7

Calibration of the *VGSSD* Option Pricing Model using a Quantum-inspired Evolutionary Algorithm

Kai Fan[1,2], Conall O'Sullivan[2], Anthony Brabazon[1], Michael O'Neill[1], and Seán McGarraghy[2]

[1] Natural Computing Research and Applications Group,
University College Dublin, Ireland.
kai.fan@ucd.ie; anthony.brabazon@ucd.ie; m.oneill@ucd.ie
[2] School of Business, University College Dublin, Ireland.
conall.osullivan@ucd.ie; sean.mcgarraghy@ucd.ie

Summary. Quantum effects are a natural phenomenon and just like evolution, the brain, or immune systems, can serve as an inspiration for the design of computing algorithms. This chapter illustrates how a quantum-inspired evolutionary algorithm (QIEA) using real number encodings can be constructed and examines the utility of the resulting algorithm on an important real-world problem, namely the calibration of an Option Pricing model. The results from the algorithm are shown to be robust and comparable to those of other algorithms, suggesting that there is useful potential to apply QIEA to this domain.

7-1 Introduction

This chapter introduces a novel option pricing calibration methodology which uses a quantum-inspired real number encoding rather than a traditional encoding representation in an evolutionary algorithm. The chapter also assesses the utility of the resulting algorithm for the purposes of calibrating an option pricing model.

Quantum mechanics is an extension of classical mechanics which models behaviours of natural systems that are observed particularly at very short time or distance scales. An example of such a system is a sub-atomic particle, such as a free electron. A complex-valued (deterministic) function of time and space co-ordinates, called the *wave-function*, is associated with the system: it describes the *quantum state* the system is in. The standard interpretation of Quantum Mechanics is that this abstract wave-function allows us to calculate probabilities of outcomes of concrete experiments. The squared modulus of the wave-function is a probability density function (PDF): it describes the

probability that an observation of, for example, a particle will find the particle at a given time in a given region of space. The wave-function satisfies the *Schrödinger equation*. This equation can be thought of as describing the time evolution of the wave-function — and so the PDF — at each point in space: as time goes on, the PDF becomes more "spread out" over space, and our knowledge of the position of the particle becomes less precise, until an observation is carried out; then, according to the usual interpretation, the wave-function "collapses" to a particular classical state (or *eigenstate*), in this case a particular position, and the spreading out of the PDF starts all over again.

Before the observation we may regard the system as being in a linear combination of all possible classical states (this is called *superposition of states*); then the act of observation causes one such classical state to be chosen, with probability given by the PDF. Note that the wave function may interfere with itself (for example, if a barrier with slits is placed in the "path" of a particle) and this interference may be constructive or destructive, that is, the probability of detecting a particle in a given position may go up or go down.

More generally, we may seek to observe properties of quantum systems other than position, e.g., energy, momentum, or the quantum *spin* of an electron, photon or other particle. [Spin is incorporated by necessity in Dirac's relativistic extension of the wave equation; in fact spin is one of the arguments of the wave-function.] Such properties are called *observables*. Observables may be either continuous (e.g., position of a particle) or discrete (e.g., the energy of an electron in a bound state in an atom). Some observables may only take finitely many values, e.g., there are only two possible values for a given particle's spin: "up" or "down". This last is an example of a *two-state system*: in such a system the quantum state ψ is a linear superposition of just two eigenstates, say $|0\rangle$ and $|1\rangle$ in the standard Dirac bra-ket notation, that is,

$$\psi = \alpha|0\rangle + \beta|1\rangle,$$

where α and β are complex numbers with $|\alpha|^2 + |\beta|^2 = 1$. Here $|0\rangle$ and $|1\rangle$ are basis vectors for a 2-dimensional complex Hilbert space. A two-state system where the states are normalised and orthogonal, as here, may be regarded as a *quantum bit* or *qubit*.[3] It is thought of as being in eigenstates $|0\rangle$ and $|1\rangle$ simultaneously, until an observation is made and the quantum state collapses to $|0\rangle$ (with probability $|\alpha|^2$) or $|1\rangle$ (with probability $|\beta|^2$). The relation $|\alpha|^2 + |\beta|^2 = 1$ captures the fact that precisely one of $|0\rangle$, $|1\rangle$ must be observed, so their probabilities of observation must sum to 1.

A *quantum computer* is one which works with qubits instead of the (classical) bits used by usual computers. Benioff [1] first considered a Turing machine which used a tape containing what we would call qubits. Feynman [10] devel-

[3] Geometrically, a qubit is a compact 2-dimensional complex manifold, called the Bloch sphere.

oped examples of physical computing systems not equivalent to the standard model of deterministic computation, the Turing machine.

In recent years there has been a substantial interest in the theory and design of quantum computers, and the design of programs which could run on such computers, stimulated by Shor's discovery of a quantum factoring algorithm which would run faster than possible clasically. One interesting strand of research has been the use of natural computing (for example Genetic Programming (GP)) to generate quantum circuits or programs (algorithms) for quantum computers [25]. (Genetic programming is an evolutionary algorithm based methodology inspired by biological evolution to find computer programs that perform a user-defined task. Therefore it is a machine learning technique used to optimize a population of computer programs according to a fitness landscape determined by a program's ability to perform a given computational task.) There has also been associated work in a reverse direction which draws inspiration from concepts in quantum mechanics in order to design novel natural computing algorithms. This is currently an area of active research interest. For example, quantum-inspired concepts have been applied to the domains of evolutionary algorithms [23, 12, 14, 27, 28], social computing [29], neuro-computing [18, 11, 26], and immuno-computing [19, 16]. A claimed benefit of these algorithms is that because they use a quantum representation, they can maintain a good balance between exploration and exploitation. It is also suggested that they offer computational efficiencies as use of a quantum representation can allow the use of smaller population sizes than typical evolutionary algorithms.

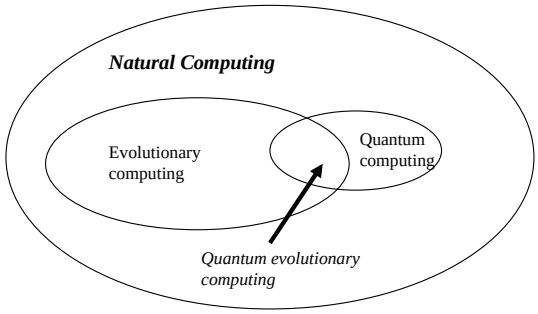


Fig. 7-1. Quantum-inspired evolutionary computing

Quantum-inspired evolutionary algorithms (QIEA) offer interesting potential. As yet, due to their novelty, only a small number of recent papers have implemented a QIEA, typically reporting good results [27, 28]. Consequently, we have a limited understanding of the performance of these algorithms and further testing is required in order to determine both their effectiveness and

their efficiency. It is also noted that although a wide-variety of biologically-inspired algorithms have been applied for financial modelling [3], the QIEA methodology has not yet been applied to the finance domain. This study addresses both of these research gaps.

Structure of Chapter

The rest of this chapter is organised as follows. The next section provides a concise overview of QIEA, concentrating on the quantum-inspired genetic algorithm. We then outline the experimental methodology adopted. The remaining sections provide the results of these experiments followed by a number of conclusions.

7-2 The Quantum-inspired Genetic Algorithm

The best-known application of quantum-inspired concepts in evolutionary computing is the quantum-inspired genetic algorithm (QIGA) [23, 12, 14, 27, 28]. The QIGA is based on the concepts of a qubit and the superposition of states. In essence, in QIGAs the traditional representations used in evolutionary algorithms (binary, numeric and symbolic) are extended to include a quantum representation.

A crucial difference between a qubit and a (classical) bit is that multiple qubits can exhibit quantum entanglement. Entanglement is when the wave function of a system composed of many particles cannot be separated into independent wave functions, one for each particle. A measurement made on one particle can produce, through the collapse of the total wavefunction, an instantaneous effect on other particles with which it is entangled, even if they are far apart. Entanglement is a nonlocal property that allows a set of qubits to be highly correlated. Entanglement also allows many states to be acted on simultaneously, unlike bits that can only have one value at a time. The use of entanglement in quantum computers is sometimes called *quantum parallelism*, and gives a possible explanation for the power of quantum computing: because the state of the quantum computer (i.e., the state of the system considered as a whole) can be in a quantum superposition of many different classical computational states, these classical computations can all be carried out at the same time.

The quantum equivalent of a classical operator on bits is an *evolution* (not to be confused with the evolution of EAs). It transforms an input to an output, e.g., by rotation or Hadamard gate, and operates without measuring the value of the qubit(s). Thus it effectively does a parallel computation on all the qubits at once and gives rise to a new superposition.

In the language of evolutionary computation a system of m qubits may be referred to as a *quantum chromosome* and can be written as a matrix with two rows:

$$\begin{bmatrix} \alpha_1 & \alpha_2 & \dots & \alpha_m \\ \beta_1 & \beta_2 & \dots & \beta_m \end{bmatrix}. \tag{7.1}$$

A key point when considering quantum systems is that they can compactly convey information on a large number of possible system states. In classical bit strings, a string of length m can represent 2^m possible states. However, a quantum space of m qubits has 2^m *dimensions* (as a complex manifold).[4] Thus, a single qubit register of length m can simultaneously represent *all* possible bit strings of length 2^m, e.g., an 8 qubit system can simultaneously encode 256 distinct strings. This implies that it is possible to modify standard evolutionary algorithms to work with very few, or even a single quantum individual, rather than having to use a large population of solution encodings. The qubit representation can also help to maintain diversity during the search process of an evolutionary algorithm, due to its capability to represent multiple system states simultaneously.

7-2.1 Representing a Quantum System

There are many ways that a quantum system could be defined in order to encode a set of binary (solution) strings. For example, in the following 3 qubit quantum system, the quantum chromosome is defined using the three pairs of amplitudes below

$$\begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{\sqrt{3}}{2} & \frac{1}{2} \\ \frac{1}{\sqrt{2}} & \frac{1}{2} & \frac{\sqrt{3}}{2} \end{bmatrix} \tag{7.2}$$

These numbers are the probabilities that a qubit (unit of information) will be observed in a particular eigenstate rather than another. Taking the first qubit, the occurrence of either state 0 or 1 is equally likely as both α_1 and β_1 have the same amplitude. Following on from the definition of the 3 qubit system, the (quantum) state of the system is given by

$$\tfrac{\sqrt{3}}{4\sqrt{2}}|000\rangle+\tfrac{3}{4\sqrt{2}}|001\rangle+\tfrac{1}{4\sqrt{2}}|010\rangle+\tfrac{\sqrt{3}}{4\sqrt{2}}|011\rangle+\tfrac{\sqrt{3}}{4\sqrt{2}}|100\rangle+\tfrac{3}{4\sqrt{2}}|101\rangle+\tfrac{1}{4\sqrt{2}}|110\rangle+\tfrac{\sqrt{3}}{4\sqrt{2}}|111\rangle \tag{7.3}$$

To provide intuition on this point, consider the system state $|000\rangle$. The associated probability amplitude for this state is $\frac{\sqrt{3}}{4\sqrt{2}}$ and this is derived from the probability amplitudes of the 0 state for each of the three individual qubits ($\frac{1}{\sqrt{2}} * \frac{\sqrt{3}}{2} * \frac{1}{2} = 0.25$). The associated probabilities of each of the individual states ($|000\rangle, |001\rangle, |010\rangle, |011\rangle, |100\rangle, |101\rangle, |110\rangle, |111\rangle$) are $\frac{3}{32}, \frac{9}{32}, \frac{1}{32}, \frac{3}{32}, \frac{3}{32}, \frac{9}{32}, \frac{1}{32}, \frac{3}{32}$ respectively. Taking the first of these states as an example, $(\frac{\sqrt{3}}{4\sqrt{2}})^2 = \frac{3}{32}$.

[4] It can be shown that, because of entanglement, an m-qubit physical system has $2^{m+1} - 2$ degrees of freedom, much larger than the $2m$ degrees a classical version would have.

7-2.2 Real-valued quantum-inspired evolutionary algorithms

In the initial literature which introduced the QIGA, a binary representation was adopted, wherein each quantum chromosome was restricted to consist of a series of 0s and 1s. The methodology was modified to include real-valued vectors by da Cruz et al., [9]. As with binary-representation QIGA, real-valued QIGA maintains a distinction between a quantum population and an observed population of, in this case, real-valued solution vectors. However the quantum individuals have a different form to those in binary-representation QIGA. The quantum population $Q(t)$ is comprised of N quantum individuals $(q_i : i = 1, 2, 3, \ldots, N)$, where each individual i is comprised of G genes $(g_{ij} : j = 1, 2, 3, \ldots, G)$. Each of these genes consist of a pair of values $q_{ij} = (p_{ij}, \sigma_{ij})$ where $p_{ij}, \sigma_{ij} \in \Re$ represent the mean and the width of a square pulse. Representing a gene in this manner has a parallel with the quantum concept of superposition of states as a gene is specified by a range of possible values, rather than by a single unique value.

The original QIGA algorithms, e.g., [12, 14] are based very closely on physical qubits, but the "quantum-inspired" algorithm of da Cruz et al. [9] used in this chapter draws less inspiration from quantum mechanics since it:

- does not use the idea of a quantum system (in particular, no qubits);
- only allows for constructive (not destructive) interference, and that interference is among "wave-functions" of *different* individuals;
- uses real numbers as weights, rather than the complex numbers which arise in superposition of states in physical systems;
- the PDFs used (uniform distributions) are not those arising in physical systems.

However, the da Cruz et al algorithm does periodically sample from a distribution to get a "classical" population, which can be regarded as a wave-function (quantum state) collapsing to a classical state upon observation.

Algorithm

The real-valued QIGA algorithm is as follows

```
Set t=0

Initialise Q(t) of N individuals with G genes

While (t < max t)
    Create the PDFs (and corresponding CDFs, which describe the probability
        distributions of real-valued random variables, see equation(6)) for
        each gene locus using the quantum individuals
    Create a temporary population, denoted E(T), of K real-valued solution
        vectors by observing Q(t) (via the CDFs)

    If (t=0) Then C(t)=E(t)
    (Note: the population C(T) is maintained between iterations of the algorithm)
    Else    E(t)=Outcome of crossover between E(t) and C(t)
            Evaluate E(t)
            C(t)= K best individuals from E(t) U C(t)
```

```
    End if

    With the N best individuals from C(t)
    Q(t+1)=Output of translate operation on Q(t)
    Q(t+1)=Output of resize operation on Q(t+1)
    t=t+1
Endwhile
```

Initialising the Quantum Population

A *quantum chromosome*, which is observed to give a specific solution vector of real-numbers, is made up of several quantum genes. The number of genes is determined by the required dimensionality of the solution vector. At the start of the algorithm, each quantum gene is initialised by randomly selecting a value from within the range of allowable values for that dimension. A gene's width value is set to the range of allowable values for the dimension. For example, if the known allowable values for dimension j are $[-75, 75]$ then q_{ij} (dimension j in quantum chromosome i) is initially determined by randomly selecting a value from this range (say) -50. The corresponding width value will be 150. Hence, $q_{ij} = (-50, 150)$. The square pulse need not be entirely within the allowable range for a dimension when it is initially created as the algorithm will automatically adjust for this as it executes. The height of the pulse arising from a gene j in chromosome i is calculated using

$$h_{ij} = \frac{1/\sigma_{ij}}{N} \tag{7.4}$$

where N is the number of individuals in the quantum population. This equation ensures that the probability density functions (PDFs) (see next subsection) used to generate the observed individual solution vectors will have a total area equal to one. Fig. 7-2 provides an illustration of a quantum gene where N=4.

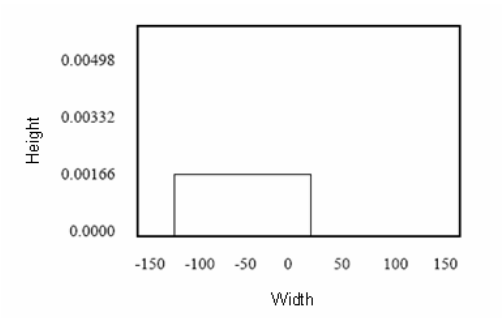


Fig. 7-2. A square pulse, representing a quantum gene, with a width of 150, centred on -50. The height of the pulse is 0.001666̇

Observing the Quantum Chromosomes

In order to generate a population of real-valued solution vectors, a series of observations must be undertaken using the population of quantum chromosomes (individuals). A pseudo-interference process between the quantum individuals is simulated by summing up the square pulses for each individual gene across all members of the quantum population. This generates a separate PDF (just the sum of the square pulses) for each gene and eq. 7.4 ensures that the area under this PDF is one. Hence, the PDF for gene j on iteration t is

$$PDF_j(t) = \sum_{i}^{j} g_{ij} \tag{7.5}$$

where g_{ij} is the square pulse of the j^{th} gene of the i^{th} quantum individual (of N). To use this information to obtain an observation, the PDF is first converted into its corresponding Cumulative Distribution Function (CDF)

$$CDF_j(x) = \int_{L_j}^{U_j} PDF_j(x)dx \tag{7.6}$$

where U_j and L_j are the upper and lower limits of the probability distribution. By generating a random number r from (0,1), the CDF can be used to obtain an observation of a real number x, where $x = CDF^{-1}(r)$. If the generated value x is outside the allowable real valued range for that dimension, the generated value is limited to its allowable boundary value. A separate PDF and CDF is calculated for each of the G gene positions. Once these have been calculated, the observation process is iterated to create a temporary population with K members, denoted E(t).

Crossover Mechanism

The crossover operation takes place between C(t) and the temporary population E(t). This step could be operationalised in a variety of ways with [9] choosing to adopt a variant of uniform crossover, without an explicit selection operator. After the K crossover operations have been performed, with the resulting children being copied into E(t), the best K individuals $\in C(t) \cup E(t)$ are copied into C(t).

Updating the Quantum Chromosomes

The N quantum chromosomes are updated using the N best individuals from C(t) after performing the crossover step. Each quantum gene's mean value is altered using

$$p_{ij} = c_{ij} \tag{7.7}$$

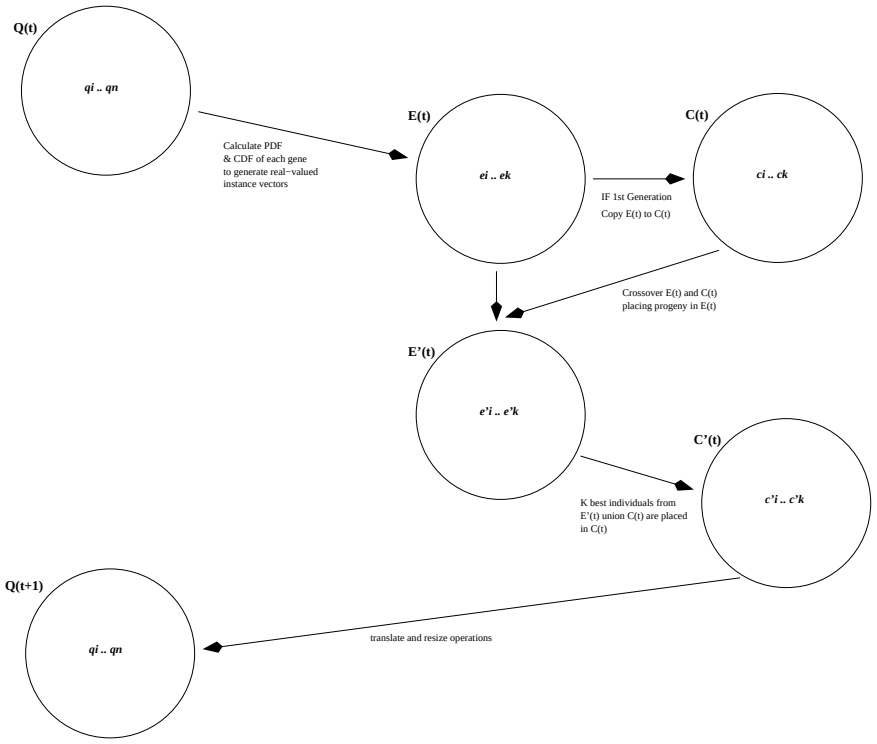


Fig. 7-3. An illustration of the process in the creation of generation t+1 from t.

so that the mean value of the j^{th} gene of the i^{th} quantum chromosome is given by the corresponding j^{th} value of the i^{th} ranked individual in $C(t)$.

The next step is to update the corresponding width value of the j^{th} gene. The objective of this process is to vary the exploration / exploitation characteristics of the search algorithm, depending on the feedback from previous iterations. If the search process is continuing to uncover many new better solutions, then the exploration phase should be continued by keeping the widths relatively broad. However, if the search process is not uncovering many new better solutions, the widths are reduced in order to encourage finer-grained search around already discovered good regions of the solution space. There are multiple ways this general approach could be operationalised. For example, [9] suggests use of the 1/5th mutation rule from Evolutionary Strategies [24] whereby

$$if\ \phi < 1/5 \text{ then } \sigma_{ij} = \sigma_{ij} g$$

$$if\ \phi > 1/5 \text{ then } \sigma_{ij} = \sigma_{ij} / g$$

$$if\ \phi = 1/5 \text{ then } \sigma_{ij} = \sigma_{ij}$$

where σ_{ij} is the width of the i^{th} quantum chromosome's j^{th} gene, g is a constant in the range $[0, 1]$ and ϕ is the proportion of individuals in the new population that have improved their fitness.

In this study we update the width of the i^{th} quantum chromosome's j^{th} gene by comparing each successive generation's best fitness function. If the best fitness function has improved (disimproved) we shrink (enlarge) the width in order to improve the local (global) search.

QIGA vs Canonical Genetic Algorithm

A number of distinctions between the QIGA above and the canonical GA (CGA) can be noted. In the CGA, the population of solutions persists from generation to generation, albeit in a changing form. In contrast, in QIGA, the population of solutions in $P(t)$ are discarded at the end of each loop. The described QIGA, unlike CGA, does not have explicit concepts of crossover or mutation. However, the adaptation of the quantum chromosomes in each iteration does embed implicit selection as the best solution is selected and is used to adapt the quantum chromosome(s). The crossover and mutation steps are also implicitly present, as the adaptation of the quantum chromosome in effect creates diversity, as it makes different states of the system more or less likely over time. Another distinction between the QIGA and the CGA is that the CGA operates directly on representations of the solution (the members of the current population of solutions), whereas in QIGA the update step is performed on the probability amplitudes of the ground states for each qubit making up the quantum chromosome(s).

7-3 Option Pricing Model Calibration

An optimisation problem in financial modelling is considered to test the performance of the QIGA. The optimisation involves calibrating an option pricing model to observed market data. Calibration is a method of choosing model parameters so that the distance between a set of model option prices and market option prices is minimised, where distance is some metric such as the sum of squared errors or the sum of squared percentage errors. The parameters can be thought to resemble the market's view on current option prices and the underlying asset price. In calibration we do not explicitly take into account any historical data. All necessary information is contained in today's option prices which can be observed in the market. Practitioners frequently calibrate option pricing models so that the models provide reasonable fits to current observed market option prices and they then use these models to price exotic derivatives or for hedging purposes. In this paper we calibrate a very recent

extension of the *Variance Gamma* option pricing model [20, 21, 22] known as the *Variance Gamma Scaled Self-Decomposable* (*VGSSD*) model [5] to FTSE 100 index option data.

A European call (put) option on an asset S_t with maturity date T and strike price K is defined as a contingent claim with payoff at time T given by $\max[S_T - K, 0]$ ($\max[K - S_T, 0]$). The well known Black-Scholes (BS) formula for the price of a call at time t on this asset is given by

$$C_{BS}(S_t, K, r, q, \tau; \sigma) = S_t e^{-q\tau} N(d_1) - K e^{-r\tau} N(d_1) \tag{7.8}$$

$$d_1 = \frac{-\ln m + \left(r - q + \frac{1}{2}\sigma^2\right)\tau}{\sigma\sqrt{\tau}} \qquad d_2 = d_1 - \sigma\sqrt{\tau} \tag{7.9}$$

where $\tau = T - t$ is the time-to-maturity, t is the current time, $m = K/S$ is the moneyness of the option, r and q are the continuously compounded risk-free rate and dividend yield and $N(\cdot)$ is the cumulative normal distribution function. Suppose a market option price, denoted by $C_M(S_t, K)$, is observed. The Black-Scholes implied volatility for this option price is that value of volatility which equates the BS model price to the market option price as follows

$$\begin{aligned} \sigma_{BS}(S_t, K) &> 0 \\ C_{BS}(S_t, K, r, \tau; \sigma_{BS}(S_t, K)) &= C_M(S_t, K) \end{aligned} \tag{7.10}$$

If the assumptions underlying the BS option pricing model were correct, the BS implied volatilities for options on the same underlying asset would be constant for different strike prices and maturities. However in reality the BS implied volatilities are varying over strike price and maturity. Given that the options are written on a single underlying asset this result seems at first paradoxical, i.e. we have a number of different implied volatilities for a single asset which should only have one measure for its volatility. Yet if we relax some of the assumptions in the BS model, such as allowing for a more complex data generating process for the asset price than the log normal stochastic process (as assumed by BS), and take into account the resulting complications, this result begins to make sense and is simply highlighting the erroneous assumptions that underpin the BS model.

Many different option pricing models have been proposed as alternatives to the BS model. Examples include stochastic volatility models and jump diffusion models which allow for more complex asset price dynamics. We examine a simple extension of a very popular option pricing model known as the Variance Gamma (VG) option pricing model. The extension of the model is called the Variance Gamma Scaled Self-Decomposable (*VGSSD*) model. The idea of the VG process is to model the continuously compounded returns of the stock price occurring on business time rather than on calendar time using a time transformation of a Brownian motion. The resulting model is a three parameter model where roughly speaking we can interpret the parameters as

controlling volatility, skewness and kurtosis, denoted respectively as σ, θ and ν, of the underlying asset returns distribution. Closed form option pricing formulae exist under the VG model [22]. The model performs well at fitting a range of options prices with different strike prices at one maturity but fails to fit option prices at several maturities. This is because the returns are independent in the VG model and this results in skewness and excess kurtosis of the VG returns density function approaching zero too quickly as maturity increases. This resulted in [5] proposing a number of alternative models, one of which is based on the VG model. The model proposes that a *VGSSD* random variable at time τ has the same distribution as a VG random variable at unit time multiplied by τ^γ, where γ is an additional parameter of the model that induces volatility clustering in the returns. The resulting random process has a scaled density function, i.e. skewness and excess kurtosis remain constant as time-to-maturity increases. The *VGSSD* model performs much better than the VG model at calibrating to a range of options across both strike price and maturity at the expense of one additional parameter. The characteristic function (Fourier transform of its density function) of the *VGSSD* process, X_τ, has a very simple form and is given by

$$\phi_{X_\tau}(u) = E[\exp(\mathrm{i}uX_\tau)] = \left(\frac{1}{1 - \mathrm{i}u\nu\theta\tau^\gamma + \frac{1}{2}u^2\nu\sigma^2\tau^{2\gamma}}\right)^{\frac{1}{\nu}} \tag{7.11}$$

where u is a Fourier transform parameter and i is the imaginary number with $\mathrm{i} = \sqrt{-1}$. The characteristic function of the logarithm of the stock price is given by

$$\phi_{\ln S_T}(u) = \exp\left\{\mathrm{i}u\left(\ln S_t + (r-q)\tau - \ln \phi_{X_\tau}(-\mathrm{i})\right)\right\}\phi_{X_\tau}(u) \tag{7.12}$$

This form ensures that the expectation of the future stock price in the risk neutral world is given by $E[S_T] = S_t \exp((r-q)\tau)$ where we recall that $\tau = T - t$. Option prices can be computed using the well known fast Fourier transform approach of [4]. A call option price with strike price K and time-to-maturity τ is given by

$$c(K,\tau) = \frac{\exp(-\alpha \ln(K))}{\pi}\int_0^{+\infty} \exp(-\mathrm{i}v\ln(K))\,\psi_\tau(v)\,\mathrm{d}v \tag{7.13}$$

where

$$\psi_\tau(v) = \frac{\exp(-r\tau)\,\phi_{\ln S_T}(v - (\alpha+1)\,\mathrm{i})}{\alpha^2 + \alpha - v^2 + \mathrm{i}(2\alpha+1)v} \tag{7.14}$$

α is a dampening parameter that must be included for numerical reasons, it is set to $\alpha = 4$ in this analysis, and the integral in equation 7.13 is computed numerically using a fast Fourier transform (FFT), see [4] for more details.

7-4 Experimental Approach

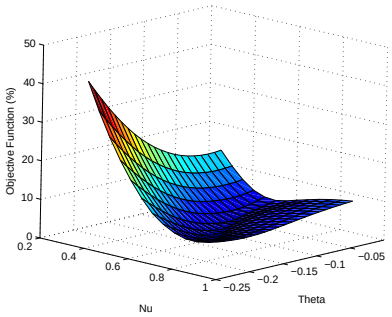
Market makers in the options markets quote BS implied volatilities rather than option prices even though they realise BS is a flawed model. Table 7-1 depicts end-of-day settlement Black-Scholes implied volatilities for FTSE 100 European options on the 17 March 2006 for different strike prices and time-to-maturities. As can be seen the BS implied volatilities are not constant across the strike price and the maturity date. These implied volatilities are converted into market call and put prices by substituting the BS implied volatilities into the Black-Scholes formula. The following input parameters were used to calculate the option prices, the index price is the FTSE 100 index itself $S_t = 5999.4$, the interest rate is the one month *Libor* rate converted into a continuously compounded rate $r = 0.0452$ and the dividend yield is a continuously compounded dividend yield downloaded from *Datastream* and is $q = 0.0306$. These prices are then taken to be the observed market option prices. Out-of-the money (OTM) put prices were used for $K < S$ and OTM call prices were used for $K > S$ in the calibration. The calibration problem now amounts to choosing an optimum parameter vector $\Theta = \{\sigma, \nu, \theta, \gamma\}$ such that an objective function $G(\Theta)$ is minimised. In this paper the objective function is chosen to be the absolute average percentage error (APE)

$$\mathrm{G}(\Theta) = \frac{1}{N}\sum_{i=1}^{N}\left|\frac{C_i - C_i(\Theta)}{C_i}\right|$$

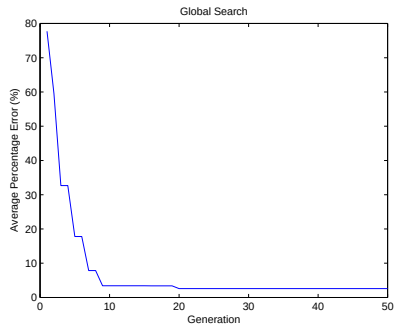
where C_i is the observed market price on the i-th option (could be a call or a put) and $C_i(\Theta)$ is the *VGSSD* model price of the i-th option with parameter vector Θ. One of the difficulties in model calibration is that the available market information may be insufficient to completely identify the parameters of a model [6]. If the model is sufficiently rich relative to the number of market prices available, a number of possible parameter vector combinations will be compatible with market prices and the objective function $G(\Theta)$ may not be convex function of Θ. A plot of the objective function versus the two parameters controlling skewness and kurtosis of the asset returns distribution, θ and ν, whilst keeping σ and γ fixed at $\sigma = 0.1175$ and $\gamma = 0.5802$ is shown in figure 7-4a. Although the objective function looks well behaved other studies show that there are some potential problems. A graph of a very similar objective function is plotted in [6], for the VG option pricing model, using DAX index option data and the potential for gradient based optimisers to converge to a local rather than the global minimum is illustrated due to the fact that certain parameters have off setting effects in the VG model. Options are priced using the VG model with FFTs by [15] who outlines the potential for option prices to explode to infinity for certain parameter values of the VG model where conditions on the parameter values that keep the characteristic function finite do not hold. These potential problems provide the motivation to use an evolutionary based optimiser for calibrating the *VGSSD* model.

Table 7-1. Market BS implied volatilities (%) for FTSE 100 index options on the 17 March 2006. The strike prices and maturities (in years) are given in the table and the other observable inputs are $S = 5999.4, r = 0.0452$ and $q = 0.0306$.

	Strike Price										
Maturity	4196.5	4796	5395.5	5695.2	5845.1	5995	6144.9	6294.7	6594	7194	7793.5
0.0959				13.76	12.41	11.13	10.44	10.94			
0.1726			15.43	13.27	12.20	11.14	10.39	10.32			
0.2493			15.21	13.28	12.32	11.37	10.59	10.22	10.26		
0.4986		18.43	15.22	13.65	12.87	12.10	11.42	10.98	10.76		
0.7479	21.24	18.28	15.45	14.07	13.38	12.70	12.06	11.55	10.96	10.76	
0.9973	20.98	18.27	15.69	14.42	13.80	13.18	12.57	12.06	11.35	10.85	
1.2466	20.85	18.33	15.94	14.76	14.18	13.60	13.04	12.54	11.79	11.09	10.80



a Objective function vs parameters



b Objective function vs generation no.

Fig. 7-4. Objective function versus model parameters ν and θ and objective function versus generation number.

7-5 Results

In all runs of the QIGA, a population size of 20 observed chromosomes was used, the algorithm was allowed to run for 30 generations, and all reported results are averaged over 30 runs. In order to provide a benchmark for the results obtained by the QIGA a deterministic Matlab optimiser called *fminsearch* was run 30 times with different initial parameter vectors. *Fminsearch*, multidimensional unconstrained nonlinear minimization, uses the Nelder-Mead simplex search method. This nonlinear optimization algorithm approximately finds a locally optimal solution to a problem with variables when the objective function varies smoothly, see Lagarias et al [17]. It is a direct search method that does not use numerical or analytic gradients. The optimiser converged to different values for the parameter vector Θ for different initialisations of Θ. The parameter vector associated with the optimal value for the objective function G was chosen to compute the average percentage error (APE) and the model prices. Figure 7-4b depicts the evolution of the global objective function G, measured using average percentage error (APE), as a function of the generation number for a single run of the algorithm. Figures 7-5a, 7-5b, 7-5c and 7-5d depict the evolution of the parameters σ, ν, θ and γ as a function of the generation number for a single run of the algorithm.

The best results from the last generation of the algorithm, averaged over 30 runs, are reported in the first column Table 7-2. The average results from the last generation of the algorithm, averaged over 30 runs, are reported in the second column of Table 7-2. As can be seen in Table 7-2 the optimal APE is very low at 1.29% (this compares favourably to the optimal matlab value of 1.27%) and the parameter values are very close to the parameter values from the Matlab optimiser(*fminsearch*).

The results from the second column of Table 7-2 are the average result from the last generation of the algorithm, averaged over 30 runs. The APE is calculated by computing a single set of model prices from the average parameter vector Θ. The APE is higher than the optimal values at 3.34% indicating that some of the runs converged too early to a suboptimal result. However the average parameter values are not that different from the optimal parameter values indicating that only a small number of the 30 runs gave poor results.

Model vs Market Implied Volatilities

In order to give a more complete picture of the results figure 7-6 depicts market and model implied volatilities versus the moneyness of the option, where moneyess is equal to $\frac{K}{S}$, for all the options used in the analysis. The model implied volatilities are calculated from the model prices using the optimal parameter values from Table 7-2. Model and market implied volatilities are plotted as opposed to model and market option prices because implied volatilities depict the calibration performance of the option pricing model in a much clearer way than option prices alone. This is because implied volatilities vary far less

with strike price and maturity than option prices themselves. The objective function depends on model option prices as opposed to model implied volatilities because optimising over implied volatilities would be very time consuming since implied volatility has to be calculated numerically given the option price and other input parameters. However, if the model option prices are very close to the market option prices, the model implied volatilities will also be very close to the market implied volatilities. Figure 7-6 outlines the success of the parsimonious *VGSSD* model in terms of calibrating to a wide range of option prices. The *VGSSD* model performs best at the shorter maturities but more flexibility would be needed to match the deep out-of-the-money put options at longer maturities. However for a four parameter model the calibration performance is still very promising.

Sensitivity Analysis

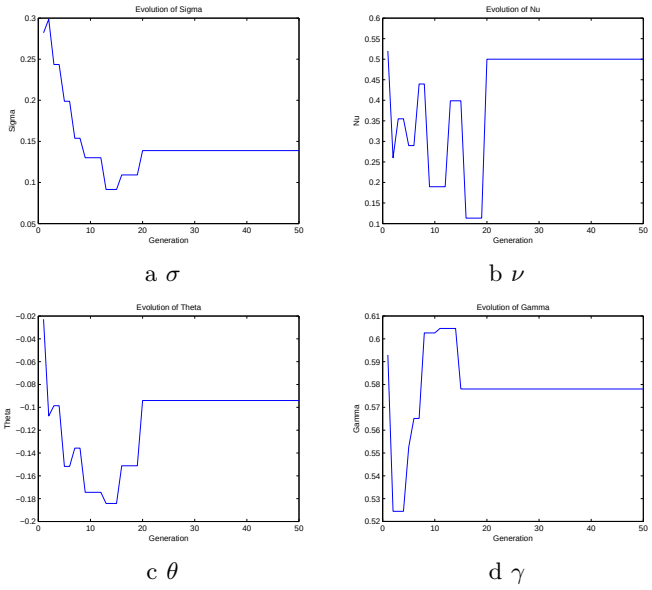
In previous expositions of the real-valued QIGA, detailed sensitivity analysis results were not reported. In order to gain greater insight into the operation of the algorithm, and to guide future applications of it, we undertook such an analysis by systematically investigating a variety of parameter settings for shrinkage, enlargement and crossover. The results of the optimal APE value as a function of the enlargement and shrink parameters are reported in table 7-3. The crossover rate is fixed at 0.3, a population size of 20 and a generation number of 50 are used. Figure 7-7a graphs these results.

The APE is less sensitive to the enlargement parameter than to the shrink parameter. The shrink parameter forces the algorithm to converge faster and this has a strong effect on the algorithms performance. The enlargement parameter causes the algorithm to widen the search space, however the crossover rate can also do this albeit using a different method, and this is why the algorithm is less sensitive to the enlargement parameter provided the crossover rate is a reasonable value. In this analysis the shrinkage parameter is set to 0.7 and the enlargement parameter is set to 1.2 as this provided a reasonable trade-off in the convergence speed of the algorithm versus the search space of the algorithm.

Table 7-4 reports the optimal APE value as a function of the shrink parameter and the crossover rate. The enlargement parameter is set to 1.2 and a population size of 20 and generation number of 50 are used. Figure 7-7b graphs these results. The algorithm performs more optimally with the crossover rate set to 0.3 and the shrink parameter set to 0.7. Low values of the crossover rate results in a smaller global search space and high values results in almost random search so intermediate values provide a reasonable trade-off between exploitation and exploration. Similarly the shrink parameter forces the algorithm to narrow the search space in the region of the current best solution, however this may be a local minima so the shrink parameter should not be made too small.

Table 7-2. Results of QIGA. The optimal and average parameter values from the last generation are averaged over 30 runs and compared with the parameter values from 30 runs of a Matlab optimiser.

Parameter	QIGA Optima	Mean	Standard deviation	Matlab optimisation
σ	0.1181	0.1034	0.0618	0.1175
ν	0.5000	0.4088	0.1058	0.4975
θ	-0.0941	-0.1067	0.0993	-0.1084
γ	0.5662	0.5231	0.1017	0.5802
APE (%)	1.29	3.34	1.04	1.27



a σ b ν

c θ d γ

Fig. 7-5. Evolution of parameters over time.

Table 7-3. This table reports the APE (%) for different enlargement and shrinkage values.

Enlarge\Shrink	0.5	0.6	0.7	0.8	0.9
1.1	34.44	22.34	4.18	2.85	2.53
1.2	3.71	5.53	3.74	10.34	4.59
1.3	4.14	16.06	6.75	4.56	17.54
1.4	5.39	12.20	3.19	3.57	3.71
1.5	54.56	3.87	2.51	7.16	4.85

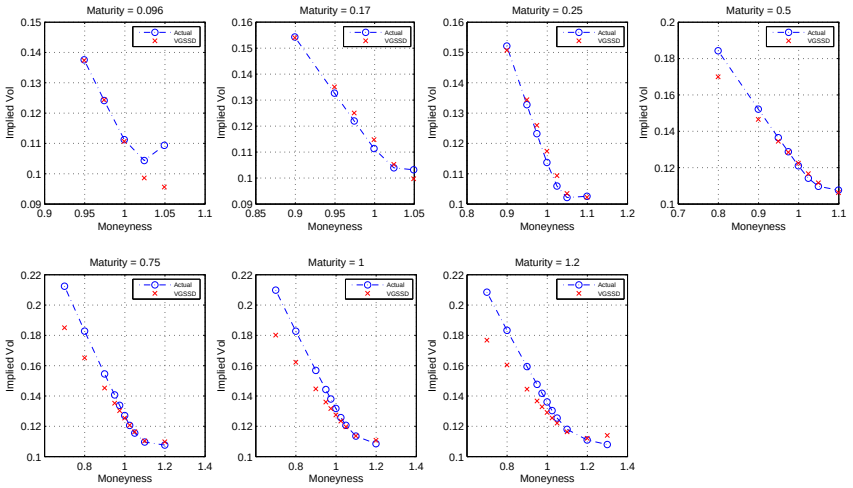
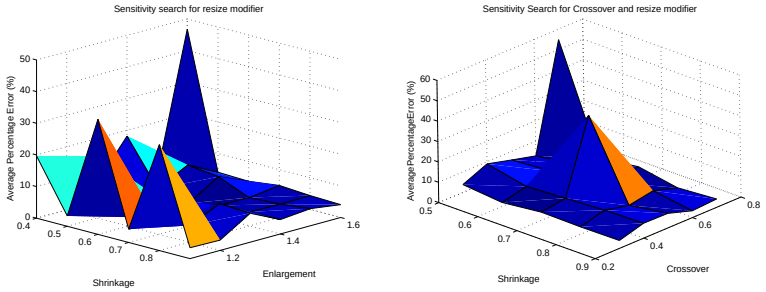


Fig. 7-6. Market and model implied volatilities.



a sensitivity b shrinkage and crossover

Fig. 7-7. Sensitivity search

Table 7-4. This table reports the APE (%) for different shrinkage and crossover values.

Crossover\Shrink	0.5	0.6	0.7	0.8	0.9
0.3	3.84	1.90	4.30	3.68	3.85
0.4	8.96	4.74	5.82	6.82	7.30
0.5	5.81	7.20	41.39	4.17	7.09
0.6	2.81	5.84	3.39	6.30	3.32
0.7	54.85	3.29	6.25	2.72	4.08

7-6 Conclusions

The real-valued QIEA is a novel form of representation which could be hybridised with multiple real-valued search algorithms apart from the GA such as PSO or DE. This chapter illustrates how a quantum-inspired evolutionary algorithm can be constructed and examines the utility of the resulting algorithm on an important problem in financial modelling known as model calibration. The results from the algorithm are shown to be robust and comparable to those of other algorithms.

This underpins earlier proof of concept exploration studies using real valued quantum-inspired evolutionary algorithm. It is also noted that this paper reports the first application of a QIGA to the financial domain. Several extensions of the methodology in this study are indicated for future work. The first extension would be to extend the real-valued QIGA to a higher dimensional setting in order to examine, and highlight, the computational benefits of QIGA. The algorithm offers substantial potential for calibrating more complex financial models than the *VGSSD* option pricing model. Future work could also assess the benefits of operationalising the key steps of the QIGA in alternative ways, for example, by implementing alternative diversity-generating mechanisms in updating C(t) or by implementing alternative mechanisms for altering σ during the algorithm.

References

1. Benioff, P. (1980). The Computer as a Physical System: A Microscopic Quantum Mechanical Hamiltonian Model of Computers as Represented by Turing Machines. *Journal of Statistical Physics*, 22, 563–591.
2. Black, F. and Scholes, M. (1973). The pricing of options and corporate liabilities, *Journal of Political Economy*, 81:37–654.
3. Brabazon, A. and O'Neill, M. (2006). *Biologically-inspired Algorithms for Financial Modelling*, Berlin: Springer.
4. Carr, P. and Madan, D.P. (1999). Option valuation using the fast Fourier transform. *Journal of Business*, 75:305–332.
5. Carr, P., H. Geman, D. Madan and M. Yor (2007). Self decomposability and options pricing. *Mathematical Finance*, 17 (1).
6. Cont, R. and Ben Hamida, S. (2005). Recovering volatility from option prices by evolutionary optimisation, *Journal of Computational Finance*, 8 (4), Summer 2005.
7. da Cruz, A., Barbosa, C., Pacheco, M. and Vellasco, M. (2004) Quantum-Inspired Evolutionary Algorithms and Its Application to Numerical Optimization Problems, *ICONIP*, 2004, pp:212-217
8. da Cruz, A., Pacheco, M., Vellasco, M. and Barbosa, C. (2005) Cultural Operators for a Quantum-Inspired Evolutionary Algorithm Applied to Numerical Optimization Problems, *IWINAC*
9. da Cruz, A., Vellasco, M. and Pacheco, M. (2006). Quantum-inspired evolutionary algorithm for numerical optimization, in *Proceedings of the 2006*

IEEE Congress on Evolutionary Computation (CEC 2006), 16-21 July, Vancouver, pp. 9180–9187, IEEE Press.

10. Feynman, R. (1982). Simulating Physics with Computers. *International Journal of Theoretical Physics*, 21 (6&7), 467–488.
11. Garavaglia, S. (2002). A quantum-inspired self-organizing map (QISOM), *Proceedings of 2002 International Joint Conference on Neural Networks* (IJCNN 2002),12-17 May 2002, pp. 1779–1784, IEEE Press.
12. Han, K-H. and Kim, J-H. (2002). Quantum-inspired evolutionary algorithm for a class of combinatorial optimization, *IEEE Transactions on Evolutionary Computation*, 6(6):580–593.
13. Han, K-H. and Kim, J-H. (2003). On setting the parameters of quantum-inspired evolutionary algorithm for practical applications, *Proceedings of IEEE Congress on Evolutionary Computing* (CEC 2003), 8 Aug-12 Dec. 2003, pp. 178–184, IEEE Press.
14. Han, K-H. and Kim, J-H. (2002). Quantum-inspired evolutionary algorithms with a new termination criterion, H_ε gate and two-phase scheme, *IEEE Transactions on Evolutionary Computation*, 8(3):156–169.
15. Itkin, A. (2005). Pricing options with the VG model using FFTs. Working Paper. Moscow State Aviation University.
16. Jiao, L. and Li, Y. (2005). Quantum-inspired immune clonal optimization, *Proceedings of 2005 International Conference on Neural Networks and Brain* (ICNN&B 2005), 13-15 Oct. 2005, pp. 461–466, IEEE Press.
17. Lagarias, J.C., J. A. Reeds, M. H. Wright, and P. E. Wright. (1998) .Convergence Properties of the Nelder-Mead Simplex Method in Low Dimensions, *SIAM Journal of Optimization*, Vol. 9, Number 1, pp.112–147, 1998.
18. Lee C-D., Chen, Y-J., Huang, H-C., Hwang, R-C. and Yu, G-R. (2004). The non-stationary signal prediction by using quantum NN, *Proceedings of 2004 IEEE International Conference on Systems, Man and Cybernetics*, 10-13 Oct. 2002, pp. 3291–3295, IEEE Press.
19. Li, Y., Zhang, Y., Zhao, R. and Jiao, L. (2004). The immune quantum-inspired evolutionary algorithm, *Proceedings of 2004 IEEE International Conference on Systems, Man and Cybernetics*, 10-13 Oct. 2002, pp. 3301–3305, IEEE Press.
20. Madan, D. and Seneta, E. (1990). The VG model for share market returns, *Journal of Business*, 63:511–524.
21. Madan, D. and Milne, F. (1991). Option pricing with VG martingale components, *Mathematical Finance*, 1(4):39–55.
22. Madan, D., Carr, P. and Chang, E. (1998). The Variance Gamma Process and Option Pricing, *European Finance Review*, 2:79–105.
23. Narayanan, A. and Moore, M. (1996). Quantum-inspired genetic algorithms, *Proceedings of IEEE International Conference on Evolutionary Computation*, May 1996, pp. 61–66, IEEE Press.
24. Rechenberg, I. (1973). *Evolutionsstrategie: Optimierung Technisher Systeme nach Prinzipien der Biologischen Evolution*, Fromman-Holzboog Verlag, Stuggart.
25. Spector, L. (2004). *Automatic Quantum Computer Programming: A Genetic Programming Approach*, Boston, MA: Kluwer Academic Publishers.
26. Tsai, X-Y., Chen, Y-J., Huang, H-C., Chuang, S-J. and Hwang, R-C. (2005). Quantum NN vs NN in Signal Recognition, *Proceedings of the Third Interna-*

tional Conference on Information Technology and Applications (ICITA 05), 4-7 July 2005, pp. 308–312, IEEE Press.
27. Yang, S., Wang, M. and Jiao, L. (2004). A genetic algorithm based on quantum chromosome, *Proceedings of IEEE International Conference on Signal Processing* (ICSP 04), 31 Aug- 4 Sept. 2004, pp. 1622–1625, IEEE Press.
28. Yang, S., Wang, M. and Jiao, L. (2004). A novel quantum evolutionary algorithm and its application, *Proceedings of IEEE Congress on Evolutionary Computation 2004* (CEC 2004), 19-23 June 2004, pp. 820–826, IEEE Press.
29. Yang, S., Wang, M. and Jiao, L. (2004). A Quantum Particle Swarm Optimization, in *Proceedings of the Congress on Evolutionary Computation 2004*, 1:320–324, New Jersey: IEEE Press.