Distributed Minimum-Time Weight Balancing over Digraphs

Themistoklis Charalambous*, Christoforos N. Hadjicostis#, and Mikael Johansson*

*Department of Automatic Control, Royal Institute of Technology (KTH)
#Department of Electrical and Computer Engineering, University of Cyprus

International Symposium on Communications, Control, and Signal Processing (ISCCSP '14)
Athens, Greece, May 2014
1 Motivation - Introduction
2 Notation and mathematical preliminaries
3 Weight balancing in digraphs
4 Minimum-time weight balancing
5 Examples
6 Concluding remarks
Applications where weight balance plays a key role:

- Synchronization

- Average consensus via linear iterations (special case of synchronization without dynamics) – applications in multicomponent systems where one is interested in distributively averaging measurements, e.g., sensor networks

- Traffic-flow problems captured by $n$ junctions and $m$ one-way streets

- Stable flocking of agents with significant inertial effects

- Pinning control, optoelectronics, biology, ...

Finite-time algorithms are generally more desirable

- they converge in finite-time

- closed-loop systems under finite-time control usually demonstrate better disturbance rejection properties
Distributed system model

- Distributed systems conveniently captured by digraphs
  1. Components represented by vertices (nodes)
  2. Communication and sensing links represented by edges

Consider a network with nodes \( (v_1, v_2, \ldots, v_N) \)
E.g., sensors, robots, unmanned vehicles, resources, etc.

- Nodes can receive information according to (possibly directed) communication links
- Each node \( v_j \) has some initial value \( x_j[0] \) (could be belief, position, velocity, etc.)
Graph notation

- **Digraph** $G = (\mathcal{V}, \mathcal{E})$
  - Nodes (system components) $\mathcal{V} = \{v_1, v_2, \ldots, v_N\}$
  - Edges (directed communication links) $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ where $(v_j, v_i) \in \mathcal{E}$ iff node $v_j$ can receive information from node $v_i$
  - In-neighbors $\mathcal{N}^-_j = \{v_i \mid (v_j, v_i) \in \mathcal{E}\}$; in-degree $D^-_j = |\mathcal{N}^-_j|$
  - Out-neighbors $\mathcal{N}^+_j = \{v_l \mid (v_l, v_j) \in \mathcal{E}\}$; out-degree $D^+_j = |\mathcal{N}^+_j|$

- **Adjacency matrix** $A$: $A(j, i) = 1$ if $(v_j, v_i) \in \mathcal{E}$; $A(j, i) = 0$ otherwise

- **Undirected graph**: $(v_j, v_i) \in \mathcal{E}$ iff $(v_i, v_j) \in \mathcal{E}$ (bidirectional links)
  - In undirected graphs, we have (for each node $j$) $\mathcal{N}^+_j = \mathcal{N}^-_j$ and $D^+_j = D^-_j = D_j$; also, $A = A^T$

- **(Strongly) connected (di)graph** if for any $i, j \in \mathcal{V}, j \neq i$, there exists a (directed) path connecting them, i.e.,

$$v_i = v_{i_0} \rightarrow v_{i_1}, \ v_{i_1} \rightarrow v_{i_2}, \ldots, \ v_{i_{t-1}} \rightarrow v_{i_t} = v_j$$
Weight-balanced digraph:
Sum of weights on incoming links = Sum of weights on outgoing links

1. \( w_{ji} > 0 \) for each edge \( (v_j, v_i) \in E \);

2. \( w_{ji} = 0 \) if \( (v_j, v_i) \notin E \);

3. \( S_j^+ = S_j^- \land v_j \in V \), where \( S_j^- = \sum_{v_i \in \mathcal{N}_j^-} w_{ji} \), \( S_j^+ = \sum_{v_l \in \mathcal{N}_j^+} w_{lj} \)
Real-weight balancing:

- *Asymptotic* weight balancing; no known bound of convergence
- *Asymptotic* weight balancing; each agent is assumed to distinguish the information coming from other agents; a global stopping time is set to stop performing the balancing
  [Priolo et al, 2013]
- *Geometric* convergence rate with known rate of convergence
  [T.C. & C.N.H., 2013]

Integer-weight balancing:

- Finite number of steps; no known bound for convergence
  [B. Gharesifard and J. Cortés., 2012]
- Finite number of steps; upper bound of $O(n^7)$
  [Apostolos Rikos, T.C. & C.N.H., 2014]
Asymptotic weight balancing over digraphs

The algorithm (1/2)

- **Setting:** Nodes distributively adjust the weights of their outgoing links such that the digraph asymptotically becomes weight-balanced; they observe but cannot set the weights of their incoming links.

- Each node $v_j$ initializes the weights of all of its outgoing links to unity, i.e., $w_{lj}[0] = 1$, $\forall v_i \in \mathcal{N}_j^+$ (different initial weights also possible).

- Nodes enter an iterative stage where node $v_j$ performs the following steps:
  1. It computes its weight imbalance defined by
     \[ x_j[k] \triangleq S_j^−[k] − S_j^+[k] \]
  2. If $x_j[k]$ is positive (resp. negative), all the weights of its outgoing links are increased (resp. decreased) by an equal amount and proportionally to $x_j[k]$, specifically, $\forall v_i \in \mathcal{N}_j^+$,
     \[ w_{lj}[k + 1] = w_{lj}[k] + \beta_j \left( \frac{S_j^−[k]}{D_j^+} - w_{lj}[k] \right), \quad \beta_j \in (0, 1) \quad (1) \]
Intuition: we compare $S_j^-[k]$ with $S_j^+[k] = D_j^+ w_{ij}[k]$. If $S_j^+[k] > S_j^-[k]$ (resp. $S_j^+[k] < S_j^-[k]$), then the algorithm reduces (resp. increases) the weights on the outgoing links.

Proposition 1

If a digraph is strongly connected, the weight balancing algorithm asymptotically reaches a steady state weight matrix $W^*$ that forms a weight-balanced digraph, with geometric convergence rate equal to $R_\infty(P) = -\ln \delta(P)$, where

$$P_{ji} \triangleq \begin{cases} 1 - \beta_j, & \text{if } i = j, \\ \beta_j/D_j^+, & \text{if } v_i \in N_j^-, \end{cases}$$

and $\delta(P) \triangleq \max\{|\lambda| : \lambda \in \sigma(P)), \lambda \neq 1\}$.
Finite-time approaches for *undirected* graphs:

- *Minimum-time* average consensus [Y. Yuan *et al*, 2009]
  (associated with final value of linear iterations)

Finite-time approaches for *directed* graphs:

- *Minimum-time* average consensus in digraphs [T.C. *et al*, 2013]
  (used the same concept for final value of linear iterations)
Distributed *finite-time* methods in graphs

- **Finite-time approaches for undirected graphs:**
  - *Minimum-time* average consensus [Y. Yuan *et al*, 2009]
    (associated with final value of linear iterations)

- **Finite-time approaches for directed graphs:**
  - *Minimum-time* average consensus in digraphs [T.C. *et al*, 2013]
    (used the same concept for final value of linear iterations)

We propose an algorithm that combines *asymptotic weight-balancing* with *distributed final value of linear iterations* and has a convergence upper bound $O(2n)$. 
The minimal polynomial associated with the matrix pair \([P, e_j^\top]\), denoted by 
\[ q_j(t) = t^{M_j+1} + \sum_{i=0}^{M_j} \alpha_i^{(j)} t^i, \]
is the monic polynomial of minimum degree \(M_j + 1\) that satisfies 
\[ e_j^\top q_j(P) = 0. \]

Easy to show (e.g., using the techniques in [Y. Yuan et al, 2009]) that 
\[ \sum_{i=0}^{M_j+1} \alpha_i^{(j)} w_j[k + i] = 0, \quad \forall k \in \mathbb{Z}_+, \]

where \(\alpha_{M_j+1}^{(j)} = 1\). Denote \(z\)-transform of \(w_j[k]\) as 
\[ W_j(z) \triangleq \mathcal{Z}(w_j[k]). \] Then, 
\[ W_j(z) = \frac{\sum_{i=1}^{M_j+1} \alpha_i^{(j)} \sum_{\ell=0}^{i-1} w_j[\ell] z^{i-\ell}}{q_j(z)}, \]
where \(q_j(z)\) is the minimal polynomial of \([P, e_j^\top]\).
Define the following polynomial:

\[ p_j(z) \triangleq \frac{q_j(z)}{z - 1} \triangleq \sum_{i=0}^{M_j} \beta_i^{(j)} z^i \]

The application of the final value theorem (FVT) yields:

\[ \phi_w(j) = \lim_{k \to \infty} w_j[k] = \lim_{z \to 1} (z - 1) W_j(z) = \frac{w_{M_j}^T \beta_j}{1^T \beta_j} \]

where

- \( w_{M_j}^T = (w_j[0], w_j[1], \ldots, w_j[M_j]) \)
- \( \beta_j \) is the vector of coefficients of the polynomial \( p_j(z) \)
Define the following polynomial:

\[ p_j(z) \triangleq \frac{q_j(z)}{z - 1} \triangleq \sum_{i=0}^{M_j} \beta_i^{(j)} z^i \]

The application of the final value theorem (FVT) yields:

\[ \phi_w(j) = \lim_{k \to \infty} w_j[k] = \lim_{z \to 1} (z - 1) W_j(z) = \frac{w_{M_j}^T \beta_j}{1^T \beta_j} \]

where

- \( w_{M_j}^T = (w_j[0], w_j[1], \ldots, w_j[M_j]) \)
- \( \beta_j \) is the vector of coefficients of the polynomial \( p_j(z) \)

How can we obtain \( \beta_j \) in the computation of final values?
Consider the vectors of differences between $2k + 1$ successive discrete-time values of $w_j[k]$ at node $v_j$ and $x_j[k]$:

$$\overline{w}_{2k}^T = (w_j[1] - w_j[0], \ldots, w_j[2k+1] - w_j[2k])$$

Let us define their associated Hankel matrix:

$$\Gamma\{\overline{w}_{2k}^T\} \triangleq \begin{bmatrix} w_j[0] & w_j[1] & \ldots & w_j[k] \\ w_j[1] & w_j[2] & \ldots & w_j[k+1] \\ \vdots & \vdots & \ddots & \vdots \\ w_j[k] & w_j[k+1] & \ldots & w_j[2k] \end{bmatrix}$$

$\beta_j$ can be computed as the kernel of the first defective Hankel matrix for $\Gamma\{\overline{w}_{2k}^T\}$

For arbitrary initial conditions $w_0$, except a set of initial conditions with Lebesgue measure zero.
Minimum-time weight balancing in digraphs
Proposed algorithm

- **Input:** A strongly connected digraph $G(V, E)$

- **Data:** Successive observations for $w_j[k]$, $\forall v_j \in V$ using simultaneous iterations of (1) for *asymptotic weight-balancing* with initial conditions $w[0] = w_0$

- **Step 1:** Each node $v_j \in V$ stores the vectors of differences $\overline{w}_{Mj}^T$ between successive values of $w_j[k]$

- **Step 2:** Increase the dimension $k$ of $\Gamma\{\overline{w}_{Mj}^T\}$, until it loses rank; store the first defective matrix

- **Step 3:** The kernel $\beta_j = (\beta_0, \ldots, \beta_{Mj-1}, 1)^T$ of the first defective matrix gives the value $\phi_w$ which is the final value of iteration (1); i.e.,

$$w_j^* = \phi_w(j) = \frac{w_{Mj}^T \beta_j}{1^T \beta_j}$$
Example borrowed by [B. Gharesifard & J. Cortés, 2010]

Concerned with the absolute balance defined as

\[ \varepsilon[k] = \sum_{j=1}^{n} |x_j[k]| \]

If weight balance is achieved, then \( \varepsilon[k] = 0 \) and \( x_j[k] = 0, \forall v_j \in V \)

\[ W^* = \begin{bmatrix}
0 & 0 & 0.7143 & 0.7143 \\
1.4286 & 0 & 0 & 0 \\
0 & 1.4286 & 0 & 0 \\
0 & 0 & 0.7143 & 0
\end{bmatrix} \]
Comparisons with other works

Total imbalance vs Number of iterations for a random graph of 50 nodes

Algorithm 1 with $\beta_j = 0.5$ for all $v_j$
Rikos & Hadjicostis, CDC 2013
Gharesifard and Cortés, Allerton 2009
Algorithm 2

Average number of iterations needed for 100 graphs of size 10, 20, ..., 100 nodes
Concluding remarks and future directions

Conclusions:
- Proposed a distributed iterative algorithm, in which each node:
  - has knowledge of its *outgoing* links
  - reaches weight balancing in *directed* graphs in *minimum-time*
  - uses only output observations at each component (finite-time history of its own values)

Future work:
- Study weight balancing in a graph with time-varying delays
- Consider noisy output observations


Thank You!

Questions?

For more information:
themisc@kth.se
chadjic@ucy.ac.cy
mikaelj@kth.se