Available online at www.sciencedirect.com
ScienceDirect
Artificial Intelligence 172 (2008) 884–898
Artificial Intelligence
www.elsevier.com/locate/artint

Robust artificial life via artificial programmed death

M.M. Olsen [a,*], N. Siegelmann-Danieli [b], H.T. Siegelmann [a]

[a] *University of Massachusetts Amherst, Department of Computer Science, 140 Governor's Drive, Amherst, MA 01003, USA*
[b] *Maccabi Health Organization, 27 Hamered Street, Tel-Aviv 68125, Israel*

Received 10 August 2007; received in revised form 3 October 2007; accepted 8 October 2007

Available online 21 December 2007

Abstract

We propose a novel approach to self-regenerating continuously-operating systems. Such systems provide best-case solutions in security surveillance or decision making centers. We introduce HADES, a self-regenerating system whose agents acknowledge their "citizenship" or faithfulness to the good of the system and are able to monitor their environment. When agents of HADES find irregularity in themselves they first try to repair, and will self-kill if repair fails. When an agent senses that there are persistent malfunctioning agents in its environment, it sends messages to entice them to self-kill. The neighbors then proceed to generate new healthy agents to replace the killed agent. We experiment with HADES on various impairments including the most difficult one of excessive regeneration of irregular aggressive agents. These agents may use all of the system's resources and thus take over the system, reminiscent of biologically grown tumors. We study how irregular growth may occur and then develop protocols of killing these agents to optimize the system's longevity. While some of the inspiration is from the immune system and tumor therapy, we contribute to the field of AI by introducing protocols for system robustness via the notion of active citizenship and the fundamental property of programmed death.

Keywords: Multi-agent system; Artificial life; Adaptive system; Regenerative system

1. Introduction

This paper describes a biologically inspired mechanism for multi-agent systems that improves robustness by enabling agents to combat anonymous malfunctioning agents. This mechanism makes it possible for the system to self-regenerate as needed, to repair agents when they are not functioning correctly, and to influence misbehaving agents to kill themselves. The remaining functioning agents can then regenerate new agents to replace the removed ones. Since these operations constitute underlying maintenance, they do not require many resources and do not disturb the continuous functionality of the system. One aspect of this mechanism is a set of life protocols that are internal to every agent and include self-checking, self-repairing, self-death, and rules governing their use. The complementing aspect of the mechanism is a communication protocol among agents that influences surrounding malfunctioning agents to "kill" themselves. This communication mechanism is described in terms of a system we call HADES, "Healing and

* Corresponding author.
E-mail addresses: molsen@cs.umass.edu (M.M. Olsen), danieli_na@mac.orgil (N. Siegelmann-Danieli), hava@cs.umass.edu (H.T. Siegelmann).

0004-3702/$ – see front matter
doi:10.1016/j.artint.2007.10.015

Agent Death Enabling Stability". This paper focuses on the fundamental design of both aspects of this mechanism, and calls for future applications to utilize the communication aspect of the paradigm.

Self-regeneration enables agents of the system to create new agents. In biology, this is used to both recover from dying cells as well as develop and expand the system [16]. In [5] agents replicate to create agent clusters that will make joint decisions, thus improving fault tolerance. [27] describes a simple robot composed of building blocks that can generate a new copy of itself by assembling additional blocks. The use of self-regeneration for development is described in [17]. In this system artificial chemicals are used to control movement of cells on a lattice, and grouped with the regenerative abilities the system is able to retain a specific shape despite dying cells. The ability of the system to obtain a specified shape despite dying cells is referred to in that work as self-repair. This use of the term self-repair is to be differentiated from the more fundamental definition, as is used in biology and will be used in our system: the cellular level repair. In cellular level repair, a self-replicating cell or agent uses error correction to make changes to itself to ensure that it grows into the proper form [8]. The system self-repair is but one consequence of the cellular repair.

Another work describes how self-regeneration replaces dying cells in a more general 3D lattice using artificial presence chemicals [7]. When a cell dies, the lack of chemical in its location will cause the neighbors to regenerate to solve the problem of cell death. In HADES the use of presence chemicals is advanced to a more efficient form by allowing the presence chemicals to linger rather than removing them with each step. This enables our agents to also recognize the center of the shape and thus keep a coherent shape without using the previously required ordering [7]. HADES will also include a deeper sense of self-death, mutations, and communication protocols.

HADES is built on the principle of self-regenerating agents where all agents follow the same basic life protocols which code for the individual activity of the agents and provide generalization and improvement over previous self-regenerative agent systems. In addition, HADES diagnoses and repairs via a multi-step protocol. The first step is self-monitoring such that an agent determines whether its own life protocol has been damaged. The second step is for the agent to repair any discovered damage within itself. It is possible that the agent will be unable to repair itself and will thus apply self-death to retain the system's health. This third step of self-death is crucial, as can be seen in biological systems. For example, cancer cells may develop due to the malfunctioning of a cell's self-death mechanism. The fourth step comprises a main concern of this paper: the ability of agents to note that at least one of their neighbors is irregular and thus send surrounding agents a message as a warning. These messages cause the receiving agents to either elevate their own level of alertness or if enough messages are received they entice the receiving agents to activate their programmed death. As will be described in the paper, all agents maintain some level of citizenship and communicate their upcoming death via signaling to entice neighboring agents to die as well. HADES studies the ability to induce self-death on an agent, allowing the system to recommend death without agents killing one another. It utilizes agent regeneration, repair, and death, as well as novel communication protocols to overcome system faults and maintain continuous operation. All the activity of HADES is cheap and will only occur as necessary, and thus can serve as an underlying maintenance mechanism to improve the robustness of any multi-agent system.

This paper is organized into the following sections: Section 2 describes more related work in diagnosis and repairing systems, Section 3 introduces HADES as a multi-agent system where agents follow goals and life protocols, Section 4 describes how an agent may become irregular and how its continued operation may negatively affect the entire system and includes the communication protocol that enables the recognition and removal of such irregular agents by their neighbors, and Section 5 describes simulation details used for the experiments and results as described in Section 6. We close the paper with conclusions.

2. Related work

For a multi-agent system to function continuously it must adapt on-line to changes in the environment and internal failures. Diagnosis of a problem is a key requirement, as is providing a plan to react to the problem [9]. Various frameworks exist for diagnosis in multi-agent systems, including domain independent diagnosis where an agent should also be able to determine a new plan if its expectations are not met [10]. Diagnosis for pre- and post-failure analysis for causal tasks can allow the system to both prevent a failure and recover from it [25]. It is argued that post-failure protocols are less domain dependent and are more crucial for the design of robust systems [25].

One way to obtain post-failure robustness is via *self-repairing* mechanisms. Repair typically follows one of two categories, "Attributive" or "Functional". Attributive repair restores attributes to their state prior to the failure to revert

any damage. Functional repair does not backtrack but instead optimally uses the remaining resources to obtain the best possible functionality [3]. As an example of Attributive repair, software components can self-monitor to determine vulnerabilities and thus remove them [22]. The analysis to determine vulnerabilities uses both static and dynamic techniques, including the Stackguard tool and predicate abstraction. Software validation techniques can be used to identify causes of vulnerabilities, enabling their removal [22]. Wireless sensor networks are also being designed with self-healing capabilities inspired by immunology to detect sensor faults and respond via a form of Functional repair. For example, [2] mimics B-cells in the immune system with scripts on monitor nodes that follow the status of the sensor nodes in the system. The monitor nodes can find failure by examining the statistical properties of the sensor readings. This system can adapt to the changes in the network caused by sensor failure via monitor nodes notifying sensors of incorrect readings, allowing them to request retraining. This combination of different node types interacting enables the system to find and react to failures [2].

Self-regeneration provides another paradigm for attaining robustness, and has been investigated for at least the last 50 years [26]. It works in a functional framework, mainly to achieve a larger system during development or after agent death. It is one of the main responses to agent death utilized in other systems, as seen in Section 1.

The paper by Roth et al. introduces a system that self-organizes to grow from a stem cell into an intelligent behaving organism [21]. The system is comprised of cells that replicate based on chemical signaling received from the surrounding environment. Since the replication control is sensitive to the environment it will cause the system to grow from a single stem cell to the desired size. This algorithmic control will also cause the system to regenerate after cells are (artificially) removed. Replicating upon need is considered a system-level repair mechanism and is different from repair mechanisms of the cell itself, which we include in HADES. The focus in [21] is on the growth and organization of the system as opposed to the later fault processing that is the main contribution of HADES.

The death of an agent may be harmful in a multi-agent system [13]. Various fault tolerant algorithms exist to react to agent death following either the *survivalist* or *citizen* concepts [4,12,23]. Both approaches are aimed at increasing the adaptability of the system and at minimizing loss of its overall functionality due to agent death. The citizen approach utilizes an external system that is alerted when an agent dies and then reallocates tasks so that the overall system continues to function correctly [12]. The survivalist approach requires each agent to be capable of dealing with all problems as an individual following a prepared set of actions for each specific problem [4,12,23]. The survivalist concept is utilized in the CNet protocol [23].

Agent death is not only a problem to the system, but can be a desired property when the system has to decrease in size or when agents are destroyed and act irregularly. When death is desired, self-destruction is preferred over agents killing other agents, providing more robust behavior [12]. [24] describes a self-managing system proposed by NASA that uses self-destruction as a last resort to deal with damage. This system first tries to self-repair, but if the repair fails a self-death mechanism is deployed to remove the broken agent. This self-death is implemented by a constant *stay alive* signal, such that if an agent no longer receives the signal it will self-destruct. In [14], regeneration, repair, and death are combined to create an artificial organism as the first step toward hardware with the ability to remove surrounding agents that may be faulty. One form of repair involves disabling all cells in the column of the faulty cell after transferring their functions to the cells in the column to their right, essentially using death to repair the organism. The other form of repair is internal, used to combat failure of the artificial molecules that control cellular actions. This repair is accomplished by removing the faulty molecule and then rearranging the remaining ones until a spare is reached, ending with the same number of molecules as was used before the failure [14].

HADES entices death as well but via the use of communication protocols, thus not requiring agents to know exactly which other agents are faulty. In [24] self-death is a default that occurs if an override is not received. Since irregularity is not the norm in most systems, this technique will continuously flood the system with messages to every agent. However in our system messages are utilized in the opposite way by being sent when irregularity has occurred and only to agents in that area, thus decreasing the overall messages that must be sent. We improve on the approach in [14] as well by giving more flexibility to agent movement. We do not require spare agents to be kept to the side until needed, but create them as necessary. HADES utilizes similar mechanisms overall, such as internal repair, regeneration, and death, but allows the agents to decide when to die themselves. We are thus able to maintain our system by suggesting death to surrounding agents and over time removing all faulty ones. Overall, HADES uses a unique combination of death, repair, regeneration, movement, and communication to retain itself by giving agents the utmost control over these actions.

3. HADES—the system

HADES is a three-dimensional lattice of agents whose protocols are biologically inspired and whose agents act locally to achieve citizenship goals. The system maintains the equilibrium goal by agents replicating if there is open space. Agents aim to maintain self-health by testing for failures that are represented by mutations to life protocols and either repairing them or inducing self-death if they are not repaired. Each agent also has a goal to retain basic distance from surrounding agents. This distance is maintained by not moving or replicating into a location closer to its neighbors than the specified distance. The system also aims to maintain itself as one cohesive unit by agents moving or replicating toward the highest density of surrounding agents, to ensure that agents do not become isolated. Finally, the agents also initiate and transfer signals among themselves to facilitate the goal of maintaining the overall system health by decreasing the number of irregular agents. The system could also be modeled in a two-dimensional space with similar mechanisms. We chose to work in three-dimensional space as it is slightly more complex and will therefore relate to more systems.

To achieve these goals each agent follows a series of life protocols that focus on the internal state of the agent. These protocols are replication, self-testing, repair, suppression, self-death, space maintenance, and movement. The life protocols are highly inter-regulated, as will be seen below. Other agent protocols consider the environment and communication with neighbors, and will be described in Section 4.1. The life protocols are inspired by cellular biology, but are applicable for self-regenerating multi-agent systems. We next describe how they are used to achieve an agent's goals.

3.1. Maintaining equilibrium via replication

All agents share the goal of keeping the equilibrium of the system. This goal is accomplished by replicating. Every agent is capable of replicating via a probability that controls how frequently an agent attempts to replicate, to ensure that it neither replicates too quickly nor too slowly. The rate of replication is a system parameter that can be changed and is shown in the Results section to be crucial. Mutation to a single life protocol may occur with small probability during replication, which will alter the agent's functioning (see Section 4).

The replication rate is biologically inspired. In breast tissue, for example, cells have an inherent probability of replication that is 0.0025 [11]. In addition, we are inspired by biology for stopping replication after a certain number of generations. For example, in breast tissue the cell can replicate up to 70 generations before being damaged, although in other tissues this number is different [1].

3.2. Maintaining self-health via repair

Agents maintain their own health by monitoring any damage (or "mutation") that occurs to their protocols. When a mutation is detected internally, the agent attempts to repair the damaged protocol. If the repair mechanism continuously fails, the agent recognizes that it may not be functioning correctly and will attempt to kill itself so that it does not damage the system. The agent is therefore preserving the system health by preserving its own health. In the case where the death protocol is damaged the use of external communication will be required.

3.3. Maintaining healthy replication rate

Since replication is a fundamental operation and its failure can cause significant damage, there is a specialized mechanism called the "suppression" protocol for controlling the effect of mutating the replication protocol. If an agent attempts to replicate more frequently than the replication probability allows, the suppression mechanism will halt the replication. This way an agent can only over-replicate if both the replication protocol and the suppression protocol are mutated, thus utilizing double mechanisms for robustness.

3.4. Maintaining space

Each agent requires a buffer around itself (Fig. 1). No healthy agent will move to a coordinate if another agent exists on an adjacent location. Also, a daughter agent will only be created if there is an empty location with no agents

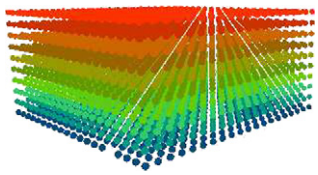


Fig. 1. 3D view of the system at its stable state. Shading is used to make the rows more visible.

adjacent to it. If an agent becomes adjacent to another agent despite these efforts, that agent will attempt to move away if it can do so without encroaching on another agent's space.

3.5. Maintaining cohesiveness via chemical presence signals

An agent maintains the cohesiveness of the system by maintaining the shortest possible distance between itself and the center of the system. This organization is accomplished via presence signals that all agents passively emit. The areas closest to the agent will therefore have a stronger chemical presence. Presence signals are continuously diffusing in all directions for a specific radius and time period. These chemicals can thus be used to determine the proximity between agents, and the direction of the center of mass. For details, see Section 5.

4. Irregular agents and the communication protocol

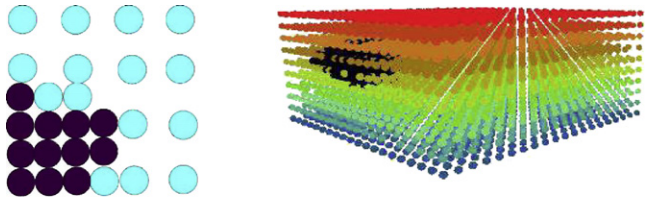
In the previous section we described the life protocols of an agent. In this section we introduce the second type of protocol, the rescue protocol. This protocol exists primarily to counteract agents that have obtained mutations. Although every individual mutation can be problematic, the worst case is when all life protocols are mutated and the agent is therefore *irregular*.

Definition 1. An agent is *irregular* when none of its protocols are functioning correctly. An irregular agent will

(1) Continuously attempt to replicate
(2) Refuse to repair itself
(3) Not induce self-death when repair fails
(4) Ignore space regulations.

An irregular agent will continue to replicate since it can neither repair nor kill itself, thus spreading its damaged life protocol to its daughters. This behavior will quickly create a cluster of irregular agents. The probability of creating an irregular agent from a healthy agent is incredibly low (based on the replication and mutation probabilities) since to prevent the agent from repairing mutations the life protocols must be ruined in a particular order: repair damaged first, then death, then suppression, and last replication. However, once an irregular agent has been introduced into the system it will over-replicate, paying no respect to available space or diffused presence signals and will quickly take over the system as seen in Fig. 2. An irregular agent in essence "pushes" a healthy agent into an adjoining lattice location if it is in its way, causing it to lose its desired space, which may hinder its ability to replicate. Therefore, as the random death slowly occurs the healthy agent count will decrease since they cannot replenish the numbers via replication.

Since irregular agents form a cluster that pushes agents closer to their neighbors as it expands, they will continue to exert physical pressure on the same area of the system. We propose taking advantage of this style of growth to find a solution that inhibits the irregular agents and supports the healthy agents. The environment and signaling algorithms will therefore enable the agents to become active citizens through maintaining their own health, monitoring neighboring agent health, and maintaining the system health. This rescue protocol sends signals that encourage neighboring agents to die. The goal is to determine how these signals should be sent such that healthy agents will remain while irregular agents are removed.



(a) Diagram (b) System

Fig. 2. Irregular agents (darkly colored) take over the system quickly by pushing the healthy agents to the side as shown in the diagram and the system screen shot. The system picture shows the beginning of an irregular agent cluster.

4.1. Communication

We propose a *rescue protocol* aimed at saving the system from the over-replication of irregular agents. Agents that suspect irregularity will alert surrounding agents by sending a signal. The receiving agents interpret this signal as a statement of irregularity in the area. If an agent receives this announcement multiple times it will conclude that it may be the problem and should activate its programmed death. Agents do not know the sender of the signals they receive, and when they send a signal it is to all agents in the surrounding area. This communication style is inspired by biology [6].

The first communication protocol within the rescue protocol is called PLEASE DIE, and is initiated by healthy agents that sense irregularity when their space is invaded by a neighbor's "push". This type of invasion occurs when a neighboring agent moves to the space an agent is occupying, causing that agent to be moved to surrounding buffer space. Due to healthy agents maintaining space the invading agent will either be an unhealthy agent or a healthy agent that has been pushed and therefore forced to move. If a pushed agent does not have empty space around it, it will push a neighboring agent in the same direction. This chain effect will continue to occur until an agent is pushed that has an open space surrounding it.

It is beneficial for an agent to only die after a specified number of signals (>1) are received so that no one agent can directly kill another one. For this purpose a threshold is used:

Definition 2. A Threshold of Signals (TOS) for signal type S is a local variable such that:

1. an agent may only activate self-death from receiving signals once the sum of all received strengths of type S reaches the TOS for type S;
2. the TOS for type S is greater than the max strength of a signal of type S.

4.2. Double messaging

Naively one may think that the PLEASE DIE signal suffices for restoring the system. However, this assumption is not the case as seen in Fig. 3(a). We thus propose a novel double message system where in addition to the PLEASE DIE signal generated by a pushed healthy agent, a secondary signal is applied. This second signal called I DIED is sent by an agent when it is dying from receiving a signal. The I DIED signal is used to alert neighboring agents that they may want to consider dying as well. This method takes advantage of the cluster structure of irregular agents: neighboring agents may be descendants or ancestors of the dying irregular agent, and are therefore likely to also be irregular.

For the rescue protocol to remove all irregular agents and retain a large percentage of healthy agents the I DIED signal must be sent by both healthy and irregular agents that die. In Fig. 3 we see that when no agents sent I DIED signals (Fig. 3(a)) the system collapsed with no healthy agents alive. When the irregular agents could not send I DIED signals (Fig. 3(c)) and only the healthy agents sent them, all healthy agents were killed as well. When dying healthy agents do not send an I DIED signal (Fig. 3(b)) the irregular agents are initially decreased but then increase while the healthy agents eventually decrease. Therefore, it is necessary to have the PLEASE DIE signal and all agents must be able to send the I DIED signal.

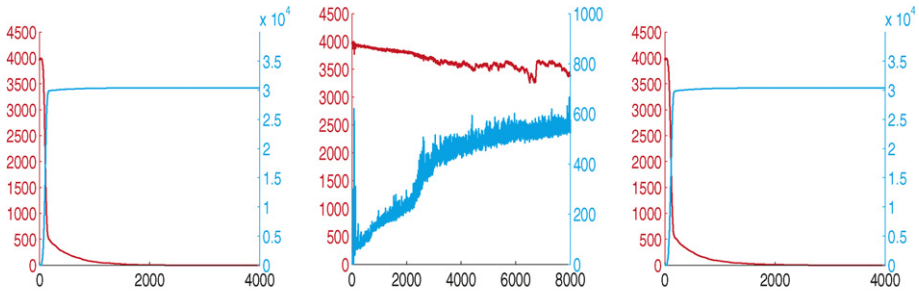


Fig. 3. Different combinations of signals will give different results. The left y-axis corresponds to the healthy agents, and the right y-axis refers to the irregular agents. The x-axis is time. Results show that both the PLEASE DIE and the I DIED signal are necessary.

When an agent receives a signal it will always correctly identify the signal type. The two types of signals exist together, but are not combined and do not affect each other. They each have their own TOS value, as well as their own parameters.

4.3. Signal parameters

Rescue signals are sent in the system for a specific radius in all directions. The strength decreases slightly with each additional distance the signal travels before reaching the radius, as described in Section 5.

Definition 3. The *radius* of a signal defines the maximum distance over which the signal is received.

Definition 4. The *strength* of a signal represents the value that is associated with it, and is decreased as it travels further from the source.

A low radius and strength coupled with a high TOS will result in many signals needing to be received before an agent will decide to die. However, a high radius and strength coupled with a low TOS value will have the opposite effect. Radius and strength can be different for each type of signal. Precise definitions of high and low for these parameters are determined in Section 6 for particular scenarios.

We assume that all agents will die when the total of signals received reaches the appropriate TOS value. Given the mutation probability, very few irregular agents would also mutate the rescue protocol if it was an option. Also, such a mutation is only harmful if it occurs in an irregular agent, since deaths of healthy agents are considered unfavorable.

5. Simulation details

For the following simulations we ran HADES as a cube bounded to size $40 \times 40 \times 20$ units. A total of 4000 healthy agents can reside in this cube at any given time since we chose the inherent distance between agents to be 2. Thus, agents only reside on every other coordinate and consider the surrounding empty spaces as buffers. However, since irregular agents do not adhere to these space constraints, they can total 32,000.

The simulation progresses by time ticks, and although agent decisions are implemented sequentially, the actual actions occur in parallel. There are two different sets of actions that can be performed: at each time step one action from the first set may be chosen, and multiple actions from the second step may be chosen (see Fig. 4). The first set of actions, only one of which may be performed at a time is:

(1) Repair: occurs if the agent is damaged. With every repair attempt there is a 0.5 probability of failure.
(2) Death: occurs by three mechanisms. Death can occur when an agent has been unable to repair its life protocols. It can also occur with a probability of 0.0024 to include other causes of death such as age. The third mechanism is via death signals sent by surrounding agents.

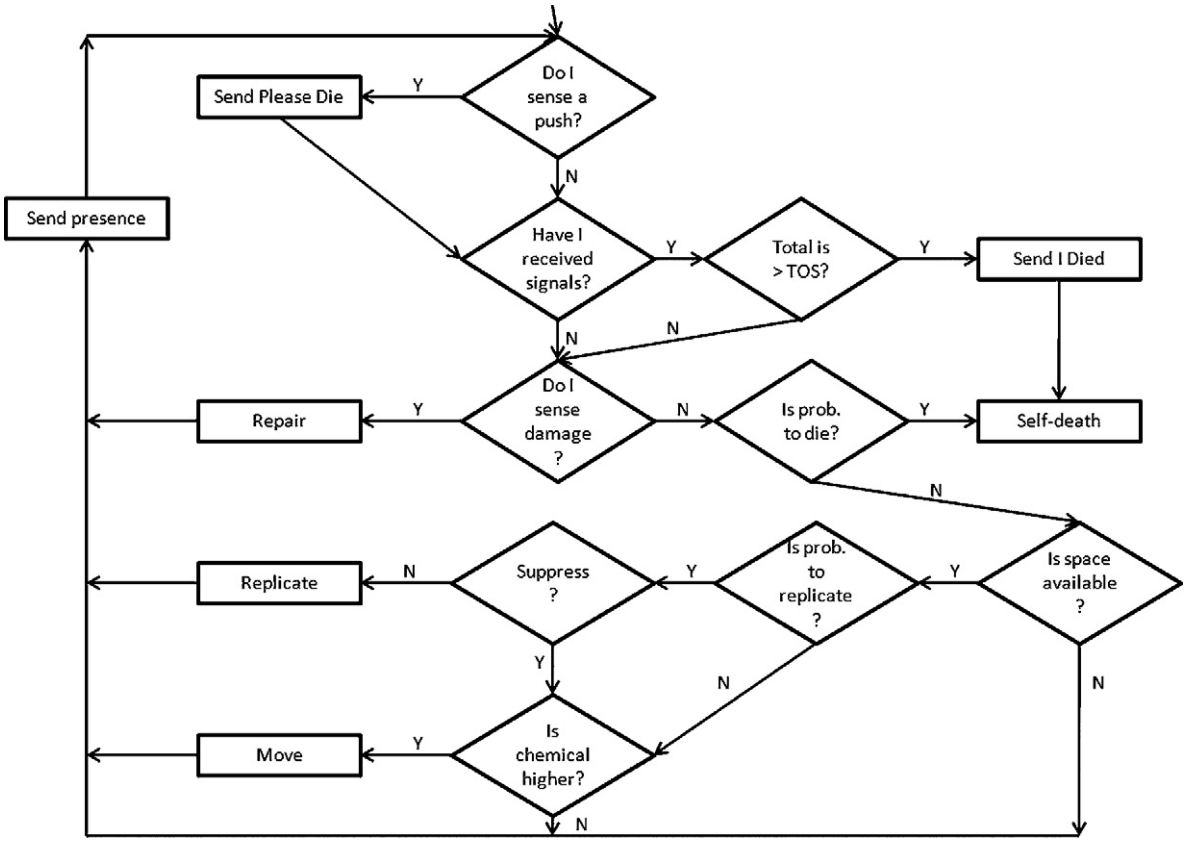


Fig. 4. The basic action controls for a healthy agent at each time step.

(3) Replication: occurs if there is available space and suppression is not activated. The preferred probability of replication is tested in Section 6. With each replication there is a 0.002 probability of mutation. It is randomly decided with equal probability which protocol to mutate among all protocols that are not currently mutated.
(4) Movement: occurs if an agent cannot replicate but there is an available adjacent space with a higher concentration of presence signals than its current spot, representing a space closer to the center of mass.

In addition to one of the above operations, at each time tick an agent is also able to perform the following actions in parallel:

(1) Emit presence signals: Presence signals will diffuse as in Algorithm 1, with the concentration of the chemical slowly increasing at each time step by $\frac{\epsilon}{MAX\_RADIUS^{\rho}}$ for each location within the distance. We used a *MAX_RADIUS* of 5, ϵ of 1, and ρ of 2. *MAX_RADIUS* represents the total radius a presence signal can travel, and thus once the signal reaches this distance from the sender the signal strength will be steady at each location until the emitting agent moves or dies. When an agent moves or dies, the signal will slowly decrease toward the original spot at the same rate that it diffused out.
(2) Participate in rescue protocol: Both PLEASE DIE and I DIED signals travel as in Algorithm 2. The meaning of the parameters is described in Section 4.3. The strength and radius were varied as described in Section 6. PLEASE DIE is only sent after a "push", whereas I DIED is only sent when an agent is dying due to receiving signals. Both signals can also be received at any time step and may induce self-death.

All probabilities are implemented via a pseudo random number generator. We chose the Mersenne Twister RNG, as it has a long period ($2^{19937} - 1$) and "good" randomness [15].

```
1: S = time_current − time_lastmove
2: if S < MAX_RADIUS then
3:     distance = 0
4:     for S > 0 do
5:         for all location(x,y,z) within distance (city block distance) do
6:             Chemicals(x, y, z) = Chemicals(x, y, z) + ϵ / MAX_RADIUS^p
7:         end for
8:         S = S − 1
9:         distance = distance + 1
10:    end for
11: end if
```

Algorithm 1. Presence signal diffusion.

```
1: distance = 1
2: for distance < radius do
3:     for all (x,y,z) location ∈ distance do
4:         send signal of strength: strength − (distance ∗ 1/(2∗radius))
5:     end for
6:     distance = distance + 1
7: end for
```

Algorithm 2. Signal propagation for rescue protocol.

6. Experimental results

Simulations were conducted to determine the signal parameters, replication probability, and various algorithmic properties that would result in eliminating the threat of irregular agents and retaining the population of healthy agents in the given scenario. All simulations ran for 8000 time ticks. Twenty different runs were executed with different random number generator seeds for each set of parameters evaluated. Although the TOS values for the PLEASE DIE and I DIED signals are separate, they were tested with the same values. Therefore, a given TOS value refers to the value that the PLEASE DIE TOS and the I DIED TOS each have separately, and they are not combined in any way. In each run we recorded the number of both irregular agents and the healthy agents since both measurements are needed to accurately describe the state of the system.

6.1. Determining parameters

We started with studying how the system behaves with different replication rates for healthy agents (Fig. 5). The replication rate must be set high enough to allow healthy agents to re-instantiate any agents that may have been lost due to random death. Without irregular agents in the system a rate near the death rate would be sufficient, but with irregular agents blocking many replications a higher rate is necessary. HADES was tested with rates of 0.01, 0.1, 0.2, 0.3, 0.4, 0.5. As seen in Fig. 5,[1] the low replication rate of 0.01 is the least resistant to irregular agents, and on average has the highest irregular agent count (around 500) and the lowest healthy agent count (less than 2500). Raising the replication rate to 0.1 improves both numbers, with irregular agents at 100 on average and healthy agents over 3500. Another increase to 0.2 decreases irregular agents to around 50 on average. The replication increases to 0.3, 0.4, and 0.5 do not change the average results from that of 0.2, therefore a replication rate of 0.2 is high enough to sustain healthy agents while diminishing irregular agents. These tests were conducted with signal radius of 1, signal strength of 3, and a PLEASE DIE TOS and I DIED TOS of 4.

[1] For colors in figures see the web version of this article.

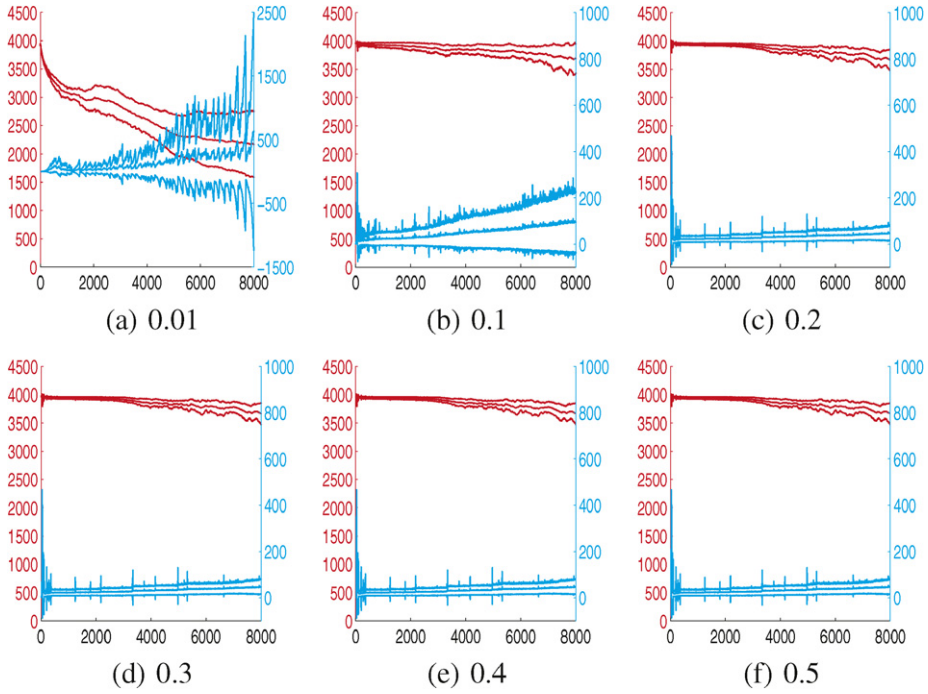


Fig. 5. Different replication probabilities with radius of 1, strength of 3, and a PLEASE DIE TOS and I DIED TOS of 4. The left y-axis corresponds to the healthy agents, and the right y-axis refers to the irregular agents. The x-axis is time. The middle line of each color is the average, with the upper and lower bounds representing the standard deviation. The best rates are 0.2 and up.

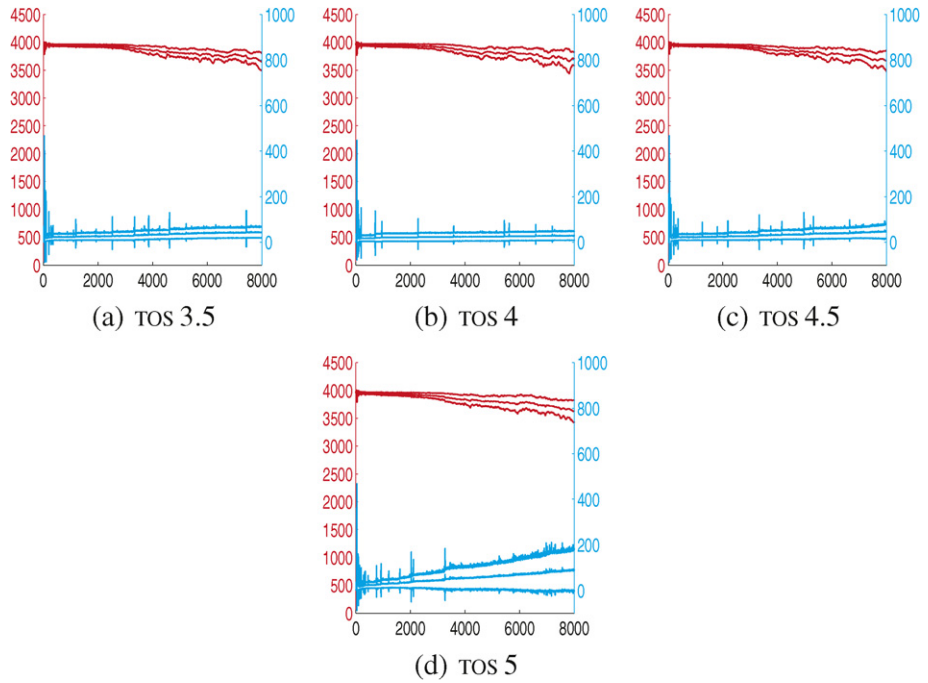


Fig. 6. Different TOS values for both the PLEASE DIE and I DIED signals with replication rate of 0.2 and signal radius of 1 and strength of 3. The left y-axis corresponds to the healthy agents, and the right y-axis refers to the irregular agents. The x-axis is time. The middle line of each color is the average, with the upper and lower bounds representing the standard deviation. The best value is 4, for target totals of both healthy and irregular agents.

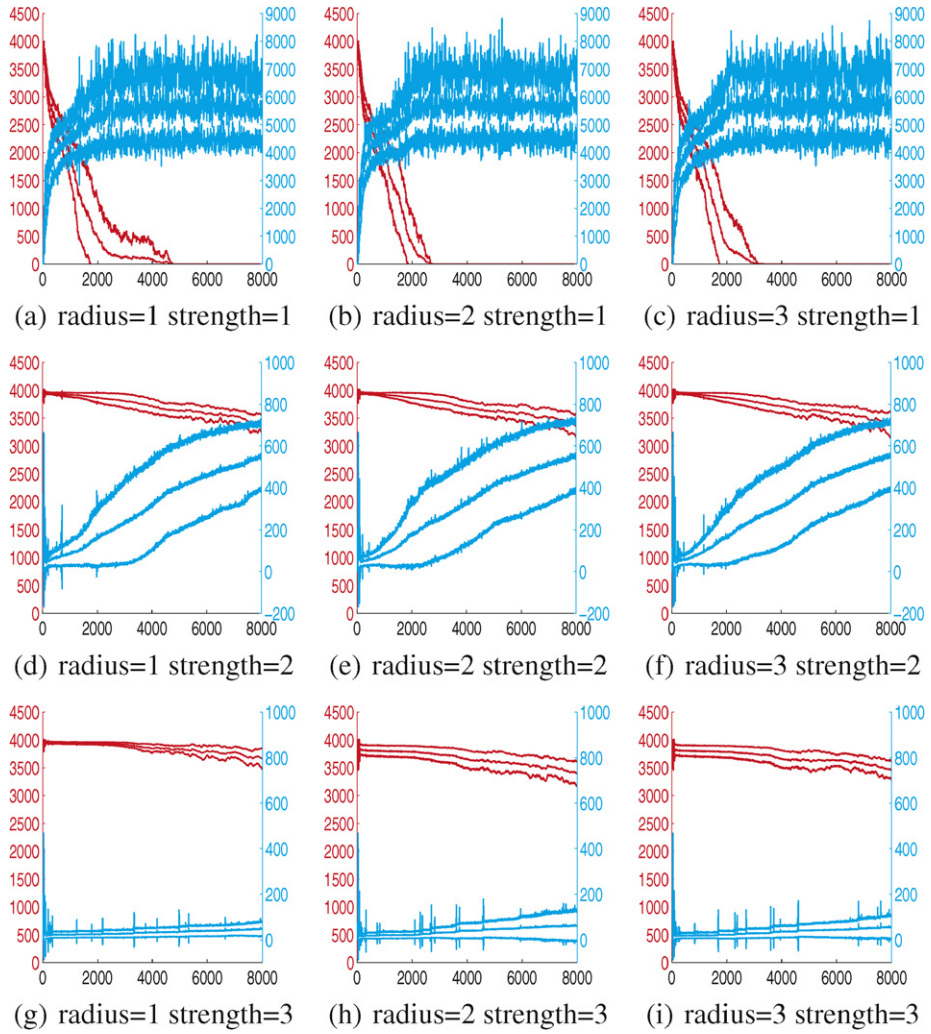


Fig. 7. Different values for signal strength and radius for both signal types. The left y-axis corresponds to the healthy agents, and the right y-axis refers to the irregular agents. The x-axis is time. The middle line of each color is the average, with the upper and lower bounds representing the standard deviation. A signal radius of 1 and strength of 3 performs the best.

We next verify the best I DIED and PLEASE DIE TOS values for the replication rate of 0.2 with a signal radius of 1 and strength of 3. As seen in Fig. 6, we tested TOS values of 3.5, 4, 4.5, and 5 for both signal types. The TOS value of 4 for both I DIED and PLEASE DIE signals is found to be the most beneficial, with the average number of irregular agents below 50. A value of 3.5 gives similar results for both healthy and irregular agents, so we could have chosen it instead and the results would be the same. A TOS of 4.5 for I DIED and PLEASE DIE signals results in a larger number of irregular agents with over 50 on average, and a value of 5 has irregular agents at 100. TOS values of 3 and less for I DIED and PLEASE DIE signals will cause agents neighboring the sending agent to die from that single signal, which makes the rescue protocol less robust as discussed in Section 4.1.

Signal radius and strength are also tested (Fig. 7), with both PLEASE DIE and I DIED having the same strength and radius. A strength of 1 is not sufficient as it results in the complete disappearance of healthy agents, and increasing the radius does not improve on that result. Increasing the strength to 2 decreases the irregular agent count by an order of magnitude, from 8000 to 800. Again, changing the radius does not significantly affect the result. For all strength of 2 scenarios the healthy agent average count is around 3500. Increasing the strength to 3 provides a more promising result, with all 3 average counts of irregular agents under 100. Setting the radius to 1 gives the lowest standard deviation, and the healthy agent average count continues to stay around 3500. Therefore, a TOS of 4, a strength of 3 and radius of 1 give the lowest average count of irregular agents and the highest average count of healthy agents.

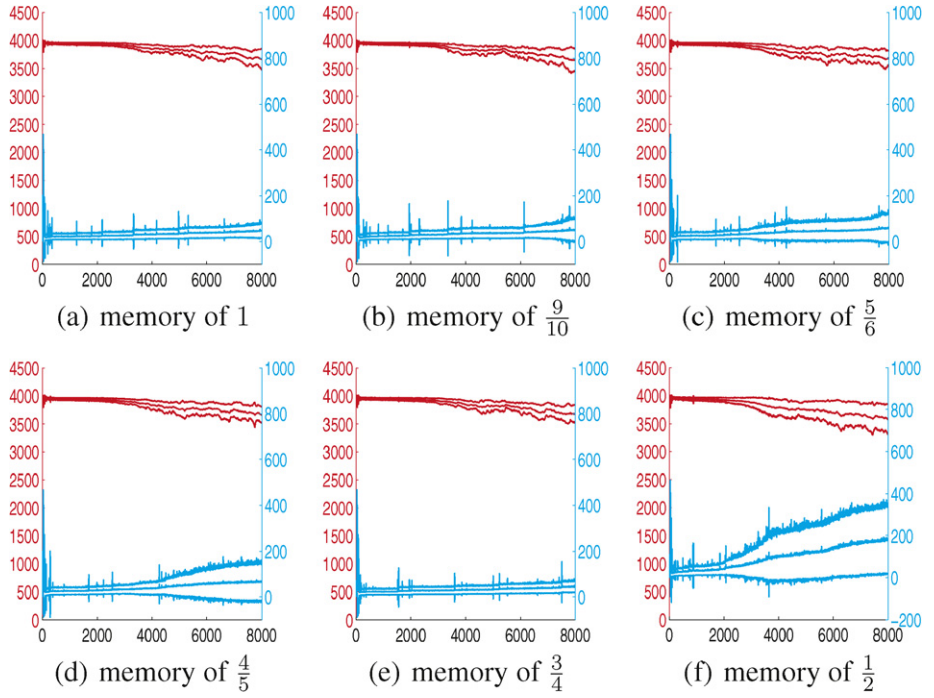


Fig. 8. Different signal retention values with replication rate of 0.2, TOS of 4, signal radius of 1 and strength of 3 for both signal types. The left y-axis corresponds to the healthy agents, and the right y-axis refers to the irregular agents. The x-axis is time. The middle line of each color is the average, with the upper and lower bounds representing the standard deviation. The best retention rate is $\frac{3}{4}$.

6.2. When to initiate self-death via signaling

An agent may bias toward recent signals and discount the older ones. Given a total of signal strengths received $SC_{k,n-1}$ for signal type k at time $n-1$, and a signal retention rate of $0 \leqslant \gamma \leqslant 1$, the total of signal strengths received over time at time n for signal type k is defined as

$$SC_{k,n} = SC_{k,n-1} * \gamma + \sum Signals\_Received_n. \tag{1}$$

We examined γ values of 1, $\frac{9}{10}$, $\frac{5}{6}$, $\frac{4}{5}$, $\frac{3}{4}$, and $\frac{1}{2}$ for both signal types. All of the results seen thus far have had γ values of 1 representing infinite memory. In this case $SC_{k,n}$ continues to grow until it reaches the signal threshold which then causes the death of the agent. By reducing γ we are increasing the time an agent survives by slowing the rate of growth of $SC_{k,n}$. Fig. 8 shows the resulting systems when replication is 0.2, signal radius is 1, and strength is 3 for both the I DIED and PLEASE DIE signals. The ability to remove irregular agents decreases as the signal retention rate decreases, except for the case of a signal retention rate of $\frac{3}{4}$. This case is actually slightly better than the infinite memory case ($\gamma = 1$), as it has higher regular agents and fewer irregular agents at the end of the runs. However, in both cases exactly 4 runs resulted in all irregular agents being removed, showing that they are very similar. The fact that the retention rate of $\frac{3}{4}$ does not lower system robustness demonstrates that due to parameter interactions, it is possible to find a successful case where agents have bounded memory. It is therefore preferable to have agents either use infinite memory or retain around 0.75 of the received signal strengths when using this set of parameters.

6.3. Delaying the start of the rescue protocol

We next investigate the end result on the system when a delay occurs between the appearance of the first irregular agent and the beginning of signaling among agents. Delay is implemented by the communication being turned off until the specified number of irregular agents exist in the system. This delay represents situations where an agent or system protocol requires some amount of irregularity prior to activation due to resource constraints or limited agent alertness. Fig. 9 shows the results of different delays when the replication rate is 0.2, signal radius is 1, and strength

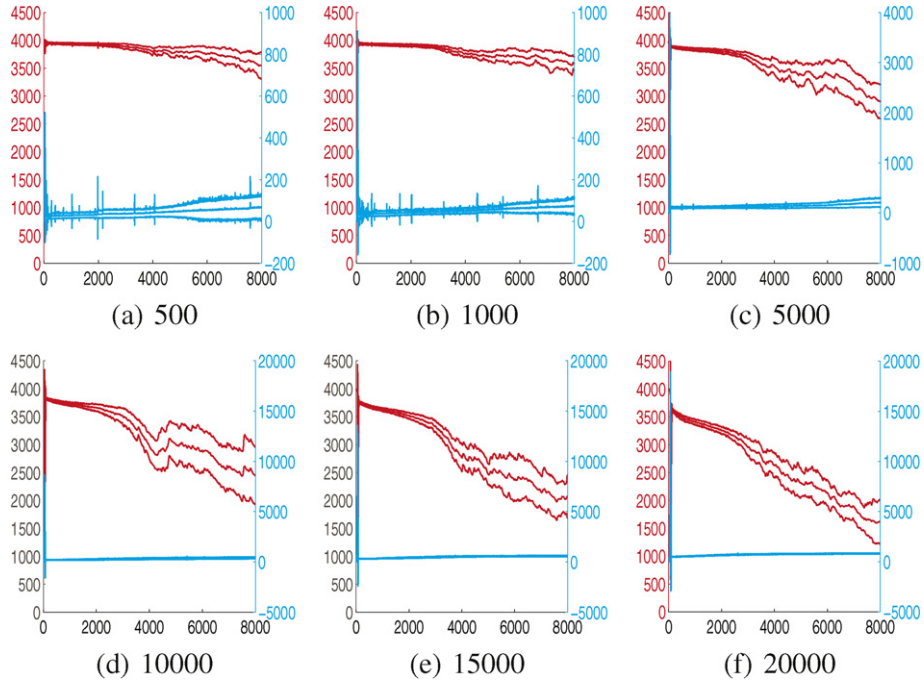


Fig. 9. Delays occur before signaling can be initiated. The left y-axis corresponds to the healthy agents, and the right y-axis refers to the irregular agents. The x-axis is time. The middle line of each color is the average, with the upper and lower bounds representing the standard deviation. Each caption represents the number of irregular agents prior to the activation of the protocol. Degradation occurs for a large delay in protocol activation.

is 3. For any of the delay periods tested, the algorithm confined the irregular agents. For delay amounts below 10,000 the average healthy agent count is at least 3000. However, for delay periods of 10,000 and greater the number of healthy agents was decreased to below 2500. The system cannot therefore recover from a delay over 10,000. The analogy between a signal delay and cancer treatment is compelling, as the signal delay can represent the stage of a tumor before it is found, which is a main factor in cancer outcome. A delay may also occur in an artificial system due to resource constraints.

7. Discussion and conclusion

HADES is a multi-agent system that is able to control and protect itself via life protocols and a rescue protocol. Its life protocols control the replication, repair, movement, and self-induced death that govern each agent in the system. These protocols would be sufficient to control HADES if errors were not possible. However, since each replication has a probability of mutation the system must be fault tolerant to deal with completely mutated agents. The rescue protocol has therefore been designed to allow agents to influence the death of neighbors due to violation of space constraints, thus enabling the system to self-maintain despite irregular agents. Our mechanism of inducing self-death is designed such that it could be applied to any self-regenerating system, making it a valuable tool for multi-agent systems.

During the design of the rescue protocol and numerous simulations we found that in order to fully extinguish the irregular agents, the protocol should include two kinds of signaling: PLEASE DIE and I DIED. In the first an agent that recognizes irregularity surrounding it sends PLEASE DIE messages, and in the second an agent that is going to die announces its death to the environment as a way of transferring the alert for irregularity to its neighbors. Without this combined set of signals, the system would not be able to remove an entire cluster of irregular agents while maintaining the overall system health. All scenarios with final averages below 100 irregular agents had at least one of the twenty runs remove all irregular agents. Therefore, our system is able to remove all irregular agents with certain random number generator values.

We demonstrated how to tune algorithm parameters to achieve the required results including an accumulating signal strength that lead to self-death, a signal radius, a signal strength, a threshold of signals, and a replication rate.

We estimated how the algorithm would function given a delayed start with a large number of irregular agents, and found that up to some threshold of delay HADES still succeeds in removing the irregular agents. However, if there is a lengthy delay HADES will not correct the irregular agent problem. We also experimented with limited agent memory with respect to rescue signals that have been received, showing that many different memory situations can still utilize this signaling technique. Our rescue protocol severely inhibits irregular agent generation even in highly complicated situations, showing a strong fault tolerance.

We intend to further investigate related scenarios for HADES and thus increase the range in which the rescue protocol succeeds. We will first continue to explore the parameter space for both the life protocol and rescue protocol. We will then consider cases where the irregular agents only partially participate in the rescue protocol. A percentage of signals received may be ignored, or the agent may fail to send the I DIED signal upon death. Both of these scenarios can lead to failure of the protocol, but further tailoring of the parameters may enable the system to be more robust, even in such cases. We will also model cases where the receiving agents activate their programmed death probabilistically when told by the protocol to activate it, and investigate how this provides different effects on the system when combined with changing reproduction and death rates. On-line tuning of parameters will be required for increased robustness.

Although our system is inspired by biology, the solution is designed for general multi-agent systems with citizenship where the agents share the goal of keeping the system functioning. As an autonomous sensor network, [2] is an example of this type of system. The improved sensor networks will be able to determine if incorrect data is initiated from a specific group of sensors, and thus send messages to shut them down. These sensors can then be repaired and re-introduced into the system. Another application for our rescue protocol is improving distributed software by adding real time repair. For instance, the system may have a process that runs concurrently to its other processes that enables it to declare fellow processes as damaged. This repair can be in the form of killing the process and then restarting a new one, as in HADES. A system that corrects C and C++ code as it runs was proposed in [18]. It does so by replicating the programs and comparing results from each replication, only using results with a majority agreement.

Repair mechanisms can also facilitate self-organization in agent systems as demonstrated in [19,20]. A group of cooperative autonomous agents working toward a system level goal can benefit from repair controlled self-organization. For instance, if a single agent's decision making fails it will put pressure on the system by providing incorrect output that could damage the system's organization. This output can eventually be repelled by the other agents via the rescue protocol, enabling the agents to re-organize the system after resetting or re-creating the faulty agent. We can improve on many systems that require continuous functioning in this way, as long as the control structures have relative autonomy.

We thus call on the AI community to open investigations on the use of regeneration and agent death by considering different systems and how the repair mechanism introduced by HADES can be applied to them. We also call on the AI community to think of their models in such a way that regeneration is more utilized.

Acknowledgements

This research was partially supported by NSF grant ECCS 0501432. This research was also performed under an appointment to the Department of Homeland Security (DHS) Scholarship and Fellowship Program, administered by the Oak Ridge Institute for Science and Education (ORISE) through an interagency agreement between the U.S. Department of Energy (DOE) and DHS. ORISE is managed by Oak Ridge Associated Universities (ORAU) under DOE contract number DE-AC05-06OR23100. All opinions expressed in this paper are the author's and do not necessarily reflect the policies and views of DHS, DOE, or ORAU/ORISE. The advise of Joe Jerry was very helpful in the beginning of the process. We also appreciate the comments and suggestions of our anonymous reviewers.

References

[1] R.C. Allsopp, H. Vaziri, C. Patterson, S. Goldstein, E.V. Younglai, A.B. Futcher, C.W. Greider, C.B. Harley, Telomere length predicts replicative capacity of human fibroblasts, Proceedings of the National Academy of Sciences 89 (1992) 10114–10118.

[2] T. Bokareva, N. Bulusu, S. Jha, Sasha: Toward a self-healing hybrid sensor network architecture, Embedded Networked Sensors (2005) 30–31.

[3] E.A. Coyle, L.P. Maguire, T.M. McGinnity, Self-repair of embedded systems, Engineering Applications of Artificial Intelligence 17 (1) (2004) 1–9.

[4] C. Dellarocas, M. Klein, J.A. Rodriguez-Aguilar, An exception-handling architecture for open electronic marketplaces of contract net software agents, in: Proceedings of the 2nd ACM Conference on Electronic Commerce, Minneapolis, 2000.
[5] A. Fedoruk, R. Deters, Improving fault-tolerance by replicating agents, in: AAMAS '02: Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems, ACM Press, New York, 2002.
[6] K. Francis, B. Palsson, Effective intercellular communication distances are determined by the relative time constants for cyto/chemokine secretion and diffusion, Proceedings of the National Academy of Sciences 94 (1997) 12258–12262.
[7] S. George, D. Evans, S. Marchette, A biological programming model for self-healing, in: SSRS '03: Proceedings of the 2003 ACM Workshop on Survivable and Self-Regenerative Systems, ACM Press, New York, 2003.
[8] S. Griffith, D. Goldwater, J. Jacobson, Self-replication from random parts, Nature 437 (2005) 636.
[9] W. Hamscher, L. Console, J. de Kleer, Readings in Model-Based Diagnosis, Morgan Kaufmann Publishers Inc., 1992.
[10] B. Horling, V. Lesser, R. Vincent, A. Bazzan, P. Xuan, Diagnosis as an integral part of multi-agent adaptability, in: Proceedings of DARPA Information Survivability Conference and Exposition, 2000, pp. 211–221.
[11] R. Humphreys, M. Krajewska, S. Krnacik, R. Jaeger, H. Weiher, S. Krajewski, J. Reed, J. Rosen, Apoptosis in the terminal endbud of the murine mammary gland: A mechanism of ductal morphogenesis, Development 122.
[12] M. Klein, J. Rodriguez-Aguilar, C. Dellarocas, Using domain-independent exception handling services to enable robust open multi-agent systems: The case of agent death, Autonomous Agents and Multi-Agent Systems 7 (2003) 179–189.
[13] S. Kumar, P.R. Cohen, H.J. Levesque, The adaptive agent architecture: Achieving fault-tolerance using persistent broker teams, ICMAS 00 (2000) 0159.
[14] D. Mange, M. Sipper, A. Stauffer, G. Tempesti, Toward self-repairing and self-replicating hardware: The embryonics approach, EH 00 (2000) 205.
[15] M. Matsumoto, T. Nishimura, Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator, ACM Trans. Model. Comput. Simul. 8 (1) (1998) 3–30.
[16] H. Meinhardt, A. Gierer, Generation and regeneration of sequence of structures during morphogenesis, Journal of Theoretical Biology 85 (3) (1980) 429–450.
[17] J. Miller, Evolving a self-repairing, self-regulating, French flag organism, in: Proceedings of GECCO, 2004.
[18] G. Novark, E.D. Berger, B.G. Zorn, Exterminator: Automatically correcting memory errors with high probability, in: Proceedings of the Conference on Programming Language Design and Implementation, 2007.
[19] M. Prokopenko, G. Poulton, D. Price, P. Wang, P. Valencia, N. Hoschke, T. Farmer, M. Hedley, C. Lewis, A. Scott, Self-organising impact sensing networks in robust aerospace vehicles, in: Idea Group, 2006, pp. 189–230, Chapter 7.
[20] M. Prokopenko, P. Wang, P. Valencia, D. Price, M. Foreman, A. Farmer, Self-organizing hierarchies in sensor and communication networks, Artificial Life 11 (4) (2005) 407–426.
[21] F. Roth, H.T. Siegelmann, R.J. Douglas, The self-construction and -repair of a foraging organism by explicitly specified development from a single cell, Artificial Life 13 (4) (2007) 347–368.
[22] H. Saidi, B. Dutertre, J. Levy, A. Valdes, Self-regenerative software components, in: SSRS '03: Proceedings of the 2003 ACM Workshop on Survivable and Self-Regenerative Systems, ACM Press, New York, 2003.
[23] R. Smith, The contract net protocol: High-level communication and control in a distributed problem solver, IEEE Transactions on Computers C-29 (12).
[24] R. Sterritt, M. Hinchey, Apoptosis and self-destruct: A contribution to autonomic agents?, in: Lecture Notes in Computer Science, vol. 3228, Springer, Berlin/Heidelberg, 2004, pp. 262–270.
[25] K. Toyama, G.D. Hager, If at first you don't succeed..., in: Proceedings of the 14th National Conference on Artificial Intelligence, 1997.
[26] J. von Neumann, Theory of Self-reproducing Automata, University of Illinois Press, Urbana, 1966.
[27] V. Zykov, E. Mytilinaios, B. Adams, H. Lipson, Robotics: Self-reproducing machines, Nature 435 (2005) 163–164.