Toy Implementations of Artificial Immune System

Akira Imada

last modified on:

June 18, 2004

Abstract

This document is for the purpose of a tutorial in order for us to kick off our new inovative research project. All the algorithms are from the papers I collected via Internet, though I modified the description to make it more clear, having a risk of my making a mistake or misunderstanding. You might find all those papers in this web-site of ours. Bellow you find a set of keywords which are originally defind in the field of biology. in reading those papers here, I recommend you to consider what is the couterpart of these keywords in conputer science/technology at the beginning, that is, what on earth do these keywords mean both in our real body and in an artificial immune system in computer simulations.

Keywords: antigen, antibody, primary/secondory-responce, cross-reactive-responce, epitope, paratope, affinity-maturatin, B-cells, T-helper-cell, Jerne's hypothesis, cronal-selection, somatic-mutatuin, hyper-mutation, VJ-recombination, receptor-editing, repertoir, $\cdots$

1 Introduction

Firstly, I propose a series of test-functions in order to observe how the algorithms work? Some of them are also commonly used ones and the others are my own proposition. The algorithms bellow are paraphrased by me imterpreting by myself despite the risk of misunderstanding. So study both my description and the one in the paper carefully and bug-reports are always welcome, as usual.

2 Real biological clonal selection

What's going on when we are exposed to virses? In order to have a bird's eye view of the idea of applying biological immune system to our computational technique, let's see, for a while, inmmune system in our or animals' body. The followings are what I paraphrased from (de Castoro and von Zuben, 2002) for the purpose. See also the keywords above, and try to give it a consideration on what they actually are?

When we are exposed to an *antigen* (*Ag*), a subpopulation of our *B-lymphocytes* (*B-cells*) derived from our *bone marrow* responds to the *Ag* by producing *antibodies* (*Ab*) in such a way that each of these B-cells secretes a single type of *Ab*'s. Those secreted *Ab*'s are relatively specific for the *Ag*. The *Ag* stimulates the B-cell by binding to these *Ab*'s via *cell-receptors*[1] with a help by yet another stimuli from *T-helper-cell* which we have in our body. Then the stimulated B-cell proliferate (divide) by secreting cells called *plasma cells* and matures itself into a terminal *Ab* (a-cell-not-dividing-any-more). Thus this process of cell division, called *mitosis*, generates a set of cells that are the *progenies* of a single cell. Plasma cells are the most active antibody secretors. B-cells divide rapidly also seceting Ab's but with a lower rate than the plasma-cells. In short, B-cell proliferate and/or differentiate into plasma cells. Furthermore, B-cells differentiate into B-memory-cells which circulate through the blood, lymph and tissures, and if exposed to the same *Ag* again, a set of high affinity *Ab*'s for the specific *Ag* that had stimulated the primary response.

Affinity maturation is a generation of diverse antibody patterns from one specific *Ab* by giving it a random genetic changes using what is called *somatic mutation*, and those generated antibody patterns are called *clones*. Then *learning* in our AI terminology corresponds to the enhancement of the relative population size and affinity of those B-cells that can recognized a given antigen, efficiently. Thus a spectrum of small low affinity clones of B cells each producing an antibody type of different affinity will be generated when the system is exposed to an *Ag*. The *repertoire* of antigen-activated B cells is diversified *hypermutation* and *receptor-editing*. Generally, *Ab*'s in the memory repetoire have, on average, a higher affinity than those of the early primary response. This phenomenon, is referred to as the *maturation of the immune response.* Those B-cells with low affinity receptors will developed entirely new ones through V(D)J recombination, which is called *receptor editing.*

3 Test Target

In this section two test suit for function optimization and three for pettern classification.

3.1 For Function Optimization

Two m-dimensional function are given bellow. Real value y is defined on multi-dimensional Eucledian-coordinate of $(x_1, x_2, \cdots, x_m)$. The difficulty of search might be controlled by changign m. Try, typically, say, $m = 20$. In order to grasp the shape of the function, a 2-dimensional version of each function is shown.

[1] Or, equivalently, they say "*epitok* of antigen binds *paratok* of antibody."

3.1.1 Ackley's Function:

This function has only one global minimum at the origin while many local minima. The goal is to locate the global minimum without being trapped by one of those local minima.

Test-Function 1 (Ackley's Function.) *Search for the point* $(x_1, \cdots, x_n)$ *in n-dimensional space which gives us the global minimum of y.*

$$y = -20 \sum_{i=1}^{n} \exp\left(-0.2\sqrt{x_i^2/n}\right) - \exp\left(\left(\sum_{i=1}^{n} \cos 2\pi x_i\right)/n\right) + 20 + e \quad (x_i \in [-30, 30]). \tag{1}$$

A two dimensional example ($n = 1$) is as follows. See Figure 1 (Top).

$$y = -20 \exp\left(-0.2\sqrt{x^2}\right) - \exp\left(\cos 2\pi x\right) + 20 + e \tag{2}$$

3.1.2 Schwefel's Function:

This function has much more local minima than the Ackley's one above and around half of those local minima is not exactly but almost equal to the global minimum, i.e., zero. Hence much difficult for a search-algorithm to locate the global minimum.

Test-Function 2 (Schwefel's Function.) *Search for the point* $(x_1, \cdots, x_n)$ *in n-dimensional space which gives us the global minimum of y.*

$$y = \sum_{i=1}^{n} (x_i \sin(|x_i|)), \qquad x_i \in [-500, 500]. \tag{3}$$

A two dimensional example ($n = 1$) is as follows. See Figure 1 (Bottom).

$$y = x \sin(|x|). \tag{4}$$

3.1.3 Multimodal Function

Typically, EC finds one global peak at one run. But we have functions which have multipul grobal peaks. Such function is called *multi-modal-function.* In order to test its behaviour of an algorithm the function with five peaks is given as

Test-Function 3 (A Multi-modal Function.)

$$y = \sin^6(5\pi x). \tag{5}$$

Also funcution which has five peaks in a similar way but the altitude is diferent from the one to the next in order for the function to be deceptive.

$$y = \exp(-2((x - 0.1)/0.9)^2) \sin^6(5\pi x). \tag{6}$$

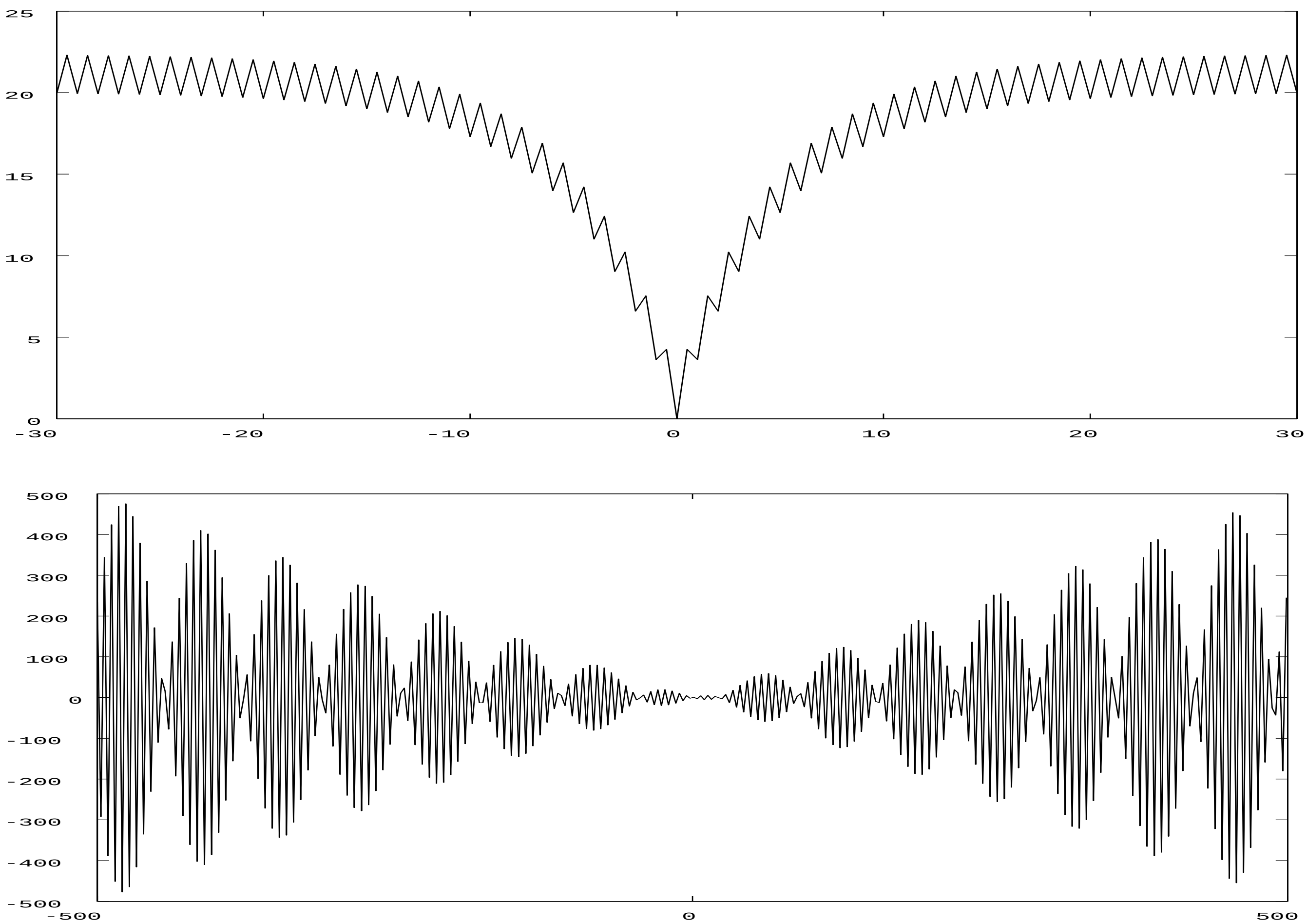

Figure 1: A two-D version of Ackley Function (Top) and Schwefel-function (Bottom).

3.2 For Pattern Classification

For Pattern Recognition/Classification, we need a set of sample for the algorithm to learn with.

3.2.1 Two classes of points drawn from Gaussian p.d.f. in 2-D space.

Here two sets of p points each drawn from two Gaussian distributions whose mean-points are different with each other in 2-D space.

Test-Function 4 (Gaussian distributed two classes) *Pick up two sets of p points each from two different Gaussian distribution to be classified by an algorithm. Give them to an algorithm so that it will be able to classify new points into either of the two classes.*

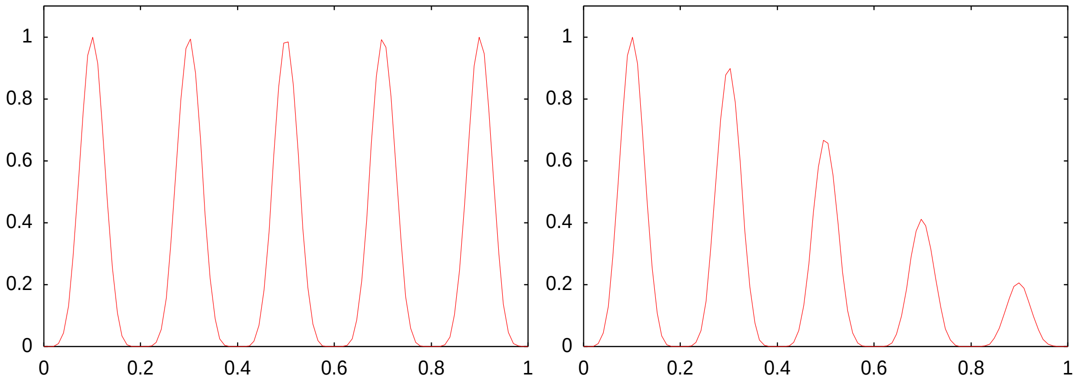

Figure 2: A multimodal function and its deception version

2-dimensional Gaussian distribution is defined as

$$p(x_1 \quad , \quad x_2) = \frac{1}{\sqrt{2\pi}\sigma_1\sigma_2\sqrt{1-\rho^2}} \cdot \exp\{(-\frac{1}{2(1-\rho^2)} \cdot \left((\frac{x_1-\mu_1}{\sigma_1})^2 - 2\rho(\frac{x_1-\mu_1}{\sigma_1})(\frac{x_2-\mu_2}{\sigma_2}) + (\frac{x_2-\mu_2}{\sigma_2})^2 \right)\} \tag{7}$$

where ρ is correlation coefficient defined as:

$$\rho = \frac{\sigma_{12}}{\sigma_1\sigma_2} \tag{8}$$

We now represent two standard deviation σ_1 and σ_2 as

$$\Sigma_i = \begin{pmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{pmatrix}$$

Then, for example, the points we pick up 50 points each from two Gaussian distributions,

$$\mu_1 = (0,0) \text{ and } \mu_2 = (1,1)$$

and

$$\Sigma_1 = \begin{pmatrix} 0.10 & 0 \\ 0 & 0.10 \end{pmatrix}, \qquad \Sigma_2 = \begin{pmatrix} 0.10 & 0 \\ 0 & 0.10 \end{pmatrix}$$

are plotted as Figure 3.2.2

3.2.2 Two classes of points uniformly distributed in high-D space

The Gaussian distribution in the space larger than 2-diminsional space is somehow troublesome for us. So, why don't we pick up points at random from two different region on high-dimensional space.

Test-Function 5 (Random points in two hypercubes) *Assume a search space in a m-dimensional Eucledian space whose coordinates x_i $(i = 1, \cdots, m)$ all lie in $[-1, 1]$.*

(a) Now we have a domain A whose coordinates x_i are all positive, that is,

$$x_i \in [0, 1] \quad i = 1, \cdots, m.$$

Also we have yet another domain B whose coordinates x_i are all negative, that is,

$$x_i \in [-1, 0] \quad i = 1, \cdots, m.$$

Create p points each from domain A and B and use them as a set of training samples. Then create the other points from A and B and make your algorithm classify them.

(b) Here we assume three domains B, C, and D whose coordinates all lie in $[-1/4, 1/4]$, $[-1, -1/2]$, and $[1/2, 1]$, respectively. Then design classifier as in (1) above.

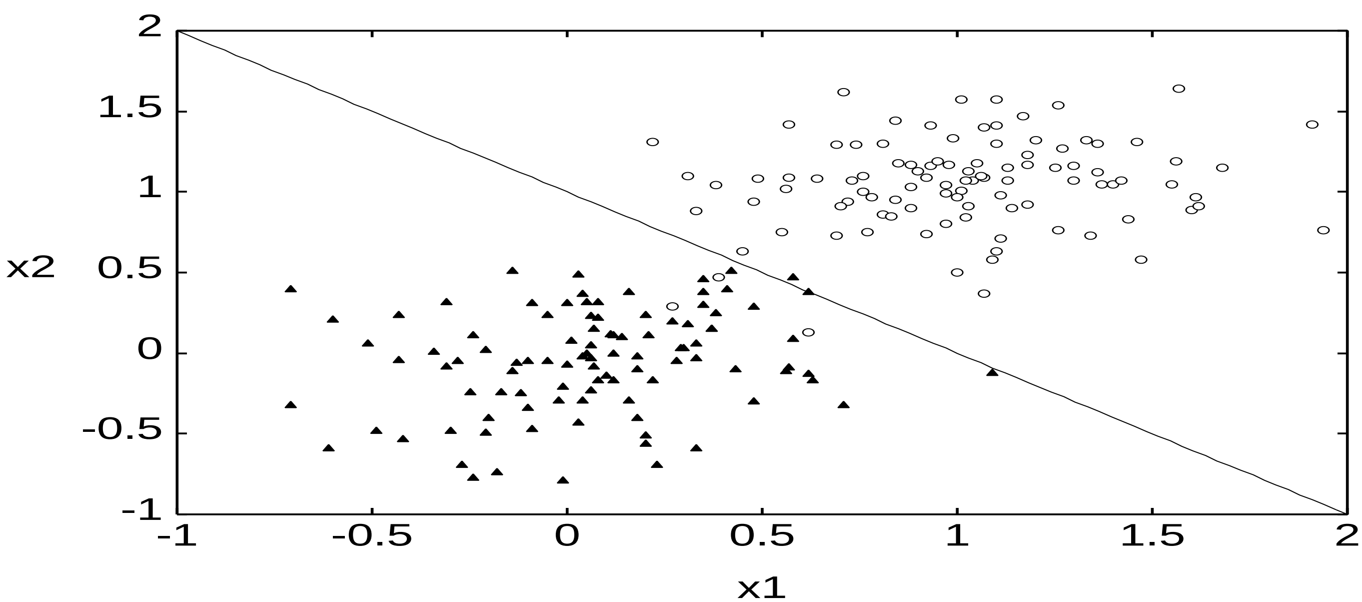

Figure 3: A sketch of the two domains each contains Gaussian distributed 50 points.

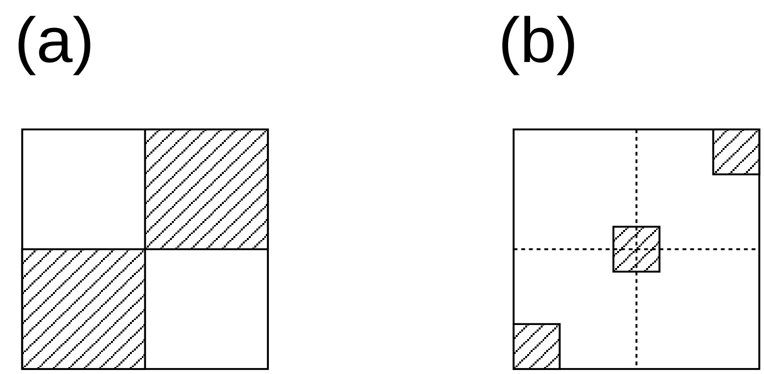

Figure 4: Two dimensional version of case (a) and case (b) in the testfunction above.

3.3 A Strange Fitness Landscape

Here, a strange fitness landscape where search is extremely difficult, if not impossible, when we use Evolutionary Computations.

3.3.1 A Needle in a Haystack

In 1987 Hinton & Nowlan[2] proposed a problem where only one configuration of 20 bits of one and zero was asked to be searched for. The search space is made up of 2^{20} points all of which except for one point are assigned fitness zero. Only one points, say, (11111111110000000000) is assigned fitness one. That is why this is called search for a needle in a haystack. See Figure 1.

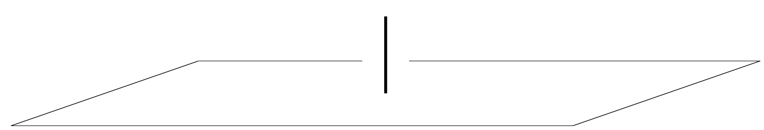

Figure 5: A fictitious sketch of fitness landscape of a needle in a haystack. The haystack here is drawn as a two-dimensional flat plane of fitness zero.

Test-Function 6 (Needle in Haystack) *Search for a point in m-dimensional space whose coordinates x_i $(i = 1, \cdots, m)$ all zero, assuming your algorithm does not know where is the point located.*

[2] Hinton G. E. and S. J. Nowlan (1987) *How Learning can Guide Evolution.* Complex Systems, 1, pp. 495–502.

4 Toy Implementations

The algorithms I describe in this section are dared to be paraphrased by me interpreting by myself, for the purpose of clearer/easier understanding for the readers, at the risk of my misunderstanding. I would appreciate it if you would study both my description and the one in the paper carefully and bug-reports are always welcome, as usual.

4.1 A pattern clusturing/recognition

4.1.1 An example for good start

The algorithm given in this subsection is from Wierzchon et al. (2002).

Algorithm 1 *Assume that both antigen and antibodies are n-dimensional real number vector whose element takes all between 0 and 1, that is, $(x_1, x_2, \cdots, x_n)$ where $x_i \in [0, 1]$.* [3]

1. *Present n antigens Ag_i $(i = 1, \cdots, n)$.*

   This might be called a training set from machine learning aspect.

2. *Generate n antibodies Ab_i $(i = 1, \cdots, n)$.*

3. *Find an initial value of NAT*

   Affinity between two antibodies Ab_j, Ab_k $(j \neq k)$ is defined as Euclidean distance d_{jk}. Then NAT is defiend the average distance between n lowest affinity.

4. *Link each of antigens to othe antigen if the distance between the two antibodies are smaller than NAT as:*

   FOR (i = 1 to n)
       FOR (j=i+1 to n)
           IF $(d(Ab_i, Ab_j) < NAT)$ Link Ab_i with Ab_j.

5. *Caluculate Stimulation-level of all the antibodies $sl(Ab_j)$ $(j = 1, 2 \cdots, n)$ as follows:*

   FOR (j = 1 to n)
       FOR (k=1 to n)
           Calculate Euclidian distance between j-th antibody and k-th antigen $d_{jk} = d(Ab_j, Ag_k)$.
       Calculate minimum value of $d_j = min\{d_{jk} \mid k = 1, \cdots, n\}$
       IF $(d_j \leq NAT)$
           $sl(Ab_j) = 1 - d_j$
       ELSE
           $sl(Ab_j) = 0$

[3] In the original paper the procedure is {Recalculate *NET* ⇒ Re-build network ⇒ Find *Stimulation* ⇒ Clone ⇒ Mutate ⇒ Purge.} The modification is according to my interpretation. If something is wrong I'm responsible not the Author of the paper.

5. *Clone cells.*

   Produce clones of each untibody Ab_j. The number of clones is determined as $c \cdot st(Ab_j)$ where c is a constant which means muximum possible number of clones

6. *Mutate cells*

   Each clone $(y_1, ..., y_m)$ is given mutation as $y_i = y_i + r \cdot \delta$ where r is a randome number which is renewed each i, and δ is either $-y_i$ or $(1 - y_i)$ with equal probability.

7. *Purge worse antibodies.*

Excersise 1 *Apply Algorithm* **??** *to Testfunction 4 and 5.*

4.1.2 Yet another pattern clusturing/recognition

The algorithm given in this subsection is from L. N. de Castoro and J. von Zuben (2002) You might find the paper in this web-page. What is recognized here is an explicit antigen population as in the previous subsection.

Algorithm 2 *Assume that both antigen and antibodies are n-dimensional real number vector whose element takes all between 0 and 1, that is,*

1. *Pick up an antigen Ag_j randomly one by one*[4] *and present it to all the antibodies.*
2. *Calculate affinity $f_i(j)$ of all the antibodies Ab_i $(i = 1, \cdots, N)$.*
3. *Select antibodies of n highest affinity.*
4. *Clone those n highest afftinity antibodies such that the number of clones is proportional to its affinity (the higher the affinity the higher the number). This constructs the repertoir C^j.*
5. *Submitt the repertoir C^j to an affinity maturation process, i.e., inversely propotrional to the affinity (the higher the affinity the lower the mutation rate), This constructs C^{j*} made up of matured clones.*
6. *Calculate affinity of all the Ab in C^{j*}.*
7. *Re-select the highest affinity Ab from C^{j*} if the affinity of this antigen is higher than the one in the memory then the old one is replaced with this Ab*
8. *The lowest d affinity antibodies from non-memory population are replaced with new randomly created*[5] *antibodies.*

[4] In the original paper it is described only "randomly".
[5] This is not in the paper but is inserted by me

4.2 Function optimization

4.2.1 A Function optimization — not necesarily standard but a good start

The algorithm given in this subsection is also from L. N. de Castoro and J. von Zuben (2002) the same as in the previous subsection. Just by modifying the algorithm for pattern recognition abov. You might find the paper in this web-page as well.

As already mentioned, the algorithm is essensially similar to the one in the previous subsection except for two points as foollows:

- In Step 1, there is no explicit antigen population to be recognized, but Here an objective function $g(\cdot)$ is to be optimized (maximized/minimized). This way, an antibody affinity corresponds to the evaluation of the objective function for the given antibody
- as there is no specific antigen population to be recognized, the whole antibody population Ab will compose the memory set and, hence, it is no longer necessary to maintain a separate memory set Abm; and
- Step 7, n antibodies are selected to compose the set Ab, instead of selecting the single best individual Ab^*.

Algorithm 3 *Assume that both antigen and antibodies are n-dimensional real number vector whose element takes all between 0 and 1, that is,*

(1) Construct a set of N antibodies $Ab_i \in \mathbf{Ab}$ $(i = 1, 2, \cdots, N)$.

(2) For each antibody Ab_i $(i = 1, 2, \cdots, N)$ produce a set of clones $\mathbf{C}^i$ whose number is proportional to current affinity of Ab_i.

(3) For each antibody find $C^{i*} \in C^i$, the clone that has the highest affinity

(4) For each pair $(Ab_i; C^{i*})$ check the degree of stimulation of Ab_i and C^{i*}. Strongly stimulated antibody is remembered in $\mathbf{Ab}_{mem}$, while weakly stimulated antibody dies.

(5) Replace a number of weakly stimulated antibodies from $\mathbf{Ab}_{rep}$, by fresh antibodies.

(6) Repeat steps (2) – (5) till a termination condition will be satisfied

4.2.2 Multimodal Optimization

4.2.3 Multi Objective Optimization

4.3 Combinatorial Optimization

this case, the use of an Integer shapespace might be appropriate, where integer-valued vectors of length L, composed of permutations of elements in the set C = 1,2,...,L, represent the possible tours. Each component of the integer vector indexes a city. The total length of each tour gives the affinity measure of the corresponding vector.

4.4 Anomaly Detection — not yet well documented

The algorithm shown in this subsection is from Gonzalez et al. [4] and hence the descriptions here are the ones that I praphrased from their article. Negative Selection (NS) Algorithm is based on *self/non-self* discrimination in the immune system. T-cells that has recognized *self* cells are eliminated, and therefore the remaining T-cells will recognize only *non-self*, i.e., foreign molecules. This is the basic principle of *anomaly detection*. Before we explor the algorithm we might to need to notice followings

Antibody representation. Both of the *self* and *non-self* cells are represented by n-dimensinal binary vectors. Antibodie's representation is different from those described so far in the previous sections. An antibody Ab_j is specified as a hypershpere, that is, by $\mathbf{X}_j$, coordinate of its center and R_j, its radious: both a continuous value.

Affinity evaluation. The affinity of an antibody Ab_j to an antigen Ag_i is defined also in different way from the previous ones. That is, $\mu_j(Ab_i) = \exp(||Ab_j - Ag_i||^2/R_j)$ where $||Ab_j - Ag_i||$ is Eucledian distance between Ab_j and Ag_i.

How an antibodies matches a self cell? To determine if Ab matches a *self* cell, the algorithm pick up the k-nearest neighbors of Ab in the *self* set. If the median distance of these k-neighbors is smaller than R_j then Ab is considered to match the *self*.

Update the location of antibodies. The algorithm is given a set of samples from *self* cells represented by n-dimensional vector. Then the algorithm tries to evolve *antibodies* so that antivodies cover the whole (*non-self*) space as much as possible. This is by repeating updates the location of the antibodies with the following two goals: (1) If an Ab maches to a *self* cell then *move* the Ab away from the *self*. The direction to move is along the vector $\eta \sum_{i=1}^{k}(X_i - Ab_j)/k$; (2) If an Ab maches to a *self* cell, then *move* the Ab away from other antigens. This keeps the antigens separated in order to maximize the covering of the *non-self* space. $\eta \sum_{j'=1}^{N} \mu_j(Ab_j) \cdot (Ab_j - Ab_{j'}) / \sum_{j'=1}^{N} \mu_j(Ab_{j'})$

Thus if an antibody matches any *self* then move the antibody away from the self on the condition that the antigen is not very old and increment the age of the antibody. If it's very old kill the antibody. Alas! On the contrary if the antibody does not match anty *self* then move the antigen away from other antigens and reset its age to zero. The algorithm is the repetition of the above procedures. To be more specific,

Algorithm 4 *Each antibodie is a hyperspher represented with the coordinate of its center, n continuous values; and its radious, also a continuous value. Both Self and non-self cells are n-dimensional binary vectors. Assume ρ is radius of antigen; η adaptation rate of step size when antigen moves which decreases from generation to generation; and τ is the time when antibody is considered to be matured.* [6]

1. *Generate a random population of antibodies.*
2. *FOR each antibodies Ab_j $(j = 1, 2, \cdots, N)$*

   (1) Select k self cells X_i $(i = 1, \cdots, k)$ which are the k nearest from Ab_j; Refer the cell whose distance is the median of these k cells to Cr_j;

   (2) IF $||Ab_j - Cr_j||$, the distance between Ab_j and Cr_j, is smaller than ρ

   THEN IF $age_j > \tau$

   THEN replace Ab_j with a new random antibodie.

   ELSE increment age_j; and move Ab_j as $Ab_j = Ab_j + \eta \cdot Dr_j$ where

$$Dr_j = (\sum_{i=1}^{k}(Ab_j - X_i))/k \tag{9}$$

   ELSE $age_j = 0$; and move Ab_j as $Ab_j = Ab_j + \eta \cdot Dr_j$

$$Dr_j = (\sum_{i=1}^{k}(Ab_j - X_i))/k \tag{10}$$

3. *WHILE stopping criteria is not satisfied repeat 2.*

References

[1] G. E. Hinton and S.J. Nowlan (1987) *How Learning can Guide Evolution.* Complex Systems, 1, pp.495–502.

[2] L. N. de Castoro and J. von Zuben (2002) *Learning and Optimization using the Clonal Selection Principle.* IEEE Transactions on Evolutionary Computation 6 (3) pp. 239–251

[3] S. T. Wierzchon and U. Kuzelewska (2002) *"Adaptive Clusters formation in an Artificial Immune System."*

[4] F. A. Gonzalez and D. Dasgupta (2003) Anomaly Detection Using Real-Valued Negative Selection Genetic Programming and Evolvable Machines, 4(4), pages 383-403, Kluwer Academic Publishers.

[6] Therefore Real-Valued-Negative-Selection might be said to be a quadrapl (ρ, η, τ, k).