Expediting Genetic Search with Dynamic Memory

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

Abstract

Exploitation of domain knowledge can expedite the process of finding solutions to new problems. This research is focused on a distributed memory system which maintains a dynamic knowledge base, in the form of memory cells, that are employed to improve search performance over time. Short-term and long-term memory models are analyzed in the context of the distributed memory system. Results indicate that genetic search performance is significantly impacted by the quantity and quality of information that is maintained in memory.

1 Introduction

Many real-world optimization problems are time sensitive, where unbounded time to find the optimal solution is not practical. In these instances, execution time can be expedited by leveraging domain knowledge, providing a "starting point" for the search algorithm. Extensive research has been performed in the context of incorporating *a-priori* knowledge to improve genetic search performance [4, 5]. However, it is still unclear how the quantity and quality of information that constitutes the knowledge base affect search performance. This provides the motivation for examining the impact of short-term and long-term memory on genetic search performance, in terms of both quantitative and qualitative metrics.

We have developed a distributed memory system (DMS) that serves as the test environment for our analysis. In this DMS, genetic search is employed as the search mechanism on the system nodes. The nodes operate independently and simultaneously, sampling and solving different pattern matching problems from a shared library. The DMS incorporates a dynamic pool of memory cells that is shared between nodes. However, information from any one cell is available at only one node at a time. The memory cells evolve with continuous feedback from each independent genetic search. Each node in the DMS employs genetic search to solve an independent problem as opposed to all nodes working together to solve a single problem, thereby differentiating this work from most parallel genetic algorithm research.

A secondary goal for each search is to improve the aggregate search performance of all nodes by sharing information about problems that may be encountered by other nodes over time. Our objective is to compare genetic search performance when knowledge is shared via short- or long-term memory (of limited size) for initial seeding rather than random initialization.

2 Distributed Memory System

Simple bit-pattern recognition problems form the bounded problem library for this work.[1] The system is trained to recognize a set of bit string patterns in a pattern library by evolving and maintaining dynamic memory cells that may be shared between nodes via communication. The memory cells (in the form of bit string patterns) are evolved by local genetic search and fed back into the system.

2.1 Operation

This DMS consists of two core operations. The first is genetic search, taking place simultaneously and continuously on each node. The second involves mobile

[1]The GA may not be the best choice for simple pattern matching problems. This application was selected so as to focus on performance metrics in the context of exploiting memory for subsequent search. We anticipate using this system for complex pattern recognition tasks in the future.

agents that circulate the memory cells between independent nodes. While the memory cells can travel to any node in the system, a cell is only useful when resident on a given node. The nodes perform pattern matching continuously. Each node has access to a pattern library, much like the immune library in Forrest, et. al., [3]. Random samples from the pattern library are taken at each node. The resident memory cells in the input queue on each respective node are compared against a selected bit string. If a match is found, a new sample is taken and the process is repeated. If no match is found, the resident memory cells are used as the initial population for genetic search.

In this system, autonomous mobile agents circulate the memory cells. From the perspective of each node, agents continuously arrive, deposit memory cells, retrieve new memory cells and transport them to new nodes (Figure 1). Meanwhile, bit strings are continuously sampled from the pattern library. At the instant that a sample is taken, the first 50 memory cells that reside in the local input queue are removed, comprising the initial population for genetic search. If the initial population is less than 50, the memory cells that reside in the local output queue are removed and added to the initial population. Finally, random bit strings are generated to satisfy any remaining discrepancy. With a complete initial population, genetic search begins for the string to match the sampled pattern.

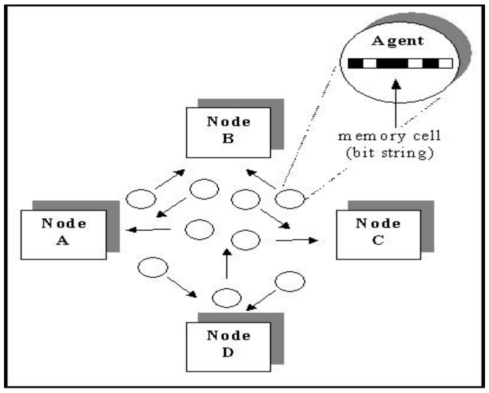

Figure 1: Agent transportation of memory cells.

2.2 Search Algorithm

The CHC adaptive search algorithm [1] is a generational genetic algorithm that has been shown to yield very good performance in optimizing a wide variety of test problems with little or no parameter tuning [6, 7]. Therefore, CHC is used as the genetic search component of our DMS. CHC begins the search process by generating $2 * N$ random samples and selecting the best N samples, where N is the population size.[2] Each of the individuals in the population is then randomly paired to form potential mating pairs. The Hamming distance between the potential mates is measured and compared with an incest threshold. If the Hamming distance is greater than the incest threshold the individuals are allowed to mate. When the mating process is completed for the $N/2$ pairs, the offspring produced compete for survival with the parent population. The best N individuals survive.

Mating in CHC is typically performed using the HUX crossover operator when a binary representation is employed. HUX is a highly disruptive crossover operator which guarantees that the offspring produced will be maximally distant (in Hamming space) from the parents. Crossover is performed by exchanging exactly half of the bits that differ between the two parents.

The incest threshold value is initially set to the expected difference between samples in the population (i.e., $L/2$, where L is the string length). The incest threshold is adaptively adjusted as search progresses. Each generation that either: a) no offspring survive or b) no matings are allowed, the incest threshold is reduced. When the incest threshold goes below 0, cataclysmic mutation is used to diverge the population. Divergence is accomplished by making $2 * N$ copies of the best individual in the population. Then 35% of the bits in all but one individual are complemented and search is restarted.

2.3 Parameters

In this DMS there are several operational parameters that can be tuned to: 1) optimize the learning curve (i.e., minimize the time to recognize the sampled patterns) and 2) minimize the memory footprint (i.e., the number of memory cells in the system). There are four parameters (*feedback percentage*, *feedback decrement*, *direct decrement* and *survival threshold*) in this DMS that directly impact memory cell survival. There are two additional parameters (*number of agents* and *agent wait time*) that effect the circulation and overall distribution of memory cells in the system.

To bound the size of the memory cell population in this system we have introduced the notion of lifetime. Memory cell lifetimes are associated with time-to-live (TTL) values. All memory cells receive the same initial TTL value when they are fed into the system. To survive, a memory cell must maintain a TTL that is greater than the survival threshold. Each time a cell

[2]The initial population in this DMS is seeded rather than randomly generated, as explained in Section 2.1.

is moved from the input to the output queue on a given node, its TTL is decremented. Figure 2 provides an overview of the operations and parameters at each node in the system. The six operational parameter descriptions [and ranges] are as follows:

1. **Number of Agents** [1 - (100*numOfNodes)] - the number of agents that exist in the system.
2. **Agent Wait Time** [100 ms - 2 sec] - the length of time that an agent waits at a node for a memory cell to become available before moving to a new node "empty handed".
3. **Feedback Percentage** [2% - 100%] - the percentage of patterns fed to the survival test from the final genetic population. String patterns are taken in order of best score to worst.
4. **Feedback Decrement** [0 - 100] - decrement applied to a memory cell's TTL when fed to the survival test from the final genetic population.
5. **Direct Decrement** [0 - 100] - decrement applied to a memory cell's TTL when fed directly from the input queue to the output queue.
6. **Survival Threshold** [1 - 100] - minimum TTL value for a memory cell's continued survival.

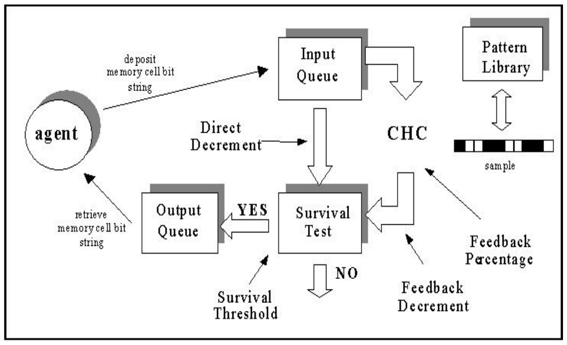

Figure 2: DMS operations and parameters.

In this DMS, the TTL parameter governs the persistence of memory cells. Short-term memory (STM) is modeled by unconditionally decrementing the TTL of memory cells each time they are used. Each cell is treated the same, regardless of its value to the system. Long-term memory (LTM) is modeled by conditional handling of the TTL value, rewarding "useful" memory cells. Upon arrival at a host, each memory cell is scored against the current pattern sample. The TTL of the memory cell is reset if its score is greater than its current TTL. Thus, in the LTM model, survival depends on possessing a high affinity (close in Hamming space) for one or more members of the pattern library.

In evaluating the STM and LTM models, we examine memory efficiency, defined in terms of: 1) minimizing the average trials to match the sample patterns (system learning curve), and 2) minimizing the memory cell count at the end of the simulation (memory footprint). Ideally, the DMS should quickly reduce the number of trials required to match sampled patterns, while limiting the growth of memory cells (Figure 3).

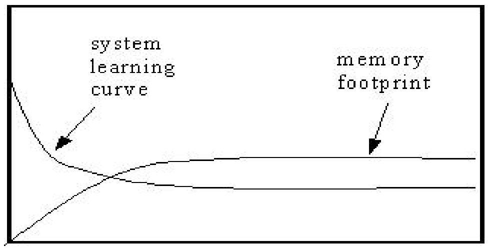

Figure 3: Model DMS behavior.

The performance of this DMS was known to depend on the values assigned to the six parameters governing the system behaviors. To fairly compare the two memory models in this DMS, we used a meta-GA to search for the best operational parameter sets for each model (Section 3). The respective parameter sets are then employed to compare the STM and LTM models. Section 4 describes the DMS simulations and provides an analysis of simulation results. Analysis of the results leads to a question of long-term stability (Section 5), where the meta-GA is again employed to search for a new DMS parameter set using new evaluation conditions. Section 6 provides insight into the effects of the STM and LTM models on genetic search, with an emphasis on the role of random genetic material.

3 Optimizing DMS Parameters Using a Meta-GA

Each evaluation for the search simulates the DMS using the operational parameters as specified by the meta-GA genes (Figure 4). The DMS is simulated for a fixed number of cycles on all nodes, where each cycle constitutes the search for a sampled pattern. Due to the stochastic nature of genetic search and the non-deterministic behavior in a multi-threaded environment, a complete DMS simulation is executed three times for each evaluation, reporting the average as the evaluation value. The evaluation function used to minimize the system learning curve and memory size is:

$$f(x) = \left[\sum_{i=0}^{i=3} (1.5 * MemCells) + avgTrials \right] /3 \quad (1)$$

The STM and LTM meta-GA searches were performed using a pattern library with 10, 64-bit strings. Two

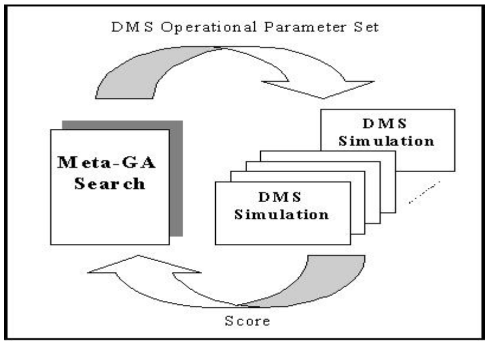

Figure 4: Meta-GA search for DMS parameter sets.

DMS nodes were concurrently simulated for 100 cycles each. For the STM model, the memory cell TTL was never reset. For the LTM model, the memory cell TTL was reset if the memory cell pattern scored higher against the current sample. The *initial* TTL was set to 100 for the simulations of both models.

3.1 Meta-GA Search Results

Table 1 shows the operational parameter sets discovered by the meta-GA searches for the two memory models. Each search was run for 8,000 trials (i.e., 24,000 simulations).[3] Both meta-GA searches discovered that a feedback rate of 2% was best for each of the two memory models. This results in a single individual being fed back to the memory cell population for each completed pattern matching cycle. The operational parameter set discovered for the LTM model specifies more agents to transport information in the DMS than does the parameter set discovered for the STM model. More agents may be required in the LTM simulations, since the agent wait time is considerably longer than the wait time specified for the STM model.

The feedback decrement values discovered for the two DMSs are similar and the direct decrement is exactly the same. However, the values discovered for the survival threshold in the two searches were dramatically different. The meta-GA search discovered a survival threshold of 50 for the DMS using a LTM model and a 12 for the STM model. The memory cells in the LTM model must be useful to the system (i.e., match a pattern sample) frequently to remain alive. A survival threshold of 50 quickly eliminates memory cells with random strings and, when combined with a direct decrement value of 3, will also eliminate memory cells with patterns that are not sampled for more than 16 cycles. The lifetime of a memory cell in the DMS using the STM model is not determined by its value to the system. Thus, a survival threshold of 12, when combined with a direct decrement of 3, allows STM cells to be exploited for ∼29 cycles.

[3]Due to the evaluation time required for each simulation, only a single meta-GA search was performed for each of the memory models proposed. Therefore, parameter sensitivity tests (i.e., varying one parameter at a time while holding the others constant) were employed to evaluate the solutions. No better solutions were discovered, indicating that the operational parameter sets at least represent local optima.

	Memory Model	
DMS Parameter	**STM**	**LTM**
Number of Agents	1	9
Agent Wait Time	338 ms	1170 ms
Feedback Rate	2%	2%
Feedback Decr.	1	0
Direct Decr.	3	3
Survival Threshold	12	50

Table 1: Parameter sets discovered by meta-GA search. Each simulation was run for 100 cycles.

4 Comparing DMS Performance for STM and LTM Models

As mentioned previously, the behavior of this DMS is stochastic in nature. Therefore, the performance of the DMS using the two memory models were compared by executing 30 independent, 100-cycle simulations, using the operational parameter sets found by the meta-GA and listed in Table 1. The results of the 30 independent simulations enabled statistical comparisons. For each simulation we measured:

1. **Final Memory Cell Count** - the total number of memory cells in the system at completion.
2. **Average Trials** - the average number of trials required to match the pattern library samples.
3. **Evaluation Value** - as given in Equation 1.
4. **Hit Rate** - the percentage of cycles where the sampled pattern was matched in the initial population (i.e., at least one memory cell used to seed the initial population matched the sample, eliminating the need for genetic search).
5. **Pattern Match Rate** - the percentage of memory cells in the system at completion that match one of the pattern library samples.
6. **Restarts/Simulation** - the number of cycles/simulation (pattern library samples) where the GA experienced at least one restart (see Section 2.2) while searching for a sampled pattern.

Table 2 gives the average values and standard error of the mean (SEM) for each of these six performance measurements. Not surprisingly, the DMS learns to recognize patterns much faster when starting with populations seeded with memory cells than when starting from random populations (i.e., no memory).

Metric	No Memory	STM	LTM
Cell count	0	87.27 (0.69)	120.33 (1.79)
Avg trials	900.78 (7.24)	177.81 (2.24)	230.22 (3.06)
Hit rate(%)	0	84.68 (0.25)	79.83 (0.27)
Match rate(%)	N/A	69.87 (0.55)	89.35 (1.29)
Restarts/Sim	0	0.06 (0.04)	2.16 (0.35)

Table 2: Average performance values and SEM (for 30 runs) using the parameter sets found by the meta-GA (Table 1). Each run equals a 100-cycle simulation.

4.1 The Value of Short-term Memory

Performance, as measured by the evaluation value, memory cell count, and average trials, is significantly better for the DMS implementing the STM model as opposed to the LTM model. This might be unexpected since traditional memory models tend to reinforce useful memory recall events (i.e., reset TTL) and delete memory cells that have not been useful for long periods of time. However, the direct feedback that allows this DMS to learn also replenishes memory cells that are of value. For example, if a memory cell contains a string that matches the pattern sampled at a given cycle, and that memory cell is used to seed the initial population for the search, the string will be duplicated in the feedback process. This propagates useful information in the DMS without resetting the TTL.

The design of this DMS does not provide for dynamically discontinuing feedback (in which case, resetting the TTL would be critical). In future work, this DMS could respond to a dynamic pattern library, making continuous feedback critical. The emergence of a dynamically maintained distribution of patterns (represented by the memory cell population) that promotes recall at all nodes in the DMS is important.

More insight into these memory models can be gained by examining the behavior of individual DMS simulations. Figure 5-a shows the trials to match each pattern sampled during a DMS simulation of 100 cycles for the STM model, as well as, the memory cell count. Figure 5-b shows the same information for a DMS simulation using the LTM model. Cycles are shown on the X-axis. The solid line shows the number of trials to discover the pattern (Y-axis on left). The dashed line shows the memory cell count (Y-axis on the right).[4]

For both simulations, the searches usually expend ∼900 trials to discover the pattern to match the sample for the first ∼15 cycles. After this learning period, the system often contains a match for the sampled pattern in the seeded initial population. Cycles where the pattern is matched in the initial population (i.e., a hit) require only 50 trials.[5] The peaks in trials indicate genetic search was required to discover the pattern (i.e., no match was found in the initial population).

Matching a sampled pattern in the initial population has a significant impact on the average trials to find a pattern. Thus, the higher average hit rate observed for the STM DMS in Table 2 results in significantly better performance than in the LTM DMS implementation.

Figure 5-a also shows that the memory cell count grows for a period of ∼40 cycles and reaches a high of ∼100 memory cells. However, in the DMS using the LTM model (Figure 5-b) the memory cell count only grows for about half as long and peaks out at a much lower number (∼70). The shorter initial growth period and smaller memory cell count peak exhibited by the DMS using the LTM model would seem to be advantageous, but actually results in worse performance when compared with the DMS using the STM model. The memory cell count in the DMS using the STM model appears to become stable after a large number of cells expire (∼40 cycles). The DMS using the LTM model never seems to become stable and the memory cell count appears to be growing somewhat toward the end of the simulation (i.e., 100 cycles).

5 Stability Analysis of the LTM Model

The trend exhibited by the DMS using the LTM model (Figure 5-b) indicates that the memory cell count would likely continue to grow in longer simulations. To test this hypothesis, DMS simulations for the two memory models were run for 300 cycles, using the same operational parameter sets (see Table 1) used to produce Figures 5-a and 5-b. Figure 5-c shows the trials to match a sample pattern at each cycle for the DMS using the STM model over 300 cycles. Not only does the memory cell count remain stable (i.e., no growth) past the initial 100 cycles, but the peaks, indicating where genetic search is needed to discover the sampled pattern, are sparse after the first ∼30 cycles. The 300-cycle simulation for the STM model is consistent with the behavior observed in 100-cycle simulations.

[4]The information on these graphs reflects the experience at a single node in the DMS system.

[5]The entire initial population is tested regardless of which of the initial patterns match.

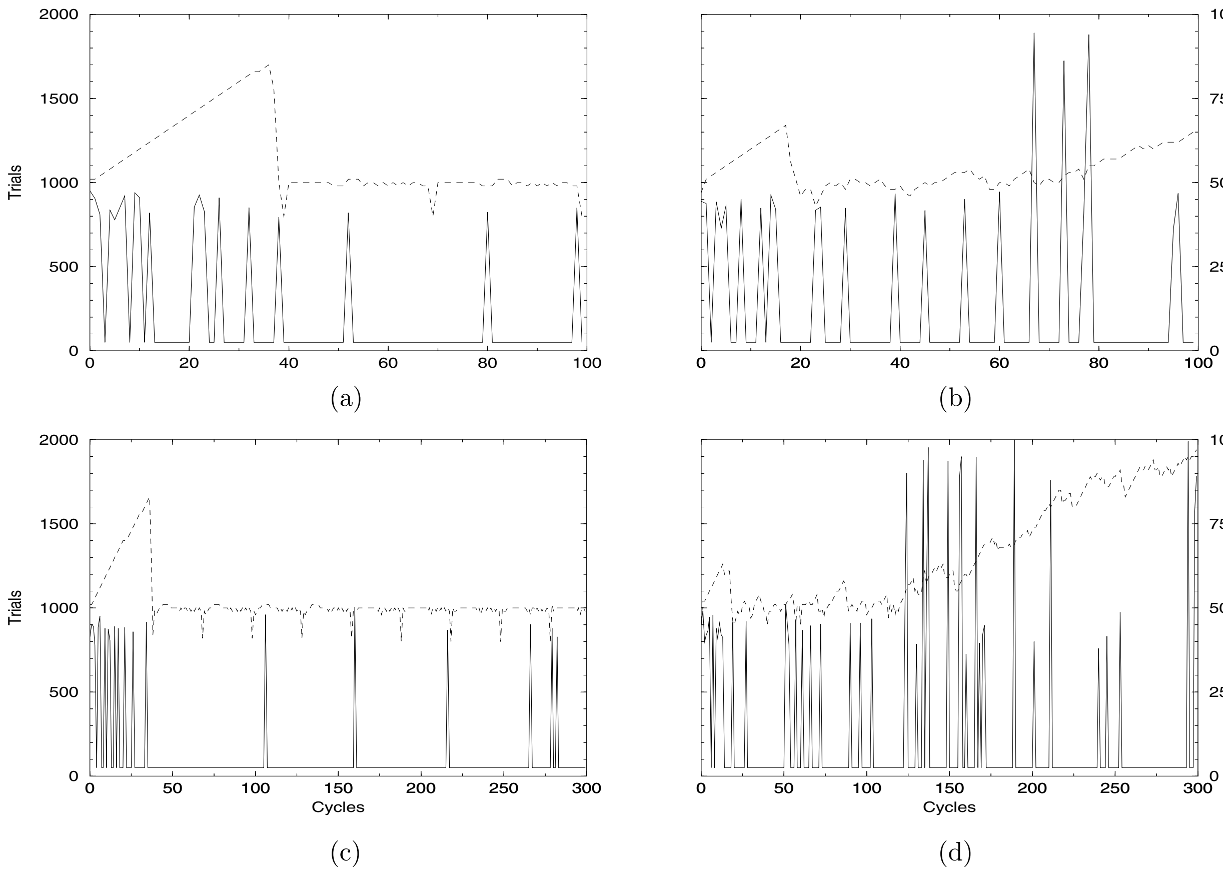

Figure 5: Trials to match sampled patterns and memory cell count at a node for a DMS simulation.

Figure 5-d shows that the memory cell count continues to grow after 100 cycles for the DMS using the LTM model, as predicted. Also, the frequency of peaks (indicating that genetic search was performed) is considerably greater than observed for the STM model simulation (Figure 5-c). This behavior indicates that the operational parameters found by the meta-GA for the DMS using the LTM model do not perform well past 100 cycles (i.e., termination point of simulations during the meta-GA search evaluations).

DMS Parameter	Memory Model		
	STM-100	**LTM-100**	**LTM-300**
Number of Agents	1	9	5
Agent Wait Time	338 ms	1170 ms	1880 ms
Feedback Rate	2%	2%	2%
Feedback Decr.	1	0	0
Direct Decr.	3	3	6
Survival Threshold	12	50	41

Table 3: Parameter sets discovered by meta-GA. The STM-100 and LTM-100 parameters sets result from the 100 cycles/simulation (Table 1) and the LTM-300 parameters result from 300 cycles/simulation.

5.1 Meta-GA Revisited

To determine if a parameter set resulting in more stable behavior could be discovered for the DMS using the LTM model, a third meta-GA search was performed using additional cycles/simulation during evaluation. The third search was performed using the same parameters, conditions and evaluation function described in Section 3, except that each simulation was run for 300 cycles instead of the original 100 cycles.

Table 3 lists the best parameters discovered after 8,000 trials for the 300 cycles/simulation evaluations (LTM-300). Table 3 also lists the parameter sets discovered for the STM and LTM models using 100 cycles/simulation during evaluation (from Table 1). The parameter sets from Table 1 are referred to as STM-100 and LTM-100 to indicate the number of cycles/simulation used during meta-search evaluations.

The LTM-300 operational parameter values discovered by the meta-GA differed in several respects from the LTM-100 parameter set. Perhaps the most critical difference is in the parameters that affect memory cell survival. The LTM-100 parameter set included a survival threshold of 50 and a direct decrement of 3. Thus, memory cells must be useful at least every ∼16 cycles to survive. In contrast, the LTM-300 parameter set has a survival threshold of 41 and direct decrement of 6. In this case, memory cells will only be tolerated for ∼10 cycles if they are not useful. This suggests that a smaller population of memory cells will be maintained.

To compare the performance of the DMS using the LTM-300 parameter set to that of the DMS using the STM-100 and LTM-100 parameter sets, 30 independent simulations were executed using each parameter set and the corresponding memory model. All simulations were run to 300 cycles (regardless of the cycles/simulation used during meta-search). Table 4 shows the average values and SEM for the final memory cell count, average trials to match a pattern, average evaluation value, hit rate, final match rate, and average restarts/simulation for the 30 independent runs of each of the three parameter sets.

The final memory cell count, average trials to match a pattern, evaluation value, and hit rate performance values are significantly better for the DMS simulations using the STM-100 parameter set than for the DMS simulation using either the LTM-100 or LTM-300 parameter set. The LTM-300 parameter set results in only marginally better performance than the LTM-100 parameter set, despite the extra 200 cycles/simulation considered by the meta-GA. The inability to discover a parameter set resulting in better performance for the LTM model, regardless of the number of cycles/simulation, indicates the superiority of the STM model for this DMS.

Figure 6 shows the trials to find pattern matches and the memory count using the LTM-300 parameter set. The memory cell count still fluctuates after 250 cycles. However, the smaller value for the survival threshold used in the LTM-300 parameter set than the value in the LTM-100 set does in fact result in a smaller memory cell count. The average final memory cell count is significantly smaller when the LTM-300 parameter set is used rather than the LTM-100 set (Table 4).

Metric	STM-100	LTM-100	LTM-300
Mem cells	100.50 (0.16)	190.70 (7.90)	128.03 (6.67)
Avg trials	117.08 (1.04)	205.56 (5.99)	276.20 (6.81)
Eval value	267.83 (1.08)	491.61 (12.95)	468.24 (15.12)
Hit rate	91.98 (0.10)	86.06 (0.38)	76.95 (0.44)
Match rate	60.50 (0.12)	92.08 (0.85)	69.44 (2.82)
Restarts	0.27 (0.10)	26.40 (2.56)	22.3 (4.70)

Table 4: Averages for 30 independent runs using the STM-100, LTM-100 and LTM-300 parameter sets. All simulations are run to 300 cycles.

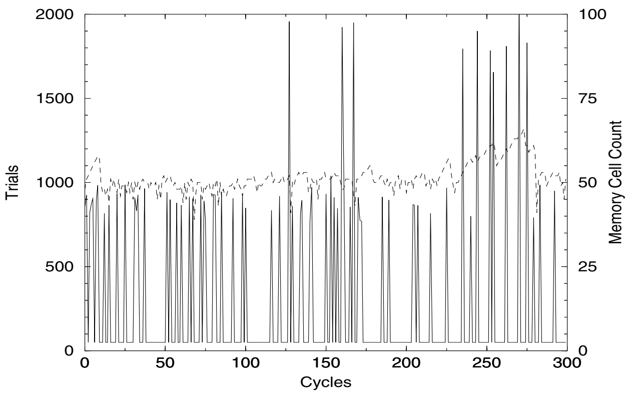

Figure 6: Trials to match patterns and memory cell count at a node for a 300 cycle DMS simulation using the LTM-300 parameter set (Table 3).

While the average final memory cell count using the LTM-300 parameters is smaller than that using the LTM-100 parameters, the average number of trials to find a matching string is significantly worse when using the LTM-300 parameter set. As previously noted, the hit rate is the most important factor in lowering the average trials required to match a sample pattern. The hit rate when using the LTM-100 parameter set is significantly higher than for the LTM-300 parameter sets. In addition, the hit rate for both of the LTM models is significantly lower than when using the STM-100 parameter set.

It is also surprising that the match rate of the final memory cell population does not correlate better with the hit rate. In fact, these two measures are inversely correlated (i.e., a high hit rate yields a lower match rate). This is due to a non-uniformly distributed memory cell population, where most of the cells match only a small fraction of the pattern library.

6 Hit Or Miss: The Role of Random Genetic Material

A side effect of the high match rate exhibited by the DMS using the LTM models (Table 4) is the occurrence of cycles where pattern samples are difficult to match. This is reflected in the large number of trials (>1200) to match a pattern (Figures 5-c, 5-d, and 6). In these instances the trials required to match a pattern are 2 to 3 times as many as when the GA uses a random initial population (i.e., no memory, Table 2). The average number of restarts/simulation for each parameter set tested is listed in Tables 2 and 4.

These events represent a restart by CHC while searching for the pattern. These restarts occur when CHC has converged on a solution that does not match the sampled pattern. This occurs when the initial population is seeded from memory cells where at least one allele in all of the seed strings disagrees with the allele at the corresponding locus of the sampled pattern. For example, if all of the seed strings have a 0 in locus 3 and the pattern sampled has a 1 in that position, the initial population will not contain the genetic material to solve the problem. CHC does not use mutation except during restarts, so the search will converge to a solution that does not match the sampled pattern. A restart will be required to find the pattern [2].

This type of "biased" seeding occurs when a large percentage of the memory cells contain strings matching patterns in the library and very few contain random strings. This condition is a result of the considerable impact of the hit rate on fitness. Memory cells containing random strings do not survive long in the DMS when the LTM model is used while memory cells with strings that match library patterns rapidly increase their representation as seen by the final match rate metric in Tables 2 and 4. In contrast, the STM model simulations contain a higher percentage of random strings in the memory cell population (i.e., significantly lower match rate). Including this random material in the initial population helps provide genetic diversity in all loci and avoids restarts more reliably.

7 Conclusions

This investigation provides an analysis of two memory models in the context of a distributed memory system. We have examined the performance impact of knowledge preservation and exploitation with respect to genetic search using CHC.

The LTM model was designed to promote and preserve high quality information, with the expectation that seeding the genetic search with this material would yield the best results. Although this model performed as expected (i.e., a high concentration of quality material is maintained), it was surprising to find that the DMS using the STM model performed significantly better with respect to average trials and memory cell count. This is due to the fact that on occasion, the LTM model causes the genetic search to be initialized with material that is not representative of the problem set (i.e., a non-uniform distribution). In these instances, the initial population is comprised of many "good" seeds, with respect to alternate pattern library samples, but not the current sample. Hence, an initial match is not available and the genetic search must discover the pattern. Even worse, there is a higher probability of a restart event, which results in performance that is significantly worse than starting with a random initial population (i.e., no memory).

The random information kept by some memory cells in the STM model actually mitigates the potential of seeding the initial population with an incorrect bias. Essentially, there exists a memory quality boundary, where highly concentrated (yet unevenly sampled) knowledge can penalize performance. This is evident from the significantly higher hit rate and better performance exhibited in simulations using the STM model. Given this behavior, we can conclude that the constant feedback mechanism in the STM model, equivalent to short term memory with reproduction, is the better of the two memory models tested for this DMS.

This research demonstrates that the exploitation of domain knowledge, or memory in this instance, can significantly expedite search performance. The STM model improved genetic search performance by a factor of five over random initialization (i.e., no memory), with respect to the number of trials needed to match a pattern. The LTM model, although performing considerably worse than STM, nevertheless demonstrated a four-fold improvement in performance over genetic search with random initialization.

References

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

[2] Larry Eshelman. Personal Communication, 2001.

[3] 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.

[4] Sushil J. Louis and Li Gong. Augmenting Genetic Algorithms with Memory to Solve Traveling Salesman Problems. In *Joint Conference on Information Sciences*, pages 109–111. Duke University, 1997.

[5] Sushil J. Louis and Fang Zhao. Domain Knowledge for Genetic Algorithms. *International Journal of Expert Systems*, 8(3):195–212, 1995.

[6] Keith E. Mathias and L. Darrell Whitley. Changing Representations During Search: A Comparative Study of Delta Coding. *Journal of Evolutionary Computation*, 2(3):249–278, 1994.

[7] Darrell Whitley, Keith Mathias, Soraya Rana, and John Dzubera. Building Better Test Functions. In L. Eshelman, editor, *ICGA-6*. Morgan Kaufmann, 1995.