Pricing Financial Option as a Multi-Objective Optimization Problem Using Firefly Algorithms

by

Gobind Preet Singh

A Thesis submitted to the Faculty of Graduate Studies of
The University of Manitoba
in partial fulfillment of the requirements of the degree of

MASTER OF SCIENCE

Department of Computer Science
University of Manitoba
Winnipeg

Thesis advisor

Dr. Ruppa Thulasiram

Author

Gobind Preet Singh

Pricing Financial Option as a Multi-Objective Optimization Problem Using Firefly Algorithms

Abstract

An option, a type of a financial derivative, is a contract that creates an opportunity for a market player to avoid risks involved in investing, especially in equities. An investor desires to know the accurate value of an option before entering into a contract to buy/sell the underlying asset (stock). There are various techniques that try to simulate real market conditions in order to price or evaluate an option. However, most of them achieved limited success due to high uncertainty in price behavior of the underlying asset. In this study, I propose two new Firefly variant algorithms to compute accurate worth for European and American option contracts and compare them with popular option pricing models (such as Black-Scholes-Merton, binomial lattice, Monte-Carlo, etc.) and real market data.

In my study, I have first modelled the option pricing as a multi-objective optimization problem, where I introduced the pay-off and probability of achieving that pay-off as the main optimization objectives. Then, I proposed to use a latest nature-inspired algorithm that uses the bioluminescence of Fireflies to simulate the market conditions, a first attempt in the literature. For my thesis, I have proposed adaptive weighted-sum based Firefly algorithm and non-dominant sorting Firefly algorithm to

find Pareto optimal solutions for the option pricing problem. Using my algorithm(s), I have successfully computed complete Pareto front of option prices for a number of option contracts from real market (Bloomberg data). Also, I have shown that one of the points on the Pareto front represents the option value within 1-2% error of the real data (Bloomberg).

Moreover, with my experiments, I have shown that any investor may utilize the results in the Pareto fronts for deciding to get into an option contract and can evaluate the worth of a contract tuned to their risk ability. This implies that my proposed multi-objective model and Firefly algorithm could be used in real markets for pricing options at different levels of accuracy. To the best of my knowledge, modelling option pricing problem as a multi-objective optimization problem and using newly developed Firefly algorithm for solving it is unique and novel.

Contents

List of Figures

List of Tables

Acknowledgments

Foremost, I would like to express my sincere gratitude to my supervisor Dr. Ruppa K Thulasiram for his continuous support during my stay at the University of Manitoba and his patience, motivation, enthusiasm, and immense knowledge. His guidance helped me during the time of research and writing of this thesis. I could not have imagined having a better advisor and mentor for my Master's study. Besides my advisor, I would like to thank the rest of my thesis committee: Dr. Pourang Irani and Dr. Yuvraj Gajpal for their precious time, encouragement, and insightful comments.

A very special thanks to my father Tejinder Pal Singh and my mother Surinder Kaur for providing me with unfailing support and continuous encouragement for all of my pursuits in life and inspiring me to follow my dreams. This accomplishment would not have been possible without them. Further, I would also like to thank my sisters, Bani Preet Kaur, Tarun Preet Kaur and my brother Agam Jyot Singh for their emotionally support and love. Also, I would like to thank my uncle Jasminder Pal Singh and my aunt Shweta Kaur for inspiring me and continuously motivating me to achieve something great in life. Special thanks also goes to all my friends and lab mates in Winnipeg for all the stimulating discussions, for the sleepless nights we worked together just before the deadlines, and for all the fun we have had in the last two years. Also, special thanks to all my friend back home in India for their support in my stressful times and for all the cheerful memories we celebrated.

Last but not the least, I would like to thank the almighty God for providing me this opportunity and granting me the capability to proceed successfully. Also, for motivating me to always take the right path in my life giving me the strength to make this world a better place.

This thesis is dedicated to grandmother, "Surjit Kaur" and my grandfather, "Harbhajan Singh". Thanks for their Love and Blessings.

Chapter 1

Introduction

Today's real world applications have become more complex, dynamic and require quick response time. In this research, I have focused on one such application from the world of finance. In financial market, every investor desires rapid information and accurate solutions to earn profits. However, problems in finance are too complex to be solved by a deterministic algorithm(s) in reasonable computing time. Therefore, researchers in the field of computational finance are constantly working towards developing efficient models and algorithms in order to help investors in their decision-making process and provide them with as accurate solution as possible.

In this research, my focus is on one such complex and fundamental problem in finance known as the option pricing problem. Option is a popular type of a derivative contract used for variety of purposes from risk analysis to portfolio management. Every investor in the financial market desires to know the accurate worth of an option contract. However, due to the dynamic and volatile nature of the financial market and complex behaviour of the option contract, accurately pricing options becomes a

difficult task.

In 1973, Fischer Black and Myron Scholes [3] along with Robert Merton [4] revolutionized the financial market by introducing a closed-form solution to price an option. However, this model (Black-Scholes-Merton (BSM) model) had some drawbacks and lacked in depicting the real market conditions appropriately. They made an assumption about the market volatility to remain constant for the underlying asset and restricted its applicability to only simpler type of options such as European option (which allows exercising (buy/sell) only at the expiration date). Following this, Cox-Ross Rubenstein et al. [5] proposed a discrete time approach known as the binomial lattice model to evaluate an option in the risk-neutral regime put forth by the BSM model. Due to the time discretization, this model allowed pricing of other styles of options as well, such as an American option (which allows exercising (buy/sell) before the expiration date). However, in order to estimate the accurate worth of a contract using the binomial lattice model, large number of computations are required. Therefore, there is always a demand for the techniques that would price option accurately and within some reasonable computing time. There are many other numerical techniques in literature such as Monte-Carlo simulation [6], fast fourier transform [7] and others [8, 9, 10]) that are used to price options. However, these techniques also had some limitations and were not able to capture the real market conditions appropriately.

Therefore, due to lack in accurate and efficient techniques to price an option contract, researchers started looking for some unconventional techniques with the aim of capturing the real market conditions as accurately as possible. One such class

of technique is nature-inspired meta-heuristic algorithms [11] such as particle swarm optimization [12], ant colony optimization [13], etc. These algorithms have gained immense popularity over the last decade and were primarily developed to efficiently solve hard problems in a reasonable amount of time. In the recent past, various meta-heuristic algorithms were proposed to solve the option pricing problem [14, 15, 16, 17] and showed satisfactory results.

Moreover, recently a new nature-inspired algorithm, Firefly algorithm [18] was introduced and in many studies [18, 19, 20] it was found to be more efficient as compared to the other nature inspired techniques, in terms of finding a better optimal solution for N-P hard problems and in terms of the speed of convergence towards an optimal solution. Getting motivation from these studies, in my research, I designed a new model where I mapped option pricing problem as a multi-objective optimization problem and used Firefly algorithm to find its solution. In contrast to the previous studies of nature inspired algorithm used for pricing option, I have aimed to consider more parameters while pricing the option and capture real market conditions more accurately. In the previous studies [14, 15, 16, 17], researchers considered option pricing problem as a single objective optimization problem not as a multi-objective optimization problem.

Rest of my thesis is organized as follows: In the next couple of subsections, I briefly introduce financial option contract, meta-heuristic algorithm and multi-objective optimization problem. In Chapter 2 and Chapter 3, I provide details about some of the conventional and unconventional option pricing techniques used in the financial market and about the the basic Firefly algorithm, respectively. In Chapter 4, I ex-

plain my proposed model to map option pricing as a multi-objective optimization problem and describe the designed Firefly algorithm used to find its solution. Further, in Chapter 5, I describe the experimental setup and explain the results from my experiments along with the evaluations and error analysis. Finally, I conclude my study in Chapter 6 and suggest few research directions to explore in future.

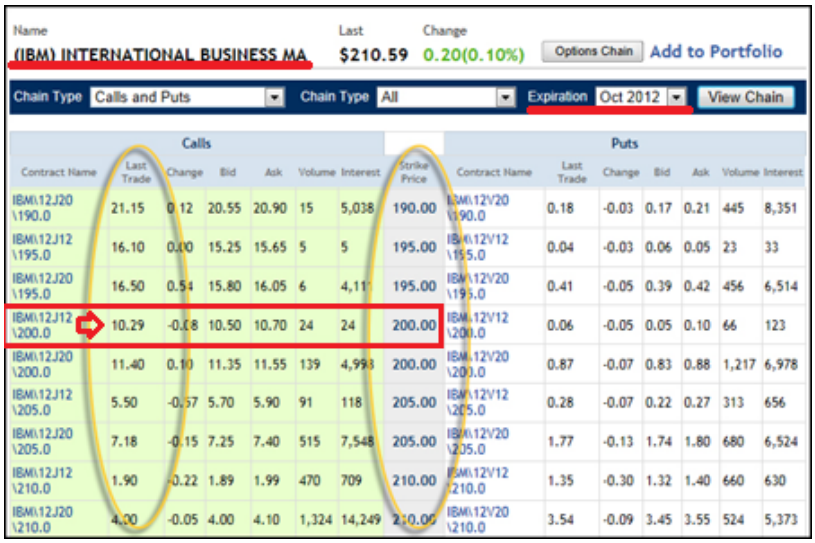
1.1 Financial Option Contract

Financial Option is a standard type of derivative contract between two parties where, one gets the right to buy/sell an asset at a fixed price on or before a certain date, whereas the other person has to oblige the decision of the first party. But in order to get into an option contract, second person is paid some premium at the time of getting into the contract. The first party who gets the right to buy/sell is known as holder whereas second party who gets into an obligation is known as the writer of option contract.

Options are categorized into two types: *call* option contract and *put* option contract. In call option, the holder gets the flexibility to purchase an asset at a fixed price for a specific period of time. Here, holder anticipates the asset price will increase (beyond the contract price) before the expiration to make profit. In put option contract holder gets the flexibility of selling an asset at a specific price for a fixed period of time. Here, holder hopes the asset price to fall (below the contract price) before the contract expires, to make profit. In both the contracts writer is obligated to fulfill the holder's decision, i.e. sell or buy the asset based on the type of contract.

Based on when and how the option is exercised (i.e., style), options can be cat-

egorized as either vanilla (European, American) or exotic options (Asian, Russian) etc. In European style of contract, holder can exercise only on the expiration date and in American style of contract, holder gets flexibility to exercise on or before the expiration date. Other styles of option are called exotic options (Asian, Russian, etc.) and are much more complex in nature. For example, in Asian options, payoff depends on the average price of the underlying asset over a certain period of time. In this research my main focus is to price American and European style of contracts.



Name (IBM) INTERNATIONAL BUSINESS MA — Last $210.59 — Change 0.20(0.10%) — Options Chain — Add to Portfolio

Chain Type: Calls and Puts — Chain Type: All — Expiration: Oct 2012 — View Chain

Calls								Puts						
Contract Name	Last Trade	Change	Bid	Ask	Volume	Interest	Strike Price	Contract Name	Last Trade	Change	Bid	Ask	Volume	Interest
IBM\12J20\190.0	21.15	0.12	20.55	20.90	15	5,038	190.00	IBM\12V20\190.0	0.18	-0.03	0.17	0.21	445	8,351
IBM\12J12\195.0	16.10	0.00	15.25	15.65	5	5	195.00	IBM\12V12\195.0	0.04	-0.03	0.06	0.05	23	33
IBM\12J20\195.0	16.50	0.54	15.80	16.05	6	4,11	195.00	IBM\12V20\195.0	0.41	-0.05	0.39	0.42	456	6,514
IBM\12J12\200.0	10.29	-0.08	10.50	10.70	24	24	200.00	IBM\12V12\200.0	0.06	-0.05	0.05	0.10	66	123
IBM\12J20\200.0	11.40	0.10	11.35	11.55	139	4,993	200.00	IBM\12V20\200.0	0.87	-0.07	0.83	0.88	1,217	6,978
IBM\12J12\205.0	5.50	-0.57	5.70	5.90	91	118	205.00	IBM\12V12\205.0	0.28	-0.07	0.22	0.27	313	656
IBM\12J20\205.0	7.18	-0.15	7.25	7.40	515	7,548	205.00	IBM\12V20\205.0	1.77	-0.13	1.74	1.80	680	6,524
IBM\12J12\210.0	1.90	-0.22	1.89	1.99	470	709	210.00	IBM\12V12\210.0	1.35	-0.30	1.32	1.40	660	630
IBM\12J20\210.0	4.00	-0.05	4.00	4.10	1,324	14,249	210.00	IBM\12V20\210.0	3.54	-0.09	3.45	3.55	524	5,373

Figure 1.1: Option chain for the October 2012 IBM contract shows the various premiums and strike prices. [1]

Pricing an option contract means finding/computing the worth of an option contract. As mentioned earlier, at the time of getting into an contract, the buyer (holder) pays a premium to the seller (writer) in return of the flexibility provided by the option contract and the value of this premium is known as the option price. For example, consider Figure 1.1, here, if a buyer wants to get into a contract where he/she gets the right to purchase 100 shares of *IBM* stock at a strike price of $200 on or before

October 2012, he/she has to pay $10.29 per share as a premium at the start of a contract to the writer. Whether the investor exercises the contract or not he/she has to pay the premium and is non-refundable.

Five basic parameters that influence the price of an option are:

- **Underlying asset price**: It is one of the most influential variables in evaluating the worth of an option contract. It is the current value of an underlying asset.
- **Strike price**: It is the predetermined price of an asset at which holder can buy/sell the asset in future, also known as the contract price.
- **Volatility**: It is the degree of price movement during the contract period, measured using the standard deviation of the underlying asset price.
- **Time until expiration**: It is the time between the start and expiration of the option contract.
- **Risk-free rate of interest**: It is the rate of return from an investment with zero risk. It signifies the minimum return an investor might expect from any investment.

Another important term in an option contract is known as the pay-off of the contract which represents the (local) profit that investor has earned on exercising the contract and is calculated using the equation: maximum(S-K, 0) and maximum(K-S, 0) for the call and put option contracts respectively [8].

In the literature, there are various models that use previously defined parameters to price an option contract and are discussed briefly in the Chapter 2.

1.2 Meta-heuristic Algorithms

These are optimization algorithms that have gained immense popularity in recent past for solving complex optimization problems and are found to be more powerful than conventional deterministic algorithms. Important characteristics these algorithms possess are: self-organization and decentralized decision making. Another important feature that make these algorithms unique and efficient is their tendency to trade-off between the exploration and exploitation [21]. During the intensification/exploitation phase these algorithms exploit the current information at hand, to find the optimal solution. In other words, we can say that during the search procedure, these algorithms explore the search space around the currently best known solution, in order to find a more fitter (better) solution. In diversification/exploration phase, these algorithms tries to ensure that they explore the search space thoroughly and efficiently. Main objective for these algorithms is to efficiently solve large scale problem in less computational time.

Nature has been a principal source of inspiration for many meta-heuristic approaches in their ability to imitate the best features in nature. Generally nature-inspired algorithms are divided into three main categories [22]: (1) Swarm intelligence, (2) Evolutionary algorithms , (3) Bacterial foraging algorithms. In this study, I will mainly focus on swarm intelligence technique where algorithms are inspired by the collective behavior of different kind natural phenomena such as swarm of fireflies or flock of birds or colony of ants. These algorithms also have been applied to many real-world problems [23, 24, 25, 26, 27] and have shown satisfactory results. For my thesis study, I am using Firefly algorithm, a latest swarm intelligence algorithm, for

pricing options.

1.3 Multi-objective Optimization

In an optimization problem when there are more than one objective function that needs to be optimized, the task to find optimum solution is known as *multi-objective optimization*. Most real-world application falls under this category due to their complexity. Fundamentally, single objective problem and multi-objective problem are very different and require contrasting approaches to get their solutions.

In multi-objective optimization problem, multiple objectives may have a conflicting behaviour; that is, optimal solution for one objective may not correspond to optimal solution for other objectives and vice-versa. Hence, we can say that each objective may correspond to different optimal solution. This can be explained more easily with an example. Let us consider buying a car as our goal where selecting the car with minimum cost and maximum performance are our optimization objectives. There are number of cars in the market ranging from few thousand dollars to few hundred thousand dollars. Similarly, performance of these cars also range from low to high. If the cost (budget constraint) is the only objective in your mind then you will definitely buy the car with the minimum price tag (represented as solution 1 in Figure 1.2). Similarly, if the performance is the main objective , you may not mind paying high price for a high performing car (represented as solution 5 in Figure 1.2). When you want a car which is low in cost and high in performance then you have to trade-off between both the objectives as they both do not go hand-in-hand and in fact have an opposite behavior. Best solution for one need not be the best for the

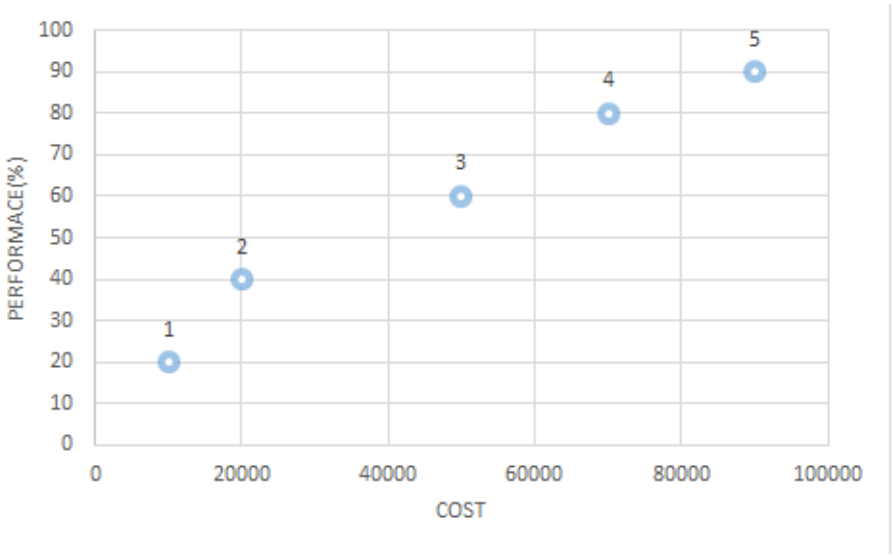


Figure 1.2: Car buying as Multi-objective problem

other. Hence, you have to decide for the car that trades-off between the cost and the performance on the basis of your needs and pocket.

In an multi-objective optimization problem we have number of trade-off solutions at hand and it is not possible to say which solution is the best for all the objectives. Hence, in any multi-objective optimization problem there can not be single optimum (best) solution that optimizes all the objectives simultaneously. There has to be a set of optimum solution with varying degree of objective values. This set of optimal solution for a multi-objective optimization problem is known as Pareto optimal solution. Joining all these possible solutions form a Pareto optimal front. In Figure 1.3 Pareto front for the various combination of bi-objective optimization problem are shown (represented using the highlighted line). For any multi-objective optimization algorithm, our main aim is to find the set of Pareto-optimal solution or the Pareto front. As it is clear by now that in a multi-objective optimization problem it is difficult to optimize one objective without adverse effects on other objective(s).

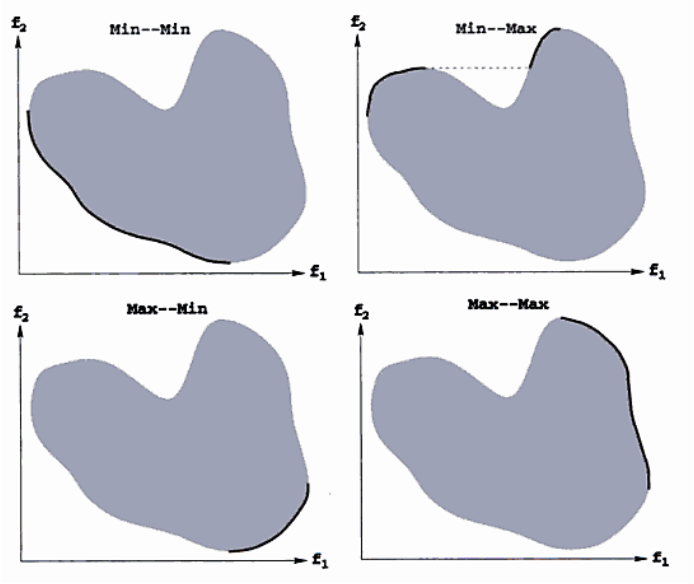


Figure 1.3: Pareto optimal solution for four combination of bi-objective optimization problem [2]

Therefore, we aim to find the Pareto front, which includes all those solution that are better in at-least one objective when compared to all other solutions in the search space but are not worse in any single objective. In other words, we can say that Pareto optimal front comprises of all the non-dominating solution from within the search space. Concept of non-domination is explained further in Chapter 4.

Chapter 2

Option Pricing Techniques

In this chapter, I discuss few conventional and unconventional option pricing techniques used in the financial market.

2.1 Black-Scholes-Merton (BSM) Model

In 1973 three researchers, Fischer Black, Myron Scholes and Robert C. Merton revolutionized the financial market by introducing a closed-form solution to price European options. Their model was inspired from partial differential equations used for the heat flow problems. For their contribution to financial market, Robert C. Merton and Myron Scholes won the Nobel Prize in Economics in 1997.

Initially, Black-Scholes-Merton model [3] was applicable only to European style of option. This model assumed that the contract can only be executed on expiration and assumed underlying asset volatility to remain constant thought out the contract period. These assumption did not capture the real market scenario but still the

model gained immense popularity and it is still being used by many practitioners to evaluate the option contract. Many mathematicians extended this Black-Scholes-Merton model to find approximate solution for various other types of options like American option and other exotic options (see for example, [9, 28]).

The solution for the Call option contract using classical Black-Scholes-Merton model [3, 4] is given by

$$c = (N(d_1) \times S_o) - (N(d_2) \times Ke^{r(T-t)})$$

where,

$$d_1 = \frac{(r + \sigma^2/2)(T-t) + ln(S/K)}{\sigma\sqrt{(T-t)}}$$

$$d_2 = d_1 - \sigma\sqrt{(T-t)}$$

and the solution for the Put option contract using classical Black-Scholes-Merton model is given by

$$p = (N(d_2) \times Ke^{-r(T-t)}) - N(-d_1) \times S_o$$

In these solutions, S_o represents the initial asset price, K represents the strike price, r represents the interest rate, σ represents volatility and t represents the expiration time.

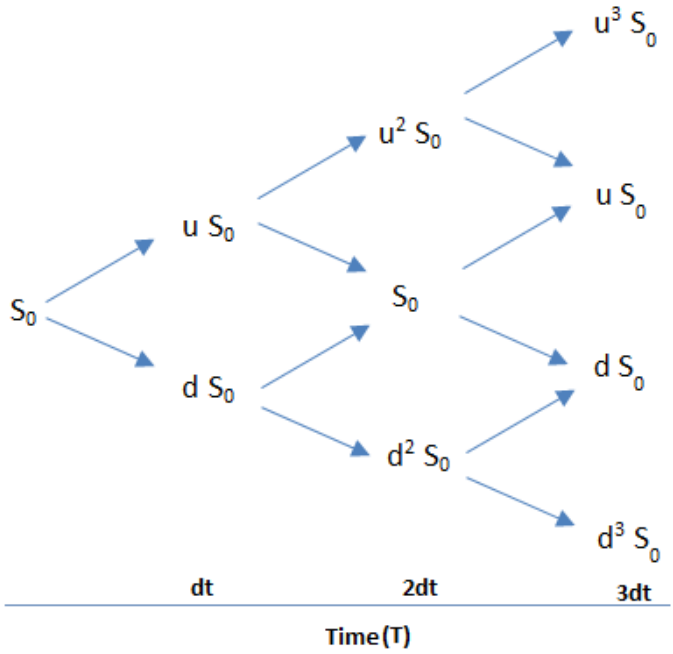


Figure 2.1: Binomial Tree

2.2 Binomial Lattice Model

As mentioned earlier, the Black-Scholes-Merton model has made an assumption of treating the volatility for the underlying asset to remain constant and hence lacks in depicting real market conditions. In 1979, Cox-Ross Rubenstein [5] proposed a discrete time approach to evaluate the option contract known as the binomial lattice model. This model involves constructing a binomial tree as shown in Figure 2.1 which depicts different possible path that an underlying asset might follow during the contract period. At each time step, underlying asset has a possibility of moving up or down with certain probability.

In the binomial tree structure S, T, σ, dt represent stock price, time, asset volatility and time difference between two steps, respectively. In Figure 2.1 other two variables u

and d, represent the amount by which stock price increases and decreases respectively. We can calculate u and d using the following equations 2.1 and 2.2 [5].

$$u = e^{\sigma\sqrt{\Delta T}} \tag{2.1}$$

$$d = e^{-\sigma\sqrt{\Delta T}} = \frac{1}{u} \tag{2.2}$$

In the binomial lattice model [5] first a binomial tree is constructed as shown in Figure 2.1 and then the pay-off is calculated at each node. Pay-off at the leaf nodes are calculated first using the simple pay-off equation, max (S-K, 0) or max (K-S, 0) for call and put option, respectively. Then, the pay-off on the rest of nodes is calculated using the equation

$$f = (e^{-rT}(pf_u + (1-p)f_d) \tag{2.3}$$

where,

$$p = \frac{e^{rt} - d}{u - d} \tag{2.4}$$

In the above equations, f_u is the pay-off at the upper node (asset price increase) and f_d is the pay-off of the lower node (asset price decrease) where p and *(1-p)* are the respective probabilities of reaching these prices. In the binomial lattice model, number of steps can vary (one step, two steps, n steps) and as we increase the number of steps (in other words, as we decrease the size of the (time) steps), model tends to move towards continuous-time BSM model. However, with very large tree, finding solutions for option value becomes computationally expensive.

2.3 Monte-Carlo Simulation for Option Pricing

Another popular model for pricing options is Monte-Carlo (MC) simulation introduced in year 1977 [6]. In Monte-Carlo simulation a process is repeated a number of times attempting to predict possible future outcomes and at the end of the simulation, distribution produced from these trials is analyzed. In the case of option pricing, outcomes are the future asset (stock) price.

In Monte-Carlo simulation, it is assumed that stock follows a risk-neutral random path and requires a model representing the behavior of the stock price. One of the most common models in finance is the geometric Brownian motion (GBM). Here, the stock evolution is described using equation 2.5 [6]:

$$dS(t) = \mu S(t)dt + \sigma S(t)dB(t) \tag{2.5}$$

where S is the asset price, μ is the drift of stock price, σ is the volatility of stock and $B(t)$ is the Brownian motion. Monte-carlo simulation for option pricing assumes Brownian motion as a normally distributed random variable with zero mean and variance T (Time to expiration). Also, it assumes drift rate equal to the risk-free interest rate (r). The future stock price using Monte-Carlo simulation is given by equation 2.6 [6, 29]

$$S(T) = S_o e^{(r-\frac{\sigma^2}{2})+\sigma\varepsilon\sqrt{T}} \tag{2.6}$$

Algorithm 1 captures the description above for pricing option using Monte-Carlo simulation [6, 29].

Algorithm 1 Monte-Carlo simulation for pricing option

1: **Input :** Intial Value(S_o), Strike Price(K), Expiration Time (T), Volatility (σ), Risk free rate of interest (r)
2: Initialize number of Simulation and t=0
3: **while** $t < Numberof Simulation$ **do**
4: Generate a Normally Distributed Random Number (ε) from range (0,1)
5: Calculate Asset Value using equation $S(T) = S_o e^{(r-\frac{\sigma^2}{2})+\sigma\varepsilon\sqrt{T}}$
6: Compute the pay-off using equation max(S(T)-K, 0) or max(K-S(T), 0) for call and put option respectively
7: **end while**
8: Average pay-off of all simulations is calculated and then discounted to compute the option price

2.4 Meta-heuristic Techniques for Option Pricing

During the last few decades, heuristic approaches have gained immense popularity and have been applied to various financial applications. Most of the problems in finance are too complex to be solved by deterministic or analytic strategies and therefore, there is a need for powerful approximation algorithms or strategies that could find their solutions in reasonable computational time. Some of the applications in finance where implementation of meta-heuristic approach have shown significant results are implied volatility [30], time series forecasting [31] and portfolio selection [32]. As all these applications lacked efficient analytical strategies or closed-form solution to find their solution, therefore, contemporary numerical techniques i.e. nature-inspired meta-heuristic algorithms were applied and proven with satisfactory results.

Techniques mentioned in the previous subsections for option pricing are efficient, however, getting accurate results from these techniques require intensive computation. Thus, computational finance researchers explored the area of nature-inspired computing to find the accurate option value and with smaller computational cost. In

1994, Hutchinson et al. [33] applied artificial neural network to find the worth of an option contract. Later, in 1998 Chidambaran et al. [16] applied genetic programming approach to evaluate the worth of an option contract. They tried to find a relation between the option price and other parameters that were used to price the option. The authors randomly generated functions (programs) relating the set of input (parameters of options contract) and underlying asset with one single output (option price). One benefit in their approach was that they have the flexibility to incorporate in their search a previously known solution from models such as BSM model, to find better approximation results. Also, the authors claimed that in most of the trials their algorithm performed better than BSM model. Later, Yin and Brabazon [34] proposed an improved adaptive genetic programming algorithm to price options and claimed their strategy as more efficient when compared to the fixed genetic programming for option pricing.

In 2002, Keber and Schuster [35] used swarm-intelligence technique, known as generalized ant programming, in order to evaluate the American style of options for non-dividend paying stocks. They have also claimed to find accurate approximation results. Further, Kumar et al. [14], studied the suitability of ant colony optimization (ACO) technique [13] to find value of an option. In [14] the authors developed two new ACO based algorithms for pricing options named "sub-optimal path generation" algorithm and "dynamic iterative" algorithm. They also studied the efficiency of their techniques to price exotic options (Asian or barrier option) and claimed their algorithm performed better than other numerical techniques like the binomial lattice model [5] and Monte-Carlo simulation [6]. Later, Thulasiram et al. [15] considered

another popular technique, particle swarm optimization (PSO) [12] for valuation of option. These authors mapped each particle of PSO to capture both the profit (pay off) and the exercise time. They also incorporated varying volatility in their algorithm to represent real market condition to price both European and American style option.

For my thesis, I am using a latest nature inspired technique, Firefly algorithm for pricing options. As stated before, according to many studies Firefly algorithm has been proven as a more efficient algorithm [18, 19, 20, 36] as compared to other nature-inspired techniques. Further details about the Firefly algorithm is described in next Chapter.

Chapter 3

Firefly Algorithm

Firefly Algorithm is one of the recent nature-inspired meta-heuristic algorithms introduced by Xin She Yang in 2008 [21]. It belongs to the group of stochastic algorithms as it uses randomization in searching for the set of solutions. It is inspired from the flashing lights of Fireflies. Pseudo code of the Firefly algorithm is described in Algorithm 2 [21].

Fireflies are one of the most charismatic creatures in nature and their behaviour have been studied by various researchers [37, 38, 39, 40, 41, 42]. Their main feature is that they have ability to generate flashing lights. These flashing lights are produced due to a biochemical process called bio-luminescence. Primary purpose for such flashing light is to act as a signaling system to attract mating partners or to warn potential predators. There are around two thousand Firefly species that produce short and rhythmic flashes and pattern observed for most of these species are unique [37].

Typically first signalers are the male trying to attract female on ground and in

Algorithm 2 Firefly Algorithm

```
Input: α (Randomness),γ (Attractiveness),β_o (Light Intensity at source)
Randomly initialize population for all Fireflies x_i where i = (1, 2, ...n)
For each individual i evaluate fitness value f(x_ik) within dimensional search space
k for k=(1,2...d)
Light intensity I_i at x_i is it fitness value f(x_i)
while t < MaxGeneration do
    for i = 1 : n do
        for j = 1 : n do
            if I_i > I_j then
                Move Firefly j towards i in all dimensions-d using equation 3.4
            end if
            Attractiveness varies with distance r via exp[−γr]
            Evaluate new solutions and update their fitness value
        end for
    end for
    Rank the Fireflies and find best of current iteration
end while
```

response female also emits continuous or flashing lights. Both the mating partners emit distinct flashing signals that are precisely timed in order to encode information like sex or species identity. Typically, females are attracted toward male flashing brighter lights. Also, it is a scientific fact [21] that the flash intensity decrease with increase in distance from source but some female species get confused between the distant flashes produced by stronger light source and closer light flashes produced by weaker light source.

Based on the flashing light characteristics, two important issues needed to be defined are: (i) variation of the light intensity and (ii) formulation of attractiveness. While formulating the Firefly algorithm Xin-She Yang made three assumptions [21]: (i) all Fireflies are uni-sexual, so that one Firefly will be attracted to all other Fireflies, (ii) attractiveness is proportional to their brightness, and for any two Fireflies, the less

brighter one will be attracted by (and thus move to) the more brighter one; however, the brightness can decrease as their distance increases, (iii) if there are no Fireflies brighter than a given Firefly, it will move randomly. In the Firefly algorithm, we treat light intensity I of Firefly as proportional to the solution fitness value; that is $I \propto F(x)$, while intensity of light follows the inverse square law and is varied according to the following equation 3.1 [21]:

$$I(r) = I_o e^{-\gamma r^2} \tag{3.1}$$

where I_o denotes light intensity of the source and γ denotes the fixed light absorption coefficient. The attractiveness β of Fireflies is proportional to the light intensities $I(r)$. Therefore, attractiveness is expressed using equation 3.2 [21]

$$\beta = \beta_o e^{-\gamma r^2} \tag{3.2}$$

where β_o is attractiveness at r=0. Light intensity I and attractiveness β is similar in some way. Intensity refers to the absolute measure of emitted light by the Fireflies and attractiveness refers to the measure of light seen by the other Fireflies.

The distance between two Fireflies in the basic Firefly algorithm is calculated using Euclidean distance and is measured using the following formula 3.3 [21]:

$$r_{ij} = |x_i - x_j| = \sqrt{\sum_{k=1}^{k=n} (x_{ik} - x_{jk})^2} \tag{3.3}$$

where n is the dimensionality of the problem. Different application may have different

expression to calculate distance between two Fireflies but in my approach I am using the Euclidean distance equation. The movement of i^{th} Firefly towards more attractive j^{th} Firefly is measured with following equation 3.4 [21]:

$$x_i = x_i + \beta_o e^{-\gamma r^2}(x_j - x_i) + \alpha\varepsilon_i \tag{3.4}$$

where ε_i is the random number drawn from Gaussian distribution. Movement of all the Fireflies basically consist of three terms: the current position of i^{th} Firefly, attraction to another Firefly and random walk with α as a randomization parameters. Value of all the parameters in a Firefly have crucial impact on search procedure of the Firefly algorithm.

For any application to be solved by an optimization algorithm, modelling the problem as an optimization problem is very crucial. Hence, in my thesis I have mapped the option pricing problem as a multi-objective optimization problem. Further, in order to find the solution for a multi-objective optimization problem, a multi-objective algorithm is also required. Therefore, in my thesis I have also developed a multi-objective variant of the Firefly algorithm to find the solution for option pricing. Both my model and algorithm to find solution for option pricing problem are explained in detail in next Chapter.

Chapter 4

Multi-Objective Firefly Algorithm for Pricing Options

In this chapter I explain my proposed model of option pricing as a multi-objective optimization problem and provide details about the solution methodology using Firefly algorithm to find it's solution. First, in Chapter 4.1, I present details on how I modelled option pricing as multi-objective optimization problem. In Chapter 4.2, I explain the algorithm used for probability computation. Then, in Chapters 4.3 and 4.4, I explain the two variants of multi-objective Firefly algorithm that I have designed and developed to find solutions for the option pricing problem.

4.1 Modelling

This part of the study is one of the most important contributions of my research. Here, I explain the modelling of option pricing problem as an optimization problem.

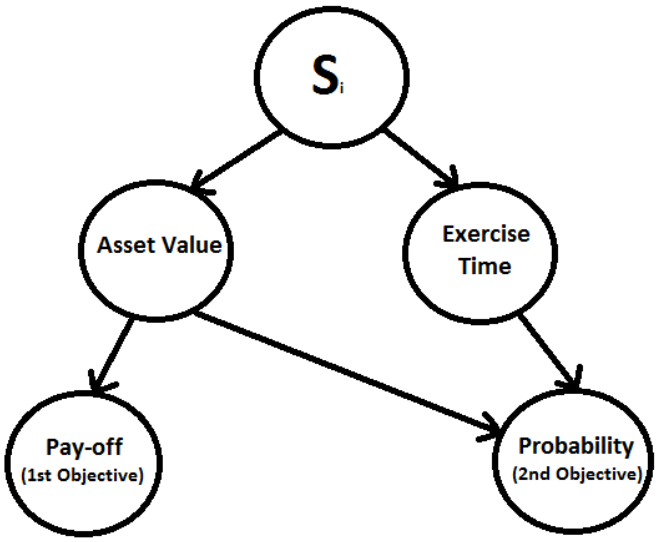


Figure 4.1: Mapping

This is a novel idea and have not been introduced in the past. For my model, I consider search space for an option pricing problem as a 2-dimensional space where each point in the space, represents a solution equal to the worth of an option contract (maximum possible profit expected on exercising the contract under given constraints).

Each solution in the search space consists of two option parameters (dimensions): value of the underlying asset and exercise time. The first dimension, the value of the underlying asset, represents any feasible value that the underlying asset can reach during the life of an option. The second dimension, the exercise time, represents any time between the start and expiration of the contract.

Considering the complexity of an option contract and volatile behavior of the underlying asset, it is not easy to find an optimal solution for an option pricing problem. Therefore, to capture the accurate behavior and find good approximation solution, I consider maximum pay-off by the contract and probability of attaining

that payoff as the two main objectives for my problem. Pay-off is the profit expected on exercising the contract and probability is the chance of earning that profit in current market condition(s). Pay-off is calculated using max(S-K, 0) for a call option and max(K-S, 0) for a put option where S is the stock price at the time of exercise and K is the strike price and probability is calculated using the algorithm explained in Chapter 4.2. To the best of my knowledge, none of the studies using nature-inspired techniques have considered probability of achieving a particular pay-off as an objective in pricing an option. Therefore, in my approach I aim to maximize the combination of both pay-off and probability in whole to accurately price the option. This makes my approach unique and novel. I have constructed the fitness function of a solution for option pricing problem as the function of two optimization objectives: profit (pay-off) and probability of achieving that profit.

$$FitnessValue(F(x)) = F(Payoff, Probability) \tag{4.1}$$

In this function, I aim to maximize both the profit and probability of achieving the pay-off simultaneously, to compute the accurate worth of the option contract. This gives rise to a multi-objective optimization problem. At this point, I would like to make a note that two objectives considered in my model behave in such a manner that when probability is high, the rate of change in pay-off is low and vice-versa. For example, let us consider a call option contract for a stock ABC whose $S_0 = 90$, $K = 90$ and $T = 60$ days. In Figure 4.2, a chart is plotted between price of stock ABC and probability of reaching that stock price. Here, we can see that as the price of stock increases pay-off also increases but the probability of attaining that pay-off decreases.

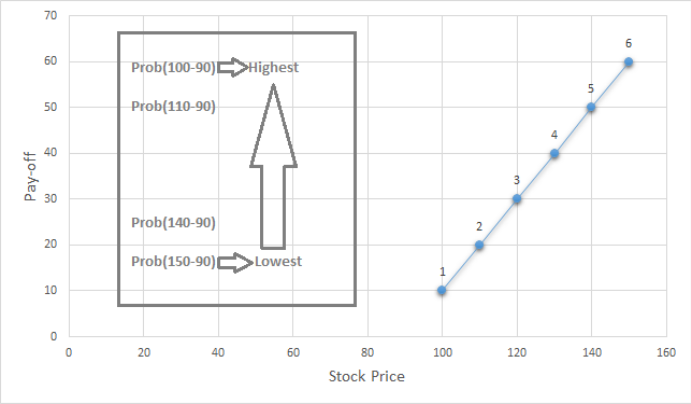


Figure 4.2: Relationship between Pay-off and Probability

Therefore, in order to efficiently optimize both the objectives and compute accurate set of solutions for the option pricing problem, I have designed and developed two variants of multi-objective Firefly algorithm to efficiently find Pareto optimal solution for option pricing problem. These algorithms are explained in Chapters 4.3 and 4.4.

4.2 Probability Computation

Investors would like to get as accurate information as possible on the probability of any underlying asset reaching a particular value in future (during the contract period). In my research, I have used the Monte-Carlo simulation to compute the probability of an asset reaching a particular value in future. This is described in Algorithm 3.

The input to the algorithm consists of four parameters: initial asset value, target asset value for which probability is to be calculated, number of trading days and volatility. To estimate the probability, we compare the target asset value to the final simulated price from Monte-Carlo simulation (Line 7). Line 8, records the number

Algorithm 3 Probability estimation using Monte-Carlo simulation

1: **Input :** Intial Value(S_o), Target Value(V), Number of Trading Days(T), Volatility (σ), Drift rate(μ)
2: **procedure** ProbCal
3: Initialize number of Simulation
4: t=0, counter=0
5: **while** $t < Numberof Simulation$ **do**
6: Generate a Normally Distributed Random Number (ε) from range (0,1)
7: Calculate Asset Value using equation $S(T) = S_o e^{(\frac{\sigma}{2})(2(1+\varepsilon\sqrt{T})-\sigma)}$
8: **if** $S(T) \geq V$ **then**
9: Counter = Counter +1
10: **end if**
11: **end while**
12: Probability = Counter / Total Simulations
13: **return** Probability
14: **end procedure**

of times the simulated random walk of an asset exceeds the target asset value. This value is divided by the total number of simulations giving the probability of an asset to reach or exceed the target value (Line 12 and 13).

4.3 Adaptive weighted sum based Firefly Algorithm

This is the first variant of multi-objective Firefly algorithm, developed to price an option as a multi-objective problem. Reasoning behind developing this algorithm was to test the model of option pricing as multi-objective optimization problem. Results obtained from these experiments were very promising due to which I developed another more advanced variant of the multi-objective Firefly algorithm, which I explain in Chapter 4.4.

4.3.1 Weighted-sum approach

One of the simplest approaches to simultaneously optimize more than one objective is the weighted-sum approach [2]. In this classical approach, it is required to combine two or more individual objectives into one composite function by assigning weight w_i to all i objective functions, based on their importance. Composite equation that combines multiple objectives is given as [2]:

$$F(x) = w_1 f_1 + w_2 f_2 + ...w_k f_k \tag{4.2}$$

In this equation, vector $[w_1, w_2, ...w_k]$ denotes weights for each objective k and $[f_1, f_2, ...f_k]$ represent their respective fitness values. Formalized equation to calculate fitness value for option pricing problem is:

$$F(x) = w_1 Payoff(x) + w_2 Probability(x) \tag{4.3}$$

where w_1 and w_2 are weights for the respective objectives. Option pricing problem involves great degree of uncertainty due to the volatile behavior of the underlying asset, which motivated me to find solutions that are accurate enough to real market values. Therefore, in my approach I simulate this optimization algorithm with different combination of weights and tried to find a set of Pareto optimal solutions, which are diverse enough and satisfy each objective at different acceptable levels. I tried to trade-off both the objectives at different possible levels and find a set of solutions that represent Pareto optimal front. However, while doing the experiments, I discovered that Pareto front for option pricing problem has convex region with non-uniform

curvature (flat curvature). In convex problems where curvature of Pareto front is non-uniform, traditional weighted-sum based approach only finds solution in the region where curvature is really high and do not find any solutions where the curvature is mostly flat [43]. A similar situation arises in my problem and I was able to find solutions only at the extreme corners of the Pareto front. To address this critical issue, I used adaptive weighted-sum method [43] to find the desired Pareto front for the problem at hand.

In adaptive weighted-sum approach, feasible search space evolves according to the nature of solutions. At each iteration, inequality constraints are imposed on the feasible search space to reduce it by a factor δ, which is the distance between the two extreme points on feasible search space. Then, a sub-optimization is performed in the new region to find optimal solution of that region. This procedure is followed until the stopping criteria is met, in order to attain final set of Pareto optimal solutions. This procedure described so far is captured in Algorithm 4.

4.3.2 Adaptive weighted-sum based Firefly Algorithm for Pricing Option

Pseudo Code for option pricing using basic weighted-sum based multi-objective Firefly algorithm is described in Algorithm 5.

The inputs to the Firefly option pricing algorithm consist of parameters for the Firefly algorithm (Line 1) and option pricing problem (Line 2). Line 3 randomly initializes the positions of n fireflies within the bounded region. The fitness value of each Firefly (Line 4) is calculated. To do so, it is needed to find the pay-off and

Algorithm 4 Procedure for Adaptive sum based approach

1: **Input :** F_imax (Solution representing maximum fitness for i_{th} objective), F_imin (Solution representing minimum fitness for i_{th} objective)
2: Normalize objective value of all the solutions using equation $f_i(x) = (f_i(x) - f_imin)/(f_imax - f_imin)$ where i=1,2.
3: Select any one of the objective(Probability in my approach) whose search space will be varied throughout the simulation (Putting constraints on search space of one objective will effect other automatically).
4: Variable δ is initialized using equation $\delta = F_{min}$ where F_{min} in minimum value of selected objective i.e. probability
5: Variable n is calculated using Equation $n = (F_{max} - F_{min})/F_{min}$ where n is number of solution to search on Pareto front.
6: **for** $i = 1 : n$ **do**
7: Run optimization algorithm defined in Algorithm 5, to find optimal solution in feasible search space bounded between F_{max} and F_{min}
8: Feasible region for selected variable is varied using equation $F_{min} = F_{min} + \delta$
9: **end for**
10: n number of solutions found represents Pareto Front for the Problem.

probability of achieving this pay-off. The pay-off is calculated using equation, max(S-K, 0) (Line 4.1) and the probability using the algorithm described in Algorithm 3 (Line 2). In Line 4.3 of Algorithm 5, I initialize the two weights, w_1 and w_2, to calculate the fitness value for each Firefly using equation 4.3. In my implementation I have selected 0.6 and 0.4 as the weights for Payoff and Probability respectively, since at each feasible search region, my algorithm is finding solution with maximum pay-off and minimum probability. Then, the algorithm iterates for t number of times for each of the fireflies. Note that as mentioned in Chapter 3, the light intensity describes the fitness value (Line 5). Therefore, all those fireflies with less fitness value will be attracted towards more brighter or fitter fireflies (Line 9-11). At each iteration of this algorithm the position of all unfit or less bright Firefly is varied using equation (3.4) with an aim to move to better locations. That is, I vary the asset value and time

Algorithm 5 Weighted Sum based Multi objective Firefly Algorithm to evaluate Call Option

1: FA Input parameter: α (Randomness), γ (Attractiveness), β_o (Light Intensity at source), Population Size (n), Number of Iteration (t)
2: Option Input Parameters: Initial Asset Value (S), Expiration Time (T), Strike Price (K), Volatility (σ), Risk free rate of interest (r)
3: Randomly initialize position for all fireflies x_i where $i = (1, 2, ...n)$
4: For each Firefly i evaluate fitness value $f(x_i)$ using following steps
  4.1: Calculate Pay-off for each Firefly x_i using equation max(S-K,0), where S represents asset value of each Firefly
  4.2: For each Firefly x_i Call procedure PROBCAL such that $Probabililty(x_i)$= PROBCAL(S,$AssetValue(x_i)$,$Time(x_i)$,σ,r)
  4.3: Initialize weight w_1 for Pay-off and w_2 for Probability
  4.4: Calculate Fitness $F(x_i) = (w_1 * Payoff(x_i)) + (w_2 * Probability(x_i))$
5: Light intensity I_i for each Firefly x_i is represented by fitness value $F(x_i)$
6: **while** $t < MaxGeneration$ **do**
7: **for** $i = 1 : n$ **do**
8: **for** $i = 1 : n$ **do**
9: **if** $I_i > I_j$ **then**
10: Move Firefly j towards i using Equation 3.4 in Asset Value and Time dimensions.
11: **end if**
12: **end for**
13: **end for**
14: Evaluate all new solutions and update their fitness value using step 4
15: Rank all the fireflies x_i on the basis of their fitness $F(x_i)$ and find best of current iteration
16: **end while**
17: Pay-off by the best individual is discounted using equation: $OptionValue = Pay - off * e^{(-rt)}$ where t is time represented by that individual.

dimension of each Firefly such that there is high probability of getting high pay-off. This process continues until the stopping criteria (t) is reached and obtain the best solution (Line 14-15). Finally, the best pay-off received is discounted to get the value of an option (Line 17).

This algorithm searches the pre-defined feasible region and finds the solution with maximum fitness. Fitness of solution for each implementation is computed on the

basis of weights defined. In my implementation, I first define the weights as 0.9 and 0.1 for payoff and probability respectively, to find solution representing maximum payoff and minimum probability. Similarly, I also find a solution with minimum payoff and maximum probability using weights as 0.1 and 0.9 for payoff and probability respectively. Later, these maximum and minimum values for each objective are passed to the adaptive weighted-sum based procedure to find all set of solutions on the Pareto front. In Adaptive weighted-sum based procedure, feasible search space is altered or evolved n number of times using Line 8 of Algorithm 4 and for each evolved region, Algorithm 5 is simulated to find optimal solution for the defined feasible region. After following this procedure I end up with n number of solutions that represent a set of solutions on the Pareto front for the option pricing application.

4.4 Non-Dominated Sorting Firefly Algorithm

In this Chapter, I explain the second variant of the multi-objective Firefly algorithm: non-dominated sorting Firefly algorithm (NSFA). Main inspiration for this algorithm is non-dominated sorting Genetic algorithm (NSGA-II) [44], one of the most popular and efficient multi-objective optimization algorithm. Reason behind developing a new multi-objective algorithm using Firefly algorithm is that in the earlier studies it was shown that for the uni-modal and multi-modal optimization problem, Firefly performed better [18, 20] than the Genetic algorithm. Therefore, we wanted to explore the performance of Firefly for the multi-objective optimization problem.

In a multi-objective optimization problem, all the objectives are considered equally

important and no single solution is said to be an optimal solution on the basis of any single objective. Therefore, the main motivation behind developing any multi-objective optimization algorithm is to find solutions that trade-off all the objectives at hand and represents true set of optimal solutions; that is, to find a set of Pareto optimal solution.

In order to find a true set of Pareto optimal solution for a multi-objective optimization problem, I have considered the search space as a combination of two non-overlapping sets S_1 and S_2 similar to NSGA-II [44], where S_1 contains all non-dominated solution and S_2 contains all dominated solution and main aim for my algorithm is to find the full set of non-dominated solution (S_1), which we consider to represent the Pareto optimal front. Therefore, I first start by explaining the concept of domination and about the procedure required for non-dominated sorting. Further, I explain the working of NSFA for the option pricing problem.

Dominance: "A solution x_1 is said to dominate other solution x_2 if both the condition mentioned below are true [2]:

1. *The solution x_1 is no worse than x_2 in all the objectives.*
2. *The solution x_1 is strictly better than x_2 in at least one objective.*"

If both the condition mentioned above are true then we can also say that the solution x_1 is *non-dominated* by the solution x_2. Also, in a situation when both the solutions x_1 and x_2 do not dominate each other, we can say that both the solutions are non-dominant to each other. Now, in order to explain the concept of dominance more thoroughly, I consider a bi-objective optimization problem with five different solutions as shown in Figure 4.3. Here, function F1 needs to be minimized and function F2

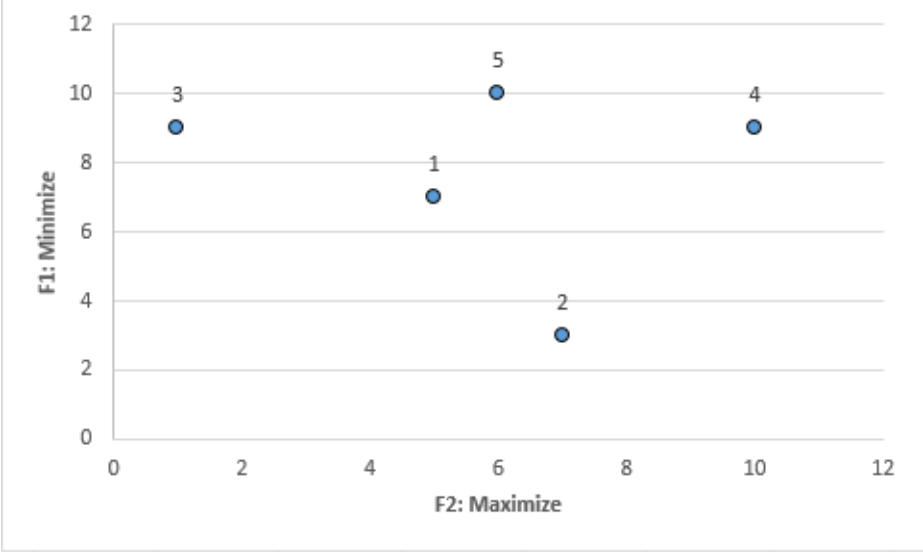


Figure 4.3: Dominance

needs to be maximized. First, let us compare solution 3 and solution 1. Value for function 2 of solution 1 is greater than solution 3 and value for function 1 of solution 1 is less than solution 3, hence both the conditions of dominance are getting satisfied and therefore, we can say that solution 3 dominates solution 1 or solution 3 is non-dominant to solution 1. Next, I take another instance and compare solution 1 and solution 4. It is visible that function 1 of solution 1 is less as compared to solution 4 but value for function 2 of solution 4 is greater than solution 1. In other words, both the conditions of dominance are not satisfied for both the solutions and hence, we can say that both these solutions do not dominate each other or are non-dominant with respect to each other. Similarly, all other solutions can also be compared with each other and a set representing non-dominated solutions from a given set of solutions can be found. I explain the procedure to find a set of non-dominated solution in more details in next paragraph.

Now, with the concept of domination described in previous paragraph, one can

compare any two solution with multiple objectives and find the one that is better. Therefore, it is now possible to find those set of solutions that are better in respect to others from a given set of solutions; that is, one can now easily find non-dominated set of solutions. I have used a very simple approach, which is similar to finding a smallest number from a given set of real numbers. In my approach each solution i is compared with every other solution in the population to check if it gets dominated by any other solution in the population and if such a solution is found then it means that solution i does not belong to non-dominated set of solutions as there exist a solution in population that is better than solution i in all the objectives. If no solution from the population is found to be better than solution i, then we can declare that solution i belongs to non-dominated solution. I explain in Algorithm 7 [2] the procedure to find non-dominated set of solutions from a population of size n.

Algorithm 6 Identify Non-Dominated set

```
1: Input:Set of solutions of size n
2: Output:Set of non-dominated solution
3: Create an empty non-dominated set N'
4: while i < n do
5:     while j < n do
6:         if Solution_j dominates Solution_i then
7:             i → i + 1
8:         else
9:             j → j + 1
10:            if j > n then
11:                Add Solution_i to N'
12:            end if
13:        end if
14:    end while
15: end while
```

After explaining the the concept of domination and procedure used to find non-dominated set, another important concept I need to explain (before getting to un-

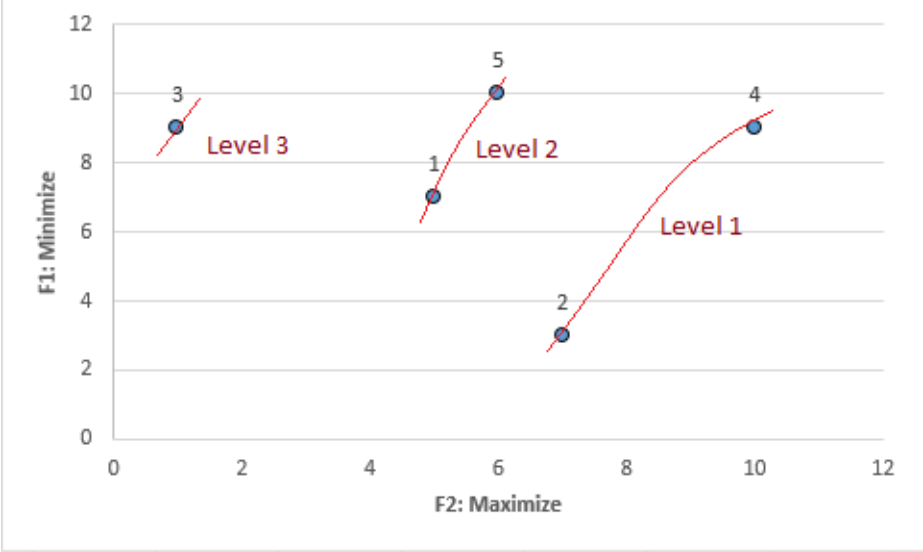


Figure 4.4: Non-dominated Sorting

derstand NSFA) is non-dominated sorting. Algorithm 7 mainly classifies population into two sets: non-dominated set and dominated set. Whereas, in NSFA the entire population need to be classified into various non-domination levels, that is, we need population to be sorted according to ascending level of non-domination. Therefore, Algorithm 7 [2] sorts the entire population such that the best non-domination solution are represented with non-dominated solutions of Level 1. Further, after removing the Level 1 non-dominated solution from the population, the next best non-domination solution are represented with non-dominated solutions of Level 2 and this process goes on until all the solutions are covered under some non-domination level. Figure 4.4 shows how a population shown in Figure 4.3 gets sorted according to their non-dominance level using Algorithm 7.

Another important concept that I need to discuss before explaining the NSFA algorithm is crowding distance metric. In any multi-objective optimization problem there are two main goals: (i) To find a set of solutions closer to the Pareto front and

Algorithm 7 Non-Dominated Sorting

1: **Input:**Population P of size n
2: Set all non-dominated sets N_j(j=1,2,...), as empty.
3: Set non-domination level counter j=1
4: **while** $P \neq \emptyset$ **do**
5: Find non-dominated set N' of population P
6: Update $N_j \rightarrow N'$
7: Remove all the solution of N' from Population P. ($P = P/N'$)
8: $j \rightarrow j + 1$
9: **end while**
10: **Return:** All non-dominated sets P_i for i=1.2.3...j.

(ii) To find a diverse set of solutions. Motivation behind using the crowding distance metric [2] is to bring diversity in the solution set. Crowding distance d_i of a solution i is a measure of search space around it which is not occupied by any other solution in the population. In other words, crowding distance is the density estimator for all the solutions in a population [2]. Also, Crowding distance is computed together for all the solutions having the same non-domination level; that is, we calculate the crowding distance for the solution with respect to the other solutions lying on the same non-domination front. For example let us consider Figure 4.5, here if we want to calculate the Crowding distance d_i of a solution i then the average distance of two solutions on either side of solution i along each of the objectives is calculated (represented by *i+1* and *i-1*) and also all the solution represented using filled dot in Figure 4.5 are consider while calculating the Crowding distance d_i of a solution i as they all belong to same non-domination level. In Algorithm 8 [2], method to calculate the crowding distance is explained in details.

Now, after covering non-dominance and crowding distance concept, I explain in details the non-dominated sorting Firefly algorithm for pricing options.

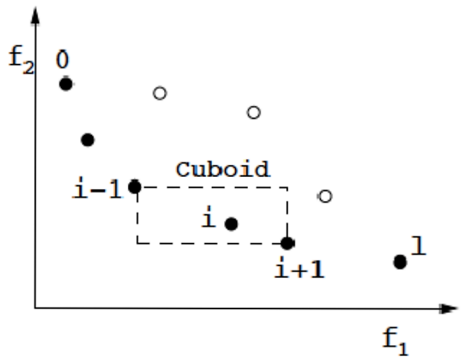


Figure 4.5: Crowding Distance

Algorithm 8 Crowding Distance Estimator

1: Let number of solutions on front F_j to be l. For each solution on front F_j, assign $d_0 = 0$
2: For each objective function m=1,2,3,,M, sort the set in ascending order of f_m.
3: For each objective function m=1,2,3,,M, assign large value to boundary solution and for all other solution i=2 to l-1 assign
$d_(i) = d_(i) + \frac{f_m^{i+1} - f_m^{i-1}}{f_m^{max} - f_m^{min}}$

4.4.1 Non-Dominated Sorting Firefly Algorithm for Option Pricing

In this Chapter, I explain the newly designed non-dominated sorting Firefly algorithm(NSFA) to find Pareto optimal solution for option pricing problem. This algorithm is similar to NSGA-II [44] in a way that I am also using the approach of non-dominated sorting and crowding distance to find Pareto optimal solution for multi-objective option pricing problem. However, as Firefly algorithm is very different from genetic algorithm, the way these approaches are used is very different.

First, in order to simulate NSFA for option pricing problem, one has to set the initial values for Firefly parameters $(\alpha, \gamma, \beta_o n, t)$ and financial option parameters (S, T, K, σ). Then, on the basis of input parameters for the financial options, up-

per bound and lower bound for both the variables S and T of a Firefly are set using the equation defined in line 3a and 3b of Algorithm 10. Here, I am using the value of volatility to set bounds for the variable stock. Explanation behind this approach is that I am trying to incorporate the real market condition in my search. Volatility is the annualized standard deviation of price movement for a stock, which means volatility depicts the magnitude of future price movement of a stock (not the direction). Therefore, using the equation defined in line 3a, future stock price movement is bounded within a range defined by the stock's current level of volatility. In my implementation, I have used twice the value of volatility ($2^*\sigma$) rather than using just the volatility, in order to increase the size of the search space.

Algorithm 9 Procedure FFAMOVE

```
1: Input: previousPopulation: A List with fireflies as its member.
2: Output: newPopulation: A List with updated position of fireflies
3: n= sizeof(previouspopulation)
4: F(x_i) and D(x_i) denotes Fitness and Crowding distance respectively for a Firefly
   x_i where i = (1, 2, 3...n)
5: procedure FFAMOVE
6:     newPopulation → previousPopulation
7:     for i = 1 : n do
8:         for j = 1 : n do
9:             if F(x_i) > F(x_j) then
10:                Move Firefly j towards Firefly i by updating it's position
                   in all the dimensions using Equation 3.4.
11:            else if F(x_i) = F(x_j) then
12:                if D(x_i) > D(x_j) then
13:                    Move Firefly j towards Firefly i by updating it's position
                       in all the dimensions using Equation 3.4.
14:                end if
15:            end if
16:        end for
17:    end for
18:    return newPopulation
19: end procedure
```

After initializing all the parameters and variables, simulation of NSFA starts by randomly setting values for the initial population, where each Firefly represents stock (S) and time (T). After setting the initial values, the pay-off and probability for each Firefly are evaluated. Further, using Algorithm 7 (lines 6 and 7), the non-domination sorting of the initial population is done and the fitness for each Firefly is assigned using equation $F(x_i) = n - NonDomLevel(x_i)$ where n is the size of population and *NonDomLevel* is the non-domination the level of *ith* Firefly. Using this equation all the Fireflies with better non-domination level will have higher fitness and will have more significant impact during the simulation, since in my implementation I am trying to maximize the fitness. Note that a Firefly with more fitness is considered to have higher light intensity.

Further, crowding distance for each Firefly is also computed using the Algorithm 8, which is used to maintain the diversity in the set of solutions. In-order to retain all the non-dominated solution found during the simulation of algorithm, I have used an external archive with the name *nonDomPopulation* that stores all the non-dominated solution. This archive is continuously updated during the simulation and after the completion of simulation non-dominated solutions are returned as the final output for the optimization problem. This archive is first initialized in line 7 of Algorithm 10 where all the non-dominated solution from the initial population are identified and stored in list *nonDomPopulation*.

Now, I explain how NSFA works in-order to efficiently search the bounded space and find a set of non-dominated solutions, which basically represents the true Pareto front for option pricing problem. In line 8 of my Algorithm 10, list *previousPopulation*

is initialized with the contents of the list containing initial population and starts the main part of the simulation where Firefly searches the bounded search space and continuously updates the contents of population. This simulation is executed until the stopping criteria is met. In line 10 of Algorithm 10, newPopulation is constructed using the Algorithm 9 where the position of each Firefly is changed with an aim to reach a better position. Here, better position means a position close to the true Pareto optimal front. Position of Fireflies are changed using the equation 3.4 in which each less fit Firefly is attracted towards a more fit Firefly. In this version of (multi-objective) Firefly algorithm I have made a slight variation. Along with comparing the fitness of each Firefly during the simulation, I am also considering the crowding distance value for each of the Firefly when the fitness of two fireflies are same/equal (lines 10 and 11 of Algorithm 9). We do this because a Firefly with higher crowding distance is considered to be in a position of comparatively less denser area. Therefore, when Firefly from a more denser area is attracted towards a Firefly in less dense area, probability of finding a more diverse set of solution increases.

After the creation of new population, both the list *newPopulation* and *previous-Population* are combined into a new population of size *2n* and named *combPopulation*. Then, this new *combPopulation* is searched for any newly found non-dominated solution and, if found, is added to the archive *nonDomPopulation*. Although this step of combining population and finding non-dominated solution requires more effort, it allows for the global non-domination check and leads to an elitist strategy. Along with adding the non-dominated solution to *nonDomPopulation*, some solutions are also removed for which the status of dominance is changed from non-dominant to

dominant on addition of new solution. This process of updating *nonDomPopulation* is continued until the stopping criteria for simulation is reached. Finally, the archive *nonDomPopulation* is returned as final solution for the problem, which is the Pareto optimal solution.

Algorithm 10 Non-Dominant Sorted Firefly Algorithm to evaluate Call Option

1: FA Input parameter: α (Randomness), γ (Attractiveness), β_o (Light Intensity at source), Population Size (n), Number of Iteration (t)
2: Option Input Parameters: Initial Asset Value (S), Expiration Time (T), Strike Price (K), Volatility (σ)
3: Initialize Upper Bound and Lower bound for each variables representing a Firefly
   a: $S_{min} = S(1 - (2 * \sigma))$ and $S_{max} = S(1 + (2 * \sigma))$
   b: $T_{min} = 1$ and $T_{max} = T$
4: Initialize position for all fireflies x_i where $i = (1, 2, ...n)$ and store the population in the list *initPopulation*
   a: Each Firefly x_i is represented by two variables: Stock and Time
   b: S_i is initialised with random real number within range S_{min} and S_{max}
   c: T_i is initialised with random Integer within a range T_{min} and T_{max}
5: For each Firefly x_i evaluate their objective function values
   a: Calculate Pay-off for each Firefly x_i using equation $max(S_i - K, 0)$, where S_i represents asset value of each Firefly x_i
   b: For each Firefly x_i Call procedure PROBCAL such that $Probabililty(x_i)$= PROBCAL(S,S_i,T_i,σ,r)
6: Sort each Firefly according to their Non Domination level and evaluate their Fitness and Crowding Density
   a: Fitness is calculated as $F(x_i) = n - NonDomLevel(x_i)$
   b: Crowding Distance is calculated by calling procedure $CrowdingSort(x_i)$
7: Identify the non-dominated solution from population and store them in the list *nonDomPopulation*
8: Assign list *previousPopulation* $\rightarrow$ *initPopulation*
9: **while** $t < MaxGeneration$ **do**
10: $newPopulation \rightarrow FFAMOVE(previousPopulation)$
11: Evaluate Fitness and Crowding Distance of *newPopulation*
12: Combine *previousPopulation* and *newPopulation* into a list *combPopulation* of size 2n
13: Identify the non-dominated set of solution from the list *combPopulation* and update the list *nonDomPopulation*. Also remove any solution from the list if it gets dominated by adding new solutions.
14: Assign *previousPopulation* $\rightarrow$ *newPopulation*
15: **end while**
16: Return the final set of *nonDomPopulation* as a set of Pareto optimal solution for Option Pricing Problem

Chapter 5

Results and Analysis

In this Chapter, I present and discuss the experimental set up organized for the study and analyze the results for accuracy and efficiency of my proposed multi-objective option pricing model for both European and American options using the Firefly algorithms (NSFA and Adaptive weighted-sum Firefly Algorithm).

5.1 Experimental setup

In my experiments, first I have used adaptive weighted-sum based Firefly algorithm for pricing European option. Based on the experiences with this algorithm I have used non-dominant sorting Firefly algorithm for further efficiency and accuracy in finding option values for complex American option.

For my experiments, I have gathered real market data from Bloomberg [45]. For European option experiments I have collected European Call Option data for S&P 500 index for July 2013. For American option experiments, I have selected 5 equities:

Apple (AAPL), Google (GOOG), Goldman Sachs (GS), Amazon (AMZN) and IBM (IBM) and have gathered their respective data for the period from January 2015 till June 2015. The efficiency and accuracy of my algorithms and quality and competency of my model is examined by validating the results against the real-world data. Also, I compare my results with results from the binomial lattice model and Monte-Carlo simulation for the American option contract and with BSM model for the European option contract. Since BSM model provides closed-form solution for the European options, I have compared my results with the BSM model for European option. However, for American option I have to compare with other numerical techniques only such as the binomial lattice model [6] and Monte-Carlo simulation [46] as there is no closed-form solution for American option. These techniques are most popular and accurate techniques for evaluating American option.

In my experiments, to estimate the volatility for American option contract, I have used volatility indices for options on individual equities, where CBOE publishes the market's expected volatility on five highly active equities (VXAPL, VXAZN, VXGOG, VXGS, VSIBM) and in order to estimate the volatility for European option, I have used CBOE volatility index (VIX), the best known estimate of volatility for S&P 500 index.

Exhaustively evaluating all the contracts on each date (approx 3500 contract/day) is very difficult (laborious). Therefore, in my study, I have selected four particular days in each month for valuation; that is, four Mondays in each month from January till June 2015 to evaluate American option and four Mondays in the month of July 2013 to evaluate European option. Further, to show the competency of my model on

a broader scale of contracts I have selected different possible types of contracts on the basis of *time to expiration* and *moneyness* ($\frac{S}{K}$), which are shown in Table 5.1 and Table 5.2 respectively for European and American option. In these tables, moneyness means by either at-the money contract ($K = S$) or out-of-the-money contract ($K > S$) or in-the-money contract ($K < S$)

Table 5.1: Variation of contracts based on Time to Expiration and Moneyness (European Option)

Expiration Period	**Moneyness (S/K)**		
	In-the-money	At-the-money	Out-of-the-money
1 Week	0.99; 0.98	1	1.01; 1.03
3 Weeks	0.95; 0.98	1	1.01; 1.03
1 Month	0.95; 0.96	1	1.01; 1.05
3 Months	0.95; 0.96	1	1.01; 1.08
6 Months	0.9; 0.95	1	1.06; 1.13
1 Year	0.9; 0.95	1	1.06; 1.13

Table 5.2: Variation of contracts based on Time to Expiration and Moneyness (American Option)

Expiration Period	**Moneyness (S/K)**		
	In-the-money	At-the-money	Out-of-the-money
1 Week	0.95; 0.98	1	1.02; 1.05
1 Month	0.95; 0.98	1	1.02; 1.05
3 Months	0.90; 0.95	1	1.05; 1.1
6 Months	0.90; 0.95	1	1.05; 1.1
1 Year	0.90; 0.95	1	1.05; 1.1

Decision on parametric value for the Firefly algorithm is very important for it's efficiency in a given application. Therefore, after trying different possible combinations of parameters, final values used in my experimentation are presented in Table 5.3. In addition to these parameters, I introduced a small variation for my implementation; that is, for the initial position of n Fireflies, I first generated 1000 different possible

Fireflies and then sorted them on the basis of their fitness from which I selected *t*op n Fireflies as the initial population. This variation helped me improve the convergence rate of the algorithm.

Table 5.3: Parameter setting for Firefly algorithm

Parameter	**Value**
Population Size	100
Number of Iteration	250
Randomization parameter (α)	0.2
Light absorption coefficient (γ)	1.0
Attractiveness value (β_o)	1.0

After creating the experimental setup, I did different types of experiments to validate my model and algorithm for both European and American option pricing. I did the evaluation of experiments in two parts: (i) I showed the efficiency of my algorithm to find the complete Pareto front for both the style of contracts and (ii) I did the analysis of the solution from my algorithm to evaluate the competency of my model.

5.2 Experimental results for European Option

In this Chapter, I discuss the results from the experiments on European style of option contract. For these experiments, I have used the simpler algorithm i.e. adaptive weighted-sum based Firefly algorithm.

5.2.1 Pareto Front for Option Value

First, I discuss the Pareto front obtained from the implementation of my model for European option. I have executed adaptive weighted-sum based Firefly algorithm on over 200 contracts with different combinations of "maturity" and "moneyness" on different possible dates. In all the experiments performed, my algorithm was successful in finding the set of Pareto optimal solutions for option pricing problem that is clearly observable from the results shown for some of the experiments in Figures 5.1 to 5.4. From Figure 5.1, in which I present the solutions found for 1 year at-the-money (when $S_0 = K$, where S_0 is the asset price at the start of contract) contract on July 1, 2013, it is clear that my algorithm successfully found the solution for maximum and minimum of both the objectives (represented by maximum payoff and minimum payoff arrow in all the figure); and it also successfully found the set of all other non-dominant solutions with respect to both objectives (represented by all the points between the maximum payoff and minimum pay off arrow). Similar behavior is also observed in other contracts shown in Figures 5.1 to 5.4, which validates the ability of Firefly algorithm to find solution for multi-objective optimization problem.

In order to show the quality (accuracy and efficiency) of the Pareto front found using my algorithm, I filtered and selected only 120 experiments (30 different types of contract as tabulated in Table 5.1 on 4 Mondays in July 2013) from over 200 experiments performed since other experiments showed similar behavior. These 120 experiments were adequate to capture all possible behavior of an option contract. For all these experiments I have plotted pay-off (y-axis), probability (x-axis) and error for all solutions on a scatter plot using data visualization tool called Tableau and

some of the visualization are shown in Figures 5.1 to 5.4. Further, the actual worth of the contracts for all these 120 selected experiments were noted from the historical real data, which were later used to determine the quality of the Pareto front; that is, after noting the real pay-off received for each contract, I compute the absolute error for all the solutions found using my algorithm by subtracting the pay-off of each solution point on Pareto front from real pay-off of that contract, which are shown in Figures 5.1 to 5.4 for each solution point. One important insight observed on analyzing all the Pareto fronts is that in 99% of the experiments, my algorithm was successful in capturing the true option value as a solution on the Pareto front with less than 2% error. Here, error is defined as the difference between the actual worth of the contract and the solution on the Pareto front which is close the actual worth of the contract obtained from my model and algorithm.

That is, in almost all the experiments my algorithm was able to find a solution, that is equal to the actual (discounted) option price of a contract as available from historical data. For example (Table 5.4), for at-the-money 1-year contract in Figure 5.1 where S&P 500 index is an underlying asset, S_T=1960.23, S_0 = 1614.96 and K=1625. The actual option value of the contract from historical data is noted as 335 and solution on the Pareto front (computed using my model and algorithm) closest to the actual worth of the contract has an option value of 328.6, which leads to an error of 6.4. This solution point is identified with an arrow in Figure 5.1 (zoomed in). Similarly, for all other contracts Firefly algorithm was able to capture the actual option values in the Pareto front as identified with arrows in Figures 5.2 to 5.4. This shows that the Firefly algorithm for option pricing using probability and pay-off as

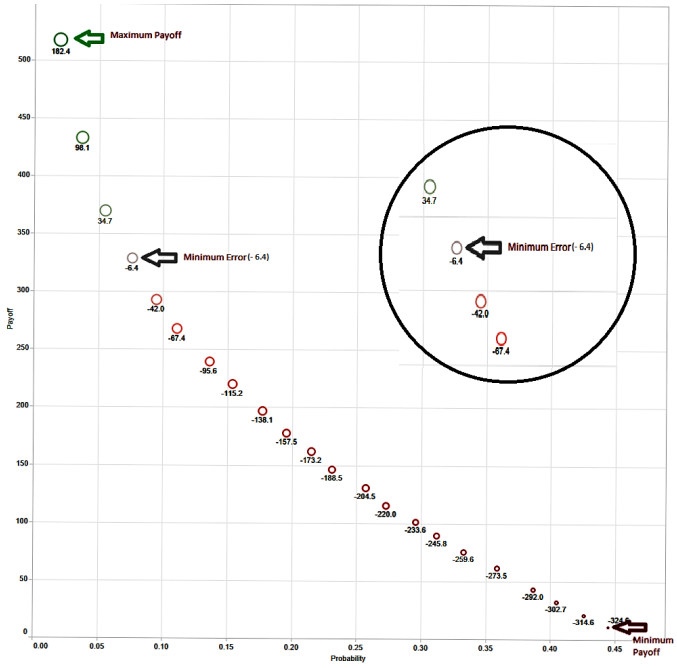


Figure 5.1: Pareto Front for at-the-money 1 year contract on July 1, 2013

objectives is very efficient and is able to capture the accurate real market scenario. Pareto optimal solutions found using my strategy is successful in accurately capturing all possible pay-off that one might get from such a contract. This also validates the competency of my model of formulating the option pricing as a multi-objective optimization problem.

Table 5.4: Experimental Results for European Option

Initial price	Exercise price	Strike price	Actual Option value	Experiment Option value	Error
1614.96	1960.23	1625	335	328.6	6.4

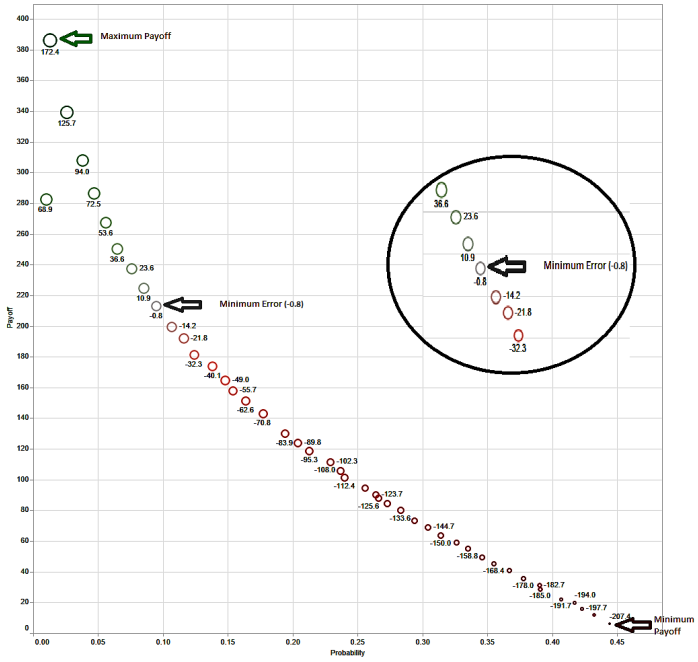


Figure 5.2: Pareto Front for at-the-money 6 months contract on July 1, 2013

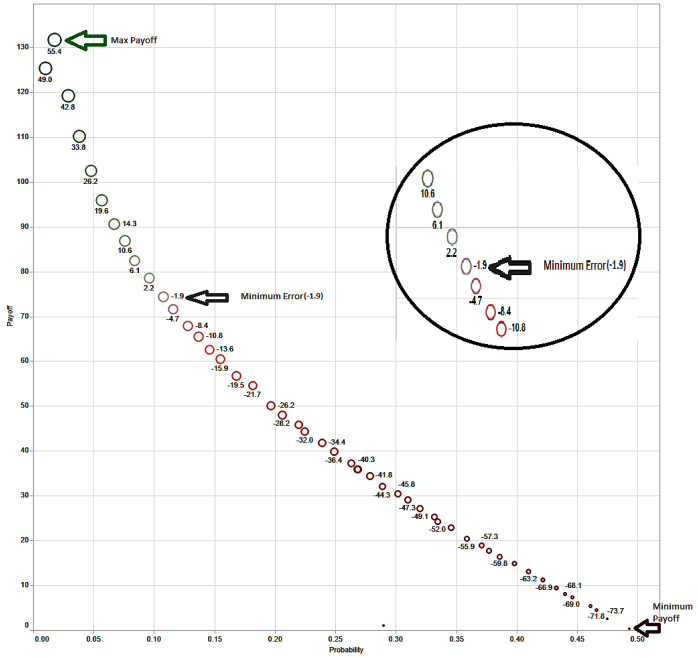


Figure 5.3: Pareto Front for at-the-money 1 month contract on July 1, 2013

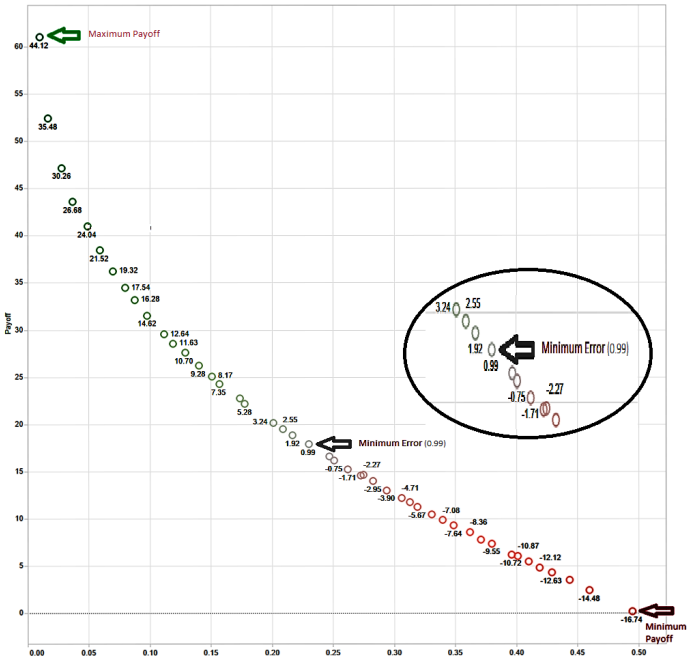


Figure 5.4: Pareto Front for at-the-money 1 week contract on July 1, 2013

5.2.2 Risk-Aware Application of the Model

In any multi-objective optimization problem, after finding the Pareto optimal solutions, we require a decision variable from the user to find an estimate on the solution for that particular problem. In my study, after validating my algorithm (chapter 4.3.2) and my model (chapter 4.1) it was necessary to analyze how investor will deploy this study to find the accurate worth of a contract. In my model, each solution on the Pareto front represents two variables: the *p*ay-off and *probability of getting that pay-off*. An investor is expected to select a solution from the Pareto front with maximum values for both of these variables, which however, is not possible due to the conflicting nature of both (as discussed in Chapter 4.1 with an example in Figure 4.2). Hence, the investor has to select a point, which trades-off between these two variables. My model and algorithm presents the Pareto front to the investor and

tells that one particular solution from the Pareto front is an accurate worth of the contract. Now, on the basis of *decision variable*, an investor may select any point on the Pareto front that estimates the worth of the contract with minimum error.

Value of an option varies with the risk capability of an investor. Therefore, in my algorithm, I map *investment risk* as the decision variable from the investor to compute the worth of a contract.In other word, I propose mapping the risk that investor may take to the probability of attaining a pay-off. In other words, on the basis of risk capability of an investor as a decision variable, the investor can select a point on the Pareto front to get an option value for a contract. For example, if an investor is not willing to take any risk, then he/she can select a solution with high *probability* (referred as "Minimum Payoff" in Figures 5.1 to 5.4) that may give less profit to the investor. Similarly, if an investor is willing to take high risk then he/she may select a solution with low probability and high payoff (referred as high payoff in Figures 5.1 to 5.4) that may give high profit to the investor but chances of getting that profit is very less. An Investor usually selects the risk factor on the basis of volatility and current market condition.

In the next chapter, I introduce a strategy to find the risk level variable for an investor using the historical data.

5.2.3 Strategy to evaluate risk level for European Option

To help investors in identifying possible risk level that would help them in finding the accurate worth of a contract, I further analyzed all my experiments and results in an attempt to find any relation among them. From my analysis I observed that

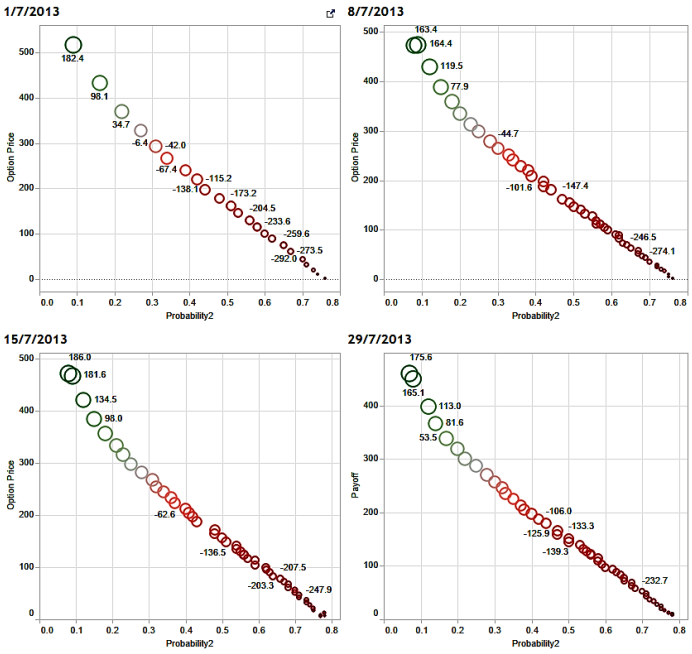


Figure 5.5: Comparison of at-the-money 1 year contract

for each particular type of contract, using historical data one can come up with an average risk or average value of probability where the error is minimum. Here, error is defined as the difference between the actual worth of the contract and the solution on the Pareto front that is closest to the actual worth of the contract obtained from my model and algorithm. In one of the analysis presented in Figure 5.5, it is clear that the amount of similarity between same type of contracts (at-the-money and one year expiration on four different days) is high.

From the Figure 5.5, I observe that on all four dates, error (shown using colours: Red for negative error and green for positive error, brightness resembles the magnitude of error) is minimum between probability range of 0.23 to 0.27 for one year at-the-money contract. Using the same approach on other types of contracts I analyzed

and noted average value of probability where error is minimum. Further, I used that noted average value to price the similar type of contract(s) on some other day (in this case July 22). Then, I compute the error between option value using the approach described above and the real worth of the contract (from real market data). In order to compare my results, I have also calculated the error between real worth of a contract and the option value calculated using classical BSM model. Results are shown in Tables 5.5 and 5.6. I have decided to compare my results with BSM model since it provides closed-form solution to European options. Also, it has found wide acceptance in financial markets and is used by investors to evaluate the worth of an option contract. From these tables, I observe that my approach described earlier in this chapter is really efficient.

From my experiments, I also observed one more crucial point that for similar type of contracts on different dates, option prices listed on the exchange have same probability of getting that option price. For example, for at-the-money 1 year contract on all selected four dates, option price asked on the exchange is 93, 87.7, 94, and 94 respectively. Option parameters on all four dates were different; that is, Initial asset price (S_0), Volatility (σ), interest (r) are different on all these dates. However, probability of attaining respective pay-off listed on exchange, for all different dates, turns out to be same (that is, 0.62). This type of behavior is also seen in all other contracts. This insight from the analysis depicts that the market tries to price the contract using same order of risk, which tends to be small. That is, they try to give the minimum price of the contract and my model is able to capture that accurately, which further proves the competency of my model.

Table 5.5: Comparison of error between Firefly algorithm and BSM model for At-the-money option pricing.

Maturity	**Error using FA algorithm**	**Error using BSM algorithm**
1 Week	1.75	14.5
6 Month	0.625	96.3
1 Year	1.14	197.8

Table 5.6: Comparison of error between Firefly algorithm and BSM model for Out-of-the-Money option pricing.

Maturity	**Error using FA algorithm**	**Error using BSM model**
1 Week	5.9	5.78
6 Month	6.45	44
1 Year	4.25	222.76

5.3 Experiment results for American Option

In this chapter, I discuss the results for the experiment on American style option contract. I used my experience of pricing European option contract as a building block to design another new and efficient Firefly algorithm for multi-objective optimization problem and used it to find worth of a more complex American option. In essence for these experiments, I have used non-dominant sorting Firefly algorithm (NSFA) to find Pareto optimal solution. Note that American style of option contract allows the holder to exercise the option anytime prior to the expiration date. Complexity arises due to the open boundary of the American option contract.

5.3.1 Pareto Front for Option Value

Here again, I will first discuss the Pareto front obtained from the implementation of my model. I have performed NSFA on over 600 contracts with different combina-

tions of "maturity" and "moneyness" on different possible dates and with different underlying stocks. In all the experiments performed, my algorithm was successful in finding the complete set of Pareto optimal solutions for American option where trade-off solutions have to be searched for the objectives: pay-off and probability. Some of the Pareto fronts computed are shown in Figures 5.6 to 5.8 from where it is clearly observable that NSFA was successful in finding the Pareto-optimal solution for multi-objective option pricing problem. For example, in Figure 5.6, which is the representation of the solutions found for 6 months at-the-money contract on January 5, 2015 for Apple stock, we can see that the NSFA successfully found the solution for maximum and minimum of both the objectives (represented by maximum payoff and minimum payoff arrow in the figure) and also successfully found the set of all other non-dominant solutions with respect to both objectives (represented by all the point between the maximum payoff and minimum pay off arrow). Similar behavior is also observed with other contracts shown in Figures 5.7 and 5.8, which validates the ability of NSFA to find solution for multi-objective optimization problem.

In order to analyze the quality (accuracy and efficiency) of the Pareto front found using my algorithm, similar to previous experiments on European option, I plotted the pay-off (y-axis), probability (x-axis) and error (shown using colours: Red for negative error and green for positive error, brightness resembles the magnitude of error) for all the solutions on a scatter plot. Some of the visualization are shown in Figures 5.6 to 5.8. Error is computed using the actual worth of the contracts. It was possible to get actual worth of all the contracts which was further used to determine the quality of the Pareto front in comparison to (real) historical data. One important

insight I observed on analyzing all the 600 computed Pareto fronts is that in 98% of the experiments my algorithm was successful in capturing the actual worth of the option contract with less than 1% error. For example (Table 5.7), Figure 5.8 presents solutions found using NSFA (at-the-money 6 months contract for Goldman Sachs). Here, S_T=218.39, S_o=193.06 and K=193. The option value of the contract from real data is noted as 25.4. In Figure 5.8, one of the solutions represents pay-off as 25.84 which is just 0.44 more than the actual value. This solution point is identified with an arrow in the Figure 5.8. Similarly in other contracts also solutions found using NSFA algorithm captured the solution with almost 0-1 % error. This observation shows the ability that my model of option pricing as multi-objective optimization is capable of capturing the accurate worth of the contract in real market conditions.

Table 5.7: Experimental Results for American Option

Initial price	Exercise price	Strike price	Actual Option value	Experiment Option value	Error
193.06	218.39	193	25.4	28.84	0.44

Finally, I conclude using the observation from this chapter that (i) non-dominant sorting Firefly algorithm efficiently computes the true Pareto-optimal solution for multi-objective option pricing problem and might be further extended to optimize any other multi-objective optimization problems (ii) my model of mapping probability and pay-off as a multi-objective optimization problem is successful in capturing the true behavior of the American style of option contracts and accurately.

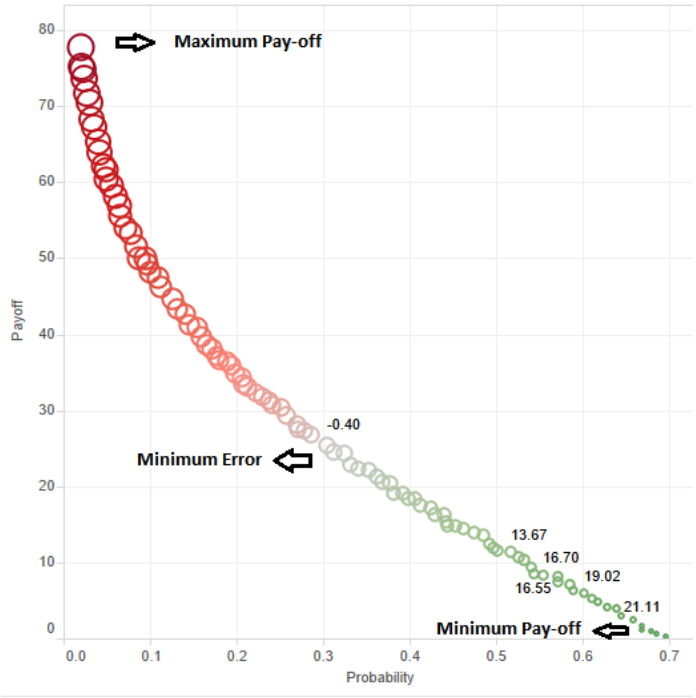


Figure 5.6: Pareto Front for at-the-money 6 months contract on January 5, 2015 for Apple Stock

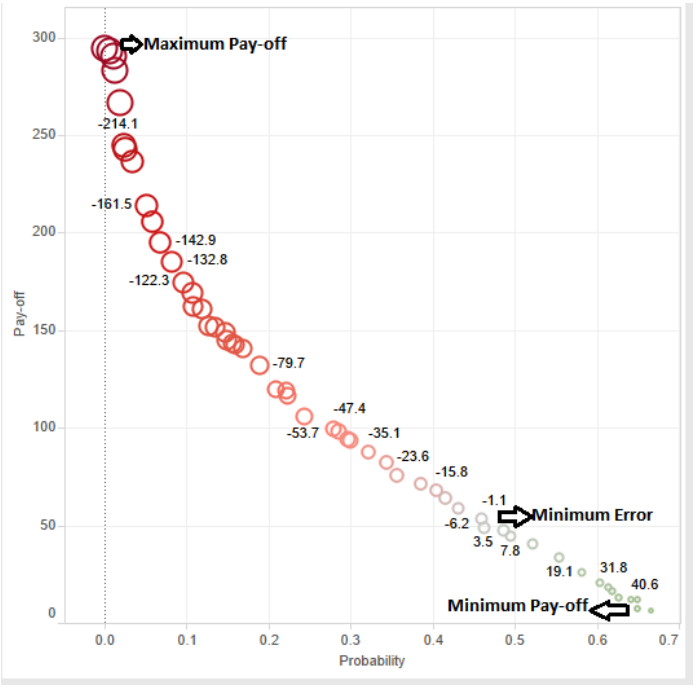


Figure 5.7: Pareto Front for at-the-money 6 months contract on January 5, 2015 for Google Stock

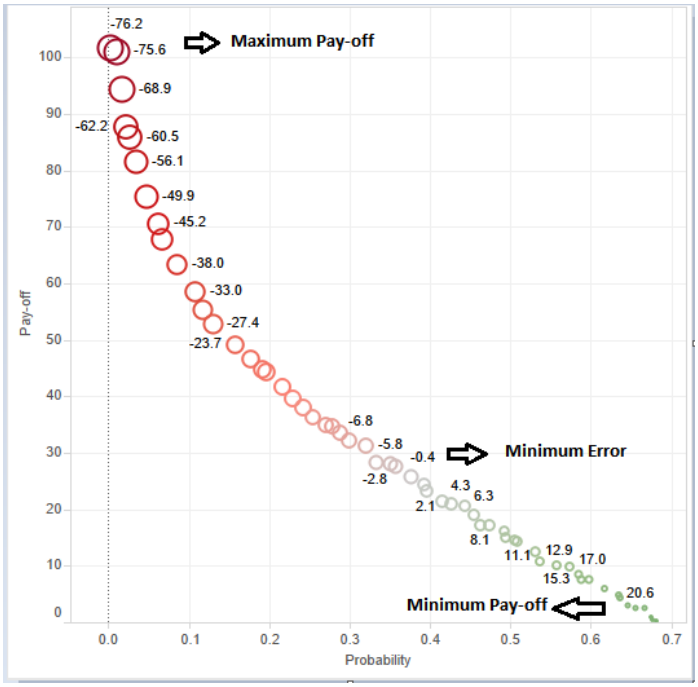


Figure 5.8: Pareto Front for at-the-money 6 months contract on January 5, 2015 for Goldman Sachs Stock

5.3.2 Risk-Aware Application of the Model

After validating my algorithm and model, I analyze how investor will deploy this study to find the accurate worth of a contract for American option the same way I studied European Contract. In my model each solution on the Pareto front, represents two variables: the *p*ay-off and *probability of getting that pay-off* and an investor is always expected to select a solution from the Pareto front that gives maximum values for both these objectives. However, it is difficult to find such a solution as both objectives have tendency of behaving in opposite manner. Therefore, the investor has to select a point, which finds a trade-off between both these objectives. My model and algorithm mainly presents a Pareto front of solutions to the investor and advise the investor to select one particular solution from the Pareto front on the basis of

their *decision variable* that estimates the accurate worth of an option with minimum error.

Financial markets are very volatile due to which it is not possible to have a single value for any option contract. Therefore, I believe that every option contract has its own value depending on the level of risk an investor may take. In other words, value of an option contract varies according to the risk taking capacity of an investor. Therefore, in my algorithm,I map *investment risk* as the *decision variable* to compute the worth of an option. In other words, I propose mapping the risk taking capacity of an investor with the probability of attaining a pay-off. On the basis of risk capacity of an investor as a decision variable, the investor may select a point on the Pareto front to get an option value for the contract. As discussed earlier, if an investor is not willing to take any risk, then he/she can select a solution with high *probability* (referred as "Minimum Payoff" in Figures 5.6 to 5.8) that may give less profit to the investor. Similarly, if an investor is willing to take high risk then he/she may select a solution with low probability and high payoff (referred to as high payoff in Figures 5.6 to 5.8) that may give high profit to the investor but chances of getting that profit is very less. Investor usually select the risk factor on the basis of volatility and current market condition.

In the next chapter, I propose a strategy for the investor to select risk variable on the basis of historical data that may help investor(s) to approximate the accurate worth of the contract.

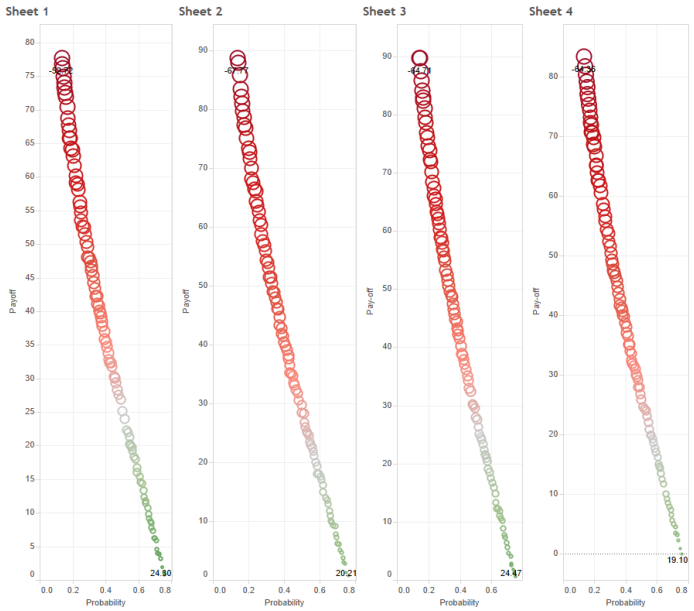


Figure 5.9: Pareto Front for at-the-money 1 Year contract on 4 Mondays in January, 2015 for Apple Stock

5.3.3 Strategy to evaluate current risk level for American option

I have analyzed all my experiments done using NSFA on the historical data in order to identify the relation between the risk level and worth of an option contract. In Figures 5.9 - 5.11, the degree of similarity in Pareto fronts for similar type of contracts is clearly visible. Therefore, in this chapter I propose a strategy of using the historical data and analyze Pareto front for similar type of contracts from previous dates in order to predict the current level of risk.

In order the analyze the accuracy of this strategy, I have done back-testing using 6 months historical data from January 2015 to June 2015. I have considered data of 4 Mondays of each month. In my strategy I analyze each and every historical contract for the considered period and note the value of probability or risk for each

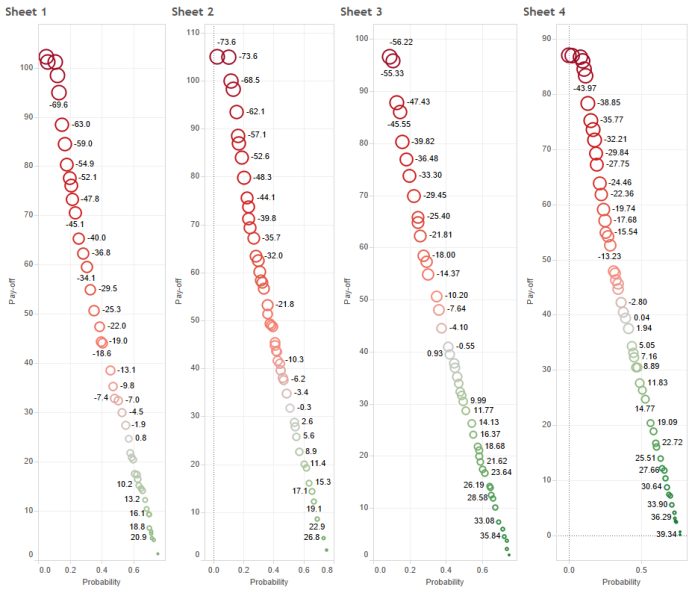


Figure 5.10: Pareto Front for at-the-money 1 Year contract on 4 Mondays in January, 2015 for Goldman Sachs Stock

contract where the error is found to be minimum. Then, I compute the average value of probability or risk for each type of contract where it has maximum chances of getting minimum error. I consider these average computed value as the average risk level for that particular type of contract and use it to compute the option price.

Table 5.8: Comparison of error between NSFA, Monte-Carlo simulation and Binomial Lattice for At-the-money option pricing

Maturity	**Error using FA algorithm**	**Error using Monte-Carlo Simulation**	**Error using Binomial Lattice**
1 Week	14	50	46
1 Month	35	51	51
3 Month	14.3	56	54.5
6 Month	29.1	37.2	37.8
1 Year	12.4	31.9	30.8

For example, if I want to evaluate a contract on 6th July, 2015 (Monday), I consider the probability of Mondays in previous 6 months. I observe the Pareto front of all

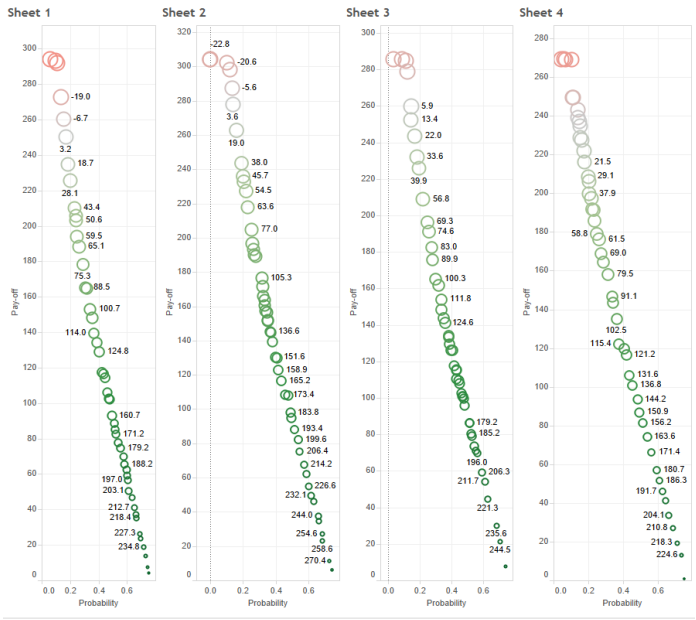


Figure 5.11: Pareto Front for at-the-money 1 Year contract on 4 Mondays in January, 2015 for Google Stock

Table 5.9: Comparison of error between NSFA, Monte-Carlo simulation and Binomial Lattice for Out-of-money option pricing

Maturity	**Error using FA algorithm**	**Error using Monte-Carlo Simulation**	**Error using Binomial Lattice**
1 Week	20	55	48
1 Month	28	55	52
3 Month	20.7	55.9	53.4
6 Month	25.2	35.8	37.2
1 Year	14.8	33.1	34.6

Table 5.10: Comparison of error between NSFA, Monte-Carlo simulation and Binomial Lattice for In-the-money option pricing

Maturity	**Error using FA algorithm**	**Error using Monte-Carlo Simulation**	**Error using Binomial Lattice**
1 Week	12	52	45
1 Month	24.5	45.8	47.1
3 Month	10.6	47.4	44.2
6 Month	22.1	34.3	35.6
1 Year	10.7	30.4	30.9

contracts on all these 24 days and noted those solutions on their front, which gives the minimum error. I compute the average probability for noted solution and consider it as the current risk level. Further, using the computed risk level I select a solution on Pareto front for the current contract and use its pay-off value as the solution. Also, I did some experiments to evaluate option price for American style of contracts for 5 stocks and have noted the error between real worth and the worth predicted using this strategy. I also compared the results with other two popular American option pricing techniques: Monte-Carlo simulation [46] and binomial lattice model [4]. Results are shown in Tables 5.8 - 5.10. It is clearly observable that the results using this strategy with Firefly algorithm is much better than the other conventional techniques. Similarly, one can explore different risk selecting strategies to predict the accurate worth of a contract.

Chapter 6

Conclusion and Future Work

The main goal of my thesis study is to solve option pricing problem using the nature-inspired optimization techniques. Two major contributions from my research are: (i) proposal of a novel model where I mapped option pricing problem as a multi-objective optimization problem, (ii) design and development of two new variants of Firefly algorithm that were used to compute the solution for the option pricing problem.

I have proposed the pay-off from the option contract and the probability of attaining that pay-off as two major optimization objectives and used them with new firefly algorithms to find Pareto optimal solution(s). Since option pricing is modelled as a multi-objective optimization problem, I present not one but a set of solutions to the investor for the option pricing problem. To the best of my knowledge this is the first attempt of studying financial option pricing problem as a multi-objective optimization using firefly algorithm.

With my experiments, I was able to show the competency of my model and accu-

racy and efficiency of my algorithm to price both the European and American style options. In my thesis, I designed experiments in order to cover a spectrum of option contracts with multiple strike prices, expiration dates and stocks. In all these experiments, my algorithm was successful in finding the Pareto front for option prices and that the solution computed from all experiments captured the real market values very accurately. Analyzing all the Pareto fronts, it was observed that in 99% of the experiments, my algorithm was successful in capturing the true option price as a solution on the Pareto front with less than 2% error. That is, in almost all the experiments, my algorithm was able to find a solution that is equal to option price of a contract as available from real (historical) market data. Therefore, I can conclude that using my model and algorithm, an investor can accurately evaluate the option contract for a given level of risk to enable him/her to decide before entering the contract.

Moreover, I proposed a strategy using historical data to help investors to approximate decision variable (risk level) in order to evaluate the accurate worth of a contract. This strategy worked best with promising results for both European and American options. Initially, I did my study on a small set of data for European contracts and later extended it to price American style of contracts with large set of data. Experiments in both cases showed significant results for this strategy. Therefore, I conclude that the strategy of using historical data to find the risk level is very efficient in terms of accuracy and can be used by investors in real financial market. This study can be extended and larger set of data (2-5 years) for each stock can be used to get a more refined historical risk level for any particular type of contract. Also, a tool can be developed, which keeps refining the risk level value continuously for multiple

type of stocks on different indices. However, for this tool, data becomes too big if we consider all the listed stocks or contracts on the wall street and therefore, various analytics from the field of big data may be needed to tackle such an issues. I leave this as a important and immediate future work.

Further, various other risk selection strategies can be explored and their effects can be studied for different trading strategies. My technique presents a set of solutions to the user and lets user to choose any one solution but the basis on which user selects the solution is something that needs to be studied further. Basically how to use the Pareto front in different ways and determining the decision variable (risk level) can be seen as a new future study. Also, Pareto front and its relation with different type of option trading strategies can be analyzed in future. My methodology can be extended to price exotic options also such as Asian or Russian option and its performance or results can be analyzed.

Bibliography

[1] "IBM option chain diagram, howpublished = http://www.investopedia.com/university/options-pricing/pricing-basics.asp, note = Accessed: 2016-07-27."

[2] K. Deb, *Multi-objective optimization using evolutionary algorithms*. John Wiley & Sons, 2001.

[3] F. Black and M. Scholes, "The pricing of options and corporate liabilities," *The journal of political economy*, pp. 637–654, 1973.

[4] R. C. Merton, "Theory of rational option pricing," *Bell journal of economics and Management Science*, pp. 141–183, 1973.

[5] J. C. Cox, S. A. Ross, and M. Rubinstein, "Option pricing: A simplified approach," *Journal of financial Economics*, vol. 7, no. 3, pp. 229–263, 1979.

[6] P. P. Boyle, "Options: A monte carlo approach," *Journal of financial economics*, vol. 4, no. 3, pp. 323–338, 1977.

[7] M. J. Brennan and E. S. Schwartz, "Finite difference methods and jump processes arising in the pricing of contingent claims: A synthesis," *Journal of Financial and Quantitative Analysis*, vol. 13, no. 03, pp. 461–474, 1978.

[8] J. C. Hull, *Options, Futures and Other Derivatives, 9th edition.* Prentice Hall, 2014.

[9] L. Jiang and C. Li, *Mathematical modeling and methods of option pricing.* World Scientific, 2005.

[10] A. Cerny, *Mathematical Techniques in Finance - Tools for Incomplete Markets.* Princeton Univ. Pres,, NJ, USA, 2004.

[11] X. S. Yang, *Nature-inspired optimization algorithms.* Elsevier, 2014.

[12] J. Kennedy, "Particle swarm optimization," in *Encyclopedia of Machine Learning.* Springer, 2010, pp. 760–766.

[13] M. Dorigo and M. Birattari, "Ant colony optimization," in *Encyclopedia of machine learning.* Springer, 2010, pp. 36–39.

[14] S. Kumar, G. Chadha, R. K. Thulasiram, and P. Thulasiraman, "Ant colony optimization to price exotic options," in *Proceedings of the IEEE Congress on Evolutionary Computation Conference*, 2009, pp. 2366–2373.

[15] R. K. Thulasiram, P. Thulasiraman, H. Prasain, and G. Jha, "Nature- inspired soft computing for financial option pricing using high performance analytics," *Concurrency and Computation: Practice and Experience - appeared on-line First*, 2014.

[16] N. K. Chidambaran, C.-W. J. Lee, and J. R. Trigueros, "An adaptive evolutionary approach to option pricing via genetic programming," 1998.

[17] G. He, N. Huang, H. Ma, J. Lu, and M. Wu, *An Improved Particle Swarm Optimization Algorithm for Option Pricing*, J. Xu, A. V. Cruz-Machado, B. Lev, and S. Nickel, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014.

[18] X. S. Yang, "Firefly algorithms for multimodal optimization," in *Stochastic algorithms: foundations and applications*. Springer, 2009, pp. 169–178.

[19] S. K. Pal, C. Rai, and A. P. Singh, "Comparative study of firefly algorithm and particle swarm optimization for noisy non-linear optimization problems," *International Journal of intelligent systems and applications*, vol. 4, no. 10, p. 50, 2012.

[20] X. S. Yang, "Multiobjective firefly algorithm for continuous optimization," *Engineering with Computers*, vol. 29, no. 2, pp. 175–184, 2013.

[21] X. S. Yang, *Nature-inspired metaheuristic algorithms*. Luniver press, 2010.

[22] W. Tang and Q. Wu, "Biologically inspired optimization: a review," *Transactions of the Institute of Measurement and Control*, 2009.

[23] X. S. Yang, S. S. S. Hosseini, and A. H. Gandomi, "Firefly algorithm for solving non-convex economic dispatch problems with valve loading effect," *Applied Soft Computing*, vol. 12, no. 3, pp. 1180–1186, 2012.

[24] K. A. Costa, L. A. Pereira, R. Y. Nakamura, C. R. Pereira, J. P. Papa, and A. X. Falcão, "A nature-inspired approach to speed up optimum-path forest clustering and its application to intrusion detection in computer networks," *Information Sciences*, vol. 294, pp. 95–108, 2015.

[25] J. Senthilnath, S. Omkar, and V. Mani, "Clustering using firefly algorithm: performance study," *Swarm and Evolutionary Computation*, vol. 1, no. 3, pp. 164–171, 2011.

[26] S. J. Nanda and G. Panda, "A survey on nature inspired metaheuristic algorithms for partitional clustering," *Swarm and Evolutionary computation*, vol. 16, pp. 1–18, 2014.

[27] T. M. Sands, D. Tayal, M. E. Morris, and S. T. Monteiro, "Robust stock value prediction using support vector machines with particle swarm optimization," in *Evolutionary Computation (CEC), IEEE Congress on*, 2015, pp. 3327–3331.

[28] R. Geske and R. Roll, "On valuing american call options with the black-scholes european formula," *The Journal of Finance*, vol. 39, no. 2, pp. 443–455, 1984.

[29] P. Wilmott, *Paul Wilmott introduces quantitative finance*. John Wiley & Sons, 2007.

[30] C. Keber and M. G. Schuster, "Generalized ant programming in option pricing: Determining implied volatilities based on american put options," in *Computational Intelligence for Financial Engineering, 2003. Proceedings. IEEE International Conference on*, 2003, pp. 123–130.

[31] G. P. Zhang, "Time series forecasting using a hybrid arima and neural network model," *Neurocomputing*, vol. 50, pp. 159–175, 2003.

[32] K. Doerner, W. J. Gutjahr, R. F. Hartl, C. Strauss, and C. Stummer, "Pareto

ant colony optimization: A metaheuristic approach to multiobjective portfolio selection," *Annals of operations research*, vol. 131, no. 1-4, pp. 79–99, 2004.

[33] J. M. Hutchinson, A. W. Lo, and T. Poggio, "A nonparametric approach to pricing and hedging derivative securities via learning networks," *The Journal of Finance*, vol. 49, no. 3, pp. 851–889, 1994.

[34] Z. Yin, A. Brabazon, and C. O'Sullivan, "Adaptive genetic programming for option pricing," in *Proceedings of the 9th Annual Conference Companion on Genetic and Evolutionary Computation*. New York, NY, USA: ACM, 2007, pp. 2588–2594.

[35] C. Keber and M. G. Schuster, "Option valuation with generalized ant programming," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2002, pp. 74–81.

[36] G. P. Singh and A. Singh, "Comparative study of krill herd, firefly and cuckoo search algorithms for unimodal and multimodal optimization," *International Journal of Intelligent Systems and Applications*, vol. 6, no. 3, p. 35, 2014.

[37] S. M. Lewis and C. K. Cratsley, "Flash signal evolution, mate choice, and predation in fireflies," *Annu. Rev. Entomol.*, vol. 53, pp. 293–321, 2008.

[38] J. R. De Wet, K. Wood, M. DeLuca, D. R. Helinski, and S. Subramani, "Firefly luciferase gene: structure and expression in mammalian cells." *Molecular and cellular biology*, vol. 7, no. 2, pp. 725–737, 1987.

[39] A. Brasier, J. Tate, and J. Habener, "Optimized use of the firefly luciferase assay

as a reporter gene in mammalian cell lines." *BioTechniques*, vol. 7, no. 10, pp. 1116–1122, 1988.

[40] B. L. Strehler and J. R. Totter, "Firefly luminescence in the study of energy transfer mechanisms. i. substrate and enzyme determination," *Archives of biochemistry and biophysics*, vol. 40, no. 1, pp. 28–41, 1952.

[41] H. H. Seliger and W. D. McElroy, "Spectral emission and quantum yield of firefly bioluminescence," *Archives of biochemistry and biophysics*, vol. 88, no. 1, pp. 136–141, 1960.

[42] M. Deluca and W. McElroy, "Purification and properties of firefly luciferase." *Methods in enzymology*, no. 57C, pp. 3–15, 1978.

[43] I. Y. Kim and O. De Weck, "Adaptive weighted-sum method for bi-objective optimization: Pareto front generation," *Structural and multidisciplinary optimization*, vol. 29, no. 2, pp. 149–158, 2005.

[44] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *Evolutionary Computation, IEEE Transactions on*, vol. 6, no. 2, pp. 182–197, 2002.

[45] L. P. Bloomberg, "Stock and option data for S&P 500 index, Apple, Google, Goldman Sachs, Amazon, IBM," 2015.

[46] F. A. Longstaff and E. S. Schwartz, "Valuing american options by simulation: a simple least-squares approach," *Review of Financial studies*, vol. 14, no. 1, pp. 113–147, 2001.