Efficient Distributed "Hormone" Graph Gradients.

Esben Hallundbæk Østergaard
Maersk Mc-Kinney Moller Institute for Production Technology
University of Southern Denmark
esben@mip.sdu.dk

Abstract

Several researchers have attempted extract the principles behind hormone gradients found in biological system, and apply them to control physically manifested distributed systems. This paper presents an efficient implementation of a graph-based version of hormone gradient mechanisms. The algorithm is based on hop-counting of the topological distance between any given vertex and the hormone emitting vertex, and can deal with dynamic changes to the topology of the system. Performance of the described algorithm is documented through a number of experiments. The algorithm was developed for use in self-reconfigurable robotics, but might very well be useful for many other applications. The use of the algorithm to provide a common coordinate system for a collection of self-reconfigurable robot modules is described, that provides pose relative to the emitting module for all modules affected by the hormone.

1 Introduction

Hormone gradients have a central role in the coordination of growth and self-repair in biological multicellular systems. Several researchers have attempted to use these same mechanisms to control physically manifested distributed systems, by using some abstraction over the perceived functioning of biological hormone gradients. This paper presents an efficient implementation of a graph-based version of hormone gradient mechanisms, based on hop-counting and spanning a tree over the topology, that can deal with dynamic changes to the topology of the system. The paper will also describe how the algorithm can be used to relay a common coordinate system for modules in a self-reconfigurable robot. Even though the focus was on self-reconfigurable robots during development, the algorithm is believed to have much wider applicability.

2 Graph-Based Hormone Gradients

As a first approximation, simple gradients are fairly easy to implement. Figure 1 illustration **a)** shows a graph representing a system of 11 vertices, with 11 edges. In **b)** a simple

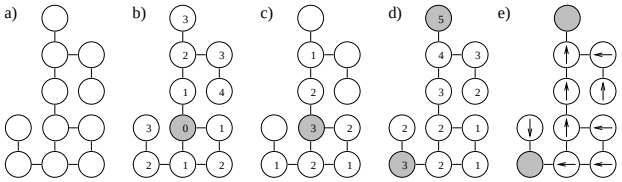

Figure 1: *A few simple graph gradients*

width first graph traversal algorithm (starting at the gray vertex) has been applied to assign a value to each vertex, where the value corresponds to the recursion depth. The result is that each vertex holds its distance to the gray vertex. In **c)** a similar algorithm is run, where a limit has been imposed on the recursion depth. The value assigned to vertices is the limit minus the recursion depth, resulting in the held value decreasing the further we get away from the gray module. This can be regarded as an abstract implementation of a hormone gradient emitted from the gray vertex. The hormone gradient is emitted with a given strength that defines the depth of the recursion (the maximum topological distance) and thus the number of affected vertices. The **c)** approach also allows us to have several emitting vertices, as shown in **d)** where two vertices are emitting a gradient, one with strength 3, and one with strength 5. In **d)** the graph algorithm is designed such that once a vertex has been assigned a value, only higher values can be assigned to that vertex.

To illustrate how this can be used as a gradient, in **e)** arrows point to the edge leading to the vertex with highest value. We see that if we start at any vertex in the graph and follow the arrows, we end up at one of the emitting vertices. Also we see that the vertex with a higher emission strength has a larger basin of attraction than the vertex with lower emission strength.

3 Agent-Based Hormone Gradients

The previous section briefly described how traditional graph traversal algorithms can produce a loose analogy to biological hormone gradients. However, in a physically distributed system we cannot directly use graph traversal algorithms.

This paper presents a distributed algorithm, providing information on topological distance in communication networks with dynamic changing topologies. Consider an agent

for each vertex in the graph, where agents asynchronously and without any global knowledge, efficiently realizes such graph hormone gradient information.

3.1 Related Work

Hormone gradients have been studied by biologists and have been found to be a key player in coordinating and controlling growth in plants and animals. Mainhardt and Gierers' [Meinhardt, 1993] [Meinhardt and Gierer, 2000] work on how hormone gradients control the forming of the tentacles of the Hydra animal describes how hormones are responsible for almost all aspects of shaping the animals body. Other relevant work includes Nishimura's [Nishimura and Sasai, 2004] work on chemo-taxis on cells in solutions with chemical gradients.

Several researchers have extracted and abstracted these mechanisms for use in the multi-robot domain. Amorphous computing [Abelson *et al.*, 2000][Nagpal, 1999][Nagpal *et al.*, 2003] is one example of research relying heavily on simulated hormone gradients, and many more examples exist, including work on cell division in artificial evolution [Hotz, 2004], and in self-reconfigurable robotics, such as Bojinov, Casal and Hogg's *scents* [Bojinov *et al.*, 2000].

All the implementations of digital hormone gradients in the distributed robotics literature rely on continuous broadcasting of a hop-count number. When an agent receives a message with a hop count number, it remembers this number and starts emitting a message with an incremented or decremented hop count number, at regular intervals. A timer keeps track on when the last message matching the stored hop count number was received. If too long time has expired, the stored hop count number is erased, and the agents stop sending messages.

As a first approximation, this continuous broadcasting approach works fine. However, there are two main problems with this approach:

- Frequent emission of messages, even if there is no change to the topology of the agents.
- Continuous broadcast style approaches can fail to deal adequately with dramatic changes to the topology.

The first point has to do with efficiency in terms of power consumption and processing power. The second point, illustrated in figure 2, is more serious.

For these two reasons, it is desirable to develop a better algorithm.

4 Features of the Algorithm

The algorithm described in this paper is an extension of E. Gafni and D. Bertsekas' work described in the paper "Distributed Algorithms for Generating Loopfree Routes in Networks with Frequently Changing Topology" [Gafni and Bertsekas, 1981]. Their algorithm has a number of properties that makes it interesting for the purpose of this paper:

1. The algorithm is distributed and works on unknown communication topologies.
2. Their algorithm spans a directed acyclic graph (DAG) over the network topology, where all directed paths lead to the destination.

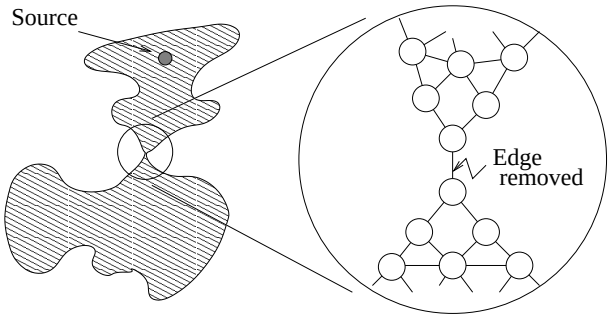

Figure 2: *Illustration of a case where continuous broadcast-style graph gradients is problematic. Assume a gradient source in the upper part of the illustrated graph, emitting a decrementing gradient (type c) from figure 1) with high enough intensity to cover the graph. When the edge is removed, we would like gradient information to disappear from the amputated part of the graph. However, depending on the exact implementation of the broadcast-style approach, the gradient hop-count number might instead drift (slowly) towards infinity or zero. The problem is, that when one vertex has timed out, it might receive an new gradient message from one of its neighbors, keeping the gradient "alive".*

3. In the event of changes to the topology of the network, the algorithm creates a new DAG using an iterative method.

From a functional point of view, the algorithm presented in this paper is an improved version of the algorithm presented by Gafni and Bersekas, with two additional features;

4. From a given vertex, the length of any directed path to the destination is equal to the shortest undirected path to the destination.
5. Each vertex knows its topological distance from the destination.

4.1 Definitions

The algorithm described below works on a graph, to create a loose analogy to chemical/hormone gradients. The algorithm is completely distributed, and is able to deal with frequently changing topology, while at the same time using very little communication. Consider an *agent* for each vertex v in the graph $G = (V, E)$. Agents have a number of ports, through which they can communicate to other agents. The algorithm uses two features per agent. Agents can emit a hormone with strength s and sense the intensity i of the hormone. Both s and i are in $\mathbb{N}$, the set of natural numbers. We require agents to know which of their ports are connected to other agents, and which are not. Given this, a digital hormone gradient can be defined as a four-tuple, $\{G, I, S, P\}$, where:

- $G = (V, E)$ is a graph, where vertices are elements of the configuration, and edges are physical connections between the elements.
- $I : V \rightarrow \mathbb{N}$ is the hormone intensity (concentration) mapping, so that $i = I(v)$ is the hormone intensity for vertex v.

- $S : V \rightarrow \mathbb{N}$ is the hormone emission strength mapping, where $s = S(v)$ is the strength s with which vertex $v \in V$ emits the hormone.

- $P : (V \times \mathbb{N}) \rightarrow (\{\overline{\diamond}\} \bigcup V)$ is the port function. The port function holds information for the state of each of the agent's ports, so that $P(v, n)$ is the state of the agent for vertex v's n'th port. The symbol $\overline{\diamond}$ is used to specify a port that is not connected, such that $\forall n \in \mathbb{N} : \forall v \in V : \forall v' \in P(v, n), \{v, v'\} \in E \Leftrightarrow P(v, n) \in V$.

The four-tuple $\{G, I, S, P\}$ describes the global state of the hormone gradient, but we are aiming at a distributed algorithm with only local information. For any given vertex $v \in V$, the local gradient state information available is $i = I(v), s = S(v)$ and for each of the j ports, $p_n = P(v, n)$ where $0 \leq n < j$. Besides this information, the algorithm requires agents to remember their parent in the spanning tree $-1 \leq f < j$ (using -1 for no parent), a timer/counter $t \in \mathbb{N}$ and an array for storing information on which ports to send messages to, $m[n] \in \{NONE, PROPAGATE\}$.

The algorithm basically works by maintaining local information on the spanning tree, such that the variable f always points to the port from which the gradient is flowing. The timer/counter is used to adjust the relative speeds of propagating gradient and deleting the gradient. Without this adjustment continuous propagation and deletion of the gradient in local cyclic areas of the graph might arise, causing much communication and slow hormone deletion. However, this adjustment introduces a parameter that is the number of time steps to delay propagate messages when spreading the gradient. Lower values cause the algorithm to be faster but more unstable.

The algorithm for each agent, implementing the desired graph-based hormone gradient behavior, is given below. Note that neither global communication nor synchronization between agents is assumed.

The **—initialize—** function resets the variables of the agent. **—emitGradeint—** is used to set the hormone emission strength of the agent. Setting to zero clears the gradient. **—updateGradient—** holds the central functionality such as sending messages, and should be called at regular time intervals. The frequency at which it is called determines the hormone propagation speed in the system. The four event functions **—connectEvent—**, **—disconnectEvent—**, **—propagateMessageRecievedEvent—** and **—deleteMessageRecievedEvent—** are very light-weight and can be called in some interrupt routine whenever the corresponding event is detected.

—initialize()—
$i := 0; s := 0; f := 0; t := 0;$
foreach $n \in \{0..j-1\}$ **do**
  $m[n] := NONE;$
end

—emitGradient(s')—
$s := s';$

—connectEvent(n)—
$m[n] := PROPAGATE;$

—disconnectEvent(n)—
if $n = f$ **then**
  $f := -1;$
end

—updateGradient()—
if $s > i$ **then**
  $i := s; f := -1;$
  foreach $n \in \{0..j-1\}$ **do**
    $m[n] := PROPAGATE;$
  end
else if $s < i \wedge f = -1$ **then**
  foreach $n \in \{0..j-1\}$ **do**
    if $p_n \neq \overline{\diamond}$ **then**
      sendDeleteMessage(n,i-1);
    end
    $m[n] := NONE;$
  end
  $i := 0;$
else if $t > 0$ **then**
  $t := t - 1;$
  if $t = 0$ **then**
    foreach $n \in \{0..j-1\}$ **do**
      if $n \neq f$ **then**
        $m[n] := PROPAGATE;$
      end
    end
  end
else
  foreach $n \in \{0..j-1\}$ **do**
    if $p_n \neq \overline{\diamond} \wedge m[n] = PROPAGATE \wedge i > 1$ **then**
      sendPropagateMessage(n,i-1); $t := 0;$
    end
    $m[n] := NONE;$
  end
end

—propagateMessageRecievedEvent(n', i')—
if $i' > i$ **then**
  $i := i'; f := n'; t :=$ **3**;// This constant (3) defines the delay before propagating the gradient, to allow delete messages to "catch up".
else if $i' < i - 1$ **then**
  $m[n'] := PROPAGATE;$
end

—deleteMessageRecievedEvent(n', i')—
if $i' \geq i$ **then**
  $f := -1;$
else
  $m[n'] := PROPAGATE;$
end

4.2 Tests of the Algorithm

The algorithm was implemented and tested on graphs with varying sizes and numbers of loops, shown in figure 3. To test the robustness of the algorithm, the different activation schemes described in [Butler *et al.*, 2002], D_0,D_1 and D_∞, were used for the agents. In D_0, agents are activated in a fixed order each time step, in D_1 the agents activation order is permuted for each time step, and in D_∞ a random agent is picked and activated n times each time step, where n is the number of agents. Figure 4 shows the number of messages passed as a function of the number of modules for three different graph types, using the D_∞ activation scheme. The

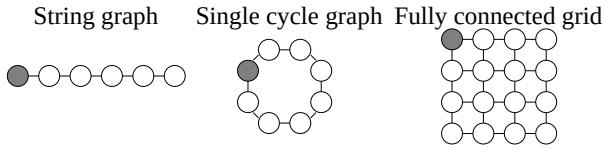

Figure 3: *The three graph types used to test performance of the distributed hormone algorithm. The gray vertex is the source of the gradient.*

data shows that the algorithm has good performance, in that the number of messages passed is linearly proportional to the number of edges in the graphs, and that the number of time steps used to propagate the hormone is proportional to the diameter of the graph.

A different set of experiments were performed for counting the number of messages used to remove the gradient again, by setting the emission strength of the emitting agent to zero. These results are shown in figure 5. The results show that the number of messages generated for removing a hormone is linearly proportional to the number of involved graph edges. Notice that removing the hormone is faster than covering the graph, due to the delay on propagating the gradient.

5 Pose Hormone Gradients

The previous section described the basic graph hormone gradient algorithm. This section describes how the algorithm has been used for controlling two self-reconfigurable robot systems, shown in figure 6. Self-reconfigurable robot control is a good application test bed for this kind of hormone gradient algorithms, because of the constantly changing topology in such a robot, and because of similarities between self-reconfigurable robots and multi-cellular organisms.

5.1 Common Pose for Self-Reconfigurable Robot Modules

The distributed hormone gradient algorithm described above can be expanded slightly to be used for propagating pose (position and orientation) relative to the emitting vertex in a self-reconfigurable robot. Basically, the expanded algorithm uses feed-forward kinematics in the generated directed acyclic graph to compute the relative pose to the emitting vertex. To realize this, it is a requirement that the relative pose between two connected vertices in the configuration can be found. We let the space $\mathbb{P}$ be the space of relative poses in a 3D vector space.

Then, a relative pose hormone is a four-tuple, $\{G, \mathcal{I}, \mathcal{S}, \mathcal{P}\}$ where:

- $G = (V, E)$ is a graph, where vertices are elements of the configuration, and edges are physical connections between the elements.
- $\mathcal{I} : V \rightarrow (\mathbb{N} \times \mathbb{P})$ is the hormone intensity (concentration) mapping, so that $i = I(v), i = \langle i_n, i_p \rangle$, where $i_n \in \mathbb{N}, i_p \in \mathbb{P}$ is the hormone intensity i_n and relative pose i_p for vertex v.

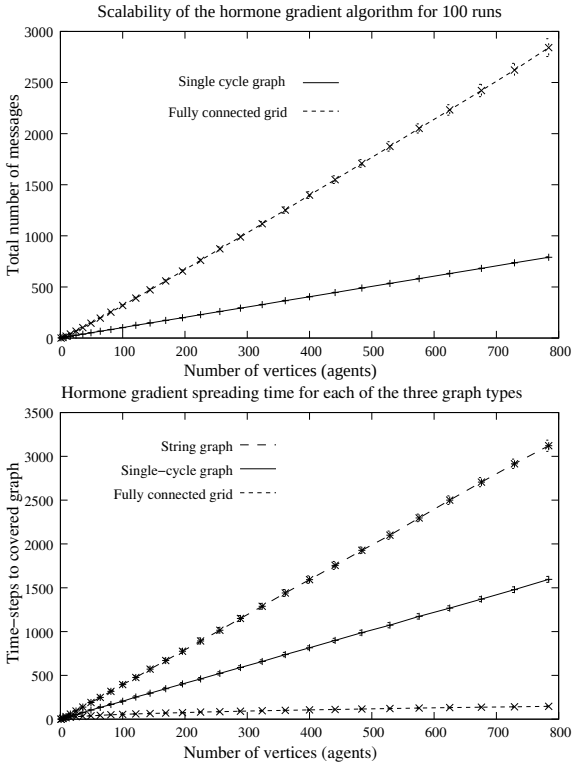

Figure 4: **Top:** *Message count for covering a graph with the hormone gradient, for varying number of vertices. Results for the string graph type is not shown, since they turned out to be the same as the results for the single-cycle graph. The variance is caused by using the D_∞ activation scheme.* **Bottom:** *The number of time steps it takes to fully cover the graph for each of the three graph types and for varying graph sizes.*

- $\mathcal{S} : V \rightarrow (\mathbb{N} \times \mathbb{P})$ is the hormone emission mapping, where $s = \mathcal{S}(v), s = \langle s_i, s_p \rangle$ is the emission strength (s_i) and the emitted pose (s_p) for vertex v.
- $P : (V \times \mathbb{N}) \rightarrow (\{\overline{\diamond}\} \bigcup (V \times \mathbb{P}))$ is the port function. The port function holds information for the state of each of agent's ports, so that $P(v, n)$ is the state of vertex v's n'th port. The symbol $\overline{\diamond}$ is used to specify a port that is not connected, such that $\forall n \in \mathbb{N} : \forall v \in V : \forall \langle v', p' \rangle \in P(v, n) : \{v, v'\} \in E \Leftrightarrow P(v, n) \in V$. Here, p' is the relative orientation between the two connected elements, such that if $p_{v'}$ is the pose of v' and p_v is the pose of v, then $p_{v'} \times p' = p_v$.

For any given module $v \in V$, the local gradient state information available is $i = \langle i_n, i_p \rangle = \mathcal{I}(v), s = \langle s_i, s_p \rangle = \mathcal{S}(v)$ and for each of the modules j connectors, $p_n = P(v, n)$ is either $\overline{\diamond}$ or $\langle v', p' \rangle$, for $0 \leq n < j$.

The principle is to do feed-forward kinematics in the directed acyclic graph generated by the graph hormone algorithm. As a consequence, each module affected by the hormone knows its relative position and orientation to the hormone source. It turns out that by exploiting that each module knows its relative position to the source, it is possible to re-

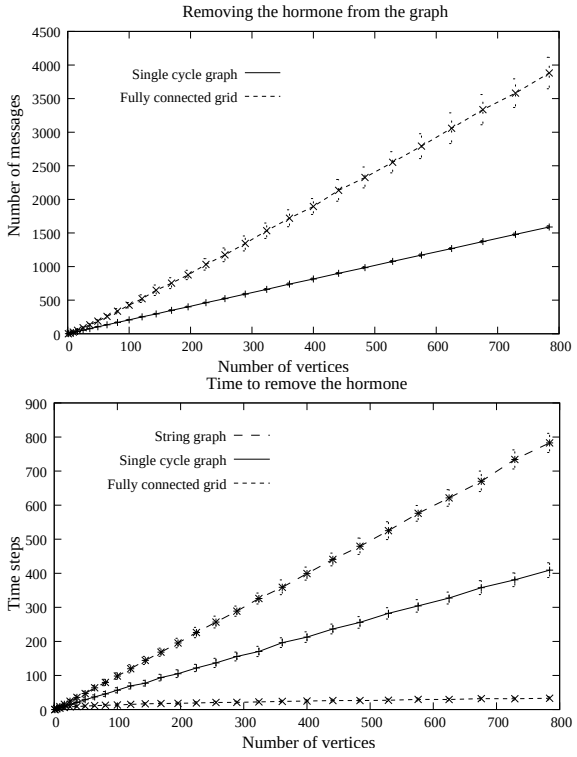

Figure 5: *Tests on removing the hormone from the graph by setting the emission strength of the emitting vertex to zero, after the hormone has fully propagated. The two graphs show the number of messages generated and the time used for the three graph families and for varying graph sizes. In the top graph the string graph results were omitted since they are identical to the single cycle results. Notice that removing the hormone is faster than propagating the hormone, but produces more communication and has higher variance. This relationship can be adjusted by changing the propagation delay.*

move the delay for propagate messages by adding an extra condition for the case of receiving a propagate message. The implementation details of the pose hormone gradient algorithm has been omitted due to space constraints.

6 Implementation and Experiments

The pose hormone algorithm has been implemented and tested on simulations of the ATRON system, figure 6, left, and on the M-TRAN system, figure 6, right.

Both the ATRON and the M-TRAN modules are composed of two parts, where the parts are connected by an actuated joint. The algorithm is implemented, such that each part of the modules corresponds to a vertex in the graph. The following will describe some experiments done on the pose hormone algorithm on the simulated M-TRAN robots.

6.1 M-TRAN, Message Count

Three test configuration designs are shown in figure 7. Each configuration comes in a form with 4, 32, 108, 256, 500 and

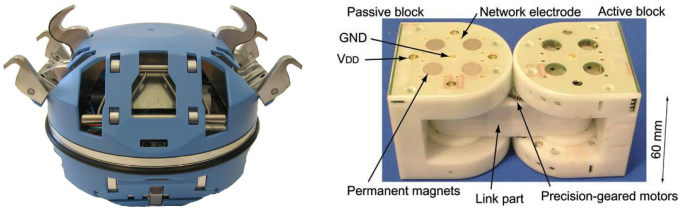

Figure 6: *Module for two self-reconfigurable robot systems.* **Left:** *The ATRON system.* **Right:** *The M-TRAN System.*

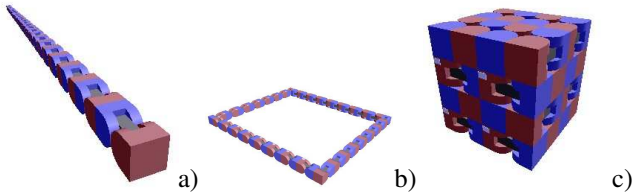

Figure 7: *The three test configurations used in the synthetic tests. Here composed of 32 modules each.*

864 modules. The three designs differ in the number of loops in the topology, where a) has no loops, b) a single loop and c) has many loops.

An experiment proceeds as follows:

1. A module in one end/corner is set to emit a hormone gradient with an emission strength of 1000.
2. Data is collected on the number of messages used to fully propagate the gradient, and on how many time steps it took.
3. Then the emission strength of the emitting module is set to zero.
4. Similarly to step two, data is collected.

The algorithm was tested using the D_0, D_1 and D_∞ activation schemes. Ten experiments were performed for each combination of configuration and activation scheme, and mean and variance were computed. The correctness of the pose information is easily verified by placing the emitting module such that the emitted pose is identical to the pose in the simulator. In this case, the relative pose hormone for any module should be identical with its pose in the simulator (provided appropriate scaling). Figure 8 shows the average number of messages, and figure 9 the communication load per connector, using the D_1 activation scheme. For deleting the hormone gradient, the number of messages passed by any connected connector was constant two.

Besides these simple tests, the algorithm was used a lot during work on control of the two self-reconfigurable robot systems, to produce a common coordinate frame of reference for locally cooperating groups of modules.

7 Discussion & Conclusion

This paper describes an efficient algorithm for realizing an abstract implementation of hormone gradients. Hormone gra-

Test Configuration	a)			b)			c)		
# modules	4	32	108	4	32	108	4	32	108
# connections	3	31	107	4	32	108	4	64	252
# messages for propagating	7	63	210	8	66	221	9	115	455
# message for deleting	14	126	420	16	128	432	16	193	720

Figure 8: *The average number of messages (rounded to nearest integer) used to propagate and delete a digital hormone gradient for nine different configurations using the D_1 activation scheme. The messages passed between "agents" within the same module are included in the data, so since each module has two "agents". The number of messages to propagate the gradient in configuration a) is exactly the number of modules plus the number of connections between modules. The data also shows that the number of messages for deleting a gradient is approximately double that of propagating the gradient.*

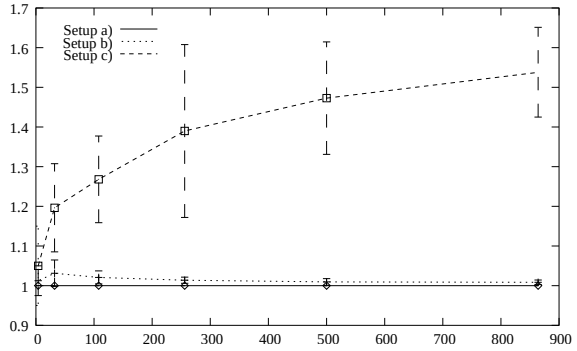

Figure 9: *Average message count per connector for propagating the hormone gradient for each of the configurations, using the D_1 activation scheme. The graph shows that configuration design c) with many topology graph cycles causes more messages per connector than a) and b). However, the message count per connector is still lower than two.*

dients are widely used in biology to coordinate and differentiate cellular behavior.

The described algorithm is based on topological distance, and is capable of dealing with changing topology in the connectivity graph using only local information and without requiring global synchronization.

The algorithm has properties similar to that of Gafni and Bertsekas, with the added benefit that the spanning DAG is, in a sense, optimal, in that any directed path tho the source has the same length as the shortest undirected path. Additionally, in contrast to the commonly used continuous broadcast method, the presented algorithm only generates communication when the topology of the graph changes, or when the gradient information changes. Furthermore, only a constant (and small) number of messages is generate per graph edge, in the event of any change to the hormone gradient. Less communication results in reduced hardware requirements, that again can reduce requirements to power, processing, price and size.

Performance of the described algorithm is documented through a number of experiments. An expansion of the hormone gradient algorithm is given which can be used to provide all self-reconfigurable robot modules affected by the hormone their pose relative to the emitting module. The algorithm was developed for use in self-reconfigurable robotics, but might very well be useful for many other applications. Future work will include further analysis of performance and analytical proofs.

References

[Abelson *et al.*, 2000] H. Abelson, D. Allen, D. Coore, C. Hanson, G. Homsy, T. Knight, R. Nagpal, E. Rauch, G. Sussman, and R. Weiss. Amorphous computing. *Communications of the ACM*, 43(5):74–82, 2000.

[Bojinov *et al.*, 2000] H. Bojinov, A. Casal, and T. Hogg. Multiagent control of self-reconfigurable robots. In *Fourth International Conference on MultiAgent Systems (ICMAS)*, pages 143–150, Boston, Massachusetts, USA, July 2000.

[Butler *et al.*, 2002] Z. Butler, K. Kotay, D. Rus, and K. Tomita. Generic decentralized control for a class of self-reconfigurable robots. In *Proceedings, IEEE Int. Conf. on Robotics and Automation (ICRA)*, volume 1, pages 809–815, Washington, DC, USA, 2002.

[Gafni and Bertsekas, 1981] E. Gafni and D. P. Bertsekas. Distributed algorithms for generating loopfree routes in networks with frequently changing topology. *IEEE Trans. on Comm.*, COM-29:11–18, 1981.

[Hotz, 2004] P. E. Hotz. Asymmetric cell division in artificial evolution. In *Congress on Evolutionary Computation (CEC)*, 2004.

[Meinhardt and Gierer, 2000] H. Meinhardt and A. Gierer. Pattern formation by local self-activation and lateral inhibition. *Bioessays*, 22(8):753–60, Aug 2000.

[Meinhardt, 1993] H. Meinhardt. A model for pattern formation of hypostome, tentacles, and foot in hydra: how to form structures close to each other, how to form them at a distance. *Developmental Biology*, 157(2):321–333, June 1993.

[Nagpal *et al.*, 2003] R. Nagpal, A. Kondacs, and C. Chang. Programming methodology for biologically-inspired self-assembling systems. In *Proceedings, AAAI Spring Symposium on Computational Synthesis: From Basic Building Blocks to High Level Functionality*, 2003.

[Nagpal, 1999] R. Nagpal. Organizing a global coordinate system from local information on an amorphous computer. *AI Memo 1666*, 1999.

[Nishimura and Sasai, 2004] S. I. Nishimura and M. Sasai. Chemotaxis of an eukaryotic cell in complex gradients of chemoattractants. In *The ninth international symposium on artificial life and robotics (AROB)*, Beppu, Oita, Japan, January 2004.