Evolutionary Computation in Financial Engineering: A Road Map of GAs and GP

By Shu-Heng Chen, Ph.D.

Editor's Note: This is an abridgement of Professor Chen's extensive article which is published in its entirety on the Financial Engineering News web site at http://www.fenews.com/1998/Issue6/chen.pdf. References have been eliminated here for sake of brevity; they are included in the full article.

Motivation

In this and previous issues, R. Baker and S. Smith have introduced the techniques of evolutionary computation and their application to financial engineering. In this article, we would like to first summarize a few major insights revealed in those two articles and then provide to a roadmap for readers to initialize their own trails.

First of all, why evolutionary computation? Where is the motivation? In engineering or applied mathematics, evolutionary computation has been extensively applied to problems whose solution space is irregular, i.e., too large and highly complex, so that it is difficult to employ conventional optimization procedures to search for the global optimum.

Now, are solution spaces for most financial optimization problems irregular? My answer is yes, and I am not alone to entertain such a view. General acceptance of this property has in fact fostered the growth of financial engineering. Take financial forecasting for example. There is a tendency to jettison the early well-accepted random walk hypothesis or the unpredictability of financial time series. Nonlinear models such as artificial neural networks have proved to be capable of beating random walks not only statistically significantly but also economically significantly. Suffice it to say, many nonlinear models have the potential to beat the random walk but the search space of nonlinear models is doubtlessly large and complex. Similarly, in technical analysis, search over the space of trading rules is obviously a daunting task.

But, is evolutionary computation the only choice for us to cope with the irregular search space? Of course not. There are many equally promising tools available in the machine learning literature. However, evolutionary computation should not be considered as a kind of optimization technique to compete with other alternative techniques but an optimization

principle to be incorporated with existing techniques. It is more a complement to than a substitute for many existing techniques.

Evolutionary Computation: A Principle for Search

What is evolutionary computation? Briefly, evolutionary computation refers to a collection of stochastic search algorithms whose designs are gleaned from natural evolution: genetic inheritance and the Darwinian principle of the survival-of-the-fittest (natural selection).

The several different styles of evolutionary algorithms each share a common feature, i.e., modeling the search process by mimicking a biological evolution process which is operated over the solution space. They are different mainly in the evolution operators involved and the representation of the solution space. For example, in genetic programming, each solution is represented by a computer program, and hence the evolution process is implemented on a society of computer programs. In financial application, this society of computer programs can be a society of option pricing formulas, trading rules, or forecasting models. As an optimization principle, genetic programming is to make the selection process of these computer programs act as natural selection.

Given this introduction, it is not hard to see that the role which evolutionary computation can play in relation to other optimization techniques can be quite cooperative rather than competitive. For example, in terms of financial forecasting, one of the most promising tools is artificial neural nets. However, designing a neural net involves a series of selections, including number of hidden layers, number of inputs, number of hidden nodes, type of transfer function, type of connection, learning styles, etc. None of them are trivial and many of them have no theoretical answer. As a result, over the last few years evolutionary programming (EP), genetic algorithms (GAs) and genetic programming (GP) have all been applied to the determination of the architecture of artificial neural nets (ANAS).

While evolutionary computation as a search principle can be extensively used in financial engineering, up to the present, applications have been largely restricted to only two styles: genetic algorithms and genetic programming. In the rest of this paper, we will go over a few applications. The areas reviewed by us are trading strategies and financial forecasting. The reason to choose these two areas is because applications over them are well-established in the sense rigorous performance evaluations are available and promising. By doing this, we unavoidably skip the youngest but very active area, i.e., option pricing.

Trading Strategies

Smith, in this issue, mentions three applications of GP to financial engineering. One of them is the development of trading strategies. The main idea of using GAs to evolve trading strategies is to encode the variable one wants to optimize, e.g., the trading strategy, as binary string and work with this string.

Genetic Programming as the Advanced

Now, we turn to a different style of evolution computation, namely, genetic programming. GP works in a pretty similar fashion to that of GAs. They both apply the operation of Darwinian selection, crossover and mutation. However, there is an essential difference between GAs and GP which makes GP a generalization of GAs: unlike GAs whose population is composed of fixed-length binary strings, the population of GP is composed of programs. In particular, each program in GP is written in its LISP S-Expression and can be depicted as a parse tree. Briefly speaking, by appropriately defining the terminal set and function set, each program can be written in LISP S-Expression.

But, why GP instead of GAs? What difference does it make? GAs in most of their applications have a finite-length string. This finite length implies a number of restrictions, such as the number of predicates and their contents.

The first issue is how many trials should be conducted before one can reach meaningful results. While there is no simple answer to this question, one has to bear in mind that both GAs and GP are stochastic search algorithms. They are stochastic in the sense that the results obtained are in some sense random. It is quite normal that different trials can have different results. Therefore, it is not appropriate to use or to evaluate the performance of GAs or GP based on a few trials.

The second issue is how to deal with a population of rules. One essential characteristic of evolutionary algorithms is that they are population-to-population, instead of point-to-point, search. In the end, what we have is not a single answer or a single rule but a population or rules. This situation can be further complicated by multiple runs, in which we have a population of populations of trading rules. For example, a population size of 500 and 100 trials provides a population of 50,000 trading rules.

The third issue is how to avoid overlearning or overfitting. It is quite common to see that EC techniques can outperform other techniques over the training sample but are outdone by others over the testing sample. Even within EC techniques, the best rule discovered after thousands of generations can turn out to be the worst performer on the testing sample. This predicament known as winner's curse in economics can be managed in several possible ways. There are two commonly used techniques employed. First, introduce a

selection period. However, this procedure reduces, but does not eliminate, the problem of overfitting." The rule that did best in the selection period turns out to be one of the poorest performers out-of-sample. The second solution is to add a penalty function (complexity function) to the fitness function to bias the selection toward simple models.

The last, but by no means the least, issue is the computing time. GP has been highly computation intensive.

Financial Forecasting

While more than a dozen techniques can be applied to financial forecasting, GP has some peculiarities in this area. First, from a theoretical viewpoint, GP should be considered particularly relevant to predicting chaotic time series, which are characteristic of many financial time series. Chaotic time series is a deterministic system which can behave quite erratically. As illustrated in
many textbooks, simple nonlinear dynamic equations, such as logistic maps, are able to exhibit this phenomenon. These equations can be simple in terms of the number of nodes and levels required to represent them. Therefore, simple mathematics can show that genetic programming can be promising in catching the regularities of these financial time series with even a few dozen data points. Nevertheless, a weakness of GP's is its sluggishness to learn the constants.

Second, a productive way to employ GAs and GP in financial forecasting is to hybridize them with other techniques. GAs or GP have been combined with other techniques such as the nearest neighbor, wavelets, and neural nets to do forecasting.

Third, perhaps the most novel approach of using GAs or GP in financial forecasting is the forecasts based on artificial financial markets. The artificial stock market consists of a computer model in which simulated traders update an array of trading rules over time. At intervals, the traders attempt to discover better sets of rules using machine learning techniques, e.g., genetic algorithms and classifier systems. The price of the stock in the market and the market behavior are entirely endogenously generated. Concluding Remarks

This paper gives a brief survey of the current development of evolutionary computation, with special emphasis on genetic algorithms and genetic programming in financial engineering. We have no intention to mislead the reader into thinking that GAs or GP is the best technique or the only promising technique. However, this paper does suggest using GAs or GP as a search principle to be incorporated with existing techniques so that further improvement can be expected. This is particularly true when our knowledge about a specific technique is not precise, and GAs or GP can be used to initialize a trial-and-error

process. Evolutionaryartificial neural networks give a perfect, though not the only,example of this kind of application. Furthermore, when technical restrictions, such as discontinuity and non-differentiability, paralyze the applicability of a particular technique, then GAs or GP can be called in as a resort. Technical analysis is one of these applications.

There is a final remark on GP. Consider what you currently have as the initial generation of the GP evolution, then after 5,50, 500, 50000, or more generations of run, GP might add something to be appreciated. If it does, you win. If it does not, what is currently known to you is, to some extent, the best available. In this sense, GP is just a computerized version of the progress of human intelligence.

Shu-Heng Chen is an Associate Professor in the AI-ECON Research Group, Department of Economics, National Chengchi University, Taipei, R.O.C. He received an M.A. (Economics) from National Taiwan University, an M.A. (Mathematics) and a Ph.D. (Economics) from the University of California, Los Angeles. He is the editor of Taiwan Journal of Political Economy. His email address is chchen@nccu.edu.tw