International Journal of Computer Information Systems and Industrial Management Applications.
ISSN 2150-7988 Volume 8 (2016) pp. 163–172
© MIR Labs, www.mirlabs.net/ijcisim/index.html

ImmunoOptiDrone - A case study on the general suitability of purpose of abstract evolutionary approaches

Kevin Downs[1] **and Duncan Anthony Coulter**[2]

[1]Academy of Computer Science and Software Engineering, University of Johannesburg,
Auckland Park, Johannesburg, South Africa
kevindwns@gmail.com

[2]Academy of Computer Science and Software Engineering, University of Johannesburg,
Auckland Park, Johannesburg, South Africa
dcoulter@uj.ac.za

Abstract: **The work investigates the general suitability of purpose of abstract evolutionary approaches by investigating the issues involved with re-factoring an existing, gene expression programming based, drone control system into an immunologically based general purpose optimization system. The re-purposed model is investigated regarding its suitability suitable for use in clonal expansion based optimization contexts made possible due to being freed from the physical constraints of real world drone hardware.**

Keywords: Clonal expansion, gene expression programming, drone control

I. Introduction

This article expands on previously introduced work [1]. This article discusses the issues associated with re-purposing an unrelated previously existing evolutionary model into a more abstract problem domain. Software systems which are based on nature and biological systems are a relatively recent development and as such the approach is somewhat prone to repeating some of the mistakes which have already been addressed in other fields. This process is analogous to the previously accepted biological phenomenon of *ontogeny recapitulating phylogeny*. In this a developing embryo seemed to go through stages in its growth which mirror the evolutionary history of the species as a whole.
In this article an evolutionary flight control software generation system for drone platforms is discussed. The underlying algorithms, architecture, and approach for the evolutionary generation of candidate flight control software is discussed followed by a discussion of the simulation environment which plays a key role during the fitness evaluation of the candidate systems. The use of simulation software is crucial for containing the escalation of costs which would arise through the use of approaches to fitness evaluation other than those performed *in silico*.
A general overview of the adopted evolutionary approach is presented in the following section followed in turn by a discussion of the drone system itself. The adaptation of this model onto the immunological paradigm is then considered by initially describing the specific immune algorithm used and its incorporation into the new model.

II. Gene Expression Programming

All forms of evolutionary computation approaches to problem operate by generating successive populations of candidate solutions and determining the suitability of those candidate solutions to solving the problem under consideration. Their respective performance in solving the problem is used as the basis of a derived fitness value which is then used in selecting a subset of the population to undergo reproduction. This reproduction takes the form of the application of a variety of genetic operators [2] which introduce variation in the population while attempting to ensure those attributes of candidate solution which contribute positively to their performance in the fitness metric are preferentially selected for in the next generation. The fitness function therefore serves to drive the evolutionary process to a point where individuals containing an acceptable fitness value, and by extension a suitable level of performance in the underlying task, are present in the population. Once this threshold is reached the algorithm is usually terminated.
The principle differences between the approaches to evolutionary computation that exist are found in either the representation of an individual within the population or the reproduction method which is used to introduce genetic variety [3]. In traditional Genetic Algorithms, GAs, these individuals within the population are expressed as linear genotypes, that is, some fixed string representation of the behaviour of the individual. Genetic Programming differs in that the genotype is expressed into a tree structure which is executed. The tree structure represents a candidate solution to the problem and the tree structure is subject to evolution

directly. These tree structures are referred to as the chromosomes' phenotype, and the application of genetic operators to these representations is therefore referred to as phenotypic evolution [3]. These genetic operators evolve the phenotype representations of the individual directly.

Gene Expression Programming, GEP, as defined by Ferreira [4], is an approach to evolutionary computation. The principle difference between GEP and other evolutionary computation approaches lies in the representation of the behaviour of an individual and in the approach to evolution. Chromosomes are represented by fixed length strings which, like in GA's, are referred to as the genotype representing that individual. This linear representation allows for easier application of genetic operators such as recombination, transposition, replication and mutation. These genotypes may then be expressed as phenotypes in order to be evaluated. These expression trees are exclusively used for the expression of the behaviour associated with the genotype, and as such no phenotypic evolution takes place [5]. The phenotype is simply evaluated and assigned a fitness.

GEP chromosomes are divided into one or more genes, which themselves are divided into both head and tail sections. The head of the gene is composed of both functions, of a certain arity, and arguments to the functions. The tail of gene is composed of only arguments to the functions present in the head of the gene. This representation mimics open reading frames when considered from a biological perspective. Open reading frames in biology are coding sequences for a gene, where the start of the gene is identified by a starting codon. The gene is then comprised of a number of amino acid codons before ending with a termination codon. In GEP, this start codon is always the start of the gene. The entire symbolic string representing the genotype is therefore considered the open reading frame and in GEP is referred to as a K-expression by Ferreira [5]. The length of these genes is calculated as the sum of the length of the tail, t, and that of the head. The length of the tail of gene is a function of the length of head, h, and the maximum arity of the functions present in the head, n. The length of the tail can therefore be defined as follows in Equation 1:

$$t = h \times (n - 1) + 1 \tag{1}$$

One such benefit of this representation is that during phenotypic expression, all invalid regions of the genotype are discarded. This eliminates the need to search the potentially infinite problem space consisting of invalid programs. Genotypes are expressed through the creation of expression trees and as such each argument that applies to a function within the head of the gene is placed in a position that corresponds to the arity of that function. Chromosomes may therefore contain some values which do not contribute to the candidate solution that they represent. These expression trees are also guaranteed to be correct with regards to the functions that appear within. Fig.1 illustrates the standard approach [5, 6].

There has been a great deal of recent work concerned with improving the standard GEP approach. This section concludes with a discussion of three such recently published works summarizing their contributions and distinguishing them from the work presented in this paper.

The first approach to improving GEP attempts to improve the genomic representation of the k-expression strings themselves [7]. This is done by having the genome define sub-functions instead of the candidate solution directly. Instead these sub-functions are used to construct the candidate solution itself. Due to their presence in the genome these sub-functions are themselves subject to improvement through evolutionary pressure. One of the promising advantages of this approach is that it builds on the key insights of GEP (genotypic / phenotypic separation and that non-coding regions act as reservoirs of diversity) by having the emergent results of the execution of these functions define the candidate solution. This is much closer to the biological analogue in that an organism is not in fact directly defined by its genome rather the genes define the molecular machinery which in turn assembles the organism.

The other two improved GEP variants which are now being discussed aim to combine GEP with artificial immune system concepts. The first of these operates in the problem domain of modelling ordinary differential equations and combines the standard GEP operators with clonal expansion at the operator level before additionally combining the approach with a memetic algorithm to improve local search [8]. The combined approach was found to outperform the standard genetic programming approach (it is worth noting that this comparison is with genetic programming, a purely tree based approach, and not with the standard GEP approach).

The last of the GEP variants under consideration [9] improved upon the original in two major ways. Firstly at the high level algorithm layer a *mirror and reset* mechanism was deployed to purge poorly performing candidate solutions from the population pool as well as to improve the exploration behaviour of the system.

The second improvement is again at the genetic operator level and likewise involves incorporates clonal expansion. It is worth noting that clonal expansion is the most suitable of the artificial immune system algorithms as it is most suitably deployed in optimisation problems as opposed to negative selection which is better suited to categorisation problems. The other immune based approaches each also have their own niche in which they are best deployed.

These approaches to improved gene expression programming differ from the ImmunoOptiDrone investigation in that they are attempts to incorporate such improvements at the algorithmic level. ImmunoOptiDrone, however, takes a more layered approach attempting to focus on the structured re-use of an existing model in a different problem domain.

III. Proposed Model

The core components of the prototype system are shown in Fig.2. The system itself uses an approach to evolutionary computation inspired by GEP called multiple-output GEP, moGEP, as defined by Mwaura and Keedwell [10]. MoGEP was developed as a method of creating control systems for robotic behaviour, specifically in problems such as avoidance and guidance. Chromosomes within MoGEP are comprised of a number of genes which are expressed individually as a number of sub expression trees. Other approaches exist, to which MoGEP is similar, such as Cartesian GP, Parisian GP and Multi Expression Programming (MEP). MoGEP differs from these other approaches in the way in which the

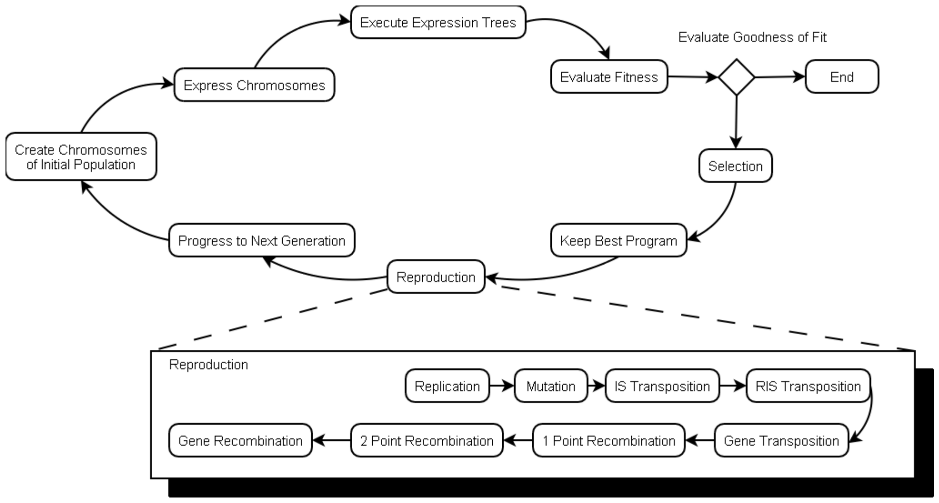

Figure. 1: The standard GEP approach [5, 6]

chromosomes are evaluated. Cartesian GP used both directed graphs, instead of tree-like structures, and also utilizes a separate, potentially identical fitness function that is used to evaluate each gene within the chromosome [10]. MoGEP also differs from Parisian GP in that in MoGEP, chromosomes code for the entire candidate solution to a problem, where in Parisian GP, the chromosome forms a sub-solution to the entire problem [10].

MoGEP has been chosen as the initial model for adaptation owing to the contribution provided by each gene within the chromosome to the solution to the problem. Initially, the MoGEP approach focussed on reading the values associated with sensors placed in positions around a simulated robot and applying forces to simulated motors through the use of various kinematic models. Unlike Mwaura and Keedwells system, in this model the state variables representing the drone in 3d space are simply updated directly based on the evaluation of their phenotype representations. As such, drones maintain state about their position within an environment. These maintained state variables include the acceleration of a drone, the facing angle, as well as their ascent vector. Removing the kinematic models required for the simulation of motors ensures that the actions taken by a drone based on the evaluation of its' phenotype are deterministic. These simplifications allow for the problem domain being modelled to be extended to n-dimensions and provide the basis for the adaptions proposed to this system.

Each of the chromosomes within this system are, as in MoGEP, divided into a number of genes, where each gene is itself comprised of both a head an a tail. Genes are, as in GEP, fixed length strings where the head is comprised of both functions and arguments, as defined in table 1. Functions used within the described system, refered to as I and G and presented in equation 2 and 3 respectively, are both functions with arity 4. I, also referred to as the "If less than or equal" function executes the *true* action of the node that appeared as argument 3 if the sensor value returned from the argument in 1 is less than or equal to the value returned from the argument in position 2. Otherwise, I, executes the *false* action of node 4. G, also referred to as the "If greater than or equal" which executes the *true* action of the node appearing in argument 3 if the sensor value returned from argument 1 is greater than the sensor value returned from argument 2. Otherwise, G performs the *false* action of the node provided as argument 4. The addition of the function to MoGEP has been performed in order to provide for greater genetic diversity with regards to the head portions of genes. This cause is further supported through the use of multiple execution options for a given node within the expression tree. Within the equations presented below, the value of $c(true)$ represents the execution of the true branch of the argument represented by c. The same is true for the second case, where $d(false)$ represents the execution of the false branch of the argument represented by d.

$$I(a,b,c,d) = \begin{cases} c(true), & \text{if } value(a) \leq value(b) \\ d(false), & \text{otherwise} \end{cases} \tag{2}$$

$$G(a,b,c,d) = \begin{cases} c(true), & \text{if } value(a) \geq value(b) \\ d(false), & \text{otherwise} \end{cases} \tag{3}$$

Arguments correspond to both sensor output and an action that is to take place depending on the position in the expression tree at which they are encountered. When an argument is encountered and evaluated within the expression tree for a gene, the value represented by the argument is always returned. This is to ensure that a value tree is executed. In the case of an argument not having an associated action, this ensures that the value is able to be utilized further towards the root of the tree, thereby ensuring that each value contributes to the candidate solution.

Within the arguments presented in table 1, the entries which appear in the *true* column represent the action which takes place when the functions at the root of 4 nodes within a subtree of the decision tree evaluates to true based on the 1st and 2nd nodes. The entries within the *false* column take place when the function evaluates to false.

Based on the discussed arguments and functions, the example phenotype representation presented below (Fig. 3) codes

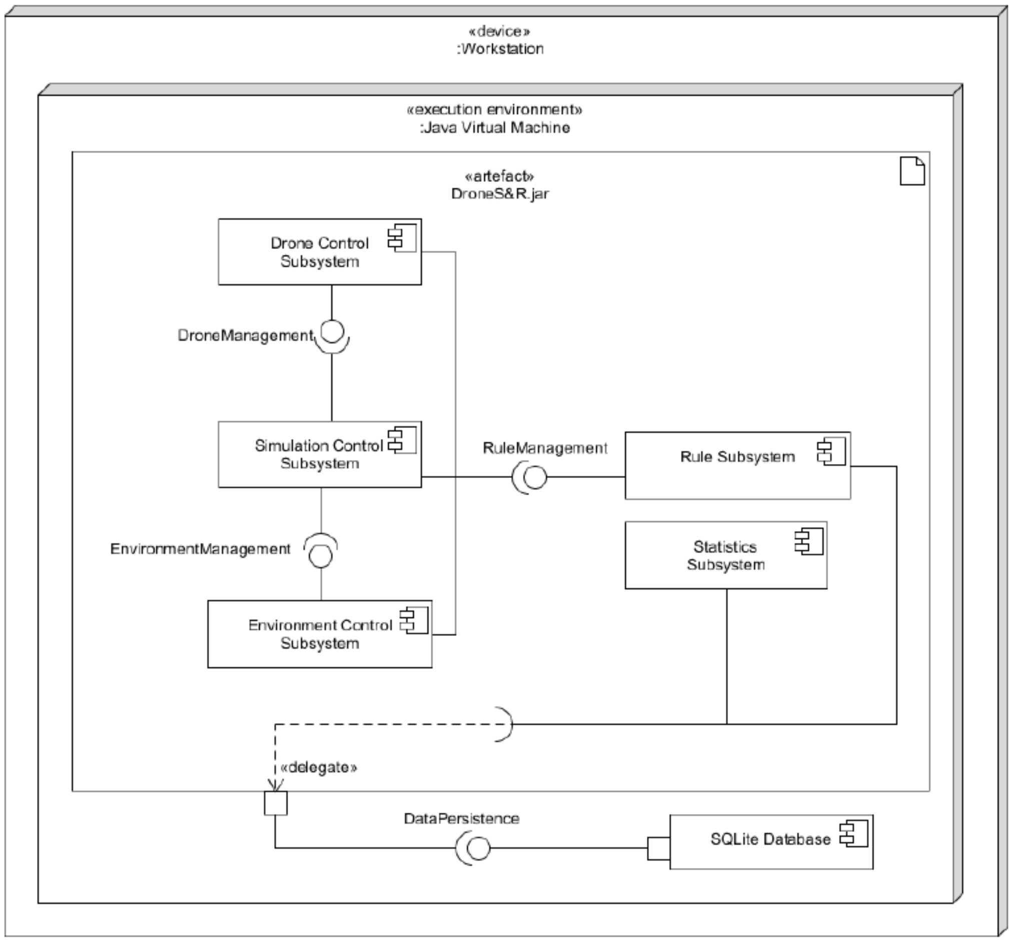

Figure. 2: Component Diagram - Evolutionary Drone Control System [11]

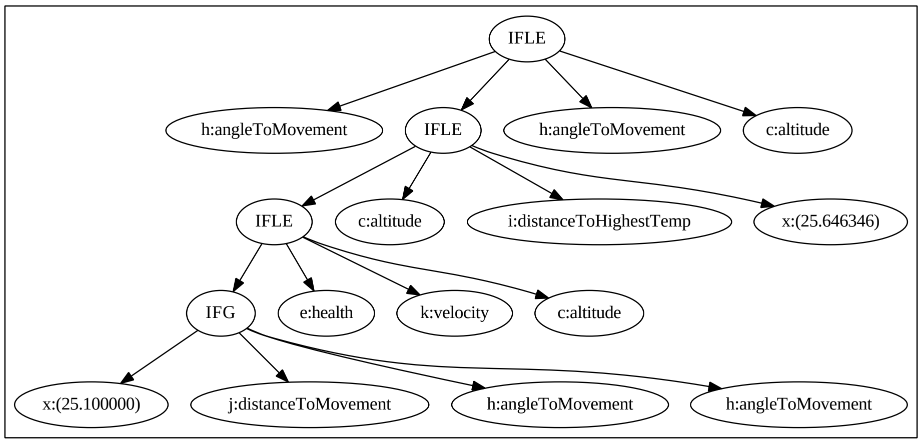

Figure. 3: Example phenotype representation for the individual, IhIIG(25.1)jhhekccixhcifkgcfab

Function Arguments			
Symbol	Returned Value	True	False
l	Ascent vector	Ascend	Descend
f	Facing angle	Rotate left	Rotate right
k	Velocity	Accelerate	Decelerate
a	X Coordinate	-	-
b	Y Coordinate	-	-
c	Altitude (Z Coordinate)	-	-
d	Power remaining for drone	-	-
e	Structural integrity of the drone	-	-
g	Angle to highest detected temperature	Rotate	-
h	Angle to movement detected	Rotate	-
i	Distance to highest detected temperature	Rotate	-
j	Distance to detected movement	Rotate	-
x	Double constant	Constant	Constant

Table 1: Table containing arguments for functions appearing in the head of a gene.

for behaviour where the drone will rotate to face the angle of movement detected if the distance to the moving object, based on results from the relevant sensors, is greater than the constant value, 25.1. Otherwise, the drone will also rotate to the angle to movement. The value returned from the function being executed will then be compared to the structural integrity of the drone. If the structural integrity of the drone is greater than the returned value, the drone will increase accelerate, otherwise it will return the ascent vector of the drone. This returned value is then compared with the current altitude, and if it is less, the drone will return the distance to the highest detected temperature. Otherwise the function will return the constant value 25.646346. This value is finally compared to the angle to movement that is detected. If the angle to movement is less than this returned value, the drone will rotate to face this angle, otherwise it will simply return it's altitude to the root of the decision tree.
In this way, the drone's state is updated based on the decisions taken throughout the execution of the various decision trees which represent these genes within the chromosome. The environment within which these chromosomes are evaluated is represented as a 3-dimensional height map in jMonkeyEngine3, an example of which is provided in 4. Height maps for the proposed system may either be rendered from images or created with use of arrays of values. This allows the height map to be adapted to represent values from a function being modelled.

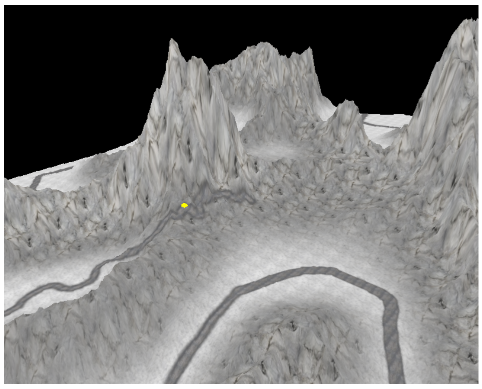
Figure. 4: Rendered environment represented as a 3-d heightmap

The individual genes are then evaluated in order, with each evaluation leading to a change in the state maintained by the drone based on the result of the execution of the phenotype structure. The proposed model evaluates the fitness of a given drone within the population based on a number of criterion. This problem is therefore considered to be a multi-objective problem where the criteria used to evaluate each drone is the distance to the object of interest, the fuel used by the drone, the total number of control cycles used, as well as the damage that has occurred to the drone during the simulation. This system has also been designed to be extensible, where the fitness function may simply be adapted given the problem domain being explored. In this way, the fitness could be considered to be a maximum or minimum value thereby allowing for the adaptation to multiple domains.

IV. Experimental Setup

In order to evaluate the viability of the initial system to the problem at hand, populations were initialized based on a number of parameters. These parameters included the number of individuals, the number of genes that each individual is comprised of, as well as the length of the head for each gene. These individuals were randomly generated and allowed to evolve according to the canonical GEP algorithm as presented by Ferreira [5]. The replacement strategy of the population for subsequent generations is performed between parents and children, where the weakest parent is always replaced by the strongest child. This takes place by way of a Parent-offspring competition [3]. The object to be located was placed in a fixed position for testing purposes and the algorithm was allowed to progress for a total of 50 generations, after which the average and best fitnesses were recorded. Roulette Wheel Selection was implemented for the selection process among the parents of the next generation [3].
In order to contrast between the various genetic operators so that the best configuration given the problem domain may be found, a number of operators have been implemented. These operators are then configured at the start of each execution of the simulation. In terms of Recombination, both One Point and Two Point Recombination were implemented at a selectable rate. Both Gene and Root Transposition were likewise implemented according to Ferreria's definitions of these operators [5]. Owing to the variety of genetic material

within each chromosome representing a solution, mutation is implemented in such a way that each allele within each gene is subject to mutation at a fixed rate. The mutation operator in use examined each allele and randomly selected either a function node, or argument node to replace the value contained within. In cases where floating point valued nodes were subject to mutation, either the value was regenerated or the preexisting value was simply modified by a fixed value, namely ± 0.1. In terms of the fitness function, presented in equation 4, the weights were fixed such that $dWeight = 10.0$, $hWeight = 2.0$, $pWeight = 1.0$, and $cWeight = 0.5$. Therefore, more emphasis was placed on the distance to the object being searched for. The problem can therefore be considered to be a minimization problem.

$$\begin{aligned} fitness(x_i) = &(dWeight * distance(x_i, p)) \\ &+ (hWeight * health(x_i)) \\ &+ (pWeight * power(x_i)) \\ &+ (cWeight * controlCycles(x_i)) \end{aligned} \tag{4}$$

V. Results

Results, as presented in table 2, are classified according to the parameters that were selected for the execution of the algorithm. Let P be the tuple comprised of the parameters for the simulation, such that:

$P = \{nGenes, hLength, mRate, rOp, rRate, tOp, tRate\}$

Where $nGenes$ is the number of genes that each drone is comprised of, $hLength$ is the length of the head of each gene, $mRate$ is the mutation rate, rOp is the recombination operator, $rRate$ is the rate at which recombination occurs, and finally tOp and $tRate$ are the transposition operator and transposition rate respectively. Presented within these results are both the average and best fitnesses at the end of 50 generations.

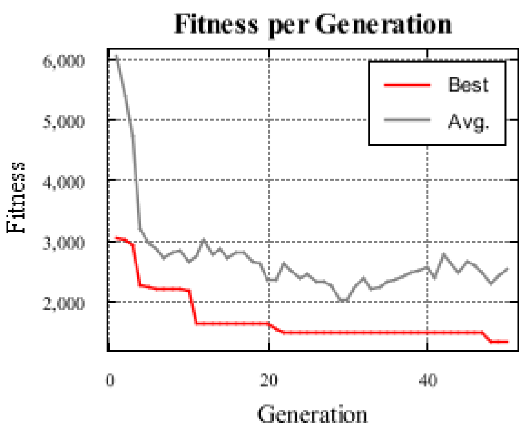

Figure. 5: Average vs. Best Fitness over the course of 50 generations of 30 Drones comprised of 30 genes and a head length of 5, using Roulette Wheel Selection, Uniform Mutation at a rate of 0.02, Two Point Recombination, and Gene Transposition at a rate of 0.01.

As can be seen within table 2, the algorithm is tunable in order to favour either exploration or exploitation within the simulation. This is evident in both the average and best fitnesses found in Test 3 compared to other configurations. This configuration, where each chromosome was comprised of 20 genes, with a head length of 6, resulted in poor convergence with the selected operations and rates, instead favouring exploration. This is in comparison with Test 6, presented in fig. 5, where the best fitness of any configuration is observed. The configuration presented in Test 6 was able to generate the best solution observed throughout the testing process, owing to the increased tendency towards exploitation during the simulation.

VI. ImmunoOptiDrone via Clonal Expansion

The vertebrate adaptive immune response has evolved in order to supplement the fixed innate immune response found in older organisms. Innate immunity provides an unchanging response to predetermined pathogens which have been identified during the evolutionary history of the organism. Adaptive immunity supplements this response by allowing an organism to develop appropriate immune responses to new pathogens encountered within the life-span of the organism. The adaptive immune response is in truth not a single response but rather a multitude of interrelated responses each dealing with a different sub-problem each governed by differing cells and effector molecules. The problem of self / non-self recognition is mediated by T-cell lymphocytes through negative selection while the problem of providing an optimal response is handled by B-cell lymphocytes via clonal expansion's positive selection [12].
In general clonal expansion is most similar to a mutation-heavy evolutionary algorithm. Fig. 6 details the steps involved in clonal expansion. Initially immature B-cells are produced en mass in the bone marrow. Each B-Cell then undergoes somatic hyper-mutation in a specialized variable region of its DNA. This region controls which peptide sequence the lymphocyte will bond to i.e. its affinity to a specific antigen. During maturation clonal expansion is used in order to filter out those lymphocytes which have affinity to peptide sequences which are not present and amplify the response of those lymphocytes whose affinity is for those antigens which are present. In order to do so antigen presenting cells provide samples on their surface (via the major histo-compatibility complex molecule) of those peptides which are present within the organism together with danger signals such as those produced when tissues are damaged. Those lymphocytes which happen to bond with the presented antigens are preferentially duplicated via the process of clonal expansion and a limited subset of those clones have their life-spans increased to serve as memory cells and thereby increase the rapidity of future responses to those antigens [13].
In order to adapt the evolutionary drone control mechanism discussed in this model to the task of function optimisation the following observations need to made. Firstly as the new problem domain is purely virtual physical constraints on drones need not apply. The original evolutionary system would have to heavily penalize drone control software which lead to crash however in the new problem domain this is not such a dire consequence. Additionally drones are not lim-

Results										
Test	nGenes	hLength	mRate	rOp	rRate	tOp	tRate	Best	Avg
1	20	5	0.04	One Point	1.00	Gene	0.01	1836.56	3204.14
2	20	5	0.02	One Point	1.00	Gene	0.01	1905.25	2204.65
3	20	6	0.02	One Point	0.7	Gene	0.01	2456.15	3360.75
4	30	5	0.02	One Point	1.00	Gene	0.01	2304.54	2980.5
5	30	5	0.04	One Point	1.00	Gene	0.01	1950.15	2845.32
6	30	5	0.02	Two Point	1.00	Gene	0.01	1450.31	2745.34
7	30	5	0.04	Two Point	1.00	Gene	0.01	2285.31	3047.51
8	30	5	0.02	One Point	1.00	Root	0.01	1845.65	2403.54
9	30	6	0.04	One Point	1.00	Root	0.01	2206.45	3014.68
10	20	6	0.04	Two Point	0.8	Root	0.02	2050.49	3405.95

Table 2: Table containing the results of various configurations utilizing Roulette Wheel Selection over the course of 50 generations of a population comprised of 30 Drones.

ited to their initial number and be dynamically created and destroyed by the system as needed. The second observation is that the fitness function of the evolutionary drone control system can be adapted so that instead of favouring drones which come closest to victims in need of rescue it instead favours those which maintain a fixed distance above the terrain's surface yet achieve a high elevation. In this way the function being optimised can be substituted for the terrain of the environment. The system can then be mapped onto the clonal expansion algorithm by having the fitness function of the underlying GEP algorithm substitute for affinity in an evolutionary algorithm layered on top of it. Individual lymphocytes are then realized by the drones themselves with new instances brought in and out according to their affinity at fixed intervals. Somatic hyper-mutation does not need to be expressly implemented instead the pre-existing mutation function of the underlying GEP can be used.

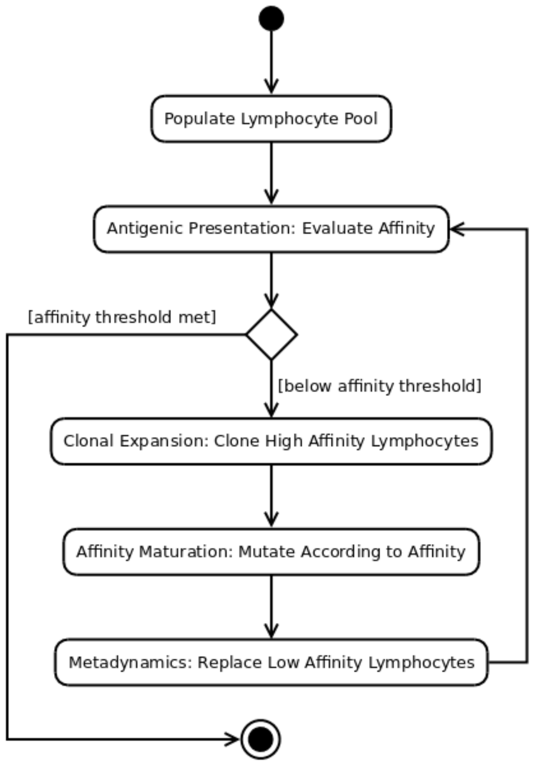
Populate Lymphocyte Pool
Antigenic Presentation: Evaluate Affinity
[affinity threshold met]
[below affinity threshold]
Clonal Expansion: Clone High Affinity Lymphocytes
Affinity Maturation: Mutate According to Affinity
Metadynamics: Replace Low Affinity Lymphocytes

Figure. 6: UML Activity Diagram - Clonal Expansion [12–14]

VII. Experimental Setup of ImmunoOptiDrone

In order to more accurately compare the use of the Clonal Selection algorithm to GEP, a number of considerations have been made. Firstly, the fitness function remains the same as that presented in equation 4. This is so that the results of adapting the system to Clonal Selection may be directly contrasted to the performance of GEP, as in the initial model. The final goal position also remains the same as that of the initial model so that the results presented in table 2 may be compared to the results of the execution of the Clonal Selection algorithm. This is simply an abstraction, as in ImmunoOptiDrone, the final position can also be seen as the highest point in terms of elevation within the simulated environment. The simulated environment may therefore be considered to be affinity landscape in which these drones function. The affinity landscape could therefore represent a function being optimized and in this way may be instead replaced by examining the highest or lowest point within this landscape.

Initially, the population of drones, referred to as lymphocytes, are initialized at a fixed starting position. This fixed starting position was selected so as to provide a baseline by which performance may be measured between the different models. This location is surrounded by local minima and maxima which the chromosomes within the initial model traversed to reach the goal position. This location also remains fixed between executions so that the performance of each execution of the model may be compared to one another. As in GEP, these lymphocytes are allowed to execute their initial randomly generated behaviours. However, in contrast to GEP, the physical constraints of the simulation environment are removed and the lympocytes may freely traverse the affinity landscape irrespective of whether they collide with the terrain or not. These lympocytes continue to execute their behaviours until they have reached the end of their lifespans or they begin to move outside of the domain of the function that defines the affinity landscape in which they function. Once these lymphocytes have reached the end of their lifespan, their final affinity is calculated based on equation 4. Based on this the problem domain can be seen as a minimization problem, where larger distances to the target position indicate a higher affinity in much the same way as GEP and affinity is therefore considered to be the difference between the target position and the final position obtained by the drone. The weights for this affinity calculation

were fixed such that $dWeight = 10.0$, $hWeight = 2.0$, $pWeight = 1.0$, and $cWeight = 0.5$. This allows for a more accurate comparisson between the results obtained for GEP and those obtained with ImmunoOptiDrone. Following this affinity calculation, the best lymphocytes are selected so that they may undergo the process of clonal expansion. That is, those lymphocytes within the population with lower affinities are selected for cloning. The n lympocytes with the best affinities are selected from the population, where $n = populationSize/portion$ and $portion$ is the sample of the population that is to be examined.
These selected lymphocytes are then cloned in relation to the population size and $cRate$ as defined in equation 5.

$$numClones() = populationSize * cRate \qquad (5)$$

Where $cRate$ is the rate at which clones are created. The number of clones generated is therefore in proportion to the initial population size. Higher numbers of clones are generated within larger lymphocyte populations.
These clones then undergo the process of affinity maturation, where these high affinity lymphocytes are mutated in proportion to their affinities. That is, higher affinities indicate larger differences between the final position of these drones and the goal position. This may be adapted to instead be the highest point within the terrain, however, for comparisson purposes this has been selected as the same point being searched for as that in the GEP approach. As the behaviours of these lympocytes is defined as a decision tree, as specified in figure 3, the mutation operators remain the same as that of the GEP approach within the initial model. That is, the original mutation operators remain in use, however, the rate at which mutation occurs is altered depending on the affinity of the lymphocytes being examined. Through empirical study, the function which determines the rate at which mutation occurs has been defined. As can be seen in equation 6, the mutation rate is defined in terms of a lymphocytes affinity, where high affinities in this case indicate worse performance, and $mRate$, where $mRate$ is tunable to allow the algorithm to exhibit either more or less mutation for these cloned lymphocytes.

$$mutationRate(affinity) = \frac{(\frac{affinity}{600} - 1)^2}{mRate} \qquad (6)$$

As can be seen in figure 7, the mutation rate for each lympocyte is in terms of both the affinity for that lymphocyte and the rate of mutation defined as $mRate$. Based on this, it can be seen that higher affinities represent lower performance, and therefore the mutation rate increases in order to improve on the performance of that lymphocyte. These lymphocytes are subject to the original mutation operators of the initial model, that is One Point Recombination and Gene Transposition at these calculated mutation rates. As sexual reproduction does not take place within the clonal selection algorithm, those operators which were present within the initial model have been disabled.
After these lymphocytes have undergone the affinity maturation process, they are allowed to execute in order to calculate their final affinities. Based on these calculated affinities, the lymphocyte clones and the original lymphocyte population are combined in order to select the new population.

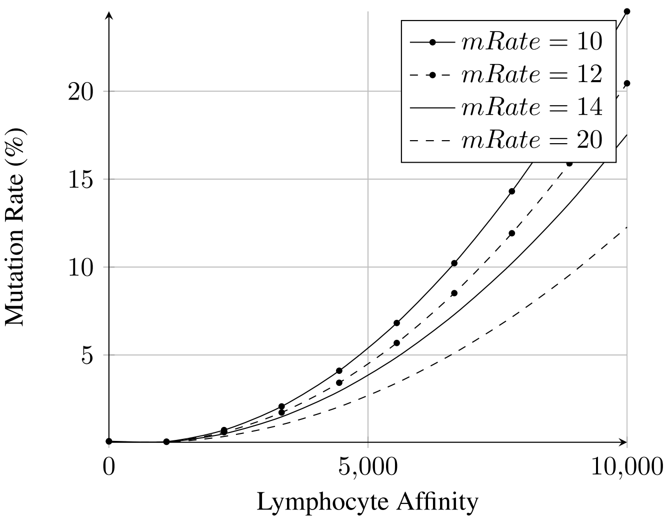

Figure. 7: Percentage of mutation based on a lymphocytes affinity with $mRate = 10$, $mRate = 12$, $mRate = 14$, and $mRate = 20$ as defined in equation 6. Lower affinities represent better performance.

This is performed by selecting the best performing lymphocytes from both of these sets and combining them. The worst performing lymphocytes are then replaced with new stochastically generated lymphocytes and the process continues. These new lymphocytes serve as the method by which randomness is injected into the algorithm and aids in preventing stagnation as the population evolves.

VIII. ImmunoOptiDrone Results

In order to determine the validity of this model a number of tests were executed, the results of which are presented in table 3. The model was initialized with variations in the population size, represented by $pSize$, and as in the initial model, the length of the head for each chromosome, $hLength$, and the number of genes present in the chromosomes, $nGenes$. The simulation was allowed to execute for a total of 50 generations in order to contrast these results to those presented in table 2. After each simulation had executed successully, both the average and best affinity exhibited was recorded. These affinities can be directly compared to the fitness of the drones recorded in the initial model owing to the use of the identical fitness function.
When comparing the perfomance of the initial model, in table 2, to that of ImmunoOptiDrone in table 3, it can be seen that the average affinity of the population of lymphocytes is consistently higher than the average fitnesses presented in the original model throughout each execution of the simulation. This is true for the best affinities of ImmunoOptiDrone as well, where despite attempting to tune the algorithm it is still unable to converge on an ideal solution. It can also be seen that this is the case despite modifications to the population size which was not considered in the original model. That is, within the original model, the population size was fixed at 30 individuals. Owing to the functioning of the clonal selection algorithm in use in ImmunoOptiDrone, the population size was allowed to fluctuate. Initially, these executions were set so that a fixed number of lymphocytes were generated,

Results								
Test	pSize	nGenes	hLength	portion	mFactor	cRate	Best Affinity	Avg. Affinity
1	30	20	6	4	14	0.1	2952.48	5046.53
2	30	20	6	4	12	0.1	1909.05	4951.18
3	30	20	6	4	10	0.1	3032	4882
4	30	20	6	5	10	0.1	3035	4827
5	30	20	6	5	10	0.2	2718	4773
7	20	20	6	5	12	0.2	2444	7428
8	30	20	6	5	14	0.3	2337	5048
9	30	20	6	5	20	0.2	2425	4642
10	30	20	6	5	12	0.2	2672	4743

Table 3: Results of configurations utilizing the ImmunoOptiDrone model over 50 generations.

however, it was observed that the population size fluctuated as the algorithm progressed. This allowed for more diversity within the population compared to the initial model, and the inability of the population to converge and exploit good solutions may also be attributed to this increased diversity within the population. The best performance of this model was achieved in Test 2 which was initialized such that $pSize = 30$, $nGenes = 20$, $hLength = 6$, $portion = 4$, with $mFactor = 12$ and $cRate = 0.1$. The performance of this execution of the ImmunoOptiDrone model is presented in figure 8.

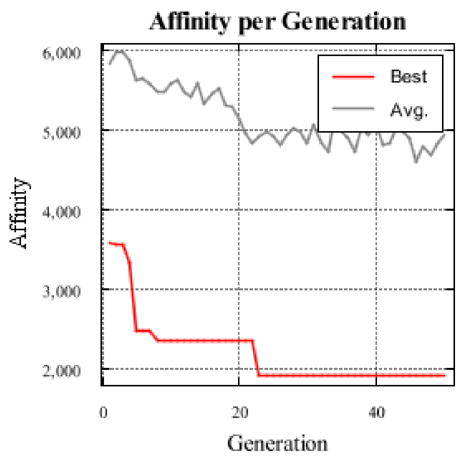

Figure. 8: Average vs. Best Affinities over the course of 50 generations of 30 Drones comprised of 20 genes and a head length of 5 within the ImmunoOptiDrone model. Lower affinities indicate better performance.

The results of the execution presented appear in figure 8, where the average vs. best affinities within the population of 30 lymphocytes over 50 generations is charted, demonstrates the inability of ImmunoOptiDrone to converge. While the average affinity of the population of lymphocytes does indeed decrease, it is still consistently higher than the affinity of the best lymphocyte within the population. When compared to the results of an execution of the initial model, presented in figure 5, the average fitness of the population does decrease and the algorithm is able to capitalize on good solutions. This behaviour was not exhibited by the ImmunoOptiDrone model, where the performance of the best lymphocyte makes large sudden improvements to its performance. These large sudden decreases can be attributed to the increased mutation rate of the clonal selection algorithm when compared to that of GEP. As the mutation rate increases with an increase in affinity, larger modifications are made in order to attempt to minimize the function being optimized. When considering the structure of the chromosomes that define the behaviour of these lymphocytes it may be seen that these large mutations serve to create even larger changes in the behaviours of these drones within the simulation. It may therefore be concluded that ImmunoOptiDrone when configured in the manner presented above, is unsuitable to this problem domain.

References

[1] Downs, KP., and Coulter DA., 2015, ”ImmunoOptiDrone—towards re-factoring an evolutionary drone control model for use in immunological optimization problems”. Advances in Intelligent Systems and Computing, pp. 327–336. Springer Science + Business Media.

[2] Mitchell, M., 1998, ”An introduction to genetic algorithms”. MIT Press, Cambridge, Mass.

[3] Engelbrecht, A., 2007, ”Computational intelligence”. John Wiley & Sons, Chichester, England. pp. 177–187.

[4] Ferreira, C., 2002, ”Gene Expression Programming in Problem Solving”. Soft Computing and Industry, pp. 635–653.

[5] Ferreira, C., 2001, ”Gene expression programming: a new adaptive algorithm for solving problems”. Complex Systems. 13, 2, pp. 87–129.

[6] Coulter, DA., 2007, ”Cellular automata driven self evolution in multi-agent systems”. MSc (Computer Science) dissertation. University of Johannesburg. pp. 17–35.

[7] Jinghui Zhong; Yew-Soon Ong; Wentong Cai., 2016, ”Self-Learning Gene Expression Programming”. IEEE Transactions on Evolutionary Computation, Volume 20. Issue 1. IEEE Conference Publications. pp. 65–80.

[8] Siqing Xue; Jie Wu, 2015, ”Modeling Systems of Ordinary Differential Equations Using Immune Based Gene Expression Programming”. Intelligent Human-Machine Systems and Cybernetics (IHMSC), Volume 1. IEEE Conference Publications. pp. 437–422.

[9] Chao-xue Wang, Jing-jing Zhang, Shu-ling Wu, Chunsen Ma, 2015, ”An improved gene expression programming algorithm based on hybrid strategy”. Biomedical

Engineering and Informatics (BMEI). IEEE Conference Publications. pp. 639–643.

[10] Mwaura, J., Keedwell, E., 2014, ”On using Gene Expression Programming to evolve multiple output robot controllers”. 2014 IEEE International Conference on E-volvable Systems.

[11] Downs, KP., 2015, ”IT00237 Deliverable 10”. University of Johannesburg.

[12] Dasgupta D., and Nino, F., 2008. ”Immunological Computation: Theory and Applications”, CRC Press, Boca Raton, Florida, United States of America, ISBN 978-1420065459.

[13] de Castro L.N., and Timmis J., 2002, ”Artificial Immune Systems: A New Computational Intelligence Approach”, Springer-Verlag London, London, United K-ingdom, ISBN 978-1-85233-594-6.

[14] Coulter, DA., 2014, ”Immunologically amplified knowledge and intentions dimensionality reduction in cooperative multi-agent systems”. PhD (Computer Science) thesis. University of Johannesburg, pp. 77.

Author Biographies

Kevin Downs Kevin was born in South Africa in 1991. He completed his BSc Information Technology degree in 2014 at the University of Johannesburg in South Africa. In 2015 he completed his BSc Honours in information technology degrees also at the University of Johannesburg.

Duncan Anthony Coulter Duncan was born in South Africa in 1983. He completed his BSc information technology in 2003, BSc Honours in computer science in 2004, MSc in computer science in 2008, and PhD in computer science in 2014. All of these were completed at the University of Johannesburg in South Africa. His research interests concern biologically inspired artificial intelligence.