Immune Programming

Payman Samadi

Supervisor: Dr. Majid Ahmadi

March 2006

Department of Electrical & Computer Engineering
University of Windsor
OUTLINE

• Introduction
• Biological Immune System
• Artificial Immune System
• Immune Programming
  • Computer Architecture
  • Algorithm
• Results
• Performance Evaluation
• Comparison of Immune Programming and Genetic Programming
• Conclusion
Introduction

• Artificial Intelligence
  Goal: Having Computers automatically solve problems without explicitly programmed to do that.

• Evolutionary Computation
  mechanizing the scientific method in an algorithmic formulation so that a machine may carry out the procedure and similarly gain knowledge about its environment and adapt its behavior to meet goals.

Different methods:
  – Evolution strategies
  – Evolutionary computing
  – Genetic algorithm
  – Genetic programming
Biological Immune system

• One of the most intricate bodily systems
• Composed of great variety of molecules, cells, and organs spread through the body
• There is no central organ controlling the immune system
• Able to recognize the organism’s own cells and tissues to prevent their inadvertent destruction.
• Main role
  • protect the organism against disease causing cells called pathogens
  • Eliminate Malfunctioning Cells
• This accomplished through RECONGNITION.
Biological Immune system

- Antigen: all elements recognizable by the immune system such as pathogens, malfunctioning cells and healthy cells.

- Antigens:
  - Self: native cells that originally belong to the organism and are harmless to its functioning.
  - Non-self: disease causing elements.

- Self / Non-Self discrimination
  - After identifying as non-self:
    - Deactivating the pathogens.
    - Saving on memory and learning: pattern recognition, clonal selection, negative selection and affinity maturation
Biological Immune system

• Pattern Recognition:
  Carried out by white blood cells (lymphocytes): B-Cell & T-Cell

B-cells and T-cells both have receptors responsible for recognizing Antigenic pattern.

• B-cells: recognizing isolated antigens from outside.
• T-Cells: recognizing parts of the complex presented by organic molecules.
Biological Immune system

• Recognition process:
  Based on matching the shape of antigen with the shape represented by the surface receptors of B-Cells and T-Cells

Binding is often is not perfect but still leads to correct recognition.

*Affinity*: Degree of binding
Biological Immune system

• Clonal Selection:
  Process in which the size of subpopulation of B-Cells and T-Cells are controlled, depending upon the extent of the infection.

• Hypermutation: subpopulation of cells that are slightly different.

• Maturation: whole process of selection and hypermutation.

• High affinity antigens are selected to become memory cells with long life spans.
Biological Immune system

• Negative Selection:

  The process of training receptors for discrimination between self and non-self antigens
Artificial Immune System

- Computational system inspired by theatrical immunology and observed immune functions, principles, and models, applied to solve the problems.

- Several analogies between EC & AIS:
  - Frameworks are inspired by biological systems
  - Employ some form of evolutionary principles
  - Can be used in largely overlapping domains
Artificial Immune System

• AIS is described using a framework specifying its
  – Representation
  – Evaluation
  – Adaptation
Artificial Immune System

• Representation scheme:
  
  – Constructed using models of antigens and antibodies
  – Recognition: shape-space model of Perelson and Oster.
  – Elements either antigen or antibody is represented by attribute string \( m \).
    
    \[
    m = \langle m_1, m_2, \cdots, m_L \rangle
    \]

• \( m \) can be integer or binary.
Artificial Immune System

• Evaluation:
  – Interaction between antigen and antibody is described by the degree of binding in terms of affinity.

• Affinity of antigen-antibody pair is related to their distance in the shape space S.

• Minkowski Distance Measure:
  \[ D_m(Ag, Ab) = \sqrt[p]{\sum_{i=1}^{L} |Ag_i - Ab_i|^p} \]
Artificial Immune System

- Adaptation:
  This procedure governs the evolution of the behavior of an AIS.

- It is a population-based algorithm.

- Population-based algorithms can be categorized as
  - Clonal Selection Algorithm: Simulation of B-cells behavior
  - Negative Selection Algorithm: Simulation of T-cells behavior
Artificial Immune System

- Clonal Selection Algorithm is based on Clonal Selection process in biological IS.
  
  1. n candidate solutions are generated and evaluated using a suitable affinity measure.
  2. n attribute strings with highest affinity are selected to proliferate by cloning; the cloning rate is proportional to affinity.
  3. Newly generated clones are subjected to hypermutation; the mutation rate is inversely proportional to its affinity.

- This algorithm results in change of antibody concentrations favoring those with high affinity, and added diversity through the process of hypermutation.
Artificial Immune System

• Negative Selection Algorithm is based on self/non-self discrimination capability in biological IS.

  1. n candidate detectors are generated.
  2. Each candidate detector, $C_i$, $i = 1,2, \ldots, n$, is compared to a set of protected elements, PE.
  3. If a match occurs between $C_i$ and PE, the detector is discarded.
  4. Otherwise (if a match does not occur), $C_i$ is stored in the detector set $D$.

• This algorithm produces a set of detectors capable to recognize non-self patterns.
Immune Programming

- Novel paradigm combining the program-like representation of solutions to problems with principles & theories of the immune system.

- An extension of immune algorithms, particularly clonal selection algorithm in AIS.

- John Koza: “Immune programming is a systematic method for getting computers to automatically solve a problem.”

- Immune Programming :
  » Computer Architecture
  » Algorithm
Immune Programming
Computer Architecture

• Automatically generated programs are executed on computer architecture.

• In IP computer architecture is stack-based.

• Stack-based advantages:
  – Small size
  – Low system complexity
  – High system performance
  – Good performance under varying conditions
A computing architecture is functionally described by an instruction set. The instruction set provides a detailed list of the operations that the machine is capable of processing, and description of the types, locations, and access methods for operands.

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nop</td>
<td>0</td>
<td>No operation</td>
</tr>
<tr>
<td>dup</td>
<td>1</td>
<td>Duplicate the top of the stack ((x \Rightarrow x x))</td>
</tr>
<tr>
<td>swap</td>
<td>2</td>
<td>Swap the top two elements of the stack ((x y \Rightarrow y x))</td>
</tr>
<tr>
<td>mult</td>
<td>3</td>
<td>Multiply the top two elements of the stack ((23 \Rightarrow 6))</td>
</tr>
<tr>
<td>add</td>
<td>4</td>
<td>Add the top two elements of the stack ((23 \Rightarrow 5))</td>
</tr>
<tr>
<td>over</td>
<td>5</td>
<td>Duplicate the second item on the stack ((x y \Rightarrow y x y))</td>
</tr>
</tbody>
</table>
Immune Programming

Algorithm

1. Initialization: n Antibodies are generated.

2. Evaluation: all of the Abs are compared to Ag and $f_i$ is generated.

3. Replacement: low-affinity antibodies are replaced.

4. Cloning: if there is no new antibody, high-affinity Abs are cloned.

5. Hypermutation: if high-affinity antibody is not cloned in previous step, it is submitted for hypermutation in this step.

6. Iteration–repertoire: step 3-5 is repeated until new AB is constructed.

Immune Programming
Algorithm - Initialization

- Initialization:
  - The initial repertoire of antibodies is randomly selected from gene libraries.
  - In IP algorithm libraries contain instructions available in instruction set of computer architecture.
  - The length of the program can be variable or fixed.
  - The size of instruction set ‘r’ and the program length ‘L’ dictates the size of the set of all antibodies.
  - $r^L$ should be large enough to ensure sufficient antibody diversity.
Immune Programming

Algorithm - Initialization

- Example:
  - $A_{b_1} = (1,3,2,5,1,4)$
    - dup mult swap over dup mult
  - $A_{b_2} = (4,1,2,2,3,0)$
    - Add dup swap swap mult nop
  - $A_{b_3} = (0,2,0,5,5,5)$
    - nop swap nop over over over
Immune Programming

Algorithm - Evaluation

- Antigen can be pairs of input-output numerical values representing desired mapping or an arithmetic expression.

- To evaluate the affinity, particular values have to be placed on the stack and the program should be executed.

- Values are randomly selected and restricted to 1-byte integer values, \([0,\ldots,255]\).

- Affinity is determined by calculating the distance (e.g. Minkowski distance, \(p=2\)) but has some disadvantages.
Immune Programming
Algorithm - Evaluation

- Disadvantages:
  - No value for programs who has not executed correctly.
  - Very large range for possible affinity values.

- Alternative method for evaluating affinity is to consider three important properties of the generated programs:
  - Executability: if program executed correctly: score $T_1$
  - Completeness: if program returns only single value: score $T_2$
  - Correctness: if program yields to correct result: score $T_3$

$$T_1 < T_2 < T_3$$
Immune Programming

Algorithm - Evaluation

- Example for definition of scores:
  \[ T_1 = 1 \]
  \[ T_2 = c(T_1 + 1) \]
  \[ T_3 = c(T_1 + T_2 + 1) \]

Maximum Affinity:
\[ f^M = c(T_1 + T_2 + T_3) \]

\[ f_i = \sum_{j=1}^{c} T_1(Pg_i, \text{arg}_j) + T_2(Pg_i, \text{arg}_j) + T_3(Pg_i, \text{arg}_j) - KL_a \]

\text{arg}_j \text{ is j-th set of arguments}

\text{La is length of the program and k is a constant of proportionality}
Immune Programming
Algorithm - Replacement

• Process of replacement:
  • A random number RAND [0, 1] is generated and compared to a parameter called probability of replacement, Pr.
  • If RAND < Pr, a new antibody is generated and placed in the new repertoire and the algorithm proceeds to step 6, iteration–repertoire.
  • If RAND > Pr, no antibody is placed in the new repertoire and the algorithm proceeds to the next step.

• These new antibodies are not considered for cloning and mutation is subsequent steps and algorithm proceeds to step 6.
Immune Programming
Algorithm – Cloning

• If a new antibody has not generated in step 3, an antibody is considered for cloning. Antibodies are selected in sequential manner.

• Process:
  – A random number RAND is generated and compared to the relative affinity of the antibody. If RAND < $f_N / i$, the antibody is cloned and put into the new generation with probability of cloning Pc. This concludes the current iteration of creating the new repertoire, step 6, and the algorithm returns to step 3, unless the new repertoire is complete, $|\text{AB'}| = n$, in which case the algorithm proceeds to step 7.

  – If RAND < $f_N / i$ but Ab$_i$ has not been cloned due to the stochastic character of the cloning process, the antibody is submitted to hypermutation.
Immune Programming
Algorithm – Cloning

- Two stochastic aspects of the cloning process:
  - First: affinity ensures that mainly high-affinity antibodies are considered for cloning. Low-affinity antibodies can be cloned but with much smaller probability.
  - Second: \( P_c \) limits the proportion of antibodies that are actually cloned (high-affinity subpopulation).
• If the high-affinity antibody selected in the previous step has not been cloned (due to $P_c$), it is submitted to the process of hypermutation.

• This process walks through the attribute string $m = (m_1, m_2, ..., m_L)$ of the antibody, $A_{bi}$, and replaces each attribute $m_j$ with a new randomly generated value.

• This replacement is driven by the probability of mutation, $P_m$, so only a portion of the attributes is actually replaced.
Immune Programming
Algorithm – Hypermutation

- Pm should be inversely proportional to affinity, so Pm is scaled by its normalized affinity. \((Pm/fin),1\) is used for actual decision for replacement.

- The hypermutation provides the algorithm with the ability to introduce new material into the repertoire and expands the solution space searched.

- The inverse proportionality of hypermutation ensures that high-affinity antibodies are disturbed only slightly while low-affinity ones are modified to a high extent.
Immune Programming

Algorithm – Iteration-repertoire

- Steps 3 to 5 (replacement, cloning and hypermutation) are repeated until a complete new repertoire, |AB'| = n, has been created.
Immune Programming

Algorithm – Iteration-algorithm

- After the new repertoire has been constructed, the generation counter (set to value $G = 1$ during initialization) is incremented, $G = G + 1$. The algorithm then proceeds iteratively through steps 2–6 (evaluation, replacement, cloning, hypermutation, iteration–repertoire) until a stopping criterion is met.

- Stop Criteria:
  - Present number of iteration
  - Fitness threshold
  - No fitness improvement
Results

• Several experiments are done to examine the performance of IP.

• Parameters:
  • $n = 100 \quad P_r = 0.5 \quad P_c = 0.1 \quad P_m = 0.2 \quad L$ is varied.

• Experiments:
  – High-order operations
  – Multiple variables
  – Factorization
  – No simplification
Results

- High-order operations:

- The goal of this experiment is to examine the performance of IP to deal with operations of high-order.

- The expression is: $x^8$

<table>
<thead>
<tr>
<th>Performance of the IP system: high-order operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Constrained program size, $L$</td>
</tr>
<tr>
<td>Number of generations, $\bar{G}$</td>
</tr>
</tbody>
</table>
Results

- Multiple variables:

- The second experiment is provided to illustrate the ability of the IP system to handle multiple variables.

- The particular case considered involves three variables \(x, y, z\) in expression \(x \cdot y + y^2 + z\).

<table>
<thead>
<tr>
<th>Performance of the IP system: multiple variables</th>
</tr>
</thead>
<tbody>
<tr>
<td>Constrained program size, (L)</td>
</tr>
<tr>
<td>Number of generations, (\overline{G})</td>
</tr>
</tbody>
</table>
Results

• Factorization:

• The goal of the third experiment is to verify whether the IP system is capable of simplifying arithmetic expressions, e.g., by factorization.

• For instance, the antigen expression $x^2*y^2$ can be reformulated as $(x*y)^2$.

<table>
<thead>
<tr>
<th>Performance of the IP system: factorization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Constrained program size, $L$</td>
</tr>
<tr>
<td>Number of generations, $\overline{G}$</td>
</tr>
</tbody>
</table>
Results

- No simplification:

- The fourth experiment examines the performance of the IP system while solving more complicated expressions that cannot be further simplified.

- The expression considered is $x^2 + y^2 + x + y$.

<table>
<thead>
<tr>
<th>Performance of the IP system: no simplification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Constrained program size, $L$</td>
</tr>
<tr>
<td>Number of generations, $\overline{G}$</td>
</tr>
</tbody>
</table>
Performance Evaluation

- Several experiments are done to study the evolution of affinity and the sensitivity of convergence to algorithm parameters.

- These experiments are:
  - Evolution of affinity
  - Sensitivity Analysis
    - Effect of repertoire size \( n \)
    - Effect of probability of replacement, \( Pr \)
    - Effect of probability of cloning, \( Pc \)
    - Effect of probability of mutation, \( Pm \)

- \( L \) is kept constant at 10 and no limits for number of generations
Performance Evaluation

- Evolution of affinity with respect to number of Generations:
Performance Evaluation

- Sensitivity of IP performance respect to population size, n:

![Graph](image-url)
Performance Evaluation

- Sensitivity of IP performance with respect to probability of replacement, \( P_r \)
Performance Evaluation

- Sensitivity of IP performance with respect to probability of cloning, $P_c$
Performance Evaluation

- Sensitivity of IP performance with respect to probability of mutation, $P_m$
Comparison of Immune Prog. & Genetic Prog.

- The expression considered for this experiment is:
  \[ x^3 + 3x^2y + 3xy^2 + y^3 = (x + y)^3 \]

- Experiments are conducted for minimal length program \( L = 5 \) and twice length \( L = 10 \).

- The stopping criteria is either success in finding a solution or a maximum number Generations. \( G_{\text{max}} = 1000 \).
## Comparison of Immune Prog. & Genetic Prog.

### Comparison of convergence of GP and IP

<table>
<thead>
<tr>
<th>$L$</th>
<th>System</th>
<th>Repertoire/population size $n$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>10</td>
</tr>
<tr>
<td>5</td>
<td>GP</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0%</td>
</tr>
<tr>
<td></td>
<td>IP</td>
<td>298</td>
</tr>
<tr>
<td></td>
<td></td>
<td>90%</td>
</tr>
<tr>
<td>10</td>
<td>GP</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0%</td>
</tr>
<tr>
<td></td>
<td>IP</td>
<td>553</td>
</tr>
<tr>
<td></td>
<td></td>
<td>90%</td>
</tr>
</tbody>
</table>
Conclusion

• IP is a new paradigm in evolutionary computing.

• IP is systematic, domain-independent method to intelligently solving programming problems with no human interaction.

• Convergence of IP is superior to stack-based GP.

• IP converges in situations that cannot be handled by GP, arising from IP’s inherently improved repertoire diversity.