

Evolution of Robust Data Distribution Among Digital Organisms

David B. Knoester, Andres J. Ramirez, Philip K. McKinley, and Betty H.C. Cheng
Department of Computer Science and Engineering
Michigan State University
East Lansing, Michigan 48824
{dk, ramir105, mckinley, chengb}@cse.msu.edu

ABSTRACT

This paper describes a study of the evolution of robust communication, specifically the distribution of data among individuals in a population, using digital evolution. In digital evolution, a population of self-replicating computer programs exists in a user-defined computational environment and is subject to instruction-level mutations and natural selection. To encourage the evolution of this cooperative behavior, we make use of “digital germlines,” a form of group-level selection similar to multicellularity in biology. The results of experiments using the AVIDA platform for digital evolution demonstrate that populations of digital organisms are capable of evolving to distribute data in a network, and that through the application of different selective pressures, these digital organisms can overcome communication obstacles such as message loss, limited bandwidth, and node failure.

Categories and Subject Descriptors

I.2.8 [Computing Methodologies]: Artificial Intelligence—*Problem Solving, Control Methods, and Search*; D.1.3 [Programming Techniques]: Concurrent Programming—*Distributed programming*; F.1.1 [Computation by Abstract Devices]: Models of Computation—*Self-modifying machines*

General Terms

Experimentation.

Keywords

Digital evolution, cooperative behavior, natural selection, multilevel selection, mutation, germline, biologically-inspired computing, communication, distributed systems.

1. INTRODUCTION

The natural world is replete with organisms that exhibit cooperative behaviors of varying complexity. For example,

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

GECCO '09, July 8–12, 2009, Montréal Québec, Canada.
Copyright 2009 ACM 978-1-60558-325-9/09/07 ...\$5.00.

nearly all species of microorganism cooperate to form extracellular structures called biofilms [1], and many social insects cooperate to build complex nests [2]. Many such cooperative behaviors involve some form of communication, for example, the alarm drumming of ants [3], the synchronization of firefly flashes [4], and quorum sensing in bacteria [5].

Many of the communication challenges faced by biological organisms are shared with distributed computing systems. For example, node churn, message loss, and network segmentation all have biological analogues in organism death, sensory impairment, and environmental dangers. Because of these close parallels, and the remarkable effectiveness of biological organisms in overcoming these hazards, recent approaches to designing distributed systems have focused on *biomimetics*, where the observed behaviors found in nature are replicated *in silico* [6]. A complementary approach to biomimetics is *evolutionary computation*, where instead of mimicking the behaviors found in nature, we harness the power of evolution and natural selection, the processes that produced those behaviors. Although still relatively early in its application to distributed systems, evolutionary computation enables the designer to search an enormous solution space, often revealing robust and non-intuitive solutions [7].

Although the evolution of communication has been studied in both biology [8] and evolutionary computation [9–11], the effects of various forms of communication failures on the evolutionary process are not yet fully understood. This paper describes a study, conducted using digital evolution [12], that examines the effects of communication failures on the evolution of a simple distributed behavior, that of distributing data from a single source to all individuals in a population. In digital evolution, a population of digital organisms exists in a user-defined computational environment. These organisms replicate, compete for resources, and are subject to mutation and natural selection. Over thousands of generations, they can evolve to survive, and even thrive, under extremely dynamic and adverse conditions. In this study we employ AVIDA [13], a digital evolution platform previously used in biology to study the evolution of complexity [14] and phenotypic plasticity [15].

To evolve a cooperative behavior for data distribution, we made use of a facility in AVIDA that enables digital organisms to exchange messages. This messaging facility includes two components, one that controls *what* data may be exchanged between organisms, and another that describes *which* organisms may exchange data. This second component takes the form of an underlying network describing the possible paths messages may follow among organisms. For

this study, we extended AVIDA with new types of underlying networks, as well as various communication hazards, including message loss, bandwidth restrictions, node churn, and network topology changes.

Our experiments demonstrate that populations of digital organisms are capable of evolving to distribute data in a network. Furthermore, these digital organisms were able to efficiently distribute data despite of the presence of communication hazards such as message loss, node churn, and bandwidth limitations. Our results indicate that evolutionary computation in general, and digital evolution in particular, offer a promising approach to designing cooperative computing systems, as well as to improve our understanding of the evolution of communication in natural organisms.

2. DIGITAL EVOLUTION AND AVIDA

Digital evolution [12] is a form of evolutionary computation that has a close relationship to evolution in biology, both from an architectural perspective (digital organisms within AVIDA self-replicate), and from an application perspective (AVIDA has primarily been used to study the evolution of complexity [14] and phenotypic plasticity [15] in nature). Recently, AVIDA has been used to study various types of cooperative behaviors for application to computing systems, including information gathering [16] and network construction [17]. In the remainder of this section, we give a brief overview of AVIDA.

2.1 Digital Organisms

Figure 1 depicts an AVIDA population and the structure of an individual organism. Each digital organism comprises a circular list of instructions (its *genome*) and a virtual CPU, and exists in a common virtual environment. Within this environment, organisms execute the instructions in their genomes, and the particular instructions that are executed determine the organism’s behavior (its *phenotype*). Different AVIDA CPU architectures have been implemented and used in various studies [13]. The architecture used in this study contains a circular list of three general-purpose registers {*AX, BX, CX*}, two general-purpose stacks {*GS, LS*}, and four special-purpose *heads*. Heads may be thought of as pointers into the organism’s genome and are similar to a traditional program counter and stack pointer. The instruction-head points to the next instruction to be executed. The flow-control head points to a location in the genome to which the instruction-head may be moved upon execution of certain instructions; it is similar to a `goto` label in the C programming language, but may be dynamically changed during execution. The read-head and write-head are primarily used during replication, and enable the organism to read and write instructions within its genome.

Instructions within an organism’s genome are similar in appearance to a traditional assembly language. These instructions enable an organism to perform simple mathematical operations, such as addition and multiplication; to manipulate heads within their genome; to sense and change properties of the environment; and to communicate with neighboring organisms. Certain instructions also enable organisms to replicate, subject to mutation, and thus spread throughout the population. Instructions in AVIDA can have different costs in terms of virtual CPU cycles. For example, a simple addition may cost only one cycle, while broadcasting a message may cost 20 cycles. A key property of AVIDA’s

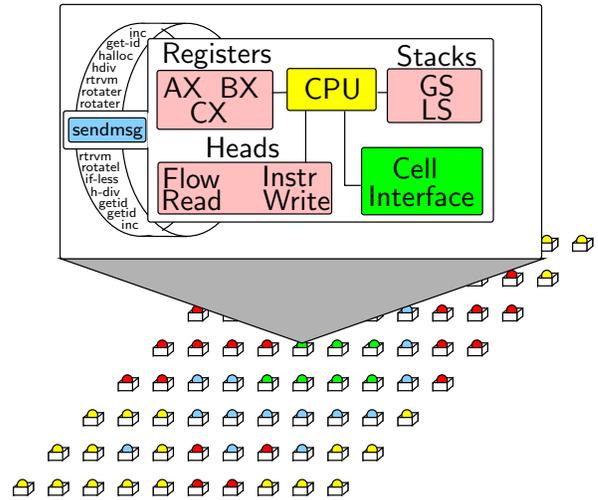


Figure 1: An Avida population containing multiple genomes (bottom), and the structure of an individual organism (top).

instruction set that differs from traditional computer languages is that it is not possible to construct a *syntactically incorrect* genome in AVIDA. Hence, while random mutations may produce many genomes that do not perform useful computation, their instruction sequences are always executable.

As shown in Figure 1, each organism in AVIDA lives in a *cell* located in a fixed location within a spatial environment. Each cell can contain at most one organism; organisms cannot live outside of cells. The topology of the environment defines the neighborhood of each cell, and is user-defined. For example, the environment topology may be configured as a grid, a torus, or as a well-mixed environment, where all cells are neighbors of each other (also referred to as a clique). Furthermore, each organism in the environment has a *facing* that defines its orientation. This facing may be used in a number of different ways. For example, an organism can send a message in the faced direction. The organism can also sense and manipulate its facing via the `get-facing` and `rotate*` instructions, respectively.

2.2 Levels of Selection

Figure 2 depicts the three different levels of selection available within AVIDA. Under the first, individual selection, organisms compete with each other for space (cells) in their environment and are responsible for their own replication; that is, organisms must execute instructions to self-replicate. In the second level, group selection, the population of digital organisms is divided into distinct subpopulations, called *demes*. Within each deme, organisms replicate, mutate, and compete with each other for space and resources. At the same time, demes also compete with each other for space and resources based on the behavior of their constituent organisms. Competition between demes can either be *synchronous*, where periodic selection is applied based on a deme-level fitness function and using traditional selection techniques, or *asynchronous*, where the behavior of each deme is monitored for a user-defined event that triggers the deme’s replication. Normally, when a deme replicates, all of its constituent organisms are replicated as well. The third level of selection available within AVIDA is most similar to

multicellularity in biology. Here, the population is again split into demes, however the organisms within each deme are homogeneous. In this case, a genome is attached to each deme, and all organisms within the deme are instantiations of that same genome. When a deme replicates, any mutations occur to the deme’s genome, which is called a *digital germline* [17]. As in the group selection case, replication when using a germline can be either synchronous or asynchronous.

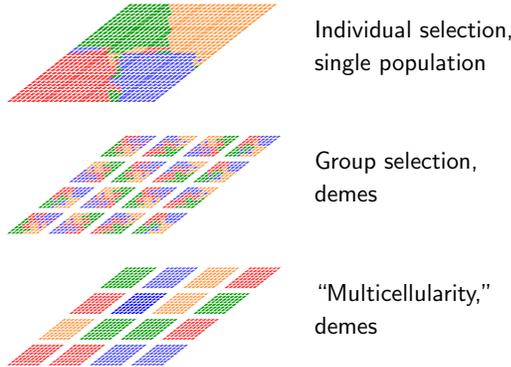


Figure 2: Levels of selection available within Avida; different shades represent different genomes.

For this study, we used *CompeteDemes*, a framework within AVIDA that enables the periodic (synchronous) replication and competition of demes, in combination with a digital germline to ensure homogeneity within demes. During the execution of an AVIDA trial, the *CompeteDemes* framework periodically calculates the fitness of each deme via a user-defined fitness function. This fitness function takes as input a single deme and all the organisms within, and produces the fitness of that deme (a floating-point number) as output. Using the resulting array of fitness values, one fitness value for each deme, the *CompeteDemes* framework then performs fitness-proportional selection, preferentially replicating those demes with higher fitness, and replacing those demes with lower fitness. For example, one may define a fitness function based on the degree of data distribution exhibited by the organisms within a deme. Over time, the *CompeteDemes* framework will then preferentially replicate those demes that are more capable of data distribution than others, resulting in a population that evolves toward higher levels of data distribution.

3. METHODS

Typically, the AVIDA user will configure the environment in which the digital organisms live and define the selective pressures that act upon the population. Once configured, multiple AVIDA trials are conducted to account for the stochastic nature of evolution. In the remainder of this section, we describe the configurations and extensions to AVIDA that were required for this study of the evolution of data distribution.

3.1 Communication

Organisms within a deme are connected via an underlying network that enables point-to-point communication. As shown in Figure 3, this network may be any of a variety of

topologies, including a torus, clique, scale-free, and a random connected network. In a torus topology, every cell is linked to eight neighbors, while in a clique (also called a complete graph or K_n), every cell is linked to every other cell. In a random network, cells are first connected via a random tree, with a configurable number of additional links connecting random pairs. Finally, scale-free networks, characterized by few cells with a large number of links (so-called “hubs”) and many cells with a small number of links, are constructed via *preferential attachment* ($\alpha = 1.0, m = 3$; configurable) [18]. Based on the topology of their network, organisms can send messages to any of their neighbors.

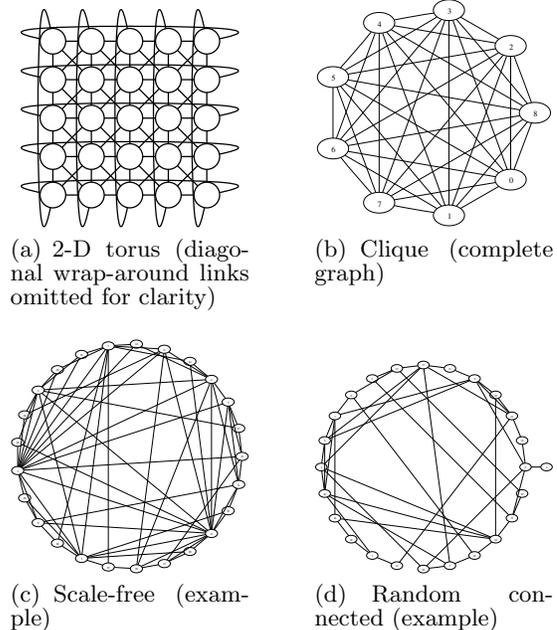


Figure 3: Topologies available within Avida. Torus, scale-free, and random topologies shown with 25 nodes; clique topology shown with 9 nodes.

An example of data distribution is depicted in Figure 4. Figure 4(a) depicts 25 organisms “living” in a deme. A single cell in each deme is designated as the initial source of data (a random number) for that deme. The individual living in that cell, shaded in this figure, is responsible for initiating data distribution by collecting the data in its cell and sending messages containing that data to its neighbors. In Figure 4(b), we see that neighbors of the source have received the data, again indicated by shading, and have in turn forwarded it to other individuals. The number of organisms that receive this data will be used as the basis for fitness functions that will be presented later. Finally, in Figure 4(c), we see that the data has been distributed to all individuals within the deme. At this point, the data distribution task is complete.

To study the evolution of data distribution in the presence of communication failures, we introduced a number of different network hazards. Congested network conditions are simulated by increasing the cost, in terms of virtual CPU cycles, of sending a message. Random message loss is used to examine the effect of messages that are lost during transmission, for example as a result of environmental interference.

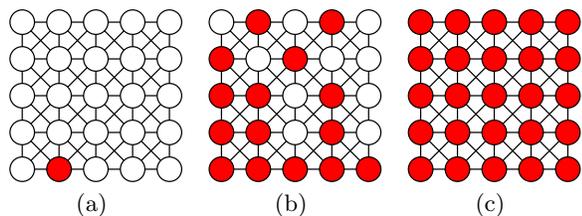


Figure 4: Depiction of data distribution in a 5×5 torus (wrap-around links omitted for clarity). In (a), data is present at a single cell only, shown by the shaded circle; (b) depicts organisms in the midst of distributing data; finally, (c) shows complete distribution of data.

Finally, node churn, where randomly selected organisms are “killed” and replaced, addresses the effect of organism death on communication, akin to transient nodes in a network.

3.2 Instructions

While no new virtual CPU instructions were needed for this study, we leveraged several mechanisms that were developed for previous studies in the evolution of distributed problem solving [16, 19]. These instructions are summarized in Table 1. Specifically, we were able to employ the messaging- and opinion-related instructions to provide communication primitives and to indicate when data has reached its destination.

Table 1: Relevant instructions for this study. All instructions are equally likely to be selected as targets for mutation.

Instruction	Description
send-msg	Sends a message to the neighbor currently faced by the caller; message contains BX and CX .
retrieve-msg	Loads the caller’s virtual CPU from a previously received message; loads BX and CX .
collect-cell-data	Loads BX with the value sensed from the caller’s cell; zero if the caller is not the source.
get-opinion	Load BX with the value of the caller’s current opinion flag.
set-opinion	Set this organism’s opinion flag to the value in BX .
rotate-left-one	Rotate this organism counter-clockwise one step.
rotate-right-one	Rotate this organism clockwise one step.

3.3 Configuration

We configured AVIDA to use the `CompeteDemes` process, outlined in Section 2.2, with specific configuration values summarized in Table 2. We used 400 demes, each comprising 25 digital organisms connected via one of four different network topologies, with each deme configured to use a germline to provide homogeneity within demes. The relatively small size of each deme was selected primarily for practical (i.e., computation time) purposes, and the default values were used for all mutation rates.

Table 2: Common Avida configurations used for the experiments described in this study.

Configuration	Value
Trials per experiment	30
Max. population size	10,000
Number of demes	400, each 5×5
Environment topology	Torus, clique, scale-free, or random
Copy mutation rate	0.0075 (per instruction)
Insertion mutation rate	0.05 (per replication)
Deletion mutation rate	0.05 (per replication)
Time slice	5 instructions per update, constant-time scheduler
CompeteDemes	Compete all demes using fitness-proportional selection (periodic, every 400 updates).

In order to control the initial conditions for each deme, we configured AVIDA to disallow self-replication of individual organisms. Each time a deme replicated as a result of the `CompeteDemes` process, the offspring deme was filled with 25 copies of the latest (possibly mutated) genome from the germline. We note that deme replication does *not* necessarily include the underlying network; in the case of random and scale-free networks, offspring are born into a new network. As the final step of AVIDA configuration, we defined the *default ancestor*, which is the starting organism for each deme within all AVIDA trials. Here, we used an organism that contained 100 `nop-C` instructions. The `nop-*` instructions perform no computation, and do not change the state of the virtual CPU. The presence of a large number of `nop-C` instructions in the default ancestor is common in AVIDA experiments, and provides evolution with a “blank tape” for mutating different instructions into the genome. We emphasize that although the default ancestor contains only 100 instructions in its genome, not only can mutations increase and decrease genome size, the genome itself is circular; once the organism executes the final instruction, execution flow wraps around to the beginning of the genome.

4. RESULTS

In this section, we describe the experiments we conducted to evolve solutions to the problem of data distribution. Each of the experiments focuses on a single aspect of this problem, and each was designed to provide insight into how different environmental factors affect the evolution of this cooperative behavior.

4.1 Distribution of data

Our initial experiment examined the effect of different underlying topologies on the evolution of data distribution behavior, without regard to efficiency of message use. Specifically, we varied the underlying topology between torus, clique, scale-free, and a random connected network (mean degree approximately 3.25), while employing a fitness function based solely on the number of individuals that received the data. We defined the fitness function for each deme as:

$$F = (1 + U)^2 \quad (1)$$

where F is the resulting fitness of the deme and U is the number of organisms within the deme that indicate they

have received data. Organisms indicate that they have received the data by executing the `set-opinion` instruction, described in Table 1, with the received data passed as a parameter.

Figure 5 plots the grand mean fitness of each deme for each of the different underlying network topologies across 30 trials. In this figure, a black horizontal bar is used to indicate the fitness at which a deme would have successfully completed the data distribution task, in this case a fitness of 676 ($F = (1 + 25)^2 = 676$). While the grand mean fitness of each treatment is below this level, we note that at least one deme in each treatment successfully distributed data to all organisms by update 20,000. Although roughly the same number of organisms received the data in each of the four different topologies, suggesting that there is little difference in performance among the different topologies, we found the fitnesses of all treatment pairs to be significantly different from each other (Mann-Whitney U -test, $p < 2 \times 10^{-6}$), with the exception of the random and scale-free treatments, which performed similarly to each other. These results illustrate that organisms are able to evolve data distribution across networks of varying connectivity.

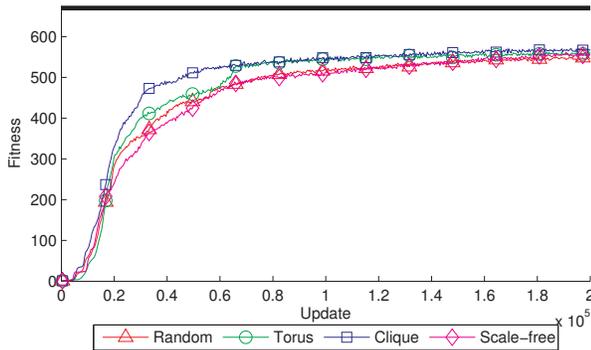


Figure 5: Fitness for random, toroidal, clique, and scale-free topologies when rewarding only for data distribution.

In this experiment, the dominant evolved genomes that solved the data distribution problem employed a fairly succinct looping behavior. Figure 6 depicts the AVIDA instructions for one such loop, drawn from a single trial of the random network treatment. Here, the entire loop is 11 instructions in length, approximately 10% of the genome; the remainder of the genome was “junk DNA,” remnants of earlier approaches to data distribution. As shown in Figure 6, organisms with this genome would either send messages that contained data sensed at their location within the deme, or forward messages that they had previously received. Additionally, each sent message was followed by a rotation so that the next message would be sent to a different neighbor.

4.2 Efficiency on differing topologies

In this experiment we examined the effect of different underlying topologies on the evolution of *efficient* data distribution. Specifically, we varied the underlying topology between torus, clique, scale-free and random connected network, while employing a fitness function based on both the number of individuals that received data as well as the number of messages that were used. We defined the fitness func-

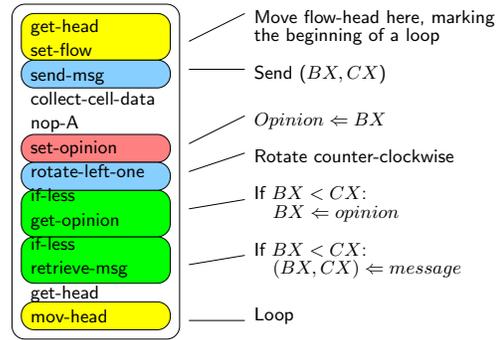


Figure 6: Representative genome fragment for solving the data distribution problem, drawn from a single trial using a random network topology. Prior to this loop, this genome contains instructions to test its location for the presence of data, setting its opinion if data is found.

tion for each deme as:

$$F = \begin{cases} (1 + U)^2 & \text{if } U < S \\ (1 + U + (S * 1000)/C)^2 & \text{if } U = S \end{cases} \quad (2)$$

where F is again the resulting fitness of the deme, U is the number of organisms within the deme that received and expressed the data, S is the number of organisms in the deme (always 25 in this study), and C is the total number of messages sent by organisms within the deme. We note that S is scaled by 1,000 to reward demes for completing the data distribution task even at the cost of a large number of messages (C).

Figure 7 plots the grand mean fitness of each deme for these treatments across 30 trials. As before, a black horizontal bar is used to indicate the fitness at which a deme successfully completed data distribution (note \log_{10} scale); any fitness improvements above this level are the result of efficiency gains. Here we see a difference in performance among the different network topologies, with the clique-based treatment reaching a fitness nearly 100 times that of the other treatments. Indeed, grand mean fitnesses of all pairs of treatments are significantly different ($p < 1 \times 10^{-6}$), and as a result we conclude that the underlying topology does play a significant role in the evolution of data distribution when efficiency is a concern.

In this experiment, the highest-performing strategy across all treatments evolved on a clique network, where only the organism at the location containing data sent messages; all other organisms remained silent, but actively listened for messages, setting their opinion once a message was received. Interestingly, the highest-performing deme on a torus network evolved a strategy that depended upon both the contents of messages, as well as the frequency with which messages were received. Pseudo-code for this strategy is shown in Algorithm 1. In effect, this strategy uses messages for *behavior suppression*, where receiving a message after an individual has set its opinion results in the suppression of future message sending.

4.3 Message cost

The purpose of this experiment was to ascertain the effect of different costs for sending messages across a network on

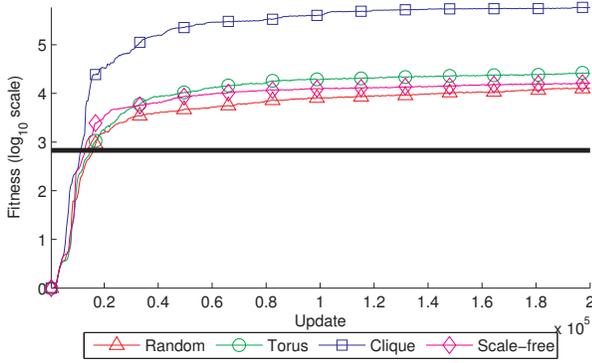


Figure 7: Fitness for random, toroidal, clique, and scale-free topologies when rewarding for efficient use of messages.

Algorithm 1 Representative evolved algorithm for data distribution in a torus when rewarding for efficiency.

Require: *opinion* is null; *location* is null for all but a single individual within each deme.
if *location* is not null **then**
 opinion \leftarrow *location*
end if
loop
 if *message* is received AND *opinion* is null **then**
 opinion \leftarrow *message*
 end if
 if *message* is not received AND *opinion* is not null **then**
 send *opinion*
 end if
 rotate
end loop

the evolution of data distribution behavior. For this and remaining studies, we use only the random connected network topology and employ the fitness function described in Equation 2. Here, however, we also varied the cost of sending messages from one organism to another while still rewarding for efficiently distributing data to all individuals. Specifically, we varied the cost of sending a message from 1 to 100 cycles per instruction, testing costs {1, 20, 40, 60, 80, 100}. Any message sending cost larger than 1 implies that sending a message is proportionally more expensive relative to other instructions, and delays the transmission of data through the network. Thus, a higher cost for sending a message is analogous to limiting the available bandwidth in a network.

Figure 8 plots the grand mean fitness of each deme for these treatments across 30 trials. As before, a black horizontal bar is used to indicate the fitness at which a deme successfully completed data distribution. These results illustrate that organisms are capable of evolving efficient behaviors for distributing data, regardless of message cost. In particular, as groups of organisms were placed in networks with higher message sending costs, they were able to evolve more efficient behaviors for data distribution. Although the grand mean fitness of most pairs of treatments were significantly different from each other ($p < 1 \times 10^{-6}$), treatments with a cost of 20, 40, and 60 were similar to each other.

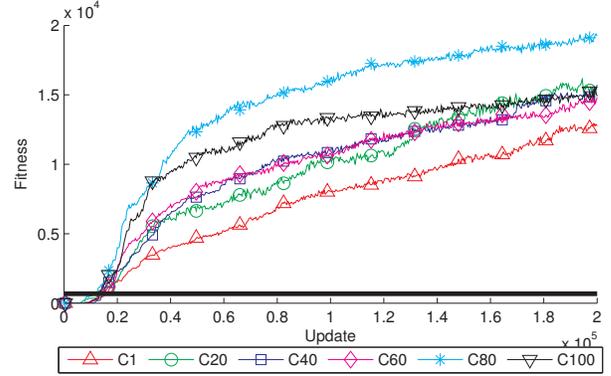


Figure 8: Fitness for a range of different message costs. The addition of a cost to send messages improves efficiency.

A side effect of increasing the cost of sending a message is that organisms have fewer opportunities to send messages, thus automatically providing a fitness advantage. Indeed, the evolved genomes verified this observation, as the strategies employed were similar to those described in Section 4.1, where efficiency was not rewarded. The treatment with a cost of 80, however, evolved a novel solution to the data distribution problem, where two interlocking loops were used to control organism behavior; pseudo-code is shown in Algorithm 2. The first loop, LOOP-A, is executed until the organism sets its opinion to either data contained at its location or the contents of a message, whichever contains a non-zero number. Once its opinion is set, however, the second loop, LOOP-B is executed, which repeatedly sends a message containing the organism’s opinion, and rotates the organism to face a new neighbor.

Algorithm 2 Representative evolved algorithm for data distribution in a random network when rewarding for efficiency and using a message cost of 80.

Require: *opinion* is null; *location* is null for all but a single individual within each deme.
LOOP-A:
 opinion \leftarrow *location* OR *message* {null treated as zero.}
LOOP-B:
 if *opinion* is null **then**
 goto LOOP-A
 end if
 send *opinion*
 rotate
 goto LOOP-B

4.4 Message loss

For this experiment, we simulated environmental interference by varying the rate of message loss. Specifically, we varied the probability (from 0 to 0.8 in intervals of 0.2) that a message would be lost. Here, each dropped message counts as a message that is successfully sent but never received at its destination. Given that we use the deme fitness function defined in Equation 2, dropped messages negatively affect fitness in two different ways: they both hinder data distribution and reduce the reward for efficient use of messages.

Figure 9 plots the grand mean fitness of each deme for these treatments across 30 trials. This plot illustrates that even moderate message loss rates result in significant drops in fitness. Somewhat surprisingly, we note that although fitness varies with message loss, message counts approach a common equilibrium across each treatment (data not shown). As in earlier experiments, the evolved behaviors followed a wait-and-forward strategy similar to that described in Algorithm 1, however the number of instructions present in these loops varied with loss rate, with higher loss rates generally having shorter loops, thus increasing the number of attempts to send messages.

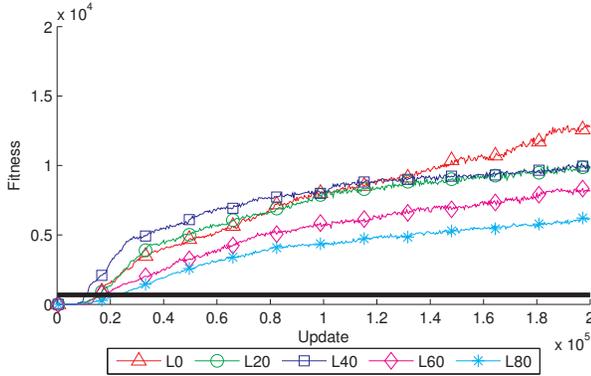


Figure 9: Fitness for a range of message loss rates.

4.5 Node churn

This experiment examined the effect of node churn, that is, node death and replacement, on the evolution of efficient data distribution. As in earlier experiments, the fitness function used here rewarded for data distribution and efficiency, however, here we also probabilistically replace organisms within each deme. When an organism is replaced, it is killed and a new organism, with its genome drawn from the germline, is instantiated in its place. We tested multiple treatments for different values of this replacement probability ranging from 0 to 1, in increments of 0.2. The frequency of checking for node replacement and associated probabilities were selected to provide, on average, 25 replacements per deme at a probability of 1.0. We note that when a node is replaced, the neighboring organisms must re-send the data to that location in the network, and that until they do so, any reward for efficiency is unavailable (as that individual cannot set its opinion without first receiving data).

Figure 10 plots the grand mean fitness of each deme for these treatments across 30 trials. Here, higher probabilities of node churn resulted in lower fitness values, indicating that replacement has a significant impact on the evolution of data distribution, and all pairs of grand mean fitnesses are significantly different from each other ($p < 1 \times 10^{-6}$). We note that in each case, the fitness of all treatments is still gradually increasing at the end of the experiment. This, in combination with the variance in fitness among treatments with a non-zero churn, indicate continued adaptation to this environment. As with the previous experiment, the strategies that evolved to solve this problem are similar to that in Algorithm 1.

Interestingly, when comparing Figure 10 to Figure 9, we see that even modest levels of node churn have a much

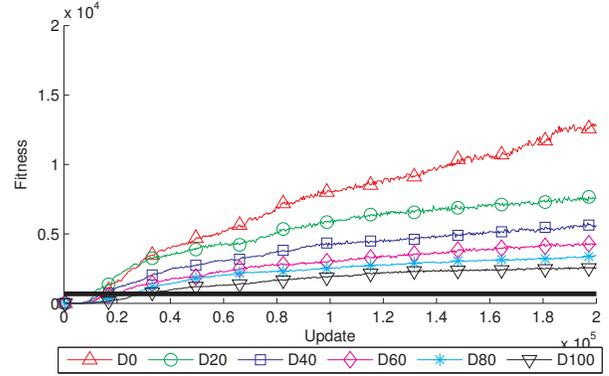


Figure 10: Fitness for a range of probabilities of node churn.

greater detrimental effect on fitness than do increased loss rates. This result suggests that designing systems that are resilient to churn would be more beneficial than reducing loss rate.

4.6 Combined cost, loss, and death

In the final experiment we examined the combined effect of message loss, message cost, and node churn at different levels, while still rewarding for efficiency. The tested configurations included: $\{loss = 0.1x, churn = 0.1x, cost = \max(1, 10x)\}$ for $x \in \{0, 1, 2, 3\}$. For example, we tested $\{loss = 0.1, churn = 0.1, cost = 10\}$.

Figure 11 plots the grand mean fitness of each deme under three different levels of loss, cost, and churn versus the control, each across 30 trials. Here we see that each treatment that includes positive rates of loss, cost, and churn have significantly lower fitness than the $\{0, 0, 0\}$ treatment.

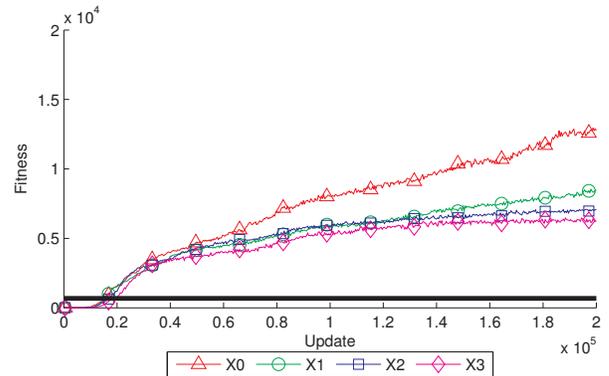


Figure 11: Fitness for a range of values for message loss, message cost, and node churn.

Evolved solutions to this experiment were similar to that described in Algorithm 2. Of all the different experiments conducted here, we note that structures similar to Algorithm 2 have appeared only in experiments containing cost, and that the strategy itself contains two different behaviors, each with a distinct period. Although it is possible that the evolution of this structure is an artifact of AVIDA, an alternative explanation is that agents within communication-based systems, especially those that include cost, operate

most effectively at multiple different frequencies. Moreover, this structure encodes multiple states, where the behavior of an individual in terms of loop frequency and messaging behavior, depends on its location in the environment and whether the individual has received a message.

5. CONCLUSION

The experiments described here demonstrate that populations of digital organisms are capable of evolving to distribute data in a network, and that through the application of different selective pressures, these digital organisms can overcome communication obstacles such as message loss, limited bandwidth, and node failure. Having analyzed the evolution of data distribution under several different environments, we can make a number of qualitative observations. First, the specific network topology used has ramifications not only for the overall performance of the evolved behaviors, but also for the speed with which those behaviors will evolve. Second, employing a cost for sending messages decreases the amount of communication, and results in a more fit behavior. Third, message loss decreases the speed of evolution as well as performance. Fourth, node churn appears to have a much greater impact on performance than does message loss, which suggests that designing systems to be resilient to node churn would be more beneficial than reducing loss rate. Finally, the effect of these environmental hazards appear to be independent of each other.

Future directions for this work include the implementation and deployment of the evolved behaviors in real networks by mapping the genome's instructions into a high-level language, as in [20]. As well, a related problem that could be addressed is the design of optimal communication networks, for example by including a parameter in the genome for the degree of preferential attachment of the underlying network. In addition, it is possible that the results presented here could be used to further our understanding of the evolution of communication in biology, particularly in environments that include communication among multiple agents, as in bacterial biofilms.

6. ACKNOWLEDGMENTS

This work was supported in part by NSF Grants CCF-0750787, CCF-0820220, and CNS-0751155; U.S. Army Grant W911NF-08-1-0495; and by a Quality Fund Grant from Michigan State University.

7. REFERENCES

- [1] M. E. Davey and G. A. O'Toole, "Microbial biofilms: from ecology to molecular genetics," *Microbiology and Molecular Biology Reviews*, vol. 64, no. 4, pp. 847–867, 2000.
- [2] D. Grimaldi and M. S. Engel, *Evolution of the Insects*. Cambridge University Press, 2005.
- [3] B. Holldobler and E. O. Wilson, *The Ants*. Harvard University Press, 1990.
- [4] J. Buck, "Synchronous Rhythmic Flashing of Fireflies, II," *The Quarterly Review of Biology*, vol. 63, no. 3, pp. 265–289, 1988.
- [5] C. M. Waters and B. L. Bassler, "Quorum sensing: Cell-to-cell communication in bacteria," *Annual Review of Cell and Developmental Biology*, vol. 21, no. 1, pp. 319–346, 2005.
- [6] W. Truszkowski, M. Hinchey, J. Rash, and C. Rouff, "NASA's swarm missions: The challenge of building autonomous software," *IT Professional*, vol. 06, no. 5, pp. 47–52, 2004.
- [7] J. R. Koza, M. A. Keane, M. J. Streeter, W. Mydlowec, J. Yu, and G. Lanza, *Genetic Programming IV: Routine Human-Competitive Machine Intelligence*. Genetic Programming, Springer, 1st ed., 2005.
- [8] M. D. Hauser, *The Evolution of Communication*. MIT Press, 1996.
- [9] G. Werner and M. Dyer, "Evolution of communication in artificial organisms," in *Artificial Life II*, pp. 659–687, 1992.
- [10] V. Trianni, T. H. Labella, and M. Dorigo, "Evolution of direct communication for a swarm-bot performing hole avoidance," in *Ant Colony Optimization and Swarm Intelligence*, pp. 131–142, 2004.
- [11] E. Tuci and C. Ampatzis, "Evolution of acoustic communication between two cooperating robots," *Advances in Artificial Life*, vol. 4648, pp. 395–404, 2007.
- [12] C. Ofria and C. Adami, "Evolution of genetic organization in digital organisms," in *Proceedings of DIMACS Workshop on Evolution as Computation*, 1999.
- [13] C. Ofria and C. O. Wilke, "Avida: A software platform for research in computational evolutionary biology," *Journal of Artificial Life*, vol. 10, pp. 191–229, 2004.
- [14] R. E. Lenski, C. Ofria, R. T. Pennock, and C. Adami, "The evolutionary origin of complex features," *Nature*, vol. 423, pp. 139–144, 2003.
- [15] J. Clune, C. Ofria, and R. T. Pennock, "Investigating the emergence of phenotypic plasticity in evolving digital organisms," in *European Conference on Artificial Life (ECAL)*, pp. 74–83, 2007.
- [16] B. Beckmann, P. K. McKinley, D. B. Knoester, and C. Ofria, "Evolution of cooperative information gathering in self-replicating digital organisms," in *Proceedings of Self-Adaptive and Self-Organizing Systems (SASO)*, July 2007.
- [17] D. B. Knoester, P. K. McKinley, and C. Ofria, "Cooperative network construction using digital germlines," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, 2008.
- [18] R. Albert and A. Barabási, "Statistical mechanics of complex networks," *Reviews of Modern Physics*, vol. 74, no. 1, pp. 47–97, 2002.
- [19] D. B. Knoester, P. K. McKinley, B. Beckmann, and C. Ofria, "Directed evolution of communication and cooperation in digital organisms," in *Proceedings of the European Conference on Artificial Life (ECAL)*, 2007.
- [20] B. E. Beckmann, L. M. Grabowski, P. K. McKinley, and C. Ofria, "Autonomic software development methodology based on Darwinian evolution (poster summary)," in *Proceedings of the 5th IEEE International Conference on Autonomic Computing*, 2008.