

Evolutionary Computation

CAP 5512

Ivan Garibay
igaribay@cs.ucf.edu
12201 Research Parkway, Suite 501
(407) 882-1163
<http://ivan.research.ucf.edu>

Spring 2007

1 Stochastic Search

“Further research into intelligence of machinery will probably be very greatly concerned with searches... There is the genetical or evolutionary search by which a combination of genes is looked for, the criterion being the survival value. The remarkable success of this search confirms to some extent the idea that intellectual activity consist mainly of various kinds of search”

–ALAN TURING, “Intelligent Machinery”, 1948

As early as the very beginnings of the science of computation, in the times of Charles Babbage and his “analytical engine”, *search* has been recognized as an important conceptual tool for problem solving. Later, in the initial stages of modern computational theory and artificial intelligence, Alan Turing proposed various kinds of search as a means to achieve machine intelligence. Since then, search, in particular *heuristic search*, has played an important and historical role in artificial intelligence. In heuristic search, a computer seeks the answer to a problem by searching through the space of all possible candidate solutions using heuristics to guide the search toward promising areas. These heuristics are simply guiding principles obtained by using knowledge about the nature of the problem being solved.

Since the very beginning, there has been an intuitive understanding that, because heuristics are based on specific knowledge of the target problem, a universally best search algorithm, which is good for all problems, does not exist. It is only recently, however, that Wolpert and Macready (1997) have provided a formal proof. In their *No Free Lunch* theorems, they show, in essence, that an algorithm that performs well in one problem class is guaranteed to perform poorly on another. More importantly, they show that all search algorithms

perform the same when averaged over all possible problems. As a result, there is a fundamental tradeoff between algorithm robustness across problems and algorithm performance.

Given that there is no universal search algorithm we are left with the task of designing algorithms that are specialized for a particular problem or a problem class. When there is an abundance of domain knowledge about the problem that we are trying to solve, a good deterministic algorithm can be designed and often its convergence to the optimal can be guaranteed. If the domain knowledge is incomplete, we still have the possibility of using some appropriate heuristic to guide the search. Unfortunately, for many interesting theoretical and practical problems, the space to be searched is too complex or too vast or both for us to devise an obvious heuristic or to collect a meaningful amount of domain knowledge. These problems are difficult. For the worst of these cases, the best we can do is random or brute-force exhaustive search. Fortunately, many of these difficult problems share a key property that can be leveraged to guide the search: there is a *positive correlation between the form and quality of candidate solutions*. In other words, small changes in the form of a solution cause small changes in the resulting quality of the solution. *Stochastic search algorithms* are the class of search algorithms based on this key property. Stochastic search algorithms are more robust than other types of search algorithms in the sense that the key property they are based upon is quite general across problems and that they require less domain knowledge. They are weaker than other search algorithms in the sense that convergence to an optimum is usually not guaranteed. However, stochastic search algorithms often outperform random search and they have been shown to have wide applicability on this class of difficult problems where no strong heuristics are available, especially on computationally hard combinatorial problems such as planning, scheduling, constraint satisfaction, and satisfiability.

The class of stochastic search algorithms includes, among others, the following algorithms, simulated annealing, stochastic hill-climbing, tabu search, ant colony optimization, and evolutionary computation.

2 Evolutionary Computation

This preservation of favorable individual differences and variations and the destruction of those which are injurious, I have called Natural Selection, or the Survival of the Fittest.

—CHARLES DARWIN, “The Origin of Species”, 1859.

Evolutionary Computation (EC) is a relatively young field. According to the *Handbook of Evolutionary Computation* (Bäck, Fogel & Michalewicz 1997), the term itself was coined in 1991 as an attempt to bring together three related research communities: *Evolutionary Programming* (Fogel 1962, Fogel 1964), *Evolution Strategies* (Rechenberg 1965) and *Genetic Algorithms* (Holland 1962,

Holland 1975, Goldberg 1989). In the early 1960's, these communities independently developed evolutionarily inspired computational paradigms that share the basis of any evolutionary process: reproduction, random variation, competition and selection. The earliest work on evolutionarily inspired computation can be traced back, at least, to the apparition of the first electronic computers, i.e., see Fraser (Fraser 1957) and Friedberg (Friedberg 1958, Friedberg, Dunham & North 1959).

In essence, evolutionary computation searches for good solutions to problems by testing large number of candidate solutions, selecting the “better” ones and randomly changing them to form a new set of candidates to be tested. This process repeats until a “good enough” solution is found or until the computational resources are exhausted.

Evolutionary computation is inspired by the process of *evolution by natural selection* originally proposed by Darwin (Darwin 1859). The EC jargon is grossly adapted from terms used in Genetics, Population Genetics, Evolutionary Biology, Molecular Biology, Evolutionary Theory and related fields. In EC, a candidate solution to a problem is known as an *individual*. A collection of such individuals form a *population*. When individuals undergo variations to generate another individuals, they are said to be *breeding*. While testing an individual, we assign a quantitative measurement of its performance known as its *fitness*. The set of individuals created by replication and variation of the current population are known as the next *generation*. The overall process of finding a good solution using this technique is known as *evolving* a solution.

Evolutionary Computation searches for “good solutions” over the space of all possible candidate solutions, the *search space*. In many cases, the variations to generate new individuals are not performed on the search space itself but on an auxiliary space, the *representation space*.

In this case, the individuals of the population are members of the representation space and are known also as *genomes*, *genotypes*, or *chromosomes*. Elements of the search space are known as *phenotypes*, and the process of mapping a genotype (structure subject to variation) into a phenotype (candidate solution) is known as *morphogenesis*.

3 Genetic Algorithms

Genetic algorithms were proposed by Holland in the context of *adaptive systems*. Their initial motivation was twofold: to understand the principles of natural adaptive systems and to construct robust artificial adaptive systems that could successfully respond to unexpected environmental changes. In contrast, Evolutionary Strategies were initially developed to evolve optimization techniques and Evolution Programming, to evolve intelligent agents. Another fundamental difference is that the main source of variation on Genetic Algorithms is recombination or *crossover*. The central role of recombination in the GAs is initially explained and further analysed by Holland in the *Building Block Hypothesis* and in its famous *Schema theorem* (Holland 1975). For the other paradigms,

```

procedure GA
{
  initialize population;
  while termination condition not satisfied do
  {
    evaluate current population;
    select parents;
    apply genetic operators to
      parents to create offspring;
    set current population equal to
      be the new offspring population;
  }
}

```

Figure 1: Basic steps of a typical genetic algorithm.

the main source of variation is mutation.

The idea of a genotype to phenotype separation, as found in nature, has been a central part of the GA since its inception. Individuals are represented by structures called genotypes. GA operators manipulate the genotypes of each individual. After a decoding process, these genotypes are translated into phenotypic structures upon which fitness is evaluated. The GA genotypes are typically, but not always, restricted to bit strings. A segment or various segments of the genotype that represent a component of a solution are usually called *genes*.

Figure 1 shows the basic steps of a GA. The initial population may be initialized randomly or with user-defined individuals. The GA then iterates thru an evaluate-select-reproduce cycle until either a stopping condition is satisfied or the computational resources are exhausted.

References

- Bäck, T., Fogel, D. B. & Michalewicz, Z., eds (1997), *Handbook of Evolutionary Computation*, IOP Publishing Ltd and Oxford University Press.
- Darwin, C. (1859), *The Origin of Species*, J. M. Dent and Sons Ltd, 1975.
- Fogel, L. J. (1962), ‘Autonomous automata’, *Industrial Research* **4**, 14–19.
- Fogel, L. J. (1964), On the Organization of Intellect, PhD thesis, University of California at Los Angeles.
- Fraser, A. (1957), ‘Simulation of genetic systems by automatic digital computers’, *Australian Journal of Biological Sciences* **10**, 484–499.
- Friedberg, R. M. (1958), ‘A learning machine: Part i’, *IBM Journal* **2**, 2–13.
- Friedberg, R. M., Dunham, B. & North, J. H. (1959), ‘A learning machine: Part ii’, *IBM Journal* **3**, 282–287.

- Goldberg, D. E. (1989), *Genetic algorithms in search, optimization, and machine learning*, Addison Wesley.
- Holland, J. H. (1962), ‘Outline for a logical theory of adaptive systems’, *ACM* **9**, 297–314.
- Holland, J. H. (1975), *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, MI.
- Rechenberg, I. (1965), Cybernetic solution path of an experimental problem, Translation 1122, Royal Aircraft Establishment Library.