Towards development of a hybrid discrete graph-based ant colony optimization algorithm with an antibody-based immune programming algorithm

Barton, Alan

Report (National Research Council of Canada. Radio and Electrical Engineering Division : ERB), 2005-04-21

National Research Council Canada
Conseil national de recherches Canada

Institute for Information Technology
Institut de technologie de l'information

NRC · CNRC

*Towards Development of a Hybrid Discrete Graph-Based Ant Colony Optimization Algorithm with an Antibody-Based Immune Programming Algorithm **

Barton, A.
April 2005

* published as NRC/ERB-1126. April 21, 2005. 5 pages. NRC 48119.

Canada

Towards Development of a Hybrid Discrete Graph-based Ant Colony Optimization Algorithm with an Antibody-based Immune Programming Algorithm*

Alan J. Barton†
Integrated Reasoning Group
Institute for Information Technology
National Research Council Canada
alan.barton@nrc-cnrc.gc.ca

ABSTRACT

An attempt is made to improve swarm-based optimization algorithms (SBOA) through an investigation and development in C of the following: *i)* Random Algorithm, *ii)* Immune Programming Algorithm (IP), *iii)* Ant Colony Optimization Algorithm (ACO), and *iv)* Hybridized ACO and IP. Preliminary experiments are performed and reported for both integer ($\mathbb{Z}$) stack-based and real ($\mathbb{R}$) model-based problem representations.

Keywords

optimization, approximation, stigmergy, emergence

1. INTRODUCTION

> We should not rely on careful examinations, we should avoid the need for it. – *J.W. Tukey, 1977*[1]

Nature inspires. Biology inspires. Computer Science may use such inspiration for the construction of new algorithms. In particular, foraging ants and the human immune system [3] have lead to the ACO [1], [7] and IP [5], [2], [8] algorithms, respectively. This work hopes to provide an explication of related algorithms from a Computer Science perspective with the intent of contributing to the development of a hybrid ant immune-system algorithm.

Given an input/output mapping, one becomes interested in determining an apriori *unknown* exact-form analytical function that fits the data. There are many possible classical optimization algorithms that could be used (least squares being one). There are also recent advances in local-search based simple agents using stigmergy (environment-based memory) that could potentially be applied if a suitable problem representation exists.

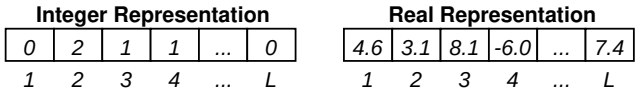

Figure 1: Two possible representations having length L. Left: $\mathbb{Z}$-Antibody Right: $\mathbb{R}$-Antibody

Inst.	Code	Description
nop	0	No Operation (enables antibodies of varying length)
dup	1	Duplicate the top of the stack ($x \Rightarrow xx$)
ste	2	Swap the top two elements of the stack ($xy \Rightarrow yx$)
mult	3	Multiply the top two elements of the stack ($23 \Rightarrow 6$)
add	4	Add the top two elements of the stack ($23 \Rightarrow 5$)
over	5	Duplicate the second item on the stack ($xy \Rightarrow yxy$)
etpo	6	The top of the stack is replaced with e^x ($x \Rightarrow e^x$). This was not in [5].

Table 1: A $\mathbb{Z}$-Antibody may be decoded into stack-based instructions using this instruction set

2. PROBLEM REPRESENTATION

At least two representations may be used (Fig.1), integer-based, which shall be referred to as a $\mathbb{Z}$-Antibody and real-based, which shall be referred to as an $\mathbb{R}$-Antibody. Decoding the representation in order to solve the problem may be done in at least two ways. The $\mathbb{Z}$-Antibody's elements may be interpreted as op-codes from an instruction set [5], but not sufficiently fine-grained in order to be useful for discovering low error models, while the $\mathbb{R}$-Antibody's elements may be considered as either op-codes from an instruction set or constants in a model.

For example, for a $\mathbb{Z}$-Antibody of length 4, with values as

*Final Project for the course COMP5900Z entitled *Swarm Intelligence* at Carleton University

†Master of Computer Science (in progress)

[1]John W. Tukey was referring to the fact that plotted data (e.g. a scatter plot) should be clearly presented to a reader when attempting to argue for a particular perspective[6].

in (Fig.1), it may be decoded as the stack-based evaluation function (using the instruction set in Table.1).

$$nop\ \ ste\ \ dup\ \ dup \tag{1}$$

For example, for an $\mathbb{R}$-Antibody of length 4, with values as in (Fig.1), it may be decoded as the model-based evaluation function using the model

$$a \cdot e^{b \cdot x} + c \cdot x + d$$

where antibody= $[a, b, c, d]$.

$$4.6 \cdot e^{3.1 \cdot x} + 8.1 \cdot x + -6.0 \tag{2}$$

It becomes apparent that the latter form is easier to read and interpret, inclining one to favour the model-based approach over the stack-based approach. Irrespective of this, $\mathbb{Z}$-Antibody and $\mathbb{R}$-Antibody based algorithms were developed and investigated (albeit briefly for some).

3. ALGORITHMS

The following algorithms were implemented in the programming language C. The Ant Colony Optimization subsection begins to explain how the mapping is made from $\mathbb{Z}$- and $\mathbb{R}$-Antibodies to patches and nests.

3.1 Random

Completely randomly generate populations of $\mathbb{Z}$- and $\mathbb{R}$-Antibodies. This algorithm may be used as a baseline over which all other algorithms should enhance.

ALGORITHM 1. Random(m, seed, t_{max})

```
1  for(t ← 1) to (t_max) do
2     for(k ← 0) to (num_ants−1) do
3        build_solution_for_ant(k)
4        compute_cost_of_solution_for_ant(k)
5        recordBestSolutionFoundSoFar()
6     endfor
7  endfor
```

— End of Algorithm —

3.2 Immune Programming

The Immune Programming algorithm as explained in [5] was implemented with the help of the PGAPack library [4]. The algorithm was enhanced in order to support various forms of elitism, which was not used in [5]. For example, from one repertoire to the next, one may keep r best affinity-valued antibodies. The IP algorithm was also enhanced to incorporate the idea of a maximum number of iterations, which was not used in [5].

ALGORITHM 2. IP (n,r,seed,t_{max},$Pr(Rep)$,$Pr(Hyp)$,$Pr(Clo)$)

```
1  init_pop_of_antibodies() do
2    for(i←0) to size(repertoire) do
3      add_new_antibody_to_repertoire()
4      for(j←0) to len(new_antibody) do
5        new_antibody[j] ←
6                          [minOpCode..maxOpCode]
7      endfor
8    endfor
9  endFcn
10 evaluateAffinityOfAllAntibodies()
11 do
12   // Elitism was not in original algorithm
13   copy_r_BestIndividualsToNewRepertoire()
14   i ← 0 // index into old repertoire
15   while(
16     constructionOfNewRepertoireNotComplete)
17   do
18     if(i = size(oldRepertoire)) do
19       i ← 0
20     endif
21     if(rand() ≤ Pr(Rep)) do
22       generateRandomAntibody() do
23         for(j←0) to len(new_antibody) do
24           new_antibody[j] ←
25                          [minOpCode..maxOpCode]
26         endfor
27       endFcn
28       addNewRandAntibodyToNewRepertoire()
29     else do
30       i ← i + 1
31       // clone or mutate?
32       if(rand() ≤ affinityOfAntibody(i)) do
33         cloneAntibodyToNewRepertoire(i)
34         if(rand() > Pr(Clo)) do
35           mutateAntibodyInNewRepertoire(i)
36         endif
37       endif
38     endelse
39   endwhile
40   //constructionOfNewRepertoireComplete
41   evaluateAffinityOfAllAntibodies()
42 while(notDone)
43   // tooSimilar || noChange || maxIterations
```

— End of Algorithm —

3.3 Ant Colony Optimization

The ACO algorithm as applied to the Quadratic Assignment Problem in [1] was implemented and modified for the purposes outlined above. Fig.2 presents possible ways for ants to forage over patches in a world, remembering their graph traversal histories, and providing solutions at the end of their life. An ant dies when it gets to the other side of the world (the world is flat) or when it reaches food and gorges itself to death. Fig.3 demonstrates another possible representation where an ant is born on any patch and walks either east or west (wrapping around the world) until it comes to its own nest and subsequently dies. Ants must make a decision about which patch to move to based on pheromone intensity on the patches in the neighbouring nest in the direction of their world tour.

Fig.3 shows a $\mathbb{Z}$-Antibody mapped onto the ant world, where Nest i is position i in the $\mathbb{Z}$-antibody. For example, if an ant is standing on a patch that has a mapped value coming from a $\mathbb{Z}$-Antibody, then that may be decoded using the instruction set in Table.1.

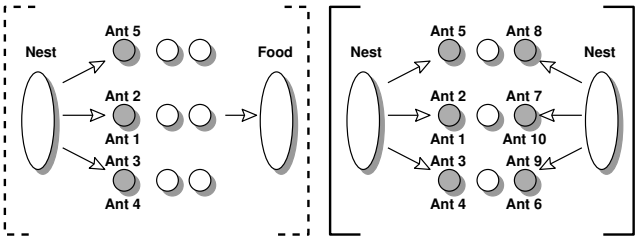

Figure 2: Two possible nest/food configurations. Left: 1 nest and one food source, where ants build a solution as they traverse the graph from left to right. Right: 2 nests (and no food) where ants simply move away from their respective nests, thereby building a solution. This variant may have ψ ants at Nest 1 and $n - \psi$ at Nest 2.

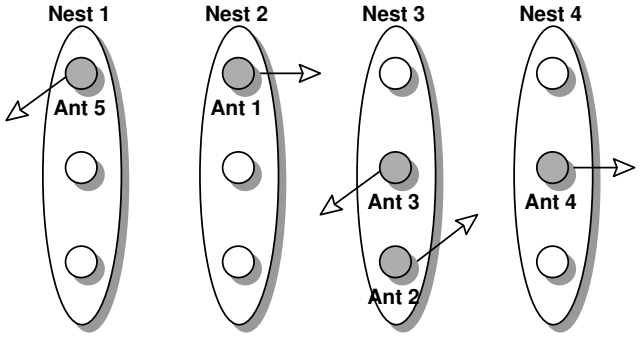

Figure 3: Ants foraging and making decisions about what patch to move to. In this scheme, every ant may stand on a patch irrespective of how many ants are already standing on the patch, but the patches only exist in the world inside of nests.

ALGORITHM 3. ACO
(n, seed, α, β, ρ, t_{max}, τ_0, Q)

```
1  assign_initial_pheromone_levels()
2  place_ants_on_locations()
3  p ← constructAPermutation()
4  for(t ← 1) to (t_max) do
5     //place_ants_on_locations()?
6     randomize(p)
7     for(k ← 0) to (num_ants−1) do
8        // consider ants in random order
9        build_solution_for_ant(p[k])
10    endfor
11    for(k ← 0) to (num_ants−1) do
12       compute_cost_of_solution_for_ant(k)
13    endfor
14    recordBestSolutionFoundByAnAnt()
15    update_pheromone_trails()
16 endfor
```

— End of Algorithm —

3.4 Hybrid

This algorithm is a hybridization of ACO and IP in the sense that ACO is used, but every *resetFreq* time steps a completely new mapping on the world is performed. This new generation of map values may be done in the same manner as upon initialization (e.g. uniform random distribution) or may be done based on the learned pheromone levels of the ants, thereby passing learned information from one world map to the next. It is hoped that the ants have learned good mapping values, from which nearby values could be generated (a local search heuristic). This is in the spirit of IP w.r.t. the generation of new antibodies based on their affinity to the antigen, which represents the problem (input/output mapping) that is being attempted to be solved.

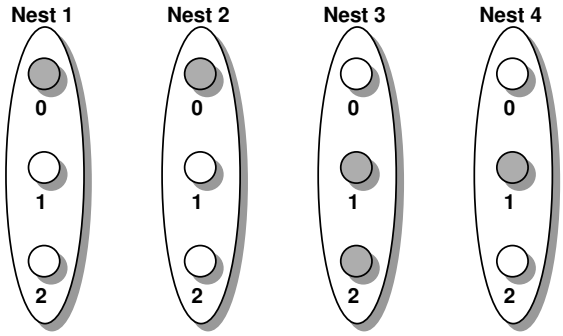

Figure 4: $\mathbb{Z}$-Antibody mapped onto the ant World, where Nest i represents position i in the antibody.

ALGORITHM 4. Hybrid
(n, seed, resetFreq, α, ρ, L, P, t_{max}, τ_0, Q)

```
1  assign_initial_pheromone_levels()
2  place_ants_on_locations()
3  p ← constructAPermutation()
4  for(t ← 1) to (t_max) do
5     randomize(p)
6     for(k ← 0) to (num_ants−1) do
7        // consider ants in random order
8        build_solution_for_ant(p[k]) do
9           for(i ← 0 to (len(antibody)) do
10             do
11                j ← nextIndexBasedOnDirection()
12                Pr(patch) ← basedOnPheromone()
13             while(rand() < Pr(patch)
14             ants[p[k]].history[i] ← j
15             // i.e. allow ant to move to patch
16             // (assign opcode to ant's solution
17          endfor
18       endFcn
19    endfor
20    for(k ← 0) to (num_ants−1) do
21       compute_cost_of_solution_for_ant(k)
22    endfor
23    recordBestSolutionFoundByAnyAnt()
24    update_pheromone_trails()
25    if(reset_freq > 0 do
26       if(t%reset_freq=0) do
27          update_map_and_pheromone_levels()
28       endif
29    endif
30 endfor
```

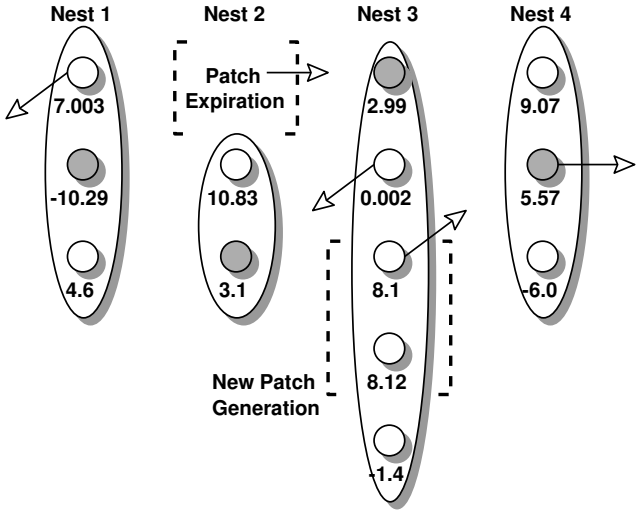

Figure 5: A static snapshot of a dynamic ant World with a mapping to the reals. $\mathbb{N}$- and $\mathbb{R}$-antibodies may be represented in this world.

— End of Algorithm —

Fig.5 demonstrates a static snapshot of how an $\mathbb{R}$-antibody may be computationally evolved from ants and immune-system concepts. A new patch representing an additional real value on an attribute in the search space that the ants are exploring may be generated in many possible ways. The current implementation generates a new real value completely randomly, but a more local based generation strategy, such as generating based on a gaussian distribution of mean μ and standard deviation σ, may lead to faster convergence.

4. EXPERIMENTAL RESULTS

All of the algorithms were implemented and executables run. Only the results of the most complex implementation are reported.

Fig.6 shows the best antibody found for a series of 100 experiments where all parameters were kept constant except the seed, which was calculated based on computer clock time. In particular, the number of ants was 200, the maximum number of iterations was 200, the mapping graph would be regenerated completely at randomly every 4 iterations (in order to give the ants enough time to search), α was 2, ρ was 0.1, τ_0 was 0.1, the strategy for updating the pheromone levels was based on normalization, the $\mathbb{R}$-antibody had length 4 and the number of patches per nest was fixed at 20.

The overall distribution of solution costs is shown in Fig.7. It can be seen to have a skewed distribution with average solution costs in the neighbourhood of 1.

The best solution (Fig.6) was found at $t = 153$, with a solution cost of 0.224718 and Seed= 1113951307. The ant started at Nest 0, and moved in an westerly direction. The complete graph traversal history is $[19, 17, 4, 12]$, where the corresponding mapped $\mathbb{R}$-Antibody (labelled as antibody1)

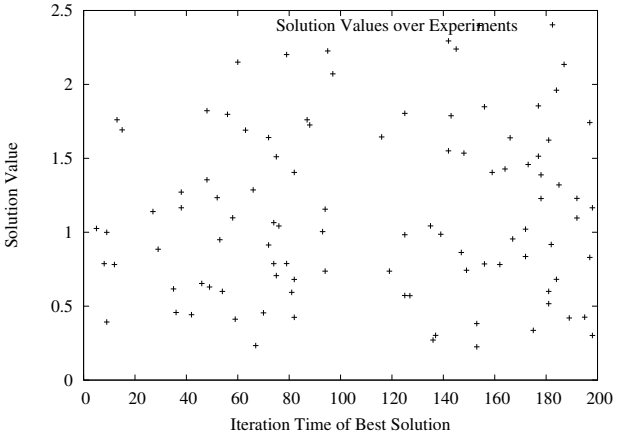

Figure 6: 100 trials with fixed algorithm parameters, except seed. Good solutions (models fitting the antigen very closely) are distributed throughout the space of time steps. Demonstrating the stochastic nature of the hybrid algorithm.

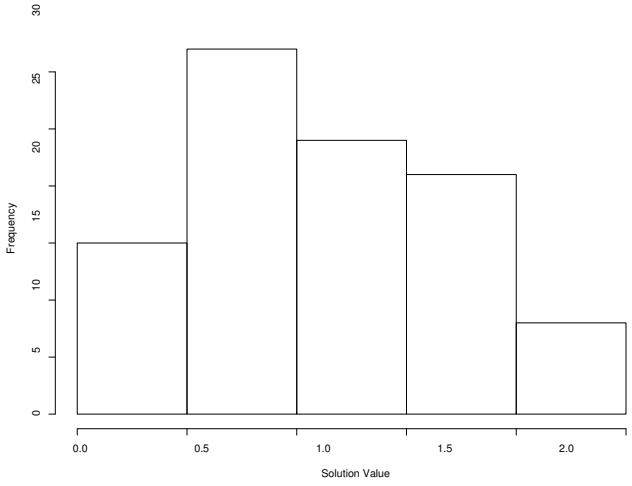

Figure 7: Distribution of 100 best solutions from trials of the hybrid algorithm with a fixed parameter set (except varying seed). Binning via Scott's Rule.

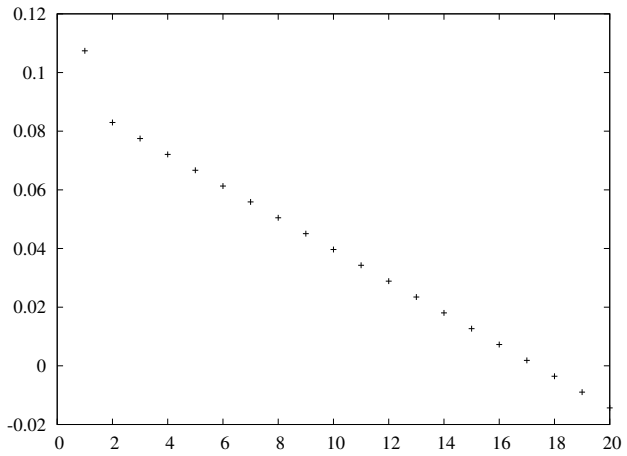

Figure 8: Residuals for best fit (out of 100 trials) to the input/output mapping problem ($f(x) = x$) for $x = 1, 2, 3, ..., 20$).

x	f(x)		antibody1	antibody2
1	1		1.103267	0.883816
2	2		2.082888	1.890011
3	3		3.077486	2.896206
...	...		...	...
19	19		18.991054	18.995326
20	20		19.985652	20.001521
$\overline{x} = 10.5$		cost	0.224718)	0.301971)
		seed	1113951307	1113926247

Table 2: First 6 values of the 20 (x,y) sampled point values used to build the solution costs

$[4.992523, -5.565050, 0.994598, 0.093692]$. These mapping values represent the constants in Eqn(2) above. This antibody solution has the residuals as indicated in Fig.8. It is interesting that the residuals display such a high correlation, as this indicates a further area for antibody improvement. Possibly, if the hybrid algorithm had t_{max} increased, the solution cost would be able to decrease in this case.

The model that was learnt is not similar to the expected solution. That is, the expected form of the solution, based on the generated data, is $f(x) = a \cdot x$, but we were giving the hybrid algorithm a vastly different underlying data model assumption (Eqn(2)) in order to investigate the robustness of the hybrid algorithm with respect to underlying incorrect assumptions.

It is interesting, that one experiment of higher cost seems to model the problem much better. For example, antibody2 in Table.2 can be seen to have discovered a much better solution (even though it is of higher cost than antibody1) because it is $[0.313425, -6.967681, 1.006195, -0.122379]$, where the first and fourth co-ordinates are closer to 0.

5. CONCLUSIONS

A number of algorithms were implemented with the goal of developing a hybrid ACO immune-system algorithm. Preliminary results look promising (further experiments are required to more fully cover the space of possible algorithm parameters) as the hybrid algorithm was able to find vastly different models to fit the data (as one would expect). Future extensions include *i)* incorporating the concept of dynamic nest size, and *ii)* investigating other local/global heuristics (e.g. w.r.t. *a)* pheromone levels, *b)* patch generation/elimination, *c)* number of ants based on the number of patches/nodes in the world/graph) that may help improve convergence time and/or solution cost.

6. ACKNOWLEDGMENTS

The author would like to thank Prof. Tony White for teaching the course and Julio Valdés, Fazel Famili and Bob Orchard from the National Research Council (NRC), Institute for Information Technology (IIT), Integrated Reasoning Group for their kind support.

7. REFERENCES

[1] E. Bonabeau, M. Dorigo, and G. Theraulaz. *Swarm Intelligence From Natural to Artificial Systems*. Number ISBN 0-19-513159-2 in Sante Fe Institute Studies in the Sciences of Complexity. Oxford University Press, 198 Madison Avenue, New York, New York 10016, 1999.

[2] L. N. D. Castro and F. J. V. Zuben. The clonal selection algorithm with engineering applications. *Workshop on Artificial Immune Systems and their Applications in Proceedings of Genetic and Evolutionary Computation Conference (GECCO)*, July 2000.

[3] J. Farmer, N. Packard, and A. Perelson. The immune system, adaptation, and machine learning. *Physica D*, 22:187–204, October 1986.

[4] D. Levine. Users guide to the PGAPack Parallel Genetic Algorithm library. *Argonne National Laboratory. Mathematics and Computer Science Division*, January 1996. `http://www.mcs.anl.gov/pgapack.html`.

[5] P. Musilek, A. Lau, M. Reformat, and L. Wyard-Scott. Immune programming. *Dept. of Electrical and Computer Engineering, University of Alberta*, April 2004.

[6] J. W. Tukey. *Exploratory Data Analysis*. 1977.

[7] Z. Wang and B. Feng. Classification rule mining with an improved ant colony algorithm. *AI 2004. Lecture Notes in Artificial Intelligence*, LNAI 3339:357–367, December 2004.

[8] S. Wierzchoń and U. Kużelewska. Stable clusters formation in artificial immune system. *First International Conference on Artificial Immune Systems*, pages 68–75, 2002.