DQM: Decentralized Quadratically Approximated Alternating Direction Method of Multipliers

Aryan Mokhtari[†], Wei Shi[⋆], Qing Ling[⋆], and Alejandro Ribeiro[†]

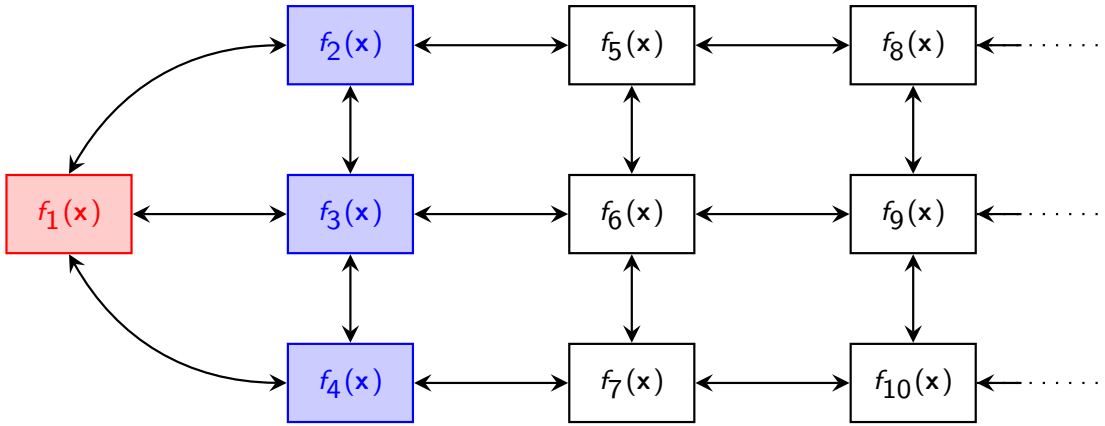
[†]University of Pennsylvania, [⋆]University of Science and Technology of China

3rd IEEE Global Conf. on Signal & Information Processing (GlobalSIP)

Orlando, FL, Dec 15, 2015

- Network with n nodes. Each node i has access to local function $f_i(\mathbf{x})$
- Collaborate to minimize global objective $\Rightarrow f(\mathbf{x}) = \sum_{i=1}^{n} f_i(\mathbf{x})$

  $\Rightarrow$ Sample subsets to train classifier

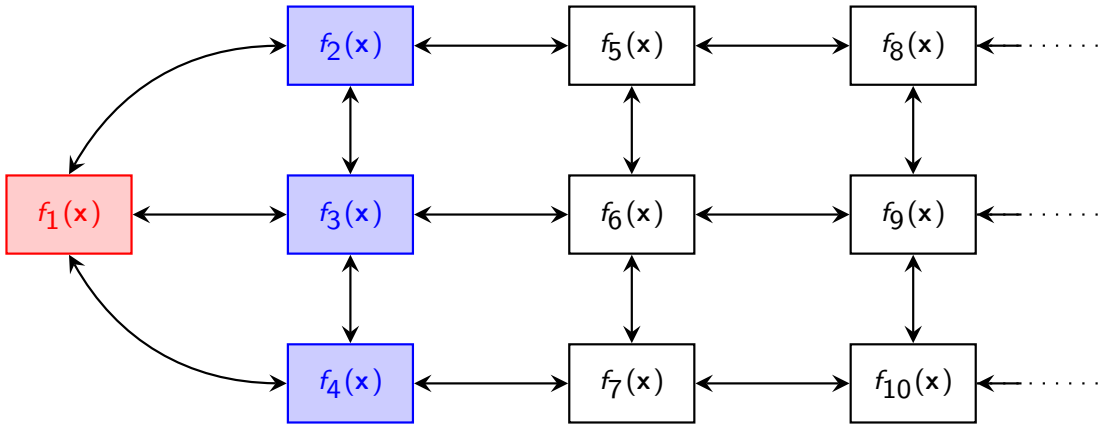


- Nodes can operate (train or estimate) locally but would benefit by sharing

  $\Rightarrow$ Cost of aggregating functions is large $\Rightarrow$ Comms and computation

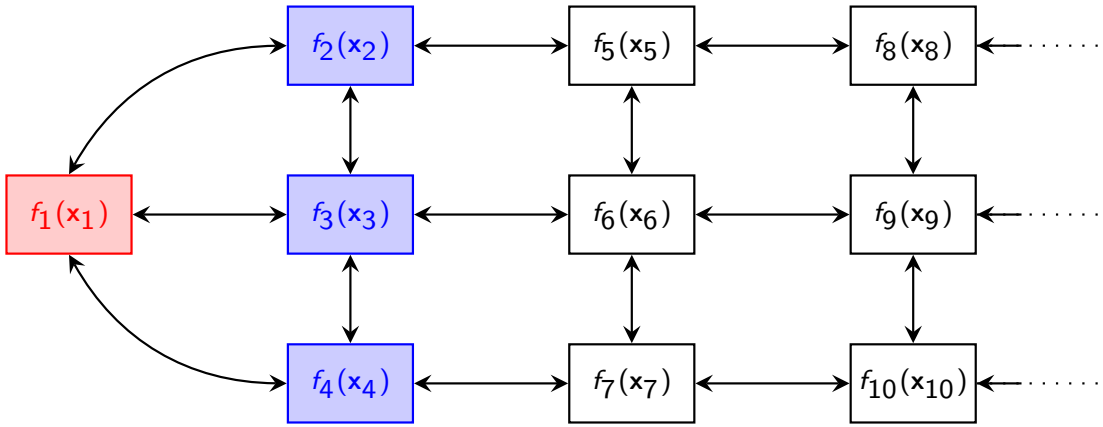
- Recursive exchanges with neighbors $j \in \mathcal{N}_i$ to aggregate global information

- Replicate common variable at each node $\Rightarrow f(\mathbf{x}_1, \ldots, \mathbf{x}_n) = \sum_{i=1}^{n} f_i(\mathbf{x}_i)$
- Enforce equality between neighbors $\mathbf{x}_i = \mathbf{x}_j$ (thus between all nodes)



- Operate recursively to enforce equality asymptotically. Differ on how.
  - $\Rightarrow$ Distributed gradient descent, recursive averaging, [Nedic, Ozdaglar '09]
  - $\Rightarrow$ Distributed dual descent, prices, [Rabbat et al '05]
  - $\Rightarrow$ Distributed ADMM, prices, [Schizas et al '08]
- DADMM is the best in terms of comm. cost $\Rightarrow$ Computationally costly

- Replicate common variable at each node $\Rightarrow f(\mathbf{x}_1, \ldots, \mathbf{x}_n) = \sum_{i=1}^{n} f_i(\mathbf{x}_i)$
- Enforce equality between neighbors $\mathbf{x}_i = \mathbf{x}_j$ (thus between all nodes)



- Operate recursively to enforce equality asymptotically. Differ on how.
  - $\Rightarrow$ Distributed gradient descent, recursive averaging, [Nedic, Ozdaglar '09]
  - $\Rightarrow$ Distributed dual descent, prices, [Rabbat et al '05]
  - $\Rightarrow$ Distributed ADMM, prices, [Schizas et al '08]
- DADMM is the best in terms of comm. cost $\Rightarrow$ Computationally costly

DQM: Approximate DADMM

- DADMM: a primal-dual method that minimizes an augmented Lagrangian
- The primal update of DADMM is computationally expensive
- DQM resolves this issue by approximating the DADMM primal update
- DQM uses a quadratic approximation of the global objective function
- This approximation leads to a quadratic program $\Rightarrow$ computationally cheap
  $\Rightarrow$ with minimal effect on convergence properties

- n nodes, m edges as $\mathcal{E} = \{(i,j)|$ i can comm. to j$\}$, $(i,j) \in \mathcal{E} \Leftrightarrow (j,i) \in \mathcal{E}$
- Problem in distributed form $\Rightarrow \min_{x_1,\dots,x_n} \sum_{i=1}^{n} f_i(\mathbf{x}_i)$, s.t. $\mathbf{x}_i = \mathbf{x}_j$, for $(i,j) \in \mathcal{E}$
- With the auxiliary variables $\mathbf{z}_{ij}$ associated with edge $(i,j) \in \mathcal{E}$

$$\{\mathbf{x}_i^*\}_{i=1}^n := \operatorname*{argmin}_{\mathbf{x}_1,\dots,\mathbf{x}_n} \sum_{i=1}^{n} f_i(\mathbf{x}_i),$$
$$\text{s.t.} \quad \mathbf{x}_i = \mathbf{z}_{ij}, \ \mathbf{x}_j = \mathbf{z}_{ij}, \ \text{for all } (i,j) \in \mathcal{E}.$$

- Try to separate the variables $\mathbf{x}_i$

- Rewrite the problem formulation in vector form
  - $\Rightarrow$ Aggregate variables $\mathbf{x} := [\mathbf{x}_1; \ldots; \mathbf{x}_n]$ and $\mathbf{z} := [\mathbf{z}_1; \ldots; \mathbf{z}_m]$
  - $\Rightarrow$ Aggregate function $f(\mathbf{x}) = f(\mathbf{x}_1, \ldots, \mathbf{x}_n) = \sum_{i=1}^{n} f_i(\mathbf{x}_i)$
- Therefore, the problem can be written as

$$\mathbf{x}^* := \underset{\mathbf{x}}{\operatorname{argmin}}\, f(\mathbf{x}), \quad \text{s. t. } \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} = \mathbf{0}.$$

- $\mathbf{A} = [\mathbf{A}_s; \mathbf{A}_d] \in \mathbb{R}^{2mp \times np}$ stacks the source $\mathbf{A}_s$ and destination $\mathbf{A}_d$ matrices
- $\mathbf{B} = [-\mathbf{I}; -\mathbf{I}] \in \mathbb{R}^{2mp \times mp}$ stacks two negative identity matrices of size mp
- ADMM can be used to solve the optimization problem

- The augmented Lagrangian of $\min_{\mathbf{x}} f(\mathbf{x}), \quad \text{s.t. } \mathbf{Ax}+\mathbf{Bz}=\mathbf{0}$ is

$$\mathcal{L}(\mathbf{x},\mathbf{z},\boldsymbol{\lambda}) := f(\mathbf{x}) + \boldsymbol{\lambda}^T(\mathbf{Ax}+\mathbf{Bz}) + \frac{c}{2}\|\mathbf{Ax}+\mathbf{Bz}\|^2.$$

- $\boldsymbol{\lambda}$ is the dual variable and $c>0$.

- The DADMM update at step k:

  $\Rightarrow$ Step 1: $\mathbf{x}_{k+1} = \operatorname{argmin}_{\mathbf{x}} f(\mathbf{x}) + \boldsymbol{\lambda}_k^T(\mathbf{Ax}+\mathbf{Bz}_k) + \frac{c}{2}\|\mathbf{Ax}+\mathbf{Bz}_k\|^2$

  $\Rightarrow$ Step 2: $\mathbf{z}_{k+1} = \operatorname{argmin}_{\mathbf{z}} f(\mathbf{x}_{k+1}) + \boldsymbol{\lambda}_k^T(\mathbf{Ax}_{k+1}+\mathbf{Bz}) + \frac{c}{2}\|\mathbf{Ax}_{k+1}+\mathbf{Bz}\|^2$

  $\Rightarrow$ Step 3: $\boldsymbol{\lambda}_{k+1} = \boldsymbol{\lambda}_k + c(\mathbf{Ax}_{k+1}+\mathbf{Bz}_{k+1})$

- Steps 2 and 3 are not costly in terms of computation time
- Step 1 can be computationally expensive $\Rightarrow$ no closed form solution

- DQM reduces computational complexity of the primal update

  $\Rightarrow$ Using a quadratic approximation of $f(\mathbf{x})$ around $\mathbf{x}_k$

- The DQM update at step k:

  $\Rightarrow$ Step 1: $\mathbf{x}_{k+1} = \underset{\mathbf{x}}{\operatorname{argmin}}\, f(\mathbf{x}_k) + \nabla f(\mathbf{x}_k)^T(\mathbf{x}-\mathbf{x}_k) + \frac{1}{2}(\mathbf{x}-\mathbf{x}_k)^T\mathbf{H}_k(\mathbf{x}-\mathbf{x}_k) + \boldsymbol{\lambda}_k^T(\mathbf{A}\mathbf{x}+\mathbf{B}\mathbf{z}_k) + \frac{c}{2}\|\mathbf{A}\mathbf{x}+\mathbf{B}\mathbf{z}_k\|^2$

  $\Rightarrow$ Step 2: $\mathbf{z}_{k+1} = \underset{\mathbf{z}}{\operatorname{argmin}}\, f(\mathbf{x}_{k+1}) + \boldsymbol{\lambda}_k^T(\mathbf{A}\mathbf{x}_{k+1}+\mathbf{B}\mathbf{z}) + \frac{c}{2}\|\mathbf{A}\mathbf{x}_{k+1}+\mathbf{B}\mathbf{z}\|^2$

  $\Rightarrow$ Step 3: $\boldsymbol{\lambda}_{k+1} = \boldsymbol{\lambda}_k + c(\mathbf{A}\mathbf{x}_{k+1}+\mathbf{B}\mathbf{z}_{k+1})$

- Step 1 is a quadratic program $\Rightarrow$ has a closed form solution
- This recursion is useful for analyzing convergence of DQM

Proposition

Consider $\mathbf{L}_u$ as the unoriented Laplacian, $\mathbf{L}_o$ as the oriented Laplacian, and $\mathbf{D} := (\mathbf{L}_u + \mathbf{L}_o)/2$ as the degree matrix. By choosing proper initial variables, the DQM iterates $\mathbf{x}_k$ can be generated as

$$\mathbf{x}_{k+1} = (2c\mathbf{D} + \mathbf{H}_k)^{-1} \left[(c\mathbf{L}_u + \mathbf{H}_k)\mathbf{x}_k - \nabla f(\mathbf{x}_k) - \boldsymbol{\phi}_k\right],$$
$$\boldsymbol{\phi}_{k+1} = \boldsymbol{\phi}_k + c\mathbf{L}_o\mathbf{x}_{k+1}.$$

- These recursions are simpler
  - $\Rightarrow$ Eliminate the auxiliary variables $\mathbf{z}_k$
  - $\Rightarrow$ Reduce the dimensionality of $\boldsymbol{\lambda}_k \in \mathbb{R}^{2mp}$ to that of $\boldsymbol{\phi}_k \in \mathbb{R}^{np}$
- How to implement these updates in a decentralized fashion?

- DQM global update can be implemented in a decentralized fashion
- Consider $\mathbf{x}_{i,k}$ and $\phi_{i,k}$ as the iterates of node i at step k

- DQM local primal update at node i

$$\mathbf{x}_{i,k+1} = \left(2cd_i\mathbf{I} + \nabla^2 f_i(\mathbf{x}_{i,k})\right)^{-1}\left[cd_i\mathbf{x}_{i,k} + c\sum_{j\in\mathcal{N}_i}\mathbf{x}_{j,k} + \nabla^2 f_i(\mathbf{x}_{i,k})\mathbf{x}_{i,k} - \nabla f_i(\mathbf{x}_{i,k}) - \phi_{i,k}\right]$$

- DQM local dual update at node i

$$\phi_{i,k+1} = \phi_{i,k} + c\sum_{j\in\mathcal{N}_i}\left(\mathbf{x}_{i,k+1} - \mathbf{x}_{j,k+1}\right).$$

(0) Initialize the local iterates $\mathbf{x}_{i,0}$ and $\phi_{i,0}$. Repeat for times $k = 0, 1, \ldots$

(1) Update the primal local iterate $\mathbf{x}_{i,k}$ as

$$\mathbf{x}_{i,k+1} = \left[2cd_i\mathbf{I} + \nabla^2 f_i(\mathbf{x}_{i,k})\right]^{-1}\left[cd_i\mathbf{x}_{i,k} + c\sum_{j\in\mathcal{N}_i}\mathbf{x}_{j,k} + \nabla^2 f_i(\mathbf{x}_{i,k})\mathbf{x}_{i,k} - \nabla f_i(\mathbf{x}_{i,k}) - \phi_{i,k}\right]$$

(2) Exchange iterates $\mathbf{x}_{i,k+1}$ with neighbors $j \in \mathcal{N}_i$.

(3) Update the local dual variable $\phi_{i,k}$ as

$$\phi_{i,k+1} = \phi_{i,k} + c\sum_{j\in\mathcal{N}_i}\left(\mathbf{x}_{i,k+1} - \mathbf{x}_{j,k+1}\right).$$

Assumption 1

- The local objective functions $f_i(\mathbf{x})$ are twice differentiable
- The Hessians $\nabla^2 f_i(\mathbf{x})$ have bounded eigenvalues $m\mathbf{I} \preceq \nabla^2 f_i(\mathbf{x}) \preceq M\mathbf{I}$

Assumption 2

- The local Hessians are Lipschitz continuous
  $\Rightarrow \|\nabla^2 f_i(\mathbf{x}) - \nabla^2 f_i(\hat{\mathbf{x}})\| \leq L\|\mathbf{x} - \hat{\mathbf{x}}\|$

Assumption 3

- The graph is connected and non-bipartite

- Consider the vector $\mathbf{u} := \begin{bmatrix} \mathbf{z} \\ \boldsymbol{\lambda} \end{bmatrix}$, and PD matrix $\mathbf{C} := \begin{bmatrix} c\mathbf{I}_{mp} & \mathbf{0} \\ \mathbf{0} & (1/c)\mathbf{I}_{mp} \end{bmatrix}$

Theorem

Consider the DQM method and let all the mentioned assumptions hold. Then, DQM converges linearly

$$\|\mathbf{u}_{k+1} - \mathbf{u}^*\|_{\mathbf{C}}^2 \leq \frac{1}{1+\delta_k}\|\mathbf{u}_k - \mathbf{u}^*\|_{\mathbf{C}}^2$$

where the sequence of positive scalars δ_k is increasing and given by

$$\delta_k = \min\left\{ \frac{(\mu-1)(c\gamma_u^2 - \eta_k\zeta_k)\gamma_o^2}{\mu\mu'(c\Gamma_u^2\gamma_u^2 + 4\zeta_k^2/c(\mu'-1))}, \frac{m - \zeta_k/\eta_k}{c\Gamma_u^2/4 + \mu M^2/c\gamma_o^2} \right\}$$

Theorem [Shi et al '14]

The sequence of iterates generated by DADMM converges linearly as

$$\|\mathbf{u}_{k+1} - \mathbf{u}^*\|_{\mathbf{C}}^2 \leq \frac{1}{1+\delta}\|\mathbf{u}_k - \mathbf{u}^*\|_{\mathbf{C}}^2$$

where $\delta = \min\left\{\frac{(\mu-1)\gamma_o^2}{\mu\Gamma_u^2}, \frac{m}{c\Gamma_u^2/4 + \mu M^2/c\gamma_o^2}\right\}$ is a positive constant.

Proposition

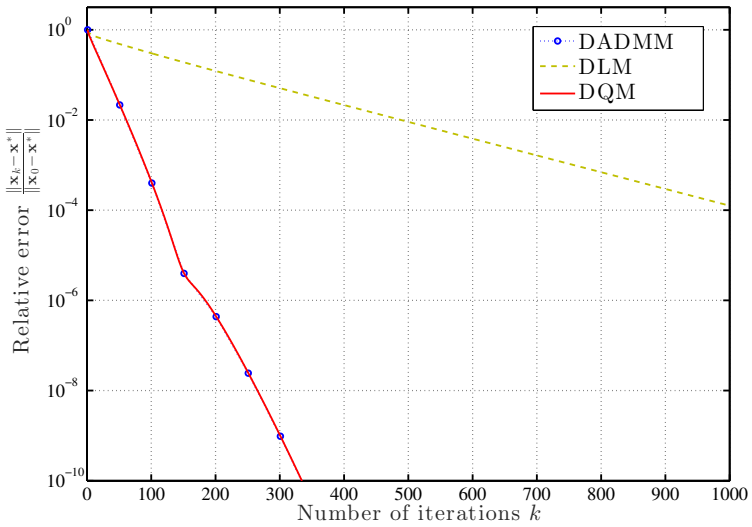
As time passes, the linear convergence factor δ_k of DQM approaches the DADMM linear convergence factor δ.

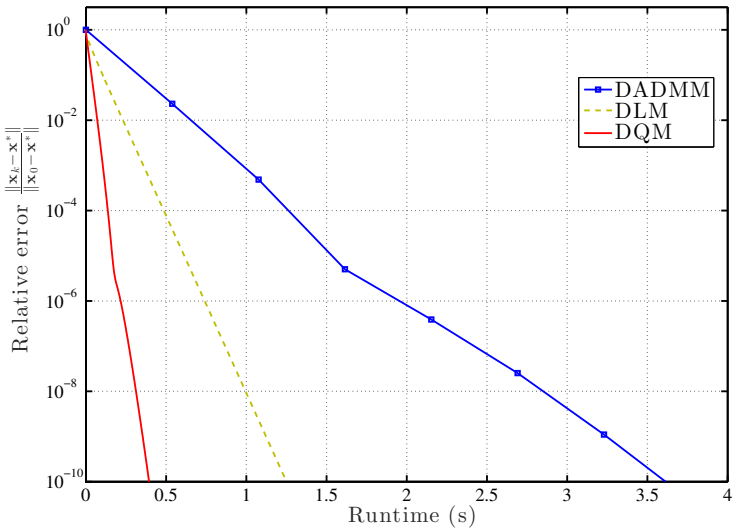
$$\lim_{k\to\infty} \delta_k = \delta$$

- The asymptotic linear rate of DQM is equal to the linear rate of DADMM

- We consider a logistic regression problem $\Rightarrow$ each node has q samples
- The graph is connected and random with connectivity ratio $r_c = 0.4$
- DLM $\Rightarrow$ a first order approx. of DADMM + proximal term

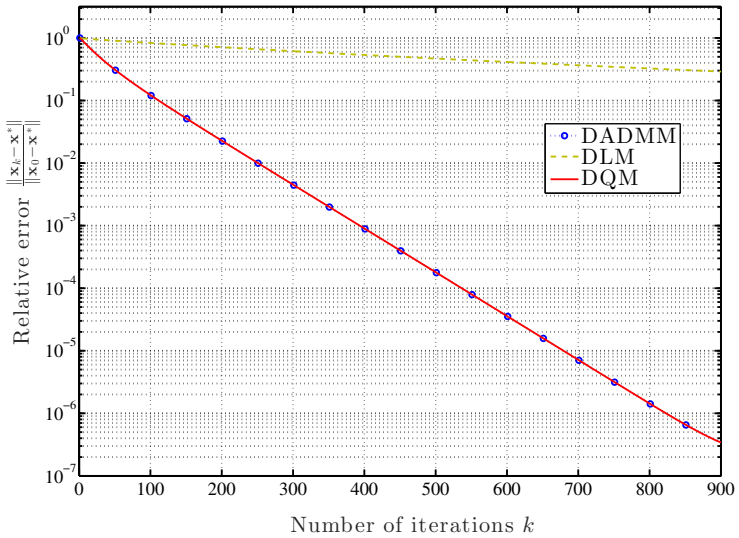
- First case $\Rightarrow$ $n = 10$ nodes and $q = 5$ samples per node

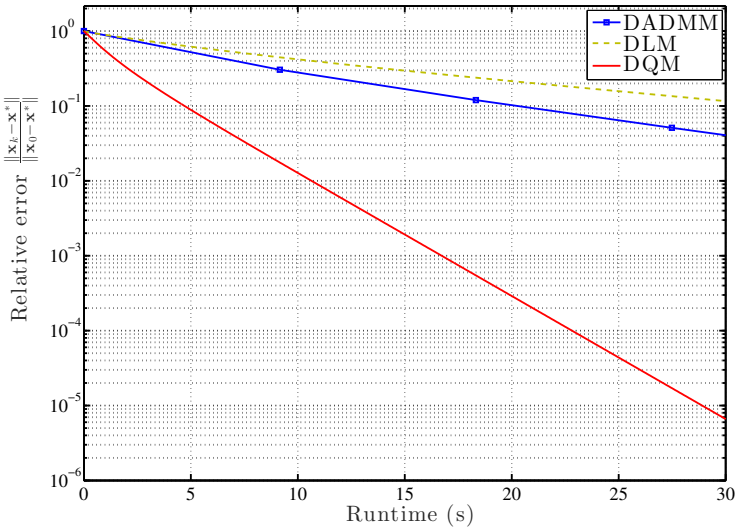




- DQM converges as fast as DADMM in terms of number of iterations
- In terms of convergence time DQM $\succeq$ DADMM $\succeq$ DLM

- Second case $\Rightarrow$ $n = 100$ nodes and $q = 20$ samples per node





- DLM (first-order method) is impractical in terms of communication cost
- DQM convergence rate is identical to the convergence rate of DADMM
- DQM outperforms both DLM and DADMM in terms of convergence time

- Introduced a network optimization formulation
  - $\Rightarrow$ Each agent has local cost function f_i
  - $\Rightarrow$ Global cost $f = \sum_{i=1}^{n} f_i$
- DQM is proposed as a second-order distributed method
  - $\Rightarrow$ Approximates the global cost by its second-order approximation
- Linear convergence is established
- Linear convergence factor of DQM approaches the one for DADMM
- DQM computational cost is significantly lower than DADMM
- Numerical experiments verify the theoretical results
  - $\Rightarrow$ DQM converges as fast as DADMM in terms of communication cost
  - $\Rightarrow$ DQM runtime to achieve a target accuracy is less than DADMM