Tutorial on Distributed Optimization

Amal Feriani

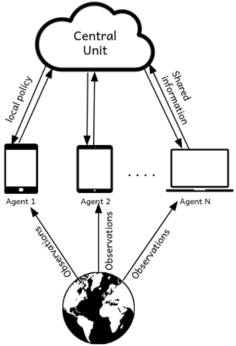
University of Manitoba

December 17, 2023

Table of Contents

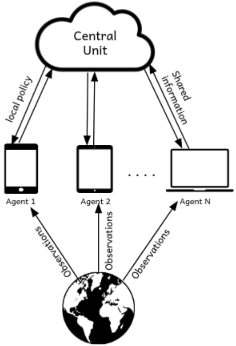
Table of Contents

Motivation

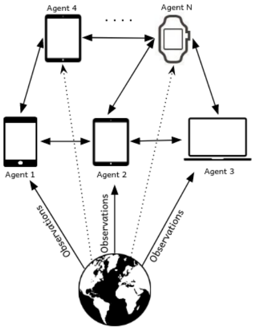


Centralized

Motivation

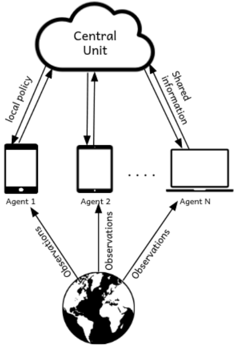


Centralized

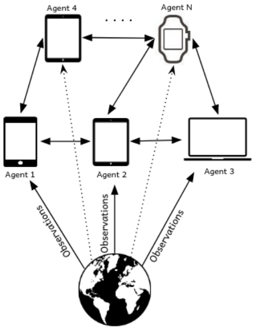


Networked Agents

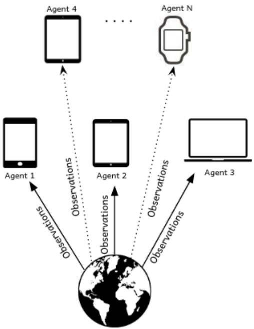
Motivation



Centralized



Networked Agents



Fully Decentralized

Motivation

- Fully decentralized learning scheme

Motivation

- Fully decentralized learning scheme
  - Non-stationarity problem
  - Local solutions may not necessarily lead to global ones
  - Non-cooperative setting

Motivation

- Fully decentralized learning scheme
  - Non-stationarity problem
  - Local solutions may not necessarily lead to global ones
  - Non-cooperative setting
- Centralized learning scheme

Motivation

- Fully decentralized learning scheme
  - Non-stationarity problem
  - Local solutions may not necessarily lead to global ones
  - Non-cooperative setting
- Centralized learning scheme
  - Existence of a central coordinator

Motivation

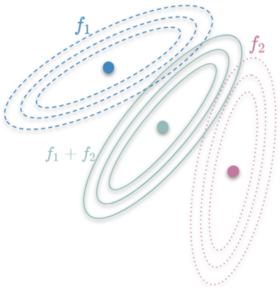
- Fully decentralized learning scheme
  - Non-stationarity problem
  - Local solutions may not necessarily lead to global ones
  - Non-cooperative setting
- Centralized learning scheme
  - Existence of a central coordinator
- Networked Optimization scheme

Motivation

- Fully decentralized learning scheme
  - Non-stationarity problem
  - Local solutions may not necessarily lead to global ones
  - Non-cooperative setting
- Centralized learning scheme
  - Existence of a central coordinator
- Networked Optimization scheme
  - Existence of a communication structure to enable coordination
  - Information exchange with neighboring nodes

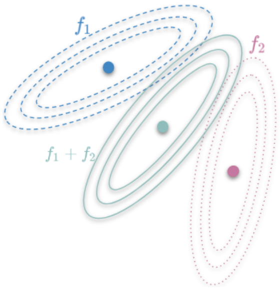
Motivation

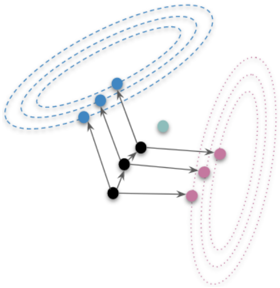
$$f(\mathbf{x}) = \frac{1}{N} \sum_{i=1}^{N} f^i(\mathbf{x}) \tag{1}$$



Motivation

$$f(\mathbf{x}) = \frac{1}{N}\sum_{i=1}^{N} f^i(\mathbf{x}) \tag{1}$$

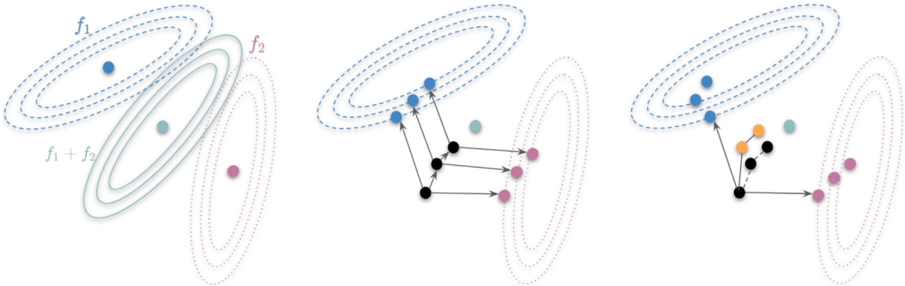




$x_k = x_{k-1} - \alpha_k \sum_{i=0}^{N} g^i(x_{k-1})$

Motivation

$$f(\mathbf{x}) = \frac{1}{N}\sum_{i=1}^{N} f^i(\mathbf{x}) \tag{1}$$

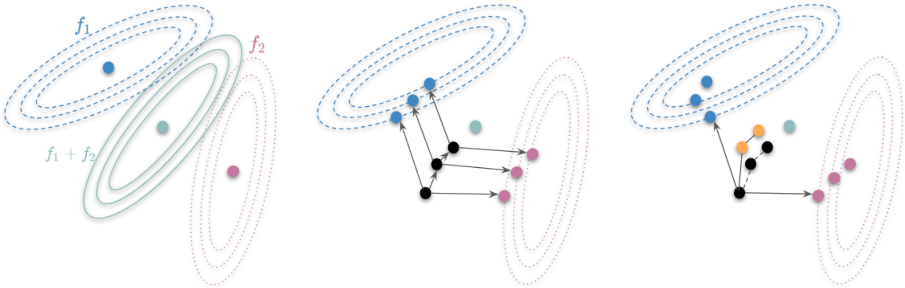


$x_k = x_{k-1} - \alpha_k \sum_{i=0}^{N} g^i(x_{k-1})$

$x_k^i = x_{k-1}^i - \alpha_k^i g^i(x_{k-1}^i)$

Motivation

$$\mathbf{f}(\mathbf{x}) = \frac{1}{N}\sum_{i=1}^{N} f^i(x) \tag{2}$$



slow (only 1 update)

Potentially faster but may not converge

Table of Contents

Decomposible Problems

- Optimization problems with *coupling* variables

$$\max_{\mathbf{x}} f(\mathbf{x}) = \frac{1}{N}\sum_{i=1}^{N} f^i(\mathbf{x})$$

Decomposible Problems

- Optimization problems with *coupling* variables

$$\max_{\mathbf{x}} f(\mathbf{x}) = \frac{1}{N}\sum_{i=1}^{N} f^i(\mathbf{x})$$

- Optimization problems with *coupling* constraints

$$\max_{x_i} \frac{1}{N}\sum_{i=1}^{N} f^i(x_i) \text{ s.t } C(\mathbf{x}) = B$$

Decomposition: Motivation

Power allocation in wireless communication

$$\max_{p_i} \sum_{i=0}^{N} \log(1 + \frac{g_i p_i}{\sigma^2})$$

$$\text{s.t} \sum_{i=0}^{N} p_i = P_{\max} \longleftarrow \text{coupling constraint}$$

Decomposition: Motivation

Decomposition benefits for MAS:

- Computational Scalability and flexibility
- Tractability
- Privacy
- ....

Table of Contents

Dual Ascent Algorithm

Consider the problem

$$\min_{\mathbf{x}} f(\mathbf{x}) \text{ s.t } A\mathbf{x} = \mathbf{b}$$

- **Lagrangian**

$$L(\mathbf{x}, \boldsymbol{u}) = f(\mathbf{x}) + (A\mathbf{x} - \mathbf{b})^T \boldsymbol{u}$$

- **Dual function**

$$g(\boldsymbol{u}) = \min_{\mathbf{x}} L(\mathbf{x}, \boldsymbol{u})$$

- **Conjugate function/Fenchel conjugate**

$$f^\star(\mathbf{y}) = \max_{\mathbf{x}}[\mathbf{y}^T\mathbf{x} - f(\mathbf{x})]$$

$$-f^\star(\mathbf{y}) = \min_{\mathbf{x}}[f(\mathbf{x}) - \mathbf{y}^T\mathbf{x}]$$

$$g(\boldsymbol{u}) = -f^\star(-A^T\boldsymbol{u}) - \mathbf{b}^T\boldsymbol{u}$$

Reminder: Conjugate functions

- If f is closed and convex, $(f^\star)^\star = f$ and

$$x \in \partial f^\star \iff y \in \partial f \iff x \in \arg\min_z f(z) - y^T z \tag{3}$$

- If f is strictly convex, $\nabla f^\star(y) = \arg\min_z f(z) - y^T z$
- $f^\star$ is always convex even if f is not convex (i.e affine in y)

Dual Ascent Algorithm

Dual Problem:

$$\max_{\boldsymbol{u}} g(\boldsymbol{u}) = min_u f^\star(-A^T\boldsymbol{u}) + \mathbf{b}^T\boldsymbol{u}$$

$$\implies \partial g(\boldsymbol{u}) = A\partial f^\star(-A^T\boldsymbol{u}) - \mathbf{b}$$

$$\implies \partial g(\boldsymbol{u}) = A\mathbf{x}^* - \mathbf{b} \text{ where } \mathbf{x}^* \in \arg\min_{\mathbf{x}} f(\mathbf{x}) + \boldsymbol{u}^T A\mathbf{x} \quad (3)$$

Apply sub/gradient methods to the dual : $u_{k+1} = u_k + t_k\partial g(u_k)$

Dual Ascent Algorithm

Dual Problem:

$$\max_{\boldsymbol{u}} g(\boldsymbol{u}) = min_u f^{\star}(-A^T\boldsymbol{u}) + \mathbf{b}^T\boldsymbol{u}$$

$$\implies \partial g(\boldsymbol{u}) = A\partial f^{\star}(-A^T\boldsymbol{u}) - \mathbf{b}$$

$$\implies \partial g(\boldsymbol{u}) = A\mathbf{x}^* - \mathbf{b} \text{ where } \mathbf{x}^* \in \arg\min_{\mathbf{x}} f(\mathbf{x}) + \boldsymbol{u}^T A\mathbf{x} \quad (3)$$

Apply sub/gradient methods to the dual : $u_{k+1} = u_k + t_k \partial g(u_k)$

Algorithm 2: Dual Sub-gradient Method

Start with an initial dual λ_0 and repeat for K iterations; t_0 initial learning rate

for $k = 1, \ldots, K$ **do**

$\quad \mathbf{x}_k \in \arg\min_{\mathbf{x}} f(\mathbf{x}) + \boldsymbol{u}_{\boldsymbol{k-1}}^T A\mathbf{x}$

$\quad \boldsymbol{u}_{\boldsymbol{k}} = \boldsymbol{u}_{\boldsymbol{k-1}} + t_k(A\mathbf{x}_k - \mathbf{b})$

end

Separable Problem Example

Primal Problem

$$\min_{x_1,x_2} f_1(x_1) + f_2(x_2)$$
$$\text{s.t } x_1 + x_2 \leq P$$

Dual Problem

$$\max_u -f_1^\star(-u) - f_2^\star(-u) - Pu$$
$$\text{s.t } u \geq 0$$

The Lagrangian is separable in f:

$$L(x_1, x_2, u) = f_1(x_1) + ux_1 + f_2(x_2) + ux_2 - Pu$$
$$g(u) = g_1(u) + g_2(u) - Pu; \quad g_i(u) = \min_{x_i} f_i(x_i) + ux_i = -f_i^\star(-u), i = 1, 2$$

Applying Dual Ascent will result in

$$u_{k+1} = u_k + t_k \text{Proj}_{u\geq 0}\left\{x_k^1 + x_k^2 - b\right\} = u_k + t_k \max\left\{x_k^1 + x_k^2 - b, 0\right\}$$
$$x_k^i = \arg\min_{x^i} f_i(x^i) + u_{k-1}x^i, \quad i = 1, 2 \implies \text{ decoupled}$$

Dual Decomposition

More generally,

$$\min_{\mathbf{x}_i} \sum_{i=1}^{B} f_i(\mathbf{x}_i) \qquad \text{separable}$$
$$\text{s.t } A\mathbf{x} = \mathbf{b} \qquad \text{not separable}$$

Dual Decomposition

More generally,

$$\min_{\mathbf{x}_i} \sum_{i=1}^{B} f_i(\mathbf{x}_i) \quad \text{separable}$$
$$\text{s.t } A\mathbf{x} = \mathbf{b} \quad \text{not separable}$$

Algorithm 4: Dual Decomposition Method (coupled constraints)

Start with an initial dual λ_0 and repeat for K iterations; t_0 initial learning rate **for** $k = 1, \ldots, K$ **do**

$\quad \mathbf{x}_k^i = \arg\min_{\mathbf{x}^i} f_i(\mathbf{x}^i) + \boldsymbol{u}_{\boldsymbol{k-1}}^T A^i \mathbf{x}^i$

$\quad \boldsymbol{u}_{\boldsymbol{k}} = \boldsymbol{u}_{\boldsymbol{k-1}} + t_k(\sum_{i=1}^{B} A^i \mathbf{x}_k^i - \mathbf{b})$

end

- The minimization problems w.r.t to x_i are independent
- Coordination is done through adjusting u to maximize $g(u)$

Dual Decomposition for coupling variables

Primal Problem

$$\min_{x_1,x_2,y} f_1(x_1,y) + f_2(x_2,y)$$

Transform the problem to

$$\begin{array}{ll} \min_{x_1,x_2,y_1,y_2} f_1(x_1,y_1) + f_2(x_2,y_2) & : y_1, y_2 \text{ are } \textbf{local copies} \text{ of } y \\ \text{s.t } y_1 = y_2 & : \textbf{consensus constraint} \end{array}$$

Algorithm 5: Dual Decomposition for coupling variables

Start with an initial dual λ_0 and repeat for K iterations; t_0 initial learning rate

for $k = 1 \ldots K$ **do**

$\quad x_k^i, y_k^i = \arg\min_{x^i,y^i} f_i(x^i,y^i) + u_{k-1}y^i, \quad i = 1,2$

$\quad u_k = u_{k-1} + t_k(y_k^1 - y_k^2)$

end

But, the convergence to a feasible iterates (i.e $y_2 - y_1 \neq 0$) is not always guaranteed.

Discussion

- The dual variable u is equivalent to a price of a shared resource
- If the resource is over-utilized (i.e $P - x_1 - x_2 < 0$), the price is increased and decreased otherwise.
- But, the price never gets negative !
- A centralized unit is needed for coordination to update u

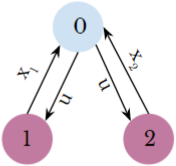


Figure: Price Coordination interpretation of DD (see slide 16)

Discussion

- **Subgradient** methods are slow, step size selection is difficult
- **Gradient** methods are applicable if the dual function is differentiable (which is not always the case, requires strict convexity of the primal function)
- The convergence guarantees of the Dual Ascent method do not necessarily guarantee the convergence of the primal (i.e Slater's Theorem)

Discussion

Problem 1

$$\min_{x_1, x_2} 2|x_1 - 2| + 4|x_2 - 4|$$
$$\text{s.t } x_1 + x_2 = 5$$
$$0 \leq x_1 \leq 10, 0 \leq x_2 \leq 10$$

The optimal solution : $x_1^* = 1, x_2^* = 4$ with optimal value is $f^* = 2$

Problem 2

$$\min_{x_1, x_2} 2(x_1 - 2)^2 + 4(x_2 - 4)^2$$
$$\text{s.t } x_1 + x_2 = 5$$
$$0 \leq x_1 \leq 10, 0 \leq x_2 \leq 10$$

The solution for this problem is $x_1^* = 4/3, x_2^* = 11/3$ and the optimal value is $f^* = 4/3$.

Discussion

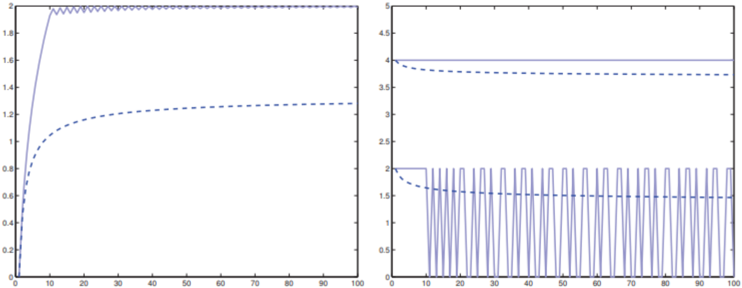


Figure: When the dual function is non-smooth, the dual objective (left, full lines) converges while the primal iterates (right, full lines) do not. When the dual function is differentiable, both objective and iterates converge asymptotically (dashed lines) [1]

Table of Contents

Augmented Lagrangian Methods (ALM)/ Methods of Multipliers

Adds a penalty term to the primal objective to ensure the smoothness of the dual function

Augmented Lagrangian Methods (ALM)/ Methods of Multipliers

Adds a penalty term to the primal objective to ensure the smoothness of the dual function

Considers the modified problem, for a parameter $\rho > 0$

$$\begin{aligned} &\min_{\mathbf{x}} f(\mathbf{x}) + \frac{\rho}{2}||A\mathbf{x} - \mathbf{b}||_2^2 \\ &\text{s.t } A\mathbf{x} = \mathbf{b} \end{aligned} \tag{4}$$

Augmented Lagrangian Methods (ALM)/ Methods of Multipliers

Adds a penalty term to the primal objective to ensure the smoothness of the dual function

Considers the modified problem, for a parameter $\rho > 0$

$$\min_{\mathbf{x}} f(\mathbf{x}) + \frac{\rho}{2}||A\mathbf{x} - \mathbf{b}||_2^2 \quad (4)$$

$$\text{s.t } A\mathbf{x} = \mathbf{b}$$

Uses the augmented Lagrangian

$$L_\rho(\mathbf{x}, \boldsymbol{u}) = f(\mathbf{x}) + \boldsymbol{u}^T(A\mathbf{x} - \mathbf{b}) + \frac{\rho}{2}||A\mathbf{x} - \mathbf{b}||_2^2 \quad (5)$$

Augmented Lagrangian Methods (ALM)/ Methods of Multipliers

Adds a penalty term to the primal objective to ensure the smoothness of the dual function

Considers the modified problem, for a parameter $\rho > 0$

$$\begin{aligned} &\min_{\mathbf{x}} f(\mathbf{x}) + \frac{\rho}{2}||A\mathbf{x} - \mathbf{b}||_2^2 \\ &\text{s.t } A\mathbf{x} = \ \mathbf{b} \end{aligned} \tag{4}$$

Uses the augmented Lagrangian

$$L_\rho(\mathbf{x}, \boldsymbol{u}) = f(\mathbf{x}) + \boldsymbol{u}^T(A\mathbf{x} - \mathbf{b}) + \frac{\rho}{2}||A\mathbf{x} - \mathbf{b}||_2^2 \tag{5}$$

Repeats, $k = 1 \dots K$

$$\begin{aligned} \mathbf{x}_k &= \arg\min_{\mathbf{x}} f(\mathbf{x}) + \boldsymbol{u}_{k-1}^T A\mathbf{x} + \frac{\rho}{2}||A\mathbf{x} - \mathbf{b}||_2^2 \\ \boldsymbol{u}_k &= \boldsymbol{u}_{\boldsymbol{k-1}} + \rho(A\mathbf{x}_k - \mathbf{b}) \end{aligned}$$

ALM convergence

- Optimality conditions

$$\text{Primal feasibility } Ax^* - b = 0; \text{ stationarity } \partial f(x^*) + A^T u^* = 0$$

- x_k minimizes $L_\rho(x, y_k)$

$$\mathbf{x}_k = \arg\min_{\mathbf{x}} f(\mathbf{x}) + \boldsymbol{\lambda}_{k-1}^T A\mathbf{x} + +\frac{\rho}{2}||A\mathbf{x} - \mathbf{b}||_2^2$$
$$\Longrightarrow 0 \in \partial f(\mathbf{x}_k) + A^T \underbrace{(\boldsymbol{\lambda}_{k-1} + \rho(A\mathbf{x}_k - \mathbf{b}))}_{\boldsymbol{\lambda}_k}$$

- the dual update makes the pair (x_k, y_k) satisfy the **stationarity** condition
- Under mild conditions, $A\mathbf{x}_k - \mathbf{b} \longrightarrow 0$ as $k \longrightarrow \infty$, KKT conditions are satisfied in the limit
- x_k and u_k converge to solutions

ALM

- The objective function stays the same
- The primal objective because strongly convex if A is full rank (f is a convex function)
- The Dual function becomes smooth $\Longrightarrow$ gradient ascent
- Better convergence guarantees compared to Dual Ascent (f can be non-differentiable, take on value $+\infty$)
- The primal objective is no longer decomposible because of the quadratic term

Proximal Operator

$$\text{prox}_{\rho f}(\tilde{x}) = \arg\min_{x} f(x) + \frac{1}{2\rho}||x - \tilde{x}||^2$$

Proximal Operator

$$\text{prox}_{\rho f}(\tilde{x}) = \arg\min_{x} f(x) + \frac{1}{2\rho}||x - \tilde{x}||^2$$

Proximal Point Algorithm (PPA)[2]

$$x_{t+1} = \text{prox}_{\rho f}(x_t) = (I + \rho \partial f)^{-1}(x_t)$$

Augmented Lagrangian and Proximal Point Methods

$$\textbf{Primal}: \min_x f(x) + g(Ax)$$
$$\textbf{Dual}: \max_u -f^\star(-A^T u) - g^\star(u) = \max_u -H(u)$$

Applying the PPA in the dual problem:

$$\begin{aligned} u_{t+1} &= \text{prox}_{\rho H}(u_t) \\ &= \arg\min_u f^\star(-A^T u) + g^\star(u) + \frac{1}{2\rho}||u - u_t||_2^2 \end{aligned}$$

Is equivalent to

$$u_{t+1} = u_t + \rho(A\tilde{x} - \tilde{y}) \text{ where}$$
$$(\tilde{x}, \tilde{y}) = \arg\min_{x,y}(f(x) + g(y) + u_t^T(Ax - y) + \frac{\rho}{2}||Ax - y||_2^2)$$

Hence, the augmented Lagrangian in the primal is the same as PPA in the dual

Example

$$\min_{\mathbf{x}} \sum_{i=1}^{N} f_i(\mathbf{x})$$

$$\begin{cases} \min_{\boldsymbol{x}_i} \sum_{i=1}^{N} f_i(\boldsymbol{x}_i) \\ \text{s.t } \mathbf{x}_i = \mathbf{x} \quad i = 1 \dots N \end{cases}$$

$g = \mathbb{1}_{\boldsymbol{x}}$ (indicator function), $A = I_N$

The augmented Lagrangian method

$$(x^i_{k+1}, x_{k+1}) = \arg\min_{x_i, x} (f_i(x_i) + u_t^T (x_i - x) + \frac{\rho}{2} ||x_i - x||_2^2)$$

$$u^i_{k+1} = u^i_k + \rho(x^i_{k+1} - x_{k+1})$$

Coupling between $\boldsymbol{x}$ and $\boldsymbol{x}_i$

Table of Contents

Alternating Direction Method of Multipliers (ADMM)

Combines the best of both methods (decomposibility and convergence guarantees) [3]

Alternating Direction Method of Multipliers (ADMM)

Combines the best of both methods (decomposibility and convergence guarantees) [3]
Consider a problem of the form:

$$\min_{\mathbf{x},\mathbf{y}} f(\mathbf{x}) + g(\mathbf{y}) \qquad (6)$$
$$\text{s.t } A\mathbf{x} + B\mathbf{y} = \mathbf{c}$$

Two blocks of variables with separable objectives

Alternating Direction Method of Multipliers (ADMM)

Combines the best of both methods (decomposibility and convergence guarantees) [3]
Consider a problem of the form:

$$\min_{\mathbf{x},\mathbf{y}} f(\mathbf{x}) + g(\mathbf{y}) \quad (6)$$
$$\text{s.t } A\mathbf{x} + B\mathbf{y} = \mathbf{c}$$

Two blocks of variables with separable objectives
The augmented Lagrangian is

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

Alternating Direction Method of Multipliers (ADMM)

Combines the best of both methods (decomposibility and convergence guarantees) [3]

Consider a problem of the form:

$$\min_{\mathbf{x},\mathbf{y}} f(\mathbf{x}) + g(\mathbf{y}) \tag{6}$$

$$\text{s.t } A\mathbf{x} + B\mathbf{y} = \mathbf{c}$$

Two blocks of variables with separable objectives

The augmented Lagrangian is

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

Applying the method of multipliers, we need to find (x_k, y_k) by solving the joint minimization:

$$(x_k, y_k) = \arg\min_{x,y} L_\rho(x, y, u_{k-1})$$

Alternating Direction Method of Multipliers (ADMM)

ADMM splits the joint minimization into two steps :

$$x_k = \arg\min_x L_\rho(x, y_{k-1}, u_{k-1}) \tag{7}$$

$$y_k = \arg\min_y L_\rho(x_k, y, u_{k-1}) \tag{8}$$

$$u_k = u_{k-1} + \rho(Ax_k + By_k - c) \tag{9}$$

$\Longrightarrow$ Splitting since we minimize over x with y fixed, and vice versa (**Gauss-Seidel method**)

Example

$$\begin{cases} \min_{\mathbf{x}_i} \sum_{i=1}^{N} f_i(\mathbf{x}_i) \\ \text{s.t } \mathbf{x}_i = \mathbf{x} \quad i = 1 \dots N \end{cases}$$

The augmented Lagrangian

$$L_\rho(x, y, u) = \sum_{i=0}^{N} f_i(x_i) + u^T(x_i - x) + \frac{\rho}{2}||x_i - x||_2^2$$

ADMM iterates

$$x_{k+1}^i = \arg\min_{x_i}(f_i(x_i) + (u_k^i)^T x_i + \frac{\rho}{2}||x_i - x_k||_2^2)$$

$$x_{k+1} = \frac{1}{N}\sum_{i=0}^{N} x_{k+1}^i + \frac{1}{\rho}u_k^i$$

$$u_{k+1}^i = u_k^i + \rho(x_{k+1}^i - x_{k+1})$$

Convergence guarantees

Under modest assumptions on f, g, the ADMM iterates satisfy, for any $\rho > 0$ [4]:

- Residual convergence/Primal feasibility: $r_k = Ax_k + By_k - c \longrightarrow 0$ as $k \longrightarrow \infty$
- Primal convergence: $f(x_k) + g(y_k) \longrightarrow f^* + g^*$
- Dual convergence: $u_k \longrightarrow u^*$

Assumption 1. The (extended-real-valued) functions $f : \mathbb{R}^n \mapsto \mathbb{R} \cup \{+\infty\}$ and and $g : \mathbb{R}^m \mapsto \mathbb{R} \cup \{+\infty\}$ are closed, proper, and convex. $\implies$ epif $= \{(x, t) \in \mathbb{R}^n \times \mathbb{R} | f(x) \leq t\}$ is a closed nonempty convex set.

Assumption 2. The unaugmented Lagrangian L has a saddle point.

Note: No further assumptions on A and B

Convergence rate: Theory still being developed but ADMM has similar convergence rate as 1st order methods (linear) (i.e [5] linear convergence rate when one of the functions is strongly convex)

Scaled form ADMM

Let the residual $r = Ax + Bz - c$ and the scaled dual variable $w = \frac{u}{\rho}$

$$u^T r + \frac{\rho}{2}||r||_2^2 = \frac{\rho}{2}||r + \frac{1}{\rho}u||_2^2 - \frac{2}{2\rho}||u||_2^2 = \frac{\rho}{2}||r + w||_2^2 - \frac{\rho}{2}||w||_2^2$$

The augmented Lagrangian becomes

$$L_\rho(x, y, w) = f(x) + g(y) + \frac{\rho}{2}||r + w|| - \frac{\rho}{2}||w||_2^2$$

And the ADMM iterates

$$x_{k+1} = \arg\min_x f(x) + \frac{\rho}{2}||Ax + By_k - c + w_k||_2^2$$

$$y_{k+1} = \arg\min_y g(y) + \frac{\rho}{2}||Ax_{k+1} + By - c + w_k||_2^2$$

$$w_{k+1} = w_k + Ax_{k+1} + By_{k+1} - c$$

The residual at iteration k, $r_k = Ax_k + By_k - c$,

$$w_{k+1} = w_0 + \sum_{i=1}^{k} r_j \longrightarrow \text{ running sum of residuals}$$

Douglas–Rachford algorithm

Consider the problem, where f and g are closed convex functions

$$\min_x f(x) + g(x)$$

Douglas–Rachford iteration [6]:

$$\begin{aligned} y_{k+1} &= y_k + \mathrm{prox}_g(2x_k - y_k) - x_k \\ x_{k+1} &= \mathrm{prox}_f(y_{k+1}) \end{aligned}$$

An equivalent formulation

$$\begin{aligned} u_{k+1} &= \mathrm{prox}_g(2x_k - y_k) \\ x_{k+1} &= \mathrm{prox}_f(y_k + u_{k+1} - x_k) \\ y_{k+1} &= y_k + u_{k+1} - x_{k+1} \end{aligned}$$

Let $w_k = x_k - y_k$

$$\begin{aligned} u_{k+1} &= \mathrm{prox}_g(x_k + w_k) \\ x_{k+1} &= \mathrm{prox}_f(u_{k+1} - w_k) \\ w_{k+1} &= w_k + x_{k+1} - u_{k+1} \end{aligned}$$

Douglas–Rachford algorithm and ADMM

ADMM is equivalent to applying Douglas–Rachford in the Dual :

$$\min_{x,y} f(x) + g(y) \text{ s.t } Ax + By = c$$

Dual Problem: $\max_z -c^T z - f^\star(-A^T z) - g^\star(-B^T z)$
Apply the Douglas–Rachford method to minimize
$\underbrace{c^T z + f^\star(-A^T z)}_{f_1(z)} + \underbrace{g^\star(-B^T z)}_{f_2(z)}$

$$u_{k+1} = \text{prox}_{f_1}(z_k + w_k) \tag{10}$$

$$z_{k+1} = \text{prox}_{f_2}(u_{k+1} - w_k) \tag{11}$$

$$w_{k+1} = w_k + z_{k+1} - u_{k+1} \tag{12}$$

Recall that the PPA algorithm in the dual is equivalent to the augmented Lagrangian in the primal

Douglas–Rachford algorithm and ADMM

(10) $\Longrightarrow$

$$\tilde{x} = \arg\min_{x} f(x) + (z_k + w_k)^T(Ax - b) + \rho/2||Ax - b||_2^2 \tag{13}$$

$$u_{k+1} = (z_k + w_k) + \rho(A\tilde{x} - b)$$

(11) $\Longrightarrow$

$$\tilde{y} = \arg\min_{y} f(y) + (z_k)^T(By) + \rho/2||A\tilde{x} + By - b||_2^2 \tag{14}$$

$$z_{k+1} = z_k + \rho(A\tilde{x} + B\tilde{y} - b) \tag{15}$$

From (13), (14) and (15), we can derive the ADMM iterates

ADMM in practice

- Modest accuracy in a handful of iterations, but it requires a large number of iterations for a highly accurate solution (like a first-order method)
- Choice of ρ can greatly influence practical convergence of ADMM:
  - ρ too large $\longrightarrow$ not enough emphasis on minimizing $f + g$
  - ρ too small $\longrightarrow$ not enough emphasis on feasibility
  - Vector Approximate Message Passing (VAMP) find the optimal ρ value under the Gaussian approximation of the messages
- Transforming a problem to ADMM format can be subtle and leads to different algorithms

ADMM for constrained convex optimization

Consider problem

$$\min_x f(x) \text{ s.t } x \in C$$

Transform the problem to ADMM format

$$\min_{x,y} f(x) + g(y) \text{ s.t } x - y = 0$$

where $g = \mathbb{I}_C$

$$x_{k+1} = \arg\min_x \left(f(x) + \frac{\rho}{2}||x - y_k + u_k||_2^2\right)$$

$$y_{k+1} = \arg\min_y \left(g(y) + \frac{\rho}{2}||x_{k+1} + u_k - y||_2^2\right) = \Pi_C(x_{k+1} + u_k)$$

$$u_{k+1} = u_k + x_{k+1} - y_{k+1}$$

ADMM for computer vision

Image Deblurring[7]

original noisy/blurred restored

Sparse + Low rank matrix decomposition[8]

(a) Original frames (b) Low-rank $\hat{L}$ (c) Sparse $\hat{S}$

Table of Contents

Consensus ADMM

Consider the problem

$$\min_{x} \sum_{i=0}^{N} f_i(x)$$

The **Consensus ADMM** starts by transforming the problem to

$$\min_{x_i, x} \sum_{i=0}^{N} f_i(x_i) \text{ s.t } x_i = x, \quad i = 1 \ldots N$$

Thus, the decomposoble ADMM iterates

$$x_k^i = \arg\min_{x_i} f_i(x_i) + \rho/2||x_i - x_{k-1} + w_{k-1}^i||_2^2 \quad i = 1, \ldots, N$$

$$x_k = 1/N \sum_{i=0}^{N} (x_k^i + w_{k-1}^i)$$

$$w_k^i = w_{k-1}^i + x_k^i - x_k, \quad i = 1, \ldots, N$$

Consensus ADMM

By taking $x_k = \bar{x} = 1/N \sum_{i=0}^{N} x_k^i$, the previous iterates can be simplified to

$$x_k^i = \arg\min_{x_i} f_i(x_i) + \rho/2||x_i - \bar{x}_{k-1} + w_{k-1}^i||_2^2 \quad i = 1, \ldots, N$$

$$w_k^i = w_{k-1}^i + x_k^i - \bar{x}_k, \quad i = 1, \ldots, N$$

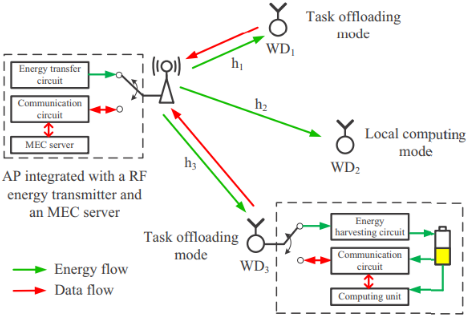
- the $x_i, i = 1, \ldots, B$ updates are done **in parallel**
- Consensus ADMM is **communication inefficient** since we need to gather all the x_k^i to update x_k and thus w_k^i
  Recently, Group ADMM (GADMM) [9] is proposed to tackle this problem, where a worker i only communicates to two neighbors

$$\min_{x_i} \sum_{i=0}^{N} f_i(x_i) \text{ s.t } x_i = x_{i+1}, \quad i = 1 \ldots N$$

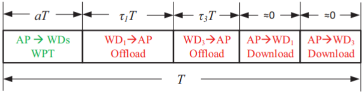
Task Offloading in MEC [10]

- MEC newtork with an AP and N wireless devices (WD)
- AP brodcasts RF energy to distributed WDs and the WDs harvest that energy for aT time, $a \in [0, 1]$ in each time frame
- Within each time frame, we assume that each WD needs to accomplish a certain computing task (pollution level) based on its local data (measurements)
- WD decides to compute the task locally (mode 0) or offload it to the AP (mode 1) (binary offloading)
- $M = M_0 \cup M_1 = \{1 \dots N\}$ the set of WDs
- Harvested energy $E_i = \mu P h_i a T, i = 1 \dots N$, P is RF energy transmit power of the AP, $\mu \in [0, 1]$ is the energy harvesting efficiency, h_i is the channel gain between the AP and the i-th WD
- Transmission time $\tau_i T$, $\tau \in [0, 1]$

Task Offloading in MEC



An example 3-user wireless powered MEC system with binary computation offloading



An example time allocation in the 3-user wireless powered MEC network. Only WD1 and WD3 selecting mode 1 offload the task to and download the computation results from the AP

- **Local computation model**

$$r_i = \nu_1 (\frac{h_i}{k_i})^{1/3} a^{1/3} \tag{16}$$

where r_i is the local computation rate (bits/s), k_i denotes the computation energy efficiency coefficient of the processor's chip and ν_1 is a fixed parameter

- **offloading mode**

$$r_i = \frac{B\tau_i}{\nu_u} \log_2(1 + \frac{\mu P a h_i^2}{\tau_i N_0}) \forall i \in M_1 \tag{17}$$

where ν_u indicates the communication overhead in task offloading, B the bandwidth and N_0 the noise power

Problem formulation

$$
\begin{aligned}
(P1): \underset{\mathcal{M}_0, a, \boldsymbol{\tau}}{\text{maximize}} \quad & \sum_{i \in \mathcal{M}_0} w_i \eta_1 \left(\frac{h_i}{k_i}\right)^{\frac{1}{3}} a^{\frac{1}{3}} && \text{(7a)} \\
& + \sum_{j \in \mathcal{M}_1} w_j \varepsilon \tau_j \ln\left(1 + \frac{\eta_2 h_j^2 a}{\tau_j}\right) && \text{(7b)} \\
\text{subject to} \quad & \sum_{j \in \mathcal{M}_1} \tau_j + a \leq 1, && \text{(7c)} \\
& a \geq 0, \ \tau_j \geq 0, \ \forall j \in \mathcal{M}_1, && \text{(7d)} \\
& \mathcal{M}_0 \subseteq \mathcal{M}, \ \mathcal{M}_1 = \mathcal{M} \setminus \mathcal{M}_0. && \text{(7e)}
\end{aligned}
$$

ADMM formulation

$$
\begin{aligned}
\underset{a,\mathbf{z},\mathbf{x},\boldsymbol{\tau},\mathbf{m}}{\text{maximize}} \quad & \sum_{i=1}^{N} w_i \Bigg\{ (1-m_i)\,\eta_1 \left(\frac{h_i}{k_i}\right)^{\frac{1}{3}} x_i^{\frac{1}{3}} \\
& + m_i \varepsilon \tau_i \ln\left(1+\frac{\eta_2 h_i^2 x_i}{\tau_i}\right) \Bigg\} \\
\text{subject to} \quad & \sum_{i=1}^{N} z_i + a \le 1, \\
& x_i = a,\, z_i = \tau_i \;\; i = 1,\cdots,N, \\
& a, z_i, x_i, \tau_i \ge 0,\;\; m_i \in \{0,1\},\;\; i = 1,\cdots,N.
\end{aligned} \tag{8}
$$

Reformulate (P1) as an equivalent integer programming problem by introducing binary decision variables m_i and additional artificial variables x_i and z_i

ADMM formulation

$$\begin{aligned} \underset{a,\mathbf{z},\mathbf{x},\boldsymbol{\tau},\mathbf{m}}{\text{maximize}} \quad & \sum_{i=1}^{N} q_i(x_i, \tau_i, m_i) + g(\mathbf{z}, a) && \text{(9a)} \\ \text{subject to} \quad & x_i = a, \tau_i = z_i \ i = 1, \cdots, N, && \text{(9b)} \\ & x_i, \tau_i \geq 0, \ m_i \in \{0,1\}, \ i = 1, \cdots, N, && \text{(9c)} \end{aligned}$$

where

$$q_i(x_i, \tau_i, m_i) = w_i \left\{ (1 - m_i)\, \eta_1 \left(\frac{h_i}{k_i} \right)^{\frac{1}{3}} x_i^{\frac{1}{3}} + m_i \varepsilon \tau_i \ln \left(1 + \frac{\eta_2 h_i^2 x_i}{\tau_i} \right) \right\},$$

and

$$g(\mathbf{z}, a) = \begin{cases} 0, & \text{if } (\mathbf{z}, a) \in \mathcal{G}, \\ -\infty, & \text{otherwise,} \end{cases} \qquad (10)$$

where

$$\mathcal{G} = \left\{ (\mathbf{z}, a) \mid \sum_{i=1}^{N} z_i + a \leq 1, a \geq 0, z_i \geq 0, i = 1, \cdots, N \right\}.$$

Notice that $i \in M_0$ can be removed from the constraint set

ADMM formulation

$$L\left(\mathbf{u},\mathbf{v},\boldsymbol{\theta}\right)=\sum_{i=1}^{N} q_i(\mathbf{u})+g(\mathbf{v})+\sum_{i=1}^{N}\beta_i\left(x_i-a\right)$$
$$+\sum_{i=1}^{N}\gamma_i\left(\tau_i-z_i\right)-\frac{c}{2}\sum_{i=1}^{N}\left(x_i-a\right)^2-\frac{c}{2}\sum_{i=1}^{N}\left(\tau_i-z_i\right)^2$$

ADMM formulation

$$L(\mathbf{u}, \mathbf{v}, \boldsymbol{\theta}) = \sum_{i=1}^{N} q_i(\mathbf{u}) + g(\mathbf{v}) + \sum_{i=1}^{N} \beta_i (x_i - a)$$
$$+ \sum_{i=1}^{N} \gamma_i (\tau_i - z_i) - \frac{c}{2} \sum_{i=1}^{N} (x_i - a)^2 - \frac{c}{2} \sum_{i=1}^{N} (\tau_i - z_i)^2$$

Step 1: maximize w.r.t to $u = \{x, \tau, m\}$

$$u_{t+1} \arg\max_{u} L(u, v_t, \theta_t) \tag{18}$$

ADMM formulation

$$L(\mathbf{u}, \mathbf{v}, \boldsymbol{\theta}) = \sum_{i=1}^{N} q_i(\mathbf{u}) + g(\mathbf{v}) + \sum_{i=1}^{N} \beta_i (x_i - a)$$
$$+ \sum_{i=1}^{N} \gamma_i (\tau_i - z_i) - \frac{c}{2} \sum_{i=1}^{N} (x_i - a)^2 - \frac{c}{2} \sum_{i=1}^{N} (\tau_i - z_i)^2$$

Step 1: maximize w.r.t to $u = \{x, \tau, m\}$

$$u_{t+1} \arg\max_{u} L(u, v_t, \theta_t) \tag{18}$$

$$\begin{cases} \underset{x_i, \tau_i \geq 0}{\text{maximize}} & w_i \eta_1 \left(\frac{h_i}{k_i}\right)^{\frac{1}{3}} x_i^{\frac{1}{3}} + \beta_i^l x_i + \gamma_i^l \tau_i - \frac{c}{2}\left(x_i - a^l\right)^2 \\ & -\frac{c}{2}\left(\tau_i - z_i^l\right)^2, \text{ if } m_i = 0, \\ \underset{x_i, \tau_i \geq 0}{\text{maximize}} & w_i \varepsilon \tau_i \ln\left(1 + \frac{\eta_2 h_i^2 x_i}{\tau_i}\right) + \beta_i^l x_i + \gamma_i^l \tau_i \\ & -\frac{c}{2}\left(x_i - a^l\right)^2 - \frac{c}{2}\left(\tau_i - z_i^l\right)^2, \text{ if } m_i = 1. \end{cases}$$

ADMM Algorithm

$$L\left(\mathbf{u},\mathbf{v},\boldsymbol{\theta}\right)=\sum_{i=1}^{N}q_i(\mathbf{u})+g(\mathbf{v})+\sum_{i=1}^{N}\beta_i\left(x_i-a\right)$$

$$+\sum_{i=1}^{N}\gamma_i\left(\tau_i-z_i\right)-\frac{c}{2}\sum_{i=1}^{N}\left(x_i-a\right)^2-\frac{c}{2}\sum_{i=1}^{N}\left(\tau_i-z_i\right)^2$$

ADMM Algorithm

$$L(\mathbf{u}, \mathbf{v}, \boldsymbol{\theta}) = \sum_{i=1}^{N} q_i(\mathbf{u}) + g(\mathbf{v}) + \sum_{i=1}^{N} \beta_i (x_i - a) + \sum_{i=1}^{N} \gamma_i (\tau_i - z_i) - \frac{c}{2} \sum_{i=1}^{N} (x_i - a)^2 - \frac{c}{2} \sum_{i=1}^{N} (\tau_i - z_i)^2$$

Step 1: maximize w.r.t to $u = \{x, \tau, m\}$

$$u_{t+1} \arg\max_{u} L(u, v_t, \theta_t) \tag{19}$$

ADMM Algorithm

$$L(\mathbf{u}, \mathbf{v}, \boldsymbol{\theta}) = \sum_{i=1}^{N} q_i(\mathbf{u}) + g(\mathbf{v}) + \sum_{i=1}^{N} \beta_i (x_i - a)$$
$$+ \sum_{i=1}^{N} \gamma_i (\tau_i - z_i) - \frac{c}{2} \sum_{i=1}^{N} (x_i - a)^2 - \frac{c}{2} \sum_{i=1}^{N} (\tau_i - z_i)^2$$

Step 1: maximize w.r.t to $u = \{x, \tau, m\}$

$$u_{t+1} \arg\max_{u} L(u, v_t, \theta_t) \tag{19}$$

$$\begin{cases} \underset{x_i, \tau_i \geq 0}{\text{maximize}} & w_i \eta_1 \left(\frac{h_i}{k_i}\right)^{\frac{1}{3}} x_i^{\frac{1}{3}} + \beta_i^l x_i + \gamma_i^l \tau_i - \frac{c}{2}\left(x_i - a^l\right)^2 \\ & -\frac{c}{2}\left(\tau_i - z_i^l\right)^2, \text{ if } m_i = 0, \\ \underset{x_i, \tau_i \geq 0}{\text{maximize}} & w_i \varepsilon \tau_i \ln\left(1 + \frac{\eta_2 h_i^2 x_i}{\tau_i}\right) + \beta_i^l x_i + \gamma_i^l \tau_i \\ & -\frac{c}{2}\left(x_i - a^l\right)^2 - \frac{c}{2}\left(\tau_i - z_i^l\right)^2, \text{ if } m_i = 1. \end{cases}$$

ADMM Algorithm

Step 2: maximize w.r.t to $v = \{z, a\}$

$$u_{t+1} \arg\max_{u} L(u_{t+1}, v, \theta_t) \tag{20}$$

ADMM Algorithm

Step 2: maximize w.r.t to $v = \{z, a\}$

$$u_{t+1} \arg\max_{u} L(u_{t+1}, v, \theta_t) \tag{20}$$

$$\begin{aligned}
&\mathbf{v}^{l+1} = \\
&\underset{\mathbf{z},a}{\arg\text{maximize}} \quad \sum_{i=1}^{N} \beta_i^l \left(x_i^{l+1} - a\right) + \sum_{i=1}^{N} \gamma_i^l \left(\tau_i^{l+1} - z_i\right) \\
&\qquad\qquad - \frac{c}{2} \sum_{i=1}^{N} \left(x_i^{l+1} - a\right)^2 - \frac{c}{2} \sum_{i=1}^{N} \left(\tau_i^{l+1} - z_i\right)^2 \\
&\text{subject to} \quad \sum_{i=1}^{N} z_i + a \leq 1, \;\; a \geq 0, \;\; z_i \geq 0, i = 1, \cdots, N.
\end{aligned}$$

ADMM Algorithm

Step 2: maximize w.r.t to $v = \{z, a\}$

$$u_{t+1} \arg\max_{u} L(u_{t+1}, v, \theta_t) \tag{20}$$

$$\begin{aligned}
&\mathbf{v}^{l+1} = \\
&\arg\underset{\mathbf{z},a}{\text{maximize}} \quad \sum_{i=1}^{N} \beta_i^l \left(x_i^{l+1} - a\right) + \sum_{i=1}^{N} \gamma_i^l \left(\tau_i^{l+1} - z_i\right) \\
&\qquad\qquad - \frac{c}{2} \sum_{i=1}^{N} \left(x_i^{l+1} - a\right)^2 - \frac{c}{2} \sum_{i=1}^{N} \left(\tau_i^{l+1} - z_i\right)^2 \\
&\text{subject to} \quad \sum_{i=1}^{N} z_i + a \leq 1, \;\; a \geq 0, \;\; z_i \geq 0, i = 1, \cdots, N.
\end{aligned}$$

Step 3: minimize w.r.t to $\theta = \{\beta, \gamma\}$

$$\begin{aligned}
\beta_i^{l+1} &= \beta_i^l - c(x_i^{l+1} - a^{l+1}), \;\; i = 1, \cdots, N, \\
\gamma_i^{l+1} &= \gamma_i^l - c(\tau_i^{l+1} - z_i^{l+1}), \;\; i = 1, \cdots, N.
\end{aligned}$$

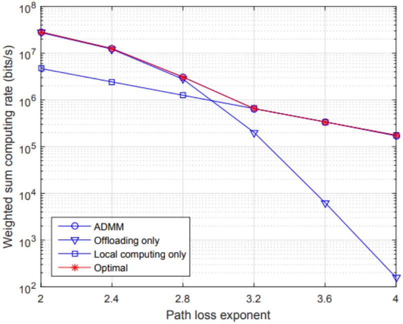
Results I

Baselines

- **Optimal**: exhaustively enumerates all the 2^N combinations of N WDs' computing modes;
- **Offloading only**: all the WDs offload their tasks to the AP, $M_0 = \{\}$;
- **Local computing only**: all the WDs perform computations locally, $M_0 = M$.

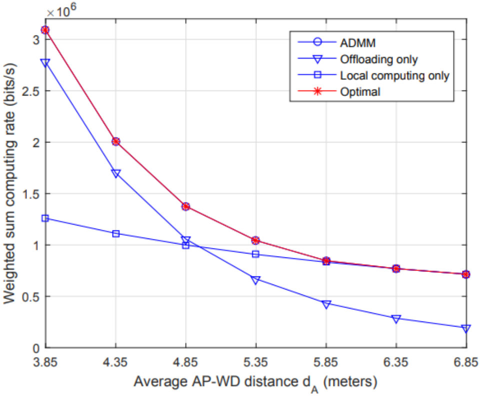
Baselines

- **Optimal**: exhaustively enumerates all the 2^N combinations of N WDs' computing modes;
- **Offloading only**: all the WDs offload their tasks to the AP, $M_0 = \{\}$;
- **Local computing only**: all the WDs perform computations locally, $M_0 = M$.



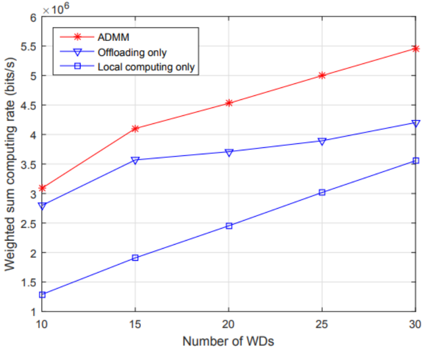
Under different path loss exponents (N=10)

Results II



Under different average AP-WD distance (N=10)

Results III


×10^6
ADMM
Offloading only
Local computing only
Weighted sum computing rate (bits/s)
Number of WDs
6
5.5
5
4.5
4
3.5
3
2.5
2
1.5
1
10
15
20
25
30

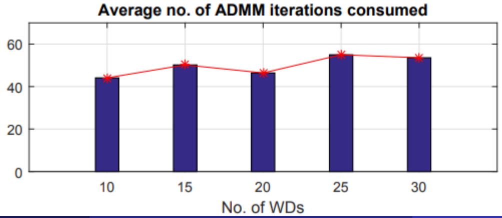

Average no. of ADMM iterations consumed
60
40
20
0
10
15
20
25
30
No. of WDs

Table of Contents

- Decomposition is necessary for decentralized optimization
- Coordination between sub-problems can be costly and communication inefficient
- ADMM offers good convergence guarantees with mild assumptions on the objective functions
- ADMM may result in different algorithms when the problem is formulated differently

What we didn't cover

- More on consensus algorithms
- Recent work on distributed gradient algorithms

Lecture Notes I

- ECE236C - Optimization Methods for Large-Scale Systems, Prof. L. Vandenberghe, UCLA
- EE364b - Convex Optimization II, Prof. S. Boyd, Stanford
- 10-725: Convex Optimization, Prof. R. Tibshirani, CMU

References I

[1] B. Yang and M. Johansson, *Distributed Optimization and Games: A Tutorial Overview*, vol. 406, pp. 109–148.
10 2010.

[2] R. T. Rockafellar, "Monotone operators and the proximal point algorithm," *SIAM journal on control and optimization*, vol. 14, no. 5, pp. 877–898, 1976.

[3] R. Glowinski and A. Marroco, "Sur l'approximation, par éléments finis d'ordre un, et la résolution, par pénalisation-dualité d'une classe de problèmes de dirichlet non linéaires," *ESAIM: Mathematical Modelling and Numerical Analysis - Modélisation Mathématique et Analyse Numérique*, vol. 9, no. R2, pp. 41–76, 1975.

References II

[4] S. Boyd, N. Parikh, and E. Chu, *Distributed optimization and statistical learning via the alternating direction method of multipliers*.
2011.

[5] R. Nishihara, L. Lessard, B. Recht, A. Packard, and M. Jordan, "A general analysis of the convergence of admm," in *International Conference on Machine Learning*, pp. 343–352, PMLR, 2015.

[6] J. Eckstein and D. P. Bertsekas, "On the douglas—rachford splitting method and the proximal point algorithm for maximal monotone operators," *Mathematical Programming*, vol. 55, no. 1-3, pp. 293–318, 1992.

[7] D. O'Connor and L. Vandenberghe, "Primal-dual decomposition by operator splitting and applications to image deblurring," *SIAM Journal on Imaging Sciences*, vol. 7, no. 3, pp. 1724–1754, 2014.

[8] E. J. Candès, X. Li, Y. Ma, and J. Wright, "Robust principal component analysis?," *Journal of the ACM (JACM)*, vol. 58, no. 3, pp. 1–37, 2011.

[9] A. Elgabli, J. Park, A. S. Bedi, M. Bennis, and V. Aggarwal, "Gadmm: Fast and communication efficient framework for distributed machine learning.," *Journal of Machine Learning Research*, vol. 21, no. 76, pp. 1–39, 2020.

[10] S. Bi and Y.-J. A. Zhang, "An admm based method for computation rate maximization in wireless powered mobile-edge computing networks," in *2018 IEEE International Conference on Communications (ICC)*, pp. 1–7, IEEE, 2018.