Polynomial Filtering for Fast Convergence in Distributed Consensus

Effrosyni Kokiopoulou, *Student Member, IEEE*, and Pascal Frossard, *Senior Member, IEEE*

***Abstract*—In the past few years, the problem of distributed consensus has received a lot of attention, particularly in the framework of ad hoc sensor networks. Most methods proposed in the literature address the consensus averaging problem by distributed linear iterative algorithms, with asymptotic convergence of the consensus solution. The convergence rate of such distributed algorithms typically depends on the network topology and the weights given to the edges between neighboring sensors, as described by the network matrix. In this paper, we propose to accelerate the convergence rate for given network matrices by the use of polynomial filtering algorithms. The main idea of the proposed methodology is to apply a polynomial filter on the network matrix that will shape its spectrum in order to increase the convergence rate. Such an algorithm is equivalent to periodic updates in each of the sensors by aggregating a few of its previous estimates. We formulate the computation of the coefficients of the optimal polynomial as a semidefinite program that can be efficiently and globally solved for both static and dynamic network topologies. We finally provide simulation results that demonstrate the effectiveness of the proposed solutions in accelerating the convergence of distributed consensus averaging problems.**

***Index Terms*—Distributed averaging, distributed consensus, polynomial filtering, sensor networks.**

I. Introduction

We consider the problem of distributed consensus [1] that has become recently very interesting especially in the context of ad hoc sensor networks. In particular, the problem of distributed average consensus has attracted a lot of research effort due to its numerous applications in diverse areas. A few examples include distributed estimation [2], distributed compression [3], coordination of networks of autonomous agents [4], and computation of averages and least squares in a distributed fashion (see, e.g., [5]–[8] and references therein).

In general the main goal of distributed consensus is to reach a global solution using only local computation and communication while staying robust to changes in the network topology. Given the initial values at the sensors, the problem of distributed averaging is to compute their average *at each sensor* using distributed linear iterations. Each distributed iteration involves local communication among the sensors. In particular, each sensor updates its own local estimate of the average by a weighted linear combination of the corresponding estimates of its neighbors. The weights that are represented in a network weight matrix W typically drive the importance of the measurements of the different neighbors.

One of the important characteristics of the distributed consensus algorithms is the rate of convergence to the asymptotic solution. In many cases, the average consensus solution can be reached by successive multiplications of W with the vector of initial sensor values. Furthermore, it has been shown in [5] that in the case of fixed network topology, the convergence rate depends on the second largest (in magnitude) eigenvalue of W, $\lambda_2(W)$. In particular, the convergence is faster when the value of $|\lambda_2(W)|$ is small. Similar convergence results have been proposed recently in the case of dynamic random network topology [9], [10], where the convergence rate is governed by the expected value of $|\lambda_2(W)|$.

The main research direction so far focuses on the computation of the optimal weights W that yield the fastest convergence rate to the consensus solution [5]–[7]. In this paper, we diverge from methods that are based on successive multiplications of W, and we rather allow the sensors to use their previous estimates, in order to accelerate the convergence rate. This is similar in spirit to the works proposed [15] and [16] that reach the consensus solution in a finite number of steps. They use, respectively, extrapolation methods and linear dynamical system formulation for fixed network topologies. In order to address more generic network topologies, we propose here to use a matrix polynomial p applied on the weight matrix W in order to shape its spectrum. Given the fact that the convergence rate is driven by $|\lambda_2(W)|$, it is therefore possible to impact on the convergence rate by careful design of the polynomial p. In the implementation viewpoint, working with $p(W)$ is equivalent to each sensor aggregating its value periodically using its own previous estimates. We further formulate the problem of the computation of the polynomial coefficients for both static and dynamic network topologies. We propose a methodology for the computation of the coefficients based on semidefinite programming (SDP), which results into an optimal solution in the case of static network topologies. In the case of dynamic topologies, we provide an effective suboptimal solution where the filter coefficients are computed based on the average weight matrix.

The rest of this paper is organized as follows. In Section II, we review the main convergence results of average consensus in both fixed and dynamic random network topologies. Next, in

Manuscript received February 25, 2008; revised August 06, 2008. First version published September 19, 2008; current version published January 06, 2009. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Hongbin Li. This work was supported in part by the Swiss NSF under NCCR IM2.

The authors are with the Signal Processing Laboratory (LTS4), Ecole Polytechnique Fédérale de Lausanne, Lausanne 1015, Switzerland (e-mail: effrosyni.kokiopoulou@epfl.ch; pascal.frossard@epfl.ch).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSP.2008.2006147

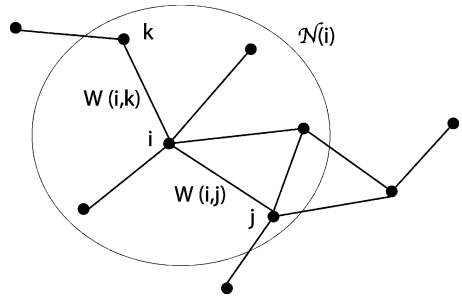

Fig. 1. Static network topology. The parameter $W(i,j) \neq 0$ describes the weight of the edge that permits the sensor i to communicate with its neighbor sensor j.

Section III, we introduce the polynomial filtering methodology and discuss its implementation for distributed consensus problems. We discuss the computation of the polynomial filter coefficients in Section IV for both static and dynamic network topologies. In Section V, we provide simulation results that verify the validity and the effectiveness of our method. Related work is finally presented in Section VI.

II. Convergence in Distributed Consensus Averaging

Let us first define formally the problem of distributed consensus averaging. Assume that initially each sensor i reports a scalar value $x_0(i) \in \mathbb{R}$. We denote by $x_0 = [x_0(1), \ldots, x_0(n)]^\top \in \mathbb{R}^n$ the vector of initial values on the network. Denote by

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_0(i) \tag{1}$$

the average of the initial values of the sensors. However, one rarely has a complete view of the network. The problem of distributed averaging therefore becomes typically to compute μ *at each sensor* by distributed linear iterations. In what follows, we review the main convergence results for distributed consensus algorithms on both fixed and dynamic network topologies.

A. Static Network Topology

We model the static network topology as an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with nodes $\mathcal{V} = \{1, \ldots, n\}$ corresponding to sensors. An edge $(i,j) \in \mathcal{E}$ is drawn if and only if sensor i can communicate with sensor j, as illustrated in Fig. 1. We denote the set of neighbors for node i as $\mathcal{N}_i = \{j | (i,j) \in \mathcal{E}\}$. Unless otherwise stated, we assume that each graph is simple, i.e., no loops or multiple edges are allowed.

In this paper, we consider distributed linear iterations of the following form:

$$x_{t+1}(i) = W(i,i)x_t(i) + \sum_{j \in \mathcal{N}_i} W(i,j)x_t(j) \tag{2}$$

for $i = 1, \ldots, n$, where $x_t(j)$ represents the value computed by sensor j at iteration t. Since the sensors communicate in each iteration t, we assume that they are synchronized. The parameters $W(i,j)$ denote the edge weights of $\mathcal{G}$. Since each sensor communicates only with its direct neighbors, $W(i,j) = 0$ when $(i,j) \notin \mathcal{E}$. The above iteration can be compactly written in the following form:

$$x_{t+1} = Wx_t \tag{3}$$

or, more generally

$$x_t = \left(\prod_{i=0}^{t-1} W\right) x_0 = W^t x_0. \tag{4}$$

We call the matrix W that gathers the edge weights $W(i,j)$ as the weight matrix. Note that W is a sparse matrix whose sparsity pattern is driven by the network topology. We assume that W is symmetric, and we denote its eigenvalue decomposition as $W = Q\Lambda Q^\top$. We also denote by $\lambda_2(W)$ the second largest (in magnitude) eigenvalue of W.

The distributed linear iteration given in (3) converges to the average for every x_0 if and only if

$$\lim_{t\to\infty} W^t = \frac{\mathbf{1}\mathbf{1}^\top}{n} \tag{5}$$

where $\mathbf{1}$ is the vector of ones [5]. Indeed, notice that in this case

$$x^* = \lim_{t\to\infty} x_t = \lim_{t\to\infty} W^t x_0 = \frac{\mathbf{1}\mathbf{1}^\top}{n} x_0 = \mu\mathbf{1}.$$

It has been shown that for fixed network topology, the convergence rate of (3) depends on the magnitude of the second largest eigenvalue $|\lambda_2(W)|$ [5]. The asymptotic convergence factor is defined as

$$r_{\text{asym}}(W) = \sup_{x_0 \neq \mu\mathbf{1}} \lim_{t\to\infty} \left(\frac{\|x_t - \mu\mathbf{1}\|_2}{\|x_0 - \mu\mathbf{1}\|_2}\right)^{1/t} \tag{6}$$

and the per-step convergence factor is written as

$$r_{\text{step}}(W) = \sup_{x_0 \neq \mu\mathbf{1}} \frac{\|x_{t+1} - \mu\mathbf{1}\|_2}{\|x_t - \mu\mathbf{1}\|_2}. \tag{7}$$

Furthermore, it has been shown that the convergence rate relates to the spectrum of W, as given by the following theorem [5].

Theorem 1: The condition in (5) holds if and only if

$$\mathbf{1}^\top W = \mathbf{1}^\top \tag{8}$$

$$W\mathbf{1} = \mathbf{1} \tag{9}$$

$$\rho\left(W - \frac{\mathbf{1}\mathbf{1}^\top}{n}\right) < 1 \tag{10}$$

where $\rho(\cdot)$ denotes the spectral radius of a matrix. Furthermore

$$r_{\text{asym}}(W) = \rho\left(W - \frac{\mathbf{1}\mathbf{1}^\top}{n}\right) \tag{11}$$

$$r_{\text{step}}(W) = \left\|W - \frac{\mathbf{1}\mathbf{1}^\top}{n}\right\|_2. \tag{12}$$

According to the above theorem, $(1/\sqrt{n})\mathbf{1}$ is a left and right eigenvector of W associated with the eigenvalue one, and the magnitude of all other eigenvalues is strictly less than one. Note finally that since W is symmetric, the asymptotic convergence

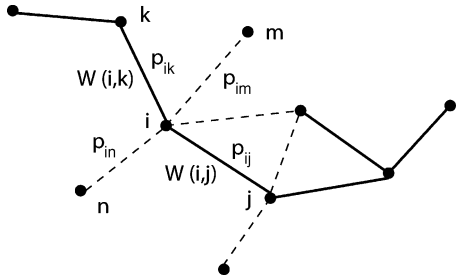

Fig. 2. Dynamic network topology. Each link is active with a probability p_{ij}. When a link is active, the parameter $W(i,j)$ describes the weight of the edge that permits the sensors i and j to communicate.

factor coincides with the per-step convergence factor, which implies that (11) and (12) are equivalent.

We give now an alternate proof of the above theorem that illustrates the importance of the second largest eigenvalue in the convergence rate. We expand the initial state vector x_0 to the orthogonal eigenbasis Q of W; that is

$$x_0 = \nu_1 \frac{\mathbf{1}}{\sqrt{n}} + \sum_{j=2}^{n} \nu_j q_j$$

where $\nu_1 = \langle (\mathbf{1}/\sqrt{n}), x_0 \rangle$ and $\nu_j = \langle q_j, x_0 \rangle$. We further assume that $\nu_1 \neq 0$. Then, (4) implies that

$$\begin{aligned} x_t = W^t x_0 &= W^t \left(\frac{\nu_1}{\sqrt{n}} \mathbf{1} + \sum_{j=2}^{n} \nu_j q_j \right) \\ &= \frac{\nu_1}{\sqrt{n}} \mathbf{1} + \sum_{j=2}^{n} \nu_j W^t q_j \\ &= \frac{\nu_1}{\sqrt{n}} \mathbf{1} + \sum_{j=2}^{n} \nu_j \lambda_j^t q_j. \end{aligned}$$

Observe now that if $|\lambda_j| < 1, \forall j \geq 2$, then in the limit, the second term in the above equation decays and

$$x^* = \lim_{t \to \infty} x_t = \frac{\nu_1}{\sqrt{n}} \mathbf{1} = \mu \mathbf{1}.$$

We see that the smaller the value of $|\lambda_2(W)|$, the faster the convergence rate. Analogous convergence results hold in the case of dynamic network topologies discussed next.

B. Dynamic Network Topology

Let us consider now networks with random link failures, where the state of a link changes over the iterations (see Fig. 2). In particular, we use the random network model proposed in [9] and [10]. We assume that the network at any arbitrary iteration t is $\mathcal{G}_t = (\mathcal{V}, \mathcal{E}_t)$, where $\mathcal{E}_t$ denotes the edge set at iteration t or, equivalently, at time instant t. Since the network is dynamic, the edge set changes over the iterations, as links fail at random. We assume that $\mathcal{E}_t \subseteq \mathcal{E}^*$, where $\mathcal{E}^* \subseteq \mathcal{V} \times \mathcal{V}$ is the set of realizable edges when there is no link failure.

We also assume that each link (i,j) fails with a probability $(1 - p_{ij})$, independently of the other links. Two random edge sets $\mathcal{E}_{t_1}$ and $\mathcal{E}_{t_2}$ at different iterations t_1 and t_2 are independent. The probability of forming a particular $\mathcal{E}_t$ is thus given by $\prod_{(i,j) \in \mathcal{E}_t} p_{ij}$. We define the matrix P as

$$P_{ij} = \begin{cases} p_{ij}, & \text{if } (i,j) \in \mathcal{E}^* \text{ and } i \neq j, \\ 0, & \text{otherwise.} \end{cases} \tag{13}$$

The matrix P is symmetric and its diagonal elements are zero, since it corresponds to a simple graph. It represents the probabilities of edge formation in the network, and the edge set $\mathcal{E}_t$ is therefore a random subset of $\mathcal{E}^*$ driven by the P matrix. Finally, the weight matrix W becomes dependent on the edge set since only the weights of existing edges can take nonzero values. Note finally that one may further introduce a probability q of network topology change in each iteration. In this case, q allows for controlling the dynamicity of the network. The network then follows the above random network model only when a change is triggered.

In the dynamic case, the distributed linear iteration of (2) becomes

$$x_{t+1}(i) = W_t(i,i) x_t(i) + \sum_{j \in \mathcal{N}_i} W_t(i,j) x_t(j) \tag{14}$$

or, in compact form

$$x_{t+1} = W_t x_t \tag{15}$$

where W_t denotes the weight matrix corresponding to the graph realization $\mathcal{G}_t$ of iteration t and $W_t(i,j)$ is its corresponding weight of entry (i,j). The iterative relation given by (15) can be written as

$$x_t = \left(\prod_{s=0}^{t-1} W_s \right) x_0.$$

Clearly, x_t now represents a stochastic process since the edges are drawn randomly. In what follows, when it is clear from the context that we refer to the random matrix W, we drop the subscript t for notational ease. The convergence rate to the consensus solution therefore depends on the behavior of the product $\prod_{s=0}^{t-1} W_s$. We say that the algorithm converges if

$$\forall x_0 \in \mathbb{R}^n, \ \lim_{t \to \infty} E \| x_t - \mu \mathbf{1} \| = 0. \tag{16}$$

We review now some convergence results from [10], which first shows the following.

Lemma 1: For any $x_0 \in \mathbb{R}^n$

$$\| x_{t+1} - \mu \mathbf{1} \| \leq \left(\prod_{s=0}^{t} \rho \left(W_s - \frac{\mathbf{1}\mathbf{1}^\top}{n} \right) \right) \| x_0 - \mu \mathbf{1} \|.$$

It leads to the following convergence theorem [10] for dynamic networks.

Theorem 2: If $E[\rho(W - (\mathbf{1}\mathbf{1}^\top/n))] < 1$, the vector sequence $\{x_t\}_{t=0}^{\infty}$ converges in the sense of (16).

We define the convergence factor in dynamic network topologies as

$$r_d(W) = E \left[\rho \left(W - \frac{\mathbf{1}\mathbf{1}^\top}{n} \right) \right].$$

This factor depends in general on the spectral properties of the induced network matrix and drives the convergence rate of (15). More generally, the authors in [17] show that $|\lambda_2(E[W])| < 1$ is also a necessary and sufficient condition for asymptotic (almost sure) convergence of the consensus algorithm in the case of random networks, where both network topology and weights are random (in particular independent identically distributed over time).

Finally, it is interesting to note that the consensus problem in a random network relates to gossip algorithms. Distributed averaging under the synchronous gossip constraint implies that multiple node pairs may communicate simultaneously only if these node pairs are disjoint. In other words, the set of links implied by the active node pairs forms a matching of the graph. Therefore, the distributed averaging problem described above is closely related to the distributed synchronous algorithm under the gossip constraint that has been proposed in [18, Sec. 3.3.2]. It has been shown in this case that the averaging time (or convergence rate) of a gossip algorithm depends on the second largest eigenvalue of a doubly stochastic network matrix.

III. Accelerated Consensus With Polynomial Filtering

A. Exploiting Memory

As we have seen above, the convergence rate of the distributed consensus algorithms depends in general on the spectral properties of an induced network matrix. This is the case for both fixed and dynamic network topologies. Most of the research work has been devoted to finding weight matrix W for accelerating the convergence to the consensus solution when sensors only use their current estimates. We choose a different approach where we exploit the memory of sensors, or the values of previous estimates in order to augment to convergence rate, since memory and computation use is cheaper than communication costs.

Therefore, we have proposed in our previous work [15] the scalar epsilon algorithm (SEA) for accelerating the convergence rate to the consensus solution. SEA belongs to the family of extrapolation methods for accelerating vector sequences, such as (3). These methods exploit the fact that the fixed point of the sequence belongs to the subspace spanned by any $\ell+1$ consecutive terms of it, where ℓ is the degree of the minimal polynomial of the sequence generator matrix (for more details, see [15] and references therein). SEA is a low-complexity algorithm, which is ideal for sensor networks and is known to reach the consensus solution in 2ℓ steps. However, ℓ is unknown in practice, so one may use all the available terms of the vector sequence. Hence, the memory requirements of SEA are $O(T)$, where T is the number of terms. Moreover, SEA assumes that the sequence generator matrix [e.g., W in the case of (3)] is fixed, so that it does not adapt easily to dynamic network topologies.

In this paper, we propose a more flexible algorithm based on the polynomial filtering technique. Polynomial filtering permits to "shape" the spectrum of a certain symmetric weight matrix in order to accelerate the convergence to the consensus solution. Similarly to SEA, it allows the sensors to use the value of their previous estimates. However, the polynomial filtering methodology introduced below presents three main advantages.

i) It is robust to dynamic topologies.
ii) It has explicit control on the convergence rate.
iii) Its memory requirements can be adjusted to the memory constraints imposed by the sensor.

B. Polynomial Filtering

Starting from a given (possibly optimal) weight matrix W, we propose the application of a polynomial filter on the spectrum of W in order to impact the magnitude of $\lambda_2(W)$ that mainly drives the convergence rate. Denote by $p_k(\lambda)$ the polynomial filter of degree k that is applied on the spectrum of W

$$p_k(\lambda) = \sum_{l=0}^{k} \alpha_l \lambda^l = \alpha_0 + \alpha_1 \lambda + \alpha_2 \lambda^2 + \ldots + \alpha_k \lambda^k. \quad (17)$$

Accordingly, the matrix polynomial is given as

$$p_k(W) = \sum_{l=0}^{k} \alpha_l W^l = \alpha_0 I + \alpha_1 W + \ldots + \alpha_k W^k. \quad (18)$$

Observe now that

$$\begin{aligned} p_k(W) &= p_k(Q \Lambda Q^\top) \\ &= \alpha_0 I + \alpha_1 (Q \Lambda Q^\top) + \ldots + \alpha_k (Q \Lambda^k Q^\top) \\ &= Q p_k(\Lambda) Q^\top \end{aligned} \quad (19)$$

which implies that the eigenvalues of $p_k(W)$ are simply the polynomial filtered eigenvalues of W, i.e., $p_k(\lambda_i(W))$, $i = 1, \ldots, n$.

In the implementation level, working on $p_k(W)$ implies a periodic update of the current sensor's value with a linear combination of its previous values. To see why this is true, we observe that

$$\begin{aligned} x_{t+k+1} &= p_k(W) x_t \quad (20) \\ &= \alpha_0 x_t + \alpha_1 W x_t + \ldots + \alpha_k W^k x_t \\ &= \alpha_0 x_t + \alpha_1 x_{t+1} + \ldots + \alpha_k x_{t+k}. \quad (21) \end{aligned}$$

A careful design of p_k may impact the convergence rate dramatically. Then, each sensor typically applies polynomial filtering for distributed consensus by following the main steps tabulated in Algorithm 1. In what follows, we propose different approaches for computing the coefficients α_l of the filter p_k.

C. Newton's Interpolating Polynomial

One simple and rather intuitive approach for the design of the polynomial $p_k(\lambda)$ is to use Hermite interpolation [33]. Recall that the objective is to dampen the smallest eigenvalues of W while keeping the eigenvalue one intact. Therefore, we assume (initially) that the spectrum of W lies in an interval $[a,1]$, and we impose smoothness constraints of p_k at the left endpoint a. In particular, the polynomial $p_k(\lambda) : [a, 1] \to \mathbb{R}$ that we seek will be determined by the following constraints:

$$p_k(a) = 0 \quad (22)$$

$$p_k(1) = 1 \quad (23)$$

$$p_k^{(i)}(a) = 0,\ i = 1, \ldots, k-1 \quad (24)$$

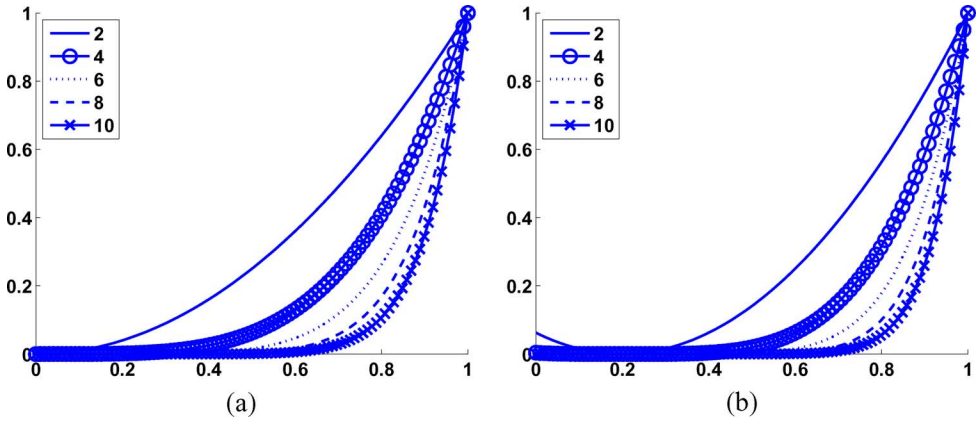

Fig. 3. Newton's polynomial of various degrees k. (a) $a = 0$ and (b) $a = 0.2$.

Algorithm 1 Polynomial Filtered Distributed Consensus

1: **Input**: polynomial coefficients $\alpha_0, \ldots, \alpha_k + 1$, tolerance ϵ.

2: **Output**: average estimate $\bar{\mu}$.

3: **Initialization**:

4: $\bar{\mu}_0 = x_0(i)$.

5: Set the iteration index $t = 1$.

6: **repeat**

7: **if** $\mathrm{mod}(t, k+1) == 0$ **then**

8: $x_t(i) = \alpha_0 x_{t-k-1}(i) + \alpha_1 x_{t-k}(i) + \alpha_2 x_{t-k+1}(i) + \ldots + \alpha_k x_{t-1}(i)$ {polynomial filter update}.

9: $x_t(i) = W(i,i)x_t(i) + \sum_{j \in \mathcal{N}_i} W(i,j)x_t(j)$.

10: **else**

11: $x_t(i) = W(i,i)x_{t-1}(i) + \sum_{j \in \mathcal{N}_i} W(i,j)x_{t-1}(j)$.

12: **end if**

13: Increase the iteration index $t = t + 1$.

14: $\bar{\mu}_t = x_t(i)$.

15: **until** $\bar{\mu}_t - \bar{\mu}_{t-1} < \epsilon$

where $p_k^{(i)}(a)$ denotes the ith derivative of $p_k(\lambda)$ evaluated at a. Intuitively, we seek a polynomial $p_k(\lambda)$ that will be zero at $\lambda = a$ [(22)], will be one at $\lambda = 1$ [(23)], and will be very small in the region close to the left endpoint a. The latter is imposed by the smoothness constraints (24). The dampening is achieved by imposing smoothness constraints of the polynomial on the left endpoint of the interval. The computed polynomial will have a degree equal to k. Finally, the coefficients of p_k that satisfy the above constraints can be computed by Newton's divided differences [33].

Fig. 3(a) shows an example of the shape of $p_k(\lambda)$ for $a = 0$ and different values of the degree k. As k increases, more smoothness constraints are imposed on a, and the dampening of the small eigenvalues becomes more effective. Interestingly, notice that since the smoothness constraints hold for free on the left of the interval $[a,1]$ as well, the filtering will work even in the case where a lies within the spectrum of W, provided that the magnitude of the filtered eigenvalues is strictly smaller than one (i.e., $|p_k(\lambda_j)| < 1$, $j \neq 1$). Therefore, we may drop the assumption that a encloses the spectrum of W. For instance, if the spectrum of W lies in [0,1], then one may choose $a = 0.2$. Fig. 3(b) shows the obtained polynomial filter in this case.

Furthermore, it is worth mentioning that the design of Newton's polynomial does not depend on the network topology or the network size. What is only needed is a left endpoint a, which roughly corresponds to the left extreme of the spectrum of W, as well as the desired degree k, which moreover may be imposed by memory constraints. This feature of Newton's polynomial is very interesting and is particularly appealing in the case of dynamic network topologies. Note, however, that the above polynomial design is mostly driven by intuitive arguments, which tend to obtain small eigenvalues for faster convergence. In the following section, we provide an alternative technique for computing the polynomial filter that optimizes the convergence rate.

IV. Polynomial Filter Design With Semidefinite Programming

A. Polynomial Filtering for Static Network Topologies

Given weight matrix W and a certain degree k, we are now interested in finding the polynomial that leads to the fastest convergence of (20), which we reproduce for convenience here

$$x_{t+k+1} = p_k(W)x_t.$$

Recall that $p_k(W) = \sum_{l=0}^{k} \alpha_l W^l$. Applying Theorem 1 to the above linear iteration, the optimal polynomial is the one that minimizes the spectral radius $\rho(p_k(W) - (\mathbf{1}\mathbf{1}^\top/n))$. Therefore, we need to solve an optimization problem where the optimization variables are the $k+1$ polynomial coefficients $\alpha_0, \ldots, \alpha_k$

and the objective function is the spectral radius of $p_k(W) - (\mathbf{1}\mathbf{1}^\top/n)$

Optimization problem : **OPT1**
$\min_{\alpha\in\mathbb{R}^{k+1}} \rho\left(\sum_{l=0}^{k}\alpha_l W^l - \frac{\mathbf{1}\mathbf{1}^\top}{n}\right)$
subject to
$\left(\sum_{l=0}^{k}\alpha_l W^l\right)\mathbf{1} = \mathbf{1}.$

Interestingly, the optimization problem OPT1 is convex. First, its objective function is convex, as stated in Lemma 2.

Lemma 2: For a given symmetric weight matrix W and degree k, $\rho(\sum_{l=0}^{k}\alpha_l W^l - (\mathbf{1}\mathbf{1}^\top/n))$ is a convex function of the polynomial coefficients α_ls.

Proof: Let $\beta, \gamma \in \mathbb{R}^{k+1}$ and $0 \leq \theta \leq 1$. Since W is symmetric, $\sum_{l=0}^{k}\alpha_l W^l$ is also symmetric. Hence, the spectral radius is equal to the matrix 2-norm. Thus, we have

$$\begin{aligned}
&\rho\left(\sum_{l=0}^{k}(\theta\beta_l + (1-\theta)\gamma_l)W^l - \frac{\mathbf{1}\mathbf{1}^\top}{n}\right)\\
&= \left\|\sum_{l=0}^{k}(\theta\beta_l + (1-\theta)\gamma_l)W^l - \frac{\mathbf{1}\mathbf{1}^\top}{n}\right\|_2\\
&= \left\|\theta\left(\sum_{l=0}^{k}\beta_l W^l - \frac{\mathbf{1}\mathbf{1}^\top}{n}\right)\right.\\
&\qquad \left.+(1-\theta)\left(\sum_{l=0}^{k}\gamma_l W^l - \frac{\mathbf{1}\mathbf{1}^\top}{n}\right)\right\|_2\\
&\leq \theta\left\|\sum_{l=0}^{k}\beta_l W^l - \frac{\mathbf{1}\mathbf{1}^\top}{n}\right\|_2 + (1-\theta)\left\|\sum_{l=0}^{k}\gamma_l W^l - \frac{\mathbf{1}\mathbf{1}^\top}{n}\right\|_2\\
&\leq \theta\rho\left(\sum_{l=0}^{k}\beta_l W^l - \frac{\mathbf{1}\mathbf{1}^\top}{n}\right)\\
&\quad + (1-\theta)\rho\left(\sum_{l=0}^{k}\gamma_l W^l - \frac{\mathbf{1}\mathbf{1}^\top}{n}\right)
\end{aligned}$$

which proves the lemma. ∎

In addition, the constraint of OPT1 is linear, which implies that the set of feasible α_ls is convex. As OPT1 minimizes a convex function over a convex set, the optimization problem is indeed convex.

In order to solve OPT1, we use an auxiliary variable η to bound the objective function, and then we express the spectral radius constraint as a linear matrix inequality (LMI). Thus, we need to solve the following optimization problem:

Optimization problem : **OPT2**
$\min_{\eta\in\mathbb{R},\alpha\in\mathbb{R}^{k+1}} \eta$
subject to
$-\eta I \preceq \sum_{l=0}^{k}\alpha_l W^l - \frac{\mathbf{1}\mathbf{1}^\top}{n} \preceq \eta I,$
$\left(\sum_{l=0}^{k}\alpha_l W^l\right)\mathbf{1} = \mathbf{1}.$

Recall that since W is symmetric, $p_k(W) = \sum_{l=0}^{k}\alpha_l W^l$ will be symmetric as well. Hence, the constraint $p_k(W)\mathbf{1} = \mathbf{1}$ is sufficient to ensure that $\mathbf{1}$ will be also a left eigenvector of $p_k(W)$. The spectral radius constraint

$$-\eta I \preceq p_k(W) - \frac{\mathbf{1}\mathbf{1}^\top}{n} \preceq \eta I$$

ensures that all the eigenvalues of $p_k(W)$, other than the first one, are less than or equal to η. Due to the LMI, the above optimization problem becomes equivalent to an SDP [19]. SDPs are convex problems and can be globally and efficiently solved. The solution to OPT2 is therefore computed efficiently in practice, where the SDP only has a moderate number of $k+2$ unknowns (including η).

B. Polynomial Filtering for Dynamic Network Topologies

We extend now the idea of polynomial filtering to dynamic network topologies. Theorem 2 suggests that the convergence rate in the random network topology case is governed by $E[\rho(W - (\mathbf{1}\mathbf{1}^\top/n))]$. Since W depends on a dynamic edge set, $p_k(W)$ now becomes stochastic. Following the same intuition as above, we could form an optimization problem, similar to OPT1, whose objective function would be $E[\rho(\sum_{l=0}^{k}\alpha_l W^l - (\mathbf{1}\mathbf{1}^\top/n))]$. Although this objective function can be shown to be convex, its evaluation is hard and typically requires several Monte Carlo simulations steps.

Recall that the convergence rate of $x_{t+1} = W_t x_t$ [see (15)] is related to the second largest (in magnitude) eigenvalue of $E[W]$, which is much easier to evaluate. Let $\overline{W}$ denote the average weight matrix $E[W]$. We then observe that

$$E\left[\rho\left(W - \frac{\mathbf{1}\mathbf{1}^\top}{n}\right)\right] \geq \rho\left(\overline{W} - \frac{\mathbf{1}\mathbf{1}^\top}{n}\right) \tag{25}$$

which is due to Lemma 2 and Jensen's inequality. The above inequality implies that in order to have a small $E[\rho(W - (\mathbf{1}\mathbf{1}^\top/n))]$, it is required that $\rho(\overline{W} - (\mathbf{1}\mathbf{1}^\top/n))$ be small. Additionally, the authors provide experimental evidence in [10], which indicates that $\rho(\overline{W} - (\mathbf{1}\mathbf{1}^\top/n))$ seems to be closely related to the convergence rate of the linear iteration $x_{t+1} = W_t x_t$.

Based on the above approximation, we propose an algorithm for polynomial filtering applied to dynamic network topologies. In particular, we propose to build our polynomial filter based on $\overline{W}$. Hence, we formulate the following optimization problem for computing the polynomial coefficients α_ls in the dynamic network topology case

Optimization problem : **OPT3**
$\min_{\eta\in\mathbb{R},\alpha\in\mathbb{R}^{k+1}} \eta$
subject to
$-\eta I \preceq \sum_{l=0}^{k}\alpha_l \overline{W}^l - \frac{\mathbf{1}^\top}{n} \preceq \eta I,$
$\left(\sum_{l=0}^{k}\alpha_l \overline{W}^l\right)\mathbf{1} = \mathbf{1}.$

OPT3 could be viewed as the analog of OPT2 for the case of dynamic network topology. The main difference is that we work on $\overline{W}$, whose eigenvalues can be easily obtained. Once $\overline{W}$ has

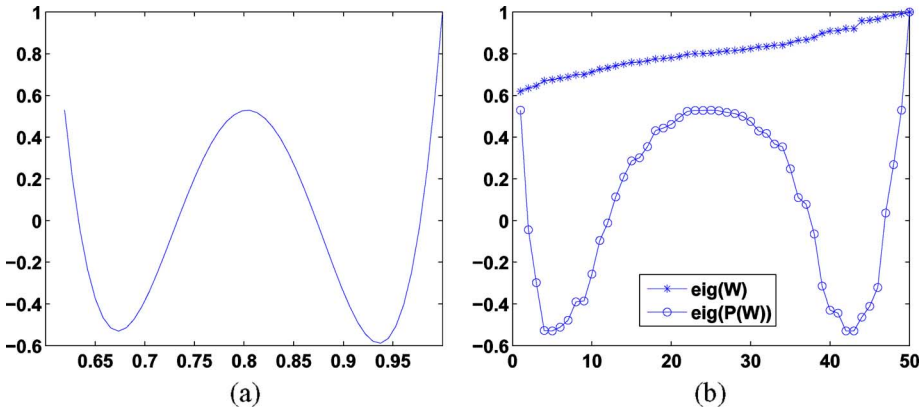

Fig. 4. Effect of polynomial filtering on the spectrum of the maximum-degree matrix of (26). (a) SDP polynomial filter $p_k(\lambda)$, $\lambda_{\min} \leq \lambda \leq 1$. (b) Effect on the spectrum $\lambda(W)$.

been computed, this optimization problem is solved efficiently by an SDP, similarly to the case of static networks.

Finally, one would expect that the bound (25) will be tighter when the successive matrices W_t are not very different. Indeed, in this case, one may argue that for infinitesimal changes of W, the function $\rho(W - (\mathbf{1}\mathbf{1}^\top/n))$ is locally linear and therefore the bound in (25) is tight. Note also that these network changing conditions correspond to most common scenarios. Furthermore, since $\overline{W}$ is symmetric, its eigenvalues are well behaved under small perturbations (due to the Bauer–Fike theorem [13]). Thus, one would additionally expect that minor changes in $\overline{W}$ will lead to small changes to the optimal polynomial coefficients.

V. Simulation Results

A. Setup

In this section, we provide simulation results that show the effectiveness of the polynomial filtering methodology. First we introduce a few weight matrices that have been extensively used in the distributed averaging literature. Suppose that $d(i)$ denotes the degree of the ith sensor. It has been shown in [5] and [6] that iterating (3) with the following matrices leads to convergence to $\mu\mathbf{1}$.

- *Maximum-degree* weights: The maximum-degree weight matrix is

$$W(i,j) = \begin{cases} \frac{1}{n}, & \text{if } (i,j) \in \mathcal{E} \\ 1 - \frac{d(i)}{n}, & i = j \\ 0, & \text{otherwise.} \end{cases} \tag{26}$$

- *Metropolis* weights: The metropolis weight matrix is

$$W(i,j) = \begin{cases} \frac{1}{1+\max\{d(i),d(j)\}}, & \text{if } (i,j) \in \mathcal{E} \\ 1 - \sum_{(i,k)\in\mathcal{E}} W(i,k), & i = j \\ 0, & \text{otherwise.} \end{cases} \tag{27}$$

- *Laplacian* weights: Suppose that A is the adjacency matrix of $\mathcal{G}$ and D is a diagonal matrix that holds the vertex degrees. The Laplacian matrix is defined as $L = D - A$ and the Laplacian weight matrix is defined as

$$W = I - \gamma L \tag{28}$$

where the scalar γ must satisfy $\gamma < 1/d_{\max}$ [5].

- *Optimal* weights, introduced in [5]: The optimal weight matrix W is determined by solving an SDP corresponding to the minimization of $\rho(W - (\mathbf{1}\mathbf{1}^\top/n))$ under the constraint that W is graph conformant (see [5] for more details). It should be noted that the number of unknowns in the case of optimal weights is equal to the number of nonzero entries of W (i.e., number of links in the network, which is on the order of $\mathrm{O}(n^2)$). This represents a limitation on the use of optimal weights for large or dynamic network topologies. On the contrary, the number of unknowns in our method is small, namely, $k+2$.

The sensor networks are built using the random geographic graph model [23]. In particular, we place n nodes uniformly distributed on the two-dimensional unit area. Two nodes are adjacent if their Euclidean distance is smaller than $r = \sqrt{\log n/n}$ in order to guarantee connectedness of the graph with high probability [23].

Note that in our SDP polynomial filtering method, for both fixed and dynamic network topology cases, the α_ls are computed offline assuming that W and, respectively, $E[W]$ are known *a priori*. Finally, the SDP programs for optimizing the polynomial filters are solved in Matlab using the SeDuMi [21] solver[1] and the YALMIP toolbox [22].

B. Static Network Topologies

First, we illustrate graphically the effect of polynomial filtering on the spectrum of W. We build a network of $n = 50$ sensors and apply polynomial filtering on the maximum-degree weight matrix W, given in (26). We use $k = 4$ and solve the optimization problem OPT2 using the maximum-degree matrix W as input. Fig. 4(a) shows the obtained polynomial filter $p_k(\lambda)$, when $\lambda \in [\lambda_{\min}(W), 1]$, where $\lambda_{\min}(W)$ denotes the smallest (algebraically) eigenvalue of W. Next, we apply the polynomial on W. Fig. 4(b) shows the spectrum of W before (star-solid line) and after (circle-solid line) polynomial filtering versus the vector index. Observe that polynomial filtering dramatically increases the spectral gap $1-|\lambda_2(W)|$, which further leads to accelerating the distributed consensus, as we show in the simulations that follow. This example illustrates the effect of a polynomial filter on the spectrum of a matrix. The detailed comparison

[1]Publically available at: http://www.sedumi.mcmaster.ca/.

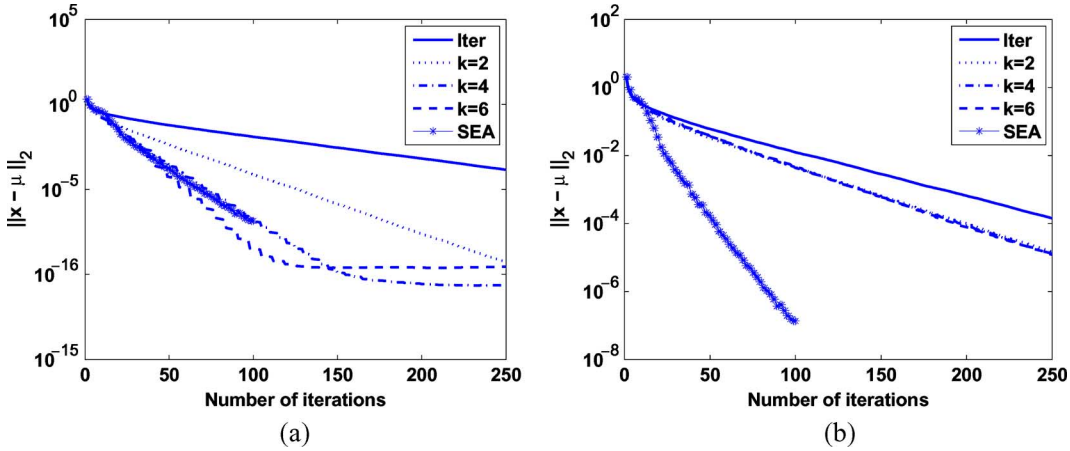

Fig. 5. Behavior of polynomial filtering for variable degree k on fixed topology using the Laplacian weight matrix. (a) SDP polynomial filtering and (b) Newton polynomial filtering.

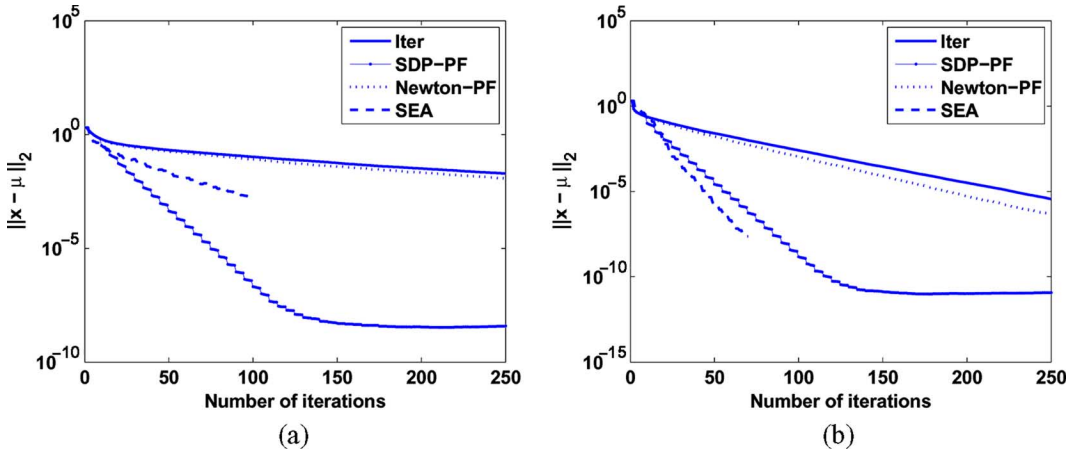

Fig. 6. Comparison between Newton's polynomial and SDP polynomial ($k = 4$) on fixed topology. (a) Maximum-degree weight matrix and (b) metropolis weight matrix.

between the convergence rates implied by the two matrices is further provided below.

We compare the performance of the different distributed consensus algorithms, with all the aforementioned weight matrices; that is, maximum-degree, metropolis, Laplacian, and optimal weight matrices for distributed averaging. We compare both Newton's polynomial and the SDP polynomial (obtained from the solution of OPT2) with the standard iterative method, which is based on successive iterations of (3). For the sake of completeness, we also provide the results of the scalar epsilon algorithm (SEA) that uses all previous estimates [15].

First, we explore the behavior of polynomial filtering methods under variable degree k from 2 to 6 with step 2. We use the Laplacian weight matrix for this experiment and $a = 0.3$. Fig. 5(a) and (b) illustrate the evolution of the average absolute error $||x_t - \mu\mathbf{1}||_2$ versus the iteration index t, for polynomial filtering with SDP and Newton's polynomials respectively. The curves that are shown, are averages over 500 random realizations of the sensor network and random (uniformly distributed) initial measurements. We also provide the curve of the standard iterative method as a baseline. Observe first that both polynomial filtering methods outperform the standard method by exhibiting faster convergence rates, across all values of k. Notice also that in the SDP method, the degree k clearly governs the convergence rate, since larger k implies more effective filtering and therefore faster convergence (i.e., the slope is steeper for larger k). Finally, the stagnation of the convergence process of the SDP polynomial filtering and large values of k is due to the limited accuracy of the SDP solver.

Next, we show the results obtained with the other two weight matrices. Fig. 6(a) and (b) shows the average convergence behavior of all methods for the maximum-degree and metropolis matrices, respectively, over 500 random experiments (sensor network and initial measurements). In both polynomial filtering methods, we use a representative value of k, namely, four. In the case of Newton's polynomial filtering, we use $a = 0.2$ for both weight matrices. Notice again that polynomial filtering accelerates the convergence of the standard iterative method (solid line). As expected, the optimal polynomial computed with SDP outperforms Newton's polynomial, which is based on intuitive arguments only.

Furthermore, we can see from Figs. 5 and 6 that in some cases the convergence rate is comparable for SEA and SDP polynomial filtering. Note, however, that the former uses all previous iterates, in contrast to the latter, which uses only the $k+1$ most recent ones. Hence, the memory requirements are smaller for

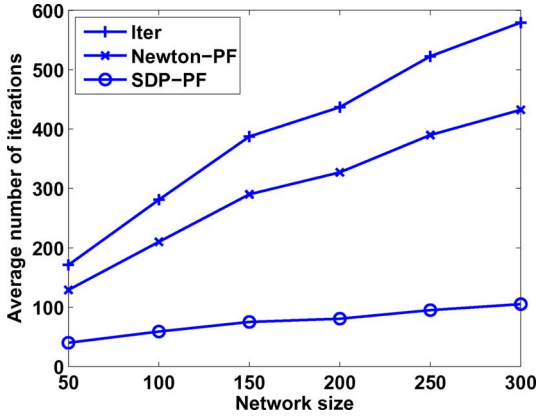

Fig. 7. Convergence time versus n with the Laplacian weight matrix.

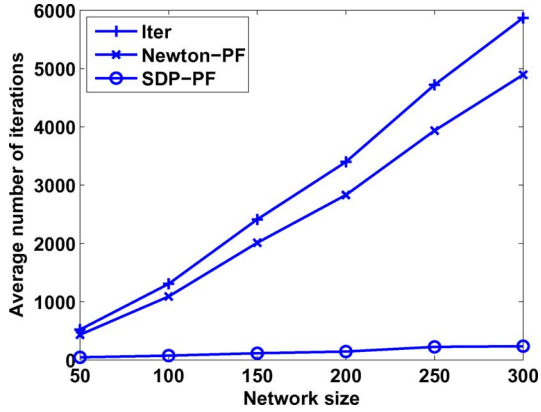

Fig. 9. Convergence time versus n with the maximum-degree weight matrix.

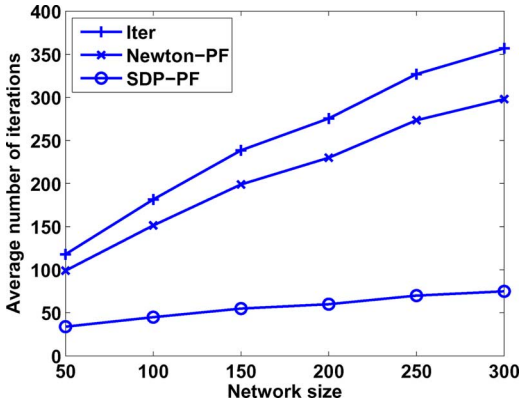

Fig. 8. Convergence time versus n with the metropolis weight matrix.

polynomial filtering, since they are directly driven by k. This moreover permits a more direct control on the convergence rate, as we have seen in Fig. 5. Interestingly, we see that the convergence process is smoother with polynomial filtering, which further permits easy extension to dynamic network topologies.

Next, we provide simulation results for different network sizes n, where n varies from 50 to 300 with step 50. In particular, we fix the tolerance δ of the absolute error $\|x_t - \mu\mathbf{1}\|_2$ to 10^{-3} and measure the average number of iterations needed by each method to reach the desired level δ of absolute error across different network sizes. This provides an estimate on the average convergence time achieved by each method over 100 random experiments. The degree k is set to four for both polynomial filtering methods. Figs. 7–9 show the obtained results for the Laplacian, metropolis, and maximum-degree weight matrices. Notice that the improvement of polynomial filtering methods on the convergence rate over the standard iterative method is pronounced in larger networks as well.

Finally, for the sake of completeness, we provide simulation results with the optimal weight matrix (see Section V-A). We apply our SDP polynomial filtering method on the optimal weight matrix and we report the results in Fig . 10(a) and (b). Fig. 10(a) shows the average convergence behavior over 100 random experiments for different values of k. It can be noted that the SDP polynomial filtering is able to accelerate the linear iteration, even when the optimal weights are used. Also, increasing k results in improvement on the convergence rate, but this seems to saturate for large values of k. Additionally, Fig. 10(b) shows the average number of iterations needed to obtain absolute error of $\delta = 10^{-3}$ when the network size n varies from ten to 50 with step ten. In this experiment, k was set to four, and the results are averages over 100 random experiments. We observe that the improvement offered by polynomial filtering stays consistent for different network sizes. Note that in this case we perform simulations for relatively small network sizes, since the solution of the optimization problem for the optimal weights is computationally intensive (recall that the number of unknowns is $\mathrm{O}(n^2)$). On the contrary, the number of unknowns in our SDP polynomial filtering method is only $k+2$ and does not depend on the network size.

C. Dynamic Network Topologies

We study now the performance of polynomial filtering for dynamic networks topologies. We build a sequence of random networks of $n = 50$ sensors. We assume that in each iteration, the network topology changes with probability q (independently from the previous iterations) and with probability $1 - q$, it remains the same as in the previous iteration. We compare all methods for different values of the probability q. We use the Laplacian weight matrix (28) and $a = 0.3$ in Newton's polynomial filtering. In the SDP polynomial filtering method, we solve the SDP program OPT3 (see Section IV-B). Fig. 11 shows the average performance of polynomial filtering for some representative values of the probability q and degree k. The average performance is computed using the median over 100 experiments. We have not reported the performance of the SEA algorithm since it is not robust to changes of the network topology.

Notice that when $k = 1$ (i.e., each sensor uses only its current value and the right previous one), polynomial filtering accelerates the convergence over the standard method for both small and large values of q. At the same time, it stays robust to network topology changes. Also, observe that in this case, the SDP polynomial outperforms Newton's polynomial. However, when $k = 2$, the roles between the two polynomial filtering methods change as the probability q increases. For instance, when $q = 0.8$, the SDP method even diverges. Thus, it appears that Newton's polynomial filtering is more robust to network changes than the SDP method. This is expected if we think that the coefficients of Newton's polynomial are computed

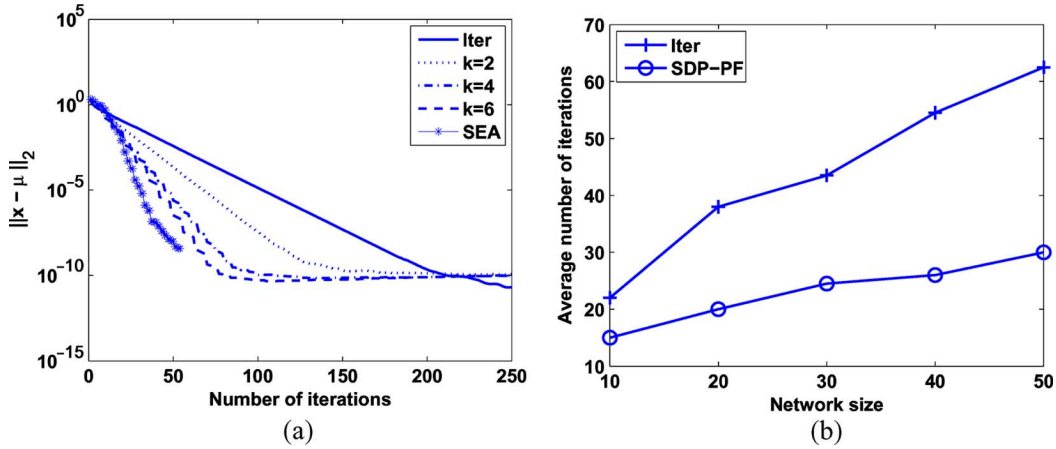

Fig. 10. Combination of SDP polynomial filtering with the optimal weight matrix. (a) SDP polynomial filtering applied to the optimal weight matrix and (b) average convergence time using the optimal weight matrix.

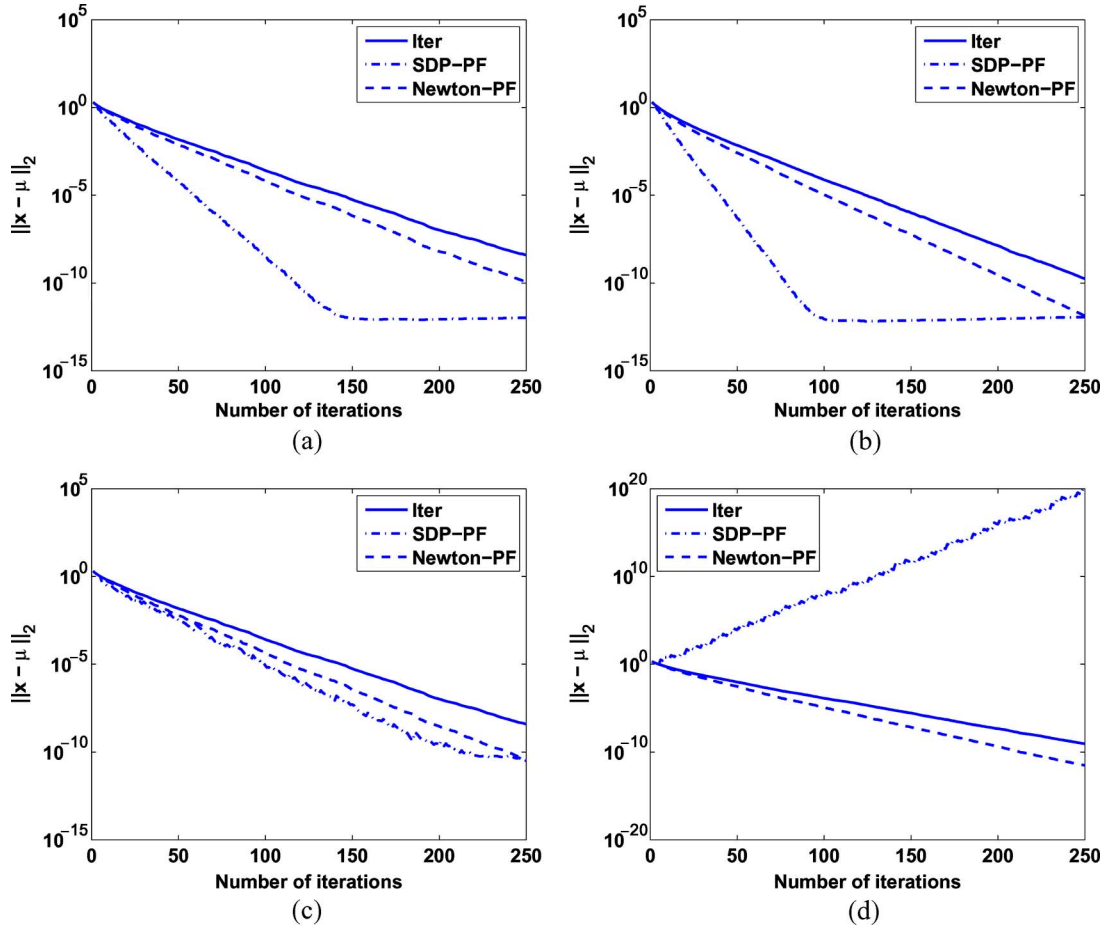

Fig. 11. Random network topology simulations with small k. q denotes the probability that the network changes at each iteration. (a) $k = 1, q = 0.2$; (b) $k = 1$, $q = 1$; (c) $k = 2, q = 0.2$; and (d) $k = 2, q = 0.8$.

using Hermite interpolation in a given interval and they do not depend on the specific realization of the underlying weight matrix. Thus, they are more generic than those of the SDP polynomial that takes $\overline{W}$ into account, and therefore they are less sensitive to the actual topology realization. Algorithms based on optimized polynomial filtering become inefficient in a highly dynamic network, whose topology changes very frequently.

We perform further simulations that correspond to larger values of k in order to investigate the behavior of Newton's polynomial filtering. Fig. 12 shows the obtained results when $k = 5, 6$, and 7. The experimental results suggest that polynomial filtering with Hermite interpolation stays robust to network changes when $k = 5$ and $k = 6$. This is even the case for $k = 7$ and small q. Therefore, we can conclude that

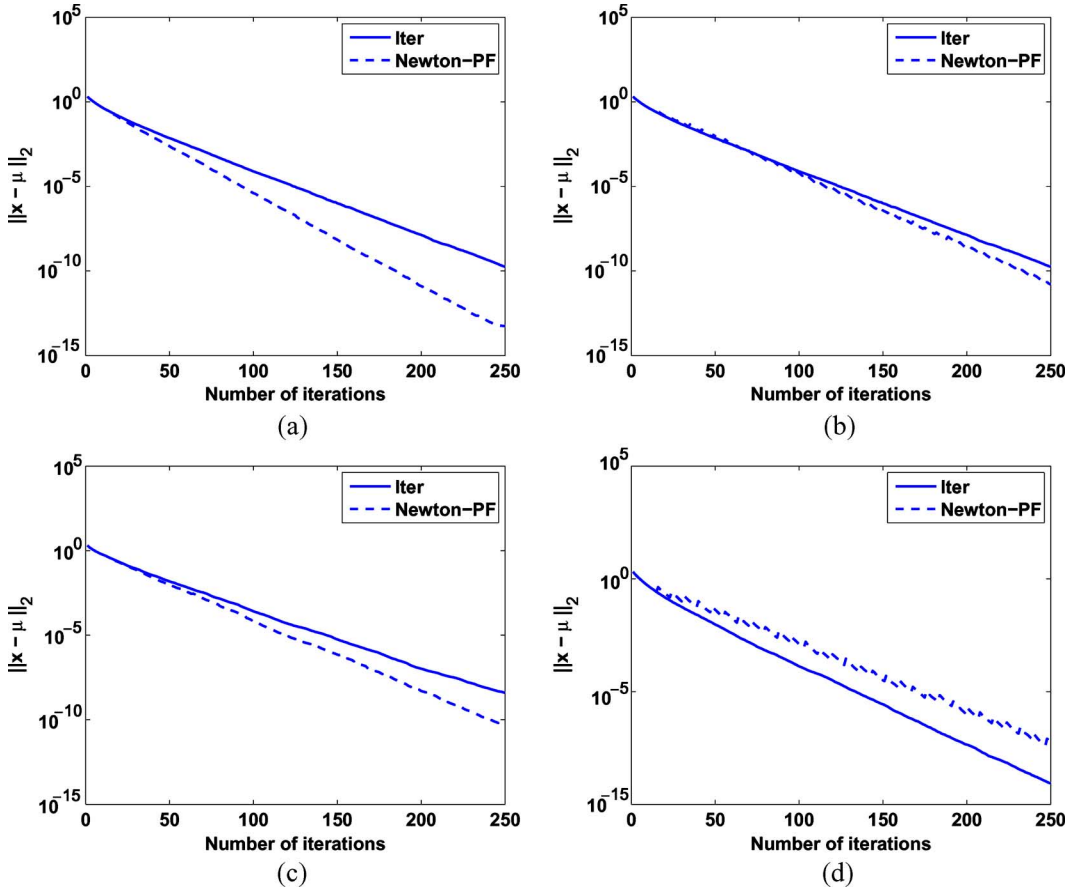

Fig. 12. Random network topology simulations with larger k. q denotes the probability that the network changes at each iteration. (a) $k = 5$, $q = 1$; (b) $k = 6$, $q = 1$; (c) $k = 7$, $q = 0.2$; and (d) $k = 7$, $q = 0.8$.

Newton's polynomial filtering is robust to network changes for moderate values of k.

VI. Related Work

Several works have studied the convergence rate of distributed consensus algorithms. In particular, the authors in [5] and [9], [10] have shown that the convergence rate depends on the second largest eigenvalue of the network weight matrix for fixed and random networks, respectively. They both use semidefinite programs to compute the optimal weight matrix, and the optimal topology. We have illustrated in the simulation section that our polynomial filtering methodology can be combined with optimal weight matrices and result in even faster convergence rates.

Recently, a few papers have appeared that focus into accelerating the convergence rate to the consensus solution via lifted Markov chains. The main idea of lifting is to distinguish the graph nodes from the states of the Markov chain and to "split" the states into virtual states that are connected in such a way that permits faster mixing. The lifted graph is then "projected" back to the original graph, where the dynamics of the lifted Markov chain are simulated subject to the original graph topology. We mention the work in [26], which proposes a fast distributed averaging algorithm for geographic random graphs. In particular, the location information of the sensors is assumed to be known and is used in order to construct a nonreversible lifted Markov chain that mixes faster than corresponding reversible chains. However, since the Markov chain is nonreversible, the stationary distribution is not uniform anymore. The authors introduce weights in order to overcome this problem, which in turn increases the communication cost. Moreover, the extension of the proposed methodology to dynamic network topologies is not straightforward. Along the same lines of lifting, Jung *et al.* [28] used nonreversible lifted Markov chains for fast gossip. The authors use the lifting scheme of [29] and propose a deterministic gossip algorithm based on a set of disjoint maximal matchings in order to simulate the dynamics of the lifted Markov chain.

In [27], the authors propose a cluster-based distributed averaging algorithm, applicable to both fixed linear iteration and random gossiping. By clustering the nodes, one may construct an overlay graph that is better connected, relative to the original graph; hence, the random walk on the overlay graph mixes faster than the corresponding walk on the original graph. The improvement in mixing time comes nevertheless at the cost of performing the clustering and forming the overlay graph. The extension of this methodology to dynamic network topologies is also not easy. On the contrary, our polynomial filtering methodology is flexible, and as we have seen in the simulations section, it can be applied successfully in both fixed and dynamic network topologies.

Recent works have also proposed to use the sensors' memory, such as [11] and [12]. The main idea in [11] is to update the value

of each sensor by a convex combination of the value obtained by linear prediction on its own previous values and the standard consensus operation (i.e., aggregating neighbors' values). Although the authors derive the optimal convex combination parameter, the linear prediction coefficients are not optimized. This is to be contrasted to our SDP polynomial filtering methodology, where we establish the optimality of the polynomial coefficients. In a different context, the effect of quantization has been studied in [12] for average consensus problems. The authors propose scalar quantizers based on predictive coding. The predictive coding scheme relies upon linear prediction using the past values of the sensors.

For the sake of completeness, we further mention in the sequel a few representative papers that address the general problem of consensus in sensor networks. The approaches based on agreement algorithms or gossiping are quite different from the work proposed in this paper. Olshevsky and Tsitsiklis in [8] propose two consensus algorithms for fixed network topologies, which build on the "agreement algorithm." The proposed algorithms make use of spanning trees and the authors bound their worst case convergence rate. For dynamic network topologies, they propose an algorithm that builds on a previously known distributed load balancing algorithm. In this case, the authors show that the algorithm has a polynomial bound on the convergence time (ϵ-convergence).

The authors in [30] study the convergence properties of agreement over random networks following the Erdős and Rényi random graph model. According to this model, each edge of the graph exists with probability p, independently of other edges and the value of p is the same for all edges. By agreement, we typically consider the case where all nodes of the graph agree on a particular value. The authors employ results from stochastic stability in order to establish convergence of agreement over random networks. Also, it is shown that the rate of convergence is governed by the expectation of an exponential factor, which involves the second smallest eigenvalue of the Laplacian of the graph.

Gossip algorithms have also been applied successfully to solving distributed averaging problems. In [18], the authors provide convergence results on randomized gossip algorithm in both synchronous and asynchronous settings. Based on the obtained results, they optimize the network topology (edge formation probabilities) in order to maximize the convergence rate of randomized gossip. This optimization problem is formulated as a semidefinite program. In a recent study, the authors in [24] and [25] have been able to improve the message complexity of the standard gossip protocols in cases where the sensors know their geometric positions. The main idea is to exploit geographic routing in order to aggregate values among random nodes that are far away in the network.

Finally, even if we have mostly considered synchronous algorithms in this paper, it is worth mentioning that the authors in [31] propose two asynchronous algorithms for distributed averaging. The first algorithm is based on blocking (that is, when two nodes update their values they block until the update has been completed) and the other algorithm drops the blocking assumption. The authors show the convergence of both algorithms under very general asynchronous timing assumptions. Along the lines of asynchronous algorithms, we mention also the *consensus propagation* framework proposed in [32], which is an asynchronous distributed protocol that is a special case of belief propagation. In the case of singly connected graphs (i.e., connected with no loops), synchronous consensus propagation converges in a number of iterations that is equal to the diameter of the graph. The authors provide convergence analysis for regular graphs.

VII. Conclusion

In this paper, we proposed a polynomial filtering methodology in order to accelerate distributed average consensus in both fixed and dynamic network topologies. The main idea of polynomial filtering is to shape the spectrum of the polynomial weight matrix in order to minimize its second largest eigenvalue and subsequently increase the convergence rate. We have constructed semidefinite programs to optimize the polynomial coefficients in both static and dynamic networks. Simulation results with several common weight matrices have shown that the convergence rate is much higher than for state-of-the-art algorithms in most scenarios, except in the specific case of highly dynamic networks and small memory sensors.

Acknowledgment

E. Kokiopoulou would like to thank Prof. Y. Saad for the valuable and insightful discussions on polynomial filtering.

References

[1] D. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*. Englewood Cliffs, NJ: Prentice-Hall, 1989.

[2] I. D. Schizas, A. Ribeiro, and G. B. Giannakis, "Consensus in ad hoc WSNs with noisy links—Part I: Distributed estimation of deterministic signals," *IEEE Trans. Signal Process.*, vol. 56, pp. 350–364, Jan. 2008.

[3] M. Rabbat, J. Haupt, A. Singh, and R. Nowak, "Decentralized compression and predistribution via randomized gossiping," in *Proc. 5th ACM Int. Conf. Inf. Process. Sensor Netw. (IPSN)*, Apr. 2006, pp. 51–59.

[4] V. D. Blondel, J. M. Hendrickx, A. Olshevsky, and J. N. Tsitsiklis, "Convergence in multiagent coordination, consensus and flocking," in *Proc. IEEE Conf. Decision Contr., Eur. Contr. Conf.*, Dec. 2005, pp. 2996–3000.

[5] L. Xiao and S. Boyd, "Fast linear iterations for distributed averaging," *Syst. Contr. Lett.*, no. 53, pp. 65–78, Feb. 2004.

[6] L. Xiao, S. Boyd, and S. Lall, "A scheme for robust distributed sensor fusion based on average consensus," in *Proc. Int. Conf. Inf. Process. Sensor Netw.*, Los Angeles, CA, Apr. 2005, pp. 63–70.

[7] L. Xiao, S. Boyd, and S. Lall, "A space-time diffusion scheme for peer-to-peer least-squares estimation," in *Proc. 5th Int. Conf. Information Processing in Sensor Networks (IPSN)*, Apr. 2006, pp. 168–176.

[8] A. Olshevsky and J. Tsitsiklis, "Convergence rates in distributed consensus and averaging," in *Proc. IEEE Conf. Decision Contr.*, San Diego, CA, Dec. 2006.

[9] S. Kar and J. M. F. Moura, "Distributed average consensus in sensor networks with random link failures," in *IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, Apr. 2007.

[10] S. Kar and J. M. F. Moura, "Sensor networks with random links: Topology design for distributed consensus," *IEEE Trans. Signal Process.*, vol. 56, no. 7, pp. 3315–3326, Jul. 2008.

[11] B. N. Oreshkin, T. C. Aysal, and M. J. Coates, "Distributed average consensus with increased convergence rate," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, Apr. 2008.

[12] M. E. Yildiz and A. Scaglione, "Differential nested lattice encoding for consensus problems," in *Proc. ACM Int. Conf. Inf. Process. Sensor Netw. (IPSN)*, Apr. 2007.

[13] G. H. Golub and C. Van Loan, *Matrix Computations*, 3rd ed. Baltimore, MD: John Hopkins Univ. Press, 1996.

[14] E. Kokiopoulou and Y. Saad, "Polynomial filtering in latent semantic indexing for information retrieval," in *Proc. 27th ACM-SIGIR Conf. Res. Develop. Inf. Retrieval*, 2004.
[15] E. Kokiopoulou and P. Frossard, "Accelarating distributed consensus using extrapolation," *IEEE Signal Process. Lett.*, vol. 14, pp. 665–668, Oct. 2007.
[16] S. Sundaram and C. N. Hadjicostis, "Distributed consensus and linear functional calculation in networks: An observability perspective," in *Proc. 6th ACM Int. Conf. Inf. Process. Sensor Netw. (IPSN)*, Apr. 25–27, 2007.
[17] A. Tahbaz-Salehi and A. Jadbabaie, "On consensus over random networks," in *Proc. 44th Annu. Allerton Conf. Commun., Contr. Comput.*, Sep. 2006, pp. 1315–1321.
[18] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Randomized gossip algorithms," *IEEE Trans. Inf. Theory*, vol. 52, pp. 2508–2530, Jun. 2006.
[19] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.
[20] Y. Saad, *Iterative Methods for Sparse Linear Systems*, 2nd ed. Philadelphia, PA: SIAM, 2003.
[21] J. F. Sturm, "Implementation of interior point methods for mixed semidefinite and second order cone optimization problems," in *Econ-Papers 73*, Aug. 2002, Center for Economic Research, Tilburg Univ..
[22] J. Löfberg, "YALMIP: A toolbox for modeling and optimization in MATLAB," in *Proc. CACSD Conf.*, Taipei, Taiwan, R.O.C., 2004.
[23] P. Gupta and P. R. Kumar, "The capacity of wireless networks," *IEEE Trans. Inf. Theory*, vol. 46, pp. 388–404, Mar. 2000.
[24] A. Dimakis, A. D. Sarwate, and M. J. Wainwright, "Geographic gossip: Efficient averaging for sensor networks," *IEEE Trans. Signal Process.*, to be published.
[25] F. Bénézit, A. G. Dimakis, P. Thiran, and M. Vetterli, "Gossip along the way: Order-optimal consensus through randomized path averaging," in *Proc. 45th Annu. Allerton Conf. Communication, Control, Computing*, Sep. 2007.
[26] W. Li and H. Dai, "Location-aided fast distributed consensus," *IEEE Trans. Inf. Theory*, Jun. 2007, submitted for publication.
[27] W. Li and H. Dai, "Cluster-based fast distributed consensus," in *IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Apr. 2007.
[28] K. Jung and D. Shah, "Fast gossip via nonreversible random walk," in *Proc. IEEE Inf. Theory Workshop (ITW)*, Mar. 2006.
[29] F. Chen, L. Lovász, and I. Pak, "Lifting Markov chains to speed up mixing," in *Proc. 31st Annu. ACM Symp. Theory Comput. (STOC '99)*, 1999, pp. 275–281.
[30] Y. Hatano and M. Mesbahi, "Agreement over random networks," in *IEEE Conf. Decision Contr.*, Dec. 2004, vol. 2, pp. 2010–2015.
[31] M. Mehyar, D. Spanos, J. Pongsajapan, S. H. Low, and R. M. Murray, "Asynchronous distributed averaging on communication networks," *IEEE/ACM Trans. Netw.*, vol. 15, pp. 512–520, Jun. 2007.
[32] C. C. Moallemi and B. V. Roy, "Consensus propagation," *IEEE Trans. Inf. Theory*, vol. 52, pp. 4753–4766, Nov. 2006.
[33] R. Schilling and S. Harris, *Applied Numerical Methods for Engineers Using MATLAB and C*, 3rd ed. New York: McGraw-Hill, 1998.

Effrosyni Kokiopoulou (S'05) received the Diploma degree in engineering from the Computer Engineering and Informatics Department, University of Patras, Greece, in 2002 and the M.Sc. degree in computer science from the Computer Science and Engineering Department, University of Minnesota, Minneapolis, in 2005. She is currently working towards the Ph.D. degree from the Swiss Federal Institute of Technology (EPFL), Lausanne.

In September 2005, she joined the Signal Processing Laboratory (LTS4) of the Electrical Engineering Institute, EPFL. Her doctoral work is under the supervision of Prof. Pascal Frossard. Her research interests include multimedia data mining, machine learning, computer vision, numerical linear algebra, and distributed data analysis.

Pascal Frossard (S'96-M'01–SM'04) received the M.S. and Ph.D. degrees in electrical engineering from the Swiss Federal Institute of Technology (EPFL), Lausanne, in 1997 and 2000, respectively.

Between 2001 and 2003, he was a Member of Research Staff with the IBM T. J. Watson Research Center, Yorktown Heights, NY, where he worked on media coding and streaming technologies. Since 2003, he has been an Assistant Professor at EPFL, where he heads the Signal Processing Laboratory (LTS4). His research interests include image representation and coding, nonlinear representations, visual information analysis, joint source and channel coding, and multimedia communications. He has been a member of the organizing or technical program committees of numerous conferences.

Dr. Frossard has been the General Chair of IEEE ICME 2002 and Packet Video 2007. He has been an Associate Editor of the IEEE TRANSACTIONS ON MULTIMEDIA since 2004 and of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY since 2006. He has been an elected member of the IEEE Image and Multidimensional Signal Processing Technical Committee since 2007, the IEEE Visual Signal Processing and Communications Technical Committee since 2006, and the IEEE Multimedia Systems and Applications Technical Committee since 2005. He has served as Vice-Chair of the IEEE Multimedia Communications Technical Committee (2004–2006) and as a member of the IEEE Multimedia Signal Processing Technical Committee (2004–2007). He received the Swiss NSF Professorship Award in 2003 and the IBM Faculty Award in 2005.