ISSN No:-2456-2165

Cost Optimization Solutions for E-Commerce Vendors

Sashwat Desai

ISSN No:-2456-2165

ABSTRACT

In today's times, retail is soaring with the advent of e-commerce and a majority of sellers are coming online to sell their products. This has resulted in an increased competition. Newer vendors are finding it difficult to compete with the bigger companies who have already established a foothold in the market. Hence winning over a customer requires top level customer satisfaction, but at the same time expenditure should be kept such that there is a considerable profit margin.

The main reason behind the selection of this topic is to provide not only the customers but also the vendors with the most cost-effective solution. In many cases, it is observed that the prices listed for products may not be the best ones. To tackle this issue, this topic has been chosen.

This topic is relevant to the current world, with respect to the fact that majority of the population has shifted to online shopping. This is due to the fact that the market giants are using a lot of money to please the customers, thereby providing and spoiling them with a number of products and delivery options to choose from. To compete in the market, the vendors need to achieve highest levels of customer satisfaction while optimizing their expenditure.

Since 2017, the percentage of online sales has risen from 34.5% to 53.9% in 2021, in the US alone. This upward trend along with the current scenario suggests that there is not going to be a lowering in the percentages of online sales anytime soon. This project makes use of the well-established Differential Evolution algorithm and realworld data.

ISSN No:-2456-2165

CHAPTER ONE

INTRODUCTION

An effective way to grow your business would be to ensure that customer satisfaction is always high. This includes providing the best available services to the customer. Although, this does not mean that the company should not make any profit. To tackle this and to ensure that the lowest cost possible is obtained, this project has been selected.

We will be examining a programme that uses Differential Evolution to optimize our cost model in this project, using real world data.

A. Overview

With the introduction of e-commerce, the retail industry has touched new heights. This forces majority of the sellers and vendors to push their products online. This is advantageous for the customers as they can order anything from daily groceries to large electronic appliances with just the click of a button. The issue here now lies in the costs for the vendor. To be able to deliver with the best quality of service, vendors should not only think about customer fulfilment but also about reducing their costs. The key is efficient product lifecycle management and structuring the flow of products through customized e-commerce logistics solutions. All these demands a proactive approach from e-commerce retailers eying to match the ever-increasing demand

But most of the retailers on e-commerce find it difficult avail logistics optimization and considers it as a business constraint on the supply chain front.

One of the ways to keep up with the ever-increasing demand is to utilize multiple depots to cater to the public. To win over a customer in such a cut-throat environment is to speed up the delivery process, but at the same time the vendors must not go over his budget, which would incur losses. Hence, cost optimization is a must.

In this project, I will be considering the cost optimization for Perishable products, especially focusing on milk. Milk is such a commodity that is delivered to lakhs of customers daily. It being a perishable product, the entire chain needs to fast, but at the same time, cost effective.

B. Basic Concepts

➢ *Supply Chain Management*

As defined in [1], a supply chain is the set of entities that are involved in the design of new products and services, procuring raw materials, transforming them into semifinished and finished products and delivering them to the end customers. Subsequently, supply chain management can be defined as the management of the entire process from design of the product to disposal of it by the customer. This includes everything from designing the product to selling it to the customer and finally the customer disposing it. The supply chain of a business process comprises mainly five activities namely,

- Purchase of materials from suppliers,
- Transportation of materials from suppliers to facilities,
- Production of goods at facilities,
- Transportation of goods from facilitates to ware houses,
- Transportation of goods from ware houses to customers.

Effective supply chain management is powerful enough to be able to grow the company's service, while reducing the costs [2].

ISSN No:-2456-2165

- *Same Day Delivery*

Same day delivery means that orders are delivered within a few hours after purchasing them, or in a chosen time window on the same day. Same day delivery is a game changer because it combines the immediate product availability of retail with the convenience of ordering from home.

ISSN No:-2456-2165

CHAPTER TWO

REVIEW OF LITERATURE

A. How retail can scale fast using logistics optimization

As described in the work of [3], retail is soaring with the help of e-commerce; many sellers are coming to sell their products online. This results in an increased competition and a wider range of products to choose from. Winning a customer in such competition involves speeding up delivery but at the same time not spending huge amounts of money on delivery. This requires logistic solutions. Big corporations are spending a lot of money to please customers, spoiling them with multiple options and various delivery routes and styles.

- *Optimization Technology*

This is the first step is mapping out the entire supply network. Next, we need to collect of information like costs of distribution, size of order/staff etc., shipping turnaround time, SKU to customer flow. Hurdles can be overcome by using scalable and responsive SCM system.

- *Fulfilling Same day Delivery*

96% prefer SDD, meaning order should be ready to ship within 3-4 hours of them placing an order. Last Mile Delivery is another term used, which means integrating the delivery process with an efficient route planning algorithm which use AI and ML to map out the most efficient and fastest route possible.

- *Developing cost-effective SCM*

Scanning the entire SCM system and automating all possible steps results in massive savings. Other areas of cost saving include – improving supplier relations, better storage strategy, using business intelligence for analysing customer demands, fast movement of supplies.

- *Managing live delivery with end-to-end visibility*

Logistics optimization ensures that the nearest and best suited driver is chosen for a timely delivery, same is done for return orders. This provides a better way to process return orders and give a better and more satisfactory customer service. The UI also enables customers to see live tracking of their order through a map, with real time driver information. The app also uses algorithms to select shortest and fastest path by taking into account factors like traffic and road blocks. By adopting such methods, 60% retainment of customers is ensured with 40% even vouching for the brand.

B. Supply Chain Management in a Dairy Industry – A Case Study

The work of [4] gives us a great overview of the problem statement that is being solved. A supply chain is an integrated system wherein a number of various business entities (i.e. suppliers, manufacturers, industrial customers, distributors, retailers) work together to address issues of both materials flow and information flow. A reference model - the Supply Chain Operations Reference model (SCOR), has been developed by the Supply-Chain Council (SCC). This process reference model contains standard description of management process and a framework of relationships among the standard processes. Ganeshan et.al. explored the basics of supply chain management from a conceptual perspective by tracing the roots of the definition and the origins of the concept from a broad stream of literature. Pyke and Cohen analysed the management of materials in an integrated supply chain and develop a Markov chain model for a three-level production distribution system. Cohen and Huchzermeier presented a survey of the literature pertaining to analytic approaches for global supply chain strategy analysis and planning. The integrated supply chain network model is developed to capture the complexities of a multi-product, multi-echelon, multi-country, multiperiod planning problem for the optimal choice of facility locations, capacity and technology used. Sabri and Beamon developed an integrated supply chain model for use in simultaneous strategic and operation supply chain planning.

ISSN No:-2456-2165

Less research has been aimed at co-ordination of procurements, production and distribution systems. In this paper an attempt has been made to develop a coordinated supply chain planning model with procurement, production and distribution systems.

A multi-objective function is then formulated to minimize cost subject to supplies, plant and distribution capacities, production and distribution through put limits and customs demand requirements. Total cost includes fixed costs of production and distribution, variable costs of production, distribution and transportation. This model consists of four echelons namely Suppliers, Plants, Distribution Centres (DCs), and Customer zones (CZs).

ISSN No:-2456-2165

CHAPTER THREE

SYSTEM MODEL

A. Mathematical Model

I have designed a basic cost function, which would be subjected to different limits and variables and then it would be minimised using an appropriate algorithm.

- *Material Cost*

  $MC = \sum A_{v,m} \times C_{v,m} \; \forall v,m$

  where v = vendor, m = material

First, we have the Material Cost, which is the cost of the raw materials going from the vendor to the facility, where the raw material will be turned into the finished product. The material cost is the product of cost per material and the total amount of material. The limits go for all vendors (v) and all materials (m).

- *Production Cost*

  $PC = \sum A_{g,f} \times C_{g,f} \; \forall f,g$ where g = goods, f = facility

Next, we have the production cost, which is the total cost of producing the finished goods. This is the product of the cost of production per goods and the total amount of goods produced. Since production is done at the facility, the limits are valid for all goods produced (g) and all facilities where the goods are produced (f).

- *Transportation Cost*

  $TC = \sum A_{m,v,f} \times C_{m,v,f} \; \forall v,f,m +$
  $\sum A_{g,f,w} \times C_{g,w,f} \; \forall f,w,g +$
  $\sum A_{g,w,c} \times C_{g,w,c} \; \forall w,c,g$

  Where v = vendor, f = facility, m = material, w = warehouse, g = goods, c = customer

Next comes the transportation cost. This function has 3 sub parts namely, transportation of raw transportation of goods from warehouse to customer. All 3 are derived similarly, they are the products of the transportation cost of 1 material/goods and the amount of goods being transported.

- *Inventory Cost*

For Inventory Cost, we have done it differently. Since we are considering milk, a perishable product, it needs to be stored in refrigerated warehouses. If we were to calculate it the way we calculated the other costs, there would be a time factor which would come into play. Hence, to not complicate things too much, we assume a constant time (let's say 6 hours), for which the milk would be stored in the warehouse. The Inventory Cost thus becomes a constant value.

- *Objective Function*

  Objective Function =

  $\text{Min} (\sum A_{v,m} \times C_{v,m} \; \forall v,m + \sum A_{g,f} \times C_{g,f} \; \forall f,g + \sum A_{m,v,f} \times C_{m,v,f} \; \forall v,f,m + \sum A_{g,f,w} \times$

  $C_{g,w,f} \; \forall f,w,g + \sum A_{g,w,c} \times C_{g,w,c} \; \forall w,c,g + IC)$

  Our objective function hence would be the minimization of the above costs.

ISSN No:-2456-2165

➢ *Limits*

$0 \leq Am,v,f \leq Am,v,f(max)$
$0 \leq Ag,f,w \leq Ag,f,w(max)$
$0 \leq Ag,w,c \leq Ag,w,c(max)$
$Ag,f(min) \leq Ag,f \leq Ag,f(max)$

We subject our model to the following limits,

- The amount of material stored in facility must not exceed the maximum value of the facility.
- The amount of goods stored in facility must not exceed the maximum value of the facility.
- The amount of material stored in warehouse must not exceed the maximum value of the warehouse.

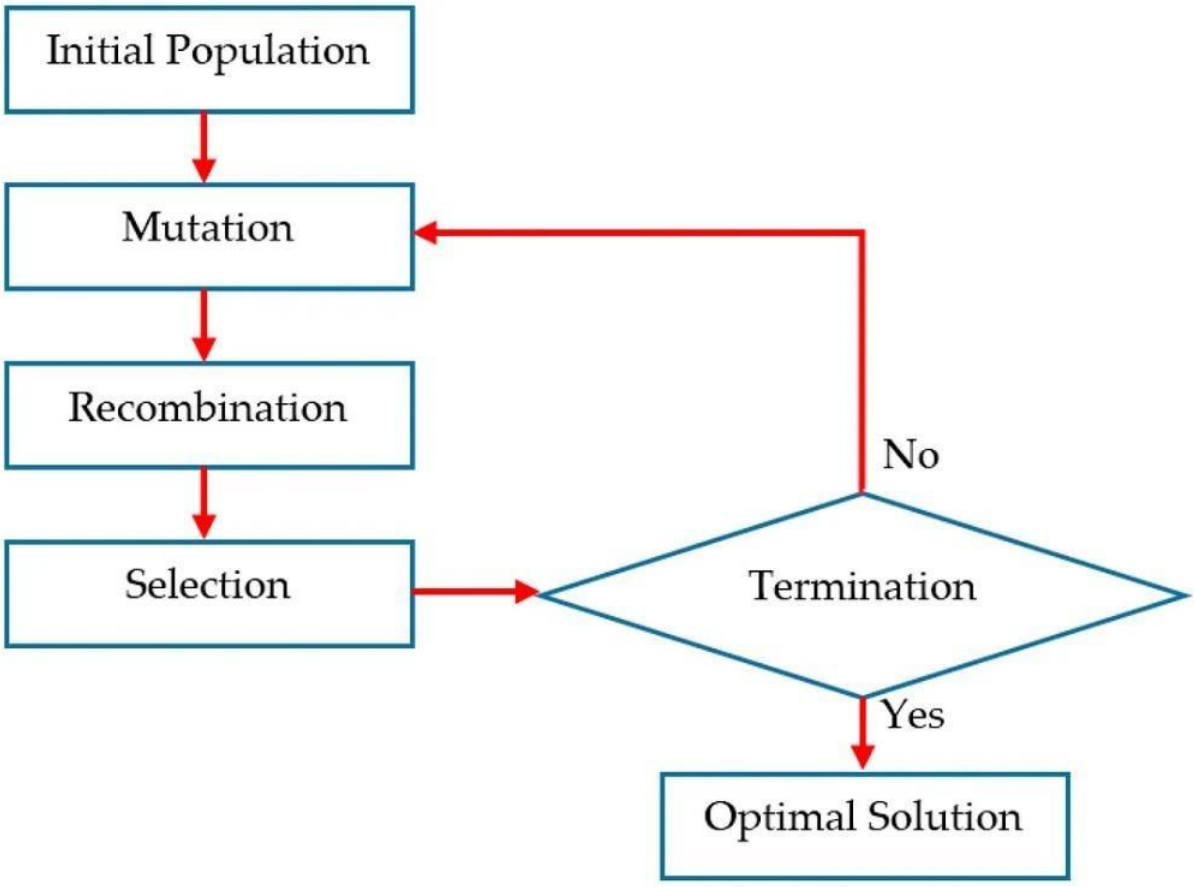
➢ *Restrictions*

$Am,v,f = (\text{Amount sold})v,f \ \forall v,m,f$
$Ag,w,c = (\text{Amount sold})g,c \forall\ w,c,g$

The restrictions applied are as follows,

- The amount of materials going from the vendor to the facility must be equal to the amount of goods sold by that vendor to that facility.
- The amount of goods going from the warehouse to the customer must be equal to the amount of goods sold to that customer.

B. Differential Evolution



[5] Fig. 1: Differential Evolution Flowchart

ISSN No:-2456-2165

As given by [5], we can say that Differential evolution (DE) is a population-based metaheuristic search algorithm that optimizes a problem by iteratively improving a candidate solution based on an evolutionary process. Differential evolution can be considered to be one of the most robust algorithms based on population. It does not make any assumption about the underlying information and can explore the entire data very speedily.

The algorithm works by having a set of initial agents and creates the next set of agents by using a set of parameters. This next set of agents created is such that they have the best objective values. If not, then they are simply discarded. This entire process keeps repeating itself until a certain criterion is not met. When this happens, the algorithm stops.

➢ *Standard Differential Evolution Algorithm*

- **Step 1**: Initialize the algorithm to G = 0, where G is the total number of iterations.
- **Step 2**: Generate 3 random individuals from the initial population, such that they are distinct and not equal to one another.
- **Step 3**: Next, form the donor vector using these individuals using the mutation function. **Step 4**: Perform crossover. This is done by developing a trial vector which is formed such that, if the index of the donor vector is less than or equal to the crossover probability then choose the donor vector. Else choose the original individual.
- **Step 5**: Evaluate if the objective value of the trial vector is less than or equal to the objective value of the individual. If so then replace the individual with the trial vector, else keep it. **Step 6**: Check if the stopping criterion has been met. If met then stop, else go to step 2.

ISSN No:-2456-2165

Data:
NP: population size, *F*: mutation factor, *CR*: crossover probability, *MAXFES*: maximum number of functions evaluations
INITIALIZATION $G = 0$; Initialize all *NP* individuals with random positions in the search space;
while $FES < MAXFES$ **do**
  for $i \leftarrow 1$ **to** *NP* **do**
    GENERATE three individuals x_{r1}, x_{r1}, x_{r1} from the current population randomly. These must be distinct from each other and also from individual x_i, i.e. $r_1 \neq r_2 \neq r_3 \neq i$
    MUTATION Form the donor vector using the formula: $\mathbf{v}_i = \mathbf{x}_{r_1} + F(\mathbf{x}_{r_2} - \mathbf{x}_{r_3})$
    CROSSOVER The trial vector $\mathbf{u}_i$ is developed either from the elements of the target vector $\mathbf{x}_i$ or the elements of the donor vector $\mathbf{v}_i$ as follows:

$$u_{i,j} = \begin{cases} v_{i,j}, & \text{if } r_{i,j} \leq CR \text{ or } j = j_{rand} \\ x_{i,j}, & \text{otherwise} \end{cases}$$

    where $i = \{1, \ldots, NP\}$, $j = \{1, \ldots, D\}$, $r_{i,j} \sim \cup(0, 1)$ is a uniformly distributed random number which is generated for each j and $j_{rand} \in \{1, \ldots, D\}$ is a random integer used to ensure that $\mathbf{u}_i \neq \mathbf{x}_i$ in all cases
    EVALUATE If $f(\mathbf{u}_i) \leq f(\mathbf{x}_i)$ then replace the individual $\mathbf{x}_i$ in the population with the trial vector $\mathbf{u}_i$
    $FES = FES + NP$
  end
  $G = G + 1$;
end

[5] Fig. 2: Standard Differential Evolution Algorithm

- Mutation Function

In each iteration, for each individual x(i), there is a donor vector v(i) generated by using the mutation function. This mutation function is what generates the new donor vectors at each iteration.

There are 6 main types of mutation functions that can be defined:

- ✓ **DE/rand/1** – $v_i = x_{r1} + F\,(x_{r2} - x_{r3})$
- ✓ **DE/best/1** – $v_i = x_{best} + F\,(x_{r1} - x_{r2})$
- ✓ **DE/rand/2** – $v_i = x_{r1} + F\,(x_{r2} - x_{r3}) + F\,(x_{r4} - x_{r5})$
- ✓ **DE/best/2** – $v_i = x_{best} + F\,(x_{r1} - x_{r2}) + F\,(x_{r3} - x_{r4})$
- ✓ **DE/current-to-best/1** – $v_i = x_i + F\,(x_{best} - x_i) + F\,(x_{r1} - x_{r2})$
- ✓ **DE/current-to-rand/1** – $v_i = x_i + \text{rand}\,(x_{r1} - x_i) + F\,(x_{r2} - x_{r3})$

In the above-mentioned functions, x_{r1}, x_{r2}, x_{r3}, x_{r4}, x_{r5} and x_{r6} are all random elements that are chosen from the range of the initial population size, such that they are mutually exclusive and their indices are not same.

F is a scaling factor that is applied to the mutation function. This scaling factor is a positive scaling parameter applied to the difference vector.

x_{best} is the element from among the population which has the best objective value.

➢ *Composite Differential Evolution (CODE)*

An extensive study done by [5] shows that Composite Differential Evolution was designed by a random combination of trial vector generation methods. This resulted in a new set of trial vectors. This combination was done by reviewing previous literature. The algorithm used by CODE is easy in implementation, along with being simple in nature.

It uses 3 main trial vector generation methods:

- Rand/1/bin
- Rand/2/bin
- Current-to-rand/1

There is also a parameter pool, which contains a set of parameter settings. During generation of the trial vector, each of the 3 trial vectors is subjected to the appropriate parameters from the pre-defined pool. If any of the 3 is better than the target, then it is selected or else it is rejected.

This method is further explained in [6].

Data: NP, $MAXFES$
INITIALIZATION $G = 0$; $FES = NP$;
while $FES < MAXFES$ **do**
  $P_{G+1} = 0$
  for $i \leftarrow 1$ **to** NP **do**
    GENERATE three trial vectors $u_{i_1,G}, u_{i_2,G}$ and $u_{i_3,G}$ for the target vector $x_{i,G}$ using the three generation schemes "rand/1/bin," "rand/2/bin," and "current-to-rand/1," each with control parameter setting randomly selected from the parameter candidate pool: $[F = 1.0, CR = 0.1]$, $[F = 1.0, CR = 0.9]$ and $[F = 0.8, CR = 0.2]$.
    EVALUATE the objective function value of the three trial vectors $u_{i_1,G}, u_{i_2,G}$, and $u_{i_3,G}$
    CHOOSE the best trial vector $u^*_{i,G}$ from the three trial vectors
    $P_{G+1} = P_{G+1} Uselect(x^*_{i,G}, u^*_{i,G})$
    $FES = FES + 3$
  end
  $G = G + 1$;
end

[5] Fig. 3: CODE algorithm

➢ *Self Adaptive Control Parameters Differential Evolution (JDE)*

As shown by [5], JDE displays a set of control parameters which are self-adapting in nature. It uses an evolution process that optimizes the control parameters itself. Despite the positive thought behind this, it is relatively difficult to implement. By having self-adapting control parameters, the entire Differential Evolution becomes flexible, in order to achieve the most optimized output. JDE output is more dependant on the values of mutation factor F, than the values of crossover probability CR.

The control parameters are defined in the following way:

$F_{i,G+1} = F_l + rand1F_u$; if $rand2 < \tau 1$
$= F_{i,G}$; otherwise

$CR_{i,G+1} = rand3$; if $rand4 < \tau 2$
$= CR_{i,G}$; otherwise

Where,

Randj is a random value in [0,1] $\tau 1$ and $\tau 2$ are the probabilities to adjust F and CR.

$F_l = 0.1$, $F_u = 0.9$

The result of this gives us a new value of F in the range [0.1,1] and CR in the range [0,1]. JDE is known to give better results to the DE/rand/1 mutation function of standard DE. A further more extensive study is given in [7].

➢ *Adaptive Differential Evolution with Optional External Archive (JADE)*

[5] gives us a look at JADE, which was suggested as a way to improve the standard DE by updating the control parameters in an adaptive manner. DE/current-to-pbest is the method used to achieve this adaptiveness. This method also comes with an optional external archive. This ensures that along with the best solution, 100p % solutions are also selected (p is in the range (0,1]).

The mutation vector is generated as shown:

$v_{i,g} = x_{i,g} + F_i (x_{pbest,g} - x_{i,g}) + F_i (x_{r1,g} - x_{r2,g})$

Where,

x^p_{best} is chosen randomly from the top 100p % solutions in the current population

F_i is the mutation factor related to x_i

Further study on this algorithm is provided by [8].

➢ *Self-Adaptive Differential Evolution (SADE)*

Along with the ability of JADE to self-adapt the control parameters, SADE also features the ability to self-adapt the trial vector generation methods, as explained in the work of [5]. This is done by learning from the previous experiences. This method consists of a candidate pool. A trial vector generation method is selected from this pool. The method of selecting is based on a probability, that is based upon the rate of success for generating better solutions within a pre- defined learning period LP.

At the start, since there is no previous data to learn from, the probability of choosing any 1 of the methods from the candidate pool are the same, which is equal to 1/K ; where K is the total number of methods present in the candidate pool. Population size would still remain a user defined value, since it will depend on the problem statement itself. The value of the mutation factor F is calculated using normal distribution having a

mean value of 0.5 and a standard deviation value of 0.3. Consequently, F would fall within the range of [-0.4,1.4].

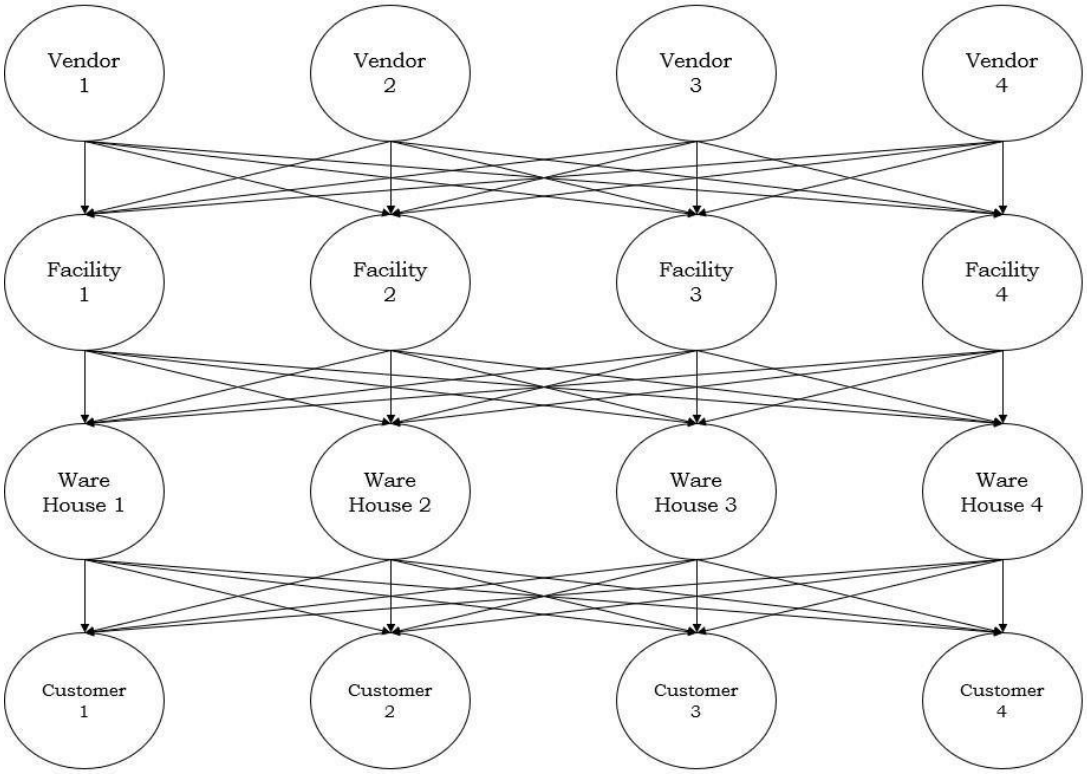
For the value of crossover probability CR, again a normal distribution is used. Initially, the mean CR_m is set to 0.5 and standard deviation STD is set to 0.1. Along with this, a $CRMemory_k$ is also assigned, which will store the CR values of the kth iteration of successful trial vector generation. These CR values that are stored in the memory are used to improve the CR values of the next iterations. After the newer trial vectors have been evaluated, the CR values of the previous iterations are replaced by the CR values of the current successful generation. As shown in [9], it is clear that the parameter settings are self-adapted by using a learning method.

C. Comparison of Genetic Algorithm and Differential Evolution

A very comprehensive work done by [10] shows that, Genetic Algorithm is generally used in cases where a solution generated by many iterations is has an attached time constraint and the mathematical solution is generally not available easily. It finds a solution in a rapid way. There is also the proof of convergence available for an advanced version of the Genetic Algorithm. The main drawback of using this algorithm is that the control parameters and the actual implementation requires a lot of adjustments and tuning.

The same work done by [10] also displays that Differential Evolution often gives better results compared to Genetic Algorithm. Differential Evolution can also be put to use in real world applications in spite of the fact that they are noisy and multi-dimensional in nature. Another advantage of it over Genetic Algorithm is that its control parameters, namely Crossover Probability CR and Mutation Factor F do not require an extensive adjustment. They can work with minimal adjustments.

D. Network Model



The first layer consists of the vendors, that will supply the raw materials to the facilities. The second layer consists of the facilities that convert the raw material to the finished goods. The third layer consists of the warehouses that will store the finished goods. The fourth layer consists of the end customers that will receive the different products.

ISSN No:-2456-2165

E. Real World Data Used

Table 1: Data collected

		MC	**PC**	**TC**			**IC**
				3 kms	51.5 kms	100 kms	
Inbound Delivery	Open	43.12	5.7 (milk) + 1 (add-on)	0.04	0.67	1.3	15000000000
	Tanker			0.03	0.47	0.91	
Outbound Dellivery	Wholesale			0.05	0.9	1.75	
	Retailer			0.03	0.45	0.87	

ISSN No:-2456-2165

CHAPTER FOUR

SYSTEM IMPLEMENTATION

A. Explanation of code implemented

```
import math

from numpy.random import rand
from numpy.random import choice
from numpy import asarray
from numpy import clip
from numpy import argmin
from numpy import min
from numpy import around
```

Fig. 4: DE Code

Here we can see that first we import all the necessary libraries. We have mainly used the ‘math’ library. Along with this, we have also imported specific sub libraries from the ‘numpy’ library.

These include:

- **Rand:** Used for creating an array of random numbers. [11]
- **Choice:** Used for creating a 1-D array of random numbers. [11]
- **Asarray:** Used to convert the input into an array. [12]
- **Clip:** Used to limit the values in an array. [12]
- **Argmin:** Returns the indices of the minimum values. [12]
- **Min:** Returns the minimum value. [12]
- **Around:** Round off evenly to the specified number of decimals. [12]

```
def obj(x):
    total = []
    for i in range(len(x)-1):
        total += f(x)
    return total
```

Fig. 5: DE Code

ISSN No:-2456-2165

Next, we define our objective function obj(x). We also define an empty array called 'total' that will store the output of the objective function.

```
def mutation(x, F):
    return x[0] + F * (x[1] - x[2])
```

Fig. 6: DE Code

Next, we define the mutation function. As discussed earlier, Standard DE has 6 main types of mutation functions. We have used DE/rand/1.

```
def check_bounds(mutated, bounds):
    mutated_bound = [clip(mutated[i], bounds[i, 0], bounds[i, 1]) for i in range(len(bounds))]
    return mutated_bound
```

Fig. 7: DE Code

Next, we define the function that will check whether the iteration is within the specified bounds or not. It does this by using the 'clip' function of the numpy module [12]. If the iteration goes beyond the specified bounds, then it simply clips it down to the specified range.

```
def crossover(mutated, target, dims, cr):
    p = rand(dims)
    trial = [mutated[i] if p[i] < cr else target[i] for i in range(dims)]
    return trial
```

Fig. 8: DE Code

Next, we define the function that will perform crossover. For this, first, the variable 'p' will generate a random value for every dimension. The trial vector is subsequently selected using a simple binary crossover. If the value of p is less than the Crossover Probability CR then the mutated vector is selected, else the target vector will be selected.

```
def differential_evolution(pop_size, bounds, iter, F, cr):
    pop = bounds[:, 0] + (rand(pop_size, len(bounds)) * (bounds[:, 1] - bounds[:, 0]))
    obj_all = [obj(ind) for ind in pop]
    best_vector = pop[argmin(obj_all)]
    best_obj = min(obj_all)
    prev_obj = best_obj
```

Fig. 9: DE Code

Now, the main function starts. First, we define the initial population 'pop' randomly, but within the specified bounds. Next, we assess the population to find the best performing vector among them, by using the 'argmin' and the 'min' functions [12]. The best performing vector is stored in best_obj.

ISSN No:-2456-2165

```
for i in range(iter):
    for j in range(pop_size):
        candidates = [candidate for candidate in range(pop_size) if candidate != j]
        a, b, c = pop[choice(candidates, 3, replace=False)]
        mutated = mutation([a, b, c], F)
        mutated = check_bounds(mutated, bounds)
        trial = crossover(mutated, pop[j], len(bounds), cr)
        obj_target = obj(pop[j])
        obj_trial = obj(trial)
        if obj_trial < obj_target:
            pop[j] = trial
            obj_all[j] = obj_trial
    best_obj = min(obj_all)
    if best_obj < prev_obj:
        best_vector = pop[argmin(obj_all)]
        prev_obj = best_obj
        print('Iteration: %d f([%s]) = %.5f' % (i, around(best_vector, decimals=5), best_obj))
return [best_vector, best_obj]
```

Fig. 10: DE Code

Now, we begin the iterations, ‘i’ is defined for the range of iterations whereas ‘j’ is defined to run across the entire initial population. Next, we choose the three candidates ‘a’, ‘b’ and ‘c’, at random, for the mutation function. These are selected such that they are not equal to the current one. After performing mutation, we check if the bounds are broken. If not, then we perform crossover. We compute the objective values for both, the target vector ‘obj_target’ and the trial vector ‘obj_trial’. After this, we perform selection. For selection, the objective values of the target and trial vectors are compared. If the trial vector has a lower objective value, then we replace the target vector with it and store the new objective value. Now we find the best performing vector at each iteration by using the ‘min’ function [12]. Next, we compare the current and the previous iteration’s objective values and store the best one. Finally, we report the progress at each iteration.

```
pop_size = 500
bounds = asarray([(-5.0, 5.0), (-5.0, 5.0)])
iter = 5
F = 0.5
cr = 0.7
```

Fig. 11: DE Code

At the end, we conclude by defining the initial population size ‘pop_size’, the bounds for iterations ‘bounds’, the number of iterations to perform ‘iter’, mutation factor ‘F’, and the crossover probability ‘CR’.

B. Implementation

➢ *Using the Rosenbrock Function*

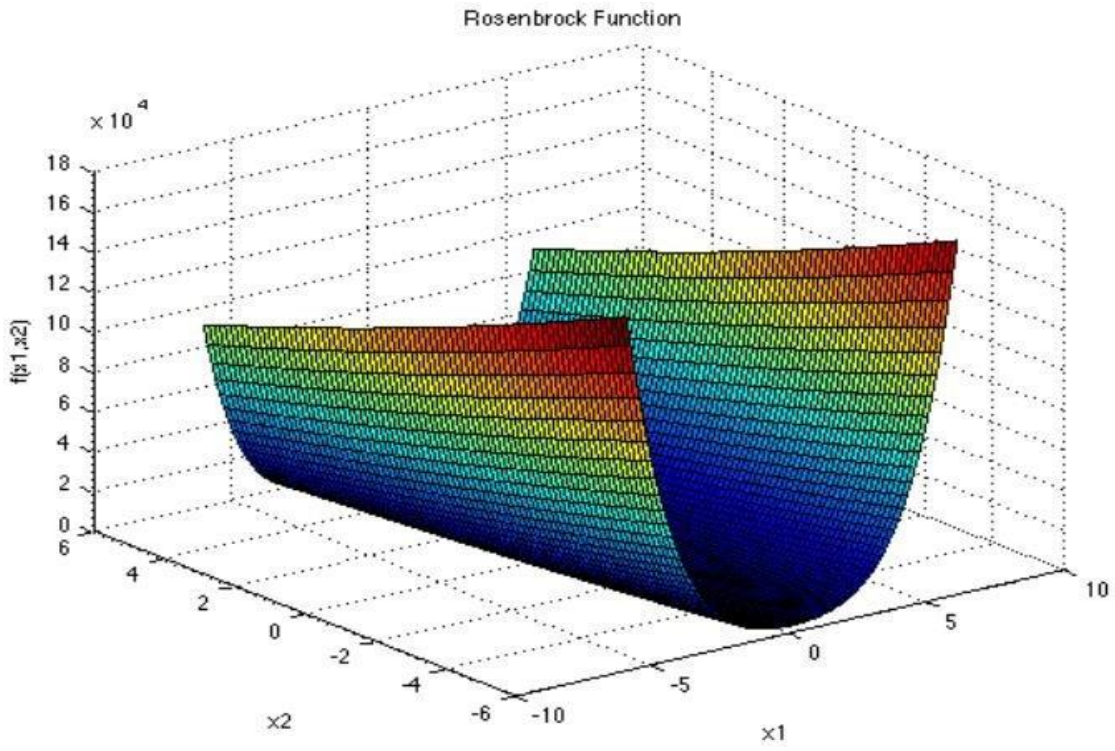
```
Iteration:  7 f([[0.97841 0.96458]]) = 0.00580
Iteration: 12 f([[1.02934 1.05441]]) = 0.00350
Iteration: 14 f([[1.02959 1.05721]]) = 0.00168
Iteration: 27 f([[0.95945 0.92093]]) = 0.00166
Iteration: 29 f([[0.96362 0.92824]]) = 0.00133
Iteration: 33 f([[0.96453 0.93014]]) = 0.00126
Iteration: 34 f([[1.02237 1.04274]]) = 0.00112
Iteration: 35 f([[1.02259 1.04614]]) = 0.00053
Iteration: 36 f([[0.99409 0.98853]]) = 0.00004
Iteration: 45 f([[1.00314 1.00583]]) = 0.00003
Iteration: 47 f([[0.99619 0.99218]]) = 0.00002
Iteration: 48 f([[1.00028 1.00065]]) = 0.00000

Solution: f([[1.00028 1.00065]]) = 0.00000
```

Fig. 12: Rosenbrock Function Optimization

The Rosenbrock Function si a popular test function used for testing such optimization algorithms. It is defined as follows [13]:

$$F(x) = \sum i = 1d\text{-}1\ [100(xi+1 - xi2)2 + (xi - 1)2]$$

The Rosenbrock Function is also known as the Banana Function. The function is unimodal in nature, whose minima lies in a valley as shown below. Despite this, the convergence is not easy to find.



[13] Fig. 13: Rosenbrock Function Graph

ISSN No:-2456-2165

➢ *Using the mathematical model designed*

- Excluding the Inventory Cost

```
Solution: f([[-1.45067  0.24073]]) = 0.03000

Process finished with exit code 0
```

Fig. 14: Result of code

- Including the Inventory Cost

```
Solution: f([[ 1.60935 -3.22577]]) = 0.03000

Process finished with exit code 0
```

Fig. 15: Result of code

ISSN No:-2456-2165

CHAPTER 5

CONCLUSION / FUTURE SCOPE

A. Conclusion

For this project, I have studied the Supply Chain Management and the effect of e-commerce on the retail sector. Along with this, a basic cost optimization model has been developed. The different costs have been identified and the objective function has been developed. The objective function is the minimization of the total cost. This minimization is to be achieved using an iterative algorithm.

I have used the very widely popular Differential Evolution for this. I have implemented the algorithm using the Rosenbrock Function first, before implementing it using the collected real world data.

In conclusion, such a solution strategy is a very viable option for such optimization problems.

B. Future Scope

In the future, the focus of supply chain management will be on leveraging advanced technologies like AI, blockchain, and IoT to enhance efficiency, lower costs, and gain real-time visibility. This will enable businesses to optimize their supply chain operations, automate processes, and have better control over their supply chain activities.

Moreover, sustainability and social responsibility will be paramount in supply chain management. Organizations will prioritize implementing environmentally friendly practices, such as minimizing their carbon footprint and collaborating with suppliers who uphold ethical and sustainable standards.

Collaboration and partnerships within the supply chain will also witness growth as companies seek to pool resources and expertise to improve efficiency and reduce costs. This trend will foster the development of more integrated and resilient supply chains capable of handling disruptions and unpredictable market conditions.

REFERENCES

[1.] Lu, Lauren & Swaminathan, Jayashankar. (2015). Supply Chain Management. International Encyclopedia of the Social & Behavioral Sciences. 10.1016/B978-0-08-097086-8.73032-7.
[2.] Kleab, K. (2017) *Important of supply chain management - IJSRP*. Available at: https://ijsrp.org/research-paper-0917/ijsrp-p6949.pdf
[3.] S. Mansuri. "How retail and E-Commerce can scale fast using logistics optimization." peerbits. https://www.peerbits.com/blog/retail-e-commerce-can-scale-fast-using-logistics- optimization.html
[4.] K. Venkata Subbaiha, " Supply Chain Management in a Dairy Industry – A Case Study", Proceedings of the World Congress on Engineering 2009 Vol I WCE 2009, July 1 - 3, 2009, London, U.K
[5.] Georgioudakis, M. and Plevris, V. (2020) 'A comparative study of differential evolution variants in constrained structural optimization', Frontiers in Built Environment, 6. doi:10.3389/fbuil.2020.00102.
[6.] Y. Wang, Z. Cai and Q. Zhang, "Differential Evolution With Composite Trial Vector Generation Strategies and Control Parameters," in IEEE Transactions on Evolutionary Computation, vol. 15, no. 1, pp. 55-66, Feb. 2011, doi: 10.1109/TEVC.2010.2087271.
[7.] J. Brest, S. Greiner, B. Boskovic, M. Mernik and V. Zumer, "Self-Adapting Control Parameters in Differential Evolution: A Comparative Study on Numerical Benchmark Problems," in IEEE Transactions on Evolutionary Computation, vol. 10, no. 6, pp. 646-657, Dec. 2006, doi: 10.1109/TEVC.2006.872133.
[8.] J. Zhang and A. C. Sanderson, "JADE: Adaptive Differential Evolution With Optional External Archive," in IEEE Transactions on Evolutionary Computation, vol. 13, no. 5, pp. 945-958, Oct. 2009, doi: 10.1109/TEVC.2009.2014613.
[9.] K. Qin and P. N. Suganthan, "Self-adaptive differential evolution algorithm for numerical optimization," 2005 IEEE Congress on Evolutionary Computation, Edinburgh, UK, 2005, pp. 1785-1791 Vol. 2, doi: 10.1109/CEC.2005.1554904.
[10.] Hegerty, B., Hung, C.-Ch. and Kasprak, K. A Comparative Study on Differential Evolution and Genetic Algorithms for Some Combinatorial Problems. tech. Marietta, GA.
[11.] Developers, N. (2022) Random sampling (numpy.random)#, Random sampling (numpy.random) - NumPy v1.24 Manual. Available at: https://numpy.org/doc/stable/reference/random/index.html
[12.] Developers, N. (2022a) Array creation routines#, Array creation routines - NumPy v1.24 Manual. Available at: https://numpy.org/doc/stable/reference/routines.array-creation.html
[13.] Surjanovic, S. and Bingham, D. (2013) Virtual library of simulation experiments:, Rosenbrock Function. Available at: https://www.sfu.ca/~ssurjano/rosen.html