JOIM
www.joim.com

Journal Of Investment Management, Vol. 3, No. 2, (2005), pp. 77–82

SURVEY OF THE LITERATURE

GENETIC ALGORITHMS

Sanjiv Ranjan Das[a]

1 Introduction

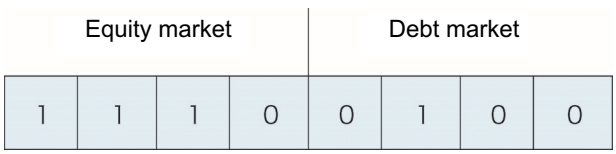
The use of genetic algorithms (GAs) in finance has focused primarily on the area of investments, in particular in the uncovering of trading rules (see Allen and Karjalainen, 1999). Genetic algorithms have also been used in the estimation of econometric models. A standard text that provides an easily accessible introduction to the subject is Goldberg (1989). A nice survey on the application of GAs to investment strategies is the book by Bauer (1994).

2 Overview of GAs

GAs deliver the optimal strategy, best trading rule, fittest parameters, etc., by means of mimicking Darwinian selection. This selection is made over hundreds of possible candidate strategies, and their offspring. The best trading rules are discovered by an evolutionary process implemented on a computer. There are two sets of rules that operate in a GA system. One, a rule that determines which candidate strategy is the best, and when the process of evolution may be halted. Two, the set of rules that generates more candidates for evaluation.

For example, in the evaluation of trading rules, the best candidate trading strategy may be, for example, the one that yields the highest risk-adjusted profits on back-testing. The stopping rule for the GA may state that no more iterations will be performed if the most profitable (fittest) rule from one iteration does not exceed that of the prior iteration by at least an amount ϵ.

Each trading rule is represented by a string of numbers. While there is nothing in the theory of GAs that specifies precisely the representation of a trading rule, it is not unusual to specify it as a string of length n, comprising zeros and ones. For example, a simple market timing trading rule may be represented as a string of eight numbers, the first four relating to the equity market and the last four to the debt markets. This eight-digit string (chromosome) comprises the "DNA" of the trading rule. By generating random strings of this length, we may create as many as $2^8 = 256$ different trading rules. The diagram below depicts one such rule:



[a] Santa Clara University, USA.

The market timing rule may be stipulated as follows: If the last 15 days' equity return exceeds the value of the number represented by the binary digits in the first four places of the rule, and the last 15 days' bond market return is less than the number in the last four digits, then it implies that a shift from bonds into equities is called for. Also, if the equity return lies below the number in the first four digits, and the bond return lies above that of the last four digits, then a shift from equity into debt is recommended. (In the diagram above, based on the binary digits, the cutoff values are 14% for equity and 4% for debt.)

Because we may generate several random binary strings, it is very easy to quickly create an initial set of trading rules. All these rules form candidates for the best strategy, and then may be back-tested to determine which is best. This mimics Darwinian selection by survival of the fittest.

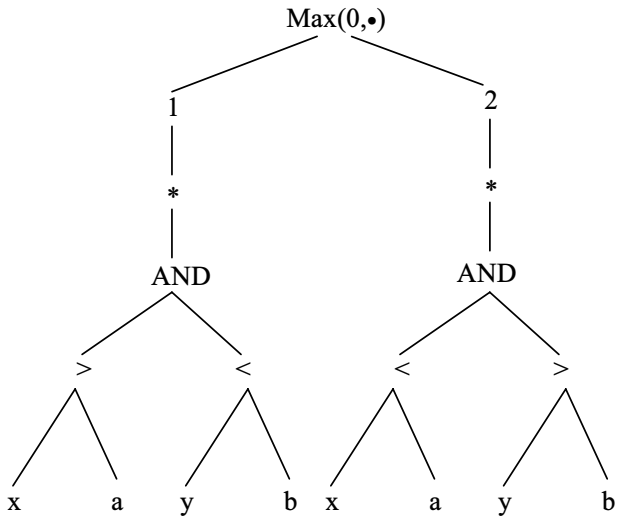
In our example, we have just 256 possible market timing rules, and hence exhaustive checking of all possible strategies is feasible. More complicated trading rules may be represented by far more than 8 bits. An n-bit rule would generate 2^n possible strategies, and for large n, it is infeasible to back-test all of them. Hence, only a limited set of rules is initialized by random generation, and the fittest of these are allowed to bear "offspring," which results in new and better rules. The beauty of GAs is that the technique very rapidly finds very good (or even the optimal) rules within a short run time.

The first random set of rules is called the initial "generation" of the population. The size of this generation is denoted $m \ll 2^n$. Every member of this population is back-tested for profitability, and its score (fitness criterion) is recorded. Each generation evolves into the next by means of three Darwinian operations (the procedure outlined below admits many variations, but usually respects the three operations in broad form):

1. *Death and birth*: A fraction δ of the population will die in the cycle from one iteration to the next. The lowest scoring δm of the rules are eliminated from the population. Next, the top βm of the population is duplicated. This birth process returns the size of the population to m if $\beta = \delta$. At the end of this stage, so far, no improved rules have been created. However, the average fitness of the population has been enhanced by the death of weaker rules and the birth of stronger ones.
2. *Crossover*: At this stage, the entire population engages in mating, resulting in $m/2$ pairings. Each pair exchanges a random substring, resulting in a crossover of DNA from one trading rule to another. For example, the bits 3–5 may be exchanged by one pair of rules. This leads to entirely new rules (children). There are many variations possible here. The substrings of bits exchanged may not be from the same starting bit, though they need to be of the same length. In another variation, commonly adopted, the same substring location is used for all crossovers in the population. In our market timing example, either the equity substring or the debt substring may be exchanged. Since the initial death and birth process has resulted in a stronger rule set, the exchange of good genetic material across pairs of rules may result in even better offspring (the theoretical literature shows that this works most of the time, barring some needle-in-haystack cases, see Goldberg, 1989, chapter 2.)
3. *Mutation*: The exchange of genetic material is a systematic way in which succeeding populations become fitter. Sometimes, idiosyncratic improvements also occur, through mutation. In order to impose this in a GA, we pick a small fraction of the population, say 1%, and flip a single random bit in the n-bit string for each of these chosen chromosomes.

At the end of these three operations, the procedure may then be repeated; each time the fitness of the

rules is computed, and the three operations performed. After some iterations, the best rule will only make a very marginal improvement and we may halt the procedure. The top few of the trading rules will be the fittest. Of course, GAs are not guaranteed to work, and there are some problems that have been shown to be "GA-hard," i.e., which require exponential time to solve. These are usually problems with extreme optima, and tend to be hard to solve for many other algorithms as well.



3 Genetic Programming

A particular variant of genetic algorithms is genetic programming (GP). A GP develops rules (for example, trading strategies) using expression trees. Hence, the basic unit of structure is not the chromosome but a free-form expression. We can explain this representation using the market timing example we previously employed. Let x denote the past 15 days' average return in the equity market, and y the same for the bond market. We denote the cutoff level of the equity return as a and that of the bond return as b. As before, we specify that if $x > a$ and $y < b$, then we shift investment into the equity market. And if $x < a$ and $y > b$, then we shift into the bond market. If neither is true we do nothing. Denoting the output variable as z, we specify that "do nothing" is denoted by $z = 0$, invest in equities is denoted $z = 1$, and investing in bonds is $z = 2$. The mathematical expression that depicts this logic is as follows:

$$z = \max\,[0, 1 \times ((x > a)\ \text{AND}\ (y < b)), \\ 2 \times ((x < a)\ \text{AND}\ (y > b))]$$

where the expressions in $(\cdot)$ are boolean and output either a 0 or 1 value. This mathematical statement may also be represented in an expression tree, which appears as follows:

GP is characterized by the expression tree. The population of candidate trading rules comprises a set of expression trees, and these trees may be randomly generated. We note the following aspects of GP:

1. There are two types of structural entities in GPs: variables and operators. In the expression tree above we had the following variables: inputs $\{x, y\}$ and constants $\{a, b, 0, 1, 2\}$. The operators came from the following set: $\{\max, \text{AND}, *, <, >\}$.
2. GP has more flexibility than the GA described in Section 2. One may easily see that the generation of flexible length random expression trees might result in more forms of rules than fixed length chromosomes.
3. The crossover operation in the case of GPs is undertaken by two parent expression trees swapping subtrees. The nodes from which the swap is undertaken need not correspond to the two parents, simply because random trees do not have the same structure, unless this is strictly imposed while generating expression trees. The children expression trees will also not have the same structure as their parents. Hence, the offspring tend to be of much richer and varied form in the case of GPs than in the case of GAs.

4. In the case of GAs, two identical parents will result in an identical child after the crossover operation. This is not necessarily the case with GPs because the nodes at which crossover occurs in both parents might be different. Hence, the subtrees that are swapped may be different.
5. The mutation operation may be undertaken by randomly replacing one tree operator with another, or a tree variable with another.

Each rule in the population of candidate rules is parsed through a fitness function to determine the ranking of rules. After iterating, the best rule is determined. The expression tree may be written in algebraic form and then applied to the purpose intended.

Overall, both GAs and GPs have been found to be extremely good at searching very large spaces for optimal or near-optimal solutions with run times that are comparable to other algorithms. The applicability of GA/GP to a wide range of problems explains its appeal to modelers. There are not many methods that provide rules and expressions that are hard to anticipate. Examination of the solution expression trees themselves often reveals intuitions that were not apparent prior to the analysis.

GAs are natural approaches to studying social interaction, especially that in the markets. Riechmann (1999) shows how efficient GAs can be in depicting evolutionary processes. The three operations in each iteration of the GA result in both, "variety generation" and "variety restriction," which makes it very useful for searching large spaces, as the algorithm, while narrowing in on an optimal solution, is simultaneously thinking outside the box as well.

In finance, the main applications have been in the areas of trading rule generation, estimation of econometric systems, and classification of firms.

4 Applications

4.1 Trading

Allen and Karjalainen (1999) applied a GP algorithm to the S&P index to determine when to be in the index and when to stay out of it. The rules they uncovered stipulated that investment in the index was called for when equity market returns are positive and market volatility is low. When the opposite is true, the rule stipulates that one stay out of the market. This rule outperforms a buy and hold strategy, but not significantly after accounting for transaction costs. However, this outcome does imply that the GP was able to develop rules with some predictability.

Neely (1999) revisited the work of Allen and Karjalainen by looking at the results on a risk-adjusted basis, and found that the GP did not find rules that surpassed buy and hold strategies after risk adjustment. The results in his paper are supportive of market efficiency. He proposed that all analyses of GAs should be undertaken on a risk-adjusted basis.

In contrast, Rodriguez *et al.* (2001) found that, even after risk adjustment, GAs outperformed buy and hold strategies in the Madrid stock market.

One would expect GAs to perform especially well in the domain of dynamic trading rules (versus static ones) since the rule space is extensive. Glaffig (2004) investigates the use of GAs in hedge fund investing. He uses a mixed GA cum neural net algorithm to develop dynamic trading rules, uncover the factors that determine the performance of these strategies, and also to propose risk management measures for hedge funds. The approach is coined FSM, for "fuzzy strategy mapping," which takes past hedge fund and market data to develop dynamic trading rules. The FSM was trained to track the CSFB Tremont hedge fund index, and also a long/short

fund. Out of sample prediction of next month's returns ranged from 65 to 85%, indicating good performance for the GA.

In an interesting paper, Linn and Tay (2001) posit that investors faced with an overwhelming quantum of information use heuristics that may be mimicked by GAs. They develop such rules using a fuzzy genetic classifier (based on a GA), and then examine how such rules, if employed by investors, might impact the process of price formation. By generating returns from artificial markets driven by GA-based trading rules, nonlinear price effects are found that are similar to that of the data from real markets. The authors thus offer a behavioral technology that may explain some of the observed price formation in markets.

4.2 Econometrics

As is apparent from the GA approach, estimation undertaken using these methods falls in the semi-parametric to non-parametric category. Czarnitzki and Doherr (2002) present a GA that is used to estimate a censored regression model. This is compared to a standard technique for estimating censored regressions, and is found to return the same estimates. However, it is shown that the GA is more stable. This application illustrates that GAs are good devices when the criterion function to be optimized is nondifferentiable in some regions. In a similar vein, Kanungo (2004) presents a GA application in the realm of logit models.

Even GAs are not always successful in converging to the global optimal, especially when many local optima exist. Pictet *et al.* (1995) analyze when standard GAs fail and propose a sharing approach across populations. If there are several local optima, they show that an approach that addresses this problem is to maintain several separate populations, not just one. In this way, diverse populations will focus on separate local optima. Only at the end of the procedure might we compare across populations.

5 Summary

In addition to the more common applications described above, a nascent area of application is that of game theory. This is a natural development. Since GAs may be used to mimic the behavior of differentially learning agents (i.e., humans), they may also be used to examine equilibria amongst these interacting agents.

Noe and Pi (1998) provide an instance of one such application in a takeover situation. They investigate the problems of free-riding and coordination failure. Simulations from a GA correspond to the theoretical predictions from the Nash hypothesis. The deviations from the Nash outcome are explained by the mimicking by the GA of experimental biases with human subjects.

As the field of behavioral finance grows, we may see more use of GAs as the medium of investigation of equilibria with interacting biases across heterogeneous agents. GAs may also be used to examine the price formation process in more detail in microstructure models. We appear to be at a cusp point—the next few years will determine if this technology will encounter widespread use, or be relegated to the graveyard of fads in social science.

References

Allen, F. and Karjalainen, R. (1999). "Using Genetic Algorithms to find Technical Trading Rules." *Journal of Financial Economics* **51**, 245–271.

Bauer, R. (1994). *Genetic Algorithms and Investment Strategies*. New York: John Wiley.

Czarnitzki, D. and Doherr, T. (2002). "Genetic Algorithms: A Tool for Optimization in Econometrics—Basic Concept and an Example for Empirical Applications." Katholic University Leuven and Center for European

Economic Research (ZEW), ZEW Discussion Paper No. 02-41.

Glaffig, C. (2004). "Risk Management of Hedge Funds using Fuzzy Neural- and Genetic Algorithms." Mimeo, Panathea Capital Partners.

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

Kanungo, R. (2004). "Genetic Algorithms: Genesis of Stock Evaluation." Asian Accounting, Finance and Business Research Unit, Cardiff Business School, Economics WPA Working Paper No. 0404007.

Linn, S. and Tay, N. (2001). "Fuzzy Inductive Reasoning and Nonlinear Dependence in Security Returns: Results from an Artificial Stock Market Environment." Working Paper, University of Oklahoma.

Neely, C. (1999). "Using Genetic Algorithms to Find Technical Trading Rules: A Comment on Risk Adjustment." Federal Reserve Bank of St. Louis Working Paper No. 99-015A.

Noe, T. and Pi, L. (1998). "Genetic Algorithms, Learning, and the Dynamics of Corporate Takeovers." Working Paper, Tulane University.

Pictet, O., Dacorogna, M., Chopard, B., Oussaidene, M., Schirru, R., and Tomassini, M. (1995). "Using Genetic Algorithms for Robust Optimization in Financial Applications." Mimeo, Research Institute for Applied Economics, Zurich.

Riechmann, T. (1999). "Learning and Behavioral Stability: An Economic Interpretation of Genetic Algorithms." *Journal of Evolutionary Economics* **9**(2).

Rodriguez, F., Martel, C. and Rivero, S. (2001). "Optimisation of Technical Rules by Genetic Algorithms: Evidence from the Madrid Stock Market." Universidad de Las Palmas de Gran Canaria—Faculty of Economic Science and Foundation for Applied Economic Research (FEDEA), FEDEA Working Paper No. 2001-14.