### **Particle swarm optimization (PSO):**

Particle Swarm Optimization [Kennedy and Eberhart, 1995] is a population based heuristic optimization algorithm inspired by social behavior of birds flocking or fish schooling.

Consider the scenario of birds looking for food. The birds first search their own neighborhood for the food source. At the end of each time step or iteration the birds decide on a location that might lead to a food source.

The birds then compare their solution with other birds solutions and move closer to the birds that are closest to the location of the food source.

Each bird (a particle) is a potential solution in the search space. This concept is formulated as PSO algorithm.

Each particle is treated as a point in a D-dimensional space. Initially, N particles are uniformly distributed in the solution space. The particles in PSO fly through the search space with a certain velocity, and change their position dynamically in the hope of reaching the food source, the destination. Therefore, position and velocity are two important parameters in the PSO algorithm.

Each particle keeps track of the best position it has encountered during its travel, and the best position traveled by the swarm of particles. The best position traveled by a particle is called the local best position, and the best position traveled by the swarm is called the global best position. At the end of each iteration, the particles calculate their next velocity, and update their positions based on the calculated velocity.

**PSO Algorithm**

Particle ***i*** is represented as ***Xi*** **=(*xi1,xi2,…..,XiD***) , which represents a potential solution to a problem in D-dimensional space.

Each particle keeps a memory of its previous best position , ***Pbest*** , and a velocity along each dimension, represented as ***Vi=(vi1,vi2,….,viD).***

For each iteration, the position of the particle with the best fitness value in the search space, designated as ***g***, and the ***P*** vector of the current particle are combined to adjust the velocity along each dimension, and that velocity is then used to compute a new position for the particle .

The method could be divided into GBEST and LBEST versions, whose main difference is their definition of the best. In the GBEST version, the particle swarm optimizer keeps track of the overall best value, and its location, obtaining thus far by any particle in the population, which is called ***gbest* (*Pgd*)**.

For the LBEST version, in addition to gbest, each particle keeps track of the best solution, called lbest(Pgd) , and it is attained within a local topological neighborhood of particles. However, the particle velocities in each dimension are held to a maximal velocity, ***Vmax***, and the velocity in that dimension is limited to *Vmax*.

Later, inertial weight was developed to better balance exploration and exploitation in order to eliminate the need for  *Vmax*.

The updating rule is given by

* *Vid(new)= W.Vid(old) + c1.rand1.(Pid + Xid) + c2.rand2.(Pgd – Xid).*
* *Xid(new)= Xid(old) + Vid(new).*

Where ***W*** is the inertia weight, ***c1*** and ***c2*** determine the relative influence of the social and cognition components (learning factors), while ***rand1*** and ***rand2*** denote two random numbers uniformly distributed in the interval [0, 1]. Recent work done by Clerc indicates that the use of a constriction factor may be necessary to ensure convergence of the PSO algorithm . PSO algorithm has been applied for many optimization problems, such as scheduling and traveling sales problems.

## **Particle swarm optimization & Linear Programing Methodology :**

The PSO concept originated as a simulation of a simplified social system. Each particle keeps track of its coordinates in the problem space which are associated with the best solution (fitness) it has achieved so far. Due to this merit, this study employs PSO algorithm for solving the LPPs(Linear Programing Problems).

The LPP is developed for decentralized planning systems in which the upper level is termed as the leader and the lower level pertains to the objective of the follower. In the LPP, each decision maker tries to optimize its own objective function without considering the objective of the other party, but the decision of each party affects the objective value of the other party as well as the decision space. This section will present the way to apply PSO algorithm for LPPs. The PSO algorithm for solving LPPs is presented as follows:

**Step 1:**

* Set up parameters including population size (the number of particles), Maximal velocity ***Vmax* ,** inertial weight ***W***, and two learning factors, (**c1** and **c2** ). Two random variables, ***rand1*** and ***rand2*** , are in the interval [0, 1].

**Step 2:**

* Set up the searching range for ***x***, which will influence the searching speed. Initialize each particle randomly with initial position, ***Xid***, within the pre-specified range and velocity, ***Vid*** , in the range of maximal speed, ***Vmax***. This study adopts the float coding method to generate the random numbers for the upper-level variables. Thus, every particle represents the real dimensional position. Then, program for variable ys in the lower level. Each particle’s position is represented as:

***Xid* = (*xi1*,…..,*xin*,*yi1*,……,*yim*).**

**Step 3 :**

* Calculate every particle’s fitness value using
* F = *CXid*.

**Step 4:**

* Update the local best position, ***Pid*** , and global best position, ***Pgd .***

**Step 5:**

* According to the updated ***Pid*** and ***Pgd***, use this Eqs. to update the velocity and position for every particle.
* *Vid(new)= W.Vid(old) + c1.rand1.(Pid + Xid) + c2.rand2.(Pgd – Xid).*
* *Xid(new)= Xid(old) + Vid(new).*
* Every particle’s velocity is constrained by the pre-determined maximal velocity ,***Vmax*** , and every particle’s position,***Xid*** , should be within the specified range: l <= xi <=u .

**Step 6:**

* Stop if the specified number of generations is satisfied; otherwise, go back to Step 3.

**Particle swarm optimization for option Pricing :**

One of the important research problems in computational finance is identifying the best time to exercise a given option while maximizing the profit of the option.

The focus of this thesis is to design an algorithm that is inspired from natural world such as school of fish, flock of birds or in general swarm.

This algorithm is expected to compute optimal option values in the given solution space at a reasonable amount of time which is better than we consider a popular nature inspired algorithm, Particle Swarm Optimization (PSO). The first focus of our research is to design and develop a sequential algorithm for the option pricing problem using basic principles of PSO.

**Mapping PSO for Option Pricing :**

In mapping the PSO to the option pricing problem, a particle in our algorithm

is defined by five parameters: id, position vector (⃗***X***), velocity vector (⃗***V*** ), local best

position (⃗***lbest***) and global best position (⃗***g best***).⃗ ***X*** corresponds to the price and⃗ ***V***

corresponds to the change in price. ***Xmax*** and ***Xmin*** correspond to the boundaries of

the solution space. Similarly, previous best position ***P i(t)*** in PSO is a particle’s best ***li*** best) in terms of option value (profit) and option exercise time; and ***Pg(t)*** in the PSO is the overall best (***g best***) among particles in the option pricing solution space.

initial particle position and velocity are already defined in Equations.

A particle is used to compute option value and capture the time when this optimum option value occurs. Therefore, each position vector (⃗***X***) has two dimensions.

The first dimension ***(X[0])*** is for stock price, and the second dimension (***X[1]***) is the

exercise time. Similarly, there are two dimensions for⃗ V ,⃗ ***lbest*** and⃗***g best***, ***Xmax*** and ***Xmin***.

Initially, the particles are distributed randomly in the solution space using initial postion Equation.

The initial velocity for the ***ith*** particle is calculated using his equation . The particles fly through

the search space with a certain velocity, and change their position dynamically until the termination condition is satisfied. In the current implementation the termination condition is set to a predetermined number of iterations.

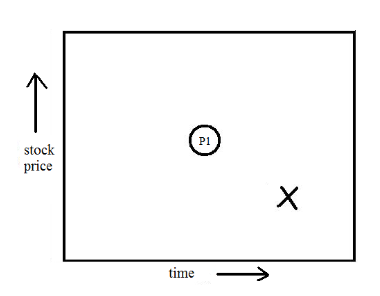
The

position and velocity updates by a particle i are calculated as follows:

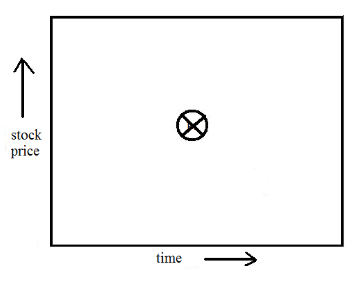
***Xi t+1[k] = Xi t [k] +Vi t+1[k]; k = 0, 1.***

***Vi t+1[k] = ω×Vi t [k]+c1×r1×(li best[k]−Xi t [k])+c2×r2×(gbest[k]−Xi t [k]); k = 0, 1.***

There are four possible scenarios to consider for a particle to update its li best[k] and gbest[k] after it moved to a new position. Figures 1- 5 demonstrate these scenarios for a call option. In these figures, “X” represents local best position, and a circle represents a new position of a particle.

**Scenario I :**

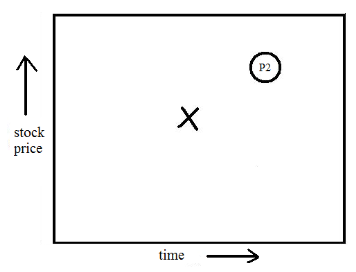
**Figure 1: A particle movement for a call option Scenario I**

A particle flies to a new position P1 (node), and checks if that new position represents better profit (higher local pay-off; that is, higher stock price for a call option) and represents an earlier time than the local best (price and time) obtained so far for that particle (see Figure 1).

**Figure 2: Local best update**

If so, the particle updates its local best position as shown in Figure 2. Recall that we are demonstrating the particles movement for a call option. Similar arguments can be extended to

a put option.

**Scenario II :**

**Figure 3: A particle movement for a call option Scenario II**

A particle flies to a new position P2, and checks if that new position represents

better profit (higher stock price for a call option) and represents a later (future)

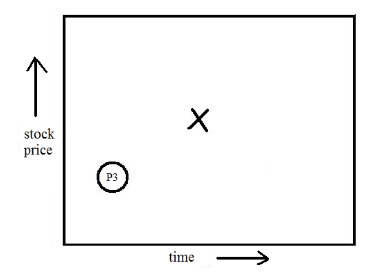
time than the local best position (price and time) obtained so far (see Figure 3)

for that particle. If so, the particle does the following:

* the option price at the new position (future time) is discounted using

***Xi[0] − K) × exp(−r × (Xi[1] – li best[1] )),*** (where K is the strike price and is the risk free interest rate), to find its value at the local best position. Recall that ***X[1]*** and ***lbest[1]*** correspond to time.

* if this discounted value is greater than the current local best option price, then the particle flies to the new position (that is, waiting for the future time is beneficial), and updates its local best position with the newly computed option price and exercise time. Otherwise, the local best remains as it is.

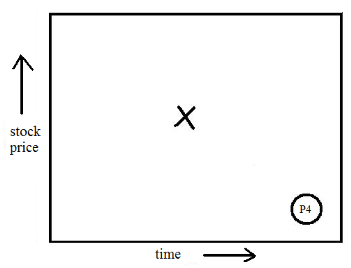
**Scenario III :**

**Figure 4: A particle movement for a call option Scenario III**

A particle flies to a new position P3, and in that position if profit is less ( i.e.lower local pay-off; that is lower stock price for a call option) and time in this new position represents an earlier time, then the particle does the following. Though the local pay-off is less at this new position, the time represented by this position is an earlier time than the local best time (see Figure 4). There

is a chance that the newly found node may move to profitable position (with higher local pay-off) at later time (iterations) than the local best position. In this case, the particle needs to search the surrounding neighborhood. This is to prevent stagnation at one location. The particle does the following :

* the future option value is discounted using ***(Xi[0]−K)×exp(−r×(libest[1]−Xi[1]))*** to calculate the current option value.
* if the discounted future option value is greater than the current local best option price, then particle updates its local best position. Otherwise, the local best remains as it .

**Scenario IV :**

**Figure 5: A particle movement for a call option Scenario IV**

The last case where there is less profit (i.e. smaller local pay-off) at a later date (see Figure 5) in time, and is not advantageous because (a) particles may have already found a better position which is profitable and corresponds to an earlier time (b) particles are close to maturity date. In this case, a particle performs no update on its local best position.

After each iteration, the best of all the local best (⃗***li best***) is calculated in the same was as the above four cases in comparison with⃗***g best***. If ***(⃗ li best)*** is better than current⃗***g best*** (global best), then⃗***g best*** will be updated. The update in local and global best positions continues until termination condition is met.

**PSO-based algorithm for Option Pricing :**

The pseudo code for a sequential PSO-based option pricing algorithm is given below:

Initialize PSO parameters: ***c1, c2, ω,⃗ Vi max,⃗ Vi min,⃗ Xi max,⃗ Xi min***

Initialize option pricing variables and parameters: S, K, r, σ, T

for i = 1 to N do

Initialize⃗ ***Xi*** by using Equation ***Xi0 = Xmin + r1 × (Xmax - Xmin)***

Initialize⃗ ***Vi*** by using Equation ***Vi0 = Xi***

Li best =⃗ ***Xi***

end for⃗

***g best = max⃗ l1best,⃗ l2best, . . . ,⃗ lNbest***

for ***Itr*** = 1 to M AXIT ER do

for i = 1 to N do

Calculate the new position (⃗***Xi I(tr))*** and velocity (⃗***Vi (Itr***)) by using Equations

***Xit+1[k] = Xit [k] + V it+1[k]; k = 0, 1.***

***and Vi t+1[k] = ω×Vi t [k]+c1×r1×(li best[k]Xi t [k])+c2×r2×(gbest[k]−Xi t [k]); k = 0, 1.*** respectively.

Check boundary conditions: If a particle flies out of boundary (solution space i.e.⃗ ***Xi max*** or⃗ ***Xi min***), then re-calculate new velocity, and check boundary conditions one more time. If a particle flies out of solution space with re-calculated

velocity, then penalize the particle with some random velocity.

Update the local best position by considering the four scenarios discussed in above .

end for⃗

***g best = max⃗ l1best,⃗ l2best, . . . ,⃗ lN best***

Update⃗***g best*** information in all particles

end for

Prin t⃗***g best***

**1-** **Unit price Proposal problem :** Bidwell Construction is faced with the task of preparing a bid for excavation work on a unit price proposal. This endeavor involves determining competitive unit prices for various excavation tasks while ensuring profitability. The company must navigate through cost calculations, markup considerations, and market dynamics to formulate a bid that not only covers expenses but also generates a desirable profit margin. This scenario presents a classic optimization problem in the realm of construction project bidding, where linear programming techniques can be employed to find the most advantageous solution.  
  
**2-Problem Overview:**The problem at hand involves several key components:

* Identifying the unit costs for different excavation tasks such as clearing, earth excavation, rock excavation, and cleanup.
* Applying a 5% markup to these unit costs to establish unit prices for the bid.
* Ensuring that the total bid amount, incorporating all unit prices, aligns with the desired budget and profit margin.
* Balancing the unit prices for rock excavation and earth excavation to maintain competitiveness in the bidding process.
* Expressing all financial transactions in terms of present worth to facilitate effective decision-making.

**3-Objectives:**

To solve this problem effectively, Bidwell Construction must:

1. Determine optimal unit prices for each excavation task that minimize the total cost of the bid while meeting profitability requirements.
2. Ensure that the bid proposal remains competitive in the market by appropriately pricing each item of excavation work.
3. Utilize linear programming techniques to optimize the bidding strategy and navigate through various constraints.
4. Interpret the results to make informed decisions regarding the final bid proposal, considering both financial considerations and market dynamics.

By addressing these objectives systematically, Bidwell Construction can develop a compelling bid proposal that not only meets the client's needs but also enhances the company's profitability and competitiveness in the construction industry.

**4-Problem Analysis:**

**Objective:** The objective is to determine the unit prices (X1, X2, X3, X4) for different items in Bidwell Construction's unit price proposal in order to maximize the total bid value.

**Constraints:**

1. **Total Bid Value Constraint:** The total bid value should include a 5% markup, which should sum up to $262,500.
   * 20,000x1+75,000x2+25,000x3+20,000x4=262,500
2. **Rock Excavation Unit Price Constraint:** The unit price for rock excavation should be higher than the unit price for earth excavation to ensure a balanced bid.
   * x3−x2≥0
3. **Unit Price Limits Constraint:** Bidwell has specified certain limits for the unit prices these constraints ensure that the unit prices fall within the specified rang.

* $1.00 ≤ x1 ≤ $4.00
* $0.50≤ x2 ≤ 3.00
* $1.50≤ x3 ≤ 6.00
* $1.00 ≤ x4 ≤ $4.00

1. **Non-Negativity Constraint:** The unit prices for all items should be non-negative.

* x1,x2,x3,x4≥0

**Decision Variables:**

* x1: Unit price for Clearing
* x2​: Unit price for Earth Excavation
* x3​: Unit price for Rock Excavation
* x4​: Unit price for Cleanup

**Objective Function:** The objective is to maximize the total bid value, which is the sum of the product of unit prices and quantities for each item.

* Z=19,606x1+67,897x2+22,632x3+17,749x4

6-References:

* 1. <https://rb.gy/aaz3oq> .
  2. <https://www.tandfonline.com/doi/pdf/10.1080/08839514.2013.805596> .
  3. <https://mspace.lib.umanitoba.ca/server/api/core/bitstreams/a081528a-e903-4a78-85cf-9767758a2a8e/content> .