Stochastic Optimization using the Genetic Algorithm

Simon Austin Lee*
Jack Baskin School of Engineering
University of California, Santa Cruz
1156 High St., Santa Cruz 95064
siaulee@ucsc.edu

Abstract

Stochastic optimization methods refer to the use and generation of random variables. This is prevelant in the class of evolutionary algorithms, in which the focus of this paper is specifically that of the *genetic algorithm*, which is used in biology, physics, computer science, and engineering to compute solutions for complex combinatorial optimization problems which we have no other way to calculate for solutions. In computer science, there remains a major unsolved hypothesis called the *"P versus NP problem"*, which discloses whether every problem whose solutions can be quickly solved for, can also be verified quickly. *P* stands for *polynomial time*, which roughly means a set of relatively easy problems to be solved for. *NP* stands for *non-deterministic polynomial time* which is a set of difficult problems to be solved for. So, if *P = NP*, this would imply that the difficult set of problems have relatively easily computed solutions. This theoretical question is the foundation of the *genetic algorithm*, which aims to find the best solution, by the evolutionary process of *natural selection*, at exponentially fast time complexities which would take years to compute if brute-forced. The essence of this algorithm is applied to research a condensed population, in order to design a distribution strategy of vaccines to minimize the number of infenctions within an epidemic outbreak.

1 Introduction

Nonlinear Dynamical Systems have wide applications in biology, physics, computer science and engineering especially in that of *combinatronics*, which is the area of mathematical studies that focuses on counting to obtain results and specific properties of finite structures. In simpler terms, finding the best combination of *things*, which will be a term used loosely to describe processes from a birds-eye view, can lead to optimized solutions for problems which simply cannot be solved for in any other way.

A clear cut example of such optimization could be seen through the *Knapsack problem*, which given a set of items, their corresponding weights, and a value, the goal is to determine the number of items to include in a collection so that the total weight remains less than or equal to the given limit while still trying to take a value as large as possible. The issues that arises within the *Knapsack Problem*, lies when there is some large n integer of items to be chosen from, in which trying to compute all the combinations by hand loses efficiency. This whole idea of efficiency leads to one of the largest unsolved theoretical hypotheses in computer science called the *P equals NP problem*, where it asks whether every problem whose solution can be quickly checked for correctness can also be computed quickly. An answer to the *P versus NP* question would open up whether problems that can be verified in polynomial time (*denoted as P*) can also be solved in polynomial time. If it turns out that $P \neq NP$, which is the more popular belief, it would mean that there are problems in non-deterministic polynomial time (*denoted as NP*) that are harder to solve than to verify: they could not be solved in polynomial time, but the answer could be verified in polynomial time.

Therefore, a solution to the *Knapsack problem* and other time-consuming problems, is the *genetic algorithm*, which is widely used in stochastic optimization because it is great at taking large, potentially massive search spaces and navigating through them, finding the solutions, to problems you might not find otherwise in a lifetime. It takes the possibility of approaching these *NP* problems, and finding very accurate approximations for them.

*Supported in part by the Directed Reading Program of the UCSC Math Department as well as ProfessorDaniele Venturi of the Baskin School of Engineering, Applied Mathematics Department

2 The Genetic Algorithm

The Genetic Algorithm (GA), is a metaheuristic algorithm inspired by *Charles Darwin's* theory of *Natural Selection*, used for solving Optimization and Search problems in machine learning. This algorithm's strict purpose is to solve for problems that would take a long time to solve, often producing one of the best approximated solutions. The ability to begin solving *NP* problems is just scratching the surface and with this algorithm there is an expectation for more complex systems and hypotheses to be approached in this manner.

2.1 How it Works

I. The way the genetic algorithm works is through bits in binary showing membership to a set, where 0 is denoted as not being a member of a particular set, and 1 being a member of a particular set. Before populating a set of possible solutions, it must be determined three factors: how long each bit sequence is, to represent the n amount of *things* that are to be selected from, how many possible solutions we want to be produced, and a limit that if succeeded willdisqualify it as a valid solution. For example, the singular binary sequence *1001010111* which signifies a singular solution, shows a bit length of 10 resembling 10 total *things* within a set, while we see that 6 *things* are apart of this solution simply by counting the number of 1's that are present in the sequence. It is important to note that the bit sequence length never increases or decreases because there are a fixed amount of items present, and that any number of bits can be flipped to 1 as long as it remains less than or equal to the limit set. Once a limit and integer n are chosen, n amount of solutions will be produced of some selected arbritary length l. This population of random possible solutions are what we refer to as a *generation* and this first generation is called generation 0 (generations can be tracked by *i-1 generations*, where i marks the number of iterations). Furthermore, because all the possible solutions were randomly generated, this evolutionary process begins in complete chaos.

II. Next comes an iterative sequence that will continue as long as there are non-zero solutions from the pervious generation. A solution is changed to zero if and only if the total sum of the collection of items succeeds the limit set in part **I**.