Proceedings of the
47th IEEE Conference on Decision and Control
Cancun, Mexico, Dec. 9-11, 2008

Increasingly Correct Message Passing Averaging Algorithms

Kurt Plarre and Francesco Bullo

Abstract— **We study averaging algorithms, when implemented in large networks of wirelessly connected elements. We extend the notion of "Increasing Correctness" (IC) which was defined for cycle-free graphs, to general graphs. An averaging algorithm that is IC has meaningful outputs at each iteration. This makes it possible to stop the algorithm at any time, and use the output values computed up to that time.**

We prove that the class of IC averaging algorithms is non-trivial. We then present a simple IC averaging algorithm that is based on ideas from Graphical Models, and study its properties. Finally, we give example applications and simulations of IC averaging algorithms.

I. INTRODUCTION

Large networks of wirelessly connected elements, such as sensor networks and robot teams promise new ways of perceiving the world and acting upon it. For example, a sensor network can be used to monitor environmental variables in a large area to study their behavior or detect events such as a forest fire. Robotic networks could be used, for example, for surveillance, reconnaissance, oil spill contention, search-and-rescue missions, and space exploration.

Algorithms for large sensor networks or robot teams must be power aware, computationally and memory efficient, scalable, operate in real-time, and take into account inherent communication constraints. See, for example, [1] and [2].

Systems such as sensor networks can generate large amounts of data. It is sometimes more efficient to compute aggregates or summaries of the data, before processing, than to process the raw data. Algorithms to compute aggregates in sensor and robot networks have been extensively studied. See, for example, [3], [4], [5]. See also [6] and the references therein.

We are interested in averaging algorithms to compute *weighted* averages in a decentralized environment. Such averaging algorithms can be compared according to different criteria. For example, convergence time, scalability, robustness to, for example, link failures and delays, and suitability to asynchronous operation. Here we study a property of some averaging algorithms, which we call "Increasing Correctness" Such property was introduced in [7] for message passing-like algorithms in loop-free graphs. In that case IC describes the fact that at each time instant, the output of the algorithm at each node in the network is the exact solution of a subproblem defined on a neighborhood of that node, and that neighborhood grows with time.

The main contributions of this paper are two. First, we give a definition of increasing correctness for averaging algorithms in general graphs, and prove that the class of IC averaging algorithms is non-trivial. Second, we introduce a simple message passing-like averaging algorithm that is IC by definition. The algorithm appears to be a novel simple version of a more sophisticated "Belief Consensus" algorithm presented in [6]. We analyze the convergence properties of our algorithm in deterministic and stochastic environments using ideas from graphical models. Finally, we apply our ideas to a spatial filtering problem and a target localization application in a simulation study.

II. SETUP

Let $G = (V, E)$, $|V| = n$ be an undirected graph, with vertex set V and edge set E. Let $a \in \mathbb{R}^n$ be a vector that represents the input to the averaging algorithm. Node v_i in V "knows" only a_i. We are interested in computing weighted averages of the a_i, i.e., expressions of the form

$$\bar{a}_\beta := \sum_{i=1}^{n} \beta_i a_i,$$

where, for $i \in \{1, \dots, n\}$, $\beta_i \in \mathbb{R}_{\geq 0}$, and $\sum_{i=1}^n \beta_i = 1$. In the algorithms we present, we will not have control over the weights β_i. Instead, we define an algorithm by local rules, and study the behavior of the computed averages.

We consider recursions of the form

$$\begin{aligned} x(t+1; a, x_0) &= f(x(t; a, x_0), a), \\ y(t; a, x_0) &= g(x(t; a, x_0), a), \end{aligned} \tag{1}$$

where $x \in \mathbb{R}^{n_x}$ is the state of the algorithm, with $x(0; a, x_0) = x_0$, and $y \in \mathbb{R}^n$ is the output, with y_i the output at node i. $f : \mathbb{R}^{n_x} \times \mathbb{R}^n \to \mathbb{R}^{n_x}$ and $g : \mathbb{R}^{n_x} \times \mathbb{R}^n \to \mathbb{R}^n$ are given maps. To simplify the presentation, of what we call an "averaging algorithm," we introduce the following notation:

(i) Given $m \in \mathbb{Z}_{>0}$, $x \in \mathbb{R}^m$, $S \subseteq \{1, \dots, m\}$, let x_S denote the subvector of x containing the entries indexed by S, in order.

(ii) Given $m, n \in \mathbb{Z}_{>0}$, $R_1, \dots, R_m \subseteq \{1, \dots, n\}$, and $S \subseteq \{1, \dots, m\}$, let $R_S := \bigcup_{i \in S} R_i$.

(iii) For each $i \in \{1, \dots, n\}$, let N_i denote the set of neighbors of node v_i, in G, and $\bar{N}_i := N_i \cup \{i\}$.

For each $i \in \{1, \dots, n\}$, let $S_i \subseteq \{1, \dots, n_x\}$ be the indices of the state variables, known to node v_i. With this notation we can introduce what we call an "averaging algorithm."

Definition 1 (Averaging algorithm): We say that (1) is an *averaging algorithm* for G, if the following holds:

This material is based upon work supported in part by AFOSR MURI Award FA9550-07-1-0528 and ONR Award N00014-07-1-0721.

Kurt Plarre is with the Department of Mechanical Engineering, and Center for Control, Dynamical Systems and Computation, University of California at Santa Barbara, Santa Barbara, CA 93106, plarre@engineering.ucsb.edu

Francesco Bullo is with the Department of Mechanical Engineering, University of California at Santa Barbara, CA 93106, bullo@engineering.ucsb.edu

978-1-4244-3124-3/08/$25.00 ©2008 IEEE

(i) $f_{S_i}(x,a) = f_{S_i}(x_{S_{\bar{N}_i}}, a_{\bar{N}_i})$ and $g_i(x,a) = g_i(x_{S_i}, a_i)$.
(ii) There exist $\beta_{i,j} \in \mathbb{R}_{\geq 0}$, $i,j = 1,\dots,n$, such that $\sum_{j=1}^n \beta_{i,j} = 1$, and $y_i(t;a,x_0) \to \sum_{j=1}^n \beta_{i,j} a_j$, as $t \to +\infty$, from any initial condition x_0.

Notice that the computation of f requires communication, while the computation of g is performed locally at each node. We denote an averaging algorithm by a tuple: $\mathcal{A} := (G, f, g)$. We give an example of a simple averaging algorithm in Section III.

We define increasing correctness in the following way.

Definition 2 (Increasing correctness): Let $\mathcal{A} = (G, f, g)$ be an averaging algorithm. We say that $\mathcal{A}$ is "increasingly correct," if for all $a \in \mathbb{R}^n$, there exists $x_0 \in \mathbb{R}^n$, $\beta_{i,j}(t) \in \mathbb{R}_{\geq 0}$, $i,j = 1,\dots,n$, and $N_i(t) \subset V$, $i = 1,\dots,n$, such that $N_i(0) = \{v_i\}$, $N_i(+\infty) = V$, and for all $t \geq 0$, $v_i \in N_i(t)$, $N_i(t+1) \supseteq N_i(t)$, $\beta_{i,j}(t) = 0$, for $j \notin N_i(t)$, $\sum_{j \in N_i(t)} \beta_{i,j}(t) = 1$, and $y_i(t;a,x_0) = \sum_{j \in N_i(t)} \beta_{i,j}(t) a_j$.

We now specialize our definition of averaging algorithm.

Definition 3 (Linear averaging algorithm): We say that an averaging algorithm $\mathcal{A} = (G, f, g)$ is linear if there exist matrices $A \in \mathbb{R}^{n_x \times n_x}$, $B \in \mathbb{R}^{n_x \times n}$, $C \in \mathbb{R}^{n \times n_x}$, and $D \in \mathbb{R}^{n \times n}$, with nonnegative coefficients, such that $f(x,a) = Ax + Ba$, and $g(x,a) = Cx + Da$. In other words, the algorithm is given by the recursion

$$\begin{aligned} x(t+1;a,x_0) &= Ax(t;a,x_0) + Ba, \\ y(t;a,x_0) &= Cx(t;a,x_0) + Da. \end{aligned}$$

We denote a linear averaging algorithm by a tuple $\mathcal{A}_L = (G, A, B, C, D)$.

The following two lemmas show that the class of IC averaging algorithm is non-trivial.

Lemma 4 (Linear consensus is IC): The linear averaging algorithm $\mathcal{A}_L = (G, A, 0, I, 0)$, with $A \in \mathbb{R}^{n \times n}$ a stochastic matrix, and $x_0 = a$, is IC.

Proof: The proof is an immediate consequence of the properties of stochastic matrices. ∎

The following lemma shows how to obtain an IC correct averaging algorithm from any linear averaging algorithm.

Lemma 5 (Constructing IC algorithms): For any linear averaging algorithm $\mathcal{A}_L = (G, A, B, C, D)$ there exists a, possibly non-linear, IC averaging algorithm.

Proof: Let $\bar{x}, e \in \mathbb{R}^{n_x}$, with e a vector of ones. The output of a linear averaging algorithm, with zero initial condition, satisfies

$$y(t;\bar{x},0) = \left(C \sum_{k=0}^{t-1} A^k B + D \right) \bar{x}, \tag{2}$$

Considering only the $i-$th entry of y, we can write (2) as

$$y_i(t;\bar{x},0) = \sum_{j=1}^n \alpha_{i,j}(t) \bar{x}_j,$$

for $i \in \{1,\dots,n\}$. The coefficients $\alpha_{i,j}(t)$ are obtained directly from (2). Letting $\beta_{i,j}(t) := \alpha_{i,j}(t) / \sum_{l=1}^n \alpha_{i,l}(t)$, we have that $\sum_{j=1}^n \beta_{i,j}(t) = 1$, and

$$\frac{y_i(t;a,0)}{y_i(t;e,0)} = \sum_{j=1}^n \beta_{i,j}(t) a_j.$$

Therefore the following defines an IC averaging algorithm.

$$\begin{aligned} x(t+1;a,0) &= Ax(t;a,0) + Ba, \\ x(t+1;e,0) &= Ax(t;e,0) + Be, \\ z(t;a,0) &= Cx(t;a,0) + Da, \\ z(t;e,0) &= Cx(t;e,0) + De, \\ y_i(t;(a,e),0) &= \frac{z_i(t;a,0)}{z_i(t;e,0)}, \qquad i = 1,\dots,n. \end{aligned}$$

∎

Lemmas 4 and 5 show that there exist a large number of IC averaging algorithms. Notice that in this paper we have considered only time invariant algorithms. More generally one could consider f and g as functions of time also.

In the next section we present an IC averaging algorithm, which is based on ideas from Graphical Models, and study some of its properties.

III. ANALYSIS OF A MESSAGE PASSING-LIKE AVERAGING ALGORITHM

Let $G = (V, E)$, $|V| = n$ be an undirected graph. Let $a_1, \dots, a_n \in \mathbb{R}$ be given numbers. As before, the averaging algorithm is defined by its state, output, and initial condition. As in message passing, we call the states "messages" $m_{i \to j}$ and the outputs "beliefs," b_i. For each edge $(v_i, v_j) \in E$, there are two messages $m_{i \to j}$, and $m_{j \to i}$. For each node $v_i \in V$, there is a belief b_i. The update rules for the messages are given by

$$m_{i \to j}(t+1;a,0) = a_i + \sum_{k \in N_{i,j}} m_{k \to i}(t;a,0), \tag{3}$$

and similarly for $m_{i \to j}(t;e,0)$. The expressions for the beliefs are

$$\begin{aligned} b_i(t;a,0) &= a_i + \sum_{k \in N_i} m_{k \to i}(t;a,0), \\ b_i(t;e,0) &= 1 + \sum_{k \in N_i} m_{k \to i}(t;a,0), \\ b_i^{\text{avg}}(t;(a,e),0) &= \frac{b_i(t;a,0)}{b_i(t;e,0)}, \end{aligned} \tag{4}$$

where $N_{i,j} := N_i \setminus \{j\}$, and we have assumed zero initial conditions. The algorithm is defined by local rules. We study properties of the algorithm and the computed averages that these rules produce.

It is not difficult to see that if the graph G is cycle-free, then $b_i^{\text{avg}}(t;a,0)$ converges after a finite number of iterations, to the correct average. It can also be proved that at each time t, $b_i^{\text{avg}}(t;a,0)$ is the exact average in the subgraph of G containing all nodes at distance up to t from v_i.

If we implement equation (3) in a general loopy graph, then the messages diverge. To avoid this problem, we use a scaling factor $\alpha \in]0,1[$. The resulting algorithm is then given by the following update rule for the messages:

$$m_{i \to j}(t;a,0) = \alpha \left(a_i + \sum_{k \in N_{i,j}} m_{k \to i}(t;a,0) \right), \tag{5}$$

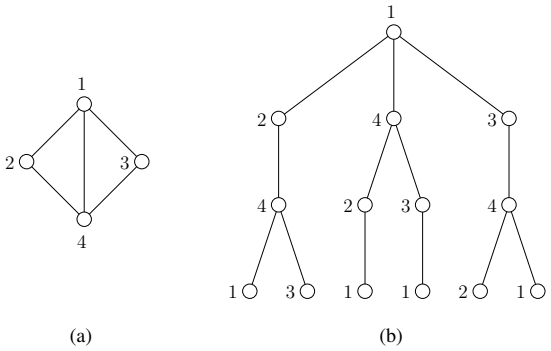

Fig. 1. Loopy graph (a), and corresponding unwrapped tree (b), rooted at v_1.

and similarly for $m_{i\to j}(t;e,0)$.

This algorithm is related to other filtering and averaging algorithms found in the literature. The weighted sum-count algorithm can be seen as a simplified version of the algorithm presented in [6], but its purpose is to compute local weighted averages, rather than the average of all the a_i as in [6]. It is also related to consensus algorithms, with the following differences: (1) it is affine, rather than linear, (2) the state is defined on the edges of the graph not the vertices, and (3) the output is a function of the state, not the state itself. The idea of running two parallel averaging algorithms and combining their outputs has been proposed before, for example, in [8] to compute maximum likelihood estimates in a distributed environment and [9]. Finally, we can also find a connection to some filtering algorithms for image restoration; e.g., see [10].

A. Unwrapped trees

The message that v_i sends to v_j at time t is a function of all messages received by v_i at time $t-1$, except from v_j. For every v_i and t, it is possible to construct a rooted tree, such that the messages received by the root node are equal to the messages received by v_i at time t. Such a tree is called an "unwrapped tree." We define such a tree in the following way.

Definition 6 (Unwrapped tree): Given a graph $G = (V,E)$, an unwrapped tree, T, rooted at v_i, for a message passing algorithm (3) is a tree, rooted at v_i, such that the nodes at distance t from the root in T, are replicas of nodes reachable from v_i in G, following treks of length t.

A "trek" is a walk without backtracking, which can have cycles, and the length of a trek W, which we denote by $|W|$, is the number of edges of W. Figure 1 shows an example of an unwrapped tree. Notice that in the unwrapped tree, the label "i" denotes "replicas" of node v_i. The size of the tree depends on the time instant, t. Unwrapped trees are used to study the behavior of message passing algorithms [11], [12].

If $\mathcal{A} = (G, A, B, C, D)$ is a linear averaging algorithm with zero initial condition, we can use the superposition principle to analyze the contribution of each node in the unwrapped tree, to the messages received by the root node, separately. This simplifies the analysis.

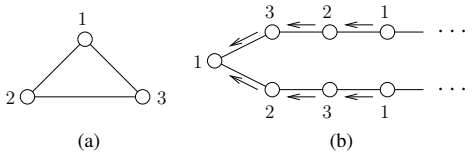

Fig. 2. Loopy graph (a), and corresponding unwrapped tree (b), rooted at v_1.

B. Steady state analysis

For example, consider the averaging algorithm in (5), with G a cycle on three nodes. The unwrapped tree, rooted at v_1 is a path. See Figure 2. We can easily see that

$$m_{2\to1}(+\infty;a,0) = a_2\left(\alpha+\alpha^4+\alpha^7+\dots\right) + a_3\left(\alpha^2+\alpha^5+\alpha^8\dots\right) + a_1\left(\alpha^3+\alpha^6+\alpha^9\dots\right),$$

and similarly for $m_{3\to1}(+\infty;a,0)$. Therefore

$$b_1(+\infty;a,0) = a_1 + m_{2\to1}(+\infty;a,0) + m_{3\to1}(+\infty;a,0) = \frac{(1+\alpha^3)a_1 + (\alpha+\alpha^2)a_2 + (\alpha+\alpha^2)a_3}{1-\alpha^3},$$

and similarly for $b_1(+\infty;e,0)$. For a general cycle on n nodes, we will have

$$b_1(+\infty;a,0) = \frac{(1+\alpha^n)a_1 + \sum_{i=2}^{n}(\alpha^{i-1}+\alpha^{n-i+1})a_i}{1-\alpha^n},$$

and similarly for $b_1(t;e,0)$.

For general graphs, we can prove the following result

Lemma 7 (Stability conditions): Consider the averaging algorithm given by (5). If $\alpha(d_{\max}-1) < 1$, where $d_{\max}$ is the maximum degree of a node in G, then the algorithm converges, and

$$b_i(+\infty;a,0) = a_i + \sum_{j=1}^{n}\left(\sum_{\text{all } W_k^{i,j}}\alpha^{|W_k^{i,j}|}\right)a_j,$$

where $W_k^{i,j}$ is any trek from v_i to v_j. The expression for $b_i(+\infty;e,0)$ is similar. Thus, the expression for $b_i^{\text{avg}}(+\infty;(a,e),0)$ is

$$b_i^{\text{avg}}(+\infty;(a,e),0) = \frac{a_i + \sum_{j=1}^{n}\left(\sum_{\text{all } W_k^{i,j}}\alpha^{|W_k^{i,j}|}\right)a_j}{1+\sum_{\text{all } W_k^{i}}\alpha^{|W_k^{i}|}},$$

where W_k^i denotes any trek starting at node v_i.

Proof: Consider the unwrapped tree, rooted at a node, say, v_i. The maximum number of nodes at depth l, in the unwrapped tree at time t is $(d_{\max}-1)^l$. The contribution of any copy of v_j at depth l to the belief at the root node is $\alpha^l a_j$. Therefore

$$|b_i(t;a,0)| \le |a_i| + \sum_{l=1}^{t}\alpha^l(d_{\max}-1)^l\max\left\{|a_i|\right\}_{i=1}^{n},$$

which is finite, for all t, if $\alpha(d_{\max}-1)<1$.

This proves that the algorithm converges. To find the limit, we observe that any path from the root node to a copy of v_j in the unwrapped tree, corresponds to a trek in G, from v_i to v_j. Thus, in the unwrapped tree, any copy of v_j contributes to only one message at the root node, and the contribution is given by $\alpha^{|W|}a_j$, where W is the path from the root node to the copy of v_j in the tree. The result then follows by adding all contributions. ■

C. Transient analysis

We are not only interested in the asymptotic output values computed by an averaging algorithm, but also in their behavior before convergence. We would like to know (1) how close those outputs are to the asymptotic ones, and (2) how meaningful they are, i.e., can those outputs be used even if they are far from their final values?

In this section we answer the first question for the algorithm given by (5). We prove the following lemma.

Lemma 8 (Convergence rate): If $\alpha(d_{\max}-1)<1$, then

$$|b_i(t;a,0)-b_i(+\infty;a,0)|\in O\left(\alpha^{t+1}(d_{\max}-1)^{t+1}\right),$$

where we used Landau's "big O" notation.

Proof: We assume that the algorithm is executed in a synchronous fashion, i.e., at each time instant t, each sensor receives (possibly empty) incoming messages from its neighbors, and sends outgoing messages to its neighbors. Following the analysis in the last section, it is not difficult to see that

$$b_i(t;a,0)=a_i+\sum_{j=1}^{n}\left(\sum_{|W_k^{i,j}|\le t}\alpha^{|W_k^{i,j}|}\right)a_j,$$

Therefore

$$\begin{aligned}|b_i(t;a,0)-b_i(+\infty;a,0)|=&\\ =&\left|\sum_{j=1}^{n}\left(\sum_{|W_k^{i,j}|>t}\alpha^{|W_k^{i,j}|}\right)a_j\right|\\ \le&\sum_{l>t}\alpha^l(d_{\max}-1)^l\max\left\{|a_j|\right\}_{j=1}^n\\ =&\frac{\alpha^{t+1}(d_{\max}-1)^{t+1}\max\left\{|a_j|\right\}_{j=1}^n}{1-\alpha(d_{\max}-1)},\end{aligned}$$

which proves the lemma. ■

Noticing that $b_i(t;e,0)\ge 1$, for all $t\ge 0$, and the fact that for any $x,\delta_x,y,\delta_y\in\mathbb{R}$, with $y\ge 1$, and $y+\delta_y\ge 1$, we have

$$\left|\frac{x+\delta x}{y+\delta y}-\frac{x}{y}\right|\le\left(1+\left|\frac{x}{y}\right|\right)\max\left\{|\delta_x|,|\delta_y|\right\},$$

by letting $x:=b_i(+\infty;a,0)$, $x+\delta_x:=b_i(t;a,0)$, $y:=b_i(+\infty;e,0)$, and $y+\delta_y:=b_i(t;e,0)$, we can prove that also

$$\left|b_i^{\text{avg}}(t;(a,e),0)-b_i^{\text{avg}}(+\infty;(a,e),0)\right|\in O\left(\alpha^{t+1}(d_{\max}-1)^{t+1}\right).$$

When the inputs to the algorithm, a_i, are estimates of a random variable a^0, we can formalize our notion of "meaningful." We do so in the following lemma.

Lemma 9 (Unbiased estimators): Let $\mathcal{A}_L=(G,A,B,C,D)$, be an IC averaging algorithm. Let a_i, $i\in\{1,\dots,n\}$, be unbiased estimates of a random variable a^0, and $a=[a_1,\dots,a_n]$. Then $b_i(t;a,x_0)$ computed using $\mathcal{A}_L$ is an unbiased estimate of a^0, for all t.

Proof: The assumption that $\mathcal{A}_L$ is IC implies that at each time t, it holds that $b_i(t;a,x_0)=\sum_{j=1}^{n}\beta_{i,j}(t)a_j$, with $\beta_{i,j}(t)\in\mathbb{R}_{\ge 0}$ and $\sum_{j=1}^{n}\beta_{i,j}(t)=1$. Taking expectations we have

$$\mathrm{E}\left[b_i(t;a,x_0)\right]=\sum_{j=1}^{n}\beta_{i,j}(t)\mathrm{E}\left[a_j\right]=\mathrm{E}\left[a^0\right].$$

This proves the lemma. ■

D. Random weights

In this section we study a randomized version of the previous algorithm. The state and output of the algorithm are the same as before, but in this case, any given message is transmitted with a certain probability, rater than with certainty. The resulting algorithm can be described as follows:

$$m_{i\to j}(t;a,0)=\alpha_{i,j}(t)\left(a_i+\sum_{k\in N_{i,j}}m_{k\to i}(t;a,0)\right),\quad(6)$$

and similarly for $m_{i,j}(t;e,0)$. Here $\alpha_{i,j}(t)=\alpha$ with probability p, and $\alpha_{i,j}(t)=0$ with probability $1-p$, with $\alpha_{i_1,j_1}(t_1)$ independent of $\alpha_{i_2,j_2}(t_2)$ for $(i_1,j_1,t_1)\neq(i_2,j_2,t_2)$. A zero message is not sent, and a message that is not received is assumed zero. The update rules of the beliefs are as before.

We notice that even in the random case, at each time t, $b_i^{\text{avg}}(t;a,0)$ is a weighted average of the a_i, although, in this case, the coefficients are random. We prove the following lemma.

Lemma 10 (Convergence in expectation): If $\alpha p(d_{\max}-1)<1$, then (6) converges exponentially in expectation, and $b_i^{\text{avg}}(t;a,0)$ is L_1 bounded.

Proof: Let $\bar{m}_{i,j}(t;a,0):=\mathrm{E}\left[m_{i,j}(t;a,0)\right]$, $\bar{m}_{i,j}(t;e,0):=\mathrm{E}\left[m_{i,j}(t;e,0)\right]$, and $\bar{\alpha}_{i,j}(t):=\mathrm{E}\left[\alpha_{i,j}(t)\right]$. Then,

$$\bar{m}_{i\to j}(t;a,0)=\bar{\alpha}_{i,j}(t)\left(a_i+\sum_{k\in N_{i,j}}\bar{m}_{k\to i}(t;a,0)\right),$$

and similarly for $\bar{m}_{i\to j}(t;e,0)$. But $\bar{\alpha}_{i,j}(t)=\alpha p$, which means that $\bar{m}_{i,j}(t;a,0)$ converges as $t\to\infty$, if $\alpha p(d_{\max}-1)<1$.

If we now define $\tilde{m}_{i,j}(t;a,0):=\mathrm{E}\left[|m_{i,j}(t;a,0)|\right]$, and $\tilde{\alpha}_{i,j}(t):=\mathrm{E}\left[|\alpha_{i,j}(t)|\right]$, we can write

$$0\le\tilde{m}_{i\to j}(t;a,0)\le\tilde{\alpha}_{i,j}(t)\left(|a_i|+\sum_{k\in N_{i,j}}\tilde{m}_{k\to i}(t;a,0)\right).$$

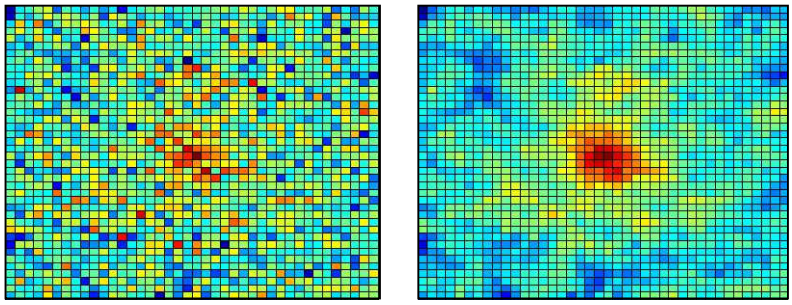

Fig. 3. Estimated temperatures before (left) and after (right) spatial filtering.

This inequality shows that, if $\tilde{\alpha}_{i,j}(t)(d_{\max}-1) \leq 1$, then each $\tilde{m}_{i\to j}(t;a,0)$ is bounded. Since $b(t;e,0) \geq 1$, for all t, we can write

$$0 \leq \mathrm{E}\left[\left|b_i^{\mathrm{avg}}(t;a,0)\right|\right] \leq |a_i| + \sum_{k\in N_i} \tilde{m}_{k\to i}(t;a,0),$$

which proves that $\mathrm{E}\left[\left|b_i^{\mathrm{avg}}(t;a,0)\right|\right]$ is also bounded, under the same conditions. ■

IV. APPLICATIONS AND SIMULATIONS

In this section we discuss two applications: Spatial filtering in a sensor network and target localization with mobile sensors.

A. Spatial filtering

A sensor network is used to estimate the temperature produced by a heat source located at the origin of the plane. Sensors are arranged on a randomly perturbed grid, where the horizontal and vertical perturbations are uniform in $[-0.25, 0.25]$. In other words each sensor location was chosen, uniformly in a square with sides of length 0.5. The temperature at location (x,y) at time t is given by

$$\theta(x,y,t) = \int_0^t \frac{1}{\tau} \mathrm{e}^{-\frac{x^2+y^2}{\tau}} d\tau,$$

where we have omitted constants, for simplicity.

Each sensor takes one noisy measurement of the field at its location. The noise at each node is Gaussian with variance $(0.2)^2$. The noises at different locations are independent. After taking measurements the sensors execute the averaging algorithm given by 5 and 4. The communication graph is a grid, i.e., each sensor communicates only with its four neighbors on the grid.

The resulting estimated field is shown in Figure 3. The left panel in Figure 3 shows the measured temperatures at each sensor. The right panel shows the estimated temperatures, after running the averaging algorithm. We can see from Figure (3) that filtering greatly improved the estimated temperature profile.

B. Target localization

A network of mobile robots must localize and move towards a number of fixed targets on the plane. Each robot can make noisy measurements of the location of the target that is closest to it. The operation of the robots is synchronous. Robots execute the following operations, in order: (1) sense, (2) initialize their estimates of the target's location, (3) execute an averaging algorithm for K steps, and (4) move according to the updated estimate. We now describe the sensing, communication, computation, and control models.

Sensing model: Each robot can take noisy measurements of the location of the target closest to it. We neglect the effect of other targets that might affect such measurements. As in [13], we use the following sensor model: $a_i^x(t) = x_j^o + r_i(t)\cos(\theta_i(t))$, $a_i^y(t) = y_j^o + r_i(t)\sin(\theta_i(t))$, where

(i) $a_i^x(t)$ and $a_i^y(t)$ are the measured horizontal and vertical positions, at node i, of the target that is closest to it, and at time t,

(ii) (x_j^o, y_j^o) represents the actual location of the target closest to robot i,

(iii) $r_i(t)$ is uniformly distributed in $[0, r_{\max}]$, and $\theta_i(t)$ is uniformly distributed in $[0, 2\pi]$.

The location of robot i at time t is $(x_i(t), y_i(t))$.

Communication model: Two robots can communicate if they are at distance less than d_{comm}. Communication is bidirectional. Robots communicate their current estimates of the location of the targets (each robot estimates the location of only one target). At each time each robot communicates with a subset of its neighbors in the communication graph.

Computation model: Each robot stores an estimate of the location of the source $(\hat{x}_i(t,k), \hat{y}_i(t,k))$. At each time, this estimate is initialized as $\hat{x}_i(t,0) = a_i^x(t)$ and $\hat{y}(t,0) = a_i^y(t)$. These estimates are updated according to the following rule:

$$\hat{x}_i'(t,k+1) = a_i^x(t) + \alpha\left(\hat{x}_i'(t,k) + \sum_{j\in N_i(t)} \hat{x}_j'(t,k)\right),$$

and similarly for $\hat{y}_i'(t+1,k)$. Here $N_i(t)$ is a *subset* of the neighbors of robot i in the communication graph. The size of $N_i(t)$ and α are chosen to make the algorithm stable. As initial conditions we set $\hat{x}_i(t,0) = a_i^x(t)$ and $\hat{y}_i(t,0) = a_i^y(t)$. The following recursion is executed in parallel to the previous two:

$$c_i(t,k+1) = 1 + \alpha\left(c_i(t,k) + \sum_{j\in N_i(t)} c_j(t,k)\right),$$

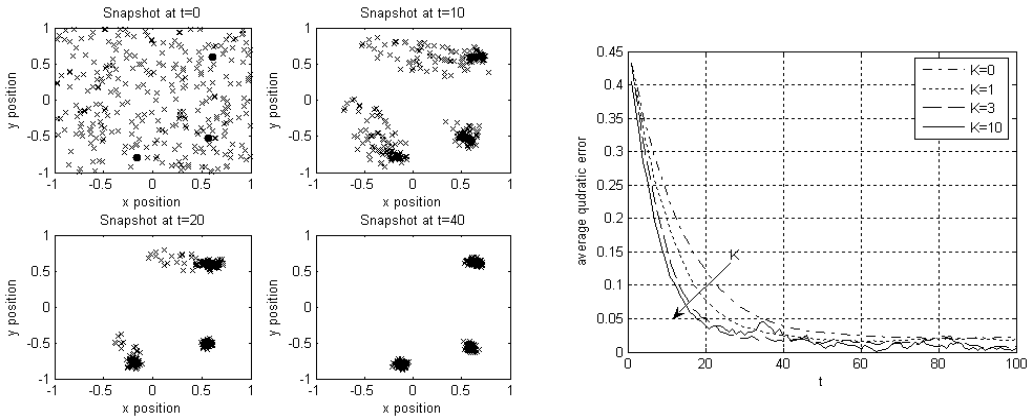

Fig. 4. Snapshots of the operation of the target localization algorithm at $t = 0, 10, 20, 40$ (left) and behavior of average (over all robots) quadratic error in estimated target location.

with initial condition $c_i(t, 0) = 1$. These update rules are executed K times. The estimates of the locations of the targets are obtained as $\hat{x}_i(t) = \hat{x}'_i(t, K)/c_i(t, K)$ and $\hat{y}_i(t) = \hat{y}'_i(t, K)/c_i(t, K)$.

By Lemma 4, we know that this averaging algorithm is IC. Hence, at each time, the estimates at each node are convex combinations of the measurements in a neighborhood of that node.

Control model: After the averaging algorithm has been executed, each robot moves towards $(\hat{x}_i(t, K), \hat{y}_i(t, K))$, by a fixed distance d_{control}. If the robot is closer to $(\hat{x}_i(t, K), \hat{y}_i(t, K))$ than d_{control}, it moves directly to that point.

Initially, the robots and targets are uniformly distributed in the square $[-1, 1] \times [-1, 1]$. In the simulations we used $d_{\text{comm}} = 0.1$, $d_{\text{control}} = 0.05$, $r_{\max} = 2$, and $n = 300$ robots. The location of the sources was chosen uniformly in $[-1, 1] \times [-1, 1]$. Figure 4 (left) shows snapshots of the operation of the algorithm for $t = 0, 10, 20, 40$. We can see that the robots cluster around the targets.

In Figure 4 (left), we compare the behavior of the average (over all robots) quadratic error in the location of the robots, with respect to time. The error in the location of a robot is the squared distance to the target that is closest to that robot. Each curve in Figure 4 (right) corresponds to a different value of K. We simulated the algorithm for $K = 0, 1, 3, 10$. The number of robots was $n = 1000$. The noise model was the same as before, but with $r_{\max} = 2$. We observe that averaging improves the rate of convergence of the robots to the targets.

V. CONCLUSIONS

We extended the notion of increasing correctness from loop-free graphs to loopy graphs, showed that the class of IC algorithms is non-trivial, and studied a simple Message Passing-like IC algorithm. We also showed the performance of the proposed IC algorithms in two interesting applications. Future work includes studying other applications of the proposed algorithms and more general linear recursions on graphs.

REFERENCES

[1] P. Gupta and P. R. Kumar. The capacity of wireless networks. *IEEE Transactions on Information Theory*, 46(2):388–404, 2000.

[2] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. A survey on sensor networks. *IEEE Communications Magazine*, 40(8):102–114, 2002.

[3] C. Intanagonwiwat, R. Govindan, and D. Estrin. A scalable and robust communication paradigm for sensor networks. In *ACM/IEEE International Conference on Mobile Computing and Networking*, Boston, MA, August 2000.

[4] S. R. Madden, M. J. Franklin, J. Hellerstein, and W. Hong. Tag: A tiny aggregation service for ad hoc wireless sensor networks. In *USENIX Symposium on Operating Systems Design and Implementation*, Boston, MA, December 2002.

[5] S. R. Madden, R. Szewczyk, M. J. Franklin, and D. Culler. Supporting aggregate queries over ad-hoc wireless sensor networks. In *Workshop on Mobile Computing Systems and Applications*, pages 49–58, Callicoon, NY, June 2002.

[6] C. C. Moallemi and B. Van Roy. Consensus propagation. *IEEE Transactions on Information Theory*, 52(11):4753–4766, 2006.

[7] K. Plarre, P. R. Kumar, and T. I. Seidman. Increasingly correct message passing algorithms for heat source detection in sensor networks. In *IEEE Conference on Sensor and Ad Hoc Communications and Networks (SECON)*, pages 470–479, October 2004.

[8] L. Xiao, S. Boyd, and S. Lall. A scheme for robust distributed sensor fusion based on average consensus. In *Symposium on Information Processing of Sensor Networks (IPSN)*, pages 63–70, Los Angeles, CA, April 2005.

[9] A. Olshevsky and J. N. Tsitsiklis. Convergence rates in distributed consensus and averaging. In *IEEE Conf. on Decision and Control*, pages 3387–3392, San Diego, CA, December 2006.

[10] A. K. Jain. *Fundamentals of Digital Image Processing*. Prentice Hall, Englewood Cliffs, NJ, 1989.

[11] Y. Weiss and W. T. Freeman. On the optimality of solutions of the max-product belief-propagation algorithm in arbitrary graphs. *IEEE Transactions on Information Theory*, 47(2):736–744, 2001.

[12] Y. Weiss. Correctness of belief propagation in Gaussian graphical models of arbitrary topology. *Neural Computation*, 13:2173–2200, 2001.

[13] R. Olfati-Saber, E. Franco, E. Frazzoli, and J. S. Shamma. Belief consensus and distributed hypothesis testing in sensor networks. In P.J. Antsaklis and P. Tabuada, editors, *Network Embedded Sensing and Control. (Proceedings of NESC'05 Worskhop)*, volume 331 of *Lecture Notes in Control and Information Sciences*, pages 169–182. Springer Verlag, New York, 2006.