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.

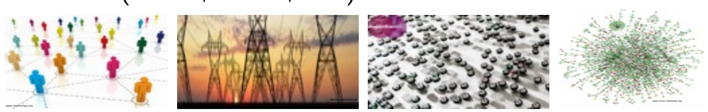
Nedich, Ozdaglar & Parrilo (2010)
Constrained consensus and optimization in multi-agent networks,
IEEE Transactions on Automatic Control, 55(4), 922–938.

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.)



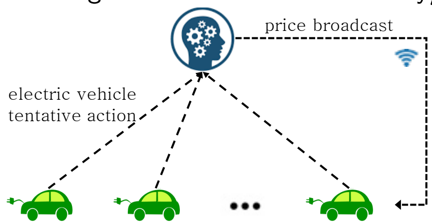
  - Large scale infrastructures
  - Multi-agent – Multiple interacting entities/users
  - Heterogeneous – Different physical or technological constraints per agent ; different objectives per agent
- Challenge : Optimizing the performance of a network ...
  - Computation : Problem size too big !
  - Communication : 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 ?

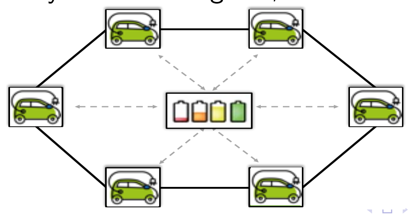
1. Scalable methodology
   - Communication :
     - Decentralized : With some central authority
     - Distributed : Only between neighbours
   - Computation : Only local ; in parallel for all agents
2. Information privacy
   - Agents do not reveal information about their preferences (encoded by objective 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

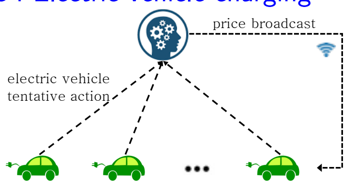


   Decentralized vs. Centralized : Agents "broadcast" only tentative information not everything
2. Distributed : Only with some agents, termed neighbours



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

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

$$\text{subject to : } x_i \in X_i, \text{ for all } i \quad \texttt{[limitations on the charging rate]}$$

Multi-agent problem classes

Cost coupled problems

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

- 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

Multi-agent problem classes

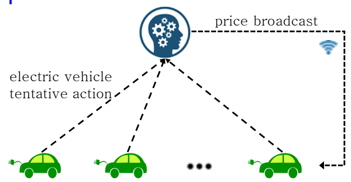
Decision coupled problems

$$\begin{aligned} &\text{minimize} \sum_{i=1}^{m} f_i(x) \\ &\text{subject to} \\ &\qquad x \in X_i, \ \forall i = 1, \ldots, m \end{aligned}$$

- Agents have a common decision : x for all agents
- Agents have separate constraint sets : X_i for agent i
- Agents have separate objective functions : f_i for agent i

Multi-agent problem classes

Constraint coupled problems : 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 independent of others consumption

$$\begin{aligned} &\text{minimize} \sum_i c_i^\top x_i && \texttt{[charging cost]} \\ &\text{subject to : } x_i \in X_i, \text{ for all } i && \texttt{[limitations on the charging rate]} \\ &\qquad \sum_i \left(A_i x_i - \frac{b}{m}\right) \leq 0 && \texttt{[power grid constraint]} \end{aligned}$$

Multi-agent problem classes

Constraint coupled problems (cont'd)

$$\begin{aligned} &\text{minimize} \sum_{i=1}^{m} f_i(x_i) \\ &\text{subject to} \\ &\qquad x_i \in X_i, \ \forall i = 1, \ldots, m \\ &\qquad \sum_{i=1}^{m} g_i(x_i) \leq 0 \end{aligned}$$

- Agents have separate decisions : x_i for agent i
- Agents have separate constraint sets : X_i for agent i
- Agents have a common constraint that couples their decisions, i.e. $\sum_i g_i(x_i) \leq 0$

Can we transform one problem class to another ?

From decision coupled to constraint coupled problems

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

- Introduce m new decision vectors, as many as the agents : $x_i, \quad i = 1, \ldots, m$
- Introduce consistency constraints : make sure all those auxiliary decisions are the same, i.e. $x_i = x$ for all $i = 1, \ldots, m$
- Price to pay : Number of constraints grows with the number of agents

Can we transform one problem class to another ?
From cost coupled to constraint coupled problems

$$\begin{aligned} &\text{minimize } \gamma = \sum_i \frac{\gamma}{m} \\ &\text{subject to} \\ &\qquad x_i \in X_i, \ \forall i = 1, \ldots, m \\ &\qquad F(x_1, \ldots, x_m) \leq \gamma \end{aligned}$$

- Introduce an additional scalar epigraphic variable γ
- Move coupling to the constraints, i.e. $F(x_i, \ldots, x_m) \leq \gamma$
- 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 ?
From decision coupled to cost coupled problems

$$\begin{aligned} &\text{minimize } F(x_1, \ldots, x_m) = \sum_i f_i(x) + \mathrm{I}_{X_i}(x) \\ &\text{subject to : no constraints} \end{aligned}$$

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

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

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

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

Part I : Decentralized algorithms
Cost coupled problems

Cost coupled problems [1]

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

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

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

Mathematical prelims : Lipschitz & Contraction mappings

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

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

- 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

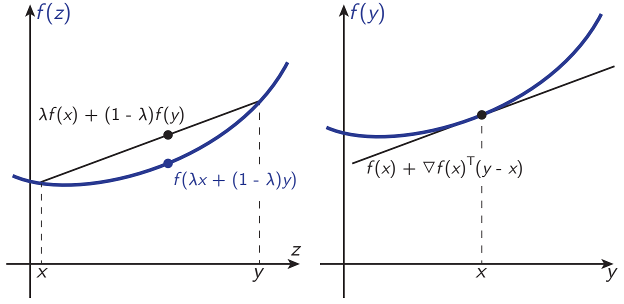
Convergence of contractive iterations

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

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

$$\|x(k) - x^\star\| \le \alpha^k \|x(0) - x^\star\|, \text{ for all } k \ge 0$$

Mathematical prelims : Convexity vs strong convexity



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

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

- 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

- Iterative algorithm

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

For each iteration $k = 1, \dots$

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

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

end for

The Jacobi algorithm

- Agents coupled via a single objective function

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

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

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

- 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 $x^\star$ of the cost coupled problem such that

$$\lim_{k\to\infty} \|x(k) - x^\star\| = 0$$

- 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 ?

The regularized Jacobi algorithm

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

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

- 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

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

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

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

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

The Gauss-Seidel algorithm

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

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

- 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 $\lim_{k\to\infty} \|F(x(k)) - F^\star\| = 0$

Summary

Decentralized algorithms for cost coupled problems

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

- 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
  $\Rightarrow$ For quadratic functions $x^\top Qx$:
  – convex if $Q \geq 0$; strongly convex if $Q > 0$
  – Strong convexity = strict convexity

Thank you for your attention !
Questions ?

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

C20 Distributed Systems
Lecture 2

Kostas Margellos

University of Oxford

Recap

Decentralized algorithms for cost coupled problems

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

- 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
  $\Rightarrow$ For quadratic functions $x^\top Qx$:
  – convex if $Q \geq 0$; strongly convex if $Q > 0$
  – Strong convexity = strict convexity

Decision coupled problems – The primal

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

subject to

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

- Decentralized solution roadmap
  1. The main algorithm for this is the Alternating Direction Method of Multipliers (ADMM)
  2. The predecessor of ADMM is the Augmented Lagrangian algorithm
  3. The Augmented Lagrangian is in turn based on the Proximal algorithm

Proximal $\Longrightarrow$ Augmented Lagrangian $\Longrightarrow$ ADMM

The proximal minimization algorithm

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

Standard minimization program

$$\text{minimize } F(x)$$

subject to : $x \in X$

Proximal minimization program

$$\text{minimize } F(x) + \frac{1}{2c}\|x - y\|^2$$

subject to : $x \in X, \ y \in \mathbb{R}^n$

- 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

The proximal minimization algorithm

- The following problems are equivalent

Standard minimization program

$$\text{minimize } F(x)$$

subject to : $x \in X$

Proximal minimization program

$$\text{minimize } F(x) + \frac{1}{2c}\|x - y\|^2$$

subject to : $x \in X, \ y \in \mathbb{R}^n$

Proximal algorithm :

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

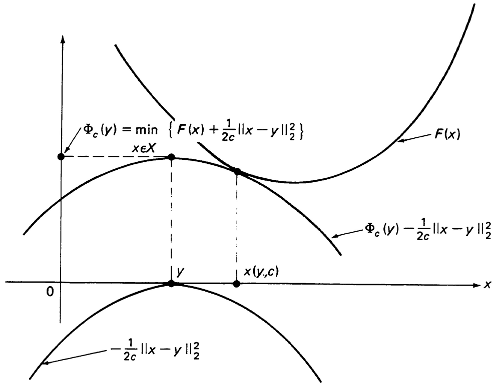
... or

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

The proximal minimization algorithm
Geometric interpretation

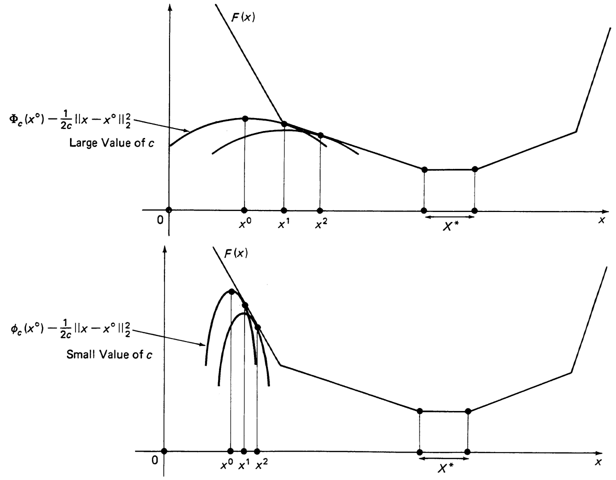
- Let $\Phi_c(y) = \min F(x) + \frac{1}{2c}\|x-y\|^2$ achieved at $x = x(y,c)$
- Hence, $\Phi_c(y) = F(x_{(y,c)}) + \frac{1}{2c}\|x_{(y,c)} - y\|^2 \leq F(x) + \frac{1}{2c}\|x-y\|^2$

$$\Rightarrow \; \Phi_c(y) - \frac{1}{2c}\|x-y\|^2 \leq F(x), \text{ with equality at } x = x(y,c)$$



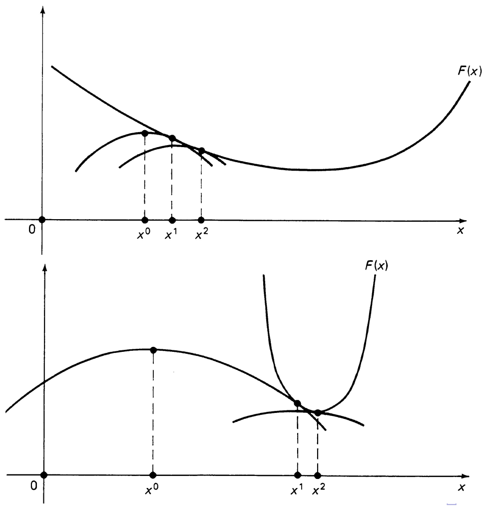
The proximal minimization algorithm
Geometric interpretation

- Effect of large and small values of c



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

$$\text{minimize}_{x\in X} \; F(x)$$
$$\text{subject to} : Ax = b$$

Augmented program

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

- 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 $X = X_1 \times \ldots \times X_m$, then $Ax = b$ models coupling among agents' decisions

The augmented Lagrangian algorithm

- Construct the Lagrangian of the augmented program

$$L_c(x,\lambda) = F(x) + \lambda^\top (Ax-b) + \frac{c}{2}\|Ax-b\|^2$$

Augmented Lagrangian algorithm :

1. $x(k+1) = \arg\min_{x\in X} F(x) + \lambda(k)^\top (Ax-b) + \frac{c}{2}\|Ax-b\|^2$
2. $\lambda(k+1) = \lambda(k) + c\big(Ax(k+1)-b\big)$

- For simplicity we assumed a unique minimum for the primal variables ; this depends on A
- Apply a primal-dual scheme : minimization for primal followed by gradient ascent for dual

The augmented Lagrangian algorithm

Augmented Lagrangian algorithm :

1. $x(k+1) = \arg\min_{x\in X} F(x) + \lambda(k)^\top (Ax-b) + \frac{c}{2}\|Ax-b\|^2$
2. $\lambda(k+1) = \lambda(k) + c\big(Ax(k+1)-b\big)$

Theorem : Convergence of Augmented Lagrangian algorithm

For any $c>0$, we have that :

1. there exists an optimal dual solution $\lambda^\star$ such that

$$\lim_{k\to\infty} \|\lambda(k) - \lambda^\star\| = 0$$

2. primal iterates converge to the optimal value $F^\star$, i.e.

$$\lim_{k\to\infty} \|F(x(k)) - F^\star\| = 0$$

Proof

Augmented Lagrangian algorithm :

1. $x(k+1) = \arg\min_{x\in X} F(x) + \lambda(k)^\top (Ax-b) + \frac{c}{2}\|Ax-b\|^2$
2. $\lambda(k+1) = \lambda(k) + c\big(Ax(k+1)-b\big)$

- Notice that the dual function of the original problem is given by

$$q(y) = \min_{x\in X} F(x) + y^\top (Ax-b)$$

where y contains the dual variables associated with $Ax \le b$

Step 1 : Equivalently write the primal minimization step as

$$\min_{x\in X} F(x) + \lambda(k)^\top (Ax-b) + \frac{c}{2}\|Ax-b\|^2$$
$$= \min_{x\in X,\ z,\ Ax-b=z} F(x) + \lambda(k)^\top z + \frac{c}{2}\|z\|^2$$

The minimizers are denoted by $x(k+1)$ and $z(k+1)$

Proof (cont'd)

Step 2 :

- Dualize the coupling constraint in Step 1 using multipliers y and consider the optimum of the dual problem

$$y^\star = \arg\max_y \left\{ \min_{x\in X} \big(F(x) + y^\top (Ax-b)\big) + \min_z \Big((\lambda(k)-y)^\top z + \frac{c}{2}\|z\|^2\Big) \right\}$$

- Using the definition of the $q(y)$ this is equivalent to

$$y^\star = \arg\max_y \left\{ q(y) + \min_z \Big((\lambda(k)-y)^\top z + \frac{c}{2}\|z\|^2\Big) \right\}$$

- The inner minimization is an unconstrained quadratic program ; calculate its minimizer by setting the objective's gradient equal to zero

$$\bar{z} = \frac{y-\lambda(k)}{c} \quad \text{and hence} \quad z(k+1) = \frac{y^\star - \lambda(k)}{c}$$

Proof (cont'd)

Step 3 :

- Substituting back the value of $\bar{z}$

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

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

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

which in turn implies that

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

Proof (cont'd)

Step 4 : Putting everything together ...

- The augmented Lagrangian primal dual scheme

1. $x(k+1) = \arg\min_{x\in X} F(x) + \lambda(k)^\top(Ax - b) + \frac{c}{2}\|Ax - b\|^2$
2. $\lambda(k+1) = \lambda(k) + c\big(Ax(k+1) - b\big)$

... is equivalent to

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

- 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. $\lim_{k\to\infty}\|\lambda(k) - \lambda^\star\| = 0$ for some optimal $\lambda^\star$
- For the primal variables we can only show something slightly weaker : they asymptotically achieve the optimal value $F^\star$

Back to decision coupled problems

Recall the equivalence between decision and constraint coupled problems

Decision coupled problem

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

Constraint coupled problem

$$\text{minimize } \sum_i f_i(x_i) \quad \text{subject to : } x_i \in X_i,\ \forall i; \quad x_i = z,\ \forall i$$

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

$$\text{minimize}_{x\in X}\ F(x) \quad \text{subject to : } Ax = b$$

Decision coupled problems

Consider the following asignements :

- 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 :

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

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

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

Decision coupled problems

Augmented Lagrangian for the reformulated constraint coupled problem

1. Primal update

$$\big(x_1(k+1),\dots,x_m(k+1),z(k+1)\big) = \arg\min_{x_1\in X_1,\dots,x_m\in X_m,z} \sum_i f_i(x_i) + \lambda_i^\top(k)(z-x_i) + \frac{c}{2}\|z-x_i\|^2$$

2. Dual update

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

- Primal update in the form cost coupled problems via a single function $\sum_i f_i(x_i) + \lambda_i(k)^\top(z-x_i) + \frac{c}{2}\|z-x_i\|^2$
- Can solve via Gauss-Seidel algorithm, alternating between minimizing with respect to $(x_1,\dots,x_m)$ and z

Decision coupled problems

Primal update : Can solve via Gauss-Seidel algorithm, alternating between minimizing with respect to $(x_1,\dots,x_m)$ and z

$$\big(x_1(k+1),\dots,x_m(k+1),x(k+1)\big) = \arg\min_{x_1\in X_1,\dots,x_m\in X_m,z} \sum_i f_i(x_i) + \lambda_i^\top(k)(z-x_i) + \frac{c}{2}\|z-x_i\|^2$$

- **Update of z :** Unconstrained quadratic minimization with respect to z. Take the derivative and set it equal to zero leads to

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

- **Update of $x_1,\dots,x_m$:** For fixed z problem is separable across agents (no longer coupled in the cost). Hence for all i,

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

Decision coupled problems

```
begin loop
```

1. Primal update for z **information from central authority**

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

2. Primal update for x_i **in parallel for all agents**

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

```
end loop
```

3. Dual update **in parallel for all agents**

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

- Nested iteration with Gauss-Seidel inner loop – Can we do any better ?

Decision coupled problems

What if we only do one Gauss-Seidel pass ?

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**

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

3. Dual update **in parallel for all agents**

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

- Does this scheme converge ? ADMM provides the answer ! Lecture 3

Summary

Decision coupled problems

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

Intriduced three different algorithms

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

Proximal $\Longrightarrow$ Augmented Lagrangian $\Longrightarrow$ ADMM

Thank you for your attention !
Questions ?

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

C20 Distributed Systems
Lecture 3

Kostas Margellos

University of Oxford

Recap

Decision coupled problems

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

Intriduced three different algorithms

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

Proximal $\Longrightarrow$ Augmented Lagrangian $\Longrightarrow$ ADMM

Recap : Augmented Lagrangian algorithm

Inner lopp : Gauss-Seidel algorithm !

begin loop

1. Primal update for z **information from central authority**

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

2. Primal update for x_i **in parallel for all agents**

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

end loop

3. Dual update **in parallel for all agents**

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

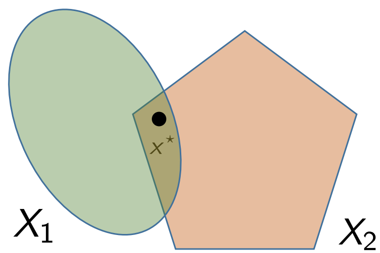
Example

Feasibility problem – part of Question 4, Example Paper

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

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

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



Example (cont'd)

- Decision coupled problem with 2 agents ; notice that $f_1(x) = f_2(x) = 0$
- 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

1. $z = \frac{x_1+x_2}{2} - \frac{\lambda_1(0)+\lambda_2(0)}{2c} = \frac{x_1+x_2}{2}$
2. $x_1 \leftarrow \arg\min_{x_1 \in X_1} -\lambda_1(0)x_1 + \frac{c}{2}\|z - x_1\|^2 = \arg\min_{x_1 \in X_1} \frac{c}{2}\|z - x_1\|^2$
   $x_2 \leftarrow \arg\min_{x_2 \in X_2} -\lambda_2(0)x_2 + \frac{c}{2}\|z - x_2\|^2 = \arg\min_{x_2 \in X_2} \frac{c}{2}\|z - x_2\|^2$

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

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

1. $z = \frac{x_1+x_2}{2}$
2. $x_1 \leftarrow \arg\min_{x_1 \in X_1} \frac{c}{2}\|z - x_1\|^2 = \Pi_{X_1}\big[z\big]$
   $x_2 \leftarrow \arg\min_{x_2 \in X_2} \frac{c}{2}\|z - x_2\|^2 = \Pi_{X_2}\big[z\big]$

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

$$\text{minimize}_{z, x_1 \in X_1, x_2 \in X_2} \ \frac{c}{2}\sum_{i=1,2}\|z - x_i\|^2$$

- Hence it converges to the minimum, which occurs when $x_1 = x_2 = z$

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

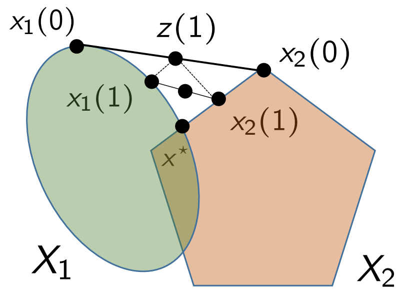
1. Averaging step : $z(k+1) = \frac{x_1(k)+x_2(k)}{2}$
2. Parallel projections :
   $x_1(k+1) = \Pi_{X_1}\big[z(k+1)\big]$ and $x_2(k+1) = \Pi_{X_2}\big[z(k+1)\big]$

Example (cont'd)

Simplified single-loop algorithm

1. Averaging step : $z(k+1) = \frac{x_1(k)+x_2(k)}{2}$
2. Parallel projections :
   $x_1(k+1) = \Pi_{X_1}\big[z(k+1)\big]$ and $x_2(k+1) = \Pi_{X_2}\big[z(k+1)\big]$

Schematic illustration of the single-loop iterations



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**

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

3. Dual update

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

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**

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

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

$$\text{minimize } F_1(x) + F_2(Ax)$$
$$\text{subject to : } x \in C_1, \; Ax \in C_2$$

Equivalent reformulation

$$\text{minimize } F_1(x) + F_2(z)$$
$$\text{subject to : } x \in C_1, \; z \in C_2$$
$$Ax = z$$

ADMM algorithm

Effectively Augmented Lagrangian with one Gauss-Seidel pass

1. $x(k+1) = \arg\min_{x \in C_1} F_1(x) + \lambda(k)^\top Ax + \frac{c}{2}\|Ax - z(k)\|^2$
2. $z(k+1) = \arg\min_{z \in C_2} F_2(z) - \lambda(k)^\top z + \frac{c}{2}\|Ax(k+1) - z\|^2$
3. $\lambda(k+1) = \lambda(k) + c\big(Ax(k+1) - z(k+1)\big)$

Theorem : Convergence of ADMM

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

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

Decision coupled problems as a special case again

Original problem

$$\text{minimize } \sum_i f_i(x)$$
$$\text{subject to : } x \in X_i, \; \forall i$$

ADMM set-up

$$\text{minimize } F_1(x) + F_2(z)$$
$$\text{subject to : } x \in C_1, \; z \in C_2$$
$$Ax = z$$

- Can be obtained as a special case of the ADMM set-up
- To see this, let $z = (z_1, \dots, z_m)$ and define $A = \begin{bmatrix} I \\ \vdots \\ I \end{bmatrix}$ (stack of identity matrices), hence $Ax = \begin{bmatrix} x \\ \vdots \\ x \end{bmatrix}$ and $Ax = z \;\Leftrightarrow\; \begin{bmatrix} x \\ \vdots \\ x \end{bmatrix} = \begin{bmatrix} z_1 \\ \vdots \\ z_m \end{bmatrix}$

Decision coupled problems (cont'd)

- Perform also the following assignments

$$F_1(x) = 0, \quad C_1 = \mathbb{R}^n$$
$$F_2(z) = \sum_i f_i(z_i), \quad C_2 = X_1 \times \dots \times X_m$$

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

1. $x(k+1) = \arg\min_{x \in \mathbb{R}^n} \lambda(k)^\top Ax + \frac{c}{2}\|Ax - z(k)\|^2$
2. $z(k+1) = \arg\min_{z_1 \in X_1, \dots, z_m \in X_m} \sum_i f_i(z_i) - \lambda(k)^\top z + \frac{c}{2}\|Ax(k+1) - z\|^2$
3. $\lambda(k+1) = \lambda(k) + c\big(Ax(k+1) - z(k+1)\big)$

Decision coupled problems (cont'd)

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

1. $x(k+1) = \arg\min_{x\in\mathbb{R}^n} \sum_i \lambda_i(k)^\top x + \frac{c}{2}\sum_i \|x - z_i(k)\|^2$
   - Unconstrained quadratic optimization
   - Setting the gradient with respect to x equal to zero we obtain

$$\sum_i \lambda_i(k) + c\sum_i (x(k+1) - z_i(k)) = 0$$

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

2. $z(k+1) = \arg\min_{z_1\in X_1,\dots,z_m\in X_m} \sum_i \left(f_i(z_i) - \lambda_i(k)^\top z_i + \frac{c}{2}\|x(k+1) - z_i\|^2 \right)$
   - Since $x(k+1)$ is fixed, fully separable across i. Minimizing the "sum" is equivalent to minimizing each individual component. Hence, for all i,

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

3. $\lambda_i(k+1) = \lambda_i(k) + c\big(x(k+1) - z_i(k+1)\big)$ (due to the structure of A)

Constraint coupled problems

Affine coupling :

$$\text{minimize } \sum_i f_i(x_i)$$
$$\text{subject to : } x_i \in X_i,\ \forall i$$
$$\sum_i x_i = 0$$

- 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

Constraint coupled problems

Original problem

$$\text{minimize } \sum_i f_i(x_i)$$
$$\text{subject to : } x_i \in X_i,\ \forall i$$
$$\sum_i x_i = 0$$

ADMM set-up

$$\text{minimize } F_1(x) + F_2(z)$$
$$\text{subject to : } x \in C_1,\ z \in C_2$$
$$Ax = z$$

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

$$F_1(x) = \sum_i f_i(x_i),\quad C_1 = X_1\times\dots\times X_m$$
$$F_2(z) = 0,\quad C_2 = \{z \mid \sum_i z_i = 0\}$$

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

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

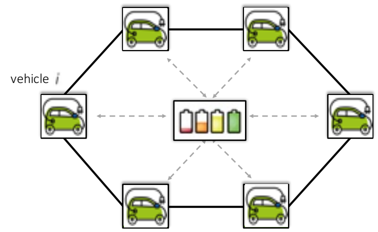
Decision coupled problems

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

subject to

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

Recall electric vehicle charging control problem



Decision coupled problem

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

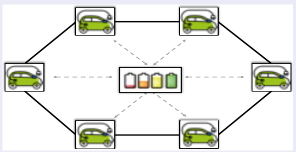
subject to

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

Distributed proximal minimization

General architecture

Step 1 : Local problem of agent i



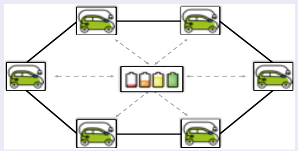
$$\left.\begin{array}{l}\text{minimize } f_i(x_i) + g_i(x_i, z_i) \\ \text{subject to} \\ x_i \in X_i\end{array}\right\} \Rightarrow x_i^*(z_i)$$

- 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

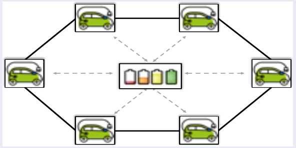


$$\left.\begin{array}{l}\text{minimize } f_i(x_i) + g_i(x_i, z_i) \\ \text{subject to} \\ x_i \in X_i\end{array}\right\} \Rightarrow x_i^*(z_i)$$

Distributed proximal minimization

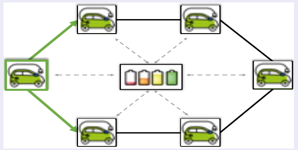
General architecture

Step 1 : Local problem of agent i

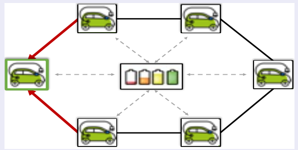


$$\left.\begin{array}{l}\text{minimize } f_i(x_i) + g_i(x_i, z_i)\\ \text{subject to}\\ x_i \in X_i\end{array}\right\} \Rightarrow x_i^*(z_i)$$

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



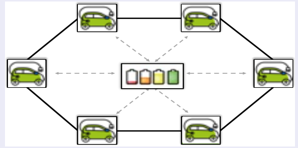
Step 2b : Receive neighbors' solutions



Distributed proximal minimization

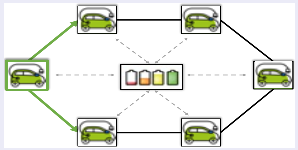
General architecture

Step 1 : Local problem of agent i

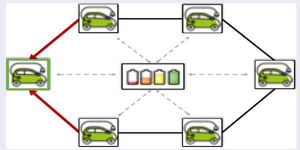


$$\left.\begin{array}{l}\text{minimize } f_i(x_i) + g_i(x_i, z_i)\\ \text{subject to}\\ x_i \in X_i\end{array}\right\} \Rightarrow x_i^*(z_i)$$

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



Step 2b : Receive neighbors' solutions

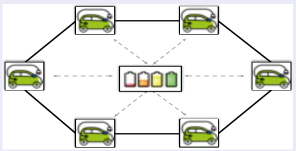


Step 3 : Update z_i on the basis of information received

Go to Step 1

Distributed proximal minimization

Local problem of agent i

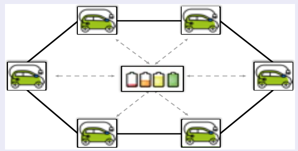


$$\left.\begin{array}{l}\text{minimize } f_i(x_i) + g_i(x_i, z_i)\\ \text{subject to}\\ x_i \in X_i\end{array}\right\} \Rightarrow x_i^*(z_i)$$

- We need to specify
  - Information vector z_i
  - Proxy 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

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**

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

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

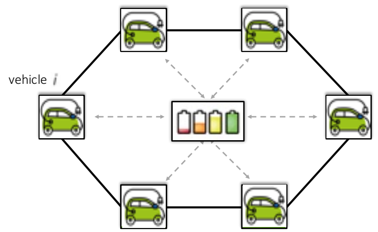
Recap : Distributed algorithms

Decision coupled problems

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

subject to

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



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 !

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 if we had access to f_i's and X_i'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. $\exists$ Ball$(\bar{x}, \rho) \subset \bigcap_i X_i$

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. $\exists$ Ball$(\bar{x}, \rho) \subset \bigcap_i X_i$
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 $\lim_{k\to\infty} c(k) = 0$, i.e. as iterations increase we penalize "disagreement" more

Algorithm analysis : Assumptions

3. Network connectivity – All information flows (eventually)

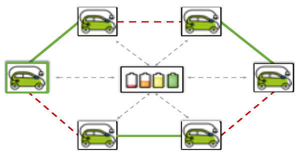
Connectivity

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

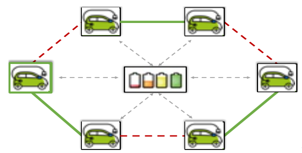
Connectivity

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

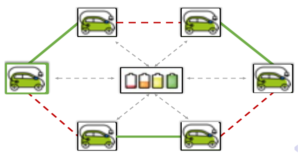
Connectivity

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

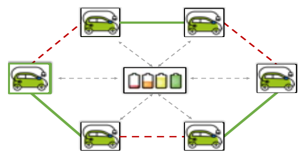
Connectivity

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

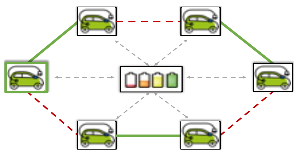
Connectivity

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 $x^\star$ of the centralized problem, i.e.,

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

- 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 :

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

1. What is the optimal solution ?
2. 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 $x_2(0) = 1$

- 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

$$\min_{x\in\mathbb{R}} \sum_{i=1,2} \alpha(x+s_i)^2$$
$$\text{subject to } \quad x \in [-M, M]$$

- 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 $x^\star = 0$

Example (cont'd)

Main distributed proximal minimization updates

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

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

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)

Main distributed proximal minimization updates

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

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

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

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

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

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 $z_i(k+1) = 0$

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

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 $z_i(k+1) = 0$

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

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

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

Example (cont'd)

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

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

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

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

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

- 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)

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

- 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

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)

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

- 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

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

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

- Notice that this is no a recursion but an identity satisfied by $x_i(k+1)$ 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. $z_i(k)$?

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

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

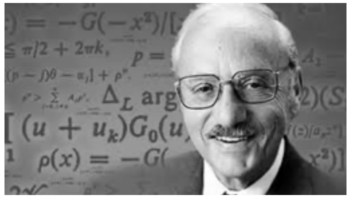
Summary

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

C20 Distributed Systems
Appendix

Kostas Margellos

University of Oxford

Condensed overview of main algorithms

Decentralized & Distributed algorithms

Part I : Decentralized algorithms

Cost coupled problems

Cost coupled problems

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

The Jacobi algorithm

Main update steps :

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

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

Convergence :

- F strongly convex and differentiable
- X_i's are all convex

The regularized Jacobi algorithm

Main update steps :

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

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

Convergence :

- F convex and differentiable and c big enough
- X_i's are all convex

The Gauss-Seidel algorithm

Main update steps **(sequential algorithm)** :

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

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

Convergence :

- F is strongly convex with respect to each individual argument, and differentiable
- X_i's are all convex

Part I : Decentralized algorithms

Decision coupled problems

Decision coupled problems

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

subject to

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

The Alternating Direction Method of Multipliers (ADMM)

Main update steps :

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**

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

3. Dual update **in parallel for all agents**

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

- Augmented Lagrangian with one Gauss-Seidel pass of the inner loop

ADMM algorithm (more general form)

Applicable to problems with two groups of variables :

$$\text{minimize } F_1(x) + F_2(z)$$

$$\text{subject to} : x \in C_1, \ z \in C_2$$

$$Ax = z$$

Main update steps :

1. $x(k+1) = \arg\min_{x \in C_1} F_1(x) + \lambda(k)^\top Ax + \frac{c}{2}\|Ax - z(k)\|^2$
2. $z(k+1) = \arg\min_{z \in C_2} F_2(z) - \lambda(k)^\top z + \frac{c}{2}\|Ax(k+1) - z\|^2$
3. $\lambda(k+1) = \lambda(k) + c\big(Ax(k+1) - z(k+1)\big)$

Convergence :

- All functions and sets are convex, and $A^\top A$ is invertible

Part II : Distributed algorithms
Decision coupled problems

Decision coupled problems

$$\text{minimize} \sum_i f_i(x)$$
subject to
$$x \in X_i, \ \forall i = 1, \ldots, m$$

Distributed proximal minimization

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

Convergence :

- Convexity of all functions and sets + Network connectivity (Lecture 4)
- Mixing weights sum up to one, forming a doubly stochastic matrix
- Step-size choice : $c(k) = \frac{\alpha}{k+1}$, $\alpha > 0$

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

Convergence :

- Same assumptions with distributed proximal minimization algorithm

Thank you for your attention !
Questions ?

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