Using Lazy Evaluation to Simulate Realistic-Size Repertoires in Models of the Immune System[1]

Derek J. Smith
Department of Computer Science
University of New Mexico
Albuquerque, NM 87131 USA
dsmith@cs.unm.edu

Stephanie Forrest
Department of Computer Science
University of New Mexico
Albuquerque, NM 87131 USA
forrest@cs.unm.edu

David H. Ackley
Department of Computer Science
University of New Mexico
Albuquerque, NM 87131 USA
ackley@cs.unm.edu

Alan S. Perelson
Theoretical Division
Los Alamos National Laboratory
Los Alamos, NM 87545 USA
asp@lanl.gov

Abstract

We describe a method of implementing efficient computer simulations of immune systems that have a large number of unique B and/or T cell clones. The method uses an implementation technique called *lazy evaluation* to create the illusion that all clones are being simulated, while only actually simulating a much smaller number of clones that can respond to the antigens in the simulation. The method is effective because only 0.001% to 0.01% of clones can typically be stimulated by an antigen, and because many simulations involve only a small number of distinct antigens. A lazy simulation of a realistic number of clones and 10 distinct antigens is 1,000 times faster and 10,000 times smaller than a conventional simulation—making simulations of immune systems with realistic-size repertoires computationally tractable.

Keywords: lazy evaluation, simulation, immune system, cross-reactive memory

1 Introduction

The B and T cell repertoires of vertebrate immune systems can recognize and respond to almost all foreign antigens, even laboratory derived ones that almost surely have never been seen in evolutionary history. The repertoire can also distinguish, at a fine level of detail, between foreign antigens and the components of the body it protects. To achieve this broad yet detailed coverage, the immune system maintains a large number of highly specific clones, where a clone is a set of cells derived from a single precursor and which almost assuredly have a unique B or T cell receptor. In this paper we discuss only the B cell repertoire; however, the method is also applicable to the T cell repertoire.

[1]Bulletin of Mathematical Biology (in press).

The murine B cell repertoire maintains 10^7 to 10^8 distinct clones (Köhler, 1976; Klinman *et al.*, 1976; Klinman *et al.*, 1977), each of which typically can be stimulated by only 10^{-5} to 10^{-4} of all possible antigens (Edelman, 1974; Nossal & Ada, 1971; Jerne, 1974). In order for an antigen to stimulate a B cell it must bind to antigen-specific receptors on the surface of the B cell.

The binding affinity between receptors and antigens is based on complementarity at the molecular level. Perelson & Oster (1979) introduced an abstract model of binding in which molecules are considered as points in a "shape space" and affinity is measured as a function of the distance between such points. Modelers have used a variety of methods to represent molecules in shape space. Segel & Perelson (1988) and DeBoer *et al.* (1992) examined one and two dimensional shape spaces in which the shape of molecules was represented by one or two real numbers, e.g. the depth or depth and width of a binding cleft or protrusion on a molecule. Affinity was then measured as a function of the Euclidean distance[2] between the shapes. Seiden & Celada (1992), Forrest & Perelson (1991), and Perelson *et al.* (1996) (after Farmer *et al.* (1986)), represented molecules as strings of 8, 32, and 64 bits respectively and measured affinity as a function of the Hamming distance[3] (or a variation on it) between them. Weisbuch & Oprea (1994) and Detours *et al.* (1996) represented molecules as strings of digits chosen from a 4 and 16 letter alphabet respectively. Smith *et al.* (1997b) determined that representing molecules as strings of 20 symbols, with each symbol chosen from a 4 letter alphabet, and affinity measured as a function of Hamming distance, as well as using a realistic-size repertoire of 10^7 B cell clones, gave good fits to immunological data important for a model of cross-reactive memory.

To make simulations of 10^7 clones computationally tractable, we use a technique called *lazy evaluation* (Friedman & Wise, 1976; Henderson & Morris, 1976). This technique (as illustrated in the next section) delays calculations, and the building of data structures, until they are needed. When not all calculations and data structures affect the result of a program, and when the relevant ones can be identified efficiently, lazy evaluation can result in significant savings in run time and memory usage. In the case of the immune system, lazy evaluation can be effective because only 0.001% to 0.01% of all clones are usually stimulated by any particular antigen, and because many simulations involve only a small number of distinct antigens.

Lazy evaluation can be programmed explicitly in traditional programming languages, or implicitly by using languages in which all evaluations are performed lazily (Turner, 1979; Turner, 1985; Hudak *et al.*, 1992). Lazy evaluation has been applied in numerous domains including: animation (Elliott & Hudak, 1997), simulation of integrated circuits (Dunne *et al.*, 1993) (and a production simulator based on Yoshino *et al.* (1987)), sound synthesis (Dannenberg *et al.*, 1992), and dictionary lookup (Lucas, 1995). In this paper we describe how lazy evaluation can be programmed explicitly in models of the immune system.

[2]Euclidean distance is the familiar square root of the sum of the squares of the differences in each dimension. The Euclidean distance between receptors $a_1, a_2 \ldots a_n$ and $b_1, b_2 \ldots b_n$, is $\sqrt{\sum_{1 \leq i \leq n} (a_i - b_i)^2}$

[3]Hamming distance is a count of the number of locations in which the receptors differ. The Hamming distance between the receptors AB$\underline{\text{D}}$CCDAD$\underline{\text{D}}$A and AB$\underline{\text{A}}$CCDAD$\underline{\text{C}}$A is 2 because they differ in the two underlined locations.

2 Algorithm

In a conventional *eager* approach to immune system simulation, computation time is taken and memory space explicitly allocated to generate all clones at the start of the simulation (Figure 1a). When an antigen is introduced, the clones that can be stimulated by it (said to be within its *ball of stimulation* (Perelson & Oster, 1979)), already exist and the simulation proceeds (Figure 1b). In the modified *lazy* simulation, no clones are generated at the start of the simulation (Figure 2a). Instead, when an antigen is introduced, the simulation is suspended while clones within the ball of stimulation of the antigen are generated (Figure 2b). In this way, all clones that could be stimulated by the antigen appear and act as in an eager simulation. The absence of the remaining clones has no effect on the simulation other than making it run faster and take less memory. Clones must not be added to regions of a ball of stimulation where they have already been created by the introduction of previous antigens—this would result in too many clones in the intersections of balls of stimulation (Figure 2c).

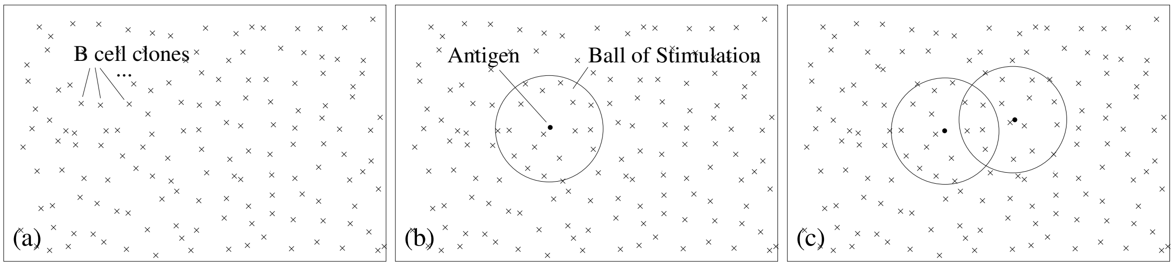

Figure 1: (a) In an eager simulation, all clones ($\times$) are generated at the start of the simulation. (b) & (c) When antigens ($\bullet$) are introduced, clones already exist and no new ones need to be generated.

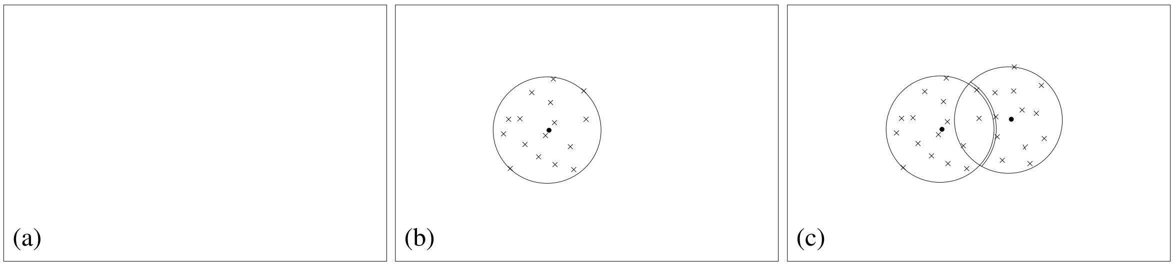

Figure 2: (a) In a lazy simulation, no clones are generated at the start of the simulation. (b) When an antigen ($\bullet$) is introduced, the simulation is temporarily halted while clones ($\times$) within its ball of stimulation are created. (c) When the ball of stimulation of a new antigen intersects that of an existing antigen, no additional clones need to be generated in the intersection as it has already been adequately populated.

For a lazy simulation to be functionally equivalent to an eager simulation, clones generated within a ball of stimulation must be added in the same distribution they would have had in an eager simulation. The correct distribution depends on how receptors and antigens are represented, and how

When a new antigen is added to the simulation

```
loop for i from 0 to r do
     loop for j from 1 to num-clones(i) do
          let clone = mutate(center, i)
              if {clone is outside the ball of stimulation
                        of all previously added antigens}
                 then {add clone to the simulation}
```

where `r` is the radius of a ball of stimulation;

where `num-clones(i)` produces a random number from the binomial distribution $B(n, p_i)$, where n is the number of clones in an eager simulation, and p_i is the probability that a clone is a radius i from an antigen;

and where `mutate(center, i)` mutates `i` distinct locations of the string representing the center of the ball of stimulation.

Figure 3: Pseudo-code describing the lazy algorithm for generating clones.

affinity between them is measured. Here we describe a lazy algorithm for receptors represented as strings of symbols, and affinity measured as a function of the Hamming distance between receptors. In an eager simulation using this representation, receptors are generated by choosing each symbol from a uniform distribution—loosely mimicking the random genetic process used by vertebrate immune systems to generate clonal diversity (Leder, 1991).

For the lazy simulation, clones must be generated only within balls of stimulation. To do this we develop a method to generate clones at radius i from the center of a ball of stimulation, and then repeat the method at radii 0 through r, where r is the radius of a ball of stimulation. The probability, p_i, that a randomly selected clone in an eager simulation is radius i from the center of a ball is given by $p_i = \binom{d}{i}\left\{\frac{(k-1)}{k}\right\}^i\left\{\frac{1}{k}\right\}^{(d-i)}$, where d is the number of symbols in the string representation of the receptor, and k is the number of possible symbols at each location in the string. Further, in an eager simulation with n clones, the probability of j clones at radius i is given by the binomial $B(n, p_i) = \binom{n}{j}\{p_i\}^j\{1-p_i\}^{n-j}$. Thus, the number of clones, $\hat{j}$, to generate at radius i from the center of a ball should be sampled from this binomial distribution. Each of the $\hat{j}$ clones is generated by changing i distinct symbols in the string that represents the receptor at the center of the ball of stimulation.

To avoid multiply generating clones in the intersections of balls of stimulation, each new clone is added to the lazy repertoire only if it is outside the balls of stimulation of all antigens already in the simulation. In the next two sections we verify that the lazy algorithm generates clones in the same distributions as the eager algorithm, and compare the algorithmic costs of the lazy and eager algorithms.

3 Verification

We generated a complex test case to check whether the lazy algorithm generates clones in the same distributions as the eager algorithm, especially in the case of multiply overlapping balls of stimulation. Following Smith *et al.* (1997b), molecules in the test were represented by strings of 20 symbols, each symbol was chosen from a four letter alphabet, and balls of stimulation had radius 5. A *seed* antigen was generated by randomly selecting each of its symbols from a uniform distribution, and 10 test antigens were generated in a cluster around the seed. Each test antigen was generated by mutating m randomly selected unique symbols of the seed, where m was chosen from a uniform distribution in the range zero to three so the balls of stimulation of the test antigens would have intersections of various sizes. Clones were generated, according to the lazy algorithm, and the number of clones at radii 0 through 5 for each antigen were counted. The experiment was replicated 100,000 times. For 10,000 of these experiments, the algorithm was metered to record the balls of stimulation that a newly generated clone fell within. These data were used to determine how much of each ball of stimulation was populated with clones generated by previous antigens.

Table 1 shows that the 10 antigens were at varying Hamming distances from each other and thus had varying overlaps. Table 2 shows that these overlaps resulted in many different proportions of balls of stimulation being populated by clones generated by prior antigens, and were thus a reasonable test of the lazy algorithm. Figure 4 shows that the observed and expected distributions are the same when compared visually, and Table 3 shows they are the same when compared statistically. Thus, the lazy algorithm worked correctly.

4 Algorithmic Cost

In this section we compare the algorithmic cost of the lazy and eager algorithms. The number of clones generated in a lazy simulation is $g \times p \times n$, where g is the number of distinct antigens in the simulation, p is the proportion of the repertoire that can be stimulated by an antigen, and n is the total number of clones in the eager simulation. Each of these clones needs to be checked to see if it falls within the ball of stimulation of any previously added antigen. Thus, the total number of checks after adding g antigens is $pn \sum_{0 \leq j < g} j = \frac{(g(g-1))}{2} pn$. If we assume that the cost of generating a clone is approximately the same as the cost of checking if a clone is in the ball of stimulation of an antigen, then the total cost of generating the lazy repertoire for g antigens is $gpn + \frac{g(g-1)}{2} pn = \frac{g^2+g}{2} pn$. The cost of the eager method is n because it generates n clones and does not have to do any checks. Comparing the cost of the lazy and eager algorithms, the lazy method is more efficient than the eager method when g is less than approximately $\sqrt{\frac{2}{p}}$.

For a realistic-size repertoire with $n = 10^7$ and p=10^{-5}, in a simulation with 10 distinct antigens, the lazy algorithm will create less than 0.01% of the clone repertoire at 0.1% of the cost of the eager algorithm. For less than 447 distinct antigens, and the same realistic-size repertoire, the lazy algorithm is lower cost than the eager algorithm (Figure 5). Simulations of more than 447 antigens, which would probably include simulations of immune networks, would be more efficient using an eager algorithm.

Antigen	Hamming distance to other antigens										Antigen string
	1	2	3	4	5	6	7	8	9	10	
1	0	4	2	1	4	4	2	3	3	2	CCBDDDBCCCABDCCDADAD
2	4	0	4	3	6	6	4	4	5	4	BCBDDDBCCCADDCCDAAAC
3	2	4	0	1	3	4	2	3	3	2	CCBDDDBCCCABACCDADAC
4	1	3	1	0	3	3	1	2	2	1	CCBDDDBCCCABDCCDADAC
5	4	6	3	3	0	6	4	5	5	4	CCCDDDBCCCABCCCDDDAC
6	4	6	4	3	6	0	3	4	5	3	CCBCDDBCCCBBDCCCADAC
7	2	4	2	1	4	3	0	1	3	2	CCBDDDBCCCDBDCCDADAC
8	3	4	3	2	5	4	1	0	4	3	CCBDDDBCCCDBDCCDACAC
9	3	5	3	2	5	5	3	4	0	3	CCBDDCDCCCABDCCDADAC
10	2	4	2	1	4	3	2	3	3	0	CCBBDDBCCCABDCCDADAC

Table 1: The pairwise Hamming distances between the 10 test antigens used in the experimental verification of the algorithm. The table is symmetric about the main diagonal because Hamming distance is commutative. The table shows that the antigens were at various Hamming distances from each other.

Antigen	Proportion of clones generated by each antigen									
	1	2	3	4	5	6	7	8	9	10
1	1.00	-	-	-	-	-	-	-	-	-
2	0.05	0.95	-	-	-	-	-	-	-	-
3	0.21	0.03	0.77	-	-	-	-	-	-	-
4	0.33	0.06	0.17	0.45	-	-	-	-	-	-
5	0.05	0.01	0.06	0.02	0.86	-	-	-	-	-
6	0.05	0.01	0.03	0.05	0.00	0.86	-	-	-	-
7	0.21	0.03	0.09	0.13	0.01	0.02	0.51	-	-	-
8	0.10	0.03	0.05	0.09	0.01	0.02	0.14	0.57	-	-
9	0.01	0.01	0.05	0.10	0.01	0.01	0.02	0.01	0.71	-
10	0.21	0.03	0.09	0.13	0.01	0.02	0.05	0.01	0.02	0.44

Table 2: The balls of stimulation of the 10 test antigens all overlapped each other; thus, many of the clones within a ball of stimulation were generated by the lazy algorithm operating on prior overlapping antigens. The proportions generated by each antigen are shown in this table. For example, for the fourth antigen, on average, 0.33 of the clones in its ball of stimulation were already generated by the first antigen, 0.06 by the second antigen, 0.17 by the third antigen, and 0.45 were generated *de novo* by the lazy algorithm on injection of the fourth antigen. The data were calculated by metering the lazy algorithm to record which balls of stimulation a newly generated clone fell within. The varying proportions suggest that the 10 antigens were a reasonable test of the lazy algorithm.

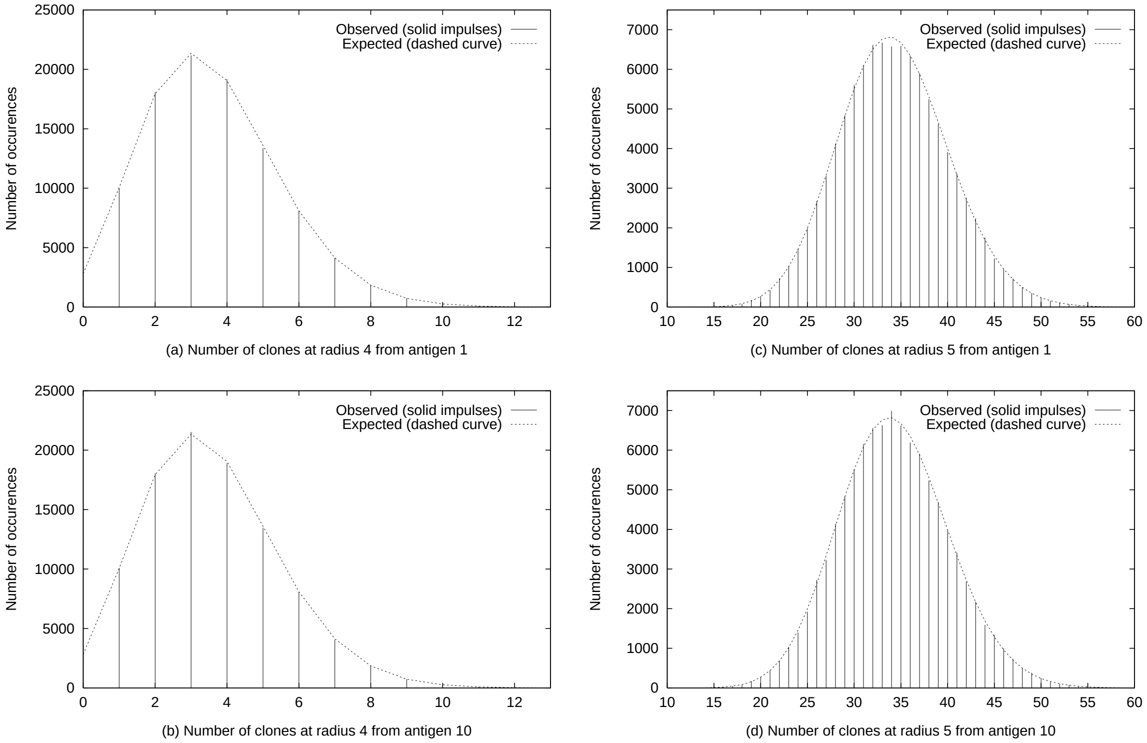

Figure 4: The expected (dashed curve) and observed (solid impulses) distributions of the number of clones at radii 4 and 5 from antigens 1 and 10. These data were collected from the application of the lazy algorithm to the sequential introduction of the 10 test antigens in 100,000 independent simulations and counting the number of clones at each radius within the ball of stimulation of each antigen. Antigens 1 and 10 are shown because they had the least and most number of clones, respectively, generated by prior antigens. Plots showing the distributions for the other antigens, and other radii, showed similar visual correspondence between the observed and expected distributions.

Radius	Observed χ^2 goodness-of-fit values for each antigen at each radius										Degrees	Critical
	1	2	3	4	5	6	7	8	9	10	of freedom	χ^2
1	0.00	0.11	0.00	1.34	0.23	0.04	0.11	0.23	1.02	0.12	1	3.84
2	4.36	1.50	0.89	3.08	1.19	2.61	1.71	4.55	0.48	1.04	2	5.99
3	5.15	3.06	5.40	0.43	0.60	2.14	4.13	4.87	3.20	1.69	4	9.49
4	13.57	10.68	22.36	17.71	6.44	11.24	12.10	7.33	17.46	10.08	12	21.03
5	39.47	44.45	31.72	31.74	45.25	39.95	35.96	48.35	28.22	54.58	40	55.76

Table 3: All observed χ^2 values (except one) were below their respective critical χ^2 value. Thus, there is no evidence (p=0.05) for rejecting the hypothesis that the observed data were in their expected distributions, and we conclude the lazy algorithm worked correctly. The one exception (antigen 3, radius 4) appears to be a Type I error, due to statistical variation, because its χ^2 value was less than the critical value when the experiment was repeated. One such error in 20 tests is to be expected at p=0.05. The data used were from the same 100,000 simulations used to make the plots of Figure 4. Not enough clones were generated at radius 0 (eight total in 100,000 simulations) to perform the test.

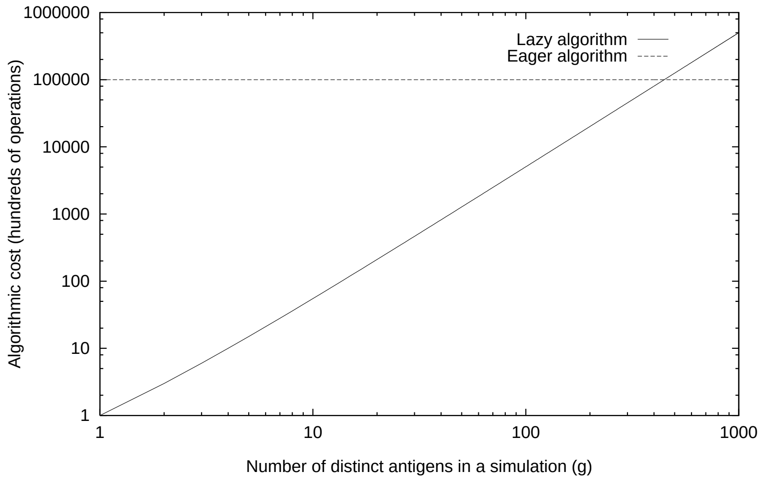

Figure 5: A comparison of the algorithmic cost of creating the B cell repertoire by the lazy and eager algorithms. The algorithmic cost was measured as the sum of the number of clones generated plus the number of checks that a clone was within the ball of stimulation of an antigen. Calculations were done for a realistic-size repertoire with $n = 10^7$ and $p = 10^{-5}$. The lazy algorithm costs less than the eager algorithm when there are less than 447 distinct antigens in a simulation, and costs 1,000 times less when there are less than 10 distinct antigens in the simulation.

When there are more than 10 distinct antigens in a realistic-size simulation, most of the algorithmic cost of the lazy method is in the g^2 comparisons of new clones with previously added antigens. In simulations involving hundreds of antigens, this g^2 cost could be reduced by various methods. One method is only to compare clones against antigens that are within distance $2r$ of the antigen for which clones are being generated. The algorithm still works correctly because antigens at greater than $2r$ distance cannot have intersecting balls of stimulation. In this case, the g^2 comparisons become a worst case, and the actual number of comparisons depends on the distances between the antigens in the simulation. Another method to reduce the g^2 comparisons is to generate clones in a ball larger than a ball of stimulation and thus avoid lazy generation for any future antigens whose balls of stimulation fall completely within the previously generated larger balls. This second method is effective when the antigens are tightly clustered.

5 Discussion

We have described an algorithm that uses lazy evaluation to generate only the subset of clones that can be stimulated by the antigens introduced into a simulation. Because this subset is typically a tiny proportion of the clone repertoire, the algorithm permits the efficient simulation of realistic-size repertoires. Correctness of the algorithm was checked by showing that, in a test case of 10 overlapping antigens, the algorithm produced clones in the same distributions as an eager algorithm. Analysis of the algorithm showed, in simulations of realistic-size repertoires involving less than 10 distinct antigens, that less than 0.01% of the expressed repertoire was created at less than 0.1% of the cost of creating a complete repertoire. We have implemented a lazy simulation of the humoral immune response that uses a realistic-size repertoire with a steady-state size of 10^7 B cell clones and a turnover of 5×10^5 B cell clones every 6 simulated hours (Smith *et al.*, 1997a). Simulations of the sequential infection by three antigens that have overlapping balls of stimulation, over a simulated period of 200 days, takes less than 2 minutes of CPU time, running in Lisp, on a Sun Ultra 2/2300.

The algorithm we described is specific for models in which receptors are represented as strings of symbols and affinity is calculated as a function of the Hamming distance between receptors. The method could also be applied to other representations of receptors and other methods of calculating affinity, by changing the calculation of the probability distribution of clones within a ball of stimulation and the method of generating clones within a ball of stimulation. The method is applicable to both *agent based* models in which each cell is represented individually, and to differential equation based models in which each clone is represented by a differential equation.

Some neural network and associative memory models have similar structure and function to the immune system models discussed above (Smith *et al.*, 1996). Thus, the lazy method can also be applied to implementation of such models. In particular, the method can be applied without modification to the *Sparse Distributed Memory* (SDM) model (Kanerva, 1988). The method can also be applied, with modifications similar to those described for other representations of receptors, to the *Cerebellar Model Arithmetic Computer* (Albus, 1981), the *Theory of Cerebellar Cortex* (Marr, 1969), *WISARD* (Aleksander *et al.*, 1984), and variations on SDM (Jaeckel, 1989a; Jaeckel, 1989b). Danforth (1997) used a lazy-like method, and a modified SDM learning rule (Danforth, 1991), to improve the performance of SDM. Danforth's method adds at most one *hard location* (the SDM equivalent of a clone) on each write to the memory, at the exact location of the write. This is in contrast to the cluster of hard locations that would be added by our method. Both methods significantly reduce the number of hard locations in a simulation (compared to an equivalent eager simulation), and distribute hard locations in accordance with the distribution of addresses used to write to the memory. Danforth's method modifies the behavior of the SDM; our method leaves the behavior unchanged, and only modifies the implementation.

General immune system models that include clones with receptors have been simulated with the order of 10^3 clones (DeBoer & Perelson, 1991; Celada & Seiden, 1996; Detours *et al.*, 1996). Lattice based cellular models that use only one or two bits to represent the concentration of highly simplified clones, and measure affinity by neighborhood on the lattice, have simulated 10^4 clones (DeBoer *et al.*, 1992a), and 10^8 clones (Stauffer & Sahimi, 1994) (the latter on a Cray-YMP). The lazy evaluation method presented here is the first to permit realistic-size repertoires of 10^7 to 10^8 clones for general immune system models.

Acknowledgments

The authors gratefully acknowledge the ongoing support of the Santa Fe Institute and its Joseph P. and Jeanne M. Sullivan program in theoretical immunology. This work was also supported by grants ONR (N00014-95-1-0364), NSF (IRI-9157644), and NIH (RR06555). Portions of this work were performed under the auspices of the U.S. Department of Energy. DJS also acknowledges the University of New Mexico Computer Science Department AI fellowship and Digital Equipment fellowship, and the help of Doug Danforth, Ron Hightower, Peter Hraber, Terry Jones, Pentti Kanerva, Catherine Macken, Ronald Moore, Francesca Shrady, Suzanne Sluizer, Paul Stanford, and Carla Wofsy. This paper was written while Forrest was visiting the MIT Artificial Intelligence Laboratory.

REFERENCES

Albus, J. S. (1981). *Brains, Behavior, and Robotics*. Byte Books, Peterborough, NH.

Aleksander, I., Thomas, W. V. & Bowden, P. A. (1984). Wisard-a radical step forward in image recognition. *Sensor Review* **4**(3), 120–124.

Celada, F. & Seiden, P. E. (1996). Affinity maturation and hypermutation in a simulation of the humoral immune response. *Eur. J. Immunol.* **26**, 1350–1358.

Danforth, D. G. (1991). Total recall in distributed associative memories. Technical Report TR 91.03, Research Institute for Advanced Computer Science, NASA Ames Research Center.

Danforth, D. G., (1997). Personal communication of work performed at the Research Institute for Advanced Computer Science, NASA Ames Research Center.

Dannenberg, R. B., Fraley, C. L. & Velikonja, P. (1992). A functional language for sound synthesis with behavioral abstraction and lazy evaluation. In: *Computer Generated Music* (Baggi, D., ed.). IEEE Computer Society Press.

DeBoer, R. J. & Perelson, A. S. (1991). Size and connectivity as emergent properties of a developing immune network. *J. Theoret. Biol.* **149**, 381–424.

DeBoer, R. J., Hogeweg, P. & Perelson, A. S. (1992). Growth and recruitment in the immune network. In: *Theoretical and Experimental Insights into Immunology* (Perelson, A. S. & Weisbuch, G., eds) pp. 223–247. Springer-Verlag, Berlin.

DeBoer, R. J., Segel, L. A. & Perelson, A. S. (1992). Pattern formation in one and two dimensional shape space models of the immune system. *J. Theoret. Biol.* **155**, 295–333.

Detours, V., Sulzer, B. & Perelson, A. S. (1996). Size and connectivity of the idiotypic network are independent of the discreteness of the affinity distribution. *J. Theoret. Biol.* **183**, 409–416.

Dunne, P. E., J.Gittings, C. J. & Leng, P. H. (1993). Sequential and parallel strategies for the demand-driven simulation of logic circuits. *Microprocessing and microprogramming* **1**, 591–525.

Edelman, G. M. (1974). Origins and mechanisms of specificity in clonal selection. In: *Cellular Selection and Regulation in the Immune System* (Edelman, G. M., ed.) pp. 1–38. Raven Press, New York.

Elliott, C. & Hudak, P. (1997). Functional reactive animation. In: *Proc. ACM SIGPLAN International Conference on Functional Programming (ICFP '97)*.

Farmer, J. D., Packard, N. H. & Perelson, A. S. (1986). The immune system, adaptation, and machine learning. *Physica D* **22**, 187–204.

Forrest, S. & Perelson, A. S. (1991). Genetic algorithms and the immune system. In: *Parallel Problem Solving from Nature* (Schwefel, H. & Maenner, R., eds) pp. 320–325. Springer-Verlag, Berlin.

Friedman, D. P. & Wise, D. (1976). CONS should not evaluate its arguments. In: *Automata, Languages and Programming* (Michaelson, S. & Milner, R., eds) pp. 257–284, Edinburgh. Edinburgh University Press.

Henderson, P. & Morris, J. M. (1976). A lazy evaluator. In: *Proc. 3rd Annual ACM symposium on Principles of Programming Languages* pp. 95–103, New York. ACM.

Hudak, P., Peyton Jones, S. L., Wadler, P. *et al.* (1992). Report on the functional programming language Haskell: A non-strict, purely functional language: Version 1.2. *ACM SIGPLAN Notices* **27**, 1–163.

Jaeckel, L. A. (1989). An alternative deisgn for a sparse distributed memory. Technical Report TR 89.28, Research Institute for Advanced Computer Science, NASA Ames Research Center.

Jaeckel, L. A. (1989). A class of designs for a sparse distributed memory. Technical Report TR 89.30, Research Institute for Advanced Computer Science, NASA Ames Research Center.

Jerne, N. K. (1974). Clonal selection in a lymphocyte network. In: *Cellular Selection and Regulation in the Immune System* (Edelman, G. M., ed.) pp. 39–48. Raven Press, New York.

Kanerva, P. (1988). *Sparse Distributed Memory*. MIT Press, Cambridge, MA.

Klinman, N. R., Press, J. L., Sigal, N. H. & Gerhart, P. J. (1976). The acquisition of the B cell specificity repertoire: the germ-line theory of predetermined permutation of genetic information. In: *The Generation of Antibody Diversity* (Cunningham, A. J., ed.) pp. 127–150. Academic Press, New York.

Klinman, N. R., Sigal, N. H., Metcalf, E. S., Gerhart, P. J. & Pierce, S. K. (1977). *Cold Spring Harbor Symp. Quant. Biol.* **41**, 165.

Köhler, G. (1976). Frequency of precursor cells against the enzyme beta-galactosidase: an estimate of the BALB/c strain antibody repertoire. *Eur. J. Immunol.* **6**, 340–347.

Leder, P. (1991). The genetics of antibody diversity. In: *Immunology: Recognition and Response* (Paul, W., ed.) pp. 20–34. W. H. Freeman, New York.

Lucas, S. M. (1995). Rapid best-first retrieval from massive dictionaries by lazy evaluation of a syntactic neural network. In: *Proc. IEEE Intnl. Conf. on Neural Networks* pp. 2237–42, New York, NY. IEEE.

Marr, D. (1969). A theory of cerebellar cortex. *J. Physiology* **202**, 437–470.

Nossal, C. J. V. & Ada, G. L. (1971). *Antigens, Lymphoid Cells and The Immune Response*. Academic Press, New York.

Perelson, A. S. & Oster, G. F. (1979). Theoretical studies of clonal selection: Minimal antibody repertoire size and reliability of self- non-self discrimination. *J. Theoret. Biol.* **81**, 645–670.

Perelson, A. S., Hightower, R. & Forrest, S. (1996). Evolution and somatic learning in V-region genes. *Res. Immunol.* **147**, 202–208.

Segel, L. A. & Perelson, A. S. (1988). Computations in shape space: A new approach to immune network theory. In: *Theoretical Immunology, Part Two, SFI Studies in the Sciences of Complexity* (Perelson, A. S., ed.) pp. 321–343. Addison-Wesley, Reading, MA.

Seiden, P. E. & Celada, F. (1992). A model for simulating cognate recognition and response in the immune system. *J. Theoret. Biol.* **158**, 329–357.

Smith, D. J., Forrest, S. & Perelson, A. S. (1996). Immunological memory is associative. In: *Workshop Notes, Workshop 4: Immunity Based Systems, Intnl. Conf. on Multiagent Systems* pp. 62–70, Kyoto, Japan.

Smith, D. J., Forrest, S., Ackley, D. H. & Perelson, A. S. (1997a). Modeling the effects of prior infection on vaccine efficacy. In: *IEEE Intnl. Conf. on Systems, Man, and Cybernetics* pp. 363–368, Orlando, Florida. IEEE.

Smith, D. J., Forrest, S., Hightower, R. R. & Perelson, A. S. (1997b). Deriving shape space parameters from immunological data. *J. Theoret. Biol. (in press)* and *Santa Fe Institute Working Paper 97-03-017* .

Stauffer, D. & Sahimi, M. (1994). High-dimensional simulation of simple immunological models. *J. Theoret. Biol.* **166**, 289–297.

Turner, D. A. (1979). A new implementation technique for applicative languages. *Software—Practice and Experience* **9**, 31–49.

Turner, D. A. (1985). Miranda: a non-strict functional language with polymorphic types. In: *Proc. Int'l Conf. on Functional Programming and Computer Architecture, Nancy, Lecture Notes in Computer Science 201* (Jouannaud, J.-P., ed.) pp. 1–16, Berlin. Springer-Verlag.

Weisbuch, G. & Oprea, M. (1994). Capacity of a model immune network. *Bull. Math. Biol.* **56**, 899–921.

Yoshino, T., Smith, D. J. & Matzke, D. J. (1987). An RTL simulator based on functional programming. *Texas Instruments Technical Journal* **4**(3).