Qualities, challenges and future of genetic algorithms: a literature review

Aymeric Vié¹

¹ Mathematical Institute, University of Oxford

November 11, 2020
Abstract

The main creator of GAs, John Holland, wrote that computer programs that evolved in ways similar to natural selection could solve complex problems that even their creators do not fully understand. 30 years later, Genetic Algorithms (GA) have been widely applied with significant achievements in the modelling of evolutionary systems, and complex optimisation problems. This review intends to summarise the key qualities, current challenges and future perspectives faced by this technique.

We first present the main merits of the GA search algorithm. Their implicit parallelism and evolutionary operators allows an optimal balance between exploration and exploitation. They thrive in identifying good solutions in large, rugged search spaces. They have desirable convergence properties, offer high flexibility, and impose very few assumptions on the nature of the solutions being evolved. They allow a realistic modelling of evolutionary systems and innovation dynamics, especially in economics, finance, and AI.

We then outline the main challenges faced by GA applications. Often criticized for their computational complexity, GAs’ computational efficiency is a first challenge to address to handle dynamic or more complex problems, likely with further use of parallelism. The difficult calibration of GAs parameters, determinant to its performance, remains an open area of investigation. The specifications of the canonical GA, notably the choice of individual representation, the design of the fitness landscape, and initial population sampling, impose some implicit constraints on the search space being explored, and the solutions being evolved, requiring further innovations to achieve open-ended evolution and robustness.

Most of GAs achievements have so far been achieved from a restricted subset of our knowledge on natural evolution and genetics. In the continuity of the first formulations of GAs, we argue that a deeper, renewed inspiration from the state of the art of biology and genetic research, carries the potential for fundamental discoveries in GA design. The evolution of evolvability, the ability to produce the most improving kind of variation for selection, interactions between mutations -epistasis-, loci mutations affecting several phenotype traits -pleiotropy-, and the modularity of biological networks, are only few examples of the many crucial advances that can push GAs capacities to a new level. Incorporating this more advanced knowledge on how natural evolution occurs allows us to engineer more complex and realistic populations and to improve efficiency, configuration and robustness of our algorithms. By considering not only DNA, but also the chromosome that carries them, as both subject to evolution, a novel class of GAs oriented towards artificial life and open ended evolution can emerge. By increasingly and more faithfully seeking inspiration from the natural process that inspired their previous successes, what could GAs achieve in the future?
# Contents

1 Introduction 4

2 Genetic Algorithms 8
   2.1 Initial sampling 9
   2.2 Fitness selection 10
   2.3 Crossover 10
   2.4 Mutation 11

3 Evolving good solutions in complex environments 12
   3.1 Genetic Algorithms as powerful complex landscapes explorers 12
      3.1.1 Exploring large search spaces 12
      3.1.2 Exploring rugged search spaces 13
   3.2 Solving complex problems 13
      3.2.1 A robust method for schema optimization 14
      3.2.2 Evolving neural networks by exploring learning architectures 14
      3.2.3 Multi-objective optimisation 15
      3.2.4 Genetic programming 15
   3.3 Emergence, novelty and open-endedness 16
      3.3.1 GAs are more than optimizers: they create novelty 16
      3.3.2 Artificial life and emergence 17

4 Limitations, challenges and perspectives 19
   4.1 Computational efficiency and cost 19
      4.1.1 Selection methods for computational efficiency 19
      4.1.2 Implicit and explicit parallelism 20
      4.1.3 Co-evolution 21
   4.2 Parameter configuration 21
      4.2.1 Parameter tuning is crucial to GA performance 21
      4.2.2 Parameter control 24
   4.3 Realism, robustness and endogeneity 26
      4.3.1 Initial population sampling 26
      4.3.2 Representation 26
      4.3.3 Fitness evaluation 29

5 Conclusion 30
1 Introduction

Inspired from Darwin’s theory of evolution, the Genetic Algorithm (GA) is an adaptive search algorithm that simulates some of the evolution processes: selection, fitness, reproduction, crossover (also denoted recombination), mutation. As a search algorithm, it produces an implicit enumeration of a subset of the search space, using population-based meta-heuristics. Evolution in a GA proceeds as the population of organisms performs an adaptive walk. In the class of population-based stochastic algorithms (evolutionary algorithms), they are stand out by their combination of parallel population-based search, and operators inspired from natural evolution. A population of individuals (organisms, strategies, individuals, objects...) characterized by a genetic sequence inducing physical characteristics, evolves under selection pressure in an artificial environment. The fittest individuals reproduce, while mutations and exchanges of genetic material explore new individual characteristics. The implicit parallelism embedded by evolution happening at each individual level, and the search efficiency allowed by the evolution operators, allow GAs to balance exploitation of promising individual genetic sequences, and exploration of new promising ones. While originally designed to the study of adaptation in natural systems (J. Holland, 1975), most have been developed as optimization tools (Whitley, 1994), in a high variety of research domains. What features of genetic algorithms allow them to be both an insightful modelling technique and a powerful optimisation tool?

While the literature in genetic algorithms is quite large, and contains a lot of information on the pros and cons of using GAs, we have found these elements to be quite scattered across numerous articles and materials. GAs are increasingly applied in a variety of research domains, and to tackle a very large set of problems. We then find a thorough compilation of the properties and specifications of GAs, that discusses their modelling and optimizing abilities in the light of recent research, to be a useful contribution to the field. We aim at highlighting both advantages and drawbacks of genetic algorithms in these two uses, identifying the key challenges they face, and drawing promising perspectives for the field.

Genetic algorithms are characterized by their ability to efficiently explore rugged, irregular search spaces, with desirable convergence properties. On an optimization perspective, GAs are particularly suitable to explore search spaces that are large, sparse, non differentiable. They excel at identifying complex interactions between the decision variables at play. They exhibit a strong performance in identifying near-best solutions in very large spaces of possible solutions, allowing to quickly deliver good solutions to quite hard problems. On a modelling perspective, the exploration behavior of GAs adequately connects with innovation and learning dynamics. They realistically represent the behavior of intelligent particle systems in which microscopic elements are evolving in a given environment with experience and open exploration.
GAs thrive in contexts where information and solution structure are scarce, and offer high flexibility. As robust, "weak" methods, GAs work well to approach phenomena or problems with unknown structure. The flexibility of the chromosome encoding and the choice of the fitness evaluation function allow to adapt to various contexts, and to formulate very few assumptions on the form of the outcome to a given evolution process. They also allow to consider a high diversity of problems in various fields, and with different objectives: simulation, engineering design, prediction, optimal control... While the choice of chromosome encoding and the necessity of a way to determine fitness are themselves assumptions that eventually set a limit to the flexibility of the GA, the literature has developed numerous variations of encoding mechanisms to tackle different problems. Notwithstanding their ability to thrive with limited information, GAs seem to work well with user intuition or knowledge of the problem that is presented, notably inputting some good candidate solutions in the initial generation of chromosomes. They appear suitable to solve complex problems even their creator does not fully understand (J. Holland, 1992).

The desirable properties of GAs combined with other methods in hybrid approaches, especially in the field of neural networks, have achieved significant achievements. The emerging approach of neuroevolution in particular, uses evolutionary algorithms to optimise neural networks (Stanley et al., 2019). Instead of training weights on stochastic gradient descent and backpropagation, neuroevolution programs evolves building blocks, hyperparameters, architectures, and even learning rules. Applications of Deep GA methods have expanded the possible uses of GAs, improved performance of neural nets, and solved high dimensional problems unsolvable by traditional reward-maximizing algorithms (Such et al., 2017). Recently, a GA trained a DNN with over four million parameters, with very competitive results (Such et al., 2017). Another hybrid domain associates evolutionary algorithms with reinforcement learning approaches, comprehensively reviewed by Drugan, 2019. Innovative solutions to improve GA efficiency and parametrisation are likely to fuel further similar progress.

In terms of modelling, which constitutes their original purpose, GAs show emergence of novel, creative behaviors, and generate complex dynamics of interest to understand the evolution of the system being represented. Due to their ability to balance exploration with exploitation of the search space, GAs constitute an adequate approach to model open-ended evolution systems, with a diversity of learning outcomes. More generally, they are adequate to describe evolutionary systems, from biology -their initial object of study- to finance, market ecology, cultural dynamics or strategies in game theory. Genetic algorithms do not only provide us with an optimal outcome: they also deliver an evolutionary trajectory towards an optimal result, outlining their policy relevance, adequacy for modelling purposes, and transparency.

Demands in computational time and efficiency impose constraints on GA de-
sign. In particular, the evaluation function must have a low computational cost, as it is iterated a high number of times, depending on the number of individuals in the generation, iterations and constraints. The population size is detrimental to computation effort, but benefits in terms of diversity. Establishing an optimal balance between exploration and exploitation, while satisfying acceptable computation efficiency targets, remains a priority research avenue. The implicit parallelism (J. Holland, 1975, 1992) of GAs have provided an adequate means to improve computational efficiency of GAs. This computational parallelism, i.e., ability to explore simultaneously different possibilities in an efficient way (Mitchell, 1998), combined with the GA as an intelligent strategy for choosing the next set of sequences to evaluate, explains the success of GAs in rapidly identifying promising regions in a very large, or irregular space. Various variants of the canonical GA, such as distributed or island genetic algorithms, have been designed to take advantage of the capacities of parallel computing.

The configuration of the program parameters, and its computation efficiency, are key determinants of the success or failure of the genetic algorithm. This successful exploration of the search space is the result of a delicate parameter configuration that balances selection pressure and preservation of diversity in the search process. If the selection pressure is too high, the population loses diversity, and the GA may prematurely converge to local optima, or lose its desirable exploration properties by only scanning the immediate-reward subset of the search space, preventing the development of more complex, fittest strategies. On the other extreme, if selection pressure is too low, the convergence speed may be quite low if even positive. A too low mutation frequency in the population causes loss of diversity and convergence to local optima, while a too high mutation frequency results into pure random search. How to achieve the right balance between diversity and selection pressure is a major challenge for genetic algorithms, and is inherently linked to the general issue of GA calibration.

Self adaptation, letting the GA configuration and operators being themselves subject to evolution, constitutes a promising future direction to tackle this challenge of parameter configuration. As we outline, a number of significant challenges faced by genetic algorithms deal with the least natural elements of the specifications. If our choices of genotype encoding, fitness evaluation function, and parameters settings are the problem, what is the solution? How can we search through a large space of possible rules, parameters values, that non-linearly and jointly impact GA performance? It turns out that this GA specification problem we are describing belongs to the category of problems GAs are evidently quite good at investigating. In the continuity of closely related fields of Genetic Programming and Automatic Programming, and General Artificial Intelligence, self-adaptation and meta-GAs were early identified as promising areas for future research.

The representation of solutions, the definition of a fitness metric, and the
Qualities, challenges and future of genetic algorithms

A. Vié

Population sampling are key challenges to design realistic, robust GAs. The free exploration of the search space is first conditioned on the way solutions are encoded. While most GAs applications choose for example a binary encoding in which chromosomes are vectors of zeros and ones, this modelling specification limits the space the GA explores. Other specifications such as integer encoding, grammatical encoding, or even the vector form, or the finite-size feature of chromosomes, are assumptions that impact the behavior and outcome of the GA. Before evaluating what selection or mutation operators are relevant to the problem we desire to solve, or the phenomena we want to study, determining how evolved objects are encoded is a crucial step.

A similar challenge holds for the evaluation function when measuring the fitness of each phenotype. While the evaluation function is often included in the problem itself, e.g. function minimization, maximizing score... But often, different alternatives are available. If we were to study how a GA would play a given game with scores, should we evolve the GA under score maximization, or win chance maximization? Would these two alternative choices impact the outcome strategy of the GA in this game? The choice of the evaluation function highlights the issue of GA robustness to perturbations of the way fitness is measured. GAs are quite efficient at exploiting loopholes, or niches in the fitness landscape, requiring a careful design of this environment.

While most of GA applications develop an exogenous fitness evaluation, endogenising fitness, as well as opening solution encoding to adaptation, may develop a new class of genetic algorithms with greater modelling and problem solving capacities.

Finally, as GAs are population-based stochastic algorithms, we naturally consider the issue of path dependence and sampling bias. We argue that while efforts in the design of initial solutions have often been left behind in previous discussions, more advanced sampling methods offer a promising direction to improve GAs regularity and diversity balance. Latin Hypercube sampling and Sobol sequences appear to us as valid candidates to formulate balanced, diverse initial populations. How the evolutionary trajectory depends on the initial population, and means to start with unbiased samples, constitutes an important key to the GA robustness.

Innovations required to solve the above challenges, acquire a deeper understanding of the functioning of GAs, and extend their use to more complex settings, notably in dynamic environments, may stem from deeper inspiration from biology. Historically inspired from genetics and evolutionary biology, we acknowledge that much of the progress achieved by GAs has been inspired by a small subset of evolutionary biology: notably selection and mutation. More recently, biology has inspired GA research on evolvability, indirect encoding, or canalization. More complex relations between genome and phenotype - pleiotropy, polygeny, modularity of genetic networks- are as many potential additions of interest to the GA design toolbox. The success GAs have enjoyed from these notions encourages us to see tighter collaborations between the fields of genetic algorithms and evolutionary biology, as an endless source of new dis-
Qualities, challenges and future of genetic algorithms

A. Vié

coveries and improvements.

We first introduce the genetic algorithm technique and its functioning in section 2. Their qualities in exploration of large, complex search spaces, optimisation qualities, and close relation to emergence and novelty, are described in section 3. The main issues and challenges faced by GAs, in particular computation efficiency, parameter configuration and robustness, are presented in section 4, that reviews our these challenges were faced, and what novel perspectives are emerging. Final section 5 concludes on the qualities and challenges of genetic algorithms, and introduces important perspectives related to renewed inspiration form biology and genetics, self-adaptation of GAs, and promising connections with open-ended evolution and AI.

2 Genetic Algorithms

In the search for creating artificial intelligence and artificial life, computers scientists have taken inspiration from natural systems. The first computers were notably used to model the human brain, imitate human learning, and simulate biological evolution (Mitchell, 1998). The first inspired neural networks, the second machine learning, and the third one evolutionary algorithms. Aiming at studying the phenomenon of evolution and adaptation in nature, at evolving a population of candidate solutions to a specific problem, or at creating self-programming algorithms, genetic algorithms appeared in the 1960s from the initial works of J. Holland (1975, 1992), Rechenberg (1978).

The biological terminology often used to describe genetic algorithms is detailed by Mitchell (1998), and we only here use some of these terms. We denote chromosome a candidate solution to a problem. Genes are the elements, or building blocks, of the chromosome. We can see each of them as encoding a trait of the solution, and the location of the trait on the gene is its locus. The different possible settings for the trait are called alleles. The collection of all chromosomes taken together is the genotype. It determines the physical characteristics of the organism, the phenotype, by a mapping between the genotype and the phenotype, that specifies how the genetic material shapes the organism features. GAs typically consider haploid individuals: characterized by a single chromosome, which is often modelled as a bit string.
Figure 1: An instance of genotype to phenotype mapping, in which the genome is encoded as a binary string. Objects evolve in the space \( \{0, 1\}^3 \). Each gene controls a given characteristic of the organism: color, size, shape.

The evolution at play in GAs takes place in a search space: the space of possible solutions, endowed with a distance metric between these solutions. The relationship between candidates’ phenotypes and fitness is often described using the fitness landscape (Wright, 1931), which can have hills, peaks, features similar to physical landscapes, or even more complex structure. GA operators of crossover, mutation, are viewed by Mitchell, 1998 as means to move a population of chromosomes (particular points in the search space) around the fitness landscape defined by the fitness function, and often, also to some extent by the other components of the population. The fitness function is a metric of the performance of each chromosome. Each iteration of the process is called a generation, and the set of generations is a run. Simplest, canonical genetic algorithms start with an initial chromosome. Each of their iterations involves three main operators: selection, crossover and mutation, that admit a large diversity of implementations. The algorithm ends with a terminal condition, often after a given number of iteration, or when a particular solution has been obtained.

### 2.1 Initial sampling

The canonical GA starts with a random population of chromosomes, and evolves this population into subsequent generations through the above iterated evolutionary operators. The objective of initial sampling is to spread uniformly the initial solutions in the search space, in order to be more likely to find promising regions (Mirjalili, 2019).

Figure 2: An example of random initial sampling for a population of 3 individuals, in the space \( \{0, 1\}^3 \).
2.2 Fitness selection

The fitness of each chromosome is determined, often by a fitness evaluation function that assigns phenotypes to a fitness value. Fitness evaluation is often assumed *exogenous* in applications of GAs and then takes this form of a function. In uses of GAs as modelling tools, fitness can be endogenised by the means of a simulation.

\[
\begin{pmatrix}
0 & 1 & 0 \\
1 & 0 & 1 \\
0 & 0 & 0 \\
\end{pmatrix}
\]

gives

\[
\begin{pmatrix}
1 \\
2 \\
0 \\
\end{pmatrix}
\]

Figure 3: An example of fitness selection. The exogenous fitness evaluation measures how close the individuals are to \( \bullet = (1 \ 1 \ 1) \). The two highest fitness scores in purple are selected as parents for crossover.

Typical selection methods are based on fitness: the classical fitness proportionate "roulette-wheel" (Mirjalili, 2019), elitism (keeping the best chromosome in the next generation, (C. Ahn and Ramakrishna, 2003)), rank selection, tournament selection (Miller, Goldberg, et al., 1997). Alternative selection methods such as truncation selection, sigma scaling, local selection proportional selection, are reviewed by (Mirjalili, 2019). In their Niched Pareto GA, Horn et al., 1994 introduced Pareto tournaments, selecting two individuals and \( n \) others in a comparison set on which to evaluate dominance. Concerns on maintaining balance and exploration have encouraged the development of selection methods rewarding diversity, such as crowding sharing rules (Horn et al., 1994), the "1/4+" selection method, explicit fitness-sharing in genetic clusters (Goldberg, Richardson, et al., 1987), novelty search (Lehman and Stanley, 2011), or quality-diversity algorithms (Pugh et al., 2016). The latter uses rank rather than fitness score to operate selection, and operates a random selection in the top 20% of the population. This is used to prevent stronger individuals from quickly dominating the population, and driving the genetic diversity down too early (Packard, 1988).

2.3 Crossover

After selecting promising parents based on fitness or rank, the crossover (recombination) step combines their genetic material to create two new solutions in the population. Most of the literature in GAs uses the single-point, or double-point crossover, even though several variants and alternatives do exist (Mirjalili, 2019). The crossover allows to explore other regions of the search space, and to combine successful lower-level solution building blocks together in order to construct higher-order good solutions.
2.4 Mutation

Maintaining diversity over generations is essential to GA performance (Horn et al., 1994). The mutation operator is essential in maintaining this diversity, as it provides an insurance against the development of uniform populations incapable of further evolution (J. Holland, 1992). Once again, a diverse set of mutation operators have been used in the literature (Mirjalili, 2019); in particular, uniform mutation that changes the value of a random allele in the chromosome (uniform mutation), or Gaussian mutations. To handle changing fitness landscapes in which the optimum changes over time, Grefenstette et al., 1992 introduced a partial hyper-mutation operator, which essentially replaces a given percentage of the population with random entries. This allows the GA to maintain a continuous level of exploration of the search space.

Now that we presented an overview of the definition and functioning of the genetic algorithm, from initial sampling to terminal conditions, and its operators, let us now move to enumerating the merits and drawbacks of this approach. First, the combination of implicit parallelism and evolutionary operators allow the GAs to be very efficient in handling large and rugged search spaces.
3 Evolving good solutions in complex environments

3.1 Genetic Algorithms as powerful complex landscapes explorers

3.1.1 Exploring large search spaces

In a stark contrast with exhaustive, enumeration or random search-based methods, GAs are suitable to solving sparse problems. That is, problems for which the number of good solutions is sparse with respect to the search space size (Whitley, 1994, Maulik and Bandyopadhyay, 2000). GAs evolve a multitude of strings over many regions simultaneously. By their ability of combining strings containing good partial solutions, GAs focus their attention on the most promising locations of the search space (J. Holland, 1992). In the binary representation approach, the schemas (building blocks) of the solutions are implicitly evaluated by partitioning the search space into a number of hyper-planes (J. Holland, 1975).

GAs in such very large, high-dimensional spaces, have reached significant performance in pattern recognition and clustering in artificial and empirical data sets with both overlapping and non-overlapping class boundaries, in various dimensions and number of cluster ranges (Maulik and Bandyopadhyay, 2000). GA-clustering algorithms notably outperformed than the traditional K-means algorithm. Similar performance was achieved in computational protein design (Street and Mayo, 1999) molecular geometry optimization by Deaven and Ho, 1993. Determining the lowest energy configurations of a collection of atoms is NP-hard and covers a very large space of possible solutions, as the number of candidates scales exponentially with the number of atoms. Nevertheless, the GA found the best structure, in spite of strong directional bonds between different structure. By so doing, the GA outperformed simulated annealing (SA). In large flow-shop sequencing problems, GAs reached near-optimal solutions more quickly than SA (Reeves, 1995), or than local discriminant analysis (Varetto, 1998), showing its relevance for large and difficult combinatorial problems. GAs have been developed by Packard, 1990 to evolve prediction models. Searching through the -huge- space of sets of conditions and predictions, the algorithm identified regions of predictability in stock market data, and obtained robust results in out of sample forecasting. Axelrod et al., 1987’s GA application to the Prisoner’s dilemma efficiently scanned over 264 (16 quadrillion) strategies, exploited some weaknesses in sample strategies to perform better than the optimal tit for tat rule (cooperate unless the opponent has defected, then defect), and developed tit for tat on its own. Recently, Vié, 2020 used a GA to identify optimal strategies in completely open environments that admitted countably many alternatives, and Mirjalili et al., 2020 reconstructed images based on random samples. Whether they explore the set of prediction rules (Waheeb and Ghazali, 2019, Han et al., 2019), of computer programs (Devarriya et al., 2020), of deep neural network architectures (Chung and Shin, 2020), strategies in games
Qualities, challenges and future of genetic algorithms

3.1.2 Exploring rugged search spaces

A traditional technique to explore and optimize in landscapes is hill-climbing, or variants of gradient descent: start at some point, and follow the path to the greater improvement to the quality of the solution. However, as the landscape becomes more rugged, complex, irregular, with many high points, tunnels, bridges or even some more convoluted topological features, finding the right hill and which way to go becomes increasingly hard (J. Holland, 1992). GAs do not require the optimisation landscape to be differentiable, uni-modal (Maulik and Bandyopadhyay, 2000) or to not exhibit particular topologies. Maintaining a population of solutions rather than a single solution, GAs are less vulnerable to premature convergence to local optima, as long as a sufficient level of population diversity is maintained (Maulik and Bandyopadhyay, 2000, Wiransky, 2020). Some examples are optimization of the Eggholder function, known for its large number of local minima, and for which we can analytically derive the global minimum, and the Himmelblau’s function that admits four global minima.

When dealing with multi-modal fitness landscapes in which several global extrema coexist, GAs have often been criticised for converging to an arbitrary instance of global extrema due to genetic drift, and sampling bias during search, rather than maintaining a stable distribution of the population around the different optima. This issue is solved by introducing randomness and different GA sampling, to obtain asymptotic unbiasedness and the adequate distribution of solutions as the number of GA runs increases: see notably such an application for the optimization of the Himmelblau’s function (Wiransky, 2020). Alternatively, the introduction of niching sharing rules to avoid clustering of solutions in one of the extrema is an adequate means to prevents genetic drift arbitrary convergence. These rules allow convergence to the four minima of the Himmelblau’s function in a single run (Wiransky, 2020).

3.2 Solving complex problems

GAs provide a robust approach for optimization problems, because of their flexibility, and efficient processing of solutions building blocks. Some major developments of GAs as problem solvers have taken place in the evolution of deep neural networks, multi-objective optimization, and more generally problems that are not fully understood by their decision maker.
3.2.1 A robust method for schema optimization

GAs are recognized as a "weak" or "robust" method (Whitley, 1994) for their applicability to a high diversity of problems, and their ability to identify good candidates with very few assumptions made on the solutions. Previous research has established guarantees of convergence. Viewing a GA as a finite-state Markov chain, Bhandari et al., 1996 showed that the canonical GA delivers the optimal string as the number of iterations goes to infinity, with a nonzero mutation probability. In well behaved landscapes, genetic algorithms results matched gradient type methods outcomes, in particular for nonlinear constrained optimization (Homaifar et al., 1994).

This efficiency and convergence result was initiated by Holland’s schema theorem. J. Holland, 1992 states that good solutions -genotypes- to a given problem tend to be composed of good schemas, i.e. building blocks: individual or groups of genes alleles. As the GA is evaluating some number n of genomes, it is by so doing evaluating the fitness of a much larger number of schemas. Holland argues that what the GA optimizes generation after generation, are not genomes, but those schemas. The Schema Theorem states that the probability of survival under mutation is higher for lower-order schemas. Short low-order schemas whose fitness is higher than average will occupy an exponentially increasing share of samples over time. In this perspective, crossover operations aim at combining good schemas together to create even better higher order schemas. By so doing, the GA is capable of extracting meaning from noisy or imprecise information, or to detect trends that are too complex to be noticed by humans or standard techniques (Metawa et al., 2017).

GAs can develop good solutions to problems we do not fully understand. For very sparse problems, the efficiency of the GA algorithm was improved when the class, or "shape" of desired solution was assumed (Deaven and Ho, 1993). GAs are however also relevant in problems for which we typically have little information on what optimal solutions would look like, and in which exploration of diverse candidates is desirable. In analysing insolvency risk with a GA, Varetto, 1998 showed that the GA obtained results faster than local discriminant with more limited contributions from the financial analyst. More recently, this low-information requirement has been useful for heart disease diagnosis (Reddy et al., 2020), design of neural network architectures (Suganuma et al., 2017), production scheduling (Nguyen et al., 2017), classification tasks such as detection of video change (Bianco et al., 2017), or determination of new functions to detect breast cancer (Devarriya et al., 2020).

3.2.2 Evolving neural networks by exploring learning architectures

Their versatility, and usually lower performance than specialized optimization method when the problem structure is well known, explain the success of hybrid methods that combine GAs with other approaches (Whitley, 1994), such as k nearest neighbor classification in dimensionality reduction (Raymer et al., 2000). Quite early on, GAs have been successfully applied to feature subset
selection (Yang and Honavar, 1998) and dimensionality reduction (Raymer et al., 2000). This preceded further combinations with neural networks, with very recent successes. The 1990s saw the first works on evolving network architecture, weights, or even the learning rules used by neural networks (Kitano, 1990, Kitano, 1994). Years later, GA-assisted neural networks have shown superior performance compared to fixed geometry neural nets and traditional nonlinear time series techniques in modelling daily foreign exchange rates (Waheeb and Ghazali, 2019), daily bitcoin prices (Han et al., 2019). Chung and Shin, 2020 used an hybrid GA with a convolutional neural network (CNN) that outperformed other models. They used the GA to identify optimal CNN architecture. A large number of recent works have applied GA methods in combination with machine learning techniques: traditional neural networks also named Evolutionary artificial neural networks (bank performance prediction: Ravi et al., 2008, cotton yarn quality Amin, 2013, time series forecasting: Donate et al., 2013, machine productivity: Azadeh et al., 2013, modelling of cogeneration processes: Braun et al., 2016, air-blast prediction: Armaghani et al., 2018, controller design: Abd-Elazim and Ali, 2018, support vector machines (parameter selection Zhao et al., 2012, intrusion detection: Ahmad et al., 2014 and Raman et al., 2017, hospitalisation expense modelling: Tao et al., 2019) and case-based reasoning (corporate bond rating: Shin and Han, 1999, bankruptcy prediction: H. Ahn and Kim, 2009).

3.2.3 Multi-objective optimisation

Multi-objective optimization has been an important area of applications of GA as optimizers (Horn et al., 1994). Notably, Lwin et al., 2014 obtained an algorithm capable of designing high quality portfolios in spite of numerous constraints, handling more than a thousand different assets, and outperforming the state of the art programs. It is not surprising other applications moved to multi-criteria decision making with environmental factors: optimal energy storage units placement (Ghofrani et al., 2013), wind farm development (Dhuny et al., 2020), dynamic greenhouse environment control (Jin et al., 2020), car engine design (Tayarani-N et al., 2014). Multi modal multi-objective optimisation has been an area of intense research in evolutionary computation more generally (Tanabe and Ishibuchi, 2020). Designing supply chain networks facing multiple constraints, such as multi stages, multi products, multi plants with maximum capacities, has also been an area of success of GA applications (Altiparmak et al., 2009). In this case, their steady state GA was shown capable of handling dynamic environment, and stochastic demands. The NP-hard problem of bank lending decisions was solved by Metawa et al., 2017 who obtained with a GA a substantial reduction in screening time, and increase in bank profit.

3.2.4 Genetic programming

The evolutionary approach of the GA inspired a field known as Genetic Programming (GP): evolution-guided design of computer programs to solve a given
task, starting from basic operators. How can computers learn to solve problems without being explicitly programmed, how can computers be designed for a given task without being told exactly how to do it? Genetic programming approaches these questions with GAs to evolve populations of computer programs, and those have succeeded to evolve correct programs to solve diverse problems, such as optimal control, discovering game strategies, planning, sequence induction, symbolic regression, image compression, robotics... (Koza and Koza, 1992, Koza, 1995). One of the early applications of this perspective was the use of GAs to engineer cellular automata to perform computations (Das et al., 1994). Recently, related applications have conducted to improve the performance of robots behavior exposed to situations or damage not encountered before (Cully et al., 2013), enabling more robust, autonomous robots that evolve similarly to animals. Much progress has been done on the design of real-time adaptive trading systems (Dempster and Jones, 2001, Dempster and Leemans, 2006, Martinez-Jaramillo and Tsang, 2009). A very recent review of the genetic programming literature has been presented by Langdon, 2020. More recent applications have focused on the design of neural network architectures (Suganuma et al., 2017), production scheduling (Nguyen et al., 2017), classification tasks such as detection of video change (Bianco et al., 2017), and the determination of new functions to detect breast cancer (Devarriya et al., 2020). In these diverse applications, the GP approach was used for its encoding openness in creating computer programs, and effectiveness in evolving efficient ones. It extends the interest of GAs in evolving not only individuals or organisms, but more complex objects such as computer programs. An important feature in this problem solving ability is the ability to generate novelty, and explore new regions in the search space. This has fueled various recent developments, from novelty search to open-endedness and the study of evolutionary systems such as ecosystems, or economic environments.

3.3 Emergence, novelty and open-endedness

3.3.1 GAs are more than optimizers: they create novelty

The successes of GAs in solving such complex problems have often led researchers to measure their progress with learning curves, fitness curves, and to emphasize the optimizing qualities of GAs. It would be restrictive and misleading to only see them as function optimizers (K. A. De Jong, 1993). While their problem solving methods have been improved by elitism (C. Ahn and Ramakrishna, 2003), i.e. keeping the best chromosome alive from one generation to the next, the canonical GAs remain deprived of a "killer instinct". They identify quite rapidly promising regions of the search space, but do not locate the exact optimum with a similar speed. Unless the problem is very specialised, in such a way that it is possible to adjust a specialised GA to take profit from this information, the canonical GA is likely to outperform over-exploiting GA-optimizers. Altering drastically the way crossover, selection or mutation function in an attempt to turn the GA into a function optimizer may alter its behavior so that
the resulting sampling biases shrink their desirable robustness and exploration abilities that are the reasons for which we used the GA as a problem solver in the first place. A key reason is the generation of novelty, which marks a decisive advantage of GAs among other search methods.

One could interpret the schema theorem as a limitation: for successful evolution to happen, intermediary rewards should exist in the fitness landscape (K. A. De Jong, 2008). In other words, the genetic algorithm would need to be rewarded for standing up, if we desired it to learn how to walk and run. This has often led to seeing GAs as vulnerable to deception, by the capacity to create a landscape that deceives the GA by hiding a high-fitness region in a larger region of lower fitness (Forrest and Holland, 1991). A related issue has been identified in multi-modal environments (Forrest and Holland, 1991), or functions with several peaks, leading the GA to premature convergence to one of the optima due to generic drift induced by random fluctuations in the sampling process. The production of novel, diverse solutions is an essential element to tackle these issues. GA adaptations with sharing or niching rules to prevent crowding of solutions (Wiransky, 2020), emphasis placed on novelty through novelty search (Lehman and Stanley, 2011) or quality-diversity search (Cully and Demiris, 2017, Pugh et al., 2016), provide both workarounds for these issues, and draw new avenues for further research. They establish connections with the general problem of artificial intelligence (Clune, 2019, Stanley et al., 2019), open-endedness in evolution (Stanley et al., 2017, Stanley, 2019) and artificial life.

3.3.2 Artificial life and emergence

GAs have been originally seen as promising to model the natural systems that inspired their design. Theories on the behavior and evolution of these systems can be implemented and tested in a GA. They allow to explore variations in details of the theories, simulate phenomena that would be difficult or even impossible to capture and analyse in a set of equations (Mitchell, 1998). They also connect evolution with information theory through that computational process lens. They provide by their ability of generate novelty, and to explore large-open search spaces, a useful tool to model the evolution living systems, both natural and human. One of the first of these ecological model was Holland’s Echo (J. H. Holland, 1999), studying how simple interactions between simple agents could create emergence of sophisticated higher-level phenomena, such as cooperation, competition, communities. These first models of artificial life using GAs were reviewed and commented by Mitchell and Forrest, 1994. Models of artificial, ”digital” evolution, are not just simulations of evolution, but mere instances, that are complex, creative and surprising (Lehman et al., 2020).

The evolutionary approach carried by the GA has relevance in the modelling of evolutionary human systems. In economics and finance, the evolution paradigm captures well the inductive theory of learning described by Arthur, 1994: agents generalise previous patterns for their future behavior, and adapt through trial and error to a changing environment. Palmer et al., 1994 created...
Qualities, challenges and future of genetic algorithms  
A. Vié

The Santa Fe Institute artificial stock market as an emergent product of evolving agent behaviors. The resulting artificial economy exhibited generated trends, speculative bubbles and crashes. The ecology these evolving agents created exhibited symbiosis, parasitism, arms races, mimicry, niche formation, speciation, all similar to natural ecosystems. Agents are co-evolving, and adapt their internal models of behavior rules to grow more sophisticated ones, by observing success and failure. The GA was used to govern the evolution of those rules, and model the creation of new ones by exploration, or innovation, providing a plausible mechanism to match empirical patterns (LeBaron et al., 1995). Since then, most of the evolutionary approaches in economics and finance have been carried by agent-based models. Most of the uses of GAs in economics have been applied works: analysis of insolvency risk (Varetto, 1998), supply chain design (Altiparmak et al., 2009), design of automated trading systems (Dempster and Jones, 2001) (Dempster and Leemans, 2006), Martinez-Jaramillo and Tsang, 2009), prediction of daily foreign exchange rates (Waheeb and Ghazali, 2019) or daily bitcoin prices (Han et al., 2019) and bank lending decisions (Metawa et al., 2017). There has been very limited theoretical follow up to the pioneering use of GAs by Palmer et al., 1994, as most of the evolutionary economics community shifted largely towards emergent macroeconomics and financial dynamics in multi-agent models. Most of these models including behavior rules encompass a limited form of evolution of these rules, most often restricted to a small number, and with little variation. We argue that in this context, GAs have a role to play. Their ability to generate novelty, explore new strategies, and account for economic learning and innovation, is relevant to domains such as market ecology, that study these economic systems from the participants types or behaviors. Allowing these behaviors to evolve in a larger space, and observing the resulting economic and financial patterns, is a promising and realistic direction for the application of GAs as modelling tools.

A plausible future of GAs lies in the modelling of continuously-adaptive systems. In the same way GAs have been used by Juzonis et al., 2012 to simulate malware evolution to forecast malware epidemic outbreaks, and contribute to the improvement of responses and threat evaluations, having models in economics, finance, but also other fields, that simulate the quasi open-ended evolution of human behavior, can shed new light on the behavior of these systems. Understanding the behavior effects of sudden changes in stock markets, improving our stress testing models to include the deep transitions in agents’ behaviors, are means to achieve more robust and more efficient economies. While horizontal (co-evolutive GAs) and vertical (meta-GAs) relations between GAs have emerged in the literature, analysing the behavior of systems of interacting “multi-GA models” may provide a new level of modelling in which agents are endowed with an adaptive set of strategies and representations of the world.

The innate connection of evolutionary algorithms with open-ended evolution, and the novelty-generating feature of GAs, can transform the modelling of evolutionary human systems and the approach to artificial intelligence. This emphasis on evolution, depth of reasoning, novelty creation and generation of complex ecological behaviors, are pathways for open-endedness and artificial
intelligence. Very early on, founding research in evolutionary algorithms saw the open-ended simulation of evolution as a path towards artificial intelligence (Fogel et al., 1966). Recently, AI-generative algorithms inspired from evolution are seen as a credible pathway towards achieving general artificial intelligence (Stanley et al., 2019, Clune, 2019). In contexts of problem solving and creative design, it can become a source of increasingly complex solutions. The novelty permitted and amplified in GAs is an essential component of such open evolution (Stanley et al., 2017). Co-evolution and novelty search have already attained significant achievements with generative adversarial networks, self-play and robot reinforcement-learning. The future of genetic algorithms, and of evolutionary algorithms in general, tends to further inspiration from natural evolution as an open-ended, divergent process. This comes however at the condition to solve some essential challenges, and realise fundamental innovations to escape the current limitations of genetic algorithms.

4 Limitations, challenges and perspectives

4.1 Computational efficiency and cost

Computational efficiency rules the GA convergence speed, and its performance. While they have often been criticized for their computational speed and complexity, and for the constraints efficiency sets on their design, GAs greatly scale with the use of parallelism and adjusted selection methods.

4.1.1 Selection methods for computational efficiency

Evolutionary algorithms have been criticized for the computational complexity, often identified to be of order $O(MN^3)$ where $M$ is the number of objectives and $N$ the population size. Indeed, comparing the respective fitness of $N$ individuals with all others, and selecting up to $N$ individuals for the next generation, contributes to a great computational cost that ought to be reduced as much as possible. In an influential article, Deb et al., 2002 introduced a faster elitist, multi-objective GA: "NSGA-II". Elitism, i.e. keeping in the next generation the current best chromosome in the population, speeds up the performance of the GA and allows to prevent the "catastrophic forgetting" of good solutions (Rudolph, 2001, Zitzler et al., 2000). NSGA-II limited computational complexity to $O(MN^2)$ using domination count, and a sharing rule for draws that gives preference to diversity. This gives us a fast non-dominated sorting procedure and a fast crowded distance operator, while computation is further improved by elitism. This specification is an example of various attempts to tweak evolutionary operators to generate more efficient procedures.

Innovations in computation efficiency have also impacted selection methods. In particular, the tournament selection method which operates a rank selection, but over a subset of the population, is computationally more efficient, and more amenable to parallel implementation (Mitchell, 1998). The size of the subset required to correctly evaluate dominance of solutions remains however
to be determined, and enters in the larger challenge of parameter configuration in GAs, that we analyse in more detail below. Recently, some metrics have been introduced to quantify the complexity and cost of given tasks for a GA: partial evaluation (Rodriguez and Ortiz, 2020) and fitness landscape analysis (Merz and Freisleben, 2000, Pitzer and Affenzeller, 2012, Wang et al., 2017). While the early applications of GAs looked for improvements in computation efficiency form the side of the operators, much larger progress has been achieved by parallelisation.

4.1.2 Implicit and explicit parallelism

Beyond the mostly static setting experimental GAs have focused on, high computational efficiency is necessary to consider more dynamic problems. By the time the problem is solved, it may have changed (Reeves, 1995). Adaptation in a sufficiently random, or unstable environment, is almost impossible (Mitchell, 1998). Natural evolution occurs in natural environments that are also changing over time. However, species evolve relatively much faster than climate, geology or fundamental environment features. If the environment changes faster than the GA populations can adapt, the effectiveness of the search becomes null. GAs have quite rarely been applied in dynamic or unstable environments for this reason. A great challenge for the development and improvement of new GAs resides in the ability to simulate an evolution that is fast enough to cope with the changes in the environment, but diverse and open enough not to overfit one particular environment instance, and stay adaptive.

As for any optimization-based procedure, the evaluation function must be fast to compute. However, as the evaluation is repeated a large number of times depending on population size, the number of constraints, and of iterations, it is a concerning issue for GAs (Whitley, 1994). Much theoretical effort has been devoted to improve the information acquired from a finite set of evaluation. Alternative paradigms for the determination of the fitness evaluation function have been proposed. In particular, J. Huang and Xie, 1998 have shown that using a fuzzy fitness evaluation function converged to results that were identical to the ones obtained by a standard GA, while considerably reducing the computation time (Laribi et al., 2004).

We emphasized the great scaling of GAs with parallelism in the context of their exploration capacities. The great synergy of GAs with parallelism, denoted "implicit parallelism" (J. Holland, 1992) mimicking the massive parallelism at play in natural systems composed of millions of individuals, provides substantial qualities in computation efficiency. GAs are able to test and exploit a large number of locations of the search space by manipulating only a few strings. The first explicit parallel implementations of GAs introduced multi-processor systems, each running a GA on own sub populations, with periodical migrations of the best solutions to other processors (Tanese, 1987). This distributed genetic algorithm with migrations was shown to perform better than the traditional one, even when each sub-population was running different parameter settings (Tanese, 1989).
Recently, approaches of parallelism related to data partitioning have been demonstrated as being more efficient in accuracy, efficiency and scalability (Alterkawi and Migliavacca, 2019). GAs enhanced with parallelism distributions have performed better than existing algorithms (Tang et al., 2017). With decomposition approaches, both implicit and explicit parallelism are applied. Different sub-populations are being evolved (L. Chen et al., 2018). Similarly to deep learning and machine learning, GAs have entered the world of high-performance computing, benefiting from the power of GPU architectures. Cheng and Gen, 2019 provide a comprehensive review of parallelism approaches, and their principal challenges. In the same way GPU computing has transformed deep learning and machine learning, it is yet to percolate in the GA community. When it will, it is likely that such highly parallelisable search technique will take immense benefits from this additional computing power. A final point of computational efficiency, linked with search speed, is co-evolution.

4.1.3 Co-evolution

Competition between candidates inspired by the Darwinian survival of the fittest is the drive of the evolution of genomes operated by the GA. Some authors have extended this framework to not only have competition within a GA, but between GAs. Inspired from co-evolution in biology, various GA applications have designed adversarial GAs to improve optimization in a related fashion to generative adversarial networks. A founding application was done by Hillis, 1990 who found that developing parasite-GAs against a GA trying to perform a classification task, significantly improved the optimization abilities of the latter. Garcia et al., 2017 used an adversarial GA network to develop network cyber-defence strategies against attacks. More recently, co-evolution has similarly contributed to applications of GAs in game theory. Vié, 2020b developed two GAs fighting each other populations to approach solutions to asymmetrical Blotto games. These are simultaneous resource allocation wargames in which two players allocate some resources over different battlefields, winning each battlefield if they have deployed more resources than their opponent. While solving versions of the game with asymmetric resource endowments was not analytically doable, co-evolving GAs allowed to approximate equilibrium strategies in this setting. They developed sophisticated and empirically consistent behaviors such as guerilla warfare, and concentration of competition. Notably, the GA with less resources learned to focus its resources on fewer battlefields. Co-evolving GAs achieved a significantly faster convergence than GAs on their own.

4.2 Parameter configuration

4.2.1 Parameter tuning is crucial to GA performance

The parameter calibration of GAs - population size, mutation rate, choice of operators... - is a critical determinant of its convergence behavior, and computational efficiency (Maulik and Bandyopadhyay, 2000, K. De Jong, 2020).
Parameters can be set at the start and fixed (parameter "tuning"), or changed during search (parameter "control"). A poorly configured GA can prematurely converge to a sub-optimal solution, or not converge at all, or converge so slowly that the entire process is essentially a waste of time. The performance of GAs is a nonlinear function of their parameters. Grefenstette, 1986 searched for optimal GA parameters using another GA, showing the efficiency of GA as meta-level optimisation techniques. The space of GAs was described as having six dimensions:

1. Population size: a large population favors diversity and mitigates premature convergence, but is detrimental to computation efficiency.

2. Crossover rate: the higher the frequency of crossover, the higher the frequency of introduction of new structures. A too high crossover rate can discard good solutions faster than selection can improve them, while a too low rate may create stagnation with a resulting lower exploration rate.

3. Mutation rate: a too large mutation rate creates an inefficient random search, while a too low mutation rate fails to prevent a given bit to remain forever in the population or can fail to mitigate premature convergence.

4. Generation gap: the percentage of the population to be replaced during each generation is optimised with respect to same above the trade-offs.

5. Scaling window: the reference to which solutions are compared may change their relative fitness, and alter the resulting fitness-proportional selection.

6. Selection strategy: pure selection, elitist selection, but also other mechanisms, are possible, and induce particular balance between selection, diversity, efficiency and convergence rate.

As it is simple to understand the impact of one parameter, the others being constant (it has been suggested that most parameters exhibited uni-modal, convex responses (Pushak and Hoos, 2018)), but hard to understand interactions of parameters, Grefenstette, 1986 developed a meta-level GA to identify high-performance GAs on some numerical test functions. The resulting GAs received a significant boost in performance. Some regularities were identified: mutation rates above 0.05 (or at 0) were usually harmful, and the optimal crossover rate and optimal mutation rates appear negatively correlated. Crossover rate appears to decrease as the population size increases. These insights were crucial in the early applications of GAs and proposed some much used baseline parameter values. Grefenstette, 1986 and Caruana and Schaffer, 1988 identified general ideal settings for elitist (non-elitist) selection strategies: a population size of 30 (80), a crossover probability of 0.95 (0.45), a mutation probability of 0.01 (0.01).

In the spirit of the no free lunch theorem of optimization for which no algorithm works universally for all optimization problems, one needs to tailor the GA to tackle different problems. The practical value of these theoretical results...
Qualities, challenges and future of genetic algorithms

A. Vié

on parameter settings remains unclear: they can be expected to vary depending on problems and the variant operators used by the GA. These early attempts to design optimal GAs remained limited by the set limitations on the space of possible GAs, and the computational cost of the meta-simulations.

Recent progress has been made in automatic parameter tuning (automatic algorithm configuration) to eliminate the limitations and drawbacks of manual parameter setting. This process can be described as a meta-optimization process that identifies the set of parameters in a configuration space for a given parametrised algorithm that maximizes a performance metric over a set of problem instances. C. Huang et al., 2019 surveys different state of the art techniques used in automatic parameter tuning, the classification of parameters into numerical (rates) and categorical (operators) types. Different tuning methods do exist: the Simple Generate-Evaluate (GEM) methods generate a set of candidate configurations, evaluate their performance and select the maximizing one. Brute-force and F-race (close to dominance selection) approaches are quite popular in SGEMs. Iterative GEMs create new configurations throughout search, and include notably heuristic search-based methods. High-level GEMs use existing tuners and search methods to generate high-quality candidates and compare them. Among these categories of tuning algorithms, Iterative GEMs stand out by their efficiency in exploring the configuration space, and their computation efficiency in doing so, as they use information from previous evaluations to generate new candidates. In particular, heuristic search-based iterative GEMs include iterative F-races (that eliminate candidates as statistic evidence grows against them, adapted to cases of many candidates: see Balaprakash et al., 2007), meta-evolutionary algorithms (meta-EAs) and ParamILS (a versatile local search approach with adaptive capping of runs to avoid unnecessary runs, introduced by Hutter et al., 2009). Two techniques are considered as the current state of the art within meta-EAs.

The Covariance Matrix Adaptation (CMA)-Evolution Strategies (CMA-ES, Hansen, 2006, Hansen, 2016) is a numerical optimization technique that samples candidates according to a multivariate normal distribution. Recombination amounts to changing the distribution mean, and mutations consist in a zero-mean perturbation. The covariance matrix of the variables is updated to maximise the likelihood of occurrence of previously successful solutions, in a related fashion to gradient descent. This method is very fast with small populations, essentially parameter free, and does makes very few assumptions on the underlying objective function. Evolution paths information is taken into account in step-size control to prevent premature convergence and from overshooting the optimal region.

The Gender-based GA (GGA, Ansótegui et al., 2009) takes advantage of the parallelism and handling of rugged landscapes features of the GA. As the way parameters interact with each others is a priori unknown, the GA becomes a great candidate. They distinguish populations in two categories, and apply different selection pressures. Only the competitive sub-population receives intra-specific competition, and struggle to mate with the noncompetitive sub-
Qualities, challenges and future of genetic algorithms

A. Vié

4.2.2 Parameter control

A single run of the GA is a stochastic path, intrinsically adaptive. Sticking to rigid parameters is in contrast with its spirit (Grefenstette, 1986). It is intuitive that different parameters settings may be optimal at different periods in time (Back, 1992; Forrest, 1993). Large mutations could be explored in the early steps to maximize exploration of promising regions of the search space, increasing selection pressure in later stages to narrow down the search to the optimal chromosomes, as does Boltzman selection (M.-R. Chen et al., 2019). Allowing a GA to dynamically modify its own parameters during a run was first suggested by Grefenstette, 1986, noting however that the evaluations done in some time interval would probably not be sufficient to conduct a realistic assessment of the search traits performance. Benefits and early approaches were surveyed by Eiben et al., 1999.

In the deterministic parameter control paradigm, parameters are changed by a fixed, predetermined rule, which can improve GA performance, but what the optimal rule depends again on the context. Adaptive parameter control uses some feedback from the search to determine the changes to implement in the parameters. However, the updating mechanism used to control parameters is externally set. Both these methods suffer from the difficulty of identifying such optimal rules of change (Eiben et al., 1999), an issue addressed by the use of
self-adaptive parameter control.

Natural evolution is itself a powerful meta-learning algorithm (Stanley et al., 2019), and was seen quite early as a natural development for GAs, with first positive theoretical results appearing with Greenwood and Zhu, 2001. Grefenstette, 1986 presented self adaptation as two dual searches taking place, one search for a solution to a given problem in a search space of possible candidates, and a second for an optimal algorithm setting in the space of possible algorithms. Adapting only the mutation rate for example, would restrict this second search to the space of algorithms with fixed mutation rate. Population size itself in natural systems is controlled by complex ecological interactions (Mitchell, 1998).

A line of research has explored self adaptation of operators rates, to let the GA choose the probability of distributions of operators. Davis, 1989 identified a key challenge for self adaptation of GAs to work: having the GA adaptation rate -and find ways to measure it- match the population adaptation rate. That "synchronization" problem appears to be the obstacle towards efficient self-adapting GAs. Hassanat and Alkafaween, 2017 enhanced GA performance with multiple crossover operators, and a meta-selection of the best to use. Meta-GAs assembled in a population with migrations were found appropriate to find good generalist parameter configurations, and to achieve performance close to specialist configurations for given problems (Clune et al., 2005). Their results also outlined the existence of shifts on the optimal parameters or operators through time, notably from crossover types, which was beneficial in short term performance, but detrimental in the longer run. Clune et al., 2005 also show how the ability to adapt exhibited by meta-GAs could lead to premature convergence to local optima. It is possible that those results apply in a specific context with a relatively low population (36 different GAs), or some restrictions of the search space, but they outline that while the meta-GA approach has some potential, it still faces important issues that need to be addressed to achieve it. In a related fashion, natural evolution failed to optimise mutation rates on rugged fitness landscapes, and selects sub-optimal mutation rates for their short term advantages (Clune et al., 2008). It appears that the dual search of a GA in the space of GAs, and the solution to a problem in a solution search space, is more complex, and amplifies the challenges they face. Parameter control is undoubtedly non-universal, and specific to each GA run (C. Huang et al., 2019). We highlight in the next section some promising directions for fundamental breakthroughs that may help us come back with new tools to these topics of parameter control by self-adaptive algorithms. Very recently, Case and Lehre, 2020 and Dang and Lehre, 2016 demonstrated theoretically and experimentally that self adaptation of GAs where parameters such a the mutation rate, are encoded with the individual chromosomes, leads to significant speedups, achieve optimal parameters as if they were known in advance. Evolutionary algorithms enhanced with self adaptation were found to have an asymptotic speed improvement over the state of the art solution for their considered problem (Case and Lehre, 2020).
4.3 Realism, robustness and endogeneity

It is not enough for GAs to be computationally efficient, and well parametrised. We also desire them to be robust in their trajectories, open in their exploration capacities, and realistic in their design. Choices of representation of individuals, and design of the fitness landscape, are essential choices to ensure realism and robustness of the approach. We open towards a greater embrace of evolution, to not only evolve solutions in complex environments, but also the GA itself, and address these limitations all together.

4.3.1 Initial population sampling

This concern for diversity starts from the very initialisation of the GA. Bias in the first generation is likely to induce further bias in search, leading potentially to premature convergence and local optima. Sampling error was early identified as one of the main difficulties faced by GAs (Forrest and Holland, 1991), and can induce premature convergence (Forrest and Mitchell, 1993). In matter of instance of evolution, studying whether and how initial conditions impact the GA outcomes provides as well valuable information on the evolution process. However, for more practical purposes, some more rigor on the initial sampling can greatly benefit GA applications. In high dimensional spaces, new methods to generate initial populations allow to maintain diversity with limited population sizes. While little attention has been paid on establishing formal desirable conditions for this initial diversity, significant results for searching in very large spaces have been achieved starting from "unbiased" initial populations (Deaven and Ho, 1995). This unbiasedness has often been achieved by simple random uniform sampling. As the object being considered by the GA grows in dimensions and complexity, random uniform sampling requires a substantially higher population size to satisfy this unbiasedness objective. To maintain the diversity of the GA initial population without sacrificing computation efficiency with excessively large population sizes, we may be interested in alternative initial seeding techniques. Recent advances have suggested advanced sampling methods as a promising means to achieve better robustness and diversity in high-dimensional simulation exploration. Latin Hypercube Sampling (Helton and Davis, 2003, see application and discussion by Jing et al., 2019) and Sobol Sequences (Sobol, 1967), that consist in generating low discrepancy sequences, i.e. quasi uniform sets of points in a high dimensional space, are now being used in the analysis of simulation models (Romain Reuillon, 2013), and can similarly improve the starting points of the GA in the search space. While the current practice of GAs is focused on random initial samples (Yakovlev et al., 2019), we maintain that the danger of sampling bias for GAs optimisation is significant, increasing in higher dimensions, and can be mitigated by the above sampling methods.

4.3.2 Representation

One of the most crucial limitation of GAs is representation, most often denoted encoding, that denotes how the action space is modelled. Various encoding
Qualities, challenges and future of genetic algorithms A. Vié

techniques exist in the literature. The most popular, the binary encoding, uses strings composed of 0s and 1s. It has the advantage of containing a higher degree of implicit parallelism, as a GA instance will contain more schemas, and most of the mutation, crossover operators have been built around the binary encoding. The encoding representation commonly used in GAs admits many variants, and exhibits a high versatility. However, binary encoding might be too narrow an encoding for some problems. Some may require integer values, some may contain strings or operators. In other cases, the number of bits necessary to encode a certain space of solutions in a binary way would simply be too large for this encoding to function. The encoding representation indirectly constrains the search space, with the issue of partial cover. Only a robust, careful representation design allows to cover its fully diversity (Juzonis et al., 2012). A founding analysis on encoding in Genetic algorithms has been done by Ronald, 1997. He notably defined the nine desirable properties for a robust encoding in the context of the schema approach:

1. Embodies the problem-relevant fundamental building blocks
2. Is amenable to a set of genetic operators to operate selection, crossover and mutation
3. Minimises epistasis (expression of one gene suppresses the action of others)
4. Allows a tractable mapping to the phenotype that allows fitness to be measured
5. Exploits an appropriate mapping from the genotype to the phenotype
6. Embodies feasible solutions and discards illegal candidates
7. Suppresses isomorphism and mitigates redundancy: many genotypes converging to a same solution point
8. Uses the smallest cardinally of an alphabet for the gene values, the binary being the best if relevant
9. Represents the problem at the correct level of abstraction

To avoid genetic hitchhiking, in which some low fitness sequences associated with highly successful alleles are maintained in spite of their low overall performance, and reduce the size of the representations GA have to search, Schraudolph and Belew, 1992 proposed a dynamic parameter encoding mechanism that adjusts in size as evolution occurs. Kumar, 2013 surveys the different encoding schemes: binary, octal, hexadecimal, permutation, value, tree encoding, and restricts their design to the principle of minimal alphabet, and principle of meaningful building blocks.

The use of GAs to evolve neural network architectures inspired a lot of research on encoding and representation. As the network size grew, so did the size of the required chromosome to encode the network structure, which led to significant issues in performance (how high a fitness can be obtained) and efficiency (how long does it take to obtain this high fitness result). Direct encoding methods represent each network connection, and struggled in encoding repeated structure, such as symmetry in a network. For complex networks, encoding the network adjacency matrix in the chromosomes may become huge and
Qualities, challenges and future of genetic algorithms

A. Vié

intractable for any search algorithm. A set of 100 nodes would require chromosomes of 5000 bits. A set of 1000 nodes would have to be encoded in 500,000-bit strings to represent each connection. One solution identified by Kitano, 1990, was grammatical encoding: encode network as grammars of structure operators, in a related fashion to genetic programming. This representation requires shorter chromosomes, as the GA evolves sets of building instructions, rather than the network structure itself. Grammatical encoding hence mitigates that issue of complexity, and can inspire innovations in encoding representations, in order to design more robust or efficient networks in neural nets, security, communications, production, supply-chain, or financial contexts.

Indirect encoding has enjoyed a recent popularity (Stanley et al., 2019): the genome is a formula, a set of instructions for generating the network, rather than a direct edge by edge encoding. Some approaches have been based on Cartesian genetic programming where the encoding being evolved is machine code with some operators (Khan et al., 2013, Turner and Miller, 2014). Inspired from genetic regulatory networks, Mattiussi and Floreano, 2007 have developed the Analog genetic encoding, that allows complexification and decomplexification of the network during the evolutionary process. How can 100 trillion connections and 100 billion neurons be encoded in a DNA encoding of 30,000 genes? (Stanley et al., 2019) Indirect encoding aims at using regularity such as symmetries or motifs, to improve compression. This has been connected with canalization: capacity of natural indirect encoding to yield robust, adaptable evolutionary paths of development (Le Rouzic et al., 2013).

As Grefenstette, 1986 point out, these representations, whether binary, finite alphabet in fixed-length chromosomes, is intrinsically inconsistent with the adaptive, evolution framework of the GA. These encoding methods are not adaptive, and as such, they restrict the search space, potentially limiting it to sub-optimal choices of lesser complexity. Any fixed length representation limits the complexity of the candidate solutions, hence the search space (Mitchell, 1998). It could even be argued that evolution not only makes the genomes bigger in size, more complex and more adaptive, but could change the genetic encoding and the mapping between genotypes and phenotypes themselves (Mitchell, 1998). As we consider the chromosomes as carrying evolving information, but also as evolving objects, we walk even further into an evolutionary paradigm. As we seek further inspiration from biology and genetics to understand how nature is encoded and mapped to phenotypes, we may move from manual design of representation to evolved representations, in which the design principles we enumerated are the product of evolution, rather than constraints on manual design. Embracing the evolution paradigm to a larger extent leads GAs to open-endedness, and vast uncertainty on their resulting behavior or properties, that is largely left to analyse, but that may contain essential innovations for the practice of GAs.
4.3.3 Fitness evaluation

While the evaluation function is often contained in the problem formulation, its design can be a difficult task (Whitley, 1994). Digital evolution is surprisingly creative in exploiting misspecified fitness functions, create unintended debugging of simulated environments (Lehman et al., 2020). The design of the fitness function may reflect some preconceptions of the experimenter on the form of the solution, include unintended loopholes that are easy for evolution programs to exploit, in the same ways well-intentioned metrics in human societies can have detrimental effects by the pressure to optimise them. Well-intentioned quantitative measures of fitness can be maximized in counter-intuitive, glitchy ways. In some other cases, digital evolution programs learned unintended regularities instead of learning the complex behaviors experimenters were aiming for (Ellefson et al., 2014), or took advantage of simulation bugs or design flaws that were previously unseen. Various stories of digital evolution algorithms "outsmarting" their creators are compiled by Lehman et al., 2020.

Evidently, fitness is not exogenous in nature, and there is no such physical thing as an objective fitness evaluation function. At best, this function would appear rather related to fuzzy logic (J. Huang and Xie, 1998). While an objective function is certainly relevant for optimization problems, subject to the design issues we highlighted above, modelling more complex, evolutionary systems could benefit from endogenous fitness determination (J. H. Holland, 1999). The fitness, or qualities of a given phenotype, may heavily depend not only on a fixed environment, but also on the other phenotypes present in the environment. When those instabilities in the environment are well understood, they can be included in the fitness evaluation function (Vié, 2020a). The GA evolving under this more uncertain environment was able to both identify the two Nash equilibria of the variants, and to develop optimal mixed responses in between.

Evolving intelligent agents requires them to explore different selves, alternative ways to represent them, and to modify them. One possible direction could be to develop a co-evolutionary paradigm in which the fitness landscape evolves with the agents. Evolving complex behaviors requires designing the fitness landscape in such a way that these intermediate rewards exist (K. A. De Jong, 2008). In fact, endogenous fitness opens to multiple phenomena observed in natural systems: predator-prey relationships, symbiosis, crowding effects. Interactions within the GA allow emergence of an evolving, autonomous ecology of phenotypes (Smith and Bedau, 2000), leading to both optimization, and understanding of the ecology behavior. It connects genetic algorithms with artificial life (Mitchell, 1998). Endogenous, not explicit, fitness, is at the root of open-ended, creative, surprising digital evolution (Lehman and Stanley, 2011, Lehman et al., 2020).
5 Conclusion

In this review, we have attempted to present in a condensed way the different properties of genetic algorithms, their merits as exploration and optimisation heuristics, and the challenges they face, notably in computation efficiency, parameter configuration, and robustness. We hope to have well covered the search space in this objective and to have identified its most promising points of its landscape.

Genetic algorithms are particularly suitable to solving sparse problems in large, rugged search spaces. They require very few assumptions on the fitness landscape properties, deal well with existence of local optima, or multiplicity of extrema. They are applicable to a wide range of problems, and are quite efficient in returning good solutions quickly. They have achieved particular performance in the evolution of neural networks, multi-objective optimisation and genetic programming. They are more than optimisers, as they generate novelty, are directly connected to artificial life as they foster emergence of complex digital ecologies, and a clear relation with open-ended evolution that can greatly improve the modelling of evolutionary, adaptive systems. Genetic algorithms have been facing significant issues in computation efficiency and cost, and in parameter configuration. Improving the efficiency of evolution as it is digitally encoded will undoubtedly involve parallelism, and the evolution of parameter configurations themselves through self-adaptation. Advances in initial sampling methods can overcome sampling bias issues.

Further inspiration from biology to incorporate in a computational form some ecological and genetic interactions, will undoubtedly allow practitioners to develop more sophisticated evolutionary algorithms, capable of evolving in changing, or more complex environments. Phenomena of gene duplication, translocation, dominance, sexual differentiation, regulatory networks, have just emerged in the field. Epistasis -interactions between mutations- and pleiotropy -mutations at one locus affect several traits- are crucial in complex and realistic fitness landscapes. Further connections with mathematical genetics and statistical mechanics are likely to provide new theoretical grounds to understand the behavior of genetic algorithms, and the properties of their operators. The technological progress in computing power and the GPU revolution (Cheng and Gen, 2019) has pushed deep learning to the front of machine learning research, and exciting times are ahead as GAs will benefit from it, with their natural scaling with parallelism (Stanley et al., 2019).

The presence of a tremendous diversity of organisms stemming from natural evolution, arguably involves the evolution of evolvability itself (Huizinga et al., 2018). Lehman et al., 2018 proposed a safe mutation approach that applies mutations through output gradients, to improve the benefits from mutations, but evolution of mutations towards beneficial ones may also be the product of the evolution of evolvability and pleiotropy, rather than a manual addition. The ability to select some dimensions of variation to be more or less likely to be explored through mutation, a phenomenon known as developmental canalization,
is barely emerging in computational simulations of evolution, but could allow the field of GA to tackle more important challenges. Open-ended, divergent evolutionary processes may be necessary for attaining this evolution of evolvability (Huizinga et al., 2018).

The latest developments of genetic algorithms towards meta-learning architectures, learning how to learn, and the endogenous generation of learning environments, have placed AI-generating algorithms as a credible means to produce general AI (Clune, 2019). Open-endedness in their evolution could lead them to produce interesting and increasingly complex discoveries, and so indefinitely (Stanley et al., 2019). The first creator of GAs, John Holland, noted that such computer programs that evolved in ways similar to natural selection, could solve complex problems even their creators do not fully understand. We add that they will do so and generate surprising novelty, provided they are endowed with a representation, a landscape that allows to go beyond manual design, and places all GA components, parameters to encoding, subject to open-ended evolution.

Acknowledgments

The author is grateful to a number of persons who contributed by their discussions, feedback, advice and recommendations to the writing of this review. I wish to particularly thank Jeff Clune, Rama Cont, Doyne Farmer, Alissa Klein-nijenhuis, Melanie Mitchell, Maarten Scholl.

Funding

This publication is based on work supported [or partially supported] by the EPSRC Centre for Doctoral Training in Mathematics of Random Systems: Analysis, Modelling and Simulation (EP/S023925/1)

References

Abd-Elazim, S., & Ali, E. (2018). Load frequency controller design of a two-area system composing of pv grid and thermal generator via firefly algorithm. Neural Computing and Applications, 30(2), 607–616.

Ahmad, I., Hussain, M., Alghamdi, A., & Alelaiwi, A. (2014). Enhancing svm performance in intrusion detection using optimal feature subset selection based on genetic principal components. Neural computing and applications, 24(7-8), 1671–1682.

Ahn, C., & Ramakrishna, R. S. (2003). Elitism-based compact genetic algorithms. IEEE Transactions on Evolutionary Computation, 7(4), 367–385.

Ahn, H., & Kim, K. (2009). Bankruptcy prediction modeling with hybrid case-based reasoning and genetic algorithms approach. Applied soft computing, 9(2), 599–607.
Qualities, challenges and future of genetic algorithms

A. Vié

Alterkawi, L., & Migliavacca, M. (2019). Parallelism and partitioning in large-scale gas using spark. *Proceedings of the Genetic and Evolutionary Computation Conference*, 736–744.

Altiparmak, F., Gen, M., Lin, L., & Karaoglan, I. (2009). A steady-state genetic algorithm for multi-product supply chain network design. *Computers & Industrial Engineering*, 56(2), 521–537.

Amin, A. (2013). A novel classification model for cotton yarn quality based on trained neural network using genetic algorithm. *Knowledge-Based Systems*, 39, 124–132.

Ansótegui, C., Sellmann, M., & Tierney, K. (2009). A gender-based genetic algorithm for the automatic configuration of algorithms. *International Conference on Principles and Practice of Constraint Programming*, 142–157.

Armaghani, D. J., Hasanipanah, M., Mahdiyar, A., Abd Majid, M. Z., Amniah, H. B., & Tahir, M. M. (2018). Airblast prediction through a hybrid genetic algorithm-ann model. *Neural Computing and Applications*, 29(9), 619–629.

Arthur, W. B. (1994). Inductive reasoning and bounded rationality. *The American economic review*, 84(2), 406–411.

Axelrod, R. et al. (1987). The evolution of strategies in the iterated prisoner’s dilemma. *The dynamics of norms*, 1–16.

Azadeh, A., Mianaei, H. S., Asadzadeh, S., Saberi, M., & Sheikhalishahi, M. (2015). A flexible ann-ga-multivariate algorithm for assessment and optimization of machinery productivity in complex production units. *Journal of Manufacturing Systems*, 35, 46–75.

Back, T. (1992). The interaction of mutation rate, selection, and self-adaptation within a genetic algorithm. *Proc. 2nd Conference of Parallel Problem Solving from Nature, 1992*.

Balaprakash, P., Birattari, M., & Stützle, T. (2007). Improvement strategies for the f-race algorithm: Sampling design and iterative refinement. *International workshop on hybrid metaheuristics*, 108–122.

Bhandari, D., Murthy, C., & Pal, S. K. (1996). Genetic algorithm with elitist model and its convergence. *International journal of pattern recognition and artificial intelligence*, 10(06), 731–747.

Bianco, S., Ciocca, G., & Schettini, R. (2017). Combination of video change detection algorithms by genetic programming. *IEEE Transactions on Evolutionary Computation*, 21(6), 914–928.

Braun, M. A., Seijo, S., Echanobe, J., Shukla, P. K., del Campo, I., Garcia-Sedano, J., & Schmeck, H. (2016). A neuro-genetic approach for modeling and optimizing a complex cogeneration process. *Applied Soft Computing*, 48, 347–358.

Caruana, R. A., & Schaffer, J. D. (1988). Representation and hidden bias: Gray vs. binary coding for genetic algorithms. *Machine learning proceedings 1988* (pp. 153–161). Elsevier.
Qualities, challenges and future of genetic algorithms

A. Vié

Case, B., & Lehre, P. K. (2020). Self-adaptation in non-elitist evolutionary algorithms on discrete problems with unknown structure. *IEEE Transactions on Evolutionary Computation*.

Chatterjee, S., Nag, R., Dey, N., & Ashour, A. S. (2018). Efficient economic profit maximization: Genetic algorithm based approach. *Smart trends in systems, security and sustainability* (pp. 307–318). Springer.

Chen, L., Deb, K., & Liu, H.-L. (2018). Investigating the effect of parallelism in decomposition based evolutionary many-objective optimization algorithms. *Michigan State University, East Lansing, USA, Tech. Rep. COIN Report*, (2018004).

Chen, M.-R., Chen, J.-H., Zeng, G.-Q., Lu, K.-D., & Jiang, X.-F. (2019). An improved artificial bee colony algorithm combined with extremal optimization and boltzmann selection probability. *Swarm and Evolutionary Computation*, 49, 158–177.

Cheng, J. R., & Gen, M. (2019). Accelerating genetic algorithms with gpu computing: A selective overview. *Computers & Industrial Engineering*, 128, 514–525.

Chung, H., & Shin, K.-s. (2020). Genetic algorithm-optimized multi-channel convolutional neural network for stock market prediction. *Neural Computing and Applications*, 32(12), 7897–7914.

Clune, J. (2019). Ai-gas: Ai-generating algorithms, an alternate paradigm for producing general artificial intelligence. *arXiv preprint arXiv:1905.10985*.

Clune, J., Goings, S., Punch, B., & Goodman, E. (2005). Investigations in meta-gas: Panaceas or pipe dreams? *Proceedings of the 7th annual workshop on Genetic and evolutionary computation*, 235–241.

Clune, J., Misevic, D., Ofria, C., Lenski, R. E., Elena, S. F., & Sanjuán, R. (2008). Natural selection fails to optimize mutation rates for long-term adaptation on rugged fitness landscapes. *PLoS Comput Biol*, 4(9), e1000187.

Cully, A., Clune, J., Tarapore, D., & Mouret, J.-B. (2015). Robots that can adapt like animals. *Nature*, 521(7553), 503–507.

Cully, A., & Demiris, Y. (2017). Quality and diversity optimization: A unifying modular framework. *IEEE Transactions on Evolutionary Computation*, 22(2), 245–259.

Dang, D.-C., & Lehre, P. K. (2016). Self-adaptation of mutation rates in non-elitist populations. *International Conference on Parallel Problem Solving from Nature*, 803–813.

Das, R., Mitchell, M., & Crutchfield, J. P. (1994). A genetic algorithm discovers particle-based computation in cellular automata. *International Conference on Parallel Problem Solving from Nature*, 344–353.

Davis, L. (1989). Adapting operator probabilities in genetic algorithms. *Proceedings of the third international conference on Genetic algorithms*, 61–69.

De Jong, K. (2020). Evolutionary computation: A unified approach. *Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion*, 327–342.
De Jong, K. A. (1993). Genetic algorithms are not function optimizers. *Foundations of genetic algorithms* (pp. 5–17). Elsevier.

De Jong, K. A. (2008). Evolving intelligent agents: A 50 year quest. *IEEE Computational Intelligence Magazine, 3*(1), 12–17.

Deaven, D. M., & Ho, K.-M. (1995). Molecular geometry optimization with a genetic algorithm. *Physical review letters, 75*(2), 288.

Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation, 6*(2), 182–197.

de Lacerda, M. G. P., de Araujo Pessoa, L. F., de Lima Neto, F. B., Ludermir, T. B., & Kuchen, H. (2020). A systematic literature review on general parameter control for evolutionary and swarm-based algorithms. *Swarm and Evolutionary Computation, 100777.*

Dempster, M. A., & Jones, C. M. (2001). A real-time adaptive trading system using genetic programming. *Quantitative Finance, 1*(4), 397–413.

Dempster, M. A., & Leemans, V. (2006). An automated fx trading system using adaptive reinforcement learning. *Expert Systems with Applications, 30*(3), 543–552.

Devarriya, D., Gulati, C., Mansharamani, V., Sakalle, A., & Bhardwaj, A. (2020). Unbalanced breast cancer data classification using novel fitness functions in genetic programming. *Expert Systems with Applications, 140*, 112866.

Dhunny, A., Timmons, D., Allam, Z., Lollechund, M., & Cunden, T. (2020). An economic assessment of near-shore wind farm development using a weather research forecast-based genetic algorithm model. *Energy, 117541.*

Donate, J. P., Li, X., Sánchez, G. G., & de Miguel, A. S. (2013). Time series forecasting by evolving artificial neural networks with genetic algorithms, differential evolution and estimation of distribution algorithm. *Neural Computing and Applications, 22*(1), 11–20.

Drugan, M. M. (2019). Reinforcement learning versus evolutionary computation: A survey on hybrid algorithms. *Swarm and evolutionary computation, 44*, 228–246.

Eiben, Á. E., Hinterding, R., & Michalewicz, Z. (1999). Parameter control in evolutionary algorithms. *IEEE Transactions on evolutionary computation, 3*(2), 124–141.

Ellefsen, K. O., Mouret, J.-B., & Chme, J. (2014). Neural modularity reduces catastrophic forgetting. *The Evolution of Learning: Balancing Adaptivity and Stability in Artificial Agents, 111.*

Fogel, L. J., Owens, A. J., & Walsh, M. J. (1966). Artificial intelligence through simulated evolution.

Forrest, S. (1993). Optimal mutation rates in genetic search. *Proc. 5th Int. Conf. Genetic Algorithms, 2–8.*

Forrest, S., & Holland, J. (1991). The royal road for genetic algorithms: Fitness landscapes and ga performance. *Toward a Practice of Autonomous Sys-
Qualities, challenges and future of genetic algorithms

A. Vié

*items: Proceedings of the First European Conference on Artificial Life, 245–254.

Forrest, S., & Mitchell, M. (1993). What makes a problem hard for a genetic algorithm? some anomalous results and their explanation. *Machine Learning, 13*(2-3), 285–319.

Garcia, D., Lugo, A. E., Hemberg, E., & O’Reilly, U.-M. (2017). Investigating coevolutionary archive based genetic algorithms on cyber defense networks. *Proceedings of the Genetic and Evolutionary Computation Conference Companion, 1455–1462.*

Ghofrani, M., Arabali, A., Etezadi-Amoli, M., & Fadali, M. S. (2013). A framework for optimal placement of energy storage units within a power system with high wind penetration. *IEEE Transactions on Sustainable Energy, 4*(2), 434–442.

Goldberg, D. E., Richardson, J. et al. (1987). Genetic algorithms with sharing for multimodal function optimization. *Genetic algorithms and their applications: Proceedings of the Second International Conference on Genetic Algorithms, 41–49.*

Greenwood, G. W., & Zhu, Q. J. (2001). Convergence in evolutionary programs with self-adaptation. *Evolutionary Computation, 9*(2), 147–157.

Grefenstette, J. J. (1986). Optimization of control parameters for genetic algorithms. *IEEE Transactions on systems, man, and cybernetics, 16*(1), 122–128.

Grefenstette, J. J. et al. (1992). Genetic algorithms for changing environments. *Ppsn, 2*, 137–144.

Han, J.-B., Kim, S.-H., Jang, M.-H., & Ri, K.-S. (2019). Using genetic algorithm and narx neural network to forecast daily bitcoin price. *Computational Economics, 1*–17.

Hansen, N. (2006). The cma evolution strategy: A comparing review. *Towards a new evolutionary computation* (pp. 75–102). Springer.

Hansen, N. (2016). The cma evolution strategy: A tutorial. *arXiv preprint arXiv:1604.00772.*

Hassanat, A. B., & Alkafaween, E. (2017). On enhancing genetic algorithms using new crossovers. *International Journal of Computer Applications in Technology, 55*(3), 202–212.

Helton, J. C., & Davis, F. J. (2003). Latin hypercube sampling and the propagation of uncertainty in analyses of complex systems. *Reliability Engineering & System Safety, 81*(1), 23–69.

Hillis, W. D. (1990). Co-evolving parasites improve simulated evolution as an optimization procedure. *Physica D: Nonlinear Phenomena, 42*(1-3), 228–234.

Holland, J. (1975). *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence.* MIT press.

Holland, J. (1992). Genetic algorithms. *Scientific American, 267*(1), 66–73.
Holland, J. H. (1999). Echoing emergence: Objectives, rough definitions, and speculations for echo-class models. *Complexity: Metaphors, models, and reality* (pp. 309–342).

Homaifar, A., Qi, C. X., & Lai, S. H. (1994). Constrained optimization via genetic algorithms. *Simulation, 62*(4), 242–253.

Horn, J., Nafpliotis, N., & Goldberg, D. E. (1994). A niched pareto genetic algorithm for multiobjective optimization. *Proceedings of the first IEEE conference on evolutionary computation. IEEE world congress on computational intelligence*, 82–87.

Huang, C., Li, Y., & Yao, X. (2019). A survey of automatic parameter tuning methods for metaheuristics. *IEEE Transactions on Evolutionary Computation, 24*(2), 201–216.

Huang, J., & Xie, W. (1998). Genetic algorithm with fuzzy fitness evaluation. *Journal of Electronics (China), 15*(3), 254–258.

Huizinga, J., Stanley, K. O., & Clune, J. (2018). The emergence of canalization and evolvability in an open-ended, interactive evolutionary system. *Artificial life, 24*(3), 157–181.

Hutter, F., Hoos, H. H., Leyton-Brown, K., & Stützle, T. (2009). Paramils: An automatic algorithm configuration framework. *Journal of Artificial Intelligence Research, 36*, 267–306.

Jin, C., Mao, H., Chen, Y., Shi, Q., Wang, Q., Ma, G., & Liu, Y. (2020). Engineering-oriented dynamic optimal control of a greenhouse environment using an improved genetic algorithm with engineering constraint rules. *Computers and Electronics in Agriculture, 177*, 105698.

Jing, Z., Chen, J., & Li, X. (2019). Rbf-ga: An adaptive radial basis function metamodeling with genetic algorithm for structural reliability analysis. *Reliability Engineering & System Safety, 189*, 42–57.

Juzonis, V., Goranin, N., Cenys, A., & Olifer, D. (2012). Specialized genetic algorithm based simulation tool designed for malware evolution forecasting. *Annales Universitatis Mariae Curie-Skłodowska, sectio AI–Informatica, 12*(4).

Khan, M. M., Ahmad, A. M., Khan, G. M., & Miller, J. F. (2013). Fast learning neural networks using cartesian genetic programming. *Neurocomputing, 121*, 274–289.

Kitano, H. (1990). Designing neural networks using genetic algorithms with graph generation system. *Complex systems, 4*, 461–476.

Kitano, H. (1994). Neurogenetic learning: An integrated method of designing and training neural networks using genetic algorithms. *Physica D: Nonlinear Phenomena, 75*(1-3), 225–238.

Koza, J. R. (1995). Survey of genetic algorithms and genetic programming. *Wescon conference record*, 589–594.

Koza, J. R., & Koza, J. R. (1992). *Genetic programming: On the programming of computers by means of natural selection* (Vol. 1). MIT press.

Kumar, A. (2013). Encoding schemes in genetic algorithm. *International Journal of Advanced Research in IT and Engineering, 2*(3), 1–7.
Langdon, W. B. (2020). Genetic programming and evolvable machines at 20. *Genetic Programming and Evolvable Machines, 21*(1), 205–217.

Laribi, M., Mlika, A., Romdhane, L., & Zeghloul, S. (2004). A combined genetic algorithm–fuzzy logic method (ga–fl) in mechanisms synthesis. *Mechanism and machine theory, 39*(7), 717–735.

Le Rouzic, A., Álvarez-Castro, J. M., & Hansen, T. F. (2013). The evolution of canalization and evolvability in stable and fluctuating environments. *Evolutionary Biology, 40*(3), 317–340.

LeBaron, B. et al. (1995). *Experiments in evolutionary finance* (tech. rep.).

Lehman, J., Chen, J., Clune, J., & Stanley, K. O. (2018). Safe mutations for deep and recurrent neural networks through output gradients. *Proceedings of the Genetic and Evolutionary Computation Conference, 117–124.*

Lehman, J., Clune, J., Misevic, D., Adami, C., Altenberg, L., Beaulieu, J., Bentley, P. J., Bernard, S., Beslon, G., Bryson, D. M., et al. (2020). The surprising creativity of digital evolution: A collection of anecdotes from the evolutionary computation and artificial life research communities. *Artificial Life, 26*(2), 274–306.

Lehman, J., & Stanley, K. O. (2011). Abandoning objectives: Evolution through the search for novelty alone. *Evolutionary computation, 19*(2), 189–223.

Lwin, K., Qu, R., & Kendall, G. (2014). A learning-guided multi-objective evolutionary algorithm for constrained portfolio optimization. *Applied Soft Computing, 24*, 757–772.

Martinez-Jaramillo, S., & Tsang, E. P. (2009). An heterogeneous, endogenous and coevolutionary gp-based financial market. *IEEE Transactions on Evolutionary Computation, 13*(1), 33–55.

Mattussi, C., & Floreano, D. (2007). Analog genetic encoding for the evolution of circuits and networks. *IEEE Transactions on evolutionary computation, 11*(5), 596–607.

Maulik, U., & Bandyopadhyay, S. (2000). Genetic algorithm-based clustering technique. *Pattern recognition, 33*(9), 1455–1465.

Merz, P., & Freisleben, B. (2000). Fitness landscape analysis and memetic algorithms for the quadratic assignment problem. *IEEE transactions on evolutionary computation, 4*(4), 337–352.

Metawa, N., Hassan, M. K., & Elhoseny, M. (2017). Genetic algorithm based model for optimizing bank lending decisions. *Expert Systems with Applications, 80*, 75–82.

Miller, B. L., Goldberg, D. E. et al. (1995). Genetic algorithms, tournament selection, and the effects of noise. *Complex systems, 9*(3), 193–212.

Mirjalili, S. (2019). Genetic algorithm. *Evolutionary algorithms and neural networks* (pp. 43–55). Springer.

Mirjalili, S., Dong, J. S., Sadiq, A. S., & Faris, H. (2020). Genetic algorithm: Theory, literature review, and application in image reconstruction. *Nature-inspired optimizers* (pp. 69–85). Springer.

Mitchell, M. (1998). *An introduction to genetic algorithms.* MIT press.

Mitchell, M., & Forrest, S. (1994). Genetic algorithms and artificial life. *Artificial life, 1*(3), 267–289.
Nguyen, S., Mei, Y., & Zhang, M. (2017). Genetic programming for production scheduling: A survey with a unified framework. *Complex & Intelligent Systems, 3*(1), 41–66.

Packard, N. H. (1988). Adaptation toward the edge of chaos. *Dynamic patterns in complex systems, 212*, 293.

Packard, N. H. (1990). A genetic learning algorithm for the analysis of complex data. *Complex Systems, 4*(5), 543–572.

Palmer, R. G., Arthur, W. B., Holland, J. H., LeBaron, B., & Tayler, P. (1994). Artificial economic life: A simple model of a stockmarket. *Physica D: Nonlinear Phenomena, 75*(1-3), 264–274.

Pitzer, E., & Affenzeller, M. (2012). A comprehensive survey on fitness landscape analysis. *Recent advances in intelligent engineering systems* (pp. 161–191). Springer.

Pugh, J. K., Soros, L. B., & Stanley, K. O. (2016). Quality diversity: A new frontier for evolutionary computation. *Frontiers in Robotics and AI, 3*, 40.

Pushak, Y., & Hoos, H. (2018). Algorithm configuration landscapes. *International Conference on Parallel Problem Solving from Nature, 271–283.*

Raman, M. G., Somu, N., Kirthivasan, K., Liscano, R., & Sriram, V. S. (2017). An efficient intrusion detection system based on hypergraph-genetic algorithm for parameter optimization and feature selection in support vector machine. *Knowledge-Based Systems, 134*, 1–12.

Ravi, V., Kurniawan, H., Thai, P. N. K., & Kumar, P. R. (2008). Soft computing system for bank performance prediction. *Applied soft computing, 8*(1), 305–315.

Raymer, M. L., Punch, W. F., Goodman, E. D., Kuhn, L. A., & Jain, A. K. (2000). Dimensionality reduction using genetic algorithms. *IEEE transactions on evolutionary computation, 4*(2), 164–171.

Rechenberg, I. (1978). Evolutionsstrategien. *Simulationsmethoden in der medizin und biologie* (pp. 83–114). Springer.

Reddy, G. T., Reddy, M. P. K., Lakshmann, K., Rajput, D. S., Kaluri, R., & Srivastava, G. (2020). Hybrid genetic algorithm and a fuzzy logic classifier for heart disease diagnosis. *Evolutionary Intelligence, 13*(2), 185–196.

Reeves, C. R. (1995). A genetic algorithm for flowshop sequencing. *Computers & operations research, 22*(1), 5–13.

Rejeb, J., & Abu Elhaij, M. (2000). New gender genetic algorithm for solving graph partitioning problems. *Proceedings of the 43rd IEEE Midwest Symposium on Circuits and Systems (Cat. No. CH37144), 1*, 444–446.

Rodriguez, A. A. O., & Ortiz, M. R. S. (2020). Partial evaluation of genetic algorithms. *Industrial and Engineering Applications of Artificial Intelligence and Expert Systems: Proceedings of the Tenth International Conference, 217.*

Romain Reuillon, S. R.-C., Mathieu Le克莱ire. (2013). Openmole, a workflow engine specifically tailored for the distributed exploration of simula-
Ronald, S. (1997). Robust encodings in genetic algorithms: A survey of encoding issues. Proceedings of 1997 IEEE International Conference on Evolutionary Computation (ICEC’97), 43–48.

Rudolph, G. (2001). Evolutionary search under partially ordered fitness sets. Universitätsbibliothek Dortmund.

Sánchez-Velazco, J., & Bullinaria, J. A. (2003). Gendered selection strategies in genetic algorithms for optimization. Survival, 8(6), 11.

Schraudolph, N. N., & Belew, R. K. (1992). Dynamic parameter encoding for genetic algorithms. Machine learning, 9(1), 9–21.

Shin, K.-s., & Han, I. (1999). Case-based reasoning supported by genetic algorithms for corporate bond rating. Expert Systems with applications, 16(2), 85–95.

Smith, R., & Bedau, M. (2000). Emergence of complex ecologies in echo. Proceedings from the international conference on complex systems on Unifying themes in complex systems, Perseus Books, 473–486.

Sobol, I. M. (1967). On the distribution of points in a cube and the approximate evaluation of integrals. Zhurnal Vychislitel’noi Matematiki i Matematicheskoi Fiziki, 7(4), 784–802.

Stanley, K. O. (2019). Why open-endedness matters. Artificial Life, 25(3), 232–235.

Stanley, K. O., Clune, J., Lehman, J., & Miikkulainen, R. (2019). Designing neural networks through neuroevolution. Nature Machine Intelligence, 1(1), 24–35.

Stanley, K. O., Lehman, J., & Soros, L. (2017). Open-endedness: The last grand challenge you’ve never heard of. While open-endedness could be a force for discovering intelligence, it could also be a component of AI itself. 

Street, A. G., & Mayo, S. L. (1999). Computational protein design. Structure, 7(5), R105–R109.

Such, F. P., Madhavan, V., Conti, E., Lehman, J., Stanley, K. O., & Clune, J. (2017). Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning. arXiv preprint arXiv:1712.06567.

Suganuma, M., Shirakawa, S., & Nagao, T. (2017). A genetic programming approach to designing convolutional neural network architectures. Proceedings of the genetic and evolutionary computation conference, 497–504.

Tanabe, R., & Ishibuchi, H. (2020). A review of evolutionary multimodal multiobjective optimization. IEEE Transactions on Evolutionary Computation, 24(1), 193–200. https://doi.org/10.1109/TEVC.2019.2909744

Tanese, R. (1987). Parallel genetic algorithm for a hypercube. Genetic algorithms and their applications: proceedings of the second International Conference on Genetic Algorithms: July 28–31, 1987 at the Massachusetts Institute of Technology, Cambridge, MA.

Tanese, R. (1989). Distributed genetic algorithms for function optimization.
Qualities, challenges and future of genetic algorithms

A. Vié

Tang, Q., Basten, T., Geilen, M., Stuijk, S., & Wei, J.-B. (2017). Mapping of synchronous dataflow graphs on mpsocs based on parallelism enhancement. *Journal of Parallel and Distributed Computing, 101*, 79–91.

Tao, Z., Huiling, L., Wenwen, W., & Xia, Y. (2019). Ga-svm based feature selection and parameter optimization in hospitalization expense modeling. *Applied Soft Computing, 75*, 323–332.

Tayarani-N, M.-H., Yao, X., & Xu, H. (2014). Meta-heuristic algorithms in car engine design: A literature survey. *IEEE Transactions on Evolutionary Computation, 19*(5), 609–629.

Turner, A. J., & Miller, J. F. (2014). Neuroevolution: Evolving heterogeneous artificial neural networks. *Evolutionary Intelligence, 7*(3), 135–154.

Varetto, F. (1998). Genetic algorithms applications in the analysis of insolvency risk. *Journal of Banking & Finance, 22*(10-11), 1421–1439.

Vié, A. (2020a). Evolutionary strategies with analogy partitions in p-guessing games. SSRN. [http://dx.doi.org/10.2139/ssrn.3645059](http://dx.doi.org/10.2139/ssrn.3645059)

Vié, A. (2020b). Genetic algorithm approach to asymmetrical blotto games with heterogeneous valuations. SSRN. [http://dx.doi.org/10.2139/ssrn.3667055](http://dx.doi.org/10.2139/ssrn.3667055)

Waheeb, W., & Ghazali, R. (2019). A new genetically optimized tensor product functional link neural network: An application to the daily exchange rate forecasting. *Evolutionary Intelligence, 12*(4), 593–608.

Wang, M., Li, B., Zhang, G., & Yao, X. (2017). Population evolvability: Dynamic fitness landscape analysis for population-based metaheuristic algorithms. *IEEE Transactions on Evolutionary Computation, 22*(4), 550–563.

Whitley, D. (1994). A genetic algorithm tutorial. *Statistics and computing, 4*(2), 65–85.

Wiransky, E. (2020). *Hands-on genetic algorithms with python*. Packt.

Wright, S. (1931). Evolution in mendelian populations. *Genetics, 16*(2), 97.

Yakovlev, S., Kartashov, O., & Pichugina, O. (2019). Optimization on combinatorial configurations using genetic algorithms. CMIS, 28–40.

Zitzler, E., Deb, K., & Thiele, L. (2000). Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary computation, 8*(2), 173–195.