Fast Self-Healing Gradients

Jacob Beal, Jonathan Bachrach, Dan Vickery, Mark Tobenkin
MIT CSAIL, 32 Vassar Street, Cambridge, MA 02139
jakebeal@mit.edu, jrb@csail.mit.edu, drv@mit.edu, mmt@mit.edu

ABSTRACT

We present **CRF-Gradient**, a self-healing gradient algorithm that provably reconfigures in $O(diameter)$ time. Self-healing gradients are a frequently used building block for distributed self-healing systems, but previous algorithms either have a healing rate limited by the shortest link in the network or must rebuild invalid regions from scratch. We have verified **CRF-Gradient** in simulation and on a network of Mica2 motes. Our approach can also be generalized and applied to create other self-healing calculations, such as cumulative probability fields.

Categories and Subject Descriptors

D.1.3 [**Programming Techniques**]: Concurrent Programming—*Distributed programming*; C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Wireless communication*; F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems—*Geometrical problems and computations*

General Terms

Algorithms, Reliability, Theory, Experimentation

Keywords

Amorphous computing, spatial computing

1. CONTEXT

A common building block for pervasive computing systems is a *gradient*—a biologically inspired operation in which each device estimates its distance to the closest device designated as a source of the gradient (Figure 1). Gradients are commonly used in systems with multi-hop wireless communication, where the network diameter is likely to be high. Applications include data harvesting (e.g. Directed Diffusion[11]), routing (e.g. GLIDER[9]), distributed control (e.g. co-fields[14]) and coordinate system formation (e.g. [2]), to name just a few.

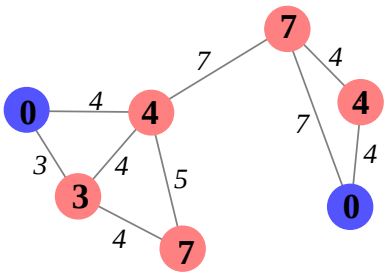


Figure 1: A gradient is a scalar field where the value at each point is the shortest distance to a source region (blue). The value of a gradient on a network approximates shortest path distances in the continuous space containing the network.

In a long-lived system, the set of sources may change over time, as may the set of devices and their distribution through space. It is therefore important that the gradient be able to self-heal, shifting the distance estimates toward the new correct values as the system evolves.

Self-healing gradients are subject to the *rising value problem*, in which local variation in effective message speed leads to a self-healing rate constrained by the shortest neighbor-to-neighbor distance in the network.

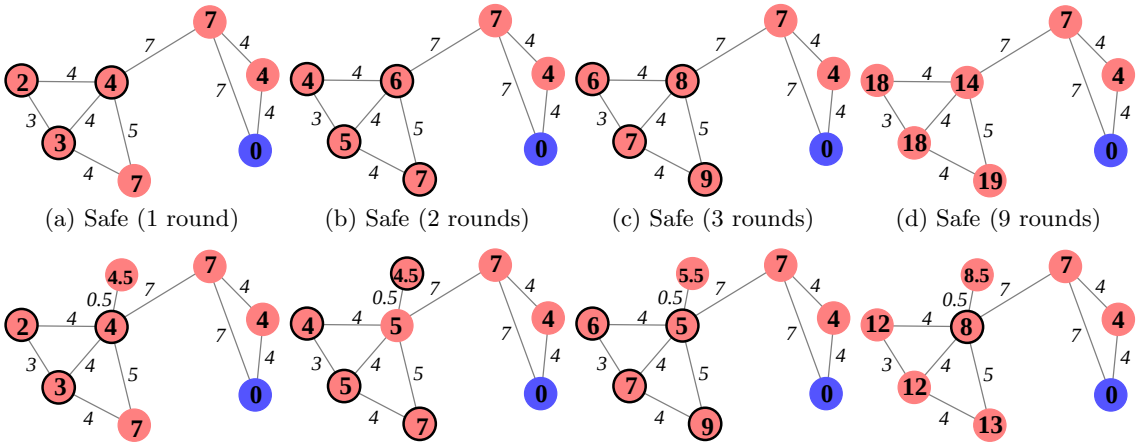
Previous work on self-healing gradients has either used repeated recalculation or assumed that all devices are the same distance from one another (e.g. measuring distance via hop-counts). Neither of these approaches is usable in large networks with variable distance between devices.

We present an algorithm, **CRF-Gradient**, that uses a metaphor of "constraint and restoring force" to address these problems. We have proved that **CRF-Gradient** self-stabilizes in $O(diameter)$ time and verified it experimentally both in simulation and on Mica2 motes. We also show that the constraint and restoring force framework can be generalized and applied to create other self-healing calculations, such as cumulative probability fields.

2. GRADIENTS

Gradients are generally calculated through iterative application of a triangle inequality constraint. In its most basic form, the calculation of the gradient value g_x of a device x is simply

SAC'08 March 16-20, 2008, Fortaleza, Ceará, Brazil



(a) Safe (1 round) (b) Safe (2 rounds) (c) Safe (3 rounds) (d) Safe (9 rounds)

(e) Problem (1 round) (f) Problem (2 rounds) (g) Problem (3 rounds) (h) Problem (9 rounds)

Figure 2: The *rising value problem* occurs when the round-trip distance between devices is less than the the desired rise per round. The figures above show self-healing after the left-most in Figure 1 stops being a source; for this example, updates are synchronous and unconstrained devices (black edges) attempt to rise at 2 units per round. The original graph rises safely (a-d), completing after 9 rounds. With one short edge (e-h), all rising becomes regulated by the short edge and proceeds much more slowly.

$$g_x = \begin{cases} 0 & \text{if } x \in S \\ \min\{g_y + d(x,y) | y \in N_x\} & \text{if } x \notin S \end{cases}$$

where S is the set of source devices, N_x is the neighborhood of x (excluding itself) and $d(x,y)$ the estimated distance between neighboring devices x and y.

Whenever the set of sources S is non-empty, repeated fair application of this calculation will converge to the correct value at every point. We will call this limit value $\bar{g}_x$.

2.1 Network Model

The gradient value of a device is not, however, instantaneously available to its neighbors, but must be conveyed by a message, which adds lag. We will use the following wireless network model:

- The network of devices D may contain anywhere from a handful of devices to tens of thousands.
- Devices are immobile and are distributed arbitrarily through space (generalization to mobile devices is relatively straightforward, but beyond the scope of this paper).
- Memory and processing power are not limiting resources.[1]
- Execution happens in partially synchronous rounds, once every Δ_t seconds; each device has a clock that ticks regularly, but frequency may vary slightly and clocks have an arbitrary initial time and phase.
- Devices communicate via unreliable broadcasts to their neighbors (all other devices within r meters distance). Broadcasts are sent at most once per round.
- Devices are provided with estimates of the distance to their neighbors, but naming, routing, and global coordinate services are not provided.
- Devices may fail, leave, or join the network at any time, which may change the connectedness of the network.

[1]Excessive expenditure of either is still bad, and memory is an important constraint for the Mica2 implementation.

2.2 Separation in Space and Time

We can reformulate the gradient calculation to take our network model into account. Let the triangle inequality constraint $c_x(y,t)$ from device y to device x at time t be expressed as

$$c_x(y,t) = g_y(t - \lambda_x(y,t)) + d(x,y)$$

where $\lambda_x(y,t)$ is the time-lag in the information about y that is available to its neighbor x. The time-lag is itself time-dependent (though generally bounded) due to dropped messages, differences in execution rate, and other sources of variability.

The gradient calculation is then

$$g_x(t) = \begin{cases} 0 & \text{if } x \in S(t) \\ \min\{c_x(y,t) | y \in N_x(t)\} & \text{if } x \notin S(t) \end{cases}$$

Our definition of the set of sources $S(t)$ and neighborhood $N_x(t)$ have also changed to reflect the fact that both may vary over time.

The most important thing to notice in this calculation is that the rate of convergence depends on the effective speed at which messages propagate through space. Over many hops, this speed may be assumed to be close to r/Δ_t (cf. [12]). Over a single hop, however, messages may move arbitrarily slowly: the time separation of two neighbors x and y is always on the order of Δ_t, while the spatial separation $d(x,y)$ may be any arbitrary distance less than r.

A device and its neighbor constrain one another. Thus, when the value of a device rises from a previously correct value, it can rise no more than twice the distance to its closest neighbor in one round; if it rises higher, then it is constrained by the neighbor's value. This applies to the neighbor as well, so after each round of rising the constraints

are no looser.

Since successive round trips between neighbors must take at least Δ_t seconds, a pair of neighbors constrain one another's distance estimates to rise at a rate no greater than $2d(x,y)/\Delta_t$ meters per second. When a device x has a value less than the correct value, its time to converge is at least

$$\max\{(\bar{g_x} - g_x(t))\frac{\Delta_t}{2d(x,y)} | y \in N_x(t)\}$$

which means that close neighbors can only converge slowly.

Worse, the dependency chains from this retarded convergence can reach arbitrarily far across the network, so that the entire network is limited in its convergence rate by the closest pair of devices. We will call this phenomenon the *rising value problem* (illustrated in Figure 2).

This can be very bad indeed, particularly given that many proposals for large networks involve some randomness in device placement (e.g. aerial dispersal). Consider, for example, a randomly distributed sensor network with 100 devices arranged in a 10-hop network with an average of 50 meters separation between devices that transmit once per second. Let us assume that the random distribution results in one pair of devices ending up only 50cm apart. If the source moves one hop farther from this pair, increasing the correct distance estimate by 50 meters, then the close pair and every device further in the network will take $50\frac{1}{2 \cdot 0.5} = 50$ seconds to converge to the new value. If they had landed 5cm apart rather than 50cm, it would take 500 seconds to converge—nearly 10 minutes!

2.3 Previous Self-Healing Gradients

To the best of our knowledge, the rising value problem has not previously been formalized. Previous work in self-healing gradients, however, may still be categorized into two general approaches. An *invalidate and rebuild* gradient discards previous values and recalculates sections of network from scratch, avoiding the rising value problem by only allowing values to decrease. An *incremental repair* gradient moves its values bit by bit until they arrive at the desired values, and previously have avoided the rising value problem by limiting the minimum link distance.

Invalidate and rebuild gradients periodically discard values across part or all of the network. This effectively avoids the rising value problem by ensuring that values that may need to change are raised above their correct values; often the value is not allowed to rise at all except during a rebuild. For example, GRAB[15] uses a single source and rebuilds when its error estimate is too high, and TTDD[13] builds the gradient on a static subgraph, which is rebuilt in case of delivery failure. These approaches work well in small networks and are typically tuned for a particular use case, but the lack of incremental maintenance means that there are generally conditions that will cause unnecessary rebuilding, persistent incorrectness, or both.

Incremental repair does not discard values, but instead allows the gradient calculation to continue running and reconverge to the new correct values. Previous work on incremental repair (by Clement and Nagpal[7] and Butera[6]) has measured distance using hop-count. This has the effect of setting $d(x,y)$ to a fixed value and therefore producing a consistent message speed through the network. If generalized to use actual distance measures, however, these approaches suffer from the rising value problem and may converge extremely slowly.

Finally, our previous work in [1] uses a hybrid solution that adds a fixed amount of distortion at each hop, producing a gradient that does not suffer from the rising value problem, but produces inaccurate values.

3. THE CRF-GRADIENT ALGORITHM

We have seen that a key problem for self-healing gradients is how to allow values to rise quickly yet still converge to good distance estimates. The **CRF-Gradient** algorithm handles this problem by splitting the calculation into *constraint* and *restoring force* behaviors (hence the acronym CRF).

When constraint is dominant, the value of a device $g_x(t)$ stays fixed or decreases, set by the triangle inequality from its neighbors' values. When restoring force is dominant, $g_x(t)$ rises at a fixed velocity v_0. The switch between these two behaviors is made with hysteresis, such that a device's rising value is not constrained by a neighbor that might still be constrained by the device's old value.

The calculation for **CRF-Gradient** implements this by tracking both a device's gradient value $g_x(t)$ and the "velocity" of that value $v_x(t)$. We use this velocity to calculate a relaxed constraint $c'_x(y,t)$ that accounts for communication lag when a device is rising:

$$c'_x(y,t) = c_x(y,t) + (\lambda_x(y,t) + \Delta_t) \cdot v_x(t)$$

We will use the original $c_x(y,t)$ to exert constraint and the new $c'_x(y,t)$ to test whether any neighbor is able to exert constraint. Let the set of neighbors exerting constraint be

$$N'_x(t) = \{y \in N_x(t) | c'_x(y,t) \leq g_x(t - \Delta_t)\}$$

CRF-Gradient may thus be formulated

$$g_x(t) = \begin{cases} 0 & \text{if } x \in S(t) \\ min\{c_x(y,t) | y \in N'_x(t)\} & \text{if } x \notin S(t), N'_x(t) \neq \emptyset \\ g_x(t - \Delta_t) + v_0 \Delta_t & \text{if } x \notin S(t), N'_x(t) = \emptyset \end{cases}$$

$$v_x(t) = \begin{cases} 0 & \text{if } x \in S(t) \\ 0 & \text{if } x \notin S(t), N'_x(t) \neq \emptyset \\ v_0 & \text{if } x \notin S(t), N'_x(t) = \emptyset \end{cases}$$

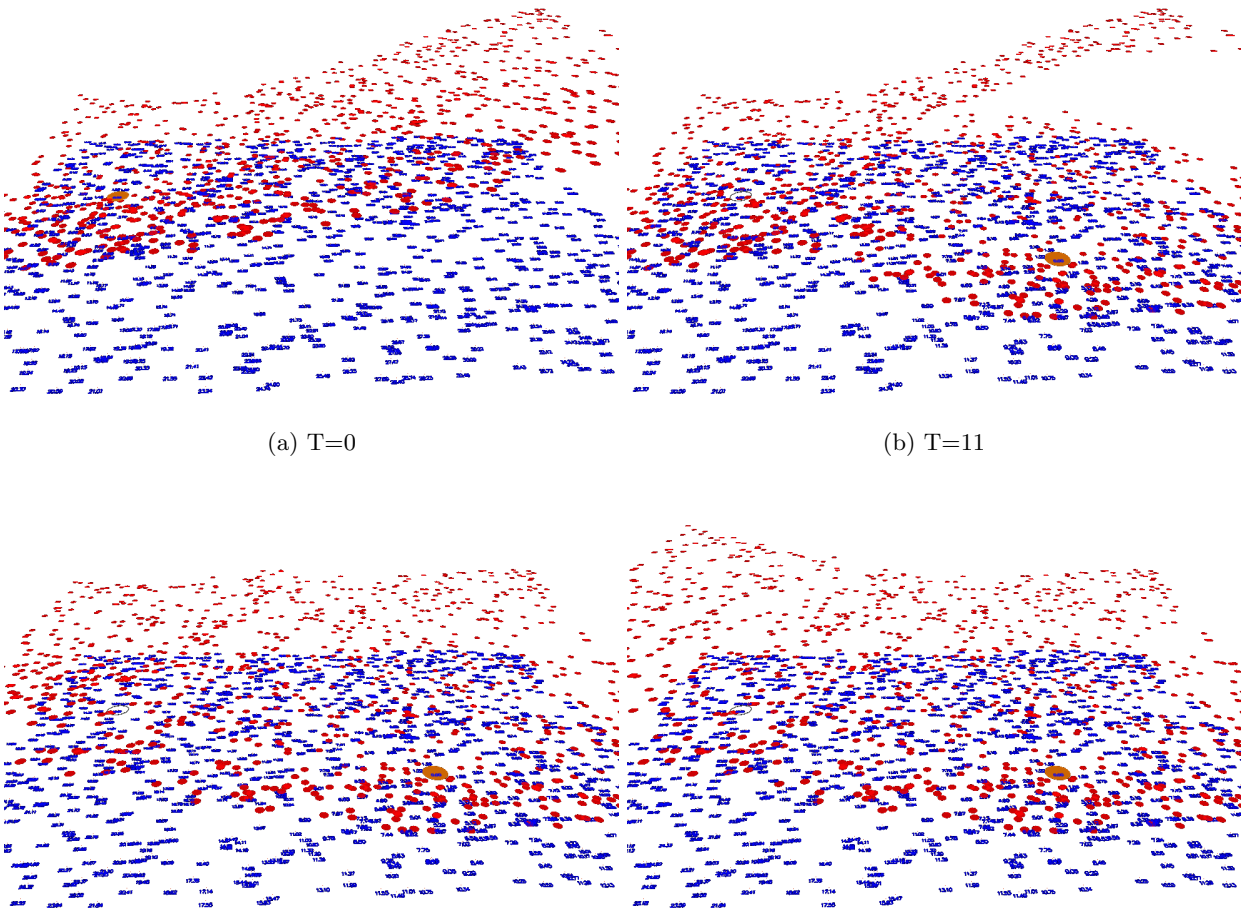
Using this calculation, the values of devices will converge to the same limit as before. The convergence is not, however, limited by short-range message speed: the value of a device rises smoothly, overshoots by a small amount, then snaps down to its correct value.

4. ANALYSIS AND VERIFICATION

We have proved that **CRF-Gradient** quickly self-stabilizes. We have further verified the behavior of **CRF-Gradient** both in simulation and on a network of Mica2 Motes.

4.1 Fast Self-Stabilization

From any arbitrary starting state, the network of devices converges to correct behavior in $O(diameter)$ time. The proof, detailed in a technical report[5], uses the *amorphous medium* abstraction[3], which considers the collection of devices as an approximation of the space they are distributed through. Self-stabilization is first proved for the continuous space, then shown to still hold for a discrete approximation



(a) T=0

(b) T=11

(c) T=31

(d) T=74

Figure 3: CRF-Gradient reconfigures in response to a change of source location (orange), running in simulation on a network of 1000 devices, 19 hops across. The network is viewed at an angle, with the value shown as the height of the red dot above the device (blue). Reconfiguration spreads quickly through areas where the new value is lower than the old (b), then slows in areas where the new value is significantly higher (c), completing 74 rounds after the source moves.

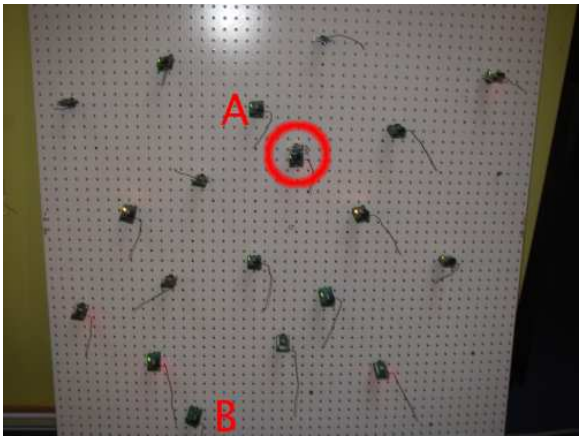


Figure 4: Our experimental network of 20 Mica2 motes, laid out in a mesh-like network with synthetic coordinates. Reception range was software-limited to 15 inches, producing a 5-hop network. Motes A and B are the two gradient sources, and the red circle contains two motes that are only 1 inch apart.

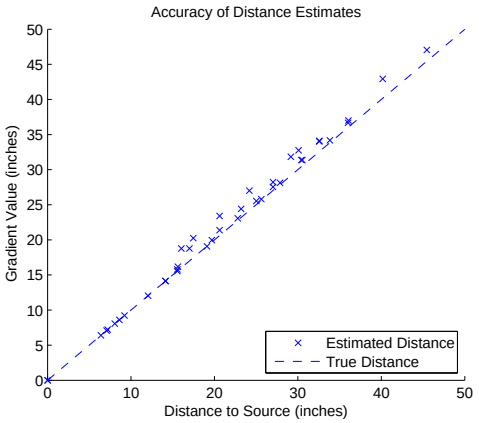


Figure 5: CRF-Gradient calculates good range estimates on our test network. The high quality of the estimates is unsurprising, given good connectivity and perfect range data, but serves to confirm that CRF-Gradient is behaving as expected.

of that space. The proof shows that convergence time is less than $6 \cdot diameter/c$, where c is the expected speed of message propagation in meters per second.

4.2 Verification in Simulation

In simulation, **CRF-Gradient** converges and reconfigures as predicted by our analysis. For example, the reconfiguration shown in Figure 3 takes place on a network of 1000 devices, placed randomly with uniform distribution to produce a network 19 hops wide. Analysis predicts that the reconfiguration should complete within $6 \cdot diameter/c = 6 \cdot 19 = 114$ rounds, and in fact it completes in 74 rounds.

4.3 Verification on Mica2 Motes

We tested **CRF-Gradient** on a network of 20 Mica2 Motes running our language Proto[4] on top of TinyOS[10]. The motes were laid out at known positions in a mesh-like network with one close pair, then supplied with perfect synthetic coordinates (Figure 4). Note that, because **CRF-Gradient** is self-stabilizing, we can expect that localization error would not disrupt the gradient as long as the coordinates are low-pass filtered to change more slowly than the convergence time.

Reception range was software-limited to 15 inches (producing a 5-hop network) in order to allow reliable monitoring of a multi-hop network through a single base-station. The frequency of neighborhood updates was set to 4 seconds.

We then ran two experiments, comparing **CRF-Gradient** with a velocity of 2 in/sec against the naive self-healing gradient described in Section 2.2. For each experiment, we started by designating a mote near the close pair (labeled A in Figure 4) as the source and allowed the gradient values to converge. We then moved the source to a mote far from the close pair (B) and waited again for values to converge. We continued moving the source back and forth between A and B, accumulating records of five moves in each direction.

We verify that the algorithm converges correctly by comparing the estimates of distance to the straight-line distance to the source. The estimates calculated by **CRF-Gradient** are unsurprisingly accurate, given the mesh-like layout and synthetic coordinates (Figure 5). The error in the estimates is entirely due to the difference between the straight-line path and the straightest path through the network.

We then compare convergence rates in two cases: when the values of the close pair fall (moves from B to A) and when the values of the close pair rise (moves from A to B). When the values of the close pair fall, the two algorithms behave similarly; when the close pair must rise, however, the naive algorithm is afflicted by the rising value problem, but **CRF-Gradient** is not (Figure 6).

5. GENERALIZATION

The constraint and restoring force approach can be generalized and applied to create other self-healing calculations besides gradients.

In the general form, we will let $c_x(y,t)$ be an any function for the constraint between two neighbors, define g_0 to be the value of a source and define $f(g,\delta)$ to be the new value after restoring force is applied to a value g for δ seconds.

The relaxed constraint $c'_x(y,t)$ is thus reformulated to

$$c'_x(y,t) = \left\{ \begin{array}{cc} c_x(y,t) & \text{if } v_x(t) = 0 \\ f(c_x(y,t), \lambda_x(y,t) + \Delta_t) & \text{if } v_x(t) \neq 0 \end{array} \right\}$$

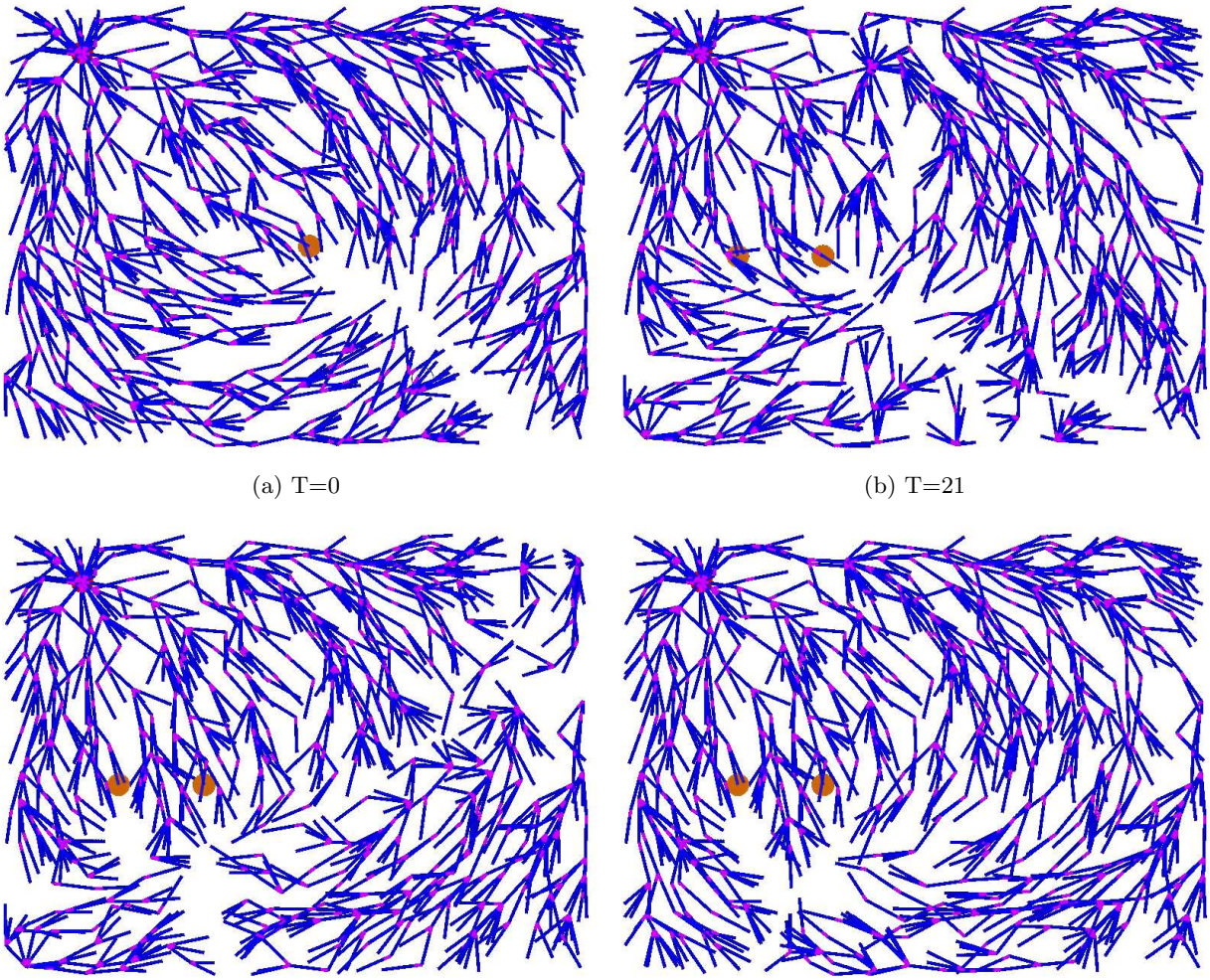
and $N'_x(t)$ to test $\leq$ for minimizing constraints and $\geq$ for maximizing constraints. For a minimizing constraint, the general constraint and restoring force calculation is

$$g_x(t) = \left\{ \begin{array}{cc} g_0 & \text{if } x \in S(t) \\ min\{c_x(y,t) | y \in N'_x(t)\} & \text{if } x \notin S(t), N'_x(t) \neq \emptyset \\ f(g_x(t-\Delta_t), \Delta_t) & \text{if } x \notin S(t), N'_x(t) = \emptyset \end{array} \right.$$

$$v_x(t) = \left\{ \begin{array}{cc} 0 & \text{if } x \in S(t) \\ 0 & \text{if } x \notin S(t), N'_x(t) \neq \emptyset \\ \frac{f(g_x(t-\Delta_t),\Delta_t) - g_x(t-\Delta_t)}{\Delta_t} & \text{if } x \notin S(t), N'_x(t) = \emptyset \end{array} \right.$$

Maximizing is the same except it uses max instead of min.

For example, we can calculate maximum cumulative probability paths to a destination (used in [8] to avoid threats).



(a) T=0

(b) T=21

(c) T=41

(d) T=90

Figure 7: A threat avoidance program built using a generalization of the constraint and restoring force approach. These images show reconfiguration in response to a change in threat location (orange), running in simulation on a network of 1000 devices, 19 hops across. The vectors from each device display the estimated minimum-threat path from that device towards the destination (upper-left corner).

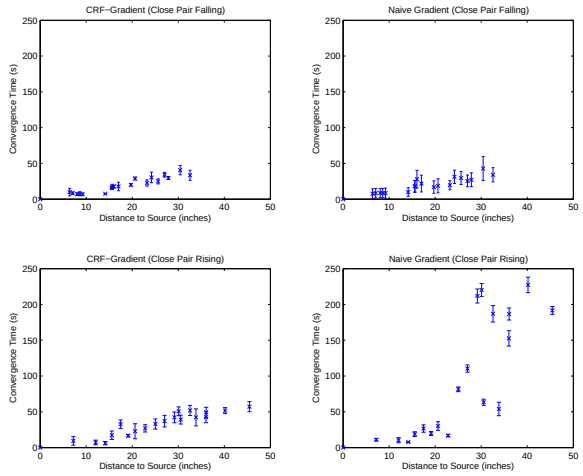


Figure 6: Convergence vs. distance for five trials using the experimental setup in Figure 4. CRF-Gradient and a naive self-healing gradient converge similarly except when the value of the close pair must rise (bottom graphs).

The maximum probability calculation is $g_0 = 1$, $c_x(y,t)$ is a maximizing constraint to g_y times the maximum integral of the probability density function along any path from x to y confined to the neighborhood, and $f(g,\delta) = g \cdot 0.99^{\delta}$.

When incorporated into a threat avoidance program similar to that in [8] and run in simulation, this calculation shifts the path in response to changing threats. Figure 7 shows a path shifting in time that appears proportional to the longest path of change.

6. CONTRIBUTIONS

We have introduced **CRF-Gradient**, a self-healing gradient algorithm that provably reconfigures in $O(diameter)$ time. We have verified **CRF-Gradient** in simulation and on a network of Mica2 motes. We also explain the *rising value problem*, which has limited previous work on self-healing gradients.

Our approach can also be generalized and applied to create other self-healing calculations, such as cumulative probability fields. This approach may be applicable to a wide variety of problems, potentially creating more robust versions of existing algorithms and serving as a building block for many pervasive computing applications.

7. REFERENCES

[1] J. Bachrach and J. Beal. Programming a sensor network as an amorphous medium. In *Distributed Computing in Sensor Systems (DCOSS) 2006 Poster*, June 2006.

[2] J. Bachrach, R. Nagpal, M. Salib, and H. Shrobe. Experimental results and theoretical analysis of a self-organizing global coordinate system for ad hoc sensor networks. *Telecommunications Systems Journal, Special Issue on Wireless System Networks*, 2003.

[3] J. Beal. Programming an amorphous computational medium. In *Unconventional Programming Paradigms International Workshop*, September 2004.

[4] J. Beal and J. Bachrach. Infrastructure for engineered emergence in sensor/actuator networks. *IEEE Intelligent Systems*, pages 10–19, March/April 2006.

[5] J. Beal, J. Bachrach, and M. Tobenkin. Constraint and restoring force. Technical Report MIT-CSAIL-TR-2007-042, MIT, August 2007.

[6] W. Butera. *Programming a Paintable Computer*. PhD thesis, MIT, 2002.

[7] L. Clement and R. Nagpal. Self-assembly and self-repairing topologies. In *Workshop on Adaptability in Multi-Agent Systems, RoboCup Australian Open*, Jan. 2003.

[8] A. Eames. Enabling path planning and threat avoidance with wireless sensor networks. Master's thesis, MIT, June 2005.

[9] Q. Fang, J. Gao, L. Guibas, V. de Silva, and L. Zhang. Glider: Gradient landmark-based distributed routing for sensor networks. In *INFOCOM 2005*, March 2005.

[10] D. Gay, P. Levis, R. von Behren, M. Welsh, E. Brewer, and D. Culler. The nesc language: A holistic approach to networked embedded systems. In *Proceedings of Programming Language Design and Implementation (PLDI) 2003*, June 2003.

[11] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: A scalable and robust communication paradigm for sensor networks. In *Sixth Annual International Conference on Mobile Computing and Networking (MobiCOM '00)*, August 2000.

[12] L. Kleinrock and J. Silvester. Optimum transmission radii for packet radio networks or why six is a magic number. In *Natl. Telecomm. Conf.*, pages 4.3.1–4.3.5, 1978.

[13] H. Luo, F. Ye, J. Cheng, S. Lu, and L. Zhang. Ttdd: A two-tier data dissemination model for large-scale wireless sensor networks. *Journal of Mobile Networks and Applications (MONET)*, 2003.

[14] M. Mamei, F. Zambonelli, and L. Leonardi. Co-fields: an adaptive approach for motion coordination. Technical Report 5-2002, University of Modena and Reggio Emilia, 2002.

[15] F. Ye, G. Zhong, S. Lu, and L. Zhang. Gradient broadcast: a robust data delivery protocol for large scale sensor networks. *ACM Wireless Networks (WINET)*, 11(3):285–298, 2005.