Assessing the Performance of Two Immune Inspired Algorithms and a Hybrid Genetic Algorithm for Function Optimisation

Jon Timmis
Computing Laboratory
University of Kent
Canterbury. Kent. CT2 7NF. UK.
J.Timmis@kent.ac.uk

Camilla Edmonds
Computing Laboratory
University of Kent
Canterbury. Kent. CT2 7NF. UK.
camillaedmonds@hotmail.com

Johnny Kelsey
Computing Laboratory
University of Kent
Canterbury.Kent. CT2 7NF. UK.
jk34@kent.ac.uk

***Abstract*- Do Artificial Immune Systems (AIS) have something to offer the world of optimisation? Indeed do they have any new to offer at all? This paper reports the initial findings of a comparison between two immune inspired algorithms and a hybrid genetic algorithm for function optimisation. This work is part of ongoing research which forms part of a larger project to assess the performance and viability of AIS. The investigation employs standard benchmark functions, and demonstrates that for these functions the opt-aiNET algorithm, when compared to the B-cell algorithm and hybrid GA, on average, takes longer to find the solution, without necessarily a better quality solution. Reasons for these differences are proposed and it is acknowledge that this is preliminary empirical work. It is felt that a more theoretical approach may well be required to ascertain real performance and applicability issues.**

I. INTRODUCTION

The field of Artificial Immune Systems (AIS) has been around for approximately 10 years [3]. There are many algorithms that are being developed within this area and it is hard to not only see the difference between algorithms, but also assess where some perform better than others (applied to the same problem). Indeed, a certain amount of criticism has been levelled at the field of AIS as not having anything new to offer in terms of quality of solutions offered or a niche application: this criticism to some degree is fair. Therefore, we have begun work to assess the usefulness and quality of AIS that are in the literature and identify their viability when compared against standard techniques (and against other AIS themselves). However, within AIS, there are a number of alternative algorithms for a given area: there is no one standard algorithm for AIS, so comparison is a large and time consuming job. To this end, we have began by selecting just two immune inspired algorithms for comparison with a standard hybrid genetic algorithm: aiNET [2] and the B-cell algorithm (BCA) [14]. These are then benchmarked against a standard hybrid genetic algorithm. These two were chosen, for a number of reasons. First, it would seem that aiNET has received a great deal of attention in the AIS literature and has been used in a number of ways [2,10,11] to name a few. Secondly, work in [14] claimed excellent results on a number of functions. Therefore, it seemed a sensible approach to assess these two algorithms first. In addition, we feel that within a new area such as AIS, reproducing results of previously published work is good practice, as often small problems can be highlighted and the research area begins to mature [4.5]. All three algorithms in this paper were re-implemented based on the aforementioned papers to ensure that the results were reproducible.

This paper begins with a brief explanation of all the techniques compared in the paper; a series of experiments and their results are then presented. Comments are offered as to the difference in performance of these algorithms and direction for future research is given. All of the code used for these experiments is available from the authors upon request. Further results and discussions that are left out of this paper can be found in [9]

II. ARTIFICIAL IMMUNE SYSTEMS IN OPTIMISATION

There is a natural parallel between the immune system and optimisation. Whilst the immune system is not specifically an optimiser, the process of the production of antibodies in response to an antigen is evolutionary in nature, and indeed does develop better antibodies to fight the invading item: hence the comparison with optimisation and the location of better solutions. The process of clonal selection describes how the production of antibodies and their maintenance as a memory of past encounters occurs. This has proven to be a source of inspiration for many people in AIS and there have been a number of algorithms developed for optimisation, inspired by the process (all to varying degrees) [3,6].

Other research into the use of AIS and optimisation include work in [19]. In that work, the authors address the

issue of designing a Genetic Algorithm (GA) with improved convergence characteristics, particularly in the field of design constraints, by creating a GA simulation of the immune system. The motivation for their work stems from the fact that genetic algorithms, when applied to design constraints, have been found to be very sensitive to the choice of algorithm parameters, which can ultimately affect the convergence rate of the algorithm. The authors use the idea of antibody/antigen binding to define a complex matching utility to define similarity between design solutions. The model created also simulates the dynamics of the immune system by creating and removing possible new solutions. Some solutions will be more specific to the problem areas, whereas others will be more generalised. However, the authors point out that both specialist and general solutions are important in the context of structural design, so they introduce a control parameter into the algorithm that enables them to control the production of specialist and general case solutions. The authors suggest their algorithm leads to a higher convergence rate when compared to a traditional GA, but indicate the need for further research and application.

The above work focused on a specific search problem in a particular domain, work in [20] adopts a more generic approach to adaptive problem solving by the use of the immune network metaphor. The authors claim the use of a network structure, but do not present the work as such, but simply immune system metaphors including B-cells, T-cells, macrophages and the Major Histocompatibility Complex (MHC). The immune algorithm given in the paper is used to produce adaptive behaviours of agents, which are used to solve problems. The algorithm is then applied to the n-TSP problem, and for small-scale problems achieves good results. The authors also experiment with removing the interaction of the T-cell in the searching algorithm and present convincing results that the effect of the T-cell on performance is significant, as the solutions found with using the T-cell result in lower cost solutions overall.

Of minor relevance to this work is utilising AIS as an approach to dynamic function optimisation (this paper is concerned with static optimisation). With respect to immune inspired approaches to the dynamic function optimisation problem, there have been fewer attempts in the literature. The first attempt appears to be [21] who proposed a simple immune inspired algorithm based on the clonal selection principle and immune network hypothesis. Additionally, recent work in [22] compared the performance of clonal selection approaches and evolutionary strategies on dynamic function optimisation. Work in [15] however, utilises the BCA algorithm described in this paper, but on more complex moving targets with more than one optimal solution.

A. opt-AiNET

The aiNET algorithm is a discrete immune network algorithm that was developed for data compression and clustering [1], and was then extended to create the algorithm opt-aiNET, which was then applied to optimisation [2]. The aiNET algorithm has subsequently been developed further and applied to areas such as bioinformatics [7] and even modeling of simple immune responses [8].

Opt-aiNET, proposed in [2], evolves a population of cells, which consists of a network of antibodies (considered as candidate solutions to the function being optimised). These undergo a process of evaluation against the objective function, clonal expansion, mutation, selection and interaction between other members of the network. In essence, opt-AiNET creates a memory set of antibodies that represent (over time) the best candidate solutions to the objective function. Opt-aiNET is capable of either unimodal or multimodal optimisation and can be characterised by five main features:

- The population size is dynamically adjustable;
- It demonstrates exploitation and exploration of the search space;
- It determines the locations of multiple optima;
- It has the capability of maintaining many optima solutions;
- It has defined stopping criteria.

Given these definitions, attention can now be turned to the algorithm itself. Figure 1 below outlines the pseudocode for opt-aiNET.

1. *Initialisation*: create an initial random population of network antibodies;
2. *Antigenic presentation*: for each antigenic pattern, do:
   - 2.1 *Clonal selection and expansion*: for each network element, determine its affinity with the antigen presented. Select a number of high affinity elements and reproduce (clone) them proportionally to their affinity;
   - 2.2 *Affinity maturation*: mutate each clone inversely proportional to affinity. Re-select a number of highest affinity clones and place them into a clonal memory set;
   - 2.3 *Metadynamics*: eliminate all memory clones whose affinity with the antigen is less than a pre-defined threshold;
   - 2.4 *Clonal interactions*: determine the network interactions (affinity) among all the elements of the clonal memory set;
   - 2.5 *Clonal suppression*: eliminate those memory clones whose affinity with each other is less than a pre-specified threshold;
   - 2.6 *Network construction*: incorporate the remaining clones of the clonal memory with all network antibodies;

3. *Network interactions*: determine the similarity between each pair of network antibodies;
4. *Network suppression*: eliminate all network antibodies whose affinity is less than a pre-specified threshold;
5. *Diversity*: introduce a number of new randomly generated antibodies into the network;

Figure 1 – opt-AiNET algorithm [3]

In step 1, the initial population consists of N network cells (randomly created). The representation employed is that of real valued shape space and each cell within the network represents a candidate solution in a real valued vector. During steps 1-4 each network cell undergoes a process of clonal expansion (reproduction) and affinity maturation (mutation and selection). Clones of each cell are mutated according to the affinity of the parent cell. The fitness represents the value of the function for the specific candidate solution. The affinity proportional mutation is performed according to the following equation:

$$c' = c + \alpha N(0.1) \quad (1)$$

$$\alpha = (1/\beta) \exp(-f^*)$$

where α is the amount of mutation, c is the parent cell, c' is the mutated clone of c, N (0,1) is a Gaussian random variable of zero mean and standard deviation of 1, β is a parameter that controls the decay of the inverse exponential function and f* is the fitness of c normalised in the interval [0..1]. As c' represents a candidate solution, it must be within the range of the function's specified domain. If c' exceeds that, then it is rejected and removed from the population.

The fitness of each clone (and parent cell) is evaluated, then the fittest individual is selected to become a memory cell: the algorithm adopts an elitist approach to achieve this by always selecting the memory cell with the highest affinity. This is an iterative process that continues until the average error value (distance from objective function) stabilises (this must be less than 0.0001). Once stabilisation occurs, the algorithm then proceeds to steps 3 and 4. Network suppression removes any similar or non-stimulated antibodies and antibodies that fall below the pre-determined suppression threshold σ. By removing similar cells, opt-aiNET prevents antibodies clustering on a single peak. This reduces the amount of cells maintained in the population.

It should be noted that the network interactions within opt-aiNET are only suppressive in nature: they do not contribute to the stimulation of the cells in any way. This is counter to the traditional immune network theory by Jerne [9], on which this algorithm is based. However, in the sprit of biologically inspired computing, this is not a major problem, as it is the inspiration people tend to seek rather than faithful models.

B. B-Cell Algorithm

Work in [14] proposed a novel algorithm, called the B-cell algorithm (BCA) which is inspired by the clonal selection process in the immune system. An important feature of the BCA is its use of a uniquue mutation operator, known a *contiguous somatic hypermutation*. The representation employed in the BCA is binary shape space, with each cell employing this encoding representing a candidate solution. Each B-cell within the population are evaluated by the objective function, *g(x)*. After evaluation by the objective function, the vector within a B-cell $\underline{v}$ is cloned to produce a *clonal pool*, *C*. It should be noted that there exists a clonal pool *C* for *each* B-cell within the population and also that all the adaptation takes place within C. The size of *C* is typically the same size as the population *P* (population size) (but this does not have to be the case). Therefore, if *P* was of size 4 then each B-cell would produce 4 clones. In order to maintain diversity within the search, a single clone is selected at random and each element in the vector undergoes a random change: subject to a certain probability. This is likened by the authors to the metadynamics of the immune system (a technique also employed in aiNET), but within the BCA a separate random clone is produced, rather than utilising an existing one. Each B-cell $\underline{v}' \in C$ is then subjected to a novel contiguous somatic hypermutation mechanism. The BCA uses a distance function as its stopping criterion for the empirical results presented below: when it is within a certain prescribed distance from the optimum, the algorithm is considered to have converged. If the optimum is unknown, then a measure of how far the optimum located so far is employed, and if no progress is made over a certain number of iterations, the search is terminated. The BCA is outlined in Figure 2.

1. Initialisation: create an initial random population of individuals *P*.
2. Antigenic Presentation: for $\underline{v} \in P$:
    2.1 *Clonal Selection and Expansion*:
    evaluate $g(\underline{v})$; then clone each *B-cell*: clone $\underline{v}$ and place in clonal pool *C*;
    2.2 Metadynamics: randomly select a clone $\underline{c}$ in *C*; randomise the vector;
    2.3 Affinity Maturation: forall $\underline{c}$ in *C*, apply the contiguous somatic hypermutation operator;
    then evaluate each clone by applying $g(v)$; if a clone has higher affinity than its parent B-cell $\underline{v}$,
    then $\underline{v} = \underline{c}$;

3.0 Cycle: repeat until a certain stopping criterion is met.

Figure 2 - Outline of the B-Cell Algorithm [based on 14]

The unusual feature of the BCA is the form of the mutation operator. This operates by subjecting *contiguous* regions of the vector to mutation. The biological motivation for this is as follows: when mutation occurs on B-cell receptors, it focuses on complementarity determining regions, which are small regions on the receptor. These are sites that are primarily responsible for detecting and binding to their targets. In essence a more focused search is undertake as in the contiguous mutation operator, rather than selecting multiple random sites for mutation, a random site (or *hotspot*) is selected within the vector, along with a random length; the vector is then subjected to mutation from the hotspot until the length of the contiguous region has been reached. This is in contrast to the method employed by aiNET and the local search functions in hybrid GA's whereby although multiple mutations take place, they are uniformly distributed across the vector, rather than being targeted at a contiguous region

III. HYBRID GENETIC ALGORITHMS

Hybrid genetic algorithms (HGAs) have, over the last decade, become almost standard tools for function optimisation and combinatorial analysis. According to [16], real-world business and engineering applications are typically undertaken with some form of hybridisation between the GA and a specialised search. The reason for this is that HGAs generally have an improved performance over traditional GAs, as has been demonstrated in such diverse areas as vehicle routing [17] and multiple protein sequence alignment [18].

As an example, within a HGA a population P is given as candidates to optimise an objective function $g(x)$. Each member of the population can be thought of as a vector $\underline{v}$ of bit strings of length $l = 64$ (to represent double-precision floating point numbers, although this does not have to be the case) where $\underline{v} \in P$ and P is the population. Hybrid genetic algorithms employ an extra operator working in conjunction with crossover and mutation, which improves the fitness of the population. This can come in many different guises: sometimes it is specific to the particular problem domain; when dealing with numerical function optimisation, the HGA is likely to employ a variant of local search. The basic procedure of a HGA is given in Figure 3. The local search mechanism functions by examining the neighbourhood of the fitness individuals within a given landscape of the population. This allows for a more specific search around possible solutions that results in a faster convergence rate to a possible solution. The local search typically operates as outlined in

Figure 4 Notice that there are two distinct mutation rates utilised: the standard genetic algorithm typically uses a very low level of mutation, and the local search function $h(x)$ uses a much higher one. This method of hybridising a GA is adopted as the model for the HGA used in this paper.

1 Initialisation: create an initial random population *(P)* of individual's $\underline{v}$;
2. Fitness evaluation: For all $\underline{v} \in P$ evaluate fitness of P($\underline{v}$) with objective function $g(x)$;
- 2.1 *Selection and crossover*: Select n number of fittest individuals and with probability p perform crossover between selected individuals;
- *2.2 Mutation*: subject t number of individuals of the population to a low level of mutation with an equally low probability;
- *2.3 Utilise hybrid function*: subject s members of the population to a hybrid search technique $h(x)$; if a higher-fitness member results, return this to the population;

3. Cycle: repeat from step (a) until a certain stopping criterion is met.

Figure 3 - Generic Hybrid GA algorithm (based on [14])

Select: copy $\underline{v}$ to $\underline{v}'$;
- *Explore neighbourhood*: apply mutation to $\underline{v}'$ with certain probability;
- *Generate number of mutations*: Subject $\underline{v}'$ to mutation:
- *Generate mutation sites*: Randomly select sites on $\underline{v}'$ and perturb bit string;

Fitness Evaluation: if $g(\underline{v}') > g(\underline{v})$, replace $\underline{v}$ so that $\underline{v}'$ in P;

Figure 4 - Example of local search mechanism for a HGA (taken form [14])

Table IV highlights the main differences between the three approaches. One main difference is that aiNET adopts an adaptive population size, whereas the BCA and HGA do not. This has the advantage of not having to specify in advance what population size you require (although an initial population size is required) and the algorithm produces enough members of the search space as it requires locating the solution. In addition, selection mechanisms vary in that aiNET selects n best plus employs an elitist mechanism, whereas both the HGA and BCA do not select n best.

IV. RESULTS

A. Experimental Protocol

In order to ascertain differences in performance, two metrics were tested. These were the number of evaluations taken to obtain a solution and the quality of the solution obtained. This is not only to be consistent with work in [14] (to allow for direct comparison) but is also standard benchmark criteria. A suitable number of functions (ranging in dimensions) were then selected from the established literature on which to perform these tests. These were taken from [12] and [13]. New implementations of the three algorithms were created and tested to ensure that results were consistent with previously published work. The aiNET, the BCA and HGA algorithms were then applied to these functions once this was established. In addition, the parameters for all algorithms were taken from previously published work: aiNet [2], BCA [14] and HGA [12] and are shown in Table I. It should be noted that where vectors consisted of bit strings of length 64 (i.e. double-precision floating point numbers) no Gray encoding was used. Each experiment was run for 50 iterations and the results averaged over the runs. Small populations for the BCA and HGA were found to be most effective (based on empirical evidence) and the reader is referred to [23] for further results on this. Additionally, the values presented for aiNET were empirically the best performing parameters for these functions. The reader is referred to [2] for further analysis of the sensitivity of aiNET to parameters.

TABLE I. ALGORITHM PARAMETERS

AINET	
Parameter	Value
Initial population size	20
Suppression threshold	0.2
Number of clones generated	10
Percentage of random new cells each iteration	40%
Scale of affinity proportion selection	100
BCA	
Initial Population size	4
Clonal pool	4
New random elements each iteration	1
HGA	
Initial Population size	4
Crossover rate	0.6
Mutation	0.001
(h)	δ in *{2,3,4,5}*

B. Comparative Results

Table III provides a set of results averaged over 50 runs for the functions being optimised. Each algorithm was executed until either the minium or maximum value was found, or 500 iterations had passed. All results reported presented with standard deviations where greater than zero. The columns in each table represent (in order from left to right): the function number, the minimum or maximum possible value for that function, the next three colums show the value located by each algorithm for that function, and the final three columns show the number of evlautions taken to achive that minium result. The number of evalutions is the cummulative value for the number of times an individual in the population called the evalute function method.

As can be seen from Table III, all three algorithms perform well at finding optimal solutions for the majority of functions. Therefore, in terms of the metric for quality of solutions there seems little to distinguish the three algorithms.

However, when the number of evaluations are taken into account, there are significant differences in the number taken to obtain the solution. The difference in values observed between algorithms were much greater than was first expected. For example, for F1 opt-aiNET and the HGA have similar numbers of evaluations (taking into account standard deviations), but for all other functions in this test scenario opt-aiNET compared against the HGA consistently employs fewer evaluations, but the quality of the locating the optimum value are the same. When compared to the BCA, in all but one case, F5, the BCA consistently performs fewer evaluations than both the opt-aiNET and HGA. What is of noticeable interest is the large standard deviation in the number of evaluations for each technique. It would seem that both AIS approached suffer from a large standard deviation, when compared to the HGA. Although the HGA still has a significant deviation. When the results were investigated in some depth, it would seem that all algorithms often locate the optimum solutions quite rapidly, just not reliably. Some thought has been given as to why these are so large. When the individual results are inspected, then occasionally, opt-aiNET reaches the optimum value very quickly (still not as fast as the BCA), thus causing such a large deviation across the vast majority of functions. Quite why the algorithm should do this is not clear, but what is clear is that all the algorithms can perform well, just not consistently well. This is somewhat true for all stochastic algorithms, but such a large standard deviation is unusual. Another thought may be that the high levels of mutation that occur in all three systems, may contribute to this feature.

C. Discussion

The authors of [14] pointed out the main reasons why they felt that the BCA performed significantly fewer evaluations that the HGA. In essence, their argument was one based on the nature of the mutation function. They argued that the contiguous nature of the mutation operator, combined with the random selection of where the mutation operation begins and ends, lends itself well to escaping local minima. Whilst this work did not investigate that claim, experimental work investigating this is ongoing.

Another important question can be asked: why does aiNET have such a large number of evaluations? The answer to this may lie in the fact that the population size is not fixed and grows to cope with the nature of the problem. This leads in turn to more members of the population being evaluated against the objective function: hence the increase in number of evaluations. Certainly from the experiments undertaken so far, this would seem to be the case. For example, for F5 the network size grows to an average of 172 cells (compared to the BCA and HGAs 4), and for F7 the network size can reach in excess of 650 individuals. Further experiments were also undertaken to see if opt-aiNET could locate as good solutions as the HGA and BCA using smaller initial size networks (with the hope to finish with a smaller network at the end). However, this was not the case as opt-aiNET seems relatively insensitive to the initial size and still evolves networks to fit the problem [9]. These results would tend to indicate that the opt-aiNET algorithm, while biologically appealing in some ways, does not perform as well as one might hope on these functions. On the contrary, opt-aiNET has shown itself to be very effective at data clustering and reduction [3], and therefore may well be better suited for that purpose, rather than for optimisation. These results, to some degree will also be relevant to CLONALG [2]. The opt-aiNET algorithm is based heavily on the CLONALG system and it would not be unreasonable to assume that CLONALG, to some degree, will suffer the same problems. However, this was not confirmed experimentally.

However, we recognise that these arguments are empirical, and maybe a more theoretical answer is required and in part we feel that there will never be a definitive answer to what is a good AIS for problem *x* or problem *y*, until a sound theoretical basis is developed. AIS is ripe for such a development, and indeed is where these authors intend to focus, rather than on continuous empirical evidence.

V. SUMMARY AND CONCLUSIONS

This paper has presented a small empirical study on two immune inspired algorithms and a GA for function optimisation. The motivation for the research was to establish (empirically) the usefulness (or not) of AIS approaches when compared to more traditional evolutionary approaches. Whilst this is not a complete survey of such immune algorithms (such a survey is way outside the scope of such a paper) the research has highlighted certain weakness of the aiNET algorithm, in terms of computational expense, when compared to another immune inspired algorithm and traditional evolutionary approach. The BCA papers (at least for these functions) to perform well in terms of quality of solution and number of evaluations when compared to both the HGA and aiNET. The possible reason for this is the nature of the mutation function, which endows the BCA with the ability to escape local optima solutions more rapidly. However, this needs to be traded of against the large standard deviation which is apparent in all three approaches: this clearly needs further investigation.

Further work is clearly required: this study is far from complete. One is tempted to undertake further empirical work, and indeed this is required to some degree. Other experiments that will be undertaken are running each algorithm for the same number of iterations and assessing number of evaluations so far and quality of solution obtained so far. In addition, experiments would include testing on more ‘real world’ data sets to see how well each algorithm coped with the messy real-world, rather than the more ‘pure’ toy problems presented here.

However, there is a further point to make and this is concerning the usefulness, or indeed, point of adopting the immune inspired approach. Clearly, there is some empirical evidence to suggest that the AIS approach is not a GA approach and that differences do exist, and in some cases results are favourable for, the AIS approach. However, this is a small study and therefore it is impossible for any more general conclusions to be drawn. What is really required is a two pronged approach. The first is to tackle more real world problems where the more traditional approaches much as GAs have failed to deliver satisfactory results. If this were to be tackled by an AIS approach and a good result be obtained, then the usefulness or not of the AIS approach would be established (at least for that problem). To date, we are not aware of such a system. The second, is the fact that a more theoretical understanding of how AIS algorithm performs is required. Adopting certain methodologies from mathematics, such as non-liner dynamics analysis (used for identifying attractors in systems) may be of benefit. This way, one will be able to identify certain properties and performance restrictions on algorithms that at present is not possible. Therefore, our approach from now, will be this two pronged tactic and we hope that results will be possible that will allow us to conclude if there is something different and useful about the AIS approach.

REFERENCES

[1] De Castro, L.N and Von Zuben, F. (2001). "aiNET: An Artificial Immune Network for Data Analysis", in Data Mining: A Heuristic Approach. Abbas, H, Sarker, R and Newton, C (Eds). Idea Group Publishing.

[2] De Castro, L.N and Timmis, J. (2002) An Artificial Immune Network for Multimodal Function Optimisation. Proc. Of IEEE World Congress on Evolutionary Computation. Pp. 669-674

[3] De Castro, L.N and Timmis, J. (2002) Artificial Immune Systems: A New Computational Intelligence Approach. Springer-Verlag.

[4] Timmis, J and Neal, M. (2001). A Resource Limited Artificial Immune System for Data Analysis. Knowledge Based Systems, 14(3-4):121-130.

[5] Knight, T and Timmis, J (2001). AINE: An Immunological Approach to Data Mining. In Cercone, N, Lin, T and Wu X. (Eds) IEEE International Conference on Data Mining. Pp. 297-304, San Jose. CA.

[6] De Castro, L. N. & Von Zuben, F. J. (2002), Learning and Optimisation Using the Clonal Selection Principle. IEEE Transactions on Evolutionary Computation, Special Issue on Artificial Immune Systems, **6**(3), pp. 239-251.

[8] Jerne, N (1975). Towards a Network theory for the Immune System. Annals of Immunology., Inst. Pasture.

[9] Edmonds, C. (2003) Artificial Immune Networks and Multimodal Optimisation. MSc Thesis. University of Kent. Canterbury.UK.

[10] Bezerra, B and De Castro, L.N. (2003) Bioinformatics data analysis using an artificial immune network. Proceedings of ICARIS 2003, eds. Timmis, J., Bentley, P. and Hart, E. Lecture Notes in Computer Science 2787, pp. Springer-Verlag, 2003.

[11] De Castro, L.N. (2003). The Immune response of an Artificial Immune Network (aiNET). In the proceedings of the Congress on Evolutionary Computation. Pp 1273-1280. Canberra. Australia.

[12] Andre, J., Siarry, P. and Dognon, T. (2001). An improvement of the standard genetic algorithm fighting premature convergence in continuous optimisation. *Advances in Engineering Software*. **32**. p. 49-60, 2001.

[13] Eiben, A and van Kemenade, C. (1995). Performance of multi-parent crossover operators on numerical function optimization problems Technical Report TR-9533, Leiden University.

[14] Kelsey, J and Timmis, J (2003). Immune Inspired Somatic Contiguous Hypermutation. In E. Cantú-Paz et al, editor, *Genetic and Evolutionary Computation Conference - GECCO 2003*, volume 2723 of*Lecture Notes in Computer Science*, Chicago. USA., July 2003. Springer-Verlag.

[15] Kelsey, J, Timmis, J and Hone, A (2003). Chasing Chaos. In R. Sarker, R. Reynolds, H. Abbass, T. Kay-Chen, R. McKay, D Essam, and T. Gedeon, editors, *Proceedings of the Congress on Evolutionary Computation*, pages 413-419, Canberra. Australia, December. IEEE.

[16] Goldberg, D. and Voessner, S. (1999) optimizing global-local search hybrids. *Proceedings of the Genetic and Evolutionary Computation Conference*. Pp 220-228. Orlando, Florida, USA,

[17] Berger, J., Sassi, J and Salois, M. (1999). A Hybrid Genetic Algorithm for the Vehicle Routing Problem with Time Windows and Itinerary Constraints. In Proceedings of the Genetic and Evolutionary Computation Conference, pp. 44--51, Orlando, Florida, USA.

[18] Nguyen, H. Yoshihara, I., Yamamori, M and Yasunaga, M. (2002) A parallel hybrid genetic algorithm for multiple protein sequence alignment. Proceedings of the 2002 Congress on Evolutionary Computation CEC2002. Pp. 309-314. IEEE Press.

[19] Hajela, P., & Yoo, J. S. (1999), "Immune Network Modelling in Design Optimization", In *New Ideas in Optimization*, D. Corne, M. Dorigo & F. Glover (eds.), McGraw Hill, London, pp. 203-215.

[20] Toma, N., Endo, S. & Yamada, K. (1999), "Immune Algorithm with Immune Network and MHC for Adaptive Problem Solving", *Proc. of the IEEE System, Man, and Cybernetics*, IV, pp. 271-276.

[21] Gaspar, A. and Collard, P. (1999) From GAs to Artificial Immune Systems: Improving Adaptation in Time Dependent Optimization. *Proceedings of the Congress on Evolutionary Computation*. Pp 1859-1866. Peter J. Angeline and Zbyszek Michalewicz and Marc Schoenauer and Xin Yao and Ali Zalzala (Eds).

[22] Walker, J and Garrett, S. (2003). Dynamic Function Optimisation: Comparing the Performance of Clonal Selection and Evolutionary Strategies. Lecture Notes in Computer Science 2787. Pages 273-284. Timmis, J., Bentley, P. and Hart, E. (Eds).

TABLE II. FUNCTIONS EMPLOYED FOR EXPERIMENTATION

Function ID	Function	Parameters
F1	$f(x) = 2(x-0.75)^2 + sin(5\pi x = 0.4\pi) - 0.125$	0<= x ,+ 1
F2	$f(x) = -\sum$ (j= 1..5) [j *sin* ((j + 1)x + j)]	-10 <= x <= 10, h = 10,
F3	$f(x,y) = a(y - bx^2 + cx - d)^2 + h(1-f) cos(x) + h$	a = 1, b = 5.1/4Π2, c=5/Π, d = 6, f = 1/8Π, -5 <= x <= 10, 0 <= y <=15
F4	$f(x,y) = \sum$ (j = 1..5) j cos[(j + 1) x + j]	-10 <= x <= 10 and –10 <= y <= 10
F5	$\sum$ (j = 1..5) j cos[(j + 1) y + j] + β[(x + 1.4513)^2 + (y + 0.80032)^2	as above but β = 1
F6	$f(x,y) = x^4/4 - x^2/2 + x/10 + y^2/2$	=10 <= x <= 10 and –10 <= y <= 10
F7	$f(x,y) = \sum$ (j = 1..5) j cos[(j + 1) x + j] $\sum$ (j = 1..5) j cos[(j + 1) y + j]	-10 <= x <= 10 and –10 <=y <= 10

TABLE III. COMPARING PERFORMANCE ON FUNCTIONS FOR ALL THREE ALGORITHMS

F(x)	Optimum	Optimum Found			No. Eval g(x)		
		BCA	AiNET	HGA	BCA	Opt-AiNET	HGA
F1	-1.12	-1.08±.04	-1.12	-1.12	3016±2252	6717 ±.538	6081±4471
F2	-12.03	-12.03	-12.03	-12.03	1219±767	41419 ± 25594	3709±2397
F3	0.40	0.40	0.39	-.40	4921±31587	6346 ± 4656	30583±28378
F4	-186.73	-186.73	-180.83	-186.73	46433±31587	363528 ± 248161	78490±6344
F5	-186.73	-186.73	-173.16	-186.73	426360±32809	346330 ± 255980	76358±11187
F6	-0.35	-0.91	-0.26	0.99	2862±351	54703 ± 29701	12894±9235
F7	-186.73	-186.73	-186.73	-186.0	14654±5277	50875 ± 45530	52581±19095

TABLE IV. SUMMARISING THE MAIN SIMILARITIES AND DIFFERENCES BETWEEN BCA, HGA AND OPT-AINET

Algorithm	Diversity	Selection	Population
BCA	Somatic Contiguous mutation	Replacement	Fixed Size
HGA	Point mutation, cross over and local search	Replacement	Fixed size
Opt-aiNET	Affinity proportional somatic mutation	Replacement by *n* fittest clones	Flexible population size