Genetic Algorithms

William H. Hsu
Department of Computing and Information Sciences
Kansas State University
234 Nichols Hall
Manhattan, KS 66506-2302
USA
voice: +1 785-532-6350
fax: +1 785-532-7353
email: bhsu@cis.ksu.edu

Genetic Algorithms

William H. Hsu, Kansas State University, USA

INTRODUCTION

A **genetic algorithm** (**GA**) is a procedure used to find approximate solutions to search problems through application of the principles of evolutionary biology. Genetic algorithms use biologically inspired techniques such as genetic inheritance, natural selection, mutation, and sexual reproduction (recombination, or crossover). Along with genetic programming (GP), they are one of the main classes of genetic and evolutionary computation (GEC) methodologies.

Genetic algorithms are typically implemented using computer simulations in which an optimization problem is specified. For this problem, members of a space of candidate solutions, called *individuals*, are represented using abstract representations called *chromosomes*. The GA consists of an iterative process that evolves a working set of individuals called a *population* toward an objective function, or fitness function. (Goldberg, 1989; Wikipedia, 2004). Traditionally, solutions are represented using fixed-length strings, especially binary strings, but alternative encodings have been developed.

The evolutionary process of a GA is a highly simplified and stylized simulation of the biological version. It starts from a population of individuals randomly generated according to some probability distribution, usually uniform and updates this population in steps called generations. Each generation, multiple individuals are randomly selected

from the current population based upon some application of fitness, bred using crossover, and modified through mutation to form a new population.

- **Crossover** – exchange of genetic material (substrings) denoting rules, structural components, features of a machine learning, search, or optimization problem
- **Selection** – the application of the fitness criterion to choose which individuals from a population will go on to reproduce
- **Replication** – the propagation of individuals from one generation to the next
- **Mutation** – the modification of chromosomes for single individuals

This chapter begins with a survey of GA variants: the simple genetic algorithm, evolutionary algorithms, and extensions to variable-length individuals. It then discusses GA applications to data mining problems, such as supervised inductive learning, clustering, and feature selection and extraction. It concludes with a discussion of current issues in GA systems, particularly alternative search techniques and the role of building block (schema) theory.

BACKGROUND

The field of genetic and evolutionary computation (GEC) was first explored by Turing, who suggested an early template for the genetic algorithm. Holland performed much of the foundational work in GEC in the 1960s and 1970s. His goal of understanding the processes of natural adaptation and designing biologically-inspired artificial systems led to the formulation of the simple genetic algorithm (Holland, 1975).

State of the field: To date, GAs have been successfully applied to many significant problems in machine learning and data mining, most notably classification, pattern detectors (Rizki *et al.*, 2002; González & Dasgupta, 2003) and predictors (Au *et al.*, 2003), and payoff-driven reinforcement learning.[i] (Goldberg, 1989).

Theory of GAs: Current GA theory consists of two main approaches – Markov chain analysis and schema theory. Markov chain analysis is primarily concerned with characterizing the stochastic dynamics of a GA system, *i.e.*, the behavior of the random sampling mechanism of a GA over time. The most severe limitation of this approach is that while crossover is easy to implement, its dynamics are difficult to describe mathematically. Markov chain analysis of simple GAs has therefore been more successful at capturing the behavior of evolutionary algorithms with selection and mutation **only**. These include evolutionary algorithms (EAs) and *evolutionsstrategie*. (Schwefel, 1977).

Successful building blocks can become redundant in a GA population. This can slow down processing and can also resulting in a phenomenon called *takeover* where the population collapses to one or a few individuals. Goldberg (2002) characterizes "steady-state innovation" in GAs as the situation where time to produce a new, more highly-fit building block (the *innovation time,* t_i) is lower than the expected time for the most fit individual to dominate the entire population (the *takeover time*, t^*). "Steady state innovation" is achieved, facilitating convergence towards an optimal solution, when $t_i < t^*$, because the "countdown" to takeover or "race" between takeover and innovation is reset.

MAIN THRUST OF THE CHAPTER

The general strengths of genetic algorithms lie in their ability to explore the search space efficiently through parallel evaluation of fitness (Cantú-Paz, 2000) **and** mixing of partial solutions through crossover (Goldberg, 2002); maintain a search frontier to seek global optima (Goldberg, 1989); and solve multi-criterion optimization problems. The basic units of partial solutions are referred to in the literature as building blocks or *schemata*. Modern GEC systems are also able to produce solutions of variable length (De Jong *et al.*, 1993; Kargupta & Ghosh, 2002).

A more specific advantage of GAs is their ability to represent rule-based, permutation-based, and constructive solutions to many pattern recognition and machine learning problems. Examples of this include induction of decision trees (Cantú-Paz & Kamath, 2003) among several other recent applications surveyed below.

Types of GAs

The simplest genetic algorithm represents each chromosome as a bit string (containing binary digits: 0s and 1s) of fixed-length. Numerical parameters can be represented by integers, though it is possible to use floating-point representations for reals. The simple GA performs crossover and mutation at the bit level for all of these. (Goldberg, 1989; Wikipedia, 2004).

Other variants treat the chromosome as a parameter list, containing indices into an instruction table or an arbitrary data structure with pre-defined semantics, *e.g.*, nodes in a linked list, hashes, or objects. Crossover and mutation are required to preserve semantics by respecting object boundaries, and formal invariants for each generation can specified

according to these semantics. For most data types, operators can be specialized, with differing levels of effectiveness that are generally domain-dependent. (Wikipedia, 2004).

Applications

Genetic algorithms have been applied to many classification and performance tuning applications in the domain of knowledge discovery in databases (KDD). De Jong *et al.* produced *GABIL (Genetic Algorithm-Based Inductive Learning)*, one of the first general-purpose GAs for learning disjunctive normal form concepts. (De Jong *et al.*, 1993). *GABIL* was shown to produce rules achieving validation set accuracy comparable to that of decision trees induced using *ID3* and *C4.5*.

Since *GABIL*, there has been work on inducing rules (Zhou *et al*., 2003) and decision trees (Cantú-Paz & Kamath, 2003) using evolutionary algorithms. Other representations that can be evolved using a genetic algorithm include predictors (Au *et al.*, 2003) and anomaly detectors (González & Dasgupta, 2003). Unsupervised learning methodologies such as data clustering (Hall *et al.*, 1999; Lorena & Furtado, 2001) also admit GA-based representation, with application to such current data mining problems as gene expression profiling in the domain of computational biology (Iba, 2004). KDD from text corpora is another area where evolutionary algorithms have been applied (Atkinson-Abutridy *et al.*, 2003).

GAs can be used to perform meta-learning, or higher-order learning, by extracting features (Raymer *et al.*, 2000), selecting features (Hsu, 2003), or selecting training instances (Cano *et al.*, 2003). They have also been applied to combine, or fuse, classification functions (Kuncheva & Jain, 2000).

FUTURE TRENDS

Some limitations of GAs are that in certain situations, they are overkill compared to more straightforward optimization methods such as hill-climbing, feedforward artificial neural networks using backpropagation, and even simulated annealing and deterministic global search. In global optimization scenarios, GAs often manifest their strengths: efficient, parallelizable search; the ability to evolve solutions with multiple objective criteria (Llorà & Goldberg, 2003); and a characterizable and controllable process of innovation.

Several current controversies arise from open research problems in GEC:

- Selection is acknowledged to be a fundamentally important genetic operator. Opinion is, however, divided over the importance of crossover verses mutation. Some argue that crossover is the most important, while mutation is only necessary to ensure that potential solutions are not lost. Others argue that crossover in a largely uniform population only serves to propagate innovations originally found by mutation, and in a non-uniform population crossover is nearly always equivalent to a very large mutation (which is likely to be catastrophic).
- In the field of GEC, basic building blocks for solutions to engineering problems have primarily been characterized using schema theory, which has been critiqued as being insufficiently exact to characterize the expected convergence behavior of a GA. Proponents of schema theory have shown that it provides useful normative guidelines for design of GAs and automated control of high-level GA properties (e.g., population size, crossover parameters, and selection pressure).

Recent and current research in GEC relates certain evolutionary algorithms to ant colony optimization (Parpinelli, Lopes, & Freitas, 2002).

CONCLUSION

Genetic algorithms provide a comprehensive search methodology for machine learning and optimization. It has been shown to be efficient and powerful through many data mining applications that use optimization and classification.

The current literature (Goldberg, 2002; Wikipedia, 2004) contains several general observations about the generation of solutions using a genetic algorithm:

- GAs are sensitive to *deceptivity*, the irregularity of the fitness landscape. This includes locally optimal solutions that are not globally optimal; lack of fitness gradient for a given step size; and jump discontinuities in fitness.
- In general, GAs have difficulty with adaptation to dynamic concepts or objective criteria. This phenomenon, called concept drift in supervised learning and data mining, is a problem because GAs are traditionally designed to evolve highly-fit solutions (populations containing building blocks of high relative and absolute fitness) with respect to stationary concepts.
- GAs are not always effective at finding globally optimal solutions, but can rapidly locate *good* solutions, even for difficult search spaces. This makes *steady-state* GAs (Bayesian optimization GAs that collect and integrate solution outputs after convergence to an accurate representation of building blocks) a useful alternative to *generational* GAs (maximization GAs that seek the best individual of the final generation after convergence).

Looking ahead to future opportunities and challenges in data mining, genetic algorithms are widely applicable to classification by means of inductive learning. GAs also provide a practical method for optimization of data preparation and data transformation steps. The latter includes clustering, feature selection and extraction, instance selection. In data mining, GAs are likely to be most useful where high-level, fitness-driven search is needed. Non-local search (global search or search with an adaptive step size) and multi-objective data mining are also problem areas where GAs have proven promising.

REFERENCES

Atkinson-Abutridy, J., Mellish, C., & Aitken, S. (2003). A semantically guided and domain-independent evolutionary model for knowledge discovery from texts. *IEEE Transactions on Evolutionary Computation*, **7**(6), 546-560.

Au, W.-H., Chan, K.C.C., & Yao, X. (2003). A novel evolutionary data mining algorithm with applications to churn prediction. *IEEE Transactions on Evolutionary Computation*, **7**(6), 532-545.

Cano, J.R., Herrera, F., & Lozano, M. (2003). Using evolutionary algorithms as instance selection for data reduction in KDD: an experimental study. *IEEE Transactions on Evolutionary Computation*, **7**(6), 561-575.

Cantú-Paz, E. (2000). *Efficient and Accurate Parallel Genetic Algorithms.* Norwell, MA: Kluwer.

Cantú-Paz, E. & Kamath, C. (2003). Inducing oblique decision trees with evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, **7**(1), 54-68.

De Jong, K.A., Spears, W.M., & Gordon, F.D. (1993). Using genetic algorithms for concept learning. *Machine Learning*, **13**, 161-188, 1993.

Goldberg, D.E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning.* Reading, MA: Addison-Wesley.

Goldberg, D.E. (2002). *The Design of Innovation: Lessons from and for Competent Genetic Algorithms.* Norwell, MA: Kluwer.

González, F.A. & Dasgupta, D. (2003). Anomaly Detection Using Real-Valued Negative Selection. *Genetic Programming and Evolvable Machines*, **4**(4), 383-403.

Hall, L.O., Ozyurt, I.B., & Bezdek, J.C. (1999). Clustering with a genetically optimized approach. *IEEE Transactions on Evolutionary Computation*, **3**(2), 103-112.

Holland, J.H. (1975). *Adaptation in natural and artificial systems,* Ann Arbor: The University of Michigan Press.

Hsu, W. H. (2003). Control of Inductive Bias in Supervised Learning using Evolutionary Computation: A Wrapper-Based Approach. In *Data Mining: Opportunities and Challenges*, Wang, J. *editor*. Hershey, PA: Idea Group Publishing.

Iba, H. (2004). Classification of Gene Expression Profile Using Combinatory Method of Evolutionary Computation and Machine Learning. *Genetic Programming and Evolvable Machines, Special Issue on Biological Applications of Genetic and Evolutionary Computation (Banzhaf, W. & Foster, J., guest editors)*, **5**(2), 145-156.

Kargupta, H. & Ghosh, S. (2002). Toward Machine Learning Through Genetic Code-like Transformations. *Genetic Programming and Evolvable Machines*, **3**(3), 231-258.

Kuncheva, L.I., Jain, L.C. (2000). Designing classifier fusion systems by genetic algorithms. *IEEE Transactions on Evolutionary Computation*, **4**(4), 327-336.

Llorà, X. & Goldberg, D. E. (2003). Bounding the effect of noise in Multiobjective Learning Classifier Systems. *Evolutionary Computation*, **11**(3), 278 – 297.

Lorena, L.A.N. & Furtado, J. C. (2001). Constructive Genetic Algorithm for Clustering Problems. *Evolutionary Computation*, **9**(3), 309 – 328.

Parpinelli, R.S., Lopes, H.S., & Freitas, A.A. (2002). Data mining with an ant colony optimization algorithm. *IEEE Transactions on Evolutionary Computation*, **6**(4):321- 332.

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.

Rizki, M.M., Zmuda, M.A., & Tamburino, L.A. (2002). Evolving pattern recognition systems. *IEEE Transactions on Evolutionary Computation*, **6**(6), 594-609.

Schwefel, H.-P. (1997). *Numerische Optimierung von Computer--Modellen mittels der Evolutionsstrategie*. Vol. 26 of Interdisciplinary Systems Research, Basel: Birkhauser Verlag.

Wikipedia (2004). *Genetic Algorithm*. Available from URL: http://en.wikipedia.org/wiki/Genetic_algorithm.

Zhou, C., Xiao, W., Tirpak, T. M., & Nelson, P.C. (2003). Evolving accurate and compact classification rules with gene expression programming. *IEEE Transactions on Evolutionary Computation*, **7**(6), 519-531.

TERMS AND DEFINITIONS

Crossover: In biology, a process of sexual recombination, by which two chromosomes are paired up and exchange some portion of their genetic sequence. Crossover in GAs is highly stylized and typically involves exchange of strings. These can be performed using a crossover bit mask in bit-string GAs, but require complex exchanges (such as partial-match, order, and cycle crossover) in permutation GAs.

Evolutionary Computation: A solution approach based on simulation models of natural selection, which begins with a set of potential solutions, then iteratively applies algorithms to generate new candidates and select the fittest from this set. The process leads toward a model that has a high proportion of fit individuals.

Generation: The basic unit of progress in genetic and evolutionary computation, a step in which selection is applied over a population. Usually, crossover and mutation are applied once per generation, in strict order.

Genetic programming (GP): see *Genetic Programming* entry.

Individual: A single candidate solution in genetic and evolutionary computation, typically represented using strings (often of fixed length) and permutations in genetic algorithms, or using "problem solver" representations – programs, generative grammars, or circuits – in genetic programming.

Meta-learning: Higher-order learning by adapting the parameters of a machine learning algorithm or the algorithm definition itself, in response to data. An example of this approach is the search-based wrapper of inductive learning, which "wraps" a search algorithm around an inducer to find locally optimal parameter values such as the relevant feature subset for a given classification target and data set. Validation set accuracy is

typically used as fitness. Genetic algorithms have been used to implement such wrappers for decision tree and Bayesian network inducers.

Mutation: In biology, a permanent, heritable change to the genetic material of an organism. Mutation in GAs involves string-based modifications to the elements of a candidate solution. These include bit-reversal in bit-string GAs and shuffle and swap operators in permutation GAs.

Permutation GA: A type of GA where individuals represent a total ordering of elements, such as cities to be visited in a minimum-cost graph tour (the Traveling Salesman Problem). Permutation GAs use specialized crossover and mutation operators compared to the more common bit string GAs..

Schema (pl. Schemata): An abstract building block of a GA-generated solution, corresponding to a set of individuals. Schemata are typically denoted by bit strings with don't-care symbols `#': for example, 1#01#00# is a schema with $2^3 = 8$ possible instances, one for each instantiation of the # symbols to 0 or 1. Schemas are important in GA research because they form the basis of an analytical approach called schema theory, for characterizing building blocks and predicting their proliferation and survival probability across generations, thereby describing the expected relative fitness of individuals in the GA.

Selection: In biology, a mechanism in by which the fittest individuals survive to reproduce, and the basis of speciation according to the Darwinian theory of evolution. Selection in GP involves evaluation of a quantitative criterion over a finite set of fitness cases, with the combined evaluation measures being compared in order to choose individuals.

[i] *Payoff-driven reinforcement learning* describes a class of learning problems for intelligent agents that receive rewards, or reinforcements, from the environment in response to actions selected by a policy function. These rewards are transmitted in the form of payoffs, sometimes strictly nonnegative. A GA acquires policies by evolving individuals, such as condition-action rules, that represent candidate policies.