Evolutionary Computation and 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 the 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 typically are 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*. For each generation, multiple individuals are selected randomly 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 article begins with a survey of the following 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 (1975) 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 applied successfully to many significant problems in machine learning and data mining, most notably classification, pattern detectors (González & Dasgupta, 2003; Rizki et al., 2002) and predictors (Au et al., 2003), and payoff-driven reinforcement learning[1] (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 evolutions strategie (Schwefel, 1977).

Successful building blocks can become redundant in a GA population. This can slow down processing and also can result 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 toward an optimal solution, when $t_i < t^*$, because the countdown to takeover, or race, between takeover and innovation is reset.

MAIN THRUST

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); to maintain a search frontier to seek global optima (Goldberg, 1989); and to 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 also are 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 in the following.

Types of Gas

The simplest genetic algorithm represents each chromosome as a bit string (containing binary digits 0 and 1) 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 be specified according to these semantics. For most data types, operators can be specialized, with differing levels of effectiveness that generally are 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 also have 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, feed-forward artificial neural networks using back propagation, 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, however, is divided over the importance of crossover vs. 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