Distributed Optimal Self-Organisation in a Class of Wireless Sensor Networks

Aditya Karnik and Anurag Kumar
Dept. of Electrical Communication Engg.
Indian Institute of Science, Bangalore, 560012, INDIA
{karnik, anurag}@ece.iisc.ernet.in

***Abstract*— The work in this paper is motivated by the idea of using randomly deployed, ad hoc wireless networks of miniature smart sensors to serve as distributed instrumentation. We argue that in such applications it is important for the sensors to self-organise in a way that optimizes network throughput. We then identify and discuss two main problems of optimal self-organisation: (i) building an optimal topology, and (ii) tuning network access parameters such as the transmission attempt rate. We consider a simple random access model for sensor networks and formulate these problems as optimisation problems. We then present centralized as well as distributed algorithms for solving them. Results show that the performance improvement is substantial and implementation of such optimal self-organisation techniques may be worth the additional complexity.**

I. Introduction

Advances in microelectronics technology have made it possible to build inexpensive, low power, miniature sensing devices. Equipped with a microprocessor, memory, radio and battery, such devices can now combine the functions of sensing, computing, and wireless communication into miniature *smart sensors*.

Since smart sensors need not be tethered to any infrastructure because of on-board radio and battery, their main utility lies in being *ad hoc*, in the sense that they can be rapidly deployed by randomly strewing them over a region of interest. This means that the devices and the wireless links will not be laid out to achieve a planned topology. During the operation, sensors would be difficult or even impossible to access and hence their network needs to operate autonomously. Moreover, with time it is possible that sensors fail (one reason being battery drain) and cannot be replaced. It is, therefore, essential that sensors *learn about each other* and *organise into a network* on their own. In the absence of a centralized control, this whole process needs to be carried out in a distributed fashion.

A smart sensor may have only modest computing power, however, the ability to communicate allows a group of sensors to collaborate to execute tasks more complex than just sensing and forwarding the information, as in traditional sensor arrays. Hence they may be involved in on-line processing of sensed data in a distributed fashion so as to yield partial or even complete results to an observer, thereby facilitating control applications, interactive computing and querying ([1], [2], [3], [4]). It is this *self-organising distributed instrumentation* aspect of sensor networks that we are interested in.

A distributed computing approach will also be energy efficient as compared to mere data dissemination since it will avoid energy consumption in long haul transport of the measured data to the observer; this is of particular importance since sensors could be used in large numbers due to their low cost yielding very high resolutions and large volumes of sensed data. Further, by "arranging computations" among only the neighbouring sensors the number of transmissions will be reduced, thereby, saving transmission energy. A simple class of distributed computing algorithms would require each sensor to periodically exchange the results of local computation with the neighbouring sensors. The more frequently such exchanges can occur, the more rapidly will the overall computation converge. The more rapid the progress of the computation the faster the variations of the spatial process that can be tracked. *Thus, in this paper our goal is to study optimal self-organisation of sensor networks from the point of view of optimizing their communication throughput.*

As an example, consider a scenario where sensors are randomly deployed in a geographical region to gather statistics of a spatial process; for example, the temperature of an environment, or the level of some chemical contamination. Suppose that the observer is interested in knowing the maximum value of the quantity being measured. Instead of each sensor sending its measurement to the observer, sensors can compute the maximum in a distributed fashion, and communicate only the result to the observer. A simple distributed algorithm for each sensor can be to collect measurements from its neighbours, compare them with its local value and only forward the maximum of these values. The communication structure most suitable for such a computation is a spanning tree which sensors can form in a distributed fashion. Moreover, a sensor needs to transmit the local maximum to its parent only after it receives the corresponding values from its children. The algorithm ensures that the observer ultimately receives the maximum value in the network.

From this example, it is clear that the higher the communication throughput of sensors, the more rapidly will the computation of the maximum proceed, thereby allowing the network to track the variations of the temperature or contamination with time. The example also shows that the network topology and the transmission protocol are critical factors that determine the communication throughput. It is in these two aspects that we study optimal self-organisation of sensor networks. To

this end, we propose a simple mathematical model for sensor networks. Instead of limiting ourselves to a particular task, we consider a general distributed computation scenario in which sensors are continuously sampling and processing a spatio-temporal process. We investigate the problem of building an *optimal topology* and tuning to an *optimal value of channel access rate* in our communication model. The model allows a concrete mathematical formulation of the problem, and is also sufficiently general so that the analysis can be applied or extended to other cases of interest. We then formulate the optimisation problems and present distributed algorithms for solving them. We also discuss the convergence and complexity issues in the algorithms.

Presently, the algorithms are synchronous. Thus, in relation to our model, their performance gain is the best possible. Our results show substantial performance improvements but at the cost of algorithmic complexity. However, compared with the performance gains, implementation of such self-organisation techniques may be worth the additional complexity. In this respect, our work should be seen as a step towards eventually understanding algorithms for self-optimizing sensor networks.

The paper is organised as follows. In Section II we review the previous work in this area. Section III discusses the model and relevance of assumption to the real sensor networks. Section IV motivates the optimisation problems. The problem of optimal network topology is discussed in Section V and tuning to an optimal channel access rate in Section VI. Section VII follows with discussion and we conclude in Section VIII. Proofs are sketched in the Appendix.

II. Previous Work

Previous work on self-organisation of ad hoc networks has largely focused on topology formation by discovering neighbours, and transmission scheduling. For example, [5] discusses a cluster formation algorithm (LCA) and a link activation algorithm based on the topology graph so formed. In [6], the DEA algorithm forms subnetworks and conflict-free schedules simultaneously in a step by step fashion. However, the generation of compatible schedules is a complex task; particularly for large networks. Moreover, the graph-based scheduling suffers in network performance as shown in [7]. [8] describes the SWAN protocol in which nodes maintain wireless links using stringent power control thereby obviating the need for transmission scheduling. However, it does not address the problem of how nodes should choose their neighbours thereby setting the topology. [9] presents a message efficient clustering algorithm for sensor networks. [10] reports experimental performance studies of a self-configuring protocol for sensor networks. Specific protocols for self-organisation are discussed in [11].

The existing literature has emphasized mainly on protocol design without much attention to the performance of the resulting network organisation. Contrary to this approach we view *performance optimisation as the objective for self-organisation*; hence our algorithms are motivated by this goal. Our formulations, we believe, are the first of their kind in this area. The model along with the analytical approach allows us to study trade-offs as well as performance gains involved in optimal self-organisation. We also substantiate our results by simulations.

III. A Model for Sensor Networks

Our model takes into account deployment, communication and distributed computation issues in a sensor network. Since our interest is in analysing local computing and communication, we have not explicitly modelled communication between the network and the observer. We consider a random access communication model; we believe that sensors will not need elaborate multiple access or transport protocols since typical packets will be small and RTS-CTS may be too much of an overhead. Moreover, it is not clear how much improvement such a scheme will lead to in dense random networks. We assume slot synchronization among sensors. LAA ([5]), SWAN ([8]) and SEEDEX ([12]) have also assumed a synchronized TDMA system. Time synchronization is vital for some sensing tasks ([13], [14]); hence our slotted time assumption may not be very restrictive. Even in the absence of time synchronization, slot synchronization can be achieved by distributed algorithms ([15], [16]). Further, it makes analysis tractable and provides useful insights.

Deployment: We assume that a large number (denoted by N) of static sensor nodes are randomly and densely located in a region. By *dense* we mean that the nearest neighbour distance is much less than the transmission range. To reconstruct the properties of a spatial process from its samples in space with low error, we need sufficiently close "space-samples". Hence the interest in dense sensor networks. In the results presented we model the spatial distribution of sensors as a two-dimensional Poisson point process of intensity λ per m^2. λ is thus a natural measure of the *spatial density* of the network. However, the model and the following analyses are applicable to any placement of sensors.

Communication: A sensor cannot transmit and receive simultaneously. All sensors transmit on a common carrier frequency using omni-directional antennas. All transmitters use the same transmit power, and all powers are normalised to this common value of transmit power. In fixed-position, short-path, flat terrain outdoor wireless, the channel variation is very slow, and there is a strong LOS path ([17]). We, therefore, consider only the path loss with exponent η. Letting d_0 denote the near field crossover distance, the power received at a distance r from a transmitter, $P_s(r) = (r/d_0)^{-\eta}$ if $r > d_0$ and $P_s(r) = 1$ if $r \leq d_0$; this model is similar to the one in [18]. We say that a transmission can be "decoded" when its signal to interference ratio (SIR) exceeds a given threshold[1] β. The transmission range (denoted by R_0) is defined as the maximum distance at which a receiver can decode a transmitter in the absence of any co-channel interference. It thus follows that a receiver being within R_0 from a transmitter does not guarantee that it

[1]Given a modulation and coding scheme, β actually governs the maximum bit error probability. For narrow-band systems $\beta > 1$ and for spread spectrum systems $\beta < 1$.

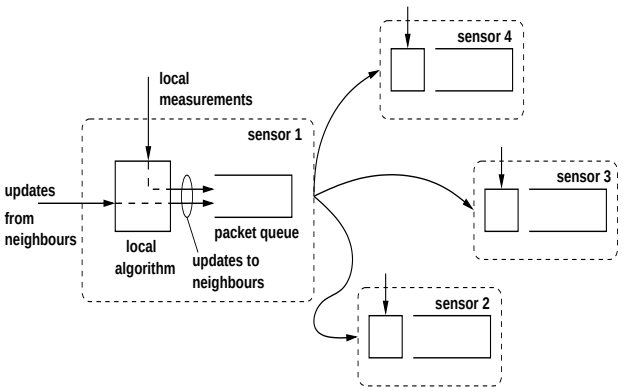


Fig. 1. A traffi c model for a sensor network carrying out measurements and distributed computation.

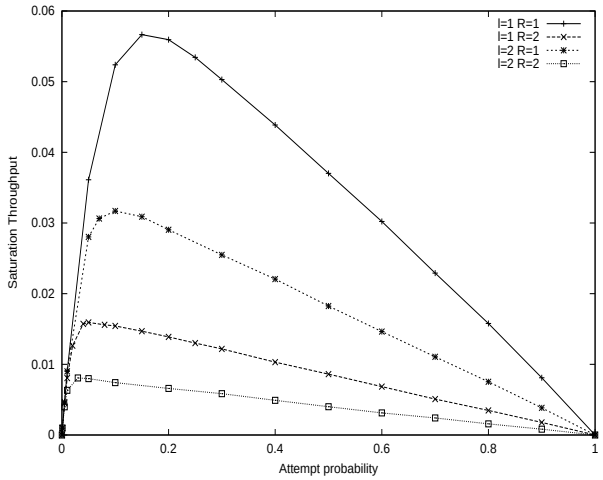


Fig. 2. Saturation throughput ($p_t^{(s)}$) variation with attempt probability (α) for $\lambda = 1$ and 2 per m^2. R equals 1 or $2m$.

will decode a packet; the SIR needs to be above β as well. Time is slotted and channel access is random, i.e., in each slot, sensor i decides to transmit with probability α_i and decides to receive with probability $(1 - \alpha_i)$ independent of anything else; α_i is called the *attempt probability* of sensor i.

Computation: We consider a continuous computing model for sensor networks ([19]). This also models the observer initiated processing ([19]) with long enough activity periods. Figure 1 shows a traffic model for a sensor network engaged in distributed computation. A sensor communicates only with certain nodes, within R_0 from it, designated as its *neighbours*. A local algorithm running on each sensor uses local measurements and updates from its neighbours to perform certain computations. The raw measurements and/or computational results to be sent to the neighbours are queued up in a packet queue as shown in Figure 1. The application is such that each packet is destined to a random neighbour uniformly chosen from the neighbours. This is not a restrictive assumption and can be attributed to isotropy of the physical process and uniformity of processing. Hence, if a sensor decides to transmit, the 1-hop destination of the head-of-the-line packet is equally likely to be any of the neighbours in the operational topology. A transmission is successful when the sensor to which it is addressed (by actually inserting a physical address in the packet header) is in the receive mode, and is able to decode the transmission. If a transmission is successful, the corresponding packet is removed from the queue (i.e., instantaneous acknowledgments are assumed). Since acknowledgments can be also lost in transmissions, this assumption gives an upper bound on the performance of sensor networks. A successfully received packet at a sensor invokes a new computation that *may* result in an update being sent to a neighbour. We model this probabilistically, i.e., the successful reception of a packet generates another packet to be sent to a neighbour with probability ν.

IV. Optimal Self-Organisation: Motivation

When a sensor network is processing a spatio-temporal process, to reconstruct the process in time, each sensor has to sample it at a specific rate (akin to a Nyquist rate). These samples, along with the packets triggered by updates from the neighbours, form an arrival process into the queue (see Figure 1). Therefore, this 'sampling rate' cannot exceed the rate at which packets are drained from the queue. More precisely, assume that $\alpha_i = \alpha, 1 \leq i \leq N$ and that each sensor has all the nodes within a fixed distance, say $R \leq R_0$ as its neighbours. Note that the probability of successful transmission of a sensor decreases with distance because of decrease in the received power. Therefore, a smaller value of R is desirable. However, a small R also means that a sensor communicates with only a few of the many sensors within its transmission range. In [20], we show that if γ denotes the arrival rate of measurements, then at a 'typical' sensor in the Poisson distributed sensor field, the packet queue is stable if $\gamma < p_t^{(s)} - (1 - \alpha)\nu$ where $p_t^{(s)}$ denotes the probability of successful transmission in a slot in saturation, i.e., when the head-of-the-line packet, after its successful transmission, is immediately replaced by another packet at each queue. $p_t^{(s)}$ is also called the saturation throughput. While processing a time varying process, most of the time each sensor will have some local measurements and/or partial results to communicate to its neighbours. Therefore, in such a scenario, the saturation throughput can be a good measure of the communication throughput.

This work is particularly motivated by Figure 2 which shows the variation of $p_t^{(s)}$ with α (we have assumed that $\alpha_i = \alpha, 1 \leq i \leq N$). We use 1000 Poisson distributed points as sensors on the plane for $\lambda = 1$ and 2 per m^2. We take $\eta = 4$, $\beta = 10$ dB and R equals 1 or $2m$. Throughputs are averaged over 1000 random point placements. Observe that, for a fixed value of λ, $p_t^{(s)}$ decreases as R increases and for a fixed value of R, $p_t^{(s)}$ decreases as λ increases. Thus, high values of $p_t^{(s)}$ decree small values of R, however, arbitrarily small values of R result in a disconnected network. Note also from Figure 2 that, for a fixed λ and R, there is a value of α which maximises $p_t^{(s)}$. From this preliminary analysis, we conclude that sensors

need to form a network that is 'optimally connected', and operated at an 'optimal attempt rate'. A precise formulation of these problems and their solutions is the objective of the following sections.

V. Optimal Network Topology: Objectives and Algorithms

Let G denote a connected weighted graph with vertex set V ($|V| = N$), edge set E and weight function $W : E \to R_+$. The weight of an edge $(i,j) \in E$ is denoted by $w(i,j)$. G can be a directed or an undirected graph. If G is directed, (i,j) denotes an edge outgoing from i to j and connectivity refers to strong connectivity. V_s denotes the set of sensors; each element in V_s is a triplet of the form (i, x_i, y_i) where $i \in \{1, 2, \ldots, N\}$ is the sensor index, and x_i and y_i are the x-coordinate and y-coordinate of i respectively.

Definition 5.1: The *transmission graph*, G_{R_0}, is the symmetric directed graph with V_s as its vertex set, and $(i,j) \in E_{R_0}$ if sensor j is within a distance R_0 of sensor i. □

Note that, G_{R_0} is a geometric random graph since sensors are randomly placed. If the number of sensors is assumed to be infinite, then [21] shows that there exists a critical number ζ for the two-dimensional Poisson sensor field such that if $\lambda\pi R_0^2 > \zeta$, G_{R_0} would contain an infinite component with nonzero probability. An infinite component does not imply that all the sensors are connected. However, N and the area of deployment both are large but finite; hence it is reasonable to expect that most of the sensors belong to the giant component. This has been found to be true in practice (see [22] for details). With these considerations, henceforth we take G_{R_0} to be a connected graph.

Denote by $\underline{\alpha}$ the vector of α_i's, $\underline{\alpha} := (\alpha_1, \alpha_2, \ldots, \alpha_N)$. Let $\underline{\alpha}$ be fixed and let for $(i,j) \in E_{R_0}$, $p_{ij}(\underline{\alpha})$ denote the probability of successful transmission from sensor i to j under $\underline{\alpha}$. Recall that, we are assuming that all sensors have packets to send. Therefore, according to our model

$$p_{ij}(\underline{\alpha}) = \alpha_i(1-\alpha_j)P\left(\frac{(\frac{d_{ij}}{d_0})^{-\eta}}{\sum_{k\neq i,j}(\frac{d_{kj}}{d_0})^{-\eta}Y_k + N_0} \geq \beta\right) \quad (1)$$

The last term in (1) is $P(\Gamma_{ij} \geq \beta)$ where Γ_{ij} denotes the SIR of a transmission from i to j. d_{kj} is the distance between k and j, d_0 is the near-field crossover distance, N_0 is thermal noise power, and Y_k is 1 if k transmits, 0 otherwise. Since $p_{ij}(\underline{\alpha})$ depends on the geometry of interferers of sensor j, $p_{ji}(\underline{\alpha})$ need not equal $p_{ij}(\underline{\alpha})$ in a random network.

A. The MAWSS Topology

Let $G' := (V', E')$ denote a subgraph of a given connected graph G. For $i \in V'$, let $d_i(G')$ denote the out-degree of node i in G'. For all $i \in V'$, let

$$\psi_i(G') := \frac{1}{d_i(G')} \sum_{(i,j)\in E'} w(i,j) \quad (2)$$

if $d_i(G') > 0$ otherwise $\psi_i(G') = 0$. Define a function ψ on G' as

$$\psi(G') := \sum_{i\in V'} \psi_i(G')$$

and let

$$\tilde{G} = \arg \max_{G'\in G_{cs}} \psi(G')$$

where, G_{cs} is the set of all connected spanning subgraphs of G. G_{cs} is nonempty since $G \in G_{cs}$. $\tilde{G}$ maximises the measure ψ over all connected spanning subgraphs of G. We call $\tilde{G}$ the *maximum average-weighted spanning subgraph* (MAWSS) of G. We will use the term MAWSS to also denote an algorithm for determining an MAWSS.

Each subgraph of G_{R_0} specifies a network topology, i.e., a set of *neighbours* for each sensor. Let $N_i(G')$ (respectively $n_i(G')$) denote the set of neighbours (respectively the number of neighbours) of i in topology G'. Now for each i define,

$$M_i(G', \underline{\alpha}) = \frac{1}{n_i(G')} \sum_{j\in N_i(G')} p_{ij}(\underline{\alpha}) \quad (3)$$

Thus, $M_i(G', \underline{\alpha})$ *equals the time average throughput of sensor* i. We have used our assumption that in transmit mode a sensor transmits a packet to one of its neighbours with probability $\frac{1}{n_i(G')}$. Let $M(G', \underline{\alpha})$ denote the network throughput, i.e., the sum of individual sensor throughputs with topology specified by G'. Now if all sensors always have packets to send then $\frac{M(G',\underline{\alpha})}{N}$ is the average saturation throughput of the network. The discussion in Section IV, therefore, motivates the problem of choosing a network topology G' so that $M(G', \underline{\alpha})$ is maximised.

Note that, the "out-degree" of a sensor in G' is simply the number of its neighbours, $n_i(G')$. It, thus, follows by comparing (2) and (3) that for a fixed $\underline{\alpha}$ if G' is a subgraph of G_{R_0}, and if for all $(i,j) \in E_{R_0}$, $w(i,j)$ equals $p_{ij}(\underline{\alpha})$, then $\psi(G')$ is $M(G', \underline{\alpha})$. Since a sensor network needs to be connected, it follows that, *the optimal topology of a sensor network is the MAWSS of its* G_{R_0}.

Proposition 5.1: MAWSS for directed and undirected graphs is NP-complete.

Proofs are presented in the Appendix.

1) A Centralized MAWSS Algorithm: In the following, we discuss directed graphs in particular, and propose a heuristic algorithm for obtaining an approximation to the MAWSS. Some notation is in order. For node i, $e_i(k)$ denotes the k^{th} heaviest outgoing edge and $w_i(k)$ denotes its weight. Ties are resolved arbitrarily. $E_1(G) := \{e_i(1) | i \in G\}$, is the set of maximum weight outgoing edges of all the nodes in G. The basic idea is the following. It is clear that the MAWSS contains $E_1(G)$. Hence if $(V, E_1(G))$ is strongly connected, we are done. If not, we convert the "maximum average weight" problem to the "minimum sum of weights" problem by a suitable transformation of $w(i,j)$ to $\bar{w}(i,j)$. We consider the transformation $\bar{w}(i,j) = w_i(1) - w(i,j)$ and and denote this weight function by $\bar{W}$. We, then, construct minimum weight

1: **if** $(V, E_1(G))$ is strongly connected **then**
2: $\quad \hat{G} = (V, E_1(G))$
3: **else**
4: $\quad$ For all $(i,j) \in E$, $\bar{w}(i,j) := w_i(1) - w(i,j)$ and set $\bar{G} = (V, E, \bar{W})$
5: $\quad$ For all $i \in V$, find $G^i_{out} = (V, E^i_{out})$, the minimum weight out-branching of $\bar{G}$ rooted at i
6: $\quad \hat{G} = (V, \cup_{i \in V} E^i_{out})$

Algorithm 1: Algorithm for finding an approximation $\hat{G}$ to the MAWSS of a directed Graph G

out-branching (directed tree or arborescence) using $\bar{w}(i,j)$ rooted at each i. Recall that, any out-branching rooted at a given node contains one and only one edge incoming to every other node. The minimum weight branchings pick out edges with small $\bar{w}(i,j)$ which are the edges with large $w(i,j)$. The resulting graph is taken as an approximation to the MAWSS. An optimal algorithm for constructing optimal branchings is presented in ([23]).

Proposition 5.2: The output $\hat{G}$ of Algorithm 1, is a strongly connected spanning subgraph of G.

2) A Distributed MAWSS Algorithm: At the time of deployment, neither G_{R_0} nor $p_{ij}(\underline{\alpha})$ is known to sensors. Over time, sensors "discover" each other by advertising their ids which can be simply their indices. Let $\underline{\alpha}$ and the locations of the sensors be fixed. At time 0, the sensors start broadcasting their ids. Let $G_n = (V_n, E_n)$ denote the subgraph of G_{R_0} discovered until time n, i.e., $V_n = V_s$ and $(i,j) \in E_n$ if there exists a time slot $m \leq n$ in which sensor j successfully received a transmission from i for the first time. $G_0 = (V_s, \phi)$. Note that G_n is a random graph. In addition to noting ids of its neighbours, a sensor also *counts the number of times it received a particular id*; the larger this number, the higher is the probability of successful transmission from that node to i. To make it precise, let $S_{ij}(n)$ denote the number of times sensor j successfully received i till time n. Then the following holds.

Proposition 5.3: Let $0 < \alpha_i < 1$ for each i. Then $G_n \to G_{R_0}$ and $\frac{S_{ij}(n)}{n} \to p_{ij}(\underline{\alpha})$ with probability 1.

The convergence of the discovery process is in itself an interesting problem since how fast G_n converges to G_{R_0} will depend on $\underline{\alpha}$. Practically, sensors will carry out the discovery process for either a pre-programmed number of slots, or during the discovery phase they will detect when the graph is connected and then stop. For this discussion we will assume that either G_{R_0} or a connected subgraph of it has been discovered and sensor i has an estimate of $p_{ij}(\underline{\alpha})$ for each (i,j) discovered; j counts the number of times it received the id from i and sends back the number to i; i divides it by the number of slots to form an estimate. A distributed version of Algorithm 1 is presented in [24]. The algorithm works by formation of node clusters, detection of cycles, selection of minimum weight cluster incoming edge in a distributed fashion. We omit the details.

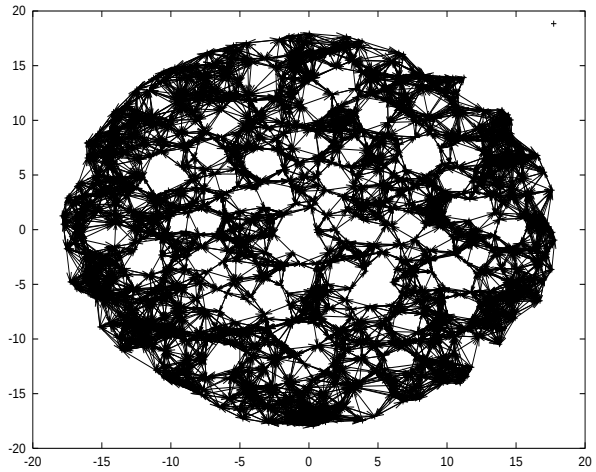


Fig. 3. Topology discovered till 500 slots (G_{500}) by 1000 sensors with $\lambda = 1$ per m^2 and $\alpha = 0.05$. x and y axis scale is distance in m.

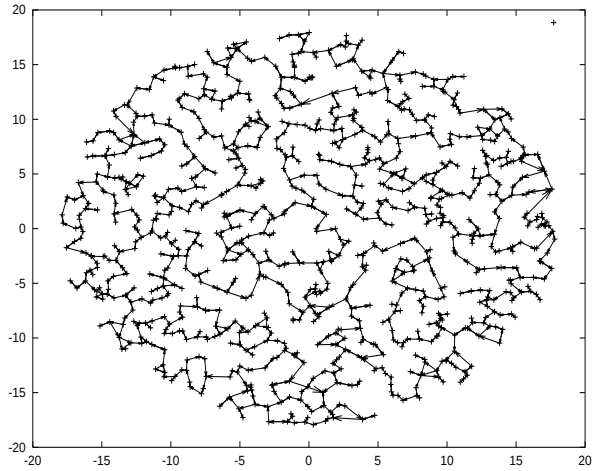


Fig. 4. MAWSS constructed from the discovered topology G_{500} of 1000 sensors with $\lambda = 1$ per m^2 and $\alpha = 0.05$. x and y axis scale is distance in m.

B. Results

The setup is as explained in Section IV. 1000 sensors form a Poisson field on the plane with $\lambda = 1$. In this set of results, we use the same value of attempt probability for each sensor. Further, two "types" of α's need to be distinguished. The first, denoted by α_d, is the attempt probability sensors use while *discovering the topology* (subscript 'd' denotes discovery). We use $\alpha_d = 0.01$ or 0.05. For $\alpha_d = 0.05$, the discovered graph is connected at 500 slots and for $\alpha_d = 0.01$ it is connected at 1000 slots. Figure 3 shows G_{500} (recall that G_n denotes the discovered graph at slot n) and Figure 4 shows the MAWSS constructed from it for $\alpha_d = 0.05$; MAWSS here refers to the graph obtained from Algorithm 1.

Once the topology formation is complete, sensors switch to an "operational value" of the attempt probability. Figure 5 shows the variation of average saturation throughput of a sensor with the operational values of α for network topologies

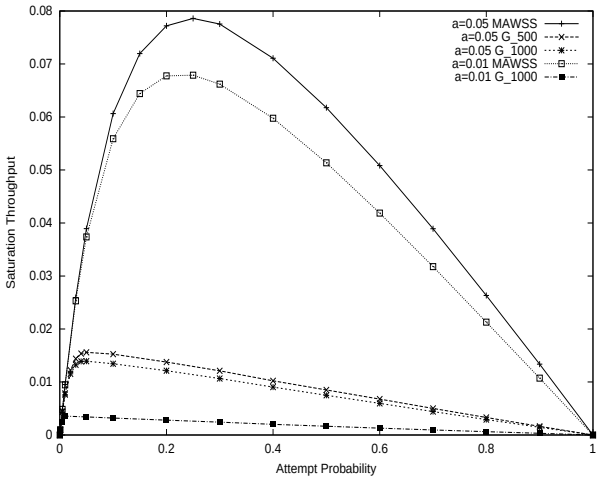


Fig. 5. Saturation Throughput variation with attempt probability (α) for different topologies: G_{500}, G_{1000}, MAWSS constructed using $\alpha_d = 0.05$ and G_{1000}, MAWSS using $\alpha_d = 0.01$.

given by G_{500}, G_{1000}, MAWSS constructed using $\alpha_d = 0.05$ and G_{1000}, MAWSS for $\alpha_d = 0.01$. Recall that, for a given topology G' and $\underline{\alpha}$, $M(G', \underline{\alpha})$ denotes the network throughput. The average saturation throughput which we plot in Figure 5 is simply $\frac{M(G',\underline{\alpha})}{N}$. Note that for $\alpha = 0.05$, throughput of G_{1000} is lower than G_{500} since it includes more edges of low probability of success discovered during additional 500 slots. A lower value of α_d tends to discover longer edges too which reduce the throughput; hence for G_{1000}, performance with $\alpha_d = 0.01$ is worse than with $\alpha_d = 0.05$. MAWSS, on the other hand, eliminates edges with low probability of success while maintaining the connectivity; hence, the maximum throughput achieved by MAWSS (for both the values of α_d) is almost five times of the corresponding discovered graphs. Note from (1) and (3) that for any connected topology the average saturation throughput at α cannot exceed $\alpha(1-\alpha)$, e.g., for $\alpha = 0.25$ the throughput can be no more than 0.1875.

VI. Optimal Attempt Probabilities: Objectives and Algorithms

Apart from the observation that MAWSS gives significant throughput improvement, a crucial observation from Figure 5 is that the throughput is maximised at a different value of α than α_d. For example, for the MAWSS constructed with $\alpha_d = 0.05$, the throughput is maximised at $\alpha = 0.25$. At this α, the throughput is almost five times of G_{500} and an order of magnitude more than that of G_{1000} with $\alpha_d = 0.01$. *Thus, it is essential to actually operate the network at a throughput-maximising value of* α.

One way to solve this problem is the following. From the number of sensors to be used and the approximate area of region to be monitored, estimate the density. Using this estimate, fix an initial value of α which will lead to fast discovery and MAWSS formation. After the topology is formed, let sensors switch to a pre-programmed α which maximises throughput for that density. Though feasible and simple this approach has some problems. The α maximising the throughput can be known for specific point placements (or averages thereof). In the field, sensors will fall as one particular sample path and then it is not clear whether that value of α will maximise the throughput. A more powerful approach is to let the sensors *learn the value of* α *over time*. We do not insist that all the sensors use the same value. First because maintaining the same value of α at every sensor at every step of the learning process is difficult. More importantly, different sensors may need different values of α to counter the local inhomogeneities in the node placement. This "learning" approach will also help sensors to reconfigure themselves if and when some sensors fail.

Definition 6.1: For a given topology, an *independent set* is a set of transmitter-receiver pairs which do not interfere with each others' transmission, i.e., if only the sensors chosen as transmitters in this set are allowed to transmit to the respective receivers, then their transmissions are successful with probability 1. □

Proposition 6.1: Let A_{max} denote the largest independent set in a given topology. Then $\underline{\alpha}$, which maximises the network throughput, is the one with $\alpha_i = 1$ for all transmitters $i \in A_{max}$ and $\alpha_i = 0$ for the rest.

It is clear from Proposition 6.1 that if sensors are allowed to use different values of α_is then the maximisation of average saturation throughput leads to a degenerate assignment. In addition, the "maximum calculation" example in Section I suggests that the overall progress of the computation will be really limited by the lowest sensor throughput in the network. This motivates the problem of *maximising the minimum of sensor throughputs*.

A. The MAXMIN Throughput

For a given network topology G, consider the following optimisation problem.

$$\max_{\underline{\alpha} \in [0,1]^N} \min_{1 \le i \le N} M_i(G, \underline{\alpha}) \tag{4}$$

In order to get some insight into the throughput functions, $M_i(G, \underline{\alpha})$, recall that $N_i(G)$ (respectively $n_i(G)$) denote the set of neighbours (respectively number of neighbours) of i. Let $\underline{\alpha}^{ij}$ denote the vector $\underline{\alpha}$ with entries α_i and α_j omitted.

Proposition 6.2: For a fixed topology G, η and β, $M_i(G, \underline{\alpha})$ has the following form.

$$M_i(G, \underline{\alpha}) = \frac{1}{n_i(G)} \sum_{j \in N_i(G)} \alpha_i (1 - \alpha_j) g_{ij}(\underline{\alpha}^{ij})$$

For each $j \in N_i(G)$, $g_{ij}(.)$ either equals 1 or there exists a set $I_{ij} \subseteq V_s \backslash \{i, j\}$ such that $g_{ij}(.)$ is a decreasing and affine function of α_k, $k \in I_{ij}$ and does not depend upon α_k, $k \notin I_{ij}$. Moreover, $g_{ij}(\underline{1}) = 0$ and $g_{ij}(\underline{0}) = 1$. □

It is clear from Proposition 6.2 that $M_i(G, .)$, $1 \le i \le N$ are continuous functions of $\underline{\alpha}$, and so is $\min_i M_i(G, .)$. Therefore, an optimum exists for the MAXMIN problem (4) by Weierstrass Theorem. Since topology G is fixed, henceforth we suppress it from the notation. It is, however, assumed that

G is connected. Let $\underline{\alpha}^*$ denote an optimum of MAXMIN and M^* denote $\min_i M_i(\underline{\alpha}^*)$. We will call $\underline{\alpha}^*$, the *MAXMIN throughput attempt probabilities* (MMTAP).

Definition 6.2: Let the sensor locations be fixed. Then for fixed β and η, j is called an interferer of i if $M_i(G, \underline{\alpha})$ is decreasing in α_j. j is called a primary interferer of i if whenever j transmits, i either decodes it or decodes none. □

Proposition 6.3: If every sensor is a primary interferer of at least one sensor, then $\underline{0} < \underline{\alpha}^* < \underline{1}$.

Consider first N collocated sensors; by collocated we mean that in any slot at most one transmission can be successful. Then $M_i(\underline{\alpha}) = \alpha_i \ \Pi_{j \neq i}(1 - \alpha_j), \ 1 \leq i \leq N$ and $\alpha_i^* = \frac{1}{N}$ which is an intuitive and desirable operating point for sensors in this scenario. Secondly, even when sensors are spatially distributed, $\underline{\alpha}^*$ equalises the throughputs, i.e.,

Proposition 6.4: $\underline{0} < \underline{\alpha}^* < \underline{1} \Rightarrow M_i(\underline{\alpha}^*) = M_j(\underline{\alpha}^*), \ 1 \leq i, j \leq N$.

The *throughput equalizing property* makes the MMTAP particularly important; with MMTAP, sensors operate at equal processing rates which is desirable in applications where computations are iterative.

B. An MMTAP Algorithm

Consider Algorithm 2, an iterative scheme to tune $\underline{\alpha}$ to the MMTAP. Π denotes projection on $[0, 1]$ and $|U(k)|$ the cardinality of set $U(k)$. $a_j(k)$ is the step size in the k^{th} iteration at sensor j. Algorithm 2 is a "generalised gradient ascent" algorithm; $\frac{1}{|U(k)|} \sum_{i \in U(k)} \frac{\partial M_i(\underline{\alpha}(k))}{\partial \alpha_j}$ being a generalised gradient of $\min_i M_i(\underline{\alpha}(k))$ at $\underline{\alpha}(k)$ ([25]). Informally the iterations can be explained as follows. $U(k)$ denotes the set of sensors whose throughput is the minimum under operating point $\underline{\alpha}(k)$. If $j \notin U(k)$, then α_j is reduced in the next iteration since $\frac{\partial M_i(\underline{\alpha})}{\partial \alpha_j} \leq 0, i \neq j$ (see Proposition 6.2). This leads to an increase in the throughput of $i \in U(k)$. If $j \in U(k)$, then α_j is increased or decreased based on how it affects others and how others affect its throughput. Thus the algorithm tries to equalize as well as maximise the sensor throughputs.

$$
\begin{aligned}
\alpha_j(0) &\in [0,1], \ j = 1, 2, \ldots, N \\
u(k) &= \min_{1 \leq i \leq N} M_i(\underline{\alpha}(k)), \ k \geq 0 \\
U(k) &= \{i | 1 \leq i \leq N, M_i(\underline{\alpha}(k)) = u(k)\} \\
\alpha_j(k+1) &= \Pi \left[\alpha_j(k) + \frac{a_j(k)}{|U(k)|} \sum_{i \in U(k)} \frac{\partial M_i(\underline{\alpha}(k))}{\partial \alpha_j} \right] \\
& \quad j = 1, 2, \ldots, N
\end{aligned}
$$

Algorithm 2: An MMTAP algorithm using generalised gradient ascent.

1) A Synchronous Distributed Stochastic Algorithm: Though fixed in form for a given placement of nodes, $M_i(.)$ is not known at sensor i and being a steady-state average, only *noisy measurements* of $M_i(.)$ are available for Algorithm 2. An unbiased estimator of $M_i(.)$, denoted by $\hat{M}_i(.)$, is $\frac{1}{\tau} \sum_{j=1}^{\tau} X_i(j)$ where $X_i(j) = 1$ if i transmits successfully in slot j, otherwise 0. τ is the number of estimation slots. Sensors also need to *estimate the gradient* of $M_i(.)$ in order to use Algorithm 2. Since we need a distributed algorithm and since IPA and LR-SF ([26]) cannot be applied in this case, an appropriate method for gradient estimation is simultaneous perturbation (SP, [27]). Instead of perturbing one component, i.e., α_i at a time to obtain the estimates of partial derivatives, in SP all α_is can be perturbed simultaneously given that perturbations for each α_i are zero mean independent random variables with some additional conditions ([27]). This way, *by choosing the perturbation amount locally, sensors can simultaneously estimate the derivatives.* In the k^{th} iteration, let $\underline{\Delta}(k)$ denote a vector of N independent Bernoulli random variables such that $\{\underline{\Delta}(k)\}$ is an independent sequence with $\underline{\Delta}(k)$ independent of $\underline{\alpha}(0), \underline{\alpha}(1), \ldots, \underline{\alpha}(k)$. Then the "central-difference estimator" of $\frac{\partial M_i(\underline{\alpha})(k)}{\partial \alpha_j}$ is $\frac{\hat{M}_i(\underline{\alpha}(k) + c(k)\underline{\Delta}(k)) - \hat{M}_i(\underline{\alpha}(k) - c(k)\underline{\Delta}(k))}{2c(k)\Delta_j(k)}$ where $c(k)$ is a scalar. SP requires $c(k) \to 0$ so that the estimator is asymptotically unbiased.

Proposition 6.5: Let in Algorithm 2, the partial derivatives of $M_i(.), \ 1 \leq i \leq N$ be replaced by their estimates (biased or unbiased). Let $a_j(k) = a(k), \ 1 \leq j \leq N, \ k \geq 0$ and $a(k)$ satisfy $\sum_{k=1}^{\infty} a(k) = \infty$ and $\sum_{k=1}^{\infty} a(k)^2 < \infty$. Then the generated sequence $\{\underline{\alpha}(k), \ k \geq 1\}$ converges a.s. to the MMTAP.

For a complete distributed implementation we now only need a way of obtaining an estimate of $\frac{\partial M_i(\underline{\alpha}(k))}{\partial \alpha_j}$ for each $i \in U(k)$ at every sensor j in iteration k. First note from the form of the derivative estimator that sensor j does not require such individual estimates to calculate the sum of the partial derivative in Algorithm 2; if $\delta(k) := \sum_{i \in U(k)} \frac{\hat{M}_i(\underline{\alpha}(k) + c(k)\underline{\Delta}(k)) - \hat{M}_i(\underline{\alpha}(k) - c(k)\underline{\Delta}(k))}{|U(k)|}$ is made known to it, it can directly obtain the required sum simply by dividing $\delta(k)$ by $2c(k)\Delta_j(k)$. $\delta(k)$ can be obtained at each sensor by first collecting it at a node designated as the "root" using the distributed computation approach as in the "maximum calculation" example discussed in Section I, and then letting the root distribute this value to all the sensors. The same approach will work since we have a "minimum calculation" problem at hand, $u(k)$ being the minimum value of sensor throughputs. Recall that MAWSS is built using trees. Therefore, the computation will proceed efficiently using the underlying tree structure. We assume that such a "computational tree" with a root has been built; we denote by C_i the children of sensor i in the tree. The distributed MMTAP algorithm proceeds in synchronized rounds; in every round k all sensors use the same values of $c(k)$ and $a(k)$. The number of slots used for estimating $M_i(.)$ is denoted by τ. Algorithm 3 describes in detail the procedure to be executed at each sensor in a round. By the computation at a sensor we mean initialising $u_i(k)$, $r_i(k)$ and $\delta_i(k)$, updating them and forwarding them to the parent (see Algorithm 3); updation commences only after the corresponding values are obtained from every child. Then the key step is to note that, at the end of the computation at sensor i, $u_i(k)$ is the minimum sensor throughput *known to sensor*

operate with $\alpha_i(k)$ for next τ slots and obtain $\hat{M}_i(\underline{\alpha}(k))$ by counting the number of times transmission is successful and dividing it by τ
randomly choose $\Delta_i(k) \in \{-1, 1\}$
operate with $\alpha_i(k)+c(k)\Delta_i(k)$ for next τ slots and obtain $\hat{M}_i(\underline{\alpha}(k) + c(k)\underline{\Delta}(k))$
operate with $\alpha_i(k)-c(k)\Delta_i(k)$ for next τ slots and obtain $\hat{M}_i(\underline{\alpha}(k) - c(k)\underline{\Delta}(k))$
set $u_i(k) = \hat{M}_i(\underline{\alpha}(k))$, $\delta_i(k) = \hat{M}_i(\underline{\alpha}(k) + c(k)\underline{\Delta}(k)) - \hat{M}_i(\underline{\alpha}(k) - c(k)\underline{\Delta}(k))$ and $r_i(k) = 1$
receive $u_j(k)$, $\delta_j(k)$, $r_j(k)$ from each child $j \in C_i$
set $U_i(k) = \{j : j \in C_i \cup \{i\}, u_j(k) = \min(u_i(k), u_j(k),\ j \in C_i)\}$
update $r_i(k) = \sum_{j \in U_i(k)} r_j(k)$, $\delta_i(k) = \sum_{j \in U_i(k)} \delta_j(k)$ and $u_i(k) = \min(u_i(k), u_j(k),\ j \in C_i)$
if root **then**
    $\delta(k) = \frac{\delta_i(k)}{r_i(k)}$
    set $\alpha_i(k+1) = \alpha_i(k) + a(k)\frac{\delta(k)}{2c(k)\Delta_i(k)}$
    forward $\delta(k)$ to each $j \in C_i$
else
    forward $u_i(k)$, $\delta_i(k)$, $r_i(k)$ to the parent in the tree
    upon receiving $\delta(k)$ from the parent set $\alpha_i(k+1) = \alpha_i(k)+a(k)\frac{\delta(k)}{2c(k)\Delta_i(k)}$ and forward $\delta(k)$ to each $j \in C_i$

Algorithm 3: Algorithmic procedure to be executed at each sensor i in round $k \geq 0$.

i in round k, i.e., the minimum among those sensors which form a subtree of the computational tree rooted at i. If $U_i^s(k)$ denotes the set of sensors in the subtree rooted at i having the minimum throughput known to i then $r_i(k) = |U_i^s(k)|$ and $\delta_i(k) = \sum_{j \in U_i^s(k)}(\hat{M}_j(\underline{\alpha}(k) + c(k)\underline{\Delta}(k)) - \hat{M}_j(\underline{\alpha}(k) - c(k)\underline{\Delta}(k)))$. It thus follows that if i is the root then $u_i(k)$ is the global minimum throughput and in Algorithm 3, $\delta(k)$ equals $\sum_{i \in U(k)} \frac{\hat{M}_i(\underline{\alpha}(k)+c(k)\underline{\Delta}(k)) - \hat{M}_i(\underline{\alpha}(k)-c(k)\underline{\Delta}(k))}{|U(k)|}$. Therefore, at every sensor j, $\frac{\delta(k)}{2c(k)\Delta_j(k)}$ is the required estimate. We omit the details.

C. Results

In this section, we study the performance of Algorithm 3 on two example sensor networks. The networks are simple enough so that $M_i(.)$s can be deduced easily (hence the optimum) and insight into the algorithm can be obtained.

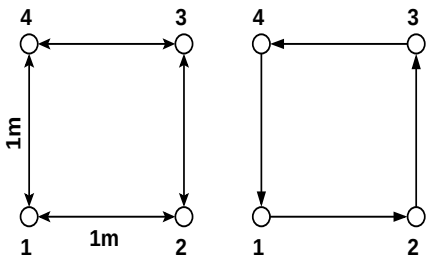


Fig. 8. 4 node network, $R_0 = 1$m, $\beta = 2$, $\eta = 4$; on left is G_{R_0} and on right is the operational topology

The first example, as shown in Figure 8, is a network of 4 sensors symmetrically placed and operating with one neighbour each. $\eta = 4, \beta = 2, R_0 = 1$m so that, $\forall\, i$ $M_i(\alpha_1, \alpha_2, \alpha_3, \alpha_4) = \alpha_i(1-\alpha_{i+1})(1-\alpha_{i+2})$, addition in the subscript being modulo 4. Note that sensor 3 is an interferer of the pair $(1, 2)$. It is also a primary interferer of sensor 2 and an interferer of 1 since $M_1(.)$ is decreasing in α_3. It is easy to see that $\underline{\alpha}* = (1/3, 1/3, 1/3, 1/3)$ and $M^* = 0.148$. (recall that M^* denotes $\min_i M_i(\underline{\alpha}^*)$).

Figure 6, and 7 show the variation of α_is and $\min_i M_i(\underline{\alpha})$ with the number of iterations in Algorithm 3. Recall that each iteration consists of three estimation intervals (τ). We have used $\tau = 1000$ slots. Adjacent point averaging has been done to show the trend in $\min_i M_i(.)$. We choose $a(k) = \frac{0.05}{k^{0.8}}$ and $c(k) = \frac{0.1}{k^{0.25}}$. Observe that within few iterations, the improvement in the performance is substantial.

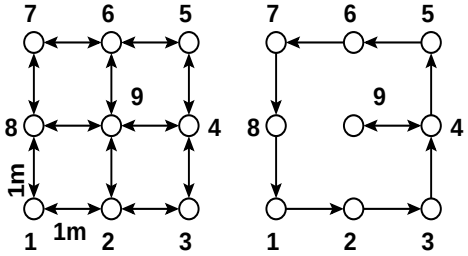


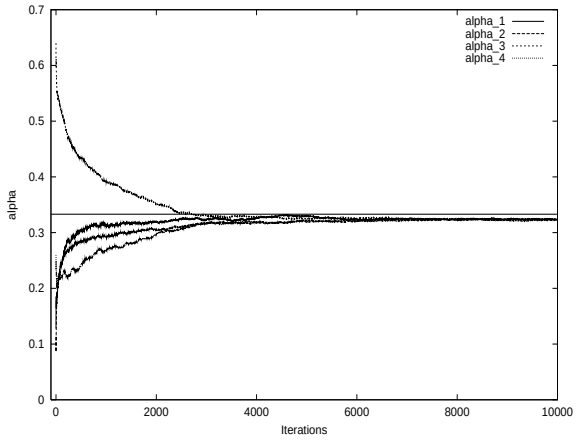
Fig. 9. 9 node network, $R_0 = 1$m, $\beta = 2$, $\eta = 4$; on left is G_{R_0} and on right is the operational topology

The second example, shown in Figure 9 is that of an asymmetric network. $\eta = 4, \beta = 2, R_0 = 1$m. The operational topology is such that

$$
\begin{aligned}
M_1(\underline{\alpha}) &= \alpha_1(1-\alpha_2)(1-\alpha_3)(1-\alpha_9) \\
&\quad (1-\alpha_4\alpha_8(1-(1-\alpha_7)(1-\alpha_6)(1-\alpha_5))) \\
M_2(\underline{\alpha}) &= \alpha_2(1-\alpha_3)(1-\alpha_4) \\
M_4(\underline{\alpha}) &= \alpha_4[(1-\alpha_5)(1-\alpha_6) + (1-\alpha_9)(1-\alpha_2) \\
&\quad (1-\alpha_8)(1-\alpha_6) \\
&\quad (1-(\alpha_5\alpha_1\alpha_3(1-\alpha_7) + \alpha_5\alpha_1\alpha_7(1-\alpha_3) + \\
&\quad \alpha_5\alpha_3\alpha_7(1-\alpha_1) + \alpha_3\alpha_1\alpha_7(1-\alpha_5) + \\
&\quad \alpha_3\alpha_1\alpha_7\alpha_5))]/2
\end{aligned}
$$

From Figure 9 is can be seen that, $M_3(.), M_5(.), M_7(.)$ and $M_9(.)$ are similar in form to $M_1(.)$ whereas M_6 and M_8 are similar in form to M_2. Note the interferers from $M_i(.)$s, e.g., sensor 3 is a primary interferer for sensor 2, so is 9. Sensors 4 and 8 do not disrupt a transmission from 1 to 2 individually but together along with at least one sensor from 5, 6 and 7 do. Thus, by Definition 6.2, 4, 5, 6, 7 and 8 are interferers of sensor 1.

Table I shows the comparison of performance of Algorithm 3 (indicated by SPSA) with the deterministic Algorithm 2 (indicated by SDA); in SDA the exact gradients obtained explicitly from the form of $M_i(.)$s are used. We choose $a(k) = \frac{0.1}{k^{0.7}}$ and $c(k) = \frac{0.1}{k^{0.15}}$. For each of the two sets of $\underline{\alpha}(0)$, the first column indicates $\underline{\alpha}(0)$, and second and



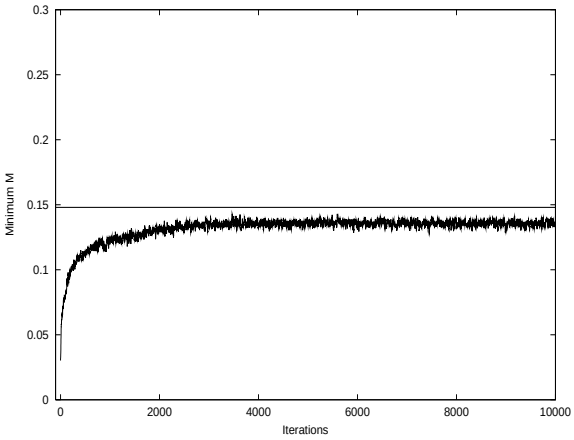
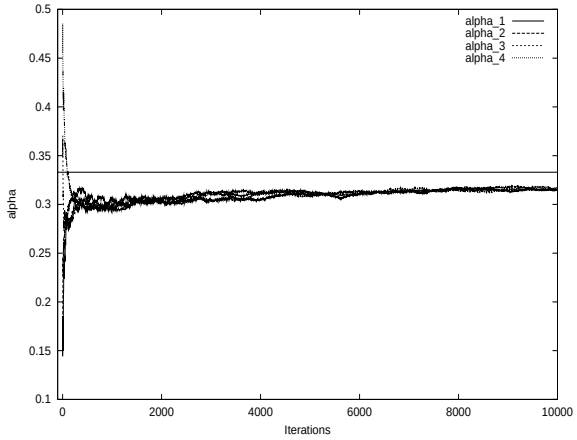


Fig. 6. 4 sensor example; variation of α_is and $\min_i M_i(.)$ with the number of iterations. $\underline{\alpha}(0) = (0.0978, 0.149, 0.61, 0.2301)$. The horizontal lines indicate $\alpha_i^* = 1/3,\ 1 \leq i \leq 4$ and $M^* = 0.148$.



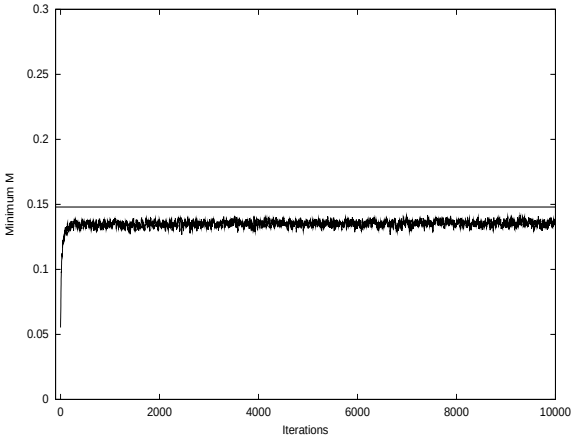


Fig. 7. 4 sensor example; variation of α_is and $\min_i M_i(.)$ with the number of iterations. $\underline{\alpha}(0) = (0.496, 0.129, 0.228, 0.074)$. The horizontal lines indicate $\alpha_i^* = 1/3,\ 1 \leq i \leq 4$ and $M^* = 0.148$.

	set 1			set 2		
	(0)	SPSA	SDA	(0)	SPSA	SDA
α_1	0.494	0.334	0.281	0.916	0.570	0.281
α_2	0.129	0.218	0.206	0.219	0.224	0.204
α_3	0.228	0.305	0.287	0.387	0.355	0.286
α_4	0.074	0.257	0.237	0.699	0.342	0.237
α_5	0.255	0.295	0.271	0.294	0.380	0.274
α_6	0.609	0.313	0.192	0.520	0.305	0.192
α_7	0.396	0.217	0.278	0.616	0.464	0.276
α_8	0.021	0.217	0.194	0.270	0.278	0.196
α_9	0.377	0.297	0.289	0.467	0.346	0.289
min $M_i(.)$	0.008	0.091	0.109	0.014	0.078	0.109

TABLE I

COMPARISON OF PERFORMANCE OF THE ALGORITHM USING GRADIENT ESTIMATES (SPSA) AND EXACT GRADIENTS (SDA) FOR 2 SETS OF INITIAL $\underline{\alpha}(0)$ WITH 20000 ITERATIONS.

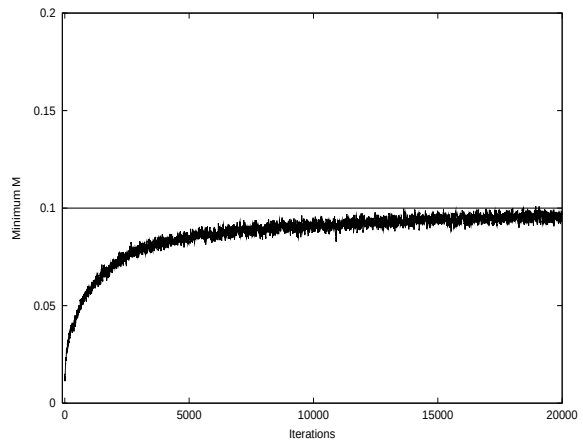


Fig. 10. 9 sensor example; variation of $\min_i M_i(.)$ with the number of iteration for set 1 $\underline{\alpha}(0)$ in Table I. The horizontal line indicates $\min_i M_i(\underline{\alpha}(20000)) = 0.109$ obtained with exact gradients (SDA).

third column indicate $\underline{\alpha}(20000)$ obtained by SPSA and SDA respectively Note that, for set 1, $\min_i M_i(\underline{\alpha}(0))$ is only 0.008 whereas with SDA $\min_i M_i(\underline{\alpha}(20000)) = 0.109$ and with SPSA it is 0.091. Similar observations hold for set 2. Figure 10 shows the trends in $\min_i M_i(.)$ with the number of iterations for the set 1 shown in Table I. Observe that, within a few iterations, the performance gains are significant.

VII. Discussion

It is essential that after their deployment, sensors organise into an optimal network as fast as possible. This is particularly true of the network topology. Our approximation algorithm for MAWSS uses branchings whose time complexity is known to be $O(N^2)$ for dense networks ([23]). The time (and message) complexity of the distributed algorithm discussed in Section V-A.2 which finds N branchings also equals $O(N^2)$ ([24]). This cost appears to be imperative for forming an optimal topology. In return, as already seen, the performance gain is substantial (Figure 5). Algorithm V-A.2 also constructs directed trees rooted at each sensor, which can be used in computational algorithms and for control information propagation; recall that, our MMTAP algorithm makes use of this fact. Our approach can be extended directly to a k-connected or a symmetric topology (if i has a link to j, j has to have the reverse link). Note that, symmetric topology problem is also NP-complete since MAWSS for undirected graphs is a special case of it. An algorithm for this problem can be found in [28]. Learning an optimal $\underline{\alpha}$ is an important but much harder problem. Our algorithm is simple and makes use of measurements made locally. Its major complexity is in obtaining the estimates of partial derivatives of throughputs at each sensor. Stochastic algorithms are constrained by the "bias-variance dilemma" ([29]), therefore, their convergence properties can be improved by careful selection of the parameters. In our examples, the starting points were chosen arbitrarily. Practically, a sensor can guess its primary interferers from the estimates of probability of successful transmission obtained during the discovery phase so that it is possible to find a good starting point for the algorithm to improve its convergence to the optimum. The most important point, however, is that the improvement within a few iterations is significant. So the network after achieving certain improvement or a target rate may stop executing the algorithm.

Interestingly, such algorithms can also be seen as a tool by which the *network slowly and continuously keeps on improving itself*. This aspect is particularly important because even if some sensors fail over time, the remaining sensors can reconfigure themselves with such an algorithm. Note that, our algorithms are measurement based hence it is be possible to extend our approach to other access schemes too. The other important advantage of stochastic algorithm is that the throughputs will be measured using the real transmissions, no special packet transmissions are required. Hence, there is no extra energy consumption. Further, they will work even in the presence of any energy saving techniques such as random sleep time and can account for energy constraints directly, for example, by upper bounding the attempt probabilities.

We designed algorithms so as to achieve optimal performance and found correspondingly higher algorithmic complexity. Our future work, therefore, is to develop asynchronous algorithms with strictly local information exchange for scalability. This paper lends support to any such effort since it shows a way to compute the global optimal performance against which the performance of other algorithms can be compared.

VIII. Conclusion

We viewed performance optimisation as the objective for self-organisation in sensor networks and argued that the rate at which a sensor network can process data in a distributed fashion is governed by its communication throughput; hence the self-organisation should be *throughput optimal*. Using a simple model, we showed that the network topology and optimal transmission attempt rate are the critical factors which determine the throughput.

We obtained the optimal topology by MAWSS formulation and discussed a distributed algorithm for it. This algorithm uses connectivity and probability of successful transmission information which can be obtained locally. It was seen in an example that such a topology gave almost five times the throughput of the original topology. The overall progress of iterative computations in a sensor network is limited by the minimum of the sensor throughputs. Therefore, maximisation of minimum throughput is an important problem. We characterised the optimum attempt probabilities, MMTAP for this problem. The MMTAP were found to have an important throughput-equalizing property. We presented a synchronous distributed stochastic algorithm for driving a sensor network to the MMTAP. The algorithm uses local throughput measurements and yields substantial performance improvement even within few iterations.

The performance improvement is at the price of algorithmic complexity. However, this work shows that the performance gains from optimal self-organisation can be substantial and such techniques need to be considered during the protocol design.

IX. Acknowledgements

This research was supported in part by a grant from the Indo-French Centre for the Promotion of Advanced Research (IFCPAR) (Project No. 2900-IT), and in part by a fellowship from the IBM India Research Lab.

Appendix

In this section, we sketch the proofs of some propositions in the paper.

Proof of Proposition 5.1: The proof for undirected graphs is by transformation to 3DM (3-dimensional matching, [30]).
MAWSS instance: Connected graph $G = (V, E, W)$, and a positive integer B.
Question: Is there a set $G_1 \in G_{cs}$ such that $M(G_1) \leq B$?
3DM instance: A set $S \subseteq W \times X \times Y$, where W, X, and Y are disjoint set having the same number q of elements.
Question: Does S contain a matching, i.e., a subset $S' \subseteq S$ such that $|S'| = q$ and no two elements of S' agree in any co-ordinate.

We construct the following gadget; R_x and X denote sets of q points such that there is an edge between points of R_x and X as shown in Figure 11 of weight 1. Each point of

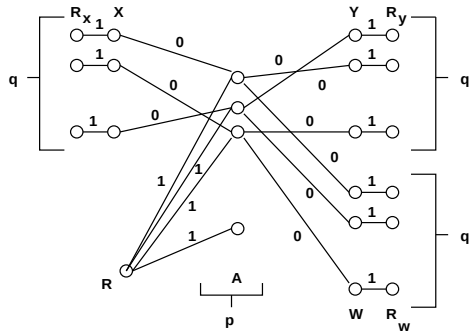


Fig. 11. Gadget for proving NP-completeness of undirected MAWSS

X has an additional edge of weight 0. Similarly for R_y, Y, R_w and W. A is a set of p points ($p > q$). R is a single node which has p edges from set A with weights 1 each. Additional edges of set X can only be connected to A. Claim: A matching exists if and only if a connected spanning graph with $M = (1+3q) + \frac{3q}{2} + (p - \frac{3q}{4})$ can be constructed.

Note that points in R_x have weight 1 each and points in X have $1/2$ each. Thus total (average) weight of points in R_x, X, R_y, Y, R_w and W is $3q + \frac{3q}{2}$. R has average weight 1. These are unchanging weights in the graph. Weights of points in A change based on how X, Y and W connect.

If a matching exists, then out of p, q points in A have 3 edges each of weight 0 (from X, Y and W), 1 edge to R and rest $p-q$ points have only 1 edge (connected to R). Thus weight of set A is $p - q + q \times 1/4 = p - 3q/4$. The resulting graph is connected with $M = (1+3q) + \frac{3q}{2} + (p - \frac{3q}{4})$.

Lemma 1.1: Let

$$f(\underline{k}) = \frac{1}{k_1+1} + \frac{1}{k_2+1} + \ldots + \frac{1}{k_p+1}$$

and consider the problem

$$\begin{array}{rcl} \max & f(\underline{k}) & \\ \text{subj.to } 0 \leq & k_i & \leq 3 \;\; 1 \leq i \leq p \\ k_1 + k_2 + \ldots + k_p & = & 3q \end{array}$$

Then, the max $f(\underline{k}) = p - 3q/4$ and is achieved when in $\underline{k}$, q k_is equal 3 and the rest $p-q$ are 0.

Lemma 1.1 implies that if a connected graph with $M = (1+3q) + \frac{3q}{2} + (p - \frac{3q}{4})$ exists, then out of p points in A, q have 3 edges connected to X, Y and W, 1 edge to R while the rest have only 1 edge to R. This implies that the matching exists (q points with edges from X, Y, W is the required matching).

NP-completeness of MAWSS for directed graphs is established by noting that STA (strong connectivity augmentation, [31]) is its special case. □

Proof of Proposition 5.2: Follows from noting that Algorithm 1 constructs a route from every node to every other node. □

Proof of Proposition 5.3: Since $0 < \alpha_i < 1$ for each i, $p_{ij}(G_{R_0}, \underline{\alpha}) > 0$ for each $(i,j) \in E_{R_0}$. Therefore, the probability that (i,j) is discovered in finite time is 1. Since N is finite, $G_n \to G_{R_0}$ in finite time with probability 1. The second limit follows from the strong law of large numbers. □

Proof of Proposition 6.1: The network throughput is upper bounded by the maximum number of successful transmissions in a slot. The upper bound is achieved by assigning $\alpha_i = 1$ for transmitter $i \in A_{max}$ and $\alpha_i = 0$ to the rest. □

Proof of Proposition 6.2: Let the sensor locations be fixed. Recall Equation 1 and Equation 3. Note that, $g_{ij}(\underline{\alpha}^{ij})$ is $P(\Gamma_{ij} \geq \beta)$ with d_{kj} fixed; recall that Γ_{ij} denotes the SIR of a transmission from i to j. If $\frac{(\frac{d_{ij}}{d_0})^{-\eta}}{\sum_{k \neq i,j}(\frac{d_{kj}}{d_0})^{-\eta} + N_0} \geq \beta$, $g_{ij}(\underline{\alpha}^{ij}) = 1$. If not, let $\underline{v}$ denote an N-dimensional vector whose each component is either 0 or 1. Let,

$$\mathcal{V} = \left\{ \underline{v} | \frac{(\frac{d_{ij}}{d_0})^{-\eta}}{\sum_{k \neq i,j}(\frac{d_{kj}}{d_0})^{-\eta} v_k + N_0} \geq \beta \right\}$$

Then, $P(\Gamma_{ij} \geq \beta) = \sum_{\underline{v} \in \mathcal{V}} \Pi_{k \neq i,j} \alpha_k^{v_k} (1-\alpha_k)^{(1-v_k)}$ Let $\underline{v}_{-m}$ denote a vector with m^{th} entry omitted and let $(\underline{v}_{-m}, v_m)$ represent $\underline{v}$. If there exists an l such that for every $(\underline{v}_{-l}, 1) \in \mathcal{V}$, $(\underline{v}_{-l}, 0) \in \mathcal{V}$, $P(\Gamma_{ij} \geq \beta)$ does not depend on l. Let I_{ij} be the set of sensors for which the previous condition fails. That $g_{ij}(\underline{\alpha}^{ij})$ is decreasing and affine in α_k, $k \in I_{ij}$ follows from the form of $P(\Gamma_{ij} \geq \beta)$.

Proof of Proposition 6.3: If $\alpha_i^* = 0$ for some i then clearly $M_i(\underline{\alpha}^*) = 0$. If $\alpha_i^* = 1$ for some i then $M_j(\underline{\alpha}^*) = 0$, $i \in P_j$ where P_j are the primary interferers of j. Proposition 6.2 implies that if $\alpha_i \in (0,1)$ for all i, $M_i(G, \underline{\alpha}) > 0$, $1 \leq i \leq N$. Hence, $\underline{0} < \underline{\alpha}^* < \underline{1}$. □

Proof of Proposition 6.4: Note that the MAXMIN problem is equivalent to the following problem.

$$\begin{array}{rcl} \max & x & \\ \text{subj.to } M_i(\underline{\alpha}) & \geq & x, \; 1 \leq i \leq N \\ x & \geq & 0 \\ \alpha_i & \in & [0,1], \; 1 \leq i \leq N \end{array}$$

Since $M_i(\underline{\alpha}) \leq 1$, $x \leq 1$. The KKT conditions for the problem imply that at if $\underline{\alpha}^*$ is regular, then there exists $\underline{\mu}^* \geq 0$ such that following holds.

$$\sum_{i=1}^{N} \mu_i^* \frac{\partial M_i(\underline{\alpha}^*)}{\partial \alpha_j} = 0 \qquad (5)$$

$$1 - \sum_{i=1}^{N} \mu_i^* = 0$$

$$\mu_i^* = 0 \text{ if } M_i(\underline{\alpha}^*) > M^*$$

Recall Definition 6.2. Let $R_j = \{i | j \in N_i(G)\}$ and $S_j = \{i | j \in I_i\}$; I_i denotes the set of interferers of i. Let $\mu_j^* = 0$ for some j. Then Equation 5 corresponding to j implies that $\sum_{i: i \neq j} \mu_i^* \frac{\partial M_i(\underline{\alpha}^*)}{\partial \alpha_j} = 0$.

Lemma 1.2: If $\underline{0} < \underline{\alpha}^* < \underline{1}$ then $\frac{\partial M_i(\underline{\alpha}^*)}{\partial \alpha_j} = 0$ if and only if $i \notin R_j \cup S_j$.

$R_j \cup S_j = \phi$ means that no sensor transmits to j and j is not interferer of any sensors. j is thus an isolated sensor

and cannot belong to a connected network. Thus Equation 5 reduces to $\sum_{i:i\in R_j\cup S_j} \mu_i^* \frac{\partial M_i(\underline{\alpha}^*)}{\partial \alpha_j} = 0$. For each such i, $\frac{\partial M_i(\underline{\alpha}^*)}{\partial \alpha_j} < 0$ (Proposition 6.2 and Lemma 1.2) and $\mu_i^* \geq 0$. It follows that for all $i \in R_j \cup S_j$, $\mu_i^* \frac{\partial M_i(\underline{\alpha}^*)}{\partial \alpha_j} = 0$ and therefore $\mu_i^* = 0$. Continuing the argument for each such i and further, let A denote the final set $\{i|\mu_i^* = 0\}$. Let $B = \{k|k \notin A\}$. Then any such k does not transmit to any node in A and no sensor in A is an interferer of k. Since G is a connected topology, this implies that for all $k \in B,\ k \notin G$. Thus, for all $i \in G$, $\mu_i^* = 0$ which implies that $M_i(G, \underline{\alpha}^*) > M^*$. However, this is a contradiction since N is finite and the minimum is achieved. This proves that $\mu_i^* > 0, 1 \leq i \leq N$ and therefore the proposition.

Remark 1.1: The crucial condition for throughput equality is that G is connected. Connectedness imposes interference since if j receives from i, it is an interferer of i by Definition 6.2 and Proposition 6.2. Therefore, even if sensor i and j are not mutually interfering or transmitting to each other, their throughputs are coupled via intermediate sensors. If there are two disconnected clusters of sensor which are non-interfering, it is clear that their throughputs need not be equal. □

Proofs of Proposition 6.5: is based on the concept of generalised gradients of generalised differentiable functions. See [28], [25] for details.

REFERENCES

[1] S. Graham and P. R. Kumar, "The Convergence of Control, Communication, and Computation," in *PWC*, 2003.

[2] W. Zhang et al., "Distributed Problem Solving in Sensor Networks," in *AAMAS*, 2002.

[3] A. Knaian, "A Wireless Sensor Network for Smart Roadbeds and Intelligent Transportation Systems," Master of Engg. thesis, MIT, 2000.

[4] B. Sinopoli et al., "Distributed control applications within sensor networks," *Proc. of the IEEE*, vol. 91, no. 3, pp. 1235–1246, 2003.

[5] D. Baker and A. Ephremides, "The Architectural Organization of a Mobile Radio Network via a Distributed Algorithm," *IEEE Trans. on Commn.*, vol. 29, no. 11, November 1981.

[6] M. Post et al., "A Distributed Evolutionary Algorithm for Reorganizing Network Communications," in *IEEE MILCOM*, 1985.

[7] J. Gronkvist and A. Hansson, "Comparison between Graph-based and Interference-based STDMA Scheduling," in *MobiHOC*, 2001.

[8] K. Scott and N. Bambos, "Formation and Maintenance of Self-organizing Wireless Networks," in *31st Asilomar Conf. on Signals, Systems and Computers*, 1997.

[9] R. Krishnan and D. Starobinski, "Message-Efficient Self-Organization of Wireless Sensor Networks," to appear in IEEE WCNC, 2003.

[10] A. Cerpa and D. Estrin, "ASCENT: Adaptive Self-Configuring Sensor Network Topologies," UCLA Technical Report UCLA/CSD-TR 01-0009, May 2001.

[11] K. Sohrabi et al, "Protocols for Self-organization of a Wireless Sensor Network," *IEEE Personal Communications*, vol. 7, no. 5, pp. 16–27, October 2000.

[12] R. Rozovsky and P. R. Kumar, "SEEDEX: A MAC Protocol for Ad Hoc Networks," in *MobiHOC*, 2001.

[13] K. Romer, "Time Synchronization in Ad Hoc Networks," in *MobiHoc*, 2001.

[14] J. Elson et al., "Fine-Grained Network Time Synchronization using Reference Broadcasts," UCLA Technical Report 020008, 2002.

[15] Y. Akaiwa et al., "Autonomous Decentralized Inter-basestation Synchronization for TDMA Microcellular Systems," in *IEEE VTC*, 1991, pp. 257–262.

[16] A. Ebner et al., "Decentralized Slot Synchronization in Highly Dynamic Ad Hoc Networks," in *Intl. Symp. Wireless Personal Multimedia Commn.*, 2002, pp. 27–30.

[17] A. Domazetovic et al., "Estimating the Doppler Spectrum of a Short-Range Fixed Wireless Channel," accepted to IEEE Commn. Letters 2002.

[18] C. Kchao and G. Stuber, "Analysis of a Direct-Sequence Spread-Spectrum Cellular Radio System," *IEEE Trans. on Commn.*, vol. 41, no. 19, pp. 1507–1516, 1993.

[19] S. Tilak et al., "A Taxonomy of Sensor Network Communication Models," *Mobile Computing and Communication Review*, vol. 6, no. 2, April 2002.

[20] A. Karnik and A. Kumar, "Performance of a Distributed Computation Model of Wireless Sensor Networks," in *National Conference on Communication (NCC), India*, 2004.

[21] E. Gilbert, "Random Plane Networks," *SIAM Journal*, vol. 9, pp. 533–543, 1961.

[22] T. Philips et al., "Connectivity Properties of a Packet Radio Network Model," *IEEE Trans. on Information Theory*, vol. 35, no. 5, pp. 1044–1047, September 1989.

[23] R. E. Tarjan, "Finding Optimal Branchings," *Networks*, vol. 7, pp. 25–35, 1977.

[24] P. Humblet, "A Distributed Algorithm for Minimum Weight Directed Spanning Trees," *IEEE Trans. on Commn.*, vol. 31, no. 6, pp. 756–762, June 1983.

[25] Y. Ermoliev and V. Norkin, "Stochastic Generalized Gradient method with Application to Insurance Risk Management," Tech. Rep. International Institute for Applied Systems Analysis IR-97-021, 1997.

[26] P. L'Ecuyer, "An Overview of Derivative Estimation," in *Conf. on Winter Simulation*, 1991.

[27] J. Spall, "Multivariate Stochastic Approximation Using a Simultaneous Perturbation Gradient Approximation," *IEEE Trans. on Automatic Control*, vol. 37, no. 3, pp. 332–341, March 1992.

[28] A. Karnik, "Performance and Optimal Self-organisation of Wireless Sensor Networks," Ph.D. thesis, Indian Institute of Science, under preparation, 2003.

[29] B. Bharath and V. Borkar, "Stochastic Approximation Algorithms: Overview and Recent Trends," *Sadhana*, vol. 24, pp. 425–452, 1999.

[30] M. Garey and D. Johnson, *Computers and Intractability*, W. H. Freeman and company, 1979.

[31] G. Frederickson and J. Ja'Ja, "Approximation Algorithms for Several Graph Augmentation Problems," *SIAM Journal of Computing*, vol. 10, no. 2, pp. 270–283, May 1981.