Option Model Calibration Using a Bacterial Foraging Optimization 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.dang@ucd.ie, anthony.brabazon@ucd.ie, m.oneill@ucd.ie
[2] School of Business, University College Dublin, Ireland
davide@ucd.ie

Abstract. The Bacterial Foraging Optimization (BFO) algorithm is a biologically inspired computation technique which is based on mimicking the foraging behavior of E.coli bacteria. This paper illustrates how a BFO algorithm can be constructed and applied to solve parameter estimation of a EGARCH-M model which is then used for calibration of a volatility option pricing model. The results from the algorithm are shown to be robust and extendable, suggesting the potential of applying the BFO for financial modeling.

1 Introduction

This paper illustrates the financial application of a biologically-inspired computation technique (see [1] for a general introduction to biologically inspired algorithms), the Bacterial Foraging Optimization (BFO) algorithm introduced by Passino [14] in 2002, which models the foraging behavior of *Escherichia coli* bacteria present in our intestines.

The algorithm has been developed and applied to solve various real-world problems [7,8,10,19], in a number of application domains. Mishra [11] shows that BFO can converge to the global optimum faster than the canonical genetic algorithm. Kim [8] suggests that the BFO could be applied to find solutions for difficult engineering design problems. In this paper, we examine the potential of applying BFO algorithm - with and without swarming effect - within the financial domain. We employ BFO to estimate the parameters of the EGARCH-M model which is a nonlinear problem. The results are used to price the volatility options.

The paper is organized as follows: Section 2 provides a concise overview of the BFO algorithm. Section 3 gives background information of volatility option pricing and section 4 outlines the experimental methodology adopted. Section 5 provides the results of these experiments followed by conclusions in Section 6.

2 The Bacterial Foraging Optimization (BFO) Algorithm

Natural selection tends to eliminate animals with poor foraging strategies and favors the propagation of genes of those animals that have successful foraging

M. Giacobini et al. (Eds.): EvoWorkshops 2008, LNCS 4974, pp. 113–122, 2008.

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 optimization process: animals search for nutrients to maximize the energy obtained per unit time spent foraging, in the face of constraints presented by its own physiology. The *E.coli* bacteria present in our intestines also undertake a foraging strategy [14].

Algorithm 1. BFO algorithm

Randomly distribute initial values for $\theta^i, i = 1, 2, ..., S$ across the optimization domain. Compute the initial cost function value for each bacterium i as J^i, and the initial total cost with swarming effect as J^i_{sw}.

for *Elimination-dispersal loop* **do**
  for *Reproduction loop* **do**
    for *Chemotaxis loop* **do**
      for *Bacterium i* **do**
        Tumble: Generate a random vector $\phi \in \mathcal{R}^D$ as a unit length random direction
        Move: Let $\theta^{new} = \theta^i + c\phi$ and compute corresponding J^{new}. Let $J^{new}_{sw} = J^{new} + J_{cc}(\theta^{new}, \theta)$
        Swim: Let m=0
        while $m < N_s$ **do**
          let m=m+1
          if $J^{new}_{sw} < J^i_{sw}$ **then**
            Let $\theta^i = \theta^{new}$, compute corresponding J^i and J^i_{sw}
            Let $\theta^{new} = \theta^i + c\phi$ and compute corresponding $J(\theta^{new})$.
            Let $J^{new}_{sw} = J^{new} + J_{cc}(\theta^{new}, \theta)$
          else
            let $m = N_s$
          end
        end
      end
    end
    Sort bacteria in order of ascending cost J_{sw}The $S_r = S/2$ bacteria with the highest J value die and other S_r bacteria with the best value split
    Update value of J and J_{sw} accordingly.
  end
  Eliminate and disperse the bacteria to random locations on the optimization domain with probability p_{ed}. Update corresponding J and J_{sw}.
end

Here, the objective is to find the minimum of $J(\theta), \theta \in \mathcal{R}^D$, where we do not have the gradient information $\nabla J(\theta)$. Suppose θ is the position of the bacterium and $J(\theta)$ represents a nutrient profile, i.e.,$J(\theta) < 0, J(\theta) = 0$ and $J(\theta) > 0$ represent the presence of nutrients, a neutral medium and noxious substances respectively. The bacterium will try to move towards increasing concentrations of nutrients (i.e. find lower values of J), search for ways out of neutral media

and avoid noxious substances (away from positions where $J > 0$). It implements a type of biased random walk.

Bacteria can also engage in a form of chemically-mediated 'social communication' during their search process. Let J^i denote the actual cost (or the nutrient surface) at the position of the ith bacterium θ^i. A bacterium that has uncovered good sources of nutrients during its search can release a chemical signal which attracts other bacteria to converge (or swarm) to its current location. This process is mediated by the release of a 'repellent signal' to ensure that the bacteria do not get too close to each other. Including this mechanism in our optimization algorithm, the problem becomes the minimization of $J^i_{sw} = J^i + J_{cc}(\theta_i, \theta)$, which represents the time-varying total cost value for bacterium i. The mathematical swarming (cell-cell signalling) function can be represented by:

$$J_{cc}(\theta^i, \theta) = \begin{cases} -M\left(\sum_{k=1}^{S} e^{-W_a\|\theta^i-\theta^k)\|^2} - \sum_{k=1}^{S} e^{-W_r\|\theta^i-\theta^k\|^2}\right) & \text{With swarming} \\ 0 & \text{No swarming} \end{cases}$$

where $\|.\|$ is the Euclidean norm, W_a and W_r are measures of the width of the attractant and repellent signals respectively, M measures the magnitude of the cell-cell signalling effect. These parameter values are chosen according to the nutrient profile used.

During the lifetime of *E.coli* bacteria, they undergo different stages such as chemotaxis, reproduction and elimination-dispersal. A description of each of these is given below. The details of the algorithm are presented in Algorithm 1.

2.1 Chemotaxis

Chemotaxis is the tendency of a bacterium to move toward distant sources of nutrients. In this process, the bacterium alternates between tumbling (changing direction) and swimming behaviors. Here, a *tumble* is represented by a unit walk with random direction $\phi \in \mathcal{R}^D$ (i.e. $\phi = \frac{\Delta}{\sqrt{\Delta^T \Delta}}$, where Δ is a vector with each element a random number on [-1,1]), a *swim* is indicated as movement in the same direction as the previous tumble. After one step move, the new position of the ith bacterium can be represented as $\theta^{new} = \theta^i + c\phi$, where $\theta^i \in \mathcal{R}^D$ indicates the position of the ith bacterium across the optimization domain. c is the chemotactic step size taken in the direction of ϕ. In this paper, we consider a fixed step size c for all bacteria.

If at θ^{new}, the total cost J^{new}_{sw} is better (lower) than the cost at θ^i, another swimming step 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.

2.2 Reproduction

After N_c chemotactic steps, a reproduction step is taken. In reproduction, the least healthy bacteria die and the other S_r healthiest bacteria each split into two

bacteria, which are then placed in the same location. The health of the bacteria are measured by J_{sw}, higher cost represents that the bacterium did not get as many nutrients during its life of foraging (hence is not as "healthy") and thus unlikely to reproduce.

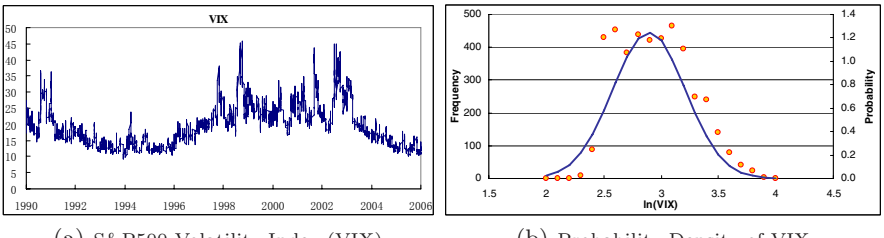
2.3 Elimination - Dispersal

Let N_{ed} be the number of elimination-dispersal steps. The elimination-dispersal step happens after N_{re} reproduction steps. In elimination-dispersal, individual bacterium is stochastically selected for elimination from the population and is replaced by a new bacterium located at a random new location within the optimization domain, according to a preset probability p_{ed}. This mimics the real-world process whereby bacteria can be dispersed to new locations, for example via wind dispersal.

3 Estimation of Volatility Option Pricing Model

Volatility is a measure of how much a stock can move over a specific amount of time. In 1993, the first measure of volatility in the overall market - the S&P 500 Volatility Index (VIX) was created. It is a 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.

Options are financial instruments that convey the right, but not the obligation, to engage in a future transaction on some underlying assets. 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. 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.



(a) S&P500 Volatility Index (VIX) (b) Probability Density of VIX

Fig. 1. Empirical Results of the VIX

Detemple and Osakwe [3] provide analytic pricing formulae for European volatility options[1] assuming that volatility follows a mean-reverting log process (MRLP). We consider MRLP to be a reasonable assumption with small misidentification error, as can be seen from Fig.1, the volatility tends to revert to some long-running average (mean-reversion properties) and the probability density function of the volatility based on unconditional distribution of VIX could be approximated by a lognormal distribution. The closed from expression for the European call option c_t written on the volatility V, with strike price K and maturity T is [3]:

$$c_t(V_t, K, \tau) = e^{-\lambda\tau}[V_t^{\phi_\tau}\exp(\frac{\alpha}{\lambda}(1-\phi_\tau) + \frac{1}{2}a_\tau^2)\mathrm{N}(d_\tau + a_\tau) - K\mathrm{N}(d_\tau)] \quad (1)$$

$$\begin{aligned} \text{where } d_\tau &= \frac{1}{a_\tau}[\phi_\tau \ln(V_t) - \ln(K) + A_\tau] \\ \phi_\tau &= \mathrm{e}^{-\lambda(\tau)} \\ A_\tau &= \frac{\alpha}{\lambda}(1-\phi_\tau) \\ a_\tau &= \frac{\alpha}{\sqrt{2\lambda}}(1-\phi_\tau^2)^{\frac{1}{2}} \end{aligned}$$

and where $\tau = T - t$ is the time-to-maturity, t is the current time, V_t is the volatility of the underlying asset at time t. The above model is a function of parameters α, λ, and σ. α/λ denotes a long run mean for log (V), and $exp\left((\alpha + \frac{1}{4}\sigma^2)/\lambda\right)\sqrt{285}$ denotes a long run mean annualized volatility (based on 285 days). These parameters for the option pricing model can be calculated as below by estimating the corresponding EGARCH-M (the exponential GARCH[2] in mean, first proposed by Nelson [13]) model and then taking the limit [4,12]:

$$\alpha = \frac{K}{2} + \frac{A_1}{\sqrt{2\pi}}\ , \quad \lambda = 1 - G_1\ , \quad \sigma = \frac{1}{2}\sqrt{(\frac{\pi-2}{\pi}){A_1}^2 + {L_1}^2} \quad (2)$$

The EGARCH-M model has estimation issues such as choice of starting values and choice of the optimization routine [15]. We employ BFO to optimize the EGARCH-M model parameters: K, G_1, A_1 and L_1, the details are described in the following section.

4 Experimental Approach

Due to the existence of noise in the newly-traded volatility option data, we calibrate the MRLP option pricing model by estimating the corresponding discrete

[1] A European call option on an asset V_t with maturity date T and strike price K is defined as a contingent claim with payoff at time T given by $\max[V_T - K, 0]$.

[2] GARCH models are popular econometric modeling methods, having been initially specified by Engle [5] and Bollerslev [2], they are specifically designed to model and forecast changes in variance, or volatility per se.

time EGARCH-M model and then taking the limit. The EGARCH-M model is an asymmetric model designed to capture the leverage effect, or negative correlation, between asset returns and volatility. The EGARCH-M (1,1) model considered in this paper is set up as follows:

Conditional mean model:

$$y_t = C - \frac{1}{2}\sigma_t^2 + \varepsilon_t \tag{3}$$

where $\varepsilon_t = \sigma_t z_t,\ z_t \sim N(0,1)$ and y_t represents the log returns of S&P 500

Conditional variance model:

$$\log\sigma_t^2 = K + G_1 \log\sigma_{t-1}^2 + A_1(|z_{t-1}|) + L_1 z_{t-1} \tag{4}$$

where $z_{t-1} = \frac{|\varepsilon_{t-1}|}{\sigma_{t-1}}$

The left-hand side of equation 4 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. In equation 3, the coefficient of σ_t^2 is fixed at -0.5, and the constant C is assumed to be 0.0005, hence the parameters to be estimated are those appeared in equation 4, namely, K, G_1, A_1, and L_1. Where K is the conditional variance constant, G_1 (GARCH term) is the coefficients 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 EGARCH-M 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-M models specified in equations 3 and 4, the objective is to maximize the log likelihood function (LLF) as follows:

$$LLF = -\frac{1}{2}\sum_{t=1}^{T}[\log(2\pi\sigma_t^2) + \frac{\varepsilon_t^2}{\sigma_t^2}] \tag{5}$$

The residuals ε_t and the conditional variances σ_t^2 are inferred by recursive substitution based on equations 3 and 4, given the observed log return series, the current parameter values and the starting value of $z_1 \sim N(0,1)$ and $\sigma_1^2 = exp(K)$. Since minimizing the negative log-likelihood ($-LLF$) is the same as maximizing the log-likelihood(LLF), we use $-LLF$ as our nutrient function (the objective function). And the goal is to minimize the $-LLF$ value, by optimizing parameters K, G_1, A_1, L_1 within the search domain.

The EGARCH-M model is fitted to the return series of S&P 500 daily index using the BFO algorithm (a modified version of Passino's original Matlab code [20]). The S&P 500 (Ticker SPX) equity index is obtained from CBOE,

with data drawn from 02/01/1990 to 30/12/2006, giving a total of 4084 daily observations.

5 Results

The estimated parameters are constrained within [-1, 1]. During the searching process for optimal parameter value, if it breaches the lower/upper bound, its value is set to be -1 or 1. In each run of the BFO, we use parameter values specified in Table 1, they are chosen based on trial and error experimentation.

Fig.2 depicts the evolution of the objective function value, measured using negative maximum likelihood ($-LLF$), as a function of the iteration number for a random single run of the algorithm for BFO with, and for comparison purposes without, the swarming effect. Obviously, the BFO algorithm containing swarming effect has quicker convergence than BFO without this mechanism, though the accuracy of the final results are not too different. Figs. 3(a), 3(b), 3(c) and 3(d) depict the evolution of the parameters K, G_1, A_1 and L_1 as a function of the iteration number for a random single run of the BFO algorithm with the swarming effect. In the early iterations BFO mainly performs global search for the optimum value and displays quick convergence. Later in the run, the focus switches to local optimal search.

Running both BFO with and without swarming effect over 30 trials, we obtain the results shown in Table 2. Where 'Optimum' provides the best results over 30 runs for the objective value $-LLF$ and relevant parameters K, G_1, A_1, L_1. The best results averaged over 30 runs and the standard deviation of the best results over 30 runs are reported in the 'Mean' and 'S.D.' respectively. In order to provide a benchmark for the accuracy of the results obtained by BFO, a Matlab (Ver.7.0.1(R14)) optimizer *fmincon*was used. This optimizer uses sequential quadratic programming (SQP) methods, which closely mimic Newton's method for constrained optimization.

From Table 2, the BFO algorithm with swarming outperforms that without swarming effect. The objective value (the minimal $-LLF$) of -14242.27 is close

Parameter Values and Definition	
D=4	Search space dimension
S=50	Bacteria population size
N_c=20	No. of chemotactic steps
N_s=4	No. of swimming steps
N_{re}=4	No. of reproduction steps
N_{ed}=2	No. of elimination-dispersal
p_{ed}=0.25	Prob. for elimination-dispersal
c=0.007	Chemotactic step size
M=10	Magnitude of swarming effect
W_a=0.2	Coefficients of attractant effect
W_r=10	Coefficients of repellent effect

Table 1. BFO Parameters

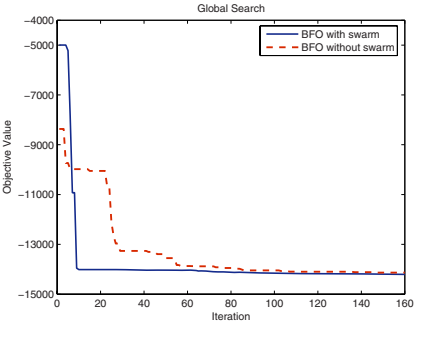
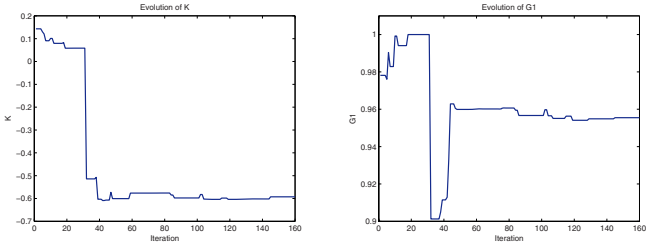
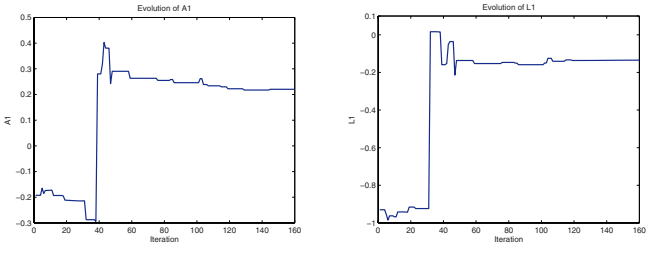


Fig. 2. Objective value vs. Iteration



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



(c) Evolution of A_1 (d) Evolution of L_1

Fig. 3. Evolution of parameters as a function of the iteration number

Table 2. Results of BFO with and without swarming effect

Method		Output	$-LLF(Obj)$	K	G_1	A_1	L_1
BFO	With Swarming	Optimum	-14242.27	-0.474	0.965	0.182	-0.121
		Mean	-14064.26	-0.685	0.956	0.382	-0.086
		(S.D.)	(206.69)	(0.219)	(0.020)	(0.230)	(0.183)
	No Swarming	Optimum	-14241.72	-0.539	0.960	0.198	-0.125
		Mean	-13841.24	-0.726	0.945	0.302	-0.119
		(S.D.)	(1072.5)	(0.212)	(0.030)	(0.224)	(0.164)
Matlab Optimizer		Optima	-14242.99	-0.491	0.963	0.178	-0.114

to (though slightly higher) than -14242.99 obtained in Matlab using the *fmincon* function. The result is acceptable and the standard deviation is small, which suggests the applicability and potential of BFO in finance. Also, the bacterial swarming improves the searching effect, which could be further investigated in the future study. The estimated optimal parameter values using BFO with swarming effects are: $K = -0.474$, $G_1 = 0.965$, $A_1 = 0.182$, $L_1 = -0.121$. 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, 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 2, the resulting stochastic volatility option pricing model parameters are: $\alpha = -0.164$, $\lambda = 0.035$ and $\sigma = 0.082$. The negative α implied mean reversion with a long run mean for log (V) of $\alpha/\lambda = -4.707$, and a long run mean annualized volatility (based on 285 days) of $exp\left((\alpha + \frac{1}{4}\sigma^2)/\lambda\right)\sqrt{285} = 0.160$ 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 Fig.1.

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

6 Conclusion and Future Work

In this paper, we applied the proposed Bacterial Foraging Optimization (BFO) algorithm for parameter estimation of a EGARCH-M model, the results of which can be used to price volatility options. Compared to traditional parameter estimation methods, BFO provides satisfactory results for this non-linear parameter estimation problem, indicating proof of concept of the potential utility of applying this algorithm to the domain of finance. The results also indicate that the BFO algorithm with a social swarming effect provides quicker convergence and more stable results compared with the BFO algorithm without the swarming mechanism.

In future work, we intend to extend the application of the BFO algorithm to harder higher-dimensional and dynamic optimization problems in finance. These complexities are widespread in financial modeling and pose serious problems for traditional, gradient-based statistic computing methods which might only give local solutions. Hence, it is important that the utility of population-based approaches such as the BFO algorithm is tested in the finance domain. It is noted that there is of yet, still a limited literature on the application of the BFO algorithm for real-world problems. Hence, it is necessary to undertake comprehensive scaling and dynamic environment benchmarking of the BFO algorithm and examine the contribution of BFO compared to other global optimium searching algorithms.

References

1. Brabazon, A., O'Neill, M.: Biologically-inspired Algorithms for Financial Modelling. Springer, Berlin (2006)
2. Bollerslev, T.: Generalized Autoregressive Conditional Heteroskedasticity. Journal of Econometrics 31(3), 307–327 (1986)
3. Detemple, J.-B., Osakwe, C.: The Valuation of Volatility Options. European Finance Review 4(1), 21–50 (2000)

4. Duan, J.-C.: Augmented GARCH (p,q) Process and its Diffusion Limit. Journal of Econometrics 79(1), 97–127 (1997)
5. Engle, R.-F.: Autoregressive conditional heteroskedasticity with estimates of the variance of U.K. inflation. Econometrica 50(4), 987–1008 (1982)
6. Hentschel, L.: All in the Family: Nesting Symmetric and Asymmetric GARCH Models. Journal of Financial Economics 39(1), 71–104 (1995)
7. Kim, D.-H., Cho, J.-H.: Intelligent Control of AVR System Using GA-BF. In: Khosla, R., Howlett, R.J., Jain, L.C. (eds.) KES 2005. LNCS (LNAI), vol. 3684, pp. 854–859. Springer, Heidelberg (2005)
8. Kim, D.-H., Abraham, A., Cho, J.-H.: A Hybrid Genetic Algorithm and Bacterial Foraging Approach for Global Optimization. Information Sciences 177(18), 3918–3937 (2007)
9. Li, M.-S., Tang, W.-J., et al.: Bacterial Foraging Algorithm with Varying Population for Optimal Power Flow. In: Giacobini, M. (ed.) EvoWorkshops 2007. LNCS, vol. 4448, pp. 32–41. Springer, Heidelberg (2007)
10. Mishra, S.: A Hybrid Least Square-Fuzzy Bacterial Foraging Strategy for Harmonic Estimation. IEEE Transactions on Evolutionary Computation 9(1), 61–73 (2005)
11. Mishra, S., Bhende, C.-N.: Bacterial Foraging Technique-Based Optimized Active Power Filter for Load Compensation. IEEE Transactions on Power Delivery 22(1), 457–465 (2007)
12. Nelson, D.-B.: ARCH Models as Diffusion Approximations. Journal of Econometrics 45(1–2), 7–38 (1990)
13. Nelson, D.-B.: Conditional Heteroskedasticity in Asset Returns: A New Approach. Econometrica 59(2), 347–370 (1991)
14. Passino, K.-M.: Biomimicry of bacterial foraging for distributed optimization and control. Control Systems Magazine, IEEE 22(3), 52–67 (2002)
15. St. Pierre, E.F.: Estimating EGARCH-M models: Science or art? The Quarterly Review of Economics and Finance 38(2), 167–180 (1998)
16. Tang, W.-J., Wu, Q.-H., Saunders, J.-R.: A Novel Model for Bacterial Foraging in Varying Environments. In: Gavrilova, M.L., Gervasi, O., Kumar, V., Tan, C.J.K., Taniar, D., Laganá, A., Mun, Y., Choo, H. (eds.) ICCSA 2006. LNCS, vol. 3980, pp. 556–565. Springer, Heidelberg (2006)
17. Tang, W.-J., Wu, Q.-H., Saunders, J.-R.: Bacterial Foraging Algorithm for Dynamic Environments. IEEE Congress on Evolutionary Computation, 1324–1330 (July 2006)
18. Tang, W.-J., Wu, Q.-H., Saunders, J.-R.: Individual-Based Modeling of Bacterial Foraging with Quorum Sensing in a Time-Varying Environment. In: Marchiori, E., Moore, J.H., Rajapakse, J.C. (eds.) EvoBIO 2007. LNCS, vol. 4447, pp. 280–290. Springer, Heidelberg (2007)
19. Ulagammai, M., Venkatesh, P., et al.: Application of Bacterial Foraging Technique Trained Artificial and Wavelet Neural Networks in Load Forecasting. Neurocomputing 70(16-18), 2659–2667 (2007)
20. Original matlab codes of Passino can be obtained from, `http://www.ece.osu.edu/~passino/ICbook/ic_index.html`