2012 IEEE Sixth International Conference on Self-Adaptive and Self-Organizing Systems Workshops

Analysis of new gradient based aggregation algorithms for data-propagation in mobile networks

Jose Luis Fernandez-Marquez,
Akla-Esso Tchao, Giovanna Di Marzo Serugendo
University of Geneva, ISS
Geneva, Switzerland
Email: Joseluis.fernandez,
Akla-Esso.Tchao, Giovanna.DiMarzo@unige.ch

Graeme Stevenson, Juan Ye,
Simon Dobson
University of St Andrews,
St Andrews, UK
Email: graeme.stevenson, juan.ye,
simon.dobson@st-andrews.ac.uk

***Abstract*—In large scale networks, agents must use partial knowledge obtained from local interactions to reason about their environment. They require efficient mechanisms to allow them to retrieve and aggregate information beyond their communication range. Even though proposals have been presented for gathering information in large scale wireless sensor networks, it is still a challenge to find an efficient and robust technique for gathering information in large scale mobile wireless networks. In this paper we propose gradients as a multi-path structure for routing and aggregating information across a network of computational mobile nodes.**

We use simulation to demonstrate that progressive aggregation done on top of a gradient improves the bandwidth usage and memory consumption. We also demonstrate self-* properties of our proposed algorithms including scalability, robustness and adaptability.

***Keywords*-Gradients; Aggregation; Distributed Networks;**

I. Introduction

The proliferation of wireless, often mobile devices, such as sensors, smart phones, tablet computers, and public displays offers untapped potential as an ad-hoc communication and computation infrastructure. Such a network enables a new class of application—one focused on highly dynamic, localised, frequent update, real-time interactions that are difficult to implement via a traditional centralised approach. Such applications are designed specifically with local interactions and awareness of partial knowledge at the fore.

Consider an application to optimising the travel arrangements for tens of thousands of people, distributed across a city, in real-time. Such an application would rely on up to the minute information about bus, train and metro location, the origin and intended destinations of citizens, traffic accidents and areas of congestion, much of this information lying beyond the host device's communication range.

In this paper we propose the gradient [1] as a structure for routing and progressively aggregating the information. Progressive aggregation of information in a network is not a new idea, it has been applied for large scale sensor networks on top of hierarchical structures, such as trees and clusters. However, these hierarchical structures present failure tolerance problems, and are difficult to maintain in mobile networks. Multi-path structures have been proposed to overcome this. In multi-path approaches, the information is sent over multiple paths at once, producing duplicates of the information while increasing robustness. Indeed, in a multi-path structure even if a connection between two nodes breaks, information is simultaneously sent along auxiliary paths. This is potentially useful in Mobile Ad-Hoc Networks, where the topology of the network is changing continuously due to their dynamic nature. However, simultaneous use of multiple paths increases the complexity of these algorithms and the creation of duplicates makes it potentially difficult to aggregate information with 100% of accuracy.

Gradients provide a complete multi-path structure, i.e., all possible paths from any node to the sink. However, unlike multi-path approaches, the gradient contains a notion of distance and direction to the node that originated the gradient. Consequently, the information is naturally routed along the shortest path created by the gradient and the information is not duplicated when it comes back to the sink. Thus, gradients produce a multi-path structure exploited using unicast communication, and using the alternative paths only in case of node failures or topological network changes. That makes gradient a good candidate for gathering information in mobile networks.

Even though gradients have been used for creating spatial structure in sensor networks, their use in mobile networks as basis for progressive aggregation is still an open issue. As far as we know, there is not any work proposing gradients as a structure for gathering and progressively aggregating data in large scale mobile networks.

This paper analyses gradients as a structure for routing and aggregating information in large mobile wireless networks. Mainly, we analyse the feasibility, performance and robustness of progressive aggregation on top of gradients.

The paper is structured as follows: the next section summarises related works. The proposed technique is introduced in Section III. Section IV describes the different aggregation algorithms. In Section V we evaluate the contribution of progressive aggregation on top of gradient. Finally, we conclude and present future works in Section VI.

II. Related work

Since the introduction of sensor networks (SN) different techniques have been proposed in order to gather and aggregate information from sensors. Nowadays, these techniques are not only required for SN but by other domains, including pervasive computing, amorphous computing, ubiquitous computing, and the Internet of Things, in order to obtain meaningful information from distributed systems, increasing agents' knowledge, and allowing them to reason and take decisions.

The state of the art of progressive data aggregation (in-network aggregation) algorithms for distributed systems has been recently settled in [2]. This survey classifies the existing networking protocols and hierarchies for progressive aggregation in three different groups: Tree-based, cluster based, and multi-path approaches.

In a tree-based approach a tree is first constructed at the sink. The requested information is sent through the tree and aggregated at intermediary nodes, that act as aggregators and routers. This approach is suitable for performing optimal and very efficient aggregation functions, however, it presents a potentially high cost for maintaining the hierarchical structure (i.e. the trees) and scarce robustness in case of link/device failures. Some existing proposals are: TAG [3], Directed Diffusion [4], and PEGASIS [5].

Analogously to tree-based approaches, cluster-based approaches build hierarchical structures, where nodes are subdivided into clusters. For each cluster, a node is responsible of aggregating and routing the information to upper levels of the structure. In dynamic environment the structure should be maintained, involving a potentially high cost. Main proposals in the literature that implement cluster-based approaches are: LEACH [6] and COUGAR [7].

In order to increase the robustness of previous approaches, multi-path approaches (e.g. Synopsis Diffusion [8]) extend the tree-based approaches sending the information to more than one single parent. This information is duplicated, and reaches the source from many different paths. Obviously, the increment of robustness is paid with an increment of the resource usage.

In this paper we propose an in-network aggregation on top of gradients. Gradients are multi-path structures that convey the notion of direction and distance to the originator of the gradient as defined in [1]. Even though gradients have been used for creating hierarchical structures, such as, trees, as far as we know, there is not any approach that exploits the multi-path capabilities of gradients for implementing progressive aggregation. A difference with tree and cluster based approaches, gradients provide all the possible paths from the data sources to the sink, being potentially more tolerant to node/link failures. Moreover, even though gradients provide a multi-path structure we exploited them using unicast. Thus, we do not duplicate information as multi-path approaches do. In our case, the redundant paths are used in case of node/link failures, producing a fast repair of the gradient structure.

III. Requesting information via gradients

Analogously to other algorithms, such as, TAG [3], we consider two phases, the distribution phase and the collection phase. To allow information to be requested and routed to the source node (i.e., the node from where the information query is sent), we propose the use of gradients as a multi-path structure created during the distribution phase.

The gradient is a multi-path structure that provides an additional information about the senders distance and direction, usually the number of hops.

Gradients were initially proposed as one of the amorphous computing paradigms [9] to estimate distances from each node to the source node. Different approaches have been presented for mobile networks [10], [11]. In this paper we assume a basic implementation of gradients [1] and we focus on how the gradient can be exploited for gathering and aggregating information in an efficient way.

The aggregation algorithms proposed in this paper are then built on top of the gradients and independent from the gradient implementation itself.

Initially we consider a network where each node is locally connected, have a local knowledge of the system, and stores a value. The different steps of the proposed algorithm are:

1) A gradient is originally created by an agent (situated in a given source node) that requests an information (i.e., MIN, MAX, AVG of the values stored in nodes). The gradient contains the distance to the source, the information requested and how it must be aggregated (e.g., a pair $(humiditySensorValue, MIN)$ is a request that should provide the minimum humidity sensor value).
2) Once the gradient is spread among the nodes, if an agent at one of the nodes can answer the request, it sends the reply value following the gradient back to the node from where the gradient was initially created.
3) When two or more reply values arrive to the same node, they are aggregated using the function specified at the gradient creation time, and the aggregated result is then sent back following the gradient.
4) Finally the agent that initially requested the information receives the desired aggregated value.

IV. Aggregation Algorithms

In this section, we present the design of several different distributed aggregation functions that can be used to calculate the sum, average, mode and max/min of a set of data. Each differs in two respects: (1) the actual reply values communicated between the nodes, (2) the function that is applied when two or more values (i.e., inputs) are co-located within the same node. Moreover, we present a couple of application scenario for each aggregation function.

A. Sum

Following the algorithm steps proposed above, when two reply values arrive at the same node they are summed, and the result is sent back following the gradient.

The function is executed when two or more pieces of information arrive at the same node are:

$$f_{sum}(a_1, a_2, \ldots, a_n) = (a_1 + a_2 + \ldots + a_n)$$

where a_n is the numeric value and n is the number of aggregated values. The aggregation result is sent following the gradient.

In the SUM algorithm the information sent among the node is a simple value. Example application where the SUM algorithm is needed are: (1) An agent wants to know the number of nodes in the system (e.g., a number of active sensors in a sensor network, or mobile phones in a given city), and (2) Given a sensor network, an agent wants to know the total battery power available in the system.

B. Average

To calculate the average, each reply value contains: i) the numeric value, and ii) a counter that indicates the number of times that this numeric value has been aggregated (i.e., summed in this case). Thus, each node initially has a pair (a, b), where a is the value and b is a counter initialised to 1. When two or more reply values arrive to the same host, they are aggregated according to the following rule:

$$f_{avg}((a_1, b_1), (a_2, b_2), \ldots (a_n, b_n)) = (\sum_{i=1}^{n} a_i, \sum_{i=1}^{n} b_i)$$

Finally the host from where the average was requested receives a pair (a, b), and the agent then calculates the average as: $avg = a/b$.

Examples of applications where the AVG algorithm is needed are: (1) Given a sensor network, an agent wants to know the average battery available at each sensor, or the average of a number of sensor measurements, and (2) Given a mobile network composed of cars, an agent wants to know the average car speed.

C. Mode

In the mode case, reply values consist of a list of pairs, with each pair representing: a unique number value in the list and the number of times that specific value has been already encountered while it is being propagated along the gradient. Thus, the aggregation rules that produce the mode are defined as follows:

$$f_m((a_1, b_1), (a_2, b_2)) = \begin{cases} (a_1, b_1), (a_2, b_2) & \text{if} \quad a_1 \neq a_2 \\ (a_1, b_1 + b_2) & \text{if} \quad a_1 = a_2 \end{cases}$$

The agent that initially requested the information will eventually receive a list of pairs, each containing different values encountered in the system and the number of times each of them has been encountered.

Examples of applications of the MODE algorithm are: (1) given a network composed of mobile phones or PDAs, an agent wants to know what is the best date to celebrate an event (i.e., most voted day is the best date), and (2) given a phone application, an agent wants to know the most frequently chosen setting among users.

D. Max/Min

When two or more reply values arrive at the same host, the MAX/MIN aggregation is processed according to the following function:

$$f_{max}(a_1, a_2, \ldots, a_n) = a_i \ s.t. \ a_i \in \{a_1, a_2, \ldots, a_n\} \text{ and } a_i \geq a_j, \forall j \in \{1, \ldots, n\}$$

Examples of applications where the MAX/MIN algorithm is needed are: (1) Given a mobile network composed of cars, an agent wants to know the maximum speed achieved in one city, and (2) Given a sensor network, an agent wants to know the maximum measured value (e.g., maximum temperature in one city).

A major aspect of these algorithms is that each node sends each value only one time. The algorithms are not replicating data among the nodes, thus, when a node sends a value it is removed from the sending node. This is important in order to get an accuracy of the result, even when network topological changes occur, or the gradient is not properly updated. As we show in the next section the algorithms reach 100% of accuracy for all cases.

V. Evaluation

In this section we evaluate the performance of the proposed algorithms, and analyse the contribution of the progressive aggregation on top of gradients. To measure this contribution, we compare our approach using progressive aggregation against aggregating the information on the source node (called "flat"). Progressive aggregation has been successfully applied on top of trees and clusters. However, it is not clear its contribution on top of gradients, because gradients provide many paths for routing the information, thus, reducing the probability of information converging on the same path. Through this section it is shown that progressive aggregation on top of gradient reduces the bandwidth usage and memory consumption, making gradients a potential multi-path structure for routing and aggregating information in mobile wireless networks. Moreover, we provide a detailed evaluation of the properties of our approach, such as scalability, robustness and adaptability.

The simulations are implemented using REPAST 3 [12].

Params	values
Space	1200m x 700m
hosts' distribution	Randomly uniform
Num. of dimensions	2
Comm. range	40
Hosts number	2000
Mobile hosts	false

Table I: Parameters Settings

A. Assumptions and Parameters settings

For the simulations we assume the following:

- A number of nodes large enough to prevent network fragmentation.
- A node can only communicate with node to which it is directly connected (i.e., nodes that are inside its communication range), and has only access to local information.
- The cost of messages and memory for the gradient creation is independent of the proposed algorithms. It is analysed separately in section V-C.

Across this evaluation we vary the different simulation parameters in order to analyse the different range of behaviour exhibited. Table I summarises the default parameter settings used in the simulations. Unless otherwise indicated, all the simulations use these settings, namely, 2000 nodes, uniformly spread over a bi-dimensional space (1200 x 700 meters), each with a communication range of 40 meters, and initially the nodes are static. The minimum number of nodes used in this simulation is 2000, which corresponds to an average adjacency of 11.47 nodes. Lowering this value increases the probability of segmentation, i.e., that some nodes are not connected to the network. Each node is initialised with a random value between $[1..1000]$, and each simulation result is provided as the average of the result among 50 runs.

B. Metrics

In these simulations we use two metrics: bandwidth usage, and the memory consumption. A formal definition of these metrics are as follows:

1) Bandwidth Usage: A message is sent each time a reply value is moved from one node to another. The size of one message is expressed in information units. For example, in the sum algorithm, which only sends a single numeric value among the nodes, the message size is 1 information unit. The bandwidth usage is expressed as the amount of information units sent during a given period of time.

Definition 1 (The bandwidth usage at time t): Let $msg(n,t)$ be the set of messages sent by a node n between 0 and t, and $s(m)$ the size of a given message m in information units, thus, the bandwidth used at time t over all nodes, $\mathcal{N}_t$, is given by:

$$BW(t) = \sum_{\substack{n \in \mathcal{N}_t \\ m \in msg(n,t)}} s(m)$$

2) Memory: For each simulation we measure the maximum memory used by any host in the system.

Definition 2 (Number of messages stored at time t): Let $mem(n,t)$ be the maximum number of messages stored at a node n between time 0 and t, the maximum number of messages stored at time t in the network is given by:

$$MEM(t) = \max_{n \in \mathcal{N}_t} \{mem(n,t)\}$$

Notice that for all simulations these metrics are reported when the algorithm have reached 100% of accuracy. Thus, the criterium for finishing each run is reached when the desired result is achieved.

C. Gradient cost

In this section we analyse the bandwidth usage and memory required to create the gradients. Namely, we vary the number of nodes and show the bandwidth usage and memory consumption. We observe how the bandwidth consumption increases as the number of nodes increases. This increment is a linear proportion of the number of nodes, so for 2000 nodes the system needs a bandwidth usage of 2000 information units and for a network size of 10000 nodes, the system needs a bandwidth usage of 10000 information units. Simulation results demonstrate that in order to create the gradient each node sends an average of one message.

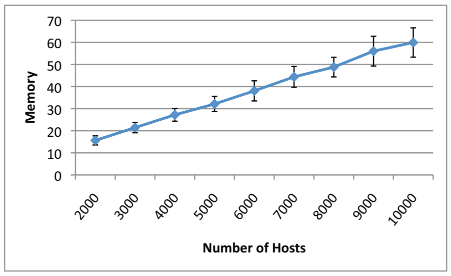

Figure 1: Gradient Cost Memory

Figure 1 shows the different memory consumption, when we vary the number of nodes. As the information is progressively aggregated (following the gradient algorithm reported in [1]), keeping only the minimum hop counter, the memory consumption is not high. Memory consumption is directly related to the number of adjacent nodes.

Notice that if the nodes are moving we need to update the gradients. The cost of update the gradient is directly related to the updating frequency. In sectionV-F we show how even when the gradient are not properly updated the increment of bandwidth usage and memory consumption is insignificant, while keeping 100% of accuracy.

D. Scalability

In this simulation we vary the number of nodes, in order to evaluate the scalability of our approach, and we compare our approach versus the flat approach (i.e. without using progressive aggregation). All the proposed aggregation functions have been compared with the flat approach. Because of space, we present only most relevant results.

By comparing Figure 2 and Figure 3, we observe that the progressive aggregation dramatically reduces the bandwidth usage among all the different number of nodes (e.g., the SUM algorithm sends around 14000 units of information per run, while SUM Flat sends an average of 140000 units of information per run).

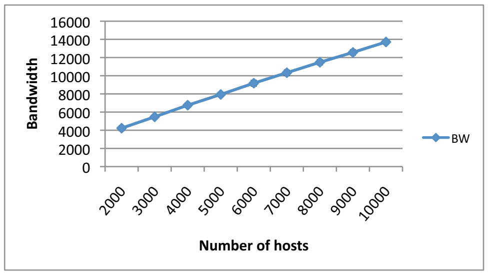

Figure 2: Scalability - SUM - Bandwidth

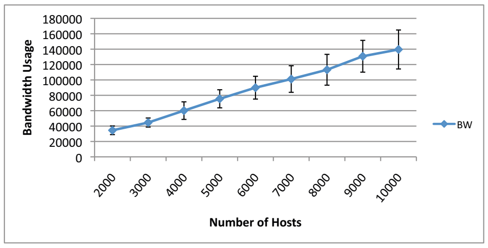

Figure 3: Scalability - SUM flat - Bandwidth

Regarding the memory consumption, Figure 4 shows the maximum memory used by our approach for the different number of nodes. Figure 5 shows the memory used by the flat approach. By comparing both Figures, we observe that the progressive aggregation (SUM in this case), reduces dramatically the maximum memory consumption. In our approach the maximum memory consumption is directly related to the number of adjacent hosts (i.e., number of neighbouring hosts), thus, the node with the highest connectivity is the one that uses more memory, since in the worst case it will receive as many information unit as its number of neighbours. For the memory consumption in the flat approach the maximum memory is directly related with the number of hosts in the system. Since, each host sends the information to the host that requires the information, and it is aggregated there.

Similar results are achieved for the other aggregation functions (i.e., AVG and MAX/MIN). The worst performance is achieved in the MODE aggregation function. Here, our approach presents similar bandwidth consumption as the flat approach. The reason is that we are calculating the MODE of values between [1..1000], and the aggregation only happens when two similar values are in the same host. However, if we implement the MODE for a voting system where

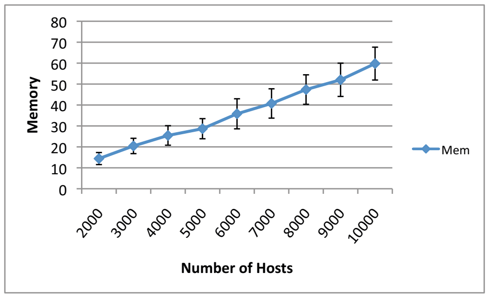

Figure 4: Scalability - SUM - Memory

the number of possibles values is a range between [1..10], the aggregation would happen more frequently, reducing dramatically the bandwidth usage and memory consumption.

This simulations show that progressive aggregation can be done on top of gradients, reducing the bandwidth usage and memory consumption, in spite of the existence of many different paths for reaching the source.

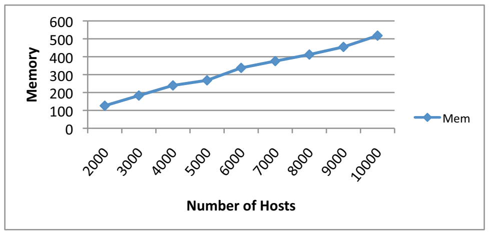

Figure 5: Scalability - SUM flat - Memory

E. Robustness

In this section we evaluate the robustness of our proposed aggregation algorithms, when the nodes values are subject to noise. To evaluate the noise tolerance of our approach, the value of each node is subject to noise. In this simulation we set the number of nodes to 3000 and we vary the noise factor. The noise is applied as follows:

$$noise(value) = value + ((2 * \theta * \gamma) - \gamma) \qquad (1)$$

where θ generates a uniform random number between [0..1] and γ is the noise factor.

Simulations show how the error produced by the noise in the SUM aggregation function remains constant for all the different noise factors. In general, the proposed algorithms behave similarly to a centralised approach in front of noise. Thus, the tolerance of the proposed aggregation algorithms depends on the mathematical properties of the aggregation function and not on the algorithm itself. Moreover, we observe that the bandwidth and memory consumption remain the same even in presence of noise.

Simulation results show that our approach is tolerance to intermittent communication failures, without a significant increment of the bandwidth usage nor memory consumption.

F. Adaptability

In this section we evaluate the performance of our proposal when the network is composed of mobile nodes. To deal with mobile nodes we implement the SUM algorithm on top of active gradients. Active gradients [1], [10], [11] are periodically updated in order to adapt the gradients' values to network topological changes. The main goal of this simulation is to evaluate how sensitive our approach is to gradients that are not properly updated. Thus, this simulation varies the frequency of updating the gradient and evaluates the bandwidth usage and memory costs.

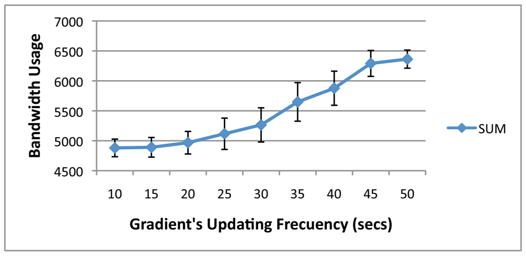

Figure 6: SUM - Active Gradients - BW

The nodes follow a Random Way Point mobility pattern. We assume that nodes are moving at a constant speed (1 meter/sec), the gradients are updated every 10s to 50s.

Figure 6 shows the bandwidth usage for different updating frequencies. Notice that the SUM algorithm is able get 100% of accuracy even though the gradient is not properly updated. Moreover, the increment of bandwidth usage is not significant. This small increment in the bandwidth usage occurs because the information follows wrong paths due to no updated gradient values. Regarding the memory consumption, simulations results show that the memory consumption does not change when nodes are moving, even if the gradients are not properly updated.

VI. Conclusions

This paper proposes a gradient based aggregation algorithm for distributed systems. This approach allows to increment the agents' knowledge by gathering and collaboratively and progressively aggregating information from the system. Agents can request information that come beyond their communication range, reply values are routed following the gradients and progressively aggregated at each node. As far as we know, the progressive aggregation has only been applied on top of tree, cluster, or multi-path structures, which are not or only in a limited form fault tolerant (i.e.,multi-path approaches have been applied to sensor networks, but it is not clear their feasibility in mobile networks), and never on top of gradients.

Even though, gradients have been used for the construction of tree structures. The combination of this progressive aggregation with gradient provides an alternative for dealing with mobile wireless networks, such as, pervasive computing, or ubiquitous computing.

Simulation results show that the progressive aggregation can be done on top a gradient reducing dramatically the bandwidth usage and memory consumption. Additionally, we analysed the behaviour of the proposed algorithm dealing with noise and host failures, demonstrating high levels of robustness against noise and failure tolerance.

In future work we plan to compare our gradient based approach against other existing approaches, namely, Directed Diffusion [4] in order to demonstrate its contribution in both, mobile (e.g. pervasive computing) and static networks (e.g. sensor networks).

Acknowledgment

Work supported by the EU SAPERE Project under contract no.256873

References

[1] J. L. Fernandez-Marquez, G. Di Marzo Serugendo, S. Montagna, M. Viroli, and J. L. Arcos, "Description and composition of bio-inspired design patterns: a complete overview," *Natural Computing*, pp. 1–25, 2012.

[2] E. Fasolo, M. Rossi, J. Widmer, and M. Zorzi, "In-network aggregation techniques for wireless sensor networks: a survey," *Wireless Communications*, vol. 14, pp. 70–87, 2007.

[3] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "Tag: a tiny aggregation service for ad-hoc sensor networks," in *OSDI*, 2002.

[4] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed diffusion: A scalable and robust communication paradigm for sensor networks," in *MOBICOM*. ACM, 2000, pp. 56–67.

[5] S. Lindsey and C. S. Raghavendra, "PEGASIS: Power-efficient gathering in sensor information systems," in *Aerospace Conference Proc.*, vol. 3, 2002, pp. 1125–1130.

[6] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-efficient communication protocol for wireless microsensor networks," in *Proc. of the 33rd Hawaii Int. Conf. on System Sciences-Volume 8*, ser. HICSS '00. Washington, DC, USA: IEEE Computer Society, 2000, pp. 8020–.

[7] Y. Yao and J. Gehrke, "The cougar approach to in-network query processing in sensor networks," *SIGMOD Rec.*, vol. 31, no. 3, pp. 9–18, Sep. 2002.

[8] H. Yu, P. B. Gibbons, S. Seshan, S. Nath, and S. Nath, "Synopsis diffusion for robust aggregation in sensor networks," in *In SenSys*. ACM Press, 2004, pp. 250–262.

[9] H. Abelson, D. Allen, D. Coore, C. Hanson, G. Homsy, J. Thomas F. Knight, R. Nagpal, E. Rauch, G. J. Sussman, and R. Weiss, "Amorphous computing," *Commun. ACM*, vol. 43, no. 5, pp. 74–82, 2000.

[10] J. Beal, J. Bachrach, D. Vickery, and M. Tobenkin, "Fast self-healing gradients," in *Proceedings of the 2008 ACM symposium on Applied computing*, ser. SAC '08. New York, NY, USA: ACM, 2008, pp. 1969–1975.

[11] J. Beal, "Flexible self-healing gradients," in *Proceedings of the 2009 ACM symposium on Applied Computing*, ser. SAC '09. New York, NY, USA: ACM, 2009, pp. 1197–1201.

[12] D. Samuelson and C. Macal, "Agent-based simulation comes of age," *OR/MS Today*, no. 4, pp. 34–38, 2006.