Chapter 10
Distributed Approximation and Tracking Using Selective Gossip

Deniz Üstebay, Rui Castro, Mark Coates and Michael Rabbat

Abstract This chapter presents selective gossip which is an algorithm that applies the idea of iterative information exchange to vectors of data. Instead of communicating the entire vector and wasting network resources, our method adaptively focuses communication on the most significant entries of the vector. We prove that nodes running selective gossip asymptotically reach consensus on these significant entries, and they simultaneously reach an agreement on the indices of entries which are insignificant. The results demonstrate that selective gossip provides significant communication savings in terms of the number of scalars transmitted. In the second part of the chapter we propose a distributed particle filter employing selective gossip. We show that distributed particle filters employing selective gossip provide comparable results to the centralized bootstrap particle filter while decreasing the communication overhead compared to using randomized gossip to distribute the filter computations.

10.1 Introduction

Many applications of wireless sensor networks require collection and processing of large amounts of data. The main challenge in fulfilling these tasks is preserving network resources such as lifetime and bandwidth. One approach to fuse and

D. Üstebay (✉) · M. Coates · M. Rabbat
Department of Electrical and Computer Engineering, McGill University, Montreal, QC, Canada
e-mail: deniz.ustebay@mail.mcgill.ca

M. Coates
e-mail: coates@ece.mcgill.ca

M. Rabbat
e-mail: michael.rabbat@mcgill.ca

R. Castro
Department of Mathematics, Eindhoven University of Technology, Eindhoven, The Netherlands
e-mail: rmcastro@tue.nl

process large amounts of data without draining network resources is to reduce the data dimensionality. We present an algorithm called selective gossip to approximate high dimensional vectors of network data in an efficient manner. Our method is based on gossip algorithms which are decentralized methods studied extensively for scalar network data. In essence, gossip algorithms utilize iterative information exchange between pairs of nodes, and asymptotically all nodes reach consensus on a network aggregate. Selective gossip applies the idea of iterative information exchange to vectors of data. Instead of communicating the entire vector and wasting network resources, our method adaptively focuses communication on the most significant entries of the vector. We prove that nodes running selective gossip asymptotically reach consensus on these significant entries, and they simultaneously reach an agreement on the indices of entries which are insignificant.

Selective gossip can be taken as a building block and used in various distributed signal processing algorithms. Here we study the distributed target tracking problem where the nodes of a sensor network collaboratively track a moving object. For problems involving nonlinear dynamics, nonlinear measurements, and non-Gaussian noise, particle filtering is the current state-of-the-art-estimation method. We propose a distributed particle filter implementation using selective gossip. In this setting, nodes maintain a shared particle filter to sequentially estimate the state of the target. The measurements taken by sensors are fused by reaching a consensus on the likelihood associated with the each particle. Selective gossip efficiently identifies particles with large weights and focuses communication resources on computing these important weights. Through a simulation study we demonstrate that selective gossip requires lower communication overhead while achieving similar accuracy as compared to the state-of-the-art distributed particle filtering approaches on a scenario involving bearings-only measurements of a maneuvering target.

This chapter is organized as follows. Section 10.2 reviews gossip algorithms. Section 10.3 discusses the distributed averaging problem for vectors. Section 10.4 proposes the selective gossip algorithm in its three versions and also provides the convergence results. Section 10.5 introduces distributed tracking problem and proposes the distributed particle filter using selective gossip. A distributed target tracking scenario is presented to illustrate the performance of this algorithm. Section 10.6 concludes the chapter with a discussion of the results.

10.2 Gossip Algorithms

Operating under energy and bandwidth constraints, wireless sensor networks require efficient and reliable methods for processing. The traditional approach of centralized processing has several drawbacks. It introduces a single point of failure to the network. Furthermore, in dense networks, the links close to the central authority can become bottlenecks. To avoid congestion and, also, to exploit processing capabilities of sensor nodes, in-network processing algorithms are proposed. In-network processing can be performed using spanning trees or Hamiltonian cycles. These are effective methods when the network topology does not change over time. However, since they require forming and maintaining routes, these methods also have significant communication overhead when nodes are mobile or wireless networking conditions are not reliable. Gossip algorithms, on the other hand, are decentralized methods which do not require specialized routes. They are known to provide robust and scalable solutions for in-network processing.

Gossip algorithms have been widely studied as solutions to distributed consensus, a problem which dates back to early work of Tsitsiklis et al. [32, 33]. This problem requires nodes to reach an agreement by using only local exchanges. It is acknowledged as a canonical problem in distributed control and signal processing (see, e.g., the surveys [7, 23]). Some example applications are cooperative control of multiple autonomous vehicles [18], parameter estimation [30], distributed optimization [22], and source localization [26].

The standard example of distributed consensus is the *average consensus* problem, where in a network of n nodes, each node v has a scalar value $x^v \in \mathbb{R}$, and the goal is to compute the average

$$\overline{x} = \frac{1}{n}\sum_{v=1}^{n} x^v, \tag{10.1}$$

at every node. Although averaging of scalars is a basic problem, it can be generalized to computation of any linear function of the node values and to averaging of vectors. Due to this capacity for generalization, algorithms that solve average consensus are attractive for a wide range of wireless sensor network applications.

Gossip algorithms can be synchronous or asynchronous. The synchronous version requires that at each iteration all nodes broadcast their values [37]. Having received the values of its neighbors, each node then updates its value with a weighted average of its value and the values it received. The asynchronous version, on the other hand, does not require synchronization and only one pair of nodes update at each iteration. In the remainder of this chapter when we refer to gossip algorithms, we refer to asynchronous gossip.

Randomized gossip algorithm describes a randomized and asynchronous version of gossip [3]. This algorithm restricts information exchange at each iteration to only a pair of neighboring nodes. Below we summarize the randomized gossip algorithm.

For a network of n nodes, let the undirected graph $\mathcal{G} = (V, E)$ represent the network connectivity where $V = \{1, \dots, n\}$ is the set of nodes, and $E \subseteq V \times V$ is the set of edges such that $(u, v) \in E$ if and only if nodes u and v can perform bidirectional wireless communication. The set of neighbors of node u (not including u itself) is denoted by $\mathcal{N}_u = \{v : (u, v) \in E\}$. The gossip iterations are indexed using $k = 1, 2, \dots,$ where $k = 0$ corresponds to the initial state. Each node $v \in V$ maintains a gossip value $x^v(k)$ which is initialized with $x^v(0) = x^v$.

Asynchronous time model [2]. A clock ticks at each node according to an independent rate-1 Poisson process. Since there are $|V| = n$ nodes, this is equivalent to there being a network coordinator running a Poisson clock with rate n, and when the coordinator's clock ticks, it assigns the tick to a node drawn uniformly from V. Each tick of the coordinator's clock corresponds to one iteration and we assume that the communication and update steps involved in each iteration occur instantaneously so that no two iterations overlap.

In a practical setting, the updates take some non-trivial amount of time. One could either tune the rate of the Poisson clocks at each node so that two updates overlap (e.g., leading to interference) with probability zero, or one could adopt a more complex scheduling mechanism to avoid interference. These issues are beyond the scope of this work.

Communication model. There is a pre-defined communication matrix P with entries $P_{u,v} \geq 0$ and $\sum_{v \in V} P_{u,v} = 1$. In addition, $P_{u,v} > 0$ if and only if $(u, v) \in E$. Suppose the kth clock tick occurs at node u. Then u contacts a random neighbor v which is drawn according to the distribution $\{P_{u,v}\}_{v \in V}$, and nodes u and v perform an update.

Update rule. When nodes u and v gossip, they update their values with the average,

$$x^u(k+1) = x^v(k+1) = \frac{1}{2}\big(x^u(k) + x^v(k)\big). \tag{10.2}$$

All other nodes $v' \in V \setminus \{u, v\}$ remain unchanged; i.e., $x^{v'}(k+1) = x^{v'}(k)$.

Intuitively, the convergence of randomized gossip is guaranteed if there exists a path between each pair of nodes so that information can flow between each pair infinitely often. Hence, one can show that, for a connected graph $\mathcal{G}$, under mild conditions on the way a random neighbor, v, is chosen, the values $x^u(k)$ converge to $\overline{x}$ at every node u as $k \to \infty$ [37]. The number of randomized gossip iterations required to achieve consensus scales with the number of nodes in the network; the rate of scaling depends on the network topology. For topologies that are generally used to model wireless sensor networks such as grids and random geometric graphs, randomized gossip converges slowly [3]. Motivated by this fact, there has been a body of work studying faster versions of gossip, e.g., [1, 6, 19, 24, 35].

Another research direction involves using gossip algorithms as a building box in complex signal processing applications (see [7] and references therein). Motivated by applications in distributed estimation, we study gossiping on vectors of data. Below we state this problem and propose selective gossip for efficient distributed approximation of vectors.

10.3 Gossiping on Vectors

The scalar average consensus problem described in the previous section can be immediately generalized to distributed averaging of vectors where, initially, each node $v \in V$ has a vector $\mathbf{x}^v \in \mathbb{R}^M$ and the aim is to compute the average

$$\bar{\mathbf{x}} = \frac{1}{n} \sum_{v=1}^{n} \mathbf{x}^v, \tag{10.3}$$

at each node v.

The basic solution to this problem is to run one scalar gossip algorithm for each dimension of the vector in parallel such that the entire average vector is computed at all nodes. Parallel gossip sessions can be implemented using the standard gossip setup with a modified update rule which involves exchanging and averaging vectors instead of scalars. Note that in practical sensor network scenarios, each wireless packet can carry only a certain amount of data and, consequently, exchanging long vectors may require several packets to be transmitted. Since energy consumption is proportional to the number of packets transmitted, exchange of long vectors instead of scalars increases the energy consumption of wireless communication. Increased number of packet transmissions also increases the bandwidth consumption of gossip updates.

However, often we only care about computing the largest entries of the average vector and not the entire vector. One example is decentralized field estimation where sensor nodes are deployed in an area to take scalar measurements [34]. Starting with local measurements, the goal is to reach a network state where each node has an approximation of the field. Transform coding is based on the idea that many natural signals are sparse (or nearly sparse) when they are transformed into a suitable domain. Hence, the signal representing the field can be well-approximated using only a few transform coefficients (those with the large magnitude). Assuming that a suitable transformation is available, one can use gossip algorithms to reach a consensus on the transform coefficients in a decentralized manner. Since only a few of the transform coefficients have large magnitudes, reaching a consensus only on these coefficients is satisfactory. The problem is that we do not know which coefficients have large magnitudes before actually computing them. Thus, any gossip algorithm that aims to decrease the communication cost by computing only these coefficients needs to also identify their locations.

Another example is the scenario where nodes must collectively decide among one of a large number of hypotheses. Initially, each node has its own data. Under the assumption that the likelihood of the data at different nodes is conditionally independent given the hypothesis, the network-wide log-likelihood of any hypothesis is simply the sum of the log-likelihoods at each node. However, if the number of hypotheses is very large, then it is more efficient for the nodes to focus their resources on computing the log-likelihood of only the most likely hypothesis or hypotheses, rather than all of them. In Sect. 10.5, we will consider the related setting of distributed particle filtering, where nodes gossip on the weights of particles which can be viewed as hypotheses.

Motivated by these applications in distributed signal processing and decision making, we study a method which adaptively identifies the largest elements of a vector while computing their values. The next section describes this method and provides results related to its performance.

10.4 Selective Gossip

To address the average consensus problem in multi-dimensional setting, we propose an efficient distributed averaging algorithm called *selective gossip* [34, 36]. Selective gossip conceptually builds on the randomized gossip algorithm described in [3]. In particular, we adopt the asynchronous time model and the communication model, explained in Sect. 10.2. The update rule, on the other hand, is where selective gossip differs from randomized gossip.

Each node $v \in V$ maintains a gossip vector $\mathbf{x}^v(k) \in \mathbb{R}^M$ at iteration k and this vector is initialized with $\mathbf{x}^v(0) = \mathbf{x}^v$. Let $x_j^v(k)$ represent the jth entry of $\mathbf{x}^v(k)$. The gossip vectors in the entire network at iteration k are denoted by $X(k) = \{\mathbf{x}^v(k)\}_{v \in V}$. Let $\bar{x}_{(i)}$ denote the ith highest entry of $\bar{\mathbf{x}}$, so that $\bar{x}_{(1)} \geq \bar{x}_{(2)} \geq \cdots \geq \bar{x}_{(M)}$.

We aim to reach a consensus on the locations and values of the largest entries of $\bar{\mathbf{x}}$. Depending on how the concept of *largest entries* is defined, the problem statement and the solution changes. Here we consider two possibilities:

1. **Threshold**. Given a non-negative threshold τ, let H_τ be the set of entries larger than the threshold, i.e.,
$$H_\tau = \{j : \bar{x}_j \geq \tau\}. \tag{10.4}$$
The goal is to have the iterates $X(k)$ approach $\mathcal{X}_\tau^*$ as efficiently as possible, where
$$\mathcal{X}_\tau^* = \left\{ \{\mathbf{x}^v\} : \text{ for all } v \in V, \begin{array}{l} x_j^v = \bar{x}_j \text{ if } j \in H_\tau \\ x_j^v < \tau \quad \text{if } j \notin H_\tau \end{array} \right\}. \tag{10.5}$$
If $X(k) \in \mathcal{X}_\tau^*$ then we say that the network has *reached consensus on* $\bar{\mathbf{x}}$ *for entries larger than the threshold* τ.
2. **Top-m**. Given a non-negative integer $m < M$, let $H_{\text{top-}m}$ be the set of entries of the highest m entries of $\bar{\mathbf{x}}$, i.e.,
$$H_{\text{top-}m} = \{j : \bar{x}_j \geq \bar{x}_{(m)}\}. \tag{10.6}$$
Note that the cardinality of $H_{\text{top-}m}$, denoted $|H_{\text{top-}m}|$, may in fact be larger than m, e.g., if $\bar{x}_{(m+1)} = \bar{x}_{(m)}$. Similar to above, the goal is to have the iterates $X(k)$ approach $\mathcal{X}_{\text{top-}m}^*$ as efficiently as possible, where
$$\mathcal{X}_{\text{top-}m}^* = \left\{ \{\mathbf{x}^v\} : \text{ for all } v \in V, \begin{array}{l} x_j^v = \bar{x}_j \quad \text{if } j \in H_{\text{top-}m} \\ x_j^v < \bar{x}_{(m)} \text{ if } j \notin H_{\text{top-}m} \end{array} \right\}. \tag{10.7}$$
If $X(k) \in \mathcal{X}_{\text{top-}m}^*$ then we say that the network has *reached consensus on* $\bar{\mathbf{x}}$ *for the largest m entries*.

Our goal is to *efficiently* reach a state $X(k) \in \mathcal{X}_\tau^*$ or $X(k) \in \mathcal{X}_{\text{top-}m}^*$. Our measure of efficiency aims to capture the amount of data communicated between nodes over the network. Specifically, we count the total number of scalar values transmitted. Of course, in order to obtain $X(k) \in \mathcal{X}^*_\tau$ or $X(k) \in \mathcal{X}^*_{\text{top-}m}$, one could run a standard distributed averaging algorithm [2, 3, 23] on each dimension, in which case standard results guarantee that $\mathbf{x}^v(k) \rightarrow \bar{\mathbf{x}}$ as $k \rightarrow \infty$ for all $v \in V$. Since $\bar{\mathbf{x}} \in \mathcal{X}^*_\tau$ and $\bar{\mathbf{x}} \in \mathcal{X}^*_{\text{top-}m}$, this achieves our objective in both cases. However, if $|H_\tau| \ll M$ or $m \ll M$, then this is wasteful since the nodes expend communication resources calculating entries which are not relevant. Selective gossip aims to achieve a network state in $\mathcal{X}^*_\tau$ or $\mathcal{X}^*_{\text{top-}m}$, but not necessarily one where any node computes the entire vector $\bar{\mathbf{x}}$. The main challenge is that the nodes do not know, a priori, the index set (H_τ or $H_{\text{top-}m}$) as it depends on the initial values, $X(0)$, and so it must also be estimated.

Below we present three versions of selective gossip; the first version addresses the threshold-based problem and the next two versions address the top-m problem.

10.4.1 Threshold Selective Gossip

Threshold selective gossip algorithm employs a threshold τ, which is fixed and known by all nodes, to determine which entries to communicate and update at each iteration. For a node $v \in V$, let $H^v_\tau(k)$ represent the entries with values higher than τ, i.e.,

$$H^v_\tau(k) = \{j : x^v_j(k) \geq \tau\}. \tag{10.8}$$

When nodes u and v wake up according to the asynchronous time model and communication model described in Sect. 10.2, they update entries that at least one of them believes to be one of the largest. Namely, they update only the entries $j \in H^u_\tau(k-1) \cup H^v_\tau(k-1)$ by setting

$$x^u_j(k) = x^v_j(k) = \frac{1}{2}\big(x^u_j(k-1) + x^v_j(k-1)\big). \tag{10.9}$$

No change is made to entries $j \notin H^u_\tau(k-1) \cup H^v_\tau(k-1)$, and these values are not transmitted in order to save energy. Also, all other nodes $v' \in V \setminus \{u, v\}$ keep their gossip vectors unchanged.

Threshold selective gossip asymptotically converges to the correct values for entries $j \in H_\tau$. Since there is no coupling between the different entries of the vector $\bar{\mathbf{x}}$, we treat each entry individually and focus on analyzing the behavior of the algorithm for a single scalar entry. Without loss of generality, let $x^v(0)$ denote the initial value for this entry at node v, let $\bar{x}$ denote the average, and let $\tau > 0$ be the given threshold. It is well known that, under the assumptions stated above, randomized gossip converges asymptotically to the average consensus [3]. Selective gossip differs from randomized gossip in that, at some iterations, two nodes may not update a particular entry. Thus, intuitively, to show convergence when $\bar{x} \geq \tau$ we just need to show that nodes gossip sufficiently often so that eventually they all have $x_v(k) \geq \tau$; at that point selective gossip is identical to randomized gossip.

Theorem 1 [34]. *Let $S(k) = \sum_{v=1}^{n} (x^v(k) - \bar{x})^2$ and suppose $\bar{x} \geq \tau$. Then*

$$\mathbb{E}[S(k)|S(0)] \leq \left(1 - \frac{1}{n^4 \operatorname{diam}(\mathcal{G})^2 \Delta_{\max}}\right)^k S(0), \tag{10.10}$$

where $\operatorname{diam}(\mathcal{G})$ *is the diameter of the network $\mathcal{G}$ and $\Delta_{\max} = \max_v |\mathcal{N}_v|$ is the maximum degree.*

Sketch of proof When a pair of neighboring nodes (u, v) decide to gossip at the kth iteration, $S(k)$ decreases such that $S(k+1) = S(k) - \frac{1}{2}\left(x^u(k) - x^t(k)\right)^2$. Taking the expectation over all pairs of neighboring nodes with non-zero probability of gossiping at iteration k, we get

$$\mathbb{E}[S(k+1)|S(k)] \leq S(k) - \frac{1}{n\Delta_{\max}}\left(x^u(k) - x^v(k)\right)^2. \tag{10.11}$$

Since consensus is not reached yet, there exists at least one node a with $x^a(k) \geq \bar{x} + \frac{1}{n}\sqrt{\frac{S(k)}{n}}$. Constructing a path from node a to any node b with $x^b(k) < \bar{x}$, we find that there exists a pair of neighboring nodes (a', b') on this path for which

$$(x^{a'}(k) - x^{b'}(k))^2 > \frac{S(k)}{n^3 \operatorname{diam}(\mathcal{G})^2},$$

and with (10.11) the statement of the theorem follows. □

Theorem 1 shows that for entries $j \in H_\tau$, selective gossip always computes the correct value in expectation. Furthermore, since $\mathbb{E}[S(k+1)|S(k)] \leq S(k)$ and $S(k) \geq 0$ for all k, the sequence $\{S(k)\colon k \geq 0\}$ is a non-negative supermartingale with respect to itself. Using the Martingale convergence theorem, one can show that the limit $S_\infty = \lim_{k\to\infty} S(k)$ exists almost surely [12]. Moreover, standard arguments [3] based on Markov's inequality can be applied to this result to show convergence in probability. Next we give the result for entries $j \notin H_\tau$.

Theorem 2 [34]. *Let $\mathcal{G} = \mathcal{K}_n$ be the complete graph. Suppose that $\bar{x} < \tau$ and $\tau - \bar{x} = c > 0$. If $S(0) > 0$ and there exists at least one node with non-zero probability of gossiping, then there exists a finite constant $K < \infty$ such that after $k \geq K$ iterations, $x^v(k) < \tau$ for all nodes v with probability 1.*

Sketch of proof In this case, one can find two nodes (u, v) such that $(x^u(k) - x^v(k))^2 \geq c^2$. Since $\Delta_{\max} = n - 1$ for the complete graph and using the bound (10.11), we get $\mathbb{E}[S(k)|S(0)] \leq S(0) - \frac{kc^2}{n(n-1)}$. Applying Markov's inequality yields

$$\Pr\left(S(k) \geq c^2|S(0)\right) \leq \frac{S(0)}{c^2} - \frac{k}{n(n-1)}.$$

Therefore, if $k \geq K = \frac{n(n-1)}{c^2} S(0)$, then $x^v(k) < \tau$, for all v with probability 1. □

Theorem 2 addresses the case where $\bar{x} < \tau$ only for the complete graph. This approach does not directly extend to general connected topologies. In particular, in the proof of Theorem 2, one cannot guarantee that the nodes u and v will be neighbors in a general topology. However, the convergence can be shown using an approach similar to that presented below for the proof of Theorem 3.

It is also worth noting that the bounds given in Theorems 1 and 2 are extremely loose since we only consider the gossiping of one pair of nodes instead of all pairs, and hence these bounds should not be taken as an indicator of the rate of convergence. In fact, it is easy to see that once all nodes agree that an entry j is in H_τ, threshold selective gossip behaves identically to randomized gossip, and so asymptotically the rates of convergence are the same as reported in [3] for randomized gossip. As illustrated in the simulations presented below, the error decay rate of threshold selective gossip, as a function of the number of scalar values transmitted, is in fact substantially faster than running randomized gossip in parallel for all entries.

10.4.2 Adaptive Threshold Selective Gossip

Threshold selective gossip requires a fixed preset threshold, τ, to determine the entries to be computed. However, having a fixed threshold is typically not practical since we may not have accurate prior knowledge of the distribution of values in the average vector. To address this problem, we describe a heuristic called *adaptive threshold selective gossip* which aims to find the appropriate threshold at each node in a decentralized way. By appropriate threshold, we mean $\tau \in (\bar{x}_{(m)}, \bar{x}_{(m+1)})$, where m is given as input to the algorithm. In other words, our heuristic deals with the top-m problem and tries to reach the index set $H_{\text{top-}m}$ by adaptively changing the threshold at every node. For this, each node keeps an estimate of the threshold as well as the gossip vectors $\mathbf{x}^v(k)$.

Let the threshold estimate of each node v be denoted by $\tau^v(k)$ at time k. The threshold estimate of each node is initialized with the mth largest entry of its gossip vector, i.e., $\tau^v(0) = x^v_{(m)}(0)$. When two nodes u and v perform a gossip update, they modify the entries $j \in H^u_{\tau^u(k-1)}(k-1) \cup H^v_{\tau^v(k-1)}(k-1)$ by setting

$$x^u_j(k) = x^v_j(k) = \frac{1}{2}\bigl(x^u_j(k-1) + x^v_j(k-1)\bigr). \qquad (10.12)$$

All other entries remain unchanged and all other nodes keep their gossip vectors unchanged. After the update, nodes u and v reassess their approximation quality. If the current threshold of node v provides fewer than m entries in $H^v_\tau(k)$, then the node decreases its threshold. If the node has more than m entries in $H^v_\tau(k)$, then the threshold value is increased. Specifically, node v updates its threshold according to the following rule

$$\tau^v(k+1) = \begin{cases} (1+c_1)\tau^v(k) & |H^v_\tau(k)| > m \\ (1-c_2)\tau^v(k) & |H^v_\tau(k)| < m \\ \tau^v(k) & |H^v_\tau(k)| = m \end{cases} \tag{10.13}$$

where $c_1, c_2 > 0$ are predefined constants. Note that we choose $c_1 \neq c_2$ as having $c_1 = c_2$ may cause undesirable oscillations in the threshold estimates.

The adaptive threshold heuristic does not have any convergence guarantees but intuitively should be more efficient than randomized gossip since it aims to compute only the largest entries of the average vector. We present simulation results in the upcoming sections to illustrate the performance of this method.

10.4.3 Top-m Selective Gossip

Since the adaptive threshold version of selective gossip is a heuristic without convergence guarantees, we propose another variation of gossip that solves the top-m problem and also has provable guarantees. Top-m selective gossip takes a positive integer m as an input and adaptively focuses communication on the largest m entries of the gossip vectors.

Let $x^v_{(m)}(k)$ denote the mth largest value in the gossip vector $\mathbf{x}^v(k)$ at node v, and let $H^v_{\text{top-}m}(k)$ denote the set of largest m indices of node v, i.e.,

$$H^v_{\text{top-}m}(k) = \{j : x^v_j(k) \geq x^v_{(m)}(k)\}. \tag{10.14}$$

When nodes u and v perform an update, they first exchange those entries of their gossip vectors which at least one of them believes to be among the m largest; i.e., they exchange values for entries $j \in H^u_{\text{top-}m}(k-1) \cup H^v_{\text{top-}m}(k-1)$. Then, they update

$$x^u_j(k) = x^v_j(k) = \frac{1}{2}\big(x^u_j(k-1) + x^v_j(k)(k-1)\big), \tag{10.15}$$

for entries $j \in H^u_{\text{top-}m}(k-1) \cup H^v_{\text{top-}m}(k-1)$, and they set $x^u_j(k) = x^u_j(k-1)$ and $x^v_j(k) = x^v_j(k-1)$ for entries $j \notin H^u_{\text{top-}m}(k-1) \cup H^v_{\text{top-}m}(k-1)$. Likewise, the gossip vectors of all nodes $v' \in V \setminus \{u, v\}$ who do not participate in the update remain unchanged; i.e., $\mathbf{x}^{v'}(k) = \mathbf{x}^{v'}(k-1)$.

Although the threshold and top-m approaches appear similar at first glance, there are subtle differences which make top-m selective gossip considerably more challenging to analyze. When the aim is to compute all entries which exceed a threshold, the updates applied to each entry of the vector can be decoupled, since the final result only depends on whether the average for that entry does or does not exceed the threshold. On the other hand, when the aim is to compute the largest m entries of the average vector, all entries are coupled since the final result depends on the rank ordering. Subsequently, a different approach is required to show convergence.

The following theorem shows that this algorithm converges asymptotically on any connected graph to a state where all nodes agree on the indices and values of the m largest entries, where m is a given parameter.

Theorem 3 [36]. *The gossip vectors generated by top-m selective gossip converge to a limit* $\{\mathbf{x}^v(k)\}_{v\in V} \to \{\tilde{\mathbf{x}}^v\}_{v\in V}$ *as* $k \to \infty$, *where*

$$\begin{aligned}\tilde{x}_j^v &= \bar{x}_j, \quad \text{for } j \in H_{top\text{-}m}, \quad v \in V,\\ \tilde{x}_j^v &< \bar{x}_{(m)}, \quad \text{for } j \notin H_{top\text{-}m}, \quad v \in V.\end{aligned}$$

Sketch of proof Let $\mathbf{x}_j(k) \in \mathbb{R}^n$ denote the jth entry at each node, $x_j^v(k)$, stacked into a vector. Observe that the update Eqs. (10.14) and (10.15) for top-m selective gossip, can be written as a collection of linear updates,

$$\mathbf{x}_j(k) = \mathbf{W}_j(k)\mathbf{x}_j(k-1), \quad \text{for } j = 1, 2, \ldots, M \tag{10.16}$$

where $\mathbf{W}_j(k)$ is time-varying and depends on the entire state $X(k)$ through the sets $H^v_{\text{top-}m}$ as described next. Let $[\mathbf{W}]_{u,v}$ the (u, v)th entry of the matrix $\mathbf{W}$. Suppose that nodes u and v perform the kth gossip update. If $j \in H^u_{\text{top-}m}(k-1) \cup H^v_{\text{top-}m}(k-1)$, then

$$[\mathbf{W}_j(k)]_{u,u} = [\mathbf{W}_j(k)]_{u,v} = [\mathbf{W}_j(k)]_{v,u} = [\mathbf{W}_j(k)]_{v,v} = \frac{1}{2}, \tag{10.17}$$

and

$$[\mathbf{W}_j(k)]_{u',u'} = 1, \quad [\mathbf{W}_j(k)]_{u',v'} = 0, \tag{10.18}$$

for all $u', v' \notin \{u, v\}$, since only nodes u and v update their gossip vector, and all other nodes make no changes. If $j \notin H^u_{\text{top-}m}(k-1) \cup H^v_{\text{top-}m}(k-1)$, then no node updates this entry of the gossip vector, and $\mathbf{W}_j(k) = I$. In particular, note that every matrix $\mathbf{W}_j(k)$ is symmetric and doubly stochastic with non-zero entries at least $1/2$.

Recent theory [16, 31] for time-varying linear systems of the form (10.16) makes it possible to characterize the behavior of the limit $\lim_{k\to\infty} \mathbf{x}_j(k)$. Specifically, for matrices such as $\mathbf{W}_j(k)$ satisfying the properties mentioned above, the limit $\tilde{\mathbf{x}}_j = \lim_{k\to\infty} \mathbf{x}_j(k)$ exists. In addition, consider the graph $G_j = (V, E_j)$ with $(u, v) \in E_j$ if $[\mathbf{W}_j(k)]_{u,v}$ infinitely often. If G_j is connected (i.e., if there is a path in G_j connecting every pair of nodes), then $\tilde{x}_j^u = \tilde{x}_j^v$ for all u, v; i.e., all nodes asymptotically reach a consensus on the jth entry of the gossip vector. Moreover, since every $\mathbf{W}_j(k)$ is doubly stochastic, $\tilde{x}_j^v = \bar{x}_j = \frac{1}{n}\sum_u x_j^u(0)$, and the nodes reach a consensus on the average. Thus, to determine which entries of the gossip vectors converge to the average (if any), we need to characterize which entries are updated infinitely often as $k \to \infty$.

From the definition of the asynchronous time model, since all nodes initiate updates according to a rate-1 Poisson process, it follows that as $k \to \infty$, every node will participate in an infinite number of updates. Each time an update is performed, the nodes u and v update those entries in the set $H^u_{\text{top-}m}(k-1) \cup H^v_{\text{top-}m}(k-1)$, which contains at least m elements (more if the two sets are not identical). Thus, there exists a set of indices $\mathcal{J}$ which are updated infinitely often, and thus for $j \in \mathcal{J}$, the limit $\tilde{\mathbf{x}}_j$ is the consensus vector with all elements equal to $\bar{x}_j$. Moreover, those indices $i \notin \mathcal{J}$ are only updated a finite number of times. It remains to be shown that $\mathcal{J} \equiv H_{\text{top-}m}$.

Suppose that $j \in H_{\text{top-}m}$. It follows that at every iteration k there exists a node u_k such that $x_j^{u_k}(k) \geq \bar{x}_j \geq \bar{x}_{(m)}$, and so $j \in H_{\text{top-}m}^{u_k}(k)$. Thus there is a non-zero probability of element j being updated at every iteration (since there is a non-zero probability that node u_k will participate in the update), and it follows that $j \in \mathcal{J}$ and $\tilde{x}_j^v = \bar{x}_j$ for all $v \in V$ and $j \in H_{\text{top-}m}$.

Next, suppose that $j \notin H_{\text{top-}m}$ and $j \in \mathcal{J}$. Since $j \notin H_{\text{top-}m}$, we have $\bar{x}_j < \bar{x}_{(m)}$. As more updates are performed on entry j, the entries $x_j^v(k)$ for all nodes v approach $\bar{x}_j$. At some time k', it is necessarily true that $\max_v x_j^v(k') < \min_v \min_{i \in H_{\text{top-}m}} x_i^v(k') \leq \bar{x}_{(m)}$. But then $j \notin H_{\text{top-}m}^u(k')$ for any node u, and so entry j will no longer be updated. Thus, the values of entries $j \notin H_{\text{top-}m}$ converge to a limit $\tilde{x}_j^v < \bar{x}_{(m)}$ which may be different at every node. $\square$

Note that the goals stated above can also be generalized to cases where one aims to reach a consensus on the largest entries in absolute value; i.e., entries with $|\bar{x}_j| \geq \tau$, or sorting the entries according to magnitude, $|\bar{x}_{(1)}| \geq |\bar{x}_{(2)}| \geq \dots$, when defining which are the m most significant. For example, in the decentralized field estimation application where transform coefficients can be computed via selective gossip, it may be more meaningful to compute the m entries (transform coefficients) with largest magnitude, rather than simply the largest m coefficients. All three versions of selective gossip can be modified to address this formulation, at the expense of more cumbersome notation. This extension has been performed for threshold selective gossip and the results, along with a comparison to the corresponding version of adaptive threshold selective gossip, are reported in [34]. We expect that a similar extension for top-m selective gossip should be possible using similar techniques.

10.4.4 Simulation Results

In this section we demonstrate the performance of selective gossip through numerical experiments. The simulation setup consists of a network of $n = 50$ nodes which are distributed uniformly at random in the unit square. The communication topology is a random geometric graph, i.e., there is an edge between two nodes that are within a distance r from each other. This distance is set to $r = \sqrt{2 \log n / n}$ so that the graph is connected with high probability [14]. The dimension of the gossip vectors is $M = 25$.

To generate an initial network state, $X(0)$, we first determine the average vector, $\bar{\mathbf{x}}$. The top panels of Figs. 10.1 and 10.2 show two different vectors $\bar{\mathbf{x}}$ used in our experiments. The first one has a clear separation between the averages of the first 5

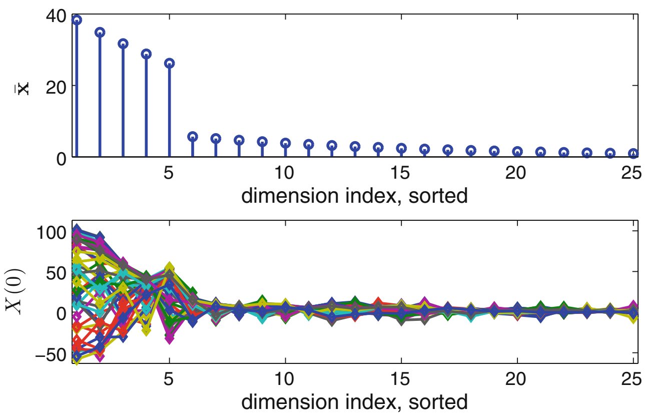

Fig. 10.1 *Top*: The average vector $\bar{\mathbf{x}}$ in descending order for initialization 1. *Bottom*: The initial state of the network with indices in the same order as $\bar{\mathbf{x}}$ above. *Diamonds* represent $x_j^v(0)$ and those that belong to the same node are connected with a *solid line*

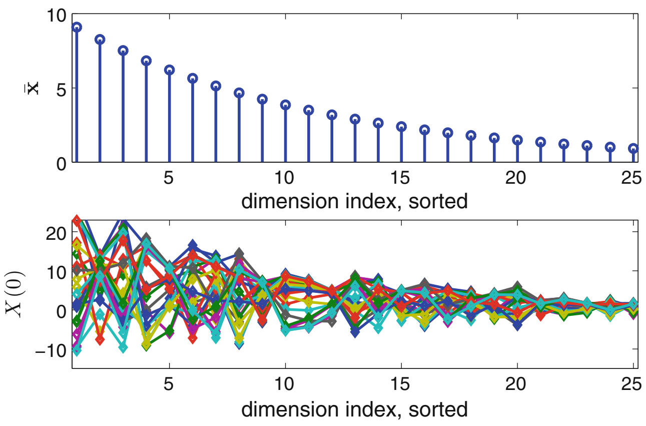

Fig. 10.2 *Top*: The average vector $\bar{\mathbf{x}}$ in descending order for initialization 2. *Bottom*: The initial state of the network with indices in the same order as $\bar{\mathbf{x}}$ above. *Diamonds* represent $x_j^v(0)$ and those that belong to the same node are connected with a *solid line*

indices and the rest, making $m = 5$ a natural choice. The second average vector is more smoothly distributed across its dimensions.

Motivated by applications in sensor networks, we assume that the node values represent measurements of natural phenomena. For each index j, we select a point μ_j uniformly at random in the unit square. Then for each node v we generate $x_j^v(0)$ such that the nodes geographically closer to μ_j will have higher values and the average over all nodes is equal to $\bar{x}_j$. The initial values are distributed such that the highest m indices at each node are not necessarily the same as $H_{\text{top-}m}$. The bottom panels of Figs. 10.1 and 10.2 illustrate the vectors $\{x_j^v(0)\}_{v\in V}$ and how they are distributed over the network.

We compare the performance of adaptive threshold selective gossip and top-m selective gossip with the performance of randomized gossip [3]. Randomized gossip is guaranteed to converge to $\bar{\mathbf{x}}$, but it is wasteful since the nodes gossip on every entry of the gossip vector at every iteration. Since randomized gossip computes every entry of $\bar{\mathbf{x}}$ it is equivalent to running top-m selective gossip with $m = M$.

The performance is measured with the mean squared error which is defined as

$$MSE(k) = \frac{1}{n}\sum_{v\in V}\sum_{j\in H_{\text{top-}m}}\left(x_j^v(k) - \bar{x}_j\right)^2.$$

Since we are interested in the amount of data that is communicated during the course of gossip, we plot the error against the number of scalars that are transmitted instead of the iterations k. Figures 10.3 and 10.4 compare the MSE of the three algorithms for different values of m. The results show the average performance over 500 different realizations of gossip.

Since randomized gossip updates all entries of gossip vectors at every iteration, its performance is the same as the performance of selective gossip for $m = 25$. In fact, for $m = 25$ all three methods perform the same, and hence their MSE curves overlap. For other values, we can see that the performance of top-m selective gossip is always better than that of adaptive threshold selective gossip.

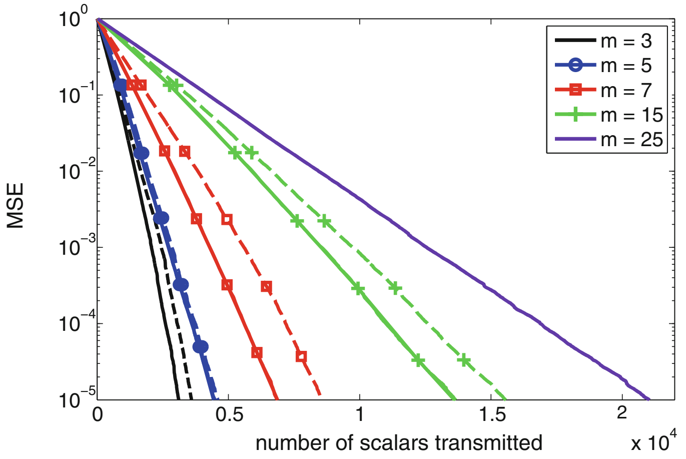

Fig. 10.3 A comparison of error performances for initialization given in Fig. 10.1. The algorithms that are compared are top-m selective gossip *(solid)* and adaptive threshold selective gossip *(dashed)* for varying m values and randomized gossip (corresponds to $m = 25$ in the plot as it updates all entries at each iteration). The plot illustrates the performance averaged over 500 realizations of gossip

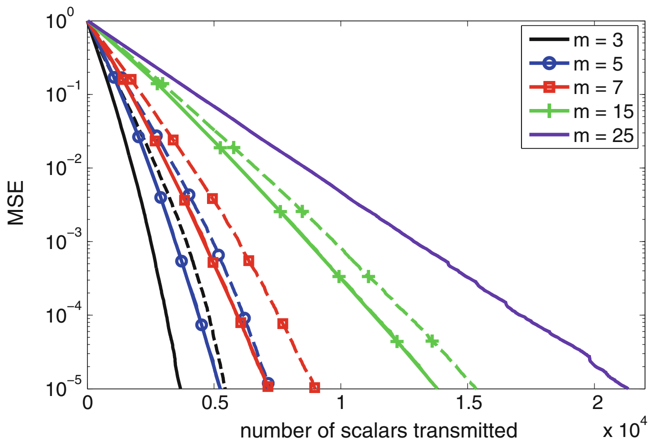

Fig. 10.4 A comparison of error performances for initialization given in Fig. 10.2. The algorithms that are compared are top-m selective gossip *(solid)* and adaptive threshold selective gossip *(dashed)* for varying m values and randomized gossip (corresponds to $m = 25$ in the plot as it updates all entries at each iteration). The plot illustrates the performance averaged over 500 realizations of gossip

The effects of varying m can be seen in Fig. 10.4 for the initialization shown in Fig. 10.1. The difference between the top-m and adaptive threshold versions of selective gossip is minimal when m is equal to the number of entries of $\bar{\mathbf{x}}$ that are significantly higher than the rest. For the initialization of Fig. 10.2, the adaptive threshold version performs worse for every m. In particular, for low values of m, top-m selective gossip computes more entries of the average vector with the same number of transmitted scalars compared to the adaptive threshold version.

To investigate how well one could hope to do using top-m selective gossip, we also implement a version of top-m selective gossip where every node clairvoyantly knows $H_{\text{top-}m}$ from the start and only updates entries $j \in H_{\text{top-}m}$ at each iteration. The corresponding results are shown in Figs. 10.5 and 10.6.

10.5 Distributed Tracking Using Selective Gossip

In this section we propose a distributed tracking algorithm that utilizes selective gossip. Before explaining the details of the algorithm, we provide some background on the problem.

Tracking is an important task in wireless sensor networks. The goal of tracking is to estimate the state of a dynamical system sequentially in time using measurements recorded by the sensors. For example, the state can be the position and the velocity of a moving target or, in the case of monitoring environmental conditions, it can represent the soil moisture and temperature. In these scenarios, we do not have direct

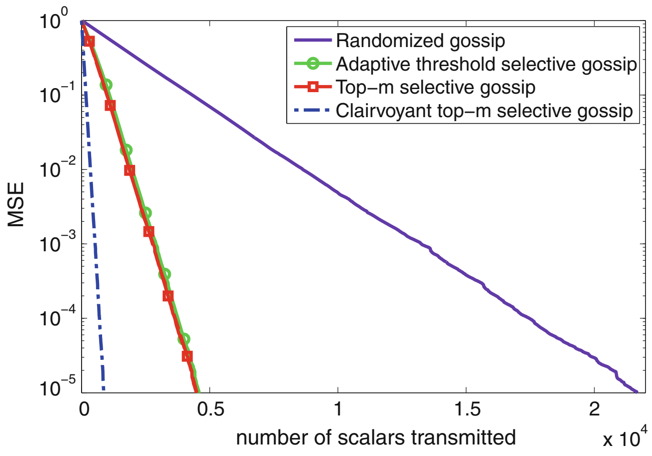

Fig. 10.5 A comparison of performances of randomized gossip, top-m selective gossip, adaptive threshold selective gossip for the initialization given in Fig. 10.1 and $m = 5$. The plot also includes clairvoyant top-m selective gossip which updates only the entries in $H_{\text{top-}m}$ at each iteration

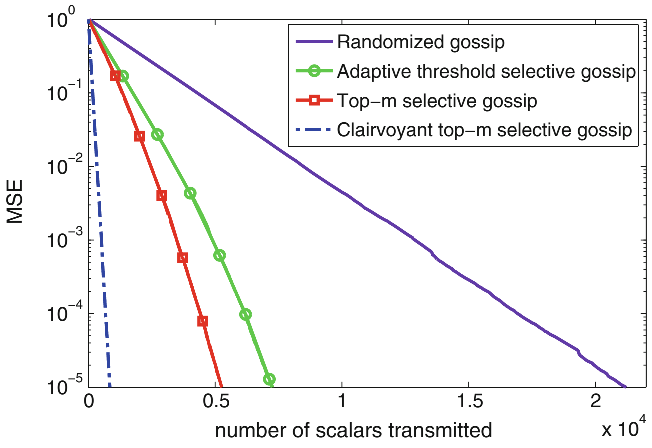

Fig. 10.6 A comparison of performances of randomized gossip, top-m selective gossip, adaptive threshold selective gossip for the initialization given in Fig. 10.2 and $m = 5$. The plot also includes clairvoyant top-m selective gossip which updates only the entries in $H_{\text{top-}m}$ at each iteration

access to the state of the dynamical system. Instead, the state can only be observed via the noise-corrupted measurements of the sensors.

The sequential estimation problem arises in many areas including robotics, tracking, financial econometrics and computer vision (see [4, 8, 28] and the references therein). The optimal estimator for this problem when the dynamics and observation models are linear and the noise distributions are Gaussian is the well-known Kalman filter. However, many practical scenarios (e.g., the tracking of a maneuvering target) involve nonlinearities and/or non-Gaussian noise, in which case the Kalman filter does not apply. Some popular approaches for more general settings are the extended Kalman filter, the Gaussian sum filter, the unscented Kalman filter, and particle filter methods (also known as sequential Monte Carlo methods) [28]. Due to their flexibility, ease of implementation, and performance, particle filter methods are widely accepted as the state-of-the-art approach to sequential estimation for the case of nonlinear dynamic models and non-Gaussian noise distributions [8, 9].

10.5.1 Sequential Estimation

In this section we review the sequential estimation problem, adopting definitions and terminology from [4, 5, 9, 28].

The state-space modeling framework describes the state of the system as an unobserved Markov process denoted by $\{\mathbf{y}_t\}_{t\in\mathbb{N}}$. The state evolution is determined by the initial distribution $p(\mathbf{y}_0)$ and the transition distribution $p(\mathbf{y}_t|\mathbf{y}_{t-1})$. The observations $\{\mathbf{z}_t\}_{t\in\mathbb{N}^+}$ are assumed to be conditionally independent given the state $\mathbf{y}_t$, and they are of marginal distribution $p(\mathbf{z}_t|\mathbf{y}_t)$. Such state-space models are also known as hidden Markov models.

The goal is to characterize the distribution of the state at the present time using the information provided by all observations received up to the present time. Let the sequence of states up to time t be denoted by $\mathbf{y}_{0:t}$ and let the sequence of observations up to time t be denoted by $\mathbf{z}_{1:t}$. We are interested in sequential estimation of the posterior distribution $p(\mathbf{y}_{0:t}|\mathbf{z}_{1:t})$ and the filtering distribution $p(\mathbf{y}_t|\mathbf{z}_{1:t})$.

The analytical solution is available as a two-stage recursion for both the posterior and filtering distributions. The stages of the recursion for the filtering distribution are termed prediction and update steps, and are presented in the following format:

$$\text{Prediction:} \quad p(\mathbf{y}_t|\mathbf{z}_{1:t-1}) = \int p(\mathbf{y}_t|\mathbf{y}_{t-1})p(\mathbf{y}_{t-1}|\mathbf{z}_{1:t-1})d\mathbf{y}_{t-1} \tag{10.19}$$

$$\text{Update:} \quad p(\mathbf{y}_t|\mathbf{z}_{1:t}) = \frac{p(\mathbf{z}_t|\mathbf{y}_t)p(\mathbf{y}_t|\mathbf{z}_{1:t-1})}{p(\mathbf{z}_t|\mathbf{z}_{1:t-1})} \tag{10.20}$$

where, assuming $p(\mathbf{y}_{t-1}|\mathbf{z}_{1:t-1})$ is available, the system model is used to predict the prior distribution at time t and the observation $\mathbf{z}_t$ is used in the second stage to update the prior via Bayes' rule.

10.5.2 Particle Filtering

Particle filters approximate the distributions $p(\mathbf{y}_{0:t}|\mathbf{z}_{1:t})$ and $p(\mathbf{y}_t|\mathbf{z}_{1:t})$ by a set of random samples termed particles. These particles are candidates for the state and their associated weights represent the accuracy of the estimate. Particle filters, also known as sequential Monte Carlo methods, have been around since the 1960s [15], but due to their computational complexity they were not widely used. The early implementations also suffered from particle degeneracy which is due to the increase in variance of weights over time. After some iterations, many particles have negligible weights and thus do not contribute to the estimation. This problem was solved in 1993 by Gordon et al. with the introduction of resampling [11].

The sequential importance resampling (SIR) particle filter maintains a weighted particle approximation $\{\mathbf{y}_{1:t}^{(i)}, w_t^{(i)}\}_{i=1}^{M}$ to estimate a posterior of interest $p(\mathbf{y}_{1:t}|\mathbf{z}_{1:t})$. The posterior is estimated by the distribution

$$\hat{p}_M(\mathbf{y}_{1:t}|\mathbf{z}_{1:t}) = \frac{1}{M}\sum_{i=1}^{M} w_t^{(i)}\delta(\mathbf{y}_{1:t} - \mathbf{y}_{1:t}^{(i)}), \tag{10.21}$$

where $\delta(\cdot)$ is the Dirac delta function.

Assuming it has a weighted particle approximation at time $t-1$, SIR propagates the particles to time t by sampling from an importance function q, evaluates the likelihoods of the extended particles, and updates the weights accordingly. A common approach is to use the prior as the importance function, i.e., $q = p(\mathbf{x}_t|\mathbf{z}_{t-1}^{(i)})$. Then there is an optional resampling step to construct a set of particles with more evenly distributed weights. Resampling replicates particles with high weights and discards particles with low weights. In [11] the prior is used as the importance function and resampling is done at every step. The authors call this implementation the bootstrap particle filter. Algorithm 1 provides the pseudo-code for the bootstrap particle filter algorithm.

10.5.3 Particle Filters in Wireless Sensor Networks

One approach to implement particle filters in networks is the leader node framework [38]. One node is selected as leader and all nodes send their measurements to this node. The leader node runs a centralized particle filter using all the information from the network. This leader node may change over time to distribute the responsibility of processing among the nodes. Being centralized, the leader node framework allows only the leader node to be queried and introduces a single point of failure. In addition, to be able to process the raw measurements of the sensors, the leader node needs to know the observation models, sensor locations, and calibration parameters of the sensors. Since only the leader node has access to the output of the particle filter, it must also make sensor management decisions such as which nodes take measurements next and with which modality.

Another approach is to distribute the computation. Each node calculates its local likelihood and the information is fused to form a global posterior. Virtually all such distributed filters rely on an assumption of conditional independence of the measure-

Algorithm 1 Bootstrap particle filter

```
// Initialization at time $t = 1$
1. For each particle $i = 1, \dots, M$ do
     • Sample $\mathbf{y}_1^{(i)} \sim q_1(\cdot)$
     • Set $w_1^{(i)} = \frac{p(\mathbf{z}_1|\mathbf{y}_1^{(i)})p(\mathbf{y}_1^{(i)})}{q_1(\mathbf{y}_1^{(i)})}$
2. end
3. Normalize weights $w_1^{(i)}$ so that $\sum_{i=1}^{M} w_1^{(i)} = 1$
4. Resample $\left\{\mathbf{y}_1^{(i)}, w_1^{(i)}\right\}_{i=1}^{M}$ to obtain $\left\{\mathbf{y}_1'^{(i)}, \frac{1}{M}\right\}_{i=1}^{M}$
5. For times $t > 1$:
// For each particle $i = 1, \dots, M$ do
     • Set $\mathbf{y}_{1:t-1}^{(i)} = \mathbf{y}_{1:t-1}'^{(i)}$
     • Sample $\mathbf{y}_t^{(i)} \sim q(\mathbf{y}_t|\mathbf{y}_{t-1}^{(i)})$
     • Set $w_t^{(i)} = \frac{p(\mathbf{z}_t|\mathbf{y}_t^{(i)})p(\mathbf{y}_t^{(i)}|\mathbf{y}_{t-1}^{(i)})}{q(\mathbf{y}_t|\mathbf{y}_{t-1}^{(i)})}$
6. end
7. Normalize weights $w_t^{(i)}$ so that $\sum_{i=1}^{M} w_t^{(i)} = 1$
8. Resample $\left\{\mathbf{y}_{1:t}^{(i)}, w_t^{(i)}\right\}_{i=1}^{M}$ to obtain $\left\{\mathbf{y}_{1:t}'^{(i)}, \frac{1}{M}\right\}_{i=1}^{M}$
```

ments made at each node given the target state. Several of these distributed particle filters require a spanning tree or Hamiltonian cycle for communication [5, 29]. Construction and maintenance of such routes can be very challenging when nodes are mobile or wireless conditions are adverse. Hence the algorithms are highly vulnerable to link and node failures.

Alternatively, gossip algorithms can be used for distributing the computation [10, 13, 17, 20, 21, 25]. The algorithm in [13] uses the expectation-maximization (EM) algorithm based on gossip to estimate the parameters of a mixture approximation to the global posterior, but it imposes significant constraints on the structure of the likelihood function. In the procedure in [25], each node forms a Gaussian approximation to a local posterior and then a gossip algorithm is used to fuse the means and covariance matrices to construct a Gaussian approximation of the global posterior. This algorithm has a much lower communication overhead, but its accuracy diminishes when posteriors cannot be adequately approximated by a Gaussian. The method presented in [17] constructs a polynomial approximation of the joint likelihood at each node using distributed averaging. Hence this algorithm also involves reduced communication overhead and is restricted to certain types of likelihood functions.

The algorithms in [10, 20] do not form parametric approximations to the posterior; instead they share particles among different nodes. In [20], particles undergo a random walk through the sensor network, and their weights are successively multiplied by a function of the local likelihood. The function is carefully chosen so that the particle weights converge to the same values that a centralized particle filter would calculate. This algorithm has attractive properties, but it only supports importance-sampling from the prior, which can lead to poor performance of a particle filtering algorithm [9]. The algorithm in [20] also has no mechanism for eliminating particles with small weights, leading to wasteful communication.

The algorithm in [10] was designed to allow sampling from a better importance distribution (one that better matches the posterior). It estimates regions of concentrated mass in the global posterior by calculating the intersection of the regions of concentration in the local posteriors. The importance sampling function is then constructed to focus on the calculated region, and the gossip procedure is used to calculate the global likelihoods and hence the particle weights. This procedure achieves high accuracy, but the communication cost (in terms of number of values exchanged) is high because the weights of all particles must be calculated, even if many are very small. Also, the computation of regions of concentration requires oversampling of particles at each node, increasing the local computation complexity.

To improve upon currently available algorithms, we propose using selective gossip in a distributed implementation of the bootstrap particle filter. The next section describes our problem statement.

10.5.4 Distributed Tracking Problem Statement

We consider a wireless sensor network consisting of n nodes and represent network connectivity as a graph, $\mathcal{G} = (V, E)$. We assume that the graph is connected, and that although nodes are unaware of the global topology, they do have the knowledge of their neighbors. The goal is to sequentially estimate a state, denoted by $\mathbf{y}_t$ at time index t. The state may represent a target's kinematics, typically position and velocity, or a set of environmental conditions, such as temperature, wind speed, or soil moisture. Let d be the dimension of the state, i.e., $\mathbf{y}_t \in \mathbb{R}^d$. At time t, node v makes a noisy measurement $\mathbf{z}_t^v$. The set of all measurements made by the network at time t is then $\mathbf{z}_t^V = \{\mathbf{z}_t^v : v \in V\}$ and the joint likelihood of these measurements is given by the function $p(\mathbf{z}_t^V|\mathbf{y}_t)$.

Nodes do not have access to the measurement modalities, noise models, or calibration parameters of other nodes in the network. Hence they cannot process raw measurements from other nodes. However we assume that the noise distributions at different nodes are conditionally independent given the state. Therefore the joint likelihood can be factorized into

$$p(\mathbf{z}_t^V|\mathbf{y}_t) = \prod_{v \in V} p(\mathbf{z}_t^v|\mathbf{y}_t), \tag{10.22}$$

where $p(\mathbf{z}_t^v|\mathbf{y}_t)$ is the likelihood of the observation made by node v.

Since the global likelihood is factorisable, its computation can be reduced to a set of local tasks at nodes followed by a final networked aggregation step. We are interested in a particle filter implementation that takes advantage of this factorization and achieves decentralized sequential estimation. Because wireless sensor networks have battery and bandwidth constraints, the distributed implementation needs to be efficient in terms of the number of values exchanged.

10.5.5 Distributed Particle Filter Using Selective Gossip

We now present our distributed particle filter algorithm which is based on the bootstrap particle filter. In this algorithm, every node in the network runs a copy of the same particle filter provided that the following two conditions hold. First, the measurements at nodes are synchronized so that measurements made at the same time index reflect the same state at all nodes. Second, the random number generators of the nodes are synchronized (e.g., the nodes use pseudo random generators initialized with the same seed). This ensures that nodes sample the same values when they are given the same set of weighted particles as input. These two conditions can be achieved via a decentralized routine that is executed before the sequential estimation.

The challenge in implementing distributed particle filters lies in the fact that the global weights depend on the measurements $\mathbf{z}_t^V$, but each node v only has access to its own measurement $\mathbf{z}_t^v$. We address this challenge by exploiting the factorization of the global likelihood and the fact that the computation of global weights is reduced to local computation tasks which need to be followed by a multiplication procedure. The local tasks can be performed independently at each node and do not require knowledge of the modality, noise, or calibration details of other nodes. Instead of multiplication, we use summation in the logarithm domain, which is suitable for distributed averaging.

We start by introducing local pre-weights $\{\phi_t^{v,(i)}\}_{i=1}^M$ where $\phi_t^{v,(i)} = n \log p(\mathbf{z}_t^v|\mathbf{y}_t^{(i)})$. Then the weight of particle i can be expressed using local pre-weights as

$$w_t^{(i)} = \frac{\exp(\frac{1}{n}\sum_{v\in V}\phi_t^{v,(i)})p(\mathbf{y}_t^{(i)}|\mathbf{y}_{t-1}^{(i)})}{q(\mathbf{y}_t|\mathbf{y}_{t-1}^{(i)})}. \tag{10.23}$$

Hence the weights can be calculated via averaging of an $M-$dimensional vector equation. Once the weights are computed, the bootstrap filter requires a normalization and resampling step so that particles are more evenly distributed. In particular, resampling discards particles with low weights and replicates the particles that have high weights. Figure 10.7 illustrates the distribution of particle weights for an example filter running with $M = 2000$ particles. Most of the particles have low weights and since particles with low weights are not to be kept, computing their values via distributed averaging wastes scarce network resources. Hence we are interested in computing only the weights that are high instead of computing all weights $\{w_t^{(i)}\}_{i=1}^M$.

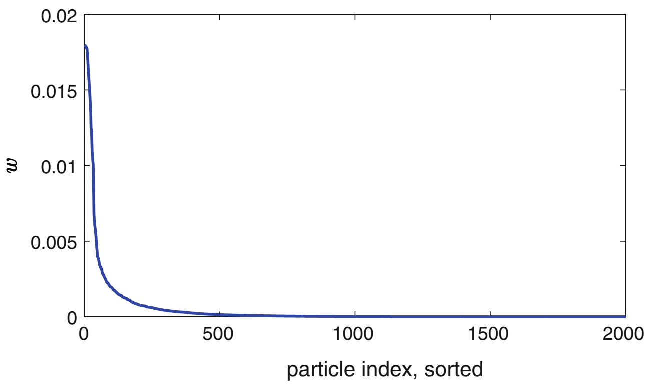

Fig. 10.7 The distribution of particle weights, sorted in descending order

Of course the challenge is that nodes do not know which weights are higher from only the local information that they have.

We propose to use selective gossip to focus communication on only the highest m weights. With the input of local pre-weights, $\{\phi^v\}_{v=1}^n$, at n nodes, and the given integer m, selective gossip identifies the set $H_{\text{top-}m}$ of the particles with the highest m weights and provides each node with the pre-weight estimates of these particles, $\{\widetilde{\phi}^{v,(H_{\text{top-}m})}\}_{v=1}^n$.

We then run a max gossip procedure to ensure that all nodes have exactly the same values, i.e., the same pre-weight vector $\widehat{\phi}^{(H_{\text{top-}m})}$. Similar to selective gossip, max gossip is based on the asynchronous time model and the communication model given in Sect. 10.2. When two nodes u and v perform a max gossip iteration, they identify the entries to update in the same way as selective gossip does. However, max gossip differs from selective gossip in that, instead of averaging, the nodes take the maximum of their previous values; i.e., nodes u and v update entries $j \in H_\tau^u(k-1) \cup H_\tau^v(k-1)$ by setting

$$x_j^u(k) = x_j^v(k) = \max\left(x_j^u(k-1), x_j^v(k-1)\right). \tag{10.24}$$

When all nodes have the exact same pre-weight values for particles in the set $H_{\text{top-}m}$, then they can compute the weights for these particles and proceed with the normalization and resampling. Since they have synchronized seeds, they will sample the same particles and reach the same set of weighted particles at the end of each step of the algorithm. The complete algorithm is described in Algorithm 2.

10.5.6 Numerical Example: Bearings Only Distributed Tracking of a Maneuvering Target

To evaluate the performance of our method, we study a distributed tracking scenario where a maneuvering target is monitored by a network of bearings sensors. Such a

Algorithm 2 Distributed Bootstrap Particle Filter with Selective Gossip

```
  // Initialization at time $t = 1$
 1. For each node $v = 1, \dots, n$ do
      • For each particle $i = 1, \dots, M$ do
          – Sample $\mathbf{y}_1^{(i)} \sim q_1(\cdot)$
          – Set $\phi^{v,(i)} = n \log p(\mathbf{z}_1^v | \mathbf{y}_1^{(i)})$
      • end
 2. end
 3. $\{\widetilde{\phi}^{v,(H_{\text{top-}m})}\}_{v=1}^n = \text{SelectiveGossip}(\{\phi^v\}_{v=1}^n, m)$
 4. $\{\widehat{\phi}^{(H_{\text{top-}m})}\}_{v=1}^n = \text{MaxGossip}(\{\widetilde{\phi}^{v,(H_{\text{top-}m})}\}_{v=1}^n)$
 5. For each node $v = 1, \dots, n$ do
      • For each particle $i \in H_{\text{top-}m}$ do
          – Set $w_1^{(i)} = \frac{\exp(\widehat{\phi}^{(i)}) p(\mathbf{y}_1^{(i)})}{q_1(\mathbf{y}_1^{(i)})}$
      • end
      • Normalize weights $w_1^{(i)}$ so that $\sum_{i \in H_{\text{top-}m}} w_1^{(i)} = 1$
      • Resample $\left\{\mathbf{y}_1^{(i)}, w_1^{(i)}\right\}_{i \in H_{\text{top-}m}}$ to obtain $\left\{\mathbf{y}_1'^{(i)}, \frac{1}{M}\right\}_{i=1}^M$
 6. end
  // For times $t > 1$:
 7. For each node $v = 1, \dots, n$ do
      • For each particle $i = 1, \dots, M$ do
          – Set $\mathbf{y}_{1:t-1}^{(i)} = \mathbf{y}_{1:t-1}'^{(i)}$
          – Sample $\mathbf{y}_t^{(i)} \sim q(\mathbf{y}_t | \mathbf{y}_{t-1}^{(i)})$
          – Set $\phi^{v,(i)} = n \log p(\mathbf{z}_t^v | \mathbf{y}_t^{(i)})$
      • end
 8. end
 9. $\{\widetilde{\phi}^{v,(H_{\text{top-}m})}\}_{v=1}^n = \text{SelectiveGossip}(\{\phi^v\}_{v=1}^n, m)$
10. $\{\widehat{\phi}^{v,(H_{\text{top-}m})}\}_{v=1}^n = \text{MaxGossip}(\{\widetilde{\phi}^{v,(H_{\text{top-}m})}\}_{v=1}^n)$
11. For each node $v = 1, \dots, n$ do
      • For each particle $i \in H_{\text{top-}m}$ do
          – Set $w_t^{(i)} = \frac{\exp(\widehat{\phi}^{(i)}) p(\mathbf{y}_t^{(i)} | \mathbf{y}_{t-1}^{(i)})}{q(\mathbf{y}_t | \mathbf{y}_{t-1}^{(i)})}$
      • end
      • Normalize weights $w_t^{(i)}$ so that $\sum_{i=1}^M w_t^{(i)} = 1$
      • Resample $\left\{\mathbf{y}_{1:t}^{(i)}, w_t^{(i)}\right\}_{i=1}^M$ to obtain $\left\{\mathbf{y}_{1:t}'^{(i)}, \frac{1}{M}\right\}_{i=1}^M$
12. end
```

scenario of tracking based on only angle measurements is generally termed bearings-only tracking (sometimes also appearing in the literature under the names passive ranging and target motion analysis [27]).

We consider a two-dimensional setup where the bearing is defined as the angle from the vertical axis of Cartesian plane to the line of sight between the observer and the target. The bearing angle is measured positive in the clockwise direction. The state of the target at time t is

$$\mathbf{y}_t = \left[y_{t,1} \ y_{t,2} \ \dot{y}_{t,1} \ \dot{y}_{t,2} \right]^T, \tag{10.25}$$

where $y_{t,1}$ and $y_{t,2}$ correspond to the position in the X and Y coordinates in the Cartesian plane and $\dot{y}_{t,1}$ and $\dot{y}_{t,2}$ are the velocity values in these coordinates. The state of the observing sensor node $v \in V$ is similarly defined as $\mathbf{y}_t^v = \left[y_1^v \ y_2^v \ 0 \ 0 \right]^T$. Note that the velocity values are equal to zero because the sensor nodes are static. We assume that each node is aware of its state. The measurement made by node v at time t is denoted by z_t^v.

The dynamics of the maneuvering target are modeled using three different motion models [28]. We assume that at any time the target makes one of the following motions: (1) constant velocity (CV), (2) clockwise coordinated turn (CT), or (3) counter-clockwise coordinated turn (CCT). The target moves according to these three motion models with probabilities of p_{CV}, p_{CT} and, p_{CCT}, respectively. We also assume that the probability of both coordinated turns are equal, i.e., $p_{\text{CT}} = p_{\text{CCT}}$, and there are no other motions available, that is $p_{\text{CV}} + p_{\text{CT}} + p_{\text{CCT}} = 1$.

The state at time $t + 1$ can be expressed as a function of the previous state $\mathbf{y}_t$ and process noise $\mathbf{v}_t$

$$\mathbf{y}_{t+1} = \mathbf{F}_t^j \mathbf{y}_t + \mathbf{G}\mathbf{v}_t, \tag{10.26}$$

where $\mathbf{F}_t^j$ is the transition matrix corresponding to the motion model $j \in \{1, 2, 3\}$ and

$$\mathbf{G} = \begin{bmatrix} T^2/2 & 0 \\ 0 & T^2/2 \\ T & 0 \\ 0 & T \end{bmatrix}. \tag{10.27}$$

Here T is the sampling interval and $\mathbf{v}_t \sim \mathcal{N}(0, \sigma_a I_{2\times 2})$ with scalar σ_a. The transition matrix corresponding to the constant velocity model is

$$\mathbf{F}_t^1 = \begin{bmatrix} 1 & 0 & T & 0 \\ 0 & 1 & 0 & T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{10.28}$$

whereas the coordinated turn models are governed by

$$\mathbf{F}_t^j = \begin{bmatrix} 1 & 0 & \frac{\sin(\Omega_t^{(j)}T)}{\Omega_t^{(j)}} & -\frac{1-\cos(\Omega_t^{(j)}T)}{\Omega_t^{(j)}} \\ 0 & 1 & \frac{1-\cos(\Omega_t^{(j)}T)}{\Omega_t^{(j)}} & \frac{\sin(\Omega_t^{(j)}T)}{\Omega_t^{(j)}} \\ 0 & 0 & \cos(\Omega_t^{(j)}T) & -\sin(\Omega_t^{(j)}T) \\ 0 & 0 & \sin(\Omega_t^{(j)}T) & \cos(\Omega_t^{(j)}T) \end{bmatrix}, \ j = 2, 3. \tag{10.29}$$

The turning rates for clockwise and counter clockwise coordinated turn models are

$$\Omega_t^2 = \frac{a}{\sqrt{(\dot{y}_{t,1})^2 + (\dot{y}_{t,2})^2}}, \qquad \Omega_t^3 = -\frac{a}{\sqrt{(\dot{y}_{t,1})^2 + (\dot{y}_{t,2})^2}}, \tag{10.30}$$

where $a > 0$ is the maneuver acceleration parameter. Note that the turning rates are nonlinear functions of the state.

The angle measurements are also a nonlinear function of the state. The measurement taken by node v at time t is modeled as

$$z_t^v = \arctan\left(\frac{y_{t,1} - y_{t,1}^v}{y_{t,2} - y_{t,2}^v}\right) + w_t, \tag{10.31}$$

where $w_t \sim \mathcal{N}(0, \sigma_\theta^2)$ is the measurement noise.

We consider a network of $n = 49$ sensor nodes, forming a grid topology. The network spans an area of 1 km^2. The initial state of the target is

$$\mathbf{y}_1 = \left[702\,\mathrm{m}\ 621\,\mathrm{m}\ 10\,\mathrm{m/min}\ 80\,\mathrm{m/min}\right]^T. \tag{10.32}$$

The target follows a trajectory for a duration of $t_{max} = 20$ min. The sensor locations and the trajectory of the target are shown in Fig. 10.8.

The particle filter at each node is initialized with the same distribution centered at the initial state of the target [28]. In particular, we assume prior knowledge of the target's initial range, speed, and course (i.e., the angle with the vertical axis of the Cartesian plane). The position components of the state are initialized using the bearing measurement recorded by the closest sensor at time $t = 1$ and the initial range $\hat{r}_1$. We assume that $\hat{r}_1 \sim \mathcal{N}(r_1, \sigma_r^2)$ where r_1 is the initial true target range and $\sigma_r^2 = r_1/8$. Similarly, the velocity components of the state are initialized using the initial speed, $\hat{s}_1$, and initial course, $\hat{c}_1$. We assume that $\hat{s}_1 \sim \mathcal{N}(s_1, \sigma_s^2)$ where s_1 is the target's true initial speed and $\sigma_s^2 = s_1/8$. Likewise, $\hat{c}_1 \sim \mathcal{N}(c_1, \sigma_c^2)$ where c_1 is the true initial course and $\sigma_c^2 = \pi/\sqrt{6}$. Note that this initialization is suitable for many problems, but there may be cases where target acquisition also needs to be performed. This is beyond the scope of the current chapter.

We model the target motion using the following parameters: the process noise is $\sigma_a = 0.1$, the acceleration parameter is $a = 30$, the probability of constant velocity model is $p_{\mathrm{CV}} = 0.6$ and the probabilities of turning clockwise or counter-clockwise are $p_{\mathrm{CT}} = p_{\mathrm{CCT}} = 0.2$.

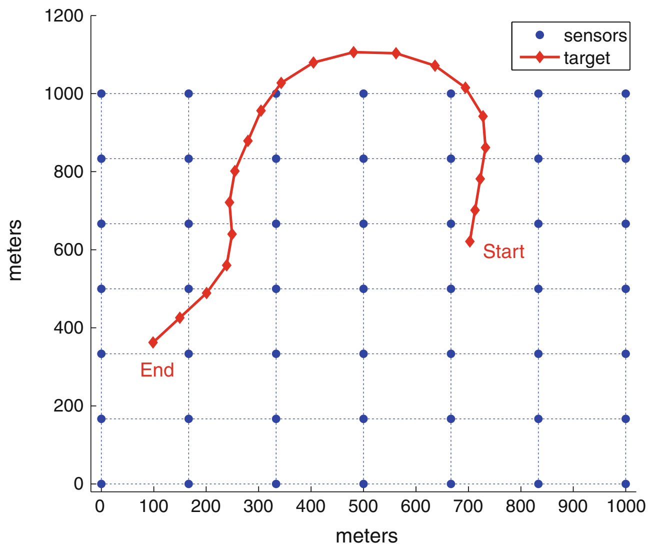

Fig. 10.8 Sensor network and the trajectory of the target. *Dashed lines* represent wireless communication links between sensors. The target makes a movement for 20 min and the markers show its location at the beginning of each minute. The start and end points of the trajectory are also marked

The nodes take measurements corrupted with additive Gaussian noise of standard deviation $\sigma_\theta = 3^\circ$. We assume that nodes have a limited sensing range, i.e., they can only provide bearing measurements for targets within their sensing range. The sensing range of each node is set to 200 m which is slightly longer than the distance between two horizontally or vertically adjacent nodes. Measurements are made only by the nodes that have the current estimate of target location within their sensing range. For the trajectory given in Fig. 10.8, up to 4 sensors take measurements at each time step. The sampling interval is $T = 1$min.

The experiment for each algorithm is repeated for 1,000 Monte Carlo trials. Let l denote the trial index. The position error for each trial l is calculated according to

$$E_t(l) = \sqrt{(\hat{y}_{t,1} - y_{t,1})^2 + (\hat{y}_{t,2} - y_{t,2})^2}, \tag{10.33}$$

where $\hat{y}_{t,1}$ and $\hat{y}_{t,2}$ are the estimated position of the target. Note that the error $E_t(l)$ is same at each node as the distributed particle filters are synchronized. The trials that exceed the error value of 250 m at any time t are considered as lost tracks. Then for the tracks that are not lost, we calculate the root-mean-squared (RMS) position error

$$RMSE(l) = \sqrt{\frac{1}{t_{max}} \sum_{t=1}^{t_{max}} E_t(l)^2}. \tag{10.34}$$

Table 10.1 A comparison of the performances of the centralized bootstrap particle filter and the distributed particle filters for $M = 2000$ and $m = 500$

Algorithm	Average RMSE	Track loss	Scalars
Centralized bootstrap	10.28 ± 6.4	0.1	–
Adaptive threshold selective gossip	11.05 ± 9.0	6.3	3.90e+06
Clairvoyant threshold selective gossip	11.09 ± 7.9	0.5	4.41e+06
Top-m selective gossip	11.01 ± 7.2	0.8	2.85e+06
Clairvoyant top-m selective gossip	10.92 ± 8.2	1.0	2.40e+06
Randomized gossip	11.17 ± 8.8	0.3	9.60e+06

For each filter the average RMS position error ± standard deviation, percentage of track loss, and the number of transmitted scalars are presented

Similarly, the communication overhead, which is represented with the number of scalars transmitted, does not include the trials that resulted in lost tracks.

We compare the performance of the distributed particle filter using the two versions of selective gossip: adaptive threshold and top-m selective gossip. To illustrate the decrease in communication cost compared to randomized gossip, we run the same algorithm with $m = N$ which corresponds to updating each entry at each gossip iteration, that is randomized gossip run in parallel for each particle weight. We also run a centralized bootstrap particle filter as a performance benchmark.

In addition, we simulate two clairvoyant versions of selective gossip. The first version, clairvoyant threshold selective gossip, represents the case where all nodes clairvoyantly know the threshold value corresponding to the largest mth entry of the average consensus vector. This is obtained by setting $\tau = \bar{\mathbf{x}}_{(m)}$ during the initialization of the algorithm. The second version, called clairvoyant top-m selective gossip, represents the case where each node clairvoyantly knows the indices of the set $H_{\text{top-}m}$ and only updates these entries. This is obtained by setting $H^{v}_{\text{top-}m}(k) = H_{\text{top-}m}$ at all nodes $v \in V$ and all iterations k. The distributed filter computations are performed with n^2 selective gossip iterations and $10n$ max gossip iterations.

For $M = 2,000$ particles and $m = 500$, Table 10.1 shows the average RMS position error, the track loss percentage and the number of scalars transmitted for each particle filter. The top-m version of selective gossip performs very close to the clairvoyant algorithms and better than the adaptive threshold selective gossip. Adaptive threshold selective gossip loses a high percentage of tracks while transmitting more scalars. Top-m selective gossip also provides performance similar to randomized gossip in terms of both error and track loss while decreasing the communication overhead more than three times. Figure 10.9 demonstrates sample tracks of the centralized and distributed particle filters, in particular the tracks with the median RMS performance for each filter.

Next, we investigate the effect of m on the performance of distributed particle filter with adaptive threshold and top-m selective gossip. Note that increasing m results in increased communication overhead. Figure 10.10 shows the percentage of lost tracks as a function of m. Figure 10.11 shows the RMS position error averaged over tracks that are not loss. We see that the distributed particle filter with top-m selective

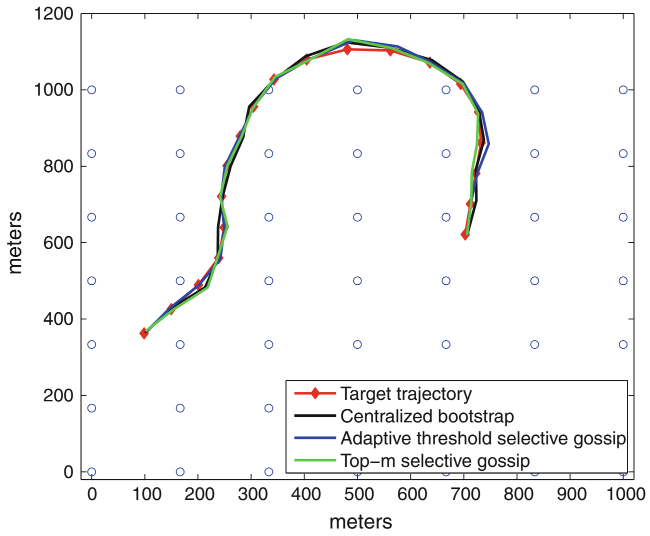

Fig. 10.9 Target trajectory and sample tracks corresponding to the median RMS position error for each filter. The unit of distance values on the axes is meter

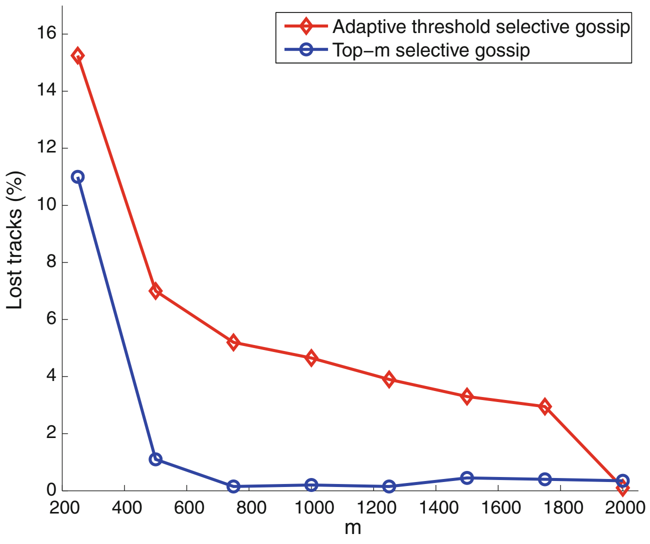

Fig. 10.10 Percentage of track loss as a function of m

gossip achieves good performance for m values 500 and more. Taken together, these results illustrate that the distributed particle filter with top-m selective gossip provides significantly better performance in terms of track loss and RMS error performance for non-divergent tracks compared to the adaptive threshold selective gossip.

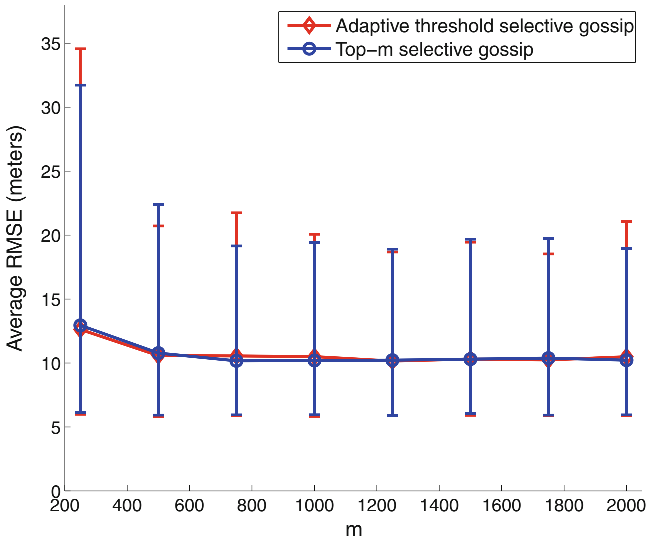

Fig. 10.11 Average RMS position error as a function of m. 95 % confidence bars are also depicted (the end points of these bars correspond to the 5 and 95 % percentiles)

10.6 Conclusions

Many complex signal processing tasks of wireless sensor networks can be formulated using distributed averaging of vector-valued network data where the vectors are possibly high-dimensional. Standard gossip algorithms, which are typically described for averaging scalar quantities, can easily be extended to the vector case by communicating all entries of the vectors. However, this is inefficient in applications where only a small percentage of the entries of the average vector is significant. This chapter presented selective gossip, an algorithm that reduces the dimension of the exchanged data by adaptively focusing communication resources on the entries which are significant for the nodes that are performing the exchange. We proved that focusing on locally significant data, nodes can asymptotically identify the locations of the significant entries of the average vector and also reach a consensus on the values of these entries. To investigate the communication overhead compared to randomized gossip, we presented a simulation study. The results demonstrate that selective gossip provides significant communication savings in terms of number of scalars transmitted. In the second part of the chapter, we proposed a distributed particle filter using selective gossip. In a target tracking scenario with bearings sensors, we showed that distributed particle filters implemented using our algorithm provide comparable results to the centralized bootstrap particle filter while decreasing the communication overhead compared to using randomized gossip to distribute the filter computations.

Our results demonstrate that selective gossip provides a decentralized and efficient building block for wireless sensor network applications. In particular, the top-m

version of selective gossip is potentially more interesting as it has convergence guarantees. This version also provides better tracking performance in the simulation setup we considered. Note that we presented selective gossip based on randomized gossip but it can be implemented with other gossip algorithms such as the synchronous gossip algorithm and faster pairwise gossip algorithms available in the literature.

The future work involves the investigation of the rates of convergence for selective gossip. Since the entries updated at each iteration depends on the vectors in the network at that iteration, the standard methods used for quantifying the convergence rate of randomized gossip do not apply.

References

1. Bénézit F, Dimakis A, Thiran P, Vetterli M (2007) Gossip along the way: Order-optimal consensus through randomized path averaging. In: Proceedings of the Allerton Conference on Communication, Control, and Computing, Monticello
2. Bertsekas DP, Tsitsiklis JN (1997) Parallel and distributed computation: Numerical methods. Athena Scientific, Belmont
3. Boyd S, Ghosh A, Prabhakar B, Shah D (2006) Randomized gossip algorithms. IEEE Trans Info Theory 52(6):2508–2530
4. Cappé O, Moulines E, Ryden T (2005) Inference in hidden Markov models. Springer-Verlag, New York
5. Coates M (2004) Distributed particle filters for sensor networks. In: Proceedings of the International Symposium on Information Processing in Sensor Networks (IPSN), Berkeley
6. Dimakis A, Sarwate A, Wainwright M (2006) Geographic gossip: Efficient aggregation for sensor networks. In: Proceedings of the International Conference on Information Processing in Sensor Networks (IPSN), Nashville
7. Dimakis AG, Kar S, Moura JMF, Rabbat MG, Scaglione A (2010) Gossip algorithms for distributed signal processing. Proc IEEE 98(11):1847–1864
8. Doucet A, de Freitas N, Gordon N (eds) (2001) Sequential Monte Carlo methods in practice. Springer-Verlag, New York
9. Doucet A, Johansen M (2010) Oxford handbook of nonlinear filtering, chapter A tutorial on particle filtering and smoothing: fifteen years later. Oxford University Press, to appear
10. Farahmand S, Roumeliotis SI, Giannakis GB (2011) Set-membership constrained particle filter: Distributed adaptation for sensor networks. IEEE Trans Signal Process 59(9):4122–4138
11. Gordon NJ, Salmond DJ, Smith AFM (1993) Novel approach to nonlinear/non-Gaussian Bayesian state estimation. IEE Proc-F 140(2):107–113
12. Grimmett GR, Stirzaker DR (2001) Probability and random processes. Oxford University Press, New York
13. Gu D (2007) Distributed particle filter for target tracking. In: Proceedings IEEE International Conference on Robotics and Automation, Rome
14. Gupta P, Kumar PR (2000) The capacity of wireless networks. IEEE Trans Info Theory 46(2):388–404
15. Handschin JE, Mayne DQ (1969) Monte Carlo techniques to estimate the conditional expectation in multi-stage non-linear filtering. Int J Control 9(5):547–559
16. Hendrickx JM, Tsitsiklis JN (2011) Convergence of type-symmetric and cut-balanced consensus seeking systems. Submitted; available at http://arxiv.org/abs/1102.2361
17. Hlinka O, Sluciak O, Hlawatsch F, Djurić PM, Rupp M (2010) Likelihood consensus: Principles and application to distributed particle filtering. In: The forty fourth Asilomar Conference on Signals, Systems and Computers (ASILOMAR)

18. Jadbabaie A, Lin J, Morse AS (2003) Coordination of groups of mobile autonomous agents using nearest neighbor rules. IEEE Trans Autom Control 48(6):988–1001
19. Kokiopoulou E, Frossard P (2009) Polynomial filtering for fast convergence in distributed consensus. IEEE Trans Signal Process 57(1):342–354
20. Lee SH, West M (2009) Markov chain distributed particle filters (MCDPF). In: Proceedings of the IEEE Conference on Decision and Control, Shanghai
21. Mohammadi A, Asif A (2011) Consensus-based distributed unscented particle filter.In: Proceedings of the IEEE Statistical Signal Processing Workshop (SSP), 237–240
22. Nedić A, Ozdaglar A (2009) Distributed subgradient methods for multi-agent optimization. IEEE Trans Autom Control 54(1):48–61
23. Olfati-Saber R, Fax JA, Murray RM (2007) Consensus and cooperation in networked multiagent systems. Proc IEEE 95(1):215–233
24. Oreshkin BN, Coates MJ, Rabbat MG (2010) Optimization and analysis of distributed averaging with short node memory. IEEE Trans Signal Process 58(5):2850–2865
25. Oreshkin BN, Coates MJ (2010) Asynchronous distributed particle filter via decentralized evaluation of Gaussian products. In: Proceedings of the ISIF International Conference on Information Fusion, Edinburgh
26. Rabbat M, Nowak R, Bucklew J (2005) Robust decentralized source localization via averaging In: Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP). Philadelphia
27. Ristic B, Arulampalam MS (2003) Tracking a manoeuvring target using angle-only measurements: algorithms and performance. Signal Process 83(6):1223–1238
28. Ristic B, Arulampalam S, Gordon N (2004) Beyond the Kalman filter: particle filters for tracking applications. Artech House, Norwood, MA, USA
29. Sheng X, Hu Y-H, Ramanathan P (2005) Distributed particle filter with GMM approximation for multiple targets localization and tracking in wireless sensor network. In: Proceedings of the International Symposium on Information Processing in Sensor Networks (IPSN), Los Angeles
30. Sundhar Ram S, Veeravalli VV, Nedić A (2010) Distributed and recursive parameter estimation in parametrized linear state-space models. IEEE Trans Autom Control 55(2):488–492
31. Touri B (2011) Product of random stochastic matrices and distributed averaging. PhD thesis, Univeristy of Illinois at Urbana-Champaign
32. Tsitsiklis JN (1984) Problems in decentralized decision making and computation. PhD Thesis, MIT
33. Tsitsiklis JN, Bertsekas DP, Athans M (1986) Distributed asynchronous deterministic and stochastic gradient optimization algorithms. IEEE Trans Autom Control 31(9):803–812
34. Üstebay D, Castro R, Rabbat M (2011) Efficient decentralized approximation via selective gossip. IEEE J Sel Top Sign Proc 5(4):805–816
35. Üstebay D, Oreshkin B, Coates M, Rabbat M (2008) Rates of convergence for greedy gossip with eavesdropping. In: Proceedings of the Allerton Conference on Communication, Control, and Computing. Monticello, pp 367–374
36. Üstebay D, Rabbat M Efficiently reaching consensus on the largest entries of a vector. In: IEEE Conference on Decision and Control (CDC) '12, Maui, HI, USA
37. Xiao L, Boyd S (2004) Fast linear iterations for distributed averaging. Syst Control Lett 53(1):65–78
38. Zhao F, Shin J, Reich J (2002) Information-driven dynamic sensor collaboration. IEEE Signal Process Mag 19(2):61–72