \\ ③ \quad & \lambda(k+1) = \lambda(k) + c(A\mathbf{x}(k+1) - \mathbf{z}(k+1)) \end{aligned}$$

Theorem : Convergence of ADMM

Assume that the set of optimizers is non-empty, and either C_1 is bounded or $A^T A$ is invertible. We then have that

- ① $\lambda(k)$ converges to an optimal dual variable.
- ② $(\mathbf{x}(k), \mathbf{z}(k))$ achieves the optimal value
If $A^T A$ invertible then it converges to an optimal primal pair

Decision coupled problems as a special case again

Equivalent problem

$$\begin{aligned} & \text{minimize } \sum_i f_i(z_i) \\ & \text{subject to : } z_i \in X_i, \quad \forall i \\ & \quad z_i = x, \quad \forall i \end{aligned}$$

ADMM set-up

$$\begin{aligned} & \text{minimize } F_1(\mathbf{x}) + F_2(\mathbf{z}) \\ & \text{subject to : } \mathbf{x} \in C_1, \quad \mathbf{z} \in C_2 \\ & \quad \mathbf{Ax} = \mathbf{z} \end{aligned}$$

- To see this, let $\mathbf{z} = (z_1, \dots, z_m)$ and define $A = \begin{bmatrix} I \\ \vdots \\ I \end{bmatrix}$ (stack of identity matrices), hence $A\mathbf{x} = \begin{bmatrix} x \\ \vdots \\ x \end{bmatrix}$ and $A\mathbf{x} = \mathbf{z} \Leftrightarrow \begin{bmatrix} x \\ \vdots \\ x \end{bmatrix} = \begin{bmatrix} z_1 \\ \vdots \\ z_m \end{bmatrix}$
- Notice that $A^T A = I$, hence invertible (algorithm expected to converge)

Decision coupled problems (cont'd)

- Perform also the following assignments

$$\begin{aligned} F_1(\mathbf{x}) &= 0, \quad C_1 = \mathbb{R}^n \\ F_2(\mathbf{z}) &= \sum_i f_i(\mathbf{z}_i), \quad C_2 = X_1 \times \dots \times X_m \end{aligned}$$

- For each block constraint, i.e. $\mathbf{x} = \mathbf{z}_i$ assign the dual vector λ_i , and let $\lambda = (\lambda_1, \dots, \lambda_m)$
- The three ADMM steps become then

- ① $\mathbf{x}(k+1) = \arg \min_{\mathbf{x} \in \mathbb{R}^n} \lambda(k)^\top A\mathbf{x} + \frac{c}{2} \|A\mathbf{x} - \mathbf{z}(k)\|^2$
- ② $\mathbf{z}(k+1) = \arg \min_{\mathbf{z}_1 \in X_1, \dots, \mathbf{z}_m \in X_m} \sum_i f_i(\mathbf{z}_i) - \lambda(k)^\top \mathbf{z} + \frac{c}{2} \|A\mathbf{x}(k+1) - \mathbf{z}\|^2$
- ③ $\lambda(k+1) = \lambda(k) + c(A\mathbf{x}(k+1) - \mathbf{z}(k+1))$

Constraint coupled problems

Affine coupling :

$$\begin{aligned} &\text{minimize} \quad \sum_i f_i(x_i) \\ &\text{subject to : } x_i \in X_i, \quad \forall i \\ &\qquad \sum_i x_i = 0 \end{aligned}$$

- Affine coupling constraint : equality with zero for simplicity
- We could have general coupling constraints $Ax = b$; see Example 4.4, Chapter 3 in [Bertsekas & Tsitsiklis 1989]
- We can still treat as an ADMM example

Decision coupled problems (cont'd)

... or equivalently (**compare with slide 5 !**)

$$\textcircled{1} \quad \mathbf{x}(k+1) = \arg \min_{\mathbf{x} \in \mathbb{R}^n} \sum_i \lambda_i(k)^\top \mathbf{x} + \frac{c}{2} \sum_i \|\mathbf{x} - \mathbf{z}_i(k)\|^2$$

- Unconstrained quadratic optimization
- Setting the gradient with respect to \mathbf{x} equal to zero we obtain

$$\sum_i \lambda_i(k) + c \sum_i (\mathbf{x}(k+1) - \mathbf{z}_i(k)) = 0$$

$$\Rightarrow \mathbf{x}(k+1) = \frac{1}{m} \sum_i \mathbf{z}_i(k) - \frac{1}{mc} \sum_i \lambda_i(k)$$

$$\textcircled{2} \quad \mathbf{z}(k+1) = \arg \min_{\mathbf{z}_1 \in X_1, \dots, \mathbf{z}_m \in X_m} \sum_i \left(f_i(\mathbf{z}_i) - \lambda_i(k)^\top \mathbf{z}_i + \frac{c}{2} \|\mathbf{x}(k+1) - \mathbf{z}_i\|^2 \right)$$

- Since $\mathbf{x}(k+1)$ is fixed, fully separable across i . Minimizing the "sum" is equivalent to minimizing each individual component. Hence, for all i ,

$$\mathbf{z}_i(k+1) = \arg \min_{\mathbf{z}_i \in X_i} f_i(\mathbf{z}_i) - \lambda_i(k)^\top \mathbf{z}_i + \frac{c}{2} \|\mathbf{x}(k+1) - \mathbf{z}_i\|^2$$

$$\textcircled{3} \quad \lambda_i(k+1) = \lambda_i(k) + c(\mathbf{x}(k+1) - \mathbf{z}_i(k+1)) \quad (\text{due to the structure of } A)$$

Constraint coupled problems

Original problem

$$\begin{aligned} &\text{minimize} \quad \sum_i f_i(x_i) \\ &\text{subject to : } x_i \in X_i, \quad \forall i \\ &\qquad \sum_i x_i = 0 \end{aligned}$$

ADMM set-up

$$\begin{aligned} &\text{minimize} \quad F_1(\mathbf{x}) + F_2(\mathbf{z}) \\ &\text{subject to : } \mathbf{x} \in C_1, \quad \mathbf{z} \in C_2 \\ &\qquad A\mathbf{x} = \mathbf{z} \end{aligned}$$

- To see this, let $\mathbf{x} = (x_1, \dots, x_m)$, $\mathbf{z} = (z_1, \dots, z_m)$ and $A = \text{identity matrix}$
- Separate *complicated* objective from *complicated* constraints

$$F_1(\mathbf{x}) = \sum_i f_i(x_i), \quad C_1 = X_1 \times \dots \times X_m$$

$$F_2(\mathbf{z}) = 0, \quad C_2 = \{\mathbf{z} \mid \sum_i z_i = 0\}$$

Constraint coupled problems

ADMM algorithm for constraint coupled problems

- ① Primal update for x_i in parallel for all agents

$$x_i(k+1) = \arg \min_{x_i \in X_i} f_i(x_i) + \lambda_i^\top(k) x_i + \frac{c}{2} \|x_i - z_i(k)\|^2$$

- ② Primal update for z information from central authority

$$z(k+1) = \arg \min_{\{z: \sum_i z_i = 0\}} - \sum_i \lambda_i^\top(k) z_i + \frac{c}{2} \sum_i \|x_i(k+1) - z_i\|^2$$

- ③ Dual update $\lambda_i(k+1) = \lambda_i(k) + c(x_i(k+1) - z_i(k+1))$

Question 6, Example paper : Solve the z -minimization analytically

- Find unconstraint minimizer and project on $\sum_i z_i = 0$
- Notice that $\lambda_1(k) = \dots = \lambda_m(k)$ for all $k \geq 1$

Part II : Distributed algorithms

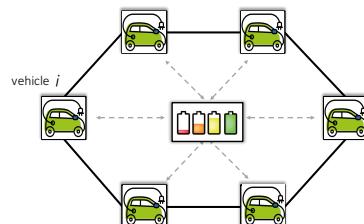
Decision coupled problems

$$\text{minimize } \sum_i f_i(x)$$

subject to

$$x \in X_i, \forall i = 1, \dots, m$$

Recall electric vehicle charging control problem



Decision coupled problem

$$\text{minimize } \sum_i f_i(x)$$

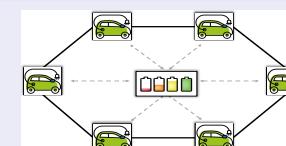
subject to

$$x \in X_i, \forall i = 1, \dots, m$$

Distributed proximal minimization

General architecture

Step 1 : Local problem of agent i



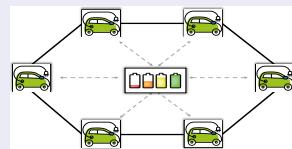
$$\left. \begin{array}{l} \text{minimize } f_i(x_i) + g_i(x_i, z_i) \\ \text{subject to } x_i \in X_i \end{array} \right\} \Rightarrow x_i^*(z_i)$$

- x_i : "copy" of x maintained by agent i NOT an element of x
- X_i : local constraint set of agent i
- z_i : information vector – constructed based on the info of agent's i neighbors
- Objective function
 $f_i(x_i)$: local cost/utility of agent i
 $g_i(x_i, z_i)$: Proxy term, penalizing disagreement with other agents

Distributed proximal minimization

General architecture

Step 1 : Local problem of agent i

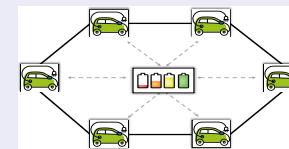


$$\begin{aligned} & \text{minimize } f_i(x_i) + g_i(x_i, z_i) \\ & \text{subject to} \\ & x_i \in X_i \end{aligned} \quad \left. \right\} \Rightarrow x_i^*(z_i)$$

Distributed proximal minimization

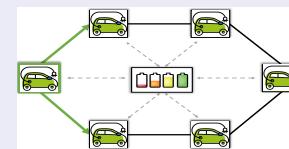
General architecture

Step 1 : Local problem of agent i

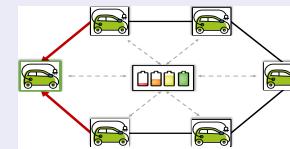


$$\begin{aligned} & \text{minimize } f_i(x_i) + g_i(x_i, z_i) \\ & \text{subject to} \\ & x_i \in X_i \end{aligned} \quad \left. \right\} \Rightarrow x_i^*(z_i)$$

Step 2a : Broadcast $x_i^*(z_i)$ to neighbors



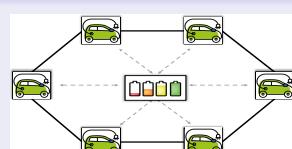
Step 2b : Receive neighbors' solutions



Distributed proximal minimization

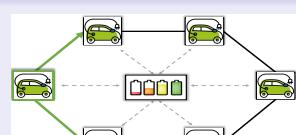
General architecture

Step 1 : Local problem of agent i

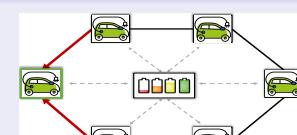


$$\begin{aligned} & \text{minimize } f_i(x_i) + g_i(x_i, z_i) \\ & \text{subject to} \\ & x_i \in X_i \end{aligned} \quad \left. \right\} \Rightarrow x_i^*(z_i)$$

Step 2a : Broadcast $x_i^*(z_i)$ to neighbors



Step 2b : Receive neighbors' solutions

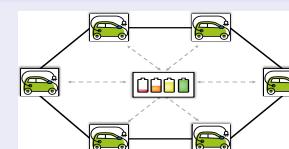


Step 3 : Update z_i on the basis of information received

Go to Step 1

Distributed proximal minimization

Local problem of agent i

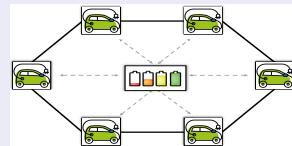


$$\begin{aligned} & \text{minimize } f_i(x_i) + g_i(x_i, z_i) \\ & \text{subject to} \\ & x_i \in X_i \end{aligned} \quad \left. \right\} \Rightarrow x_i^*(z_i)$$

- We need to specify
 - Information vector z_i
 - Proxy term term $g_i(x_i, z_i)$
- Note that these terms change across algorithm iterations
 - We need to make this dependency explicit

Distributed proximal minimization

Local problem of agent i at iteration $k + 1$



$$z_i(k) = \sum_j a_j^i(k) x_j(k)$$

$$x_i(k+1) = \arg \min_{x_i \in X_i} f_i(x_i) + \frac{1}{2c(k)} \|x_i - z_i(k)\|^2$$

- Information vector

- $z_i(k) = \sum_j a_j^i(k) x_j(k)$
- $a_j^i(k)$: how agent i weights info of agent j

- Proxy term

- $\frac{1}{2c(k)} \|x_i - z_i(k)\|^2$: deviation from (weighted) average
- $c(k)$: trade-off between optimality and agents' disagreement

Proximal minimization algorithm

① Averaging step in parallel for all agents

$$z_i(k) = \sum_j a_j^i(k) x_j(k)$$

② Primal update for x_i in parallel for all agents

$$x_i(k+1) = \arg \min_{x_i \in X_i} f_i(x_i) + \frac{1}{2c(k)} \|x_i - z_i(k)\|^2$$

- No dual variables introduced – primal only method

- All steps can be parallelized across agents – no central authority!

Contrast with the ADMM algorithm

ADMM algorithm

① Primal update for z information from central authority

$$z(k+1) = \frac{1}{m} \sum_i x_i(k) - \frac{1}{mc} \sum_i \lambda_i(k)$$

② Primal update for x_i in parallel for all agents

$$x_i(k+1) = \arg \min_{x_i \in X_i} f_i(x_i) - \lambda_i(k)^T x_i + \frac{c}{2} \|z(k+1) - x_i\|^2$$

③ Dual update in parallel for all agents

$$\lambda_i(k+1) = \lambda_i(k) + c(z(k+1) - x_i(k+1))$$

Distributed proximal minimization

① Averaging step in parallel for all agents

$$z_i(k) = \sum_j a_j^i(k) x_j(k)$$

② Primal update for x_i in parallel for all agents

$$x_i(k+1) = \arg \min_{x_i \in X_i} f_i(x_i) + \frac{1}{2c(k)} \|x_i - z_i(k)\|^2$$

- Does this algorithm converge?

- If yes, does it provide the same solution with the centralized problem (had we been able to solve it)?

Summary

ADMM algorithm

- Convergence theorem
- Decision coupled problems come as an example

Distributed algorithms

- ... for decision coupled problems
- Step-size (proxy term) is now iteration varying
- Connectivity requirements become important
- When does it converge ? [Lecture 4](#)

Thank you for your attention !
Questions ?

Contact at :
kostas.margellos@eng.ox.ac.uk

C20 Distributed Systems Lecture 4

Kostas Margellos

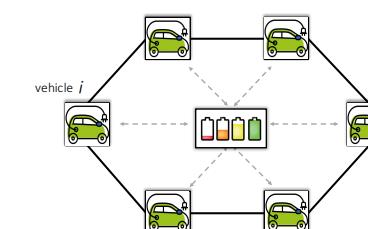
University of Oxford



Recap : Distributed algorithms

Decision coupled problems

$$\begin{aligned} & \text{minimize} \sum_i f_i(x) \\ & \text{subject to} \\ & \quad x \in X_i, \forall i = 1, \dots, m \end{aligned}$$



Proximal minimization algorithm

Proximal minimization algorithm

① Averaging step in parallel for all agents

$$z_i(k) = \sum_j a_j^i(k) x_j(k)$$

② Primal update for x_i in parallel for all agents

$$x_i(k+1) = \arg \min_{x_i \in X_i} f_i(x_i) + \frac{1}{2c(k)} \|x_i - z_i(k)\|^2$$

- No dual variables introduced – primal only method
- All steps can be parallelized across agents – no central authority !

Algorithm analysis : Assumptions

① Convexity and compactness

- $f_i(\cdot)$: convex for all i
- X_i : compact, convex, non-empty interior for all i
 \Rightarrow There exists a Slater point, i.e. $\exists \text{Ball}(\bar{x}, \rho) \subset \cap_i X_i$

Distributed proximal minimization

① Averaging step in parallel for all agents

$$z_i(k) = \sum_j a_j^i(k) x_j(k)$$

② Primal update for x_i in parallel for all agents

$$x_i(k+1) = \arg \min_{x_i \in X_i} f_i(x_i) + \frac{1}{2c(k)} \|x_i - z_i(k)\|^2$$

- Does this algorithm converge ?
- If yes, does it provide the same solution with the centralized problem
 $(\text{had we been able to solve it if we had access to } f_i\text{'s and } X_i\text{'s})$?

Algorithm analysis : Assumptions

① Convexity and compactness

- $f_i(\cdot)$: convex for all i
- X_i : compact, convex, non-empty interior for all i
 \Rightarrow There exists a Slater point, i.e. $\exists \text{Ball}(\bar{x}, \rho) \subset \cap_i X_i$

② Information mix

- Weights $a_j^i(k)$: non-zero lower bound if link between $i - j$ present
 \Rightarrow Info mixing at a non-diminishing rate
- Weights $a_j^i(k)$: form a doubly stochastic matrix (sum of rows and columns equals one)
 \Rightarrow Agents influence each other equally in the long run

$$\sum_j a_j^i(k) = 1, \forall i$$

$$\sum_i a_j^i(k) = 1, \forall j$$

Algorithm analysis : Assumptions

③ Choice of the proxy term

- $\{c(k)\}_k$: non-increasing
- Should not decrease too fast

$$\sum_k c(k) = \infty \quad [\text{to approach set of optimizers}]$$

$$\sum_k c(k)^2 < \infty \quad [\text{to achieve convergence}]$$

- E.g., harmonic series

$$c(k) = \frac{\alpha}{k+1}, \text{ where } \alpha \text{ is any constant}$$

Notice that $\lim_{k \rightarrow \infty} c(k) = 0$, i.e. as iterations increase we penalize "disagreement" more

Algorithm analysis : Assumptions

③ Network connectivity – All information flows (eventually)

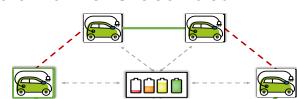
Connectivity

Let (V, E_k) be a directed graph, where V : nodes/agents, and $E_k = \{(j, i) : a_j^i(k) > 0\}$: edges Let

$$E_\infty = \{(j, i) : (j, i) \in E_k \text{ for infinitely many } k\}.$$

(V, E_∞) is strongly connected and (kind of) periodic, i.e., for any two nodes there exists a path of directed edges that connects.

- Any pair of agents communicates infinitely often,
- Intercommunication time is bounded



Algorithm analysis : Assumptions

③ Network connectivity – All information flows (eventually)

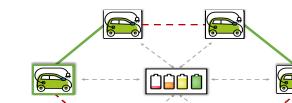
Connectivity

Let (V, E_k) be a directed graph, where V : nodes/agents, and $E_k = \{(j, i) : a_j^i(k) > 0\}$: edges Let

$$E_\infty = \{(j, i) : (j, i) \in E_k \text{ for infinitely many } k\}.$$

(V, E_∞) is strongly connected and (kind of) periodic, i.e., for any two nodes there exists a path of directed edges that connects.

- Any pair of agents communicates infinitely often,
- Intercommunication time is bounded



Algorithm analysis : Assumptions

③ Network connectivity – All information flows (eventually)

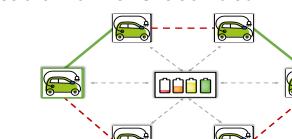
Connectivity

Let (V, E_k) be a directed graph, where V : nodes/agents, and $E_k = \{(j, i) : a_j^i(k) > 0\}$: edges Let

$$E_\infty = \{(j, i) : (j, i) \in E_k \text{ for infinitely many } k\}.$$

(V, E_∞) is strongly connected and (kind of) periodic, i.e., for any two nodes there exists a path of directed edges that connects.

- Any pair of agents communicates infinitely often,
- Intercommunication time is bounded



Algorithm analysis : Assumptions

- ③ Network connectivity – All information flows (eventually)

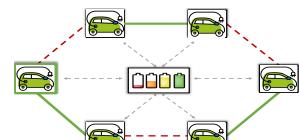
Connectivity

Let (V, E_k) be a directed graph, where V : nodes/agents, and $E_k = \{(j, i) : a_j^i(k) > 0\}$: edges Let

$$E_\infty = \{(j, i) : (j, i) \in E_k \text{ for infinitely many } k\}.$$

(V, E_∞) is strongly connected and (kind of) periodic, i.e., for any two nodes there exists a path of directed edges that connects.

- Any pair of agents communicates infinitely often,
- Intercommunication time is bounded



Algorithm analysis : Assumptions

- ③ Network connectivity – All information flows (eventually)

Connectivity

Let (V, E_k) be a directed graph, where V : nodes/agents, and $E_k = \{(j, i) : a_j^i(k) > 0\}$: edges Let

$$E_\infty = \{(j, i) : (j, i) \in E_k \text{ for infinitely many } k\}.$$

(V, E_∞) is strongly connected and (kind of) periodic, i.e., for any two nodes there exists a path of directed edges that connects.

- Any pair of agents communicates infinitely often,
- Intercommunication time is bounded



Convergence & optimality

Theorem : Convergence of distributed proximal minimization

Under the **structural + network assumptions**, the proposed proximal algorithm converges to some minimizer x^* of the centralized problem, i.e.,

$$\lim_{k \rightarrow \infty} \|x_i(k) - x^*\| = 0, \text{ for all } i$$

- Asymptotic agreement and optimality
- Rate no faster than $c(k)$ – “slow enough” to trade among the two objective terms, namely, agreement/consensus and optimality
- There are ways to speed things up : **Average gradient tracking methods**, i.e. instead of exchanging their tentative decisions, agents exchange their tentative gradients.

Example

Two-agent problem

Let $\alpha > 0$ and $1 < M < \infty$, and consider the problem :

$$\begin{aligned} & \text{minimize}_{x \in \mathbb{R}} \alpha(x+1)^2 + \alpha(x-1)^2 \\ & \text{subject to } x \in [-M, M] \end{aligned}$$

- ➊ What is the optimal solution ?
- ➋ Compute it by means of the distributed proximal minimization algorithm using
 - Time-invariant mixing weights $a_j^i(k) = \frac{1}{2}$ for all iterations k
 - Take $c(k) = \frac{1}{k+1}$
 - Initialize with $x_1(0) = -1$ and $x_2(0) = 1$

- ➌ Treat this as a two-agent decision coupled problem

Example (cont'd)

Two-agent problem equivalent reformulation

Let $\alpha > 0$ and $1 < M < \infty$, $s_1 = 1$, $s_2 = -1$, and consider

$$\min_{x \in \mathbb{R}} \quad \sum_{i=1,2} \alpha(x + s_i)^2$$

subject to $x \in [-M, M]$

- Agents' objective functions : $f_i(x) = \alpha(x + s_i)^2$, for $i = 1, 2$
- Objective function becomes : $2\alpha x^2 + 2\alpha$. Since $\alpha > 0$ its minimum is achieved at $x^* = 0$

Example (cont'd)

Main distributed proximal minimization updates

- Information mixing for $i = 1, 2$ (under our choice for mixing weights) :

$$z_i(k) = \frac{x_1(k) + x_2(k)}{2}$$

- Local computation for $i = 1, 2$:

$$\begin{aligned} x_i(k+1) &= \Pi_{[-M,M]} \left[\frac{z_i(k) - s_i 2\alpha c(k)}{2\alpha c(k) + 1} \right] \\ &= \begin{cases} \min \left(\frac{z_i(k) - s_i 2\alpha c(k)}{2\alpha c(k) + 1}, M \right), & \text{if } \frac{z_i(k) - s_i 2\alpha c(k)}{2\alpha c(k) + 1} \geq 0 \\ \max \left(\frac{z_i(k) - s_i 2\alpha c(k)}{2\alpha c(k) + 1}, -M \right), & \text{otherwise,} \end{cases} \end{aligned}$$

- What happens to $z_i(k)$ under our initialization choice ?

Example (cont'd)

Main distributed proximal minimization updates

- Information mixing for $i = 1, 2$ (under our choice for mixing weights) :

$$z_i(k) = \frac{x_1(k) + x_2(k)}{2}$$

- Local computation for $i = 1, 2$:

$$x_i(k+1) = \arg \min_{x_i \in [-M, M]} \alpha(x_i + s_i)^2 + \frac{1}{2c(k)} \|x_i - z_i(k)\|^2$$

- Information mixing is the same for all agents : $z_1(k) = z_2(k)$
- Local computation : Find unconstrained minimizer and project it on $[-M, M]$
- Unconstrained minimizer :

$$\frac{z_i(k) - s_i 2\alpha c(k)}{2\alpha c(k) + 1}$$

Example (cont'd)

We will show by means of induction that $z_1(k) = z_2(k) = 0$

- Step 1** : For $k = 0$, and since $x_1(0) = -1$ and $x_2(0) = 1$, we have that

$$z_i(0) = \frac{x_1(0) + x_2(0)}{2} = 0, \text{ for } i = 1, 2$$

- Step 2** : Induction hypothesis $z_1(k) = z_2(k) = 0$

- Step 3** : Show that $z_i(k+1) = 0$

$$\begin{aligned} x_i(k+1) &= \begin{cases} \min \left(\frac{-s_i 2\alpha c(k)}{2\alpha c(k)+1}, M \right), & \text{if } \frac{-s_i 2\alpha c(k)}{2\alpha c(k)+1} \geq 0 \\ \max \left(\frac{-s_i 2\alpha c(k)}{2\alpha c(k)+1}, -M \right), & \text{otherwise,} \end{cases} \\ &= -s_i \frac{2\alpha c(k)}{2\alpha c(k)+1}, \end{aligned}$$

where the first equality is due to the induction hypothesis, and the second is due to the fact that $\left| \frac{-s_i 2\alpha c(k)}{2\alpha c(k)+1} \right| < 1$ and $M > 1$, so the argument is never "clipped" to $\pm M$

Example (cont'd)

We will show by means of induction that $z_1(k) = z_2(k) = 0$

- ① Step 1 : For $k = 0$, and since $x_1(0) = -1$ and $x_2(0) = 1$, we have that

$$z_i(0) = \frac{x_1(0) + x_2(0)}{2} = 0, \text{ for } i = 1, 2$$

- ② Step 2 : Induction hypothesis $z_1(k) = z_2(k) = 0$

- ③ Step 3 : Show that $z_i(k+1) = 0$

$$\begin{aligned} x_i(k+1) &= \begin{cases} \min\left(\frac{-s_i 2\alpha c(k)}{2\alpha c(k)+1}, M\right), & \text{if } \frac{-s_i 2\alpha c(k)}{2\alpha c(k)+1} \geq 0 \\ \max\left(\frac{-s_i 2\alpha c(k)}{2\alpha c(k)+1}, -M\right), & \text{otherwise,} \end{cases} \\ &= -s_i \frac{2\alpha c(k)}{2\alpha c(k)+1} \end{aligned}$$

- Since $s_1 + s_2 = 0$ we then have that

$$z_i(k+1) = \frac{x_1(k+1) + x_2(k+1)}{2} = -\frac{\alpha c(k)}{2\alpha c(k)+1}(s_1 + s_2) = 0$$

Distributed projected gradient algorithm

Main update steps :

- ① Averaging step **in parallel** for all agents

$$z_i(k) = \sum_j a_j^i(k) x_j(k)$$

- ② Primal update for x_i **in parallel** for all agents (projection step)

$$x_i(k+1) = \Pi_{X_i}[z_i(k) - c(k) \nabla f_i(z_i(k))]$$

- Looks similar with the distributed proximal minimization
- $\nabla f_i(z_i(k))$ denotes the gradient of f_i evaluated at $z_i(k)$
- The x -update is no longer "best response" but is replaced by the gradient step

$$z_i(k) - c(k) \nabla f_i(z_i(k))$$

projected on the set X_i

Example (cont'd)

Since $z_i(k) = 0$ for all k , the x -update steps become

x -update steps for $i = 1, 2$,

$$\begin{aligned} x_i(k+1) &= -s_i \frac{2\alpha c(k)}{2\alpha c(k)+1} \\ &= -s_i \frac{2\alpha}{2\alpha+k+1} \end{aligned}$$

- As iterations increase, i.e. $k \rightarrow \infty$ we obtain that

$$\lim_{k \rightarrow \infty} x_i(k+1) = 0 = x^*$$

- In other words, the distributed proximal minimization algorithm converges to the minimum of the decision coupled problem

Distributed projected gradient algorithm

Main update steps :

- ① Averaging step **in parallel** for all agents

$$z_i(k) = \sum_j a_j^i(k) x_j(k)$$

- ② Primal update for x_i **in parallel** for all agents (projection step)

$$x_i(k+1) = \Pi_{X_i}[z_i(k) - c(k) \nabla f_i(z_i(k))]$$

- The proxy term $c(k)$ plays the role of the (diminishing) step-size along the gradient direction
- Convergence to the optimum under the same assumptions with distributed proximal minimization algorithm

Distributed projected gradient algorithm

Relationship with distributed proximal minimization

- Proximal algorithms can be equivalently written as a gradient step

$$\begin{aligned} \mathbf{x}_i(k+1) &= \arg \min_{\mathbf{x}_i \in \mathcal{X}_i} f_i(\mathbf{x}_i) + \frac{1}{2c(k)} \|\mathbf{x}_i - \mathbf{z}_i(k)\|^2 \\ \Leftrightarrow \mathbf{x}_i(k+1) &= \Pi_{\mathcal{X}_i} [\mathbf{z}_i(k) - c(k) \nabla f_i(\mathbf{x}_i(k+1))] \end{aligned}$$

- Notice that this is no a recursion but an identity satisfied by $\mathbf{x}_i(k+1)$ as this appears on both sides of the last equality
- What happens if we replace in the right-hand side the most updated information available to agent i at iteration k , i.e. $\mathbf{z}_i(k)$?

$$\mathbf{x}_i(k+1) = \Pi_{\mathcal{X}_i} [\mathbf{z}_i(k) - c(k) \nabla f_i(\mathbf{z}_i(k))]$$

- ... we obtain the distributed projected gradient algorithm !

Summary

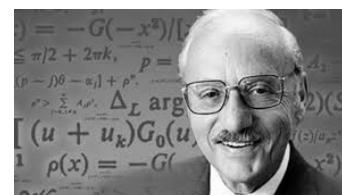
Distributed algorithms for decision coupled problems

- Distributed proximal minimization
 - Step-size (proxy term) is now iteration varying
 - Convergence under assumptions on step-size, mixing weights and network connectivity
- Distributed projected gradient
 - Rather than “best response” performs projected gradient step
 - Same convergence assumptions with proximal minimization

True optimization is the revolutionary contribution of modern research to decision processes.

– George Dantzig, November 8, 1914 – May 13, 2005

Thank you for your attention !
Questions ?



Contact at :
kostas.margellos@eng.ox.ac.uk