

## Chapter 2

# GENETIC ALGORITHMS IN ECONOMICS AND FINANCE: FORECASTING STOCK MARKET PRICES AND FOREIGN EXCHANGE — A REVIEW

Adrian E. Drake

*Sirius International Insurance Corporation*  
*ABB Financial Services Group*  
*Stockholm, Sweden*  
*adrian.drake@se.abb.com*

Robert E. Marks

*Australian Graduate School of Management*  
*Universities of New South Wales and Sydney*  
*UNSW Sydney 2052 Australia*  
*bobm@agsm.edu.au*

### Abstract

After a brief overview of the history of the development and application of genetic algorithms and related simulation techniques, this chapter describes alternative implementations of the genetic algorithm, their strengths and weaknesses. Then follows an overview of published applications in finance, with particular focus on the papers of Bauer, Pereira, and Colin in foreign exchange trading. Many other rumored applications remain unpublished.

**Keywords:** Genetic Algorithms, Genetic Programming, Finance, Forecasting, Exchange Rates, Trading Rules

### Introduction

The increased availability of computing power in the past two decades has been used to develop new techniques of forecasting. Today's com-

to find rules formed on the basis of financial ratios which were indicating future bankruptcy. Genetic algorithms were able to come up with simple rules that consistently outperformed the Multiple Discriminant Analysis approach by more than 10%.<sup>58</sup>

Genetic algorithms have also been used for **credit scoring**. This is an universal business problem, having to assess the risk associated with any form of investment. Banks have been using models based on genetic algorithms, reporting considerable savings in some cases. Genetic algorithm approaches to this problem have been superior to others (such as neural nets) because of the transparency of the achieved results.<sup>59</sup>

Other applications of genetic algorithms in finance include database mining, training neural networks, financial spreadsheets, filtering insurance applications, investment portfolio selection and management,<sup>60</sup> as well as trading rules for stock or bond markets.<sup>61</sup> The latter application, creating rules for **trading in stock or bond markets**, is frequently illustrated in literature and has a very important characteristic: Even a small amount of performance improvement (through the use of genetic algorithms) is likely to yield significant payoffs! This of course also applies to trading rules for foreign exchange markets, which will be covered below.

Trading in stock markets can take advantage of the ability to predict time series containing market data. In one genetic-algorithm approach, 18 market indicators were included in the model. In this approach, the genetic algorithm's chromosomes consisted of a list of 36 real-valued numbers. Two of these numbers were associated with each of the 18 market indicators, one showing a low value of the indicator and one denoting a high value. Each chromosome was evaluated by determining the prediction system's historical performance when each indicator value had fallen between the low and the high values associated with it on the chromosome. After letting the genetic algorithm evolve the population, the result was a set of chromosomes that described combinations of parameter values correlated with high performance of the prediction system. The user of this system now only need wait until current indicator values fall into the ranges specified by one of these chromosomes. She then can perform a trade based on the predictions of the system.<sup>62</sup> The technique has been reported to perform well.<sup>63</sup>

A study of asset-allocation strategies was performed by Bauer and Liepins.<sup>64</sup> They examined strategies that involved switching between an investment in a Standard & Poor's 500 fund and a small-firm equity fund. The investor decides every month whether to be 100% in S&P 500 or 100% in small-firm stocks. He has a five-year investment horizon and wants to maximize terminal wealth. A particular switching strategy is

represented as a 60-character bit string. The results from test runs for 12 five-year periods dating back to 1926 impressively demonstrated the genetic algorithm's ability to quickly find attractive problem solutions. The algorithm converged on average in approximately 50 generations. The average solution was within 0.3 percent of the optimal value.

An increase in complexity by incorporating transaction costs into the switching was easily accommodated by the genetic algorithm through changes in the evaluation function. It still performed reasonably well, showing average solutions that were within 3 percent of the optimal solution. The complexity was magnified even further by adding two more assets to the investment alternatives and assuming differential transaction costs. The genetic algorithm still performed quite well, despite the increase in complexity. The terminal wealth for the best performing individual was on average 94.5% of the optimal value.

Bauer and Liepins concluded that the power and flexibility of genetic algorithms enables searches of problem spaces in an extremely efficient manner. The algorithm converged quickly to optimal or near-optimal solutions, and variations of the basic problem could easily be accommodated through representation and by the evaluation function. Simplicity and flexibility are major advantages of genetic algorithms when programming. The authors suggest that genetic algorithms may be used in detecting and evaluating intra-day trading patterns.<sup>65</sup>

A major problem they found, however, was the representation issue. No generally accepted application code exists on this yet. This is a difficult but crucial decision that must be made before applying genetic algorithms to a particular problem. Another hindrance when applying the genetic algorithm to finance applications is the fact that tailoring it to the problem's specifications might produce better results. For example, one could hybridize the genetic algorithm with other algorithms currently used in the domain. As much as this promises better results, it is more of an art, requiring more knowledge and experience than when just using a genetic algorithm. Other problems are the task of coming up with a suitable evaluation function and the availability of CPU time. The latter problem of course is being eased by computing resources constantly becoming cheaper and more powerful.<sup>66</sup>

The potential benefits of using genetic algorithms have been called mind-boggling, and Bauer and Liepins note that the genetic algorithm's efficiency suggests that they might be of use in real-time trading applications.<sup>67</sup> The more one wants to find out about current use of genetic algorithms in trading applications, however, the less people are willing to tell, referring to "proprietary knowledge". Does part of the answer to the question "If you're so smart, why aren't you rich?" lie herein?

putational capacity and the widespread availability of computers have enabled development of a new generation of intelligent computing techniques, such as expert systems, fuzzy logic, neural networks, and genetic algorithms. These "intelligent" computing techniques are concerned with performing actions that imitate human decision makers. They are flexible and so can adapt to new circumstances. They can learn from past experience. Moreover, they can find solutions to problems unsolvable by traditional means. Of the computing techniques mentioned above, genetic algorithms are the most powerful and yet the most simple innovation. They are showing very promising results in improving or replacing existing statistical methods.<sup>1</sup>

The genetic algorithm is a surprisingly simple problem-solving technique, inspired by biological evolution. It is based on an artificially simulated process of natural selection, or survival of the fittest, known as Darwinian evolution. Genetic algorithms have emerged as a powerful general-purpose search-and-optimization technique and have found applications in widespread areas.<sup>2</sup>

## 1. History of Genetic Algorithms

Darwin's natural selection is the inspiration for genetic algorithms. Computational problems often require searching through a large number of possible solutions. What can help in this case is the application of parallelism, where many possibilities are explored simultaneously, and an intelligent strategy for selecting the next set of sequences to evaluate is used. These and other helpful features can be observed in the search for constantly evolving fitness criteria in natural evolution.<sup>3</sup> Scientists have been impressed by the remarkable power of natural evolution for a long time, and had been trying to mimic it in procedures for computer programs. The first attempts to combine computer science and evolution were made in the late 1950s and early 1960s. They were at first relatively unsuccessful, mainly because they relied too much on the operator of mutation, the evolution process that was emphasized by biology in those days.<sup>4</sup>

By the mid-1960s, John Holland at the University of Michigan had added the operators of mating and cross-over to mutation and was able to create a programming technique that he believed could solve difficult problems by mimicking natural selection. Since then, he has been regarded as the founder of the field of generic algorithms. Research in this area has been extended by two scholars of Holland's, and by many others all over the world, who have created several variations of Hol-

land's original genetic algorithms and applied them in many different areas (Nissan 1995).

Today, the term "evolutionary algorithms" is used as an umbrella term for problem-solving computer optimization techniques that are based on some mechanisms of natural evolution as key elements in their design and implementation.<sup>5</sup> The major evolutionary algorithms are genetic algorithms, genetic programming, evolution strategies, and evolutionary programming, all being evolution-based systems with a population of individuals that undergo some transformations, during which the individuals or their "genes" "fight" for survival.<sup>6</sup>

Genetic algorithms are the most popular form of evolutionary algorithms. We will examine them in detail below. Genetic programming is a modification of genetic algorithms for evolving computer programs, rather than simple strings (Koza 1993). Evolution strategies (Evolutionstrategien) is a form of evolutionary algorithms that uses non-string representation. It was created in Germany and relies more on the operator of mutation. Evolutionary programming is very similar to evolution strategies but it was developed independently. It has no restriction on solution representation.<sup>7</sup> The boundaries between the different branches of evolutionary algorithms have been blurred somewhat in recent years. When people use the term "genetic algorithms" today, it is not always clear whether they relate to Holland's original conception or to one of the numerous modifications and further developments.<sup>8</sup>

Before we move to an description of how genetic algorithms function and how they can contribute in forecasting foreign exchange markets, we discuss a wide range of genetic algorithms applications, which shows how widely genetic algorithms are already being used.

## 2. Applications of Genetic Algorithms

Genetic algorithms have a number of characteristics that make them very suitable for use in applications. They can solve hard problems quickly and reliably. They are broadly competent algorithms that can solve problems with many hard-to-find optima. Since genetic algorithms use very little problem-specific information, they are easy to connect to existing application codes, making them easy to interface with existing simulations and models. Genetic algorithms are also easy to hybridize, enabling the input of problem-specific information.<sup>9</sup>

As the theory and practice of genetic algorithms were being developed, they were being used in a wide spectrum of real-world applications. The benefits of genetic algorithms in engineering can be illustrated by their application in the **design of turbine blades** for jet engines, as done



by General Electric. The design of turbines involves more than 100 variables. The resulting search space contains more than  $10^{387}$  points.<sup>10</sup> The use of genetic algorithms in this case rendered an improvement of 2 percent in efficiency, an immense gain in this field.<sup>11</sup>

The **design of communications networks** has also been aided by genetic algorithms. The genetic algorithm approach was found to perform well in this setting, cutting design time from two months to two days and saving up to 10 million dollars per design.<sup>12</sup> In addition, genetic algorithms have been applied to problems where the network topology was fixed. They have been found to perform well in this area as well, where the goal of the software was to carry the maximum possible amount of data with the minimum number of transmission lines and switches in an existing network topology.<sup>13</sup>

Genetic algorithms have been used in **controlling and managing gas pipelines** that consist of many branches carrying different amounts of gas. The solution cut costs, optimized energy use, minimized false leak alarms as well as creating a set of rules for responding properly to punctures in the pipelines.<sup>14</sup>

In distribution, companies have used genetic algorithms for what is commonly known as the **classical traveling salesman problem**. Manufacturing companies must make efficient use of production capacity, while shipping the most goods on the fewest trucks using the best routes. The new model, based on genetic algorithms, efficiently searches through almost limitless possibilities to find the schedule which makes the best use of company resources and also cuts delivery time.<sup>15</sup>

Another type of problem that has been addressed successfully with genetic algorithms is **scheduling**. This is the problem of finding the optimal sequence to execute a set of operations in a way that a certain set of constraints are not violated. Usually, one attempts to maximize the utilization of individuals or machines and to minimize the cost or time required for completing the entire process. Conflicts may arise when one individual or machine is scheduled for more than one operation at the same time, or when more resources than available are utilized. Some tasks may have priority over others.<sup>16</sup>

Likenesses of the faces of criminals from witnesses' descriptions are important for **criminal identification**. Artists' sketches have been used for many years, but nowadays this task can be done by a genetic algorithm application, developed by the psychology department of New Mexico State University. The system presents several computer-generated faces and then asks the witness to rate the faces in terms of which looks most like the criminal. It is much easier for a human mind to recognize similarity in a face than to recall and describe facial features. The

system takes the witness' rating and creates new faces. The process continues until a likeness of the criminal's face evolves. The system has proven to be very successful in tests, and New Mexico State University has applied for a patent.<sup>17</sup>

In **computer chip manufacturing**, Texas Instruments has used genetic algorithms for finding a design on the smallest piece of silicon possible. The new chip took 18 percent less space by using a strategy of cross connections that no human had thought of.<sup>18</sup>

There are of course many more applications that could be added to this list and the number is growing every day.<sup>19</sup> Genetic algorithms have also been used in financial applications, such as credit scoring, bankruptcy prediction, database mining, and for currency trading, which we will discuss later.

We can see that genetic algorithms offer the potential of assisting a broad range of different applications. They imitate natural evolution. Hence, they are designed to cope with the most challenging tasks imaginable. After all, natural evolution has been fine-tuned during billions of years of evolution.

### 3. Description of Genetic Algorithms

#### 3.1. Representation

Natural evolution operates on encodings of biological entities at the level of chromosomes, rather than on the entities themselves. Similarly, genetic algorithms operate on representations of solutions to problems. Since they work with encoded parameters of the optimization problem, the choice of a representation form has a large impact on the performance. There are different ways of encoding solutions, and probably no single best way for all problems. The performance of genetic algorithms depends on the choice of a suitable representation technique.<sup>20</sup>

Most genetic algorithms applications use Holland's fixed-length simple binary coding. This is historically the most widely used representation.<sup>21</sup> Each chromosome is comprised of zeroes and ones, with each bit representing a gene. A conceptually simpler technique would be the real-number representation, in which each variable being optimized is represented as a conventional floating-point number. Another approach might use decimal digits only and many other encoding techniques are possible. Researchers in genetic algorithms are divided on this issue. In the following, the fixed-length simple binary coding will be used to explain how the genetic algorithm works.<sup>22</sup>

The terminology used in the genetic algorithms literature is a mix of expressions originating in natural genetics and in computer science.

Many researchers in the area of genetic algorithms came from the biology domain, therefore the terminology of natural genetics seems to have prevailed. In order to obtain a better understanding, Table 2.1 summarizes the correspondence between natural and artificial terminology.<sup>23</sup>

Table 2.1. Comparison of Natural Genetics and Genetic Algorithms Terminology

Natural genetics	Genetic algorithms
chromosome	string
gene	bit, feature, character, or detector
allele	feature value, value of a gene
locus	position
genotype	structure, coded solution
phenotype	behaviour, parameter set, solution alternative, decoded structure, decoded solution, or a point in the solution space

### 3.2. Genetic Algorithms Step by Step

The typical genetic algorithm consists of a number of steps on its way to finding optimal solutions. Different authors describe different numbers of steps, but all include the characteristic procedures that a genetic algorithm runs through.<sup>24</sup> These will be covered in the following.

**3.2.1 Initialization.** The first step is the creation of an initial population of solutions, or chromosomes. The population of chromosomes is generally chosen at random, for example, by flicking a coin or by letting a computer generate random numbers. There are no hard rules for determining the size of the population. Generally, population size is chosen to be between 100 and 200. Larger populations guarantee greater diversity and may produce more robust solutions, but use more computer resources. The initial population must span a wide range of variable settings, with a high degree of diversity among solutions in order for later steps to work effectively.<sup>25</sup>

The population will converge through the application of the genetic operators. Convergence means that the initially diverse population of chromosomes that has been constructed by a random number generator tends to become more similar as it undergoes the genetic recombinations. The population may even become identical after many generations.<sup>26</sup> If explicit knowledge about the system that has to be optimized is at hand,

it may be included in the initial population, for example, by restricting the initialization to a domain of interest.<sup>27</sup>

**3.2.2 Fitness Evaluation.** In the next step, the fitness of the population's individuals is evaluated. In biology, natural collection means that chromosomes that are more fit tend to produce more offspring than do those that are not as fit. Similarly, the goal of the genetic algorithm is to find the individual representing a particular solution to the problem which maximizes the objective function, so its fitness is the value of the objective function for a chromosome.<sup>28</sup> Genetic algorithms can of course also solve minimization problems.<sup>29</sup>

The fitness function (also called objective function or evaluation function) is used to map the individual's chromosomes or bit strings into a positive number, the individual's fitness. The genotype, the individual's bit string, has to be decoded for this purpose into the phenotype, which is the solution alternative. Once the genotype has been decoded, the calculation of the fitness is simple: we use the fitness function to calculate the phenotype's parameter values into a positive number, the fitness. The fitness function plays the role of the natural environment, rating solutions in terms of their fitness.<sup>30</sup>

**3.2.3 Selection.** In the third step, the genetic algorithm starts to reproduce. The individuals that are going to become parents of the next generation are selected from the initial population of chromosomes. This parent generation is the "mating pool" for the subsequent generation, the offspring.<sup>31</sup> Selection determines which individuals of the population will have all or some of their genetic material passed on to the next generation of individuals. The object of the selection method is to assign chromosomes with the largest fitness a higher probability of reproduction. There are many alternative selection procedures, a live-or-die turning point for the chromosomes.<sup>32</sup> The most common way is Holland's spinning of the weighted roulette wheel.

**Roulette wheel selection** gets its name from the fact that the algorithm works like a roulette game. Segments of the roulette wheel are allocated to population individuals in proportion to the individual's relative fitness score (see Figure 2.1).<sup>33</sup> Selection of parents is carried out through successive spins of the roulette wheel. The selection process is random, but fitter individuals will be more likely to be selected because of the higher probability of getting picked. This is a simple technique of rewarding fitter individuals, but because possible random fluctuations it is not ideal. With the relatively small population sizes typically used in typical genetic algorithms, an unfavorable series of spins of the



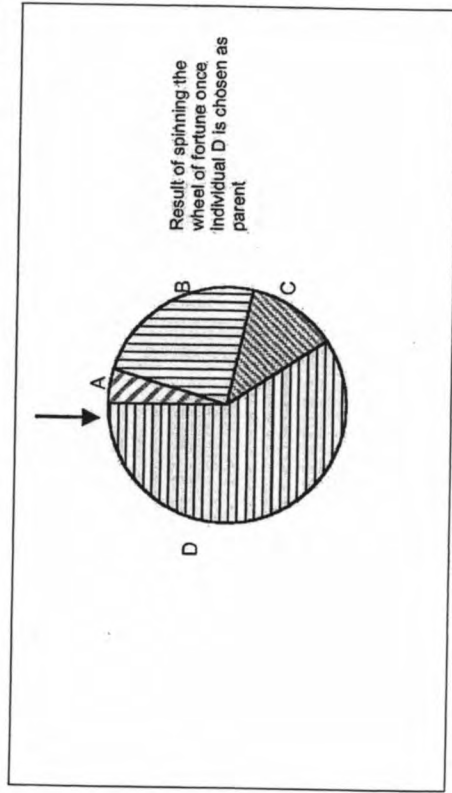


Figure 2.1. Roulette wheel selection: a larger segment implies larger fitness.

roulette wheel can select a disproportionately large number of unfit parent chromosomes.<sup>34</sup>

There are many other ways of accomplishing effective selection, such as stochastic remainder selection, elitism selection, linear fitness selection, genitor selection, ranking selection with roulette wheel, or tournament selection.<sup>35</sup>

**Tournament selection** can involve two or more individuals at a time. In the two-party tournament selection, for example, pairs of strings are randomly chosen from the initial population. Their fitness values are compared and a copy of the better performing individual becomes part of the mating pool. The tournament will be performed repeatedly until the mating pool is filled. That way, the worst performing string in the population will never be selected for inclusion in the mating pool.<sup>36</sup>

The choice of the selection method has a major impact on the balance between "exploitation" and "exploration".<sup>37</sup> Exploitation means taking advantage of information already obtained, while exploration refers to searching new areas. Holland illustrates this with the chess example.<sup>38</sup>

After finding a good strategy for playing chess, one can of course solely concentrate on exploiting this strategy. But this behaviour inhibits the discovery of new, possibly superior, strategies. Improvements will only be discovered by trying new and more risky strategies, by exploring.

If we solely applied the selection procedure, only exploitation would occur. But one attribute of genetic algorithms that makes them superior to conventional problem-solving methods is their remarkable balance

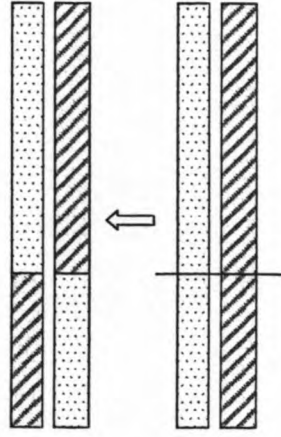


Figure 2.2. Crossover diagram.

between exploitation and exploration.<sup>39</sup> Exploration is being achieved by the genetic operators, to be examined in the next section.

### 3.2.4 Genetic Operators.

**Crossover.** The primary exploration operator in genetic algorithms is crossover, a version of artificial mating. If two strings with high fitness values are mated, exploring some combination of their genes may produce an offspring with even higher fitness. Crossover is a way of searching the range of possible solutions based on the information obtained from the existing solutions.

There are many ways in which crossover can be implemented, such as one-point crossover, two-point crossover,  $n$ -point crossover, or uniform crossover.<sup>40</sup> In the following, we will stay with the simplest form, Holland's one-point crossover technique.

Single-point crossover is the simplest form, yet it is highly effective. First, two potential parents are selected from the population. Then, a random position on the bit string is selected, the cross point or cut point. The bits to the left of the cross point on parent one are combined with the bits to the right of the cut point in parent 2 to form child 1. The opposite segments are combined to create child 2 (see Figure 2.2).<sup>41</sup>

Both children will tend to be different from either of their parents yet retain some features of both. If both parents had high fitness (which is likely because of the selection procedure), then there is a high chance that at least one of the children is as fit or as fitter than either parent. In this case, further selection procedures will favor the child's reproduction, otherwise they will favor the extinction of its genetic information, since it will not be selected as a parent for the next generation.

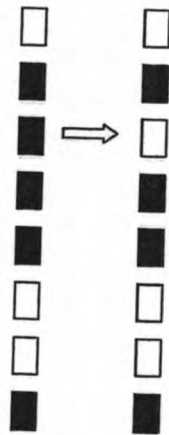


Figure 2.3. Mutation diagram.

In order to counter the possibility that most or all of the crosses produce children of lower fitness, a simulated coin is flipped before the crossover operation is performed. The coin is biased in a way that it lets crossover be performed more often than is required to pass the parents' genetic information into the next generation unchanged.

**Mutation.** If crossover is the main operator of genetic algorithms that efficiently searches the solution space, then mutation could be called the "background operator" that reintroduces lost alleles into the population.<sup>42</sup> Mutation occasionally injects a random alteration for one of the genes (see Figure 2.3).<sup>43</sup> Similar to mutation in nature, this function preserves diversity in the population. It provides innovation, possibly leading to exploration of a region of better solutions. Mutation is performed with low probability. Applied in conjunction with selection and crossover, mutation not only leads to an efficient search of the solution space but also provides an insurance against loss of needed diversity.<sup>44</sup>

**Inversion.** Another operator in Holland's genetic algorithms was the inversion operator. Inversion is a process that shifts the locus of one or more genes in a chromosome from one point to another. This does not change the meaning of the genotype, in that a genotype before and after inversion will still decode to the same phenotype. This is the same as in natural genetics, where the function of a gene is independent of its position in the chromosome.<sup>45</sup> In the genetic algorithm setting, inversion has not been found to be useful.<sup>46</sup>

### 3.2.5 Evaluate Offspring and Create a New Population.

Having applied the genetic operators, the new individuals must be evaluated in order to assess their fitness. Hence, the children undergo evaluation through the fitness function, similar to the process in Section 3.2.2. Once they have been assessed, the new, better offspring will re-

place those chromosomes of the population that perform the weakest, according to the fitness function.

**3.2.6 Termination Criteria.** The last step is the verification of a termination criterion. If the termination criterion is not satisfied, the genetic algorithm returns to Section 3.2.3 and continues with the selection and the genetic operators. The termination criterion is satisfied when either the maximum number of generations is achieved or when the genotypes (the structures) of the population of individuals converges. The maximum number of generations is set by the creator of the genetic algorithm before running it, which ensures that the genetic algorithm does not continue indefinitely. Convergence may occur in two ways: first, convergence of the genotype structure occurs when all bit positions in all strings are identical. In this case, crossover will have no further effect.<sup>47</sup> It is also possible that there is phenotypic convergence without genotypic convergence—identical behaviour with dispersant structures—this will happen when only certain, converged genes are tested.

### 3.3. Why do Genetic Algorithms Work?

We have visited the steps of a genetic algorithm and have seen that it is based on relatively simple procedures. But how can a limited number of individuals efficiently explore vast search spaces with trillions of different points? This question leads us to the Schema Theorem or Fundamental Theorem of Genetic Algorithms,<sup>48</sup> which states that each time an individual is processed, the genetic algorithm is sampling not simply one but many points in the search space.

With binary-digit-string genotypes, each position in the string is limited to the characters 0 and 1. We also need the "don't care" symbol (\*), meaning that its value could be 0 or 1. The sample string 00\*\*11 matches all of the string of the following set {000011, 000111, 001011, 001111}. Strings including the don't care symbol are called **schemata** or **similarity templates**. The success of genetic algorithms stems from the fact that they implicitly search for schemata with higher fitness. In a bit string of length five there are only  $2^5 = 32$  individuals, but  $3^5 = 243$  schemata. Every time a genetic algorithm processes a single individual, it actually processes many schemata. Hence, a genetic algorithm that deals with a population of a few hundred or thousand strings in fact samples a vastly greater number of solutions. This **implicit parallelism**, combined with the fact that bad schemata are quickly eliminated, accounts for the genetic algorithm's central advantage over other problem-solving techniques and makes it a powerful search tool.<sup>49</sup>

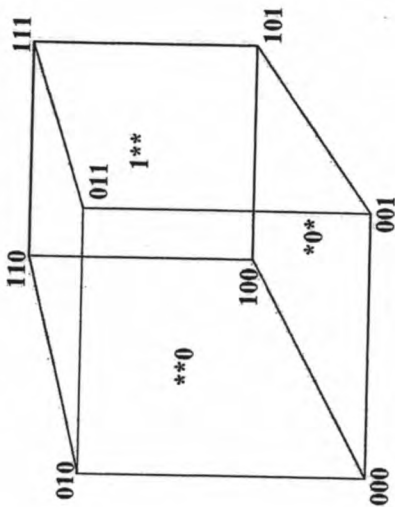


Figure 2.4. Visualization of schemata as hyperplanes in 3-dimensional space.

Although both crossover and mutation destroy existing schemata, they are necessary for building better ones. The degree of destruction is dependent on the order and the length of the schemata, where length is the distance between the first 0 or 1 and the last 0 or 1. For example, in  $1^{***}0^*$ , the length is 4. Order refers to the number of fixed positions (0 or 1) in the schema. Highly fit, short, low-order schemata are called **building blocks**. They tend to stay intact, because when crossover chops up schemata as it recombines individuals, short schemata have a higher chance of surviving crossover intact. Low-order schemata process more individuals and are also more likely to stay intact.<sup>50</sup>

A good visualization of genetic algorithms and schemata is supplied by the geometric representation. The easiest representation is attained for strings and schemata of length 3 in a three-dimensional vector space (see Figure 2.4).<sup>51</sup> Points in the space represent string or schemata of order 3, lines are schemata of order 2, planes in the space are schemata of order 1. The whole space is covered by the schema of order 0:  $\{***\}$ .<sup>52</sup>

### 3.4. Summary

Genetic algorithms mimic natural reproduction and provide a simple yet powerful tool that can be easily transformed into a computer program. Further assessment uncovers the following benefits and drawbacks.

The **advantages** of genetic algorithms are that they can quickly and reliably solve problems that are hard to tackle by traditional means. Genetic algorithms can interface with existing applications, they are extensible, and they are easy to hybridize. Existing knowledge need

not be discarded. Implicit parallelism makes genetic algorithms a very efficient optimization algorithm. Their greatest property is the ability to find approximate solutions to combinatorially explosive problems.<sup>53</sup>

There are, however, also a number of **disadvantages** cited in literature. For one, there is the fact that genetic algorithms may find only near-optimal solutions. There is the difficulty of verifying the optimality of a solution, especially when genetic algorithms have been applied on a problem that would be computationally impossible to solve when using conventional techniques.<sup>54</sup> Further restrictions are the difficulties of choosing a suitable representation technique, and making the right decision regarding the choice of the selection method and the genetic operator probabilities.<sup>55</sup>

## 4. Genetic Algorithms and Financial Applications

After this excursion into the technical world of the genetic algorithms, we return to financial markets. Before examining the technique's usefulness in forecasting foreign exchange markets, we take a closer look at the wide range of genetic algorithms applications in the financial domain, an area well-documented in the literature.

Genetic algorithms appear to be particularly well-suited for financial modelling applications because of three reasons:<sup>56</sup>

- They are payoff-driven. Payoffs can mean improvements in predictive power or returns over a benchmark. There is an excellent match between the problem-solving tool and the issues to be addressed.
- Genetic algorithms are inherently quantitative. They are well-suited to parameter optimization.
- Genetic algorithms allow a wide variety of extensions and constraints that cannot be provided in traditional methods. They are robust, revealing a remarkable balance between efficiency and efficacy. The genetic algorithm's use of a coding, the search from a population, its blindness to auxiliary information, and its randomized operators contribute to its robustness.<sup>57</sup>

Corporate distress and bankruptcy have been an area of research since the 1930s. The most common modelling technique in this area is one dating back to the 1960s: Multiple Discriminant Analysis. Genetic algorithms have been used to induce rules for **bankruptcy prediction** based on balance-sheet information. The genetic algorithm's task was



to find rules formed on the basis of financial ratios which were indicating future bankruptcy. Genetic algorithms were able to come up with simple rules that consistently outperformed the Multiple Discriminant Analysis approach by more than 10%.<sup>58</sup>

Genetic algorithms have also been used for **credit scoring**. This is an universal business problem, having to assess the risk associated with any form of investment. Banks have been using models based on genetic algorithms, reporting considerable savings in some cases. Genetic algorithm approaches to this problem have been superior to others (such as neural nets) because of the transparency of the achieved results.<sup>59</sup>

Other applications of genetic algorithms in finance include database mining, training neural networks, financial spreadsheets, filtering insurance applications, investment portfolio selection and management,<sup>60</sup> as well as trading rules for stock or bond markets.<sup>61</sup> The latter application, creating rules for **trading in stock or bond markets**, is frequently illustrated in literature and has a very important characteristic: Even a small amount of performance improvement (through the use of genetic algorithms) is likely to yield significant payoffs! This of course also applies to trading rules for foreign exchange markets, which will be covered below.

Trading in stock markets can take advantage of the ability to predict time series containing market data. In one genetic-algorithm approach, 18 market indicators were included in the model. In this approach, the genetic algorithm's chromosomes consisted of a list of 36 real-valued numbers. Two of these numbers were associated with each of the 18 market indicators, one showing a low value of the indicator and one denoting a high value. Each chromosome was evaluated by determining the prediction system's historical performance when each indicator value had fallen between the low and the high values associated with it on the chromosome. After letting the genetic algorithm evolve the population, the result was a set of chromosomes that described combinations of parameter values correlated with high performance of the prediction system. The user of this system now only need wait until current indicator values fall into the ranges specified by one of these chromosomes. She then can perform a trade based on the predictions of the system.<sup>62</sup> The technique has been reported to perform well.<sup>63</sup>

A study of asset-allocation strategies was performed by Bauer and Liepins.<sup>64</sup> They examined strategies that involved switching between an investment in a Standard & Poor's 500 fund and a small-firm equity fund. The investor decides every month whether to be 100% in S&P 500 or 100% in small-firm stocks. He has a five-year investment horizon and wants to maximize terminal wealth. A particular switching strategy is

represented as a 60-character bit string. The results from test runs for 12 five-year periods dating back to 1926 impressively demonstrated the genetic algorithm's ability to quickly find attractive problem solutions. The algorithm converged on average in approximately 50 generations. The average solution was within 0.3 percent of the optimal value.

An increase in complexity by incorporating transaction costs into the switching was easily accommodated by the genetic algorithm through changes in the evaluation function. It still performed reasonably well, showing average solutions that were within 3 percent of the optimal solution. The complexity was magnified even further by adding two more assets to the investment alternatives and assuming differential transaction costs. The genetic algorithm still performed quite well, despite the increase in complexity. The terminal wealth for the best performing individual was on average 94.5% of the optimal value.

Bauer and Liepins concluded that the power and flexibility of genetic algorithms enables searches of problem spaces in an extremely efficient manner. The algorithm converged quickly to optimal or near-optimal solutions, and variations of the basic problem could easily be accommodated through representation and by the evaluation function. Simplicity and flexibility are major advantages of genetic algorithms when programming. The authors suggest that genetic algorithms may be used in detecting and evaluating intra-day trading patterns.<sup>65</sup>

A major problem they found, however, was the representation issue. No generally accepted application code exists on this yet. This is a difficult but crucial decision that must be made before applying genetic algorithms to a particular problem. Another hindrance when applying the genetic algorithm to finance applications is the fact that tailoring it to the problem's specifications might produce better results. For example, one could hybridize the genetic algorithm with other algorithms currently used in the domain. As much as this promises better results, it is more of an art, requiring more knowledge and experience than when just using a genetic algorithm. Other problems are the task of coming up with a suitable evaluation function and the availability of CPU time. The latter problem of course is being eased by computing resources constantly becoming cheaper and more powerful.<sup>66</sup>

The potential benefits of using genetic algorithms have been called mind-boggling, and Bauer and Liepins note that the genetic algorithm's efficiency suggests that they might be of use in real-time trading applications.<sup>67</sup> The more one wants to find out about current use of genetic algorithms in trading applications, however, the less people are willing to tell, referring to "proprietary knowledge". Does part of the answer to the question "If you're so smart, why aren't you rich?" lie herein?

## 5. Genetic Algorithms and Foreign Exchange

The literature in the area of genetic algorithms in forecasting on foreign-exchange market is viewed mainly from the perspective of speculators and traders. The genetic algorithm is used to come up with some trading rule that will generate profits. Most of the research is concentrated on finding technical trading rules. But we will first look at an approach assuming a fundamental perspective.

### 5.1. Fundamental Trading Strategies

In his paper on genetic algorithms and exchange rates, Bauer focused on fundamental strategies for trading currencies.<sup>68</sup> Fundamental strategies rely on economic relationships. The economic variables Bauer primarily chose to include in his model are: money supply, inflation, interest rates, and economic activity. He limited his study to the US dollar and the Deutchmark.

Bauer developed trading rules for the monthly allocation of assets either into U.S. dollars or Deutchmarks. He used a simple genetic algorithm with fixed-length binary-string representation and a population size of 100. One-point crossover was performed with a probability of 0.6, and mutation with a probability of 0.001.<sup>69</sup> Simple ranking selection was chosen and the genetic algorithm was limited to 300 generations. Each binary string represented a specific foreign-exchange monthly trading rule. Each month, the trading rule signalled whether to invest money in U.S. dollars or in Deutchmarks. The fitness measure was the U.S.-dollar value at the end of the five-year testing period. Transaction costs and interest were ignored in the calculations of the terminal value.

Bauer reported that the genetic algorithm found only near-optimal solutions in most cases, which he conquered by running multiple trials. The genetic algorithm found trading rules with substantially greater fitness levels than when using enumerative search. It also outperformed the buy-and-hold alternative. The performance of the rules in some intervals was mixed. None of the rules outperformed the buy-and-hold strategy during 1989-1990, an abnormal period, characterized by Germany many expanding its money supply for financing the German reunification process.<sup>70</sup>

Hence, this application of a genetic algorithm for finding trading rules in the foreign exchange market based on fundamental economic variables is limited but shows promising results. Bauer recommends the following extensions and variations of his basic methodology for future research: the substitution or addition of other macroeconomic variables or even technical variables. Genetic-algorithm specifics such as representation,

crossover probability, or the fitness function could also be altered in a sensitivity analysis.

### 5.2. Technical Trading Rules

Greater attention in the application of genetic algorithms in foreign exchange markets has been paid to the detection of trading rules based on technical data. Technical strategies rely on past price and trading-volume information. These variables have a far more short-term character and are hence more useful for the day-to-day trading and in real-time trading applications than are the macro-economic fundamentals.

Technical trading rules use historical price and volume data to predict the direction of future price movements. They then give either a buy or a sell signal.<sup>71</sup> Genetic algorithms have been applied differently in the area of technical trading rules. They have been used for optimizing conventional methods and for designing trading rules.

#### 5.2.1 Optimizing Technical Trading Rules.

Genetic algorithms can be applied in the search for the optimal parameter threshold values for technical trading models. The most elementary way of processing data is the **simple moving average** (see Figure 2.5).<sup>72</sup> It is simply calculated by taking an average of the prices of the last  $L$  days. The parameter  $L$  represents the length of the period of the moving average. If the actual price of a currency rises above the moving average, a buy signal is issued. When the price falls below the moving average, the model issues a sell signal.

This kind of a simple moving average can work well in trending markets. It ignores short-term changes and follows large moves. But since exchange rates usually show high volatility in a relatively narrow range, the simple moving average will lose money. To avoid this, the **double moving average** uses a combination of long-term and short-term moving averages (see Figure 2.6).<sup>73</sup> When the shorter moving average rises above the longer moving average from below, a buy signal is issued. The foreign currency is then held until the shorter moving average falls below the longer moving average, when it is sold.<sup>74</sup>

Another technique that has been used by technical traders is the **filter rule**: if the current investment position in a certain foreign currency is short, a buy signal is issued when its price rises by  $x\%$  above the minimum local price (that is, the minimum observed in the historical series of prices since the last transaction). The foreign currency is then bought and held until the price falls  $x\%$  below the maximum local price (that is, the maximum price observed since the last transaction), when the currency is sold.<sup>75</sup>



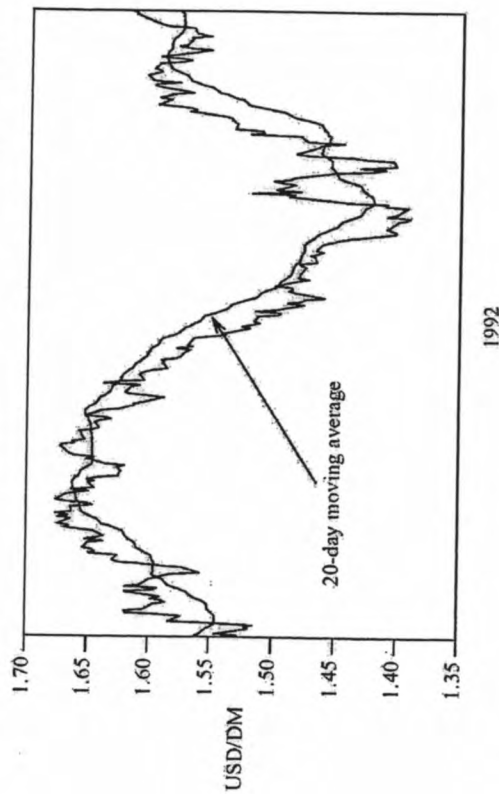


Figure 2.5. The simple moving average of the U.S./ DM exchange rate.

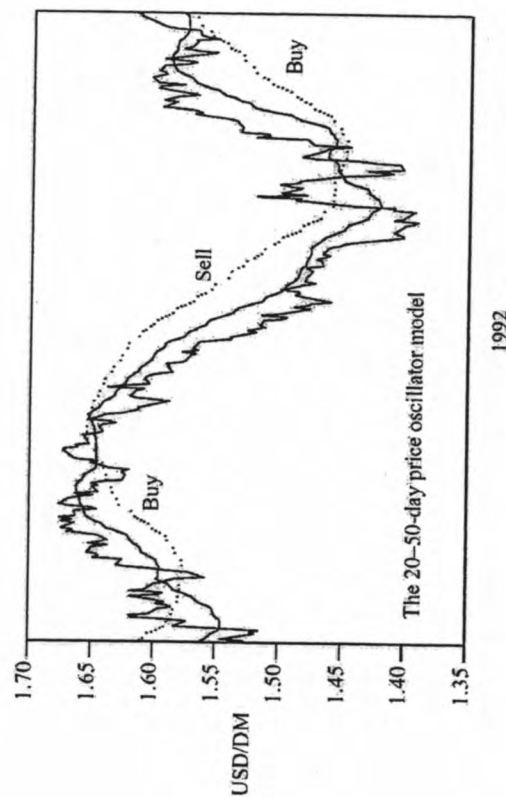


Figure 2.6. The double moving average of the U.S./ DM exchange rate.

A genetic algorithm can now be applied for searching for the optimal threshold values for the trading rule parameters (i.e., for the length  $L$  of the respective moving averages and for the percentage level  $x$ ). Pereira has done this in a study for the U.S. dollar / Australian dollar

exchange rate.<sup>76</sup> Each trading rule required slightly different genetic-algorithm features (i.e., population size). Still, all used fixed-length binary representation, the genitor ranking selection procedure, and the trading rule fitness was evaluated by the profit-maximization criterion. Crossover and mutation occurred with a probability of 0.6 and 0.001, respectively,<sup>77</sup> and the maximum number of generations was limited to 50 in all genetic algorithms.<sup>78</sup>

After letting the genetic algorithm find the optimal parameters for the three different rules and after applying these rules for trading U.S.\$ / AUS\$, the results showed profitable in-sample returns. The search for the complex double-moving-average parameter values was performed more than fifty times faster when using the genetic algorithm than when applying an exhaustive grid-search technique.

The out-of-sample profitability of the trading rules was much lower than in the rule-building period. Still, the trading rules rendered positive returns, with the filter rule being most profitable. But, the study did not consider risk, nor did it include any trading costs, such as transaction costs, taxes, or costs of information. The author concluded that more complex, statistically significant models must include risk and trading costs. A genetic algorithm is a suitable technique for these explosively complex trading systems.<sup>79</sup>

### 5.2.2 Genetic Programming for Rule Induction.

After applying genetic algorithms for optimizing traditional prediction techniques, we will now examine whether they could also be used for the automatic induction of foreign-exchange trading models. We want to find a way of letting the computer develop its own strategies that can model the foreign exchange market's dynamics, based on a number of different inputs.

Suppose we have three different models:<sup>80</sup>

- if (10-day moving average > 0) then buy \$ else sell \$ (model 1)
- if (15-day moving average > 0) then buy \$ else sell \$ (model 2)
- if (25-day moving average > 0) then buy \$ else sell \$ (model 3)

We would like to create a genetic algorithm-based system that produces statements such as:

- if (model 1 says BUY) then BUY else SELL
- if (model 1 and model 2 say BUY) then BUY else SELL
- if (models 1 and 2 say BUY) OR (model 3 says BUY) then BUY else SELL



In this case it is unclear how to represent the statements in terms of a binary string, or how to perform crossover or mutation. In addition, each individual in the population can have a different length and complexity. This will handicap the crossover mechanism.

At this point, a new technique comes into play. **Genetic programming** is an important spin-off from genetic algorithms aiming at automatically evolving computer programs to solve problems.<sup>81</sup> It uses data structures of varying complexity that can grow or shrink through appropriately defined crossover and mutation operators.<sup>82</sup>

The representation now occurs by using a natural mapping between algebraic statements of the form shown above and a binary tree structure. The mutation operator now alters any one quantity in the set of data fields of a binary tree's node. The crossover operator selects two expressions with varying probability and it picks a crossover site in each. A copy of the first parent tree is made and the subtree consisting of the nodes below the crossover site in the second tree is implanted onto the first tree at the first tree's crossover site, replacing the original subtree.<sup>83</sup>

The trading system that has been created based on these specifications generated average returns of about 50% over the 7-year test period, or about 7% per year. Transaction costs were not taken under consideration, and the potential amounts that could have been made or lost by trading for the interest rate differentials between the currency pairs were ignored. Colin concludes that major diversification gains can be rendered by generating numerous non-correlated models (internal diversification), or by diversifying across markets.<sup>84</sup> The main disadvantage of a multiple-model approach is, however, the significant investment that has to be made in computer technology in order to be able to handle the large number of signals generated. Nevertheless, Colin recommends extensions of the model including several markets or economic variables, such as using interest rates and commodity indices, etc.<sup>85</sup>

## 6. Summary

Genetic algorithms have been presented as robust general-purpose optimization techniques. These very efficient problem-solving techniques have found to be useful in many engineering applications as well as in the domain of finance. They can quickly solve difficult problems, they can easily be tailored to a specific problem, and they are easy to interface with existing models. They possess the ability to find approximate solutions to combinatorially explosive problems. This near-optimality has also been viewed as a weakness, though. Further obstacles are the difficulties

of choosing the representation form suitable to a specific problem, as well as the right selection method and the genetic operator probabilities.

Because of the huge potential profits, the genetic algorithm is a very interesting tool for financial applications. In the area of foreign exchange markets, it has been applied in fundamental as well as in technical trading models. Genetic algorithms seem to provide very promising results. Research, however, is still only in its infancy. Early publications on tests in this area report that the models were able to generate profits, but the assumptions that have been made were restricting the models.

## Notes

1. Feldman and Treleaven (1994), p. 195
2. Feldman and Treleaven (1994), p. 199
3. Mitchell (1996), pp. 4-5
4. Holland (1992a), p. 44
5. Heitkoetter and Beasley. (1996), p. 1
6. Michalewicz (1994), pp. 1-3
7. Biethahn and Nissen (1995), pp. 4-33
8. Mitchell (1996), p. 3
9. Goldberg (1994), pp. 114-115
10. There are  $10^{81}$  elementary particles in the known universe.
11. Hughes (1990), p. 22
12. Goldberg (1994), p. 117
13. Davis and Coombs (1987), pp. 252-256
14. Holland (1992a), p. 49
15. Goldberg and Ling Jr. (1985), pp. 154-159 and Grefenstette et al. (1985), pp. 160-168
16. Syswerda (1991), pp. 332-349
17. Caldwell and Johnston (1991), pp. 416-421
18. Kennedy (1996), p. 1
19. For an extensive overview of the wide area of genetic algorithms applications, refer to Biethahn and Nissen (1995), pp. 48-62 or Goldberg (1989), pp. 126-129
20. Davis (1991), p. 4
21. Davis (1994), p. 137
22. For an overview of the work on the representation problem, refer to Koza (1993), pp. 63-68
23. Goldberg (1989), p. 22
24. For an overview of the genetic algorithms steps, refer to Holland (1992a), p. 46, Goldberg (1989), pp. 15-18, Davis (1991), p. 5, Bauer (1994), pp. 11-17 and 55-71, Biethahn and Nissen (1995), pp. 7-13, and Fogel (1995), pp. 89-97.
25. Especially the crossover operator (see Section 3.2.4 below).
26. Bauer (1994), p. 293

27. Biethahn and Nissen (eds., 1995), p. 9
28. Davis (1994), p. 135
29. Bauer (1994), p. 49
30. Michalewicz (1994), pp. 17-20
31. Goldberg (1989), p. 15
32. Goldberg (1994), p. 113
33. Biethahn and Nissen(eds., 1995, p.10)
34. Mitchell (1996), p. 167
35. For an overview of various alternative selection techniques, refer to Goldberg and Deb or Bäck (1996), pp. 163-195.
36. Goldberg (1994), p. 113
37. Kingdon and Feldman (1995), p. 96
38. Holland (1992a), p. 47
39. Holland (1992a), p. 47
40. Bauer (1994), pp. 94-96
41. Bauer (1994), p. 63
42. Biethahn and Nissen (1995), pp. 9-10
43. Bauer (1994), p. 65
44. Goldberg (1994), p. 114
45. Mitchell (1996), p. 159
46. Davis (1991), p. 21
47. Kingdon and Feldman (1995), p. 92
48. Goldberg (1989), p. 33
49. Holland (1992a), p. 46
50. Goldberg (1989), pp. 41-45
51. Koza (1993), p. 34
52. Goldberg (1989), pp. 53-54
53. Feldman and Treleven (1994), p. 201
54. Bodnovich (1995), p. 1
55. Feldman and Treleven (1994), p. 201
56. Leinweber and Arnott (1995), p. 12
57. Goldberg (1989), p. 10
58. Kingdon and Feldman (1995), pp. 109-110
59. Davis (1994), p. 143
60. Edelbuttel (1996a), pp. 1-23 and Edelbuttel (1996b), p. 21
61. For an overview of financial applications of genetic algorithms, refer to Biethahn and Nissen (1995), pp. 49-59.
62. Davis (1994), pp. 141-142
63. Goldberg (1994), p. 116

## REFERENCES

64. Bauer and Liepins (1992), pp. 89-100
65. Bauer and Liepins (1992), p. 97
66. Davis (1994), pp. 145-146
67. Bauer and Liepins (1992), p. 97
68. Bauer (1995), pp. 253-263
69. as recommended in Bauer (1994), pp. 62-65
70. Bauer (1995), p. 262
71. Pereira (1996), p. 28
72. Colin (1994), p. 152
73. Colin (1994), p. 154
74. Pereira (1996), p. 28
75. Pereira (1996), p. 28
76. Pereira (1996), pp. 27-34
77. As recommended in Bauer (1994), pp. 62-65
78. Good instructions for setting up a genetic algorithm for this kind of problems is given in Colin (1994), pp. 152-160.
79. Pereira (1996), pp. 33-34
80. Colin (1994), p. 160
81. Koza (1993), and Biethahn and Nissen (1995), p. 25
82. Colin (2000), p. 176
83. Koza (1992), pp. 225-226
84. Colin (2000), pp. 183-186
85. Colin (2000), p. 188

## References

- Bäck, T. (1996). *Evolutionary Algorithms in Theory and Practice, Evolutionary Strategies, Evolutionary Programming, Genetic Algorithms*. New York and Oxford: Oxford University Press.
- Barnett, W. A., C. Chiarella, S. Keen, R. E. Marks, and H. Schnabl (eds., 2000). *Commerce, Complexity, and Evolution: Topics in Economics, Finance, Marketing, and Management: Proceedings of the Twelfth International Symposium in Economic Theory and Econometrics*. Cambridge: Cambridge University Press.
- Bauer, R. J. Jr. (1994). *Genetic Algorithms and Investment Strategies*. New York: Wiley.
- Bauer, R. J. Jr. (1995). "Genetic Algorithms and the Management of Exchange Rate Risk," in J. Biethahn and V. Nissen (eds.), *Evolutionary Algorithms in Management Applications*, 253-263.

- Bauer, R. J. Jr. and G. E. Liepins (1992). "Genetic Algorithms and Computerized Trading Strategies," in O'Leary and Watkins (eds.), *Expert Systems in Finance*, 89-100.
- Belew, R. K. and L. B. Booker (eds., 1991). *Proceedings of the Fourth International Conference on Genetic Algorithms*. San Mateo: Morgan Kaufmann.
- Biethahn, J. and V. Nissen (eds., 1995). *Evolutionary Algorithms in Management Applications*. Berlin, Heidelberg, New York: Springer-Verlag.
- Bodnovich, T. (1995). "Genetic Algorithms in Business and their Supportive Role in Decision Making," <http://business.kent.edu/~tbodnovi/ga.html>, November.
- Caldwell, C. and V. S. Johnston (1991). "Tracking Criminal Suspect Through 'Face-space' with a Genetic Algorithm," in Belew and Booker (eds.), *Proceedings of the Fourth International Conference on Genetic Algorithms*, 416-421.
- Colin, A. M. (1994). "Genetic Algorithms for Financial Modeling," in Deboeck (ed.), *Trading on the Edge: Neural, Genetic, and Fuzzy Systems for Chaotic Financial Markets*, 148-173.
- Colin, A. M. (2000). "A Genetic-programming-based Approach to the Generation of Foreign-exchange Trading Models," in Barnett, Chiarella, Keen, Marks, and Schnabl (eds.), *Commerce, Complexity, and Evolution: Topics in Economics, Finance, Marketing, and Management: Proceedings of the Twelfth International Symposium in Economic Theory and Econometrics*, 173-189.
- Davis, L. (ed., 1991). *Handbook of Genetic Algorithms*. New York: Van Nostrand Reinhold.
- Davis, L. (1994). "Genetic Algorithms and Financial Applications," in Deboeck (ed.), *Trading on the Edge: Neural, Genetic, and Fuzzy Systems for Chaotic Financial Markets*, 133-147.
- Davis, L. and S. Coombs (1987). "Genetic Algorithms and Communication Link Speed Design: Theoretical Considerations," in Grefenstette (ed.), *Genetic Algorithms and their Applications: Proceedings of the Second International Conference on Genetic Algorithms*, 252-256.
- Deboeck, G. J. (ed., 1994). *Trading on the Edge: Neural, Genetic, and Fuzzy Systems for Chaotic Financial Markets*. New York: Wiley.
- Eddelbüttel, D. (1996a). "A Genetic Algorithm for Passive Management: Creating Index Funds with Fewer Stocks," <http://rosebud.sps.queensu.ca/~edd/papers.html> 26 December.
- Eddelbüttel, D. (1996b). "A Hybrid Genetic Algorithm for Passive Management," <http://rosebud.sps.queensu.ca/~edd/papers.html> 26 December.

## REFERENCES

- Feldman, K. and P. Treleaven (1994). "Intelligent Systems in Finance," *Applied Mathematical Finance*, 1(2), December, 195-207.
- Fogel, D. B. (1995). *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*. New York: IEEE Press.
- Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Reading, Massachusetts: Addison-Wesley.
- Goldberg, D. E. (1994). "Genetic and Evolutionary Algorithms Come of Age," *Communications of the ACM*, 37(3), March, 113-119.
- Goldberg, D. E. and K. Deb (1991). "A Comparative Analysis of Selection Schemes Used in Genetic Algorithms," in Rawlins (ed.), *Foundations of Genetic Algorithms*, 69-93.
- Goldberg, D. E. and R. Lingle Jr. (1985). "Alleles, Loci, and the Traveling Salesman Problem," in Grefenstette (ed.), *Proceedings of an International Conference on Genetic Algorithms and their Applications*, 154-159.
- Grefenstette, J. J. (ed., 1985). *Proceedings of an International Conference on Genetic Algorithms and their Applications*. Hillsdale: Erlbaum.
- Grefenstette, J. J. (ed., 1987). *Genetic Algorithms and their Applications: Proceedings of the Second International Conference on Genetic Algorithms*. Hillsdale: Lawrence Erlbaum.
- Grefenstette, J. J., R. Gopal, B. Rosmaita, and D. V. Gucht (1985). "Genetic Algorithms for the Traveling Salesman Problem," in Grefenstette (ed.), *Proceedings of an International Conference on Genetic Algorithms and their Applications*, 160-168.
- Hettketter, J. and D. Beasley (1996). "Q1: What Are Evolutionary Algorithms (EAs)?" <http://www.cis.ohio-state.edu/hypertext/faq/usenet/ai-faq/genetic/part2/faq-doc-1.html>
- Holland, J. H. (1992a). "Genetic Algorithms," *Scientific American*, July, 44-50.
- Holland, J. H. (1992b). *Adaptation in Natural and Artificial Systems*, 2nd ed. Cambridge: MIT Press.
- Hughes, M. (1990). "Improving Products and Processes—Nature's Way," *Industrial Management & Data Systems*, 6, 22-25.
- Kennedy, S. (1996). "Introduction to Genetic Algorithms," <http://www.axcelis.com:80/articles.html>
- Kingdon, J. and K. Feldman (1995). "Genetic Algorithms and Applications to Finance," *Applied Mathematical Finance*, 2(2), 89-116.
- Koza, J. R. (1992). "The Genetic Programming Paradigm: Genetically Breeding Populations of Computer Programs to Solve Problems," in



- Soucek and IRIS Group, *Dynamic, Genetic, and Chaotic Programming: The Sixth Generation*, 203–321.
- Koza, J. R. (1993). *Genetic Programming*. Cambridge: MIT Press.
- Leinweber, D. J. and R. D. Arnott (1995). "Quantitative and Computational Innovation in Investment Management," *The Journal of Portfolio Management*, 21(2), Winter, 8–15.
- Michalewicz, Z. (1994). *Genetic Algorithms + Data Structures = Evolution Programs*, 2nd ed. Berlin, Heidelberg, New York: Springer-Verlag.
- Mitchell, M. (1996). *An Introduction to Genetic Algorithms*. Cambridge: MIT Press.
- O'Leary, D. E. and P. R. Watkins (eds., 1992). *Expert Systems in Finance*. Amsterdam and New York: North-Holland.
- Pereira, R. (1996). "Selecting Parameters for Technical Trading Rules Using Genetic Algorithms," *Journal of Applied Finance and Investment*, 1(3), July/August, 27–34.
- Rawlings, G. J. E. (ed., 1991). *Foundations of Genetic Algorithms*. San Mateo: Morgan Kaufmann.
- Soucek, B. and IRIS Group (1992). *Dynamic, Genetic, and Chaotic Programming: The Sixth Generation*. New York: John Wiley.
- Syswerda, G. (1991). "Schedule Optimization Using Genetic Algorithms," in Davis (ed.), *Handbook of Genetic Algorithms*, 332–349.

## Chapter 3

# GENETIC PROGRAMMING: A TUTORIAL WITH THE SOFTWARE SIMPLE GP

Shu-Heng Chen

AI-ECON Research Center, Department of Economics, National Chengchi University  
Taipei, Taiwan 11623  
chchen@nccu.edu.tw

Tzu-Wen Kuo

AI-ECON Research Center, Department of Economics, National Chengchi University  
Taipei, Taiwan 11623  
kuo@atecon.org

Yuh-Pyng Shieh

Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan 106  
arpyng@turing.csie.ntu.edu.tw

## Abstract

This chapter demonstrates a computer program for tutoring genetic programming (GP). The software, called **Simple GP**, is developed by the **AI-ECON Research Center** at National Chengchi University, Taiwan. Using this software, the instructor can help students without programming background to quickly grasp some essential elements of GP. Along with the demonstration of the software is a list of key issues regarding the effective design of the implementation of GP. Some of the issues are already well noticed and studied by financial users of GP, but some are not. While many of the issues do not have a clear-cut answer, the attached software can help beginners to tackle those issues on their own. Once they have a general grasp of how to implement GP effectively, many advanced materials prepared in this volume are there for further exploration.