Distributed Self-Tuning of Sensor Networks

Aditya Karnik[1], Anurag Kumar[1] and Vivek Borkar[2]

[1] Dept. of Electrical Communication Engineering,
Indian Institute of Science, Bangalore-560012, India
karnik@ece.iisc.ernet.in, anurag@ece.iisc.ernet.in
[2] School of Technology and Computer Science,
Tata Institute of Fundamental Research, Mumbai-400005, India
borkar@tifr.res.in

Abstract. This work is motivated by the need for an ad hoc sensor network to autonomously optimise its performance for given task objective and constraints. We consider the problem of maximising the minimum communication throughput of sensors in an ad hoc sensor network and investigate adaptive algorithms using which sensors can ''tune" to the optimal channel attempt rates in a distributed fashion; the optimality is in the sensor of maximising the minimum throughput. The adaptive scheme is a stochastic gradient algorithm derived from the augmented Lagrangian formulation of the optimisation problem. Our algorithms are promising and can be seen as a step towards developing optimally self-organising architectures for sensor networks.

1 Introduction

Equipped with a microprocessor, memory, radio and battery, "smart sensors" bring the new dimension of embedded computing, communication and control into distributed instrumentation based on ad hoc sensor networks. A smart sensor may have only modest computing power, but the ability to communicate will allow a group of sensors to collaborate to execute tasks more complex than just sensing and forwarding the information. Thus sensors may process sensed data on-line in a distributed fashion and yield partial or even complete results to an observer. This would not only facilitate computing but also save energy by avoiding long haul transport of sensed data to the observer. However, the practicality of such an approach may be limited by the rate at which sensors can collaboratively process data, since, unlike conventional distributed processing machines which have high speed communication buses between their processors, a sensor network has low speed wireless links, lacks a centralised co-ordinator, is, in many cases, ad hoc in nature and is energy constrained. *It is, therefore, imperative that sensors autonomously optimise their computing rate for given task objectives and constraints.* A simple class of distributed computing algorithms would require each sensor to periodically exchange the results of local computation with neighbouring sensors. The more frequently such exchanges can occur, the more rapidly will the overall computation converge. Hence, by optimising their communication throughput sensors can optimise their computing rates.

Consider a sensor network comprising n nodes, indexed by $i \in \{1, 2, \cdots, N\}$; we denote this set by V_s. Time is slotted, and each packet occupies one slot. The problem of learning a good topology has been addressed in [8]. We, therefore, assume that the topology is already learnt, and a sensor knows which other sensors are its neighbours. Sensor i attempts a transmission in any slot with probability α_i independent of everything else. If a sensor decides to transmit, it addresses its transmission to one of its neighbours randomly [3]; this can be attributed to isotropy of the physical process and uniformity of processing. We assume that a transmission is successful if its signal-to-interference ratio at the receiver is above a given threshold β; the signal is assumed to undergo only path loss with exponent η. Details of the model can be found in [8]. Let N_i (resp. n_i) denote the set (resp. number) of neighbours of sensor i. Let $\underline{\alpha}$ denote the vector of attempt probabilities and $M_i(\underline{\alpha})$ the throughput of sensor i under the attempt probabilities $\underline{\alpha}$. $\underline{\alpha}^{ij}$ denotes the vector $\underline{\alpha}$ with entries α_i and α_j omitted. Then assuming that sensing and computing is such that each sensor always has a packet to transmit, for a given topology, and given values of η and β ([8]),

$$M_i(\underline{\alpha}) = \frac{1}{n_i} \sum_{j \in N_i} \alpha_i (1 - \alpha_j) g_{ij}(\underline{\alpha}^{ij}), \;\; i = 1, 2, \ldots, N \tag{1}$$

[3] This is not a restrictive assumption and may be relaxed without any change in the following analysis.

where for each $j \in N_i$, $g_{ij}(.)$ 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 $\alpha_k,\ k \in I_{ij}$ and does not depend upon $\alpha_k,\ k \notin I_{ij}$. Moreover, $g_{ij}(\underline{1}) = 0$ and $g_{ij}(\underline{0}) = 1$. The parameters β and η appear implicitly in the function $g_{ij}(.)$.

Suppose N sensors are collocated, i.e., they are located such that in any slot at most one transmission can be successful. Assume that sensors sample a temporal process, e.g., temperature and transmit their values to an observer which then calculates the maximum of them. It is easy to see that the rate at which an observer can compute the maximum is the minimum of N sensor throughputs. This observation also holds for spatially distributed sensors. In general for a large class of tasks, the overall progress of the computation will be limited by the lowest sensor throughput in the network, hence, the problem we are interested in is

$$\max_{\underline{\alpha} \in [0,1]^N} \min_{1 \leq i \leq N} M_i(\underline{\alpha}) \tag{2}$$

Henceforth we will refer to the problem in (2) as MAXMIN and denote by $\underline{\alpha}^*$ the optimal MAXMIN transmission attempt probabilities (MMTAP). M^* denotes the optimal objective function value. When the sensors are collocated, for each i, $M_i(\underline{\alpha}) = \alpha_i \ \Pi_{j \neq i}(1-\alpha_j)$ and $\alpha_i^* = \frac{1}{N}$ which is an intuitive and desirable operating point for sensors in this scenario; note that as $N \to \infty$, $M^* \to \frac{1}{Ne}$ while the computing rate cannot exceed $\frac{1}{N}$. Even when sensors are spatially distributed, $\underline{\alpha}^*$ is *throughput equalising*, i.e., $M_i(\underline{\alpha}^*) = M_j(\underline{\alpha}^*),\ 1 \leq i,j \leq N$ ([8]). This property makes MMTAP particularly important; with MMTAP, sensors operate at equal processing rates which is desirable in applications where computations are iterative. It should be clear that $\underline{\alpha}^*$ depends on the sensor placement and topology, hence cannot be computed offline and preset in sensors because of the very nature of random deployment of sensor networks. Therefore, *it is essential that sensors adaptively learn the optimal attempt probabilities.* Investigation of such a scheme is the subject of this paper.

"Minimax" problems have been one of the motivations for non-differentiable optimisation theory ([13]). A generalised gradient method for MAXMIN was investigated in [8]. The other approaches for minimax are [6] (sequential quadratic programming), [5] (parametric embedding), [11] (conversion to smooth function) etc. These methods have good rates of convergence, however, as such they are not amenable to distributed implementation necessary for sensor networks. We, therefore, obtain a gradient-based iterative scheme which can be distributed.

This paper is organised as follows. In Section 2 we present an adaptive algorithm for the MAXMIN. Its distributed version is presented in Section 3. Implementation of the algorithm by considering a special functional form of the sensor throughputs is discussed in Section 4. In Section 5 we present stochastic algorithms in which the throughputs are optimised using measurements. In Section 6 we discuss the implications of our optimisation to global computing and conclude in Section 7.

2 An Adaptive Algorithm for MAXMIN

Note that the MAXMIN is equivalent to the following problem.

$$\begin{aligned} \max \quad & x \\ \text{subj. to} \quad & M_i(\underline{\alpha}) \geq x,\ i = 1,2,\ldots,N \\ & x \geq 0 \\ & \alpha_i \in [0,1],\ i = 1,2,\ldots,N \end{aligned} \tag{3}$$

Since $M_i(\underline{\alpha}) \leq 1$, $x \leq 1$. Let $\tilde{x} := (x, \underline{\alpha})$. Though the objective function is concave the constraints are not concave in $\tilde{x}$. As an illustration, let $M_1(\alpha_1,\alpha_2) = \alpha_1(1-\alpha_2)$. For $\tilde{x}_1 = (\frac{1}{2}-\epsilon, \frac{1}{2}+\epsilon, (\frac{1}{2}-\epsilon)^2)$ and $\tilde{x}_2 = (\frac{1}{2}+\epsilon, \frac{1}{2}-\epsilon, (\frac{1}{2}+\epsilon)^2)$, $M_1(\alpha_1,\alpha_2) - x = 0$ but for $\tilde{x}_3 = \frac{\tilde{x}_1+\tilde{x}_2}{2}$, $M_1(\alpha_1,\alpha_2) - x < 0$.

With $\underline{\mu}$ denoting the vector of dual variables corresponding to the constraints $M_i(\underline{\alpha}) \geq x$, let us consider the dual objective function.

$$\begin{aligned} L(\underline{\mu}) &= \sup_{\{1 \geq x \geq 0, 0 \leq \alpha_i \leq 1, 1 \leq i \leq N\}} \left(x + \sum_{i=1}^{N} \mu_i (M_i(\underline{\alpha}) - x) \right) \\ &= \sup_{\{1 \geq x \geq 0, 0 \leq \alpha_i \leq 1, 1 \leq i \leq N\}} \left(x(1 - \sum_{i=1}^{N} \mu_i) + \sum_{i=1}^{N} \mu_i M_i(\underline{\alpha}) \right) \end{aligned}$$

Note that there is one dual variable μ_i for each sensor i. Let us first assume that the sensors are collocated. Therefore $\sum_{i=1}^{N} M_i(\underline{\alpha}) \leq 1$. Let $\tilde{\mu} = \max\{\mu_i, 1 \leq i \leq N\}$. It follows that $\sum_{i=1}^{N} \mu_i M_i(\underline{\alpha}) \leq \tilde{\mu}$. The bound is achieved by choosing $\alpha_i = 1$ if $\mu_i = \tilde{\mu}$ (if there are multiple such nodes, one is chosen arbitrarily) and $\alpha_j = 0,\ j \neq i$. Thus, we find that

$$L(\underline{\mu}) = \begin{cases} 1 - \sum_{i=1}^{N} \mu_i + \max_{1 \leq i \leq N} \mu_i & \sum_{i=1}^{N} \mu_i < 1 \\ \max_{1 \leq i \leq N} \mu_i & \sum_{i=1}^{N} \mu_i \geq 1 \end{cases}$$

and $\inf L(\underline{\mu}) = \frac{1}{N}$. The optimal dual value $(\frac{1}{N})$ is only a loose bound on the MAXMIN throughput which was shown to be $\frac{1}{N}\left(1 - \frac{1}{N}\right)^{(N-1)}$. Therefore, there is a duality gap. This observation can be generalised to spatially distributed sensors. In general, (3) even lacks local concavity structure. An important implication of this is that a primal-dual algorithm will not work for our problem. Recall that a primal-dual algorithm for an equality constrained problem $\max_{\{h(\underline{x})=0, \underline{x} \in R^n\}} f(\underline{x})$ consists of sequential maximisations of the form $\max_{\underline{x} \in R^n} L(\underline{x}, \underline{\lambda}(k))$ where $L(\underline{x}, \underline{\lambda}(k)) = f(\underline{x}) + \underline{\lambda}(k)^T h(\underline{x})$ yielding vectors $\underline{x}(k)$ followed by updates of the form $\lambda_i(k+1) = \lambda_i(k) - a(k) h_i(\underline{x}(k))$ where $a(k)$ is the step size parameter.

The approach, therefore, is to *convexify* the problem by adding to the cost function a penalty term that prescribes a high cost to infeasible points ([1]). For sufficiently large penalty parameter, the convexified problem has locally convex structure, hence primal-dual algorithms can be applied to it. In the augmented Lagrangian method the penalty term is the square of the norm of constraint functions. We apply this technique to the MAXMIN problem. We follow the development in [2] where the problem is stated with equality constraints. Hence, we introduce slack variables $y_i\ i = 1, 2, \ldots, N$ to convert each inequality constraint $(M_i(\underline{\alpha}) - x \geq 0)$ into an equality constraint. Thus, the convexified objective function of the MAXMIN is

$$x - \frac{\sigma}{2} \| M(\underline{\alpha}) - x\underline{1} - \underline{y} \|^2$$

where, $\underline{1}$ denotes a vector of 1's. σ is the penalty parameter. If $\sigma > \bar{\sigma}$, where $\bar{\sigma}$ depends on the objective and constraint functions, then a primal-dual algorithm applied to the convexified problem will yield the optimal solution of the original problem, namely MMTAP, provided the starting point is sufficiently close to the optimal point ([2]). Since calculating $\bar{\sigma}$ is a difficult task in general, it is increased from iteration to iteration so that ultimately it exceeds $\bar{\sigma}$. By increasing σ the algorithm can also be made globally convergent ([2]). Recall that $\tilde{x} := (x, \underline{\alpha})$. The Lagrangian of the convexified problem is

$$L(\tilde{x}, \underline{\mu}, \underline{y}; \sigma) = x - \frac{\sigma}{2} \| M(\underline{\alpha}) - x\underline{1} - \underline{y} \|^2 + \sum_{i=1}^{N} \mu_i (M_i(\underline{\alpha}) - x - y_i) \tag{4}$$

Referring to the previous discussion, a primal-dual algorithm in this case will require maximisation of $L(\tilde{x}, \underline{\mu}, \underline{y}; \sigma)$ followed by an update of the multipliers. The augmented Lagrangian (4) can be first maximised in slack variables $\underline{y}$. It can be shown that, this yields $y_i = \max\left(0, -\frac{\mu_i}{\sigma} + M_i(\underline{\alpha}) - x\right)$. Thus, the slack variables can now be eliminated from (4) to yield

$$L(\tilde{x}, \underline{\mu}; \sigma) = x - \frac{1}{2\sigma} \sum_{i=1}^{N} (\max(\mu_i - \sigma(M_i(\underline{\alpha}) - x), 0))^2 + \frac{1}{2\sigma} \sum_{i=1}^{N} \mu_i^2 \tag{5}$$

Algorithm 2.1 is the required primal-dual algorithm, called the multiplier algorithm, for the convexified problem. $\mu_i(k+1)$ is updated essentially as $\mu_i(k) - a(k)(M_i(\underline{\alpha}) - x - y_i)$ with the step size equaling the penalty parameter ([2]). Note that $(M_i(\underline{\alpha}) - x - y_i)$ is the i^{th} constraint and y_i is as given above. $\sigma(k)$ is usually chosen as γ^k for some $\gamma > 1$. Algorithm 2.1 solves a sequence of concave maximisation problems (8). A gradient ascent method can be used for these maximisations using

$$\frac{\partial L(\tilde{x}, \underline{\mu}; \sigma)}{\partial x} = 1 - \sum_{i=1}^{N} \max(\mu_i - \sigma(M_i(\underline{\alpha}) - x), 0) \tag{6}$$

$$\frac{\partial L(\tilde{x}, \underline{\mu}; \sigma)}{\partial \alpha_j} = \sum_{i=1}^{N} \max(\mu_i - \sigma(M_i(\underline{\alpha}) - x), 0) \frac{\partial M_i(\underline{\alpha})}{\partial \alpha_j} \tag{7}$$

Convergence of Algorithm 2.1 is proved under regularity of $\tilde{x}^*$ and second order sufficiency of $(\tilde{x}^*, \underline{\mu}^*)$ ([3]).

Algorithm 2.1 A Multiplier Algorithm for MMTAP

$$\tilde{x}(k+1) = \arg\max_{\tilde{x}} L(\tilde{x}, \underline{\mu}(k); \sigma(k)), \ k \geq 0 \tag{8}$$

$$\underline{\mu}(k+1) = \max(\underline{\mu}(k) - \sigma(k)(M(\underline{\alpha}(k+1)) - x(k+1)\underline{1}), 0) \tag{9}$$

3 A Distributed Multiplier Algorithm

The constrained problem (3) does not have a structure suitable for distributed implementation in sensor networks. The idea is to modify (3) by introducing dummy variables, u_i's, at each sensor i as a local copy of "x" and then equalising them by additional equality constraints, i.e.,

$$\begin{aligned} \max \quad & x \\ \text{subj.to} \quad & M_i(\underline{\alpha}) \geq u_i, \ i = 1, 2, \ldots, N \\ & u_i = x, \ i = 1, 2, \ldots, N \\ & x, \alpha_i, u_i \in [0, 1], \ i = 1, 2, \ldots, N \end{aligned} \tag{10}$$

(3) and (10) are equivalent in a sense that optimal x and $u_i, \ i = 1, 2, \ldots, N$ for this problem equal x^*, and optimal $\underline{\alpha}$ is MMTAP. Let $\tilde{u} := (x, u_1, \ldots, u_N, \alpha_1, \ldots, \alpha_N)$. Following a procedure identical to the one that yielded (5) from (4) the augmented Lagrangian for (10) is

$$L(\tilde{u}, \underline{\mu}, \underline{\lambda}; \sigma) = x - \frac{1}{2\sigma}\sum_{i=1}^{N}(\max(\mu_i - \sigma(M_i(\underline{\alpha}) - u_i), 0))^2 + \frac{1}{2\sigma}\sum_{i=1}^{N}\mu_i^2 + \sum_{i=1}^{N}\lambda_i(x - u_i) - \frac{\sigma}{2}\sum_{i=1}^{N}(x - u_i)^2$$

where, $\underline{\lambda}$ is the vector of dual variables corresponding to the constraints $u_i = x$. It is now straightforward to show that for (10) an iteration equivalent to (8) can be obtained by Gauss-Seidel iterations. In Gauss-Seidel algorithm, the maximisations are carried out successively for each component. Let m index the subiteration for obtaining the maximum in (8). With the multipliers fixed at $\underline{\lambda}(k)$ and $\underline{\mu}(k)$, and the penalty parameter fixed at $\sigma(k)$, we now display the Gauss-Seidel update at the $(m+1)^{th}$ subiteration for the maximisations in (8) applied to (10).

$$x(m+1) = \frac{1 + \sum_{i=1}^{N}\lambda_i(k)}{N\sigma(k)} + \frac{1}{N}\sum_{i=1}^{N}u_i(m) \tag{11}$$

$$u_i(m+1) = \frac{1}{2}\left(x(m+1) - \frac{\lambda_i(k)}{\sigma(k)}\right) + \frac{1}{2}\min\left(x(m+1) - \frac{\lambda_i(k)}{\sigma(k)}, M_i(\underline{\alpha}(m)) - \frac{\mu_i(k)}{\sigma(k)}\right) \tag{12}$$

$$\alpha(m+1) = \arg\max_{\underline{\alpha}} L(x(m+1), \underline{\alpha}(m), \underline{u}(m+1), \underline{\mu}(k), \underline{\lambda}(k); \sigma(k)) \tag{13}$$

Informally (11)-(13) can be explained as following. x represents the "global notion of throughput" and u_i's the "local target throughputs". The local targets are "synchronised" via (11); note that for large values of $\sigma(k)$, x is updated as the average of the local target throughputs in the previous iteration; note that $u_i^* = M_i(\underline{\alpha}^*)$ and $x^* = \frac{1}{N}\sum_{i=1}^{N}M_i(\underline{\alpha}^*)$ due to throughput equality ([8]). For this to happen, sensors send their respective λ_i's and u_i's to the "root" which updates x. Note that, each λ_i and u_i need not be sent separately to the root; sensors can directly yield the sum and the average (see (11)) by a simple distributed algorithm ([8]). The root then distributes the updated value of x to all the sensors. Once x is received, (12) gives the new local target to which sensors tune their α_i's using (13) in parallel; (13) can be implemented using a gradient ascent method. (11)-(13) are repeated until convergence to a maximum of the augmented Lagrangian where upon each sensor updates μ_i's and λ_i's as:

$$\mu_i(k+1) = \max(\mu_i(k) - \sigma(k)(M_i(\underline{\alpha}(k+1)) - x(k+1)), 0) \tag{14}$$

$$\lambda_i(k+1) = \lambda_i(k) - \sigma(k)(u_i(k+1) - x(k+1)) \tag{15}$$

Practically, (11)-(13) will be repeated only a finite number of times; convergence is guaranteed even with inexact maximisation provided the error in maximisation goes to zero with iterations ([2]). A particularly efficient way is to iterate (11)-(13) just once (see alternating directions method ([4])).

Algorithm 2.1 and its distributed version as discussed above not only need values of $M_i(.)$ at specific values of $\underline{\alpha}$ but also their gradients: see (7) and (13). We discuss two mechanisms.

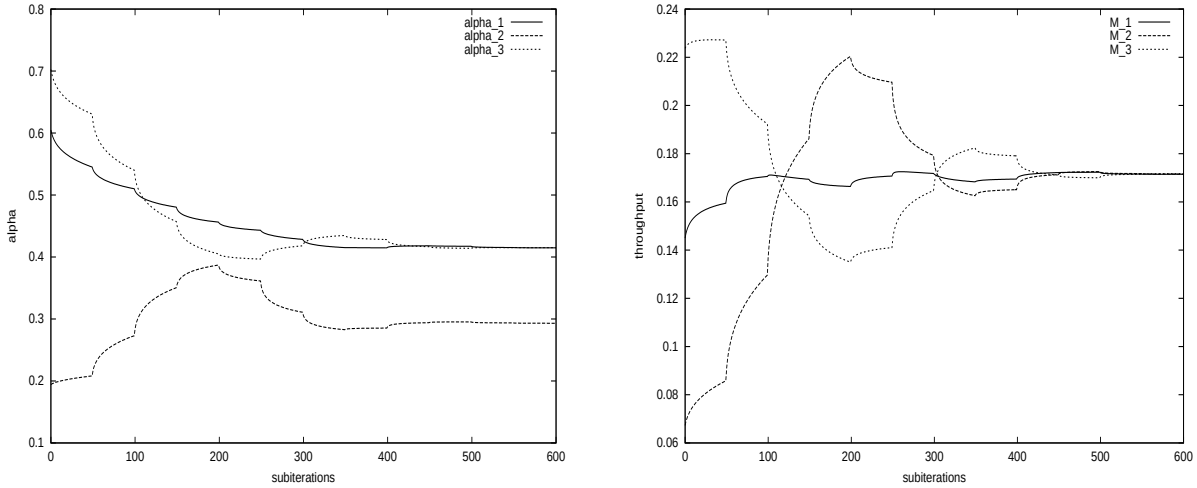

Figure 1. Evolution of $\underline{\alpha}$ and throughputs of 3 node sensor network for initial conditions $\underline{\alpha}_0 = (0.61, 0.19, 0.71)$ and $\underline{\mu}_0 = (0.3, 0.3, 0.3)$. $\underline{\alpha}^* = (0.41, 0.29, 0.41)$ and $M(\underline{\alpha}^*) = 0.1715$.

4 Deterministic Approach: $M_i(.)$ known

The first approach is to assume that for a successful reception at a sensor in a receive mode only its neighbours be silent; in other words function $g_{ij}(\underline{\alpha}^{ij})$ in (1) is of the form $\Pi_{k \in N_j}(1-\alpha_k)$. Then sensors can construct functional forms of $M_i(.)$ by exchanging neighbour lists and use the iterative scheme discussed above. As an example, consider a simple sensor network of 3 sensors such that $N_1 = \{2\}$, $N_2 = \{1, 3\}$, $N_3 = \{2\}$. β and η are such that,

$$
\begin{aligned}
M_1(\underline{\alpha}) &= \alpha_1(1-\alpha_2)(1-\alpha_3) \\
M_2(\underline{\alpha}) &= \alpha_2 \frac{(1-\alpha_1)+(1-\alpha_3)}{2} \\
M_3(\underline{\alpha}) &= \alpha_3(1-\alpha_2)(1-\alpha_1)
\end{aligned}
$$

$\sum_{i=1}^{N} \mu_i \frac{\partial M_i(\underline{\alpha})}{\partial \alpha_j} = 0, 1 \leq j \leq 3$ yields

$$
\begin{aligned}
\mu_1(1-\alpha_2)(1-\alpha_3) - \mu_2\frac{\alpha_2}{2} - \mu_3\alpha_3(1-\alpha_2) &= 0 \\
-\mu_1\alpha_1(1-\alpha_3) + \mu_2\frac{2-\alpha_1-\alpha_3}{2} - \mu_3\alpha_3(1-\alpha_1) &= 0 \\
-\mu_1\alpha_1(1-\alpha_2) - \mu_2\frac{\alpha_2}{2} + \mu_3(1-\alpha_1)(1-\alpha_2) &= 0
\end{aligned}
$$

By symmetry $\alpha_1^* = \alpha_3^*$ and $\mu_1^* = \mu_3^*$. Then the set of equations yields, $\mu_1^* = \mu_3^* = \frac{1}{2(1+\alpha_1^*)}$, $\mu_2 = \frac{\alpha_1^*}{1+\alpha_1^*}$, $\alpha_2^* = \frac{1-2\alpha_1^*}{1-\alpha_1^*}$ and $\alpha_1^* = \sqrt{2}-1$. Thus, $\underline{\alpha}^* = (\sqrt{2}-1, 1-\frac{1}{\sqrt{2}}, \sqrt{2}-1)$, $\underline{\mu}^* = (\frac{1}{2\sqrt{2}}, 1-\frac{1}{\sqrt{2}}, \frac{1}{2\sqrt{2}})$, and $M(\underline{\alpha}^*) = (\sqrt{2}-1)^2 \approx 0.1715$. $\tilde{x}^*$ can be shown to be regular. Further, $\nabla^2_{\tilde{x}\tilde{x}} L(\tilde{x}^*, \underline{\mu}^*)$ is negative definite which in view of strict complementarity ($\mu_i > 0$, $i = 1, 2, 3$) shows second order sufficiency. Numerical results are shown in Figure 1. $\sigma(k) = \gamma^k$ with $\gamma = 1.2$. We use decreasing step sizes, $a(m) = \frac{0.1}{(m+1)^{0.7}}$, in the gradient ascent method in (8). Since checking whether the gradient of Lagrangian is near zero is infeasible in a sensor network, we use a fixed number, 50, of subiterations in (8). Abscissa is the number of subiterations, thus convergence is achieved in only 12 "iterations". Observe that, the sensor throughputs equalise.

5 Stochastic Approach: Estimation of $M_i(.)$

Our previous assumption regarding $g_{ij}(.)$, though prevalent in ad hoc network literature, does not hold in general. Moreover, the particular form of $M_i(.)$ in (1) is the result of our modelling assumptions (e.g.,

only path loss, every sensor always has a packet to transmit etc.); in many cases only an approximate analytical form is known for node throughputs ([15]). Therefore, an approach which allows general forms of $g_{ij}(.)$ and more importantly not dependent on an analytical form is to let *sensors optimise their throughputs based on the measurements of their throughputs* ([9]).

Being a steady-state average, only *noisy measurements* of $M_i(.)$ are available. 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 $M_i(.)$. It can be shown that infinitesimal perturbation analysis (IPA) and likelihood ratio-score function (LR-SF) methods of gradient estimation ([10]) are not applicable to our problem ([7]). An appropriate method for gradient estimation in a distributed algorithm is simultaneous perturbation (SP, [14]) since sensors can simultaneously estimate the derivatives by choosing the perturbation amount locally.

Currently there exists no general convergence result for the multiplier algorithm in a stochastic setting ([16]). We will, therefore, consider two versions of the quadratic penalty method ([2]). In algorithm 1, update (8) remains unchanged except values are now replaced by their estimates, however, the multiplier $\underline{\mu}(k) = 0,\ k \geq 1$[4]. In Algorithm 2, $\underline{\mu}(k) = 0,\ k \geq 1$ and (8) is replaced by the following iteration,

$$\tilde{x}(k+1) = \tilde{x}(k) + a(k)\tilde{\nabla}L(\tilde{x}(k), \underline{\mu}(k); \sigma(k))$$

$\tilde{\nabla}L(.)$ denotes the gradient estimate. Similarly for the distributed version. We do not establish the convergence of stochastic algorithms in this paper. Instead we will consider a realistic network of 100 sensors to look into the performance of our stochastic algorithms; the optimal throughput and attempt probabilities are actually unknown for this network. We assume that sensors, each of transmission range $4m$, are deployed randomly in a square field of area $400m^2$. Figure 2 shows their transmission graph[5] and a computationally optimal network topology ([8]); scale in Figure 2 is distance in m. We assume $\eta = 4$ and $\beta = 7$dB. Throughput and gradient estimation is done over 1000 slots. We choose gain sequence

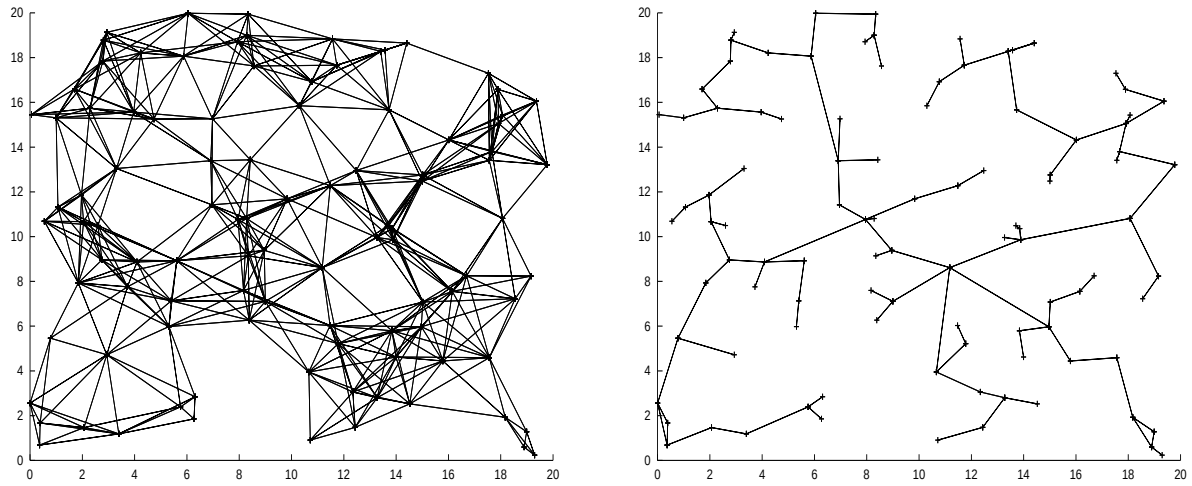

Figure 2. Transmission graph and computing topology of a random 100 node sensor network.

$a(k) = \frac{0.1}{(k+1)^{0.7}}$ and perturbation sequence $b(k) = \frac{0.1}{(k+1)^{0.15}}$. $\sigma(k) = \gamma^k$; $\gamma = 1.8$ and 1.003 for Algorithm 1 and 2 respectively. The number of subiterations of (8) in Algorithm 1 is 200. $\alpha_i(0) = 0.1$, $1 \leq i \leq 100$. Figure 3 shows how $\min_{1 \leq i \leq N} M_i(.)$ improves with iterations; the actual estimates of sensor throughputs are used and the resulting graph is smoothed by 5-point adjacent averaging to show the trend. The algorithms appear to have reached a throughput value of $0.032 - 0.035$ packets/slot starting from 0.01 packets/slot.

[4] actually any bounded sequence of $\underline{\mu}(k),\ k \geq 1$ is sufficient for convergence

[5] a transmission graph is a graph with V_s, the set of sensors, as the vertex set and an edge (i, j) is in the edge set if sensors i and j are within each other's range

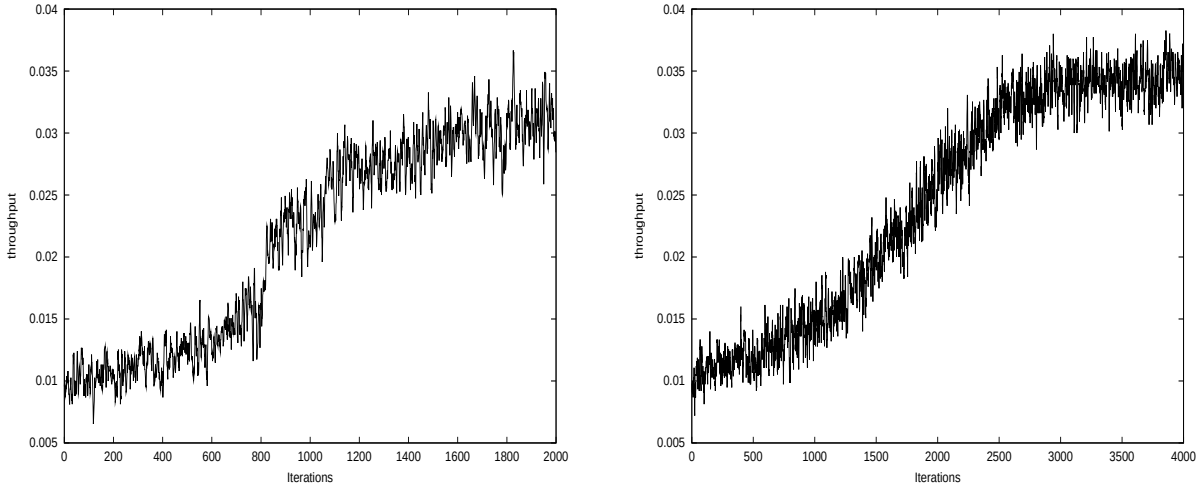

Figure 3. Evolution of the *estimated* minimum sensor throughput for stochastic Algorithm 1 (left) and Algorithm 2 (right).

6 Discussion

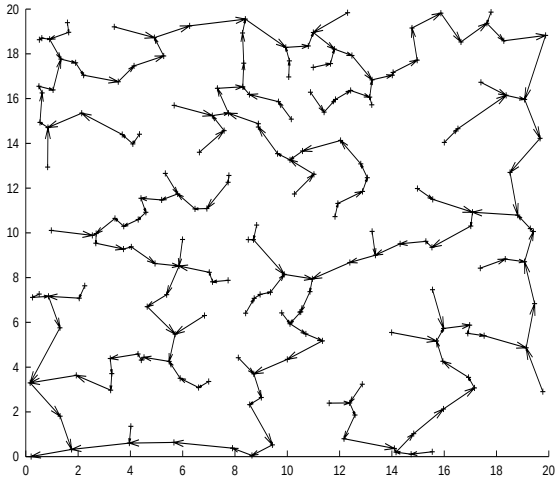

Figure 4. An optimal tree of 200 sensors for maximum computation and delivery to the observer at the origin.

Figure 3 shows that for the specified initial condition, the algorithms lead to an improvement of more than 100% in the minimum sensor throughput. An important question now is *how much improvement in the computing performance does this optimisation lead to?* To answer this consider a scenario where each sensor measures the local value of some environmental variable say temperature. An observer, placed at the origin, is interested in tracking the time-varying maximum temperature. The task of 200 sensors deployed randomly in a square $20m \times 20m$ is to deliver the maximum temperature values to the observer. Assume that sensors sample the temperature at a given rate synchronously. Instead of each sensor sending its value to the observer, sensors can compute the maximum in a distributed fashion. They do this by first constructing a tree optimised for throughput (see Figure 4, [8]) and then following a simple computing algorithm: each sensor receives values forwarded by all its children, compares them with its own and only forwards the maximum to its parent. The observer is guaranteed to receive the correct maximum values. This "aggregation" algorithm can be applied to many other functions such as average, sum etc. Thus it will be clear that the following discussion is not specific to "maximum computation" but holds for a variety of sensor tasks of practical interest.

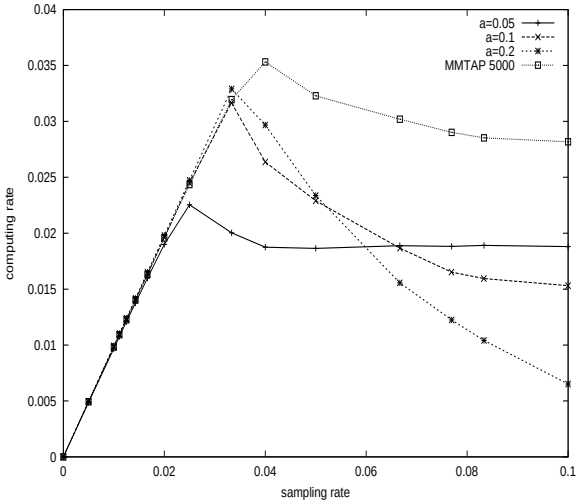

Figure 5. Computing rate (maximum calculations per slot) vs sampling rate. MMTAP is tuned for each value of sampling rate.

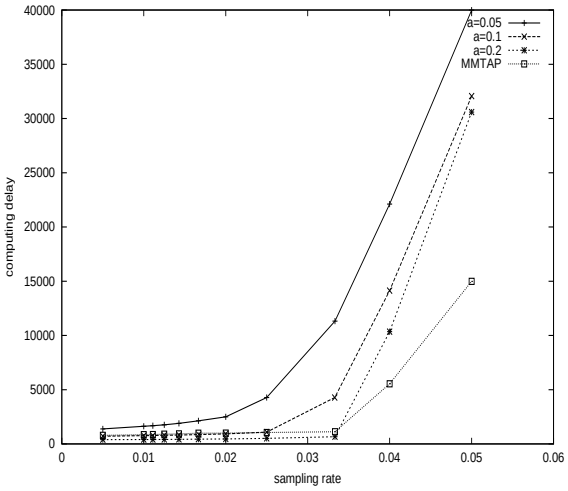

Figure 6. Computing delay (average calculation time) vs sampling rate. MMTAP is tuned for each value of sampling rate.

This computing network can be seen as a queueing system with the sensor network as its "complex server". The set of samples taken simultaneously at each sensor acts as a "customer" which requires a "service time" equal to the time required for the calculation of the maximum over this set by the sensor network. The computing rate is the rate at which the maximum is calculated by the network. Thus, the system is stable only if the rate at which sensors sample the temperature is less than the computing rate. The computing rate being a complex function of the attempt probabilities, a "good value of $\underline{\alpha}$" for a given sampling rate cannot be known a priori. Figure 5, which shows the variation of the computing rate with the sampling rate for various values of $\underline{\alpha}$, illustrates this point. Observe that when $\alpha_i = 0.05$ for each i, sampling rates only less than 0.02 samples per slot can be handled. When $\alpha_i = 0.1$ for each i this threshold is around 0.025 samples per slot whereas for $\alpha_i = 0.2$ for each i, it is about 0.033 samples per slot. Thereafter the system becomes unstable; see Figure 6 which shows that the computing delay, i.e., the time required for maximum calculation, increases rapidly with the sampling rate beyond this threshold. Further, at low sampling rates, such as 0.01 samples per slot, the computing delay incurred by $\alpha_i = 0.2$ is less than that obtained by $\alpha_i = 0.05$ and is thus preferable.

It is, therefore, necessary that sensors adapt their attempt probabilities for a given sampling rate. MMTAP is important in this scenario because the global maximum computation is limited by the lowest sensor throughput in the network. The performance of MMTAP shown in Figure 5 was obtained by *tuning attempt probabilities on-line for every value of the sampling rate*, i.e. while sensors are computing. This is possible because throughput measurements do not need special packets. Observe from Figure 5 that with MMTAP, sensors are able to adapt to sampling rates. Moreover, MMTAP leads to higher sampling rates as compared to the preset values of $\underline{\alpha}$ without compromising the delay performance. This is important also from the point of view of task reprogramming, i.e., sensor network may be reprogrammed in the field to sample at a different rate than the one decided at the time of deployment. Recall that for the analysis we had assumed that each sensor always has a packet to transmit. This happens at large sampling rates when the network is almost saturated. Figure 5 then also shows that under this assumption the computing performance of the sensors tuned to MMTAP is substantially better than every value of $\underline{\alpha}$ considered for comparison.

Thus our results not only show that MMTAP adapts and improves the computing performance in scenarios such as discussed above but also that with our algorithms sensors are able to tune to MMTAP. However, these algorithms involve two network-wide estimation phases and one synchronisation phase (update (11)) at the root node per iteration and in a real scenario may take much longer time to tune to the optimal values. Stochastic algorithms are constrained by the 'bias-variance dilemma'. Their convergence may be improved by appropriate selection of parameters. Secondly, optimality can be traded for acceptable improvement. Most importantly, such an algorithm can work in the background and can be seen as a tool by which the network continuously keeps on improving itself. An important advantage of

stochastic algorithms is that 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. However, communication required for (11) and (7) is substantial, limiting the extent to which this approach can be distributed. This is the cost of optimising a coupled performance measure. We are currently investigating ways by which a sensor can obtain an estimate of the global performance locally. In some cases, slot synchronisation could be achieved since for some sensing tasks, time synchronization is vital ([12]. In such cases, an approach discussed in Section 4 may be practically useful.

7 Conclusion

In this paper, we proposed an iterative scheme for sensors to adaptively learn the channel attempt probabilities which maximise the minimum throughput in the network in a distributed fashion. We showed that with such a scheme a sensor network can optimise itself for given task objectives. We designed algorithms to achieve the optimal performance and found correspondingly higher communication 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.

8 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.

References

1. D. Bertsekas. Convexification Procedures and Decomposition Algorithms for Large-Scale Nonconvex Optimization Problems. *Journal of Optimization Theory and Applications*, 29:169–197, 1979.
2. D. Bertsekas. *Constrained Optimization and Multiplier Methods.* Academic Press, 1982.
3. D. Bertsekas. *Nonlinear Programming.* Athena Scientific, 1995.
4. D. Bertsekas and J. Tsitsiklis. *Parallel and Distributed Computation.* Prentice Hall, 1989.
5. F. Guerra and G. Lopez. A Parametric Embedding for the Finite Minimax Problem. *Mathematical Methods of Oper. Res.*, 49:359–371, 1999.
6. S. Han. Variable Metric Methods for Minimizing a Class of Nondifferentiable Functions. *Mathematical Programming*, 20:1–13, 1981.
7. A. Karnik. *Performance and Optimal Self-organisation of Wireless Sensor Networks.* PhD thesis, Indian Institute of Science, 2004. under preparation.
8. A. Karnik and A. Kumar. Distributed Optimal Self-Organisation in a Class of Wireless Sensor Networks. In *IEEE INFOCOM*, 2004.
9. H. Kushner and D. S. Clark. *Stochastic Approximation Methods for Constrained and Unconstrained Systems.* Springer-Verlag, 1978.
10. P. L'Ecuyer. An Overview of Derivative Estimation. In *Conf. on Winter Simulation*, 1991.
11. G. Di Pillo, L. Grippo, and S. Lucidi. A Smooth Method for the Finite Minimax Problem. *Mathematical Programming*, 60:187–214, 1991.
12. K. Romer. Time Synchronization in Ad Hoc Networks. In *MobiHoc*, 2001.
13. N. Shor. *Minimization Methods for Nondifferentiable Functions.* Springer, 1985.
14. J. Spall. Multivariate Stochastic Approximation Using a Simultaneous Perturbation Gradient Approximation. *IEEE Trans. on Automatic Control*, 37(3):332–341, March 1992.
15. F. Tobagi. Modelling and Performance Analysis of Multihop Packet Radio Networks. *Proc. of the IEEE*, 75(1):135–154, January 1987.
16. I. Wang and J. Spall. A Constrained Simultaneous Perturbation Stochastic Approximation Algorithm Based on Penalty Functions. In *American Control conf.*, 1999.