Self-Optimisation of Sensor Networks as a Game

Aditya Karnik and Anurag Kumar

Indian Institute of Science, Bangalore-560012, INDIA
karnik@ece.iisc.ernet.in, anurag@ece.iisc.ernet.in

Abstract. The ability to communicate and compute allows a group of sensors to collaboratively process sensed data on-line rather than just sensing and forwarding the information. However, as an ad hoc wireless network without centralised control, sensors need to optimise their computing rates autonomously for given task objectives and constraints. Communication being the bottleneck for distributed computation in a sensor network, in this work, we formulate optimisation of computing rates as a game in which cost for each sensor is a measure of its communication time with the neighbours. Our game formulation not only leads to an explicit characterisation of the Nash equilibrium but also to a simple iterative scheme by which sensors can adaptively learn the equilibrium attempt probabilities using only the estimates of transmission and reception times from their local measurements.

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. The on-board processor and radio permit smart sensors to collaboratively process the sensed data, thus, giving them an ability to execute tasks more complex than just forwarding the sensed information. Distributed processing within the network will also save energy expended in long haul transport of data, however, its practicality will be limited by the rate at which sensors can collaboratively process data. The physical process being monitored will have certain processing requirements and unlike conventional distributed processing machines which have high speed communication buses between their processors, a sensor network has low speed wireless links, lacks centralised control, 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.

Consider a scenario in which an observer is interested in tracking the maximum value of a spatio-temporal process, for example, the temperature or the concentration of a pollutant in a region. To do so, n sensors are deployed to sample the process. Let us for the moment assume that the sensors are collocated so that only one transmission can be successful at a time and that the observer is accessible to all of them. Further, time is slotted and nodes attempt transmission to the observer randomly in each slot. We will assume that calculations at the observer are almost instantaneous. Sensors sample the process continuously at a rate akin to a Nyquist rate and store them in a packet queue, to be transmitted one at a time to the observer. Samples are time-stamped so that the observer can extract the maximum from samples with the same time-stamp and thus can track the time-varying maximum; we will ignore interpolation, spatial correlation and accuracy issues. Since the actual calculation is instantaneous and the sensors are collocated, the rate of computation of the maximum equals the rate at which samples of the same time-stamp are received by the observer from all sensors. Let us denote by α_i the attempt probability of sensor i. Then by somewhat laborious algebra, it is possible to arrive at a closed form for the mean computation time, $\tau(n)$, as a function of $\alpha_i,\ 1 \leq i \leq n$. For example,

$$\tau(2) = \frac{\alpha_1(1-\alpha_2)(\alpha_1+\alpha_2(2-3\alpha_1)}{\alpha_2(1-\alpha_1)(\alpha_1+\alpha_2-2\alpha_1\alpha_2)^2} + \frac{\alpha_2(1-\alpha_1)(\alpha_2+\alpha_1(2-3\alpha_2)}{\alpha_1(1-\alpha_2)(\alpha_1+\alpha_2-2\alpha_1\alpha_2)^2}$$

In this calculation, we assumed that the sampling rate is such that the packet queue at a sensor always has a sample, and thus in every slot sensor i attempts transmission with probability α_i. $\tau(2)$ is minimised at $\alpha_1^* = \alpha_2^* = 0.5$; in general $\tau(n)$ is minimised at $\alpha_i^* = \frac{1}{n},\ 1 \leq i \leq n$.

In general, finding a functional form of the computation rate of a given task in a sensor network 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 objective only on local measurements. One such approach was proposed in [7] and [8] by us. We argued that the higher the throughput of the sensors the faster will the overall computation proceed and hence, considered the problem of maximising the minimum throughput in the network by tuning transmission attempt probabilities at each sensor. We call attempt probabilities which maximise the minimum throughput, MMTAP. The distributed stochastic algorithms proposed for MMTAP in [7] and [8] were found to have a high communication overhead because of coupled objective function. In this paper, we investigate the possibility of optimising the computation time in a sensor network with the objective of finding simple and scalable algorithms.

A global computation proceeds in steps comprising of certain local computations at each sensor. Thus, the optimisation of the local computation rate at each sensor will lead to a higher global computation rate. Equally importantly such an optimisation can be done in a distributed fashion. 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; hence a functional form of even the local computation time cannot be obtained in the general case. 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 rate of reception and transmission is optimised at each sensor, the local computations will proceed rapidly so will the overall global computation. With these insights, we take two approaches to this problem. First, we view local computation as a set, rather than particular sequence, of transmission and receptions; hence we seek to minimise the time to complete say m transmissions to the neighbours and n receptions from them. Secondly, since the rate of local computation depends on the rate at which sensors transmit as well as receive, we minimise a measure of the average communication time with the neighbours. It is this approach that we will discuss in this paper.

The paper is organised as follows. In Section 2, we present our problem formulation. Section 3 discusses an algorithm for collocated sensor networks. The algorithm is generalised to spatially distributed sensors in Section 4. In Section 5 we discuss the implications of our optimisation to global computing and conclude in Section 6.

2 A Game Formulation

Consider a general 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. A sensor communicates only with certain other sensors within its transmission range (denoted by R_0) designated as its neighbours. To keep the notation simple, we assume that this relation is symmetric. This assumption does not in any way affect the analysis; we consider an example of directed links in Section 5. We consider a simple random access model for sensor transmissions: 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[1]. Let N_i (resp. n_i) denote the set (resp. number) of neighbours of sensor i. By our previous assumption, $j \in N_i \Leftrightarrow i \in N_j$. Let $\underline{\alpha}$ denote the vector of attempt probabilities. $\underline{\alpha}^{ij}$ denotes the vector $\underline{\alpha}$ with entries α_i and α_j omitted. We say that a transmission can be "decoded" when its signal to interference ratio (SIR) exceeds a given threshold β. We consider only the signal path loss with exponent η. Details of the model can be found in [7]. A transmission is successful when the sensor to which it is addressed (by actually inserting a physical address in the packet header) is in the receive mode, and is able to decode the transmission. Let $p^t_{ij}(\underline{\alpha})$ (resp. $p^r_{ij}(\underline{\alpha})$)) denote the probability of successful transmission from i to j (resp. successful reception by i from j). Thus,

$$p^t_{ij}(\underline{\alpha}) = \frac{\alpha_i}{n_i}(1-\alpha_j)g_{ij}(\underline{\alpha}^{ij})$$
$$p^r_{ij}(\underline{\alpha}) = \frac{\alpha_j}{n_j}(1-\alpha_i)g_{ji}(\underline{\alpha}^{ij})$$

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. 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 probability of successful

[1] This can be attributed to isotropy of the physical process and uniformity of processing

transmission by i, $p_i^t(\underline{\alpha})$ is simply $\sum_{j \in N_i} p_{ij}^t(\underline{\alpha})$. 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. It is clear that

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

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.

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

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. 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).

3 Collocated Sensors

Proposition 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 the above 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 EAP if and only if for each i, $(\alpha_i - \alpha_i^*)\frac{\partial F_i(\underline{\alpha}^*)}{\partial \alpha_i} \geq 0$, i.e., $(\alpha_i - \alpha_i^*)(n\alpha_i^* - 1) \geq 0$. The result now follows easily. $\alpha_i^* = \frac{1}{n}$, $1 \leq i \leq n$ is an intuitively desirable operating point for each sensor. $\frac{1}{n}$ equalises sensor throughputs as well maximises the minimum throughput in the network ([7]).

3.1 A Stochastic Distributed Algorithm

While the solution of the 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 $k \geq 0$,

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

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 2. *Let $a(k) = a$, $k \geq 0$ such that $0 < a < \frac{2}{K}$. Then Algorithm 1 converges to the Nash equilibrium.*

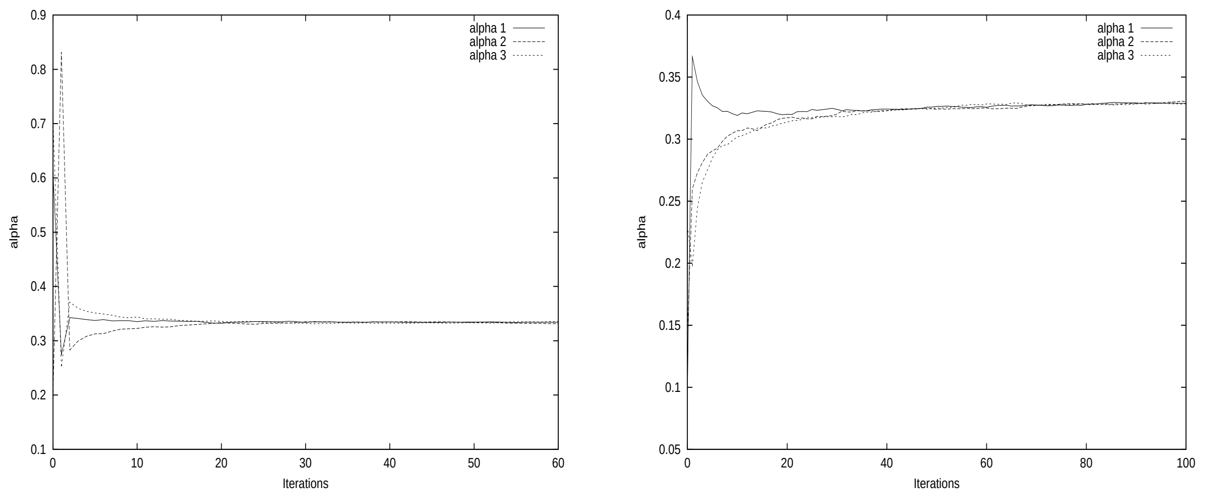

Figure 1. Sensor network of 3 collocated nodes. Stochastic case. $\underline{\alpha}_0$ is $(0.614, 0.19, 0.714)$ and $(0.098, 0.143, 0.23)$. $\alpha_i^* = 1/3,\ i = 1, 2, 3$.

Note that Algorithm 1 is actually a projected gradient algorithm in this case. Its convergence follows from [1] (Proposition 3.4). The gradient has a particularly simple form.

$$\frac{\partial F_i(\underline{\alpha}(k))}{\partial \alpha_i} = \frac{-T_i^t(\underline{\alpha}(k))}{\alpha_i(k)} + \frac{1}{n-1}\sum_{j \neq i} \frac{T_{ij}^r(\underline{\alpha}(k))}{1-\alpha_i(k)} \tag{2}$$

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 3. *Let in Algorithm 1, 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.*

Proposition 3 can be proved using standard stochastic approximation arguments ([9]). 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 1 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.

4 Spatially Distributed Sensor Networks

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

$$\alpha_i(k+1) = \Pi\left[\alpha_i(k) - a(k)\left(\frac{-\tilde{T}_i^t(\underline{\alpha}(k))}{\alpha_i(k)} + \frac{1}{n_i}\sum_{j \in N_i} \frac{\tilde{T}_{ij}^r(\underline{\alpha}(k))}{1-\alpha_i(k)}\right)\right],\ i = 1, 2, \ldots, n \tag{3}$$

$\tilde{T}$ denotes its estimate; gradient estimation is as explained earlier. Convergence of the algorithm can be proved on the lines of [11]. Assume that the Nash equilibrium is in the interior of $[\alpha_{min}, \alpha_{max}]$.

Denote by $h(\underline{\alpha})$ the vector $\left[\frac{\partial F_1(\underline{\alpha})}{\partial \alpha_1} \frac{\partial F_2(\underline{\alpha})}{\partial \alpha_2} \cdots \frac{\partial F_n(\underline{\alpha})}{\partial \alpha_n}\right]^T$ and by $H(\underline{\alpha})$ the Jacobian matrix of $h(\underline{\alpha})$. The key proposition is that $H(\underline{\alpha})$ is uniformly positive definite, i.e., $H(\underline{\alpha}) > \epsilon I$ for some $\epsilon > 0$ and for all $\underline{\alpha}$. We defer the proofs to the longer version of the paper and motivate the result using simple examples. Consider a network of 3 sensors such that $N_1 = \{2\}$, $N_2 = \{1, 3\}$, $N_3 = \{2\}$ and when 1 transmits to

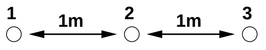

Figure 2. 3 node network, $R_0 = 1$m, $\beta = 2$, $\eta = 4$.

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. Therefore,

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

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 ([11]). Let $g(\underline{\alpha}) = -h(\underline{\alpha})$ and take $\|g(\underline{\alpha})\|^2$ as the Lyapunov function. The convergence of (3) in this example now follows from [11]. Figure 3 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.

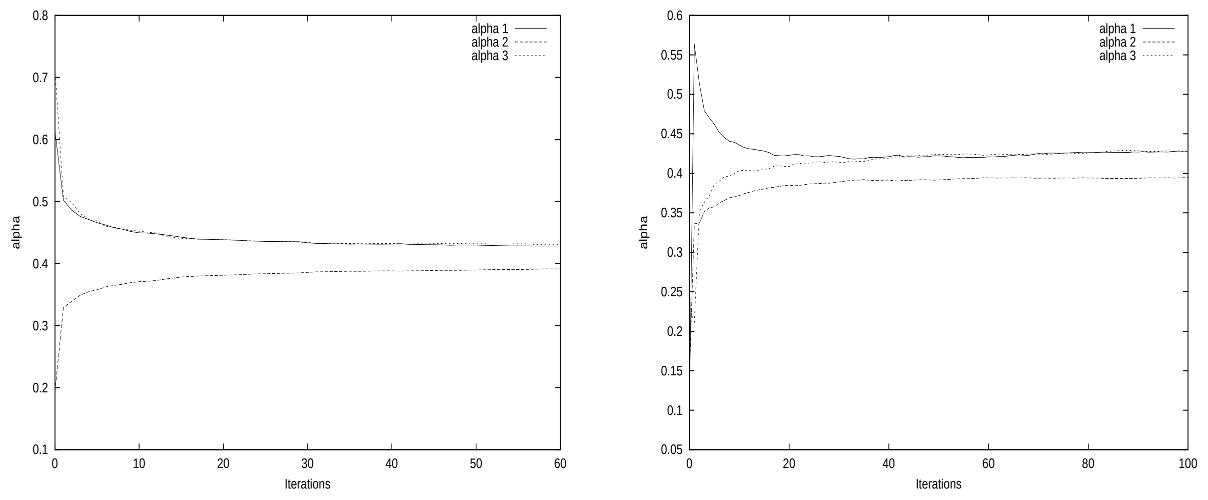

Figure 3. Asymmetric 3 sensor network. Stochastic case. $\underline{\alpha}_0 = (0.614, 0.19, 0.714)$ and $(0.098, 0.143, 0.23)$. $\underline{\alpha}^* = (0.432, 0.396, 0.432)$.

Figure 4 shows a network of 4 sensors symmetrically placed and operating with two neighbours each. $\eta = 4, \beta = 2, R_0 = 1$m so that,

$$F_i(\underline{\alpha}) = \frac{1}{\alpha_i(1-\alpha_{i+1})(1-\alpha_{i+2})} + \frac{1}{2}\left(\frac{1}{\alpha_{i+1}(1-\alpha_i)(1-\alpha_{i+3})} + \frac{1}{\alpha_{i+3}(1-\alpha_i)(1-\alpha_{i+1})}\right) \quad (4)$$

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 5 shows convergence of (3) applied to the 4 sensor network shown in

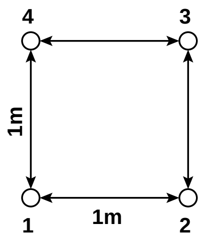

Figure 4. 4 node network, $R_0 = 1\text{m}$, $\beta = 2$, $\eta = 4$.

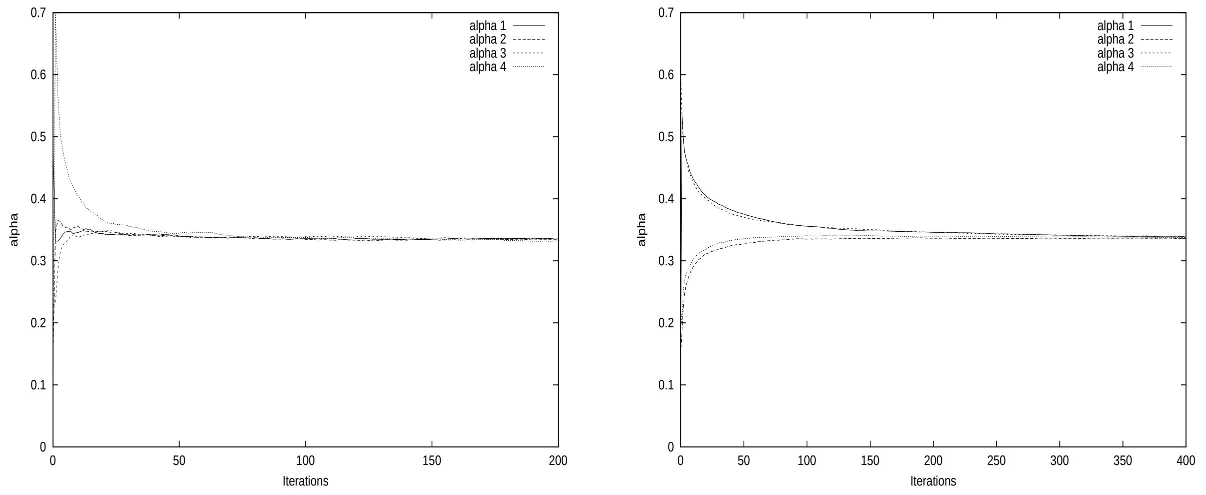

Figure 5. A 4 sensor network. Stochastic case. $\underline{\alpha}_0 = (0.496, 0.129, 0.228, 0.074)$ and $(0.098, 0.149, 0.61, 0.23)$, $\alpha_i^* = 1/3,\ i = 1, 2, 3, 4.$.

Figure 4. The 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.

5 Discussion

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. Since individual sensors cannot assess the global processing speed one way to effectively optimise the network in a distributed fashion is to optimise the local computing speeds. 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. We, therefore, believe that a game theoretic approach is appropriate in this context too as in more conventional wireless channels, e.g. Aloha ([10], [5], [4]). Our game formulation not only leads to an explicit characterisation of the Nash equilibrium but also to a simple iterative scheme by which sensors can adaptively learn the equilibrium attempt probabilities. An important question is *what is the computing performance of sensors if the attempt probabilities are tuned to EAP?*

Consider first the maximum calculation example discussed in Section 1 for 16 spatially distributed sensors as shown in Figure 6. Sensor 10 acts as the observer. Sensors follow 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. Note that, the rate at which sensors sample the temperature must be less than the rate at which the maximum is computed by the network. This can be understood by viewing this computing network as a queueing system in which the sensor network acts as a complex "server" and the set of samples taken simultaneously at each sensor as a "customer" with "service time" equal to the time required for the calculation of the maximum over the set by the server. A "good value of $\underline{\alpha}$" for a given temperature sampling rate cannot be known a priori since the computing rate (rate of maximum calculations by the network per slot) is a complex function of $\underline{\alpha}$. This is illustrated by Figure 7 which shows the variation of the computing

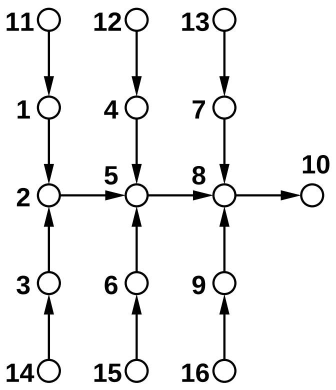

Figure 6. A 16 sensor network deployed to calculate the maximum.

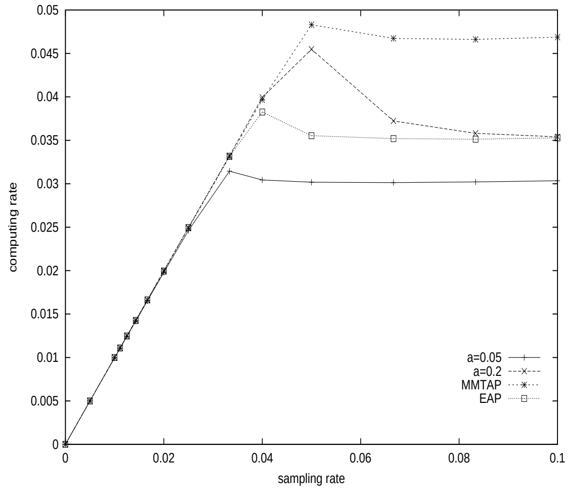

Figure 7. Computing rate vs sampling rate for the 16 sensor network computing maximum of the sensed values.

rate with the sampling rate for various values of $\underline{\alpha}$. Observe 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. Therefore, it is necessary to adapt attempt probabilities to the sampling rates (see [8]). One way is to tune them to MMTAP (Section 1, [8]). 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 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.

For greater accuracy, each sensor can estimate the process in its spatial neighbourhood from its own and its neighbours' measurements and estimate the maximum in this neighbourhood. It then proceeds to collect the corresponding values from its children and forwards the maximum of them. Thus each sensor will require to receive as well as transmit. We consider such a scenario in Figure 8 which shows a network of 200 sensors deployed randomly in a square of $20m \times 20m$ to calculate the maximum in the region. The topology has been optimised for throughput ([6]). The observer is at the origin. Sensors use a tree embedded in the network to forward the maximum values to the observer. At each sensor the raw measurements to be sent to the neighbours are queued up in a packet queue. The actual estimation algorithm is not of interest in this work, therefore, we assume that if a sensor decides to transmit, it chooses a neighbour randomly and if the transmission to that neighbour is successful, the packet is

Table 1. Comparison of computing rate for various $\underline{\alpha}$ assuming each sensor has infinite store of samples.

$\underline{\alpha}$	$\alpha_i = 0.05, \forall i$	$\alpha_i = 0.1, \forall i$	$\alpha_i = 0.2, \forall i$	MMTAP	EAP
Computing rate	0.0027	0.0015	0.000013	0.0012	0.0010

removed from the queue[2]. If the chosen neighbour is the parent in the tree and a maximum calculation is locally complete, i.e., the corresponding results have been obtained from the children in the tree then the result of the local maximum calculation is forwarded to it. We consider a scenario in which each sensor has infinite store of samples. Figure 9 shows convergence of attempt probabilities of sensors 1, 5, 10 and 34 to EAP. The computing rates are shown in Table 1. EAP adapts fairly well and has performance comparable to MMTAP and $\alpha_i = 0.1$ for each i. However, the performance with $\alpha_i = 0.05$ for each i is much higher than that with EAP. Nash equilibria are known to be inefficient ([3]). We believe that energy constraints can be used as a mechanism to enforce social optimum.

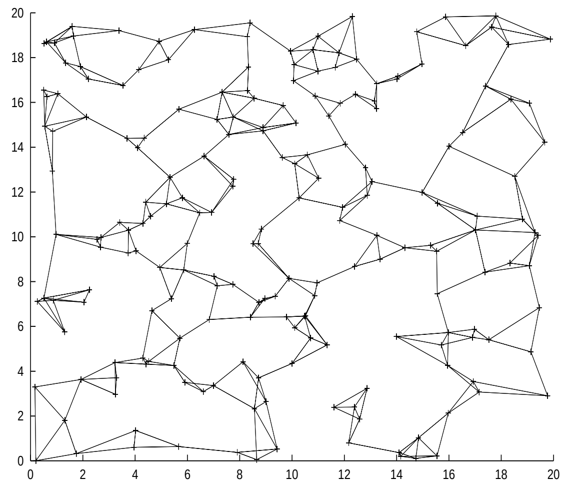

Figure 8. 200 sensors deployed randomly in a square of $20m \times 20m$ to calculate the maximum in the region. The observer is at the origin.

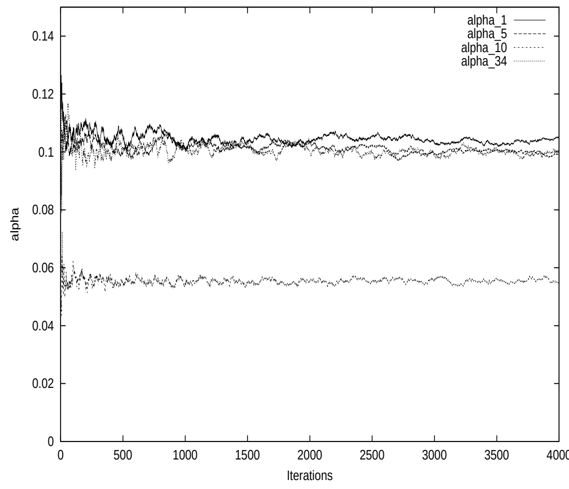

Figure 9. Convergence of attempt probabilities of sensors 1, 5, 10 and 34 to EAP.

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 self-organising sensor networks based on a particularly simple cost function is discussed in [2]. It also presents an asynchronous algorithm for updating attempt probabilities. However, it needs a sensor 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 tied to an analytical form; with simulation based gradient estimation schemes it may be possible to extend our algorithm to a different access scheme. Moreover, in our results attempt probabilities were tuned to EAP assuming a saturated network, however, it will be possible for sensors to 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.

6 Conclusion

In this paper, we formulated the problem of optimisation of the computing rate in sensor networks as a game and proposed an iterative scheme for sensors to adaptively learn the equilibrium attempt

[2] Sending each measurement to each neighbour will increase accuracy

probabilities with local measurements. Our approach is promising and can be seen as a step towards developing optimally self-organising architectures for sensor networks.

7 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 and J. Tsitsiklis. *Parallel and Distributed Computation.* Prentice Hall, 1989.
2. V. Borkar and A. Kherani. Self-Organisation of Wireless Sensor Networks as a Distributed Game. preprint, 2003.
3. P. Dubey. Inefficiency of Nash Equilibria. *Mathematics of Operations Research*, 11:1–8, 1986.
4. E. Altman et al. Slotted Aloha as a Stochastic Game with Partial Information. In *Wiopt'03: Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks*, 2003.
5. 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.
6. A. Karnik. *Performance and Optimal Self-organisation of Wireless Sensor Networks.* PhD thesis, Indian Institute of Science, 2004. under preparation.
7. A. Karnik and A. Kumar. Distributed Optimal Self-Organisation in a Class of Wireless Sensor Networks. In *IEEE INFOCOM*, 2004.
8. A. Karnik, A. Kumar, and V. Borkar. Distributed Self-tuning of Sensor Networks. In *WiOpt'04,: Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks*, 2004.
9. H. Kushner and D. S. Clark. *Stochastic Approximation Methods for Constrained and Unconstrained Systems.* Springer-Verlag, 1978.
10. A. MacKenzie and S. Wicker. Selfish Users in Aloha: A Game-Theoretic Approach. In *VTC*, 2001.
11. J. Rosen. Existence and Uniqueness of Equilibrium Points for Concave N-Person Games. *Econometrica*, 33(3):520–534, 1965.