Distributed Consensus
CS289


Distributed Consensus

Average Consensus

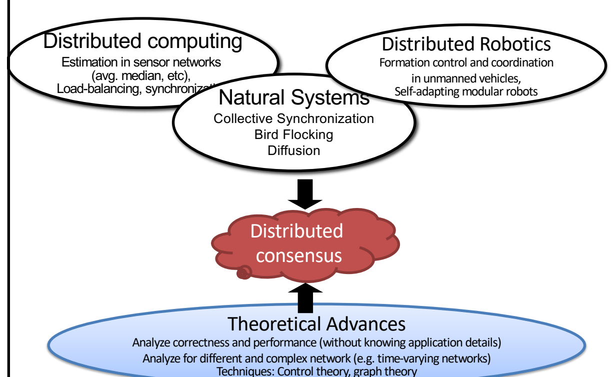
- The Average Height Problem
- The Equal Candy Problem

Distributed Consensus in "Real" Distributed Systems

- **Estimation** in distributed sensors (avg, median, product)
- **Load-balancing** in computer networks
- **Natural Phenomena** (diffusion, quorum-sensing)
- **Synchronization** (heartbeat, distributed antennas, wireless)
- **Flocking** and formation control (fish and birds, UAVs)
- **Environmentally-adaptive** robotic systems



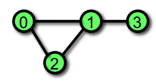

Why recognizing "similarity" matters



Outline

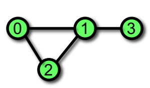
- Part I
  - We will look at the distributed consensus problem from the readings, and go through the mathematical analysis.
- Part II
  - I will show how ideas from distributed consensus have been used recently to show analytically why/how synchronization and flocking work

How do we solve the Problem?



- Problem:
  - Given a Graph G= (V,E) undirected, connected
  - Each node i in V has some initial value xi(0)
  - Each node i has some neighbors nbrs(i)
  - Nodes must cooperate to compute the average of initial values.

How do we solve the Problem?

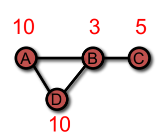


$x_i(0)$ = initial value
$x_i(t+1) = x_i(t) + \alpha\Delta x_i$
where $\Delta x_i = \sum[x_k(t) - x_i(t)]$
and k = nbrs(i)

Notice that its NOT OBVIOUS that this locally greedy (myopic) should work...

- Problem:
  - Given a Graph G= (V,E) undirected, connected
  - Each node i in V has some initial value xi(0)
  - Each node i has some neighbors nbrs(i)
  - Nodes must cooperate to compute the average of initial values.
- Answer:
  - *Intuition:* look at how you differ from your local neighbors, and move in the right direction to reduce your disagreement.

$x_i(0)$ = initial value
$x_i(t+1) = x_i(t) + \alpha\Delta x_i$
where $\Delta x_i = \sum[x_k(t) - x_i(t)]$
and k = nbrs(i)

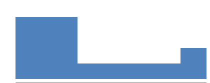


Globally, we can see that the average is 7 (i.e. 28/4)

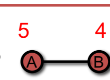
….But locally, for node C, its own value will first go down.

MYOPIC

Think of a line graph (continuous set of nodes) Information must travel, but it can also "slosh" around. How do we know it will ever settle?

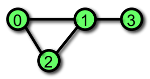


Lets say we let α=1
Then this will just flip flop



In fact, requires $\alpha < 1/dmax$
If α=1/2 or α=1/3, this example will work

Distributed Consensus



$x_i(0)$ = initial value
$x_i(t+1) = x_i(t) + \alpha\Delta x_i$
where $\Delta x_i = \sum[x_k(t) - x_i(t)]$
and $k = nbrs(i)$

- Interesting Properties of this Algorithm
  - Simple node behavior (Anonymous, leaderless, no params)
  - Self-maintaining (provides inherent robustness)
  - It works! (provably so if $\alpha < 1/dmax$)
- Provably Correct
  - Will converge to average, on any undirected connected graphs
  - Time depends on (a) distance to answer (b) network topology

How do we prove this?

Distributed Consensus

- From a local point of view (node)

$x_i(t+1) = x_i(t) + \alpha\Delta x_i$
$\Delta x_i = \sum[x_k(t) - x_i(t)]$ where k = nbrs(i)
$= (\sum x_k(t) - N_i . x_i(t))$ where N_i = number of nbrs

- From a global point of view (state matrix)

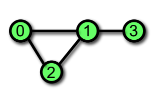
Distributed Consensus

- From a local point of view (node)

$x_i(t+1) = x_i(t) + \alpha\Delta x_i$
$\Delta x_i = \sum[x_k(t) - x_i(t)]$ where k = nbrs(i)
$= (\sum x_k(t) - N_i . x_i(t))$ where N_i = number of nbrs

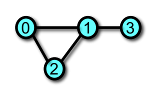
- From a global point of view (state matrix)

$$\begin{matrix}\Delta x0\\ \Delta x1\\ \Delta x2\\ \Delta x3\end{matrix} = \begin{vmatrix} -N0 & 1 & 1 & 0 \\ 1 & -N1 & 1 & 1 \\ 1 & 1 & -N2 & 0 \\ 0 & 1 & 0 & -N3 \end{vmatrix} \begin{matrix} x0(t)\\ x1(t)\\ x2(t)\\ x3(t)\end{matrix}$$



In matrix form: $\Delta X = -L\ X(t)$

Graph Laplacian



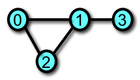
Turns out **"L"** is a famous matrix!!!

$\boxed{\Delta X = -L\ X(t)}$

$$\begin{vmatrix} -N0 & 1 & 1 & 0 \\ 1 & -N1 & 1 & 1 \\ 1 & 1 & -N2 & 0 \\ 0 & 1 & 0 & -N3 \end{vmatrix}$$

L = D-A
(Degree matrix - Adj matrix)

Graph Laplacian



Turns out **"L"** is a famous matrix!!!

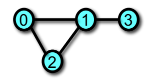
$\boxed{\Delta X = \mathbf{-\ L}\ X(t)}$

```
| -N0  1  1  0 |
|  1 -N1  1  1 |
|  1  1 -N2  0 |
|  0  1  0 -N3 |
```

L = D-A
(Degree matrix - Adj matrix)

Definition: Spectral properties of a matrix A
eigenvalues (scalar) = `v1 v2 v3 ... vn` (scalars)
eigenvectors (vector) = `e1 e2 e3 ... en`
For matrix A, **`A.e1 = v1.e1`** *(eigen decomposition)*

Graph Laplacian



Turns out **"L"** is a famous matrix!!!

$\boxed{\Delta X = \mathbf{-\ L}\ X(t)}$

```
| -N0  1  1  0 |
|  1 -N1  1  1 |
|  1  1 -N2  0 |
|  0  1  0 -N3 |
```

L = D-A
(Degree matrix - Adj matrix)

Definition: Spectral properties of a matrix A
eigenvalues (scalar) = `v1 v2 v3 ... vn` (scalars)
eigenvectors (vector) = `e1 e2 e3 ... en`
For matrix A, `A.e1 = v1.e1` *(eigen decomposition)*

If G is a undirected connected graph, then for L(G):
v1 = 0 and e1 = [a a a a a ...] and is unique (for undirected/connected)
v2 = algebraic connectivity and is > 0 (how "dense" the graph is)
the other v_s and e_s are also "magical"

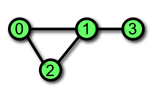
Back to Distributed Consensus

- From a local point of view (node)

```
xi(t+1) = xi(t) + αΔxi
Δxi= ∑[xk(t) – xi(t)]    where k = nbrs(i)
   = ∑xk(t) – Ni.xi(t)   where Ni = number of nbrs
```

- From a global point of view (state matrix)

```
Δx0     | -N0  1   1   0 |   x0(t)
Δx1  =  |  1  -N1  1   1 |   x1(t)
Δx2     |  1   1  -N2  0 |   x2(t)
Δx3     |  0   1   0 -N3 |   x3(t)
```



Captures the decentralized process: $\Delta X = \mathbf{-\ L}\ X(t)$

Proving the algorithm works

```
X(t+1) = X(t) + αΔX
```
where $\Delta X = \mathbf{-\ L}\ X(t)$

- Prove Correctness:
  - When it stops, the answer must be the average
  - It always stops, from any initial condition

Proving the algorithm works

$$\boxed{\begin{array}{l} X(t+1) = X(t) + \alpha\Delta X \\ \text{where } \Delta X = -\mathbf{L}\, X(t) \end{array}}$$

- Prove Correctness:
  - When it stops, the answer must be the average
  - It always stops, from any initial condition
- If G is undirected and connected
  1. **Consensus is a unique fixed point**
  2. **The Consensus is the average of initial values**
  3. **This is a stable fixed point**

Proving the algorithm works

$$\boxed{\begin{array}{l} X(t+1) = X(t) + \alpha\Delta X \\ \text{where } \Delta X = -\mathbf{L}\, X(t) \end{array}}$$

- Prove Correctness:
  - When it stops, the answer must be the average
  - It always stops, from any initial condition
- If G is undirected and connected
  1. **Consensus is a unique fixed point**
     - Stops when $\Delta X = -L\, X(t) = 0$
     - As we saw earlier, $v1 = 0$, $e1 = [a\ a\ a\ a\ a..]$ *(and $v2 > 0$)*
  2. **The Consensus is the average of initial values**
  3. **This is a stable fixed point**

Proving the algorithm works

$$\boxed{\begin{array}{l} X(t+1) = X(t) + \alpha\Delta X \\ \text{where } \Delta X = -\mathbf{L}\, X(t) \end{array}}$$

- Prove Correctness:
  - When it stops, the answer must be the average
  - It always stops, from any initial condition
- If G is undirected and connected
  1. **Consensus is a unique fixed point**
     - Stops when $-L\, X(t) = 0$
     - As we saw earlier, $v1 = 0$, $e1 = [a\ a\ a\ a\ a..]$ *(and $v2 > 0$)*
  2. **The Consensus is the average of initial values**
     - The process is conservative! The total mass (sum of values) remains constant at each time step. ($N.a$ = sum of initial values)
  3. **This is a stable fixed point**

Proving Stability

- Metric of "disagreement"

  (at time t, what's the system error?)

$$\boxed{M(t) = \sum (x_i(t) - avg)^2 \qquad \text{sum of squared error}}$$

- Prove that with each step, the dynamics of this system will cause this disagreement to be reduced
  - At each step, I reduce the disagreement by a fraction that depends on topology…

$$\boxed{M(t+1) \le M(t) - 2.v2.M(t)}$$

  - While initial convergence may be slow, reaction to perturbations is extremely fast!

Beyond Simple Consensus

Generalizable

- Directed graphs (strongly connected) [OS, T]
- Time-varying graphs [T, FL, OS]
- Gossip graphs [G]
- Distributed homeostasis (constraints) [F]
- *Applications:* Flocking, Synchronization, Vehicle formations, Sensor fusion, Self-adaptive robotic systems.

Citations

- [OS] Olfati-Saber, Murray, 2003
- [FL] Tanner, Jadbabaie, Pappas, 2003
- [G] Kempe et al 03, Xiao & Boyd 2004, Xiao et al 06
- [T] Luc Moreau, CDC 2003
- [F] Fax and Murray, 2004.

Outline

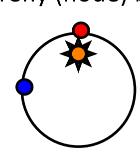
- Part I
  - We will look at the distributed consensus problem from the readings, and go through the math.
- Part II
  - I will show how ideas from distributed consensus have been used recently to show analytically why/how synchronization and flocking work

PART II

- **Synchronization**
  - Mirollo and Strogatz, SIAM 1990.
  - Izhikevich, IEEE Trans on Neural Networks,1999
  - Lucarelli and Wang, Sensys, 2004.
- **Flocking**
  - Reynolds (1987), Vicsek (1994)
  - Tanner, Jadbabaie, Pappas, CDC, 2003 (2)
  - Olfati-Saber, Murray, CDC 2003
  - *Review:* Olfati-Saber, Fax, Murray, 2007
- Both can be seen as a form of collective consensus

Mirollo and Strogatz Sync (1990)

How does a firefly (node) behave?



$o_i(t+1) = o_i(t) + \Delta o_i$

$\Delta o_i = 1/T + jump(o_i).p_i(t)$

Where $p_i(t) = 1$ if some neighbor fired ("pulse")
A simple jump function is $jump(o_i) = c.\ o_i$
One can understand how this behaves for 2 oscillators

Lucarelli and Wang, 2004

Local Point of View (slightly modified)

$\Delta o_i = 1/T + (c.\ o_i)\ .\ \sum p_k(t)$
where $p_k(t)$ = 1 if nbr k fired

If c is very small, then
Can applying Theorem by Izhikevich (1999)
can transform a pulse system to a continuous system

$\Delta O_i = e.\ 1/T.\ \sum (O_k(t) - O_i(t))$

Global Point of View

$\Delta O = -\alpha L\ O(t)$

Laplacian => Consensus!!
Speed of synchronization is affected by v2
(L&W proved a transformation for all jump functions that satisfy M&S criteria)

Flocking

- Reynold's Rules
  - Nearest neighbor behavior
  - Combination: cohesion, repulsion, alignment

  What do these rules guarantee?

- Tanner et al: What defines a Flock?
  - All flock members align their heading
  - All flock members achieve desired spacing
  - *A connected flock remains connected (not proved)*

- Alignment is like consensus
  - Problem is that the network changes at each step
  - Need to prove Consensus over time-varying topologies!!

Flocking Mathematically

r_i and v_i = position and velocity of node i
$v_i(t+1) = v_i(t) + \Delta v_i$

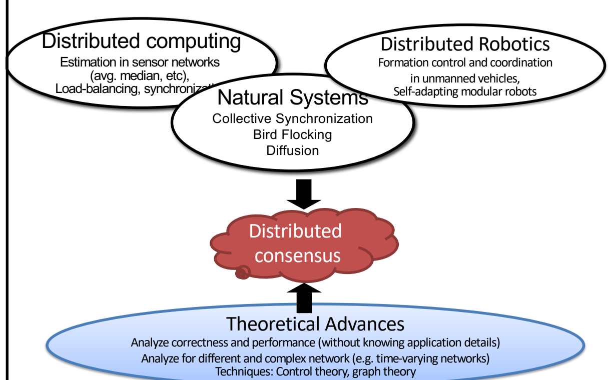
Δv_i= align-with-nbrs (consensus)
+ maintain "good" distance with nbrs

$\Delta v_i = \sum[v_k(t) - v_i(t)] + \sum \text{gradient } f(r_{ik})$
$f(r_{ik})$ = infinity if too close, 0 if perfect, high if too far

Globally
$\Delta v = -Lv(t)$ + other term

Problem is, the topology changes at every step!
old world: $v(t) = A^t\, v(0)$
new world: $v(t) = A(t).A(t-1).....A(1)A(0)\ v(0)$
But it still works!!!!

Why recognizing "similarity" matters



Swarm Intelligence