Distributed Optimization in Multiagent Systems
The Consensus Problem
The Consensus Problem

No single agent knows the target function to optimize.
The Consensus Problem

No single agent knows the target function to optimize
Formally

\[
\min_{x \in X} \sum_{n=1}^{N} f_n(x)
\]

- \(N = \) number of nodes / agents
- \(X = \mathbb{R}^d\)
- \(f_n\) is the cost function of agent \(n\)
- Two agents \(n\) and \(m\) can exchange messages if \(n \sim m\)

Numerous works on that problem
Early work: Tsitsiklis ’84
Example #1: Wireless Sensor Networks

\[ Y_n = \text{random observation of sensor } n \]
\[ x = \text{unknown parameter to be estimated} \]

\[ p(Y_1, \cdots, Y_N; x) = p_1(Y_1; x) \cdots p_N(Y_N; x) \]

The maximum likelihood estimate writes

\[ \hat{x} = \arg \max_x \sum_n \ln p_n(Y_n; x) \]

[Schizas’08, Moura’11]
Example #2: Machine Learning

Data set formed by $T$ samples $(X_i, Y_i)$ ($i = 1 \ldots T$)

- $Y_i =$ variable to be explained
- $X_i =$ explanatory features

$$\min_x \sum_{i=1}^{T} \ell(x^T X_i, Y_i) + r(x)$$

Split data into $N$ batches

$$\min_x \sum_{n=1}^{N} \sum_{i} \ell(x^T X_{i,n}, Y_{i,n}) + r(x)$$

n.b.: some problems are more involved (I. Colin’16)

$$\min_x \sum_{i} \sum_{j} f(x; X_i, Y_i, X_j, Y_j) + r(x)$$
Example #3: Resource Allocation

Let $x_n$ be the resource of an agent $n$

- Agents share a resource $b$: $\sum_n x_n \leq b$
- Agent $n$ gets reward $R_n(x_n)$ for using resource $x_n$
- Maximize the global reward

$$\max_{x: \sum_n x_n \leq b} \sum_{n=1}^N R_n(x_n)$$

The dual of a sharing problem is a consensus problem.
Networks

Parallel:

Distributed:
Outline

Distributed gradient descent

Distributed Alternating Direction Method of Multipliers (D-ADMM)

Total Variation Regularization on Graphs
Outline

Distributed gradient descent

Distributed Alternating Direction Method of Multipliers (D-ADMM)

Total Variation Regularization on Graphs
Adapt-and-combine (Tsitsiklis’84)

- [Local step] Each agent $n$ generates a temporary update

$$\tilde{x}^{k+1}_n = x^k_n - \gamma_k \nabla f_n(x^k_n)$$

- [Agreement step] Connected agents merge their temporary estimates

$$x^{k+1}_n = \sum_{m \sim n} A(n, m) \tilde{x}^{k+1}_m$$

where $A$ satisfies technical constraints (must be doubly stochastic)
Adapt-and-combine (Tsitsiklis’84)

- **[Local step]** Each agent $n$ generates a temporary update
  \[
  \tilde{x}^{k+1}_n = x^k_n - \gamma_k \nabla f_n(x^k_n)
  \]

- **[Agreement step]** Connected agents merge their temporary estimates
  \[
  x^{k+1}_n = \sum_{m \sim n} A(n, m) \tilde{x}^{k+1}_m
  \]
  where $A$ satisfies technical constraints (must be doubly stochastic)

**Convergence rates** (e.g. [Nedic’09], [Duchi’12])
- Decreasing step size $\gamma_k \to 0$ is needed in general
- Sublinear converges rates
More problems

1. **Asynchronism**
   Some agents are active at time \( n \), others aren’t
   Random link failures

2. **Noise**
   Gradients may be observed up to a random noise (online algorithms)

3. **Constraints**

   Minimize \( \sum_{n=1}^{N} f_n(x) \) subject to \( x \in C \)

   \[
   \tilde{x}_n^{k+1} = \text{proj}_C [x_n^k - \gamma_k (\nabla f_n(x_n^k) + \text{noise})]
   \]

   \[
   x_n^{k+1} = \sum_{m \sim n} A_{k+1}(n, m) \tilde{x}_m^{k+1}
   \]
Distributed stochastic gradient algorithm

Under technical conditions,

**Convergence (Bianchi et al.'12):** $x_n^k$ tends to a KKT point $x^*$

**Convergence rate (Morral et. al'12):** If $x^* \in \text{int}(C)$

\[
\sqrt{\gamma_k^{-1}}(x_n^k - x^*) \xrightarrow{\mathcal{L}} \mathcal{N}(0, \Sigma_{OPT} + \Sigma_{NET})
\]

- $\Sigma_{OPT}$ is the covariance corresponding to the centralized setting
- $\Sigma_{NET}$ is the excess variance due to the distributed setting

**Remark:** $\Sigma_{NET} = 0$ for some protocols which can be characterized
Outline

Distributed gradient descent

Distributed Alternating Direction Method of Multipliers (D-ADMM)

Total Variation Regularization on Graphs
Alternating Direction Method of Multipliers

Consider the generic problem

$$\min_x F(x) + G(Mx)$$

where $F, G$ are convex. Rewrite as a constrained problem

$$\min_{z=Mx} F(x) + G(z)$$

The augmented Lagrangian is:

$$\mathcal{L}_\rho(x, z; \lambda) = F(x) + G(z) + \langle \lambda, Mx - z \rangle + \frac{\rho}{2} \|Mx - z\|^2$$

**ADMM**

$$x^{k+1} = \arg \min_x \mathcal{L}_\rho(x, z^k; \lambda^k) \rightarrow \text{only } F \text{ needed}$$

$$z^{k+1} = \arg \min_z \mathcal{L}_\rho(x^{k+1}, z; \lambda^k) \rightarrow \text{only } G \text{ needed}$$

$$\lambda^{k+1} = \lambda^k + \rho(Mx^{k+1} - z^{k+1})$$
Back to our problem

All functions \( f_n : X \to \mathbb{R} \) are assumed convex. Consider the problem:

\[
\min_{u \in X} \sum_{n=1}^{N} f_n(u)
\]

Main trick: Define

\[
F : x = (x_1, \ldots, x_N) \mapsto \sum_{n} f_n(x_n)
\]

Equivalent problem:

\[
\min_{x \in X^N} F(x) + \iota_{sp(1)}(x)
\]

where \( \iota_{sp(1)}(x) = \begin{cases} 
0 & \text{if } x_1 = \cdots = x_N \\
+\infty & \text{otherwise} 
\end{cases} \)

- \( F \) is separable in \( x_1, \ldots, x_N \)
- \( G = \iota_{sp(1)} \) couples the variables but is simple
ADMM illustrated

Set $\bar{x}^k = \frac{1}{N} \sum_n x_n^k$

Algorithm (see e.g. [Boyd’11])

For all $n$, 
\[
\lambda_n^k = \lambda_{n}^{k-1} + \rho(x_n^k - \bar{x}^k)
\]
\[
x_{n}^{k+1} = \text{arg min}_{y} f_n(y) + \frac{\rho}{2} \| \bar{x}^k - \rho^{-1} \lambda_n^k - y \|^2
\]

1. Transmit current estimates

Diagram:

1. Transmit current estimates

\[
\begin{align*}
\lambda_n^k &= \lambda_{n}^{k-1} + \rho(x_n^k - \bar{x}^k) \\
x_{n}^{k+1} &= \text{arg min}_{y} f_n(y) + \frac{\rho}{2} \| \bar{x}^k - \rho^{-1} \lambda_n^k - y \|^2
\end{align*}
\]
ADMM illustrated

Set $\bar{x}^k = \frac{1}{N} \sum_n x_n^k$

Algorithm (see e.g. [Boyd’11])

For all $n$, 

$$\lambda_n^k = \lambda_n^{k-1} + \rho (x_n^k - \bar{x}^k)$$

$$x_n^{k+1} = \arg \min_y f_n(y) + \frac{\rho}{2} \| \bar{x}^k - \rho^{-1} \lambda_n^k - y \|^2$$

2. Compute average $\bar{x}^k$
ADMM illustrated

Set $\bar{x}^k = \frac{1}{N} \sum_n x^k_n$

Algorithm (see e.g. [Boyd’11])

For all $n$, $\lambda_n^k = \lambda_n^{k-1} + \rho(x_n^k - \bar{x}^k)$

$x_{n}^{k+1} = \arg \min_y f_n(y) + \frac{\rho}{2} \| \bar{x}^k - \rho^{-1} \lambda_n^k - y \|^2$

3. Transmit $\bar{x}^k$ to all agents
ADMM illustrated

Set \( \bar{x}^k = \frac{1}{N} \sum_n x_n^k \)

Algorithm (see e.g. [Boyd’11])

For all \( n \),
\[
\lambda_n^k = \lambda_n^{k-1} + \rho (x_n^k - \bar{x}^k) \\
x_n^{k+1} = \arg \min_y f_n(y) + \frac{\rho}{2} \| \bar{x}^k - \rho^{-1} \lambda_n^k - y \|^2
\]

4. Compute \( \lambda_n^k, x_n^{k+1} \) for all \( n \)
ADMM illustrated

Set $\bar{x}^k = \frac{1}{N} \sum_n x_n^k$

Algorithm (see e.g. [Boyd'11])

For all $n$, 

$$\lambda_n^k = \lambda_n^{k-1} + \rho (x_n^k - \bar{x}^k)$$

$$x_n^{k+1} = \arg \min_y f_n(y) + \frac{\rho}{2} \| \bar{x}^k - \rho^{-1} \lambda_n^k - y \|^2$$

4. Compute $\lambda_n^k, x_n^{k+1}$ for all $n$

The algorithm is parallel but not distributed on the graph
Subgraph consensus

Let $A_1, A_2, \ldots, A_L$ be subsets of agents

$A_1 = \{1, 3\}$, $A_2 = \{2, 3\}$, $A_3 = \{3, 4, 5\}$
Subgraph consensus

Let $A_1, A_2, \ldots, A_L$ be subsets of agents

$A_1 = \{1, 3\}$, $A_2 = \{2, 3\}$, $A_3 = \{3, 4, 5\}$

$$\begin{pmatrix} x_1 \\ x_3 \end{pmatrix} \in \text{sp}\left( \frac{1}{1} \right)$$
Subgraph consensus

Let $A_1, A_2, \ldots, A_L$ be subsets of agents

$$A_1 = \{1, 3\}, \ A_2 = \{2, 3\}, \ A_3 = \{3, 4, 5\}$$

Consensus within subgraphs $\iff$ global consensus

$$(\begin{array}{c} x_1 \\ x_3 \end{array}) \in \text{sp}(\frac{1}{1})$$

$$(\begin{array}{c} x_2 \\ x_3 \end{array}) \in \text{sp}(\frac{1}{1})$$
Subgraph consensus

Let $A_1, A_2, \ldots, A_L$ be subsets of agents

$A_1 = \{1, 3\}$, $A_2 = \{2, 3\}$, $A_3 = \{3, 4, 5\}$
Subgraph consensus

Let $A_1, A_2, \cdots, A_L$ be subsets of agents

$A_1 = \{1, 3\}, \ A_2 = \{2, 3\}, \ A_3 = \{3, 4, 5\}$

consensus within subgraphs $\Leftrightarrow$ global consensus
Example (Cont.)

The initial problem is

$$\min_{x \in \mathbb{X}^N} F(x) + G(Mx)$$

where

$$Mx = \begin{pmatrix} x_1 \\ x_3 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix}$$

that is:

$$M = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

and where $G$ is the indicator function of the subspace of vectors of the form

$$\begin{pmatrix} \alpha \\ \alpha \\ \beta \\ \beta \\ \delta \\ \delta \end{pmatrix}$$
Distributed ADMM illustrated

For all $n$, $\Lambda_n^k = \Lambda_n^{k-1} + \rho(x_n^k - x_n^k)$

$x_n^{k+1} = \arg\min_y f_n(y) + \frac{\rho|\sigma_n|}{2} \|x_n^k - \rho^{-1}\Lambda_n^k - y\|^2$

where $|\sigma_n| = \text{number of “neighbors” of } n$

1. For each subgraph, compute average $\bar{x}_{A_1}^k$
Distributed ADMM illustrated

Distributed ADMM (early works by [Schizas'08])

For all \( n \),

\[
\Lambda_n^k = \Lambda_n^{k-1} + \rho(x_n^k - \chi_n^k)
\]

\[
x_{n+1}^k = \arg \min_y f_n(y) + \frac{\rho|\sigma_n|}{2} \| \chi_n^k - \rho^{-1} \Lambda_n^k - y \|^2
\]

where \( |\sigma_n| = \text{number of "neighbors" of } n \)

1. For each subgraph, compute average \( \overline{x}_{A_2}^k \)
Distributed ADMM illustrated

For all $n$, 
\[
\Lambda_n^k = \Lambda_n^{k-1} + \rho (x_n^k - \chi_n^k)
\]

\[
x_n^{k+1} = \arg \min_y f_n(y) + \frac{\rho |\sigma_n|}{2} \| \chi_n^k - \rho^{-1} \Lambda_n^k - y \|^2
\]

where $|\sigma_n| = \text{number of “neighbors” of } n$

1. For each subgraph, compute average $\overline{x}_{A_\ell}^k$
Distributed ADMM illustrated

Distributed ADMM (early works by [Schizas’08])

For all \( n \),

\[
\Lambda_n^k = \Lambda_n^{k-1} + \rho (x_n^k - \chi_n^k)
\]

\[
x_n^{k+1} = \arg\min_y f_n(y) + \frac{\rho |\sigma_n|}{2} \|\chi_n^k - \rho^{-1} \Lambda_n^k - y\|^2
\]

where \( |\sigma_n| = \text{number of "neighbors" of } n \)

2. For each \( n \), compute \( \chi_n^k = \text{Average}(\bar{x}_{A_\ell}^k : \ell \text{ s.t. } n \in A_\ell) \)
Distributed ADMM illustrated

Distributed ADMM (early works by [Schizas'08])

For all $n$, \[
\Lambda_n^k = \Lambda_n^{k-1} + \rho(x_n^k - x_n^k)
\]
\[
x_n^{k+1} = \arg \min_y f_n(y) + \frac{\rho|\sigma_n|}{2} \|x_n^k - \rho^{-1}\Lambda_n^k - y\|^2
\]

where $|\sigma_n| = \text{number of “neighbors” of } n$

3. For each $n$, compute $\lambda_n^k$ and $x_n^{k+1}$
Linear convergence of the Distributed ADMM

Assumption: \( H_\star := \sum_n \nabla^2 f_n(x_\star) > 0 \) at the minimizer \( x_\star \)

\[
\|x_n^k - x_\star\| \sim \alpha^k \quad \text{as} \; k \to \infty
\]

- [Shi et al.’ 13] non-asymptotic bound but pessimistic
- [Iutzeler et al.’ 16] asymptotic but tight

\[
\frac{1}{k} \log \|x^k - 1 \otimes x_\star\| \quad \text{as a function of} \; k
\]
Example: ring network

Define $\alpha = \lim_{k \to \infty} \|x_n^k - x^*\|^{1/k}$

Set $f_n : \mathbb{R} \to \mathbb{R}$ and $f''(x^*) = \sigma^2$

$\alpha \geq \sqrt{\frac{1 + \cos \frac{2\pi}{N}}{2(1 + \sin \frac{2\pi}{N})}}$ with equality when $\rho = \frac{\sigma^2}{2 \sin \frac{2\pi}{N}}$
Asynchronous D-ADMM

- All agents must complete their arg min computation before combining
- The network waits for the slowest agents

Our objective: allow for asynchronism
Revisiting ADMM as a fixed point algorithm

Set $\zeta^k = \lambda^k + \rho z^k$. Fact: $\lambda^k = P(\zeta^k)$ where $P$ is a projection.

ADMM can be written as a fixed point algorithm [Gabay,83] [Eckstein,92]

$$\zeta^{k+1} = J(\zeta^k)$$

where $J$ is firmly non-expansive i.e.,

$$\|J(x) - J(y)\|^2 \leq \|x - y\|^2 - \|(I - J)(x) - (I - J)(y)\|^2$$
Random coordinate descent

Introducing the block-components of $\zeta^{k+1} = J(\zeta^k)$:

$$
\begin{pmatrix}
\zeta_{1}^{k+1} \\
\vdots \\
\zeta_{\ell}^{k+1} \\
\vdots \\
\zeta_{L}^{k+1}
\end{pmatrix}
= 
\begin{pmatrix}
J_{1}(\zeta^k) \\
\vdots \\
J_{\ell}(\zeta^k) \\
\vdots \\
J_{L}(\zeta^k)
\end{pmatrix}
$$
Random coordinate descent

If only one block $\ell = \ell(k + 1)$ is active at time $k + 1$:

$$
\begin{pmatrix}
\zeta_1^{k+1} \\
\vdots \\
\zeta_{\ell}^{k+1} \\
\vdots \\
\zeta_{L}^{k+1}
\end{pmatrix}
= 
\begin{pmatrix}
\zeta_1^k \\
\vdots \\
J_{\ell}(\zeta^k) \\
\vdots \\
\zeta_{L}^k
\end{pmatrix}
$$
Random coordinate descent

If only one block $\ell = \ell(k + 1)$ is active at time $k + 1$:

\[
\begin{pmatrix}
\zeta_{1}^{k+1} \\
\vdots \\
\zeta_{\ell}^{k+1} \\
\vdots \\
\zeta_{L}^{k+1}
\end{pmatrix}
= 
\begin{pmatrix}
\zeta_{1}^{k} \\
\vdots \\
J_{\ell}(\zeta^{k}) \\
\vdots \\
\zeta_{L}^{k}
\end{pmatrix}
\]

Convergence of the Asynchronous ADMM [Iutzeler’13]

This algorithm still converges if active components are chosen at random

Main idea: For a well-chosen norm $\| . \|$ and a fixed point $\zeta^*$ of $J$, prove

\[
\mathbb{E} \left( \| \zeta^{k+1} - \zeta^* \|^2 \mid \mathcal{F}_k \right) \leq \| \zeta^k - \zeta^* \|^2
\]

$\Rightarrow \zeta^k$ is getting “stochastically” closer to $\zeta^*$
Asynchronous ADMM explicitied

Activate two nodes $A_\ell = \{m, n\}$
Asynchronous ADMM explicitation

Activate two nodes $A_\ell = \{m, n\}$

- Agent $n$ computes

$$x_{n}^{k+1} = \arg\min_{x} f_{n}(x) + \sum_{j \sim k} \left( \langle x, \lambda_{j,n}^{k} \rangle + \frac{\rho}{2} \|x - \bar{x}_{j,n}^{k}\|^2 \right)$$

and similarly for Agent $m$. 

![Node illustration]

21/25
Asynchronous ADMM explicited

Activate two nodes $A_\ell = \{m, n\}$

- Agent $n$ computes
  \[ x_n^{k+1} = \text{arg min}_{x} f_n(x) + \sum_{j \sim k} \left( \langle x, \lambda_{j,n}^k \rangle + \frac{\rho}{2} \|x - \bar{x}_{j,n}^k\|^2 \right) \]
  and similarly for Agent $m$.

- They exchange $x_m^{k+1}$ and $x_n^{k+1}$

- Agent $n$ computes
  \[ \bar{x}_{m,n}^{k+1} = \frac{1}{2}(x_m^{k+1} + x_n^{k+1}), \]
  \[ \lambda_{m,n}^{k+1} = \lambda_{m,n}^k + \rho \frac{x_n^{k+1} - x_m^{k+1}}{2} \]
  and similarly for Agent $m$. 
Generalization: Distributed Vũ-Condat algorithm

- Vũ-Condat algorithm generalizes ADMM (allows “gradients” evaluations)
- Distributed Vũ-Condat algorithm is applicable using the same principle
- Bianchi’16, Fercoq’17 provide a random coordinate descent version
- The algorithm is asynchronous at the node level and not at the edge level
Stochastic Optimization

\[
\min_{x \in \mathcal{X}} \sum_{n=1}^{N} \mathbb{E}(f_n(x, \xi_n))
\]

- Law of $\xi_n$ unknown, but revealed on-line through random copies $\xi_1^n, \xi_2^n, \ldots$
- **Stochastic approximation**: at time $k$, replace the unknown function $\mathbb{E}(f_n(., \xi_n))$ by its random version $f_n(., \xi^k_n)$
  
  Example: stochastic gradient descent

- **Thesis of A. Salim**: Stochastic versions of generic optimization algorithms (Forward-Backward, Douglas-Rachford, ADMM, Vô-Condat, etc.)

- Byproduct: distributed stochastic algorithms
Outline

Distributed gradient descent

Distributed Alternating Direction Method of Multipliers (D-ADMM)

Total Variation Regularization on Graphs
Total variation regularization (1/2)

Notation: On a graph $G = (V, E)$, the total variation of $x \in \mathbb{R}^V$ is

$$TV(x) = \sum_{\{i,j\} \in E} |x_i - x_j|$$

General problem:

$$\min_{x \in \mathbb{R}^V} F(x) + TV(x)$$

- Trend filtering: $F(x) = \frac{1}{2} \|x - m\|^2$ where $m \in \mathbb{R}^V$ are noisy measurements
- Graph inpainting: complete possibly missing measurements on the nodes

Proximal gradient algorithm:

$$x_{n+1} = \text{prox}_{\gamma TV}(x_n - \gamma \nabla F(x_n))$$

- Computing $\text{prox}_{TV}$ is difficult over large unstructured graphs
- But efficient algorithms exist for 1D-graphs (Mammen’97) (Condat’13)
Total variation regularization (2/2)

**Write TV as an expectation:** Let $\xi$ be simple random walk in $G$ of fixed length

$$TV_G(x) \propto \mathbb{E}(TV_\xi(x))$$

**Algorithm (Salim’16) At time $n$,**

- Draw a random walk $\xi_{n+1}$
- Compute $x_{n+1} = \text{prox}_{\gamma_n TV_\xi} (x_n - \gamma_n \nabla F(x_n)) \to$ easy, 1D

Hidden difficulty: one should avoid loops when choosing the walk...

---

Trend filtering example. Cost function vs time(s). Stochastic block model $10^5$ nodes, $25.10^6$ edges.

Blue: Stochastic proximal gradient, Green: dual proximal gradient, Red: dual L-BFGS-B