Name : Subhankar Chakraborty

Roll Number : EE17B031

Title: Experiments on Primal-dual distributed algorithm for constrained convex optimization

Reference : Primal-dual stochastic distributed algorithm for constrained convex optimization [1]

Contributions : Implemented a constrained Primal-dual distributed algorithm from scratch in Python and ran numerical experiments on it. Used Metropolis weights with time varying graphs for convergence.

Department of Electrical Engineering

End Term Paper Submission for EE5121

Instructor : Dr. Rachel Kalpana Kalaimani

September 2, 2020

I. Abstract

This report investigates distributed convex optimization problems over an undirected and **connected** network, where each node's variable lies in a private constrained convex set, and overall nodes aim at collectively minimizing the sum of all local objective functions. Each local objective function and constrained set cannot be shared with others. A primal-dual distributed algorithm is presented to address the distributed convex optimization problems, where each node updates its state by resorting to projected gradient descent on its constrained set. A few numerical examples and tests have been presented to check the convergence and stability of the algorithm. An implementation in Python is presented here.

II. Introduction

A. Distributed Convex Optimization

Distributed convex optimization problems recently have received much attention and been extensively investigated in control system [2–4], sensor network [5–7], machine learning[8–10], source localization [11, 12], and wireless system [13–15], to name a few. In these scenarios, there is no node or agent that acts as the fusion center in a network, and the tasks including computation and data transmission are distributed over all the nodes. In such a distributed manner, the system can avoid the risk of network congestion and computation overload. Generally, in distributed convex optimization the autonomous networked nodes cooperatively minimize a global objective function, given by the sum of each node's private local objective functions, via the communication with their neighbors only.

In this report, we focus on the case that each node's variable is constrained to a closed convex set and each constrained set cannot be shared with other nodes. In addition, the average of a number of local instantaneous functions is set as the local objective function. Some of the experiments, however, take a single function at a node instead of an average as from an implementation or convergence perspective, this makes little to no difference. From each node, the local objective function being an average of many functions or being a single function is the same thing.

B. Metropolis Weights

Given a network of processes where each node has an initial scalar value, the problem is of computing their average asymptotically using a distributed, linear iterative algorithm. At each iteration, each node replaces its own value with a weighted average of its previous value and the values of its neighbors. Xiao et al. [16, 6] introduce the Metropolis weights, a simple choice for the averaging weights used in each step. They show that with these weights, the values at every node converge to the average, provided the infinitely occurring communication graphs are jointly connected[16].

III. Notation Used

We write 1_m as an m-dimentional vector with all entries being one. We view notation $\bigotimes$ as the Kronecker product [17]. Denoted by $\|x\|$ is the Euclidean norm of vector x. Given a vector v and a symmetric positive semi-definite matrix A, we represent the A-weighted norm as $\|v\|_A = \sqrt{v^T Av}$. We use $P_{X_i}[v]$ as the projection of vector v on a convex set X_i. Other notations used for the proofs in [1] can be seen from there.

IV. Problem Formulation

Consider a network consisting of m nodes over an undirected and connected graph $\mathbb{G} = \{\mathbb{V}, \mathcal{E}, \mathbb{A}\}$ where $\mathbb{V} = \{1, ..., m\}$ is a collection of nodes, $\mathcal{E} \subset \mathbb{V} \times \mathbb{V}$ is a collection of edges and $\mathbb{A} = [a_{ii}] \in \mathbb{R}^{m \times m}$ represents the adjacency matrix satisfying $a_{ij} = a_{ji} > 0$ if $(i, j) \in \mathcal{E}$, and $a_{ij} = 0$ if $(i, j) \notin \mathcal{E}$. We let $\mathcal{N}_i = \{j | (i, j) \in \mathcal{E}\}$ to indicate the neighborhood set of node i. Denote a diagonal matrix $\mathcal{D} = \text{diag}\{\sum_{j=1}^{j=m} a_{1j}, ..., \sum_{j=1}^{j=m} a_{mj}\}$. Besides, the Laplacian matrix $\mathbb{L}$ of $\mathbb{G}$ is defined as $\mathbb{L} = \mathcal{D} - \mathbb{A}$ which is a symmetric and semi-definite matrix satisfying $\mathbb{L}1_m = 0$.

Specially, we focus on an undirected and **connected** network, where m nodes cooperatively address the constrained convex optimization problem as:

$$\begin{aligned} \underset{x \in \mathbb{R}^n}{\text{minimize}} \quad & f(x) = \sum_{i=i}^{m} f_i(x) = \sum_{i=1}^{m} \frac{1}{q_i} \sum_{j=1}^{q_i} f_i^j(x) \\ \text{subject to} \quad & x \in X = \cap_{i=1}^{m} X_i \end{aligned} \tag{1}$$

where x is a global variable, in each local node objective function $f_i : \mathbb{R}^n \to \mathbb{R}, i \in \mathbb{V}$, is defined as the average of q_i local instantaneous functions $f_i^j : \mathbb{R}^n \to \mathbb{R}, j = 1, .., q_i$, and $X_i \in \mathbb{R}^n$ is a closed convex set of node i. Each f_i and X_i are only possessed by node i and cannot be shared by other nodes. Denote x^* as the global optimization solution to problem (1). In problem (1), m nodes try to minimize the global objective function $f(x)$ in the constrained set X that is an intersection of m private constrained sets X_i while each node can communicate with its neighbors only. Problem (1) can be applied to scenarios with machine learning problems where to obtain the optimal classifier x^*, a large training set with a total of $\sum_{i=1}^{m} q_i$ training samples is distributed in m nodes for parallel processing conduct [18].

A. Problem reformulation

Lemma 1: *Problem (1) can be rewritten in the following form [19].*

$$\begin{aligned} \underset{z \in \mathbb{R}^{mn}}{\text{minimize}} \quad & f(z) = \sum_{i=i}^{m} f_i(x_i) = \sum_{i=1}^{m} \frac{1}{q_i} \sum_{j=1}^{q_i} f_i^j(x_i) \\ \text{subject to} \quad & Lz = 0, z \in X \end{aligned} \tag{2}$$

where $z = [x_i^T, ...x_m^T]^T, L = \mathbb{L} \bigotimes I_n$ and $X = \prod_{i=1}^m X_i$ is the cartesian product. If $z^* = [x_1^*, ..., x_m^*]$ is a global optimal solution to problem (2), we have $x_i^* = x_j^* = x^*, \forall (i,j) \in \mathbb{V}$ by $Lz^* = 0$. We denote assumptions which are necessary for a later analysis. All the proofs can be found in [1] which have been skipped from this report for the sake of brevity.

Assumption 1: Each local instantaneous function f_i^j is strongly convex with parameter μ, i.e., $\forall i \in \mathbb{V}$ and $j \in \{1, ..q_i\}$, we have

$$(\nabla f_i^j(a) - \nabla f_i^j(b))^T (a-b) \geq \mu \|a-b\|^2, \forall a, b \in \mathbb{R}^n \tag{3}$$

We know that the sum of strongly convex functions with parameter μ is strongly convex with parameter μ. Hence, from Assumption 1 it follows that $f_i, i \in \mathbb{V}$ and f are also strongly convex with parameters μ.

Assumption 2: The gradient of each local instantaneous function ∇f_i^j is Lipschitz-continous with parameter L_f, i.e., $\forall i \in \mathbb{V}$ and $j = \{1, ..., q_i\}$, we have

$$\|\nabla f_i^j(a) - \nabla f_i^j(b)\| \leq L_f \|a-b\|, \forall a, b \in \mathbb{R}^n \tag{4}$$

Likewise, from Assumption 2 it follows that the gradients of $f_i, i \in \mathbb{V}$ and f are also Lipschitz-continous with parameter L_f.

Define the Lagrangian function as:

$$L_a(x, \lambda) = f(x) + \lambda^T L x \tag{5}$$

where $\lambda \in \mathbb{R}^{mn}$ is the Lagrangian multiplier vector. Please take care of the fact that there is some loose notation in use here. x used in equation (5) is not the same as the one used in problem (1). The former is an mn dimension vector whereas the latter is an n dimensional vector. x used in equation (5) is used in the same meaning as z in problem (2). In this section, we use x in the same meaning as z in problem (2) unless otherwise mentioned.
Then the primal problem of problem (2) can be written as:

$$\min_{x \in \mathbb{X}} \quad \max_{\lambda \in \mathbb{R}^{mn}} \quad L_a(x, \lambda) \tag{6}$$

The corresponding dual problem is:

$$\max_{\lambda \in \mathbb{R}^{mn}} \quad \min_{x \in \mathbb{X}} \quad L_a(x, \lambda) \tag{7}$$

Lemma 2: *A saddle point (x^*, λ^*) is considered as a primal-dual solution pair to problems (6) and (7) when the following condition is satisfied [20]*

$$L_a(x^*, \lambda) \leq L_a(x^*, \lambda^*) \leq L_a(x, \lambda^*) \tag{8}$$

where x and λ represent the primal and dual variables, respectively.

Remark 2: According to Lemma 2, when (x^*, λ^*) is a saddle point of $L_a(x, \lambda)$, it is known that x^* is a primal optimization solution to problem (6), equivalently, a global optimization solution to problem (2). Thus, we can address distributed optimization problem (2) by resorting to a saddle method.

V. Algorithms

In this section, we discuss an existing primal-dual distributed algorithm for problem (2).

A. Primal-dual distributed algorithm

A primal-dual distributed algorithm for problem (2) is proposed by Lei et al. [21] on the basis of the augmented lagrangian function. Let x_k^i and λ_k^i be the local estimates for the primal and dual variables of node i at time k, respectively. In [21], at time k the variables x_k^i and λ_k^i are updated as

$$\begin{aligned} x_{k+1}^i = P_{X_i}[x_k^i - \alpha \nabla f_i(x_k^i) - \alpha \sum_{j=1}^m a_{ij}(x_k^i - x_k^j) \\ -\alpha \sum_{j=1}^m a_{ij}(\lambda_k^i - \lambda_k^j)] \end{aligned} \tag{9}$$

$$\lambda_{k+1}^i = \lambda_k^i + \alpha \sum_{j=1}^m a_{ij}(x_{k+1}^i - x_{k+1}^j) \tag{10}$$

where α represents a constant step-size or learning rate, and $\nabla f_i(x_k^i)$ gives the gradient of f_i at x_k^i . In essence, equations (9) and (10) could be viewed as a gradient-descent algorithm for finding the primal-dual solution pair (so called a saddle point) of the augmented Lagrangian function $\tilde{L}_a(x, \lambda) = L_a(x, \lambda) + \frac{1}{2} x^T L x$.
Note that f_i is the set of q_i local instantaneous functions available at node i. At time k, updating equation (9) requires that each node i evaluates the true gradient of f_i as follows:

$$\nabla f_i(x_k^i) = \frac{1}{q_i} \sum_{j=1}^{q_i} \nabla f_i^j(x_k^i) \tag{11}$$

VI. Metropolis Weights

A. Time varying Graphs

Consider a set of interconnected processes (nodes of a network) $\mathcal{N} = \{1, 2, ..n\}$, each with an initial scalar value $x_i(0)$. The allowed communication pattern of the processes varies with time, and is specified by the set of active edges $\mathcal{E}(t)$ at time $t = 0, 1, 2....$ The edges are undirected, with $(i,j) \in \mathcal{E}(t)$ meaning the processes i and j can communicate with each other at time t. Define $W_{i,j}(t)$ as the linear weight on $x_j(t)$ at node i and $\mathcal{N}_i(t) = \{j | (i,j) \in \mathcal{E}(t)\}$ denotes the set of neighbors of node i at time t. The question is how to choose the weights $W_{i,j}(t)$

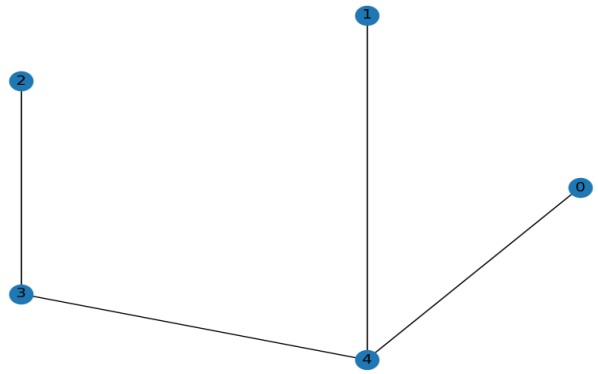

Fig. 1: Randomly generated graph

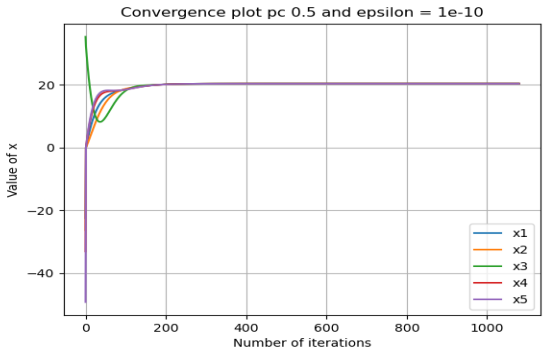

Fig. 2: Convergence plot of x values

such that every $x_i(t)$ converges to the average of their initial values, i.e.,

$$\lim_{t\to\infty} \quad x_i(t) = \frac{1}{n}\sum_{i=1}^{n} x_i(0),\ i = 1,\ldots,n \tag{12}$$

More details and background can be found in [16].

B. Formulating the weights

Set $d_i(t) = |\mathcal{N}_i(t)|$. Clearly, $d_i(t)$ is the degree of node i at time t. The Metropolis weights on a time varying graph $(\mathcal{N}(t), \mathcal{E}(t))$ are then defined as [16]

$$W_{ij}(t) = \begin{cases} \frac{1}{1+max\{d_i(t),d_j(t)\}}, & \text{if} \quad (i,j) \in \mathcal{E}(t) \\ 1 - \sum_{k\in\mathcal{N}_i(t)} W_{ik}(t), & \text{if} \quad i = j \\ 0 & \text{otherwise} \end{cases}$$

In other words, the weight on each edge is one over one plus the larger degree at its two incident nodes, and the self-weights $W_{ii}(t)$ are chosen so the sum of weights at each node is 1. More details about the motivation, history and convergence proofs behind the Metropolis weights can be found in [16].

An example graph is shows in Figure 1. The corresponding Metropolis weights are given below.

$$\mathbf{W} = \begin{bmatrix} 0.75 & 0. & 0. & 0. & 0.25 \\ 0. & 0.75 & 0. & 0. & 0.25 \\ 0. & 0. & 0.67 & 0.33 & 0. \\ 0. & 0. & 0.33 & 0.42 & 0.25 \\ 0.25 & 0.25 & 0. & 0.25 & 0.25 \end{bmatrix}$$

VII. Numerical Experiments

The simulations are conducted in an undirected connected network of m nodes with the probability of edges p_c. For the sake of simplicity of implementation, we choose $q_i = 1, i = \{1, ..m\}$, i.e, the objective at each node is now just one single function instead of being the average of q_i functions. The local instantaneous function $f_i(x)$ is given by:

$$f_i(x) = \|x - a_i\|^2 \tag{13}$$

where $x, a_i \in \mathbb{R}^n$. The m nodes cooperatively address the following distributed optimization problem.

$$\begin{aligned} \underset{x\in\mathbb{R}^n}{\text{minimize}} \quad & \sum_{i=1}^{m} f_i(x) \\ \text{subject to} \quad & x \in X \end{aligned} \tag{14}$$

Where the constrained set $X = \cap_{i=1}^{m} X_i$. For the adjacency matrix, we use the Metropolis weights described in Section VI.

A. Example plot

Here we choose $m = 5$ and $n = 1$. p_c is taken to be 0.5. Each of the constrained sets is taken to be the positive half of the real line, ie, $\forall i = \{1, .., m\}$,

$$X_i = \{x | x >= 0\}$$

The convergence criteria is set to be $\epsilon = 10^{-10}$. The value of α is set to be 0.01. Initial x, λ values at each node and a_i values are randomly generated, the details of which can be found in the code. The pseudo code is given below.

```
initialize a;
initialize x values;
initialize lambda values:
while algorithm has not converged:
    generate a random graph with edge\
    probability pc;
    calculate the metropolis weights;
    update x;
    update lambda;
    save x values for plotting;
```

In one run of this experiment, the a values are given by $[a_1, a_2, a_3, a_4, a_5] = [15, -2, 19, 33, 37]$. The convergence plot is shown in Figure 2. The x values, as expected, converge to $[20.4, 20.4, 20.4, 20.4, 20.4]$

B. Time taken to converge

In this section, we repeat the experiment varying m from 2 to 51 and check the time taken for convergence in each

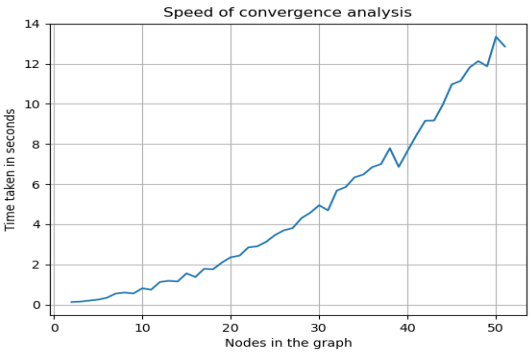

Fig. 3: Complexity analysis

case. The plot is shown in Figure 3. A few observations can be made.

- The time taken, as expected, increases with the number of nodes
- The rate of increase is quite clearly higher than a straight line, denoting the fact that the complexity in m is higher than a linear factor.

References

[1] Youcheng Niu, Haijing Wang, Zheng Wang, Dawen Xia, and Huaqing Li. Primal-dual stochastic distributed algorithm for constrained convex optimization. *Journal of the Franklin Institute*, 356(16):9763–9787, 2019.

[2] Robin L Raffard, Claire J Tomlin, and Stephen P Boyd. Distributed optimization for cooperative agents: Application to formation flight. In *2004 43rd IEEE Conference on Decision and Control (CDC)(IEEE Cat. No. 04CH37601)*, volume 3, pages 2453–2459. IEEE, 2004.

[3] Yi Wang, Zhongjun Ma, and Guanrong Chen. Distributed control of cluster lag consensus for first-order multi-agent systems on quad vector fields. *Journal of the Franklin Institute*, 355(15):7335–7353, 2018.

[4] Li Dai, Qun Cao, Yuanqing Xia, and Yulong Gao. Distributed mpc for formation of multi-agent systems with collision avoidance and obstacle avoidance. *Journal of the Franklin Institute*, 354(4):2068–2085, 2017.

[5] Michael Rabbat and Robert Nowak. Distributed optimization in sensor networks. In *Proceedings of the 3rd international symposium on Information processing in sensor networks*, pages 20–27, 2004.

[6] Lin Xiao, Stephen Boyd, and Sanjay Lall. A scheme for robust distributed sensor fusion based on average consensus. In *IPSN 2005. Fourth International Symposium on Information Processing in Sensor Networks, 2005.*, pages 63–70. IEEE, 2005.

[7] Huaqing Li, Shuai Liu, Yeng Chai Soh, and Lihua Xie. Event-triggered communication and data rate constraint for distributed optimization of multiagent systems. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 48(11):1908–1919, 2017.

[8] Bei Wang, Zhichao Li, and Xuefeng Yan. Multi-subspace factor analysis integrated with support vector data description for multimode process monitoring. *Journal of the Franklin Institute*, 355(15):7664–7690, 2018.

[9] Felix Brockherde, Leslie Vogt, Li Li, Mark E Tuckerman, Kieron Burke, and Klaus-Robert Müller. Bypassing the kohn-sham equations with machine learning. *Nature communications*, 8(1):1–10, 2017.

[10] Renato LG Cavalcante, Isao Yamada, and Bernard Mulgrew. An adaptive projected subgradient approach to learning in diffusion networks. *IEEE Transactions on Signal Processing*, 57(7):2762–2774, 2009.

[11] Chiang-Cheng Chiang and Mon-Han Chen. Robust adaptive fuzzy control of uncertain nonlinear time-delay systems with an unknown dead-zone. In *2008 IEEE International Conference on Fuzzy Systems (IEEE World Congress on Computational Intelligence)*, pages 381–387. IEEE, 2008.

[12] Usman A Khan, Soummya Kar, and José MF Moura. Diland: An algorithm for distributed sensor localization with noisy distance measurements. *IEEE Transactions on Signal Processing*, 58(3):1940–1947, 2009.

[13] Yifeng He, Ivan Lee, and Ling Guan. Distributed algorithms for network lifetime maximization in wireless visual sensor networks. *IEEE Transactions on Circuits and Systems for Video Technology*, 19(5):704–718, 2009.

[14] Jianli Zhao, Yingyou Wen, Ruiqiang Shang, and Guangxing Wang. Optimizing sensor node distribution with genetic algorithm in wireless sensor network. In *International Symposium on Neural Networks*, pages 242–247. Springer, 2004.

[15] Huaqing Li, Qingguo Lü, and Tingwen Huang. Convergence analysis of a distributed optimization algorithm with a general unbalanced directed communication network. *IEEE Transactions on Network Science and Engineering*, 6(3):237–248, 2018.

[16] Lin Xiao, Stephen Boyd, and Sanjay Lall. Distributed average consensus with time-varying metropolis weights. *Automatica*, 2006.

[17] Wikipedia. Kronecker product — Wikipedia, the free encyclopedia. http://en.wikipedia.org/w/index.php?title=Kronecker%20product&oldid=967348003, 2020. [Online; accessed 28-July-2020].

[18] Aryan Mokhtari and Alejandro Ribeiro. Global convergence of online limited memory bfgs. *The Journal of Machine Learning Research*, 16(1):3151–3181, 2015.

[19] Qingshan Liu and Jun Wang. A second-order multi-agent network for bound-constrained distributed op-

timization. *IEEE Transactions on Automatic Control*, 60(12):3310–3315, 2015.

[20] Stephen Boyd, Stephen P Boyd, and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[21] Jinlong Lei, Han-Fu Chen, and Hai-Tao Fang. Primal–dual algorithm for distributed constrained optimization. *Systems & Control Letters*, 96:110–117, 2016.