Genetic algorithms for modelling and optimisation

Article *in* Journal of Computational and Applied Mathematics · December 2005

DOI: 10.1016/j.cam.2004.07.034

1 author:

John Mccall
Robert Gordon University

104 PUBLICATIONS **828** CITATIONS

Available online at www.sciencedirect.com
SCIENCE DIRECT®
Journal of Computational and Applied Mathematics 184 (2005) 205–222
JOURNAL OF COMPUTATIONAL AND APPLIED MATHEMATICS
www.elsevier.com/locate/cam

Genetic algorithms for modelling and optimisation

John McCall

School of Computing, Robert Gordon University, Aberdeen, Scotland, UK

Received 27 February 2004; received in revised form 7 July 2004

Abstract

Genetic algorithms (GAs) are a heuristic search and optimisation technique inspired by natural evolution. They have been successfully applied to a wide range of real-world problems of significant complexity. This paper is intended as an introduction to GAs aimed at immunologists and mathematicians interested in immunology. We describe how to construct a GA and the main strands of GA theory before speculatively identifying possible applications of GAs to the study of immunology. An illustrative example of using a GA for a medical optimal control problem is provided. The paper also includes a brief account of the related area of artificial immune systems.

MSC: 68T05; 92B05; 49-04; 92D15

Keywords: Genetic algorithms; Immunology; Optimisation; Evolution

1. Introduction

The purpose of this paper is to provide immunologists and mathematicians interested in immunology with an introduction to genetic algorithms (GAs), as a computational technique with potential applications in mathematical immunology. We begin, in Section 2, with some background on the emergence of GAs as a research discipline, inspired by abstraction from natural evolution. The detailed structure of a GA with simple examples of its component parts is presented in Section 3. In Section 4, we explore the key advances that have been made in the theoretical understanding of how GAs operate, though it is a subject where successful application far outstrips the theory. Potential applications of GAs to the study of immunology are introduced in Section 5, which also includes an illustrative account of an application

E-mail address: jm@comp.rgu.ac.uk.

0377-0427/$ - see front matter
doi:10.1016/j.cam.2004.07.034

of GAs to modelling and optimisation in chemotherapy. Section 5 also contains a brief account of the related area of artificial immune systems. The main conclusions of the paper are presented in Section 6.

2. Background

GAs are a heuristic solution-search or optimisation technique, originally motivated by the Darwinian principle of evolution through (genetic) selection. A GA uses a highly abstract version of evolutionary processes to evolve solutions to given problems. Each GA operates on a population of artificial *chromosomes*. These are strings in a finite alphabet (usually binary). Each chromosome represents a solution to a problem and has a *fitness*, a real number which is a measure of how good a solution it is to the particular problem.

Starting with a randomly generated population of chromosomes, a GA carries out a process of fitness-based selection and recombination to produce a successor population, the next generation. During recombination, parent chromosomes are selected and their genetic material is recombined to produce child chromosomes. These then pass into the successor population. As this process is iterated, a sequence of successive generations evolves and the average fitness of the chromosomes tends to increase until some stopping criterion is reached. In this way, a GA "evolves" a best solution to a given problem.

GAs were first proposed by John Holland [17] as a means to find good solutions to problems that were otherwise computationally intractable. Holland's Schema Theorem [17], and the related building block hypothesis [15], provided a theoretical and conceptual basis for the design of efficient GAs. It also proved straightforward to implement GAs due to their highly modular nature. As a consequence, the field grew quickly and the technique was successfully applied to a wide range of practical problems in science, engineering and industry. GA theory is an active and growing area, with a range of approaches being used to describe and explain phenomena not anticipated by earlier theory [22]. In tandem with this, more sophisticated approaches to directing the evolution of a GA population are aimed at improving performance on classes of problem known to be difficult for GAs [25,16,26]. The development and success of GAs have greatly contributed to a wider interest in computational approaches based on natural phenomena and it is now a major strand of the wider field of Computational Intelligence, which encompasses techniques such as Neural Networks, Ant Colony Optimisation, Particle Swarm Optimisation and Artificial Immunology [13].

Evolution strategies (ESs) developed independently of GAs and began at approximately the same time, in the late 1960s. In common with GA, ESs use a string representation of solutions to some problem and attempt to evolve a good solution through a series of fitness-based evolutionary steps. Unlike GA, an ES will typically not use a population of solutions but instead will make a sequence of mutations of an individual solution, using fitness as a guide [2]. Although of independent origin, the two fields have grown together and evolutionary computation is sometimes now used as an umbrella term for the whole area.

3. The structure of a genetic algorithm

A GA is constructed from a number of distinct components. This is a particular strength because it means that standard components can be re-used, with trivial adaptation in many different GAs, thus easing implementation. The main components are the chromosome encoding, the fitness function, selection, recombination and the evolution scheme.

3.1. Chromosome encoding

A GA manipulates populations of *chromosomes*, which are string representations of solutions to a particular problem. A chromosome is an abstraction of a biological DNA chromosome, which can be thought of as a string of letters from the alphabet {A,C,G,T}. A particular position or *locus* in a chromosome is referred to as a *gene* and the letter occurring at that point in the chromosome is referred to as the *allele value* or simply *allele*. Any particular representation used for a given problem is referred to as the GA *encoding* of the problem. The classical GA uses a bit-string representation to encode solutions. Bit-string chromosomes consist of a string of genes whose allele values are characters from the alphabet {0,1}. For problems where GAs are typically applied, solution sets are finite but so large that brute-force evaluation of all possible solutions is not computationally feasible. It is not uncommon for a GA to be operating on bit strings of length 100, giving a solution space consisting of 2^{100}–10^{30} individuals. The interpretation of these strings is entirely problem dependent. For example, a bit string of length 20 might be used to represent a single integer value (in standard binary notation) in one problem, whereas, in another, the bits might represent the presence or absence of 20 different factors in a complex process. It is a strength of GAs that common representations can be used in this way for a multiplicity of problems, allowing the development of common processing routines and operators and making it faster and easier to apply GAs to new situations. On the other hand, the consequence is that the chromosome encoding alone will contain only limited problem-specific information. Much of the meaning of a specific chromosome for a particular application is encoded in the second component of a GA, the fitness function.

3.2. Fitness

The fitness function is a computation that evaluates the quality of the chromosome as a solution to a particular problem. By analogy with biology, the chromosome is referred to as the genotype, whereas the solution it represents is known as the phenotype. The translation process can be quite complicated. In timetabling and manufacturing scheduling GAs, for example, a chromosome is translated into a timetable or set of scheduled activities involving large numbers of interacting resources. The fitness computation will then go on to measure the success of this schedule in terms of various criteria and objectives such as completion time, resource utilisation, cost minimisation and so on. This complexity is reminiscent of biological evolution, where the chromosomes in a DNA molecule are a set of instructions for constructing the phenotypical organism. A complex series of chemical processes transforms a small collection of embryonic cells containing the DNA into a full-grown organism, which is then "evaluated" in terms of its success in responding to a range of environmental factors and influences.

Here, we present a very simple example, the OneMax problem, which has long been used as a test problem by the GA community. The problem is to simply use a process of evolutionary search to maximise the number of 1s in a bit string of length n. There is therefore a straightforward encoding—all chromosomes are bit strings of length n. The fitness function is simply a count of the number of 1s in a chromosome. So there are 2^n possible chromosomes and each has a fitness ranging from 0 to n. The optimal chromosome for the OneMax problem of length n is clearly the string consisting of n 1s and so it provides an easy-to-implement test case for different approaches to GA design.

In Section 4, we describe the use of a GA to evolve a good multi-drug cancer chemotherapy regime. The problem is to administer a combination of up to D anti-cancer drugs to a patient over n treatment interval, subject to a set of dosage and toxicity constraints, with the overall objective of eradicating a

cancerous tumour. Each chromosome is a bit string of length $4nD$ and represents the dosage of each drug applied in each interval in a straightforward manner. The fitness of a chromosome is calculated from a formula involving a differential equation model of tumour response to chemotherapy and a set of penalties applied for each clinical constraint violated by the treatment. In this case, all problem-specific information is contained in the fitness evaluation. The chromosomes for the chemotherapy problem are indistinguishable from those for the OneMax problem of length $4nD$.

3.3. Selection

A GA uses fitness as a discriminator of the quality of solutions represented by the chromosomes in a GA population. The selection component of a GA is designed to use fitness to guide the evolution of chromosomes by selective pressure. Chromosomes are therefore selected for recombination on the basis of fitness. Those with higher fitness should have a greater chance of selection than those with lower fitness, thus creating a selective pressure towards more highly fit solutions. Selection is usually with replacement, meaning that highly fit chromosomes have a chance of being selected more than once or even recombined with themselves. The traditional selection method used is Roulette Wheel (or fitness proportional) selection. This allocates each chromosome a probability of being selected proportional to its relative fitness, which is its fitness as a proportion of the sum of fitnesses of all chromosomes in the population [15]. There are many different selection schemes. Random Stochastic Selection explicitly selects each chromosome a number of times equal to its expectation of being selected under the fitness proportional method. Tournament Selection first selects two chromosomes with uniform probability and then chooses the one with the highest fitness. Truncation Selection simply selects at random from the population having first eliminated a fixed number of the least fit chromosomes. A full account of the various selection schemes used in the literature and guidance on when it is most appropriate to use them can be found in [1].

3.4. Recombination

Recombination is the process by which chromosomes selected from a source population are recombined to form members of a successor population. The idea is to simulate the mixing of genetic material that can occur when organisms reproduce. Since selection for recombination is biased in favour of higher fitness, the balance of probabilities (hopefully) is that more highly fit chromosomes will evolve as a result. There are two main components of recombination, the genetic operators crossover and mutation. Genetic operators are nondeterministic in their behaviour. Each occurs with a certain probability and the exact outcome of the crossover or mutation is also nondeterministic.

The crossover operator represents the mixing of genetic material from two selected parent chromosomes to produce one or two child chromosomes. After two parent chromosomes have been selected for recombination, a random number in the interval [0,1] is generated with uniform probability and compared to a pre-determined “crossover rate”. If the random number is greater than the crossover rate, no crossover occurs and one or both parents pass unchanged on to the next stage or recombination. If the crossover rate is greater than or equal to the random number, then the crossover operator is applied. One commonly used crossover operator is one-point crossover. A crossover point between 0 and n is chosen with uniform probability. Child chromosomes are then constructed from the characters of the first parent occurring

before the crossover point and the characters of the second parent occurring after the crossover point. We illustrate this on a length 10 bit-string encoding as follows:

Parent one:	1 1 1 0 1 0 0 1 1 0
Parent two:	0 0 1 0 0 1 1 1 0 0
crossover point:	↑
Child one:	1 1 1 0 0 1 1 1 0 0
Child two:	0 0 1 0 1 0 0 1 1 0

There are many alternative forms of crossover operation. One-point crossover generalises straightforwardly to 2- and multi-point crossover operations, where a sequence of crossover points is chosen along the chromosome length and the child chromosomes are constructed from the allele values of the two parents, interchanging at each crossover point. Uniform crossover constructs a child by selecting uniformly between parent allele values at each locus. Algorithms also differ with respect to whether one or more children are created from the crossover operation. After crossover, the resultant chromosome(s) will be passed on to the mutation stage.

Mutation operators act on an individual chromosome to flip one or more allele values. In the case of bit-string chromosomes, the normal mutation operator is applied to each position in the chromosome. A random number in the interval [0,1] is generated with uniform probability and compared to a predetermined "mutation rate". If the random number is greater than the mutation rate, no mutation is applied at that position. If the mutation rate is greater than or equal to the random number, then the allele value is flipped from 0 to 1 or vice versa. Mutation rates are typically very small (e.g., 0.001).

3.5. Evolution

After recombination, resultant chromosomes are passed into the successor population. The processes of selection and recombination are then iterated until a complete successor population is produced. At that point the successor population becomes a new source population (the next generation). The GA is iterated through a number of generations until appropriate topping criteria are reached. These can include a fixed number of generations having elapsed, observed convergence to a best-fitness solution, or the generation of a solution that fully satisfies a set of constraints. There are several evolutionary schemes that can be used, depending on the extent to which chromosomes from the source population are allowed to pass unchanged into the successor population. These range from *complete replacement*, where all members of the successor population are generated through selection and recombination to *steady state*, where the successor population is created by generating one new chromosome at each generation and using it to replace a less-fit member of the source population. The choice of evolutionary scheme is an important aspect of GA design and will depend on the nature of the solution space being searched. A widely used scheme is *replacement-with-elitism*. This is almost complete replacement except that the best one or two individuals from the source population are preserved in the successor population. This scheme prevents solutions of the highest relative fitness from being lost from the next generation through the nondeterministic selection process.

Recently, there has been considerable research interest in estimation of distribution algorithms (EDAs) [25,26]. In an EDA, the source population is used to build a probabilistic model of the distribution of allele values in chromosomes with high fitness scores. The EDA then samples the model to generate chromosomes for the successor population, replacing the traditional recombination component. Typically, the

model building and sampling process is more computationally expensive than traditional recombination. However, EDAs have been shown to outperform traditional GAs on problems where the value of the fitness function is dependent on a high level of interaction (linkage) between allele values at different positions on the chromosome.

3.6. GA design

There are many choices that have to be made in designing a GA for a given application. The choice of encoding will depend on the nature of the problem. Nonbit-string representations are now common, and include sequences of integer or floating point values. As the size of the allele set expands, for example, where the strings consist of floating point numbers, the set of possible chromosomes becomes considerably greater. Many modern (or nonclassical) GAs use a range of representational approaches to ensure that the set of possible chromosomes is a close match for the set of possible solutions of the problem. Having selected an encoding, there are many other choices to make. These include: the form of the fitness function; population size; crossover and mutation operators and their respective rates; the evolutionary scheme to be applied; and appropriate stopping/re-start conditions. The usual design approach is a combination of experience, problem-specific modelling and experimentation with different evolution schemes and other parameters. Several examples of nonclassical GAs can be found in [9] and [21]. Part G of [1] contains several examples of real-world applications. A typical design for a classical GA using complete replacement with standard genetic operators might be as follows:

(1) Randomly generate an initial source population of P chromosomes.
(2) Calculate the fitness, $F(c)$, of each chromosome c in the source population.
(3) Create an empty successor population and then repeat the following steps until P chromosomes have been created.
    (a) Using proportional fitness selection, select two chromosomes, c1 and c2, from the source population.
    (b) Apply one-point crossover to c1 and c2 with crossover rate pc to obtain a child chromosome c.
    (c) Apply uniform mutation to c with mutation rate pm to produce c′.
    (d) Add c′ to the successor population.
(4) Replace the source population with the successor population.
(5) If stopping criteria have not been met, return to Step 2.

In practice, the flexibility and robustness of GAs mean that implementing a first-cut GA for a particular application is usually straightforward using standard design choices. Refining this to the point where an application-specific GA outperforms other available approaches is often more problematic. GAs have proved particularly useful for problems such as scheduling where there are large numbers of constraints and where traditional gradient-based optimisation methods are not applicable.

3.7. Resources

There are a number of resources freely available on the Internet for those interested in applying GAs in their own area. A good place to start is the Genetic Algorithms Archive, maintained by the US Navy

Centre for Applied Research in Artificial Intelligence (http://www.aic.nrl.navy.mil/galist/). The site is a long-established resource for the genetic algorithm and evolutionary computation communities and contains lists of research groups, downloadable software and links to related sites of interest. The Archive also maintains an archive of postings to the EC Digest mailing list (formerly GA-List).

A wide range of downloadable software is available to assist rapid development of GAs. Toolkits are available in many programming languages and vary widely in the level of programming skill required to utilise them. Mathematicians are likely to find GAOT, the Genetic Algorithm Toolbox for Matlab, the easiest way to begin experimenting with GAs. The Toolbox implements a GA as a set of Matlab functions, which can be redefined and reconfigured to suit different applications. GAs using binary and floating point encodings can be defined and a range of standard recombination operators are also supplied. The Toolbox website also provides some example GAs to solve some standard problems. GAOT can be downloaded from http://www.ie.ncsu.edu/mirage/GAToolBox/gaot/.

4. Theoretical approaches to genetic algorithms

Although GAs have enjoyed significant practical success, attempts to establish a theoretical account of their precise operation have proved more difficult. There are two goals for a satisfactory theory of GAs. The first is to explain which classes of problem GAs are particularly suitable for and why. The second is to provide techniques and approaches for optimal design and implementation of GAs, as there are many choices of structure and parameters to be made.

4.1. Schemata and building blocks

The Schema Theorem [17] has stood for a long time as a central result of GA theory. It attempts to explain how the evolutionary processes in a GA can locate optimal or near-optimal solutions, even though they only ever sample a tiny fraction of the set of all possible solutions. A schema is a pattern within a chromosome defined by fixing the values of specific chromosome loci. A schema defines a set of chromosomes, namely all those containing the pattern.

A schema can be specified using bits to define the pattern and instances of a wildcard symbol at places in the chromosome not specified by the pattern. Thus, a schema is a string of symbols from the alphabet {0, 1, *}. For example, a schema for 10-bit chromosomes might be specified using the string

$$01 * * * 100 * *.$$

Chromosomes which belong to this schema include

0110110000,
0100010001,
0111110011.

Chromosomes that do not match the pattern do not belong to the schema and include

1010110000,
1111100000,

all chromosomes belonging to the schema **********.

For a particular schema H, we define the *length* l_H to be the difference of the allele positions of the first and last defined bits of H. The *order* of H is the number of defined bits. For example, the schema 01***100** has 5 defined bits and so has order 5. The last defined bit is in position 8 and the first is in position 1, and so the schema has length $8 - 1 = 7$.

The Schema Theorem describes how schemata featuring in highly fit chromosomes have a greater expectation of propagating through successive populations as a GA evolves and is stated as follows.

Schema Theorem. *Let H be a schema and let* $m_H(i)$ *be the number of chromosomes belonging to H present in population i of an evolving GA. Then the expectation of the number of chromosomes belonging to H in population* $i + 1$, *denoted* $m_H(i + 1)$, *is given by the formula*

$$m_H(i+1) = F_H(i) m_H(i) \left[1 - p_C \frac{l_H}{l-1}\right] [(1 - p_m)^{o_H}],$$

where $F_H(i)$ *is the relative fitness of* H, *defined to be the average fitness of all those chromosomes in the population belonging to* H *divided by the average fitness of all chromosomes in the population*; p_C *is the crossover probability*; p_m *is the mutation probability*.

The formula assumes that the GA uses fitness proportional selection and takes into account the possibility that the genetic operators of crossover and mutation can act to disrupt schema H. It is worth noting that the formula does not include a term for new instances of schema H being introduced to the population by genetic operators and so really gives a lower bound for the expectation.

The Schema Theorem allows one to reason about the ways in which particular patterns are likely to propagate and so gain an understanding about how the performance of a particular GA is affected by the various design choices one must make.

The theorem shows that, all else being equal, those schemata with relative fitness greater than 1 will be increasingly represented in the successor population, whereas those with relative fitness less than 1 will be decreasingly represented. This is purely the effect of selective pressure implemented in the fitness proportional selection process. However, in the presence of genetic operators, other factors in the formula can become very significant. In particular, as the length of H becomes close to $l - 1$, the term relating to crossover, $1 - p_C(l_H/(l-1))$, approaches $1 - p_C$. In other words, schemata with a length approaching the length of the chromosome are almost certain to be disrupted by crossover when it occurs. Also, as the order of H increases, the term relating to mutation, $(1 - p_m)^{o_H}$, decreases exponentially. In other words, schemata involving many defined bits are quite likely to be disrupted by mutation when it occurs.

Consequently, we can conclude that short length, low order, above averagely fit schemata are increasingly sampled in subsequent generations. Such schemata are known as *building blocks* and can be thought of as "partial solutions" in that chromosomes containing them tend to be of higher fitness. Intuitively, a chromosome containing many building blocks will be a near-optimal solution. The Building Block Hypothesis [15] states that GAs achieve near-optimal performance by preferentially selecting and juxtaposing building blocks.

Ideally, the notion of building blocks should be of particular utility when choosing an encoding for a GA. Knowledge of the application domain could be used to ensure that the values of factors known to interact are encoded in neighbouring loci. Also, genetic operators could be refined to disrupt a chromosome only at locations where problem-specific knowledge indicates that it makes sense, thereby increasing the

probability that the evolutionary process will fruitfully juxtapose building blocks and quickly arrive at an optimal solution.

In practice, however, problem knowledge is seldom sufficient to do this. Usually, GAs are applied to problems where solution sets are very poorly understood. There is no guarantee that, for any given problem, an encoding can be found, or even exists, that contains building blocks. Moreover, it is known that, even where building blocks do exist, the BBH breaks down for many theoretical and practical problems because other factors impede the evolution. In order to explore the ways in which GAs process schemata, Mitchell et al. [23,24] defined the Royal Road problems. For these problems, the fitness function is explicitly defined in terms of certain "preferred" schemata allowing hypotheses about schema processing to be directly tested. Mitchell et al. showed that certain alleles can become fixed at suboptimal values simply because they happen to be present in chromosomes that contain preferred schemata. This process is known as hitchhiking, and can impede the juxtaposition of building blocks.

Also, many examples have been constructed of deceptive problems, where "false" building blocks exist—i.e., short order schemata that do provide a fitness benefit but lead the search away from optimal chromosomes. Deceptive behaviour can also be observed in practical applications of GAs (see, for example, [15]).

4.2. The simple GA

In recent years, alternative approaches to GA theory that do not rely on schemata have been investigated. Vose [33] develops a mathematically sophisticated theory of random heuristic search, where a sequence of populations of elements are generated using a nondeterministic transition rule. The space of all possible populations, Ω, is a (high-dimensional) simplex over Z_2 and the theory explores evolution of the population as the trajectory of the RHS through this simplex. A simple genetic algorithm (SGA) is defined to be an example of an RHS where the transition rule can be factored as a composition of selection and mixing (mutation and crossover). A population in the sense of SGA can be thought of as a probability distribution which could be used to generate (*bit-string*) chromosomes. Therefore, it is represented as a vector of probabilities, one corresponding to each possible chromosome. The probabilities provide barycentric coordinates for the location of the population in Ω. For chromosomes of length n, an SGA evolves through a population simplex of dimension 2^n.

SGA theory provides a powerful mathematical framework in which to explore fundamental properties of GAs as evolutionary processes. What is the expected locus in Ω of the SGA at time t? Under what conditions will an SGA converge on a global optimum? How do these conditions relate to design choices when developing a GA, such as mutation and crossover operators and different selection schemes? What are the fixed points of the evolution and to what extent do they correspond with local or global optima?

SGA theory has produced some interesting results, in particular theoretical conditions on the fitness function, selection and mixing operators under which the SGA can be proven to converge. However, as the population size becomes small relative to Ω, the granularity of the set of points in Ω that can be realised by a particular sampled population decreases to the extent that the evolution is forced to diverge from the expected trajectory. In all practical GAs, population size is very small compared to Ω and so the predictions of SGA theory can only be regarded as strongly suggestive of expected GA behaviour. Interestingly, under certain conditions, the SGA exhibits the phenomenon of punctuated equilibria, where long periods of relative stasis are punctuated by sudden changes in location within Ω. This phenomenon

has been observed in practical studies of GAs and, indeed, long periods of stasis followed by sudden evolutionary leaps are also believed to be a feature of the natural evolution of species.

It is also possible to recover the notion of schemata in SGA theory by introducing equivalence relations on Ω, which respect the transition function in the sense that the natural transition function on the space of equivalence classes, $\tilde{\Omega}$, is well defined. In [33] it is demonstrated that, for the crossover component of the transition function to be compatible with the equivalence relation, the equivalence classes must be schemata. This is important because classical theories of GAs assert that they evolve successfully by processing schemata. However, Vose goes on to demonstrate that the proportional selection component of the partition function cannot, in general, be well defined on "nontrivial" schemata, thus suggesting that a full explanation of GA evolution cannot be based solely on schema processing.

4.3. Statistical mechanics

A significant issue in the study of the dynamics of GAs is the high dimensionality of the space of possible populations. This means that any attempt to derive microscopic evolution equations is fraught with computational difficulties. An analogy can be made here with the study of the time evolution of particles in a gas. Whilst, for classical physics at least, the equations of motion are deterministic, the sheer size of the dimension of the phase space means that their solution is computationally infeasible, and so it is impossible to realistically pursue this approach to model the behaviour of a macroscopic quantity of gas. Instead, traditionally, the solution has been to use statistical mechanics to derive macroscopic or bulk quantities that can be used to provide a simpler, though much more manageable, description of the characteristics of the system by averaging over the ensemble of possible states.

There have been several attempts to analyse genetic algorithms using statistical mechanical techniques. In particular, Prugel-Bennet et al. [28] have developed a statistical-mechanics theory of the evolution of GAs that have a fitness function based on the Ising model for low-energy states of a spin glass. This formalism is then used to investigate the dynamics of the GA search. This approach, while promising a highly satisfactory account of GA dynamics, is limited by assumptions about macroscopic properties that may not hold for particular GAs. It is known, for example, that these assumptions are not true for Royal Road GAs.

5. Model fitting and optimisation

The purpose of this section is to explore possible applications of GAs to immunology. We identify some ways in which application of GAs are most likely to be of benefit, illustrating this with an existing GA application in the field of cancer chemotherapy. The section concludes with a description of artificial immune systems. These form a distinct area of research in their own right but are sufficiently related to GAs, and to the general topic of immunology, to require inclusion in this paper.

5.1. Model fitting

In recent years, there have been a number of advances in understanding the dynamics of immune systems in response to viral infection, that have been achieved through a combination of in vivo or in vitro experiment and computational simulation using quantitative theoretical models, sometimes referred to as in silico experimentation. A recent article by Chakraborty et al. [7] provides several examples of

success and argues that the development of complex hierarchies of computer simulations, integrated with experimental approaches, provides the best hope for progress in understanding emergent properties of immune systems in terms of processes over a range of scales from atomic to macroscopic.

However, this agenda presents significant challenges for computational modellers, particularly in the areas of system identification, model validation and computational complexity. Systems need to be sufficiently realistic to provide predictive understanding and this implies complexity. At the same time, complex simulations whose properties are not well understood risk merely replacing one intractable problem with another.

In mathematical immunology, a typical approach is to use low-dimensional systems of coupled differential equations to study the dynamics of immune systems. In [3] for example, a multi-compartmental model is developed to study the growth and elimination of virus and immune cells in various organs and tissues. Typical aims of this sort of study are to identify intercompartmental transfer rates, to reproduce and explain phenomena observed in vivo and to explore scenarios under which particularly beneficial or pathological responses can occur, with the intention of identifying useful insight that can be transferred back into treatment or other in vivo or in vitro studies.

It is often difficult to achieve a realistic compartmental model. Solutions to the differential equations that faithfully capture observable behavioural phenomena depend crucially on the values of intercompartmental flow rates and other system parameters. These are often difficult or impossible to measure directly and so values are estimated from available data using parameter estimation techniques. GAs may have a role to play in supplementing traditional approaches to parameter estimation. In [5], a GA is used to fit parameters to model the permeability of oil-bearing rock strata as a function of logging and core-sampling data. In this case, solutions to the problem are fixed predictive models, and both the parameters and the functional form of each model are encoded in its chromosome representation. A chromosome is evaluated against the logging and core-sampling data by calculating the sum of absolute residuals and the fitness function applies selective pressure to minimise this sum. The GA was able to outperform the standard algorithm previously used in this area and suggested that the predictive model most commonly used should have an additive rather than a multiplicative form.

5.2. Optimal control

A major objective of modelling immune response is to understand the dynamics of virus or disease elimination through the action of the immune system. An example of particular interest is the vaccination process, where the immune system is exposed to “dead” virus cells and responds by producing antibodies. This prepares the system against exposure to the live virus. It is known, [3] that the distribution of antigen around the various organs and tissues in the host significantly affects the immune response, self–nonself discrimination, tolerance and autoimmunity and immunopathology [11,12,29,35,36]. Bocharov et al. [3] use a compartmental model of the dynamics of lymphocytic choriomeningitis virus in mice to examine the severity of the disease as a function of the virus dose and the host’s immune status. They use the model to explore the disease dynamics for a variety of different initial configurations of the system and show that small differences in the kinetic parameters account for large differences in the disease course. They conclude that “The problem of how virus elimination could be achieved with least immunopathology for the host still remains open for HBV, HCV, HIV and other infections and requires a combination of experimental approaches and theoretical analysis based on mathematical models of systemic virus dynamics and lymphocyte recirculation.”

An interesting area of GA application that has relevance for this sort of problem is the solution of optimal control problems. Optimal control is a well-known mathematical technique based on the calculus of variations. One formulates an objective function based on a system response and a control parameter vector. The response is related to the control using a set of equality and inequality constraints that represent the differential equations underlying the system and any side constraints that require to be applied. These are then solved as a system using the Euler–Lagrange equations, resulting in a value for the control vector that gives the optimal response.

For problems involving highly nonlinear systems, this is in theory attractive. For the vaccination problem, for example, one might formulate an objective in terms of virus elimination, with a control vector representing dosages and timings of vaccination. The constraints would represent the coupled differential equations of the compartmental model. Additional constraints would be added to prevent lethal levels of the virus accumulating during the course of the disease and the objective function would be augmented by a penalty term relating to the level of immunopathology in the host.

However, in most realistic examples, the optimal control equations quickly become mathematically and computationally intractable. In such circumstances, it becomes infeasible to solve for an optimal solution. All one can do is to search heuristically for the best solution that can be found at reasonable computational cost. GAs can be particularly useful in this regard for a number of reasons. Firstly they are particularly good at finding feasible solutions in highly constrained problems. Secondly, they are able to handle large numbers of constraints without significant performance penalties compared to other techniques. Thirdly, they are naturally suited to situations where a numerical simulation can be used to determine the outcome of a particular choice of control parameters. In a typical GA application to optimal control, the control parameters are encoded as chromosomes and the simulation is used as a fitness function to evaluate the outcome in terms of the objective function augmented by constraint penalties. We now illustrate this with an example of a GA used to design cancer chemotherapy treatments.

5.3. Example

In cancer chemotherapy, tumour development is controlled by delivery of toxic drugs subject to a number of constraints. The conflicting nature and nonlinearity of the constraints, coupled to the combinatorial possibilities for multi-drug schedules, make it difficult to solve the problem of cancer chemotherapy optimisation by means of empirical clinical experimentation [6] or by means of traditional optimisation methods [19,31].

Anti-cancer drugs are usually delivered according to a discrete dosage programme in which there are n doses given at times $t_1, t_2, \ldots, t_n$. In the case of multi-drug chemotherapy, each dose is a cocktail of d drugs characterised by the concentration levels $C_{ij}, i = \overline{1,n}, j = \overline{1,d}$ of anti-cancer drugs in the blood plasma. Optimisation of the treatment is achieved by modification of these variables. Therefore, the solution space of the chemotherapy optimisation problem is the set of control vectors $\mathbf{c} = (C_{ij})$ representing the drug concentration profiles.

Various models can be used to simulate the response of a tumour to chemotherapy. The most popular is the Gompertz growth model with a linear cell-loss effect [30,34]. The model takes the following form:

$$\frac{\mathrm{d}N}{\mathrm{d}t} = N(t)\left[\lambda \ln\left(\frac{\Theta}{N(t)}\right) - \sum_{j=1}^{d} \kappa_j \sum_{i=1}^{n} C_{ij}\{H(t-t_i) - H(t-t_{i+1})\}\right], \tag{1}$$

where $N(t)$ represents the number of tumour cells at time t; λ, Θ are the parameters of tumour growth; $H(t)$ is the Heaviside step function; C_{ij} denote the concentrations of anti-cancer drugs; and κ_j are quantities representing the efficacy of anti-cancer drugs. The κ_j are estimated using clinical trial results [20].

A cancer chemotherapy treatment may be either curative or palliative. Curative treatments attempt to eradicate the tumour. It is believed that chemotherapy alone cannot eradicate cancer, but that if the overall tumour burden is held below a certain level, other mechanisms (e.g., the immune system or programmed cell death) will remove remaining tumour cells. Palliative treatments are applied only when a tumour is deemed to be incurable and have the objective of maintaining a reasonable quality of life for the patient for as long as possible. The objectives of curative and palliative treatments conflict with each other in the sense that drug regimes which tend to minimise tumour size are highly toxic and therefore have a negative effect on patient quality of life. Moreover, it has been shown that a severe regime that fails to cure can result in a shorter patient survival time (PST) than a milder palliative regime [19].

Treatments must be constrained in a number of ways. In general the constraints of chemotherapeutic treatment are formulated as follows:

$$\begin{aligned}
&g_1(\mathbf{c}) = C_{\max\ j} - C_{ij} \geqslant 0, \quad \forall i = \overline{1,n},\ j = \overline{1,d},\\
&g_2(\mathbf{c}) = C_{\text{cum}\ j} - \sum_{i=1}^{n} C_{ij} \geqslant 0, \quad \forall j = \overline{1,d},\\
&g_3(\mathbf{c}) = N_{\max} - N(t_i) \geqslant 0, \quad \forall i = \overline{1,n},\\
&g_4(\mathbf{c}) = C_{s\text{-eff}\ k} - \sum_{j=1}^{d} \eta_{kj} C_{ij} \geqslant 0, \quad \forall i = \overline{1,n},\ k = \overline{1,m}.
\end{aligned} \tag{2}$$

Constraints $g_1(\mathbf{c})$ and $g_2(\mathbf{c})$ specify the boundaries for the maximum instantaneous dose $C_{\max\ j}$ and the maximum cumulative dose $C_{\text{cum}\ j}$ for each drug used. Constraints $g_3(\mathbf{c})$ limit the tumour size during treatment. Finally, constraints $g_4(\mathbf{c})$ restrain the toxic side effects of multi-drug chemotherapy. The factors η_{kj} represent the risk of damaging the kth organ or tissue (such as heart, bone marrow, lung, etc.) by administering the jth drug.

The optimisation objective for curative treatment is now formulated as

$$\begin{aligned}
&\text{minimise } J_1(\mathbf{c}) = \int_{t_1}^{t_n} N(\tau)\,\mathrm{d}\tau\\
&\text{subject to state (1) and the constraints (2).}
\end{aligned} \tag{3}$$

If the primary objective cannot be achieved, a palliative treatment will be sought to prolong the PST, denoted here by T:

$$\text{maximise } J_2(\mathbf{c}) = \int_0^T 1 \cdot \mathrm{d}\tau \quad \text{subject to (1) and (2).} \tag{4}$$

Infeasible solutions $\mathbf{c}$ are penalised using metrics

$$d_s = \max^2\{-g_s(\mathbf{c}), 0\}, \quad s = \overline{1,4} \tag{5}$$

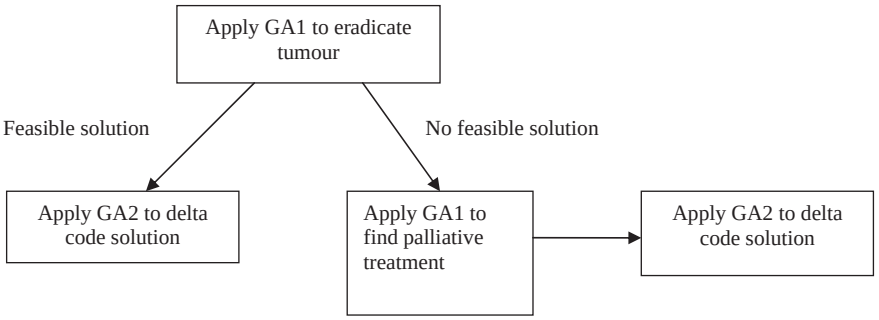

Fig. 1. The optimisation process.

to measure the distance from the feasible region corresponding to the toxicity constraints $g_s(\mathbf{c})$ formalised in (2). Applying penalty coefficients P_s to each metric, each objective is augmented to take the constraints into account as follows.

$$\tilde{J}_i(\mathbf{c}) = J_i(\mathbf{c}) - \sum_{s=1}^{4} P_s d_s, \quad i = 1, 2. \tag{6}$$

Treatment optimisation is accomplished using two genetic algorithms, GA1 and GA2. GA2 is a delta-coding GA operating on the chromosomes of GA1. Fig. 1 shows how GA1 and GA2 are used to search for curative regimes and, failing that, palliative solutions.

Multi-drug chemotherapy control vectors $\mathbf{c} = (C_{ij})$, $i = \overline{1, n}$, $j = \overline{1, d}$ are encoded as bit strings. The representation space I can be expressed as a Cartesian product

$$I = A_1^1 \times A_1^2 \times \cdots \times A_1^d \times A_2^1 \times A_2^2 \times \cdots \times A_2^d \times \cdots \times A_n^1 \times A_n^2 \times \cdots \times A_n^d \tag{7}$$

of allele sets A_i^j. GA1 uses a 4-bit scheme $A_i^j = \{a_1a_2a_3a_4 : a_k \in \{0, 1\}\ \forall k = \overline{1, 4}\}$, so that each concentration C_{ij} takes an integer value in the range of 0–15 concentration units. In general, with n treatment intervals and up to 2^p concentration levels for d drugs, there are up to 2^{npd} chromosomes. The assumptions made in [20], giving 2^{400} chromosomes, are not unrealistic. For GA2, a sign scheme, $A_i^j = \{0, 1, -1\}$ is used, representing either no change or an alteration of $\pm\Delta C_j$ to the drug levels C_{ij}.

A GA1 chromosome $x \in I_{\mathrm{GA1}}$ can be expressed as

$$x = \{a_1a_2a_3 \ldots a_{4nd} : a_k \in \{0, 1\}\ \forall k = \overline{1, 4nd}\}. \tag{8}$$

A GA2 chromosome $x \in I_{\mathrm{GA2}}$ can be expressed as

$$x = \{a_1a_2a_3 \ldots a_{nd} : a_k \in \{0, 1, -1\}\ \forall k = \overline{1, nd}\}. \tag{9}$$

Then the encoding functions on I_{GA1} and I_{GA2} will have the following forms:

$$\mathrm{GA1}: \quad C_{ij} = \Delta C_j \sum_{k=1}^{4} 2^{4-k} a_{4d(i-1)+4(j-1)+k}, \qquad \forall i = \overline{1, n}, \quad j = \overline{1, d}, \tag{10}$$

$$GA2: \quad C'_{ij} = C_{ij} + a_{d(i-1)+j}\Delta C_j, \qquad \forall i = \overline{1, n}, \quad j = \overline{1, d}. \tag{11}$$

A separate numerical tumour response simulation routine is used to evaluate each chromosome. A regime is deemed to be curative if the tumour is reduced to 1000 or fewer cells for three consecutive treatment intervals. The tumour size is then reduced to zero for the remainder of the treatment. For each chromosome, the simulation returns the sequence of tumour sizes at the end of each interval. Fitness functions for GA1 and GA2 are now obtained by calculation of the augmented objective functionals (6).

The GA-simulation system can now be used as a treatment design tool by oncologists [4]. The oncologist can explore a wide range of treatment scenarios. Parameters for tumour growth rates, drug effectiveness rates, toxic effects and constraint settings will vary with drugs used, cancer type, delivery method and patient population. Different response models will be appropriate for different cancer types and it is straightforward to replace the simulation component as required. For a given scenario, the system will suggest optimal treatment strategies. The oncologist can then evaluate these before deciding upon suitable novel regimes for clinical experiment. Potentially, this represents a substantial cost saving, both in financial and human terms. Although conflicting objectives were handled sequentially in this case, genuine multi-objective optimisation can also be attempted using GAs [8,32,37]. A multi-objective GA optimisation of chemotherapy design can be found in [27].

GAs have been successfully applied to a wide range of optimal control problems. In general though, *bit-string* chromosomes become problematic. Many problems require high precision values for control variables. Chromosome lengths quickly become unwieldy when trying to achieve this with bit strings. A lot of work has been done in this area using chromosomes whose alleles are floating-point numbers as opposed to bits. Most elements of GA construction remain unchanged; however, it is necessary to re-define what is meant by mutation and crossover. A number of genetic operators can be defined in analogy with the *bit-string* case and some alternative approaches (for example, averaging alleles) become possible due to the richer structure of the number system used. A full account of GAs for optimal control is given in [21].

5.4. Artificial immune systems

To complete this section, we briefly review the area of artificial immune systems (AIS). AIS are a problem-solving heuristic technique inspired by the human immune system in the same way that GAs are inspired by evolution. AIS are a computational technique that use biological immune systems as a problem-solving metaphor in the same way that genetic algorithms use biological evolution. Key properties of immune systems are translated into the data structures and workflow of AIS. Early collaborations in this area between researchers in genetic algorithms, immunology and complexity theory include Kauffman et al. [18] and Forrest et al. [14].

De Castro [10] identifies a number of ways in which AIS are applied. For example, in biological immune systems, clonal selection creates immune cells that match a particular antigen by a process of proliferation and differentiation biased towards producing cells that match the antigen. This process is mimicked by some AIS used for learning and optimisation tasks. The analogy is that naive immune cells represent solutions to a learning problem represented by the antigens. Clonal selection is then an evolutionary process that results in highly fit solutions to the learning problem. A typical algorithm would work along the following lines. Candidate solutions (to a learning problem say) are encoded as (bit) strings. Each solution is presented with a set of antigens (representing a training set for the learning problem, also encoded as strings) and an affinity measure is calculated. Those solutions with the highest affinity to the antigens are preferentially selected for cloning. Once cloned, the copies of selected solutions are then

"differentiated" using one of a set of possible mutation operators. The process then iterates until some stopping condition is reached. This approach bears many similarities to a GA. However, the particular selection, reproduction and mutation processes are quite distinctive to AIS.

Another feature of immune systems that can be reflected in an AIS is self–nonself discrimination. The property that T-cells that can discriminate only nonself can be evolved through a negative selection process is an important one for distributed computer systems where viruses, spam emails and other attacks are a large and growing problem. Algorithms that can monitor computer systems to detect anomalous programs and files promise to be of great use in combatting this problem. De Castro [10] provides a top-level workflow for a typical AIS implementing negative selection to solve some anomaly detection problem. He also identifies areas of commonality and complementarity between AIS and other "Soft Computing" paradigms such as GAs and Neural Networks and provides some useful suggestions for the development of more sophisticated hybrid algorithms.

6. Conclusion

GAs are a heuristic solution search technique inspired by natural evolution. They are a robust and flexible approach that can be applied to a wide range of learning and optimisation problems. They are particularly suited to problems where traditional optimisation techniques break down, either due to the irregular structure of the search space (for example, absence of gradient information) or because the search becomes computationally intractable. In this paper, we have described the major components and properties of a GA, with examples of implementations on particular problems. This provides a starting point for researchers in immunology interested in applying GAs to problems of their own interest. Simple implementations of GAs can be made with a small amount of up front effort. More significant applications ought to emerge as experience is gained and the "right" problems for GAs to address are identified. Parameter-fitting and optimal control have been speculatively identified in this paper as areas where the use of GAs might assist in the design and analysis of multi-compartmental models of biological immune systems.

The review of the main strands of GA theory contained in Section 4 demonstrates that, in general, what is understood theoretically about how GAs operate is limited in scope and applicability to special cases that do not include the vast majority of nonclassical GA applications. There is a strong parallel here to the attempts of mathematical immunologists to model immune systems. GAs and AIS are both greatly simplified abstractions of the natural processes that they represent. However, computer scientists trying to understand and use these approaches are now grappling with problems very similar to those faced by biologists in understanding how complex systems operate over a range of scales. It is to be hoped that the insights and experience of both research communities can be fruitfully recombined.

Acknowledgements

I would like to thank the anonymous JCAM referee for his kind and helpful comments.

References

[1] T. Bäck, D.B. Fogel, Z. Michalewicz (Eds.), Handbook of Evolutionary Computation, IOP Publishing, 1997.
[2] T. Bäck, F. Hoffmeister, H.-P. Schwefel, A survey of evolution strategies, in: R.K. Belew, L.B. Booker (Eds.), Proceedings of the Fourth International Conference on Genetic Algorithms, Morgan Kaufmann, Los Altos, CA, 1991.

[3] G. Bocharov, P. Klenerman, S. Ehl, Modelling the dynamics of LCMV infection in mice: II. Compartmental structure and immunopathology, J. Theor. Biol. 221 (2003) 349–378.
[4] J. Boyle, D. Henderson, J. McCall, H. McLeod, J. Usher, Exploring novel chemotherapy treatments using the WWW, Internat. J. Med. Inform. 47 (1–2) (1997) 107–114.
[5] D.F. Brown, S.J. Cuddy, A.B. Garmendia-Doval, J.A.W. McCall, Prediction of permeability in oil-bearing strata using genetic algorithms, Proceedings of the IASTED International Conference on Artificial Intelligence and Soft Computing, July 2000.
[6] J. Cassidy, H. McLeod, Is it possible to design a logical development plan for an anti-cancer drug?, Pharm. Med. 9 (1995) 95–103.
[7] A.K. Chakraborty, M.L. Dustin, A.S. Shaw, In silico models for cellular and molecular immunology: successes, promises and challenges, Nat. Immunol. 4 (10) (2003) 933–936.
[8] C. Coello, An updated survey of evolutionary multiobjective optimization techniques: state of the art and future trends, Proceedings of the 1999 Congress on Evolutionary Computation, IEEE Service Center, Washington, DC, 1999, pp. 3–13.
[9] L. Davis, Handbook of Genetic Algorithms, Van Nostrand Reinhold, New York, 1991.
[10] L.N. De Castro, Artificial immune systems as a novel soft computing paradigm, Soft Comput. J. 7 (7) (2003).
[11] P.C. Doherty, Anatomical Environment as a determinant in viral immunity, J. Immunol. 155 (1995) 1023–1027.
[12] P.C. Doherty, A.M. Hamilton-Easton, D.J. Topham, J. Riberdy, J.W. Brooks, R.D. Cardin, Consequences of viral infections for lymphocyte compartmentalization and homeostasis, Semin Immunol. 9 (1997) 365–373.
[13] A.P. Engelbrecht, Computational Intelligence An Introduction, Wiley, New York, 2002.
[14] S. Forrest, A.S. Perelson, Genetic algorithms and the immune system in: H.-P. Schwefel, R. Maenner (Eds.), Parallel Problem Solving from Nature, Lecture Notes in Computer Science, vol. 496, Springer, 1991, pp. 320–325.
[15] D.E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley, Reading, MA, 1989.
[16] G.R. Harik, F.G. Lobo, D.E. Goldberg, The compact genetic algorithm, in: D.B. Fogel (Ed.), Proceedings of the IEEE Conference on Evolutionary Computation 1998 (ICEG'98) IEEE Service Centre, Piscataway, NJ, 1998, pp. 523–528.
[17] J.H. Holland, Adaptation in Natural and Artificial Systems, The University of Michigan Press, Ann Arbor, MI, 1975.
[18] S.A. Kauffman, E.D. Weinberger, A.S. Perelson, Maturation of the Immune Response Via Adaptime Walks on Affinity Landscapes, in: A.S. Perelson (Ed.), Theoretical Immunology, Part One, SFI Studies in the Science of Complexity, Addison-Wesley Publishing Company, Reading, MA, 1988, pp. 349–382.
[19] R. Martin, K. Teo, Optimal Control of Drug Administration in Cancer Chemotherapy, World Scientific, Singapore, New Jersey, London, Hong Kong, 1994.
[20] J. McCall, A. Petrovski, A decision support system for chemotherapy using genetic algorithms, in: M. Mouhammadian (Ed.), Computational Intelligence for Modelling, Control and Automation, IOS Press, 1999, pp. 65–70.
[21] Z. Michaelewicz, Genetic Algorithms + Data structures = Evolution Programs, third ed., Springer, Berlin, 1999.
[22] M. Mitchell, An Introduction to Genetic Algorithms, MIT Press, Cambridge, MA, 1998.
[23] M. Mitchell, S. Forrest, J.H. Holland, The royal road for genetic algorithms: fitness landscapes and GA performance, in: F.J. Varela, P. Bourgine (Eds.), Toward a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life, MIT Press, Cambridge, MA, 1992.
[24] M. Mitchell, J.H. Holland, S. Forrest, When will a genetic algorithm outperform hillclimbing, in: J.D. Cowan, G. Tesauro, J. Alspector (Eds.), Advances in Neural Information Processing Systems, Morgan Kaufmann, Los Altos, CA, 1994.
[25] H. Mühlenbein, G. Paaß, From recombination of genes to the estimation of distributions I. Binary parameters, Parallel problem solving from nature, in: H.-M. Voigt, W. Ebeling, I. Rechenberg, H.-P. Schwefeleds (Eds.), Lecture Notes in Computer Science, vol. 1141, Springer, Berlin, 1996, pp. 178–187.
[26] M. Pelikan, D.E. Goldberg, F.G. Lobo, A Survey of Optimization by Building and using Probabilistic Models, University of Illinois Genetic Algorithms Laboratory, Urbana, IL, IlliGAL Report No. 99018, 1999.
[27] A. Petrovski, J. McCall, Multi-objective optimisation of cancer chemotherapy using evolutionary algorithms, in: E. Zitzler et al. (Ed.), Evolutionary Multi-Criterion Optimization Lecture Notes in Computer Science, 1993, Springer, Berlin, 2001, pp. 531–545.
[28] A. Prügel-Bennett, J.L. Shapiro, An analysis of genetic algorithms using statistical mechanics, Phys. Rev. Lett. 72 (9) (1994) 1305–1309.
[29] T.E. Starzl, R.M. Zinkernagel, Antigen localization and migration in immunity and tolerance, N. Engl. J. Med. 339 (1998) 1905–1913.
[30] G. Steel, Growth Kinetics of Tumours, Clarendon, Oxford, 1977.

[31] G. Swan, Role of optimal control theory in cancer chemotherapy, Math. BioSci. 101 (1990) 237–284.
[32] D. Veldhuizen, G. Lamont, Multiobjective evolutionary algorithms: analyzing the state-of-the-art, Evolut. Comput. 8 (2) (2000) 125–147.
[33] M.D. Vose, The Simple Genetic Algorithm, MIT Press, Cambridge, MA, 1999.
[34] T. Wheldon, Mathematical Models in Cancer Research, Adam Hilger, Bristol, Philadelphia, 1988.
[35] R.M. Zinkernagel, Immunity to viruses, in: W. Paul (Ed.), Fundamental Immunology, third ed., Raven Press, New York, pp. 1121–1250 (Chapter 34).
[36] R.M. Zinkernagel, C. Kelly, How antigen influences immunity, Immunologist 5 (1993) 114–120.
[37] E. Zitzler, K. Deb, L. Thiele, Comparison of multi-objective evolutionary algorithms: empirical results, Evolut. Comput. 8 (2) (2000) 173–195.