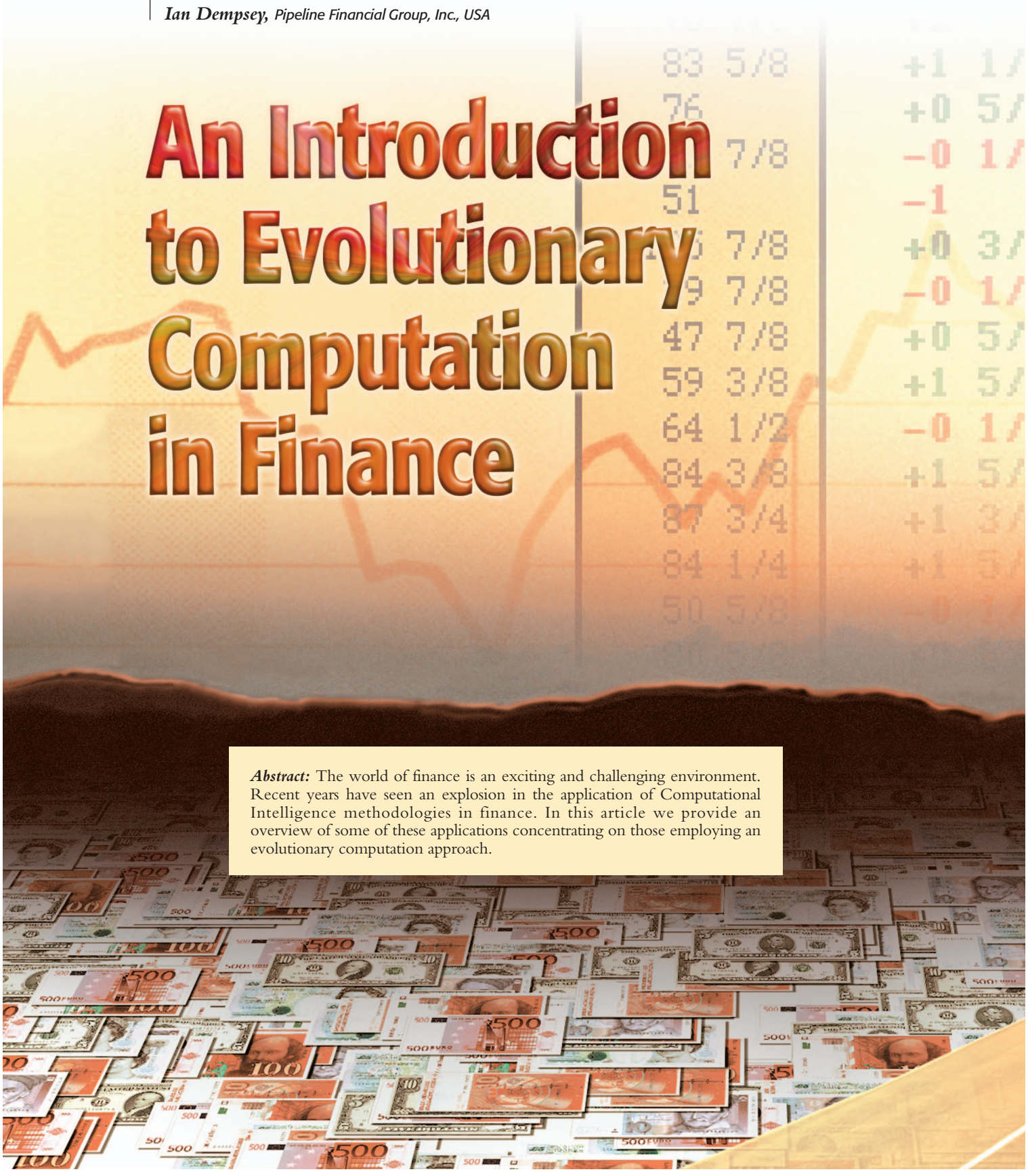
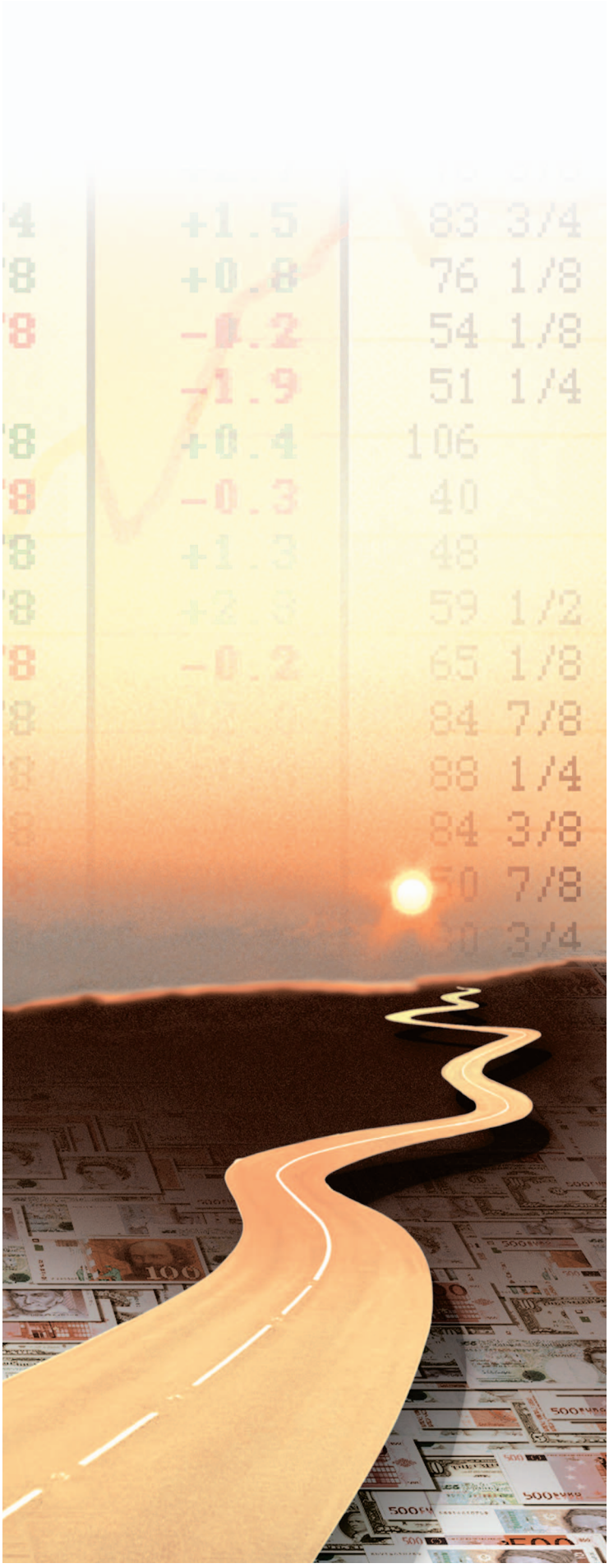
Anthony Brabazon, Michael O'Neill,
University College Dublin, IRELAND
Ian Dempsey, Pipeline Financial Group, Inc., USA

An Introduction to Evolutionary Computation in Finance



Abstract: The world of finance is an exciting and challenging environment. Recent years have seen an explosion in the application of Computational Intelligence methodologies in finance. In this article we provide an overview of some of these applications concentrating on those employing an evolutionary computation approach.

1556-603X/08/$25.00©2008IEEE



©ARTVILLE

I. Introduction

Applications of Computational Intelligence are expanding rapidly in the financial world. In the area of algorithmic trading alone, industry estimates by the Aite Group predict that by 2010, 50% of U.S., 28% of European and 16% of Asian order flow will be executed automatically via trading algorithms [1]. With about 8.5 billion shares currently being traded daily in the US this would equate to the automatic trading of $120 billion of stock in current money terms.

The modeling and trading of financial markets is challenging for several reasons. Many factors plausibly impact on markets including interest rates, exchange rates, the rate of economic growth, and notably in recent times, liquidity. We lack a hard theory as to how all these (and other) factors effect the prices of financial assets, partly because the effects can be non-linear, time-lagged, and non-stationary. Other aspects of financial markets which make them challenging for modelers include: the lack of multiple market histories with which to test our theoretical models; the emergent nature of markets; and the inherently unpredictability of some factors that can impact on markets such as natural disasters. Despite these difficulties there is an extensive appetite amongst market participants for new computational approaches. One family of computational algorithms that have attracted significant interest in recent years are Natural Computing algorithms.

A. Natural Computing

Natural computing can be broadly defined as the development of computer programs and computational algorithms using metaphorical inspiration from systems and phenomena that occur in the natural world. The inspiration for natural computing methodologies typically stem from real-world phenomena which exist in high-dimensional, dynamic environments, characteristics which fit well with the nature of financial markets. Prima facie, this makes natural computing methods interesting for financial applications.

Natural Computing algorithms can be clustered into different groups depending on the aspects of the natural world upon which they are based. The main clusters are Neurocomputing, Evolutionary Computing, Social Computing, Immunocomputing, Physical Computing, and Developmental & Grammatical Computing (see Figure 1).

Neurocomputing algorithms typically draw inspiration from simplified models of the workings of the human brain/nervous system. The predominant neurocomputing paradigms include Multi-layer Perceptrons, Self Organising Maps, Radial Basis Function Networks, and Adaptive Resonance Theory. Artificial Neural Networks (ANNs) can be used to construct models for the purposes of prediction, classification and clustering, and are non-parametric modeling tools as the model is developed directly from the data.

Digital Object Identifier 10.1109/MCI.2008.929841

Evolutionary Computation is based upon neo-Darwinian principles of evolution. It is a population-based approach to problem-solving where multiple candidate solutions are maintained in parallel. Genetic search operators are applied to breed high-quality solutions as subsequent generations are created using fitness-based selection. The fitness of a candidate solution is a measure of its quality at solving the problem at hand.

Social Computing Popular variants of these algorithms adopt a Swarm metaphor, and include algorithms inspired by the flocking and schooling behavior of birds and fish, and the behaviors observed in social insects such as ants. These social systems exhibit a number of characteristics which facilitate problem-solving including, self-organization, flexibility, robustness, and direct/indirect communication between members of the population. These algorithms are population-based like their Evolutionary Computation counterparts, and they operate by allowing the population of problem-solvers to communicate their relative success in solving the problem to each other.

Immunocomputing includes a family of algorithms which turn to the complex and adaptive biological immune system of vertebrates to inspire their design. The natural immune system represents an intricate network of specialized chemicals, cells, tissues and organs with the ability to recognize, destroy and remember an almost unlimited number of foreign bodies, and to protect the organism from misbehaving cells in the body. These properties are especially useful for tasks such as classification and optimization. Practical applications of immunocomputing include financial pattern-recognition such as the identification of potentially fraudulent credit card transactions, the identification of financially at-risk companies and the identification of market 'state'.

Physical Computing draws inspiration from the physical processes of the natural world to design computational algorithms. These algorithms draw inspiration from phenomena such as Simulated Annealing and Quantum Mechanics.

Developmental & Grammatical Computing are a family of Natural Computing algorithms which borrow from both a developmental and a grammar metaphor. Grammatical Computing refers to algorithms which adopt concepts from linguistic grammars and are dominated by the generative form of grammars. Generative grammars are used to construct a sentence(s) in the language specified by the grammar, and this generative process is metaphorically similar to the developmental process in biology which produces a complex, multi-cellular organism from a single embryonic cell. Generative grammars have been used in Natural Computing as a convenient representation by which Developmental systems can be realized in-silico.

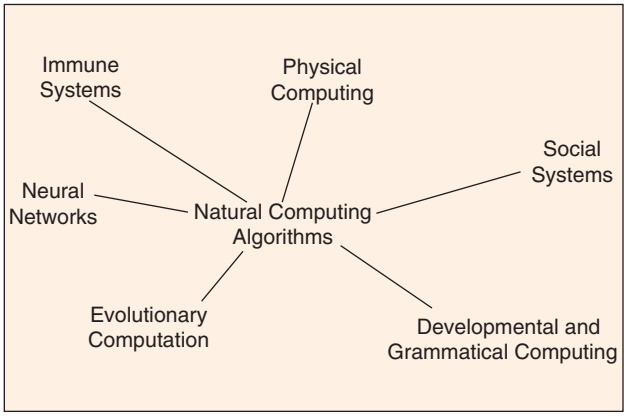


FIGURE 1 An overview of natural computing algorithms.

Most implementations of Developmental & Grammatical Computing, including Genetic Programming and its Grammatical variants, have also embedded an Evolutionary Algorithm (which is typically used to drive the search process). These algorithms have enjoyed notable success on real-world problems, with the most inspiring examples including John Koza et al's [42] use of a developmental form of Genetic Programming to routinely design analog circuits that outperform those designed by human experts. Some of these circuits have been patentable inventions in their own right. Another noteworthy application of Genetic Programming is Lohn et al's [46] development of an Antenna for the NASA Space Technology 5 mission.

A growing community of researchers are engaged in the application of Natural Computing, and in particular Evolutionary Computation, methodologies in Finance as illustrated by the number of Conferences, Workshops and Special Sessions in this area. Examples of these include the annual track on Evolutionary Computation in Finance and Economics at the IEEE Congress on Evolutionary Computation, the IEEE Symposium on Computational Intelligence for Financial Engineering (CIFEr), the annual international Conference on Computational Intelligence in Economics & Finance (CIEF), and EvoFIN the European Workshop on Evolutionary Computation in Finance held annually as part of Evo*.

II. Evolutionary Computation

Since Charles Darwin popularized the theory of Natural Selection, the driving force behind evolution, molecular biology has unraveled some of the mysteries of the components that underpin these earlier theories (e.g., most notably the existence and structure of DNA). Neo-Darwinism, which represents the accumulation of knowledge about the process of evolution at a molecular level in conjunction with Darwin's evolutionary theory, has given rise to a powerful family of problem solving algorithms known collectively as Evolutionary Computation (EC).

The origin of EC can be traced back at least to the beginning of Computer Science [41] with the writings of Turing [75] where he discusses the possibility of "*genetical or evolutionary search*" as one of the methods that might underpin the "*the intelligence of machinery*". Implementations of EC existed in the 1950s with the Pioneers of EC popularizing their ideas in subsequent decades including Holland [35], Fogel, Owens and Walsh [26], Rechenberg [64], Schwefel [67], Goldberg [28], De Jong [19] and Koza [39]. Fogel [25] presents an interesting collection of some of the pioneering papers in the field.

The natural process of evolution sees species being positively or negatively selected depending on their relative success in

surviving and reproducing in their current environment. Differential survival and variety generation during reproduction provide the engine for evolution [17], [69]. EC implements this process in a crude fashion as outlined in Figure 2.

In EC, the equivalent of DNA is the representation or encoding that is adopted by each member of the population (often referred to as an *individual* or *candidate solution*) to solve a specific problem type. This individual representation can range from simple fixed-length binary strings or reals to more complex data structures such as graphs and even variable-length computer code that can be executed. For example, a simple binary string could be used to encode decisions as to whether or not to include an input variable in a solution. The selected inputs might then be passed into a multi-layer perceptron ANN (MLP) to produce a signal such as to buy or sell a position in a market. We will now introduce Genetic Programming which provides a more complex representation that can allow model structures to be co-evolved with parameters.

A. Genetic Programming

A particularly powerful form of EC, in terms of the flexibility of its representation and its prowess as a model-induction method, is Genetic Programming. Genetic Programming (GP) traditionally distinguishes itself from other EAs in two fundamental ways. Instead of evolving fixed-length (e.g., binary) strings a variable-length representation is adopted by GP. As GP is a model induction method it is not always known a-priori what the structure or size of a desired solution might be, and to this end the number of components that make up a candidate solution must itself be open to the process of evolutionary search. Secondly, unlike other EAs which represent an indirect encoding of a potential solution, evolutionary search can be directly applied to the solution (e.g., computer code in the form of a Lisp S-expression).

An example of a GP individual in the form of a Lisp S-expression is given in Figure 3. Here we can also observe how individuals of this form can be manipulation using the genetic search operator of sub-tree crossover. Sub-tree mutation operates by randomly deleting a sub-tree and then randomly generating a brand new sub-tree of a random size to replace the original. Using this representation it is possible to incorporate various programming constructs such as conditions, iterations, recursion and modules/functions. It is also common to allow the inclusion and context of use of these constructs to be evolved over time using what are referred to as *Architecture Altering Operators* [41].

During the creation and manipulation of these tree-based individuals during a GP algorithm requires paying careful attention to syntactical issues. In other words, valid trees must be produced after any operation. This property, known as *closure*, is ensured through the design of search operators which respect the tree structure maintaining syntactically legal solutions. A particularly useful extension of the standard approach to GP is the adoption of explicit grammars to guide the creation and maintenance of GP individuals. While some of these *grammatical* approaches to GP have been in existence from the early days of GP (e.g., [4], [31]) they are enjoying increased popularity in recent years partly due to the flexibility that they afford the practitioner in allowing domain knowledge to be incorporated to guide the search process through a grammar specification, and the ease with which this grammar specification can be modified to change the structure of the output solutions (e.g., [27], [29], [30], [32], [37], [38], [60], [63], [65], [68], [76], [77]). There are many useful resources which can be used to discover more about Genetic Programming for the interested reader, including [7], [39]–[42], [61].

B. Grammatical Evolution

Grammatical Evolution (GE) is one of the most popular grammatical approaches to GP [20], [54]–[57], [66] and has been adopted for financial modeling (e.g., [10], [20]).[1] In addition to the notion of explicit grammars, GE borrows additional principles from molecular biology. The most powerful of these is the genotype-phenotype map. Earlier research in GP has shown some of the potential benefits of such a mapping [6], [38] and GE further exploits this representation to create a highly-modular and flexible approach to program/model induction. An example of this is the fact that alternative search engines can be adopted to explore the model space (e.g., Particle Swarm Optimization and Differential Evolution have been adopted [58], [59]). The flexibility of GE is such that even with the presence of the genotype-phenotype map, traditional tree-based search operators of crossover and mutation can be adopted in place of the genotype search operators effectively transforming GE into a standard form of GP with the grammar used during the initialization of the population [30]. Of course, it is also entirely possible to combine search operators that are focused on both the genotype and phenotype combining the benefits of each approach.

C. GP and GE in Financial Modeling

One particularly interesting aspect of genetic programming and grammatical evolution is that both the solution form and associated parameters are co-evolved during the evolutionary process. This offers particular utility in financial modeling, as

[1]Further information on GE and pointers to code can be found at http://www.grammatical-evolution.org.

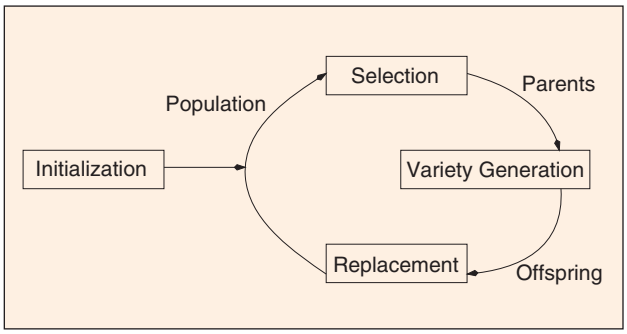


FIGURE 2 The process cycle of evolutionary computation.

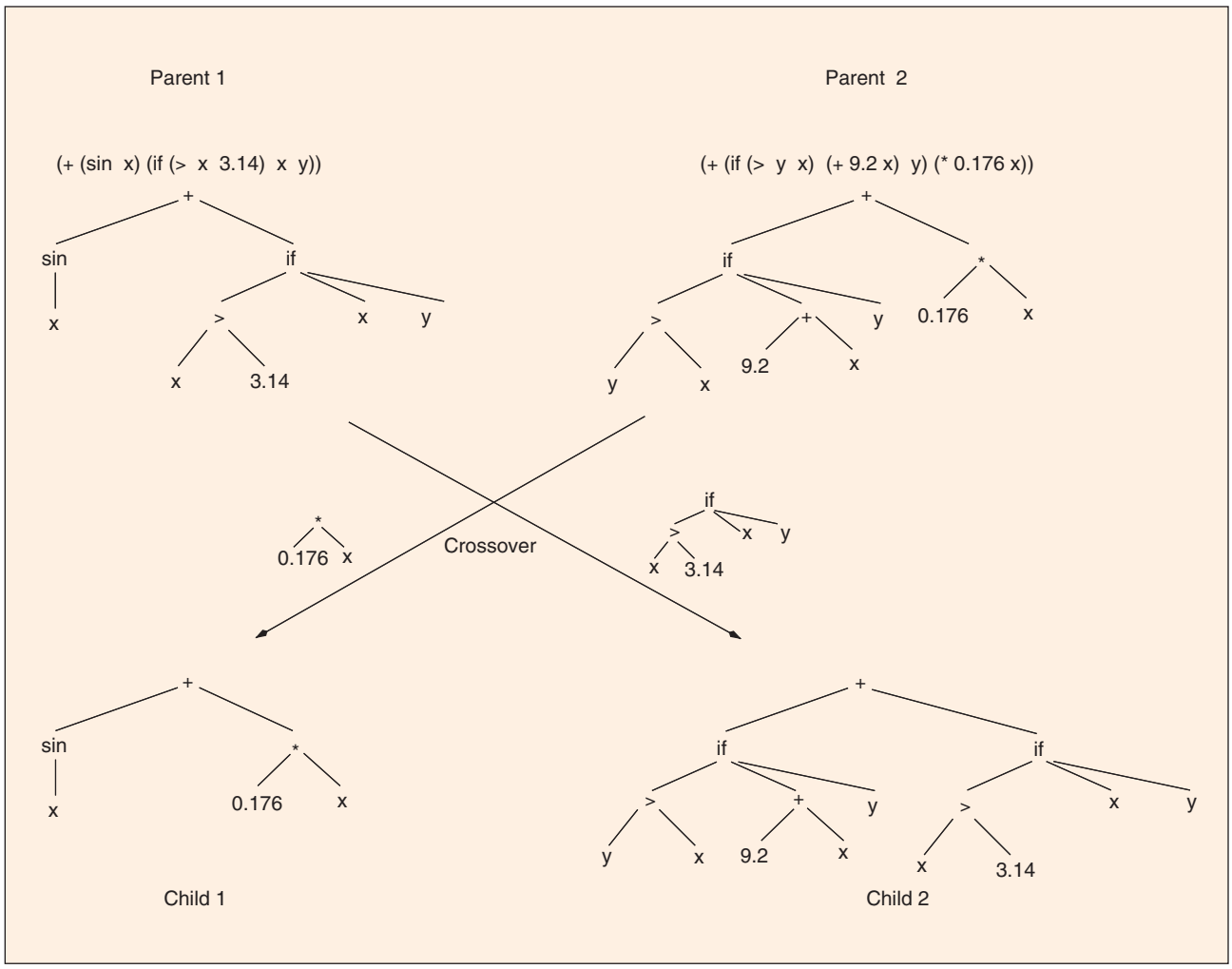


FIGURE 3 Individuals of a genetic programming population are typically represented as Syntax Trees of Lisp S-expressions. The figure illustrates how subtrees of each individual are exchanged during the process of crossover.

the environment is often data rich but theory poor. Typically, while many plausible explanatory variables exist but the interrelationship among the relevant variables is poorly understood, although some domain knowledge may exist. This suggests that model induction methodologies will have particular utility. Unlike black-box methodologies such as neural networks, GP and GE offer the potential to generate human-readable rules. This is of particular importance in (for example) financial trading systems where human decision makers will want to have insight into the trading rationale. Another advantage of GP and GE is that they permit the incorporation of domain knowledge, and the generation of 'solutions' of a particular form. This allows the financial user to (for example) seed the evolutionary process with their current trading strategies, in order to see what improvements the evolutionary process can uncover. More generally, all evolutionary algorithms allow the incorporation of complex fitness functions, which is of particular importance in finance as fitness is generally a complex amalgam of return and risk. Recent years have also seen an explosion in the quantity and quality of electronic financial information available, hence, the practicality of applying evolutionary methodologies in finance has increased.

D. Applications of EC in Finance

Applications of EC in Finance can be broadly categorized as being either Optimization or Model Induction (an extensive literature on agent-based modeling in finance also exists, but is not discussed in this paper).

1) Optimization

Optimization applications, such as portfolio selection, abound in finance. Often, because the underlying financial problem is high dimensional and complex, the only practical optimization methodologies are general purpose heuristics such as genetic algorithms, evolutionary strategies, differential evolution, and particle swarm optimization.

2) Model Induction

While optimization applications in finance are important, often the underlying model or data generating process is not known.

Hence, the task is often to 'recover' or discover an underlying model from a dataset. This is usually a difficult task as both the model structure and associated parameters must be uncovered.

Evolutionary approaches have been widely applied in finance since the late 1980s. Initially, attention was primarily focused on the application of GAs for model parameter optimization and variable selection [8] but from the mid 1990s increasing attention has been placed on the use of GP for financial forecasting and trading system induction. The sophistication of GP applications in these areas has increased markedly in recent years. GP has also seen increasing application to other areas in finance, such as derivatives pricing and volatility prediction.

In the following sections we briefly introduce a range of financial applications to which EC methodologies have been applied, including financial forecasting, credit risk assessment, portfolio optimization, asset pricing, and algorithmic trading. An excellent starting point in exploring the range of EC applications in finance can be found in [11]–[15].

III. Financial Forecasting

Financial markets are affected by a myriad of interacting economic, political and social events. The relationship between these factors and financial asset prices is not well understood and, moreover, is not stationary over time. Most theoretical financial asset pricing models are based on strong assumptions which are often not met in real-world asset markets. This offers opportunities for the application of model induction methodologies in order to 'recover' the underlying data generating models.

The 1980s and early 1990s saw a plethora of studies applying (initially) neural networks and (later) GA/GP for forecasting of financial time series. Typically, these studies used measures of goodness of fit drawn from statistics such as Mean Squared Error (MSE), Sum of Squared Error (SSE), Mean Absolute Percentage Error (MAPE) etc. as their fitness function, the object being to uncover or train a model using historical data, which 'fit' that data well. Unsurprisingly, the choice of fitness function usually has a critical impact on the behavior of resulting model, hence a model constructed using one fitness metric would not necessarily perform well on another. While these studies did indicate that models could be constructed that fit historic data fairly well, it was less clear as to whether these models could be used for trading purposes. Another common finding was that the quality of the forecasts diminished over time, as the time period since model training increased. Developments in later papers included the use of more sophisticated methods for pre-processing the raw time-series inputs and the use of sliding-window retraining in order to update the forecasting models.

IV. Credit Risk Assessment

Credit risk assessment is an important component of the lending process. Examples of decisions where credit scoring could be useful include, should a loan of $X be extended to a firm, should a customer be allowed to purchase goods on credit, or what credit limit should be offered to a customer on their credit card?

In all of these applications, the object is to develop a model which will provide a metric of credit-worthiness from a series of explanatory variables. Typically, in assessing (for example) corporate credit-worthiness, explanatory variables can include data drawn from the financial statements of the firm, data drawn from financial markets (such as share price), general macro-economic data, and non-financial, firm-specific information. In assessing personal consumer credit-worthiness explanatory variables can include income, age, occupation, current employment status, past borrowing record etc.

A practical problem in constructing risk-assessment models is that there is no clear theoretical framework for guiding the choice of explanatory variables or model form. In the absence of an underlying theory, most published work on credit rating employs a data-inductive modeling approach. This produces a high-dimensional combinatorial problem, as the modeler is attempting to uncover a good set of explanatory variables and model form.

An illustration of an early credit risk assessment model is provided by Altman's [3] classic study in which five ratios were selected and then combined to produce a linear discriminant classification model for corporate bankruptcy. A Z score was calculated for each company and this value determined whether the company was classified as likely to go bankrupt or likely to remain solvent. The original Altman classifier had the form:

$$Z = 0.012X_1 + 0.014X_2 + 0.033X_3 + 0.006X_4 + 0.999X_5,$$

where:

X_1 = working capital to total assets
X_2 = retained earnings to total assets
X_3 = earnings before interest and taxes to total assets
X_4 = market value of equity to book value of total debt
X_5 = sales to total assets

As the range of computational intelligence techniques for classification have expanded, each new technique has been applied in turn to credit scoring and corporate failure prediction. The domain offers particular potential for evolutionary automatic programming methodologies such as genetic programming or grammatical evolution as these methods can produce human-readable credit decision rules. This can be important in some countries as lenders may be required to justify decisions not to grant loans. Another advantage of GP and GE is that the rule-evolution process can be seeded using existing domain knowledge.

An interesting feature of the problem is that there are quite differing costs associated with type 1 and type 2 classification errors. Failing to grant credit to a good customer costs the lender the lost interest (or 'profit') margin, whereas lending to

a customer who does not repay the loan loses the entire capital sum. Hence, it is common when developing credit risk models to recognize that the costs of misclassification are asymmetric. Another feature of the problem when training classifiers is that of unbalanced databases. The rate of loan non-repayment or bankruptcy is usually quite low, hence historical databases of customer characteristics and loan outcome typically contain relatively few examples of credit failure.

A closely-related application is the prediction of bank failure [44], with many regulatory authorities using risk models in order to assess which financial institutions require the closest scrutiny. Obviously, for these applications it is important that the regulatory authority can verify the correctness of the underlying prediction model, hence methodologies which can incorporate expert knowledge and produce interpretable decision rules, such as fuzzy systems and GP, are of particular interest.

V. Portfolio Optimization

One of the classic (multi-objective) optimization problems in finance is that of portfolio optimization, where the object is to invest a fixed amount of money in a diverse set of assets (a portfolio) in order to maximize return while minimizing a risk measure. The solution to this is a Pareto front (see Figure 4). Once the frontier is uncovered, the final choice of portfolio is determined by the investor's risk preference.

The classic Markowitz mean-variance model [49], [50], which underpinned the original formulation of the portfolio selection problem, assumes that investors wish to maximize their return (typically measured as the expected return of their portfolio) and minimize their risk (typically measured as variance or the standard deviation of their portfolio's return). This produces a risk return trade-off between a linear return measure and a single convex, non-linear, risk measure. The Pareto efficient frontier is found by varying the risk target and maximizing on the return measure.

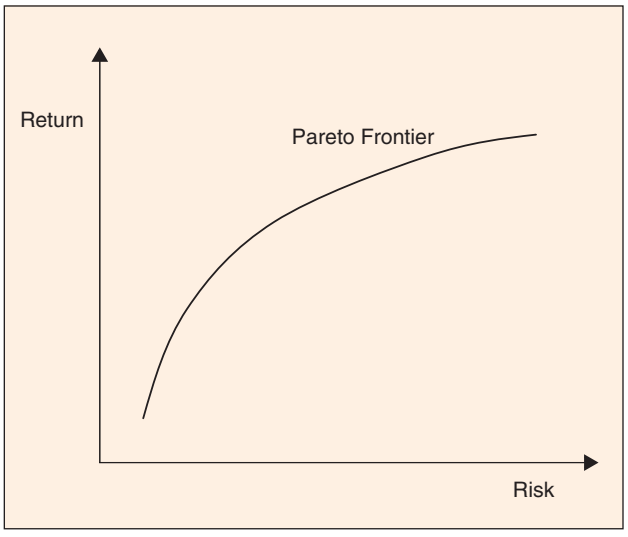


FIGURE 4 Pareto frontier. The points that correspond to the risk-return of the set of portfolios that are Pareto optimal.

Hence, the goal of the Markowitz model is to find an optimal portfolio p of $i = 1, \ldots, N$ assets, each with a weighting w_i such that the return μ_p is maximized while minimizing the variance of the return σ_p. This formulation produces a quadratic programming problem and several algorithms exist to tackle this.

However, this formulation ignores a host of constraints that impact on real-world portfolio optimization including, *cardinality constraints* (a limit on the number of assets which can be held in the portfolio), or *threshold limits* on the amount of investment in any single asset, $w_i \leq m_i;\ i = 1, \ldots, N$, where m_i is the threshold amount for asset i. Other constraints may include industry sector (or *concentration*) constraints, and round lot constraints. These constraints can lead to non-convex, non-differential models. In addition, some constraints may be hard and others may be soft. Hence, real-world portfolio optimization can present a difficult, high-dimensional, constrained optimization problem, which is beyond the capabilities of traditional optimization methods. In this setting, heuristic methods such as EC are of particular interest. A rich literature has emerged applying evolutionary algorithms for portfolio optimization (see [5], [47], [70]).

An approach which is gaining ground for portfolio optimization is stochastic programming. Stochastic programming (SP) is designed to assist decision-making when there is uncertainty as to future events (such as asset returns). In traditional portfolio optimization there is a strong assumption that past asset returns and their variability, are an accurate guide to future returns and variability. Hence, historical estimates of these items are used in the optimization process. In contrast, Stochastic Programming recognizes that future outcomes may be better approximated using probability distributions. A common approach in stochastic programming is to reduce uncertainty as to future events to a number of 'scenarios' and the aim is to uncover a solution which performs well under all these scenarios. Hochreiter [34] introduces an evolutionary stochastic portfolio optimization methodology and illustrates its application using a set of structurally different risk measures, which include, Standard Deviation, Mean-absolute Downside Semi Deviation, Value-at-Risk, and Expected Shortfall. Recent work has also seen the application of co-evolutionary MOEAs for portfolio optimization [21].

VI. Pricing Complex Financial Instruments

The range of assets traded on financial markets has expanded enormously over the past twenty years, moving far beyond the trading of shares and debt instruments to encompass *financial derivatives*. A financial derivative can be defined as a financial instrument (for example, a contract), the value of which is based on the value or values of one or more underlying assets. Derivatives can be based on the value of equities, debt, market indices, currencies, commodity prices etc. Two of the best known forms of derivative are *futures* and *options*. A future is a contract to buy or sell a specific quantity of an asset, at a specified price, at a specified time in the future, whereas an option

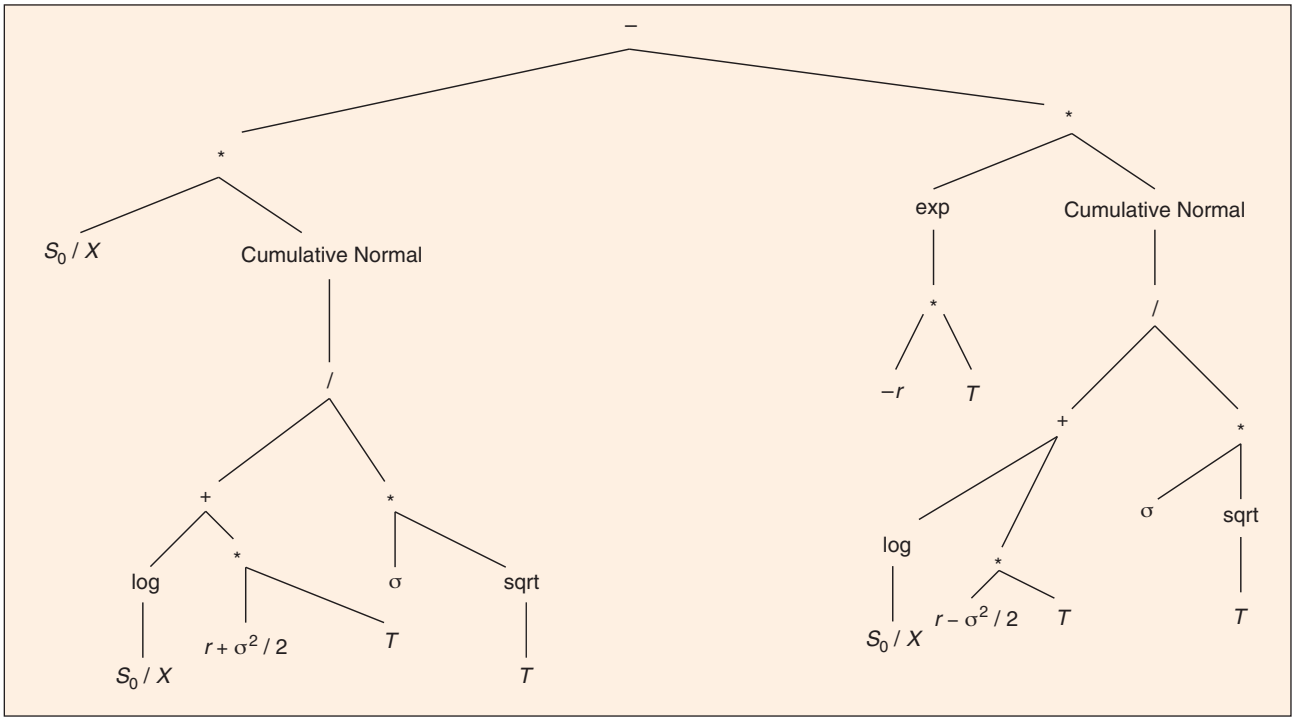


FIGURE 5 A stylized tree representation (some branches are compressed) of the Black Scholes model for pricing a non-dividend paying European call option. There are five factors that affect the price of the option, S_0: the underlying asset price, X: the exercise price of the option, T: the time to maturity (of the option), r: the risk free rate of return and σ: the expected volatility of the asset price.

gives the buyer the right (but not the obligation) to buy/sell a financial asset at a specified price, on or between specified future dates.

The key issue for investors wishing to trade in derivatives is the determination of the fair price for the derivative. For some standard derivatives, closed-form pricing equations have been determined, for example, the Nobel prize-winning Black-Scholes model [9], [52] for pricing European options (Figure 5 illustrates a tree representation of the Black Scholes model).

In reality, some of the key assumptions in the Black-Scholes model do not hold in real-world option markets, and hence the model does not explain observed option prices correctly. Due to the complexity in developing closed form theoretical models for options pricing, the domain is particularly amenable to model-induction techniques such as GP. The payoff to the development of high-quality option pricing models the potential identification of pricing gaps between the theoretical fair price of an option and its current market price. Any such gaps would produce trading opportunities.

VII. Trading

Financial trading has seen a large number of applications of EC. Typically these studies take one of two approaches, using either *fundamental data* or *price/volume data* (technical analysis).

A. Fundamental Investment

Taking the example of investing in a share, fundamental investment concentrates on the use of accounting information about the company, as well as industry and macroeconomic data, in order to identify companies which are mispriced by the market. In other words, the object is to identify shares which are good value (underpriced by the market), or shares which are overpriced by the market (and therefore are candidates for 'shorting'). In this approach, the investor needs to develop stock screening rules in order to decide which shares in which to invest. These rules were formulated manually in decades before computers. With a natural computing algorithm such as the GA, a large range of stock filter rules can be searched efficiently in order to find the highest-quality rules. In this approach, each individual in the population corresponds to a potential stock filter rule. The utility of these rules are tested using historical data, with the best rule (or set of rules) then being used for investment purposes (Figure 6). More generally, GP methods can be applied to evolve the structure of the filter rules.

B. Technical Analysis

In contrast to investors using a fundamental investment approach, technical analysts attempt to identify imbalances in the supply and demand for a financial asset using information from the time-series of the asset's trading price and volume. Usually, investors who adopt a technical analysis approach look to combine technical indicators (pre-processed price and volume time series data about a financial asset), in order to produce a 'trading signal'. For example, one 'technical indicator' that technical analysts could consider is the *moving average*

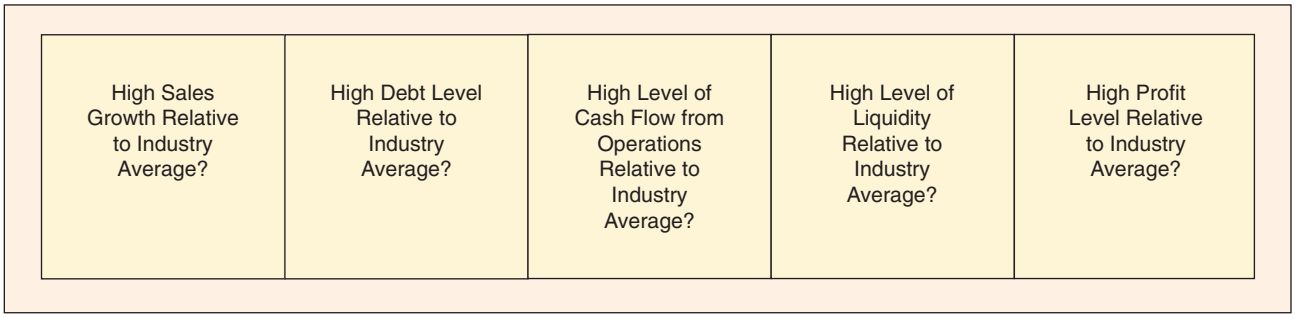


FIGURE 6 String encoding of a number of fundamental indicators. Each indicator can be coded as a 0 (no) or 1 (yes).

convergence-divergence (MACD) oscillator, calculated by taking the difference of a short-run and a long-run moving average. If the difference is positive, it is taken as a signal that the market is trending upward. For example a buy signal could be generated when the shorter moving average crosses the longer moving average in an upward direction. A sell signal could be generated in a reverse case. Therefore, a sample MACD trading rule could be:

$$IF\ x\text{-day MA of price} \geq y\text{-day MA of price}$$
$$\text{THEN Go Long ELSE Go Short}$$

where $x < y$ (for example $x = 10$ and $y = 50$). The MACD oscillator is a crude band-pass filter, removing both high-frequency price movements and certain low-frequency price movements, depending on the precise moving average lags selected. In essence, the choice of the two lags produces a filter which is sensitive to particular price-change frequencies. In a recursive fashion, more complex combinations of moving averages of values calculated from a MACD oscillator can themselves be used to generate trading rules.

In past decades, the search for apparently useful technical indicators (or combinations of these) was undertaken manually by investors who back tested various indicators on historical financial data. GP allows the automation of this process, with the concurrent vast expansion of the search space which can be feasibly searched.

Of course, as financial markets comprise a dynamic system, the utility of any static trading system can be expected to degrade over time. One basic way of examining the characteristics of a trading system is to use an equity curve (Figure 7).

While a plentiful literature exists on the application of EC approaches to design simple trading systems, many of these applications fall a long way short of presenting the true complexities in trading financial markets, particularly when attempting to trade large blocks of financial assets. The next section introduces *algorithmic trading* and briefly describes some of these real-world complexities.

Sentiment Analysis

Traditionally, stock trading models incorporated quantitative data, drawn from the market, financial statements or macroeconomic data. One source of data which has attracted increasing attention as an input into trading models in recent years is text data drawn from either Internet message boards or the financial press.

One of the early studies that combined text data with evolutionary methodologies was Thomas and Sycara (2002) [71] which examined whether measures of message volume (as distinct from message content) on internet message boards could be used as an effective predictor of stock price movements. The study used a GP methodology to build trading rules based on the message volume data for a selection of the largest Russell 1,000 stocks.

While initial studies looked at raw message count information, the next step is to consider the content or 'sentiment' of these messages in order to assess whether investors are (un)favorably disposed towards a stock [43], [62].

Using text data in trading models has been made easier by the commercial availability of 'tagged' databases of financial news, for example, the Dow Jones Elementized News Feed which places discrete pieces of news, keywords, timestamps, symbols and other crucial data, into XML-tagged fields for easy parsing and direct embedding into trading programs.

VIII. Algorithmic Trading

Algorithmic trading is defined here as the use of computer programs to assist in the execution of the trading of financial assets. It can encompass systems which decide on certain aspects of the order such as the timing, price, or even the final quantity of the order. Algorithmic trading may be used in any investment strategy, including market making, inter-market spreading, arbitrage, or speculation.

Today trading algorithms are executing an ever increasing number of trades on market centers across the world. In the U.S. their rise has been brought about through a series of technological and regulatory changes. Since 2001 with the move to decimalization of the US equity markets, and the widespread acceptance of electronic market places, the average trade size has declined from 1,200 shares per transaction in 2000 to 300 shares today [53]. This in turn has led to an explosion in the number of trades executed and a narrowing of spreads with

large institutional orders taking longer to execute. As a result, investors wishing to trade large blocks risk tipping their hand to the marketplace. Trading algorithms seek to optimally execute these orders, using the vast amounts of data produced by the market place and submitting appropriately sized smaller orders to various destinations with the aim of achieving best execution. In reaching this goal an entire ecology of different trading algorithms have been designed to perform under different market conditions, with recent innovations (such as Pipeline Financial's Algorithm Switching Engine) intelligently switching between these algorithms depending on current market conditions.

A. Market Micro-Structure

In designing a trading algorithm attention must be paid to the structure of the market being traded and the different ways in which investors can interact with this market.

Whether buying or selling, an order can be placed as either a *market order* (the order is executed at the prevailing market price) or *limit order* (the order is submitted to the destination at a specific price). The trade-off here is that a market order will provide guaranteed execution though there is no control over the price at which that order will be executed. With the limit order, there is a guarantee over price but not over execution. If a limit order is priced away from the prevailing market price it will enter the destination market's book and wait until the orders that were ahead of it are executed in a price/time priority. Thus a certain probability of execution can be arrived at.

Today most market places operate an electronic double auction limit order book. This is a system whereby all the orders to buy and all the orders to sell are displayed. At the top of the book are the orders that are best priced to buy or to sell and the difference between these two best prices is the *spread*. Thus humans or algorithms can use this data as inputs in deciding upon the size and price of their orders. In Table 1 an example of a limit order book is provided. The left-hand side displays all the orders looking to buy with the best priced orders at 132.19 with a cumulative 3300 shares and the best ask or offer priced at 132.20 with 1800 shares. A limit order to buy submitted to the best bid at 132.19 would reside in this book until the 3300 shares ahead of it were first executed. A market order to buy would be executed instantly at 132.20 for a likely maximum of 1800 shares but would have incurred the

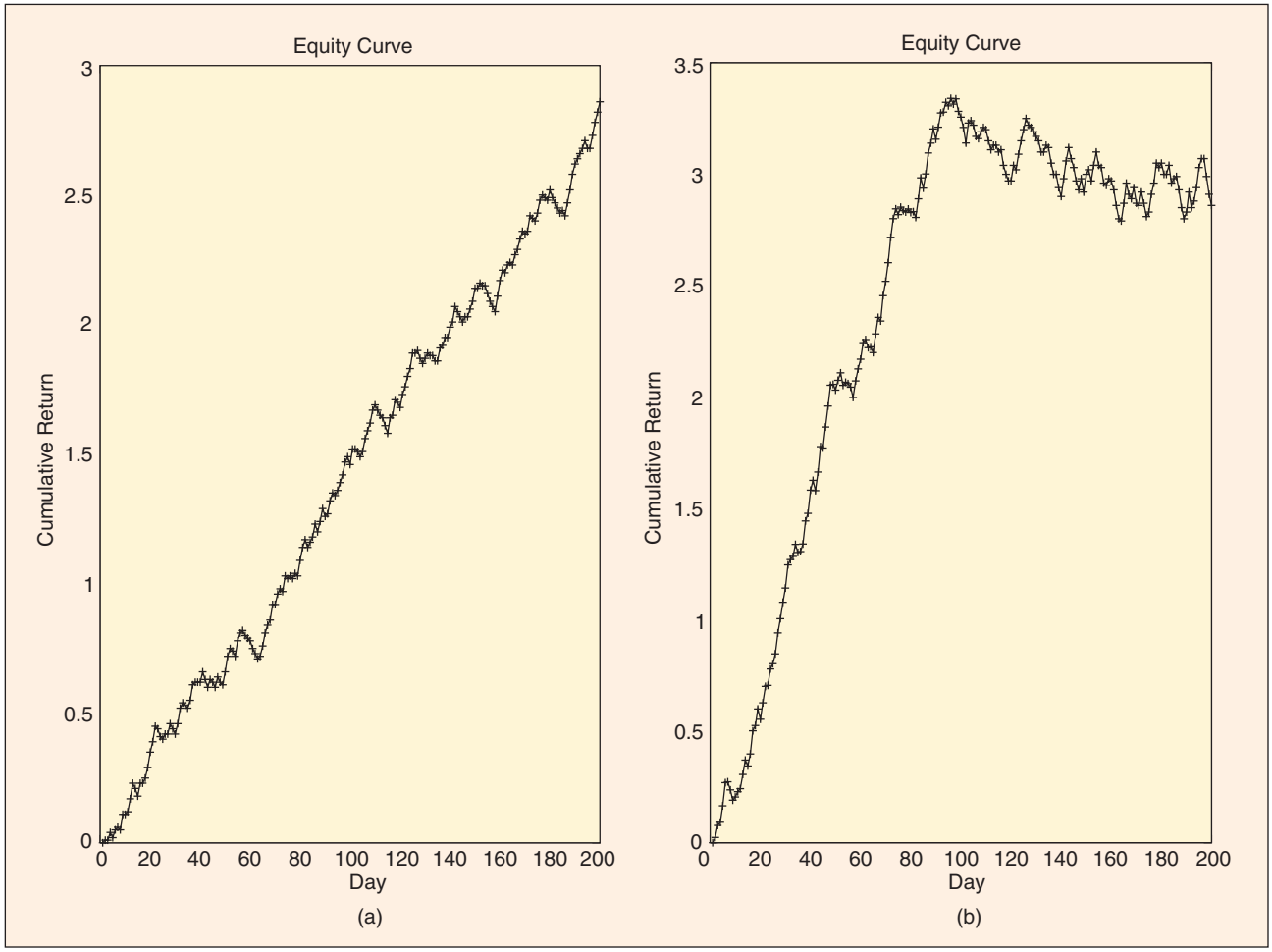


FIGURE 7 Sample equity curves showing cumulative returns on the y axis and time on the x axis. (a) Exhibiting gradual return accumulation. (b) Suggesting that model needs retraining.

Arbitrage

Arbitrage trades seek to make profits by exploiting price differences of identical (or similar) financial instruments, on different markets or in different forms. For example, suppose a share traded on one stock exchange for $23.78 and on another for $23.82, an investor could arbitrage by buying at the lower price and selling simultaneously at the higher price. As would be expected, arbitrage opportunities tend to be closed very quickly and transactions costs of buying and selling can negate apparent arbitrage possibilities.

A simple example of an arbitrage play based on *Put-Call Parity* is illustrated in [74]. In essence, the concept underlying this trade is that the price of a 'long' position on an asset and an associated put (the right to sell that asset in the future at a specified price) must be equal to the price of a long call on the same asset and a long position in a risk-free bond. If either the put or call option are mispriced the investor can, in theory, make a risk-free gain by constructing a portfolio of the four financial instruments. The above example, describes an arbitrage opportunity between the cash market (for the asset) and the options market. More generally, arbitrage opportunities can also exist between cash and futures markets and between futures and options markets.

An alternative approach to waiting for arbitrage opportunities to emerge and then trying to trade them, is to anticipate or forecast opportunities in advance of the actual mispricing occurring. Markose, Tsang and Er [48] illustrate this approach using a GP model to predict arbitrage opportunities between the FTSE-100 index futures and options market up to ten minutes in advance of the arbitrage opportunity arising.

cost of the spread which in this case is 1 penny. Needless to say, limit order books are highly dynamic with multiple parties submitting and canceling orders throughout the trading day.

This process becomes more complicated when the fee/rebate cost structure of the destination market is accounted for and more sophisticated order types are employed. The fee/rebate structure is designed to encourage flow to a destination market by awarding a small rebate to orders that provide liquidity (orders that are posted on the limit order book) and charging a fee for orders that take liquidity (market orders).

In an attempt to provide customers with the ability to hide their order sizes more effectively many market places have implemented *reserve* and *hidden* order types. These order types, used with a limit price, enable an order to partially or completely hide the size of their order on the limit book. Such an order could be submitted to the best bid in Table 1 with a size of 3000 shares but with instructions to only display 100 shares and maintain the rest in reserve. This has the effect of not scaring away the offer by hiding from the market that there may be a large buyer of the stock.

TABLE 1 Sample order book for a stock with volume (measured in 100s) and price information for bid and ask.

BID VOL	PRICE	PRICE	VOL ASK
33	132.19	132.20	18
13	132.18	132.21	21
34	132.17	132.22	24
6	132.16	132.23	14
10	132.14	132.24	9
7	132.13	132.25	1

Stemming from this, a new breed of market place has emerged in the recent past known as *dark pools*. These destinations do not display limit order book information but may provide indications of interest in a stock. When an order arrives at a dark pool, generally no size, price or side information is leaked to the market sealing in much of the information that leads to other parties gaining knowledge of the order. If a contra order arrives to the same dark pool in the same stock at a price that is within the limit of the resident order, the trade can be executed at the midpoint of the spread of the best displayed bid and ask of the traditional destinations. This provides an execution between two natural counter parties where each receives price improvement and incurs no market impact. Trade sizes at dark pools can also be much larger than their displayed counterparts as customers can submit large block orders to dark pools without tipping their hand to the market place. Today over forty dark pools are available in the U.S. from independent broker dealers such as Pipeline Trading Systems, Liquidnet and Posit, to large institutional broker dealers such as Credit-Suisse' Crossfinder and Goldman Sachs Sigma-X and broker dealer consortia in BIDS and Level.

B. Trade Execution

The goal of a trade execution algorithm is essentially to minimize *market impact* when trading large blocks of shares. Market impact occurs as each trade alters the price of a stock. As the market learns of the presence of a large buyer (or seller) market players will begin to game their behavior in order to take advantage of this information. In order to hide this information a large order is usually broken up into many smaller orders and traded over a longer period but in doing so the order is exposed to a greater degree of risk. For more on market impact see [23], [36], [78].

A model, inspired by the Capital Asset Pricing Model, that takes into account value at risk (VAR) for trade execution was first proposed by Almgren and Chriss [2]. Using such a schedule an algorithm will initially trade at a greater speed so as to reduce the VAR of the portfolio position with the rate of trading slowing down later in the day as the value of the smaller position poses less risk. Recently work by Al Janabi [51] has been conducted to include liquidity at risk into trade scheduling that accounts for the probability of liquidity on the other side of the trade being present. However these schedules should be viewed more as guides as they can incur *opportunity cost* because they do not take into account favorable price movements or good opportunities where an algorithm should seek to increase its pace of execution and

adopt a more conservative stance when the price moves away while at the same time maintaining an eye on completing the entire order. Market conditions are prone to change at any moment and given such an environment a trading strategy must be flexible enough to adapt and optimize so as to complete the order at the best prices while minimizing impact to the price.

Despite the importance of optimizing trade execution, there has been relatively little attention paid in the literature to the application of evolutionary methodologies for this task. One exception is Lim and Coggins [45] who used a GA to evolve a trading schedule in order to optimize trade execution performance using order book data from the Australian Stock Exchange. In this study, the approach taken was to initially split each trade into a series of N equal sized orders, and the object was to evolve the timing strategy for the execution of each of these N orders during a single trading day. Each order was submitted as a limit order at the best ask or bid prevailing at the time the order was submitted, depending on whether the investor was seeking to buy or sell shares. A simple traditional trading strategy could be to submit one of these orders every ten minutes. However, there is no guarantee that a ten minute order spacing would produce good results in terms of minimizing market impact. Lim and Coggins [45] used a GA to uncover good quality timings for each order by evolving a chromosome of N genes, where each gene encoded the maximum lifetime that the order would remain on the order book (if it had not already been executed) before it was automatically ticked over the spread (for example, a limit buy order being repriced to the current ask) to close out the trade. Any uncompleted trades at the end of the day were closed out the same way. Hence, the GA evolved the maximum time that each order would be exposed to the market before being crossed over the spread.

C. Algorithm Design

A trading algorithm is made up of a variety of decision making components each of which may be optimized to take into account a multitude of quantitative analytics and prevailing market conditions. When deciding whether or not to submit an order to the market place an algorithm must decide on an order's:

- ❏ Timing—when should the order be placed and/or what interval of time should there be between orders (what is the schedule?)
- ❏ Type—should the order be a market, limit, reserve, hidden order?
- ❏ Sizing—what size order should be sent to the market?
- ❏ Pricing—at what price should the order be, aggressive or passive?
- ❏ Destination—there are many market destinations and types, which one will provide the best conditions of execution for the order?
- ❏ Management—if a limit order has been submitted, how should this order be managed?

A variety of fitness functions could be designed to drive the evolution of the trading strategy but a common metric of trade execution performance is its *Volume Weighted Average Price* (VWAP)

$$\text{VWAP} = \frac{\sum (\text{Price} \cdot \text{Volume})}{\sum (\text{Volume})} \tag{1}$$

The VWAP of a strategy can be calculated and benchmarked against (for example) the overall VWAP for that share during the period of the trading strategy's execution. The aim is to evolve a strategy which produces as competitive a VWAP as possible.

In the example in the previous subsection where a trade execution rule was evolved, the basic structure of the execution rule is determined in advance (number of trades etc.) and the task of GA was to parameterize the rule and optimize a type of trading schedule. Another approach which could be applied, is to use GP to evolve the structure of the process for each of the decisions outlined above. A set of these processes that perform well together can then be combined to form a trading algorithm giving rise to the possibility of co-evolving the various aspects of a trading algorithm.

In the U.S. market a vast array of trading algorithms are available to traders with many of these designed to cater to specific needs or tailored for certain markets or market conditions. At Pipeline Financial these algorithms are viewed as *tactics*. These tactics are classified under six categories:

1) Hidden—Places hidden or reserve orders to displayed market places
2) Dark—Interacts with various dark pool market destinations
3) Peg—Maintains a presence on the inside bid or ask as it moves through different price points
4) Participate—Use market or limit orders on various markets to participate wherever the stock is being traded.
5) Opportunist—Monitor the market for price or liquidity opportunities.
6) Stealth—Pings destinations with aggressive limit orders so stealthily take liquidity or to achieve a greater rate of participation.

An optimal trading strategy can learn to employ these tactics under their ideal conditions at the right time. Pipeline Financial's Algorithm Switching Engine analyzes over forty variables to determine the market state and then selects the tactic that will achieve the lowest market impact while maintaining a desired execution rate. As can be seen, the design of real-world trading algorithms is substantially more complex than most published studies indicate.

D. Benchmarking Algorithmic Trading

A key problem in algorithmic trading is evaluating the performance of the algorithm itself. When algorithmic trading began simple benchmarks were used such as those outlined in Eq. (1) (a VWAP calculation). Here the average price obtained (or paid) by the algorithm is compared against the VWAP attained

over the same period of trading or over the whole day. This metric, while easy to calculate, can break down when the size of the order represents 10% or more of the daily volume of the stock, as the order itself will form a large part of the VWAP calculation and a couple of prints below VWAP will generate a superior average price for the algorithm. It is also possible to tailor an algorithm, relatively simply, to match VWAP by having the algorithm participate at an equal proportion to the volume profile of the stock over the course of the day [33].

In recent years as trading algorithms have become more sophisticated, VWAP has lost some importance as a benchmark. Other benchmark metrics include implementation shortfall and arrival price but like VWAP these metrics can also be very context sensitive and might not accurately reflect algorithmic performance. For example, if implementation shortfall was the chosen benchmark, an algorithm to buy stock, in a tanking market, will look great regardless of whether it has done a good job or not as the price at the close of the order will generally be lower than when the algorithm began its work. Here a more accurate measure would be to analyse the market impact and the deviation in the trajectory of the stock from that of the market indices.

Because the nature of market impact and opportunity cost is so dynamic, accurately predicting performance without actually committing the capital to testing the algorithm in a real environment is a very difficult task. Here the development of artificial multi-agent markets and practical market impact models can go a long way towards simulating the effects and behavior a trading algorithm would have on the real markets [24].

Some Financial Terms Explained

Options: A derivative of equities that provide the holder with the option to buy or sell a stock at a certain price by a certain date. There are two types of options: calls and puts. A call gives the holder the right to buy a stock at a certain price and a put allows the holder to sell at a certain price. Options may be bought long or sold short.

Futures: Synthetic contracts with cash settlement that represent the price views of investors on a security for a certain date.

Derivatives: Synthetic securities that are based on underlying asset(s). The price will fluctuate as a function of the price of the underlying asset(s).

Iceberg orders: These can be a reserve order that is displaying a certain amount on the national markets but holding the majority of the order in reserve or it can refer to the size a trader keeps private while he sends a portion of an order out to his broker.

Dark Liquidity: Resides in hidden order types or Dark Pools. Used by traders to hide information on the size/side of their order from the market place.

Sniffers/Pingers & Snipers: Types of trading algorithms used to source and target dark liquidity.

IX. Conclusion

An excellent review by Chen & Kuo (2002) [16] listed nearly 400 papers that had been published by 2001 on the use of EC in computational finance and economics. Several hundred additional papers have been published since then illustrating the continued growth in this applications area. A number of reasons for this continued growth can be identified, including:

- ❏ the increasing availability of data in electronic form,
- ❏ the increasing power of computers which alters the relative costs of financial theory vs inductive modeling methodologies,
- ❏ master's level courses in computational finance and economics are becoming staple offerings in leading colleges
- ❏ the Red Queen effect.

In considering future directions for research at the nexus of computational intelligence and finance, the key development currently taking place is the maturing of research from simplistic 'proof of concept' studies to (hard!) real-world finance problems. This transition requires the deepening of the realism of the financial problems addressed and the development of multi-disciplinary research teams with both CI and finance expertise.

Disclaimer

The opinions and inferences in this article are solely those of the authors and do not necessarily reflect the view and opinion of Pipeline Financial Group or Pipeline Trading Systems, LLC. This article is for informational purposes only and is not soliciting any action.

References

[1] Algos 3.0, *Developments in Algorithmic Trading, Traders Magazine 2007*. Special Report. SourceMedia's Custom Publishing Group.

[2] R. Almgren and N. Chriss, "Optimal Execution of Portfolio Transactions," *Journal of Risk 3*, pp 5–39. Incisive Media. 1999.

[3] E. Altman, "Financial Ratios, Discriminant Analysis and the Prediction of Corporate Bankruptcy," *Journal of Finance*, vol. 23, pp. 589–609, 1968.

[4] H.J. Antonisse, "A grammar-based genetic algorithm," in *Foundations of Genetic Algorithms*, G. J. E. Rawlins, Ed. Indiana University, Bloomington, USA: Morgan Kaufmann, 15–18 July, 1990, pp. 193–204.

[5] S. Arone, A. Loraschi, and A. Tettamanzi, "A genetic approach to portfolio selection, Neural Network World," *International Journal on Neural and Mass-Parallel Computing and Information Systems*, vol. 3, pp. 597–604, 1993.

[6] W. Banzhaf, "Genotype-phenotype-mapping and neutral variation—A case study in genetic programming," in *Lecture Notes in Computer Science 866, Parallel Problem Solving from Nature III*, Springer, 1994, pp. 322–332.

[7] W. Banzhaf, P. Nordin, R.E. Keller, F.D. Francone, *Genetic Programming—An Introduction; On the Automatic Evolution of Computer Programs and its Applications*. Morgan Kaufmann, 1998.

[8] R. Bauer, *Genetic Algorithms and Investment Strategies*. New York: Wiley, 1994.

[9] F. Black and M. Scholes, "The Pricing of Options and Corporate Liabilities," *Journal of Political Economy*, vol. 81, pp. 637–659, 1973.

[10] A. Brabazon and M. O'Neill, *Biologically Inspired Algorithms for Financial Modelling*. Springer: Berlin. 2006.

[11] A. Brabazon and M. O'Neill (eds), *Natural Computing in Computational Finance*. Springer: Berlin, 2008.

[12] A. Brabazon, and M. O'Neill, (eds), *Computational Intelligence in Finance*. Springer: Berlin (forthcoming). 2009.

[13] S.-H. Chen, *Genetic Algorithms and Genetic Programming in Computational Finance*. (ed)., Kluwer Academic Publishers, 2002.
[14] S.-H. Chen, *Evolutionary Computation in Economics and Finance*. Physica–Verlag, 2002.
[15] S.-H. Chen, P. Wang, and T.-W. Kuo, (eds) *Computational Intelligence in Economics and Finance (vol. II)*, Springer: Berlin, 2007.
[16] S.-H. Chen and T.-W. Kuo, "Evolutionary Computation in Economics and Finance: A Bibliography," in *Evolutionary Computation in Economics and Finance* (ed). Physica-Verlag, 2002.
[17] C. Darwin, *On the Origin of the Species by Means of Natural Selection, or the Preservation of Favoured Races in the Struggle for Life (reprinted 1985)*. London: Penguin Books, 1859.
[18] K. Deb, *Multi-objective Optimization using Evolutionary Algorithms*. John Wiley & Sons, 2001.
[19] K.A. De Jong, *Evolutionary Computation: A Unified Approach*. MIT Press, 2006.
[20] I. Dempsey, *Grammatical Evolution in Dynamic Environments*. PhD Thesis, University College Dublin, 2007.
[21] R. Drezewski and L. Siwik, "Co-Evolutionary Multi-Agent System for Portfolio Optimization," in *Natural Computing in Computational Finance*, A. Brabazon and M. O'Neill (eds), pp. 271–299, Springer: Berlin, 2008.
[22] J.D. Farmer, A. Gerig, F. Lillo, and H. Waelbroeck, "What Determines Market Impact? How Efficient Prices Coexist with the Heavy Tails of Supply and Demand," *Presented at the Haas School of Business Finance Seminar*, 2/28/2008.
[23] J.D. Farmer and N. Zamani, "Mechanical vs. Informational Components of Price Impact," *Eur. Phys. J.* vol. B55, pp. 189–200, 2007. Santa Fe Institute Working Paper 06–08–034, Santa Fe, NM, 2006.
[24] J.D. Farmer, P. Patelli, and I.I. Zovko, "The Predictive Power of Zero Intelligence in Financial Markets," *Supplementaty Information PNAS,* USA vol. 102, no. 6, 2005.
[25] D.B. Fogel, (Ed.) *Evolutionary Computation: The Fossil Record*. IEEE Press, 1998.
[26] L. Fogel A. Owens, and M. Walsh, *Artificial Intelligence through Simulated Evolution*. New York: Wiley, 1966.
[27] A. Geyer-Schulz, "Fuzzy Rule-Based Expert Systems and Genetic Machine Learning," 2nd ed., ser. *Studies in Fuzziness and Soft Computing*, Heidelberg: Physica-Verlag, 1996, vol. 3, 1996.
[28] D.E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Reading, MA: Addison-Wesley, 1989.
[29] F. Gruau, "On using syntactic constraints with genetic programming," in *Advances in Genetic Programming 2*, P.J. Angeline and K.E. Kinnear, Jr., Eds. Cambridge, MA, USA: MIT Press, ch. 19, 1996, pp. 377–394.
[30] R. Harper and A. Blair, "A structure preserving crossover in grammatical evolution," in *Proceedings of the 2005 IEEE Congress on Evolutionary Computation*, D. Corne, Z. Michalewicz, M. Dorigo, G. Eiben, D. Fogel, C. Fonseca, G. Greenwood, T. K. Chen, G. Raidl, A. Zalzala, S. Lucas, B. Paechter, J. Willies, J. J. M. Guervos, E. Eberbach, B. McKay, A. Channon, A. Tiwari, L. G. Volkert, D. Ashlock, and M. Schoenauer, Eds., vol. 3. Edinburgh, UK: IEEE Press, 2–5 Sept., 2005, pp. 2537–2544.
[31] J. Hicklin, *Application of the genetic algorithm to automatic program generation*. Master's thesis, University of Idaho, Moscow, ID, 1986.
[32] N.X. Hoai, R.I. McKay, and D. Essam, "Some experimental results with tree adjunct grammar guided genetic programming," in *Genetic Programming, Proceedings of the 5th European Conference, EuroGP 2002*, ser. LNCS, J. A. Foster, E. Lutton, J. Miller, C. Ryan, and A. G. B. Tettamanzi, Eds., vol. 2278. Kinsale, Ireland: Springer-Verlag, 3–5 Apr. 2002, pp. 228–237.
[33] D.D. Hobson, "VWAP and Volume Profiles," *Journal of Trading Spring*, vol. 1, no. 2, pp. 38–42, 2006.
[34] H. Hochreiter, "Evolutionary Stochastic Portfolio Optimization," in *Natural Computing in Computational Finance*, A. Brabazon and M. O'Neill (eds), pp. 67–87, Springer: Berlin, 2008.
[35] J. Holland, *Adaptation in Natural and Artificial Systems*. Ann Arbour: The University of Michigan Press, 1975.
[36] G. Iori, M.G. Daniels, J.D. Farmer, L. Gillemot, S. Krishnamurty, and E. Smith, "An Analysis of Price Impact Function in Order-Driven Markets," *Physica A*, vol. 324, no. 1–2, pp. 145–151, 2003.
[37] M. Keijzer and V. Babovic, "Dimensionally aware genetic programming," in *Proceedings of the Genetic and Evolutionary Computation Conference*, W. Banzhaf, J. Daida, A. E. Eiben, M. H. Garzon, V. Honavar, M. Jakiela, and R. E. Smith, Eds., vol. 2. Orlando, Florida, USA: Morgan Kaufmann, 13–17 July 1999, pp. 1069–1076.
[38] R.E. Keller and W. Banzhaf, "Genetic programming using genotype-phenotype mapping from linear genomes into linear phenotypes," in *Genetic Programming 1996: Proceedings of the First Annual Conference*, J. R. Koza, D. E. Goldberg, D. B. Fogel, and R. L. Riolo, Eds. Stanford University, CA, USA: MIT Press, 28–31 July 1996, pp. 116–122.
[39] J.R. Koza, *Genetic Programming*. MIT Press, 1992.
[40] J.R. Koza, *Genetic Programming II: Automatic Discovery of Reusable Programs*. MIT Press, 1994.
[41] J.R. Koza, D. Andre, F.H. Bennett III, M. Keane, *Genetic Programming 3: Darwinian Invention and Problem Solving*. Morgan Kaufmann, 1999.
[42] J.R. Koza, M. Keane, M.J. Streeter, W. Mydlowec, J. Yu, G. Lanza, *Genetic Programming IV: Routine Human-Competitive Machine Intelligence*. Kluwer Academic Publishers, 2003.
[43] F. Larkin and C. Ryan, "Good News: Using news feeds with genetic programming to predict stock prices," in O'Neill, M. et al (eds), *Proceedings of the 11th European Conference on Genetic Programming (EuroGP 2008)*, Springer, 2008, pp. 49–60, LNCS 4971.
[44] C. Lee, C. Quek, and D. Maskell, "A brain-inspired fuzzy neuro-predictor for bank failure analysis," in *Proceedings of the IEEE International Conference on Evolutionary Computation (CEC 2006)*, IEEE Press, 2006, pp. 7927–7934.
[45] M. Lim and R. Coggins, "Optimal trade execution: An evolutionary approach," in *Proceedings of the IEEE International Conference on Evolutionary Computation (CEC 2005)*, IEEE Press, 2005, pp. 1045–1052.
[46] J.D. Lohn, D.S. Linden, G.S. Hornby, W.F. Kraus, A. Rodriguez, S. Seufert, "Evolutionary Design of an X-Band Antenna for NASA's Space Technology 5 Mission," *Proc. 2004 IEEE Antenna and Propagation Society International Symposium and USNC/URSI National Radio Science Meeting*, 2004, vol. 3, pp. 2313–2316.
[47] A. Loraschi and A. Tettamanzi, M. Tomassini, and P. Verda, "Distributed genetic algorithms with an application to portfolio selection problems," in D. Pearson, N. Steele and R. Albrecht (eds), *Artificial Neural Networks and Genetic Algorithms*, Springer, 1995, pp. 384–387.
[48] S. Markose, E. Tsang, and H. Er, "Evolutionary Decision Trees for Stock Index Options and Futures Arbitrage," in *Genetic Algorithms and Genetic Programming in Computational Finance*, Shu-Heng Chen (ed)., Kluwer Academic Publishers, 2002, pp. 281–308.
[49] H. Markowitz, "Portfolio Selection," *Journal of Finance*, vol. 1, no. 7, pp. 77–91, 1952.
[50] H. Markowitz, *Portfolio Selection: Efficient Diversification of Investments*. John Wiley & Sons, 1959.
[51] Maxin A. M. Al Janabi, "Integrating Liquidity Risk Factor into a Parametric Value at Risk Method," *Journal of Trading Summer 2008*, pp. 76–87. Institutional Investor. 2008.
[52] R. Merton, "Rational Theory of Option Pricing," *Bell Journal of Economics and Management Science*, vol. 4, pp. 141-183, 1973.
[53] NYSE Euronext.
[54] M. O'Neill, C. Ryan, *Grammatical Evolution: Evolutionary Automatic Programming in an Arbitrary Language*. Kluwer Academic Publishers, 2003.
[55] M. O'Neill, *Automatic Programming in an Arbitrary Language: Evolving Programs in Grammatical Evolution*. PhD thesis, University of Limerick, 2001.
[56] M. O'Neill, C. Ryan, "Grammatical Evolution," *IEEE Trans. Evolutionary Computation*, 2001.
[57] M. O'Neill, C. Ryan, M. Keijzer, M. Cattolico, "Crossover in Grammatical Evolution," *Genetic Programming and Evolvable Machines*, Kluwer Academic Publishers, vol. 4, no. 1, 2003.
[58] M. O'Neill, "A. Brabazon, Grammatical Swarm: The generation of programs by social programming," *Natural Computing*, vol. 5, no. 4, pp. 443–462, 2006.
[59] M. O'Neill, A. Brabazon, "Grammatical Differential Evolution," In *Proceedings of IC-AI.* CSREA Press, 2006, pp. 231–236.
[60] N. Paterson and M. Livesey, "Evolving caching algorithms in C by genetic programming," in *Genetic Programming 1997: Proceedings of the Second Annual Conference*, J.R. Koza, K. Deb, M. Dorigo, D.B. Fogel, M. Garzon, H. Iba, and R.L. Riolo, Eds. Stanford University, CA, USA: Morgan Kaufmann, 13–16 July 1997, pp. 262–267.
[61] R. Poli, W.B. Langdon, and N.F. McPhee, *A Field Guide to Genetic Programming*. Published via http://lulu.com and freely available at http://www.gp-field-guide.org.uk. 2008.
[62] G. Pui, C. Fung, C.J. Yu, and W. Lam, "Stock Prediction: Integrating Text Mining Approach using Real-Time News," in *Proceedings of the 2003 IEEE International Conference on Computational Intelligence for Financial Engineering*, IEEE Press, 2003, pp. 395–402.
[63] A. Ratle and M. Sebag, "Genetic programming and domain knowledge: Beyond the limitations of grammar-guided machine discovery," in *Parallel Problem Solving from Nature—PPSN VI 6th International Conference*, ser. LNCS, M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J. J. Merelo, and H.-P. Schwefel, Eds., vol. 1917, Paris, France: Springer Verlag, 16–20 Sept. 2000, pp. 211–220.
[64] I. Rechenberg, *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Stuggart: Frommann-Holzboog, 1973.
[65] R. Rothlauf and M. Oetzel, "On the locality of grammatical evolution," *Genetic Programming, Proceedings of the 9th European Conference, EuroGP 2006*, ser. LNCS, P. Collet, M. Tomassini, M. Ebner, S. Gustafson and A. Ekart, Eds., vol. 3905. Budapest, Hungary: Springer-Verlag, 10–12 Apr., 2006, pp. 320–330.
[66] C. Ryan, J.J. Collins, and M. O'Neill, "Grammatical Evolution: Evolving Programs for an Arbitrary Language," *Proc. of the First European Workshop on GP*, Springer-Verlag, 1998, pp. 83–95.
[67] H.-P. Schwefel, *Evolutionstrategie und numerische Optimierung*. Dissertation, Technische University at, Berlin, 1975.
[68] Y. Shan, R.I. McKay, R. Baxter, H. Abbass, D. Essam, and N.X. Hoai, "Grammar model-based program evolution," in *Proceedings of the 2004 IEEE Congress on Evolutionary Computation*, Portland, Oregon: IEEE Press, 20–23 June 2004, pp. 478–485.
[69] H. Spencer, *The Principles of Biology*. vol. 1, London and Edinburgh: Williams and Norgate, 1964.
[70] F. Streichert, H. Ulmer, and A. Zell, "Evaluating a Hybrid Encoding and Three Crossover Operators on the Constrained Portfolio Selection Problem," in *Proceedings of CEC 2004*, IEEE Press, 2004a, pp. 932–939.
[71] J. Thomas and K. Sycara, "GP and the Predictive Power of Internet Message Traffic," in *Genetic Algorithms and Genetic Programming in Computational Finance*, Shu-Heng Chen (ed)., Kluwer Academic Publishers, 2002, pp. 81–102.
[72] E. Tsang and J. Li, "EDDIE for Financial Forecasting," in *Genetic Algorithms and Genetic Programming in Computational Finance*, Shu-Heng Chen (ed)., Kluwer Academic Publishers, 2002, pp. 161–174.
[73] E. Tsang and S. Martinez-Jaramillo, "Computational Finance," *IEEE Computational Intelligence Society Newsletter*, pp. 8–13, Aug. 2004.
[74] W. Tung and C. Quek, "GenSoOPATS: A brain-inspired dynamically evolving option pricing model and arbitrage system," in *Proceedings of CEC 2005*, IEEE Press, 2005, pp. 1722–1729.
[75] A.M. Turing, "Intelligent Machines," In Ince, D.C. (Ed.), 1992, *Mechanical Intelligence: Collected Works of A.M. Turing*, North-Holland, 1948, pp. 107–128.
[76] P.A. Whigham, Context free grammar and genetic programming, tr cs20/94, Dept of Computer Science, University of New South Wales@ADFA, Tech. Rep.
[77] M.L. Wong and K.S. Leung, "Applying logic grammars to induce sub-functions in genetic programming," in *1995 IEEE Conference on Evolutionary Computation*, vol. 2, Perth, Australia: IEEE Press, 29 Nov. 1, Dec., 1995, pp. 737–740.
[78] D. Farmer, A. Gerig, F. Lillo, and H. Waelbroeck, "What Determines Market Impact? How Efficient Prices Coexist with the Heavy Tails of Supply and Demand," Presented at the *Haas School of Business Finance Seminar*, 2/28/2008.