A Clustered Incremental Gradient Method

Anonymous Author(s)
Affiliation
Address
email

Abstract

This paper reviews the optimization algorithm described in the paper "A Convergent Incremental Gradient Method With Constant Step Size" by Blatt et al [1] and presents a novel algorithm that is similar to it but more suitable for implementation in wireless sensor networks and unlike [1] takes advantage of the parellellism present in wireless sensor networks. The algorithm presented in [1] is an incremental gradient algorithm for minimizing the sum of continuously differentiable functions and requires a single gradient evaluation per iteration and uses a constant step size. For the case that the gradient is bounded and Lipschitz continuous, the method visits regions in which the gradient is small infinitely often. Under certain unimodality assumptions, global convergence for this method was shown and for the quadratic case, a global linear rate of convergence was proved. This algorithm was claimed to be more suitable for wireless sensor networks. Although our review confirms the convergence properties of the algorithm in [1] to hold, it finds the algorithm to be not suitable for use in wireless sensor networks and the new algorithm developed in the process tries to address the vulnerabilities in [1] as regards wireless sensor networks while requiring roughly the same number of gradient evaluations as the Incremental Aggregate Algorithm described in [1] but at the same time taking advantage of the parallelism available in Wireless Sensor Networks.

1 Introduction

For an optimization problem concerning the sum of L functions satisfying the aforementioned required properties:

$$\text{minimize } f(x) = \sum_{l=1}^{L} f_l(x),\ x \in \Re^p \tag{1}$$

where $\Re^p$ is the p-dimensional Euclidean space, and $f_l : \Re^p \rightarrow \Re$ are continuously differentiable scalar functions on $\Re^p$. The method in [1] generates a sequence of $\{x^k\}_{k\geq 1}$ as follows. Given arbitrary L initial points x^1, x^2,..,x^L an aggregated gradient, denoted by d^L is defined as $\sum_{l=1}^{L} \bigtriangledown f_l(x^l)$.

For $k \geq L$,

$$x^{k+1} = x^k - \mu \frac{1}{L} d^k \tag{2}$$

$$d^{k+1} = d^k - \bigtriangledown f_{(k+1)_L}(x^{k+1-L}) + \bigtriangledown f_{(k+1)_L}(x^{k+1}) \tag{3}$$

where μ is a positive constant step size chosen small enough to ensure convergence, $(k)_L$ denotes (k modulo L) when (k modulo L) > 0 otherwise L with representative class $\{1, 2, ..., L\}$. We should note that in [1] the definition of $(k)_L$ is given as (k modulo L). We believe that it was a mistake in writing not in implementation. This leaves out the L^{th} sensor from participating in the optimization process and thus the result will never converge to the optimal value. Secondly we would also like to note that in this definition the term $\frac{1}{L}$ is used in (1) for easier understanding but

in the diagrams used to present the results in [1] μ is considered as the only co-efficient of d^k. In this case we followed the same convention for easier comparison.

The authors of [1] state that this method is suitable for optimization problems arising in wireless sensor networks where each sensor has access to one function and optimizing the sum of all the functions through a centralized processing system is not feasible due to bandwidth constraints. In such scenarios algorithms like steepest descent or adaptive stepsize algorithms are not feasible as they would require the evaluation of the gradient of the complete objective function or the complete objective function itself for estimating the descent direction or for adapting the stepsize to ensure convergence, which in turn would require sending information to a centralized processing system from all the sensors. But with the proposed algorithm, in a wireless sensor network with L processors, once the initialization process is done, upon receiving x^{k-1} and d^{k-1} from processor $(k-1)_L$, processor $(k)_L$ computes x^k and d^k according to (2) and (3), respectively and transmits them to processor $(k+1)_L$. Note that $\bigtriangledown f_{(k)_L}(x^{k-L})$ in (3) is available at processor $(k)_L$ since it was the last gradient computed at that processor.

Given the authors are aiming their work towards distributed wireless networks, a good choice for comparison was the standard incremental gradient algorithm with constant stepsize. The standard incremental gradient can be written as:

$$x^{k+1} = x^k - \mu(k) \bigtriangledown f_{(k)_L}(x^k) \tag{4}$$

$\mu(k)$ was kept constant and equal to μ, to avoid experimentation on determining the optimal convergence rate for the stepsize and also to avoid evaluation of the complete objective function or its gradient. It was shown in the paper that in appropriate situations where the proposed algorithm can be theoretically proved to work, with sufficiently small stepsize, it approaches the optimal point faster than the standard incremental gradient algorithm, whereas the standard incremental gradient algorithm hits a limit cycle.

For brevity we exclude the proof of convergence given in [1]. The authors of [1] made the following assumptions about the functions in context:

(1) $ASSUMPTION1$. $\bigtriangledown f_l(x)$, $l = 1, ..., L$, satisfy a Lipschitz condition in $\Re^p$, i.e there is a positive number M_1 such that for all x, $\overline{x} \in \Re^p$, $\| \bigtriangledown f_l(x) - \bigtriangledown f_l(\overline{x})\| \leq M_1 \|x - \overline{x}\|$, $l = 1, ..., L$.

Assumption 1. implies that $\bigtriangledown f(x)$ also satisfies Lipschitz condition that is for all x, $\overline{x} \in \Re^p$, $\| \bigtriangledown f(x) - \bigtriangledown f(\overline{x})\| \leq M_2 \|x - \overline{x}\|$, where $M_2 = LM_1$.

(2) $ASSUMPTION2$. There exists a positive number M_3 such that for all $x \in \Re^p$, $\| \bigtriangledown f_l(x)\| \leq M_3$, $l = 1, ..., L$.

Assumption 2. implies that for all $x \in \Re^p$, $\| \bigtriangledown f(x)\| \leq M_4$, where $M_4 = LM_3$.

(3) $ASSUMPTION3$. $f(x)$ has a unique global minimum at x^*. The Hessian $\bigtriangledown^2 f(x)$ is continuous and poisitive definite at x^* .

(4) $ASSUMPTION4$. For any sequence $\{t^k\}_{k=1}^{\infty}$ in $\Re^p$ if $\lim_{k\to\infty} f(t^k) = f(x^*)$ or $\lim_{k\to\infty} \| \bigtriangledown f(t^k)\| = 0$, then $\lim_{k\to\infty} t^k = x^*$.

Assumption is equivalent to saying that for each neighborhood, υ of x^* there exists $\eta > 0$ such that if $f(x) - f(x^*) < \eta$ or $\| \bigtriangledown f(x)\| < \eta$ then $x \in \upsilon$.

It was also proved that under Assumption 3. there exists a neighborhood υ of x^* and positive constants A_1, A_2, B_1, B_2 such that for all $x \in \upsilon$:

$$A_1\|x - x^*\|^2 \leq f(x) - f(x^*) \leq B_1\|x - x^*\|^2$$
$$A_2\|x - x^*\|^2 \leq \| \bigtriangledown f(x)\|^2 \leq B_2\|x - x^*\|^2$$

Then they finally proved that if the the step size, $\mu < min\{\frac{1}{9M_1}, \frac{1}{M_2M_5}, \frac{\eta}{3M_1M_3}\sqrt{\frac{A_2n}{B_1}}\}$, then $\lim_{k\to\infty} x^k = x^*$.

Quadratic functions, although are common and important do not satisfy all the assumptions above, for example its gradient is not bounded. In case of quadratic functions, the authors proved that their algorithm converge linearly.

Through our review we came to the conlcusion that the algorithm, in [1], does hold the claimed convergence properties, it is not suitable for wireless sensor networks where sensors are not totally reliable due to power constraints and harsh environmental conditions. Thus reliying on the sensors sending messages in a cyclic manner is not a robust method as some sensors may be damaged during the processing while breaking the cycle and making the algorithm fail. More over the algorithm also does not take advantage of the parallel processing power that is generally available in a wireless sensor network. The model of computation generally applied in wireless sensor network is different from the model that makes this algorithm suitable. We first discuss the general model of a wireless sensor network and describe what are the pitfalls of [1] when applied in such a scenario and then we describe the new algorithm that we have developed while comparing with the Incremental Gradient Algorithm of [1] and the Standard Incremental Algorithm.

Often the sensors in a wireless sensor network are clustered with one sensor acting as a cluster head. This is done to take advantage of data aggregation and in such a case the algorithm can be modified such that instead of only one sensor evaluating the gradient of the function it is tied with, and updating the aggregate gradient and using it for finding the next iterate, all the sensors in one particular cluster will evaluate the gradients of their corresponding functions and send their evaluated gradients to the cluster head where the gradients will be summed up and the sum will be used to update the aggregate gradient by removing the previous sum of gradients for this cluster and adding the new sum to the aggregate gradient and this updated aggregate gradient will be used to calculate the new iterate.

Thus given there are N_c clusters, with cluster c starting from sensor $Start_c$ and ending at sensor End_c, and N_c initial points x^1, x^2,..,x^{N_c}. A clustered aggregate gradient denoted by d^{N_c} is

$$\sum_{c=1}^{N_c} \sum_{l=Start_c}^{End_c} \phi_{(l,c)} \bigtriangledown f_l(x^c)$$

where $\phi_{(l,k)}$ is equal to 1 if the l^{th} sensor was working at the time of evaluating $\bigtriangledown f_l(x^k)$. And L_k is the number of sensors properly functioning according to the knowledge of the cluster head that is evaluating the value of x^{k+1} using x^k. Our algorithm which we will refer to as Clustered Aggregate Gradient method is given as

For $k \geq N_c$,

$$x^{k+1} = x^k - \mu \frac{1}{L_k} d^k \tag{5}$$

$$d^{k+1} = d^k - \sum_{l=Start_{(k+1)_{N_c}}}^{End_{(k+1)_{N_c}}} \bigtriangledown f_l(x^{k+1-N_c}) + \sum_{l=Start_{(k+1)_{N_c}}}^{End_{(k+1)_{N_c}}} \bigtriangledown f_l(x^{k+1}) \tag{6}$$

where μ is a positive constant step size chosen small enough to ensure convergence, $(k)_{N_c}$ denotes (k modulo N_c) when (k modulo N_c) > 0 otherwise N_c with representative class $\{1, 2, ..., N_c\}$.

So once the initialization process is done, upon receiving x^{k-1} and d^{k-1} from the cluster $(k-1)_{N_c}$, processor the cluster head of cluster $(k)_{N_c}$ broadcasts x^{k-1} to all the nodes in the cluster, and each node returns the calculated gradient to the cluster head, which then uses these gradients to calculate tx^k and d^k according to (4) and (5), respectively and transmits them to processor the cluster $(k+1)_{N_c}$. Note that $\sum_{l=Start_{(k)_{N_c}}}^{End_{(k)_{N_c}}} \bigtriangledown f_l(x^{k-N_c})$ in (5) is available at the cluster head of the cluster $(k)_{N_c}$ since it was the last gradient computed at that processor.

This method follows the general way sensor networks communicate, and given the gradient evaluations are done in parallel each iteration in this way will not take significantly more time than one iteration of the method in [1]. And from our results we found that both the algorithm that we developed and the algorithm in [1] converge after roughly the equal number of gradient evaluations.

Given that in each cluster for our case the gradients are attempted to be evaluated in parallel we will achieve significant speedups when gradient evaluations take considerable time compared time required for communication. Moreover our algorithm will still work when one or two sensors stop working in a cluster or are put to sleep as long as the cluster head works. It will still work when a new sensor is elected as cluster head as long as the previous cluster head could pass along the last evaluated sum of gradients for this cluster to the new cluster head before the calculation of the iterate happens.

2 Clustering in wireless sensor networks

Wireless sensor networks (WSNs) are generally used to monitor environment or for military field surveillieance. In most cases the number of sensors in a network range from hundreds to tens of thousands and they are left unattended. These sensors observe the environment and the reports are collected by base stations. The sensors usually use limited capacity batteries which cannot be replaced and thus energy efficiency is a great concern. Moreover the lifetime of the sensors are not totally reliable as sensors may be damaged by the harsh conditions of the environment or their power may run out. These sensors sometimes have a processor attached to them, so that they can perform small calculations within them rather than relying on the base station to perform all the calculation.

Given the vast number of sensors deployed in a WSN, it is hardly possible for all the sensors to send their observed data directly to the base station as they are non-uniformly distributed over large areas; if they tried to send the data directly to the base station, many of them will be out of reach and the ones which are far away would run out of energy long before the ones which are close. If data from each sensor was sent naively by routing through other sensors, the sensors close to the base station would run out of their power source very quickly, given they would have to work most in the routing process. Thus to alleviate these problems, data aggregation is used, where multiple messages from multiple sensors can be combined by a sensor or node into one message shorter than all the messages combined, so that the nodes which will route this message uses less energy compared to what they would have used if they routed the individual messages. To facilitate this data aggregation, when the sensors are deployed in a WSN, they are deployed as clusters, with each cluster having its member sensors and a cluster head which can be any sensor like the other sensors or it could have more sophisticated hardware and software with a better power source.

In general the sensors in a cluster send their observations to the cluster head where the observations are aggregated and routed to the base station through other cluster heads, sometimes the sensors at the border between two clusters participate in the routing process. This whole process may be initiated by the base station through a query. A schematic diagram is shown in Figure 3.

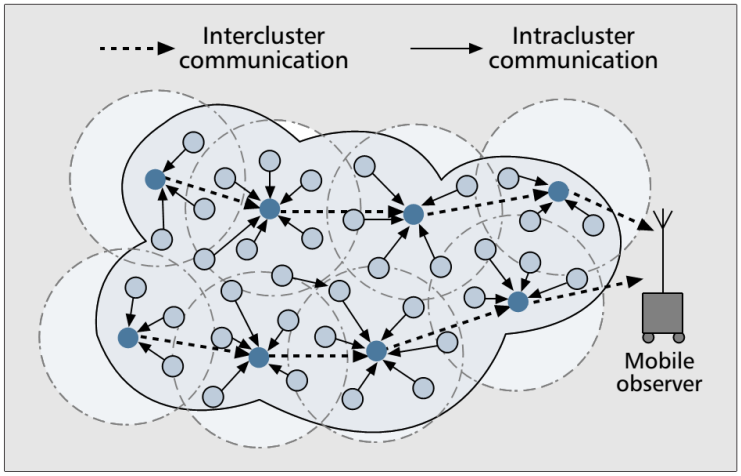


Figure 1: Illustration of data flow in a clustered network [3].

3 Convergence

We should also note that the convergence rate for the general case was also not derived in [1] but the proof of convergence was provided and for the quadratic case, linvear convergence was proved. We can for our algorithm reuse the proofs used in [1]. If we consider N_c functions, where each function is the sum of the functions accessible by the sensors in each cluster, then we can view each cluster as a processor with one function of the N_c functions and then we are just applying IAG on the objective function but with N_c functions making up the objective function this time. If assumption 1 and 2 hold on the individual functions of the sensors, they will also hold on their sums and given the total objective function remains the same, assumption 3 and 4 on the total objective function will still hold. Thus the same proofs for given in [1] for convergence apply. In case we are interested in the upper bound of step size then we can calculate them too. If for the modified model of the problem (sum of N_c functions to be minimized) , we denote the modified parameters by using a $'$ sign, $L' = N_c$ instead of L the other parameters M_1', M_2' for the proofs can also be derived by using the properties of the functions building each of the N_c functions. In the case all the N_c clusters are of equal size S and the functions are all very similar then we can approximate M_1', M_2' directly using the properties of differentiation of sums of functions as M_1' will approximately be equal to $S \times M_1$ and M_2' will be $\frac{1}{S}M_2$ and so on.

Intuitively speaking if the steepest descent algorithm converges and so does [1]. ICAG is supposed to converge given ICAG is a better approximation to steepest descent algorithm than IAG of [1]. But more work is intended to be done in the future to establish a global convergence rate.

To facilitate comparison with [1] we use the same functions as those used in [1]. In [1] the authors make use of the function

$$g(x) = c^2 \left[\frac{|x|}{c} - log\left(1 + \frac{|x|}{c}\right) \right]$$

In their numerical experiment they performed, each sensor collects a single noisy measurement of the polution level and the estimate of the average pollution level is found by minimizing the objective function defined by

$$f(x) = \sum_{l=1}^{L} f_l(x)$$

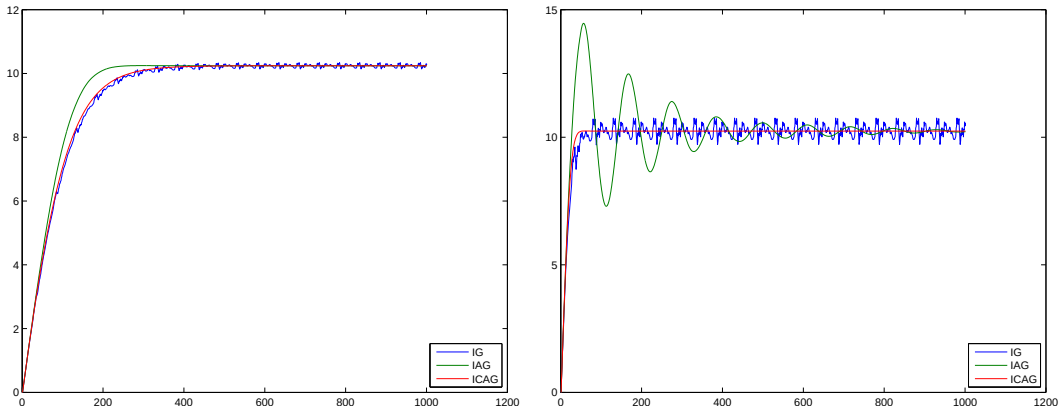
where $x \in \Re$, and

$$f_l(x) = \frac{1}{L} g(x - y_l)$$

where y_l is the measurement collected by sensor l. There were 50 sensors thus $L = 50$. To take into account faulty sensors, half of the samples were generated according to a Gaussian distribution with mean $m_1 = 10$ and unit variance ($\sigma_1^2 = 1$) and the other half was generated according to a Gaussian distribution with mean $m_2 = 10$ and ten times higher variance ($\sigma_2^2 = 10$). The coefficient c was chosen to be 10. The authors call this the robust Fair estimation problem. And for ICAG we considered equal sized clusters and each cluster containing 5 sensors, thus forming 10 clusters in total.

We ran experiments trying to minimize the objective function and found the claims of [1] to hold. We found that IAG has an optimal step size with which it takes least number of iterations to converge to the answer up 4 decimal places. We found the optimal stepsize upto two decimal place using inspection. The interesting thing we found with this is that using this step size ICAG converges slightly slower than IAG and requires a bit more number of iterations than IAG as can be seen in Figure 4.a . And we can also observe that IAG reaches the optimum point faster than the standard incremental algorithm which hits a limit cycle.

The reason for this can be explained. Let us consider two quadratic functions with the same minimum point, using IAG would calculate the aggregate gradient by taking the gradient of two functions



(a) Optimal μ for IAG = 0.95 (b) Optimal μ for ICAG = 4.78

Figure 2: Trajectories taken by the IG, IAG and IAG methods for the robust Fair estimation problem.

	Optimal μ for IAG	Optimal μ for ICAG	Ratio of μ of ICAG to IAG	Iterations for IAG	Iterations for ICAG	Ratio of iterations of IAG to ICAG
Fair Estimation	0.95	4.78	**5.03**	447	89	**5.02**
Sum of quadratics	0.37	1.86	**5.03**	371	73	**5.08**

Table 1: Comparison of IAG and ICAG.

at two different points, one slightly away from the optimal point than the other, but if there is only one cluster with two nodes, ICAG will sum up the gradients at the same point. Using the gradient that is a bit far from the minimum point here serves to give momentum towards the minimum point as both the gradients of the two functions will have the same sign but the one further will have a bit higher magnitude. And given in our experiment all the functions were unimodal and had their minimums near the mean we used , and we started far from the mean, this case was expected.

But the intersting thing that we found in our experiments is that if we made the clusters roughly equal in size and of size S then the optimal step size for ICAG would be roughly S times higher than that of IAG and the algorithm would converge with $\frac{1}{S}$ the iterations that IAG would require with its optimal step size as shown in 4.b and illustrated in table 1. Thus both the algorthms require roughly equal number of gradient evaluations. But in ICAG we are attempting to do S gradient evaluations in parallel. We use the word "attempt" because different sensors might be at different distances from the cluster head, thus all the nodes might not be starting their processing at the same time as the command to do so may not reach every node exactly at the same time, but still this parallelism can give us significant performance boost as will be explained in the next sections. We tried to see the convergence rate with the sum of quadratic functions of the form $f_l(x) = \frac{1}{L}(x - y_l)^2$ and got similar results, only that the optimal stepsizes varied this time but the ratio between the optimal stepsize of IAG and ICAG remained roughly equal.

4 Suitability in wireless sensor networks

4.1 Robustness to sensor damage

As mentioned before the lifetime of the sensors in a wireless sensor network is not reliable due to rought environmental conditions and power constraints, the IAG described in [1] works in a cyclic manner, i.e each sensor participates in the process one after another in a cyclic manner but the damage to one sensor will break the cycle causing the algorithm to fail. If a sensor that needs to send its message to the next sensor detects the damage of the recieving sensor and sends the message to another sensor, to perform the iterations, the aggregate gradient will not be updated properly as the

portion of gradient that the damaged sensor contributed to the whole aggregate gradient will not be removed properly and will cause the algorithm converge to a different place than it would converge without ever considering the sensor that got damaged.

In case of ICAG, as long as the cluster head is not damaged, damaged to the other sensors within a cluster will not give erroneous results as the algorithm will converge to the point where it would have converged without ever considering the damaged sensors, as the aggregate algorithm will be updated properly because the cluster head will still remove the sum of the gradients of all the members within the cluster calculated in the last iteration performed by it and add the new gradients of the surviving sensors and from there on, it will converge to the correct point. Even the damage of the cluster head can be handled. Often in a sensor network , the cluster heads are shuffled so that one sensor is not burdened too much, in such cases or even when shuffling is not done, the cluster head could after calculating the sum of the gradients pass along the data to the node within the cluster that is most likely to become the next cluster head, (generally it is based upon which node has the most power left). This way even the failure of the cluster head will not make the algorithm fail. If such a backup mechanism were to be used in IAG then every node will have to save its evaluated gradient to a neighboring node, which would have been too much of an overhead. Moreover if cluster heads are made more sophisticated in terms of hardware, software and power source which is done in many sensor networks, probability of their failure will be very low and thus the probability of the algorithm failing will be greatly reduced.

We test our algorithm by damaging some sensors at certain point within the iteration for ICAG and for the IAG and standard incremental gradient method we start with the damaged network. We started with 50 sensors and 10 equally sized clusters, at iteration 30 we damaged 15 sensors from different clusters without destroying a whole cluster, and the result is shown in Figure **??**. The damage causes the curve for ICAG to overshoot a little but not much and it converges where IAG and IG methods converge while starting with the damaged cluster. This damage causes the convergence to get a delayed a bit and it converges to 4 decimal places in 112 iterations, unlike 89 iterations as before.

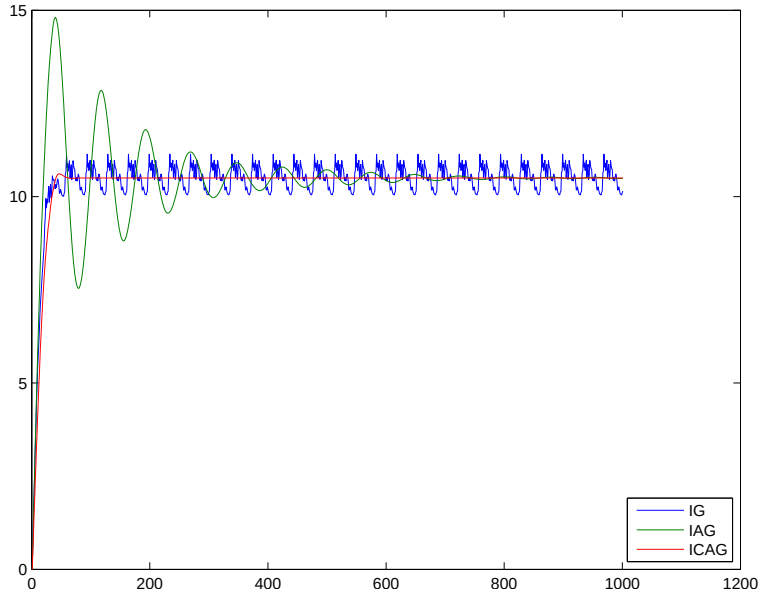


Figure 3: Illustration of data flow in a clustered network [3].

4.2 Exploitation of parallelism

Given ICAG needs approximately $\frac{1}{S}$ times less iterations than IAG with each algorithm performing with its optimal stepsize, and S being the cluster size of the wireless sensor network. We cannot directly claim $ICAG$ is S times faster, given in each iteration $ICAG$ needs to evaluate S gradients.

Table 2: Summary of Power Requirements of Various Sensors Available for Motes [2].

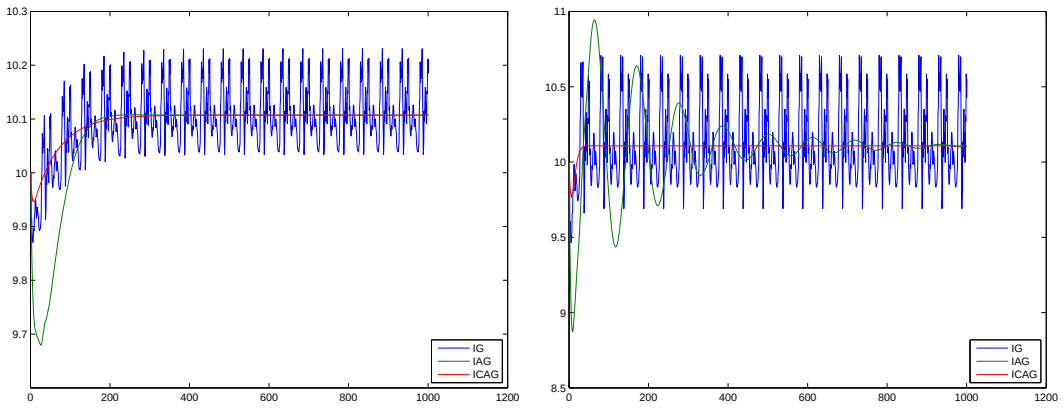
Sensor	Time per Sample (ms)	Startup Time (ms)	Current (mA)	Energy Per Sample, mJ
Weather Board Sensors				
Solar Radiation [TAOS, Inc. 2002]	500	800	0.350	.525
Barometric Pressure [Intersema 2002]	35	35	0.025	0.003
Humidity [Sensirion 2002]	333	11	.500	0.5
Surface Temp [Melexis, Inc. 2002]	0.333	2	5.6	0.0056
Ambient Temp [Melexis, Inc. 2002]	0.333	2	5.6	0.0056
Standard Mica Mote Sensors				
Accelerometer [Analog Devices, Inc.]	0.9	17	0.6	0.0048
(Passive) Thermistor [Atmel Corporation]	0.9	0	0.033	0.00009
Magnetometer [Honeywell, Inc.]	.9	17	5	.2595
Other Sensors				
Organic Byproducts[10]	.9	> 1000	5	> 15

In one iteration of the ICAG, the cluster head would need to broadcast the value of the previous iterate and aggregate gradient it recievied form another cluster to all the $S-1$ nodes, and all the nodes after evaluating their gradients would have to send the gradients to the cluster head by a form of many to one communication, and this many to one communication may be serialized if the cluster head can recieve only one message at a time. If the evaluation of gradient took negligible time, then in case of IAG for S iterations, there would be S transmissions and S recieves, but in case of ICAG for one iteration, there is 1 broadcast from the cluster head, $S-1$ reception done in parallel, and possibly $S-1$ transmissions and $S-1$ recieves done serially and then again 1 send and 1 receive done by the cluster head done to communicate with other cluster heads. The many to one communication can be done in parallel if the cluster head has multiple antennas for recieving, but we disregard such a possiblity . Thus approximately the communication overhead in S iterations of IAG is slightly lower than that of 1 iteration of ICAG. But if the nodes in a cluster communicate through routing among themselves, then one iteration of ICAG has more communication overhead than S iterations of IAG and in that case if gradient evaluation do not take considerable time compared to communication overhead then ICAG will end up being the slower of the two. So for analyzing gradient evaluation time , we look at the table 2.

Thus according to the table we can see that although the time for sensing and waking up varies greatly among the sensors, so in cases where time to sense takes a longer time compared to communication time, evaluating the gradients in parallel will be of great help because the gradient will be evaluated using the data that is sensed. Moreover the number of sensors deployed in a real world sensor netowork is large as stated before, and ranges from hundreds to tens of thousands. If the time for sensing the environment is large, in those cases, IAG may just take too much time for the results to be of any use. For example the solar radiation sensosrs take quite comparatively large time for sensing the data it is interested in, and serial sensing may just take too much time and within that time the level of radiation in the environment could change drastically to steer away the optimal solution and the achieved solution may not be useful anymore, this is assuming no sensor is damaged and the IAG algorithm continues for a reasonable time to converge.

4.3 Warm start and reclustering

Sometimes we may have a very good estimate of the optimal point, for example the pollution level of water to be measured may be very close to that what was measured yesterday. In such a case it may be desirable to start with yesterday's measured value, or say have a warm start and require less iterations. In case of IAG while initializing, the starting sensors affect initials point a lot due to the absense of contribution from other sensors in the initial steps. If the starting sensors are faulty, it can deviate the solution a lot from the warm start. Given the functions are very similar, while the points to be evaluated are far away from the optimal point, they have similar gradient, but near the optimal point, their gradients may vary a lot, thus if initialization is done near the optimal point, we will see more wiggling of the trajectory of the algorithms initially for IAG, but in case of ICAG as all the gradients within a cluster are used, this phenomenon will be subdued. Thus the initial iterates calculated by ICAG are more reliable than IAG in presence of faulty sensors. This becomes handy when the topology of the network is changed, for example reclustering is done where ICAG will



(a) Optimal μ for IAG = 0.95 (b) Optimal μ for ICAG = 4.78

Figure 4: Trajectories taken by the IG, IAG and IAG methods for the robust Fair estimation problem.

have to start again or a sensor is damaged that has to cause IAG to start again and we are likely to start with the last value that was calculated in the iteration done before the topology was changed.

In a sensor network it is desirable to do reclustering to improve energy efficiency [3]. It is expected that the new clusters will be formed using the sensors from the neighboring clusters, thus it may not be hard for the last cluster head(s) to provide the last calculated aggregate gradient(s) to the new cluster heads, and using the last iterate and aggregate gradient, a very good result can be achieved as the previous aggregate gradient will resist the initial clusters form diverting the optimal point too much, which might have occured if the clusters starting the algorithm after reclustering contanied a lot of faulty sensors.

To test this hypothesis we make the first sensor among the 50 sensors measure zero, and give them a very warm start of 10 when the optimal value is very close to 10 (10.1073). We see the trajectory of IAG greatly deviates form the optimal point compared to ICAG, thus showing that ICAG is suitable for topology change in general and facilitates reclustering.

4.4 Less complex active cycle

In a sensor network it is important to put the sensors in snooze mode or sleep mode when it does not need to work, clustering facilitates this mehtod and with ICAG the mechanism is pretty simple, as after the cluster head sends the messages to the member nodes and after the member nodes have send their respective gradients to the cluster head they can go to sleep mode. It is for the cluster head we have to go into complex details as to when it has to recieve messages from other cluster heads and when to wake up the other cluster head to whom this cluster head has to send its message. But in case of IAG this complicacy exists for all the nodes.

4.5 Conclusion and future work

We have implemented the IAG algorithm that was developed in [1], and developed another algorithm ICAG and addressed the weaknesses of IAG while trying to solve them with ICAG. IAG although has some favorable properties like that of converging, requiring only one gradient evaluation per iteration and requiring constant stepsize, thus avoiding the need of evaluationg the whole objective function or its gradient per iteration or any other experimentation needed for adapting stepsize, still this is not exactly suitable for wireless sensor networks as it lacks robustness to sensor loss, does not exploit parallellism of the network to its advantage. We addressed these issues and developed ICAG. We have not done theoretical convergence analysis and that has to be done too. And again we assumed that at each iteration L^k or the number of sensors alive is known by the cluster heads, which may not be accurate but then again this is not going to hinder the calculation much as far as L^k do not vary that much as it will only change the step size a bit and given the algorithm works for step sizes larger than the optimal one, it may not be that much of a problem. The algorithm could slightly be modified so that each cluster head passes on its known number of

alive clusters to the next one and given the number of gradient evaluations it recieves compared to the previous iteration performed in the cluster it can update the number and so on.

But further work can be done, as we assumed that the clusters in our calculation correspond to one physical cluster, it might be interesting to find out through real life experiments what happens if the clusters are formed of two or more physical clusters or even by using less than one physical cluster and thus how clusters will be formed in consideration to the fact that reclustering will be done and with a given convergence rate, the logical clustering could itself be an optimization problem.

References

[1] Doron Blatt, Alfred Hero, and Hillel Gauchman. A convergent incremental gradient method with constant step size. *SIAM J. OPTIM*, 2004.

[2] Samuel R. Madden, Michael J. Franklin, Joseph M. Hellerstein, and Wei Hong. Tinydb: An acquisitional query processing system for sensor networks. *ACM Trans. Database Syst.*, 30(1):122–173, March 2005.

[3] O. Younis, M. Krunz, and S. Ramasubramanian. Node clustering in wireless sensor networks: recent developments and deployment challenges. *Network, IEEE*, 20(3):20–25, May 2006.