Fast Self-Stabilization for Gradients

Jacob Beal[1], Jonathan Bachrach[2], Dan Vickery, and Mark Tobenkin

[1] BBN Technologies, Cambridge, MA 02138, USA,
jakebeal@bbn.com
[2] MIT CSAIL, Cambridge, MA 02139, USA

Abstract. Gradients are distributed distance estimates used as a building block in many sensor network applications. In large or long-lived deployments, it is important for the estimate to self-stabilize in response to changes in the network or ongoing computations, but existing algorithms may repair very slowly, produce distorted estimates, or suffer large transients. The CRF-Gradient algorithm[1] addresses these shortcomings, and in this paper we prove that it self-stabilizes in $O(diameter)$ time—more specifically, in $4 \cdot diameter/c + k$ seconds, where k is a small constant and c is the minimum speed of multi-hop message propagation.

1 Context

A common building block for distributed 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).[3] 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[2]), routing (e.g. GLIDER[3]), distributed control (e.g. co-fields[4]) and coordinate system formation (e.g. [5]), to name just a few.

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. As a result, self-healing gradient algorithms have suffered from potentially very slow repair, distorted estimates, or large transients during repair. The **CRF-Gradient** algorithm[1, 6] addresses these problems using a metaphor of "constraint and restoring force."

In this paper, we prove that the **CRF-Gradient** algorithm self-stabilizes in $O(diameter)$ time—more specifically, in $4 \cdot diameter/c + k$ time, where k is a small

[3] Note that "gradient" can mean either the vector operator or a value that changes over space (e.g. chemical concentration in a developing embryo). Historically, the operation we discuss has inherited its name from the latter use, due to its common use in biologically-inspired systems.

constant and c is the minimum speed of multi-hop information propagation. Section 2 and Section 3 review gradients, the rising value problem, and the **CRF-Gradient** algorithm. Section 4, the bulk of the paper, is devoted to formal analysis of the **CRF-Gradient** algorithm.

2 Gradients and the Rising Value Problem

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

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

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 device.

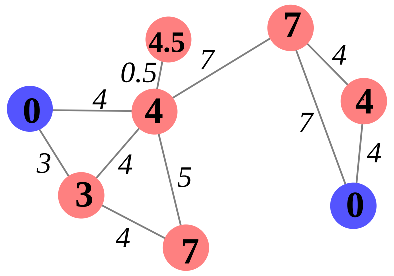

Fig. 1. A gradient is a scalar field where the value at each device is the shortest distance to a source region (blue). The value of a gradient on a network approximates shortest path distances in the continuous space containing the network.

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). The *diameter* of this network is the maximum graph distance between devices.

- Memory and processing power are not limiting resources.
- Every device has a copy of the same program, which is executed periodically to update the state of the device. 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 all other devices within r meters distance. These devices within r are called neighbors. Broadcasts are sent once per round, halfway between executions.
- 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.

Note that, although we use the simplistic unit disc model for communication and assume no measurement error, the it is straightforward to extend the results in this paper to more realistic models. The results derive from the relationship between the speed at which information propagates through the network and the rate at which distance estimates increase as it propagates. Adjusting the model will only change that constants (in the algorithm and in its convergence time) that are derived from this relationship.

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-varying (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. [7]) for networks where transmission and propagation delay is short compared to Δ_t, as is often the case in wireless networks. Over a single hop, however, messages may move arbitrarily slowly: the time separation of two neighbors x and y is always

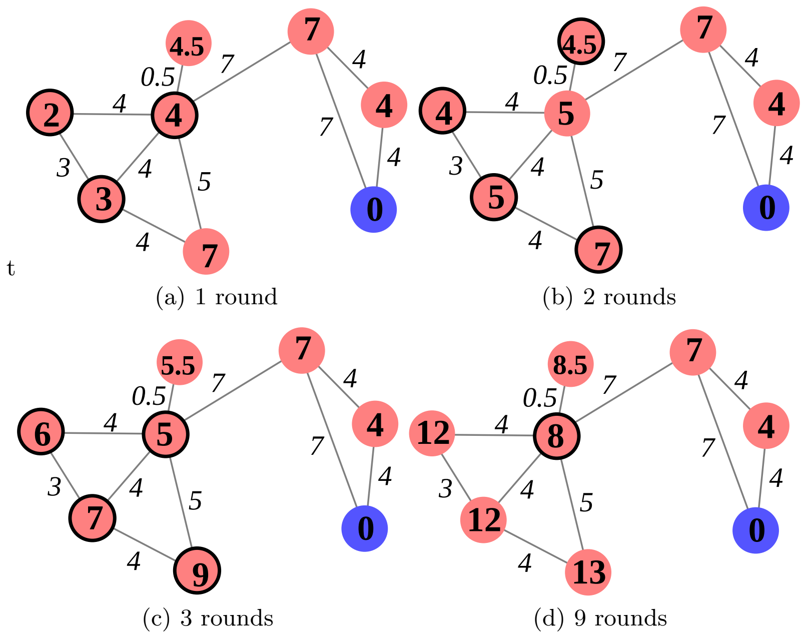
(a) 1 round (b) 2 rounds (c) 3 rounds (d) 9 rounds

Fig. 2. The *rising value problem* causes repair to be limited by the shortest edge in the network, as when the left-most device in Figure 1 stops being a source. Here updates are synchronous and unconstrained devices (black edges) attempt to rise at 2 units per round.

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 $\bar{g_x}$, 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 at least $50\frac{1}{2 \cdot 0.5} = 50$ seconds to converge to the new value. If they had landed 5cm apart rather than 50cm, it would take over 500 seconds to converge—nearly 10 minutes!

3 The CRF-Gradient Algorithm

The **CRF-Gradient** algorithm avoids the rising value 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 behavior mode is indicated by the "velocity" $v_x(t)$ of a device's value and the switch between 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.

This switch is implemented by defining the subset of neighbors $N'_x(t)$ allowed to exert constraint as:

$$N'_x(t) = \{y \in N_x(t)|c_x(y, t) + (\lambda_x(y, t) + \Delta_t) \cdot v_x(t - \Delta_t) \leq g_x(t - \Delta_t)\}$$

The hysteresis comes from the v_x term: when rising, $v_x(t-\Delta_t)$ is positive and the constraint is loosened by the amount a device's value might rise while information is making a round trip between the device and its neighbor. Then **CRF-Gradient** may be formulated

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

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

These update equations avoid the rising value problem: the value of a device rises smoothly, overshoots by a small amount, then snaps down to its correct value.

3.1 Other Self-Healing Gradients

Self-healing gradients be categorized into two general approaches: either *incremental repair* or *invalidate and rebuild*. **CRF-Gradient** is an example of incremental repair: at each step, devices attempt to move their values up or down towards the correct value. Other work on incremental repair (by Clement and Nagpal[8] and Butera[9]) has measured distance using hop-count—effectively setting $d(x,y)$ to a fixed value and therefore producing a consistent message speed through the network—and suffer from the rising value problem if generalized to use distance instead of hop-count. A hybrid solution in [10] adds a fixed amount of distortion at each hop, exchanging the rising value problem for inaccurate values.

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. For example, GRAB[11] uses a single source and rebuilds when its error estimate is too high, and TTDD[12] 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.

4 Analysis

We show that **CRF-Gradient** converges in $O(diameter)$ time by proving self-stabilization, where the network converges to correct behavior from an arbitrary starting state. Self-stabilization also gives an upper bound on the rate at which the algorithm can adapt to changes in the network or the source region. In other work[1], we have verified the expected behavior of **CRF-Gradient** both in simulation and on a network of Mica2 Motes. A technical report[6] outlines a proof of self-stabilization under a continuous space/time abstraction.

4.1 Algorithm State

In order to prove self-stabilization, we must first make explicit what state is stored at devices—the mathematical formulation leaves this implicit. There are a total of nine variables used by **CRF-Gradient**, plus the phase timer used to schedule the next event on a device, all summarized in Table 1.

Variable	Type	Range
Δ_t	Constant	$(0, \infty)$
v_0	Constant	$(0, \infty)$
$S(t)$	Input	$\{true, false\}$ per device
$g_x(t - \Delta_t)$	State	$[0, \infty)$
$v_x(t - \Delta_t)$	State	$\{0, v_0\}$
$N_x(t)$	State	[see below]
$\lambda_x(y, t)$	State	$[0, \infty)$
$g_y(t - \lambda_x(y, t))$	State	$[0, \infty)$
$d(x, y)$	State	$(0, r]$
$phase$	State	$[0, \Delta_t)$

Table 1. Variables and ranges used by **CRF-Gradient**: self-stabilization begins with arbitrary values in all state variables.

These state variables may be considered in three different categories: local, neighborhood, and algorithmic. Since the *phase* just determines relative order of execution, any possible value of *phase* is consistent with the algorithm. The neighborhood variables can be implemented several different ways. For this discussion, we assume that the messages broadcast by each device contain its unique ID, $g_x(t)$, and whatever localization information is needed to allow the receiver to compute $d(x, y)$. $N_x(t)$ is then the set of unique IDs in a device's record of its neighborhood. When a message arrives from a neighbor, a device adds the information to its neighborhood with $\lambda_x(y, t) = -phase + \Delta_t/2$, replacing any previous information about that neighbor. Before each execution, Δ_t is added to the $\lambda_x(y, t)$ of each neighbor. If $\lambda_x(y, t)$ goes above a fixed timeout T, the neighbor is deleted. Neighborhood state is correct if the values for each neighbor reflect the history and physical location of that device, and this simple mechanism guarantees that, from arbitrary state, this will become the case once T seconds have elapsed: neighbors refresh their state each round and entries for non-existent neighbors are flushed after T seconds. We shall assume that T is a small constant and neglect it henceforth.

This leaves only the algorithmic variables, $g_x(t - \Delta_t)$ and $v_x(t - \Delta_t)$. The latter is correct whenever it reflects the amount that g_x changed in the last update, and thus becomes correct after a single round. The correct behavior of $g_x(t - \Delta_t)$ depends on the source region $S(t)$. If the source region is non-empty, then its value must be equal to $d(x, S(t))$. If the source region is empty, then every value in the network must float upwards: formally, there must exist a time t_f such that for every $x \in D$, the gradient value rises at v_0 thereafter:

$g_x(t) \geq g_x(t_f) + v_0(t - t_f - \Delta_t)$. Subtracting $v_0\Delta_t$ in the expression allows the rise to occur in discrete steps.

The next section is devoted to showing the self-stabilization of g_x.

4.2 Proof of Self-Stabilization

From any arbitrary starting state, the network of devices converges to correct behavior in $O(diameter)$ time—specifically, in time less than $4 \cdot diameter/c + k$, where c is the minimum speed of message propagation in meters per second and k is a small constant.

We prove this by first finding upper and lower bounds on how quickly information propagates through the network. We use these bounds to show that the minimum values of a network quickly constrain the values of all other devices, and that without a constraining source the minimum values rise steadily. Together, these lead to a proof that the network rapidly converges to either correct values or a steady rise, depending on whether any sources exist.

For the purposes of this proof, we will assume that the source region remains fixed, there are no failures, that clock frequency does not vary between devices, and that neighbor distance estimates have no error.

A reminder of terms from the network model in Section 2.1: D is the network of devices that execute once every Δ_t seconds, broadcasting to all neighboring devices within r meters on the half-round. We will additionally augment our network model with the following definitions and assumptions:

- Messages are assumed to arrive instantly, with overhead time absorbed into the 1/2 round delay between execution and transmission.
- The distance between non-neighbors is defined recursively, through the network: $d(x, y) = min(\{d(x, z) + d(z, y) | z \in N_x(t)\})$. The distance between regions will be the minimum of the distance between pairs of devices in each region.
- No device has two neighbors on any ray emanating from itself.[4] This ensures that the release of constraint propagates quickly across multiple hops.
- $g_X(t)$ is the set of gradient values in a set of devices $X \subseteq D$ at time t. Likewise, $d_X(Y, Z)$ is the minimum distance between devices in Y and Z on paths confined to X.
- We will define the *forward lag* $L_x(y, t)$ to be the time-lag between an event at device x at time t and the next equivalent event at device y where the value $g_x(t)$ can constrain $g_y(t)$ along a path equal to $d(x, y)$. For neighboring devices, $L_x(y, t)$ is always in the range $\frac{1}{2}\Delta_t$ to $\frac{3}{2}\Delta_t$, and $L_x(y, t) = \lambda_y(x, t + L_x(y, t))$. Across multiple hops, we define $L_x(y, t)$ recursively as $L_x(y, t) = min(\{L_x(z, t) + L_z(y, t + L_x(z, t)) | z \in N_x(t) \text{ s.t. } d(x, z) + d(z, y) = d(x, y)\})$
- The restoring velocity v_0 is bounded by $v_0 \leq c/4$.

Given these definitions, we can begin the proofs, starting with a bounding of speeds over multi-hop distances.

[4] Such a network can be produced by adding a small amount of randomness to location.

Lemma 41 (Multi-Hop Speed) *Given devices $x, y \in D$ at time t, where x and y are not neighbors, the speed at which information propagates across the shortest path between them is bounded below by $c = \frac{1}{3}\frac{r}{\Delta_t}$ and above by $C = 2\frac{r}{\Delta_t}$.*

Proof. Assume without loss of generality that information is propagating from x to y, and consider the chain of hops between x and y along the shortest path.

Each pair of successive hops must move more than r distance or else the first element of the pair could be omitted. Thus the total number of hops is strictly less than $2d(x, y)/r$.

The time-lag across a single hop is at most $\frac{3}{2}\Delta_t$, between two devices with phases in the the worst alignment. Multiplying time per hop by number of hops, we see that the total time to propagate across distance $d(x, y)$ is strictly less than $3\frac{\Delta_t \cdot d(x,y)}{r}$.

Speed is distance divided by time, so we may establish a lower bound:

$$c = d(x, y)/3\frac{\Delta_t \cdot d(x, y)}{r}$$

$$c = \frac{1}{3}\frac{r}{\Delta_t}$$

The upper bound proceeds similarly, with the maximum distance per hop r and the lowest time-lag across a single hop $\frac{1}{2}\Delta_t$, yielding a bound of:

$$C = 2\frac{r}{\Delta_t}$$

We now show a loose bound for how the least values in a region bound the values of the whole region over time:

Lemma 42 *Let $R \subseteq D$. At time t_0, let $g_0 = \min(g_R(t_0))$, and define the minimum region M as the set of devices with minimal value, $M = \{x | x \in R, g_x(t_0) = g_0\}$.*

Then at time $t > t_0 + \frac{3}{2}\Delta_t$, every device $z \in R$ with $d_R(z, M) < c \cdot (t - t_0)$ has value $g_z(t) < g_0 + v_0\Delta_t + d_R(m, z) + v_0 \cdot (4\Delta_t\frac{d_R(m,z)}{r} + t - t_0)$

Proof. Consider a pair of neighboring devices, $x, y \in D$. If x executes at time t_x, producing value $g_x(t_x)$, then at time $t + L_x(y, t_x)$ y executes, producing a value

$$g_y(t_x + L_x(y, t_x)) < g_x(t_x) + d(x, y) + v_0 \cdot (L_x(y, t_x) + 2\Delta_t)$$

This bound is the decision threshold for constraint, raised by one round of restoring force.

Now consider an arbitrary pair of non-neighboring devices, $m \in M$ and $z \in D$. By Lemma 41, we know that if $d_R(m, z) < c \cdot (t - t_0)$ (and the elapsed time is enough to go at least one hop), then values from m will have time to propagate constraint to z along a shortest path between them.

Because m has value g_0 at time t_0, we know that at a time $t_m \in (t_0, t_0 + \Delta_t]$ it must compute a value $g_m(t_m) \leq g_0 + v_0\Delta_t$. The first execution at z that can be constrained along the shortest path by the value $g_m(t_m)$ occurs at time $t_m + L_m(z, t_m)$, which we will call t_z.

Accumulating the neighbor constraint across at least $2\frac{d_R(m,z)}{r}$ hops, we thus have the following constraint on the value of z:

$$g_z(t_z) < g_m(t_m) + d_R(m,z) + v_0 \cdot \left(L_m(z,t_m) + 4\Delta_t \frac{d_R(m,z)}{r}\right)$$

For an arbitrary $t > t_z$, this can have risen to at most:

$g_z(t) < g_m(t_m) + d_R(m,z) + v_0 \cdot (L_m(z,t_m) + 4\Delta_t \frac{d_R(m,z)}{r} + \Delta_t \left\lfloor \frac{(t-t_z)}{\Delta_t} \right\rfloor)$

where the floor is due to the fact that t_z is the time of an execution. Eliminating the floor and substituting for t_z we have:

$g_z(t) < g_m(t_m) + d_R(m,z) + v_0 \cdot \left(L_m(z,t_m) + 4\Delta_t \frac{d_R(m,z)}{r} + t - L_m(z,t_m) - t_m\right)$

$$g_z(t) < g_m(t_m) + d_R(m,z) + v_0 \cdot \left(4\Delta_t \frac{d_R(m,z)}{r} + t - t_m\right)$$

Substituting in the definitions for t_m and $g_m(t_m)$ gives

$$g_z(t) < g_0 + v_0\Delta_t + d_R(m,z) + v_0 \cdot \left(4\Delta_t \frac{d_R(m,z)}{r} + t - t_0\right)$$

Conversely, we can show how quickly values will rise when there are no constraints.

Lemma 43 (Floating Island Lemma) *Given a region $R \subseteq D - S(t_0)$ with no sources at time t_0, let g_0 be the minimum value of $g_R(t_0)$. Unless acted on by a constraint from a source outside of R, the gradient value for every device $x \in R$ at time $t > t_0$ is $g_x(t) \geq g_0 + v_0 \cdot (t - t_0 - \Delta_t)$*

Proof. Assume for contradiction that this is false: then there must be some device $x \in R$ that executes at time $t > t_0$ such that $g_x(t) < g_0 + v_0 \cdot (t - t_0 - \Delta_t)$. Since there are no sources in R, the value $g_x(t)$ must have been calculated using either $g_x(t - \Delta_t)$ (if unconstrained) or $g_y(t - \lambda_x(y,t))$ (if constrained). Iterating this, we can construct a dependency chain for $g_x(t)$ of constrained and unconstrained steps going backward to time t_0, grounding in an execution (real or apparent from phase) that occurs in the range $(t_0 - \Delta_t, t_0]$.

Assume this chain consists entirely of unconstrained steps. Each step goes backward in time Δ_t, so the number of steps backward is $\lceil \frac{t-t_0}{\Delta_t} \rceil$. Each of these steps decreases the value by $v_0\Delta_t$, so we have

$$g_x(t_0) = g_x(t) - v_0\Delta_t \cdot \left\lceil \frac{t - t_0}{\Delta_t} \right\rceil$$

Since the ceiling operator may raise the value of $\frac{t-t_0}{\Delta_t}$ by as little as zero, we know that

$$g_x(t_0) \leq g_x(t) - v_0\Delta_t \cdot \left(\frac{t - t_0}{\Delta_t}\right)$$

$$g_x(t_0) \leq g_x(t) - v_0 \cdot (t - t_0)$$

and substituting in our assumption for $g_x(t)$ produces

$$g_x(t_0) < g_0 + v_0 \cdot (t - t_0 - \Delta_t) - v_0 \cdot (t - t_0)$$

$$g_x(t_0) < g_0 - v_0 \Delta_t$$

which is a contradiction since g_0 is the minimum value.

Since each unconstrained step lowers the value by $v_0\Delta_t$, at least two unconstrained steps must be replaced by constrained steps. Steps need not be replaced at a 1:1 ratio, but the replacement steps must cover the same time-span–in this case at least $2\Delta_t$. Each unconstrained step can take a maximum of $\frac{3}{2}\Delta_t$ seconds, so there must be at least two such steps. Between them, these steps must cover a distance of at least r (otherwise the first and last devices would be neighbors, and since there are no collinear 3-cliques, the dependency chain could not visit the middle device). Replacing unconstrained steps with constrained steps thus can only decrease the distance if $r < 2v_0\Delta_t$, which is false by assumption.

Theorem 44 *The* **CRF-Gradient** *algorithm self-stabilizes in* $4 \cdot diameter/c + k$ *time, where* k *is a small constant.*

Proof. First, note that once a device is constrained by the source, it will always be constrained by the source—it can only relax towards a shorter path. The relaxation is finished within the transit time of information along the shortest path to the source.

Let t_0 be the time when self-stabilization begins.

If the source region is not empty, then by Lemma 43 we know that every device x not constrained by a source has a value at time t of $g_x(t) \geq v_0 \cdot (t - t_0 - \Delta_t)$. No device in the network needs a value greater than *diameter*, so they must rise to at most $diameter + \frac{5}{2}v_0\Delta_t$ in order to become constrained by a source. Setting $g_x(t)$ to this target value,

$$diameter + \frac{5}{2}v_0\Delta_t = v_0 \cdot (t - t_0 - \Delta_t)$$

we solve for $t - t_0$:

$$diameter + \frac{7}{2}v_0\Delta_t = v_0 \cdot (t - t_0)$$

$$(t - t_0) = \frac{diameter}{v_0} + \frac{7}{2}\Delta_t$$

We thus have a race between two processes, the outward flow of constraint from the source region and the upward rise of $g_x(t)$ value which are below their ultimate level. By Lemma 41, we know that constraint flows outward across multiple hops at a minimum speed of $c = \frac{1}{3}\frac{r}{\Delta_t}$. Since this propagation rate is much faster than v_0, we may expect that any distant device x will be constrained immediately after it rises above $d(x, S) + \frac{7}{2}v_0\Delta_t$, bringing the total time for stabilization to at most

$$\frac{diameter}{v_0} + \frac{11}{2}\Delta_t$$

(adding in a round to go above the threshold and another to snap down to the constraint). Given our assumption that $v_0 \leq c/4$, we take $v_0 = c/4$ to yield a bound of $4 \cdot \mathit{diameter}/c + \frac{11}{2}\Delta_t$

Now consider the case when the source region is empty. Let $t_f = t_0 + 4\frac{\mathit{diameter}}{c}$. Assume that t_f is not an acceptable time: this means there is some device $x \in D$ and time t such that $g_x(t) < g_x(t_f) + v_0 \cdot (t - t_f - \Delta_t)$. Constructing dependency chains, as in Lemma 43, the same logic shows that $g_x(t)$ must have at least one constraint step going through a neighbor y that occurs after time t_f. Since y is a neighbor, either the y is also violating the bound (and thus must be constrained by a neighbor of its own at an earlier time), or else the execution at which the constraint is applied happens precisely once, at the time t_x during the period $(t_f, t_f + \Delta_t]$. Assume the latter case (the former reduces to it by switching which device is under consideration).

Because y constrains x at t_x, y must not have been rising in its previous execution—otherwise the difference between y and x must have shrunk, meaning x is not constrained by y, or stayed the same, meaning x was rising also and will still not be constrained by y. Thus the next step of the dependency chain must be a constraint step to some neighbor z of y. We can apply the same argument iteratively to show the dependency chain must be all constraint steps back to the base time t_0. There must be at least $4\frac{\mathit{diameter}}{c\Delta_t}$ of these steps, each pair moving at least r distance (by the same argument as in Lemma 43). This gives us a bound on the post-constraint value of x: $g_x(t_x) \geq g_0 + 2r\frac{\mathit{diameter}}{c\Delta_t}$, where g_0 is the minimum gradient value in the network at time t_0. Substituting in $c = \frac{1}{3}\frac{r}{\Delta_t}$, we can simplify this:

$$g_x(t_x) \geq g_0 + 2r\frac{\mathit{diameter}}{\frac{1}{3}\frac{r}{\Delta_t}\Delta_t}$$

$$g_x(t_x) \geq g_0 + 6 \cdot \mathit{diameter}$$

By Lemma 42, we know that because there is a minimum g_0 in the network at time t_0, that at time t_f the value of x is bounded by

$$g_x(t_f) < g_0 + v_0\Delta_t + \mathit{diameter} + v_0 \cdot \left(4\Delta_t\frac{\mathit{diameter}}{r} + t_f - t_0\right)$$

Since we know that $g_x(t_f) > g_x(t_x) + v_0\Delta_t$ (or else it's not a violation), we can connect these two equations together to get

$$g_0 + 6 \cdot \mathit{diameter} + v_0\Delta_t < g_0 + v_0\Delta_t + \mathit{diameter} + v_0 \cdot \left(4\Delta_t\frac{\mathit{diameter}}{r} + t_f - t_0\right)$$

$$5 \cdot \mathit{diameter} < v_0 \cdot \left(4\Delta_t\frac{\mathit{diameter}}{r} + t_f - t_0\right)$$

$$5 \cdot \mathit{diameter} < v_0 \cdot \left(16\frac{\mathit{diameter}}{c}\right)$$

$$v_0 > \frac{5}{16}c$$

which is false by assumption. Thus the no-source case has a convergence time bounded above by $4 \cdot \mathit{diameter}/c$.

From the structure of these proofs, it appears that they should generalize to show self-stabilization for **CRF-Gradient** on devices with error in distance measurements and clock rate, as well as with non-unit disc communication, albeit with a great deal more proof complexity and producing slightly worse bounds. The permissiveness of the algorithm's constraints will need to be increased slightly to allow for error as well, however.

5 Contributions

We have proved that the **CRF-Gradient** algorithm self-stabilizes in $O(diameter)$ time—more specifically, in $4 \cdot diameter/c + k$ time, where k is a small constant, c is the minimum speed of multi-hop information propagation, and the restoring velocity is bounded $v_0 \leq c/4$. This result also implies fast self-healing following changes in network structure or source region, and the incremental nature of the repair means that there will often be useful values even while repair is going on.

In other work[1], we have verified that **CRF-Gradient** exhibits the predicted behavior both in simulation and on a network of Mica2 motes. The algorithm 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 distributed computing applications.

References

1. Beal, J., Bachrach, J., Vickery, D., Tobenkin, M.: Fast self-healing gradients. In: ACM Symposium on Applied Computing. (March 2008)
2. Intanagonwiwat, C., Govindan, R., Estrin, D.: 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)
3. Fang, Q., Gao, J., Guibas, L., de Silva, V., Zhang, L.: Glider: Gradient landmark-based distributed routing for sensor networks. In: INFOCOM 2005. (March 2005)
4. Mamei, M., Zambonelli, F., Leonardi, L.: Co-fields: an adaptive approach for motion coordination. Technical Report 5-2002, University of Modena and Reggio Emilia (2002)
5. Bachrach, J., Nagpal, R., Salib, M., Shrobe, H.: 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)
6. Beal, J., Bachrach, J., Tobenkin, M.: Constraint and restoring force. Technical Report MIT-CSAIL-TR-2007-042, MIT (August 2007)
7. Kleinrock, L., Silvester, J.: Optimum transmission radii for packet radio networks or why six is a magic number. In: Natl. Telecomm. Conf. (1978) 4.3.1–4.3.5
8. Clement, L., Nagpal, R.: Self-assembly and self-repairing topologies. In: Workshop on Adaptability in Multi-Agent Systems, RoboCup Australian Open. (January 2003)
9. Butera, W.: Programming a Paintable Computer. PhD thesis, MIT (2002)

10. Bachrach, J., Beal, J.: Programming a sensor network as an amorphous medium. In: Distributed Computing in Sensor Systems (DCOSS) 2006 Poster. (June 2006)
11. Ye, F., Zhong, G., Lu, S., Zhang, L.: Gradient broadcast: a robust data delivery protocol for large scale sensor networks. ACM Wireless Networks (WINET) **11**(3) (2005) 285–298
12. Luo, H., Ye, F., Cheng, J., Lu, S., Zhang, L.: Ttdd: A two-tier data dissemination model for large-scale wireless sensor networks. Journal of Mobile Networks and Applications (MONET) (2003)