Proceedings of the 2008 ACM Genetic and Evolutionary Computation Conference (GECCO), Atlanta, Georgia, July 2008.

Selection for Group-Level Efficiency Leads to Self-Regulation of Population Size

Benjamin E. Beckmann, Philip K. McKinley, and Charles Ofria
Department of Computer Science and Engineering
3115 Engineering Building
Michigan State University
East Lansing, Michigan 48824
{beckma24,mckinley,ofria}@cse.msu.edu

ABSTRACT

In general, a population will grow until a limiting factor, such as resource availability, is reached. However, increased task efficiency can also regulate the size of a population during task development. Through the use of digital evolution, we demonstrate that the evolution of a group-level task, requiring a small number of individuals, can cause a population to self-regulate its size, even in the presence of abundant energy. We also show that as little as a 1% transfer of energy from a parent group to its offspring produces significantly better results than no energy transfer. A potential application of this result is the configuration and management of real-world distributed agent-based systems.

Categories and Subject Descriptors

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

General Terms

Experimentation

Keywords

Artificial life, digital evolution, self-regulation, multi-agent systems, selection, cooperative behavior.

1. INTRODUCTION

In 2004, malignant neoplasms, or cancer, caused the deaths of 7.6 million people worldwide, and an estimated 12.3 million new cases were discovered [25]. Cancer is caused by the breakdown of *apoptosis*, the natural processes by which cells die. The failure of this process effectively destabilizes a human body's ability to regulate its population of cells, ultimately causing the untimely deaths of millions of people. Similarly, cancerous problems can arise in agent-based computational systems when the number of agents in the system grows out of control. For example, if the number of detector agents in an artificial immune system is not properly controlled, then the system's ability to detect a threat may degrade due to resource limitations and a corresponding increase in false positives, leading to system-wide quality of service (QoS) degradation or even failure [17]. Furthermore, global limitations on the number of agents within a decentralized system may not be possible due to a lack of knowledge caused by communication, synchronization, and time constraints. Therefore, if agent overpopulation can cause an unacceptable decrease in a system's QoS, then self-regulation of population size is a desirable feature.

Our investigations focus on how the harnessing of *digital evolution* (DE) [1] can contribute to the design or synthesis of robust distributed agent-based systems [20]. In a DE system, individuals, or digital organisms, self-replicate and evolve to perform tasks in a user defined computational environment. Instead of a traditional fitness-based selection process, in DE an organism's ability to self-replicate drives natural selection. This method of selection more closely matches that of the natural world and can provide insight into the evolutionary process [2], often revealing unexpected and strikingly clever solutions [16].

Many similarities can be drawn between the capabilities of a digital organism and an agent in a distributed system. Both are capable of replication, local computation, environmental interactions, and communication with other individuals. In addition, these capabilities can be leveraged and coupled within a group to produce collaborative behaviors, i.e., swarms of agents, enabling the completion of a complex task through the self-organization of individuals.

This work investigates the role that group-level energy efficiency can play in natural selection, in particular, its effects on the self-regulation of a group's population. For example, when a group is selected for replication, what happens if its previous energy gains are ignored completely, partially, or not at all? What effects does group-level efficiency have on the number of individuals required to complete the task and their behavior? Does energy abundance increase or decrease the time required to evolve a group-level task? In addition to providing evidence that helps to answer these questions, we will also discuss the application of the results to the design of agent-based distributed systems.

GECCO'08, July 12–16, 2008, Atlanta, Georgia, USA.

The remainder of this paper is organized as follows. Section 2 provides background on agent-based systems, self-regulating populations, energy based selection, and digital evolution. Section 3 describes Avida, the digital evolution platform used in this study. Section 4 presents our experimental setup and results, followed by conclusions and future work in Section 5.

2. BACKGROUND

As computing becomes more pervasive and decentralized, many design techniques have been proposed to handle the increasing complexity, including autonomic agent-based systems [14]. In these systems, individual agents collaborate to perform a task based on administrative goals, and the system as a whole is self-managing. Therefore, after such a system is configured and initiated, little or no human interaction is required. However, proper initial configuration is essential to ensure the agents do not inhibit the system. For example, in some artificial immune systems, the lifetime of a detector agent is determined *a priori*. If improperly configured, an agent's lifetime can limit the system's ability to function properly, resulting in either false positive or false negative detection of threats [12]. In addition to an agent's lifetime, the number of agents in the system can also affect QoS. Continuing our example, if the number of detectors in an artificial immune system is too small, then threats can go undetected, whereas if they are too numerous, the system can suffer from resource limitations. Autonomically adapting values associated with these management concerns (agent lifetime and number of agents) can directly affect a system's responsiveness, robustness, resiliency, and efficiency [5].

Research on population size regulation in agent-based systems has appeared in the genetic algorithm literature [3,7,8]. However, in these works, the strategy for varying the population size is static and may require global knowledge. For example, in [8], an individual's chances of survival is decreased by a fixed percentage every generation, fixing the maximum life span of an individual. Also, the reproduction rate of the population is determined by its diversity: when the population diversity is low, the reproduction rate is also low. To avoid a reproductive slowdown, many individual mutations are applied simultaneously to a large portion of the population when its diversity drops below a threshold, increasing the populations diversity and reproductive rate.

In addition to evolutionary computation methods, work on population size regulation has also been done in systems-related fields [4, 28]. However, many of these works use predefined rules based on stigmergic communication. The method described in this paper does limit the lifetime of an individual, however this limit is greater than 10 times the maximum ever observed. In addition, the method also limits the maximum population size, however the population rarely reaches this maximum capacity. In contrast, this method does not predetermine the number of individuals required to solve a problem, or allow information to be stored in the environment for us in stigmergic communication. We will show that this method is capable of evolving to solve a group-level problem while self-regulating the population size.

If selection is based solely on how well an individual completes a task, and energy efficiency is ignored, then solutions can evolve to be successful, but their translation into real devices can produce suboptimal results. For example, as shown in [19], when a robot was evolved to search for an object without any energy constraints, it evolved to spiral out from its starting position and ignore possibly useful sensor readings. In contrast, when energy was included in the fitness evaluation, the robot actively polled its sensors and chose a more direct path to the target, increasing its energy efficiency. Energy efficiency has also been used indirectly in the study of the evolutionary process. In [10], a mobile robot was evolved to move for an extended period of time using a rechargeable battery. As long as the robot returned to the recharging sector of the environment before its battery was depleted, it could continue to move, increasing its fitness. Through the evolutionary process the robot evolved to recharge itself and survive until it was stopped due to a hard time limit. These direct and indirect uses of energy demonstrate the theory of diminishing returns. Ideally, a computer system should automatically operate close to this point, optimizing its performance. Commonly, the configuration required to achieve such performance, such as the number of agents within a system, is computed *a priori* as in artificial immune systems [13] and particle swarm optimization [22]. However, research into run-time optimization of resources through self-* properties has increased in recent years [18,23,27] leading to exciting projects including NASA ANTS [26], and Swarmanoid [6]. Our focus here is on the effects energy efficiency can have on evolution, specifically, the effects that energy transferred from one generation to the next generation can have on the evolvablity of group-level tasks.

3. AVIDA BACKGROUND & EXTENSIONS

Avida is a well established artificial life platform used to study evolutionary biology [2, 15, 16, 21, 29]. In Avida, individuals, or digital organisms, compete for space in a two-dimensional grid of cells, shown at the bottom of Figure 1. Each cell can contain at most one organism comprised of a circular list of instructions (its genome) and a virtual CPU that executes those instructions, shown at the top of Figure 1. An organism's virtual CPU is made up of three general purpose registers (AX, BX, CX), two general purpose stacks, and special purpose heads which point to locations within the organism's genome. These heads are used to control instruction execution and flow, plus facilitate replication. The execution of an instruction costs both virtual CPU cycles and energy. Different instructions can be assigned different CPU cycle and energy costs. The competitiveness of an organism within a population depends on its ability to balance these costs while effectively executing instructions to complete user defined *tasks*. A task is a mechanism to reward or punish an organism that has successfully performed a specific function.

Self-replication is achieved when a (parent) organism copies its genome and executes a DIVIDE instruction, effectively creating two offspring organisms, one of which replaces the parent. The two newly produced offspring equally split the energy of the parent organism after a small percentage (5%) has been decayed. In general, the copying of instructions by the parent organism is imperfect, resulting in offspring genomes that are not identical to that of the parent. In the experiments described herein, these copy mutations are turned off. To introduce variation into the population, group-level mutations, described later, are used.

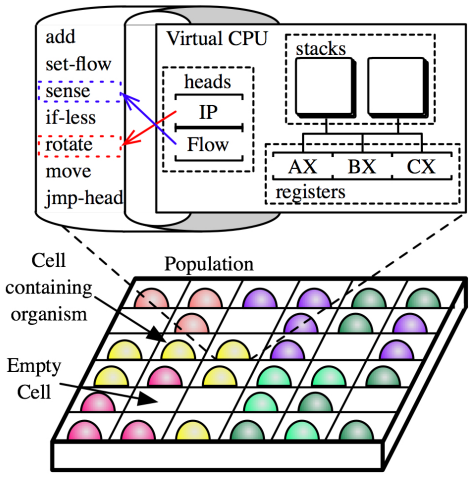


Figure 1: Avida population (bottom) and composition of a digital organism: genome (top left), virtual CPU (top right)

Instead of an explicit fitness function, the competition for space drives selection in Avida; those organisms who replicate faster are more successful than those who do not. An organism's energy is used to calculate its metabolic rate using Equation 1. An organism's metabolic rate is inversely proportional to a user defined limit on the total number of instructions an organism can execute before its energy is depleted, assuming no new energy influx and all instructions cost 1 energy unit. Probabilistically, an organism with a higher metabolic rate will execute more instructions at a higher energy cost per instruction, calculated by Equation 2, than an organism with a lower metabolic rate.

$$metabolic\ rate = \frac{stored\ energy}{instructions\ before\ zero\ energy} \quad (1)$$

$$actual\ energy\ cost = metabolic\ rate \times energy\ cost \quad (2)$$

An Avida population is initialized by injecting it with a single ancestral organism capable only of self-replication. Along with the instruction sequence required to replicate, the ancestral organism also contains 85 no-operation instructions. These instructions have no effect on the ancestral organism's observed behavior, or *phenotype*, excluding its gestation time. They do provide the evolutionary process more room to work, which decreases the probability that a single mutation will disrupt an organism's replication cycle. However, if an organism's replication cycle is broken by a mutation, it will no longer have the ability to replicate, thereby removing its genetic code from the population.

In addition to local computation and self-replication, a digital organism is also capable of inter-organism messaging, movement, and environmental sensing. Messaging functionality is provided by a BROADCAST instruction, which collects the contents of two virtual CPU registers and transmits them in a single message to every organism within a user defined radius. For example, Figure 2 depicts three possible broadcast radii of an organism S. If the organism's broadcast radius is set to 2 then every organism residing in a cell marked with a number less than or equal to 2 will receive a copy of a transmitted message. We note that the results presented in Section 4 use a broadcast radius of 3, however, a broadcast radius of 1 was also tested and produced similar results. In addition to messaging, an organism can also move to a neighboring cell by executing the MOVE instruction. An organism will always move to the cell that it is facing. For example, if the organism S in Figure 2 is facing right and it executes a MOVE it will relocate to the cell marked with a +1 . An organism can also change its facing by executing a ROTATE instruction. Upon birth, an organism initially faces its parent. Besides messaging and movement, an organism can also sense its local environment. The operation of the SENSE instruction will be discussed in Section 4.1.

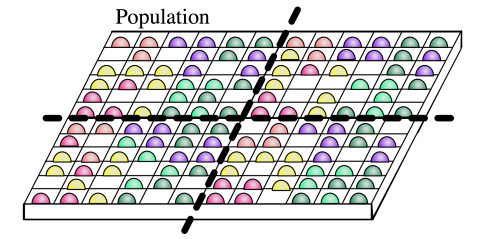
3	3	3	3	3	3	3
3	2	2	2	2	2	3
3	2	1	1	1	2	3
3	2	1	**S**	+1	2	3
3	2	1	1	1	2	3
3	2	2	2	2	2	3
3	3	3	3	3	3	3

Figure 2: Example grid containing an organism S, and the cells reached by broadcasting with varying radii.

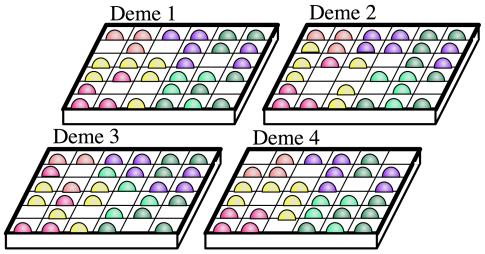
The combination of local computation and environmental interaction enables an organism to explore its environment and cooperate with others to perform a task. To encourage cooperative behavior an organism can be rewarded for completing an individual task that is a building block for a group-level behavior. For example, an organism could be rewarded for alerting its group of an important target when the group is surveying an area. Once a rewarded task is completed, the organism receives an influx of energy and its metabolic rate is recalculated. By efficiently performing individual tasks an organism can increase its metabolic rate, giving it a competitive advantage. In addition, by decomposing a group-level behavior into individual building blocks, the Avida user can encourage the evolution of a complex cooperative behavior.

In addition to promoting selection by rewarding individual tasks, Avida also allows for group-level selection through the use of *demes*. A deme is a independent subgroup within a population. As shown in Figure 3, a single population can be divided into multiple independent demes. The demes are identical in size and topology. When initialized, a deme is seeded with a single organism, and that organism is provided with a baseline amount of energy units.

Avida supports multilevel selection [30], specifically individual and deme-level selection. To enable deme-level selection, a deme is replicated when it satisfies a *deme-level predicate*, more generally thought of as a group-level behavior, such as flocking or consensus. Once a deme has satisfied a deme-level predicate, it is selected for replication and will replace itself and another randomly selected deme. Upon deme replication, prior to creating new offspring demes, mutations are applied to the genome within the parent deme. During this mutation process each instruction in the genome is subject to a 0.75% chance of being mutated to a random



(a) single population



(b) four independent demes

Figure 3: Depiction of a single Avida population without and with demes. Dashed lines in Figure 3(a) represent a division of the population into the demes shown in Figure 3(b).

instruction. The newly created genome and its ancestral genomes make up the *germ line* of an offspring deme. The newly created genome is used in the seed organism for the new demes. In addition to deme-level predicates, a deme's age is also used as a trigger for deme replication. This replication trigger allows for the bootstrapping of the evolutionary process by introducing mutations into a deme's germ line. Figure 4 depicts the initial injection of the ancestral organism into every deme, and both age and predicate based deme replication methods.

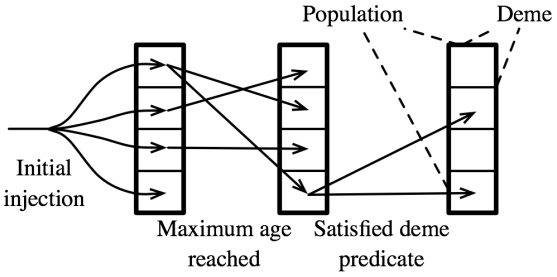


Figure 4: Example showing deme initialization and replication of germ lines

While individual organisms within a deme are able to replicate, those replications do not involve mutations to the genome. Hence, all organisms within a deme are genetically identical. Floreano *et.al.* [9] have previously shown that this approach is effective in evolving cooperative behavior.

4. SELF-REGULATING POPULATION

Distributed agents are commonly used in event detection systems, such as wireless sensor networks and artificial immune systems. Agents can act both independently [11, 13] and cooperatively [24]. For example, in [11] agents independently detect the presence of a forest fire, but collaborate to determine its perimeter and notify local authorities. However, the QoS provided by this type of reconnaissance service, capable of surveying its environment and ascertaining strategic environmental features, is susceptible to agent under- and overpopulation. In general, the number of agents required for reconnaissance depends on the desired outcome. For example, if time is limited, more agents may be used to cover an area than when time is not an issue. However, if resource usage is also important, the number of agents may need to be restricted. Furthermore, some level of cooperation among agents is required to effectively survey an environment and report events.

In this work, we focus on the evolution of a cooperative deme-level reconnaissance task, specifically investigating the effects of a heritable energy trait on the evolution of this behavior in a multi-organism system. We will show, through experimentation, that a small energy transfer from one generation to the next can decrease amount of time required to evolve a gourp-level task and can promote self-regulation of the groups population.

4.1 Experimental Setup

In these experiments, a population is divided into 100 independent demes, each consisting of 49 cells arranged in a 7×7 grid, as shown in Figure 5. Each cell within a deme is marked by an integer denoting the cell's contents: empty (-1), a "nest" (0), or a target (> 0). Each deme contains exactly one nest cell, located in its center, and one randomly located target cell; all other cells are empty. An organism can sense what type of cell it resides in by executing the COLLECT-CELL-DATA instruction, which reads the value stored in the cell into a register in the organism's virtual CPU. In the experiments described here a single deme-level predicate is used. To satisfy this predicate, a message containing the target cell's ID (a random positive integer stored in the target cell) must be received by an organism currently residing in the nest. Minimally, this predicate requires two organisms to cooperate: one to send the message and one to receive it. Upon the satisfaction of this predicate the satisfying deme is replicated, as shown in Figure 4.

-1	-1	-1	-1	-1	-1	-1
-1	>0	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	0	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1

Figure 5: Deme setup with a nest (0), target (> 0), and empty (-1) cells.

To encourage the evolution of the desired behavior, two organism-level tasks are rewarded. The simplest task rewards an organism that enters the target cell, with an energy bonus equal to the baseline energy given to a seed organism (1000 energy units). Incorporating this task into the environment encourages organisms to forage for the target cell.

However, this task does not require the organism to take any action or even have knowledge that it is in the target cell. To encourage active sensing and reporting of the target cell's ID, the second organism-level task rewards an organism for sending the target cell's ID in a message. However, before this task can be rewarded, an organism must gain access to the target cell's ID either by finding the target cell (encouraged by the first task) and collecting its ID, or by receiving it in a message. After the organism has gained access to the target cell's ID, it must send the ID to an organism on the nest in order to receive a reward. Once this final step is completed, the organism will receive a reward of 200 energy units. By performing these tasks an organism can increase its energy and gain a competitive advantage. However, it is conceivable that an organism could evolve to repeatedly complete either or both tasks. To discourage this type of hyperactivity, a limit is placed on the number of times an organism can receive a reward for each task. In addition, higher energy and virtual CPU cycle costs are assigned to all sensing, messaging, and movement instructions, mimicking the costs associated with performing these operations on physical hardware.

4.2 Evolved Foraging Behavior

Our experiments produced demes capable of satisfying the deme-level predicate. Before evaluating the effects of various parameter settings on the evolutionary process, let us first describe a strategy that evolved frequently in our runs. We note that an organism cannot glean information about the location of the target cell from the environment unless it is occupying that cell. Hence, the only way an organism can find the target cell is by performing a random search. However, organisms did evolve to take advantage of the constant location of the "nest" cell and the topology of the environment. Specifically, through the use of the GET-CELL-XY instruction, which places the organism's current (x,y) coordinates in two of its registers, and the IF-EQU register comparison instruction, organisms repeatedly moved back and forth along the deme diagonal. This oscillatory behavior, depicted in Figure 6, enables an organism to move while remaining near and frequently entering the "nest" cell.

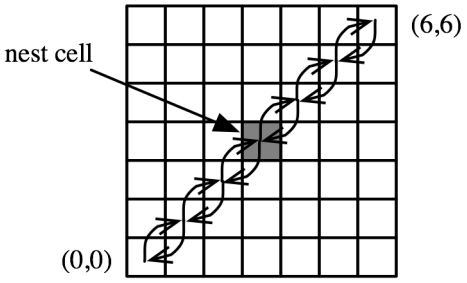


Figure 6: Example path resulting from organism moving back and forth on deme diagonal.

4.3 Varied Energy Transfer

To perform the following experiments, we extended Avida to allow a percentage of a parent deme's energy to be passed to its offspring. The passing of energy allows it to be a heritable feature, thereby enabling selection based indirectly on energy efficiency. By varying the amount of energy passed to the offspring deme, we are able to assess the effects of energy heritability on the evolution of a deme's ability to satisfy the deme-level predicate. We varied the amount of energy passed to the next generation in four different treatments: 0%, 1%, 5%, or 10%. Additional, higher levels of energy transfer were also tested, however, none was significantly different than the results observed in the 10% treatment. To measure the effect of energy transfer on the evolution of the behavior to satisfy a deme-level predicate, we compare each treatment based on the mean gestation time of a deme (time to complete deme-level task), and the mean number of organisms within a deme. We also use organism gestation time to evaluate the effects of energy transfer on the evolution of the deme-level task.

Figure 7 plots the effect, on the mean gestation time of a deme, of varying the percentage of energy transferred from the parent deme to the offspring. The plot shows a significant difference between the 0% treatment and all other treatments after $50,000$ updates. For example, the Wilcoxon rank-sum test calculates a p-value of 0.0025 when an α of 0.001 is used in the comparison of the 0% and 1% treatments. This plot suggests that as little as 1% energy transfer from a parent to an offspring can significantly increase a deme's ability to evolve a deme-level task, when compared to the 0% treatment. This result can be attributed to the fact that an organism injected into a deme in the 0% treatment is given the baseline amount of energy, which eliminates any energy advantage that could have been achieved by the parent deme, effectively slowing (but not stopping) the evolutionary process, as shown by the persistent downward slope in Figure 7. For example, if organisms in a deme increase their energy in the 0% treatment, then the deme will be more likely to be replicated. After replication, however, the energy level of the organisms in the offspring deme is reduced to the baseline, decreasing the deme's probability of replicating again. On the other hand, if energy is transferred to organisms in an offspring deme, the higher organism baseline energy level gives the deme a competitive advantage, albeit a small one.

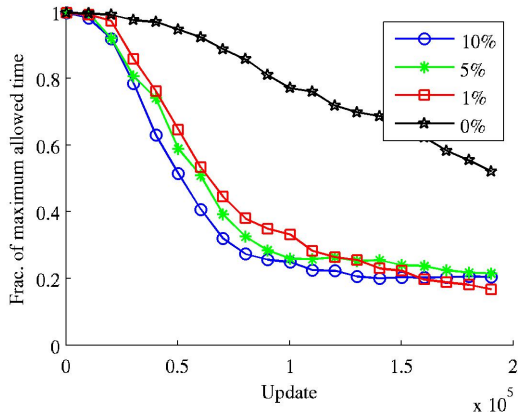


Figure 7: Average fraction of total possible time to complete a deme-level task using multiple energy transfer percentages. Results are mean of 30 runs.

In addition to increasing the evolvablity of a system, a small transfer of energy can also promote the evolution of a self-regulating population during the deme's development.

Figure 8 plots the mean population size of demes in all four treatments. This plot reveals a mean increase in deme population size in the three non-zero treatments during the beginning of a run followed by a continual reduction after about the first quarter, eventually finishing below the 0% treatment. In contrast, the 0% treatment does not exhibit much variation in deme population size.

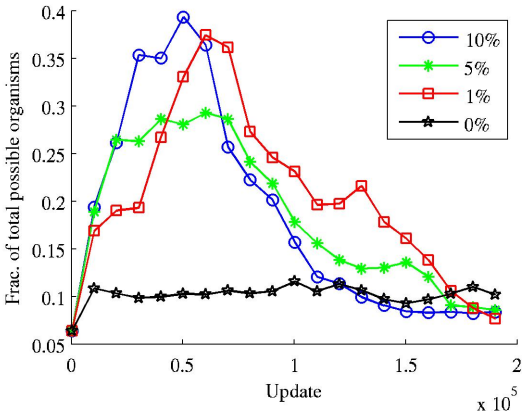


Figure 8: Average fraction of total possible organisms per deme using multiple energy transfer percentages. Results are average of 30 runs.

In addition, organisms in the 0% treatment do not perform individual tasks at the same level as organisms in the 1% treatment, as shown in Figure 9. However, we note a convergence of the task completion statistics toward the end of both treatments, which is a byproduct of the deme replacement method and the decrease in deme gestation time. Specifically, the drop in task completion levels in the 1% treatment are caused by demes that are replaced before they perform a task. The lower levels of individual task completion in the 0% treatment are due to an absence of a selective pressure to complete these tasks and collect additional energy. In addition, since the organisms collect little additional energy, they are not able to increase the population in their deme above the level achievable with the baseline energy. However, even without a fluctuating population, the evolutionary process selects demes in the 0% treatment that satisfy the deme-level predicate, but this process requires more time than when energy is transfered, as seen in Figure 7.

The reduction in deme population size observed in the non-zero treatments in Figure 8 suggests that organisms have evolved in one of three ways. Either the organisms have (1) increased their level of cooperation, enabling them to satisfy the deme-level predicate more quickly, thereby reducing time for deme replication (supported by the decline in average deme gestation time shown in Figure 7), or (2) their replication rate has been slowed such that each organism reproduces less often, giving the group more time to satisfy the predicate before producing offspring, or (3) some combination of both. Figure 10 shows the mean gestation time of an organism for the 0% and 1% treatments. (The other non-zero treatments produced results similar to the 1% treatment and are omitted due to space limitations.) Error bars are omitted from the figure because there is no significant difference between the two treatments. We note that in

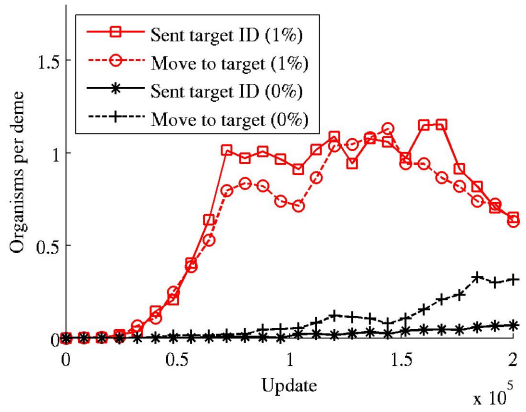


Figure 9: Average number of organisms in current demes who have performed either of the two individual tasks. Results are average of 30 runs.

both the 0% and 1% treatments, the mean organism gestation time increases with time. This phenomenon occurred in all energy transfer levels tested. In contrast, the gestation time of Avida organisms typically decreases over time, as shown during the beginning of both treatments, because of selective pressures at the organism-level to become a more efficient self-replicatior and produce more offspring. This result shows that the pressure to become a more efficient self-replicatior can be overcome by performing selection at the deme-level.

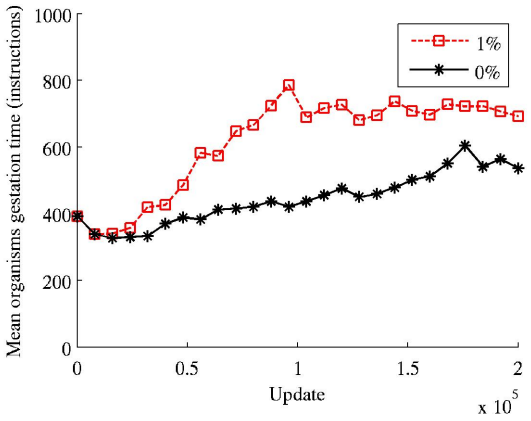


Figure 10: Mean of organism gestation times. Results are the average of 30 runs.

4.4 Abundant Energy

In the previous treatments, the amount of energy an organism could gain during its lifetime was limited by a restriction on the number of times it could receive a reward for completing an individual task. To investigate the effects of abundant energy, we removed this limitation. Repeating the previous treatments with abundant energy, we observed no significant differences in the results. Figure 11 displays the mean deme gestation time and total number of organisms per deme for the 0% and 1% treatments when energy

accumulation is not limited. By inspecting Figure 11, it can be determined that the same pressures that caused the populations in the previous treatments to self-regulate are still present, even when energy is abundant. In addition, energy abundance does not significantly affect the gestation of individual organisms. These results suggest that energy abundance has little to no effect on the evolution of demes that satisfy the deme-level predicate. The minimal impact of energy abundance can be classified as a byproduct of diminishing returns: As an organism completes more tasks and accumulates additive energy rewards, it pays a higher energy cost per instruction because of its increased metabolic rate. Once the organism reaches the point where it costs more energy to perform a task than it receives in return, additional task completion begins to have a negative effect on the organism's metabolic rate. Therefore, the evolutionary process must balance diminishing returns with the selective pressure to accumulate additional energy by increasing an organism's gestation time.

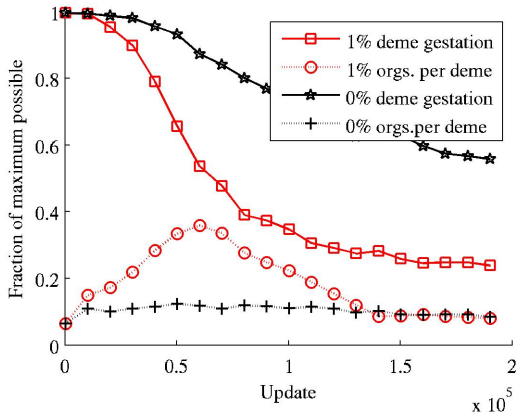


Figure 11: Fraction of total possible organisms per deme and fraction of maximum deme gestation time when energy is abundant and 0% or 1% of the parent deme's energy is transfered to the offspring. Results are representative of 30 runs.

The minimal effect of energy abundance on the evolution of a cooperative reconnaissance task suggests that deme-level selection is robust, at least in this case, to organism-level perturbation. In both the energy abundant and energy limited case, incorporating energy heritability into deme-level selection reduces the time required to evolve cooperative reconnaissance. In addition, the evolutionary process increases the quality of the solution by evolving a self-regulating population.

5. CONCLUSIONS

Through the use of digital evolution and the Avida system, we have shown that a population can evolve to self-regulate its size during group-level task development when as little as 1% of the parent deme's total energy is transferred to the offspring demes. In addition, we provide evidence that an increase in organism gestation time occurs when demes evolve to be more proficient at satisfying the deme-level predicate. In particular, an increase in the gestation time of organisms allows a deme more time to satisfy the deme-level predicate with fewer total organisms, which translates into a more energy-efficient deme. Furthermore, we have shown that abundant resources have little effect on the evolution of this deme-level behavior.

In these experiments, the evolutionary process is balancing opposing selective pressures to decrease an organism's gestation time with the pressure to decrease a deme's gestation time. These two pressures are opposing because decreasing the gestation time of an organism will increase the number of births per deme, thereby increasing the amount of energy lost due to energy decay during replication. Whereas, a decrease in deme gestation time implies that fewer instructions are executed by its constituents, which translates into an energy savings. Since the deme-level predicate used in these experiments requires cooperation, evolution favors extending an organism's gestation time to allow more time to search for the target before replication occurs. These factors promote the natural selection of demes that satisfy the deme-level predicate while selecting against inefficient organisms, effectively encouraging deme-level efficiency.

In an agent-based distributed system, both individual lifetime and population size are important concerns for developers. Mismanagement of either of these two concerns can cause a disruption of a system's required QoS. Through the transfer of energy and deme-level selection, digital evolution has produced a system that can effectively self-manage both of these concerns in addition to completing a desired task in an efficient manner. We intend to extend this work into the design of real-world systems capable of self-regulating their populations and adapting the lifetime of their agents to fulfill the requirements of a particular domain. Specifically, we intend to apply these results to the design of an agent-based event detection system for use in a wireless sensor network. By drawing inspiration from natural systems and harnessing the evolutionary process which produced those systems, we hope to provide tools capable of handling the escalating complexity of future distributed computing systems.

6. REFERENCES

[1] C. Adami. *Introduction to artificial life*. Springer-Verlag New York, Inc., New York, NY, USA, 1998.

[2] C. Adami, C. A. Ofria, and T. C. Collier. Evolution of biological complexity. *Proceedings of the National Academy of Sciences*, 97(9):4463–4468, April 2000.

[3] J. Arabas, Z. Michalewicz, and J. J. Mulawka. Gavaps - a genetic algorithm with varying population size. In *International Conference on Evolutionary Computation*, pages 73–78, 1994.

[4] M. Bakhouya and J. Gaber. Adaptive approach for the regulation of a mobile agent population in a distributed network. In *Proceedings of the Proceedings of The Fifth International Symposium on Parallel and Distributed Computing*, pages 360–366, Washington, DC, USA, 2006. IEEE Computer Society.

[5] A. Bieszczad, T. White, and B. Pagurek. Mobile agents for network management. *IEEE Communications Surveys*, 1998.

[6] M. Dorigo. Swarmanoid project. http://www.swarmanoid.org, January 2008.

[7] H. Eskandari, C. D. Geiger, and G. B. Lamont. FastPGA: A dynamic population sizing approach for solving expensive multiobjective optimization problems. In *Proceedings of the 4th International Conference on Evolutionary Multi-Criterion Optimization*, volume 4403 of *Lecture Notes in Computer Science*, pages 141–155. Springer, 2006.

[8] C. Fernandes and A. C. Rosa. Self-regulated population size in evolutionary algorithms. In *PPSN*, volume 4193 of *Lecture Notes in Computer Science*, pages 920–929. Springer, 2006.

[9] D. Floreano, S. Mitri, S. Magnenat, and L. Keller. Evolutionary conditions for the emergence of communication in robots. *Current Biology*, 17:514–519, March 2007.

[10] D. Floreano and F. Mondada. Evolution of Homing Navigation in a Real Mobile Robot. *IEEE Transactions on Systems, Man and Cybernetics Part B : Cybernetics*, 26(3):396–407, 1996.

[11] C.-L. Fok, G.-C. Roman, and C. Lu. Rapid development and flexible deployment of adaptive wireless sensor network applications. In *Proceedings of the 25th IEEE International Conference on Distributed Computing Systems*, pages 653–662, Washington, DC, USA, 2005. IEEE Computer Society.

[12] M. Glickman, J. Balthrop, and S. Forrest. A machine learning evaluation of an artificial immune system. *Evolutionary Computation*, 13(2):179–212, 2005.

[13] S. A. Hofmeyr and S. A. Forrest. Architecture for an artificial immune system. *Evolutionary Computation*, 8(4):443–473, 2000.

[14] J. O. Kephart and D. M. Chess. The vision of autonomic computing. *Computer*, 36(1):41–50, 2003.

[15] R. E. Lenki, C. A. Ofria, T. C. Collier, and C. Adami. Genome complexity, robustness and genetic interactions in digital organisms. *Nature*, 400:661–664, 1999.

[16] R. E. Lenski, C. Ofria, R. T. Pennock, and C. Adami. The evolutionary origin of complex features. *Nature*, 423:139–144, 2003.

[17] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker. Search and replication in unstructured peer-to-peer networks. In *16th ACM International Conference on Supercomputing*, New York, USA, June 2002.

[18] G. Mainland, D. C. Parkes, and M. Welsh. Decentralized, adaptive resource allocation for sensor networks. In *Proceedings of the 2nd USENIX Symposium on Networked Systems Design and Implementation*, Boston, MA, USA, May 2005.

[19] G. McHale and P. Husbands. Incorporating energy expenditure into evolutionary robotics fitness measures. In *Proceedings of the Tenth International Conference on the Simulation and Synthesis of Living Systems*, pages 206 – 212, Cambridge, MA, USA, 2006. MIT Press.

[20] P. McKinley, B. Cheng, C. Ofria, D. Knoester, B. Beckmann, and H. Goldsby. Harnessing digital evolution. *Computer*, 41(1):54–63, January 2008.

[21] C. Ofria and C. O. Wilke. Avida: A software platform for research in computational evolutionary biology. *Artificial Life*, 10:191–229, March 2004.

[22] J. Pugh and A. Martinoli. Multi-robot learning with particle swarm optimization. In *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pages 441–448, New York, NY, USA, 2006. ACM.

[23] S. M. Sadjadi and P. K. McKinley. Transparent self-optimization in existing corba applications. In *Proceedings of the First International Conference on Autonomic Computing*, pages 88–95, Washington, DC, USA, 2004. IEEE Computer Society.

[24] G. Simon, M. Maroti, A. Ledeczi, G. Balogh, B. Kusy, A. Nadas, G. Pap, J. Sallai, and K. Frampton. Sensor network-based countersniper system. In *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems*, pages 1–12, New York, NY, USA, 2004. ACM Press.

[25] A. C. Society. Global cancer facts & figures, 2007.

[26] W. Truszkowski, M. Hinchey, J. Rash, and C. Rouff. Nasa's swarm missions: The challenge of building autonomous software. *IT Professional*, 6(5):47–52, 2004.

[27] E. Tuci, R. Gross, V. Trianni, F. Mondada, M. Bonani, and M. Dorigo. Cooperation through self-assembly in multi-robot systems. *ACM Transactions on Autonomous and Adaptive Systems*, 1(2):115–150, 2006.

[28] T. White, B. Pagurek, and D. Deugo. Management of mobile agent systems using social insect metaphors. In *21st IEEE Symposium on Reliable Distributed Systems*, pages 410–415, 2002.

[29] C. O. Wilke, J. Wang, C. A. Ofria, C. Adami, and R. E. Lenki. Evolution of digital organisms at high mutation rate leads to survival of the flattest. *Nature*, 412:331–333, 2001.

[30] D. S. Wilson. Introduction: Multilevel selection theory comes of age. *The American Naturalist*, 150(S1-S4), July 1997.