An Accelerated Decentralized Stochastic Proximal Algorithm for Finite Sums

H. Hendrikx, F. Bach and L. Massoulié — INRIA - Ecole Normale Supérieure - PSL Research University - MSR-INRIA Joint Centre

Decentralized Optimization

Setting: n nodes jointly optimize the sum of their local functions:

$$\theta^* = \arg\min_{\theta\in\mathbb{R}^d} \left\{ F(\theta) \hat{=} \sum_{i=1}^{n} \left(\sum_{j=1}^{m} f_{i,j}(\theta) + \frac{\sigma_i^2}{2}\|\theta\|^2 \right) \right\}. \quad (1)$$

Each function $f_{i,j}$ is convex and $(L_{i,j})$-smooth. Nodes are connected by a network abstracted as a graph of spectral gap γ. Computing the gradient of one $f_{i,j}$ takes time 1 and communicating with a neighbor takes time τ. κ is the condition number of F.

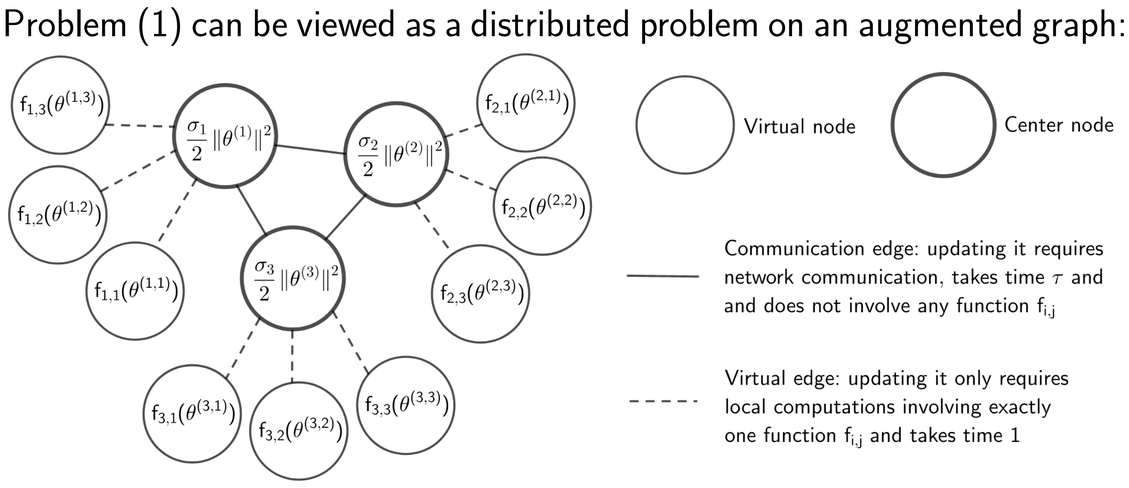
How efficiently can this problem be solved?

- Single machine stochastic algorithm: $O\left(nm + \sqrt{nm\kappa}\right)$
- Decentralized batch algorithm: $O\left(\sqrt{\kappa}\left(m + \tau\gamma^{-1/2}\right)\right)$
- Existing decentralized stochastic algorithm: $O\left((m+\kappa+\gamma)(1+\tau)\right)$

Is it possible to find a fast stochastic decentralized algorithm?

An augmented graph approach

Problem (1) can be viewed as a distributed problem on an augmented graph:



The dual formulation with edge equality constraints can be written as:

$$\min_{\lambda\in\mathbb{R}^{(nm+E)\times d}} \quad \underbrace{\frac{1}{2}\lambda^T A^T \Sigma^{-1} A\lambda}_{\text{SC + smooth}} \quad + \quad \underbrace{\sum_{i=1}^{n}\sum_{j=1}^{m} \tilde{f}_{i,j}^*(A\lambda^{(i,j)})}_{\text{convex + non-smooth}}$$

where A is such that $Ae_{ij} = \mu_{ij}(e_i - e_j)$, $\tilde{f}_{i,j}^*(x) = f_{i,j}^*(x) - \frac{1}{2L_{i,j}}\|x\|^2$ and Σ is diagonal such that $\Sigma_{i,i} = \sigma_i$ and $\Sigma_{ij,ij} = L_{i,j}$. There are $E + nm$ dual variables, i.e. one for each edge of the augmented graph.

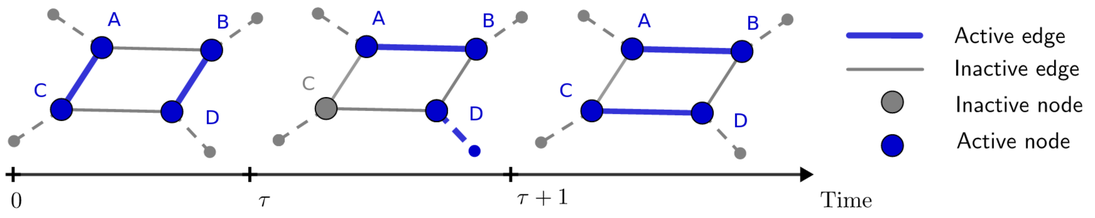
We develop an arbitrary sampling extension of **Accelerated Proximal Coordinate Gradient** (Lin, Lu, Xiao, 2015) and apply it to this problem:

– Virtual edge (i,j): local stochastic update of node i using $f_{i,j}$

– Communication edge (k,ℓ): Nodes k and ℓ exchange parameters

Local synchrony

Nodes execute the updates (A,C), (B,D), (A,E), (D), (A,B), (C,D)



Theorem: The number of updates performed per unit of time scales linearly with the size of the graph

Main References

- K. Scaman, F. Bach, S. Bubeck, Y.T. Lee and L. Massoulié. Optimal Algorithms for Smooth and Strongly Convex Distributed Optimization in Networks, *International Conference on Machine Learning, 3027–3036, 2017.*
- Q. Lin, Z. Lu, and L. Xiao. An accelerated randomized proximal coordinate gradient method and its application to regularized empirical risk minimization. SIAM Journal on Optimization, 25(4), 2244-2273 (2015).
- H. Hendrikx, F. Bach and L. Massoulié. Accelerated decentralized optimization with local updates for smooth and strongly convex objectives, AISTATS (2019)

ADFS: The algorithm

ADFS$(A, (\sigma_i), (L_{i,j}), (\mu_{k\ell}), (p_{k\ell}), \rho)$

$\sigma_A = \lambda_{\min}^+(A^T\Sigma^{-1}A)$, $\tilde{\eta}_{k\ell} = \frac{\rho\mu_{k\ell}^2}{\sigma_A p_{k\ell}}$, $R_{k\ell} = e_{k\ell}^T A^\dagger A e_{k\ell}$ // *Initialization*

$x_0 = y_0 = v_0 = z_0 = 0^{(n+nm)\times d}$

for $t = 0$ to $K-1$ **do** // *Run for K iterations*

  $y_t = \frac{1}{1+\rho}(x_t + \rho v_t)$

  Sample edge (k,ℓ) from the augmented graph with probability $p_{k\ell}$

  $z_{t+1} = v_{t+1} = (1-\rho)v_t + \rho y_t - \tilde{\eta}_{k\ell} W_{k\ell}\Sigma^{-1} y_t$

  if (k,ℓ) is the virtual edge between node i and virtual node (i,j) **then**

    $v_{t+1}^{(i,j)} = \text{prox}_{\tilde{\eta}_{ij}\tilde{f}_{i,j}^*}\left(z_{t+1}^{(i,j)}\right)$ // *Virtual node update using $f_{i,j}$*

    $v_{t+1}^{(i)} = z_{t+1}^{(i)} + z_{t+1}^{(i,j)} - v_{t+1}^{(i,j)}$ // *Center node update*

  end if

  $x_{t+1} = y_t + \frac{\rho R_{k\ell}}{p_{k\ell}}(v_{t+1} - (1-\rho)v_t - \rho y_t)$

end for

return $\theta_K = \Sigma^{-1} v_K$ // *Return primal parameter*

where $\rho^2 \leq \min_{k\ell} \frac{\lambda_{\min}^+(A^T\Sigma^{-1}A)}{\Sigma_{kk}^{-1} + \Sigma_{\ell\ell}^{-1}} \frac{p_{k\ell}^2}{\mu_{k\ell}^2 R_{k\ell}}$.

It is then possible to adjust p_{comp}, the probability to sample a virtual edge, in order to get the best rates:

Theorem: Executing ADFS to reach error ε takes time:

$$T_\varepsilon = O\left(m + \sqrt{m\kappa} + (1+\tau)\sqrt{\frac{\kappa}{\gamma}}\right)\log(\varepsilon^{-1})$$

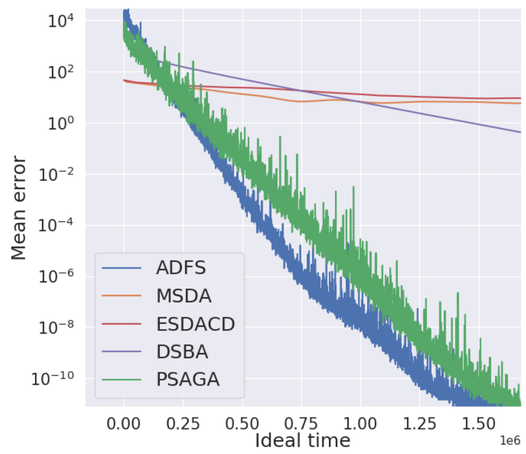
On n machines, ADFS can process n times more samples than optimal stochastic algorithms run on 1 machine as long as n is not too high.

Linear case: If $f_{i,j}(\theta) = g(X_{i,j}^T\theta)$ then proximal updates are cheap (1D subproblems), and virtual node variables can be stored as a simple scalar coefficient.

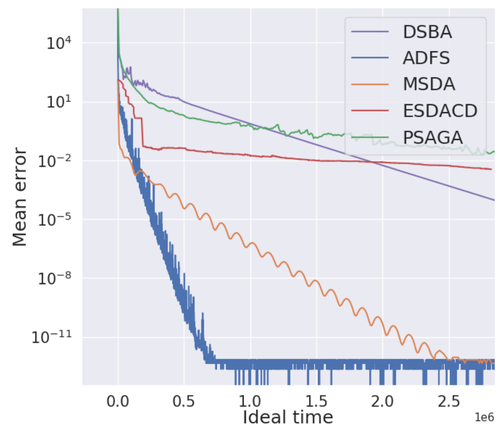
Experiments (Logistic Regression)

Experiments are run with $m = 10^4$ points per node and $\sigma_i = 1$. Idealized times are reported to avoid accounting for hardware or implementation.

Comparison with Decentralized algorithms



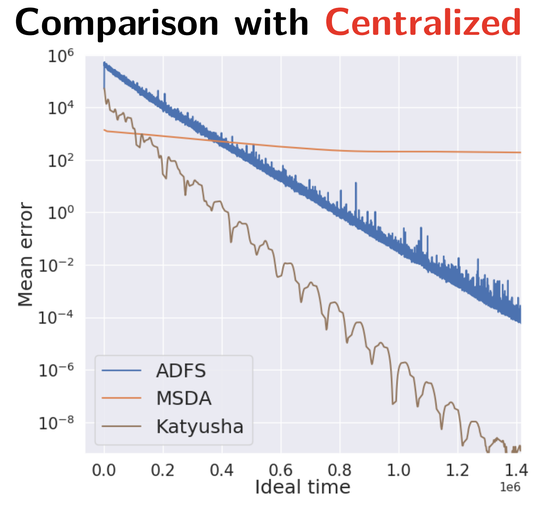
2×2 Grid, Higgs



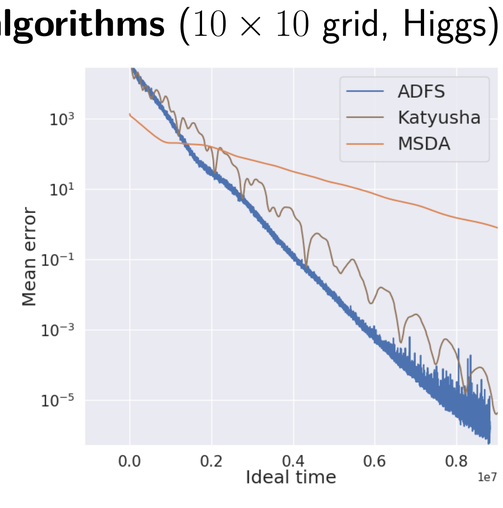
10×10 Grid, Covtype

ADFS efficiently distributes optimal single-machine approaches, thus outperforming all decentralized algorithms on large networks

Comparison with Centralized algorithms (10×10 grid, Higgs)



$\tau = 5$



$\tau = 50$

ADFS is competitive with centralized algorithms, and can outperform them when communication delays are high.