Cairo University

Faculty of Computers and Artificial Intelligence

Term Research Project

"Supply chains and Inventory control using Differential Evolution algorithm"

Department: Operations Research and Decision Support

Course Name: Computational Intelligence

Course Code: DS313/DS351

Instructor: Assoc. Prof. Ayman Ghoneim - Assoc. Prof. Sally Kassem

Team Members

Num.	Name	ID
1	Mohamed Mohamed Hassan	20221239
2	Mostafa Ahmed Ashour	20220339
3	Mariam Mohamed Hanafy Saleh	20221146
4	Basma Mmdouh Tawfik	20221036

Table of Contents

Supply Chain Management:

An Overview

1. Overview & Definition:

Supply chain management (SCM) is a vital business discipline that oversees the flow of goods, services, and information from raw material suppliers to end customers. It aims to maximize customer value and secure a sustainable competitive advantage by coordinating activities like sourcing, procurement, production, and distribution across multiple organizations. SCM builds on existing management theories, requiring significant adaptations in organizational structure, information systems, and strategic planning. Unlike traditional approaches that focus on isolated parts, SCM emphasizes managing the entire supply chain holistically, ensuring seamless integration and efficiency.

2. **Evolution & History:** The evolution of supply chains reflects a shift from fragmented, linear processes to integrated, global systems:

   - **Early Days**: Supply chains were simple, individualized links between manufacturers, warehouses, and retailers, often plagued by miscommunication and poor coordination.
   - **1950s**: Interest in SCM grew with the advent of inventory management software, enhancing control in manufacturing.
   - **1970s**: Material Requirements Planning (MRP) emerged, aligning raw material flows with production schedules.
   - **1980s**: Manufacturing Resources Planning (MRP-II) expanded to cover broader resources, paving the way for Enterprise Resource Planning (ERP) systems.
   - **Present**: SCM now focuses on integration, transparency, and global coordination, enabling firms to design products in one country, manufacture in another, and sell worldwide, driven by advancements in logistics and technology.

3. **Stages (SCOR Model):** The Supply Chain Operations Reference (SCOR) model outlines five essential stages of SCM:
    1. **Plan**: Crafting strategies to meet customer needs and designing an effective supply chain.
    2. **Develop**: Building robust supplier relationships and planning transportation, delivery, and payment logistics.
    3. **Make**: Manufacturing, testing, packaging, and scheduling products for delivery.
    4. **Deliver**: Managing logistics, fulfilling customer orders, and ensuring timely delivery.
    5. **Return**: Addressing customer service, handling returns, and resolving inquiries.

    These stages provide a timeless framework for optimizing supply chain operations.

4. **Supply Chain Decisions:** SCM decisions fall into two categories:
    - **Operational Decisions**: Day-to-day tasks like inventory control and production scheduling.
    - **Strategic Decisions**: Long-term choices shaping the supply chain's design, including:
        1. **Location Decisions**: Choosing sites for facilities and optimizing product flow paths.
        2. **Production Decisions**: Determining what to produce, where, and how to allocate resources.
        3. **Inventory Decisions**: Managing stock levels across the supply chain to balance cost and availability.
        4. **Transportation Decisions**: Selecting transport modes to optimize cost, speed, and reliability.

    These decisions drive both immediate efficiency and long-term competitiveness.

5. **Modelling Approaches:** SCM encounters several challenges
   - **Distribution Network Configuration**: Determining the optimal number and location of facilities.
   - **Distribution Strategies**: Selecting efficient delivery and transportation methods.
   - **Information Sharing**: Integrating data on demand, inventory, and forecasts across the chain.
   - **Inventory Management**: Balancing stock levels to avoid overstocking or shortages.
   - **Cash Flow**: Coordinating payment and financial transactions.

   A notable issue is the **Bullwhip Effect**, where uncoordinated actions amplify demand fluctuations. Improved communication and forecasting can mitigate this problem.

6. **Applications of SCM:** SCM enhances efficiency across various sectors
   - **Enterprise Resource Planning (ERP)**: Integrates functions to streamline processes and reduce costs.
   - **Information Technology (IT)**: Facilitates data flow, improving coordination and decision-making.
   - **Warehousing**: Optimizes storage to support logistics and customer service.
   - **Retail Industry**: Balances cost and service to meet consumer demands, as exemplified by Walmart.
   - **Healthcare**: Reduces costs and improves care by streamlining supply chains.

   These applications underscore SCM's broad relevance and impact.

7. **Conclusion:**
   SCM is a cornerstone of modern business, driving organizational success through efficient supply chain coordination. Companies like Dell, Walmart, Zara, Amul, and Asian Paints demonstrate their value in achieving competitive advantages. Far beyond basic logistics, SCM integrates

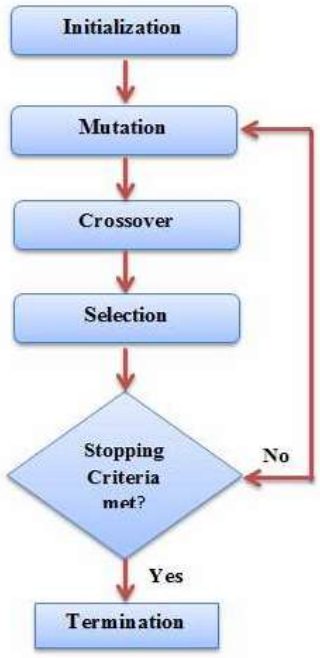
multiple disciplines, demanding strong managerial skills and strategic vision to ensure high performance across industries.

Literature Review on Differential Evolution

What is Differential Evolution (DE)?

Differential Evolution (DE) is an optimization algorithm introduced by Storn and Price in 1996. It's a **metaheuristic** method, meaning it uses smart, high-level strategies (combining randomness and local search) to solve complex problems where finding the absolute best solution is tough. DE is especially good for **continuous optimization problems**, think of tweaking numbers to minimize costs or maximize efficiency. It works similarly to genetic algorithms by:

1. Starting with a **population** of possible solutions.
2. **Mutating** and **crossing over** these solutions using a simple formula.
3. **Selecting** the best ones to move forward, generation after generation.



Paper Structure: The paper is split into four sections:

- **Introduction**: Explains DE basics.
- **Literature Review**: Covers three areas (modifications, hybridization, applications).
- **Analysis**: Breaks down trends and stats from the review.
- **Conclusion**: Wraps up findings and suggests future work.

Key Points from the Literature Review:

1. Modifications of Classic DE

Researchers have tweaked DE to make it better by adjusting:

- **Parameters**: The modifications can be classified into five criteria, which are population, learning procedures, mutation functions, control parameters, and vector generation.

The table shows the list of modifications for each author:

Table 10: The list of modifications for each author

Author	Modifications
Qin, Huang, and Suganthan(2008)	Vector generation
	control parameters
Das et al.(2009)	Vector generation
	Mutation functions
Gong et al.(Gong et al., 2010)	Learning procedures
Brest and Maucec(Brest & Maučec, 2011)	Learning procedures
Mallipeddi et al.(2011)	Mutation functions
	control parameters
	Vector generation
Wang, Cai, and Zhang(2011)	Learning procedures
	Vector generation
	control parameters
L. Wang et al. (2012)	Learning procedures
Caraffini et al.(2013)	Learning procedures
Tanabe and Fukunaga(2013)	control parameters
Tanabe and Fukunaga(Tanabe & Fukunaga, 2014)	Population

- Results for the previous changes:

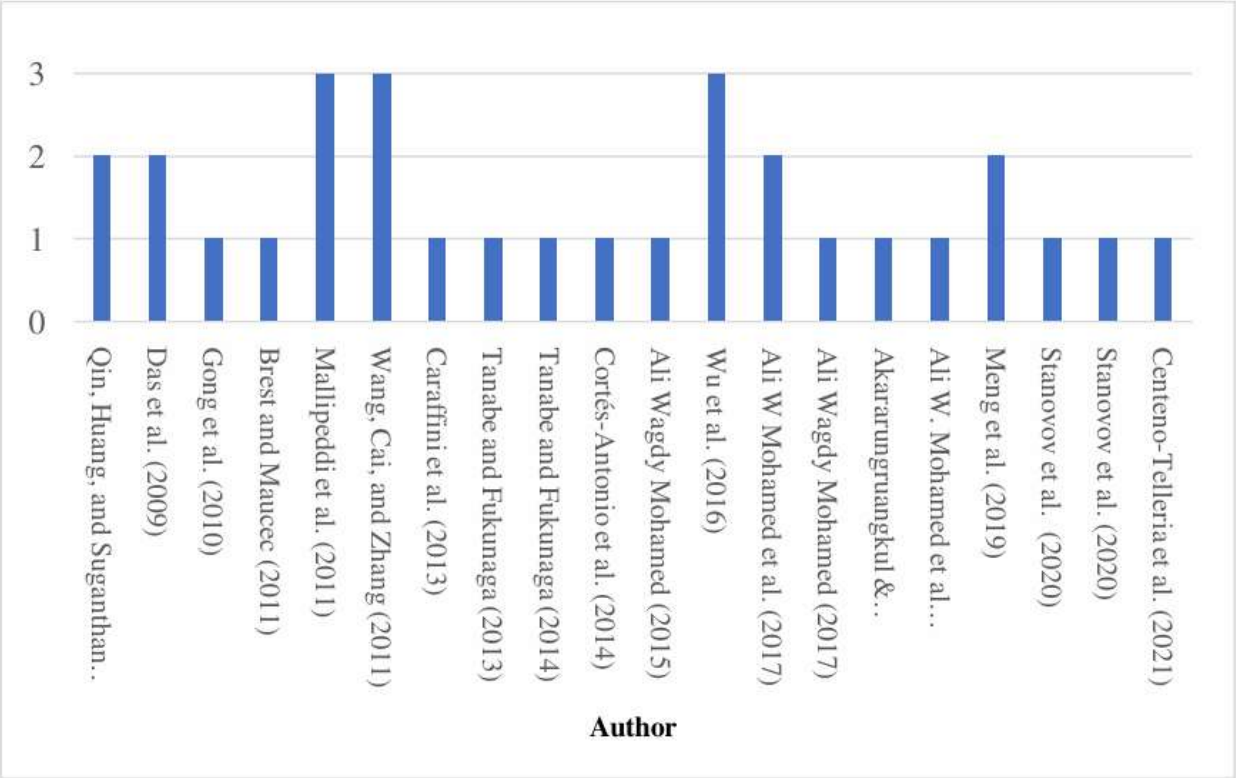


Figure 3: The relationship between authors and their modifications

- **Strategies**: Examples include:
    - **SaDE**: Self-adaptive DE that learns from past runs to adjust settings.
    - **DEGL**: Balances exploring new solutions and exploiting good ones.
    - **EPSDE** and **CoDE**: Use multiple mutation strategies for better results.
- Focus areas: Most tweaks target **learning procedures** and **mutation functions**, less so on population or vector generation.

2. Hybridization with Other Algorithms

DE is often combined with other methods to boost performance:

- Popular partners: **Cuckoo Search** (12%), **Genetic Algorithms** (12%), **Particle Swarm Optimization** (9%).
- Goal: Mix DE's strengths (exploring solutions) with others' advantages (faster convergence).

Table 11: The hybridized approaches with DE

Hybridization	**The percentage of researches**
Ant colony Optimization.	9%
The Ant Lion Optimizer.	6%
Artificial bee colony algorithm.	6%
Cuckoo search algorithm.	12%
Bat Algorithm.	9%
Firefly's Algorithm.	6%
Harmony search Algorithm.	6%
Simulated annealing Algorithm.	6%
Genetic Algorithm.	12%
Grey wolf optimizer.	3%
Whale optimization algorithm	3%
Particle swarm optimization algorithm	9%
BlackHole algorithms	3%
Biography-based optimization	6%
Hill climbing	3%

3. Applications of DE

DE solves problems across many fields:

- **Electrical/Power Systems**: Optimizing power flow or economic dispatch.
- **Manufacturing/Operations**: Scheduling or process optimization.
- **Image Processing/Pattern Recognition**: Enhancing images or classifying data.
- **Bioinformatics**: Medical data analysis.

- **Robotics**: Path planning.

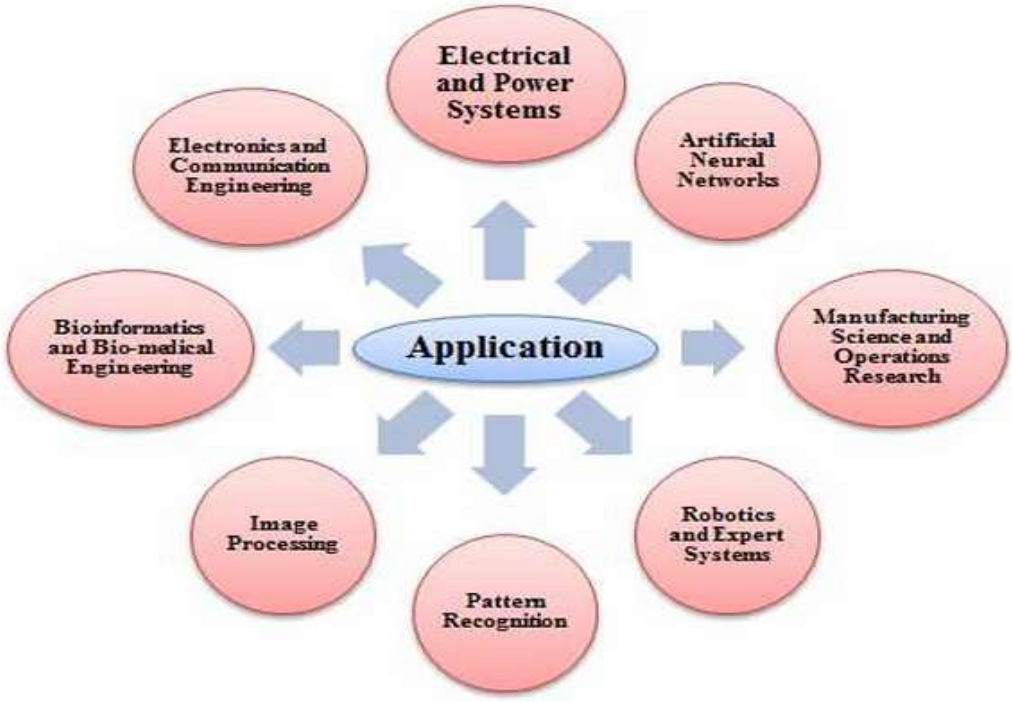
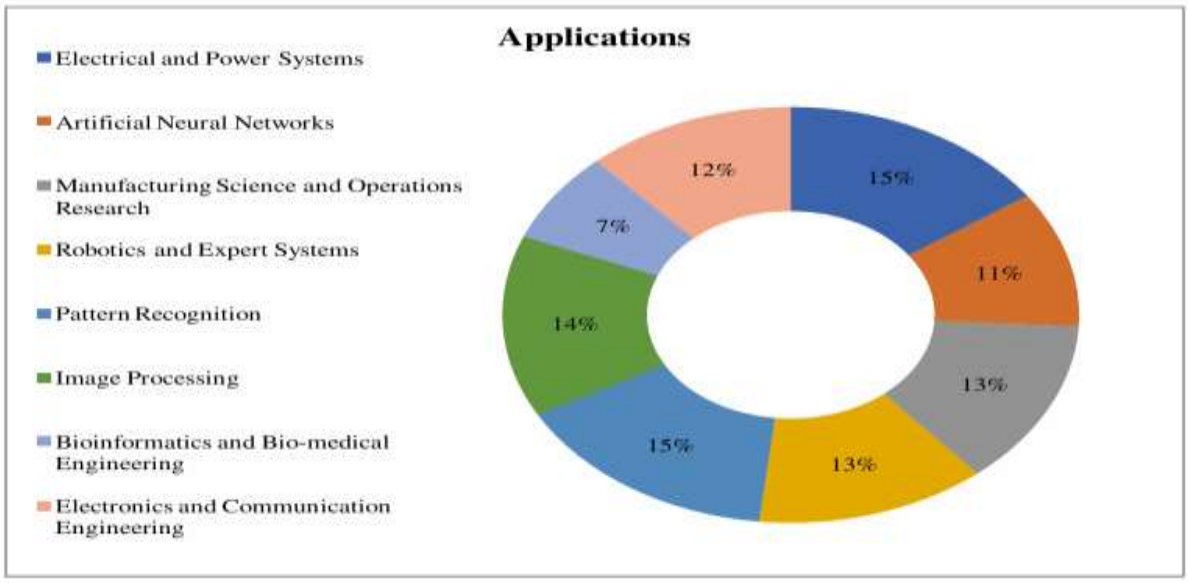


Figure 2: Diverging Radial graph of DE applications in different areas.



4. Conclusion:

This study examines the Differential Evolution (DE) algorithm, with a focus on parameter modifications, hybrid methods, applications, and publication trends from 2008 to 2021. It notes that the most significant parameter changes were made by Mallipeddi et al. (2011), Wang et al. (2011), and Wu et al. (2016), whereas vector generation and population parameters received minimal attention, indicating a need for further research in these areas. The analysis also reveals a continued interest in DE research, evidenced by numerous publications up to 2021.

Differential Evolution for Supplier Selection Problem: A DEA-based

1. Problem

In today's highly competitive global market, an efficient and effective supply chain is critical to the success of any business. One of the most significant challenges in supply chain management is selecting suitable suppliers. The complexity of supplier selection arises from the need to consider multiple criteria, such as cost, quality, reliability, and delivery performance. In many cases, thousands of potential suppliers are available, and identifying the optimal one becomes an overwhelming task. Poor supplier selection can negatively affect an organization’s operational and financial health, leading to increased costs, reduced customer satisfaction, and potential losses in market share.

Traditional supplier selection methods have employed various optimization techniques, such as linear programming, multi-objective optimization, and the analytic hierarchy process (AHP). However, many of these methods fail to capture the complexity and dynamics of real-world scenarios, where data may be imprecise, criteria may conflict, and supplier performance varies across different metrics.

To address these challenges, this paper proposes a novel approach that combines Data Envelopment Analysis (DEA) with the Differential Evolution (DE) algorithm. DEA is a non-parametric technique used to assess the relative efficiency of decision-making units (DMUs), specifically the suppliers in this context. DE, on the other hand, is an evolutionary optimization algorithm that has proven effective in solving complex and constrained optimization problems. By integrating DEA with DE, the authors aim to enhance the robustness and accuracy of supplier selection.

2. Approach

The approach presented in the paper integrates DEA with DE to evaluate and select the most efficient suppliers from a set of candidates. DEA is utilized to model the supplier selection problem by assessing each supplier's efficiency based on multiple input and output criteria. The CCR (Charnes, Cooper, and Rhodes) model of DEA, which assumes constant returns to scale, is employed in this study. The primary idea behind DEA is to calculate an efficiency score for each supplier by comparing the weighted sum of outputs to the weighted sum of inputs.

The input criteria used in the DEA model include: (Minimize)

1. **Price (PR):** Cost of goods or services.
2. **Late Deliveries (LD):** Number or frequency of delayed deliveries.
3. **Rate of Rejected Parts (RR):** Proportion of defective or rejected items.

The output criterion used is: (Maximize)

1. **Service Quality (SQ):** A measure of supplier reliability or customer satisfaction.

Goal: For each supplier, calculate an **efficiency score** between 0 and 1. A score of 1 indicates the supplier is fully efficient, meaning it performs optimally relative to others, while a score less than 1 indicates inefficiency.

DEA Concept:

DEA evaluates efficiency as the ratio of the weighted sum of outputs to the weighted sum of inputs:

$$\text{Efficiency} = \frac{\text{Weighted Sum of Outputs}}{Weighted\ Sum\ of\ Inputs}$$

However, instead of directly computing this ratio, the CCR model transforms it into a linear programming problem. For each supplier m (where m = 1, 2, …, 12) we:

- Maximize its weighted output.
- Constrain its weighted input to equal 1 (normalization).
- Ensure that, using the same weights, no supplier's efficiency exceeds 1.

The weights (W for the output and Z_1, Z_2, Z_3 for the inputs) are variables determined by the model to give each supplier its best possible efficiency score.

Mathematical Model Formulation:

The DEA model aims to maximize the efficiency score of each supplier, defined as:

1. Decision Variables: These are the weights assigned to the output and inputs, which are optimized to compute efficiency.
   - w: Weight for the output, Service Quality (SQ).
   - Z_1: Weight for the input, Price (PR).
   - Z_2: Weight for the input, Late Deliveries (LD).
   - Z_3: Weight for the input, Rate of Rejected Parts (RR).
2. Parameters: These are the observed data for each supplier.
   - SQ_m: Service Quality of supplier m (output, to be maximized).
   - PR_m: Price of supplier m (input, typically to be minimized).

- LD_m: Percentage of Late Deliveries for supplier m (input).
- RR_m: Percentage of Rejected Parts for supplier m (input).

3. Objective Function:

The goal is to maximize the efficiency of supplier m, represented by its weighted output. Since there's only one output (Service Quality), the objective is:

$$\text{Max } W \cdot SQ_m$$

Subject to:

- $Z_1 \cdot PR_m + Z_2 \cdot LD_m + Z_3 \cdot RR_m = 1$
- $W \cdot SQ_n - (Z_1 \cdot PR_n + Z_2 \cdot LD_n + Z_3 \cdot RR_n) \leq 0 \quad ; \quad \forall n=1,2,\ldots,12$
- $W \geq 0, Z_1 \geq 0, Z_2 \geq 0, Z_3 \geq 0$

- **Normalization Constraint:** The weighted sum of inputs for supplier m is normalized to 1, ensuring a consistent basis for comparison:

$$Z_1 \cdot PR_m + Z_2 \cdot LD_m + Z_3 \cdot RR_m = 1$$

- **Efficiency Constraints:** For each of the n = 1, 2, …, 12 suppliers, the efficiency (ratio of weighted output to weighted inputs) must not exceed 1 when using the same weights:

$$W \cdot SQ_n - (Z_1 \cdot PR_n + Z_2 \cdot LD_n + Z_3 \cdot RR_n) \leq 0 \quad ; \quad \forall n=1,2,\ldots,12$$

  Ensuring that the efficiency of supplier m is maximized without overrating any other supplier.

- Non-negativity Constraints: All weights must be non-negative to maintain practical interpretability:

$$W \geq 0, Z_1 \geq 0, Z_2 \geq 0, Z_3 \geq 0$$

This model is solved separately for each supplier to determine their efficiency scores.

4. **Implementation Steps Using Differential Evolution (DE):**

4.1. DE Parameters:

- Population Size (NP): 100 individuals (weight vectors).
- Scale Factor (F): 0.5, controlling the amplification of differential variations.
- Crossover Rate (Cr): 0.9, determining the probability of incorporating mutant components.
- Maximum Iterations: 3000, setting the stopping criterion.

4.2. Initialize the Population:

- Create an initial population of 100 individuals (candidate solutions).
- Each individual represents a potential solution: a set of weights [Z, W_1, W_2, W_3].
- Random values are generated for each weight between 0 and 1.
- To ensure the DEA model constraint is satisfied, the input weights are normalized so that their weighted sum, when applied to the inputs of the evaluated supplier, equals 1. Mathematically:

$$Z_1 \cdot PR_m + Z_2 \cdot LD_m + Z_3 \cdot RR_m = 1$$

  - Suppose a randomly generated weight vector is [Z1 = 0.2 , Z2 = 0.4, Z3 = 0.4]
  - The input data for supplier m is: PR_m = 250, LDm = 5, RRm = 4
  - Calculate the weighted sum: 0.2 • 250 + 0.4 • 5 + 0.4 • 4 = 53.6
  - Normalize each weight by dividing by 53.6.

$$Z1 = \frac{0.2}{53.6}, \quad Z2 = \frac{0.4}{53.6}, \quad Z3 = \frac{0.4}{53.6}$$

- These normalized weights are then used in the DEA evaluation.

4.3. Fitness Evaluation:

- Each candidate solution is evaluated by calculating:

$$E_m = W \times SQ_m$$

- Check constraints for all suppliers n:

$$W \cdot SQ_n - (Z_1 \cdot PR_n + Z_2 \cdot LD_n + Z_3 \cdot RR_n) \leq 0 \quad ; \quad \forall n=1,2,\ldots,12$$

- Count the number and degree of constraint violations.

4.4. Constraint Handling (Pareto Ranking):

DE doesn't handle constraints directly, so a Pareto ranking method is used

Comparison Rules:

- If both solutions satisfy all constraints (feasible), choose the one with the higher efficiency (objective value).
- If one solution is feasible and the other isn't, pick the feasible one.
- If both are infeasible, select the one that violates the constraints less (smaller violation).

This method ensures that feasible solutions are prioritized, while still allowing the algorithm to work with infeasible ones during optimization.

4.5. Perform Mutation: For each target vector $X_{i,G}$ (where i is the individual index and G is the generation):

- Randomly pick three distinct vectors $X_{r1,G}$, $X_{r2,G}$, $X_{r3,G}$ from the population.
- Calculate the mutant vector with $F = 0.5$.:

$$V_{i,G} = X_{r1,G} + F(X_{r2,G} - X_{r3,G})$$

4.6. Crossover (Binomial): Increase diversity mutant vector by mixing components of the mutant vector $V_{i,G}$ with the target vector $X_{i,G}$.

Let:

- $V_{i,G} = \{v_{1,i,G}, v_{2,i,G}, \ldots v_{D,i,G}\}$ be the mutant vector.
- $X_{i,G}$ be the target vector.
- Then trial vector $U_{i,G}$ is generated by:

$$u_{j,i,G} = \begin{cases} v_{j,i,G} & if \quad rand_j \leq Cr \vee j = j_{rand} \\ x_{j,i,G} & otherwise \end{cases}$$

- C_r = Crossover Probability, j_{rand} is any number from 1,2,…., D

4.7. Selection ("Survival of the fittest" principle):

- Compare the new trial vector $U_{i,G}$ against its parent (target vector) $X_{i,G}$.
- Whichever has the higher DEA-efficiency moves on to the next generation:

$$X_{i,G+1} = \begin{cases} U_{i,G} & if \ f(U_{i,G}) \leq f(X_{i,G}) \\ X_{i,G} & Otherwise \end{cases}$$

4.8. Stopping Criterion:

- For instance, The DE loop (mutation → crossover → selection) runs until **either**:
  - You hit the maximum of **3,000 generations**, or
  - The average efficiency across the population stops improving.

5. Experimental Data & DEA Model:

Suppliers (DMUs): 12

Inputs: Price (PR), Late Deliveries (LD), Reject Rate (RR)

Output: Service Quality (SQ)

- The data is detailed in this table:

Table 1 Data of 12 different suppliers [7]

Criteria	Inputs			Output
Suppliers	Price (PR)	Late deliveries (LD) %	Rate of rejected (RR) %	Service quality (SQ)
1	290	7	3	95
2	240	3	5	98
3	300	4	6	12
4	255	5	3	100
5	295	10	8	65
6	250	3	3	110
7	245	7	4	92
8	285	6	4	73
9	270	6	6	75
10	270	12	4	81
11	285	3	5	112
12	275	5	8	85

- **DEA model** of the Kth DMU will be:

$$
\begin{aligned}
&Max\, SQ_m \\
&s.t. \\
&z_1\, PR_m + z_2\, LD_m + z_3\, RR_m = 1 \\
&w\, SQ_n - (z_1\, PR_n + z_2\, LD_n + z_3\, RR_n) \leq 0 \\
&\forall n = 1, \ldots, m, \ldots 12
\end{aligned}
$$

6. Parameter Setting for DE Algorithm

- The Differential Evolution (DE) algorithm optimizes the DEA model for supplier selection.

- The following parameters are specified:

Table 2 Parameter setting for DE

Pop size (*NP*)	100
Scale factor (*F*)	0.5
Crossover rate (*Cr*)	0.9
Max iteration	3000

- The algorithm is coded in DEV C++, with uniform random numbers generated via the rand() function.
- Efficiency scores for each supplier are computed as the average over 30 runs, with termination after 3000 iterations.

6.1. Weight Selection

- Weights for the input (PR, LD, RR) and output (SQ) criteria are generated using uniform random numbers from the rand() function.

7. Results

- Results of all DMUs are given in this table:

Table 3 Average efficiency and weighted of 12 suppliers in 30 runs

Suppliers	Value of input and output weight				Efficiency
	Z[1]	Z[2]	Z[3]	W	
1	0	0	0.33337	0.00909282	0.863818
2	0.00352436	0.0225639	0.0172946	0.0102042	1
3	0.00333337	0	0	0.00757585	0.0909102
4	0	0	0.333337	0.00909102	0.909102
5	0.00338987	0	1.42974e017	0.00830173	0.539612
6	0.000230935	0.158028	0.156064	0.00909102	1
7	0	0	0.250003	0.00681827	0.62728
8	0.00350881	0	9.0241e-017	0.00797458	0.582144
9	0.00370374	1.62202e017	0	0.00841761	0.631321
10	0.00370374	0	7.15911e-018	0.00841761	0.681826
11	0.000131411	0.317989	0.0038805	0.00892868	1
12	0	0.191415	0.00536679	0.00536679	0.456178

- Results:

- Supplier 2: Efficiency = 1 (fully efficient)
- Supplier 6: Efficiency = 1
- Supplier 11: Efficiency = 1
- Supplier 3: Efficiency ≈ 0.0909 (least efficient)

- A histogram of efficiency scores for all 12 suppliers.

Fig. 1 Histogram of all suppliers with their efficiency score

- Summarization for efficiency Scores for all 12 Suppliers:

Table 4 Suppliers efficiency score

Suppliers	Efficiency	Suppliers	Efficiency
1	0.863818	7	0.62728
2	1	8	0.582144
3	0.0909102	9	0.631321
4	0.909102	10	0.681826
5	0.539612	11	1
6	1	12	0.456178

8. Discussions and Conclusion

- Suppliers 2, 6, and 11 are the most efficient, each with a score of 1, making them ideal choices.
- Supplier 3 is the least efficient (~0.0909)

- Combination of suppliers 2, 6, and 11 would be the recommended supplier set while the company needs single-item suppliers.
- Combining DEA and DE offers an effective framework for multi-criteria supplier selection.
- The approach identifies efficient supplier combinations, improving supply chain decisions.
- Numerical results confirm DE's efficacy in solving DEA-based supplier selection problems, providing a valuable tool for decision-makers.

Implementation of Differential Evolution for

Supplier Efficiency in Practice

1. Introduction

This report documents the implementation of a Differential Evolution (DE**)** algorithm to solve a Data Envelopment Analysis (DEA) problem for evaluating supplier efficiency. The goal is to optimize input and output weights (Z_1, Z_2, Z_3, W) such that the efficiency of each supplier is maximized while adhering to DEA constraints.

2. Implementation Details

2.1 Encoding, Operators, and Constraint Handling

Encoding

1- Each solution (individual) is represented as a 4-dimensional vector:

   [Z1, Z2, Z3, W]

2- Normalization constraint: Z1*PR +Z2*LD + Z3*RR =1

Operators

1- Mutation (promotes exploration by introducing diversity)
   - A mutant vector is generated using V = a + f * (b - c)
   - Where f = 0.5 and a, b, c are generated randomly
2- Crossover (binomial)
   - A trial vector is created by mixing components from the target and mutant vectors.
   - Cr = 0.9 (high crossover rate to encourage exploitation).
   - Justification: A high Cr ensures that most dimensions are inherited from the mutant, improving convergence.

3- Selection
   - The trial vector replaces the target vector if it has:
     1- Fewer constraint violations,
     2- or Higher efficiency (if no violations).

Constraint Handling

1- Penalty Function:
   - If a solution violates DEA constraints (W*SQ / (Z1*PR + Z2*LD +Z3*RR) <= 1)

     penalty (Violations) is applied.

   - Solutions with zero violations are prioritized.

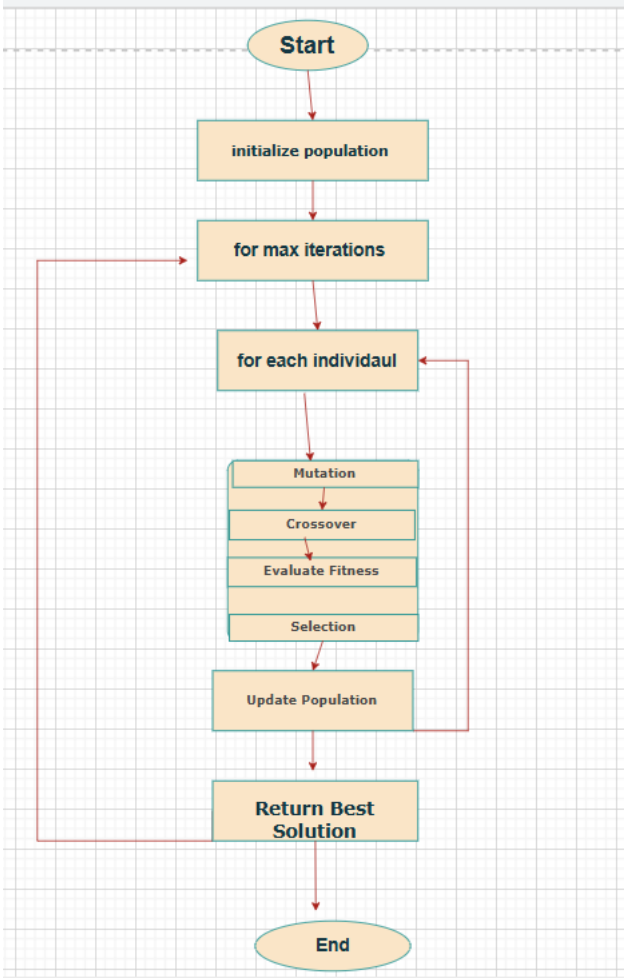
2- Justification: This ensures feasibility while temporarily allowing the algorithm to explore infeasible regions.

3. Logical Flow of the Program

The algorithm follows these steps:

- Initialization
    - Generate a population of random weight vectors (Z1, Z2, Z3, W) normalized to satisfy Z1*PR + Z2*LD + Z3*RR = 1
- Main DE Loop
    - For each individual:
        1- Mutation: Create a mutant vector.
        2- Crossover: Generate a trial vector.
        3- Fitness Evaluation: Compute efficiency and constraint violations.
        4- Selection: Retain the better solution (trial or target).
- Termination
    - After max iterations = 100 iterations, the best solution is selected.

Flow chart for Algorithm



4. Algorithm parameters

- Population Size = 100: balances exploration and computational cost.
- Scaling Factor f = 0.5: Provides moderate mutation strength.
- Crossover Rate Cr = 0.9: Encourages exploitation by inheriting most mutant traits.
- Max iteration 100: Ensures convergence without excessive runtime

5. Results

5.1 Result: Small dataset (5 suppliers)

```
Optimized weights and efficiency for all suppliers:
   Supplier        Z1        Z2        Z3         W  Efficiency
0         2  0.014412  0.074280  0.034752  0.039341    0.999988
1         4  0.002260  0.024903  0.039013  0.008177    0.999978
2         1  0.000119  0.006375  0.506189  0.015775    0.937984
3         5  0.007421  0.000000  0.000000  0.018166    0.539388
4         3  0.004646  0.002102  0.021605  0.012542    0.098249

Optimal Weights for Top Supplier:
Supplier      2.000000
Z1            0.014412
Z2            0.074280
Z3            0.034752
W             0.039341
Efficiency    0.999988
```

- Supplier #2 is more efficient

5.2 Result Medium Dataset (8 suppliers)

```
Optimized weights and efficiency for all suppliers:
   Supplier        Z1        Z2        Z3         W  Efficiency
0         6  0.004605  0.024534  0.068146  0.012993    0.999977
1         2  0.003881  0.000328  0.000000  0.008828    0.927892
2         4  0.011981  0.000000  0.148039  0.031236    0.892676
3         7  0.003001  0.000000  0.001140  0.006841    0.850658
4         1  0.000133  0.001045  0.212618  0.006079    0.844584
5         8  0.003493  0.000000  0.000000  0.007939    0.582064
6         5  0.003302  0.000000  0.000000  0.007499    0.500342
7         3  0.050331  0.000000  0.000000  0.114379    0.090902

Optimal Weights for Top Supplier:
Supplier      6.000000
Z1            0.004605
Z2            0.024534
Z3            0.068146
W             0.012993
Efficiency    0.999977
```

Supplier #6 is more efficient

5.3 Result full dataset (12 suppliers)

```
Optimized weights and efficiency for all suppliers:
    Supplier        Z1        Z2        Z3         W  Efficiency
0          6  0.004509  0.002508  0.013647  0.010687    0.999970
1         11  0.000112  0.076164  0.003671  0.002396    0.962252
2          2  0.006301  0.004447  0.000000  0.014437    0.927364
3          4  0.000280  0.000000  0.060220  0.002273    0.901844
4          7  0.010468  0.000000  0.000000  0.023791    0.853384
5          1  0.000194  0.000000  0.025929  0.001132    0.802086
6         12  0.007592  0.001614  0.000376  0.017301    0.700618
7         10  0.010937  0.000000  0.000000  0.024856    0.681776
8          9  0.004774  0.000000  0.000099  0.010852    0.631146
9          8  0.005465  0.000000  0.000000  0.012413    0.581799
10         5  0.003877  0.000000  0.000000  0.008812    0.500770
11         3  0.006917  0.000000  0.000000  0.015716    0.090884

Optimal Weights for Top Supplier:
Supplier      6.000000
Z1            0.004509
Z2            0.002508
Z3            0.013647
W             0.010687
Efficiency    0.999970
```

Supplier #6 is more efficient

📌 Fuzzification and Defuzzification Process

As part of the preprocessing steps in this project, I applied **fuzzification and defuzzification** on the price values (PR) of the suppliers to handle uncertainty and make the data more flexible for optimization. Below, I explain both steps as implemented in the code:

◆ 1. Fuzzification (Creating the Fuzzy Price Range)

To begin with, I used a **triangular membership function** to represent the fuzziness of each supplier's price. Here's how I did it:

First, I calculated the **standard deviation (σ)** of the price column across all suppliers.

Then, for each price value:

- I defined three points for the triangular function:
- a = price - σ (left end of triangle)
- b = price (peak of the triangle, where membership = 1)
- c = price + σ (right end of triangle)

After that, I randomly generated a value x within the range [a, c].

- Depending on the value of x, I calculated its **membership degree (μ)** based on its position within the triangle using the following rules:
- If a < x < b: μ = (x - a) / (b - a)
- If x == b: μ = 1
- If b < x < c: μ = (c - x) / (c - b)
- Otherwise: μ = 0
- This step allowed me to convert each crisp price value into a fuzzy representation, capturing possible variation around the original price.

◈ 2. Defuzzification (Getting a Crisp Price from the Fuzzy Set)

- After fuzzification, I needed to select a realistic crisp value to use in optimization. This was done using **α-cut-based defuzzification**:
- For each randomly chosen x from the triangle, I also generated a random α in the range [0, 1].
- If the membership value μ of x was **greater than or equal to α**, I accepted x as the new defuzzified price.
- If not, I repeated the process (up to 1000 times) until a suitable x was found.
- If no value passed the α-cut condition after 1000 tries, I simply kept the original price b as a fallback.
- This method helped add slight randomness (within reason) to the price values while keeping them close to the original range, mimicking real-world uncertainty in supplier pricing.

✓ Result

- After running this process, all price values in the 'PR' column were updated to new values that reflect both the original data and the uncertainty introduced by the fuzzification step. These updated values were then used in the optimization phase to evaluate and improve supplier efficiency and make it more realistic.

References:

1. Jauhar, S., Pant, M., & Deep, A. (2014). Differential Evolution for Supplier Selection Problem: A DEA Based Approach. *Advances in Intelligent Systems and Computing*, 343–353. https://doi.org/10.1007/978-81-322-1771-8_30
2. *Literature Review on Differential Evolution Algorithm - Journal of University of Shanghai for Science and Technology*. (2021, June 24). Journal of University of Shanghai for Science and Technology. https://jusst.org/literature-review-on-differential-evolution-algorithm/
3. Malik, A. K., Singh, A., Simranjit, & Garg, C. P. (2010). *Supply chain management- an overview*. *2*(2), 97–101. https://www.researchgate.net/publication/284727581_Supply_chain_management-_an_overview