Immune Inspired Somatic Contiguous Hypermutation for Function Optimisation

Johnny Kelsey and Jon Timmis

Computing Laboratory, University of Kent
Canterbury. Kent. CT2 7NF. UK
{jk34,jt6}@kent.ac.uk

Abstract. When considering function optimisation, there is a trade off between quality of solutions and the number of evaluations it takes to find that solution. Hybrid genetic algorithms have been widely used for function optimisation and have been shown to perform extremely well on these tasks. This paper presents a novel algorithm inspired by the mammalian immune system, combined with a unique mutation mechanism. Results are presented for the optimisation of twelve functions, ranging in dimensionality from one to twenty. Results show that the immune inspired algorithm performs significantly fewer evaluations when compared to a hybrid genetic algorithm, whilst not sacrificing quality of the solution obtained.

1 Introduction

The problem of function optimisation has been of interest to computer scientists for decades. Function optimisation can be characterised as, given an arbitrary function, how can the maximum (or minimum) value of the function be found. Such problems can present a very large search space, particularly when dealing with higher-dimensional functions. Genetic algorithms (GAs) though not initially designed for such a purpose, however, they soon began to grow in favour with researchers for this task. Whilst the standard GA performs well in terms of finding solutions, it is typical that for more complex problems, some form of hybridisation of the GA is performed: typically, an extra search mechanism is employed as part of the hybridisation, for example hill climbing, to help the GA perform a more effective local search near the optimum [10].

In recent years, interest has been growing in the use of other biologically inspired models: in particular the immune system, as witnessed by the emergence of the field of Artificial Immune Systems (AIS). AIS can be defined as adaptive systems inspired by theoretical immunology and observed immune functions and principles, which are applied to problem solving [5]. This insight into the immune system has led to an ever increasing body of research in a wide variety of domains. To review the whole area would be outside the scope of this paper, but work pertinent to this paper is work on function optimisation [4], extended with an immune network approach in [6] and applied to multi-modal optimisation.

Other germane and significant papers include [19], where work there is considered on multi-objective optimisation. However, work proposed in this paper varies significantly in terms of population evolution and mutation mechanisms employed.

This paper presents initial work into the investigation of immune inspired algorithms for function optimisation. A novel mutation mechanism has been developed, loosely inspired by the mutation mechanism found in B-cell receptors in the immune system. This coupled with evolutionary pressure observed in the immune system, leads to the development of a novel algorithm for function optimisation. Experiments with twelve different functions have shown the algorithm to perform significantly fewer evaluations when compared to a standard hybrid GA, whilst maintaining high accuracy on the solutions found.

This paper first outlines a hybrid genetic algorithm which might typically be used for function optimisation. Then there follows a short discussion on immune inspired algorithms which outlines the basis of the theoretical framework underpinning AIS. The focus of the paper then turns to the novel B-cell algorithm, followed with the presentation and initial analysis of the first empirical results obtained. Conclusions are drawn and future research directions are explored.

2 Hybrid Genetic Algorithms

Hybrid genetic algorithms (HGAs) have, over the last decade, become almost standard tools for function optimisation and combinatorial analysis: according to Goldberg et. al., real-world business and engineering applications are typically undertaken with some form of hybridisation between the GA and a specialised search [10]. The reason for this is that HGAs generally have an improved performance, as has been demonstrated in such diverse areas as vehicle routing [2] and multiple protein sequence alignment [16]).

As an example, within a HGA a population P is given as candidates to optimised 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 1. 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 described in figure 2. 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, so we have $\delta << \rho$. This process outlined in figure 2. 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 individuals $\underline{v}$;
   a) Fitness evaluation: $\forall \underline{v} \in P$: evaluate fitness of $P(\underline{v})$ with objective function $g(x)$;
   b) Diversity
      i. Selection and crossover: Select $n$ number of fittest individuals and with probability $p$ perform crossover between selected individuals;
      ii. Mutation: subject $t$ number of individuals of the population to a low level of mutation with an equally low probability;
   c) 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;
   d) Cycle: repeat from step (a) until a certain stopping criterion is met.
```

Fig. 1. Generic Hybrid GA algorithm

```
1. Select: copy $\underline{v}$ to $\underline{v}'$;
2. Explore neighbourhood: apply mutation to $\underline{v}'$ with probability $\rho$;
   a) Generate number of mutations: Subject $\underline{v}'$ to mutation: $N_{mut} = f(\rho)$;
   b) Generate mutation sites: for $N_{mut}$, randomly select sites on $\underline{v}'$ and perturb bit string;
3. Fitness Evaluation: if $g(\underline{v}') > g(\underline{v})$, replace $\underline{v}$ so that $\underline{v}' \in P$;
```

Fig. 2. Example of local search mechanism for a HGA

3 Artificial Immune Systems

There has been a growing interest in the use of the biological immune system as a source of inspiration to the development of computational systems [5]. The natural immune system protects our bodies from infection and this is achieved by a complex interaction of white blood cells called B Cells and T Cells. Essentially, AIS is concerned with the use of immune system components and processes as inspiration to construct computational systems. This insight into the natural immune system has led to an increasing body of work in a wide variety of domains. Much of this work emerged from early work in theoretical immunology [13], [8] and where mathematical models of immune system process were developed in an attempt to better understand the function of the immune system. This acted as a mini-catalyst for computer scientists, examples being work on on computer security [9] and virus detection [14]. Researchers realised that, although the computer security metaphor was a natural first choice for AIS, there are many other potential application areas that could be explored such as machine learning [18], scheduling [12] and optimisation [4].

Recent work in [5] has proposed a framework for the construction of AIS. This framework can described in three layers. The first layer is one of representation of the system, this is termed shape space and define the components for the system. A typical shape space for a system may be binary, where elements within each component can take either a zero or one value. The second layer is one of affinity measures: this allows for the measurement of the *goodness* of the component when measured against the problem. In terms of optimisation, this would be in terms of how well the values in the component performed with respect to the function being optimised. Finally, immune algorithms control the interactions of these components in terms of population evolution and dynamics. Such basic algorithms include negative selection, clonal selection and immune network models. These can be utilised as *building blocks* for AIS and augmented and adapted as desired. At present, clonal selection based algorithms have been typically used to build AIS for optimisation. This is the approach adopted in this paper. Work in this paper can be considered as an augmentation to the framework in the area of immune algorithms, rather than offering anything new in terms of representation and affinity measures.

3.1 An Immune Algorithm for Optimisation

Pertinent to work in this paper is work in [4]. Here the authors proposed an algorithm inspired by the workings of the immune system, in a process known as clonal selection. There are other examples of immune inspired optimisation such as [11], however these will not be discussed here. The reader is directed to [5] for a full review of these techniques. Clonal selection is the process by which the immune system is said to respond to invading organisms (pathogens, which then become antigens). The process is conceptually simple: the immune system is made up of cells known as T-cells and B-cells (all of which have receptors on them which are capable of recognising antigens, via a binding mechanisms analogous to a lock and key). When an antigen enters the host, receptors on B-cells and T-cells attach themselves to the antigens. These cells become stimulated through this interaction, with B-cells receiving stimulation from T-cells that attach themselves to similar antigen. Once a certain level of stimulation is reached, B-cells begin to clone at a rate proportional to their affinity to the antigen. These clones undergo a process of affinity maturation: this is achieved by the mutation of the clones at a high rate (known as somatic hypermutation) and selection of the *strongest* cells, some of which are retained as memory cells. At the end of each iteration, a certain number of random individuals are inserted into the population, to maintain an element of diversity.

Results reported for CLONALG (CLONal ALGorithm), which captures the above process, seem to indicate that it performs well on function optimisation [4]. However, from the paper it was hard to extract an exact number of evaluations and solutions found, as these were not presented other than in graphical form. Additionally, a detailed comparison between alternative techniques was never undertaken, so it has proved difficult to fully assess the potential of the algorithm.

The work presented in this paper (undertaken independently of and contemporaneously to the above work) is a variation of clonal selection, which applies a novel mutation operator and a different selection mechanism, which has been found to greatly improve on optimisation performance on a number of functions.

4 The B-Cell Algorithm

This paper proposes a novel algorithm, called the B-cell algorithm (BCA), which is also inspired by the clonal selection process. An important feature of the BCA is its use of a unique mutation operator, known as *contiguous somatic hypermutation.* Evidence for this in the immunological literature is sparse, but such examples are [17], [15]. Here the authors argue that mutation occurs in *clusters* of regions within cells: this is analogous to contiguous regions. However, in the spirit of biologically inspired computing, it is not necessary for the underlying biological theory to be proven, as computer scientists are interested in taking inspiration from these theories to help improve on current solutions. As will be shown the BCA is different to both CLONALG and HGAs in a number of ways. The BCA and motivation for the algorithm will now be discussed.

The representation employed in the BCA is one of a N-dimensional vector of 64-bit strings (as in the HGA above), known as Binary Shape Space within AIS, which represents bit-encoded double-precision numbers. These vectors are considered to be the B-cells within the system. Each B-cell within the population are evaluated by the objective function, $g(x)$. More formally, the B-cells are defined as a vector $\underline{v} \in P$ of bit strings of length $l = 64$ where P is the population. Empirical evidence indicates that an efficient population size for many functions is low in contrast with genetic algorithms; a typical size would be $\sharp P \in [3..5]$. The BCA *can* find solutions with higher P, but it converges more rapidly to the solution (using less evaluations of $g(x)$) with a smaller value for P. Results were obtained regarding this observation, but are not presented in this paper.

After evaluation by the objective function, 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 (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, one clone is selected at random and each element in vector undergo a random change, subject to a certain probability. This is akin to the *metadynamics* of the immune system, a technique also employed in CLONALG, but here 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 precise form of this mutation operator will be explored in more detail below.

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. The BCA is outlined in figure 4.

```
1. Initialisation: create an initial random population of individuals $P$;
2. Main loop: $\forall \underline{v} \in P$:
   a) Affinity Evaluation: evaluate $g(\underline{v})$;
   b) Clonal Selection and Expansion:
      i. Clone each B-cell: clone $\underline{v}$ and place in clonal pool $C$;
      ii. Metadynamics: randomly select a clone $\underline{c} \in C$; randomise the vector;
      iii. Contiguous mutation: $\forall \underline{c} \in C$, apply the contiguous somatic hypermutation operator;
      iv. Affinity Evaluation: evaluate each clone by applying $g(\underline{v})$; if a clone has
          higher affinity than its parent B-cell $\underline{v}$, then $\underline{v} = \underline{c}$;
3. Cycle: repeat from step (2) until a certain stopping criterion is met.
```

Fig. 3. Outline of the B-Cell Algorithm

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 undertaken. This is in contrast to the method employed by CLONALG and the local search function $h(x)$, whereby although multiple mutations take place, they are uniformly distributed across the vector, rather than being targeted at a contiguous region (see figure 4). Contrastingly, as also shown in figure 4, the contiguous mutation operator, rather than selecting multiple random sites for mutation, a random site (or *hotspot*) is chosen within the vector, along with a random length; the vector is then subjected to mutation from the hotspot onwards, until the length of the contiguous region has been reached.

5 Results

Both the HGA and BCA were tested on a number of functions ranging in complexity from one to twenty dimensions, taken from [1] and [7]. It was not possible to obtain results for all functions for the CLONALG, but results for certain functions were taken from [4] for comparative purposes. In total twelve functions were tested. The parameters for the HGA were derived according to standard heuristics, with a crossover rate of 0.6 and a mutation rate of 0.001: the local search function $h(x)$ incorporated a mutation rate of $\delta \in \{2, 3, 4, 5\}$ per vector. The BCA had a clonal pool size equal to the population size. It should be noted

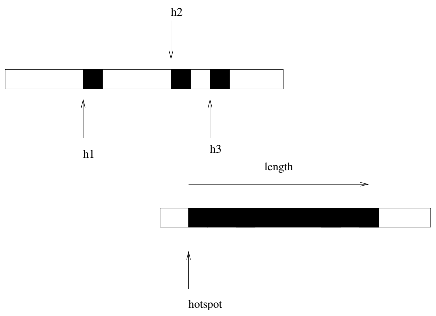

Fig. 4. Multiple-point and contiguous mutation

that all vectors consisted of bit strings of length 64 (i.e double-precision floating point numbers) and no Gray encoding was used on either the HGA or BCA.

Each experiment was run for 50 iterations and the results averaged over the runs. The functions to be optimised are given in table 1. Some of the functions may seem quite simple e.g. **f1**, **f9** with one and two dimensions respectively. However, **f12** is of twenty dimensions. An interesting characteristic of function **f11** is the presence of a second best minimum away from the global minimum. Function **f12** has a product term introducing an interdependency between the variables; this is intended to disrupt optimisation techniques that work on one function variable at a time [7].

5.1 Overview of Results

When monitoring the performance of the algorithms, two measures were employed: these were the quality of the solution found, and the number of evaluations taken to find the solution. The number of evaluations of the objective function is a measure adopted in many papers for assessing the performance of an algorithm; in case the algorithm does not converge on the optimum, the distance measure can give an estimate of how proximity to the solution. Table 2 provides a set of results averaged over 50 runs for the optimised functions. It it noteworthy that the results presented are for a population size of only 4 individuals, in order to allow for direct comparisons to be made; it should also be noted that results were obtained for population sizes ranging from 4 to 40 for both algorithms. It was found that the performance difference between the two algorithms was similar as the population size was increased. As the population sizes increased for both algorithms, the number of evaluations increased, with occasional effect on the quality of the result obtained (in terms of quality of solution found). As can be seen from table 2 both the hybrid GA and BCA perform well in finding the optimal solutions for the majority of functions. Notable exceptions are **f7** and **f9** where neither algorithm found a minimal value. In terms of the metric for quality of solutions then there seems little to distinguish the

Table 1. Functions to be Optimised

Function ID	Function	Parameters
f1	$f(x) = 2(x - 0.75)^2 + \sin(5\pi x - 0.4\pi)$ - 0.125	$0 \leq x \leq 1$
f2 (Camelback)	$f(x,y) = (4 - 2.1x^2 + \frac{x^4}{3})x^2 + xy + (-4 + 4y^2)y^2$	$-3 \leq x \leq 3$ and $-2 \leq x \leq 2$
f3	$f(x) = -\sum_{j=1}^{5}[j \sin((j+1)x + j)]$	$-10 \leq x \leq 10$
f4 (Branin)	$f(x,y) = a(y - bx^2 + cx - d)^2 + h(1 - f)\cos(x) + h$	$a = 1, b = \frac{5.1}{4\pi^2}, c = \frac{5}{\pi}$, $d = 6, f = \frac{1}{8\pi}, h = 10$ $-5 \leq x \leq 10$, $0 \leq y \leq 15$ $0 \leq y \leq 15$
f5 (Pshubert 1)	$f(x,y) = \sum_{j=1}^{5} j \cos[(j+1)x + j]$	$-10 \leq x \leq 10$ and $-10 \leq y \leq 10$ and $\beta = 0.5$
f6 (Pshubert 2)	$\sum_{j=1}^{5} j \cos[(j+1)y + j]$ $+\beta[(x + 1.4513)^2 + (y + 0.80032)^2$	as above but $\beta = 1$
f7	$f(x,y) = x \sin(4\pi x) - y \sin(4\pi y \pi) + 1$	$-10 \leq x \leq 10$ and $-10 \leq y \leq 10$
f8	$y = \sin^6(5\pi x)$	$-10 \leq x \leq 10$ and $-10 \leq y \leq 10$
f9 (quartic)	$f(x,y) = \frac{x^4}{4} - \frac{x^2}{2} + \frac{x}{10} + \frac{y^2}{2}$	$-10 \leq x \leq 10$ and $-10 \leq y \leq 10$
f10 (Shubert)	$f(x,y) = \sum_{j=1}^{5} j \cos[(j+1)x + j]$ $\sum_{j=1}^{5} j \cos[(j+1)y + j]$	$-10 \leq x \leq 10$ and $-10 \leq y \leq 10$
f11 (Schwefel)	$f(\overrightarrow{x}) = 418.9829n - \sum_{i=1}^{n} x_i \sin(\sqrt{\lvert x_i \rvert})$	$-512.03 \leq x_i \leq 511.97, n = 3.$
f12 (Griewangk)	$f(\overrightarrow{x}) = 1 + \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod \cos(\frac{x_i}{\sqrt{i}})$	$n = 20$ and $-600 \leq x_i \leq 600$

two algorithms. This at least confirms that the BCA is performing sensibly on the functions. However, when the number of evaluations are taken into account, then a different picture emerges. These are highlighted in the table 2 and are presented as a compression rate, so the lower the rate, the fewer the number of evaluations the BCA algorithm performs when compared to the HGA. As can be seen from the table, for the majority of the functions reported, the BCA performed significantly fewer evaluations on the objective function than the HGA, but without compromising quality of the solution.

Table 2. Averaged results over 50 runs, for a population size of 4. Standard deviations are given where it was non-zero

f(x)	Min.	Minimum Found		No. Eval. of g(x)		Compression Rate
		BCA	HGA	BCA	HGA	
f1	-1.12	-1.08(±.49)	-1.12	1452	6801	21.35
f2	-1.03	-1.03	-0.99(±.29)	3016	12658	23.81
f3	-12.03	-12.03	-12.03	1219	3709	32.87
f4	0.40	0.40	0.40	4921	30583	16.09
f5	-186.73	-186.73	-186.73	46433	78490	59.16
f6	-186.73	-186.73	-186.73	42636	76358	55.84
f7	1	0.92	0.92(±.03)	333	870	38.28
f8	1	1.00	1.00	132	484	27.27
f9	-0.35	-0.91	-0.99(±.29)	2862	15894	18.01
f10	-186.73	-186.73	-186	14654	52581	27.87
f11	0	0.04	0.04	67483	131147	51.46
f12	1	1	1	44093	80062	55.07

The difference between the number of evaluations is striking. The BCA takes fewer evaluations to converge on the optimum in every case, as the percentage difference in number of evaluations illustrates. On average, it would appear that the BCA performs at least half as many evaluations as the HGA. Further experiments need to be done in comparison with other techniques, in order to further gauge evaluation performance. This is outside the scope of this paper, but is earmarked for future research.

Clearly, the BCA is not performing like the HGA. When compared to the CLONALG results, it should be noted that CLONALG also found optimal solutions for **f7** but the number of evaluations was not available.

5.2 Why Does the BCA Have Fewer Evaluations?

The question of why the BCA converges on a solution with relatively few evaluations of the objective function is one which has not yet been fully explored as part of this work, but is clearly a major avenue for investigation. It is possible that the performance of this algorithm is problem dependant (as is the case with GA's) and that the mutation operator is specifically well suited to the nature of the data representation.

It is possible that the responsibility for rapid convergence lies with the contiguous somatic hypermutation operator. Consider a fitness landscape with a number of local optima and one global optimum. Now consider a B-cell that is trapped on a local optimum; a purely local search mechanism would be unable to extricate the B-cell, since that would mean first moving to a point of lower fitness. If the mutation regime were limited to a small number of *point mutations*, it would only be able to explore its immediate neighbourhood in the fitness landscape, and so it is unlikely that it would be able to escape the local optimum.

However, the random length utilised by the contiguous somatic hypermutation operator means that it is possible for the B-cell to explore a much wider area of the fitness landscape than just its immediate neighbourhood. The B-cell may be able to jump off of a local optimum and onto the slopes of the global optimum. In much the same way, the contiguous somatic hypermutation operator can also function in a more narrow sense, analogous to local search, exploring local points in the fitness space, depending on the value of length.

Despite their intuitive appeal, these are far from formal arguments; more work will need to be undertaken to verify this hypothesis.

5.3 Differences between HGA, BCA, and CLONALG

It is important to identify, at least at a conceptual level, differences in these approaches. It should be noted that, although the BCA is clearly an evolutionary algorithm, the authors do not consider it to be a genetic or hybrid genetic algorithm: a canonical GA employs a deliberately low mutation rate, and emphasises crossover as the primary operator. Similarly, the authors do not consider the BCA to be a memetic algorithm, despite superficial similarities. It is noted that a more rigorous analysis of differences is required, but that has been earmarked for future research. It is the aim of this section to merely highlight conceptual differences for the reader. Table 3 summarises the main similarities and differences. However, it is worth expanding on these slightly.

Table 3. Summarising the main similarities and differences between BCA, HGA and CLONALG

Algorithm	Diversity	Selection	Population
BCA	Somatic Contiguous mutation	Replacement	Introduction of random B-cell. Fixed size.
HGA	Point mutation, crossover and local search	Replacement	Fixed size.
CLONALG	Affinity proportional somatic mutation	Replacement by n fittest clones	Introduction of random cells, flexible population fixed size memory population

Two major differences are the mutation mechanisms and the frequency of mutation that is employed. Both BCA and CLONALG have high levels of mutation, when compared to the HGA. However, the BCA mutates a contiguous region of the vector, whereas the other two select multiple random points in the vector space. As hypothesised above, this may give the BCA a more focused search, which helps the algorithm to converge with fewer evaluations. It is also noteworthy that neither AIS algorithms employ crossover, as this does not occur within the immune system.

The replacement of individuals within the population also varies between algorithms. Within both the HGA and BCA, when a new clone has been evaluated and is found to be better than an existing member of the population, the existing member is simply replaced with the new clone. Alternatively, in CLONALG a number n of the memory set are replaced, rather than just one. However, it should be noted that within the HGA the concept of a clone does not exist, as crossover rather than cloning is employed. This means that within the BCA there is a certain amount of enhanced parallelism, since copies of the cloned B-cell have a chance to explore the immediate neighbourhood within the vector space, by providing extra coverage of the neighbourhood. In contrast, it is again hypothesised that the HGA loses this extra parallelism through the crossover mechanism.

6 Conclusions and Future Work

This work has presented an algorithm inspired by how the immune system creates and matures B-cells, called the B-cell algorithm. A striking feature of the B-cell algorithm is its performance in comparison to a hybrid genetic algorithm. A unique aspect of the BCA is its use of a contiguous hypermutation operator, which, it has been hypothesised, is responsible for its enhanced performance. A first test would be to use this operator in a standard GA to assess the performance gain (or not) that the operator brings. This will allow for useful conclusions to be drawn about the nature of the mutation operator. A second useful direction for future work would be to further test the BCA against other algorithms and widen the scope and type of functions tested; another would be to test its inherent ability to optimise multimodal functions. It has been noted that CLONALG is suitable for multimodal optimisation [4] as an inherent property of the algorithm; it would be worthwhile evaluating if this is the case for the BCA. Perhaps the most illuminating piece of work would be to test the hypothesis regarding the effect of the contiguous hypermutation operator on convergence of the algorithm.

References

1. Andre, J., Siarry, P. and Dognon, T. An improvement of the standard genetic algorithm fighting premature convergence in continuous optimisation. *Advances in Engineering Software*. **32**. p. 49–60, 2001.
2. Berger, J., Sassi, J and Salois, M. A Hybrid Genetic Algorithm for the Vehicle Routing Problem with Time Windows and Itinerary Constraints, Proceedings of the Genetic and Evolutionary Computation Conference, 1999, 1, 44–51, Orlando, Florida, USA, Morgan Kaufmann. 1-55860-611-4,
3. Burke E.K., Elliman D.G. and Weare R.F., A hybrid genetic algorithm for highly constrained timetabling problems, 6th International Conference on Genetic Algorithms (ICGA'95, Pittsburgh, USA, 15th-19th July 1995), Morgan Kaufmann, San Francisco, CA, USA, pages 605–610, 1995

4. de Castro L. Von Zuben F. Clonal selection principle for learning and optimisation. *IEEE Transactions on Evolutionary Computation.* 2002.
5. de Castro L and Timmis J. Artificial immune systems: a new computational intelligence approach Springer-Verlag. ISBN 1-85233-594-7. 2002
6. de Castro L and Timmis J. An artificial immune network for multimodal optimisation *In 2002 Congress on Evolutionary Computation. Part of the 2002 IEEE World Congress on Computational Intelligence*, pages 699–704, Honolulu, Hawaii, USA, May 2002. IEEE.
7. Eiben, A and van Kemenade, C. Performance of multi-parent crossover operators on numerical function optimization problems Technical Report TR-9533, Leiden University, 1995.
8. Farmer, J.D., Packard, N.H., and Perelson, A. The Immune System, Adaptation and Machine Learning. *Physica*, 1986. **22**(D): p. 187-204
9. Forrest S., Hofmeyr S. and Somayaji S. *Computer Immunology.* Communications of the ACM. 40(10). pages 88–96. 1997
10. Goldberg, D. and Voessner, S. Optimizing global-local search hybrids, *Proceedings of the Genetic and Evolutionary Computation Conference*, 1, 13–17, Morgan Kaufmann, Orlando, Florida, USA, 1-55860-611-4, 220–228, 1999.
11. Hajela, P. and Yoo, J. Immune network modelling in design optimisation. In *New Ideas in Optimisation.* D. Corne, M. Dorigo and F. Glover (eds), McGraw-Hill. pp. 203–215, 1999.
12. Hart, E. and Ross, P. The evolution and analysis of a potential antibody library for use in job-shop scheduling. In *New Ideas in Optimisation.* Corne, D., Dorigo, M. and Glover, F.(eds), p. 185–202, 1999.
13. Jerne, N.K. Towards a network theory of the immune system. *Annals of Immunology*, 1974. **125C**: p. 373–389.
14. Kephart, J. A biologically inspired immune system for computers. *Artificial Life IV. 4th International Workshop on the Synthesis and Simulation of Living Systems.* MIT Press, 1994.
15. Lamlum, H., et. al. The type of somatic mutation at APC in familial adenomatous polyposis is determined by the site of the germline mutation: a new facet to Knudson's 'two-hit' hypothesis. *Nature Medicine*, 1999, **5**: pages 1071–1075.
16. Nguyen, H. Yoshihara, I., Yamamori, M. and Yasunaga, M. A parallel hybrid genetic algorithm for multiple protein sequence alignment, *Proceedings of the 2002 Congress on Evolutionary Computation CEC2002,* 309–314, 2002, IEEE Press.
17. Rosin-Arbesfeld, R., Townsley, F. and Bienz, M. The APC tumour suppressor has a nuclear export function. *Letters to nature*, 2000, **406**: pages 1009–1012.
18. Timmis, J. and Neal, M. A resource limited artificial immune system for data analysis. *Knowledge Based Systems.* **14**(3-4): p. 121–130, 2001.
19. Coello, C. Coello and Cruz Cortes, N. An approach to solve multiobjective optimization problems based on an artificial immune system, *Proceedings of the 1st International Conference on Artificial Immune Systems (ICARIS)* 1, 212–221, 2002