The Digital Guesswork: a Philosophical Appraisal of Genetic Algorithms and their Epistemology

Abstract

The present paper argues that a specific type of computation, the evolutionary computation, has a particular epistemic status and it is an important instance in which a philosophical concept of evolution can draw a bridge between AI and the practice of biological sciences. Philosophers focused on the relation between computation on one hand, philosophical logic and philosophy of mind on the other hand. My aim is to explore the epistemological aspects of evolutionary computation from a philosophy of science perspective. I argue that this type of computation has important consequences for the mainstream epistemology and for the philosophy of numerical simulations used in science. The genetic algorithms illustrate an "upward epistemology" from data to theories and is relevant in the context of scientific discovery.

1 Introduction: Why Computational Philosophy?

Why do we need a philosophical analysis of artificial life? This question can be incorporated in a more general topic: "philosophy of computation". This ilk of philosophical analysis does not have name, so I adopt here the neural denomination "philosophy and computation" as an umbrella concept Beavers (2011); Floridi (2011). As several distinctive paradigms of computation are preeminent, I urge philosophers to be more nuanced in approaching computation as a whole. In a slogan form, I claim that "philosophically, not all computations are equal". I focus on one type of computation—evolutionary computation, and argue for its novel epistemological aspects, especially in the context of scientific knowledge. Adopting Beavers' terminology, my paper falls under the category of "epistemology (including philosophy of science) and evolutionary computation" or "epistemology of evolutionary computation".

Why computation and epistemology? Relating computer science to knowledge is a challenging task. Are computers in any significant way a source of knowledge? I take for granted that contemporary philosophy of mind, philosophy of language, cognitive science, philosophical logic and philosophy of mathematics would not look the same without the advancements in artificial intelligence and computer science of the last six decades. Some philosophers have not kept the pace with their peers: old-school ethicists, metaphysicians, epistemologists, and, surprisingly, philosophers of science had barely shown interest in computation. Nonetheless, in the past twenty years or so, epistemology of computation has become increasingly pervasive; ditto about the philosophy of science and computation. For an recent approach to "computational epistemology" and its relation to mainstream epistemology see Hendricks (2007) esp. Ch. 7. Another take-home lesson, which I premise my argument on, is that the history of computation is an central component of the "computational epistemology". In my paper I highlight some important moments in the history of evolutionary computation, and, equally important, I incite philosophers to take a look at the promising future of this area.

2 A Deflationist Position

Some philosophers, perhaps a majority, look with a jaundiced eye to "philosophy and computation" and disbelieve that computation is philosophically interesting. Here is a template for a deflationist argument about computation (A. Lovelace had raised similar worries in the 1850s): if computations are dummy tools or techniques with little philosophical relevance, and if epistemology is not about tools, artifacts, or techniques, but about human knowledge and philosophy of science is about the advancement of scientific knowledge by humans then no matter how important they are to the advancement of knowledge, they do not deserve a philosophical privileged position and standard epistemology/philosophy of science would suffice to analyze them. In the early discussions on the foundations of computation, this question emerged frequently: what is the epistemological difference between the output of a computational process (be it computer program) and what an army of numerical analysts equipped with "slide rules" can do? Philosophers wondered about the epistemic shift, what was the novel element that was not present in the pre-existing work of numerical analysts? Whether computation is epistemologically interesting depends on another question about artificial intelligence Samuel (1959):

> $\mathbf{Q_1}$: How can computers be made to do what needs to be done, without being told exactly how to do it?

The computational deflationist argues for a *negative* answer to $\mathbf{Q_1}$. Some prevalent claims in the philosophy of mind echo this claim: machines do not think, do not discover and do not invent. An assumption for a negative answer to $\mathbf{Q_1}$—strongly opposed by the AI community, is Simon (1969):

> [1] A computer program is no better than the assumptions which it was built on.

The deflationist shows that Turing-like computation is a direct illustration of [1]. Turing machines are programmed to solve one, *known* problem, based on a set of fixed rules and the majority of computers around us are in fact such "glorified slide rules". Algorithms are similar to "analytic machines" because, among other things, the input *always* determines the output. But the rules according to which the input determines the output are pre-programmed by a human being. In this sense, the deflationary epistemologist claims that there is no element of novelty in the results of a computer program as far *human* knowledge is concerned.

Computation in science, its application, the numerical simulation, and the respective scientific method of "computational science" gained some momentum in the philosophy of science literature in the last decade or so Humphreys (2009); Frigg and Reiss (2009); Hartmann (2008); Humphreys (2004). A numerical simulation is a procedure used to build models involving equations that we cannot solve analytically and when no exact solutions are available or when the exact solutions are not useful. Simulations are currently used to replace experiments and observations when data are sparse or inexistent. It is almost impossible to imagine a 21^{th} century science that jettisons of numerical simulations or computation methods. But until recently, philosophers of science have not paid attention to the astonishing importance of numerical simulations for the practice of science. This is derivative from the glorified slide rule sketched above: computational science is a technique or tool in the hands of scientists, hence, their second-class philosophical status.

E. Winsberg takes numerical simulations as mediators between theories and experiments and the metaphor is calling them "mongrels", between experiments and theories. Like models, they retain a form of autonomy from both the theory and from experiments. One aspect

of the numerical simulation is crucial to my argument: witness that in the "glorified slide rule" argument, the theory or theoretical assumptions (or rules) based on which the numerical simulations are supposed to work have to be already known to the scientists. To perform a numerical simulation, one needs a full-fledged theory, already interpreted, already equipped with a set of partial differential equations of the temporal dynamics of a physical system or an idealized form of it. Whenever the analytic solution is discovered, a real experiment is possible or new data is available, the numerical simulation can be thrown away. As E. Winsberg suggests, numerical simulations have a *downward epistemology*: "we start from a theory and justify inferences from it" Winsberg (2001):

> [2] Computation in science is always used to justify an existing theory (or model).

One conclusion is that numerical simulations are second-order citizens of the world of scientific methods and work as *replacements* for other more robust instruments of knowledge: theories, models and experiments. Another foreseeable consequence of [2] is:

> [3] Numerical simulations need a top-down account of scientific justification.

In philosophy of science, the computational deflationist asks again: do numerical simulations add something new to the philosophy of theories, models, experiments, analogies or similarities? Some philosophers of science, seemingly the majority, take them as epistemologically subordinate and inferior, in the sense of brute tools of mechanization of the lowliest part scientific work because they provide scientific knowledge *only when* other, more rigorous ways of representing the real world fail—similar to the glorified slide rules Eason et al. (2007); Frigg and Reiss (2009). Numerical simulations introduce idealizations, approximations and even falsifications in order to provide predictions or new results. They are all *brute* techniques, when analytical methods or more *elegant* methods are not available. On all these accounts, a simulation is not a reliable source of knowledge. Whenever the analytic solution is discovered, a real experiment is possible or new data is available, numerical simulations can be stamped out.

Among the most vocal dissenters, R. Frigg and J. Reiss argue that we do not need a new epistemology in dealing with numerical simulations because the specific problems they raise are not philosophical in nature, but "mostly of a mathematical, psychological or sociological nature" Frigg and Reiss (2009). As tools in applied mathematics, they combine the information the model provides with the existing data and do not bring anything new to philosophy of science.

3 And the Anti-deflationism One

Other philosophers of science are more enthusiastic and believe that new epistemology and methodology are needed for numerical simulations as they lie somewhere between traditional theoretical sciences and the empirical methods of experimentation and observation. For P. Galison, from the beginning, simulations have a new epistemology, as a new method of extracting information from physical measurements, as well as a new metaphysics that presupposed discrete entities interacting through stochastic processes Galison (1996). E. Winsberg suggests that philosophers should be more liberal in interpreting numeric simulations as "a process of knowledge creation, and one in which epistemological issues loom large" and as "a deeply creative source of scientific knowledge" Winsberg (2010).

In my argument I side with Winsberg and add to his program an analysis of the epistemology of evolutionary computation. I concede that the computational deflationism is partially true for some types of computation, but unsatisfactory when it comes to evolutionary computation.

The majority of the garden variety of programs, and perhaps the most successful, fall under the "glorified slide rule" category. But philosophers should not fall for the tyranny of the majority. The deflationism, with its Procrustean strategy, forces all computations into the "glorified slide rule" bedframe. In order to deter philosophers from the "glorified slide rule" stance, I focus on genetic programming and algorithms in evolutionary computation. Evolutionary computation is still a minority, but as my argument goes, it cuts deeper into the computational epistemology.

There are roughly three philosophical contexts of computation: "computation as a rational (logical) process", "computation as mind" and "computation as life". In the first context, computation is related to the general theory of rationality which should include artificial rational agents. Whether a computer needs to follow rules in manipulating symbols or on the contrary needs to mimic the natural learning are issues at the very heart of "computation as mind". They have stirred genuine cultural wars between philosophers (H. Dreyfus and J. Searle are perhaps the leading figures) and the AI community. I show that in the latter context, at least as controversial as the second, novel philosophical issues emerge—more or less loosely related to mainstream epistemology and to philosophy of science. Analyzing "computation as life" paradigm augments and diversifies epistemology, sheds light on simulations in the context of scientific discovery and achievement of new scientific knowledge. I conclude that the computational deflationism is not the preferable one—at least in this latter context.

3.1 Computation and Life

What is the relation between biology and computational processes? In the wake of computer programming, Von Neumann asked "the oldest and the most fundamental of all questions about simulation" Keller (2003):

> **Q_2:** How closely can a mechanical simulacrum be made to resemble an organism?

First off, the biomimetic strategy of interpreting algorithms as "optimal search" was prefigured by A. Turing Turing (1948, 1950).[1] S. Ulam, one of the creators of the Monte Carlo method, suggested that the right question when relating mathematics and computer science to biology is not: "What mathematics can do for biology?", but: "What biology can do for mathematics" Ulam (1972).

Answers to $\mathbf{Q_2}$ are multi-faceted and philosophers have, I claim, a deep interest in analyzing this analogy. One answer is *informational complexity*. Both DNA and computer programs are complex informational processes. G. Chaitin, the father of the algorithmic information theory, offered recently an indicative definition for this area of research Chaitin (2011):

> **Metabiology**: a field parallel to biology that studies the random evolution of artificial software (computer programs) rather than natural software (DNA), and that is sufficiently simple to permit rigorous proofs or *at least heuristic arguments* as convincing as those that are employed in theoretical physics. (my emphasis).

Chaitin assumes that there is an artificial analogue of the natural DNA and life. The analogy cuts both ways: one can interpret a self-reproducing algorithm as more complex than an ordinary algorithm. And one can see the DNA as an algorithm that "outputs" or "computes" the functions of an organism. One possible outcome of "metabiology" is that we may learn about the complexity of natural life by investigating artificial programs. One direction of research in

[1] For a deep analysis of Turing's connectionism, see Teuscher (2002); Copeland and Proudfoot (1996). It worth noting that Turing did prefigure evolutionary computation; see Fogel (1998).

"computation and life" based on complexity is cellular automata, and there are a couple of attempts to discuss their metaphysical and methodological consequences from a philosophical perspective.

Another answer to $\mathbf{Q_2}$, central to my argument, is the idea that life and computation can be both interpreted as *search for optimality*. In the 1930s S. Wright interpreted a biological species as a system that evolves in time by exploring a multi-peaked landscape heuristic of optimal solutions to a "fitness problem" Wright (1932). The operation of optimization of search which is typically performed by an algorithm can mimic a living organism that over a long period of evolution fits the environment. On the other hand the process of adaptation and evolution is not smooth. Organisms are subjected to *random* mutations. Is it a good idea to add randomness to algorithms? There are several types of *stochastic* algorithms each of them being more or less *biomimetic* in their nature. Biomimetic strategies are widely used in robotics and artificial intelligence, but they are almost ignored by philosophers. In genetic programming randomness is not a brute tool, as we shall see in the next sections.

After Turing's serendipitous proposal, evolutionary computation was rediscovered and reinvented at least ten times between the 50s and early 80s Fogel (1998). The milestone is John Holland's seminal work *Adaptation in Natural and Artificial Systems* (1975), which impacted the community of programmers only in the 1980s. Following Turing and von Neumann, Holland was able to see the potential of using the knowledge on natural adaptation process to improving search techniques and applied the principles of natural selection directly to problem-solving algorithms.[2] One fundamental difference, not available in Turing's time, is that selection occurs better at the level of population, not at the level of individuals. Genetic algorithms are iterative procedures of *searching* for the optimal solution to a problem P. They are based on the metaphor of biological processes in which organisms: (a) *non-consciously* adapt to the "environment" P and (b) are selected by a *supraindividual* mechanism such as selection.[3]

Genetic algorithms start from a given number of initial individuals randomly distributed in a given space, called the initial population. The genetic algorithm transforms individuals, each with an associated value of fitness, into a new generation by using the principles of survival-of-the-fittest, reproduction of the fittest and sexual recombination and mutation. Similar to Wright's landscape, the genetic algorithm finds "the most suitable" or the "best so far" solution to the problem by breeding individuals over a number of generations.

The procedure can be stopped by a termination condition: when the sought-for level of optimality is reached or when all the solutions converge to one candidate. The fitness function estimates the fitness to breeding of individuals in accordance with the principle of survival and reproduction of the fittest:

- Better individuals are more likely to be selected than inferior individuals.
- Reselection is allowed.
- Selection is stochastic.

The genetic algorithm ends with a *termination condition* that can be the satisfying of a success predicate or completing a maximum number of steps. The success predicate depends on the user's choice and can be deemed as a pragmatic criterion. The winner is designated at the "best-so-far" individual as the result of the run.

[2] See the background of genetic algorithms in Tomassini (1995); Koza et al. (1999, 2003); Affenzeller (2009); Olariu and Zomaya (2006). An interesting approach posing some foundational questions is de De Jong (2006)

[3] I take here algorithms as abstract, mathematical objects, whereas programs as their concrete instantiation on a machine. A sensitive difference is between genetic algorithms, genetic programming and genetic strategies. See de De Jong (2006).

3.2 Diagram of a Standard Generic Algorithms

Here is an abstract implementation of a genetic algorithm:

```
produce an initial population of
                                individuals      (1)

    WHILE `termination' not met do               (2)
        evaluate the fitness of all
                                individuals      (3)

        select fitter individuals for
                                reproduction     (4)
        produce new individuals                  (5)

        generate a new population by
                inserting some new
                good individuals and
                by discarding some
                'bad' individuals                (6)

        mutate some individuals                  (7)

    ENDWHILE                                     (8)

Call the individual(s) who satisfy
        the 'termination' condition
        the ''best-fit-so-far''                  (9)
```

Before discussing a concrete implementation of genetic algorithms, some paramount features are worth nothing: randomness and stochasticity, and unrepeatability.

Genetic algorithms can be or not random, depending on the mutation operator occurring in step (7) or by selecting the individuals for reproduction in step (5). An algorithm becomes deterministic if exactly one parent is *identically* reproduced or if two parents are combined without adding or losing information based solely on their fitness. "Evolution programming", a precursor of evolutionary computation, was developed in the 1960s with no sexual reproduction, only by mutations at the level of the individuals. The paradigm is adaptation and selection based on competition. The idea of genetic mutation was not well understood in the 1960s: see Fogel (1998); Fogel et al. (1966).

The principles of recombination, selection, and mutation are basically "operators" in the algorithm to generate new individuals. Crossover takes two individuals called parents and produces two new individuals, the offspring, by swapping substrings of the parents. Randomly choosing two parents to mate or randomly deleting or adding information from the parents will make the algorithm stochastic. Mutation is a background redistribution of strings to prevent premature convergence to *local* optima.

In this scenario, weak individuals may survive "by luck" and fit individuals may not be drawn to reproduce. The advantage of a random mutation is that at least some populations, ideally a few only, could escape the traps which deterministic methods may be captured by,

and end up with an unexpected and novel result. For very complex problems, this biomimetic procedure can output results which are definitely not accessible to deterministic algorithms if a delicate balance between the mechanism of selection that decrease variation and those that increase variation (mutation) has been achieved.

Because of the stochastic element, the best are not guaranteed to be selected, and the worst are not eliminated. One can say that the algorithm favors the best and marginalizes the unfit. The selection is not entirely "greedy" hill climbing in the search space. One mechanism is the fitness-proportionate selection: individuals are selected according to their fitness. Other is a tournament in which two individuals are chosen randomly from a population of fit individuals.

Third, genetic algorithms are stochastic in two major respects: both the operation of selection and reproduction are random. That means the results (offsprings) are not direct results of the input data (the parents). This is in direct analogy with the way we can run the tape of life and every time a different rational agent will emerge as the "better-to-fit". There are several consequences to how we interpret the epistemology of the genetic algorithms:

> different runs of the genetic algorithms with the same initial population will render in general different results.
>
> they do not find the global optimum solution to a problem with certainty, but have a better chance to find it stochastically.

Second, a genetic algorithm is abstract and needs an interpretation. To implement a genetic algorithm in a concrete computer program one needs a representation of the individual, a problem-specific fitness function assigning a value to each individual (in this sense, the fitness function is fully defined) and a concrete termination condition. The frequently used representation of an individual is a fixed-length character string similar to a chromosome that encodes potentially the solution to the problem. But it is more challenging to pick the fitness function, the initial population, its size, the mutation rate, crossover strength, or the selection method. Sometimes, an effective terminator condition is sometimes impossible to determine. One can see in what sense genetic programming is *prima facie* under the spell of [2].

These difficulties of genetic algorithms are compensated by their effectiveness in global search. Remember that they maintain a population of solutions which are constantly updated with fitter new individuals and hence avoid local optima. For a certain complexity of the search space, a genetic algorithm has a better chance to find the global optimum. This, I claim, radically changes the epistemological aspects of genetic algorithms. They are very efficient in solving "hard problems" where little or nothing is known about the sought-for structure and when discovering new structures trumps the process of evaluating existing knowledge.

4 EUREQA and its Applications

As advertised, Eureqa is a genetic software designed to discover equations and symmetries that scientists haven't been able to discover Lab (2009). The individuals in this case can be equations, invariants or symmetry groups of a specific set of data. But in general individuals may be models and scientific heuristic methods of search—not necessary mathematical objects. Each individual is tested against a bank of experimental data. Many individuals do not make sense mathematically or do not meet some consistency criteria, so they are killed. Some may fit the data better than others. The software saves these individuals for "breeding", cross-combining a 'father' with a 'mother'. It is claimed that over hundreds of thousands of generations, some extremely fit individuals emerge.

Schmidt and Lipson refined the algorithm until it could handle one of the most complex systems of all: The chaotic double pendulum. Such a pendulum swings its arms in a way that's virtually impossible to predict. Because there is seemingly no pattern whatsoever, it would be almost impossible for a human to find an equation describing the motion of the pendulums. Their evolutionary algorithm, though, was able to breed an equation describing the kinetic and potential energy of the system. Not only that, the equation 'discovered' by Eureqa shows that energy is always conserved. It had 'rediscovered' the first law of thermodynamics, one of those immutable laws of nature.

With Eureqa, not only analytical functions have been discovered from empirical data, but also structures which are highly relevant to physical sciences: Hamiltonians, Lagrangians, laws of conservation, symmetries, and other invariants Schmidt and Lipson (2009). The algorithm was able to infer the *optimal* form of the double pendulum Hamiltonian by avoiding trivial or meaningless solutions. A set of syntactical constraints can be imposed upon the combination rules, the search is constraint by "naturalness", "interestingness" or "meaningfulness". For these types of simulations, genetic algorithms are highly appropriate and are successfully used in discovering regularities or patterns in scientific data. Schmidt and Lipson showed how a genetic program discovers *natural* and *meaningful* invariants for physical systems such as double pendulums or other simple oscillators. Here "relevance" plays a special role because it offers a clear way to separate trivial conservations of a quantity from real useful "invariants". How do we pick the non-trivial ones? Schmidt and Lipson propose a very interesting criterion for non-triviality of invariance: the candidate equations should predict connections between dynamics of subcomponents of the system, which is absolutely crucial for the modeling of mechanisms Schmidt and Lipson (2009). Schmidt and Lipson fed both synthetic and physical data into the algorithm. The algorithm does not produce a single set of equations, but a small set of candidates. In the spirit of the **GA** procedure, the next step is to optimize the balance between the predictive power and the complexity/parsimony of each candidate. From a statistical analysis, they inferred that terms that are frequently used and are more complex have also *meaning*.

5 *A Concrete Result*

To illustrate, let us focus on a recent result by Schmidt and Lipson who showed how genetic programming discovers *natural* and *meaningful* invariants for physical systems such as double pendulums, systems with two springs and other simple oscillators Schmidt and Lipson (2009).[4] With Eureqa, they have discovered not only analytical functions from empirical data, but structures which are highly relevant to physical sciences: Hamiltonians, Lagrangians, laws of conservation, symmetries, and other invariants Schmidt and Lipson (2009). Their algorithm was able to infer the *optimal* form of the double pendulum Hamiltonian by avoiding trivial or meaningless solutions. Their algorithm was able to infer the *optimal* form of the double pendulum Hamiltonian by avoiding trivial or meaningless solutions. Schmidt and Lipson refined the algorithm until it handled the chaotic double pendulum. The genetic algorithm to find an equation describing the kinetic and potential energy of the system and the conservation of energy for this system.

It is claimed that by moving away from discovering equations that fit a set of data, genetic algorithms have become more human-friendly. The type of constraints the programmer imposes upon the search for symbolic regression is "naturalness", "interestingness" or "mean-

[4] The former can exhibit chaotic behavior at certain energies.

ingfulness" of the results. For these types of simulations, it seems that genetic programs are highly appropriate.

Here "relevance" plays a special role. It is easy to find conservation of a quantity or invariants for any given system: in fact there are infinite numbers of conserved quantities for any given system. How do we pick the non-trivial ones? Schmidt and Lipson propose a criterion for non-triviality of invariance: the candidate equations should predict connections between dynamics of subcomponents of the system. More precisely, the conservation equations should be able to predict connections among derivatives of groups of variables over time, relations that we can also readily calculate from new experimental data Schmidt and Lipson (2009).

Schmidt and Lipson fed both synthetic and physical data into the algorithm. The algorithm was able to infer energy laws for each system: Hamiltonians and Lagrangean equations. The algorithm does not produce a single set of equations, but a small set of candidates for the analytical solutions. In the spirit of the **GA** procedure, the next step is to optimize the balance between the predictive power and the complexity/parsimony of each candidate. By calculating the Pareto front of the dependence predictive ability versus parsimony, Schmidt& Lipson found that there are two cliffs where predictive ability jumps rapidly at some small change in complexity.[5]

Schmidt and Lipson warn that their algorithm does not provide a meaning to the discovered analytic solutions. What if we simulate the same type of system but with different masses, elastic constants, lengths etc.? After a dimensional analysis the algorithm was able to identify the metric units of the coefficients involved Schmidt and Lipson (2009). Bootstrapping can also be used to infer laws for more complex systems. From a statistical analysis, they inferred that terms that are frequently used and are more complex have also *meaning*. Trigonometric terms representing potential energy, squared velocities that are associated to kinetic energy and so on.

6 Three Rebuttals and Three Paths

Three problems. Here are some problems that impact generally my analysis. First, unlike Turing computation genetic programs are not universal computational tools. They are worth employing only when other methods of searching failed—especially in cases when we know almost nothing about the structure of the search space. Otherwise, genetic algorithms are most likely outrun by more conventional methods, even by "brute" stochastic search methods. Your car, your phone or your desktop computer do not run genetic programs, and even applications of EUREQA in science are really limited.Second, genetic algorithms are certainly not the only stochastic algorithms used to mimic scientific discovery. In fact there are algorithms, mostly stochastic, that start from data and generate models or even laws of nature: deterministic algorithms are also in the game here. Algorithms based on induction were created in the 1980s by H. Simon, P. Langley, J. Shrager, etc.: for example, "BACON" was able to rediscover Kepler's Law, Prout's hypothesis about atomic structure etc.[6] But BACON was inherently limited: there was no optimization, no combinatorial procedure between previous results, and no termination condition. As some philosophers dissented, such programs cannot surpass the

[5] For computational and pragmatic reasons, we can pick the expression that sits on the edge of the "Pareto cliff", the one that offer the best balance between complexity and predictability.

[6] Other "discovery programs" were: OCCAM, GALILEO, HUYGENS, etc. See Simonton (2004). The philosophical literature on BACON can be found in Simon et al. (1981) and accompanying papers in the same volume.

known problem of induction and do provide explanations.[7]

Three paths. Last but not least, the solutions of a genetic program are in many cases *inscrutable*. Genetic programs are able is to produce symbolic forms of possible laws from data by avoiding some of the problems that mired other techniques: the over-fitting problem, stickiness to local maxima/minima, etc. A deflationist can complain that there is no way to follow the solution of a genetic algorithm, therefore it is not justified. The only aspect which is epistemologically accessible to the scientist is comparing results and deciding the best fit. Previous generations and the evolution itself is in many cases too complicated to follow or alternatively is too stochastic to constitute a justification per se.

Despite these problems, as a first philosophical stab to genetic algorithms, my analysis complements and augments the existing literature on computational epistemology and on numerical simulation. I claim that analyzing them provides a bunch of "philosophical models" that help us understanding some controversial issues in epistemology and philosophy of science. Second, they anticipate perhaps the way scientific methodology and knowledge may look in a couple of decades.

First, in respect of mainstream epistemology, genetic algorithms shed light on a very controversial problem regarding the relation between evolution and rationality. Second, in philosophy of science, by analyzing Schmidt's and Lipson's result, one can see a case of "upward epistemology" in computational science and a rare case in which computation can be potentially used in the context of discovery, not only in the context of justification. Third, and this is more important for my present argument, the genetic algorithms shed lights on the relation between rationality and evolution.

6.1 Rationality and Evolution

Concerning the latter path, several authors (Quine, Dennett, Fodor, i.a.) have suggested that agents which are more rational will survive. In other words, evolution rules out quickly agents which are irrational. According to such a philosophical doctrine, (a) evolution produces individuals which are good approximations to an optimally well designed system and (b) optimally well-designed systems are rational agents. Without further evidence, I claim that the debate between rational and irrational evolutionary agents can be investigated further by scrutinizing genetic algorithms. Genetic algorithms can illustrate the relationship between intuitive and analytic thought and ultimately between the irrationality of luck, the rationality of optimization and the intricate process of evolution de Sousa (2007). Remember that for the same input and on the same computer, the same algorithm does not output the same result at every run. But there is always a demand for convergence to an objective "best run" which i suggests that genetic algorithms can be used in tracking objectivity, rationality and, finally, truth by the mean of a set of stochastic procedures. The general question that one can ask is whether there is a convergence to all genetic algorithms towards a "best-from-better" solution.

Ditto about scientific discovery: in a couple of decades we may (or may not) find ourselves using extensively genetic algorithms (or perhaps a combination of genetic algorithms and neural networks) in scientific discovery. Paraphrasing van Fraassen, science teaches us that the world is richer than what we can conceive with our "naked mind". I interpret genetic algorithms as tools of exploring inconceivable possibilities. Science does not ascertain only to accurate and exact descriptions of the actual world, but explores classes of possible systems—more or less close to the real systems, which are not real in themselves. For example, science is interested in questions such as: "why X does not occur?" where X is a non-actual event that is *relevant* to

[7] An interesting parallel between genetic algorithms and discovery program can be found in Simonton (2004)

real events. Scientists conceive possible systems that could have been instantiated in the world, but aren't. One aspect worth mentioning is that simulations create new possibilities which are not limited by conceivability and definitely not limited by computability in a strong sense. Nevertheless, genetic algorithms are not "blind" explorers, but *optimal* explorers. They help us surpassing the divide between what is conceivable and what is not conceivable in a more or less rational way through optimization. If the human discoverer is limited by conceivability, then genetic algorithms may well reveal in the future that they can transgress this human limit.

P. Humphreys talks about numerical simulations as "epistemic enhancers". In answering some dissenters Frigg and Reiss (2009); Stockler (2000), he claims that the central philosophical novelty of simulations is that we see how humans are "pushed away from the center of the epistemological enterprise" Humphreys (2009). I second here Humphreys, but one component of this dialectic, is the stochastic and heuristic methods as well as chance in the context of scientific discovery. They definitely widen the domain of what is scientifically *discoverable*. For P. Humphreys, simulations are epistemic enhancers as they broaden the set of "what counts as observable and mathematically tractable". "For just as observability is the region in which we, as humans, have extended ourselves in the realm of the senses, so computability is the region in which we have extended ourselves in the realm of mathematical representations." Humphreys (2004).

Without a proof, I claim that in formal epistemology genetic algorithms illustrate the difference between assessment and discovery in terms of reliability. They can decouple our scientific endeavor from our own expectations and help us with *not* looking for the car keys under the street lamp, so to speak. A practical application would be the discovery of "laws" or numerical generalizations where we do not think there are any—and warn us to stop looking for laws where chances are that there are none.

6.2 The Upward Epistemology

Schmidt and Lipson's results are intriguing. One first question is whether we can assimilate genetic algorithms to simulations and if they are simulations, what is simulated? Philosophers acknowledge that some simulations are bottom-up (the cellular automata are good examples) because they are not based on a model or theory of the data, but on general hypotheses and not on "explicit theoretical considerations" Frigg and Reiss (2009); Winsberg (2010). In the lines of Winsberg's suggestion, genetic algorithms help philosophers understand the process of discovery and developing/optimizing novel theories as they display the delicate interplay between rational search and brute search for a solution. Moreover, and this is the normative claim of my paper, genetic algorithms could or should contribute in the future to the scientific discovery. What is also important is to see the role some non-theoretical values play in the process of the termination condition in which a human decides when to terminate the search. This evidently induces an epistemic risk and hence an element of conventionality perhaps in tracking objectivity. All these philosophically rich areas are worth studying.

With its upward epistemology, the genetic algorithm shed new light on several hotly-debated topics in philosophy of science: a) the role of chance in scientific discovery; b) how we discover and optimize regularities or patterns in data c) the role of mathematics as a *constraint* on natural science, rather than a guiding force.

7 The "Three Armies" Objection

An objection to my argument runs like this: adding to the metaphor of the glorified slide rule a couple of elements may resuscitate the deflationism argument. She can contend that nothing genetic algorithms can do, could not have been done by an army of analytic modelers with slide rules, an army of gamblers and an army of breeders. In other words, genetic algorithms add two elements to the background knowledge needed in programming but does not promote simulations from their relatively minor epistemic role. This new form of deflationism, call it the "three armies argument" is hard to dismantle and my only suggestion is to "wait-and-see", but at least be prepared for the moment when a genetic algorithm will be able to discover something that we did not know and that we could not even in principle know. But it is clear that the type of background knowledge involved in scientific discovery may involve in the future evolution, computer programming, and, last but not least, awareness of the role luck and serendipity play in the advancement of our knowledge.

8 Conclusion

It is reassuring to see how evolutionary computation weds the Digital Age with the Biological Age. Purportedly, the whole second half of the 20^{th} century was dominated by advancements in both areas: DNA in 1952, computer programming in the 60s. But the discovery of genetic algorithms and its widespread usage in late 1980s marks the beginning of a new era when the Digital meets the Biological. And the philosopher can only wait for the foreseeable moment of the informational singularity when artificial intelligence will compete with humans Chalmers (2010).

My humble philosophical prediction is that genetic algorithms, or some more "evolved" offspring of theirs, will be there at the "singularity" party–if there shall be any.

References

Affenzeller, M. (2009). *Genetic Algorithms and Genetic Programming: Modern Concepts and Practical Applications*. Number v. 6 in Numerical insights. CRC Press, Boca Raton, Fla.

Beavers, A. F. (2011). Recent developments in computing and philosophy. *Journal for General Philosophy of Science*, 42(2):385–397.

Chaitin, G. (2011). Metaphysics, metamathematics, and metabiology. In Zenil, H., editor, *Randomness Through Computation: Some Answers, More Questions*. World Scientific.

Chalmers, D. (2010). The singularity: A philosophical analysis. *Journal of Consciousness Studies, 17*, 9(10):765.

Copeland, B. J. and Proudfoot, D. (1996). On Alan Turing's anticipation of connectionism. *Synthese*, 108(3):361–377.

de De Jong, K. A. (2006). *Evolutionary Computation*. MIT Press: A Bradford Book, Cambridge MA, 1st edition.

de Sousa, R. (2007). *Why Think? The Evolution of the Rational Mind*. Oxford University Press, USA.

Eason, R., Rosenberger, R., Kokalis, T., Selinger, E., and Grim, P. (2007). What kind of science is simulation? *Journal of Experimental & Theoretical Artificial Intelligence*, 19(1):19–28.

Floridi, L. (2011). *The Philosophy of Information*. Oxford University Press, USA.

Fogel, D. B., editor (1998). *Evolutionary Computation: The Fossil Record*. Wiley-IEEE Press, 1 edition.

Fogel, L., Owens, A., and Walsh, M. (1966). *Artificial Intelligence through Simulated Evolution*. John Wiley.

Frigg, R. and Reiss, J. (2009). The philosophy of simulation: Hot new issues or same old stew? *Journal for Epistemology*, 169(3):593–613.

Galison, P. L. (1996). Computer simulations and the trading zone. In Galison, P. and Stump, D. J., editors, *The Disunity of science : boundaries, contexts, and power*. Stanford University Press, Stanford Calif.

Hartmann, S. (2008). *Modeling in Philosophy of Science*. Ontos Verlag, Heusenstamm bei Frankfurt. Book, Whole.

Hendricks, V. F. (2007). *Mainstream and Formal Epistemology*. Cambridge University Press, 1 edition.

Humphreys, P. (2004). *Extending Ourselves: Computational Science, Empiricism, and Scientific Method*. Oxford University Press, USA.

Humphreys, P. (2009). The philosophical novelty of computer simulation methods. *Synthese*, 169(3):615–626.

Keller, E. (2003). Models, simulation, and 'Computer experiments'. In Radder, H., editor, *The Philosophy of Scientific Experimentation*, pages 198–215. University of Pittsburgh Press.

Koza, J. R., III, F. H. B., Andre, D., and Keane, M. A., editors (1999). *Genetic Programming III: Darwinian Invention and Problem Solving*. Morgan Kaufmann, 1st edition.

Koza, J. R., Keane, M., Streeter, M., Mydlowec, W., Yu, J., and Lanza, G., editors (2003). *Genetic Programming IV: Routine Human-Competitive Machine Intelligence*. Kluwer Academic Publishers, Norwell, Mass.

Lab, C. M. (2009). Eureqa.

Olariu, S. and Zomaya, A. Y., editors (2006). *Handbook of Bioinspired Algorithms and Applications*. Chapman and Hall/CRC, 1 edition.

Samuel, A. (1959). Some studies in machine learning using the game of checkers. *IBM JOURNAL OF RESEARCH AND DEVELOPMENT*, 3:211–229.

Schmidt, M. and Lipson, H. (2009). Distilling Free-Form natural laws from experimental data. *Science*, 324(5923):81–85.

Simon, H. A. (1969). *The Sciences of the Artificial*. M.I.T. Press, Cambridge, Mass.

Simon, H. A., Langley, P. W., and Bradshaw, G. L. (1981). Scientific discovery as problem solving. *Synthese*, 47(1):1–27.

Simonton, D. K. (2004). *Creativity in Science: Chance, Logic, Genius, and Zeitgeist*. Cambridge University Press, Cambridge.

Stockler, M. (2000). On modeling and simulation. In Carrier, M., Massey, G., and Ruetsche, L., editors, *Science at century's end : philosophical questions on the progress and limits of science*, pages 355–373. University of Pittsburgh Press, Pittsburgh.

Teuscher, C. (2002). *Turing's connectionism: an investigation of neural network architectures*. Springer.

Tomassini, M. (1995). A survey of genetic algorithms. *Annual Reviews of Computational Physics*, 3:87–118.

Turing, A. (1948). Intelligent machinery. Technical report, National Physical Laboratory, London.

Turing, A. M. (1950). Computing machinery and intelligence. *Mind*, 59(236):433–460.

Ulam, S. M. (1972). Some ideas and prospects in biomathematics. *Annual Review of Biophysics and Bioengineering*, 1(1):277–292.

Winsberg, E. (2001). Simulations, models, and theories: Complex physical systems and their representations. *Philosophy of Science*, 68(3):S442–S454.

Winsberg, E. (2010). *Science in the Age of Computer Simulation*. University of Chicago Press.

Wright, S. (1932). The roles of mutation, inbreeding, crossbreeding and selection in evolution. In *Proc of the 6th International Congress of Genetics*, volume 1, page 356366.