CPCA: A Chebyshev Proxy and Consensus based Algorithm for General Distributed Optimization

Zhiyu He, Jianping He*, Cailian Chen and Xinping Guan

Shanghai Jiao Tong University

May 2020

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

Distributed Optimization

$$\min_{x} \frac{1}{N}\sum_{i=1}^{N} f_i(x) \quad \text{s.t. } x \in \bigcap_{i=1}^{N} X_i$$

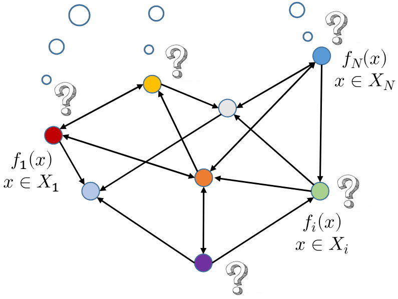


Figure 1 An illustration of distributed optimization

▶ What is distributed optimization?

Distributed optimization enables agents in networked systems to collaboratively solve the problem of optimizing the average of local objective functions.

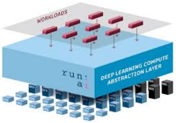
▶ Why not centralized optimization?

- possible lack of central authority
- efficiency, privacy-preserving, robustness and scalability issues[1]

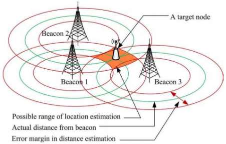
[1] A. Nedić *et al.*, "Distributed optimization for control," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 1, pp. 77–103, 2018

Distributed Optimization: Application Scenarios

- Distributed optimization empowers networked multi-agent systems

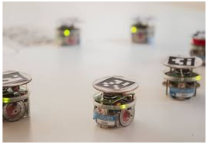


(a) Distributed Learning[2]



(b) Distributed Localization in Sensor Networks[3]

(c) Distributed Coordination in Smart Grid[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.

$$\min \sum_{i=1}^{N} f_i(P_i),$$

$$\text{s.t.} \sum_{i=1}^{N} P_i = P_d,$$

$$\text{s.t.} \ \underline{P_i} \leq P_i \leq \overline{P_i},$$

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

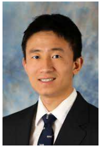
Developments of Distributed Optimization

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 (DGD, EXTRA, Push-DIGing)

non-convex optimization algorithms (SONATA, ZONE)

---

[6] A. Nedic *et al.*, *IEEE Trans. Autom. Control*, 2009, [7] W. Shi *et al.*, *SIAM J. Optim.*, 2015, [8] A. Nedic *et al.*, *SIAM J. Optim.*, 2017, [9] G. Scutari *et al.*, *Math. Program.*, 2019, [10] 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[11], Fast-DGD[12], EXTRA[13], DIGing[14], Acc-DNGD[15], ZONE[16], SONATA[17]...

    feature: combine (sub)gradient descent with consensus, so as to drive local estimates to converge in the primal domain
  - **Dual-based Methods:** Dual Averaging[18], D-ADMM[19], DCS[20], MSDA[21], MSPD[22], ...

    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

    cons: hard to be extended to deal with time-varying or directed graphs

[11] A. Nedic *et al.*, *IEEE Trans. Autom. Control*, 2009, [12] D. Jakovetić *et al.*, *IEEE Trans. Autom. Control*, 2014, [13] W. Shi *et al.*, *SIAM J. Optim.*, 2015, [14] A. Nedic *et al.*, *SIAM J. Optim.*, 2017, [15] G. Qu *et al.*, *IEEE Trans. Autom. Control*, 2019, [16] D. Hajinezhad *et al.*, *IEEE Trans. Autom. Control*, 2019, [17] G. Scutari *et al.*, *Math. Program.*, 2019, [18] J. C. Duchi *et al.*, *IEEE Trans. Autom. Control*, 2011, [19] W. Shi *et al.*, *IEEE Trans. Signal Process.*, 2014, [20] G. Lan *et al.*, *Math. Program.*, 2017, [21] K. Scaman *et al.*, in *Proc. Int. Conf. Mach. Learn.*, 2017, [22] K. Scaman *et al.*, in *Adv Neural Inf Process Syst*, 2018.

Motivations

Two notable unresolved issues within the existing works

- growing load of oracle queries with respect to the iterations

  ▷ results from the requirements of evaluations of gradients or values of local objectives at one or several points within every iteration

  $\Longrightarrow$ the selection of step-sizes also influences convergence speeds, which complicates the analysis

- hardness of achieving iterative convergence to the global optimal points

  ▷ results from the nonconvex nature of the general objectives

Is it possible to overcome these issues?

Contributions

Main contributions of this work

- We propose a novel algorithm, CPCA, leveraging polynomial approximation and consensus
- 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

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

Assumptions

1. $\mathcal{G}$ is a static, connected and undirected graph.
2. Every $f_i(x)$ is Lipschitz continuous on X_i.
3. All X_i are closed, bounded and convex sets.

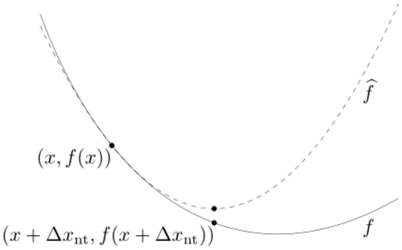
Note

- The assumptions we made on graphs and objectives are common within the literatures.
- The extension to time-varying directed graphs is feasible, and is presented in our recent work.

Key Ideas

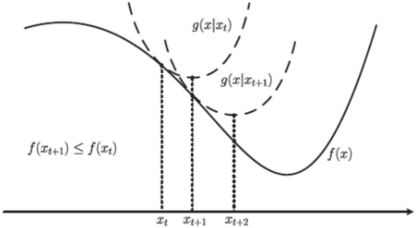
- **Inspirations**

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 3 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

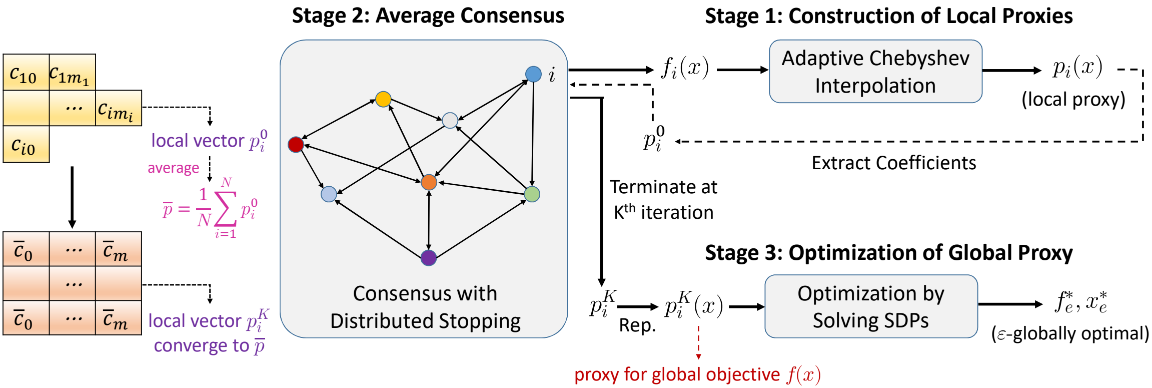


Figure 4 The architecture of CPCA

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[23] to obtain $p_i(x)$.
  3. Maintain p_i^0 storing the Chebyshev coefficients of $p_i(x)$'s derivative through certain recurrence formula.

[23] 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

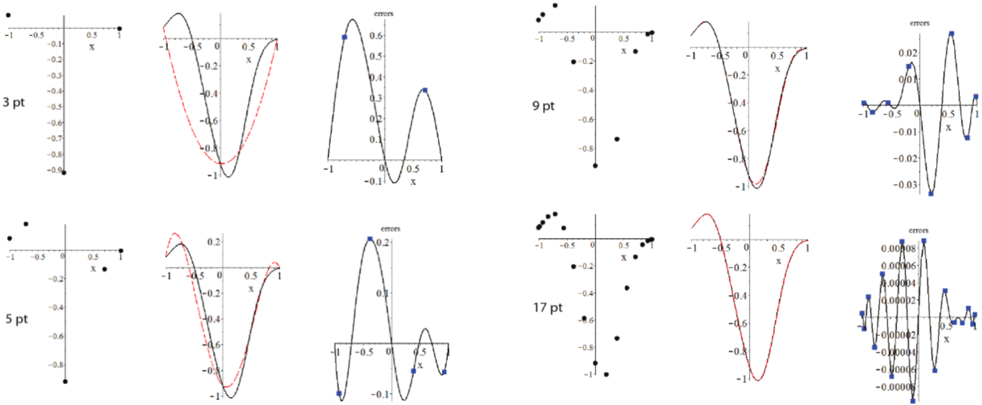


Figure 5 An illustration of Adaptive Chebyshev Interpolation

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

$$f_1(x) = \frac{1}{2}e^{0.1x} + \frac{1}{2}e^{-0.1x}$$

↓ Adaptive Interpolation

$$p_1(x) = \sum_{j=0}^{4} c_j T_j\left(\frac{x}{3}\right)$$

↓ recurrence formula

$$p_1^0 = [1.0226, 0, 0.0303, 0, 1.1301\times 10^{-4}]^T$$

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

◦ Case II

$$f_2(x) = \frac{1}{4}x^4 + \frac{2}{3}x^3 - \frac{1}{2}x^2 - 2x$$

↓ Adaptive Interpolation

$$p_2(x) = \sum_{j=0}^{4} c_j T_j\left(\frac{x}{3}\right)$$

↓ recurrence formula

$$p_2^0 = [5.3437, 7, 17.25, 9, 6.75]^T$$

(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[24] for certain rounds.

[24] 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**

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

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[25], 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 $\max_{i,j \in \mathcal{V}} \left\|p_i^0 - p_j^0\right\|_\infty$.

[25] 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[26] into consensus iterations.

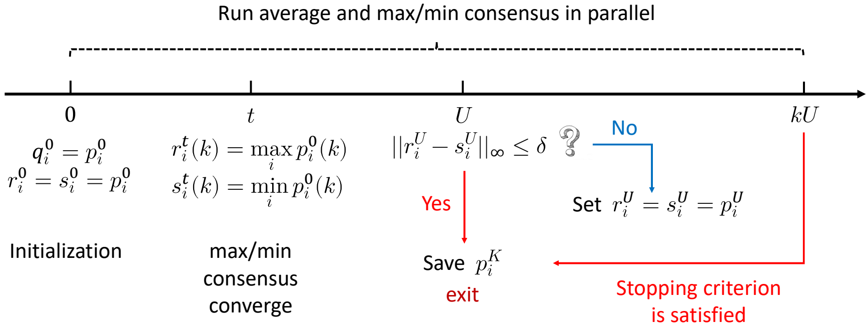


Figure 6 An illustration of average consensus with distributed stopping

[26] 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.

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

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**

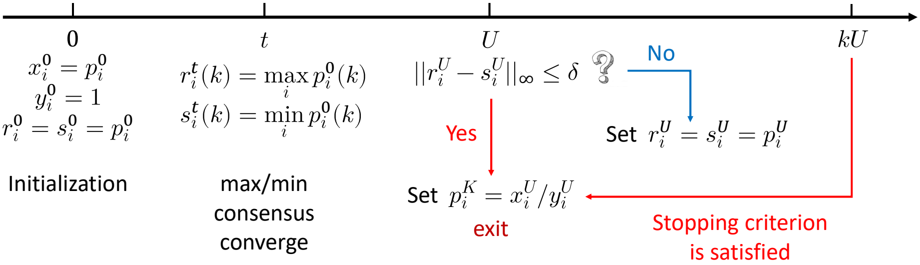


Figure 7 An illustration of push-sum consensus with distributed stopping

Optimize Polynomial Proxy based on Stationary Points

- **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.
    - ▷ 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[27].

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

[27] 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

$$
\begin{aligned}
\max_{t,Q,Q'} \quad & t \\
\text{s.t.} \quad & c_0 = t + \sum_{u,v \text{ even}} (-1)^{\frac{u+v}{2}} \left(bQ'_{uv} - aQ_{uv}\right) \\
& c_i = \frac{1}{2} \sum_{(u,v)\in\mathcal{A}} \left(bQ'_{uv} - aQ_{uv}\right) + \frac{1}{4} \sum_{(u,v)\in\mathcal{B}} \left(Q_{uv} - Q'_{uv}\right), \quad i = 1, \ldots, m \\
& Q, Q' \in \mathbb{S}_+^m,
\end{aligned}
$$

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

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

- CPCA ensures that every agent obtains ϵ-optimal solutions for any arbitrarily small given tolerance ϵ.

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.

- ϵ is used to set ϵ_1 and ϵ_2 (both equal to $\epsilon/2$) to regulate the stages of initialization and iteration, so as to guarantee the meet of the precision requirement.

Complexities of CPCA

Table 1 Complexities of CPCA

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

N: the size of the network $\quad$ 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 [28]	I	$\mathcal{O}\left(\frac{d}{\epsilon}\right)$	/	$\mathcal{O}\left(\frac{d}{\epsilon}\right)$
SONATA[29]	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.

[28] Y. Tang *et al.*, *arXiv e-prints*, arXiv:1908.11444, 2019, [28] 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)[30] (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

[30] 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

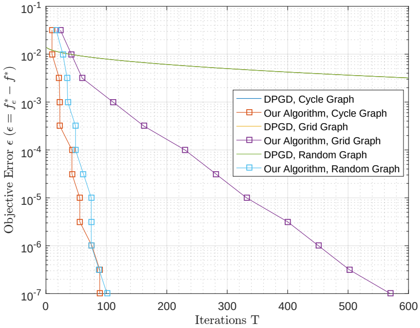
$$f_i(x) = a_i x^4 + b_i x^3 + c_i x^2 + d_i x + e_i, \quad x \in X_i = [-3, 3],$$

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.

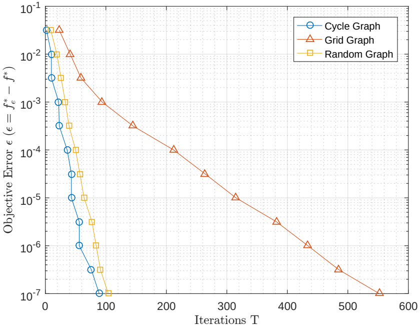
Note: Case I: convex objectives **Case II**: non-convex objectives

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 8 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[31]

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

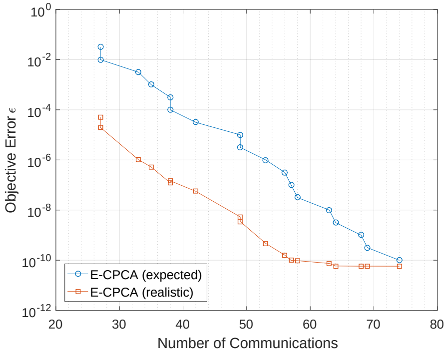
The *nonconvex but Lipschitz* objectives we choose are

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

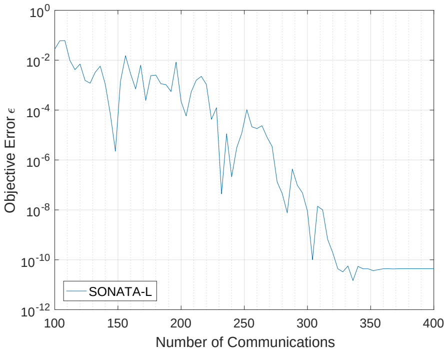
[31] G. Scutari *et al.*, "Distributed nonconvex constrained optimization over time-varying digraphs," *Math. Program.*, vol. 176, no. 1-2, pp. 497–544, 2019.

Numerical Experiments

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



(a) E-CPCA



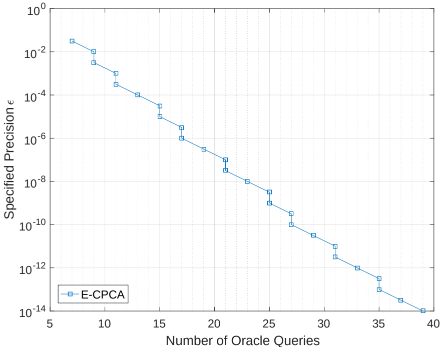
(b) SONATA-L

Figure 9 Comparison of both algorithms regarding inter-agent communications

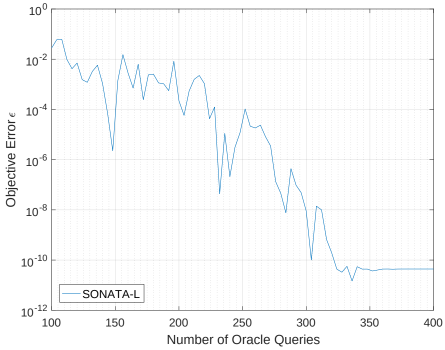
Note: E-CPCA is more communication-efficient due to its integrated rapidly convergent consensus protocols.

Numerical Experiments

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



(a) E-CPCA



(b) SONATA-L

Figure 10 Comparison of both algorithms regarding inter-agent communications

Note: Nor the increase of N or worsening of network's connectivity will change the curve in Fig. 10a.

Summary

We present a Chebyshev Proxy and Consensus-based Algorithm (CPCA) to solve a class of distributed nonconvex optimization problems

- with Lipschitz univariate objectives and convex local constraint sets,
- over static undirected graphs.

Features of CPCA

- able to address the problem with nonconvex objectives and obtain ϵ globally optimal solutions
  - ▷ originates from the idea of optimizing the polynomial proxy instead
- free from evaluations of gradients or functions within the iterations, and is computationally efficient
  - ▷ results from the scheme of simply employing average consensus to update coefficient vectors

Summary

We also discuss some possible improvements of CPCA

- incorporate distributed stopping mechanism for consensus

  $\Longrightarrow$ make CPCA communication-efficient

- transform the optimization of polynomial proxies to SDPs

  $\Longrightarrow$ make all the errors theoretically controllable

- employ push-sum consensus when applied to time-varying directed graphs

  $\Longrightarrow$ the formulation and analysis of Extended-CPCA (E-CPCA) is presented in our recent work

Future Works

Future works include

- 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.

Thank you for listening!