International Journal of Swarm Intelligence and Evolutionary Computation
ISSN: 2090-4908

Shah, Int J Swarm Intel Evol Comput 2014, 4:2
DOI: 10.4172/2090-4908.1000123

Research Article | Open Access

Cuckoo Search Optimization for Black Scholes Option Pricing

Manas Shah*

Department of Aerospace, Indian Institute of Technology, Bombay, India

Abstract

Black Scholes option pricing model is one of the most important concepts in modern world of computational finance. However, its practical use can be challenging as one of the input parameters must be estimated; implied volatility of the underlying security. The more precisely these values are estimated, the more accurate their corresponding estimates of theoretical option prices would be. Here, we present a novel model based on Cuckoo Search Optimization (CS) which finds more precise estimates of implied volatility than Particle Swarm Optimization (PSO) and Genetic Algorithm (GA).

Keywords: Black scholes model; Cuckoo search optimization; Particle swarm optimization; Genetic algorithm

Introduction

F Black and M Scholes formulated a theoretical model to price options in 1973 [1]. This model is basically a partial differential equation and produces closed form solution for the price of European Call and European Put options with assumption that underlying security price follows lognormal distribution. According to this model, option price (P) depends on various input variables such as S, current security price; K, strike price of the option contract; R, risk free rate of return; T, expiration time of the option contract; IV, implied volatility of the security. In practice, the implied volatility of security can only be estimated, and its estimation has been an interesting subject of research [2,3]. The objective of the vast research has been to determine 'good' estimate of implied volatility which can be then used to calculate the theoretical price of an option. Subsequently, the investors can then lookout for mispriced option in market and generate arbitrage profits.

The complexity is that the implied volatility is a non-linear function of other input parameters and hence we need to apply some type of optimization technique. Newton-Raphson method, a traditional calculus based optimization technique was applied by S Manaster and G Koehler [4]. Their study exposes two major drawbacks of this application:

- For many options, no value of implied volatility can justify the observed option price.
- It is highly sensitive to the starting point; if one fails to have a correct starting value, convergence might not occur.

Also, for calculus based optimization techniques, finding a global optimum for nonlinear function for which analytical derivative cannot be found is problematic as they tend to hung up on local optimum. Later, K. Bruce analyzed with Genetic Algorithm (GA), an evolutionary optimization technique, where he demonstrated GA can more precisely estimate accurate values of implied volatility than calculus based optimization methods [5]. Post that, S Lee, J Lee, D Shim and M Jeon applied Particle Swarm Optimization (PSO), another evolutionary optimization technique [6]. They proved that implied volatilities obtained through PSO are much closer to accurate values of implied volatility than GA.

A new evolutionary search algorithm, called Cuckoo Search (CS), based on cuckoo bird's behavior has been developed by X Yang and S Deb [7]. CS algorithm is known to provide more robust and precise results than PSO algorithm [8, 9]. In this paper, we apply CS to Black Scholes model and compare its performance to PSO and GA. The present paper intends to show that CS can more effectively estimate the accurate values of implied volatility than PSO and GA.

The paper is organized as follows: Section 2 briefly describes Black Scholes model and CS technique. In Section 3, we apply CS, PSO and GA methods to Black Scholes model for European Call option price and compare the results. In section 4 and 5, we discuss the results and draw the conclusion.

Overview

Black scholes model and implied volatility

Our study considers European Call option only. The Black Scholes model for Call option is as follows:

$$C = S \cdot N(\mathrm{d}_1) - X \cdot e^{r} f^{T} \cdot N(d_2) \qquad (1)$$

$$d_2 = d_1 - \sigma T^{\frac{1}{2}} \qquad (2)$$

$$d_1 = \frac{\left(\ln\left(\frac{s}{x}\right) + rfT\right)}{\left(\sigma T^{\frac{1}{2}}\right) + 0.5\sigma T^{\frac{1}{2}}} \qquad (3)$$

where C is the call option price, S is the current security price, X is the strike price, T is the time remaining until expiration expressed as a percent of a year, r_f is the current continuously compounded risk-free interest rate, σ is the implied volatility of security price and N(·) is the standard normal cumulative distribution function.

The model makes certain assumptions which include:

- The options are European options, i.e., they can be exercised only at expiration.

***Corresponding author:** Manas Shah, Department of Aerospace, Indian Institute of Technology, Bombay, India, Tel: 918268590985; E-mail: mns167@gmail.com

Received September 12, 2015; **Accepted** October 24, 2015; **Published** October 29, 2015

Citation: Shah M (2015) Cuckoo Search Optimization for Black Scholes Option Pricing. Int J Swarm Intel Evol Comput 4: 123. doi: 10.4172/2090-4908.1000123

- The risk free rate and implied volatility remain constant over the period of analysis.
- The underlying security price follows the log normal distribution.
- The underlying security does not pay any dividends.

In practice, the supply and demand dynamics for a particular option governs it price and gives us the market price of that option. Whereas the implied volatility in the equation, as discussed, cannot be directly observed and hence should be deduced using numerical techniques.

To explain mathematically, the Black Scholes model determines the option price as follows:

$$c = f(\sigma) \tag{4}$$

where f is basically the Black Scholes pricing model that depends on σ, along with S, X, r_p, T. In terms of σ, the function f is monotonically increasing, which means higher value of σ leads to higher value of option price (C). Hence, using inverse function theorem, there can be at most only one value of σ that would result in a particular value of C.

So now, assuming an inverse function $g = f^{-1}$ such that:

$$\sigma \bar{c} = g(\bar{c}, .) \tag{5}$$

where $\bar{c}$ is the market price and $\sigma \bar{c}$ is the volatility implied by the market price $(\bar{c})$, or the implied volatility. As this now cannot be expressed as closed form solution, we need to use numerical techniques to obtain solution for implied volatility.

Cuckoo search optimization

With recent development, metaheuristic algorithms inspired by nature are widely used to solve hard optimization problems. These algorithms are based on random Monte-Carlo technique, guided by some nature inspired intelligence, particularly evolution and swarm intelligence. For all such nature inspired algorithms, the rudimentary idea is balance between discovering larger area of search space (exploration) and probing a limited region of search space (exploitation).

Cuckoo Search Algorithm, a search based on behavior of Cuckoo bird was proposed by X Yang and S Deb [7,10]. It is primarily based on brood parasitism exhibited by some Cuckoo species. In this algorithm, a pattern (Cuckoo) corresponds to a nest and each single attribute of the pattern corresponds to a Cuckoo-egg. A general system equation of this algorithm is based on general system equation of random walk algorithm as follows:

$$X_{g+1;i} = X_{g;i} + \alpha \otimes levy(\lambda) \tag{6}$$

where α is the step size which depends on the problem in hand, although is mostly 1, g denotes the number of current iteration (g=1, 2, 3… maxcycle), i denotes the i^{th} pattern, the product ⊗ means entry wise multiplication and levy flight essentially provides random walk where random step length is drawn from Levy's distribution which has an infinite variance with an infinite mean.

$$levy \sim u = t^{-\lambda}, (1 \prec \lambda =\prec 3) \tag{7}$$

The initial value of j^{th} attribute of i^{th} pattern is determined as follows:

$$X_{g=o;j,i} = rand \cdot (up_i - low_i) + low_i \tag{8}$$

where low_i and up_i are the lower and upper search-space limits of jth attributes, respectively.

This algorithm detects best solution X_{best} at the beginning of each iterative step. Also at this point, step scale factor is calculated as follows:

$$\phi = \left(\frac{\Gamma(1+\beta) \cdot \sin\left(\pi \cdot \frac{\beta}{2}\right)}{\Gamma\left(\left(\frac{1+\beta}{2}\right) \cdot 2^{\left(\frac{\beta-1}{2}\right)}\right)} \right)^{\left(\frac{1}{\beta}\right)} \tag{9}$$

Where β denotes Levy distribution parameter and Γ denotes gamma function. In the standard implementation of this algorithm [10], β=1.5 has been advised.

The evolution of i^{th} pattern X_i starts with the donor vector v, where $v=X_i$. Then, step size is being calculated as follows:

$$stepsize_j = 0.01 \cdot \left(\frac{u_j}{v_j}\right)^{\frac{1}{\beta}} \cdot (v - X_{best}) \tag{10}$$

where u=φ randn [D] and v=randn [D]. The randn [D] function generates a uniform integer between [1 D].

In the next step of the CK algorithm, the donor pattern υ is randomly mutated as follows:

$$v := v + stepsize_j \cdot randn[D] \tag{11}$$

The X_{best} pattern is then calculated. The unfeasible patterns are then manipulated as follows:

$$v_i := \begin{Bmatrix} X_i + rand \cdot (X_{r_1} - X_{r_2}) rand_i \succ p_0 \\ X_i \qquad else \end{Bmatrix} \tag{12}$$

Where mutation probability value (p) =0.25 is advised [10].

Numerical Experiments and Results

We present here a comparison between the performance of CS, PSO and GA for finding implied volatility for the Black Scholes model. For this comparison, we selected 80 different pairs of Strike Price (K), Time to maturity expressed as a percent of a year (T), Risk free rate (R) for a particular Security price (S) and its implied volatility (σ). With this data in hand, we calculated the actual call option price (C) using Black Scholes model for all this 80 pairs. Now the main problem of Black Scholes model is finding an estimate of σ, σc with which the estimated price of the call should be same as the actual price of call option. Thus, now our fitness function is |estimated call value - actual call value|, which should be minimized. The Table 1 shows the input parameters used for 80 different pairs.

In all the test, number of evaluations is same $2*10^4$ (CS: population 20, number of iteration 1,000, PSO: population 20, number of iteration 1,000 and GA: population 100, number of iteration 200). In GA, we use the uniform crossover with 0.6 probability and bit change mutation with 0.05/bit probability. While in PSO, fixings are selected as c1=c2=2, where c1 and c2 are acceleration constants. As this parameter selection is very important, we relied on the values proposed in [11]. Also, the inertia weight is decreased from 0.9 to 0.4 over the iterations as advised to provide improved results [6]. For CS, Levy distribution parameter (β)=1.5 and mutation probability value (p)=0.25 were selected as advised [7,10]. All tests were repeated twenty times and the median values obtained for 80 different pairs were noted as the result. Table 2 summarizes its performance

Clearly, we find that GA is the worst performer among the three

Sr. No.	Security Price	Strike Price	Time in years	Risk free rate	Actual Option Price
1	100	80	0.25	5.00%	20.99377703
2	100	85	0.25	5.00%	16.05615438
3	100	90	0.25	5.00%	11.13257152
4	100	95	0.25	5.00%	6.41211107
5	100	100	0.25	5.00%	2.66483222
6	100	105	0.25	5.00%	0.69609080
7	100	110	0.25	5.00%	0.10593325
8	100	115	0.25	5.00%	0.00927180
9	100	120	0.25	5.00%	0.00047815
10	100	125	0.25	5.00%	0.00001513
11	100	80	0.5	5.00%	21.97555886
12	100	85	0.5	5.00%	17.10663271
13	100	90	0.5	5.00%	12.30675237
14	100	95	0.5	5.00%	7.83218141
15	100	100	0.5	5.00%	4.19226962
16	100	105	0.5	5.00%	1.81050374
17	100	110	0.5	5.00%	0.61657434
18	100	115	0.5	5.00%	0.16476202
19	100	120	0.5	5.00%	0.03480478
20	100	125	0.5	5.00%	0.00589583
21	100	80	0.75	5.00%	22.94725197
22	100	85	0.75	5.00%	18.15634732
23	100	90	0.75	5.00%	13.47894224
24	100	95	0.75	5.00%	9.15372394
25	100	100	0.75	5.00%	5.54330534
26	100	105	0.75	5.00%	2.93521951
27	100	110	0.75	5.00%	1.34446169
28	100	115	0.75	5.00%	0.53146121
29	100	120	0.75	5.00%	0.18199751
30	100	125	0.75	5.00%	0.05440470
31	100	80	1	5.00%	23.90997789
32	100	85	1	5.00%	19.19968426
33	100	90	1	5.00%	14.62883762
34	100	95	1	5.00%	10.40528429
35	100	100	1	5.00%	6.80495771
36	100	105	1	5.00%	4.04609699
37	100	110	1	5.00%	2.17394516
38	100	115	1	5.00%	1.05419583
39	100	120	1	5.00%	0.46249651
40	100	125	1	5.00%	0.18446286
41	100	80	0.25	10.00%	21.97520733
42	100	85	0.25	10.00%	17.09875304
43	100	90	0.25	10.00%	12.22880790
44	100	95	0.25	10.00%	7.47846671
45	100	100	0.25	10.00%	3.44555060
46	100	105	0.25	10.00%	1.03895204
47	100	110	0.25	10.00%	0.18727956
48	100	115	0.25	10.00%	0.01965625
49	100	120	0.25	10.00%	0.00122041
50	100	125	0.25	10.00%	0.00004646
51	100	80	0.5	10.00%	23.90172625
52	100	85	0.5	10.00%	19.14788092
53	100	90	0.5	10.00%	14.42157126
54	100	95	0.5	10.00%	9.86246020
55	100	100	0.5	10.00%	5.85027298
56	100	105	0.5	10.00%	2.87952287
57	100	110	0.5	10.00%	1.14072077
58	100	115	0.5	10.00%	0.35904362
59	100	120	0.5	10.00%	0.08990118
60	100	125	0.5	10.00%	0.01808986
61	100	80	0.75	10.00%	25.78106792
62	100	85	0.75	10.00%	21.14888416
63	100	90	0.75	10.00%	16.55674254
64	100	95	0.75	10.00%	12.12604497
65	100	100	0.75	10.00%	8.11634938
66	100	105	0.75	10.00%	4.85749642
67	100	110	0.75	10.00%	2.55885849
68	100	115	0.75	10.00%	1.17713973
69	100	120	0.75	10.00%	0.47242634
70	100	125	0.75	10.00%	0.16607396
71	100	80	1	10.00%	27.61440702
72	100	85	1	10.00%	23.10065239
73	100	90	1	10.00%	18.63085853
74	100	95	1	10.00%	14.30401155
75	100	100	1	10.00%	10.30815093
76	100	105	1	10.00%	6.88268631
77	100	110	1	10.00%	4.21674484
78	100	115	1	10.00%	2.35762781
79	100	120	1	10.00%	1.20124683
80	100	125	1	10.00%	0.55871636

Table 1: Input parameters.

Performance	Genetic Algorithm	Particle Swarm	Cuckoo Search
Mean Percentage Error	2.26E-03	1.41E-11	1.34E-11
Root Mean Square Error	3.30E-05	1.08E-12	1.03E-12

Table 2: Performance.

methods under study. To further observe and study the difference in performance between CS and PSO, we conducted the same test but with less number of iterations and increased them subsequently. Table 3 shows the comparison of performance for CS and PSO as we increase the number of iterations from 50 to 500, in step of 25.

We can see that with number of iterations being less than 200. CS travels more space and performs less efficiently than PSO. But as we increase the number of iterations, the Root Mean Square Error as well as Absolute Percentage Error for CS falls down quickly and beyond 200, it remains significantly lower than PSO. Figure 1 and Figure 2 shows the convergence of two methods as we increase the iterations. Figure 1 is the analysis with less number (<250) of iterations while Figure 2 is with greater number (>250) of iterations.

Discussion

CS optimization is a relatively recent search heuristic method. Similar to other evolutionary optimization techniques like PSO and GA, they move from one set of point to another with likely improvement using combination of nature based deterministic and probabilistic rules. For all such nature inspired algorithms the elementary concern is balance between exploration and exploitation. Here, we discuss these fundamentals for CS and PSO.

As we see in our study, CS technique explores larger search space and then exploits good found solutions more efficiently than PSO, estimating a more accurate value. Hence, the CS algorithm supplies more

Number of Iterations	Mean Percentage Error		Root Mean Square Error	
	Particle Swarm	Cuckoo Search	Particle Swarm	Cuckoo Search
50	1.78E-02	3.44E-02	0.0002975	0.000556
75	5.56E-03	1.34E-02	0.0001039	0.0002306
100	2.61E-03	5.23E-03	5.06E-05	9.11E-05
125	1.54E-03	2.36E-03	2.57E-05	3.75E-05
150	8.14E-04	9.41E-04	1.23E-05	1.58E-05
175	4.96E-04	5.65E-04	8.15E-06	1.00E-05
200	2.56E-04	2.47E-04	4.33E-06	4.17E-06
225	1.48E-04	1.26E-04	2.95E-06	2.00E-06
250	9.55E-05	6.13E-05	1.54E-06	1.01E-06
275	5.48E-05	2.82E-05	1.07E-06	5.14E-07
300	2.86E-05	1.24E-05	6.05E-07	2.04E-07
325	1.99E-05	7.80E-06	4.18E-07	1.20E-07
350	1.85E-05	4.60E-06	3.82E-07	8.31E-08
375	7.26E-06	2.55E-06	1.42E-07	4.55E-08
400	6.57E-06	1.36E-06	1.21E-07	2.75E-08
425	2.75E-06	7.10E-07	6.68E-08	1.50E-08
450	2.20E-06	3.40E-07	4.56E-08	5.39E-09
475	8.70E-07	1.50E-07	1.64E-08	3.03E-09
500	8.30E-07	1.00E-07	1.62E-08	1.90E-09

Table 3: Comparisons between PSO and CS.

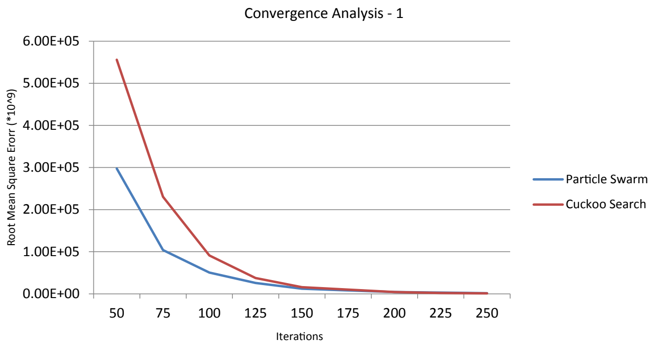

Figure 1: Convergence analysis 1.

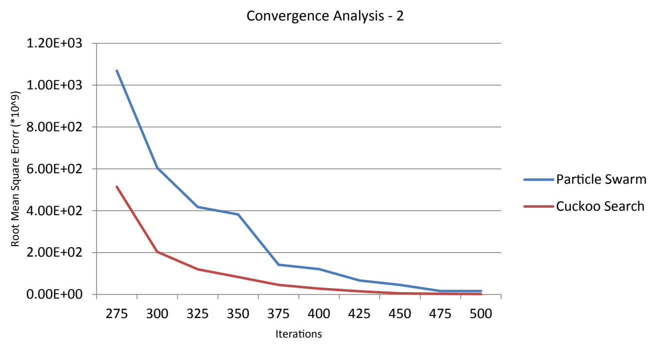

Figure 2: Convergence analysis 2.

robust and precise results [8]. As analyzed in [12], guaranteed global convergence, controlled global search and local search capabilities and global search using Levy' flights, are the reasons for better performance of cuckoo search. The randomization feature via Levy' flights and the controlled global search lead to better exploration and exploitation respectively. Also, CS technique is more computationally efficient than PSO as it uses less number of parameters [12]. The drawback of CS technique is that its convergence rate depends on parameter selection [13].

Conclusion

This paper demonstrates the usefulness and efficiency of CS optimization method in estimating implied volatility for option pricing. We first showed that CS and PSO estimated value of implied volatility is much closer to the actual values than those by GA. When CS was further compared with PSO, we observed relatively greater error for less number of iterations, however significantly lower error as the number of iterations were increased. This paper signifies high potential for widespread use of CS method in field of Finance, where accuracy is of highest importance.

Acknowledgments

Dr. R. P. Shimpi, Professor at Aerospace Department – I.I.T. Bombay, provided assistance with implementation of Cuckoo Search technique.

References

1. Black F, Scholes M (1973) The pricing of options and corporate liabilities, Journal of Political Economy 81: 637-653.
2. Krausz J (1985) Option parameter analysis and market efficiency tests. A simultaneous solution approach, Applied Economics 17: 885-897.
3. Corrado C, Miller T (1993) Optimal volatility estimates based on Black-Scholes implied standard deviations, Working Paper, University of Missouri.
4. Manaster S, Koehler G (1982) the calculation of implied variances from the BlackScholes model: a note. The Journal of Finance 37: 227-230.
5. Bruce K (2000) Black-Scholes option pricing via genetic algorithms. Applied Economics Letters 7.
6. Lee S, Lee J, Shim D, Jeon M (2007) Knowledge-Based Intelligent Information and Engineering Systems, Lecture Notes in Computer Science 4692: 85-92.
7. Yang X, Deb S (2010) Engineering optimisation by cuckoo search, International Journal of Mathematical Modelling and Numerical Optimisation 1: 330-343.
8. Adnan M, Razzaque M(2013) A comparative study of particle swarm optimization and Cuckoo search techniques through problem-specific distance function, International Conference of Information and Communication Technology.
9. Yang X, Deb S (2014) Cuckoo search: recent advances and applications, Neural Computing and Applications 24: 1: 169-1.
10. Civicioglu P, Besdok E (2013) A conceptual comparison of the Cuckoo search, particle swarm optimization, differential evolution and artificial bee colony algorithms, Artificial Intelligence Review 39: 315-346.
11. Yang X, Deb S (2009) Cuckoo search via Levy flights, Proceeding of World Congress on Nature and Biologically Inspired Computing (NaBIC 2009), India. IEEE Publication, USA 210-214.
12. Yang X, Deb S (2014) Cuckoo search: recent advances and applications, Neural Computing and Applications 24: 1: 169-1.
13. Valian E, Mohanna S, Tavakoli S (2011) Improved Cuckoo Search Algorithm for FeedForward Neural Network Training, International Journal of Artificial Intelligence and Applications (IJAIA) 2: 3.

Citation: Shah M (2015) Cuckoo Search Optimization for Black Scholes Option Pricing. Int J Swarm Intel Evol Comput 4: 123. doi: 10.4172/2090-4908.1000123