Distributed Nonconvex Optimization via Polynomial Approximation

Presenter: Jianping He*

Coauthors: Zhiyu He, Cailian Chen and Xinping Guan

IWIN-FINS

Department of Automation

Shanghai Jiao Tong University

May 2021

*Corresponding author: Jianping He, Email: jphe@sjtu.edu.cn

Contents

Distributed Optimization

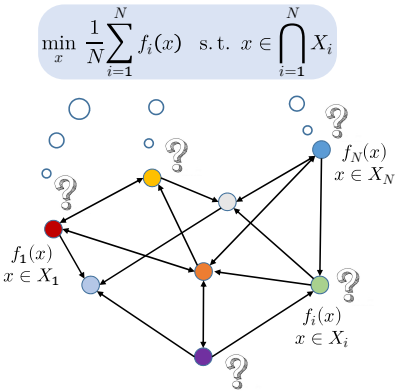


Figure 1 An illustration of distributed optimization

- What is distributed optimization?

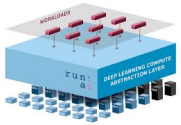
to enable agents in networked systems to collaboratively optimize the average of local objective functions.

- Why not centralized optimization?
  - possible lack of central authority
  - efficiency, privacy-preserving, robustness and scalability issues[1]

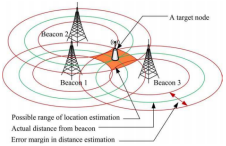
[1] A. Nedić *et al.*, "Distributed optimization for control," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 1, pp. 77–103, 2018

Distributed Optimization: Application Scenarios

- Distributed optimization empowers networked multi-agent systems

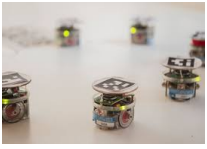


(a) Distributed Learning[2]



(b) Distributed Localization in Sensor Networks[3]

(c) Distributed Coordination in Smart Grids[4]



(d) Distributed Control of Multi-robot Formations[5]

Figure 2 Application scenarios of distributed optimization

[2] S. Boyd *et al.*, *Found. Trends Mach. Learn.*, 2011, [3] Y. Zhang *et al.*, *IEEE Trans. Wireless Commun.*, 2015, [4] C. Zhao *et al.*, *IEEE Trans. Smart Grid*, 2016, [5] W. Ren *et al.*, *ROBOT AUTON SYST.*, 2008.

Distributed Optimization: Application Scenarios

- **Distributed Learning**

Suppose that the training sets are so large that they are stored separately at multiple servers.

We aim to train the model so that the overall loss function is minimized.

$$\min_x F(x) = \sum_i f_i(x),$$
$$f_i(x) = \sum_{j \in \mathcal{D}_i} l_j(x),$$

where $\mathcal{D}_i$ denotes local dataset, and $f_i(\cdot), l_j(\cdot)$ denote loss functions.

- **Distributed Coordination in Smart Grid**

We aim to coordinate the power generation of a set of distributed energy resources, so that ▷ demand is met, ▷ total cost is minimized.

$$\min \sum_{i=1}^{N} f_i(P_i),$$
$$\text{s.t.} \sum_{i=1}^{N} P_i = P_d,$$
$$\text{s.t.} \ \underline{P_i} \le P_i \le \overline{P_i},$$

where $f_i(\cdot)$ denotes the function of generation cost of each energy resource.

Preliminaries: Gradient Descent

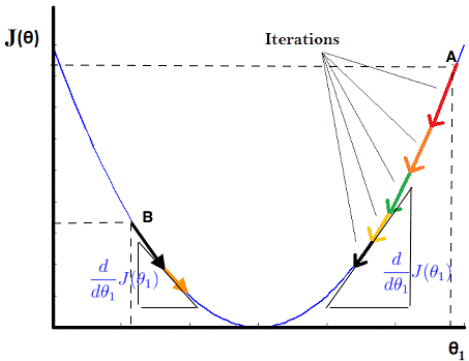


Figure 3 An illustration of gradient descent

Convex Optimization

$$\min_{x\in\mathbb{R}^n} f(x), \quad f(x) \text{ is convex.}$$

Update Rule of GD

$$x_i^{t+1} = x^{t+1} - \alpha^t \nabla f(x^t)$$

α^t can be fixed or chosen from backtracking line search method.

Convergence Rates

- $\mathcal{O}(1/t)$, for convex and smooth $f(x)$
- $\mathcal{O}(\rho^t)$, i.e., linear, for strongly convex and smooth $f(x)$

Preliminaries: Average Consensus

▶ **Question**

After taking an exam, every student i knows his/her own score $x_i \in [0, 100]$.

If students can only know the scores of themselves and their close friends, how can they figure out the average score $\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i$ of the whole class?

▶ **Thoughts**

- Continuously communicate with close friends $\Longrightarrow$ Local information can be propagated
- Properly utilize the available info $\Longrightarrow$ What about taking linear combinations? OK!

Preliminaries: Average Consensus

- **Update Rule**

$$x_i^{t+1} = \sum_{j \in \mathcal{N}_i} w_{ij} x_j^t, \qquad \text{(scalar form)}$$

$$x^{t+1} = W x^t, \qquad \text{(vector form)}$$

where $x^t = [x_1^t, \ldots, x_N^t]^T$, $W = [w_{ij}]_{i,j=1}^N$.

- **Necessary and Sufficient Condition for Average Consensus**

Average consensus is achieved (i.e., $\lim_{t\to\infty} x_i^t = \bar{x}$) *iff*

$$1^T W = 1^T, \quad W1 = 1, \quad \rho(W - 11^T/N) < 1.^6$$

[6] L. Xiao *et al.*, "Fast linear iterations for distributed averaging," *Syst. Control. Lett.*, 2004

Distributed Gradient Descent

Convex Distributed Optimization $\min_{x\in\mathbb{R}^n} \quad f(x) = \frac{1}{N}\sum_{i=1}^{N} f_i(x), \quad \forall i, f_i(x)$ is convex.

Distributed Gradient Descent (DGD)[7]

$$x_i^{t+1} = \sum w_{ij} x_j^t - \alpha^t \nabla f_i(x_i^t)$$

Averaging for reaching consensus — Local GD for reaching optimality

Assumptions • diminishing step sizes α^t • W doubly stochastic • bounded gradients $\|\nabla f_i\| \leq L$

Q: Why diminishing α^t? **A:** If not, even if we are at the global optimizer x^*, the algorithm may not stop, since the local gradient is not necessarily 0.

Sub-linear Convergence

$$f(\hat{x}_i^t) - f^* \sim \mathcal{O}\left(\frac{1}{\sqrt{t}}\right), \quad \hat{x}_i^t = \frac{1}{t}\sum_{k=0}^{t-1} x_i^k$$

can be improved to linear convergence rates with Gradient Tracking[8]

[7] A. Nedic *et al.*, *IEEE Trans. Autom. Control*, 2009, [8] P. Di Lorenzo *et al.*, *IEEE Trans. Signal Inf. Pr.*, 2016, J. Xu *et al.*, *IEEE Trans. Autom. Control*, 2017

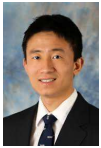
Developments of Distributed Optimization

A. Nedich
ASU

W. Shi
Princeton

A. Olshevsky
BU

G. Scutari
Purdue



M. Hong
UMN

DGD
undirected graph
sub-linear rate
(2009)

EXTRA
undirected graph
linear rate
(2015)

Push-DIGing
directed graph
linear rate
(2017)

SONATA
directed graph
(2019)
1st-order

ZONE
undirected graph
(2019)
0th-order

1st-order convex
optimization algorithms

non-convex
optimization algorithms

[9] A. Nedic *et al.*, *IEEE Trans. Autom. Control*, 2009, [10] W. Shi *et al.*, *SIAM J. Optim.*, 2015, [11] A. Nedic *et al.*, *SIAM J. Optim.*, 2017, [12] G. Scutari *et al.*, *Math. Program.*, 2019, [13] D. Hajinezhad *et al.*, *IEEE Trans. Autom. Control*, 2019.

Developments of Distributed Optimization

- We classify existing distributed optimization algorithms into two categories:
  - **Primal Methods:** Distributed (sub)Gradient Descent[14], Fast-DGD[15], EXTRA[16], DIGing[17], Acc-DNGD[18], ZONE[19], SONATA[20]...

    feature: combine (sub)gradient descent with consensus, so as to drive local estimates to converge in the primal domain
  - **Dual-based Methods:** Dual Averaging[21], D-ADMM[22], DCS[23], MSDA[24], MSPD[25], ...

    feature: introduce consensus equality constraints, and then solve the dual problem or carry on primal-dual updates to reach a saddle point of the Lagrangian
- Please refer to [T. Yang *et al.*, *Annu Rev Control*, 2019] for a recent comprehensive survey.

[14] A. Nedic *et al.*, *IEEE Trans. Autom. Control*, 2009, [15] D. Jakovetić *et al.*, *IEEE Trans. Autom. Control*, 2014, [16] W. Shi *et al.*, *SIAM J. Optim.*, 2015, [17] A. Nedic *et al.*, *SIAM J. Optim.*, 2017, [18] G. Qu *et al.*, *IEEE Trans. Autom. Control*, 2019, [19] D. Hajinezhad *et al.*, *IEEE Trans. Autom. Control*, 2019, [20] G. Scutari *et al.*, *Math. Program.*, 2019, [21] J. C. Duchi *et al.*, *IEEE Trans. Autom. Control*, 2011, [22] W. Shi *et al.*, *IEEE Trans. Signal Process.*, 2014, [23] G. Lan *et al.*, *Math. Program.*, 2017, [24] K. Scaman *et al.*, in *Proc. Int. Conf. Mach. Learn.*, 2017, [25] K. Scaman *et al.*, in *Adv Neural Inf Process Syst*, 2018.

Motivations

General Distributed Optimization

$$\min_{x \in X} \quad f(x) = \frac{1}{N} \sum_{i=1}^{N} f_i(x)$$

$f(x)$: possibly noncovex

Generic Methods with Gradient Tracking

$$x_i^{t+1} = \mathcal{F}_t \left(\sum w_{ij} x_j^t, s_i^t \right)$$

$$s_i^{t+1} = \sum w_{ij} s_j^t + \underbrace{\nabla f_i(x_i^{t+1}) - \nabla f_i(x_i^t)}_{\text{eval of gradients at every itr}}$$

Two notable unresolved issues within the existing work

- growing load of oracle queries with respect to iterations
  - ▷ results from evaluations of gradients or values of local objectives within every iteration
- hardness of achieving iterative convergence to global optimal points
  - ▷ results from the nonconvex nature of general objectives

Contributions

Main contributions of this work

- We propose a novel algorithm, leveraging polynomial approximation, consensus and SDP theories.
- CPCA has the advantages of
  - able to obtain ϵ globally optimal solutions $\Longleftarrow$ ϵ is any arbitrarily small given tolerance
  - computationally efficient $\Longleftarrow$ the required 0^{th}-order oracle queries are independent of iterations
  - distributively terminable once the precision requirement is met
- We provide a comprehensive analysis of the accuracy and complexities of CPCA

Problem Formulation

The constrained distributed nonconvex optimization problem we consider is

$$\begin{aligned} \min_{x} \quad & f(x) = \frac{1}{N} \sum_{i=1}^{N} f_i(x), \\ \text{s.t.} \quad & x \in X = \bigcap_{i=1}^{N} X_i, \quad X_i \subset \mathbb{R}. \end{aligned}$$

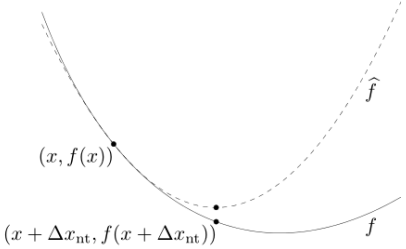
Note

- We only require possibly non-convex univariate $f_i(x)$ to be Lipschitz continuous on convex X_i.
- We assume that $\mathcal{G}$ is an undirected graph. The extension to time-varying directed graphs is presented in our recent work.

Key Ideas

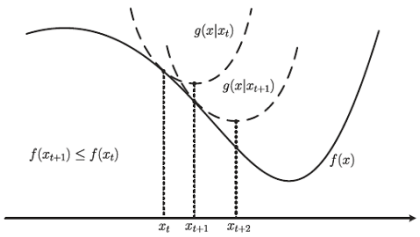
- **Inspirations**

  Approximation is closely linked with optimization.



(a) Newton's method

Source: S. Boyd *et al.*, *Convex optimization*. 2004



(b) Majorization-Minimization Algorithm

Source: Y. Sun *et al.*, *IEEE Trans. Signal Process.*, 2016

Figure 4 Optimization algorithms based on approximation

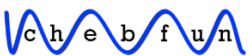
Both of them are based on local approximations. What if global approximations?

Key Ideas

- **Inspirations**

  Researchers use Chebyshev polynomial approximation to substitute for the target function defined on an interval, so as to make the study of its property much easier.

$$f(x) \approx p(x) = \sum_{i=0}^{m} c_i T_i \left(\frac{2x - (a+b)}{b-a} \right), \quad x \in [a, b].$$



Chebfun Toolbox for MATLAB

- **Insights**

  turn to optimize the approximation (i.e. the proxy) of the global objective, to obtain ϵ-optimal solutions for any arbitrarily small given error tolerance ϵ

  - use average consensus to enable every agent to obtain such a global proxy
  - optimize locally the global proxy by finding its stationary points, or solving SDPs

Overview of CPCA

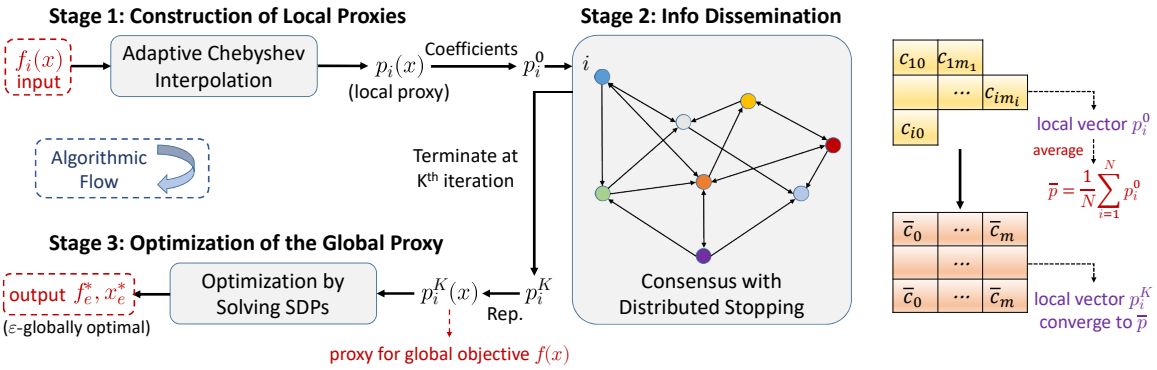


Figure 5 The architecture of CPCA

Initialization: Construction of Local Chebyshev Proxies

- **Goal**

  Construct the Chebyshev polynomial approximation $p_i(x)$ for $f_i(x)$, such that

  $$|f_i(x) - p_i(x)| \leq \epsilon_1, \quad \forall x \in X,$$

  where $X = \bigcap_{i=1}^{N} X_i \triangleq [a, b]$.

- **Details**

  1. Run a finite number of max/min consensus iterations in advance to obtain the intersection set X.
  2. Use Adaptive Chebyshev Interpolation[26] to obtain $p_i(x)$.
  3. Maintain p_i^0 storing the Chebyshev coefficients of $p_i(x)$'s derivative through certain recurrence formula.

[26] J. P. Boyd, *Solving Transcendental Equations: The Chebyshev Polynomial Proxy and Other Numerical Rootfinders, Perturbation Series, and Oracles*. SIAM, 2014, vol. 139.

Initialization: Construction of Local Chebyshev Proxies

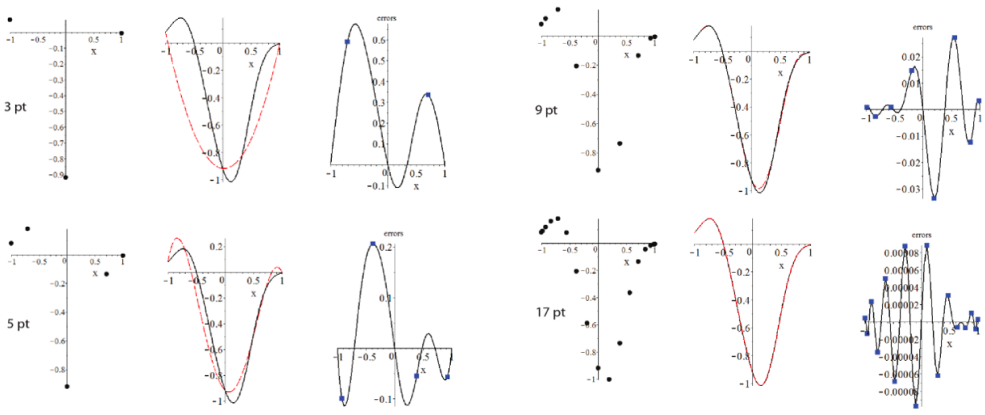
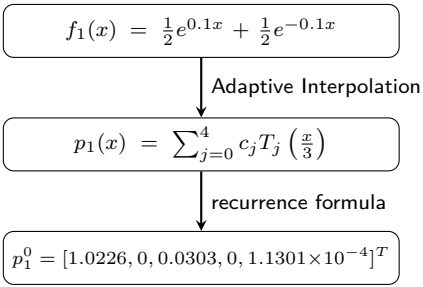


Figure 6 An illustration of Adaptive Chebyshev Interpolation

Source: J. P. Boyd. SIAM, 2014, vol. 139

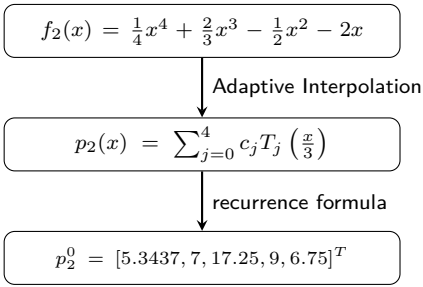
Initialization: Construction of Local Chebyshev Proxies

- **Examples**
  - ▷ Setup: precision requirement $\epsilon_1 = 10^{-6}$, constraint set $X = [-3, 3]$
    - ○ Case I



(In fact, $|f_1(x) - p_1(x)| \leq 4.8893 \times 10^{-8},\ x \in X$.)

    - ○ Case II



(In fact, $|f_2(x) - p_2(x)| \leq 1.7036 \times 10^{-14},\ x \in X$.)

Iteration: Consensus-based Update of Local Vectors

- **Goal**

  Make local vectors p_i^K converge to the average $\bar{p}$ of all the initial values p_i^0, i.e.,

  $$\max_{i \in \mathcal{V}} \left\| p_i^K - \bar{p} \right\|_\infty \leq \delta,$$

  where

  $$\delta = \frac{\epsilon_2}{1 + \frac{b-a}{2}\left(\ln m + \frac{3}{2}\right)}$$

  is proportional to the given precision ϵ_2, with $m = \max_{i \in \mathcal{V}} m_i$.

- **Strategies**

  Run linear time average consensus[27] for certain rounds.

[27] A. Olshevsky, *SIAM J. Optim.*, 2017.

Iteration: Consensus-based Update of Local Vectors

- **Further Assumption:** Every agent in the network knows an upper bound U on N.
- **Iteration Rules**

$$\begin{cases} p_i^k = q_i^{k-1} + \dfrac{1}{2}\displaystyle\sum_{j\in\mathcal{N}_i} \dfrac{q_j^{k-1} - q_i^{k-1}}{\max(d_i, d_j)}, \\ q_i^k = p_i^k + \left(1 - \dfrac{2}{9U+1}\right)(p_i^k - p_i^{k-1}). \end{cases}$$

The number of iterations K is set as

$$K \leftarrow \max\left(\left\lceil \frac{\ln(\delta/2\sqrt{2U}\|r_i^U - s_i^U\|_\infty)}{\ln \rho} \right\rceil, U\right),$$

where $\rho = \sqrt{1 - 1/(9U)}$ is the decaying rate of the error[28], and r_i^k, s_i^k are two variables updated based on max/min consensus, so that $\|r_i^U - s_i^U\|_\infty$ equals to $\max_{i,j\in\mathcal{V}} \left\|p_i^0 - p_j^0\right\|_\infty$.

[28] A. Olshevsky, *SIAM J. Optim.*, 2017.

Iteration: Consensus-based Update of Local Vectors

Lemma 1

With $K \sim \mathcal{O}\left(N \log\left(\frac{N \log m}{\epsilon_2}\right)\right)$ *iterations, we have*

$$\max_{i \in \mathcal{V}} \left\| p_i^K - \bar{p} \right\|_\infty \leq \delta.$$

- The proximity between p_i^K and $\bar{p}$ translates to

$$|p_i^K(x) - \bar{p}(x)| \leq \epsilon_2,$$

where $p_i^K(x)$, $\bar{p}(x)$ are the Chebyshev polynomials recovered from p_i^K, $\bar{p}$, respectively.

Iteration: Consensus-based Update of Local Vectors

- The order of K can be brought down to $\mathcal{O}\left(N \log\left(\frac{\log m}{\epsilon_2}\right)\right)$ by incorporating distributed stopping mechanism[29] into consensus iterations.

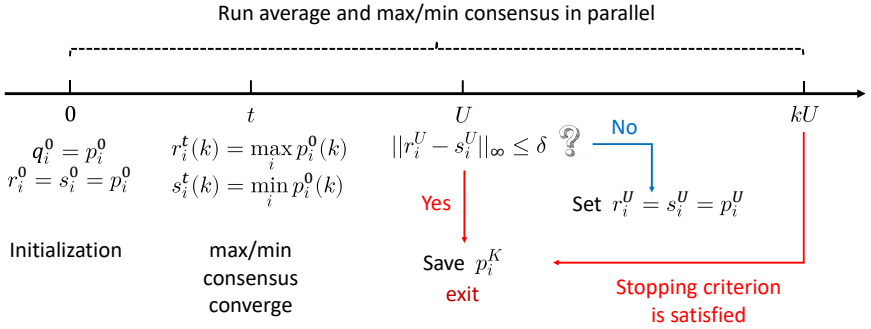


Figure 7 An illustration of average consensus with distributed stopping

[29] V. Yadav *et al.*, in *Proc. 45th Annu. Allerton Conf.*, 2007.

Iteration: Consensus-based Update of Local Vectors

- **When CPCA is extended to time-varying digraphs, the iteration rules become**
  - Set $x_i^0 \leftarrow p_i^0,\ y_i^0 \leftarrow 1$, and update x_i^t and y_i^t according to push-sum average consensus

$$x_i^{t+1} = \sum_{j=1}^{N} a_{ij}^t x_j^t, \quad y_i^{t+1} = \sum_{j=1}^{N} a_{ij}^t y_j^t,$$

where a_{ij}^t is set as $1/d_i^{\text{out},t}$ if $j \in \mathcal{N}_i^{\text{in},t}$, and 0 otherwise.

Note: $p_i^t \triangleq x_i^t / y_i^t$ converges to $\bar{p}$ geometrically.

  - Update auxiliary variables r_i^t and s_i^t in parallel according to max/min consensus.

$$r_i^{t+1}(k) = \max_{j \in \mathcal{N}_i^{\text{in},t}} r_j^t(k), \quad s_i^{t+1}(k) = \min_{j \in \mathcal{N}_i^{\text{in},t}} s_j^t(k), \quad k = 0, \ldots, m.$$

These variables are reinitialized as $p_i^t \triangleq x_i^t / y_i^t$ every U iterations.

Iteration: Consensus-based Update of Local Vectors

- **Iteration rules of CPCA when extended to time-varying digraphs**

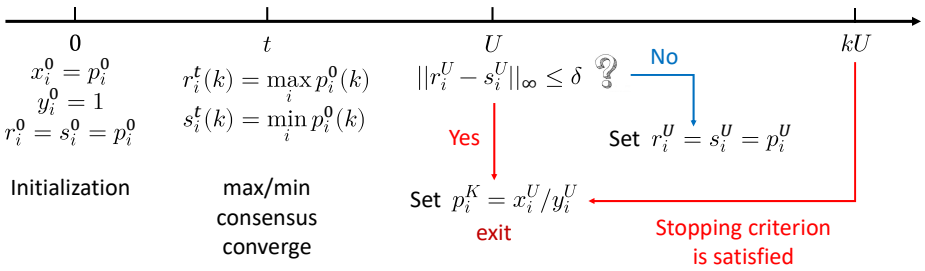


Figure 8 An illustration of push-sum consensus with distributed stopping

Optimize Polynomial Proxy

- **Goal**

  Agent i optimize the polynomial proxy $p_i^K(x)$ recovered from p_i^K.

- **Intuitions**

  ▷ After the initialization, we have $|\bar{p}(x) - f(x)| \leq \epsilon_1,\ x \in X$.

  After the iteration, we have $|p_i^K(x) - \bar{p}(x)| \leq \epsilon_2,\ x \in X$.

  ▷ If we set $\epsilon_1 = \epsilon_2 = \frac{\epsilon}{2}$, it follows that $|p_i^K(x) - f(x)| \leq \epsilon,\ x \in X$.

  ▷ The difference between the optimal values of $f(x)$ and $p_i^K(x)$ is less than ϵ.

  $\Longrightarrow$ The points in the optimal set X_e^* of $p_i^K(x)$ are ϵ-optimal solutions of the considered problem.

Optimize Polynomial Proxy Based on Stationary Points

- **Procedures**
  1. Recover the polynomial proxy $p_i^K(x)$ from p_i^K.
  2. Construct the colleague matrix M_C from p_i^K, and compute its real eigenvalues. (These are the stationary points of $p_i^K(x)$.)

$$M_C = \begin{bmatrix} 0 & 1 & & & & & \\ \frac{1}{2} & 0 & \frac{1}{2} & & & & \\ & \frac{1}{2} & 0 & \frac{1}{2} & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & \frac{1}{2} & 0 & \frac{1}{2} & \\ -\frac{c_0}{2c_m} & -\frac{c_1}{2c_m} & -\frac{c_2}{2c_m} & \cdots & \frac{1}{2} - \frac{c_{m-2}}{2c_m} & -\frac{c_{m-1}}{2c_m} \end{bmatrix}_{m \times m}$$

  3. Compute and compare the critical values of $p_i^K(x)$, and take the optimal points to form X_e^*.

Optimize Polynomial Proxy Based on Stationary Points

- **Why are the eigenvalues of M_C exactly the stationary points of $p_i^K(x)$?**
  - ▷ Note that for Chebyshev polynomials, we have

$$\frac{1}{2}T_{k-1}(x) + \frac{1}{2}T_{k+1}(x) = xT_k(x).$$

Let $v = [T_0(x), \ldots, T_{n-1}(x)]^T$. If x is the root of $\mathrm{d}p_i^K(x)/\mathrm{d}x = 0$, then $M_C v = xv$. Hence, the n roots of $\mathrm{d}p_i^K(x)/\mathrm{d}x = 0$ correspond to n eigenvalues of M_C.

Compare: The roots of $p(x) = a_0 + a_1 x + \ldots + a_n x^n = 0$ are the eigenvalues of

$$C = \begin{bmatrix} 0 & 1 & & & \\ & 0 & 1 & & \\ & & \ddots & \ddots & \\ & & & 0 & 1 \\ -\frac{a_0}{a_n} & -\frac{a_1}{a_n} & \cdots & -\frac{a_{n-2}}{a_n} & -\frac{a_{n-1}}{a_n} \end{bmatrix}.$$

Note: This method is suitable for numerical computations, but involves some errors that can't be theoretically characterized.

Alternative: Optimize Polynomial Proxy by Solving SDPs

- **Goal**

  Agent i optimize the polynomial proxy $p_i^K(x)$ recovered from p_i^K.

- **Intuitions**

  - The optimization of $p_i^K(x)$ on $[a, b]$ is equivalent to

$$\max_{x,t} \; t \quad \text{s.t. } p_i^K(x) - t \text{ is non-negative, } x \in [a, b].$$

  - For $g(x) \triangleq p_i^K(x) - t$, its non-negativity on $[a, b]$ holds if and only if it can be expressed as

$$g(x) = \begin{cases} (x-a)h_1(x) + (b-x)h_2(x), & \text{if } m \text{ is odd,} \\ h_1(x) + (x-a)(b-x)h_2(x), & \text{if } m \text{ is even,} \end{cases}$$

  where $h_1(x), h_2(x)$ are sum of squares (SOS), and are of even degree[30].

  - SOS is linked with positive semi-definiteness. $\Longrightarrow$ The problem can be transformed to a SDP.

[30] Y. Nesterov, "Squared functional systems and optimization problems," in *High performance optimization*, Springer, 2000.

Alternative: Optimize Polynomial Proxy by Solving SDPs

- **Procedures**

  Suppose $p_i^K = [c_0, c_1, \ldots, c_m]^T$. When m is odd, the SDP reformulation is

$$
\begin{aligned}
\max_{t,Q,Q'} \quad & t \\
\text{s.t.} \quad & c_0 = t + Q_{00} + Q'_{00} + \frac{1}{2}\sum_{u=1}^{m}\left(Q_{uu} + Q'_{uu}\right) + \frac{1}{4}\sum_{|u-v|=1}\left(Q_{uv} - Q'_{uv}\right) \\
& c_j = \frac{1}{2}\sum_{(u,v)\in\mathcal{A}}\left(Q_{uv} + Q'_{uv}\right) + \frac{1}{4}\sum_{(u,v)\in\mathcal{B}}\left(Q_{uv} - Q'_{uv}\right), \quad j = 1, \ldots, m, \\
& Q \in \mathbb{S}_+^{\lfloor m/2\rfloor+1},\ Q' \in \mathbb{S}_+^{\lfloor (m-1)/2\rfloor+1},
\end{aligned}
$$

where $\mathcal{A} = \{(u,v) | u+v = i \vee |u-v| = i\}$, $\mathcal{B} = \left\{(u,v) | u+v = i-1 \vee |u-v| = i-1 \vee |u+v-1| = i \vee \left||u-v|-1\right| = i\right\}$.

Note: • SDP can be efficiently solved through the use of CVX, which employs the interior-point method.

• An error tolerance ϵ_3 can be set to help terminate the solving procedure.

Accuracy of CPCA

Theorem 2

With CPCA, every agent obtains ϵ-optimal solutions for the considered problem, i.e.,

$$|f_e^* - f^*| \leq \epsilon,$$

where f^* is the optimal value.

- Every agent obtains ϵ-optimal solutions for any arbitrarily small given tolerance ϵ.
- ϵ is used to set certain parameters to regulate the stages of initialization and iteration.

Complexities of CPCA

Table 1 Complexities of CPCA

Stages	Elementary Operations	0^{th}-order Oracle Queries	Inter-communications
initialization	$\mathcal{O}\big(m^2 \log m\big)$	$\mathcal{O}(m)$	0
iteration	$\mathcal{O}\big(N \log\big(\frac{N \log m}{\epsilon}\big)\big)$	0	$\mathcal{O}\big(N \log\big(\frac{N \log m}{\epsilon}\big)\big)$
solve	$\mathcal{O}\big(m^3\big)$	0	0
whole	$\mathcal{O}\big(N \log\big(\frac{N \log m}{\epsilon}\big)\big)$	$\mathcal{O}(m)$	$\mathcal{O}\big(N \log\big(\frac{N \log m}{\epsilon}\big)\big)$

N: the size of the network m: the largest order of the polynomial approximations

Note:
- The oracle complexities are independent of N.
- m is relevant to the smoothness of objectives, and will not be very large generally (e.g, $10 \sim 10^2$).

Complexities of CPCA

Table 2 Comparisons of CPCA and Other State-of-the-arts for Nonconvex Distributed Optimization

Algorithms	**Networks**	**Oracles**		**Communications**
		0^{th}-order	1^{st}-order	
Alg. 1 [31]	I	$\mathcal{O}\left(\frac{d}{\epsilon}\right)$	/	$\mathcal{O}\left(\frac{d}{\epsilon}\right)$
SONATA[32]	II	/	$\mathcal{O}\left(\frac{1}{\epsilon}\right)$	$\mathcal{O}\left(\frac{1}{\epsilon}\right)$
CPCA	I	$\mathcal{O}(m)$	/	$\mathcal{O}\left(\log\left(\frac{\log m}{\epsilon}\right)\right)$
E-CPCA	II	$\mathcal{O}(m)$	/	$\mathcal{O}\left(\log\frac{m}{\epsilon}\right)$

Note: • I and II refers to static undirected and time-varying directed graphs, respectively.

• N denotes the number of agents, and m denotes the maximum degree of local approximations.

[31] Y. Tang *et al.*, *arXiv e-prints*, arXiv:1908.11444, 2019, [32] G. Scutari *et al.*, *Math. Program.*, 2019.

Numerical Experiments

▶ Optimization Over Static Undirected Graphs

Algorithms to Compare

- CPCA
- Distributed Projected sub-Gradient Descent (D-PGD)[33] (with step size $\eta_t = \frac{5}{4} \cdot \frac{N}{t}$).

Network Models

The network has $N = 36$ agents, and $\mathcal{G}$ varies from:

- 9-cycle graph
- 6×6 grid graph
- Erdos-Renyi random graph with connectivity probability 0.4

[33] A. Nedic *et al.*, "Constrained consensus and optimization in multi-agent networks," *IEEE Trans. Autom. Control*, vol. 55, no. 4, pp. 922–938, 2010.

Numerical Experiments

Objective Functions

- **Case I**: the objective functions are

$$f_i(x) = a_i e^{b_i x} + c_i e^{-d_i x}, \quad x \in X_i = [-3, 3],$$

where $a_i, c_i \sim \mathcal{U}(0, 1),\ b_i, d_i \sim \mathcal{U}(0, 0.2)$.

- **Case II**: the objective functions are

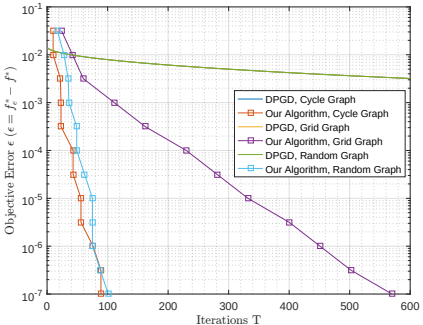
$$f_i(x) = a_i x^4 + b_i x^3 + c_i x^2 + d_i x + e_i, \quad x \in X_i = [-3, 3],$$

where a_i to e_i satisfy normal distributions, with μ being $1/4, 2/3, -1/2, -2$ and 0 respectively, and σ all being 0.1.

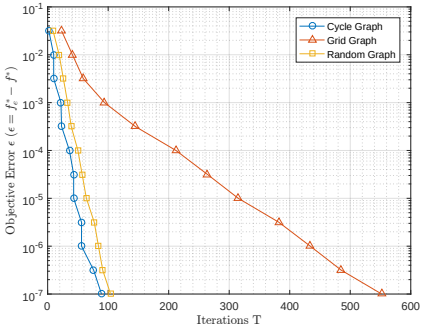
Note: **Case I**: convex objectives **Case II**: non-convex objectives

Numerical Experiments

- Horizontal axis: Number of Iterations
- Vertical axis: Objective Error ϵ



(a) Simulation Results for Case I



(b) Simulation Results for Case II

Figure 9 Comparison of CPCA and D-PGD

Note: ○ linear v.s. sub-linear convergence ○ applicable to the cases with non-convex objectives

Numerical Experiments

▶ Optimization Over Time-varying Directed Graphs

Algorithms to Compare

- E-CPCA
- SONATA-L[34]

Network Models

Consider a network of $N = 40$ agents, each of which has 2 out-neighbors besides itself at time t.

- one is on a fixed cycle
- the other is chosen uniformly at random

Objective Functions

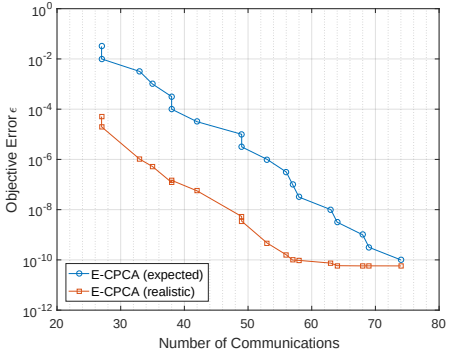
The *nonconvex but Lipschitz* objectives we choose are

$$f_i(x) = \frac{a_i}{1 + e^{-x}} + b_i \log(1 + x^2), \quad x \in X_i = [-5, 5], a_i \sim \mathcal{N}(10, 2), b_i \sim \mathcal{N}(5, 1).$$

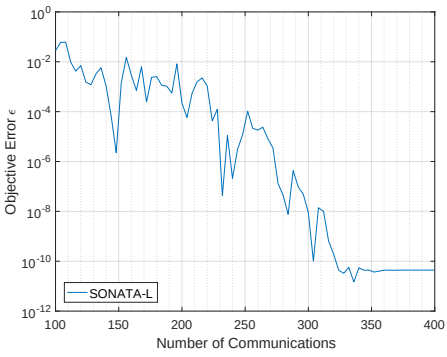
[34] G. Scutari *et al.*, "Distributed nonconvex constrained optimization over time-varying digraphs," *Math. Program.*, vol. 176, no. 1-2, pp. 497–544, 2019.

Numerical Experiments

- Horizontal axis: Number of Communications
- Vertical axis: Objective Error ϵ



(a) E-CPCA



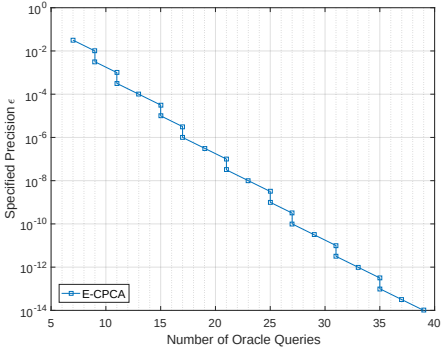
(b) SONATA-L

Figure 10 Comparison of both algorithms regarding inter-agent communications

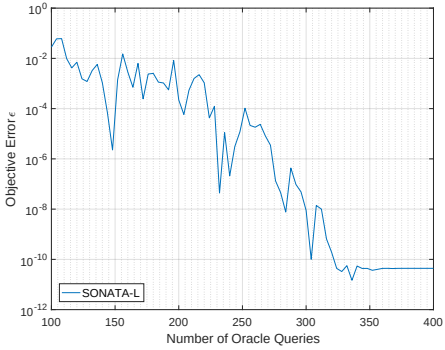
Note: E-CPCA is more communication-efficient due to its integrated rapidly convergent consensus protocols.

Numerical Experiments

- Horizontal axis: Number of Oracle Queries
- Vertical axis: Objective Error ϵ



(a) E-CPCA



(b) SONATA-L

Figure 11 Comparison of both algorithms regarding oracle queries

Note: Nor the increase of N or worsening of network's connectivity will change the curve in Fig. 11a.

Related Work on Privacy-Preserving Distributed Optimization

- **Motivation**

  DO involves local exchange of information (e.g., states or gradients)

  $\Longrightarrow$ may lead to disclosure of sensitive objective functions, constraints or state trajectories of agents

- **Prevention Strategies**

  - **Cryptography**[35]

    feature: use techniques like homomorphic encryption to prevent leakage to unauthorized users

    cons: costs associated with encryption and decryption are high

  - **Randomization**[36,37]

    feature: use differentially private mechanisms to perturb messages, so as to provide confidentiality

    cons: trade-off between accuracy and privacy[38]

[35] Y. Lu *et al.*, *Automatica*, 2018, [36] S. Han *et al.*, *IEEE Trans. Autom. Control*, 2017, [37] E. Nozari *et al.*, *IEEE Trans. Control Netw. Syst.*, 2018, [38] T. Ding *et al.*, *IEEE Trans. Autom. Control*, 2021.

Design of Privacy-Preserving Information Dissemination

- **Requirement**

  in consensus-based info dissemination, preserve the privacy of local objective function $f_i(x)$

  $\Longrightarrow$ reduced to the aim of preserving the privacy of local initial vector p_i^0

- **Building Block**

  push-sum average consensus[39] $\Longrightarrow$ achieve averaging in time-varying directed networks

- **Key Ideas**

  1. perturb the initial vector p_i^0 with noise θ_i to obtain $\tilde{p}_i^0$
  2. append $\tilde{p}_i^0$ block-by-block to current states *(first K_1 iterations)*

     $\Rightarrow$ hide useful perturbed initial values within iterations
  3. separately subtract the added noise θ_i *(next $K_2 - K_1$ iterations)*

     $\Rightarrow$ ensure exact convergence; mitigate the negative impact of persistent noises on convergence rates

[39] D. Kempe *et al.*, "Gossip-based computation of aggregate information," in *Proc. 44th Annu. IEEE Symp. Found. Comput. Sci.*, 2003

Data-Privacy

- **Aim** preserve the privacy of local objective function $f_i(x)$

  $\Longrightarrow$ transformed to the problem of preserving the privacy of local initial vector p_i^0

- **Performance Metric**

  (α, β)-data-privacy[40] - measure of estimation accuracy and disclosure probability

$$\Pr\left\{\|\hat{p}_i - p_i^0\|_1 \leq \alpha\right\} \leq \beta.$$

  β is an upper bound for the probability that a rather accurate estimation of p_i^0 is obtained.

- **Notations**

  $K_2 - K_1$: number of iterations where subtractions of noises are intermittently performed

  $f_{\theta_i(k)}(y)$: probability density function of the k-th element of the added noise θ_i

[40] J. He *et al.*, "Preserving data-privacy with added noises: Optimal estimation and privacy analysis," *IEEE Trans. Inf. Theory*, 2018.

Data-Privacy

Theorem 3

D-CPOA achieves (α, β)-data-privacy for p_i^0, where

$$\alpha = \sum_{k=1}^{m_i+1} \alpha_k, \quad \alpha_k \geq 0, \ \forall k = 1, \ldots, m_i + 1,$$

$$\beta = \prod_{k=1}^{m_i+1} \Big[\big(1 - p^{K_2 - K_1}\big) h_i(\alpha_k) + p^{K_2 - K_1} \Big], \quad h_i(\alpha_k) = p \max_{\nu \in \Theta} \int_{\nu - \alpha_k}^{\nu + \alpha_k} f_{\theta_i(k)}(y) \mathrm{d}y + \gamma.$$

- α_k is the estimation accuracy corresponding to each element of $p_i^0 \in \mathbb{R}^{m_i+1}$.
- β is the product of a set of bounds β_k for disclosure probabilities corresponding to elements of p_i^0.
- β_k is derived via the law of total probability based on whether the adversaries own the full knowledge of the in-neighborhood of agent i at each time (this event has a probability $\leq p$).

Comparison with Other Distributed Optimization Algorithms

Table 3 Comparison of D-CPOA and Other Distributed Optimization Algorithms

Algorithms	**Nonconvex Objectives**	**Networks**		**Privacy Guarantee**	**Exact Convergence**
		Time-varying	Digraph		
Push-DIGing[41]		✓	✓		✓
SONATA[42]	✓	✓	✓		✓
Algorithm[43]		Cloud-based		DP*	trade-off**
Algorithm[44]		✓	✓	DP	trade-off
D-CPOA	✓	✓	✓	data-privacy	✓

* "DP" stands for "differential privacy". ** There is a trade-off between accuracy and privacy.

[41] A. Nedic *et al.*, *SIAM J. Optim.*, 2017, [42] G. Scutari *et al.*, *Math. Program.*, 2019, [43] S. Han *et al.*, *IEEE Trans. Autom. Control*, 2017, [44] E. Nozari *et al.*, *IEEE Trans. Control Netw. Syst.*, 2018.

Current Efforts

We aim to solve problems with $d \in \mathbb{N}$, i.e., local objective functions are multivariate.

- **Inspirations**

  There is a recent method that achieves global optimization of smooth functions based on zeroth-order information[45].

- **Settings**

  nonconvex and m-times continuously differentiable objective $f(x)$, a bound region Ω that optimal solutions are known to fall into, function values $f(x_j)$ at n randomly sampled points x_j in $\Omega (j = 1, \ldots, n)$, and other technical assumptions

- **Results**

  the absolute difference between the obtained solution and the global optimum is of the order of $\mathcal{O}\left(n^{-m/d+1/2+3/d}\right) \Longrightarrow$ close to the lower bound $\mathcal{O}\left(n^{-m/d}\right)$[46]

[45] A. Rudi *et al.*, "Finding global minima via kernel approximations," *arXiv preprint arXiv:2012.11978*, 2020, [46] E. Novak, "Deterministic and stochastic error bounds in numerical analysis," , vol. 1349, 2006.

Current Efforts

▶ **Key Ideas of this Recent Method**

1. perform the following transformation

$$\min_{x\in\Omega} f(x) \implies \max_{c\in\mathbb{R}} c \quad \text{s.t. } \forall x \in \Omega,\ f(x) \geq c, \tag{1}$$

The equivalence follows from the fact that (x^*, t^*) is optimal for RHS *iff* x^* is optimal for LHS and $t^* = f(x^*)$.

2. add the penalization, and approximate the infinite set of inequality constraints in RHS of (1) by a set of equality constraints in

$$\max_{c\in\mathbb{R},\ B\in\mathbb{S}^n_+} c - \lambda \operatorname{Tr}(B) \qquad \text{s.t. } \forall j \in \{1,\ldots,n\},\ f(x_j) - c = \Phi_j^\top B \Phi_j,$$

based on sampled values $f(x_1), \ldots, f(x_n)$. Note: Φ_j is the column of a matrix obtained based on kernels

3. solve the above semidefinite programming reformulation (e.g., use the interior-point method)

Current Efforts

▶ Our Plans

We wish to borrow ideas from this recent work and design a distributed nonconvex optimization algorithm that

- finds the approximate globally optimal solutions
- has a consensus-based iteration rule free from function evaluations

  $\rightarrow$ this goal is achievable since function values are needed a priori and are not iteratively evaluated

  $\rightarrow$ then cumulative costs of queries do not grow with the iterations and are effectively reduced

Current Efforts

▶ Ongoing Efforts

- design distributed strategies based on the idea of this recent work. There are some issues to be addressed.
  - If we choose the same sampled points for all agents, then local information is coupled in the equality constraints
    $$\sum_{i=1}^{N} f_i(x_j) - c = \Phi_j^\top B \Phi_j, \quad j = 1, \ldots, n.$$
    We need to find strategies to cope with such coupled constraints.
  - To effectively solve SDPs via the interior-point method, we need to calculate the Hessians of logarithmic barrier functions. Distributed strategies may require the exchange and updates of these Hessians, which can add to the burden of communication.
- investigate in the distributed setting, whether the dependence of the error ϵ on n, m and d remains the same compared to that in the centralized setting.

Future Work

Future work includes

- Apply the proposed proxy-based algorithm to deal with practical problems arising in distributed learning, coverage control, and other applications relating to multi-agent systems.
- Leverage the idea of introducing polynomial approximation to deal with problems with multivariate noncovex objectives.

If You Are Interested

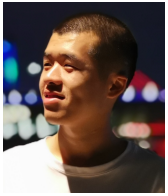
- You are warmly welcomed to visit the following websites for the paper and slides.

  Paper: `https://arxiv.org/pdf/2008.00252.pdf` `https://arxiv.org/pdf/2101.06127.pdf`

  Slides: `https://iwin-fins.com/wp-content/uploads/2020/04/slides.pdf`

Thank you for listening!

Acknowledgment



Zhiyu He, SJTU

Yushan Li, SJTU

Florian Dörfler, ETH

IWIN-FINS, SJTU

IWIN, SJTU