Agent Support for Genetic Search in an Immunological Model of Sparse Distributed Memory

Keith E. Mathias and Jason S. Byassee
TRW Systems
16201 E. Centretech Pkwy.
Aurora, CO 80011
keith.mathias, jason.byassee@auc.trw.com

Abstract

This research focuses on agent migration strategies and communication behaviors in a sparse distributed memory implementation based on the human immune system. Evaluation of various agent strategy/behavior combinations is measured in the context of genetic search performance at multiple, independent system nodes. Results indicate that agent behaviors which promote and enhance information exchange between distributed nodes yield the best performance.

1 Introduction

Our research involves a sparse distributed memory (SDM) where the theoretical memory capacity far outweighs the physical memory space (i.e., the ratio of memory cells to items represented is 1 to n, where $n >> 1$). This model is based in part on the human immune system, wherein memory persists in the form of a relatively modest population of antibodies (10^7 to 10^8) with a high affinity to a much greater number (10^{12} to 10^{16}) of possible antigen strains [5]. An effective SDM must develop and maintain a sufficient (with respect to quality) population of memory cells so that associative recall is not only feasible, but efficient.

The memory cell population in this SDM is sparsely distributed in representation space and physically distributed in the execution environment. In this system, mobile software agents circulate a limited number of memory cells between system nodes (Figure 1). Distributed, independent genetic search is leveraged in order to develop a system-wide memory cell population. Emergent behavior at the system level is a result of interactions between simultaneous and independent genetic searches, as well as, local feedback decisions.

Significant work has been performed with respect to agent strategies and enhanced distributed communication performance [3, 4]. This research differs in that we seek to examine the impact of mobile agents with respect to their migration strategies and communication behaviors to improve genetic search performance. Improved genetic search performance in turn, results in a more efficient SDM.

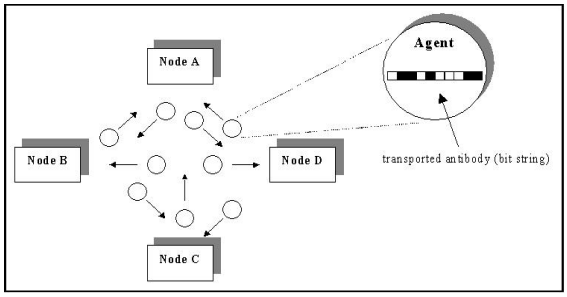

Figure 1: Agent Circulation of Antibodies.

2 Sparse Distributed Memory

This investigation is influenced by previous work that incorporates genetic algorithms in an immune system model to explore pattern recognition [2]. We have modeled the problem space using Hamming space (i.e., bit strings). In training the SDM, the objective is to dynamically develop "immunity" to patterns that are repeatedly re-introduced from a fixed library of patterns. Immunity is achieved by evolving a memory cell population that generalizes to adequately represent a much larger set of random bit string patterns. The random set of bit string patterns that must be matched is known here as the **antigen library**. The system population of memory cells, known as **antibodies**, consists of a small (relative to the size of the antigen library) collection of bit string patterns. Antibody evolution (i.e., system level learning) is a result

of isolated genetic search, local feedback decisions, and the ability of mobile agents to maintain an adequate distribution and diversity of antibodies between system nodes.

2.1 System Operation

This SDM consists of two core operations. The first is genetic search, taking place simultaneously and continuously on each node. The second involves mobile agents that circulate antibodies between the system nodes. Genetic search is used to perform pattern matching at each node, where each node randomly samples from a common antigen library, similar to Forrest, et. al. [2]. When an antigen sample is taken, the resident antibodies in the input queue on each respective node are compared against the sample. If a match is found, the search is complete and the node prepares for a new antigen sample. If a match is not found, the antibody population in the local input queue is used to seed the initial population for genetic search. The sampling of an antigen and searching for a match constitute a **cycle**.

The final population for each independent search includes the solution (i.e., antibody pattern matching the sampled antigen) and antibodies that are similar to the antigen, but not necessarily perfect matches. This provides the opportunity to feed patterns (in the form of antibodies) back into the system that are similar to the current antigen sample. The system antibody population subsequently evolves with representatives that have high affinity for the antigen library.

In this SDM, the mobile agents operate autonomously. From the perspective of each node, agents continuously arrive, deposit antibodies in the input queue, retrieve new antibodies from the output queue and transport them to new nodes (Figure 2). Meanwhile, patterns are continuously sampled from the antigen library as described above. When a sample is taken from the antigen library that must be matched, an initial population of 50 individuals is constructed to start genetic search. To take advantage of the system's learned knowledge, the initial population is comprised of: 1) antibodies taken from the local input queue, 2) copies of antibodies currently waiting in the output queue and 3) mutated copies of antibodies from steps 1 and 2. Copying and mutating antibodies is repeated until the initial population is complete.

In order to bound the size of the antibody population while promoting quality information in the system, we have introduced a survival scoring mechanism based on 1) age and 2) affinity to antigen library samples. This rewards antibodies for survival time (long-term reward) and for scoring well against the current antigen sample (short-term reward). The survival score is the sum of the *age* and $affinity$ values.

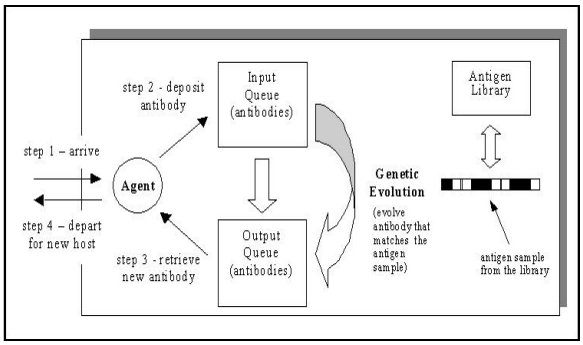

Figure 2: Simultaneous Activities At Each Node.

The *age* of an antibody corresponds to the number of nodes that it has visited since creation. A new antibody has an age of zero, and this value is subsequently incremented by one for each new search in which it is used as an initial seed. The $affinity$ is based on the percentage of bits that match antigen samples. This value is initially set to 100, giving newly created antibodies a chance to survive infancy. The value is subsequently decremented at each feedback step. The affinity value is reset to the affinity for the current antigen sample if that score is greater than the current affinity.

At the conclusion of every genetic search on each system node (when a given antigen sample is matched), a competition takes place to determine which antibodies are fed back into the system. At each competition, individuals in the antibody population that were in the input queue prior to genetic search (i.e., seeds) are compared with individuals from the final search population. The highest scoring antibodies are fed back into the system, and the remainder are discarded. A search *feedback threshold* allows individuals from the final search population that are not exact matches to be competitive in the feedback competition.

2.2 Pattern Matching Application

These experiments were designed to examine the impact of agent behavior and agent mobility strategies on the performance of this SDM. Performance in this context is measured with respect to the work necessary to discover the antibody strings that match the antigens sampled at the nodes in the system over the course of time. The antibody population consists of bit strings that are used to seed the population at local nodes for genetic search in order to match the patterns sampled from the antigen library.

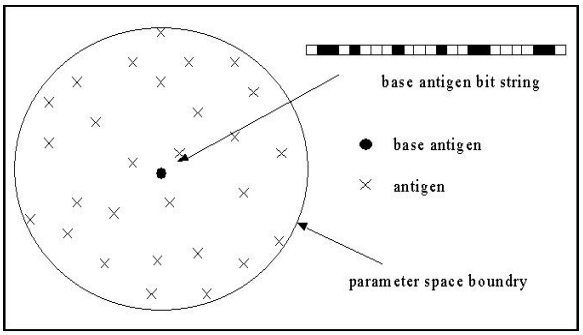

Figure 3: Antigen Library Distribution.

In each of these experiments, a parameter space is defined surrounding a randomly generated bit string, known as the **base antigen string**. An antigen library is generated in a binomial distribution around the base antigen, bounded by a given Hamming distance (Figure 3). The distribution reflects the frequency of occurrence of samples based on their Hamming distance from the base antigen. Thus, bit strings that are closer to the base antigen string are included in the library less frequently than those that are farther from the base string. This distribution is similar to that of B and T cell clones as modeled by Smith, et. al. [6], in simulations of immune system models.

In training the SDM, success is measured by the ability to generate antibodies quickly (i.e., few trials) that match strings sampled from the antigen library. A trivial solution to this problem is to generate one or more antibodies that match each antigen sampled (i.e., specialists). However, in a sparse distributed memory, the address space can be orders of magnitude larger than the instantiated address locations[6]. We used an antigen to antibody ratio in this system of 10:1. This means that, on average, one antibody must represent ten antigens. The hypothesis is that good system-wide performance requires an antibody circulation scheme that promotes an antibody population comprised of "generalist" antibodies (as opposed to specialists).

2.3 Migration Strategies and Communication Behaviors

Within the context of the SDM described, we have devised three agent migration strategies and three agent communication behaviors for investigation. This results in nine agent strategy/behavior combinations. The three agent migration strategies are as follows:

1. **Random Migration** - In this migration strategy, agents circulate antibodies by moving at random between system nodes.
2. **Directed Migration** - This migration strategy is intended to promote maximum diversity by correlating agent movement with the specific antibody that is being transported. Each node maintains a *most recently sent queue* that consists of a single entry representing every other node in the system. Each entry associates a node ID with the last antibody transported by an agent to the respective remote node from the local node. When an agent retrieves an antibody to transport, the Hamming distance between that antibody and every other entry in the queue is measured. The agent selects the destination node based on the entry that is furthest away in Hamming space.
3. **Cyclic Migration** - Cyclic migration agents move in a fixed pattern between system nodes. Each agent generates a random itinerary upon creation that includes a single visit to each node (i.e., Hamiltonian cycle).

The three communication behaviors are as follows:

1. **Always Communicate** - This is a simple behavior that requires no thinking, or decision process, on behalf of the agent. The agent simply deposits the transported antibody into the input queue of the node on which it arrives.
2. **Just In Time (JIT) Communication** - Agents search for a host that has just sampled an antigen from the library and is ready to begin genetic search. Agents continue to move until a host in this state is found, and then they deposit the transported antibody, "just in time" to begin genetic search.
3. **Load Balanced Communication** - Agents have a tendency to move away from other agents when exhibiting a load balancing communication behavior. This behavior forces agents to move away from "busy" nodes, thereby evenly distributing the antibody population among the system nodes.

2.4 CHC Algorithm

There are numerous genetic search approaches that could be used in the context of this SDM. We have chosen to incorporate the CHC adaptive search algorithm for antibody evolution at the local nodes. The CHC adaptive search algorithm [1] is a generational genetic algorithm that has been shown to yield very good performance for optimizing a wide variety of test problems and requires no parameter tuning [7]. Mating in CHC is performed by randomly pairing parents

and applying the HUX crossover operator. HUX exchanges exactly half of the bits that differ between the mates. Crossover is only performed if the differences between mates is greater than a threshold that is dynamically adjusted during the search process. This is known as *incest prevention* and serves to slow genetic convergence appreciably. Selection is performed using the $(\mu + \lambda)$ strategy, preserving the best N individuals from the child and parent populations, where N is the population size. When the population does converge, the search process is restarted via *cataclysmic mutation*. The population is filled with copies of the best individual and then 35% of the bits in all but one individual are complemented and search is restarted.

3 Experimental Conditions

For these experiments, the three migration strategies were paired with all three of the communication behaviors. These strategy/behavior combinations were also compared against the performance of a control strategy wherein no agents were used (i.e., no communication between nodes was possible). Table 1 identifies the system parameter values used for all experiments.[1] To emulate a sparse distributed memory, we maintained a 10:1 antigen to antibody ratio. To adequately sample the antigen pool, each simulation consisted of 5,000 cycles[2] at each of four nodes. This allows examination of a full range of behavior without spurious states (due to early termination) and an even sampling distribution of the patterns from the antigen library.

4 Results

All nine of the agent migration strategy/ communication behavior combinations were simulated for 30 independent runs. Combining all agent communication behaviors with all of the agent migration strategies provides a complete factorial design, supporting analysis of variance (ANOVA) testing. This allowed us to determine if any of the migration strategies or communication behaviors resulted in a statistically significant performance advantage or disadvantage (i.e., significant main effect).

[1]To support a fair comparison between the migration strategy/communication behavior pairs, the parameter values for the *total number of agents*, *antibody mutation rate*, and *search feedback threshold* were established via a separate search using a meta-GA. The purpose of the meta-GA was to find a good, if not optimal, set of parameter values for operation of this system.

[2]Genetic search is used to find an antibody that matches an antigen sample at each cycle except when a perfect match resides in the initial genetic population.

Parameter	Value
System Nodes	4
Antigen Library Size	320
System Antibody Population	32
Antibody String Length (bits)	32
Parameter Radius (bits)	5
Cycles (antigen samples/node)	5000
Time Between Search at each Node (msec)	20
Total Agents	24
Antibody Mutation Rate (bits)	3
Search Feedback Threshold (bits)	5

Table 1: System Operational Parameters.

Table 2 shows the average number of trials (and standard error of the means - SEM) to match antigen samples for each of the strategy/behavior combinations. The random agent migration strategy paired with the always communicate behavior expended the least amount of work (i.e, fewest trials), on average, to discover the antibodies that match the antigens sampled in the simulations. However, this performance advantage is only statistically significantly better than a few of the other cells in Table 2[3] (particularly the directed/JIT combination). The average trials to match the antigens sampled using the cyclic agent migration strategy are significantly worse than any of the other strategy/behavior combinations. ANOVA tests confirm this fact as a significant main effect.

Comm. Behavior	Migration Strategy		
	Random	Directed	Cyclic
Always	283.1 (0.73)	285.1 (1.19)	300.8 (1.69)
JIT	285.5 (0.66)	286.2 (0.72)	301.1 (1.87)
Load Bal	284.3 (1.46)	287.1 (2.61)	306.2 (2.13)

Table 2: Average Trials to Match Antigen Samples. When no agents are present, 298.7 trials (SEM = 1.49) are needed to match the pattern, on average.

The cyclic migration strategy combined with the load balancing communication behavior results in the worst performance, relative to all other strategy/behavior combinations. This performance is 10 standard errors worse than the random migration, always communicate runs and more than 2 standard errors worse than the experimental runs with the cyclic/JIT implementation. In fact, it is even inferior to the performance of simulations where no agents were present in the system. The average trials to match the antigens sampled when no agents are present (i.e., antibodies are not circulated) is 298.7 (SEM = 1.49).

[3]This may be due to the stochastic nature of the simulations contributing more noise than the variance between the strategy/behavior combinations.

It is important to note that the system learns and performs significantly better than genetic search on an equivalent problem when a random initial population is used. For example, on average, CHC solves a 32-bit one-max[4] problem in 504 trials (SEM = 9.12), when beginning with a random initial population.

4.1 An Emergent Behavior

Although the goal of genetic search at each local node is to match antigen library samples, the search efforts combined with the local survival decisions result in a globally emergent behavior. The communication of information via agents consistently resulted in an interesting phenomenon, notably the discovery and propagation of the string pattern used to create the antigen library (i.e., the base antigen string). Table 3 shows the average percentage of the system's final antibody population occupied by copies of strings matching the base antigen for each respective strategy/behavior experiment. This is referred to as the **saturation rate**.

Comm. Behavior	Migration Strategy		
	Random	Directed	Cyclic
Always	100.0% (0.00)	100.0% (0.00)	89.7% (1.04)
JIT	100.0% (0.00)	100.0% (0.00)	89.1% (1.35)
Load Bal	96.1% (0.74)	97.2% (0.68)	82.9% (3.40)

Table 3: Antibody Population Saturation Rate. When no agents are present, the average saturation rate is 91.1% (SEM = 0.50).

ANOVA testing confirms the trends evident by visual inspection as significant. First, the load-balancing communication behavior does not allow the base antigen to saturate the system, regardless of the agent migration strategy employed. The cyclic agent migration strategy also prevents the base antigen string from saturating the antibody population. This seems obvious in hindsight as the cyclic migration strategy is the most restrictive of the migration strategies. Visitation by a given agent is not equally likely at all nodes at each time step for this migration strategy. This restriction is so severe in fact, that the results were comparable to runs where no agents were present in the system. While simulations using no agents did discover this base antigen string, the simulations yielded an average saturation rate of 91.1% (SEM = 0.50).

The discovery of antibodies that match the base antigen string cannot be a result of searches in which the base antigen is sampled from the antigen library. Antibodies fed back to the system must meet or exceed a *feedback threshold* of five bits.

In searching for strings to match samples from the antigen library, each node contributes strings to the system antibody competition that have a large number of bits in common with the base antigen. This may or may not be sufficient for a given antibody to survive the feedback competition and be propagated to other nodes. However, those strings that are close to the base antigen string in Hamming distance will also likely score well against other antigens, if kept in the system antibody population. This causes the system antibody population to accumulate alleles in common with the base antigen string. When the antibody population is viewed as a probability vector that represents the percentage of 0- or 1-bits at each locus over the strings in the antibody population, this vector will approximate the base antigen string more accurately over time.

Eventually, an antibody matching the base antigen is a by-product of a search for another antigen library sample. Antibody copies of the base antigen string will likely perform well in the feedback competitions at each node, and chances of survival in the system will be better than average. After surviving in the antibody population for several cycles, the age weighting guarantees future survival.

The base string is very rarely useful in exactly matching any string in the antigen library (a 1 in 320 chance), yet this string serves as a good seed string for the genetic search. The discovery of the base string may or may not be an optimal system-wide strategy for learning how best to reduce the number of trials required to evolve an antibody that matches an antigen sample. For example, the discovery of four antibodies that divide the antigen library into equally sized attraction basins, based on Hamming distance, might work as well as, or better than, a single generalist. Regardless, the discovery of the base string is an interesting example of local behavior that facilitates emergent global behavior.

The best performances shown in Table 2 generally correspond with complete saturation (Table 3), yet there is not a perfect correlation. For example, the load balancing/random migration implementation performs quite well, but does not exhibit complete saturation. Therefore, saturation of the antibody population with the base antigen must not be the only factor in obtaining good performance.

[4]A one-max problem is equivalent to finding a matching bit-string using an evaluation score that reports the number (or percentage) of bits matching another pattern.

4.2 Propagation of Information

To further explore the correlation between the propagation of quality information and the efficiency of discovering antibody/antigen matches, we measured the average number of cycles required to: 1) discover the base string, 2) propagate the base string to all nodes *after* it has been discovered, and 3) saturate the system after a copy of the base string has been seen at all nodes. Table 4 shows these results.[5]

Comm. Behavior	Migration Strategy		
	Random	Directed	Cyclic
Avg. Cycles to Discover Base String			
Always	138.4 (24.5)	234.4 (51.7)	123.4 (19.9)
JIT	147.2 (23.6)	153.0 (26.6)	182.3 (35.7)
Load Bal	195.4 (37.2)	153.6 (32.4)	123.5 (20.9)
Avg. Additional Cycles to Circulate Base String to All Nodes			
Always	36.4 (12.4)	60.4 (22.1)	818.7 (77.7)
JIT	35.6 (12.4)	58.8 (17.8)	1018.9 (135.1)
Load Bal	19.4 (4.3)	89.4 (31.5)	1278.1 (130.3)
Avg. Additional Cycles to Saturate With Base String			
Always	1083.9 (154.4)	1106.6 (178.0)	*3906 (0)
JIT	865.1 (86.8)	973.2 (107.4)	*3798.5 (831.5)
Load Bal	*2944.5(410.3)	*2943.4 (378.5)	*2013 (0)

Table 4: Average Cycles (SEM) to Discover, Circulate and Saturate the Antibody Population.

ANOVA testing shows that there is no significant main effect in the time taken to *discover* the base antigen by any of the strategy/behavior combinations. Surprisingly, the average cycles for the experimental runs using a cyclic agent migration strategy and the always and load-balancing communication behaviors are better than the other strategies at discovering the base string (but not significantly so in most cases, due to large SEM values). The no agent strategy required, an average of 163 cycles (SEM=19) to discover the base string. This is comparable with most cells in Table 4.

There is a significant main effect seen in the number of cycles required to propagate the base string to all of the nodes after it has been discovered. In fact, it is at this stage of the simulation that those strategy/behavior combinations that incorporate the cyclic agent migration strategy experience a significant disadvantage, as compared to the other strategy/behavior combinations. In fact, the number of cycles needed by the cyclic agent migration strategy to propagate the base string to all other nodes after discovery is comparable with having no agents in the system. On average, the SDM runs where no agents are employed require 1158 cycles (SEM=102) after the initial discovery of the base string, until all nodes have independently discovered the base string.

There is also a weak main effect that indicates that the load-balancing behavior is slower at propagating the base string to all nodes after discovery than either the always or JIT communication behaviors. However, this trend is to be treated carefully, as there is an obvious exception. The random migration strategy that incorporates the load-balancing behavior appears to be considerably faster at propagating the base string among all of the nodes. We performed several repetitions of the complete factorial design and this was the only occurrence of this rapid propagation of information (while all other trends were verified).

The ANOVA tests could not be performed for the average number of cycles between complete circulation and saturation due to the fact that all 30 experimental runs for every strategy/behavior combination did not saturate. However, it can be observed that the random and directed strategies that use the load-balancing behavior do not propagate the base string nearly as well as when the always and JIT communication behaviors were employed.

4.3 Performance at Various Stages of the Simulation

There is an obvious difference in the ability of the strategy/behavior combinations to propagate information (although that information does not always appear to expedite search speed). It seemed prudent to test the hypothesis that the discovery of the base string does in fact affect the number of trials to match an antigen. Table 5 shows the average number of trials (and SEM) required to match an antigen during the stages relative to: 1) discovering the base string, 2) propagating the base string to all nodes after the first discovery, 3) between circulating the base to all nodes and saturation occurring, and 4) after saturation.[5]

ANOVA tests show that there is indeed a significant main effect where the discovery of the base antigen reduces the average number of trials required to match a sampled antigen. This holds true for all strategy/behavior combinations but could not be confirmed for the final two stages of simulation (i.e., after circulation and after saturation), since all runs did not saturate. The cyclic migration strategies performed consistently worse than the random and directed migration strategies, although it is not statistically significant. Therefore, the average cycles for the cyclic agent migration strategy between base string circulation among all nodes and population saturation (Table 4) must account for the significant performance

[5]The * indicates that all 30 runs did not saturate. Average cycles reported, include only those runs that did saturate.

Comm. Behavior	Migration Strategy		
	Random	Directed	Cyclic
Stage 1 - Prior to Base String Discovery			
Always	380.0 (5.46)	371.8 (3.96)	392.5 (9.34)
JIT	375.1 (4.60)	381.8 (4.79)	373.7 (2.67)
Load Bal	371.5 (2.98)	390.0 (9.98)	385.7 (5.63)
Stage 2 - Between Base String Discovery & Circulation			
Always	346.7 (2.71)	344.6 (2.59)	338.8 (2.51)
JIT	350.6 (2.95)	355.9 (3.19)	337.9 (2.60)
Load Bal	346.3 (2.72)	353.4 (3.18)	339.2 (2.19)
Stage 3 - Between Base String Circulation & Saturation			
Always	*291.5 (2.70)	291.0 (2.77)	*278.3 (0.0)
JIT	290.5 (1.92)	292.7 (2.20)	*279.1 (0.7)
Load Bal	*282.4 (2.66)	*284.1 (1.33)	*282.1 (0.0)
Stage 4 - After Saturation			
Always	*278.2 (0.33)	278.3 (0.29)	N/A
JIT	280.8 (0.28)	280.2 (0.30)	N/A
Load Bal	*276.0 (0.47)	*275.3 (0.61)	*274.6 (0.0)

Table 5: Average Trials to Match Antigen Samples During Critical Stages of Simulation.

differences observed in Table 5. This is also consistent with the infrequent saturation rates exhibited by the cyclic migration strategy.

Figure 4 shows the number of trials required to match an antigen for the first 1,500 samples of the 5,000 cycle simulation at one of the four nodes for a single representative run. Trials are shown on the Y-axis while cycles are shown on the X-axis. The open circles indicate the trials required to match an antigen during a particular cycle, and the black line represents the running average (lag = 100). The base string is first discovered at cycle 259. The trials to discover a match for the antigen samples begins to decrease at this point. The running average reaches a low of approximately 250 trials by cycle 410, where the system antibody population saturates with the base string.

4.3.1 The Effects of Seeding Genetic Search in the SDM Simulation

An unusual behavior observed in Figure 4 is the occurrence of searches that expend two to three times the normal number of trials to find an antibody/antigen match. This is indicative of seeding the initial population for the CHC search in a biased manner, risking the incidence where the correct allele is not present in any member of the initial population. Since CHC does not employ mutation, except at divergences, the search will converge to an antibody string that does not match the antigen sample, and hence cataclysmic mutation will be performed to restart the search. Such an event can significantly impact the number of trials necessary to find the matching string.

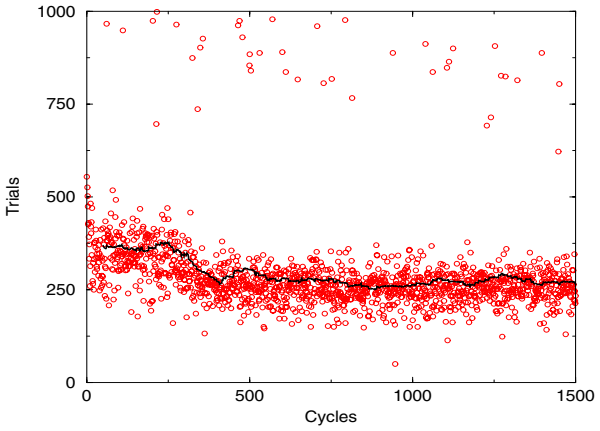

Figure 4: Search Profile At A Single Node For First 1,500 Cycles.

This phenomenon occurs here due to the seeding procedure. Each genetic search is started by seeding the population with three-bit mutations of the antibodies available in the input queue, where each antibody in the input queue seeds an equal fraction of the genetic population. The expected difference between the base string and an antigen sample is five bits. Hence, the expected difference between antibodies that are fed back from the genetic search (i.e., at least Hamming distance five from the antigen just matched) and a another antigen sample is ten bits. Therefore, the occurrence of restarts is not unexpected. It is important then, that genetic meta-search was used to discover the seeded mutation value (i.e., three bits) as opposed to arbitrary determination.

5 Conclusions

Evidence from this study illustrates that seeding the initial population with a single "generalist" pattern can expedite genetic search for other related patterns. In this context, a generalist pattern can form an effective sparse representation for a library of patterns. This SDM learns that a good strategy for reducing the work necessary to match antigen library samples is to evolve and propagate antibodies matching the base antigen string. Performance analysis reveals that providing feedback from the final population used in genetic search is sufficient to discover such a generalist, even when the genetic material does not contain precise matches for the antigen samples.

It is clear that the use of mobile agents to circulate genetic material between nodes expedites the discovery and propagation of the base antigen string. Without agents to circulate the information, each node must

discover the base string independently. The number of trials required to match an antigen is significantly reduced when the base antigen is discovered and its representation (antibody copies) is shared in the system via mobile agents. This is evident from comparing the performances of the simulations using random and directed migration strategies with the performances of simulations implementing cyclic migration, where communication is hindered, or those containing no agents, where communication is non-existent. Simulations using the cyclic agent migration strategy are unable to propagate the base string throughout the system and do not consistently saturate the antibody population.

Additional analysis of system behavior provides insight into the operational dynamics of each agent communication behavior. Although not impacting total trials, the load balancing behavior did not saturate the population in all instances. The relatively large number of cycles required for this communication behavior to saturate the population with copies of the antibody matching the base antigen (Table 4) gives rise to the hypothesis that agents implementing this behavior may be hiding quality material while "looking" for non-busy nodes.

On average, the JIT communication behavior provides more antibodies to begin each search. This behavior did in fact expedite the antibody population saturation, as evident from the number of cycles to saturate the population with the base string (Table 4), although it did not seem to significantly impact the performance metric used in the experiments. Examining metrics beyond total average trials suggests that additional experiments run with different performance criteria (such as reducing the number of cycles per node) could very well serve as a significant discriminator among communication behaviors.

The directed agent migration strategy was designed to promote maximum antibody diversity in the system. This objective was not realized with respect to improved genetic search performance (Table 2). In fact, directed migration did not perform quite as well as random migration during several simulation stages (Tables 4 and 5). We surmise that *near* real-time knowledge (as opposed to real-time) contained in the *most recently sent queue* mitigates the anticipated advantage of antibody diversity promotion. This is a result of the decentralized implementation, where an instantaneous global snapshot of node state is not available.

Thorough study of agent behaviors and migration strategies is a valuable performance analysis exercise. This investigation illustrates that various communication implementations can yield surprising results. A restrictive agent strategy (cyclic migration), conducive to uneven visitation, performed worse than simulations using no agents. Agent implementations employing more complex strategies and behaviors (such as those that are based on current system state or require coordination) are not always performance leaders. Our results indicate that a greater degree of agent autonomy, where agents make simple, independent decisions, facilitates expedited genetic search that improves sparse distributed memory performance.

References

[1] Larry Eshelman. The CHC Adaptive Search Algorithm. How to Have Safe Search When Engaging in Nontraditional Genetic Recombination. In G. Rawlins, editor, *Foundations of Genetic Algorithms*, pages 265–283. Morgan Kaufmann, 1991.

[2] Stephanie Forrest, Robert Smith, Brenda Javornik, and Alan Perelson. Using genetic algorithms to explore pattern recognition in the immune system. *Journal of Evolutionary Computation*, 1(3):191–211, 1993.

[3] Ashraf Iqbal, Joachim Baumann, and Markus Straßer. Efficient algorithms to find optimal agent migration strategies. Technical Report 1985/05, Universitat Stuttgart Fakultat Informatik, Bericht Nr., 1998.

[4] Frederick Knabe. Performance oriented implementation strategies for a mobile agent language. In *Lecture Notes in Computer Science Series*, pages 229–243. Springer-Verlag, 1997.

[5] Derek Smith, Stephanie Forrest, and Alan Perelson. Immunological Memory is Associative. In Dipankar Dasgupta, editor, *Artificial Immune Systems and Their Applications*, pages 105–112. Springer, 1998.

[6] Derek Smith, Stephanie Forrest, Alan Perelson, and David Ackley. Using lazy evaluation to simulate realistic-size repertoires in models of the immune system. *Bulletin of Mathematical Biology*, 60:647–658, 1997.

[7] Darrell Whitley, Keith Mathias, Soraya Rana, and John Dzubera. Building Better Test Functions. In L. Eshelman, editor, *Proceedings of the Sixth International Conference on Genetic Algorithms*. Morgan Kaufmann, 1995.