Distributed Self-Tuning of Sensor Networks

Aditya Karnik, Anurag Kumar
Dept. of Electrical Communication Engineering,
Indian Institute of Science, Bangalore-560012, India
{karnik,anurag}@ece.iisc.ernet.in
and
Vivek Borkar
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. Arguing that communication is the main bottleneck for distributed computation in a sensor network we formulate two approaches for optimisation of computing rates. The first is a team problem for maximising the minimum communication throughput of sensors and the second is a game problem in which cost for each sensor is a measure of its communication time with its neighbours. We investigate adaptive algorithms using which sensors can tune to the optimal channel attempt rates in a distributed fashion. For the team problem, the adaptive scheme is a stochastic gradient algorithm derived from the augmented Lagrangian formulation of the optimisation problem. The game formulation not only leads to an explicit characterisation of the Nash equilibrium but also to a simple iterative scheme by which sensors can learn the equilibrium attempt probabilities using only the estimates of transmission and reception times from their local measurements. Our approach is promising and should be seen as a step towards developing optimally self-organising architectures for sensor networks.

I. 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 will in fact be essential for sensor based control applications. Starting from fairly simple, nevertheless essential, tasks such as delivering the maximum of the sensed values of say, temperature, to the observer, data processing in a sensor network will include data combining, spatial estimation, etc. For example, each sensor can estimate the maximum in its neighbourhood from its own and its

neighbours' measurements and then deliver the maximum value of the temperature. This would not only facilitate overall task objectives but also save energy by avoiding long haul transport of sensed data to the observer. However, the practicality of such an implementation 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.

However, finding a functional form of the computation rate of a given task is a formidable task since it depends on, among other things, the way the computation is "arranged" using a distributed algorithm. In many cases such a form may not even exist. Even if such a form is known optimising it would be a centralised problem since individual sensors will hardly have any notion of the global computing rate. Therefore, we can base our optimisation problem only on objectives local to the sensors. Note that, a global computation proceeds in steps comprising of certain local computations at each sensor. Thus, optimisation of the local computation rate at each sensor will lead to a higher global computation rate. We develop two approaches with this insight.

- The first approach is to note that 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. In fact we argue that the overall progress of the global computation will be limited by the minimum throughput in the network hence an appropriate optimisation problem is to maximise the minimum throughput of sensors.
- The second approach is based on an observation that a computation at a sensor will span over a number of slots during which the exchange of data will take place as a sequence of local measurements, receptions from the other sensors and transmissions for conveying local data or partial results to the other sensors. The actual sequence will be determined by the way the computation is organised, however, communication being the main bottleneck the local computation time will be dominated by the time taken for receptions as well as transmissions. Thus if the communication time is optimised at each sensor, the local com-

putations will proceed rapidly and so will the global computation. We, therefore, propose to minimise at each sensor a measure of the average communication time with the neighbours.

Since the transmissions (or channel access) are not centrally co-ordinated in sensor networks, in view of the wireless medium, the transmission attempt rate (or channel access rate) is the most crucial parameter in determining throughput as well as average communication time of sensors[1]. A small value of attempt rate at each sensor means that opportunities of successful transmissions are often lost and a high value means numerous simultaneous transmissions leading to high interference and therefore transmission failures. Hence in both the approaches, the transmission attempt rate will be the optimisation parameter. We now proceed as follows. We formulate maximisation of the minimum throughput of sensors as a team problem and minimisation of communication time as a game problem in which cost for each sensor is a measure of its communication time with its neighbours. It should be clear that the optimal attempt rate 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. We, therefore, investigate adaptive algorithms using which sensors can "tune" to the optimal transmission attempt rates in a distributed fashion. For the team problem, the adaptive scheme is a stochastic gradient algorithm derived from the augmented Lagrangian formulation of the optimisation problem. The game formulation not only leads to an explicit characterisation of the Nash equilibrium but also to a simple iterative scheme by which sensors can learn the equilibrium attempt probabilities using only the estimates of transmission and reception times from their local measurements. Our algorithms are promising and can be seen as a step towards developing optimally self-organising architectures for sensor networks.

This paper is organised as follows. In Section II we discuss a generic model of sensor networks. Section III presents the team problem and Section IV the game problem. In Section III-E and Section IV-C we discuss the implications of our optimisation approaches to global computing. The paper concludes with Section V.

[1]With fairly simple hardware, sensors may not have sophisticated power control. It is, therefore, reasonable to assume that transmission powers are fixed and will only play a static role in throughput performance of sensors. The transmission attempt rate, on the other hand, can be easily modified.

II. Model of a Sensor Network

Consider a sensor network comprising N nodes, indexed by $i \in \{1, 2, \cdots, N\}$; we denote this set by V_s. We assume that these sensors are engaged in a generic task which involves continuous sensing and computation, for example, a typical task which requires sensors deployed in a geographical region to monitor a spatio-temporal process, say, contamination in an environment. This also models processing over long, possibly critical, activity periods ([23]). Therefore, it is assumed that the sensors are always active. However, it will be clear that our approach and the algorithms can be extended to the energy saving techniques such as random sleep times. A local algorithm running on each sensor uses local measurements and updates from the other sensors to perform certain computations; for example, calculating the "maximum" of the sampled values as discussed in Section III-E and Section IV-C. A sensor, therefore, communicates with certain other sensors within its transmission range (denoted by R_0) designated as its neighbours; neighbours of each sensor are specified by the network topology. The problem of learning a good topology has been addressed in [13]. We, therefore, assume that the topology is already learnt, and a sensor knows which other sensors are its neighbours. N_i (resp. n_i) denotes the set (resp. number) of neighbours of sensor i. Time is slotted, and each packet occupies one slot. We consider a random access model for sensor transmissions: sensor i attempts a transmission in any slot with probability α_i independent of everything else; α_i is called the *attempt probability* of sensor i. $\underline{\alpha}$ denotes the vector of attempt probabilities. This simple attempt model is motivated by noting that sensors may not need elaborate multiple access schemes 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. Time synchronization is vital for some sensing tasks ([18], [8]); hence our slotted time assumption is not very restrictive. The raw measurements and/or computational results to be sent to the neighbours will be queued up in a packet queue. We assume that if a sensor decides to transmit, it addresses the transmission of the head-of-the-line packet in its queue to one of its neighbours randomly[2]; this can be attributed to isotropy of the physical process and uniformity of processing. All sensors transmit on a common carrier frequency using the same transmit power and omni-directional antennas. A transmission from sensor i to j is said to be successful if j is in the receive mode and the signal-to-interference ratio at j is above a given threshold β; the signal is assumed to undergo

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

only path loss with exponent η[3]. A detailed discussion of the model can be found in [13] and references therein.

III. A TEAM FORMULATION

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

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

where, $M_i(\underline{\alpha})$ denotes the throughput of sensor i under the attempt probabilities $\underline{\alpha}$. Henceforth we will refer to the problem in (1) as MAXMIN and denote by $\underline{\alpha}^*$ the optimal MAXMIN transmission attempt probabilities (MMTAP). M^* denotes the optimal objective function value.

Let $\underline{\alpha}^{ij}$ denote 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, or in other words the network is saturated, for a given topology, and given values of η and β ([12]),

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

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 $g_{ij}(.)$. Recall that in our model, in transmit mode a sensor transmits a packet to one of its neighbours with probability $\frac{1}{n_i}$.

When the sensors are collocated, it follows from (2) that, 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., $0 < \underline{\alpha}^* < 1$ implies $M_i(\underline{\alpha}^*) = M_j(\underline{\alpha}^*),\ 1 \le i, j \le$

[3]Since our algorithms are measurement based, these assumptions are not crucial.

N ([12]). This property makes MMTAP particularly important; with MMTAP, sensors operate at equal processing rates which is desirable in applications where computations are iterative.

"Minimax" problems have been one of the motivations for non-differentiable optimisation theory ([20]). A generalised gradient method for MAXMIN was investigated in [13]. The other approaches for minimax are [10] (sequential quadratic programming), [9] (parametric embedding), [17] (conversion to smooth function) etc. Although these methods have good rates of convergence, 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.

A. An Adaptive Algorithm for MAXMIN

Note that the MAXMIN is equivalent to the following problem.

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

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.

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

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

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

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 ([2]). 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 [3] 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 ([3]). 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 ([3]). 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 $\underline{y}$. 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 III.1 A Multiplier Algorithm for MMTAP

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

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

Algorithm III.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 ([3]). 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 $\gamma > 1$. The penalty term

$$-\frac{1}{2\sigma}(\max(\mu_i - \sigma(M_i(\underline{\alpha}) - x), 0))^2 + \mu_i^2$$

in (5) corresponding to the i^{th} constraint is continuously differentiable in $\tilde{x}$ since $M_i(\underline{\alpha}) - x$ is continuously differentiable in $\tilde{x}$ ([4]). Therefore, maximisation in (8) can be achieved using a gradient ascent method with

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

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

Convergence of Algorithm III.1 is proved under regularity of $\tilde{x}^*$ and second order sufficiency of $(\tilde{x}^*, \underline{\mu}^*)$ ([4]).

B. A Distributed Multiplier Algorithm

The constrained problem (3) does not have a structure suitable for distributed implementation in sensor networks. Our approach 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.,

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

(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

$$L(\tilde{u}, \underline{\mu}, \underline{\lambda}; \sigma) = \tag{11}$$
$$x - \frac{1}{2\sigma}\sum_{i=1}^{N}(\max(\mu_i - \sigma(M_i(\underline{\alpha}) - u_i), 0))^2 + \frac{1}{2\sigma}\sum_{i=1}^{N}\mu_i^2 + \sum_{i=1}^{N}\lambda_i(x - u_i) - \frac{\sigma}{2}\sum_{i=1}^{N}(x - u_i)^2$$

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. Let m index the sub-iteration 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 $(m+1)^{th}$ sub-iteration for the maximisations in (8) applied to (10). Recall that, in Gauss-Seidel algorithm, the maximisations are carried out successively for each component. Thus, differentiating (11) with respect to x and equating to zero, we get,

$$x(m+1) = \frac{1 + \sum_{i=1}^{N}\lambda_i(k)}{N\sigma(k)} + \frac{1}{N}\sum_{i=1}^{N}u_i(m) \tag{12}$$

Now using $x(m+1)$ obtained as above to maximise (11) with respect to u_i leads to the following update for $u_i(m+1),\ i = 1, 2, \ldots, N$.

$$u_i(m+1) = \frac{1}{2}\left(x(m+1) - \frac{\lambda_i(k)}{\sigma(k)}\right) + \frac{1}{2}\min\left(x(m+1) - \frac{\lambda_i(k)}{\sigma(k)}, M_i(\underline{\alpha}(m)) - \frac{\mu_i(k)}{\sigma(k)}\right) \tag{13}$$

Finally, the maximiser of (11) given $x(m+1)$ and $\underline{u}(m+1)$ is

$$\alpha(m+1) \ = \ \arg\ \max_{\underline{\alpha}}\ L(x(m+1), \underline{\alpha}(m), \underline{u}(m+1), \underline{\mu}(k), \underline{\lambda}(k); \sigma(k)) \tag{14}$$

It is essential to note that for large enough $\sigma(k)$ $L(\tilde{u}, \underline{\mu}, \underline{\lambda}; \sigma)$ has the required concave structure to justify this maximisation process ([3]).

Informally (12)-(14) can be explained as follows. x represents the "global notion of throughput" and u_i's the "local target throughputs". The local targets are "synchronised" via (12); 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 as discussed in Section I. 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 (12)) by a simple distributed algorithm implemented on the underlying MAWSS tree (see [12]). The root then distributes the updated value of x to all the sensors. Once x is received, (13) gives the new local target to which sensors tune their α_i's using (14) in parallel; (14) can be implemented using a gradient ascent method.

(12)-(14) are repeated until convergence to a maximum of the augmented Lagrangian where upon each sensor updates μ_i's and λ_i's as:

$$\mu_i(k+1) = \max(\mu_i(k) - \sigma(k)(M_i(\underline{\alpha}(k+1)) - x(k+1)), 0) \quad (15)$$

$$\lambda_i(k+1) = \lambda_i(k) - \sigma(k)(u_i(k+1) - x(k+1)) \quad (16)$$

Practically, (12)-(14) will be repeated only a finite number of times; convergence can be proved with inexact maximisation provided the error in maximisation goes to zero with iterations ([3]). A particularly efficient way is to iterate (12)-(14) just once (alternating directions method, [5]).

Algorithm III.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 (14). We discuss two mechanisms.

C. 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 (2) 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 $N_1 = \{2\}$, $N_2 = \{1, 3\}$, $N_3 = \{2\}$. (see Figure 1). β and η are such that, when 1

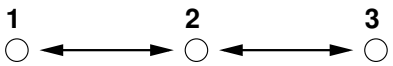


Fig. 1. An example network of 3 sensors.

transmits to 2, its transmission is successful only if 2 and 3 both do not transmit, similarly for 3. However, when 2 transmits to 1, only 1 needs to be receiving, a simultaneous transmission by 3 does not cause this transmission to fail. Similarly for transmission from 2 to 3. Thus,

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

KKT conditions ([4]) for (3) imply $\sum_{i=1}^{N}\mu_i\frac{\partial M_i(\underline{\alpha})}{\partial\alpha_j} = 0, 1 \le j \le 3$ which yields

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

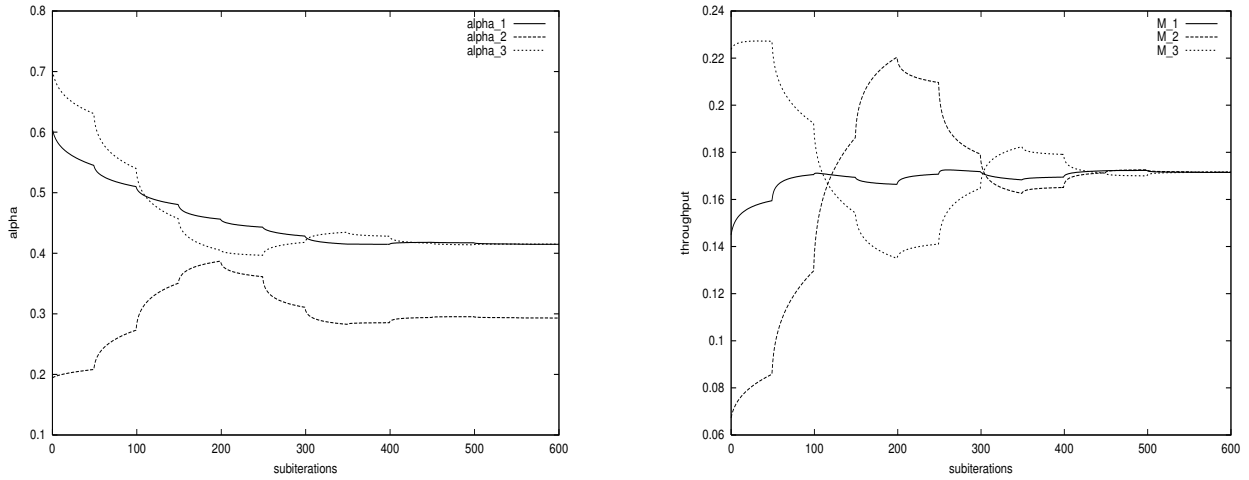


Fig. 2. Evolution of $\underline{\alpha}$ and throughputs of 3 node sensor network for initial conditions $\underline{\alpha}_0 = (0.614, 0.19, 0.714)$ and $\underline{\mu}_0 = (0.3, 0.3, 0.3)$. $\underline{\alpha}^* = (0.41, 0.29, 0.41)$ and $M(\underline{\alpha}^*) = 0.1715$.

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 2. $\sigma(k) = \gamma^k$ 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 sub-iterations in (8). Abscissa is the number of sub-iterations, thus convergence is achieved in only 12 "iterations". Observe that, the sensor throughputs equalise.

D. 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 (2) 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 ([24]). 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* ([14]).

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

Algorithm III.2 A Stochastic Quadratic Penalty Algorithm for MMTAP

$$x(k+1) = x(k) + a(k)\left(1 - \sum_{i=1}^{N} \max(\sigma(k)(x(k) - \hat{M}_i(\underline{\alpha}(k))), 0)\right) \quad (17)$$

$$\alpha_j(k+1) = \alpha_j(k) + a(k)\left(\sum_{i=1}^{N} \max(\sigma(k)(x(k) - \hat{M}_i(\underline{\alpha}(k))), 0)\frac{\partial \hat{M}_i(\underline{\alpha}(k))}{\partial \alpha_j}\right) \quad (18)$$

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 ([15]) are not applicable to our problem ([12]). An appropriate method for gradient estimation in a distributed algorithm is simultaneous perturbation (SP, [21]) since sensors can simultaneously estimate the derivatives by choosing the perturbation amount locally. In the k^{th} iteration, let $\underline{\Delta}(k)$ denote a vector of N independent Bernoulli random variables taking values in $\{-1, 1\}$ 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 $\frac{\partial M_i(\alpha)(k)}{\partial \alpha_j}$ 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.

Currently there exists no general convergence result for the multiplier algorithm III.1 in a stochastic setting ([25]). Unlike stochastic approximation algorithms in which the step size parameters is decreasing (under certain conditions), in this case, the step size $\sigma(k)$ increases with k. Therefore, we shall consider a quadratic penalty algorithm ([3]). In this algorithm, $\underline{\mu}(k) = 0,\ k \geq 1$ and (8) is replaced by the following iteration,

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

$\hat{\nabla} L(.)$ denotes the gradient estimate. Using (6) and (7), and gradient estimates as discussed above, Algorithm III.2 shows the stochastic quadratic penalty iterations. Thus, there is no nested maximisation and $\sigma(k)$ is increased in every iteration (see (6) and (7)). It is straightforward to apply this algorithm to the distributed case (10). A convergence results for the stochastic quadratic penalty algorithm as ours has been proved in [25] and [22]. The conditions are nontrivial to verify; we omit their discussion here.

As an example, consider a network of 4 sensors as shown in Figure 3. $\eta = 4, \beta = 2, R_0 = 1$m so that, for each i, $M_i(\underline{\alpha}) = \alpha_i(1 - \alpha_{i+1})(1 - \alpha_{i+2})$, addition in the subscript being modulo 4. It is easy to see that $\underline{\alpha}* = (1/3, 1/3, 1/3, 1/3)$ and $M^* = 0.148$. Figure 4 shows the variation

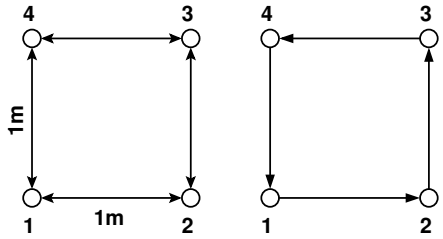


Fig. 3. 4 node network, $R_0 = 1$m, $\beta = 2$, $\eta = 4$; on left is G_{R_0} and on right is the operational topology.

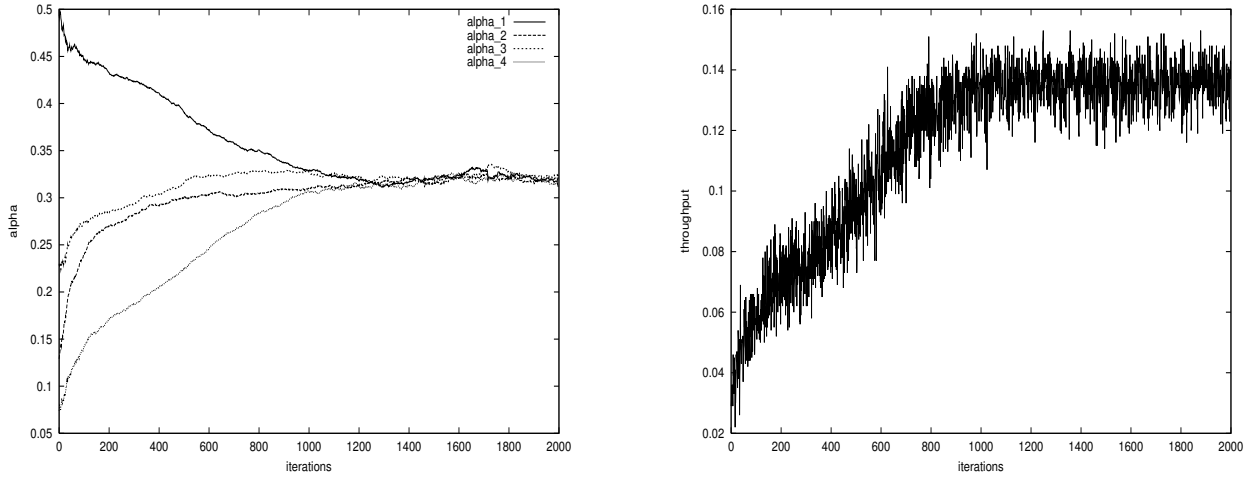


Fig. 4. Evolution of $\underline{\alpha}$ and $\min_i M_i(.)$ of 4 node sensor network for initial conditions $\underline{\alpha}_0 = (0.494, 0.129, 0.228, 0.074)$ and $\alpha_i^* = 1/3\, 1 \leq i \leq 4$ and $M(\underline{\alpha}^*) = 0.148$.

of α_is and $\min_i M_i(\underline{\alpha})$ with the number of iterations in Algorithm III.2. Each iteration consists of three estimation intervals (τ). We have used $\tau = 1000$ slots. $a(k) = \frac{0.1}{(k+1)^{0.65}}$, $c(k) = \frac{0.1}{(k+1)^{0.12}}$ and $\sigma(k) = \gamma^k$ with $\gamma = 1.003$. Observe that within few iterations, the improvement in the performance is substantial.

E. Discussion

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 5, [12])

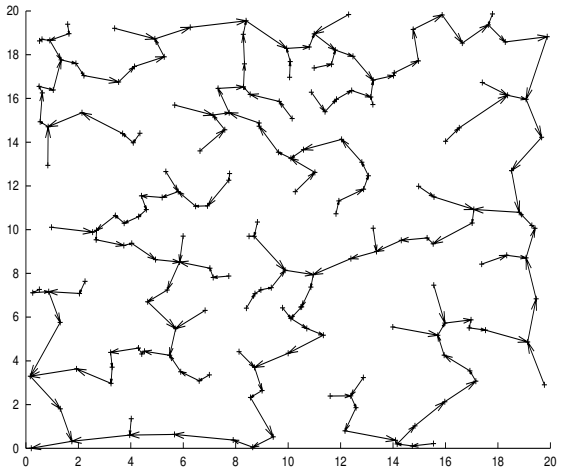


Fig. 5. An optimal tree of 200 sensors for maximum computation and delivery to the observer at the origin.

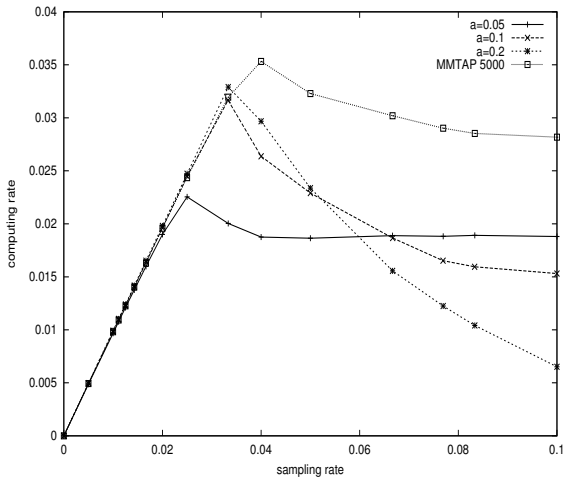


Fig. 6. Computing rate (maximum calculations per slot) vs sampling rate. MMTAP is tuned for each value of sampling rate.

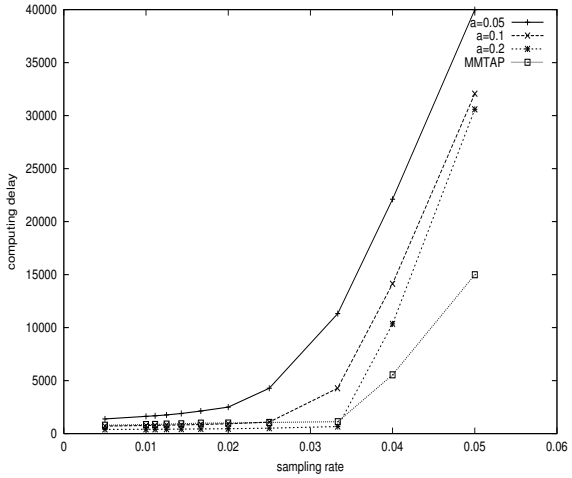


Fig. 7. Computing delay (average calculation time in slots) vs sampling rate. MMTAP is tuned for each value of sampling rate.

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.

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 6, 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 7 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 6 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 6 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 6 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 (12)) 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' ([6]). 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. In some cases, slot synchronisation could be achieved since for some sensing tasks, time synchronization is vital ([18]. In such cases, an approach discussed in Section III-C may be practically useful. However, communication required for (12) and (7) is substantial, limiting the extent to which this approach can be distributed. This cost of optimising a coupled performance measure motivates us to consider a game formulation whereby an adaptive algorithm requiring strictly local measurement can be derived.

IV. A GAME FORMULATION

A sensor network derives its utility from collaborative processing by sensors in contrast with traditional networks where the main utility is point-to-point communication on demand from users. Though sensors collaborate at the task execution level, at the local computing level they compete for successes of their transmissions and reception by the very nature of wireless channel and access protocols. Hence a game theoretic approach can be appropriate in this context too as in more conventional wireless networks, e.g. Aloha ([16], [11], [1]). To keep the notation simple, we assume that the neighbour relation is symmetric, i.e., $j \in N_i \Leftrightarrow i \in N_j$. This assumption does not in any way affect the analysis; we consider an example of directed links in Section IV-C. Let $p_{ij}^t(\underline{\alpha})$ (resp. $p_{ij}^r(\underline{\alpha})$)) denote the probability of successful transmission from i to j (resp. successful reception by i from j). Then,

$$\begin{aligned} p_{ij}^t(\underline{\alpha}) &= \frac{\alpha_i}{n_i}(1-\alpha_j)g_{ij}(\underline{\alpha}^{ij}) \\ p_{ij}^r(\underline{\alpha}) &= \frac{\alpha_j}{n_j}(1-\alpha_i)g_{ji}(\underline{\alpha}^{ij}) \end{aligned}$$

This follows from our assumptions that every sensor always has a packet to transmit and that in transmit mode a sensor transmits a packet to one of its neighbours with probability $\frac{1}{n_i}$ (see Section II and Equation 2). As in (2) the term $g_{ij}(.)$ captures the probability that interference to a transmission from i to j from other simultaneous transmissions is below the required threshold.

The probability of successful transmission by i, $p_i^t(\underline{\alpha})$ is simply $M_i(\underline{\alpha})$; hence equals $\sum_{j\in N_i} p_{ij}^t(\underline{\alpha})$. Let $T_i^t(\underline{\alpha})$ denote the expected time until a successful transmission by i and $T_{ij}^r(\underline{\alpha})$ denote the expected time until a successful reception by i from j. Now since the network is assumed to be saturated, the system is independent from slot to slot, it is, thus, clear that $T_i^t(\underline{\alpha})$ and $T_{ij}^r(\underline{\alpha})$ have the following simple forms.

$$T_i^t(\underline{\alpha}) = \frac{1}{p_i^t(\underline{\alpha})} \quad \text{and} \quad T_{ij}^r(\underline{\alpha}) = \frac{1}{p_{ij}^r(\underline{\alpha})}$$

Define, $F_i(\underline{\alpha}) = T_i^t(\underline{\alpha}) + \frac{1}{n_i}\sum_{j\in N_i} T_{ij}^r(\underline{\alpha})$. $F_i(.)$ is a reasonable measure of the the average time for communication with the neighbours. $T_i^t(.)$ is the cost of having small transmission attempt rates whereas $T_{ij}^r(.)$ penalises large values of α_i since a large value of α_i will tend to reduce the probability successful reception. Each sensor tries to locally minimise $F_i(\underline{\alpha})$, i.e., each node solves the following optimisation problem.

$$\min_{\alpha_{min}\le\alpha_i\le\alpha_{max}} F_i(\underline{\alpha})$$

where $0 < \alpha_{min} < \alpha_{max} < 1$. $0 < \alpha_{min}$ ensures a minimum transmission attempt rate for each sensor whereas $\alpha_{max} < 1$ accounts for the energy constraint; note that for battery energy E and transmission power P, $\frac{E}{\alpha_i P}$ is an upper bound on the lifetime of sensor i (considering other sources of energy consumption such as microprocessor and memory). The Nash equilibrium exists for this game since the "payoff function" $F_i(.)$ is strictly convex in α_i and the strategy space for each user is compact and convex. We will denote by $\underline{\alpha}^*$ the vector of the Nash equilibrium attempt probabilities (EAP).

A. Collocated Sensors

Proposition 4.1: If the sensors are collocated and each sensor has every other sensor as its neighbour then the Nash Equilibrium is unique and for each i, $\alpha_i^* = \min(\alpha_{max}, \max(\frac{1}{N}, \alpha_{min}))$.

In this scenario, $p_{ij}^t(\underline{\alpha}) = \frac{\alpha_i}{N-1}(1-\alpha_j)\Pi_{k\neq i,j}(1-\alpha_k)$ and $p_i^t(\underline{\alpha}) = \alpha_i\Pi_{k\neq i}(1-\alpha_k)$. Similarly, $p_{ij}^r(\underline{\alpha}) = \frac{\alpha_j}{N-1}(1-\alpha_i)\Pi_{k\neq i,j}(1-\alpha_k)$. Thus,

$$F_i(\underline{\alpha}) = \frac{1}{\alpha_i\Pi_{k\neq i}(1-\alpha_k)} + \sum_{j\neq i}\frac{1}{\alpha_j(1-\alpha_i)\Pi_{k\neq i,j}(1-\alpha_k)}$$

From the optimality conditions for convex programming, it is clear that $\underline{\alpha}^*$ is an EAP if and only if for each i, $(\alpha_i - \alpha_i^*)\frac{\partial F_i(\underline{\alpha}^*)}{\partial\alpha_i} \ge 0$, i.e., $(\alpha_i - \alpha_i^*)(N\alpha_i^* - 1) \ge 0$. The result now follows easily. $\alpha_i^* = \frac{1}{N}$, $1 \le i \le N$ is an intuitively desirable operating point for each sensor. $\frac{1}{N}$ equalises sensor throughputs as well maximises the minimum throughput in the network (see Section III).

1) A Stochastic Distributed Algorithm: While the solution of this simple problem is explicit, in practice the value of N would not be known at each of the sensors. Further, N could be slowly changing as sensors "die" owing to battery failure. We seek a distributed algorithm for solving this optimisation problem. Question: since we know the solution to the problem why not just estimate N in a distributed fashion? Answer: an explicit solution will not be available for more complex sensor network topologies; an algorithm that does not utilise the explicit form of the solution may lead to insights for solving more complex problems. Consider the following algorithm. Π denotes projection on $[\alpha_{min}, \alpha_{max}]$ and $a(k)$ the step size in iteration k. For $k \geq 0$,

$$\alpha_i(k+1) = \Pi\left[\alpha_i(k) - a(k)\frac{\partial F_i(\underline{\alpha}(k))}{\partial \alpha_i}\right],\ i = 1, 2, \ldots, N \tag{19}$$

Note that for all i, $F_i(.)$ are identical; we will denote them by F. $F(\underline{\alpha}) \geq 0$ for all $\underline{\alpha} \in [\alpha_{min}, \alpha_{max}]$ and $F(.)$ is continuously differentiable. Further, $\nabla F(.)$ is differentiable, hence locally Lipshitz which, on the compact set $[\alpha_{min}, \alpha_{max}]$, implies Lipshitz continuity of $\nabla F(.)$. Let K denote the Lipshitz constant of $\nabla F(.)$.

Proposition 4.2: Let $a(k) = a,\ k \geq 0$ such that $0 < a < \frac{2}{K}$. Then Algorithm 19 converges to the Nash equilibrium.

Note that Algorithm 19 is actually a projected gradient algorithm in this case. Its convergence follows from [5] (Proposition 3.4). The gradient has a particularly simple form.

$$\frac{\partial F_i(\underline{\alpha}(k))}{\partial \alpha_i} = \frac{-T_i^t(\underline{\alpha}(k))}{\alpha_i(k)} + \frac{1}{N-1}\sum_{j \neq i}\frac{T_{ij}^r(\underline{\alpha}(k))}{1-\alpha_i(k)} \tag{20}$$

In a distributed implementation, $T_i^t(\underline{\alpha})$ and $T_{ij}^r(\underline{\alpha}),\ j \neq i$ need to be *estimated* since their functional form would not be known. These estimates can be obtained only by local measurements. Sensor i estimates the time its takes for successful transmission; this will be done by noting the time-difference between the successful transmissions and taking their average. It also keeps track of the times when it receives the addressed transmissions from each of the other sensors to itself successfully; again the averaged time-difference between the successful receptions from say j is an estimate of $T_{ij}^r(\underline{\alpha}(k))$. These estimates can be obtained simultaneously for all neighbours or one at a time, i.e., tracking receptions from only one neighbour at a time. Note that these estimates are unbiased and the measurement noise sequence is i.i.d.

Proposition 4.3: Let in Algorithm 19, the partial derivatives be replaced by their estimates obtained as discussed above. Let $a(k)$ satisfy $\sum_{k=1}^{\infty} a(k) = \infty$ and $\sum_{k=1}^{\infty} a(k)^2 < \infty$. Then the generated sequence $\{\underline{\alpha}(k),\ k \geq 0\}$ converges a.s. to the Nash equilibrium.

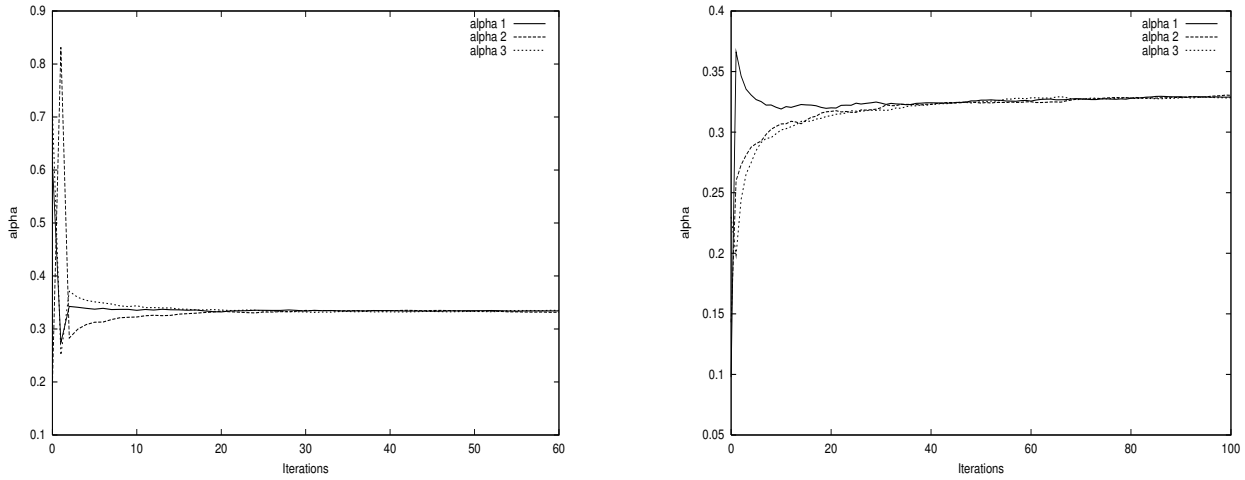


Fig. 8. Sensor network of 3 collocated nodes. Stochastic algorithm. $\underline{\alpha}_0$ is $(0.614, 0.19, 0.714)$ and $(0.098, 0.143, 0.23)$. $\alpha_i^* = 1/3,\ i = 1, 2, 3$.

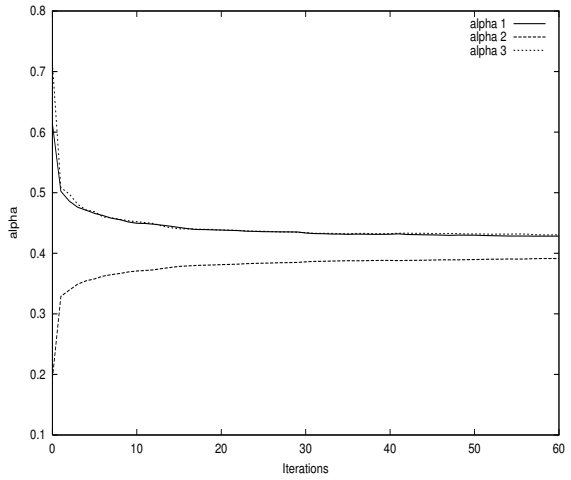
Proposition 4.3 can be proved using standard stochastic approximation arguments ([14]). As an example we consider the case when $N = 3$ and sensors estimate the gradient. $\alpha_{min} = 0.01$ and $\alpha_{max} = 0.99$. Therefore, the unique Nash equilibrium $\alpha_i^* = \frac{1}{3},\ i = 1, 2, 3$ is an interior solution. Figure 8 shows variation of α_i's with iterations for two initial conditions: $\underline{\alpha}_0$ is $(0.098, 0.143, 0.23)$ and $(0.614, 0.19, 0.714)$. We choose gain sequence $a(k) = \frac{0.0025}{(k+1)^{0.6}}$. The partial derivatives are estimated over an interval of 1000 slots. α_i's are updated at the end of each estimation interval. Thus, the time required for an iteration corresponds to the estimation interval. Observe from Figure 8 that the algorithm converges to EAP only within $80 - 100$ iterations.

B. Spatially Distributed Sensor Networks

The previous results can be generalised to a scenario where sensors are spatially distributed. The algorithm takes the following form.

$$\alpha_i(k+1) = \Pi\left[\alpha_i(k) - a(k)\left(\frac{-\tilde{T}_i^t(\underline{\alpha}(k))}{\alpha_i(k)} + \frac{1}{n_i}\sum_{j\in N_i}\frac{\tilde{T}_{ij}^r(\underline{\alpha}(k))}{1-\alpha_i(k)}\right)\right],\ i = 1, 2, \ldots, N \qquad (21)$$

$\tilde{T}$ denotes its estimate; gradient estimation is as explained earlier. Convergence of the algorithm can be proved on the lines of [19]. Assume that the Nash equilibrium is in the interior of $[\alpha_{min}, \alpha_{max}]$. Denote by $h(\underline{\alpha})$ the vector $\left[\frac{\partial F_1(\underline{\alpha})}{\partial \alpha_1}\frac{\partial F_2(\underline{\alpha})}{\partial \alpha_2}\ldots\frac{\partial F_n(\underline{\alpha})}{\partial \alpha_n}\right]^T$ and by $H(\underline{\alpha})$ the Jacobian matrix of $h(\underline{\alpha})$. If $H(\underline{\alpha})$ can be shown to be uniformly positive definite, i.e., $H(\underline{\alpha}) > \epsilon I$ for some $\epsilon > 0$ and for all $\underline{\alpha}$, convergence of Algorithm 21 can be obtained. However, it is difficult to



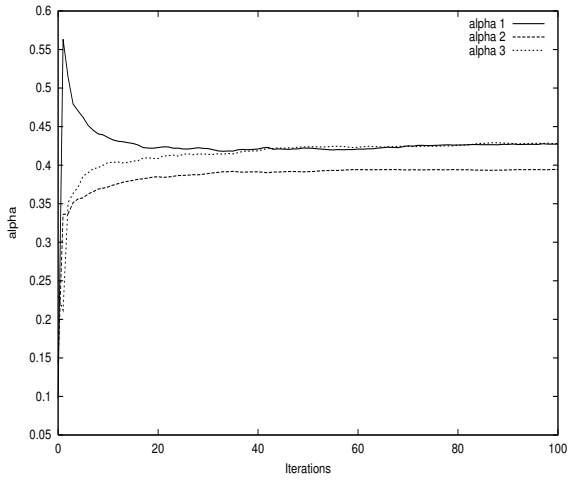


Fig. 9. Asymmetric 3 sensor network. Algorithm 21). $\underline{\alpha}_0 = (0.614, 0.19, 0.714)$ and $(0.098, 0.143, 0.23)$. $\underline{\alpha}^* = (0.432, 0.396, 0.432)$.

prove this result for general topologies. We, therefore, motivate the result using simple examples. Recall the network of 3 sensors in Figure 1. It is easy to see that,

$$\begin{aligned} F_1(\underline{\alpha}) &= \frac{1}{\alpha_1(1-\alpha_2)(1-\alpha_3)} + \frac{2}{\alpha_2(1-\alpha_1)} \\ F_2(\underline{\alpha}) &= \frac{2}{\alpha_2(2-\alpha_1-\alpha_3)} + \frac{1}{2}\left(\frac{1}{\alpha_1(1-\alpha_2)(1-\alpha_3)} + \frac{1}{\alpha_3(1-\alpha_2)(1-\alpha_1)}\right) \\ F_3(\underline{\alpha}) &= \frac{1}{\alpha_3(1-\alpha_2)(1-\alpha_1)} + \frac{2}{\alpha_2(1-\alpha_3)} \end{aligned}$$

From the optimality conditions for convex programming, $\underline{\alpha}^*$ is EAP if and only if for each i, $(\alpha_i - \alpha_i^*)\frac{\partial F_i(\underline{\alpha}^*)}{\partial \alpha_i} \geq 0$. Let $\alpha_{min} = 0.01$ and $\alpha_{max} = 0.99$. Then the interior solution is $(0.432, 0.396, 0.432)$. Further, it can be shown that the Jacobian, $H(\underline{\alpha})$, in this case is uniformly positive definite. This also proves the uniqueness of EAP ([19]). Recall that the MMTAP for this network are $(\sqrt{2}-1, 1-\frac{1}{\sqrt{2}}, \sqrt{2}-1)$ (Section III-C). Let $g(\underline{\alpha}) = -h(\underline{\alpha})$ and take $\|g(\underline{\alpha})\|^2$ as the Lyapunov function. The convergence of (21) in this example now follows from [19]. Figure 9 shows the variation of α_i's with iterations for two initial conditions: $\underline{\alpha}_0 = (0.098, 0.143, 0.23)$ and $(0.614, 0.19, 0.714)$. $a(k) = \frac{0.0035}{(k+1)^{0.6}}$ and the estimation interval is 1000 slots.

The network of 4 sensors symmetrically placed and operating with two neighbours each in Figure 3 gives,

$$F_i(\underline{\alpha}) = \frac{1}{\alpha_i(1-\alpha_{i+1})(1-\alpha_{i+2})} + \frac{1}{2}\left(\frac{1}{\frac{\alpha_{i+1}}{2}(1-\alpha_i)(1-\alpha_{i+3})} + \frac{1}{\frac{\alpha_{i+3}}{2}(1-\alpha_i)(1-\alpha_{i+1})}\right) \quad (22)$$

for $i = 1, \ldots, 4$; addition in the subscript being modulo 4. It can be shown that the unique EAP is $\alpha_i^* = \frac{1}{3},\ i = 1, \ldots, 4$. Figure 10 shows convergence of (21) applied to this network. The

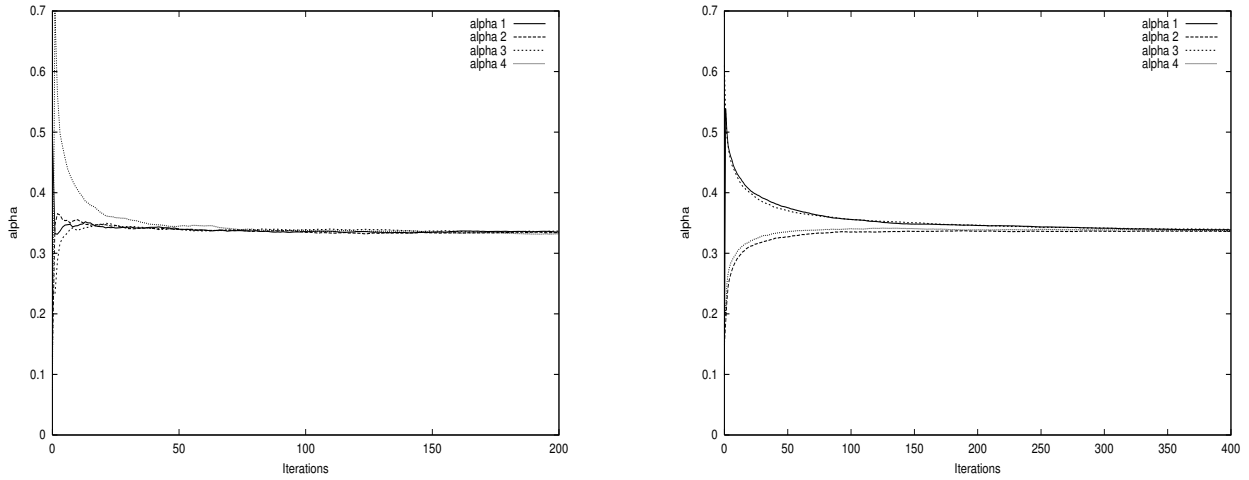


Fig. 10. A 4 sensor network. Algorithm 21). $\underline{\alpha}_0 = (0.496, 0.129, 0.228, 0.074)$ and $(0.098, 0.149, 0.61, 0.23)$, $\alpha_i^* = 1/3,\ i = 1, 2, 3, 4$..

initial conditions are $\underline{\alpha}_0 = (0.098, 0.143, 0.23)$ and $(0.614, 0.19, 0.714)$. $a(k) = \frac{0.002}{(k+1)^{0.6}}$ and the estimation interval is 1000 slots. Note from Section III-D that for this network the MMTAP are the same as the EAP. However, Figure 10 and 3 show that convergence to EAP is comparatively faster than convergence to MMTAP for the same initial conditions.

C. Discussion

We now address the question of the computing performance of sensors if the attempt probabilities are tuned to EAP. Consider the maximum calculation scenario discussed in Section III-E for 16 spatially distributed sensors as shown in Figure 11. Sensor 10 acts as the observer. Figure 12 shows the variation of the computing rate with the sampling rate for various values of $\underline{\alpha}$. In Section III-E we already identified the limitations of configuring the attempt probabilities a priori and need to *adapt* them to the sampling rates. For this example, observe from Figure 12 that when $\alpha_i = 0.05$ for each i, sampling rates less than 0.033 samples per slot can handled whereas when $\alpha_i = 0.2$ for each i this value is around 0.04 samples per slot. Thereafter the system becomes unstable and the computing delays become very large. The first approach then is to tune to the MMTAP. MMTAP is important in this scenario because the global maximum computation is limited by the lowest sensor throughput in the network. As discussed earlier, tuning to MMTAP incurs high communication overhead. This makes EAP particularly important.

In our game formulation, we had assumed that every sensor receives as well as transmits. In this example, sensors 11 to 16 do not receive from any sensor. We, therefore, fix their attempt

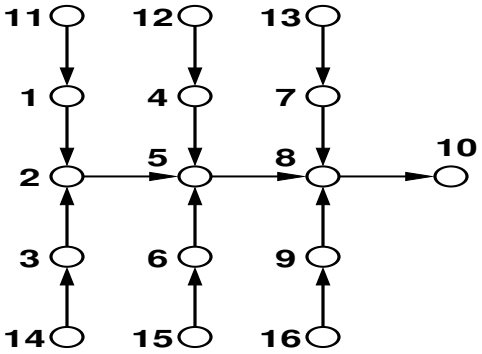


Fig. 11. A 16 sensor network deployed to calculate the maximum.

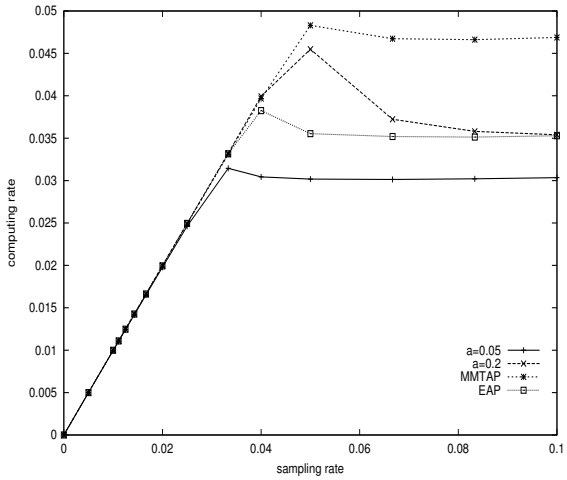


Fig. 12. Computing rate vs sampling rate for the 16 sensor network computing maximum of the sensed values.

probabilities to some value while the other sensors tune to EAP; in the results shown $\alpha_i = 0.12$ for $i = 11, \ldots, 16$. Since tuning to MMTAP is based on throughput measurements, all sensors (including 11 to 16) optimise their attempt rates to MMTAP. MMTAP and EAP are obtained by considering the network to be saturated. These are then used in the actual maximum calculation task with sampling. Thus, in our results EAP and MMTAP are not actually adapted to the sampling rates. However, the probabilities of successful transmission and reception are the lowest in a saturated network. Thus, MMTAP and EAP in our results are conservative. In this sense, our results show the least performance that can be expected with MMTAP and EAP. Further, the computing performance in a saturated network can be inferred from these results since at large sampling rates the network is almost saturated. Observe that with MMTAP, the computing performance is the best among the cases considered. EAP performs better than the case when $\alpha_i = 0.05$ for each i and performs fairly well as compared to $\alpha_i = 0.2$. Thus, EAP can handle sampling rates up to 0.033 samples per slot as compared to around 0.05 samples per slot with MMTAP. Further, it can be said that if each sensor has an infinite store of samples, with attempt probabilities tuned to MMTAP, sensors can compute the maximum at the rate of 0.046 calculations per slot while with EAP the rate is 0.032 calculations per slot. Note that performance of EAP depends on the attempt rates selected for sensors 11 to 16. Further, since every sensor has only one other sensor (its parent) to transmit to, sensors tend to have higher attempt rates. Energy can be used as a powerful constraint to control transmission attempt rates in such scenarios.

These preliminary results are encouraging. Sensors are able to optimise their attempt rates with a simple algorithm and are able to perform computationally fairly well. A game formulation for wireless networks based on a particularly simple cost function is discussed in [7]. It also presents an asynchronous algorithm for updating attempt probabilities. However, it needs a wireless node to keep and exchange information regarding attempt probabilities and connectivity about sensors up to four hops away. Our algorithm, on the other hand, uses estimates based only the local measurements, no exchange of information is necessary. Secondly, the assumption of slotted time leads to particularly simple gradient estimation scheme, however, the approach based on measurements is not dependent on an analytical form; with simulation based gradient estimation schemes it may be possible to extend our algorithm to a different access scheme. In our results attempt probabilities were tuned to EAP assuming a saturated network. A future work is to investigate how sensors can adapt to EAP while actually sampling and computing. Finally, we have discussed minimisation of computation time, however, in general sensors will need to optimise their performance under various constraints, e.g., battery power. In such a case, our formulation can be easily extended to work in the presence of any energy saving techniques such as random sleep times.

V. Conclusion

Having identified the need for sensors to autonomously optimise their computing rates for give objectives and constraints, in this paper we formulated two optimisation approaches. The first, a team problem, is to maximise the minimum throughput of sensors and the second a game problem in which cost for each sensor is a measure of its communication time with its neighbours. We proposed measurement based iterative schemes for sensors to adaptively learn the optimal and equilibrium channel attempt probabilities in a distributed fashion. For the team problem, the communication overhead limits the implementation of this scheme. However, the adaptive algorithm for the game formulation requires strictly local measurements. We showed that with these algorithms a sensor network can adapt itself for given task objectives. The strength of our measurement based approach lies in the fact that many of the assumptions can be relaxed and the algorithms can be extended to other channel access schemes, energy saving mechanisms, etc. This should be seen as a step towards developing optimally self-organising architectures for sensor networks.

VI. 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] E. Altman, R. El Azouzi, and T. Jimenez. Slotted Aloha as a Stochastic Game with Partial Information. In *Wiopt'03: Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks*, 2003.
[2] D. Bertsekas. Convexification Procedures and Decomposition Algorithms for Large-Scale Nonconvex Optimization Problems. *Journal of Optimization Theory and Applications*, 29:169–197, 1979.
[3] D. Bertsekas. *Constrained Optimization and Multiplier Methods*. Academic Press, 1982.
[4] D. Bertsekas. *Nonlinear Programming*. Athena Scientific, 1995.
[5] D. Bertsekas and J. Tsitsiklis. *Parallel and Distributed Computation*. Prentice Hall, 1989.
[6] B. Bharath and V. Borkar. Stochastic Approximation Algorithms: Overview and Recent Trends. *Sadhana*, 24:425–452, 1999.
[7] V. Borkar and A. Kherani. Random Access in Wireless Ad Hoc Networks as a Distributed Game. In *WiOpt'04,: Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks*, 2004.
[8] J. Elson, L. Girod, and D. Estrin. Fine-Grained Network Time Synchronization using Reference Broadcasts. UCLA Technical Report 020008, 2002.
[9] F. Guerra and G. Lopez. A Parametric Embedding for the Finite Minimax Problem. *Mathematical Methods of Oper. Res.*, 49:359–371, 1999.
[10] S. Han. Variable Metric Methods for Minimizing a Class of Nondifferentiable Functions. *Mathematical Programming*, 20:1–13, 1981.
[11] Y. Jin and G. Kesidis. Equilibria of a Non-cooperative Game for Heterogeneous Users of an Aloha Network. *IEEE Comm. Letters*, 6:282–284, 2002.
[12] A. Karnik. *Optimal Self-organisation of Ad Hoc Wireless Sensor Networks*. PhD thesis, Indian Institute of Science, April 2004.
[13] A. Karnik and A. Kumar. Distributed Optimal Self-Organisation in a Class of Wireless Sensor Networks. In *IEEE INFOCOM*, 2004.
[14] H. Kushner and D. S. Clark. *Stochastic Approximation Methods for Constrained and Unconstrained Systems*. Springer-Verlag, 1978.
[15] P. L'Ecuyer. An Overview of Derivative Estimation. In *Conf. on Winter Simulation*, 1991.
[16] A. MacKenzie and S. Wicker. Selfish Users in Aloha: A Game-Theoretic Approach. In *VTC*, 2001.
[17] G. Di Pillo, L. Grippo, and S. Lucidi. A Smooth Method for the Finite Minimax Problem. *Mathematical Programming*, 60:187–214, 1991.
[18] K. Romer. Time Synchronization in Ad Hoc Networks. In *MobiHoc*, 2001.
[19] J. Rosen. Existence and Uniqueness of Equilibrium Points for Concave N-Person Games. *Econometrica*, 33(3):520–534, 1965.
[20] N. Shor. *Minimization Methods for Nondifferentiable Functions*. Springer, 1985.
[21] J. Spall. Multivariate Stochastic Approximation Using a Simultaneous Perturbation Gradient Approximation. *IEEE Trans. on Automatic Control*, 37(3):332–341, March 1992.
[22] J. Spall and J. Cristion. Model-free Control of Nonlinear Stochastic Systems with Discrete-time Measurements. *IEEE Trans. on Automatic Control*, 43(9):1178–1200, 1998.
[23] S. Tilak, N. Abu-Ghazaleh, and W. Heinzelman. A Taxonomy of Sensor Network Communication Models. *Mobile Computing and Communication Review*, 6(2), April 2002.
[24] F. Tobagi. Modelling and Performance Analysis of Multihop Packet Radio Networks. *Proc. of the IEEE*, 75(1):135–154, January 1987.
[25] I. Wang and J. Spall. A Constrained Simultaneous Perturbation Stochastic Approximation Algorithm Based on Penalty Functions. In *American Control conf.*, 1999.