Genetic algorithms in astronomy and astrophysics

Vinesh Rajpaul

Astrophysics, Cosmology and Gravity Centre (ACGC), Department of Astronomy, University of Cape Town, Private Bag X3, Rondebosch 7701, Republic of South Africa

E-mail: vinesh.rajpaul@uct.ac.za

Abstract. Genetic algorithms (GAs) emulate the process of biological evolution, in a computational setting, in order to generate good solutions to difficult search and optimisation problems. GA-based optimisers tend to be extremely robust and versatile compared to most traditional techniques used to solve optimisation problems. This paper provides a very brief introduction to GAs and outlines their utility in astronomy and astrophysics.

1. Introduction

Many interesting mathematical problems can be reformulated as global optimisation problems; the solution of systems of algebraic or even differential equations, for example, can be cast quite naturally in terms of optimisation. The same holds true for the all-important inverse problems that are ubiquitous in the physical sciences, i.e. problems where one seeks to transform experimental data into model parameters in order infer properties of the physical system being studied (a simple example: choosing parameters to minimise a χ^2-statistic when fitting a Voigt-profile to a spectral line).

The goal of a global optimisation problem is, given a so-called *cost function*[1] $f : \mathcal{D} \subseteq \mathbb{R}^n \to \mathbb{R}$, to find the point $\vec{x}^* \in \mathcal{D}$ such that:

$$\forall \vec{x} \in \mathcal{D} : f(\vec{x}) \geqslant f(\vec{x}^*); \tag{1}$$

$f(\vec{x}^*)$ is called the global minimum[2]. A local minimum, $f(\hat{\vec{x}})$, is defined by the condition:

$$\forall \vec{x} \in \mathcal{D}, \exists \delta > 0 : \left\| \vec{x} - \hat{\vec{x}} \right\| < \delta \Rightarrow f(\vec{x}) \geqslant f(\hat{\vec{x}}). \tag{2}$$

Whereas finding an arbitrary local minimum of a function is a relatively straightforward task, especially if one has a good "first guess" – extremely efficient techniques exist to solve such local optimisation problems – determining the *global* minimum of a function is a far more challenging problem. Real-world cost functions tend to be nonlinear, discontinuous and/or hugely multimodal, and there is no fool-proof approach to locating their global minima.

Most established approaches – whether deterministic, stochastic or (meta)heuristic – to solving global optimisation problems yield excellent results on a limited class of problems, but have drawbacks that tend to cripple them when faced with certain (reasonably) difficult

[1] Depending on the context, cost functions are also referred to as *energy functions* or *objective functions*.

[2] Maximisation of $f(\vec{x})$ is, of course, equivalent to minimisation of $g(\vec{x}) := -f(\vec{x})$.

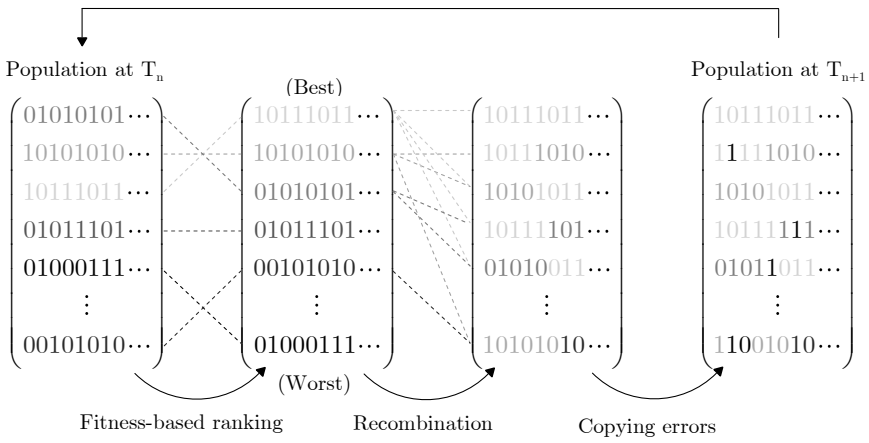

Figure 1. Schematic to illustrate the workings of a simple binary-coded genetic algorithm. Each bit represents a gene; here the genes of high-fitness solutions are given lighter colours.

problems. For example, they might get stuck too easily in local minima, they might be thwarted by discontinuous functions or they might be too slow to be of practical value when faced with enormous search spaces [1].

Evolutionary algorithms, inspired by biological evolution, are metaheuristic optimisation algorithms that tend to yield "good enough" results on a very wide range of (even extremely difficult) optimisation problems. So-called *genetic algorithms* form one of the most successful subsets, and certainly the most popular subset, of evolutionary algorithms.

2. Genetic algorithms: the basic idea

Genetic algorithms, or GAs for short, draw inspiration from population genetics (and, like all evolutionary algorithms, from evolutionary biology in general) and they incorporate, in a computational setting, notions such as natural selection/survival of the fittest, reproduction, genetic recombination, inheritance and mutation. The first GA-based optimiser was proposed in the mid-1970s [2], and since then a great variety of modifications and improvements to the basic algorithm have been developed, including mechanisms without any biological analogues [3].

In spite of the rich variety of their potential incarnations, most GAs share a basic working scheme: they start with a population of many candidate solutions (called individuals or phenotypes), associate with each solution an encoded version of the phenotype (called a chromosome, genotype or an individual's genetic material) and also a measure of the solution's fitness (quality). This fitness function is often simply the additive inverse of the cost function to be optimised. Then, by repeated application of "genetic operators" mainly at the genotypic level, they cause the population as a whole to increase in phenotypic fitness, i.e. they cause the solutions to evolve towards optimality.

A typical (though simplistic and by no means general or optimal) working scheme for a genetic algorithm is as follows:

1. construct a random initial population of genotypes;
2. decode the genotypes and evaluate their phenotypic fitness; if the fittest phenotype matches the user-defined target fitness (or other termination criterion), `break`, otherwise continue;
3. produce offspring by randomly selecting and recombining genetic material from the current population, favouring individuals with high phenotypic fitness;

4. introduce, with some low probability, random changes (copying errors) into the genetic material of the offspring;
5. replace low-fitness members of the old population with the offspring created in the previous step, and `goto` step 2.

The selective recombination of genetic material exploits good solutions to build even better ones (the unique mechanism of information transfer within the population of candidate solutions is often cited as the distinguishing feature of GAs [3]), and the random mutations serve to inject entirely new and potentially favourable material into the gene pool that could not be obtained by recombining the genetic material of existing individuals.

Figure 1 illustrates the working scheme of a simple GA where the solutions are encoded as binary strings[3]. It may be shown that given enough time, and subject to a few reasonable assumptions, a GA will always converge to the global optimum of a cost function[4] [4, 6].

3. GAs: pros and cons

Relative to more conventional optimisation algorithms, GA-based optimisers offer a number of striking advantages, some of which are outlined below.

Robustness. GA-based optimisers can handle – with aplomb – problems with multimodal or low-contrast objective functions, multiple objectives and/or problems where the parameter spaces have a very high dimensionality [1].

Simplicity. In order to solve a given optimisation problem, GAs require only a single, unambiguous measure of the quality (fitness) of candidate solutions. They do not require, for example, gradients or Hessian matrices, the computation of which might in some problems be prohibitively difficult or impossible. Moreover it is a relatively easy task to develop a working GA from scratch, and the ideas underpinning GAs are intuitively accessible.

Speed. Apart from the intrinsically high speed with which GAs tend to explore large parameter spaces [4], they are *embarrassingly parallel*, meaning very little effort is required to transform a serial implementation to a parallel implementation. Thus they are well-suited to exploiting high-performance hardware (multi-core workstations, clusters, GPGPUs etc.).

Versatility. A single GA-based optimiser can be expected to yield "good enough" results on a very wide class of problems – from something as simple, for example, as fitting a three-parameter Gaussian to some data, to something as complex as minimising a Buckingham potential in a molecular configuration problem with hundreds of parameters – and it is easy to incorporate problem-specific knowledge (and constraints) into a GA-based solver. The widespread adoption of GAs in fields such as engineering, manufacturing, chemistry, biology and economics bears testimony to their great versatility [3].

To illustrate the great robustness and versatility of a typical GA, consider the following cost function proposed by Charbonneau [7]:

$$f(x, y; n) = -[16x(1-x)y(1-y)\sin(n\pi x)\sin(n\pi y)]^2, \tag{3}$$

where $x, y \in [0, 1]$ and $n \in \mathbb{N}$. For $n = 13$, for example, it may be shown that $f(x, y)$ has 169 local minima on its domain, only one of which is the global minimum[5]; moreover, the minima are separated by steep walls and there is little contrast between many of the minima (see figure 2).

[3] Most early GAs encoded solutions as binary strings, both for the sake of simplicity and supposed theoretical optimality; a large body of empirical evidence, however, indicates that in practice is is usually preferable to work directly with floating-point representations of solutions! [1, 3, 4, 5]

[4] Of course this knowledge is of little practical value; of more importance is the *rate* of convergence to the global optimum, though unfortunately with GAs this rate is highly problem-dependent and difficult to estimate *a priori*.

[5] In general, $f(x, y; n)$ will have n^2 local minima; for n odd, there will be a unique global minimum, and for n even, 4 of the local minima will be global minima.

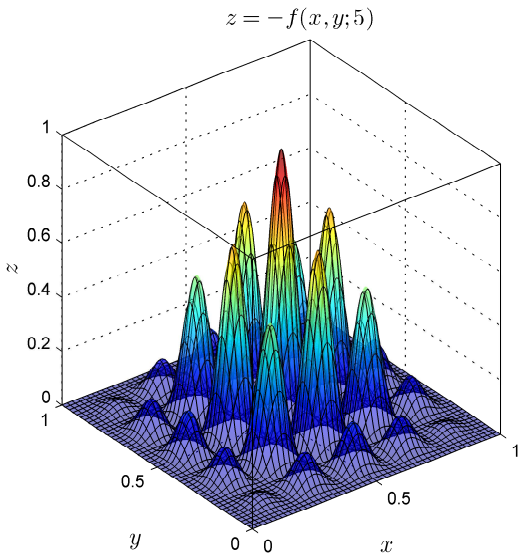

Figure 2. Surface plot of the function $f(x, y; n)$ defined by equation 3 for the case $n = 5$. In this case there are $n^2 = 25$ local optima on the domain $x, y \in [0, 1]$.

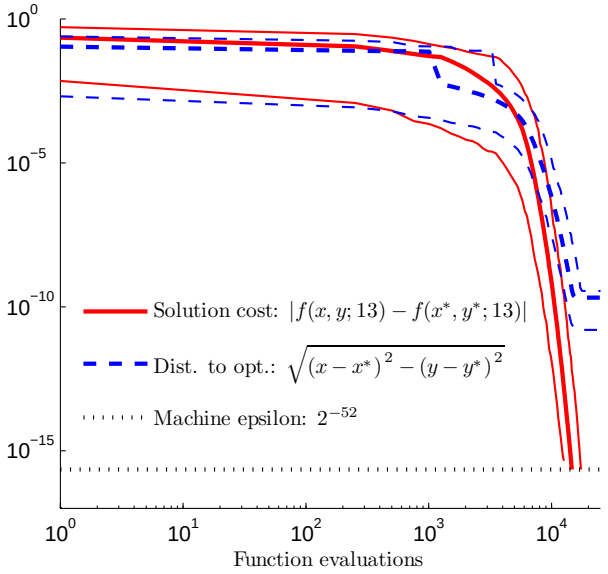

Figure 3. Performance of a GA-based optimiser applied to the $n = 13$ case of $f(x, y; n)$; the thick lines denote median performance in 10000 trials, and the thin lines, upper and lower 3σ limits.

Figure 3 illustrates how a GA-based optimiser fared on the (rather challenging) $n = 13$ problem: in 10000 trials, the algorithm converged to the global minimum every single time, with the minimum location determined to a median accuracy of about one part in a billion after only $\sim 10^4$ function evaluations (or a fraction of a second on a modern workstation). For comparison, a blind random search would require $\sim 10^{18}$ evaluations to guarantee similar accuracy!

Although this performance is impressive in its own right, it is worth emphasising that it took mere minutes to adapt an existing GA-based optimiser[6] to solve this problem, and that the algorithm control parameters were not optimised in any way for this new problem.

An obvious question arises: *why* do GAs work as well as they do? This topic is far beyond the scope of this survey paper but suffice it to say that a universally-accepted explanation has not yet been developed. Holland's famous *Schema Theorem* has long been touted as providing an explanation for GAs' success [2], although more recently it has become apparent that this theorem provides insight only into the workings of simplistic GAs; and even then, it is not clear whether the assumptions underlying the theorem are tenable [8, 9].

Unsurprisingly, in spite of all their attractive features, GAs also have their share of disadvantages (more or less in accordance with Wolpert and Macready's famous "no free lunch" theorem [10]). GAs might be called "Jacks of all problems, but masters of none": optimising a GA's performance on a given problem is often difficult or impossible, and in order to achieve near-optimal performance it is usually necessary to hybridise a GA with problem-specific heuristics. For example, they tend to be better at locating than at fine-tuning solutions: once a GA is in the vicinity of a global optimum, it is usually a good idea to let a local optimiser take over [1]. GAs can be inefficient on simple problems where the computational expense of applying the genetic operators outweighs that of evaluating the function to be optimised; conversely, on problems

[6] This GA, coded by the author, used floating-point encoding, dynamically-adjusted mutation rates and tournament-style selection of reproducing partners.

where each cost function evaluation is extremely expensive (each evaluation might require a long simulation to be run!), a GA-based forward modelling approach can be impractical.

Finally the (currently) limited theoretical understanding of GAs is regarded by some, quite understandably, as a major drawback and this might explain their relatively slow uptake in the physical sciences [7].

4. Applications: astronomy and astrophysics

This section presents a sample of the numerous and diverse applications that genetic algorithms have found in astronomy and astrophysics. For brevity's sake, only one or two short but representative examples have been drawn from different subfields. *Astrophysical dynamics.* Wahde and Donner developed a method for reliably determining the orbital parameters of interacting galaxies and applied their method to both artificial and real data [11]. Their method is based on a GA that searches very efficiently through the large space of possible orbits; indeed, the authors argue that GAs are ideally suited for investigations of tidally interacting galaxies, where large multimodal search spaces must be searched in order to constrain a large number of model parameters. Canto *et al* devised an interesting variant of the canonical GA which they applied successfully to various problems, including the challenging task of finding the orbital parameters planets orbiting 55 Cancri, based on radial velocity measurements of the aforesaid stellar system [12].

Physical and observational cosmology. Although Monte Carlo methods seem to predominate in cosmology, GAs have already found a number of applications in the field. To mention just a couple: Nesseris and Shafieloo used GAs to reconstruct the expansion history of the universe in a model-independent manner and thence, in conjunction with the so-called *Om statistic*, they derived a null test on the cosmological constant model ΛCDM [13]; via GAs, Allanach *et al* were able to answer some important questions related to the discrimination of SUSY-breaking models, and in particular to quantify the measurements necessary to tell different SUSY-breaking scenarios apart [14]; and Bogdanos and Nesseris used GAs to analyse Type Ia SNe data and to extract model-independent constraints on the evolution of the dark energy equation of state [15]. The latter authors note that as a non-parametric method, GAs provide a convenient model-independent platform for cosmological data analysis that can minimise bias due to premature choice of e.g. a dark energy model.

Gravitational lens modelling. Gravitational microlensing is an ideal technique for probing the galactic population of faint or dark objects such as substellar objects, stellar remnants, MACHOs and exoplanets. Though very successful, theoretical microlensing models tend to be complex and their associated inverse modelling problems are notoriously difficult. The author of this paper has recently been developing GAs to speed up this difficult modelling, with a view to being able to model ongoing events approximately in real time (i.e. on a timescale of minutes rather than weeks or months!) and thereby to facilitate better-informed observations and thus more useful observational data. Results of this work are expected to be published in early 2012. As another example, Liesenborgs *et al* presented a GA-based, non-parametric technique for inferring the projected lensing-mass distributions in strongly lensed systems [16].

Stellar spectrum fitting. Performing fits to stellar spectra is a nontrivial but important undertaking; from fitted models one can infer a veritable multitude of stellar properties. Baier *et al* were able to combine radiative transfer codes with a GA to produce an automated procedure for fitting the dust spectra of AGB stars. Their GA-based routine dramatically improved extant fits made with more traditional methods and provided a quantitative platform from which to compare different models [17]. In a similar vein, Mokiem *et al* used a parallelised GA as the basis for an autonomous fitter of spectra of massive stars with stellar winds [18].

Stellar structure modelling. Metcalfe and Charbonneau implemented a highly-parallelised and distributed GA to determine the globally optimal parameters for stellar models. The efficient,

parallel exploration of parameter space made possible by the GA-based optimisation led to some important results in the field of white dwarf astroseismology, including the unexpected resolution of a then-puzzling discrepancy between stellar evolution model and astroseismic inferences of He-layer masses in DBV white dwarfs [19].

Telescope scheduling. Autonomous telescope scheduling is a difficult task that requires dynamic adjustment of numerous observational constraints whilst trying to ensure the efficient achievement of many different scientific objectives. Kubanek developed an easy-to-implement yet robust approach to a robotic telescope scheduling problem, based on a GA that seeks out Pareto-optimal solutions (telescope schedules) [20].

5. Conclusions

This paper introduced genetic algorithms, mentioned some of their strengths (and weaknesses) and finally illustrated their utility in astronomy and astrophysics. For those who would like to learn more about GAs (or other evolutionary algorithms), there are many fine books on the subject: to mention just a couple, Michalewicz's book [4] gives an excellent introduction with a theoretical leaning, and Haupt's book provides an equally good though more "hands-on" treatment [3]. Goldberg's seminal tutorial-style book [21], one of the most widely-cited works in all of computer science, also serves as an outstanding reference.

The author of this paper would welcome correspondence from anyone who would like to discuss evolutionary algorithms, perhaps with a view to applying them in their own work.

Acknowledgments

The author wishes to thank the University of Cape Town and the National Research Foundation for the provision of financial support.

References

[1] Charbonneau P 2002 *An Introduction to Genetic Algorithms for Numerical Optimization* (Boulder, CO: National Center for Atmospheric Research)
[2] Holland J H 1975 *Adaptation in Natural and Artificial Systems* (Ann Arbor, MI: The University of Michigan Press)
[3] Haupt R L and Haupt S E 2004 *Practical Genetic Algorithms* (Toronto: Wiley-Interscience)
[4] Michalewicz Z 1996 *Genetic Algoritms + Data Structures = Evolution Programs* (Berlin, AL: Springer)
[5] Wright A H 1991 *Foundations of Genetic Algorithms* ed Rawlins G J (San Mateo, CA: Morgan Kaufmann) pp 205–18
[6] Eiben A E, Aarts E H L and van Hee K M 1991 *Proc. 1st Workshop on Parallel Problem Solving from Nature* (London: Springer-Verlag) pp 4–12
[7] Charbonneau P 1995 *ApJS* **101** 309–34
[8] Syswerda G 1989 *Proc. 3rd Int. Conf. on Genetic Algorithms* ed Schaffer D J (San Mateo, California: Morgan Kaufmann Publishers, Inc.) pp 2–9
[9] Wright A H, Vose M D and Rowe J E 2003 *Proc. 2003 Int. Conf. on Genetic and Evolutionary Computation: Part II* (Berlin, Heidelberg: Springer-Verlag) pp 1505–17
[10] Wolpert D H and Macready W G 1997 *IEEE Trans. Evol. Comp.* **1** 67–82
[11] Wahde M and Donner K J 2001 *A&A* **379** 115–24
[12] Cantó J, Curiel S and Martínez-Gómez E 2009 *A&A* **501** 1259–68
[13] Nesseris S and Shafieloo A 2010 *MNRAS* **408** 1879–85
[14] Allanach B C, Grellscheid D and Quevedo F 2004 *J. High Energy Phys.* **7** JHEP07(2004)069
[15] Bogdanos C and Nesseris S 2009 *J. Cosmol. Astropart. Phys.* **5** JCAP05(2009)006
[16] Liesenborgs J, De Rijcke S and Dejonghe H 2006 *MNRAS* **367** 1209–16
[17] Baier A, Kerschbaum F and Lebzelter T 2010 *A&A* **516** A45
[18] Mokiem M R, de Koter A, Puls J, Herrero A, Najarro F and Villamariz M R 2005 *A&A* **441** 711–33
[19] Metcalfe T S, Nather R E and Winget D E 2000 *ApJ* **545** 974–81
[20] Kubanek P 2008 *Genetic Algorithm for Robotic Telescope Scheduling* (Granada: University of Granada Press)
[21] Goldberg D E 1989 *Genetic Algorithms in Search, Optimization and Machine Learning* 1st ed (Boston, MA: Addison-Wesley Longman Publishing Co., Inc.)