文章编号:2095-6134(2022)01-0127-07

# 分布式有限时间重球法*

曲志海,陆捷†

(上海科技大学信息科学与技术学院, 上海 201210; 中国科学院上海微系统与信息技术研究所, 上海 200050;
中国科学院大学, 北京 100049)

(2020 年 1 月 6 日收稿; 2020 年 2 月 26 日收修改稿)

**Qu Z H, Lu J. Distributed finite-time-consensus-based heavy-ball algorithm[J]. Journal of University of Chinese Academy of Sciences, 2022, 39(1): 127-133. DOI: 10.7523/j.ucas.2020.0009.**

**摘 要** 结合有限时间共识算法及一阶加速算法重球法提出分布式有限时间重球法。本算法的优点为可以保证所有节点在每个周期都达到共识,同时达到与集中式重球法相同阶数的收敛速率。通过数值仿真将该算法与其他分布式优化算法应用于机器学习问题上,展现了该算法的优良性能。

**关键词** 分布式优化;算法设计;有限时间共识算法;重球法

**中图分类号**:TN99 **文献标志码**:A **DOI**:10.7523/j.ucas.2020.0009

# Distributed finite-time-consensus-based heavy-ball algorithm

QU Zhihai, LU Jie

(*School of Information Science and Technology, ShanghaiTech University, Shanghai* 201210, *China*;
*Shanghai Institute of Microsystem and Information Technology, Chinese Academy of Sciences, Shanghai* 200050, *China*;
*University of Chinese Academy of Sciences, Beijing* 100049, *China*)

**Abstract** Based on the finite-time-consensus algorithm and the heavy-ball algorithm which is a first order accelerate algorithm, a distributed optimization algorithm is proposed. The algorithm can achieve consensus after every periodic updates. The non-ergodic convergence rate is at the same order of the centralized heavy-ball algorithm. In addition, the numerical examples compare our algorithm with other state-of-art distributed optimization algorithms on machine learning problems and show the competitive performance.

**Keywords** distributed optimization; algorithm design; finite-time-consensus algorithm; heavy-ball algorithm

随着多智体系统的发展,分布式优化算法受到广泛的重视并应用于诸多领域,如资源调度[1]、分布式传感器系统参数估计[2]、协同控制[3]及对分布式增强学习[4]等问题的相关研究。现存的分布式算法大多以各个节点目标函数梯度作为下降方向,以渐进收敛的方式达成最优共识状态,如分布式次梯度算法[5]、EXTRA[6]、DIGing[7]等。此类算法需要足够多迭代算法才能收敛到共识状态。另一类算法则以有限次迭代达成共识的 FTC 算法[8]为基础,在一个迭代周期内

* 国家自然科学基金(61603254)资助

† 通信作者,E-mail: lujie@shanghaitech.edu.cn

所有节点达到共识,如 FADO[9]、FTC-NM[10]。其中 FADO 算法结合 FTC 与次梯度算法,因次梯度算法仅使用了函数基本的一阶信息,算法不能得到理想的收敛速率。FTC-NM 算法利用二阶信息达到局部二次收敛速率。虽然收敛速率理想但需要传输二阶导数矩阵不适用于通信受限的场景中。考虑通信量与收敛速率的折中,本文中提出一种 FTC 算法与重球法相结合的分布式一阶算法。

本文贡献如下:提出基于 FTC 的分布式一阶加速优化算法。给出收敛性分析及算法参数的选取范围。证明本文算法与集中式重球法算法相同的收敛阶数。最后通过在随机生成的大规模网络上求解机器学习问题并与多种算法进行对比以体现算法的优良性能。

**符号说明** 本文中使用 $\mathcal{G}=(\mathcal{V},\varepsilon)$,表示无向网络,其中 $\mathcal{V}:=\{1,\cdots,N\}$ 表示网络中节点的集合,$\varepsilon=\{(i,j)\mid i,j \text{ 直接相连}\}$。使用 $\mathcal{N}_i:=\{j\mid(i,j)\in\varepsilon\}$ 表示节点 $i$ 的邻居组成的集合。向量二范数用 $\|\cdot\|$ 表示,向量内积通过 $\langle\cdot,\cdot\rangle$ 表示。使用 $\nabla f(\boldsymbol{x})$ 表示函数 $f$ 在点 $\boldsymbol{x}$ 处的导数。用 $w_{ij}$ 表示矩阵 $\boldsymbol{W}$ 中 $i$ 行 $j$ 列的元素。使用 $\deg(i)$ 表示节点 $i$ 的度,也就是 $\mathcal{N}_i$ 集合中所包含的元素个数。以 e 为底的指数函数表示为 $\exp(\boldsymbol{x})$。用 $\boldsymbol{e}_i$ 表示第 $i$ 个元素为 1,其他元素为 0 的向量。使用 $\mathbb{R}^n$ 表示一个 $n$ 维实数向量,$\mathbb{R}^{n\times n}$ 表示 $n\times n$ 的实数矩阵。使用 $\mathrm{dom}(f)$ 表示函数 $f$ 的定义域。用粗体 $\boldsymbol{1},\boldsymbol{0}$ 表示对应维度全 1,全 0 的向量。$\boldsymbol{x}^{\mathrm{T}}$ 上标表示向量 $\boldsymbol{x}$ 的转置。

## 1 问题描述

假设网络中共有 $N$ 个节点,每个节点有自己的目标函数 $f_i(\boldsymbol{x})$。在分布式协同优化中考虑如下问题

$$\min_{\boldsymbol{x}\in\mathbb{R}^d} F(\boldsymbol{x}):=\frac{1}{N}\sum_{i=1}^{N} f_i(\boldsymbol{x}), \tag{1}$$

通过变量复制,问题(1)可以等价为如下问题

$$\begin{aligned}\min_{\boldsymbol{x}\in\mathbb{R}^{Nd}}\quad & \frac{1}{N}\sum_{i=1}^{N} f_i(\boldsymbol{x}_i),\\ \text{s.t.}\quad & \boldsymbol{x}_1=\cdots=\boldsymbol{x}_N.\end{aligned} \tag{2}$$

此处 $\boldsymbol{x}$ 为 $\boldsymbol{x}_1,\cdots,\boldsymbol{x}_N$ 堆叠成的向量。等价问题(2)在分布式系统中被广泛应用,由于通信受限,并非所有节点都可以实时保持变量值相同,所以考虑每个节点 $i$ 拥有自己的变量 $\boldsymbol{x}_i$。为了分析算法的收敛速率引入以下描述函数性质的假设。

**假设 1**($L_i$-光滑函数)假设函数 $f_i(\boldsymbol{x})$ 为 $L_i$-光滑,则函数 $f_i(\boldsymbol{x})$ 满足如下性质,即对任意 $\boldsymbol{x},\boldsymbol{y}\in\mathrm{dom}(f_i)$,如下关系成立

$$f_i(\boldsymbol{y})\leqslant f_i(\boldsymbol{x})+\langle\nabla f_i(\boldsymbol{x}),\boldsymbol{y}-\boldsymbol{x}\rangle+\frac{L_i}{2}\|\boldsymbol{y}-\boldsymbol{x}\|^2,$$

$L_i$-光滑的定义等价于函数 $f_i$ 的一阶导数 $L_i$-Lipschitz 连续,是对函数变化程度的一种约束。

**假设 2**($\nu$-限制性强凸函数)假设函数 $F(\boldsymbol{x})$ 为 $\nu$-限制性强凸,则函数 $F(\boldsymbol{x})$ 满足如下性质,对 $\forall\boldsymbol{x}\in\mathrm{dom}(F)$,$\boldsymbol{x}^*:=\mathrm{argmin}F(\boldsymbol{x})$,$\exists\nu>0$ 如下成立

$$F(\boldsymbol{x})-\min F\geqslant\nu\|\boldsymbol{x}-\boldsymbol{x}^*\|^2,$$

$\nu$-限制性强凸是比强凸更弱的条件。

**假设 3**(强制函数)函数 $f_i(\boldsymbol{x})$ 为强制函数,则当 $\|\boldsymbol{x}\|\to+\infty$ 时,$f_i(\boldsymbol{x})\to+\infty$。

对目标函数的假设会用于收敛性分析,为保证各个节点达成共识,需要对网络进行如下常见假设

**假设 4**(网络连通)网络中的每一个节点至少存在一条到其他任意节点的路径。

只有在连通的网络中各个节点才能通过通信获取网络中的相关信息并进行协同。

**假设 5** 加权矩阵 $\boldsymbol{W}$ 为行随机矩阵而且当 $(i,j)\in\varepsilon$ 时 $w_{ij}>0$,其他情况下 $w_{ij}=0$,且 $\boldsymbol{W}\boldsymbol{1}=\boldsymbol{1}$。

常见的 $\boldsymbol{W}$ 如 $w_{ij}=\dfrac{1}{\max\{\deg(i),\deg(j)\}}$,$(i,j)\in\varepsilon$ 为局部度加权矩阵。文献[11]将加权矩阵对分布式共识算法的影响建模为优化问题,给出了可以达到最优收敛速度的加权矩阵及多种通用的加权矩阵。

## 2 FTC 算法

不失一般性,本节考虑 $\boldsymbol{x}_i^t\in\mathbb{R}$。因为下述操作均为线性操作所以容易扩展到高维 $\mathbb{R}^d$。在满足假设 4 的情况下,经典的共识迭代算法为

$$\boldsymbol{x}_i^{t+1}=\sum_{j\in\mathcal{N}_i} w_{ij}\boldsymbol{x}_j^t,\ \forall i\in\mathcal{V}, \tag{3}$$

其中 $\boldsymbol{W}$ 是满足假设 5 的加权矩阵。当满足假设 4,假设 5 时,通过式(3)所有节点的状态 $\boldsymbol{x}_i^t$ 将渐进收敛于

$$\lim_{t\to\infty}\boldsymbol{x}^t=\boldsymbol{\Phi}\boldsymbol{x}^0,$$

此处 $\boldsymbol{\Phi}:=\boldsymbol{1}\boldsymbol{\pi}^{\mathrm{T}}$,$\boldsymbol{\pi}\in\mathbb{R}^n$ 是加权矩阵 $\boldsymbol{W}$ 特征值为 $\boldsymbol{1}$ 对应的归一化左特征向量。通过 Perron-

Frobenius 定理[12]可知存在 $\lim\limits_{t\to\infty}\boldsymbol{W}^t=\boldsymbol{\Phi}$。这意味着 $\lim\limits_{t\to\infty}\boldsymbol{x}_i^t=\boldsymbol{\pi}^{\mathrm{T}}\boldsymbol{x}^0=:\bar{\boldsymbol{x}}^0$。

Sundaram 和 Hadjicostis[8]通过单边 $z$ 变换及 Jordan 分解证明 FTC 算法可以通过有限次迭代(3)达到 $\bar{\boldsymbol{x}}^0$ 并定义了相对节点的最小多项式。

**定义 1**(节点 $i$ 的最小多项式[8]) $\boldsymbol{q}_i(t)$ 是多项式次数最低且满足 $\boldsymbol{e}_i^{\mathrm{T}}\boldsymbol{q}_i(\boldsymbol{W})=\boldsymbol{0}^{\mathrm{T}}$ 的首一多项式。其中 $\boldsymbol{W}$ 为满足假设 5 的加权矩阵。

节点 $i$ 只需要 $D_i\leqslant N$ 次加权迭代式(3)即可计算 $\bar{\boldsymbol{x}}(0)$。存在 $\boldsymbol{\alpha}^{(i)}:=[\alpha_0^{(i)},\alpha_1^{(i)},\cdots,\alpha_{D_i}^{(i)}]\in\mathbb{R}^{D_i+1},D_i\geqslant 0$[8],

$$q_i(\xi)=(\xi-1)\sum_{l=0}^{D_i}\alpha_l^{(i)}\xi^l,\ \alpha_{D_i}^{(i)}=1. \tag{4}$$

下面介绍 $\boldsymbol{\alpha}^{(i)}$ 的一种分布式计算方法[13]。以勒贝格测度下为 0 的集合中的起始状态 $\boldsymbol{x}^0$ 进行式(3)迭代(最多 $N$ 次),组成如下 Hankel 矩阵

$$\boldsymbol{H}(i,k):=\begin{bmatrix} y_i^1 & y_i^2 & \cdots & y_i^{k+1}\\ y_i^2 & y_i^3 & \cdots & y_i^{k+2}\\ \vdots & \vdots & & \vdots\\ y_i^{k+1} & y_i^{k+2} & \cdots & y_i^{2k+1}\end{bmatrix},$$

其中 $\boldsymbol{y}_i^{k+1}=\boldsymbol{x}_i^{k+1}-\boldsymbol{x}_i^k$。$\boldsymbol{\alpha}^{(i)}$ 在第一个缺秩 Hankel 矩阵 $\boldsymbol{H}(i,k)$ 的零空间中。而判断 Hankel 矩阵是否满秩用到 SVD,其复杂度为 $\mathcal{O}(k^3)$,所以计算 $\boldsymbol{\alpha}^{(i)}$ 需要进行 $\sum_{k=1}^{D_i}\mathcal{O}(k^3)=\mathcal{O}(D_i^4)$。同样 $\boldsymbol{\alpha}^{(i)}$ 有多种求解方法[8,13],均与相对应的网络规模相关。为简化 $\boldsymbol{\alpha}^{(i)}$ 的计算可使用网络分类方法[10]。

**定理 1**(通过 $q_i$ 计算共识[8])当假设 4,假设 5 满足且 $\boldsymbol{W}^{\mathrm{T}}$ 为行随机矩阵时,则共识可通过前 $D_i$ 次式(3)迭代得到,即

$$\sum_{l=0}^{D_i}\hat{\alpha}_l^{(i)}\boldsymbol{x}_i(l)=\frac{1}{N}\sum_{i=1}^{N}\boldsymbol{x}_i(0),\hat{\alpha}_l^{(i)}=\frac{\alpha_l^{(i)}}{\sum\limits_{l=0}^{D_i}\alpha_l^{(i)}}.$$

公式(4)中给出了 $\alpha_l^{(i)}$。

定理 1 给出了计算共识的方法,此方法是文中的信息融合的关键所在。

## 3　FTCHB 算法设计

结合 FTC 算法与重球法本章节提出算法 1 (finite-time-consensus-based heavy-ball method, FTCHB)。为避免二阶算法的巨大通信与计算量及经典梯度算法缓慢的收敛速率,本文决定结合使用历史信息的重球法以获取下降方向以求解问题(1)。重球法属于一阶算法,仅需要对梯度进行计算,避免了二阶算法计算量与通信量的缺陷。此外重球法在病态条件数的情况下可以保持良好的收敛速率。使用 FTC 算法辅助以达到共识。

**算法 1:FTCHB**

**初始化:**

1 对每个节点 $i\in\mathcal{V}$,随机生成起始状态 $\boldsymbol{x}_i(0)$。选择合适的步长参数 $\gamma_k$ 与动量参数 $\beta_k$,每个节点通过第 3 节的方法计算 $\hat{\boldsymbol{\alpha}}^{(i)}$。

2 **for** $t=1,2,\cdots,N-1$ **do**

3　　$\boldsymbol{x}_i(t)=\sum_{j\in\mathcal{N}_i}w_{ij}\boldsymbol{x}_j(t-1)$.

4 **end for**

5 $\boldsymbol{x}_c^{-1}(i)=\boldsymbol{x}_c^0(i)=\sum_{l=0}^{D_i}\hat{\alpha}_l^{(i)}\boldsymbol{x}_i(l)$.

**主要迭代部分:**

6 **for** $k=0,1,2,\cdots$ **do**

7　　$\boldsymbol{x}_c^{k+1}(i)=\boldsymbol{x}_c^k(i)-\gamma_k\nabla f_i(\boldsymbol{x}_c^k(i))+\beta_k(\boldsymbol{x}_c^k(i)-\boldsymbol{x}_c^{k-1}(i))$.

8　　$\boldsymbol{x}_i(0)=\boldsymbol{x}_c^{k+1}(i)$.

9　　**for** $t=1,2,\cdots,N-1$ **do**

10　　$\boldsymbol{x}_i(t)=\sum_{j\in\mathcal{N}_i}w_{ij}\boldsymbol{x}_j(t-1)$.

11　　**end for**

12　　$\boldsymbol{x}_c^{k+1}(i)=\sum_{l=0}^{D_i}\hat{\alpha}_l^{(i)}\boldsymbol{x}_i(l)$.

13 **end for**

算法初始化阶段各个节点分别计算自己 FTC 算法的相关参数 $\boldsymbol{\alpha}^{(i)}$(详见第 2 节)。算法中使用 $\boldsymbol{x}_i(t)$ 表示节点 $i$ 在内循环中第 $t$ 步的状态,使用 $\boldsymbol{x}_c^k(i)$ 表示节点 $i$ 在第 $k$ 次外循环迭代更新中状态。步骤 2~步骤 5 可以理解为通过 FTC 算得初始状态的共识即 $\boldsymbol{x}_c^0(i)=\frac{1}{N}\sum_{i=1}^{N}\boldsymbol{x}_i(0)$。在主要迭代部分中,每个内部循环只需要各个节点与其邻居节点通信各自的状态。步骤 7 使得每个节点通过重球法进行局部的更新,梯度信息 $\nabla f_i(\boldsymbol{x}_c^k(i))$ 和动量信息 $\boldsymbol{x}_c^k(i)-\boldsymbol{x}_c^{k-1}(i)$ 通过后面步骤 9~步骤 11 进行融合,步骤 12 利用定理 1 达成当前迭代的共识。

注意算法参数 $\gamma_k,\beta_k$ 选取方面各个节点使用相同的值,但参数值可通过 FTC 算法进行完全分布式计算。为简洁,算法表示仅通过表达式描述参数传递的过程。每次用到 $\boldsymbol{x}_j(t-1),j\in\mathcal{N}_i$ 的时候都需要节点间进行通信。

## 4　收敛性分析

本节给出 FTCHB 算法的收敛性分析。注意到重球法中的更新方向单调下降方向,所以无法采用文献[10]中的方法进行分析。为分析收敛

速率我们尝试通过找到一个 Lyapunov 函数以辅助收敛性证明。首先给出下降引理,并根据下降引理构造 Lyapunov 函数。此后在 $L$ -光滑与凸假设下得到一个目标函数 $F(\boldsymbol{x})$ 有非各态历经的 $\mathcal{O}(1/k)$ 收敛速率以及在 $\nu$ -限制性强凸情况下目标函数 $F(\boldsymbol{x})$ 线性收敛速率。

注意到算法 1 中步骤 3~步骤 7 以及步骤 9~步骤 12 中更新是在每个内循环中使用 FTC 算法进行共识计算。这意味着对每次迭代 $k$, 有 $\boldsymbol{x}_c^k(i) = \boldsymbol{x}_c^k(j), \forall i,j \in \mathcal{V}$。因此定义 $\boldsymbol{x}_c^k := \boldsymbol{x}_c^k(i)$ 以简化符号。

上面的等价形式在证明中起着重要作用。算法 1 利用了变量复制后问题(2)的形式。下面通过 FTC 算法给出算法 1 的集中式等价解释。集中式理解方式允许我们参考重球法中的收敛性分析方法[14-15]。下面利用算法 1 主要迭代的等价形式结合文献[15]中的集中式处理分析方法获取非各态历经的收敛速率。

**引理 1** 假设 $f_i(\boldsymbol{x})$ 是凸函数且一阶可导,且满足假设 1, $\min F > -\infty$。假设动量系数 $\{\beta_k\}_{k\geqslant 0} \subseteq [0,1)$。通过选择步长

$$\gamma_k = \frac{2(1-\beta_k)e}{L},$$

其中: $L = \sum_{i=1} L_i/N, e \in (0,1)$ 为常数。$\boldsymbol{x}_c^k$ 为算法 1 迭代生成的状态。那么有

$$\left[F(\boldsymbol{x}_c^k) + \frac{\beta_k}{2\gamma_k}\| \boldsymbol{x}_c^k - \boldsymbol{x}_c^{k-1}\|^2\right] - \left[F(\boldsymbol{x}_c^{k+1}) + \frac{\beta_{k+1}}{2\gamma_{k+1}}\| \boldsymbol{x}_c^{k+1} - \boldsymbol{x}_c^k\|^2\right] \geqslant \frac{(1-e)L}{2e}\| \boldsymbol{x}^{k+1} - \boldsymbol{x}^k\|^2 (>0),$$

**证明**: 因为 $F(\boldsymbol{x}) = \sum_{i=1}^N f_i(\boldsymbol{x})/N$, 故有 $\nabla \boldsymbol{F}(\boldsymbol{x}) = \sum_{i=1}^N \nabla f_i(\boldsymbol{x})/N$。通过步骤 7 ~ 步骤 12,相当于将函数 $\gamma_k \nabla \boldsymbol{F}(\boldsymbol{x})$ 及动量 $\beta_k(\boldsymbol{x}_c^k(\mathrm{i}) - \boldsymbol{x}_c^{k-1}(\mathrm{i}))$进行了分布式计算。通过算法 1 的迭代步骤 7~步骤 12 及定理 1 可以等价为如下形式

$$\boldsymbol{x}_c^{k+1} = \boldsymbol{x}_c^k - \gamma_k \nabla \boldsymbol{F}(\boldsymbol{x}_c^k) + \beta_k(\boldsymbol{x}_c^k - \boldsymbol{x}_c^{k-1}), \quad (5)$$

在等价迭代式(5)及假设条件满足的情况下参考文献[15]中引理 1 的证明。

引理 1 说明 $F(\boldsymbol{x}_c^k) + \frac{\beta_k}{2\gamma_k}\| \boldsymbol{x}_c^k - \boldsymbol{x}_c^{k-1}\|^2$ 是非增的,所以 Lyapunov 函数可以设置为 $F(\boldsymbol{x}_c^k) + \frac{\beta_k}{2\gamma_k}\| \boldsymbol{x}_c^k - \boldsymbol{x}_c^{k-1}\|^2$。但是为了得到收敛速率给出如下形式的 Lyapunov 函数定义

$$V(\boldsymbol{x}_c^k) := F(\boldsymbol{x}_c^k) + \delta_k \| \boldsymbol{x}_c^k - \boldsymbol{x}_c^{k-1}\|^2 - F^*,$$

这里 $\delta_k := \frac{\beta_k}{2\gamma_k} + \frac{1}{2}\left(\frac{1-\beta_k}{\gamma} - \frac{L}{2}\right)$。给出的 $\delta_k$ 的形式,第 1 部分 $\frac{\beta_k}{2\gamma_k}$ 与 $F(\boldsymbol{x}_c^k)$ 结合代入引理 1 并注意到第 2 部分中 $\frac{1-\beta_k}{\gamma_k} - \frac{L}{2} \geqslant \frac{(1-e)L}{2e}$, 通过整理可得

$$V(\boldsymbol{x}_c^k) - V(\boldsymbol{x}_c^{k+1}) \geqslant \frac{L(1-e)}{4e} \times (\| \boldsymbol{x}_c^{k-1} - \boldsymbol{x}_c^k\|^2 + \| \boldsymbol{x}_c^k - \boldsymbol{x}_c^{k+1}\|^2) \geqslant 0,$$

这说明定义的 $V(\boldsymbol{x}_c^k)$ 也是非增的函数。下面引理 2 给出 $V(\boldsymbol{x}_c^k)$ 的上界在后面定理证明中起到关键作用。

**引理 2** 假设引理 1 中的条件均满足,用 $\overline{\boldsymbol{x}_c^k}$ 表示 $\boldsymbol{x}_c^k$ 到 $\operatorname{argmin} F(\boldsymbol{x})$ 上的投影,假设 $\overline{\boldsymbol{x}_c^k}$ 存在那么有

$$V(\boldsymbol{x}_x^k)^2 \leqslant \epsilon_k(V(\boldsymbol{x}_c^k) - V(\boldsymbol{x}_c^{k+1})) \times (2\| \boldsymbol{x}_c^k - \boldsymbol{x}_c^k\|^2 + \| \boldsymbol{x}_c^k - \boldsymbol{x}_c^{k-1}\|^2),$$

这里 $\epsilon_k := \frac{4e\delta_k^2}{(1-e)L} + \frac{4e}{(1-e)L\gamma_k^2}$.

**证明**:结合算法迭代更新的等价形式(5)及文献[15]中的引理 2 的证明可得。

由引理 2 可知 $V(\boldsymbol{x}_c^k)^2 < +\infty$ ,即 $\sup_k V(\boldsymbol{x}_c^k) < +\infty$ 有上界。如果假设 $F$ 为强制函数则有 $\sup_k\{\| \boldsymbol{x}_c^k\|\} \leqslant +\infty$。另外为了证明 $\mathcal{O}(1/k)$ 的收敛速率,假设 $\boldsymbol{x}_c^k$ 与最优解的集合最大距离为有限值,即

$$R := \sup_k \| \boldsymbol{x}_c^k - \boldsymbol{x}^*\|^2 < +\infty,$$

此处的 $\boldsymbol{x}^* \in \operatorname{argmin}_{\boldsymbol{x}\in\mathbb{R}^d} F(\boldsymbol{x})$ 是问题(1)的一个最优解。

**定理 2** 引理 1 中的假设成立的情况下,若 $0 < \inf_k \beta_k \leqslant \beta_k \leqslant \beta_0 < 1$ 并且 $F(\boldsymbol{x})$为强制函数,通过算法 1,有

$$F(\boldsymbol{x}_c^k) - F^* \leqslant \frac{4R\sup_k\{\epsilon_k\}}{k},$$

**证明**:结合算法迭代更新的等价形式(5)及文献[15]定理 1 的证明可得。

注意这里的收敛是相对于算法外层的共识变

量 $\boldsymbol{x}_c^k$ (即算法中的 $\boldsymbol{x}_c^k(i)$ )的收敛,内层循环中的变量 $\boldsymbol{x}_i(l)$ 并不能保证定理 2 中的收敛速率。

下面说明加上限制性强凸的假设之后,算法可以获得的非各态历经的线性收敛速率。

**定理 3** 假设定理 2 中的条件均满足并且 $F$ 满足假设 2,那么有

$$F(\boldsymbol{x}_c^k) - F^* \leqslant V(\boldsymbol{x}_c^0)\left(\frac{\ell}{1+\ell}\right)^{k+1},$$

这里 $\ell := \sup_k\left\{\epsilon_k\left(\frac{1}{\delta_k}+\frac{2}{\nu}\right)\right\}$

**证明**:结合算法迭代更新的等价形式(5)及文献[15]定理 2 的证明可得,注意对应文献[15]中的线性收敛速率常数应为 $\omega := \ell/(1+\ell)$。

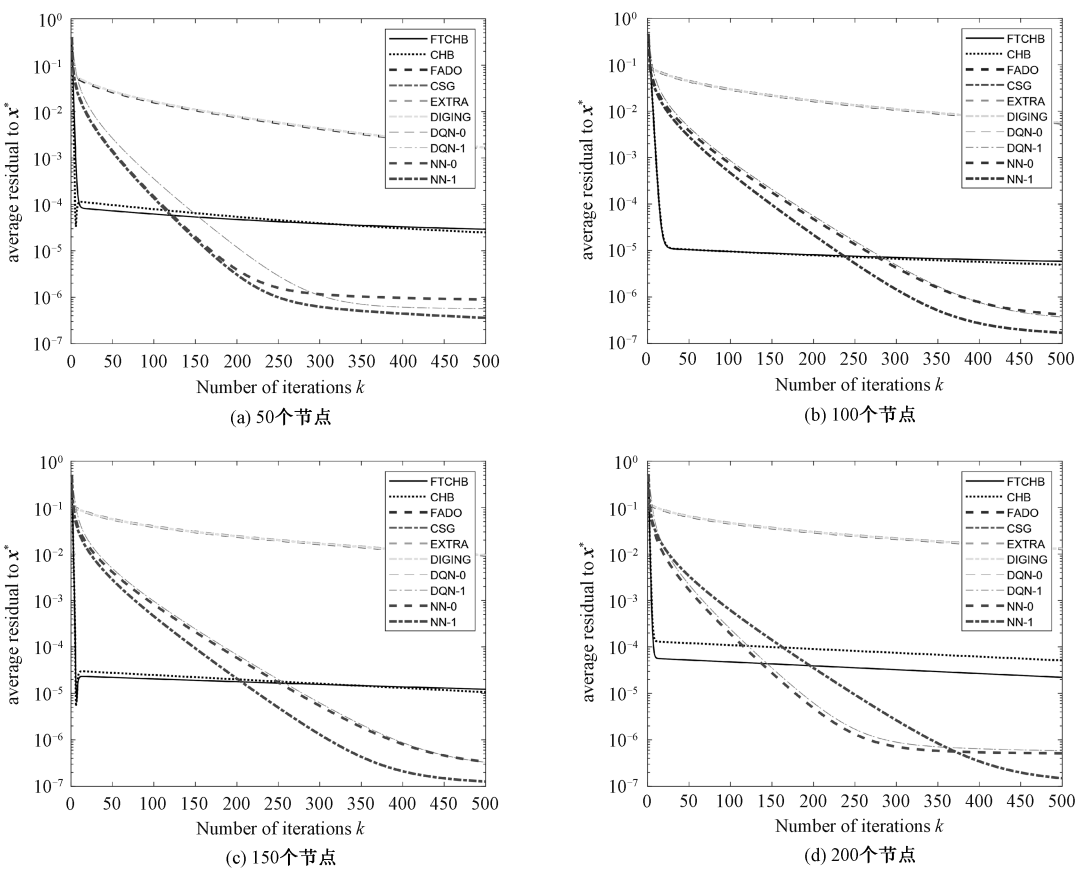
在限制性强凸的条件下有 $\|\boldsymbol{x}_c^k - \overline{\boldsymbol{x}_c^k}\|^2 \leqslant (F(\boldsymbol{x}_c^k) - F^*)/\nu \leqslant V(\boldsymbol{x}_c^0)\left(\frac{\ell}{1+\ell}\right)^{k+1}/\nu$, 即 $x_c^k$ 也线性收敛到最优解集。

## 5　数值仿真

本章节通过数值仿真体现 FTCHB 算法的有效性。图 1 展现不同规模网络下的优越性,图 2 展现不同问题参数下的性能。模型目标函数满足假设 1,2,3 均可以使用本文算法。常见的如最小二乘问题、机器学习问题、资源分配问题。本文的数值仿真考虑机器学习常见的岭回归问题[16],岭回归问题通过在目标函数中加入求解参数的正则项有效地处理了机器学习中的过拟合问题,因此得到广泛应用。在此问题中假设每个节点的目标函数为 $f_i(\boldsymbol{x}) = \frac{\lambda}{2}\|\boldsymbol{x}\|^2 + \sum_{l=1}^{q_i}\log[1+\exp(-v_{il}\boldsymbol{u}_{il}^{\mathrm{T}}\boldsymbol{x})]$, 此时整体目标函数为

$$F(\boldsymbol{x}) = \frac{N\lambda}{2}\|\boldsymbol{x}\|^2 + \sum_{i=1}^{N}\sum_{l=1}^{q_i}\log[1+\exp(-v_{il}\boldsymbol{u}_{il}^{\mathrm{T}}\boldsymbol{x})],$$

其中:$N$ 为网络中节点的总数,$\lambda > 0$ 为岭回归所对应的正则化参数,$q_i$ 表示节点 $i$ 上分配的数据个数,在此仿真中 $q_i = 6$。$\boldsymbol{u}_{il}$ 表示节点 $i$ 上的第 $l$ 个数据,而 $v_{il} \in \{-1, +1\}$ 表示此数据对应的标签。在我们的仿真中随机生成数据 $\boldsymbol{u}_{il} \in \mathbb{R}^3$, 其中每个维度满足均值为±10,方差为 1 的正态分布。每个节点的 6 个数据中有 3 个正样本, 3 个负样本。对于给定随机网络的生成,首先生成 $N-1$ 条边,构成一个满足假设 4 的连通网络,在

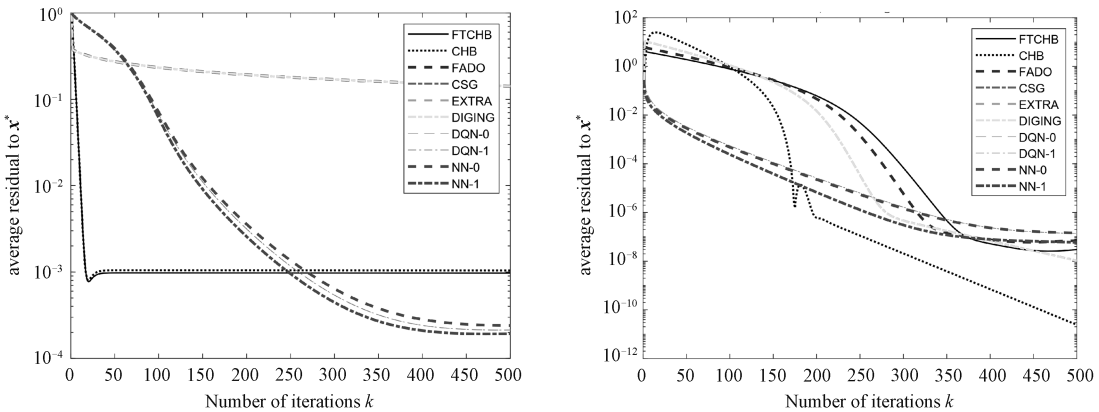



(a) 50个节点　(b) 100个节点

(c) 150个节点　(d) 200个节点

**图 1　不同网络规模的收敛结果比较($\lambda=1$)**

Fig. 1　Convergence result under different network sizes($\lambda=1$)




(a) $\lambda$=0.01 (b) $\lambda$=10

**图 2 不同 $\lambda$ 情况下收敛结果的比较($N$=100)**

Fig. 2 Convergence result with different $\lambda$ values($N$=100)

此基础上随机选择没有连接的节点添加链接直至链接的边数达到我们设置的节点平均度的要求,在此数值仿真中网络节点平均度设置为 5。通过 YAMLIP 软件包[17]进行集中式求解问题(1)获取参考最优解。网络加权矩阵设置为 $\boldsymbol{w}_{ij}=\frac{1}{\max\{\deg(i),\deg(j)\}+2}$。

我们对比了一阶算法中的集中式的重球法(CHB)、FADO[9]、集中式的梯度下降法(CSG)、EXTRA[6]、DIGING[7]以及二阶算法中的 DQN-0,1[18],NN-0,1[19]。图片中的横坐标为迭代次数,一次迭代表示进行一次节点变量 $\boldsymbol{x}_c^k(i)$ 更新对应的计算,也即一次算法 1 的步骤 7。这是因为在算法 1 中只有梯度计算为计算量大的操作,其他步骤仅为加权求和的线性操作。对于其他比较算法图中的横坐标增加 1 是一次方向更新。图片的纵坐标为网络中节点状态到最优解的残差平均值,即 $\frac{\sum_{i=1}^{N}\|\boldsymbol{x}_i-\boldsymbol{x}^*\|^2}{N}$。对所有算法参数的选取均为手调最优值,即 500 次迭代后对应最小残差平均值的参数。

从图 1,图 2 可以看出除 FTCHB 外,相比其他一阶算法收敛比较接近,二阶算法之间的收敛也比较接近。与此同时 FTCHB 相对于其他一阶算法可以更快地收敛到更高的精度。二阶算法收敛最终收敛精度超过一阶算法。对比不同网络规模的情况,通过图 1 可得知在网络规模变化的情况下 FTCHB 算法与其他算法的相对关系没有改变并且可以较快收敛到较高精度。从图 2 可以看出当 $\lambda$ = 10(对应条件数 $3.46\times10^3$)时所有算法表现均强于 $\lambda$ = 0.01(对应条件数 $3.7\times10^6$),这可以看出条件数对一阶算法影响较大而对二阶算法影响较小。比较图 2(a)和 2(b)两图可知 FTCHB 作为一阶算法在条件数大时可以很快收敛到较高精度。

此外,通过数值仿真可以看到设计算法的等价形式与集中式的重球法求解的最优值相差无几。两者有一定差别的主要原因是在算法初始化步骤 1 过程中的计算误差,如何减小此误差是一个单独的开放问题。

## 6 总结

本文提出一种解决基于共识的多智体系统协同优化问题的分布式算法 FTCHB。通过结合 FTC 及重球法,在凸假设和 $L$-光滑假设下可以达到 $\mathcal{O}(1/k)$ 的速率,并且加入限制性强凸的条件之后目标函数可以达到非各态历经性的线性收敛速率。

### 参考文献

[1] Beck A, Nedić A, Ozdaglar A, et al. An O(1/k) gradient method for network resource allocation problems[J]. IEEE Transactions on Control of Network Systems, 2014, 1(1): 64-73.

[2] Rabbat M, Nowak R. Distributed optimization in sensor networks [C] // Proceedings of the 3rd International Symposium on Information Processing in Sensor Networks 2004. April 26-27, 2004. Berkeley, California, USA. New York: Association for Computing Machinery, 2004: 20-27.

[3] Shamma J S. Cooperative control of distributed multi-agent systems[M]. New York: John Wiley & Sons, Ltd, 2007.

[4] Wai H T, Yang Z, Wang Z, et al. Multi-agent reinforcement learning via double averaging primal-dual optimization[C] //

Bengio S, Wallach H, Larochelle, et al. Advances in Neural Information Processing Systems 31: New York: Curran Associates, 2018:9649-9660.

[5] Nedic A, Ozdaglar A. Distributed subgradient methods for multi-agent optimization[J]. IEEE Transactions on Automatic Control, 2009, 54(1): 48-61.

[6] Shi W, Ling Q, Wu G, et al. EXTRA: an exact first-order algorithm for decentralized consensus optimization[J]. SIAM Journal on Optimization, 2015, 25(2): 944-966.

[7] Nedić A, Olshevsky A, Shi W. Achieving geometric convergence for distributed optimization over time-varying graphs[J]. SIAM Journal on Optimization, 2017, 27(4): 2597-2633.

[8] Sundaram S, Hadjicostis C N. Finite-time distributed consensus in graphs with time-invariant topologies[C]//2007 American Control Conference, July 9-13, 2007, New York NY, USA. IEEE, 2007: 711-716.

[9] Mai V S, Abed E H. Local prediction for enhanced convergence of distributed optimization algorithms[J]. IEEE Transactions on Control of Network Systems, 2018, 5(4): 1962-1975.

[10] Qu Z H, Wu X Y, Lu J. Finite-time-consensus-based methods for distributed optimization [C]// 2019 Chinese Control Conference (CCC). July 27-30, 2019, Guangzhou, China. IEEE, 2019: 5764-5769.

[11] Xiao L, Boyd S. Fast linear iterations for distributed averaging [J]. Systems & Control Letters, 2004, 53(1): 65-78.

[12] Horn R A, Johnson C R. Matrix analysis[M]. 2nd ed. Cambridge: Cambridge University Press, 1985, 191-200.

[13] Sundaram S, Hadjicostis C N. Distributed function calculation and consensus using linear iterative strategies[J]. IEEE Journal on Selected Areas in Communications, 2008, 26(4): 650-660.

[14] Ghadimi E, Feyzmahdavian H R, Johansson M. Global convergence of the heavy-ball method for convex optimization [C]//2015 European Control Conference (ECC). July 15-17, 2015, Linz, Austria. IEEE, 2015: 310-315.

[15] Sun T, Yin P H, Li D S, et al. Non-ergodic convergence analysis of heavy-ball algorithms[C]// Proceedings of the AAAI Conference on Artificial Intelligence. AAAI, 2019, 33(1): 5033-5040.

[16] Murphy K P. Machine learning: a probabilistic perspective [M]. Cambridge, Massachusetts: MIT press, 2012:290-295.

[17] Löfberg J. YALMIP: a toolbox for modeling and optimization in MATLAB[C]//2004 IEEE International Conference on Robotics and Automation (IEEE Cat. No. 04CH37508). September 2-4 2004, Taipei, Taiwan, China. IEEE, 2004: 284-289.

[18] Bajović D, Jakovetić D, Krejić N, et al. Newton-like method with diagonal correction for distributed optimization [J]. SIAM Journal on Optimization, 2017, 27(2): 1171-1203.

[19] Mokhtari A, Ling Q, Ribeiro A. Network Newton distributed optimization methods [J]. IEEE Transactions on Signal Processing, 2017, 65(1): 146-161.