Estimation of *EGARCH* Volatility Option Pricing Model using Bacterial Foraging Optimisation Algorithm

Jing Dang[1,2], Anthony Brabazon[1], Michael O'Neill[1], and David Edelman[2]

[1] Natural Computing Research and Applications Group,
University College Dublin, Ireland.
jing.dang1@ucdconnect.ie; anthony.brabazon@ucd.ie; m.oneill@ucd.ie
[2] School of Business, University College Dublin, Ireland.
davide@ucd.ie

Summary. The bacterial foraging optimisation algorithm is a novel evolutionary computation technique, which is based on mimicking the foraging behavior of E.coli bacteria. This chapter illustrates how a bacterial foraging optimisation (BFO) algorithm can be constructed and compared with the canonical Genetic Algorithm (GA) through testing of benchmark functions. The utility of the resulting algorithm is further extended to the financial domain, solving parameter estimation of EGARCH models, which can then be applied for pricing volatility options. The results from the algorithm are shown to be robust and extendable, suggesting the potential of applying BFO as a substitute to the conventional statistical computing techniques used in parameter estimations of financial models.

Key words: Bacterial foraging optimisation (BFO), volatility option, exponential GARCH (EGARCH)

1 Introduction

This chapter introduces a novel evolutionary computation technique, the bacterial foraging optimisation (BFO) algorithm, which models the foraging behavior of *Escherichia coli* bacteria as an optimisation process. It has been proposed and introduced recently by Passino[26] in 2002.

The proposed algorithm has been developed and applied to solve real problems. The basic foraging strategy is made adaptive through a Takagi-Sugeno fuzzy system by Mishra [21], dealing with harmonic estimation for a signal distorted with additive noise, the proposed algorithm does not rely on Newton-like gradient descent methods, and has great improvements in error percentage as well as the processing time compared with the conventional discrete Fourier transform and genetic algorithm method. Kim uses a hybrid system based on conventional genetic algorithm and bacterial foraging optimisation algorithm [17, 18] to tune a PID Controller for an automatic voltage regulator (AVR) system, the proposed approach is very efficient for solving global optimisation problems. Ulagammai trained wavelet neural network (WNN) using bacterial foraging optimisation algorithm for load forecasting (LF) as used in electric power system operations[32].

As shown above, BFO already has successful applications in solving real engineering problems, demonstrating satisfactory accuracy and stability. In this chapter, we further examine the ability of BFO to solve optimization problems, applying it to optimise nonlinear financial problems. We consider a basic BFO algorithm as proposed by Passino[26]. The various adaptive versions of the proposed BFO, and its ability to solve dynamic problems is beyond the scope of this chapter, while interested readers could refer to recent work by Tang et al[29, 30, 31, 19].

The ability of BFO to find optimal results is firstly assessed through the testing of six major benchmark functions, and compared with the canonical GA. It is thought that the comparative study would provide further insights to improve the BFO algorithm. We then examine the ability of applying this algorithm within financial domain. We apply BFO substituting traditionally adopted statistical computing methods to estimate a EGARCH model. These estimated parameters are used to approximate the volatility option pricing model.

1.1 Structure of Chapter

The rest of this chapter is organized as follows. The next section provides a concise overview of BFO, concentrating on the different steps involved in the algorithm. The following section illustrates the comparative study of BFO and GA, using testing results by six major benchmark functions. We then outline the experimental methodology adopted to estimate an option pricing model. The remaining sections provide the results of these experiments followed by a number of conclusions.

2 The Bacterial Foraging Optimisation (BFO) algorithm

Natural selection tends to eliminate animals with poor foraging (methods for locating, handling and ingesting food) strategies and favor the propagation of genes of those animals that have successful foraging strategies, since they are more likely to enjoy reproductive success. After many generations, poor foraging strategies are either eliminated or shaped into good ones. This activity of foraging led researchers to use it as optimisation process: animals take action to maximise the energy obtained per unit time spent foraging, in the face of constraints presented by its own physiology (e.g., sensing and cognitive capabilities) and environment (e.g., density of prey, risks from predators, physical characteristics of the search area).

The E.coli bacteria present in our intestines also undergo a foraging strategy. During the lifetime of E.coli bacteria, they undergo different stages such as chemotaxis, reproduction and elimination-dispersal. Chemotaxis is the ability of the bacterium to move toward distant sources of nutrients. In this stage, an E.coli bacterium alternates between swimming and tumbling (changing direction). In reproduction, the least healthy bacteria die and the other healthiest bacteria each split into two bacteria, which are then placed in the same location. In elimination-dispersal, any one bacterium is eliminated from the total set by dispersing it to a random location. An outline of the BFO algorithm is presented in Algorithm 1.

Algorithm 1. Canonical BFO algorithm

```
Randomly distribute initial values for θ^i, i = 1, 2, ..., S across the optimisation
domain

for Elimination-dispersal loop do
    for Reproduction loop do
        for Chemotaxis loop do
            for Bacterium i do
                Compute the value of cost function J(i, j)
                Let J_last = J(i, j)
                Tumble: Generate a random variable φ(j) as a unit length
                random direction
                Move: let θ^i(j + 1) = θ^i(j) + C(i) * φ(j)
                Compute J(i, j + 1) with θ^i(j + 1)
                Swim: let m=0
                while m < N_s do
                    let m=m+1
                    if J(i, j + 1) < J_last then
                        let J_last = J(i, j + 1)
                        let θ^i(j + 1) = θ^i(j) + C(i) * φ(j) and compute the
                        new J(i, j + 1)
                    else
                        let m = N_s
                    end
                end
            end
        end
        Let J^i_health = Σ_{j=1}^{N_c+1} J(i, j) be the health of bacterium i.
        Sort bacteria in order of ascending cost J_health
        The S_r = S/2 bacteria with the highest J_health value die and other
        S_r bacteria with the best value split
    end
    Eliminate and disperse the bacteria to random locations on the
    optimisation domain with probability p_ed
end
```

As depicted in Algorithm.1, the BFO algorithm considered in this chapter contains three steps, namely, chemotaxis, reproduction, and elimination-dispersal. A description of each of these steps is as follows:

A. Chemotaxis

Microbiological studies show that E.coli bacteria move by their flagella, or biological engines referred to by many biologists. When all the flagella rotate counterclockwise, the E.coli bacteria move forward, when all the flagella rotate clockwise, the E.coli bacteria slow down and tumble in its place. The foraging of E.coli bacteria is accompanied by the alternation of the two modes of operation its entire lifetime, and the bacteria are able to find nutrients, avoid noxious substances. The chemotactic step is achieved through tumbling and swimming via Flagella, which is illustrated in Figure 1.

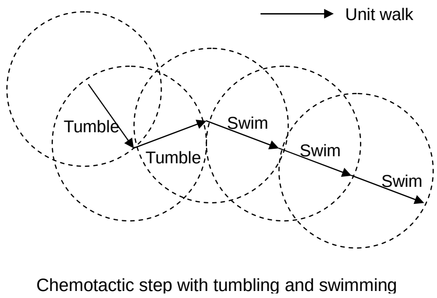


Fig. 1. Chemotactic Step

In the existing BFO algorithm, a *tumble* is represented by a unit walk with random direction, and a *swim* is indicated as a unit walk with the same direction in the last step. After one step move, the position of the ith bacterium can be represented as

$$\theta^i(j+1) = \theta^i(j) + C(i) * \phi(j)$$

where $\theta^i(j)$ indicates the position of the ith bacterium at the jth chemotactic step. $C(i)$ is the step size taken in the random direction, which is specified by $\phi(j)$, a unit length random direction. Let $J(i,j)$ denotes the cost at the position of the ith bacterium $\theta^i(j)$. If at $\theta^i(j+1)$, the cost $J(i,j+1)$ is better(lower) than the cost at $\theta^i(j)$, another step of swimming is taken, and is continued as long as it continues to reduce the cost, but only up to a maximum number of steps, N_s. This means that the bacterium will tend to keep moving if it is headed in the direction of an increasingly favorable environment.

B.Reproduction

After N_c chemotactic steps, a reproduction step is taken. Let N_{re} be the number of reproduction steps to be taken. The accumulated cost of each bacterium is calculated as the sum of the cost during its life, i.e., $\sum_{j=1}^{N_c+1} J(i,j)$. All bacteria are sorted in order of ascending accumulated cost(higher accumulated cost represents that a bacterium has lower fitness value, which means it did not get as many nutrients during its life of foraging and thus unlikely to reproduce). In the reproduction step, only the first half of population survive and a surviving bacterium splits into two identical ones, which are placed at the same location. Thus, the population size of bacteria is kept constant.

C.Elimination - dispersal

The elimination-dispersal step happens after a certain number of reproduction steps. A bacterium is chosen according to a preset probability p_{ed}, to be dispersed and moved randomly to a new position within the optimisation domain.

The chemotactic step provides a basis for local search, the reproduction step speeds the convergence, and the elimination-dispersal step prevent the local optimum trapping effectively. For the BFO algorithm employed in this chapter, we did not consider the social swarming effect during the chemotactic stage, where cell-released attractants are used to signal other cells that they should swarm together. The ways representing swarming effect can be different, Passino [26] suggests a cell-cell attractant/repellant function, however, it remains a question of how much more marginal benefits it brings in by adding complexity of the basic algorithm. For this reason, we focus our study of the basic form of BFO algorithm, where chemotactic step size parameter $C(i)$ is adapted to control the convergence speed, i.e., $C(i)$ starts from $\frac{Range\ of\ search\ domain}{100}$, and shrinks after each reproduction step.

The basic form BFO bears many similarities to the canonical GA, in the next section, we continue by presenting a comparison of BFO and the canonical GA on some benchmark problem instances in order to assess the searching ability of BFO for global optimum.

3 Comparative study with GA

The comparison with GA is undertaken to understand the relative performance characteristics of BFO. Then it might be possible to build newer versions of BFO, for example, hybrid GA-BFO algorithms.

BFO and GA are all population based search algorithms. As shown from Table 1, the nutrient concentration function and the fitness function used in BFO and GA respectively are both types of landscape. In BFO, bacteria in the most favorable environments gain a selective advantage for reproduction, which is similar to the Selection process in GA. In BFO, the bacteria with higher fitness (lower cost) split into two children, which are at the same

Table 1. Comparison of BFO and canonical GA

BFO	GA
Nutrient concentration function	Fitness function
Bacterial reproduction	Selection
Bacterial splitting	Crossover
Elimination and dispersal	Mutation

concentration, whereas in GA, with crossover they generally end up around their parents on the fitness landscape. In BFO, elimination-dispersal results in physical dispersion in a geographical area, and mutation in GA results genotypical changes, but they both can help jumping out of the local optimum trap during the search.

Note: While implementing the BFO algorithm in Matlab, instead of using an inner for loop for each bacterium i within the chemotactic loop [3], we process the population of all the bacteria together in a matrix form, which improves the algorithm speed significantly.

Benchmark Function Tests

Six major static benchmark functions[8, 25] are chosen to test the ability of BFO to find the global minimum. We test these benchmark functions within 5 dimensions, so that the result would provide implications to the 5-dimensional financial problem we consider in section 4.2. In order to get a better understanding of BFO, the results are compared to the canonical GA results. Details of the benchmark functions are shown in Table 2 and Figure 2.

Table 2. Benchmark functions

f	Function	Mathematical representation	Range	$f(x_i^*)$	x_i^*
f_1	Sphere	$f(x) = \sum_{i=1}^{p} {x_i}^2$	$-5.12 \leq x_i \leq 5.12$	0	0
f_2	Schwefel	$f(x) = \sum_{i=1}^{p} (\sum_{j=1}^{i} x_j)^2$	$-65.536 \leq x_i \leq 65.536$	0	0
f_3	Rosenbrock	$f(x) = \sum_{i=1}^{p-1} 100(x_{i+1} - {x_i}^2)^2 + (1 - x_i)^2$	$-2.048 \leq x_i \leq 2.048$	0	1
f_4	Rastrigin	$f(x) = 10p + \sum_{i=1}^{p} ({x_i}^2 - 10cos(2\pi x_i))$	$-5.12 \leq x_i \leq 5.12$	0	0
f_5	Ackley	$f(x) = 20+e-20exp\left(-0.2\sqrt{\frac{1}{p}\sum_{i=1}^{p} {x_i}^2}\right) - exp\left(\frac{1}{p}\sum_{i=1}^{p}\cos(2\pi x_i)\right)$	$-30 \leq x_i \leq 30$	0	0
f_6	Griewangk	$f(x) = \sum_{i=1}^{p} \frac{{x_i}^2}{4000} - \prod_{i=1}^{p} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	$-600 \leq x_i \leq 600$	0	0

f_1: Sphere's function (also known as De Jong's function 1) is the simplest test function, which is continuous, convex and unimodal.

[3]The original Matlab code of the algorithm by Passino can be found on the web address `http:/www.ece.osu.edu/~passino/ICbook/ic_index.html`

f_2: Schwefel's function produces rotated hyper-ellipsoids with respect to the coordinate axes. It is continuous, convex and unimodal.

f_3: Rosenbrock's function (also known as De Jong's function 2, or Banana function) is a classic optimization problem. The global optimum is inside a long, narrow, parabolic shaped flat valley. To find the valley is trivial, however convergence to the global optimum is difficult and hence this problem has been repeatedly used to assess the performance of optimization algorithms.

f_4: Rastrigin's function is based on function 1 with the addition of cosine modulation to produce many local minimum. Thus, the test function is highly multimodal [4]. However, the location of the minimum are regularly distributed.

f_5: Ackley's function is a widely used multimodal test function.

f_6: Griewangk's function is similar to Rastrigin's function. It has many widespread local minimum. However, the location of the minimum are regularly distributed.

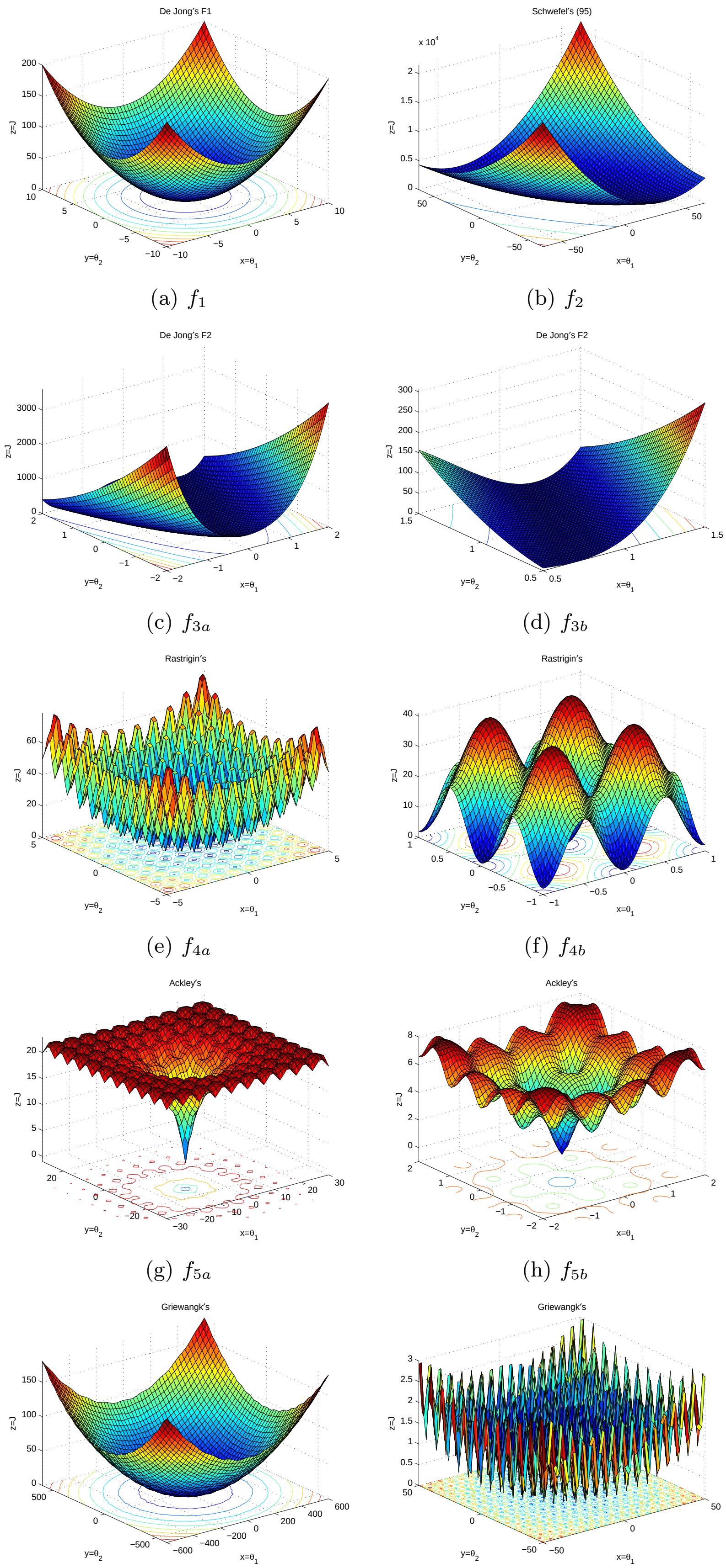
The parameters used for BFO are shown in Table 3. The value are chosen according to trial and errors before, to make balance between search speed and accuracy. The chemotactic step size $C(i)$ starts from $\frac{Range\ of\ search\ domain}{100}$, and shrinks after each reproduction step. The larger $C(i)$ results in the jump out of local optimum during the early search, and the shrank $C(i)$ make the convergence slower, exploring more accurate value around the local minimum.

Table 3. Initializing BFO Parameters

Parameters	Definition
$D = 5$	Dimension of the search space
$S = 50$	Number of bacteria (population size)
$N_c = 20$	Maximum number of chemotactic steps
$N_s = 4$	Maximum number of swimming steps
$N_{re} = 2$	Maximum number of reproduction steps
$S_r = S/2$	Number of bacteria for reproduction/splitting in two
$N_{ed} = 2$	Maximum number of elimination-dispersal steps
$p_{ed} = 0.25$	The probability that each bacterium will be eliminated/dispersed
$C(i)$	Chemotactic step size for bacterium i

To make a fair comparison between BFO and GA, we use same population size of 50, the same dimension of 5, and the same number of iterations as 100. Where in GA, the number of iterations is equivalent to the number of generations, and in BFO, number of iterations is the count of total chemotactic steps taken for each bacterium at the end of the algorithm. We set crossover rate of 0.7, and mutation rate of 0.05.

[4] A function is multimodal if it has two or more local optimum



(a) f_1 (b) f_2

(c) f_{3a} (d) f_{3b}

(e) f_{4a} (f) f_{4b}

(g) f_{5a} (h) f_{5b}

(i) f_{6a} (j) f_{6b}

Fig. 2. Two dimensional visualization of benchmark functions

The results are shown in Table 4, where both BFO and GA are run 30 times. The first column lists the minimal (optimal) objective value found during the 30 runs within the whole population. The second and third column lists the mean and standard deviation for the minimal value of 30 runs. The *Time* column shows the averaged processing time taken for each run.

Table 4. Results of BFO and GA with 30 runs for benchmark functions testing

Algorithm	Best	Mean	S.D.	Time(s)
$f_1 : Sphere's\ function$				
BFO	0.000376	0.00194	0.00103	0.047
GA	0.002016	0.00202	1.32e-018	1.07
$f_2 : Schwefel's\ function$				
BFO	0.1717	6.55	9.508	0.049
GA	0.1982	0.55	0.614	1.21
$f_3 : Rosenbrock's\ function$				
BFO	0.03989	0.578	0.73	0.050
GA	0.06818	2.46	1.43	1.17
$f_4 : Rastrigin's\ function$				
BFO	2.032	10.4	3.84	0.058
GA	0.399	0.7841	0.69	1.06
$f_5 : Ackley's\ function$				
BFO	0.0361	3.11	4.14	0.085
GA	1.0895	1.0895	9.03e-016	1.14
$f_6 : Griewangk's\ function$				
BFO	0.3271	0.687	0.17	0.113
GA	0.7067	0.722	0.015	1.20

As suggested in Table 4, BFO can find more accurate minimal(optimal) results than GA except for f_4- Rastrigin′s function. f_4 is a highly multimodal function, where BFO is more likely to be trapped in the local minimum in this case. The average processing time to run BFO is much shorter than GA for same number of iterations. And though BFO found accurate minimal values, the mean and standard deviation is rather high in function f_2, f_4 and f_5.

The global search process for f_3 using BFO and GA is illustrated in Figure 3, obviously from the graph, GA converges much earlier with fewer iteration steps compared with BFO, however, BFO can find more optimal objective value with more iterations. There is always trade-off between accuracy and speed, and BFO gives good balance with acceptable accuracy and search speed compared with GA. It is then reasonable to apply BFO solving real financial problems, such as parameter estimation considered in the next section. Traditionally statistical computing methods are usually employed in finance for parameter estimation, it requires gradient information about the objective function and often requires the initial estimates of optimising parameters, however, BFO is not confined by these. In the following section, we illustrate

BFO estimating parameters of a EGARCH model for the purpose of volatility option pricing.

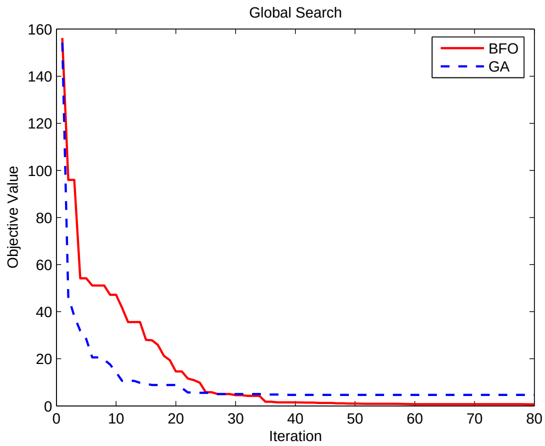


Fig. 3. Objective Value vs. Number of Generations. GA converges earlier than BFO, however, BFO can find more optimal objective value with more iterations.

4 Volatility Option Pricing Model

4.1 Volatility Option

Volatility is a measure of how much a stock can move over a specific amount of time. The more variability there is in the price changes of the stock or index, the higher the volatility. It is defined as the standard deviation of daily percentage changes of the stock price. Options are financial instruments that convey the right, but not the obligation, to engage in a future transaction on some underlying security. For example, the buyer of a European call option has the right, but not the obligation to buy an agreed quantity of a particular security (the underlying instrument) from the seller of the option at a certain time (the expiration date) for a certain price (the strike price).

In February 2006, options on the S&P500 volatility index (VIX Options) began trading on the Chicago Board of Exchange(CBOE), which is the first product on market volatility to be listed on an regulated securities exchange. The S&P500 Volatility Index (VIX) was created in 1993 as the first measure of volatility in the overall market. VIX is designed to reflect investors' consensus view of expected stock market volatility over the next 30 days. VIX is the widely disseminated benchmark index commonly referred to as the market's "fear gauge" and serves as a proxy for investor sentiment - rising when investors are anxious or uncertain about the market and falling during times of confidence or complacency. VIX options offer investors the ability to make trades based on their view of future direction or movement of the VIX, and option buyers have the advantage of limited risk. VIX options also offer the opportunity to hedge volatility risk of a portfolio, distinct from price risk.

The growing literature on volatility options has emerged after the 1987 stock market crash. Brenner and Galai [3, 4] first suggested options written on a volatility index that would serve as the underlying asset. Towards this end, Whaley [33] constructed VIX (currently termed VXO), a volatility index based on the S&P 100 option's implied volatilities [5] traded in CBOE. Ever since, other implied volatility indices have also been develop (e.g., VDAX in Germany, VXN in CBOE, VX1 and VX6 in France). Various models to price volatility options written on the instantaneous volatility have also been developed (e.g., Whaley [33], Grunbichler and Longstaff [11], and Detemple and Osakwe [7]). These models differ in the specification of the assumed stochastic process, and the assumptions made about the volatility risk premium. For example, Grunbichler and Longstaff [11] specify a mean reverting square root diffusion process for volatility. Their framework is similar to that of Hull and White [15], Stein and Stein [28] and others. Since volatility is not trading at the time they assume that the premium for volatility risk is proportional to the level of volatility. This approach is in the spirit of the equilibrium approach of Cox, Ingersoll and Ross [6]. A more recent paper by Detemple and Osakwe [7] also uses a general equilibrium framework to price European and American style volatility options. They emphasize the mean-reverting in log volatility model.

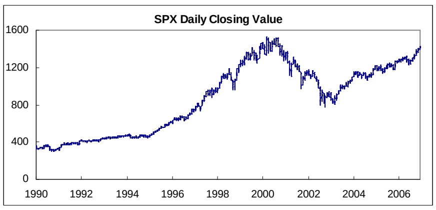
The literature on option pricing under stochastic volatility can be grouped into two categories - the bivariate diffusion and GARCH (generalized autoregressive conditional heteroskedasticity [6]) approaches. The former strand approaches option pricing with stochastic volatility in a diffusion framework, assuming that the function of asset price and the volatility of an asset follow stochastic processes. The latter develops the option pricing model in a GARCH framework. GARCH models are popular econometric modelling methods, having been firstly specified by Engle [10] and Bollerslev [1], they are specifically designed to model and forecast changes in variance, or volatility per se. These two strands of option pricing models are unified by a convergence result that the GARCH option pricing model weakly converged to a bivariate diffusion option pricing model[23, 9].

Figure 4 provides some empirical results about the volatility of S&P500 index, based on sample period from 02/01/1990 to 30/12/2006, the data source

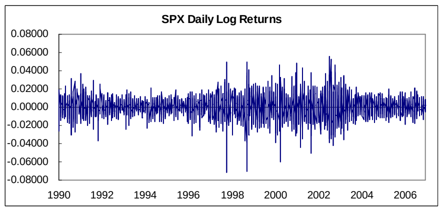
[5] Implied volatility is simply the volatility that makes the theoretical value of an option equal to the market price of an option.

[6] the "heteroskedasticity" term refers to a condition which exists when the differences between actual and forecast values do not have a constant variance across an entire range of time series observations.

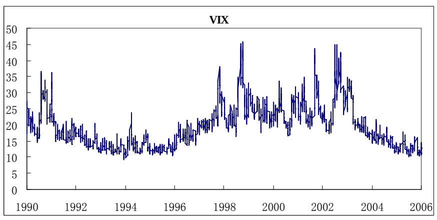
is CBOE: 4(a) shows the daily closing values of the S&P 500 equity index in the sample period. There appears no long-run average level about which the series evolves. This is evidence of a nonstationary time series. 4(b) illustrates the continuously compounded returns (the log returns) [7] associated with the price series in Figure 4a. In contrast to the price series in 4(a), the log returns appear to be quite stable over time, and the transformation from prices to returns has produced a stationary time series. 4(c) shows the closing level of the S&P 500 Volatility Index (VIX) during the sample period. We could intuitively find the volatility clustering effect, where large volatility movements are more likely to be succeeded by further large volatility movements of either sign than by small movements. 4(d) gives an example of the probability density function of VIX, the dots represents the frequency of VIX occurred within range of values in the x-axis, and the belled curve line represents the probability of ln(VIX) for a normal distribution (with mean $\mu = 2.89$, and standard deviation σ=0.32). It shows that VIX tends to follow a lognormal distribution.



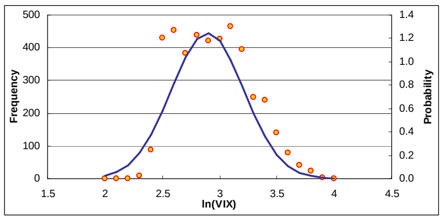
(a) S&P500 Daily Closing Prices



(b) S&P500 Daily Log Returns



(c) S&P500 Volatility Index (VIX)



(d) Probability Density of VIX

Fig. 4. Empirical Results on Volatility of S&P500

We can see from the empirical results in Figure 4, that the volatility rates of S&P 500 are higher over certain periods and lower in others, and that periods of high volatility tend to cluster together. Therefore, we would expect

[7] Denoting the successive price observations made at times $t-1$ and t as P_{t-1} and P_t respectively, then we could obtain the continuously compounded returns as $R_t = \log\frac{P_t}{P_{t-1}} = \log P_t - \log P_{t-1}$, this is the preferred method for most financial calculations since the log returns are more stationary and continuously distributed.

the volatilities to be correlated to some extent. Also it is noticeable that the volatility tends to revert to some long-running average (mean-reversion properties). In this chapter, we consider the mean-reverting log process (MRLP) option pricing model, proposed by Detemple and Osakwe [7]. The relevance of this model is motivated by (i) substantial empirical evidence supporting the EGARCH (Exponential GARCH) model of Nelson [24] and (ii) the fact that EGARCH converges to a Gaussian process that is mean reverting in the log and thus matches our MRLP specification.

4.2 EGARCH Pricing Model

With the existence of too much noise in the newly traded volatility option data, we calibrate the MRLP option pricing model by estimating the corresponding EGARCH model and then taking the limit. The exponential GARCH (EGARCH) model [8] is an asymmetric model designed to capture the leverage effect, or negative correlation, between asset returns and volatility.

The EGARCH [9] (1,1) model considered in this chapter is set up as follows:

The conditional mean model:

$$y_t = C - \frac{1}{2}\sigma_t^2 + \epsilon_t \tag{1}$$

$$\text{where } \epsilon_t = \sigma_t z_t, \text{ and} z_t \sim N(0,1)$$
$$y_t = \log(\tfrac{S_t}{S_{t-1}}) \text{ , (the log returns of S\&P)}$$

The conditional variance model:

$$\log\sigma_t^2 = K + G_1 \log\sigma_{t-1}^2 + A_1[|z_{t-1}| - E(|z_{t-1}|)] + L_1 z_{t-1} \tag{2}$$

$$\text{Where } z_{t-1} = \tfrac{|\epsilon_{t-1}|}{\sigma_{t-1}}$$
$$E(|z_{t-1}|) = \sqrt{2/\pi} \quad \text{, if } z_t \sim \text{Gaussian}$$

Duan [9] shows that under the locally risk-neutralized probability measure Q, the asset return dynamic takes the form in equation 1 (also refers to Detemple [7] and Hentschel [14]).

There are five parameters to be estimated using BFO, namely, C, K, G_1, A_1, and L_1, the search domain is from -1 to 1. C is the conditional mean constant, K is the conditional variance constant, G_1 (GARCH term) is the coefficients

[8] The EGARCH model was proposed by Nelson [24] , the nonnegativity constraints as in the linear GARCH model are taken out and so there are no restriction on the parameters in this model.

[9] The EGARCH model specified here is often referred to as the EGARCH in Mean (EGARCH-M) model, since the conditional variance term σ^2 in the variance equation also appears in the mean equation

related to lagged conditional variances, A_1 (ARCH term) is the coefficients related to lagged innovations, L_1 is the leverage coefficients for asymmetric EGARCH-M(1,1) model. The coefficient of σ^2 (GARCH in Mean term) is fixed at $-\frac{1}{2}$, hence not being estimated.

The left-hand side of equation 2 is the log value of the conditional variance. This implies that the leverage effect is exponential, rather than quadratic, and the forecasts of the conditional variance are guaranteed to be nonnegative. The presence of leverage effects can be tested by the hypothesis that $\alpha_3 < 0$. The impact is asymmetric if $\alpha_3 \neq 0$.

The weak limit of this model converges to the unique strong solution of the MRLP (mean-reverting log process) stochastic volatility diffusion model. The limiting process is:

$$d\,ln(S_t) = (r - \delta - \frac{1}{2}V^2)dt + V_t(\rho dZ_{1t} + \sqrt{1-\rho^2}dZ_{2t}) \tag{3}$$

$$d\,ln(V_t) = (\alpha - \lambda ln(V_t))dt + \sigma dZ_{1t} \tag{4}$$

Detemple and Osakwe ([7]) derived analytic pricing formulae for European volatility options as a functions of parameters α, λ, σ and ρ, based on the MRLP volatility diffusion model. Where α/λ denoting a long run mean for log (V), $exp\left((\alpha + \frac{1}{4}\sigma^2)/\lambda\right)\sqrt{285}$ denoting a long run mean annualized volatility (based on 285 days), and ρ represents the correlation between Z_1 and Z_2. These parameters for the option pricing model can be calculated as below [9]:

$$\begin{aligned} \alpha &= \frac{K}{2} + \frac{A_1}{\sqrt{2\pi}} \\ \lambda &= 1 - G_1 \\ \sigma &= \frac{1}{2}\sqrt{{L_1}^2 + (\frac{\pi-2}{\pi}){A_1}^2} \\ \rho &= \frac{L1}{2\sigma} \end{aligned} \tag{5}$$

We employ BFO to optimize the EGARCH model parameters: C, K, G_1, A_1 and L_1, details is explained in the following part.

4.3 EGARCH Estimation using BFO

The EGARCH model can be estimated by maximum likelihood estimation (MLE). The idea behind maximum likelihood parameter estimation is to determine the parameters that maximize the probability (likelihood) of the sample data. From a statistical point of view, the method of maximum likelihood is considered to be more robust and yields estimators with good statistical properties. Although the methodology for maximum likelihood estimation is simple, the implementation is mathematically intense. For the EGARCH models specified in equation 1,2, the objective is to maximise the log likelihood function (LLF) as follows:

$$LLF = -\frac{1}{2}\sum_{t=1}^{T}[\log(2\pi\sigma_t^2) + \frac{\epsilon_t^2}{\sigma_t^2}] \tag{6}$$

Given the observed log return series, the current parameter values, and the starting value of $z_1 \sim N(0,1)$, $\sigma_1^2 = exp(K)$, the σ_t^2 and ϵ_t are inferred by recursive substitution based on the conditional mean/variance equation (equation 1 and 2):

$$\begin{aligned}
&\sigma_t^2 = exp(K + G_1 \log\sigma_{t-1}^2 + A_1[|z_{t-1}| - E(|z_{t-1}|)] + L_1 z_{t-1}) \\
&z(t) = (-C + y_t + \frac{1}{2}\sigma_t^2)/\sigma_t \\
&\epsilon_t = \sigma_t z_t
\end{aligned} \tag{7}$$

The log-likelihood function then uses the inferred residuals ϵ_t and conditional variances σ_t^2 to evaluate the appropriate log-likelihood objective function in equation

We employ BFO as an optimisation tool searching for the optimal parameters and maximising the log-likelihood objective function. Since minimising the negative log-likelihood ($-LLF$) is the same as maximising the log-likelihood(LLF), we use $-LLF$ as our nutrient function (the objective function). And the goal is to minimize the $-LLF$ value, by optimising parameters C, K, G_1, A_1, L_1 within the search domain.

5 Results

The EGARCH model is fitted to the return series of S&P 500 daily index using BFO algorithm. The S&P 500 (Ticker SPX) equity index is obtained from CBOE, with the sample period from 02/01/1990 to 30/12/2006, for a total of 4084 daily observations (the first price data at 02/01/1990 is used to calculate return series, hence the return series have 4083 observations).

The parameters used in the BFO algorithm are listed in Table 5. They are chosen based on benchmark function results, and adjusted on trial and errors for this particular problem. $C(i)$ is decided by varying it from 0.01 to 0.1 in step of 0.01, and running the BFO algorithm for 10 trials in each case respectively. The best result to achieve the mean of the minimal J is obtained for $C(i) = 0.008$.

Figure 5 depicts the evolution of the objective function, measured using negative maximum likelihood ($-LLF$), as a function of the iteration number for a single run of the algorithm. Figures 6(a), 6(b), 6(c), 6(d) and 6(e) depict the evolution of the parameters C, K, G_1, A_1, and L_1 as a function of the iteration number for a single run of the algorithm. In the early generations BFO mainly performs global search for the optimum value, with quicker convergence than the latter generations, where local optimal search is focused.

Table 5. BFO Parameters

Dimension of the search space: $D = 5$
Population size: $S = 50$
Chmotactic steps: $N_c = 20$
Swimming steps: $N_s = 4$
Reproduction steps: $N_{re} = 4$
Number of bacteria for reproduction/splitting: $S_r = S/2$
Elimination-dispersal steps: $N_{ed} = 2$
Probability that each bacterium will be eliminated/dispersed $p_{ed} = 0.25$
Chemotactic step size for bacterium i: $C(i) = 0.08$

From the 40th iteration, the optimal objective value becomes worse and the effect lasting for a few generations, this is due to the elimination-dispersal step conducted in iteration 40, by allowing the optimal value to be worse, we can jump out of the local minimum, and moving towards global optimum.

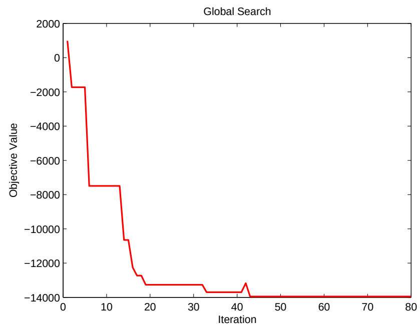
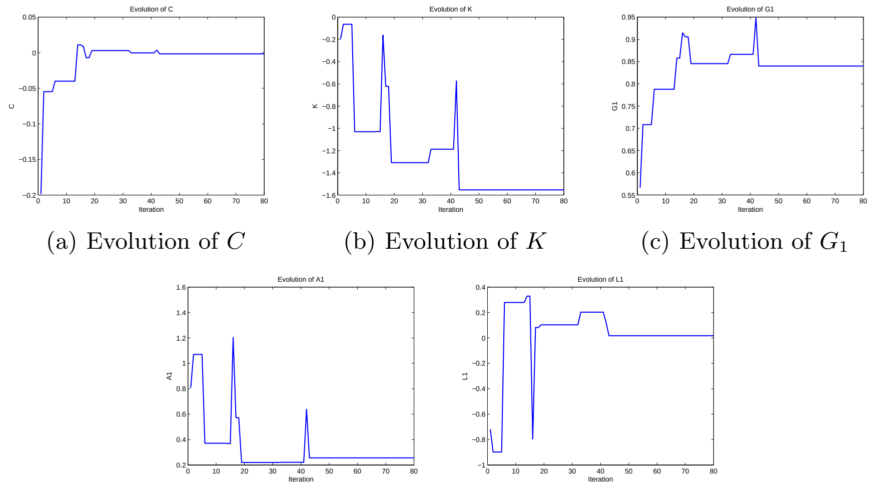


Fig. 5. Objective Value vs. Iteration

The best results over 30 runs are reported in the second column of Table 6. The best results averaged over 30 runs are reported in the third column. The standard deviation of the best results over 30 runs are reported in the fourth column. In order to provide a benchmark for the results obtained by BFO, a Matlab optimising function *fmincon* was used. The function *fmincon* uses sequential quadratic programming (SQP) methods, which closely mimic Newton's method for constrained optimization. It requires information about the gradient of the objective function and initial estimates of the optimising parameters, while BFO does not require these. Running BFO over 30 trials, we obtain the results shown in Table 6.

From Table 6, we obtain the the optimal objective (the minimal $-LLF$) value of -14180.98, which is slightly lower than -14244.13 obtained in Matlab using the default *fmincon* function. The result is reasonably acceptable and



(a) Evolution of C (b) Evolution of K (c) Evolution of G_1

(d) Evolution of A_1 (e) Evolution of L_1

Fig. 6. Evolution of parameters over generations

Table 6. Results of BFO with 30 runs

Parameter	Optima	Mean	Standard Deviation	Matlab optimisation
$-LLF$ (Objective)	-14180.98	-14099.87	32.3997	-14244.13
C	0.0005	0.0003	0.00067	0.0002
K	-0.3588	-0.301	0.0478	-0.3643
G_1	0.9107	0.904	0.0056	0.9610
A_1	0.1634	0.235	0.0489	0.1782
L_1	-0.1197	-0.0495	0.0473	-0.1184

the standard deviation is relatively small, indicating the stability of BFO algorithm. The estimated optimal parameters value are: $C = 0.0005$, $K = -0.3588$, $G_1 = 0.9107$, $A_1 = 0.1634$, $L_1 = -0.1197$. The leverage effect term L_1 is negative and statistically different from zero, indicating the existence of the leverage effect in future stock returns during the sample period. With the flexibility of BFO, it is believed that by further evolving BFO parameters such as chemotactic step size $C(i)$, number of chemotactic steps N_c etc, we can improve the accuracy of the results, however, there is always trade off between accuracy(achieved by adding complexity to the algorithm) and convergence speed.

Based on the above results and equation 5, the resulting stochastic volatility option pricing model parameters are: $\alpha = -0.1142, \lambda = 0.0893, \sigma = 0.0775$ and $\rho = -0.7722$. The negative correlation ρ corresponds to the asymmetric relationship between returns and changes in volatility, i.e., the leverage effect. The negative α implied mean reversion with a long run mean for log (V) of $\alpha/\lambda = -1.2790$, and a long run mean annualised volatility (based on 285 days) of $exp\left((\alpha + \frac{1}{4}\sigma^2)/\lambda\right)\sqrt{285} = 4.7783$ percent. The speed of reversion λ, is small, indicating strong autocorrelation in volatility which in turn implies volatility clustering. These are consistent with the empirical results found from Figure 4.

Furthermore, based on the estimated parameters of the volatility option pricing model, hedgers can manage their risk/volatility in the existing investment/portfolio. Traders can also use the generated theoretical volatility options prices as a trading guide to make arbitrage/speculating profits.

6 Conclusion

In this chapter, we introduced and assessed the recently proposed bacterial foraging optimisation (BFO) algorithm. It bears many similarities to the existing GA, through a further comparative study between BFO and GA and from the testing results using six major benchmark functions, we find that BFO can find a satisfactory trade-off between the global and local search. It has fast running speed which has special implications to the dynamic problems. However, while implementing BFO solving multimodal problems, more efforts should be done to choose parameters before coming to an optimal estimation. In solving our financial problem, the EGARCH volatility option pricing model, BFO shows its applicability and flexibility - not dependent on the gradient information about the nutrient (objective) function. This has further implications to estimate more complicated financial econometric models. BFO mimics social foraging behavior of bacteria, this is similar to the behavior of individual investors in the financial market, where they might share information and have herding effect, hence it worth further investigation of BFO with swarming effect and the applications in the financial market.

References

1. Bollerslev T (1986). Generalized Autoregressive Conditional Heteroskedasticity. Journal of Econometrics 31, 307-327.
2. Brabazon A and O'Neil M (2006). Biologically Inspired Algorithms for Financial Modelling. Berlin Springer.
3. Brenner M and Galai D (1989). New Financial Instruments for Hedging Changes in Volatility. Financial Analyst Journal, Jul/Aug, 61-65.
4. Brenner M and Galai D (1993). Hedging Volatility in Foreign Currencies. Journal of Derivatives, 1, 53-59.
5. Cox J-C and Ross S-A (1976). The Valuation of Options for Alternative Stochastic Processes. Journal of Financial Economics 3, 145-166.
6. Cox J-C, Ingersoll J-E and Ross S-A (1985). An Intertemporal General Equilibrium Model of Asset Prices. Econometrica 53, 363-384.

7. Detemple J-B and Osakwe C (2000). The Valuation of Volatility Options. European Finance Review. Vol.4, No.1, pp.21-50.
8. Digalakis J-G, Margaritis K-G (2000). An experimental study of benchmarking functions for Genetic Algorithms. Proceedings of IEEE Conference on Transactions, Systems, Man and Cybernetics, Vol.5, pp.3810-3815.
9. Duan J-C (1997). Augmented GARCH (p,q) Process and its Diffusion Limit. Journal of Econometrics 79, 97-127.
10. Engle R-F (1982). Autoregressive conditional heteroskedasticity with estimates of the variance of U.K. inflation. Econometrica 50, 987-1008.
11. Grunbichler A and Longstaff F (1996). Valuing Futures and Options on Volatility. Journal of Banking and Finance 20, 985-1001.
12. Heston S-L (1993). A Closed-Form Solution for Options with Stochastic Volatility with Applications to Bond and Currency Options. Review of Financial Studies, Vol.6, No. 2, pp237-343.
13. Heston S-L and Nandi S (1997). A Closed-Form GARCH Option Pricing Model. Working Paper 97-9, Federal Reserve Bank of Atlanta.
14. Hentschel L (1995). All in the Family: Nesting Symmetric and Asymmetric GARCH Models. Journal of Financial Economics 39, 71-104.
15. Hull J and White A (1987). The Pricing of Options on Assets with Stochastic Volatility. Journal of Finance 42, 281-300.
16. Hull J (2006) Options, Futures, and Other Derivatives (Sixth Ed.), Person Prentice Hall.
17. Kim D-H and Cho J-H (2005). Intelligent Control of AVR System Using GA-BF. Springer (KES 2005), LNAI 3684, pp.854-859, 2005.
18. Kim D-H, Abraham A and Cho J-H (2007). A Hybrid Genetic Algorithm and Bacterial Foraging Approach for Global Optimization. Information Sciences 177 (2007) 3918-3937.
19. Li M-S, Tang W-J, et al (2007). Bacterial Foraging Algorithm with Varying Population for Optimal Power Flow. Springer (EvoWorkshops 2007), LNCS 4448, pp.32-41.
20. Liu Y and Passino K-M (2002). Biomimicry of Social Foraging Bacteria for Distributed Optimization: Models, Principles, and Emergent Behaviors. Journal of Optimization Theory and Applications. Vol.115, No.3, 603-628.
21. Mishra S (2005). A Hybrid Least Square-Fuzzy Bacterial Foraging Strategy for Harmonic Estimation. IEEE Transactions on Evolutionary Computation, Vol.9, No.1.
22. Mishra S and Bhende C-N (2007). Bacterial Foraging Technique-Based Optimized Active Power Filter for Load Compensation. IEEE Transactions on Power Delivery, Vol.22, No.1.
23. Nelson D-B (1990). ARCH Models as Diffusion Approximations. Journal of Econometrics, 45, 7-39.
24. Nelson D-B (1991) Conditional Heteroskedasticity in Asset Returns: A New Approach. Econometrica, Vol.59, No.2, pp. 347-370
25. Ortiz-Boyer D, Hervás-Martínez C and García-Pedrajas N (2005) CIXL2: A Crossover Operator for Evolutionary Algorithms Based on Population Features. Journal of Artificial Intelligence Reserach, Vol 24, pp.1-48.
26. Passino K-M (2002). Biomimicry of bacterial foraging for distributed optimization and control, Control Systems Magazine. IEEE Vol. 22, Iss. 3, 52 - 67.

27. Ramos V, Fernandes C and Rosa A-C (2005). On Ants, Bacteria and Dynamic Environments. Natural Computing and Applications Workshop (NCA 2005), IEEE Computer Press 25-29, Sep.2005.
28. Stein E-M and Stein J-C (1991). Stock Price Distributions with Stochastic Volatility: An Analytical Approach. Review of Financial Studies 4, 727-752.
29. Tang W-J, Wu Q-H and Saunders J-R (2006). A Novel Model for Bacterial Foraging in Varying Environments. Springer (ICCSA 2006), LNCS 3980, pp.556-565.
30. Tang W-J, Wu Q-H and Saunders J-R (2006). Bacterial Foraging Algorithm for Dynamic Environments, IEEE Congress on Evolutionary Computation (CEC 2006), pp.1324-1330, Jul.2006.
31. Tang W-J, Wu Q-H and Saunders J-R(2007). Individual- Based Modeling of Bacterial Foraging with Quorum Sensing in a Time-Varying Environment. Springer (EvoBIO 2007), LNCS 4447, pp.280-290.
32. Ulagammai M, Venkatesh P, et al.(2007). Application of Bacterial Foraging Technique Trained Artificial and Wavelet Neural Networks in Load Forecasting. Neurocomputing (2007) 10747.
33. Whaley R-E (1993). Derivatives on Market Volatility: Hedging Tools Long Overdue. Journal of Derivatives 1, 71-84
34. Whaley R-E (2000). The Investor Fear Gauge, Journal of Portfolio Management 26, 3,Spring 2000.