西安交通大學
XI'AN JIAOTONG UNIVERSITY

多智能体系统中的一致性问题与分布式协同优化

姓名 刘济衔　　IWIN-FINS Lab

摘　要

本报告主要介绍了一个月以来关于网络化多智能体系统中的一致性 (Consensus) 和分布式协同优化的学习内容.

在一致性问题方面，首先介绍了一致性的研究现状，之后从一个例子出发，辨析了 Consensus、Average Consensus 的概念，给出了收敛和平均一致性的条件，进一步引出了我们在一致性问题中所关心的问题

- 在什么样的条件下状态向量的每一点都可以收敛到同一个值?(**consensus**)
- 这样一个收敛值在什么样的条件下是每一点的初值之和的平均值? (**Average Consensus**)
- 刻画一个实际网络的图和对应的矩阵形式应当满足什么样的特性才能够使得设计的算法最终满足收敛?

并进行谱分析，给出了对应的收敛条件和证明。

在分布式协同优化方面主要学习了 DGD, ADMM, EXTRA, Push-sum Based 及其相关算法并用 MATLAB 分别对多智能体系统连续与离散情况下的一致性演化和分布式协同优化的经典算法进行的模拟仿真，对一些经典论文进行了结果复现。

在多智能体一致性问题与分布式优化的学习之外的文献阅读中，了解了关联噪声 (correlated noise) 对于加强谐振子同步 (Oscillator synchronization) 的影响，了解了机器学习特别是联邦学习 (Federated learning) 在复杂网络，多智能体系统与分布式系统中的应用和利用图神经网络 (Graph neural network) 加速收敛的方法。

目录

1 引言

This section is chew by kitties and puppies.

2 多智能体协同控制与一致性

2.1 研究现状

自上世纪 60 年代 DeGroot 提出 statistical consensus theory 以来，协同控制不断发展，从单体到多体，从无向图到有向图，从集中式到分布式，从时不变网络到时变网络，从无噪声到有噪声，协同控制与一致性演化得到了广泛的研究，与之相应的凸优化或非凸优化算法 (如 GD,DGD,EXTRA,Push-sum,SONATA,ZONE 等) 也获得长足发展，在互联网，传感器网络 (WSNs)，车联网 (V2X)，生物与非生物集群 [1]，智能电网 (如级联故障分析 [2]) 等方面得到了广泛的应用。

网络动态系统共识问题的理论框架是是在 Fax 和 Murray[3]，[4] 的早期工作的基础上提出的。由 olfat - saber 和 Murray 在 [5] 和 [6] 中提出的。在 Jadbabaie 等人 [7] 的工作中出现了关于不计算任何目标函数而达成一致的对齐问题 (Aligment) 的研究。进一步的，在 [8] 和 [9] 中对这项工作进行了进一步的理论扩展，并讨论了有向图情况下网络中定向信息流的处理，如图1所示。

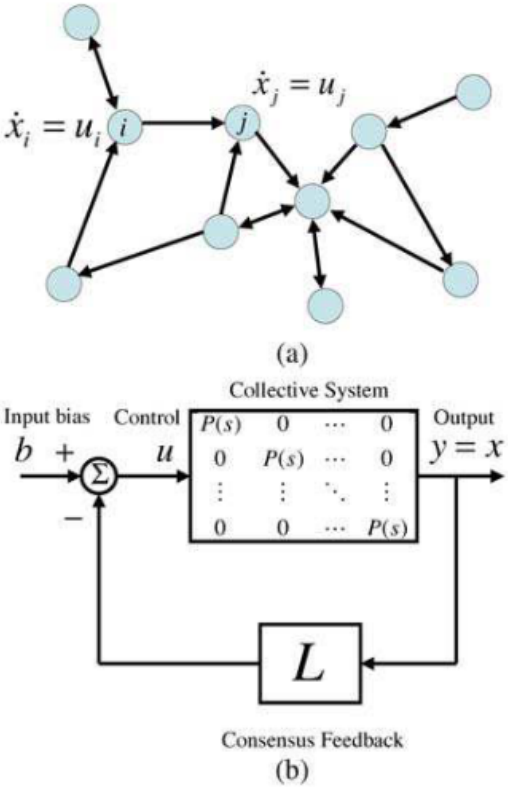

图 1: MIMO 系统下有向图中 Consensus Algorithms 的两种等效表达

除此之外，Olfat - saber 和 Murray 也在 [10] 中总结了关于网络拓扑的图结构与谱特性分析，集群行为 (collective behavior of flocks and swarms)，信息融合 (Data Fusion) 等方面的研究方向和主要结果。

2.2 从一个例子出发

现在首先来看一个简单的例子，在如图2所示的网络结构下，我们定义离散时间下的迭代过程为 $x(k+1)=Ax(K)$，为了讨论的方便起见，我们考虑这样一种平均算法

$$x_i(k+1)=\text{mean}\left(x_i(k),\{x_j(k)\}\right),\ \text{for all neighbors } j \tag{1}$$

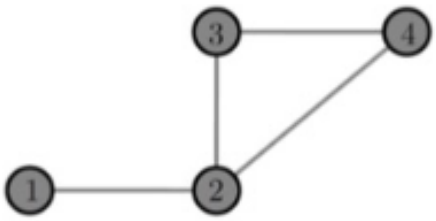

图 2: 第一个例子

我们设网络中的四个节点的初值分别为 $x_1=25,x_2=20,x_3=24,x_4=27$ 对于

$$A=A_0=\begin{bmatrix}1/2 & 1/2 & 0 & 0\\ 1/4 & 1/4 & 1/4 & 1/4\\ 0 & 1/3 & 1/3 & 1/3\\ 0 & 1/3 & 1/3 & 1/3\end{bmatrix} \tag{2}$$

我们有数值仿真结果如图3所示

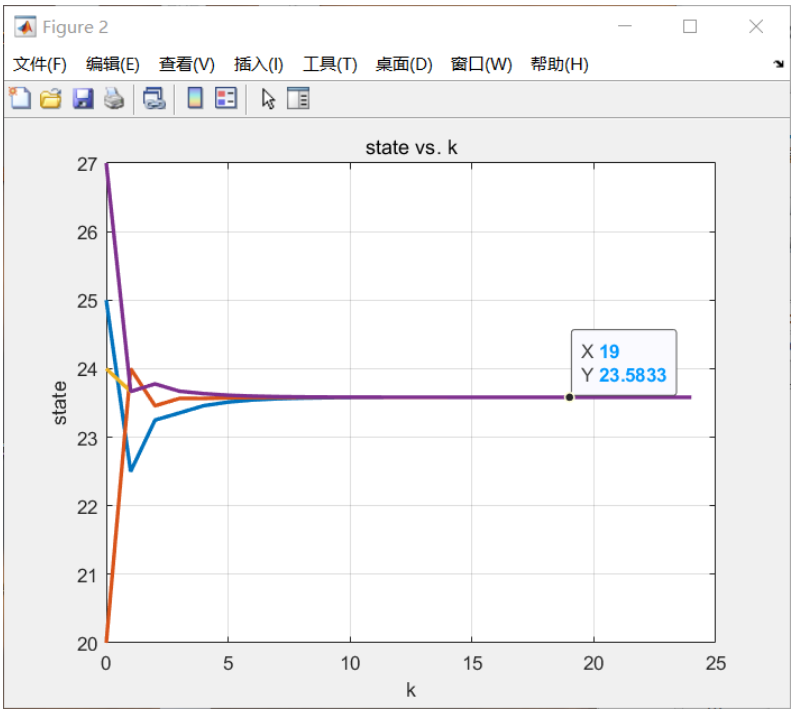

图 3: $A=A_0$ 时的迭代图样

对于

$$A = A_1 = \begin{bmatrix} 3/4 & 1/4 & 0 & 0 \\ 1/4 & 1/4 & 1/4 & 1/4 \\ 0 & 1/4 & 5/12 & 1/3 \\ 0 & 1/4 & 1/3 & 5/12 \end{bmatrix} \tag{3}$$

我们有数值仿真结果如图4所示

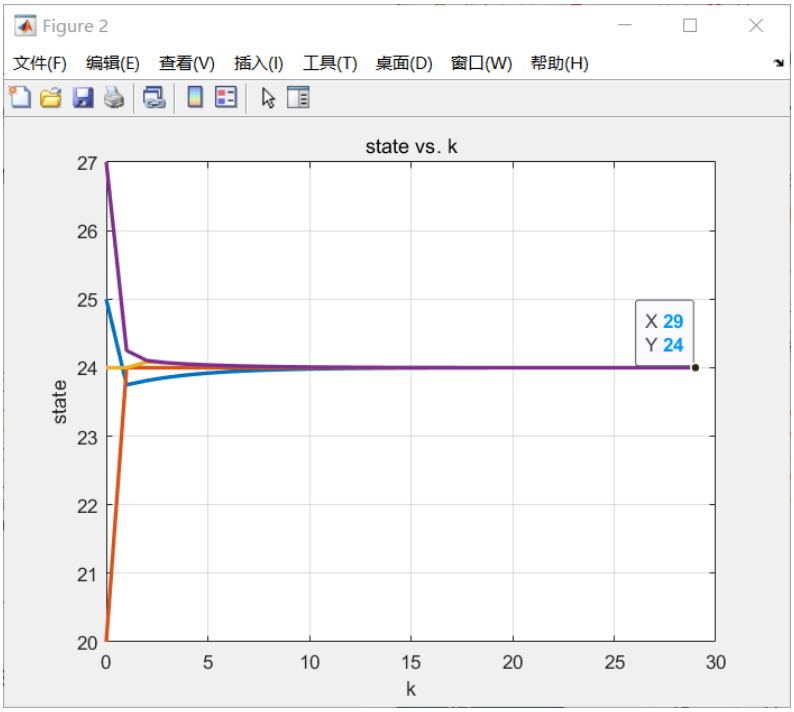

图 4: $A = A_1$ 时的迭代图样

从以上两个仿真结果中不难看出，我们可以得到如下结论

- 两次仿真结果的状态向量的每一点的代表值最终均收敛。(**consensus**)
- 第二次仿真结果的状态向量的每一点的代表值最终收敛到每一点的初值之和的平均值，而第一次的仿真结果却不行 (**Average Consensus**)

首先观察这两个矩阵共有的特性，它们都是非负矩阵，它们都是行随机 (Row stochastic) 矩阵且对应的网络均为强联通的。而他们的不同点是 A_1 是对称且满足双随机协议而 A_0 不满足 (事实上接下来我们会说明究竟是对称还是双随机协议导致的 Average Consensus)(见6.1)。

根据以上有趣的发现，我们可以自然而然地引出想要讨论的问题为:

- 在什么样的条件下状态向量的每一点都可以收敛到同一个值?(**consensus**)
- 这样一个收敛值在什么样的条件下是每一点的初值之和的平均值? (**Average Consensus**)
- 刻画一个实际网络的图和对应的矩阵形式应当满足什么样的特性才能够使得设计的算法最终满足收敛?

2.2.1 Consensus in Networks

假设有向图的拓扑结构由 $G=(V,E)$ 表示，其中 $V=1,2,3...n$ 表示顶点集合，$E\subseteq V\times V$ 表示图中边的集合。对于满足 $N_j=j\in V:(i,j)\in E$ 的顶点 j，称为 i 的邻居。

一个简单的到达一致性的 consensus algorithm 可以表示为

$$\begin{aligned}\dot{x}_i(t)=\sum_{j\in N_i}(x_j(t)-x_i(t))+b_i(t),\quad x_i(0)=z_i\\ \in\mathbb{R},b_i(t)=0.\end{aligned}\tag{4}$$

引入拉普拉斯矩阵 L 满足:

$$l_{ij}=\begin{cases}-1, & j\in N_i\\ |N_i|, & j=i.\end{cases}\tag{5}$$

则问题可以写为:

$$\dot{x}=-Lx\tag{6}$$

2.2.2 Information Consensus

达到 consensus 是指状态向量 x 中的每一项均有:

$$x_1=x_2=...=x_n\tag{7}$$

即 $x=\alpha\mathbf{1}$，其中 $\mathbf{1}=(1,1...,1)^T$ 一种 distributed consensus algorithm 可以表示为

$$\dot{x}_i(t)=\sum_{j\in N_i}a_{ij}\left(x_j(t)-x_i(t)\right)\tag{8}$$

对于无向图而言最终可以达到 consensus 满足 $\alpha=\frac{1}{n}\sum_i x_i(0)$，对于有向图而言，想要达到平均一致性（$AverageConsensus$）需要满足 $\sum_{j\neq i}a_{ij}=\sum_{j\neq i}a_{ji}$ for all $i\in V$，即平衡有向图（$BalancedDgraph$）具体证明见6.1中的有向图的收敛性分析。

对于无向图而言以上问题还可以化为 $\cdot x=-\bigtriangledown\varphi(x)$，其中 $\varphi(x)=\frac{1}{2}x^TLx$，具体的收敛性证明只需要将有向图中的非对称矩阵改为对称矩阵即可。

2.3 仿真结果

2.3.1 问题 1：有向图-Continuous-time Consensus

对于如图5所示的网络（[11] 第 27 页）而言，假设所有边的权重均为 1，

代入

$$dotx_i=u_i=\sum_j a_{ij}\left(x_j-x_i\right)=\sum_j a_{ij}x_j-\underbrace{\sum_j a_{ij}}_{d_i}x_i,\tag{9}$$

矩阵形式为

$$\dot{x}=-\underbrace{(D-A)}_{L}x=-Lx\tag{10}$$

其中，D 为表示每一点度数的对角矩阵，A 为邻接矩阵。

仿真结果如图6所示

注意离散时间下采用 ODE45 求解器而非迭代

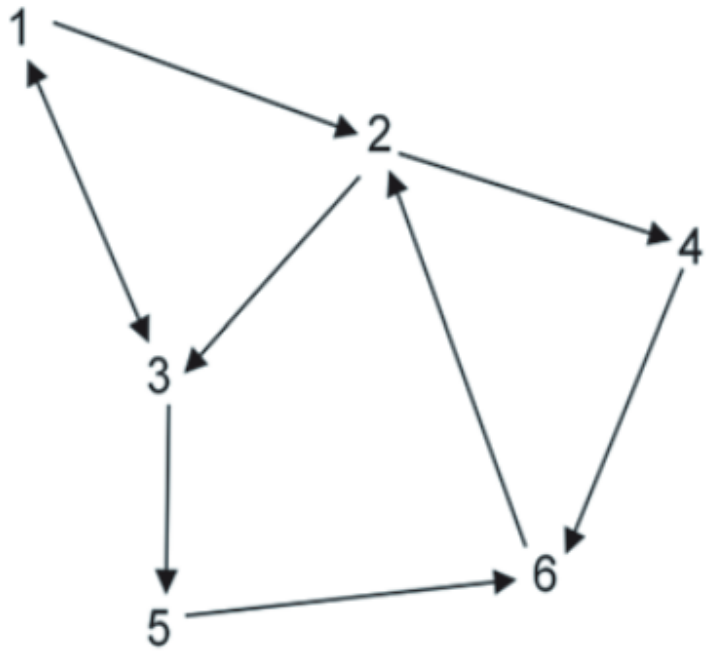

图 5: 一个有向图

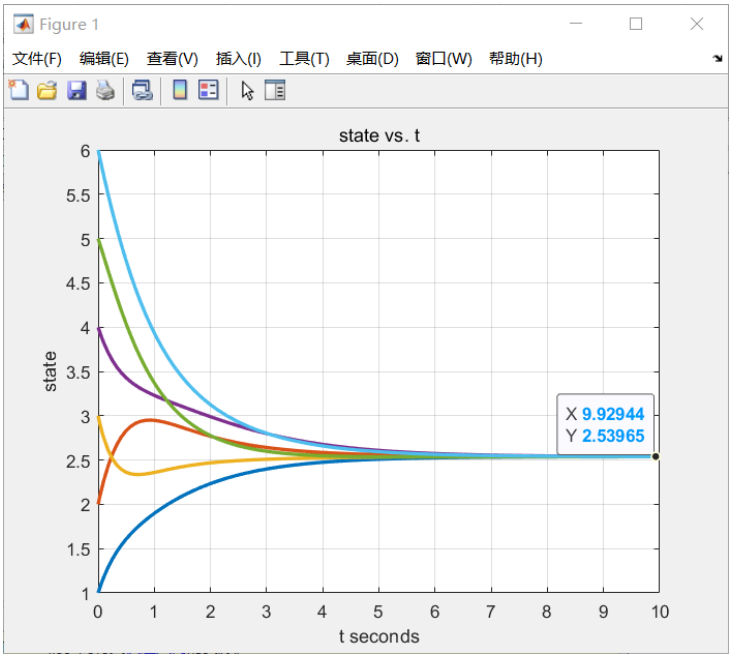

图 6: Continuous-time Consensus

2.3.2 问题 2：有向图-Discrete-time Consensus

若将时间离散，则有迭代关系满足

$$x_i(k+1) = x_i(k) + \epsilon \sum_{j \in N_i} a_{ij} \left(x_j(k) - x_i(k) \right), \tag{11}$$

即

$$x(k+1) = Px(k), \tag{12}$$

其中 $P = I - \epsilon L(Perron)$ ， ϵ 为迭代的时间步长

仿真结果如图7所示

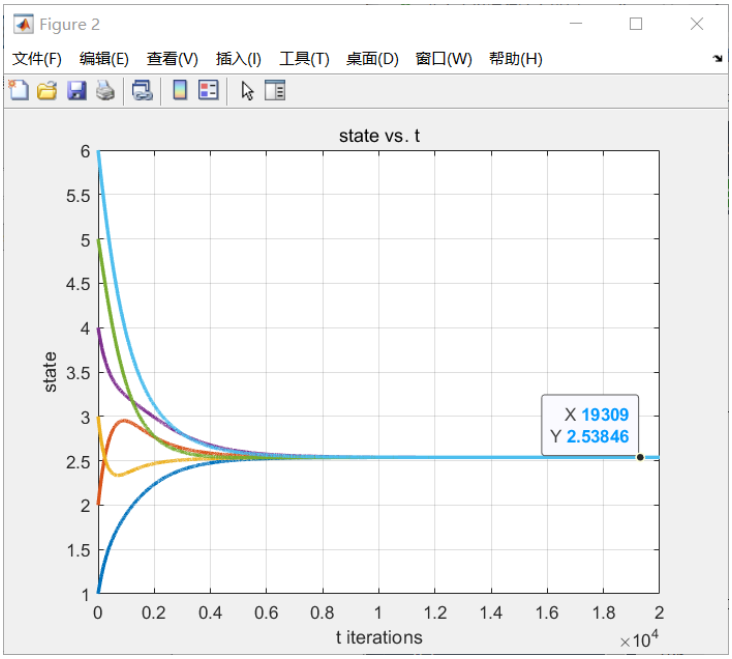

图 7: Discrete-time Consensus

代码分别为：

```
%%inputs
function xdot=inputs(t,x,L)
xdot = -L*x;
end

%%pb 1
A=[0 0 1 0 0 0;1 0 0 0 0 1;
   1 1 0 0 0 0;0 1 0 0 0 0;
   0 0 1 0 0 0;0 0 0 1 1 0]
%calculate the row sum
D=diag(sum(A,2));
%time limits
t0=0;tf=10;
%laplacians
```

```
L= D-A;
%initial conditions
x0=[1 2 3 4 5 6];
[t,x]=ode45(@inputs,[t0 tf],x0,[],L);
figure(1)
plot(t,x,LineWidth=2);
xlabel('t seconds')
ylabel('state')
title('state vs. t')
grid on

%%pb 2
A=[0 0 1 0 0 0;1 0 0 0 0 1;
   1 1 0 0 0 0;0 1 0 0 0 0;
   0 0 1 0 0 0;0 0 0 1 1 0]
%calculate the row sum
D=diag(sum(A,2)); I=eye(size(A,1)); alpha=0.01
%iteration
F = I-0.001*(D-A);N=20000;
x=zeros(6,N);x0=[1,2,3,4,5,6]'; x(:,1)=x0;
%[t,x]=ode45(@inputs,[t0 tf],x0,[],L);
for K=2:N
    if K==2
        x(:,K)=F*x0;
    else
        x(:,K)=F*x(:,K-1);
    end
end
t=0:N-1
figure(2)
plot(t,x,LineWidth=2);
xlabel('t iterations')
ylabel('state')
title('state vs. t')
grid on

%% example 1 A1
A=[3/4 1/4 0 0;
   1/4 1/4 1/4 1/4;
   0 1/4 5/12 1/3;
   0 1/4 1/3 5/12;]
%iteration
N=30;
```

```
x=zeros(4,N);x0=[25,20,24,27]'; x(:,1)=x0;
%[t,x]=ode45(@inputs,[t0  tf],x0,[],L);
for K=2:N
    if K==2
        x(:,K)=A*x0;
    else
        x(:,K)=A*x(:,K-1);
    end
end
t=0:N-1
figure(2)
plot(t,x,LineWidth=2);
xlabel('k')
ylabel('state')
title('state vs. k')
grid on
```

3 分布式协同优化

3.1 研究现状

对于分布式优化而言，我们总是存在这样一个目标：通过每个节点与邻居节点进行信息交互，最终协同实现全局优化的目标 $F(x)=\frac{1}{n}\min\limits_{x\in R^n}\sum\limits_{i=1}^{N}f_i(x)$，最终每个智能体自身状态收敛到全局最优解。

在分布式算法当中，收敛的保证是十分重要的，所以总是需要设计合适的手段来保证迭代过程中收敛性，最开始很自然的一种想法是设计逐渐减小的迭代的时间步长（衰减步长）就可以保证对应问题的收敛性，但收敛速度较慢。(DGD & Push-sum Based)

最近的研究热点是设计快速收敛到最优解的分布式加速优化算法. 这些算法大多采用固定步长，另外要求局部目标函数光滑且强凸。(EXTRA & Push-DIGing)

在分布式优化算法方面，近年来的一些有代表性的进展主要有：

- Nedich 于 2009 年提出分布式梯度下降 (DGD)(Undirected graph & sub-linear rate)[12]
- 2014 年 Nedich 与 Olshecsky 提出基于 Push-Sum Protocol 的算法 (Directed graph & sub-linear rate)[13]
- 2015 年,施伟和凌青提出了一种新算法–EXTRA(exact first-order algorithm)(Undirected graph & linear rate)[14]
- 2017 年，Olshecsky 提出了 Push-DIGing 算法 (Directed graph & linear rate)[15]
- 2019 年，Scutari 提出了 SONATA 算法 (Directed graph & non-convex)[16]
- 2019 年，Hong 提出了 ZONE 算法 (Undirected graph & non-convex)[17]

一些综述文章有 [18]，[19]，[20]，[21]...

3.2 分布式梯度下降法 (DGD)

3.2.1 DGD 的原理

DGD 算法的迭代步骤为

$$x_i(k+1) = \sum_{j=1}^{N} w_{ij}(k)x_j(k) - \alpha s_i(k) \tag{13}$$

其中, $w(k)_{ij}$ 是通信网络在时刻 k 边 (j,i) 的权重, 权重矩阵 $W(k) = [w_{ij}(k)]$ 为双随机矩阵,$s_i(k)$ 是局部目标函数 $f_i(x)$ 的次梯度, 步长 $\alpha(k) > 0$ 非增, 且满足

$$\sum_{k=0}^{\infty} \alpha(k) = \infty, \sum_{k=0}^{\infty} \alpha^2(k) < \infty \tag{14}$$

3.2.2 DGD 仿真结果

对于2.2描述的问题，我们如果假设有其目标函数为 $\sum_{j\in N(i)} e^{\frac{1}{2}(xi-xj)^2}$，显然为强凸函数，将初值 $x_1 = 1$，$x_2 = 8$，$x_3 = 0.3$，$x_4 = 0.7$ 代入得到数值仿真结果如图8和图9所示

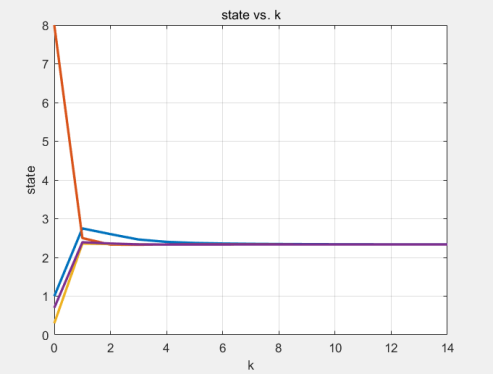

图 8: 状态向量随迭代次数的变化图

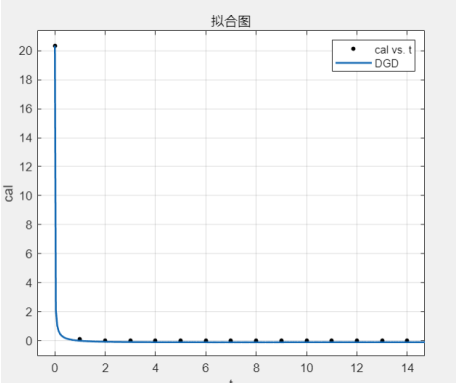

图 9: 相对误差随迭代次数的变化图

其中，用 $a\frac{ln(t)}{\sqrt{t}}$ 进行拟合的残差平方和 (SSE) 为 0.143，R 方为 0.9996，拟合效果好。说明收敛速率为 $O(\frac{ln(t)}{\sqrt{t}})$

对应的代码为

```
%% DGD
A=[3/4  1/4  0   0;
    1/4  1/4  1/4  1/4;
    0  1/4  5/12  1/3;
    0  1/4  1/3  5/12];
%iteration
N=2000;
N2=15;
x=zeros(4,N);x0=[1,8,0.3,0.7]'; x(:,1)=x0;
dt=0.01
for K=2:N
    if K==2
```

```
        x(:,K)=A*x0;
    else
        alpha=1/K;
        x(:,K)=A*x(:,K-1)-alpha*gra_qrad(x(:,K-1),A);
    end
end
x=x(:,1:N2);
t=0:N2-1;
figure(1)
plot(t,x,LineWidth=2);
xlabel('k')
ylabel('state')
title('state vs. k')
grid on

%%calculation of error
cal=sum(1/2*(x-repmat(x(:,N2),1,N2)).^2);
t=0:N2-1;
figure(2)
plot(t,cal,LineWidth=1);
xlabel('k')
ylabel('state')
title('error')
grid on


function qradsum=gra_qrad(x,A)
    B=zeros(length(A));C=zeros(length(B));
    B(find(A~=0))=1;
    B=B-eye(length(A));
    for i=1:length(A)
        for j=1:length(A)
            if B(i,j)==0
                C(i,j)=0;
            else
                C(i,j)=x(i)-x(j);
            end
        end
    end
    K=abs(C).*exp(1/2.*(C.^2));
    qradsum=sum(K,2);
end
```

3.3 交叉方向乘子法 (ADMM)

我们现在来考虑这样一个问题，对于 $\min f(x)+g(x)$ 而言，这里 $f(x)$ 和 $g(x)$ 都是凸函数，此时目标函数有两个，整体优化可能不太方便，现在想要交替优化两项，考虑将这个优化问题写为 [22]

$$\begin{aligned} &\min f(x)+g(x) \\ &s.t. x=z \end{aligned} \tag{15}$$

对应的增广拉格朗日函数为 $L_c(x,z,v)=f(x)+g(x)+v^T(x-z)+0.5c\|x-z\|_2^2$

在上面的步骤中，使用增广拉格朗日函数交替优化原变量 x,z，和对偶变量 v，交替优化的两个步骤如下所示

1. $\{x^{k+1},z^{k+1}\}=\underset{x,z}{argmin}\, f(x)+g(z)+v^T(x-z)+0.5c\|x-z\|_2^2$
2. $v^{k+1}=vk+c(x^{k+1}-z^{k+1})$

在上面步骤 1 中，优化了两个变量 x,z，现在把这一步运用分块坐标轮换法，进一步地交替优化，拆解为两步，并且将增广拉格朗日函数后两项配方法合并得到

1. $x^{k+1|t+1}=\underset{x}{argmin}\, f(x)+\frac{c}{2}\|x-z^{k+1|t}+\frac{v^k}{c}\|_2^2$
2. $z^{k+1|t+1}=\underset{z}{argmin}\, g(z)+\frac{c}{2}\|z-x^{k+1|t+1}+\frac{v^k}{c}\|_2^2$

这两步多次交替迭代，然后再迭代一次对偶变量 v，这种方法即是交替方向乘子法 (Alternating Direction Method of Multipliers, ADMM)。这里给出几个 [23] 书中例子的仿真结果。每个示例脚本都显示了原始残余 (Primal residual $\|r^k\|$)、原始可行性容差 ϵ^{pri}、对偶残余 (Dual residual $\|s^k\|$) 和对偶可行性容差 ϵ^{dual}。有关这些数据的更多细节，请参阅 [23] 第 3.3 节。还包括目标价值图和迭代的原始和对偶残差图。注意到，任何特定迭代的目标值都可能低于真正的解值 p^*，因为迭代不一定是可行的 (例如，即使约束是 $x-z=0$，我们也可以得到对某些迭代 k 而言，约束 $x^k-z^k\neq 0$)。[1]

3.3.1 Basis Pursuit

对于 [23] 中的 Basis Pursuit 示例的结果如10所示：

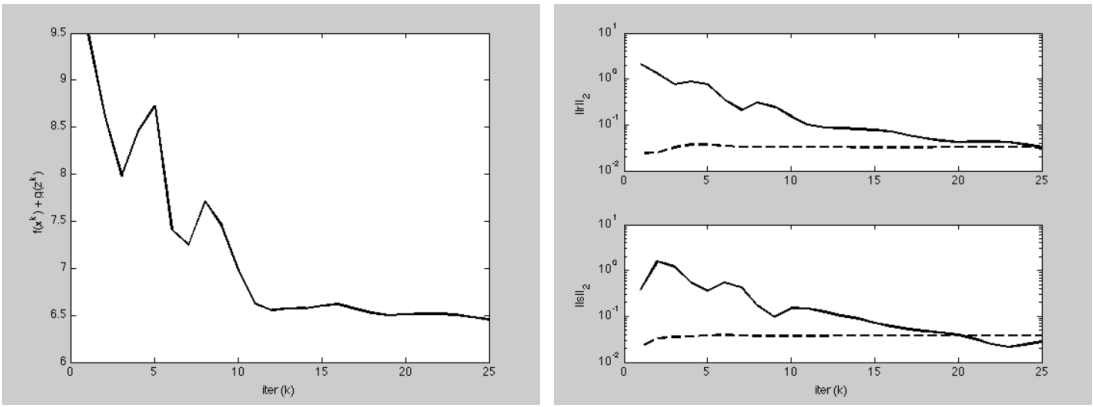

图 10: Basis Pursuit

3.3.2 LASSO

对于 [23] 中的 LASSO 示例的结果如图11所示：

[1] Open source:https://web.stanford.edu/~boyd/papers/admm

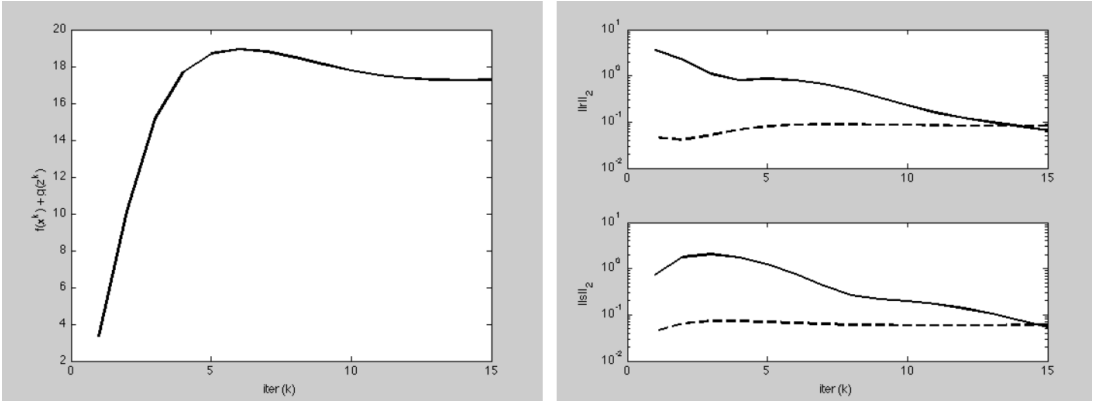

图 11: LASSO

3.3.3 Support vector machine

对于 [23] 中的 SVM(Support vector machine) 示例的结果如图12所示：

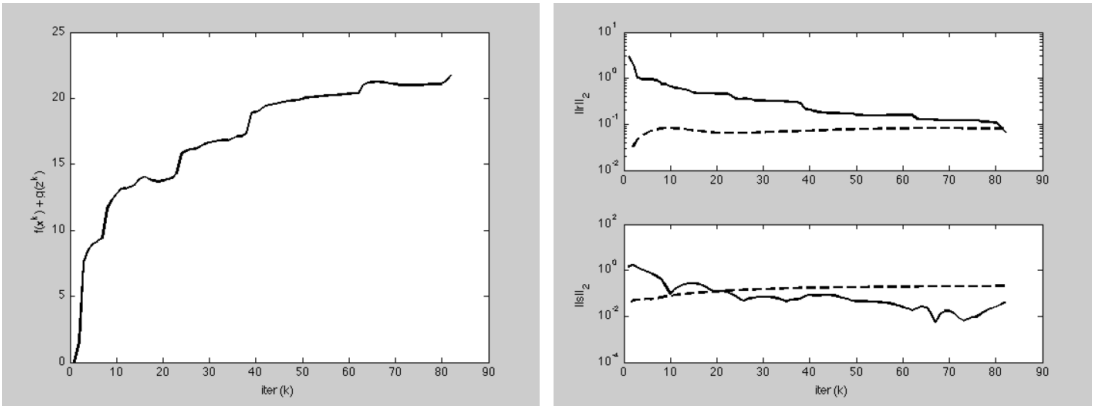

图 12: SVM

3.4 Push-sum Based Algorithm

3.4.1 Push-sum Based 算法的研究现状

Push-sum Based 的分布式算法得到了广泛的研究，它采用了一种不同类型的一致性算法。该算法由于其形式，也被称为双线性迭代或比率一致性 (Ratio consensus) 算法，它涉及到两个变量的比率，根据相同的线性动力学进化，但在初始点的选择上有所不同。

这一算法最早是由 Kempe 等人基于 Push-sum protocol 提出 [24], 2014 年, Nedich 与 Olshecsky 提出了 Subgradient push algorithm，为有向图和时变网络的分布式优化提供了一种工具 [13]，之后，Olshecsky 又提出了固定步长的 Push-DIGing 算法 [21]。

3.4.2 Push-sum 的仿真结果

对于一个给定的 WSN 网络，利用 push-sum 算法分别给出了不同断边个数下 (表示时变图) 的相对误差随迭代次数的变化，收敛速率与 $O(\frac{ln(t)}{\sqrt{t}})$ 符合较好。

论文代码如下：

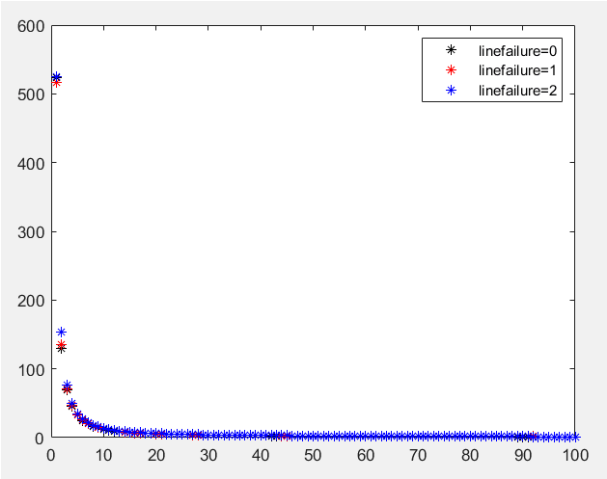

图 13: Push-sum

```
    clear;
vv=100;
su=[0  0  0]';
for i=1:vv
    b2(i)=0;
end
yy=0;
for v=1:vv
    cc=30000*10000;
    alpha=0.25;
    xa=[10  100  10  100];
    ya=[100  90  70  80];
    za=[10  10  100  100];
    for k=1:4
        p(1,1,k)=50;
        p(2,1,k)=50;
        p(3,1,k)=50;
    end
    i_p=[60  90  60];
    for i=1:4
        t(i)=(sqrt((i_p(1)-xa(i))^2 +(i_p(2)-ya(i))^2 +...
        (i_p(3)-za(i))^2))/cc;
    end
    for j=1:100
        for i=1:4
        %nv(i)=0.001*randn(1);
        nv(i)=0;
        end
    tt=t+nv;
```

```
        for i=1:4
            a=p(1,j,i)-xa(i);
            b=p(2,j,i)-ya(i);
            c=p(3,j,i)-za(i);
            d=tt(i);
            e=a^2+b^2+c^2;
            f(i)=(e^0.5-cc*d)^2;
            fdx(i)=(e^0.5-cc*d)*(e^(-0.5))*2*a;
            fdy(i)=(e^0.5-cc*d)*(e^(-0.5))*2*b;
            fdz(i)=(e^0.5-cc*d)*(e^(-0.5))*2*c;
        end
    [err(:,j)]=pushsumcc1(f);
    [g(:,1)]=pushsumcc1(fdx);
    [g(:,2)]=pushsumcc1(fdy);
    [g(:,3)]=pushsumcc1(fdz);
    for k=1:4
        gg=[];
        for i=1:3
            gg=[gg g(k,i)];
        end
        ggg(k,:)=gg;
    end
    for i=1:4
        p(:,j+1,i)=p(:,j,i)-alpha*ggg(i,:)';
    end
    end
        x=1:100;
        b2=b2+err(1,:);
        uu=p(:,100,1);
        su=su+uu;
        yy=yy+err(1,100);
end
su=su/vv
yy=yy/100
b2=b2/vv;
plot(x,b2,'*k')

%%pushsum line failure=0 as an example

function[st]=pushsumcc1(s)
T=14;
w=[0.25 0.25 0.25 0.25];
for i=1:4
```

```
        for j=1:4
            ss(i,j)=0;
            ww(i,j)=0;
        end
    end
    for i=1:4
        ss(i,i)=s(i);
        ww(i,i)=w(i);
    end
    for t=1:T
        for n=1:4
            s(n)=sum(ss(n,:));
            w(n)=sum(ww(n,:));
            st(n,t)=s(n)/w(n);
        end

        for i=1:4
            for j=1:4
                ss(i,j)=0;
                ww(i,j)=0;
            end
        end
    % rr represents the number of link failures
    rr=0;

    V=[1 2 3 4];
    M(:,:)=nchoosek(V,2);
    pp=[];
    mm=1;
    if rr==0
        pp=[];
    else
        for k=1:100
            num=ceil(6*rand);
            l=0;
            for j=1:mm-1
                if pp(j)==num
                    l=l+1;
                end
            end
            if l==0
                pp(mm)=num;
                mm=mm+1;
```

```
            end
            if length(pp)==rr
                break;
            end
        end
    end
    for n=1:4
        kk=ceil(3*rand(1));
        if kk<n
            k=kk;
        end
        if kk>=n
            k=kk+1;
        end
    ss(n,n)=0.5*s(n);
    ww(n,n)=0.5*w(n);
    ss(k,n)=0.5*s(n);
    ww(k,n)=0.5*w(n);
    for ii=1:rr
        r=pp(ii);
        m=M(r,1);
        mm=M(r,2);
        if ((k==m & n==mm)|(k==mm & n==m));
            ss(k,n)=0;
            ww(k,n)=0;
        end
    end
    end
end
st=st(:,T);
end
```

3.5 EXTRA

作为一种分布式加速优化算法，EXTRA 采用固定步长，克服了 DGD 在迭代过程中收敛较慢的问题，取得了在对数坐标系下线性的收敛速率 $O(1/t)$。

3.5.1 固定步长对于 DGD 算法的影响

固定步长下 DGD 算法只能够非精确收敛，首先我们假设 $\{x^k\}$ 收敛，并且 $x^* = \lim\limits_{k\to\infty} x^k$，根据 DGD 迭代过程可知

$$\mathbf{x}^* = W\mathbf{x}^* - \alpha\nabla\mathbf{f}(\mathbf{x}^*) \tag{16}$$

假定每个节点能够收敛到同一个点 x^*，所以有 $\mathbf{x}^* = \mathbf{1}x^{*T}$，再根据双随机矩阵的性质 $W1 = 1$，

可以得到

$$W\mathbf{x}^* = W\mathbf{1}x^{*T} = \mathbf{1}x^{*T} = \mathbf{x}^* \tag{17}$$

也就是。又因为 α 是一个固定的数，所以有 $\Delta f_i(x^*)$。这实际上是很难达到的，因为这要求，对于每个智能体的最优解都是一样的，也就是 x^* 同时最小化了所有的智能体的目标函数，所以我们不能这样要求。退一步说即便是这样的话，那么我们只需要求解其中一个智能体即可，就没必要这么麻烦了。

一个自然而然但又很巧妙的想法是这样，既然同时最小化是很难达到的，那么我们可以尝试只要求 $\sum_{i=1}^{n} \Delta f_i(x^*) = 0$，就得到了 EXTRA 算法中的一个关键步骤。

3.5.2 EXTRA 算法的原理

前面讲到了分布式梯度算法 DGD 在固定步长下，大概率是非精确收敛，而步长趋于零又会导致收敛变慢。EXTRA 就是解决这个问题，它实现了固定步长下的精确收敛。

EXTRA 算法的具体过程是

Algorithm 1: EXTRA

Choose $\alpha > 0$ and mixing matrices $W \in \mathbb{R}^{n\times n}$ and $\tilde{W} \in \mathbb{R}^{n\times n}$;

Pick any $\mathbf{x}^0 \in \mathbb{R}^{n\times p}$;

1. $\mathbf{x}^1 \leftarrow W\mathbf{x}^0 - \alpha\nabla\mathbf{f}(\mathbf{x}^0)$;
2. **for** $k = 0, 1, \cdots$ **do**

$\mathbf{x}^{k+2} \leftarrow (I + W)\mathbf{x}^{k+1} - \tilde{W}\mathbf{x}^k - \alpha\left[\nabla\mathbf{f}(\mathbf{x}^{k+1}) - \nabla\mathbf{f}(\mathbf{x}^k)\right]$;

end for

其中，最关键的两个地方是 EXTRA 算法首先使用了**两次迭代**而不是 DGD 与 Push-sum protocol 中的一次迭代，其次引入了一个**新矩阵** $\tilde{W} = \frac{I+W}{2}$，这两个步骤使得 EXTRA 算法具有固定步长的同时也保证了收敛性，使收敛速率从 $O(\frac{ln(t)}{\sqrt{t}})$ 变为了 $O(\frac{1}{t})$。

3.5.3 EXTRA 的仿真结果

对 [14] 结果部分中的第一种情形下的问题进行仿真可以得到实验结果如图14所示，可以明显的看出在纵坐标采用对数坐标系时 EXTRA 的收敛速率相较于 DGD 要快得多。

4 其他

4.1 学习外的文献阅读

除了多智能体一致性学习与分布式优化外，在这一个月以来还看了一些有意思的文献，一个有意思的现象是对于同步 (Synchronization) 而言，Martineau 等人发现关联噪声可以加强在谐振子同步 [25]，相关工作发表在 Physical Review Letters 中。

此外，机器学习尤其是联邦学习 (Federated learning) 在复杂网络，多智能体系统与分布式系统中的应用也十分广泛 [26][27],等人的工作展示了利用图神经网络加速机器人集群收敛的方法 [28][29]。

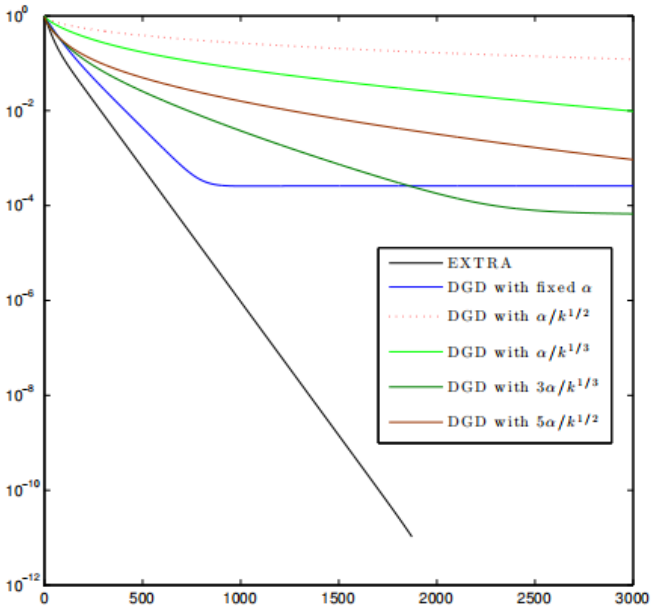

图 14: EXTRA vs. DGD

由于时间原因，其他的算法诸如适用于有向图的 DEXTRA，Push-DIGing 仅仅只是了解了一个大概，还没有进行代码复现，麦吉尔大学的 Mahmoud 在他的硕士论文中对以上几种算法利用 Python 给出了复现结果 [30][2]。

4.2 存在的问题

在进行 DGD 算法的仿真过程中，我发现对于例子一而言的复现总是存在这样一种现象，就是结果收敛与否还受到初值的影响，改变步长可以是初值的可行域的范围增大却不能消除这样一种现象，但是在 Nedich 的证明中并没有提及，事实上收敛性的满足只需要要求图结构的一致联合联通即可。

5 结论

本报告通过分别介绍一致性和分布式的协同优化算法的研究现状出发，首先探究了一个简单的多智能体系统的结构和问题，辨析了 Consensus、Average Consensus 的概念，给出了收敛和平均一致性的条件。

在分布式协同优化方面主要介绍了 DGD，ADMM，EXTRA，Push-sum Based 及其相关算法，用 MATLAB 分别对这些分布式协同优化算法进行的模拟仿真，对一些经典论文进行了结果复现。

6 附录

6.1 Convergence Analysis for Directed Networks

具体证明见 [10][11]。

[2]Open Source:https://github.com/MidoAssran/maopy

参考文献

[1] T. Vicsek, A. Czirok, E. Ben-Jacob, I. Cohen, and O. Sochet, "Novel type of phase transition in a system of self-driven particles," *Physical Review Letters*, vol. 75, no. 6, 1995.

[2] F. Dorfler and F. Bullo, "Synchronization and transient stability in power networks and non-uniform kuramoto oscillators," in *IEEE*, 2009.

[3] J. A. Fax, "Optimal and cooperative control of vehicle formations," Ph.D. dissertation, California Institute of Technology., 2002.

[4] Fax, J., Alexander, Murray, Richard, and M., "Information flow and cooperative control of vehicle formations." *IEEE Transactions on Automatic Control*, vol. 49, no. 9, pp. 1465–1476, 2004.

[5] R. O. Saber and R. M. Murray, "Consensus protocols for networks of dynamic agents," in *American Control Conference, 2003. Proceedings of the 2003*, 2003.

[6] R. Olfati-Saber and R. Murray, "Consensus problems in networks of agents with switching topology and time-delays," *IEEE Transactions on Automatic Control*, vol. 49, no. 9, pp. 1520–1533, 2004.

[7] A. Jadbabaie, J. Lin, and A. Morse, "Coordination of groups of mobile autonomous agents using nearest neighbor rules," *IEEE Transactions on Automatic Control*, vol. 48, no. 6, pp. 988–1001, 2003.

[8] L. Moreau, "Stability of multiagent systems with time-dependent communication links," *IEEE Transactions on Automatic Control*, vol. 50, no. 2, pp. 169–182, 2005.

[9] R. Wei and R. W. Beard, "Consensus seeking in multiagent systems under dynamically changing interaction topologies," *IEEE Transactions on Automatic Control*, vol. 50, no. 5, pp. 655–661, 2005.

[10] R. Olfati-Saber, J. A. Fax, and R. M. Murray, "Consensus and cooperation in networked multi-agent systems," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 215–233, 2007.

[11] Q. Li, "Cooperative control of multi-agent systems," *Dissertations Theses - Gradworks*, vol. 7, no. 8, pp. 822 – 831, 2009.

[12] A. Nedic and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE Transactions on Automatic Control*, vol. 54, no. 1, pp. 48–61, 2009.

[13] A. Nedic and A. Olshevsky, "Distributed optimization over time-varying directed graphs," 2014.

[14] W. Shi, Q. Ling, G. Wu, and W. Yin, "Extra: An exact first-order algorithm for decentralized consensus optimization," *SIAM Journal on Optimization*, vol. 25, no. 2, 2014.

[15] A. Nedich, A. Olshevsky, and W. Shi, "Achieving geometric convergence for distributed optimization over time-varying graphs," 2016.

[16] G. Scutari and Y. Sun, "Distributed nonconvex constrained optimization over time-varying digraphs," *Mathematical Programming*, vol. 176, no. 3, 2019.

[17] D. Hajinezhad, M. Hong, and A. Garcia, "Zone: Zeroth order nonconvex multi-agent optimization over networks," *IEEE Transactions on Automatic Control*, pp. 1–1, 2019.

[18] 谢佩, 游科友, 洪奕光, and 谢立华, "网络化分布式凸优化算法研究进展," 控制理论与应用, vol. 35, no. 7, p. 10, 2018.

[19] 杨涛 and 柴天佑, "分布式协同优化的研究现状与展望," 中国科学：技术科学, vol. 50, no. 11, p. 12, 2020.

[20] A. Nedi and L. Ji, "Distributed optimization for control," *Annual Review of Control Robotics and Autonomous Systems*, vol. 1, no. 1, pp. 77–103, 2018.

[21] A. Nedic, A. Olshevsky, and M. G. Rabbat, "Network topology and communication-computation tradeoffs in decentralized optimization," *Proceedings of the IEEE*, vol. 106, no. 5, pp. 953–976, 2017.

[22] S. Boyd and L. Vandenberghe, *Convex Optimization*. Convex Optimization, 2004.

[23] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations Trends in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2010.

[24] D. Kempe, A. Dobra, and J. Gehrke, "Gossip-based computation of aggregate information," in *44th Annual IEEE Symposium on Foundations of Computer Science, 2003. Proceedings.* IEEE, 2003, pp. 482–491.

[25] S. Martineau, T. Saffold, T. Chang, and H. Ronellenfitsch, "Enhancing synchronization by optimal correlated noise," 2022.

[26] T. C. Silva and L. Zhao, "Machine learning in complex networks || complex networks," vol. 10.1007/978-3-319-17290-3, no. Chapter 2, pp. 15–70, 2016.

[27] Y. Liu, Y. Kang, T. Zou, Y. Pu, Y. He, X. Ye, Y. Ouyang, Y.-Q. Zhang, and Q. Yang, "Vertical federated learning," *arXiv preprint arXiv:2211.12814*, 2022.

[28] E. Tolstaya, F. Gama, J. Paulos, G. Pappas, V. Kumar, and A. Ribeiro, "Learning decentralized controllers for robot swarms with graph neural networks," in *Conference on robot learning.* PMLR, 2020, pp. 671–682.

[29] Q. Li, F. Gama, A. Ribeiro, and A. Prorok, "Graph neural networks for decentralized multi-robot path planning," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).* IEEE, 2020, pp. 11 785–11 792.

[30] M. S. Assran, *Asynchronous subgradient push: Fast, robust, and scalable multi-agent optimization.* McGill University (Canada), 2018.