Isolines: Energy-efficient Mapping in Sensor Networks

Ignacio Solis and Katia Obraczka
{isolis, katia}@cse.ucsc.edu
Computer Engineering Department
University of California, Santa Cruz

April 15, 2005

Abstract

This paper introduces a novel energy efficient data aggregation algorithm that targets spatially correlated data in sensor networks. *Isolines* aggregation works by detecting *isolines* which are the lines in a contour map. Energy efficiency is achieved by having only the nodes that detect the isoline report to the sink. Simulation results show that *isoline* aggregation can lead to significant energy savings (some scenarios reported that no aggregation can send close to 150% more bytes than *isolines* aggregation) with adequate data accuracy. We also compared *isolines* against *polygon* aggregation, our implementation of an approach representing existing spatially-correlated data aggregation mechanisms. Our results report that *isolines* exhibit higher accuracy with a slight advantage in energy efficiency.

1 Introduction

In-network aggregation has been employed quite successfully as an effective energy savings technique in power-constrained, data-driven sensor networks. The main idea behind in-network aggregation is to process data as it flows from sensor nodes to information sinks.

A number of existing aggregation algorithms focus on temporally-correlated data. Our own *cascading timers* [1], *directed diffusion* [2], and some of the aggregation mechanisms proposed by Tiny Aggregation (TAG) [3] process sensor data as it is produced periodically and flows over a data collection tree rooted at the sink and spanning all relevant sensors. Cascade timers show that the timing model used by the aggregation algorithms is critical to achieve energy efficiency without sacrificing delay.

In this paper, we describe a novel aggregation technique that targets spatially-correlated data. In particular, we address applications that are continuously monitoring varying conditions of a given geographic region (e.g., temperature, rain fall, radiation, etc.) and, as a result, generate a "contour map" of the sensed variable.

The proposed algorithm takes advantage of the spatial correlation of data in these monitoring scenarios. Energy efficiency is achieved by, instead of having all nodes send their readings to the sink, having only a few nodes report to the sink. Ideally, only the nodes with important information will report. Our approach defines the important information to be the isolines of a map.

Isolines are basically *isopleths* (from the Greek *iso* - same and *pleth* - value), a line composed of points of the same value. When these lines are drawn on a map we get a contour map, like the one shown in Figure 1. Areas encompassed by isolines lie within a certain value range and we call them isoclusters.

For instance, if we were to generate a contour temperature map, an isotherm, the temperature ranges are defined and then nodes would be grouped into areas that exhibit temperatures within the defined ranges. To construct an isotherm we do not need to collect data from all the nodes in the region being monitored; it is sufficient to collect the isolines, draw them on our map and "color in" the correspond-

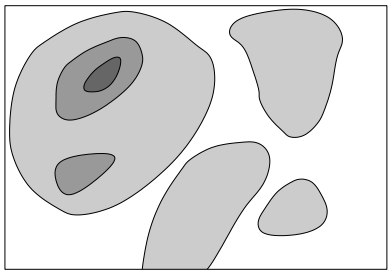

Figure 1: An isograph

ing areas. This is how energy efficiency is attained.

Through extensive simulations, we evaluate *isocluster* aggregation and compare it against no aggregation and our implementation of aggregation using polygons. *Polygon* aggregation is the approach used by both e-Scan [4] and *isobars* [5], both of which are discussed in more detail in Section 2 below. Our results show that *isoclusters* can achieve significant energy savings when compared to no aggregation while yielding adequate data accuracy. It outperforms *polygon* aggregation in terms of accuracy with slightly higher energy savings.

The remainder of this paper is structured as follows. We discuss related work in Section 2 and describe *isocluster* aggregation in Section 3. Sections 4 and 5 present our experimental methodology and evaluation results, respectively. In Section 6, we present our concluding remarks as well as directions for future work.

2 Related Work and Previous Approaches

There are mainly two other approaches that target spatially-correlated data aggregation, namely *eScan* [4] and *isobars* [5]. *eScan* focuses on monitoring the sensor network itself, in particular the remaining energy in the nodes. It queries sensing nodes which, in turn, report their remaining energy. This is done via a data collection tree established at query propagation time. When the data is being reported back to the sink, nodes aggregate the information as it flows. Aggregation is done by grouping readings that meet a certain criteria. For a set of readings to be added they need to be geographically adjacent and they need to be in the same value range.

Data is aggregated into polygons of similar value and represented by the corresponding polygon's coordinates. This approach has a few drawbacks. For one, the aggregation is done as the data flows down the collection tree. This is not always the most efficient way. If two nodes close by are in the same value range but are in different branches of the tree, their values is not aggregated until they reach a common ancestor, which might not be nearby. Also, for a node to aggregate the coordinates of a polygon it needs to know the exact location of the nodes. Assuming geographic location encoding requires more bytes than node identifiers, propagating location information results in significant additional overhead.

In contrast, *isoclusters* employ localized aggregation by detecting *isolines* and neighboring cluster members. Hence, aggregating down the path is not necessary. Location information is only needed at the sink and can be collected once.

The *isobar* mapping approach is part of the the advanced aggregation techniques proposed in TAG [5]. Here nodes are part of a grid. A node's location is based on its position on the grid. Data is collected by aggregating into polygons of nodes with similar readings. On a heavily populated grid, aggregation yields good results. If the grid is sparse, or if packets are dropped, or if energy efficient is favored over accuracy, *bounding boxes* are used for defining the polygons. A bounding box is created around an area to be aggregated. Cuts are then made to the bounding box to approximate the shape of the polygon. The more cuts, the more data that needs to be reported and the better the accuracy. Less cuts mean decreased accuracy, but less data needs to be sent, improving energy efficiency.

Isobar mapping suffers from similar problems as eScan since it also performs aggregation as the data flows towards the sink. Node location is represented by the corresponding grid coordinates minimizing the need to transmit real location information

Both *eScan* and *isobars* are based on using polygons to aggregate data of similar value produced by neighboring nodes. We will compare the perfor-

mance of *isoclusters* against (1) no aggregation and (2) *polygon* aggregation, our implementation of the aggregation mechanism underlying both *eScan* and *isobars*.

3 Isolines

The goal of *isoline* aggregation is to optimize data collection by reducing redundant transmissions based only on local information. It uses the concept of *isolines* (or *isopleths*), i.e., lines of the same value, which are used to represent information in contour maps. The basic idea is that nodes will only report to the sink if the detect an isoline; otherwise, no report is generated. Isolines are detected based on neighborhood information gathered through a neighbor-to-neighbor protocol.

The neighbor-to-neighbor protocol, or NNP, exchanges sensed value between nodes. It uses a basic push approach. Nodes decide when they need to communicate their sensed information to their neighbors. This happens when the node is started and when data changes cause an isoline to appear or disappear. First-time reports (i.e., when nodes start), allow the network to detect new nodes quickly. Periodic transmission serves to refresh state maintained by neighboring nodes. Reports are also generated upon significant changes in sensed data.

Isoline detection is a very simple yet elegant method of collecting information efficiently to draw contour maps: a node compares its reading with the reading of all neighboring nodes. If the readings lie in different sides of an isoline, then a report needs to be generated. For example, if the isolines measure multiples of 10, then a node with a sensed value of 35 with a neighbor whose value is 42 is able to detect that there is (at least) one isoline of value 40 passing between itself and said neighbor.

Once the existence of an isoline has been determined, it needs to be reported to the data collection sink. Reporting an isoline consists of sending the node's sensed value and the value of the neighboring node across the isoline to the sink. This could be optimized by reporting only the value of the isoline, saving a few extra bytes, but loosing some accuracy.

The detection of the isoline is symmetric, i.e., both the node and its neighbor detect it. The node who reports the isoline is the one closest to the sink (according to hop count). If both nodes are at the same distance, the node with the lowest reading will send the message.

Nodes will only report to the sink when there are isolines around. This might lead to problems since we will assume that on the absence of report no isoline exists. To handle failures, we also implement probabilistic reporting so nodes broadcast their information periodically even when they do not need to do so. This also helps on the accuracy of the map generated and can be tuned if necessary.

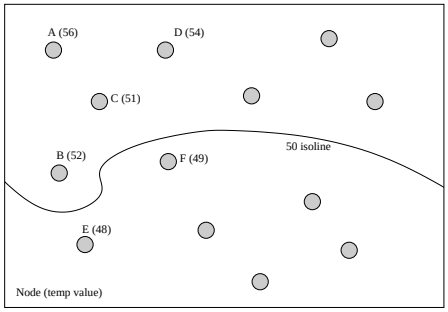

Figure 2: A temperature isoline

Nodes that are within the same isoline are said to belong to the same *isocluster*. Figure 2 depicts a temperature isoline where temperatures within 10-degree ranges belong to the same isocluster. Nodes *A* and *C* are part of the same isocluster and will know, upon exchanging sensed values, that an isoline does not go between them. On the other hand, nodes *B* whose value is 52 and *E* who is measuring a temperature of 48 will detect the 50-degree isoline when they compare values.

4 Evaluation

In our experiments, we use two different sensor network deployments, namely: (1) a 400X400m field monitored by a 16X16 sensor grid evenly spaced at 25m intervals, and (2) a 800X800 field using 32X32 grid. A snapshot of the data being sensed is generated and temperature is used as an example of the

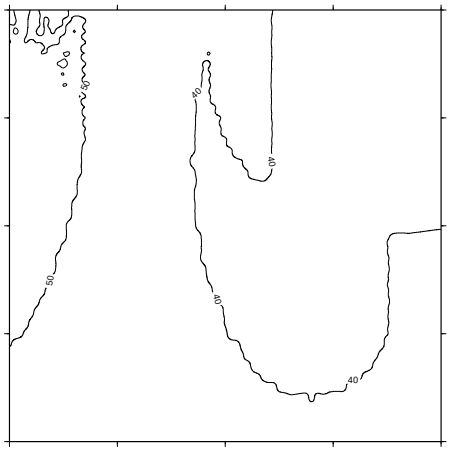

Figure 3: Real map

information being reported by the sensor network.

We should point out that, although, in these experiments, nodes are placed according to a grid pattern, *isoline* aggregation is not specific to grid placement. For random node placement, more careful determination of what nodes report is needed to avoid unnecessary redundancy. Future work will address this issue specifically.

Our implementation of *polygon* aggregation is as follows. When a node receives data from its children (according to the data distribution tree rooted at the sink), it aggregates it into polygons and sends only the vertices.

All nodes in the polygon are assigned the average value of the polygon's range; for example, in the case of a [40-50) temperature range, nodes take the value of 45. Similarly to *isobar* aggregation [5], we use the sensors' grid coordinates as their locations. Reports from nodes may contain multiple polygons if multiple value ranges have been aggregated. *Polygon* aggregation uses the PolyBoolean library [6]. In the no aggregation case, nodes just send their local information to the sink.

For efficiency purposes, we use cascading timers [1] as the timing model as it allows nodes to wait for their children to report without increasing the delay of the data collection. For both *polygon* and *isoline* aggregation, data is grouped in ranges of 10, that is, from [0-10),[10-20), etc.

As the experimental platform, we employ the ns2 [7] network simulator. For medium access control, nodes use CSMA at 115Kbps. FLIP [8] was used as our network protocol. The experiment tries to reproduce the map in Figure 3, which represents reality. This map is generated using 40X40 values. Node identifiers and location are 2 bytes long. Temperature information is also 2 bytes.

The sink node, which is placed at the center of the map, starts by broadcasting a query for the map at time 1s. From time 3s to 4s, nodes report their temperature readings. The simulation is stopped at time 5s. The transmission range of nodes was set to 40m. The data points used to compute the tabulated results in Section 5 are obtained by averaging over 10 runs.

5 Results

Even though figure 3 defines reality, we will use Figure 4 as our idealized map to get performance bounds. The map in Figure 4 is generated when no aggregation is used, i.e., all the nodes are reporting their readings. This means that we have all the information we can possibly have; the only way to be more accurate is by deploying more nodes increasing sensor density.

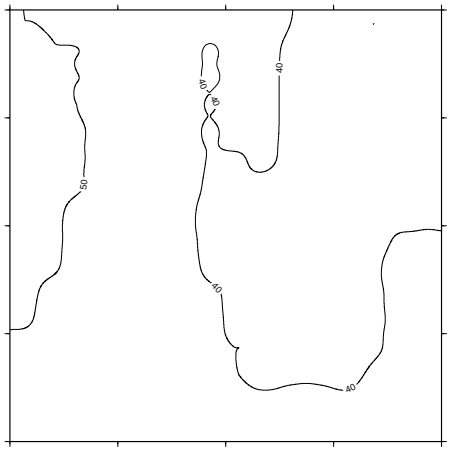

Figure 4: A map using all sensor readings

Our main goal is to achieve similar accuracy to the map obtained by not aggregating data, while making the collection process energy-efficient. Figures 5

and 6 present the maps generated using *isoline* aggregation, while Figures 7 and 8 were obtained using *polygon* aggregation. Figures 6 and 8 present the same maps as Figures 5 and 7, respectively, superimposed atop points representing the readings actually received at the sink.

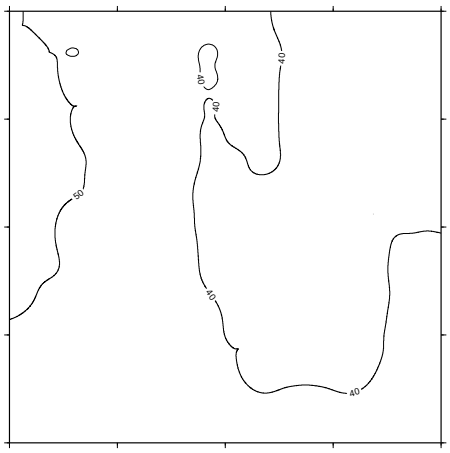

Figure 5: Map generated with *isoline* aggregation

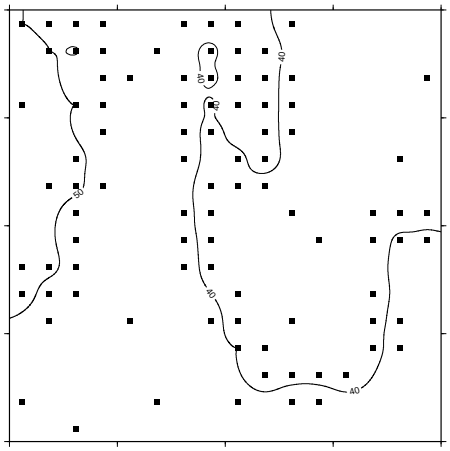

Figure 6: Map generated with *isoline* aggregation plus reporting sensors

Graphing the results obtained from the simulations helps us visualize how the aggregation algorithms perform. We should point out that the graphing tools we use, which interpolate the data points to generate the map, have not been optimized using the assumptions that be can made in the specific case of aggregation algorithms. For example, in the case of *isolines*, if there are no readings received from an area, then it is an indication that an isoline does not exist. In the case of *polygon* aggregation, reported areas should be graphed as polygons. Moreover, some of the anomalies in the graphs are caused by lost packets. However, since we do not insert artificial drops, the effects are not very evident. Packet losses can still occur caused, for example, by collisions.

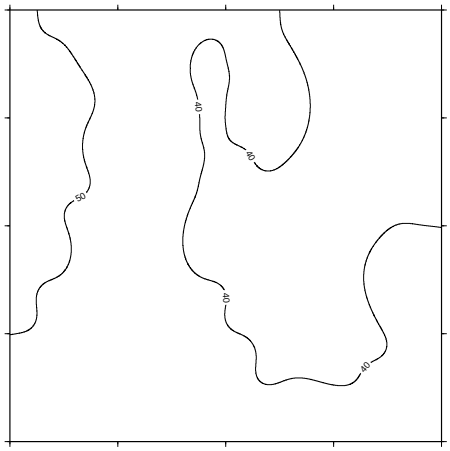

Figure 7: Map generated with *polygon* aggregation

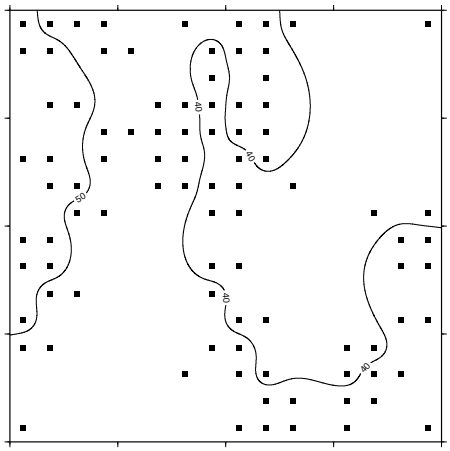

Figure 8: Map generated with *polygon* aggregation plus reporting sensors

In order to quantify how similar the maps generated by the different aggregation approaches are, we compute the average distance between corresponding points obtained from no aggregation, *isoline*, and *polygon* aggregation. We used the `ngmath` part of the NCAR Graphics Library[9] to interpolate the

40X40 reality data points. Tables 1 and 2 summarize these results.

	Reality (degrees)	No agg. (%)
No Agg.	1.21 (sd=0.01)	0
Isolines	1.59 (sd=0.12)	31.4%
Polygons	2.98 (sd=0.18)	146.3%

Table 1: Map similarity for the 16X16 sensor field

	Reality (degrees)	No agg. (%)
No Agg.	1.24 (sd=0.08)	0
Isolines	1.83 (sd=0.24)	47.6%
Polygons	3.48 (sd=0.36)	180.6%

Table 2: Map similarity for the 32X32 sensor field

The first column shows the average difference and corresponding standard deviation between each aggregation algorithm and the reality. For no aggregation, the map obtained when all nodes report differs on average by 1.21 degrees (with a standard deviation of 0.01) in the 16X16 sensor deployment. Similarly, the average difference between *isolines* and reality and *polygons* and reality is 1.59 and 2.98, respectively.

Since the best any algorithm can perform is when all nodes report their readings and no aggregation is performed. This won't obtain a "perfect score" because we can't sample at infinite density. Hence we use no aggregation as the performance upper bound and compare to it in column 2. In the 16X16 map, isoline aggregation yields a difference of only 31%, while the map generated by *polygon* aggregation differs by 146%, about 5 times more than *isolines*. For the 32X32 map, with four times the sensors, we observe 48% and 181% difference, respectively.

On a first analysis, these differences may seem too high. However, when translating them into actual sensed data, they fall into perspective. For example, a point that has the value of 43 degrees might get mapped to 44.6 with *isolines* and 46 with *polygon* aggregation. Note that both of these algorithms are not trying to map reality on a point-to-point basis. Instead, they try to aggregate data by doing this 'lossy compression' into groups of values. To quantify this, we define the *group similarity* metric which calculates how many of the points are mapped into the correct value group. It basically measures if the contours look the same. Table 3 presents these results for both 16X16 and 32X32 sensor fields.

Reality	16X16	32X32
No Agg.	95.0% (sd=0.1)	93.3% (sd=1.1)
Isolines	93.3% (sd=0.6)	91.8% (sd=1.7)
Polygons	92.5% (sd=3.9)	85.8% (sd=4.9)

Table 3: Group similarity

We observe that both algorithms perform reasonably according to this metric, with *isolines* exhibiting better performance than *polygons*, especially in the 32X32 sensor field scenario. The *group similarity* metric tries to express the similarity between the contour maps. For example, Figures 3 (reality) and 4 (no aggregation) are 95% similar. These results also show that larger fields (with the same sensor density) are harder to map. This is particularly true for *polygon* aggregation, whose performance degrades as the network size grows.

Reality	Small	Large
No agg	13826 (sd=340)	80635 (sd=1221)
Isolines	7897 (sd=384)	32549 (sd=1095)
Polygons	8311 (sd=263)	34200 (sd=918)

Table 4: Bytes sent as energy efficiency.

Recall that the main goal of data aggregation is to achieve energy efficiency by transmitting less information. Table 4 shows the number of bytes sent by all three approaches. We observe that no aggregation transmits 75%- and 148% more data than *isolines* in the 16X16 and 32X32 sensor field scenarios, respectively. With no aggregation, every node needs to transmit its information to the sink which may results in redundant data traveling multiple hops, wasting precious resources along the way. Both *isoline* and *polygon* aggregation try to reduce the number of transmissions, minimizing data redundancy by aggregating spatially correlated information.

We should also point out that *isoline* aggregation was also designed with temporal aggregation in mind. That is, over time, nodes broadcast information only when local readings change (i.e., they only need local knowledge). Since we simulated a single data collection snapshot, our results do not showcase

this feature of *isoline* aggregation.

Another observation is that no aggregation does not scale. As the number of nodes increases, forwarding packets from every node to the sink becomes prohibitively expensive, especially in power-constrained environments. This is especially true when larger areas report very similar values. To capture unevenly distributed values more accurately, the range of the isoclusters could be tuned accordingly, for example, report isolines every 5 degrees.

The disadvantage of using *polygon*-based aggregation lies in the fact that a node cannot make a localized decision of whether or not its information is redundant. Aggregation is limited to nodes on the same branch of a tree and only happens when a shared ancestor exists. If this ancestor node is far away, redundant information will propagated closer to the sink.

6 Conclusions and Future Work

In this paper, we introduced a novel data aggregation algorithm that targets spatially correlated data. *Isoline* aggregation uses local information from neighbors to group nodes that report similar readings. Energy efficiency is achieved by having only a subset of the nodes, i.e., the ones next to the isolines, report to the sink.

Simulation results show that *isoline* aggregation can lead to significant energy savings (some scenarios reported that no aggregation can send close to 150% more bytes than *isoline* aggregation) with adequate data accuracy. We also compared *isolines* against *polygon* aggregation, our implementation of an approach representing existing spatially-correlated data aggregation mechanisms (e.g., *eScan* and *isobars*). Our results report that *isolines* exhibit higher accuracy with a slight advantage in energy efficiency.

As future work, we plan to further study how network density affects our readings. This is directly related to grid placement and the selection of which nodes need to report. We are also investigating mechanisms for continuously monitoring the sensor network and performing temporal aggregation without sacrificing accuracy.

References

[1] I. Solis and K. Obraczka, "The impact of timing in data aggregation for sensor networks," *Proceedings of the IEEE International Conference on Communications (ICC)*, June 2004.

[2] C. Intanagonwiwat, R. Govindan and D. Estrin, "Directed diffusion: A scalable and robust communication paradigm for sensor networks," in *Proceedings of the International Conference on Mobile Computing and Networking (MobiCom)*. ACM, August 2000.

[3] S.R. Madden, M.J. Franklin, J.M. Hellerstein, and W. Hong, "TAG: a tiny aggregation service for ad-hoc sensor networks," in *Proceedings of the Symposium on Operating Systems Design and Implementation, OSDI*, December 2002.

[4] J. Zhao, R. Govindan and D. Estrin, "Residual energy scans for monitoring wireless sensor networks," in *Proceedings of the IEEE Wilress Communications and Networking Conference (WCNC'02)*, March 2002, pp. 17–21.

[5] J. M. Hellerstein, W. Hong, S. Madden and K. Stanek, "Beyond average: Towards sophisticated sensing with queries," in *Proceedings of the 2nd International Workshop on Information Processing in Sensor Networks (IPSN '03)*, March 2003.

[6] M. Leonov, "PolyBoolean Library," February 2004, http://www.complex-a5.ru/polyboolean/.

[7] VINT, "The Network Simulator NS-2," http://www.isi.edu/nsnam, November 2001.

[8] I. Solis and K. Obraczka, "FLIP: A flexible interconnection protocol for heterogeneous internetworking," *ACM/Kluwer Mobile Networking and Applications (MONET) Special on Integration of Heterogeneous Wireless Technologies*, 2004.

[9] National Center for Atmospheric Research, "NCAR Graphics Library," February 2004, http://ngwww.ucar.edu/ng4.3/.