Bol. Soc. Paran. Mat.
©SPM – E-ISSN-2175-1188
SPM: www.spm.uem.br/bspm

(3s.) **v. 2025 (43)** : 1–10.
ISSN-0037-8712
doi:10.5269/bspm.77389

Modern Non-linear Optimization Algorithm of large-Scale for Combinatorial Approximate Methods

Abbas Musleh Salman[*], Asmaa N. Al-Janabi and Ahmed Hadi Hussain

ABSTRACT: In this paper, we compare two methods, stochastic gradient descent (SGD) and gradient descent (GD), which are optimization algorithms used to minimize loss functions in machine learning. GD updates the model parameters by calculating the gradient over the entire dataset before taking a step. This ensures stable convergence but is computationally expensive. On the other hand, SGD updates the parameters after processing a single random data point, making it much faster but introducing noise. GD follows a smooth path to a minimum, while SGD takes a noisy, winding path, sometimes exceeding a local minimum but also escaping it. For large datasets, GD becomes inefficient, while SGD scales well and is typically used in deep learning. To balance stability and efficiency, both methods aim to find the optimal parameters for machine learning models, with GD focusing on accuracy and SGD on speed.

Key Words: Optimization algorithm, loss function, convergence, computational cost, optimal control, nonlinear optimization.

Contents

1 Introduction **1**

2 Some Definitions and Terms **2**
2.1 Gradient function 2
2.2 Objective function 2
2.3 Complex test function 2

3 Comparison Between Stochastic Gradient Descent (SGD) and Gradient Descent (GD) **3**
3.1 Gradient Descent (GD) 3
3.1.1 Mathematical Formulation 3
3.2 Stochastic Gradient Descent (SGD) 3
3.2.1 Mathematical Formulation of SGD 3

4 Comparison of Stochastic Gradient Descent (SGD) and Gradient Descent (GD) **4**

5 Numerical Results for Different Graphs **4**

6 Algorithms: Linear Regression with Gradient Descent (Batch and Stochastic) **6**
6.1 Summary 8

7 Conclusions **8**

1. Introduction

Regarding machine learning and model optimization, (SGD) and (GD) are two of the most popular optimization algorithms. These methods are used to minimize the error function and improve the performance of the model by updating the parameters in an efficient manner. Thus, (GD) and (SGD) are two fundamental optimization algorithms. (GD) computes the gradient using the entire dataset before updating the parameters. This makes it stable but computationally expensive, especially for large datasets. However, after processing a single random data point, (SGD) updates the model [1,2,3,4,5]. This makes SGD much faster but also introduces randomness, resulting in noisier updates. While GD is more stable

[*] Corresponding aauthor

Submitted June 19, 2025. Published August 12, 2025
2010 *Mathematics Subject Classification:* 30D05, 11M06.

and converges smoothly, SGD can escape local minima and is widely used in deep learning. To balance speed and stability, a regularization called Mini-Batch Gradient Descent processes small batches of data at a time [6,7,8,9,10]. Both GD and SGD play a crucial role in training machine learning models, ensuring that they learn the optimal parameters efficiently [11,12,13,14,15]. Active learning and prioritizing which data points to classify next. Data selection and identifying the most useful labeled examples. Self-training and pseudo-classification and evaluating the quality of pseudo-classifications [16,17,18,19].Given the extensive applications of modern nonlinear optimization algorithms in a wide range of fields, such as engineering and communications, we aim to cover everything that will provide valuable insights to decision makers.

2. Some Definitions and Terms

2.1. Gradient function

If we define the model's loss function as L and the labeled data set as $D = \{(x_i, y_i)\}$ the definition gradient function can be expressed as:

$$G = (x_i, y_i) = \frac{\partial L}{\partial (x_i, y_i)}. \tag{2.1}$$

This gradient indicates how sensitive the loss function is to changes in a particular labeled example (x_i, y_i). If the gradient is large, it means the example is highly influential in defining the model's decision boundary.

2.2. Objective function

If $f(x)$ represents the objective function, the goal of an optimization algorithm is to find the optimal parameters x^* that either **Minimize** the function:

$$x^* = \arg\min_x f(x). \tag{2.2}$$

Or Maximize the function:

$$x^* = \arg\max_x f(x). \tag{2.3}$$

In machine learning, the objective function is often a loss function (for supervised learning) or a reward function (for reinforcement learning).for example Linear Regression

- Objective: Minimize Mean Squared Error (MSE),
- Function:

$$f(\theta) = \frac{1}{n}\sum_{i=1}^{n} (y_i - \hat{y}_i)^2 . \tag{2.4}$$

And Logistic Regression / Neural Networks

- Objective: Minimize Binary Cross-Entropy
- Function:

$$f(\theta) = -\frac{1}{n}\sum_{i=1}^{n} [y_i \log \hat{y}_i + (1 - y_i) \log (1 - \hat{y}_i)] . \tag{2.5}$$

2.3. Complex test function

A complex test function for optimization is a mathematical function with multiple local minima / maxima, making it challenging for optimization algorithms to find the global optimum. These functions are often used to evaluate optimization techniques, such as gradient-based methods, genetic algorithms, or swarm intelligence [20,21]. For Example of the Levy function is a well-known complex test function with multiple local minima, making it a good benchmark for global optimization algorithms

$$\begin{aligned} f(x,y) = &\sin^2(\pi w_1) + (w_1-1)^2\left[1+10\sin^2(\pi w_1+1)\right] \\ &+ (w_2-1)^2\left[1+10\sin^2(2\pi w_2)\right]. \end{aligned} \tag{2.6}$$

Where $w_1 = 1 + \frac{x_i - 1}{4} \quad, i = 1, 2$.

3. Comparison Between Stochastic Gradient Descent (SGD) and Gradient Descent (GD)

Both Stochastic Gradient Descent (SGD) and Gradient Descent (GD) are popular optimization algorithms in machine learning and deep learning that seek to minimize a given objective function (such as a loss function). However, they differ in how they update the model parameters [10]. For example, GD computes the gradient over the entire dataset at each step, while SGD updates parameters using a single randomly selected data point at each step.

3.1. Gradient Descent (GD)

In order to find an ideal collection of parameters that yields the lowest possible value of the loss function, gradient descent is an iterative process that modifies a set of parameters (i.e., features of a dataset). Gradient descent computes the gradient of the loss function across the whole dataset at each step, updating the model parameters.

3.1.1. Mathematical Formulation. If the cost function $J(\theta)$ is differentiable, GD updates parameters as follows:

$$\theta_{t+1} = \theta_t - \alpha \nabla J(\theta_t). \tag{3.1}$$

Where α is the rate of learning, $\nabla J(\theta_t)$ is the cost function's gradient in relation to θ. and θ = model parameters.

Theorem 3.1 ***Convergence of Gradient Descent for Convex Functions:***
Let $J(\theta)$ be a convex, continuously differentiable function with a Lipschitz continuous gradient, meaning there exists a constant $L > 0$ such that:

$$\|\nabla J(\theta_1) - \nabla J(\theta_2)\| \le L\|\theta_1 - \theta_2\|, \forall \theta_1, \theta_2. \tag{3.2}$$

If we choose a fixed learning rate $0 < \alpha \le \frac{2}{L}$, then Gradient Descent satisfies:

$$J(\theta_1) - J(\theta^*) \le \frac{C}{t}, \tag{3.3}$$

for some constant C, meaning Gradient Descent converges at a rate of $O\left(\frac{1}{t}\right)$ to the optimal solution θ^*

3.2. Stochastic Gradient Descent (SGD)

An optimization technique called stochastic gradient descent (SGD) is used to minimize functions, especially in deep learning and machine learning. It is an efficient variation of Gradient Descent (GD), designed to handle large datasets.

3.2.1. Mathematical Formulation of SGD. As an alternative to calculating the gradient across the whole dataset, SGD updates parameters using a single random data point (x_i, y_i). Given a cost function $\mathrm{J}(\theta)$ that depends on a dataset $\mathrm{D} = \{(x_i, y_i)\}_{\mathrm{i=1}}^{\mathrm{N}}$, the update rule for SGD is given by:

$$\theta_{t+1} = \theta_t - \alpha \nabla J_i(\theta_t). \tag{3.4}$$

Where

- $\alpha \nabla J_i(\theta_t)$ is the cost function's gradient calculated using a single randomly selected sample (x_i, y_i).

- θ_t are the parameters at iteration t.
- α is the rate of learning.

Theorem 3.2 ***Convergence of Gradient Descent for Convex Functions:***
Let $J(\theta)$ be a convex and differentiable function with a unique minimum at θ^*. If the learning rate sequence α_t satisfies:

$$\sum_{t=1}^{\infty} \alpha_t = \infty, \sum_{t=1}^{\infty} \alpha_t^2 = \infty, \tag{3.5}$$

then the sequence of updates θ_t generated by Stochastic Gradient Descent (SGD) converges in expectation to the global minimum θ^* :

$$\mathbb{E}\left[J\left(\theta_t\right)\right] \rightarrow J\left(\theta^*\right) \text{ as } t \rightarrow \infty. \tag{3.6}$$

A common choice for the learning rate schedule is $\alpha_{\mathrm{t}} = \frac{1}{\mathrm{t}}$, which satisfies the theorem. In deep learning, adaptive learning rate methods such as Adam, RMS prop, and Momentum are often used to improve convergence speed and stability. Therefore, the SGD convergence theorem provides theoretical guarantees for the effectiveness of the algorithm under appropriate conditions [11]. It explains why SGD performs well in large-scale optimization problems and why tuning the learning rate schedule is critical to achieving stable convergence.

4. Comparison of Stochastic Gradient Descent (SGD) and Gradient Descent (GD)

Use GD when the dataset is small and requires stable convergence. Use SGD when the dataset is large and requires faster training. We observe that the GD (gradient descent) method moves smoothly towards the minimum at $x = -2.5$. We observe that the SGD (stochastic gradient descent) method moves more randomly due to random noise, but is still able to converge. Note the Figure 1. Where:

1. GD is smooth and stable but slower.
2. SGD fluctuates due to randomness but can converge faster.
3. SGD can escape local minima, which is useful in deep learning.

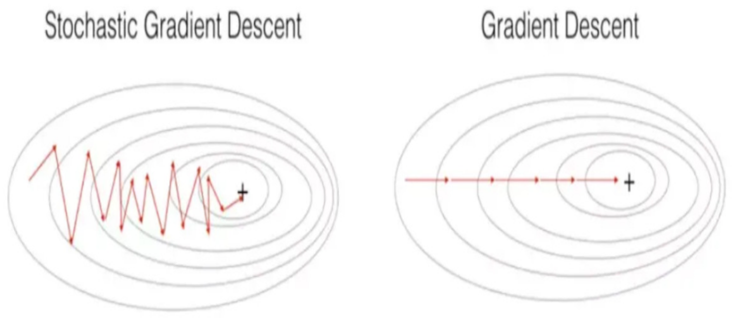

Figure 1: Gradient descent (GD) and stochastic gradient descent (SGD).

5. Numerical Results for Different Graphs

Analysis of numerical results using Python The graphs with the largest number of repetitions were tested using both methods. Three graphs containing 20 repetitions were solved, as shown in Figure 2. We noticed through the comparison between the two methods: When using 20 iterations, it was found that the Gradient Descent (GD) method is more stable but slower when dealing with large data, so it is suitable for small data. However, with the same number of iterations, it was found that the best method is Stochastic Gradient Descent (SGD), which is faster but has more fluctuations, which makes it ideal for deep learning and is used for large data and reaching the optimal solution, as shown in table 1.

Table 1: Optimization Comparison (20 Iterations) between (GD) and (SGD)

Iteration	Gradient Descent	Stochastic GD
0	10.0000	10.0000
1	8.5671	8.5636
2	7.8860	7.8736
3	6.8405	6.8180
4	4.8882	4.8525
5	3.2434	3.2413
6	3.5387	3.5326
7	3.5598	3.5531
8	3.5586	3.5532
9	3.5587	3.5385
10	3.5587	3.5496
11	3.5587	3.5519
12	3.5587	3.5702
13	3.5587	3.5687
14	3.5587	3.5592
15	3.5587	3.5537
16	3.5587	3.5660
17	3.5587	3.5617
18	3.5587	3.5426
19	3.5587	3.5565
20	3.5587	3.5565

The final results of the stepwise descent method after 20 iterations, where the optimal solution is reached slowly. Where the final X: 3.5587, the final cost: 4.0418 While the final results of the stepwise Stochastic GD method after 20 iterations, where the optimal solution is reached faster than the first. Where the Final X: 3.5565, Final Cost: 4.0419. Note the table 1.

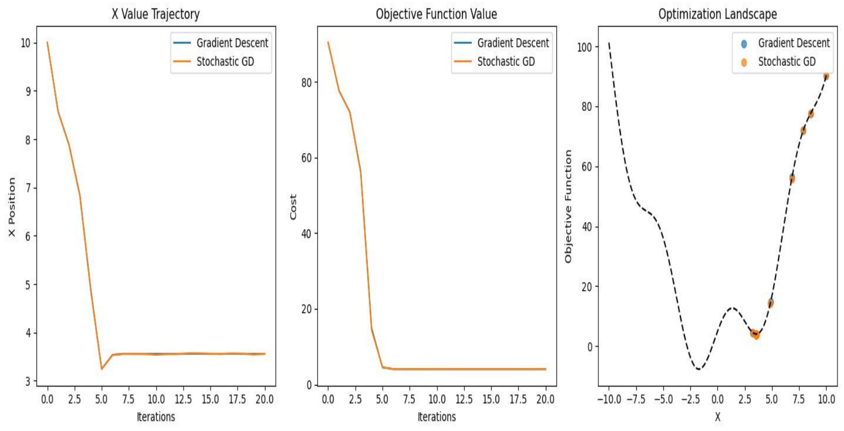

Figure 2: Test results between Gradient Descent (GD) and Stochastic Gradient Descent (SGD).

Also in the Figure 4 we notice the convergence between the two methods (GD) and (SGD) using

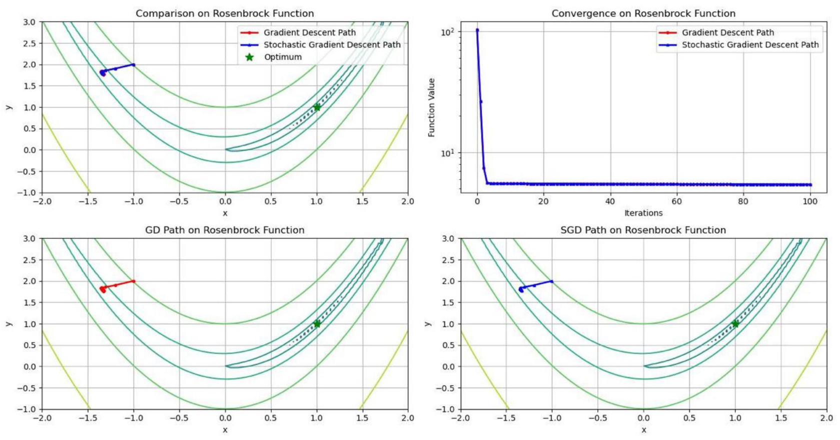

Figure 3: Comparison between (GD) and (SGD)

Python programming and as also shown in the diagram below, Figure 5 and Algorithm 1 and Algorithm 2 we found that the Gradient Descent method approaches the optimal solution faster and with 20 iterations.

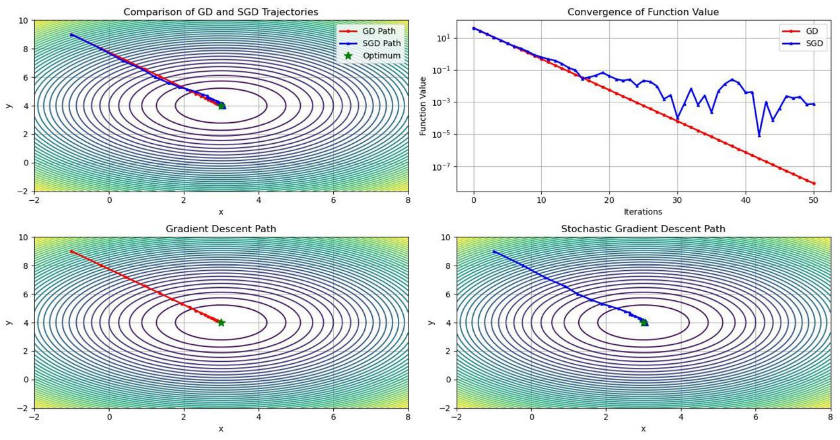

Figure 4: Convergence between (GD) path and(SGD) path

6. Algorithms: Linear Regression with Gradient Descent (Batch and Stochastic)

The algorithm begins by generating a synthetic dataset for a simple linear regression problem 100 data samples are created, each with 2 features. Features are randomly distributed between -1 and 1. The target variable y is generated using a linear formula:

$$y = 3x_1 + 2x_2 + noise \tag{6.1}$$

Noise is added using a random normal distribution to simulate real-world data variation. As shown in the algorithms below and in the diagram Figure 5

The stochastic gradient descent algorithm is a variation of traditional gradient descent, but instead of using the entire dataset to compute the gradient, the stochastic gradient descent algorithm uses only one training example at a time. As in Algorithm 2

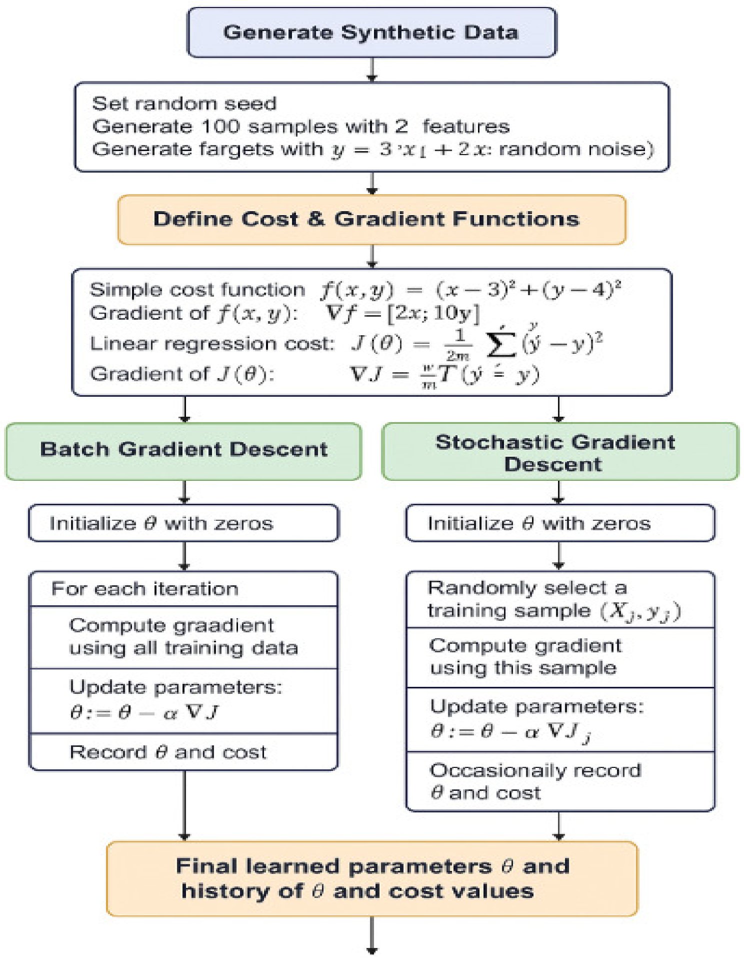
Generate Synthetic Data
Set random seed
Generate 100 samples with 2 features
Generate fargets with $y = 3 \cdot x_1 + 2x$: random noise)
Define Cost & Gradient Functions
Simple cost function $f(x,y) = (x-3)^2 + (y-4)^2$
Gradient of $f(x,y)$: $\nabla f = [2x; 10y]$
Linear regression cost: $J(\theta) = \frac{1}{2m} \sum (\acute{y} - y)^2$
Gradient of $J(\theta)$: $\nabla J = \frac{w}{m} T (\acute{y} - y)$
Batch Gradient Descent
Initialize θ with zeros
For each iteration
Compute graadient using all training data
Update parameters: $\theta := \theta - \alpha \nabla J$
Record θ and cost
Stochastic Gradient Descent
Initialize θ with zeros
Randomly select a training sample (X_j, y_j)
Compute gradient using this sample
Update parameters: $\theta := \theta - \alpha \nabla J_j$
Occasionally record θ and cost
Final learned parameters θ and history of θ and cost values

Figure 5: Algorithm flow: Training a linear regression model using gradient descent

Algorithm 1: Batch Gradient Descent

```
1. Initialize parameters $\theta$ with zeros
2. For each iteration do:
     a. Compute gradient of the cost function using all training data
     b. Update parameters using: $\theta := \theta - \alpha \cdot \nabla J$
     c. Record current $\theta$ and cost for analysis
3. Return final $\theta$ and the history of updates
```

Algorithm 2: Stochastic Gradient Descent

```
1. Initialize parameters $\theta$ with zeros
2. For each iteration do:
     For each training example do:
         a. Randomly select a single sample $(x_j, y_j)$
         b. Compute gradient using this sample
         c. Update parameters: $\theta := \theta - \alpha \cdot \nabla J_j$
     Occasionally (every 10 samples), record current $\theta$ and cost
3. Return final $\theta$ and the history of updates
```

6.1. Summary

Table 2 provides a summary of the comparison between gradient descent (GD) and stochastic gradient descent

Table 2: Summary of the comparison between gradient descent (GD) and stochastic gradient descent (SGD)

Feature	Gradient Descent (GD)	Stochastic Gradient Descent (SGD)
Update Frequency	After processing the entire dataset	After each data point
Computation Cost	High (slower for large datasets)	Low (faster updates)
Convergence Path	Smooth and stable	Noisy and zigzagged
Accuracy	More precise updates	Less precise due to randomness
Scalability	Inefficient for big data	Scales well with large datasets
Local Minima Escape	May get stuck in local minima	Can escape local minima due to noise
Best Use Case	Small/medium datasets	Large datasets, deep learning
Deterministic?	Yes (same updates each time)	No (randomness affects updates)
Memory Usage	High (stores full dataset)	Low (processes one data point at a time)
Example Algorithm	Batch Gradient Descent	Online Learning, Deep Learning

7. Conclusions

Through the practical and applied study and using the Python program, we noticed when comparing the two methods, we concluded that GD is more stable but slower when dealing with big data. Therefore, it is suitable for small data. However, the better method is Stochastic Gradient Descent (SGD), which is faster but has more fluctuations, making it ideal for deep learning and used for big data. To reach a compromise between the two methods, Mini-Batch Gradient Descent can be used, where the coefficients are updated after each small data point instead of a single or complete data point.

References

1. El-yahyaoui, Y., Lafhim, L., *Optimization problems with nonconvex multiobjective generalized Nash equilibrium problem constraints*, Communications in Combinatorics and Optimization, (2024). DOI: 10.22049/cco.2024.29435.1993.

2. Pandey, R., Pandey, Y., Singh, V., *Robust optimality and duality for bilevel optimization problems under uncertain data*, Communications in Combinatorics and Optimization, (2025). DOI: 10.22049/cco.2025.29950.2236.

3. Kheirfam, B., Nasrollahi, A., *A second-order corrector wide neighborhood infeasible interior-point method for linear optimization based on a specific kernel function*, Communications in Combinatorics and Optimization, (2021). DOI: 10.22049/CCO.2021.27044.1185.

4. Niu, F., Recht, B., Christopher, R., Wright, S. J., *Hogwild!: A Lock-Free Approach to Parallelizing Stochastic Gradient Descent*, (2011), 1-22.

5. Alridha, A. H., Salman, A. M., Mousa, E. A., *Numerical optimization software for solving stochastic optimal control*, J. Interdiscip. Math. 26, 889-895, (2023). DOI: 10.47974/JIM-1525.

6. Hasan, A. H., Salman, A. M., Al-Jilawi, A. S., *The Applications of NP-hardness optimizations problem*, J. Phys. Conf. Ser. 1818, 012179, (2021). DOI: 10.1088/1742-6596/1818/1/012179.

7. Ahmed, A., Al-Jilawi, A. S., *Mathematical programming computational for solving NP-hardness problem*, J. Phys. Conf. Ser. 1818, 012137, (2021). DOI: 10.1088/1742-6596/1818/1/012137.

8. Salman, A. M., Alridha, A., Hussain, A. H., *Some topics on convex optimization*, J. Phys. Conf. Ser. 1818, 012171, (2021). DOI: 10.1088/1742-6596/1818/1/012171.

9. Hasan, A., Salman, A. M., *Numerical Optimization Approach for Solving Production Planning Problem Using Python language*, 03, (2022). ISSN: 2660-5309.

10. Ahmed, A., Salman, A. M., *Exploring optimization algorithms for challenging multidimensional optimization problems: A comparative approach*, AIP Conf. Proc. 3097, 080025, (2024).

11. Salman, A. M., Al-Jilawi, A. S., *Solving nonlinear optimization problem using approximation methods*, (2022). DOI: 10.53730/ijhs.v6nS3.5699.

12. Salman, A. M., Al-Jilawi, A. S., *Applications of maximum independent set*, AIP Conf. Proc. 2398, 060015, (2022).

13. Salman, A. M., Al-Jilawi, A. S., *Combinatorial Optimization and Nonlinear Optimization*, J. Phys. Conf. Ser. 1818, 012134, (2021). DOI: 10.1088/1742-6596/1818/1/012134.

14. Hasan, A., Alsharify, F., Al-Khafaji, Z., *A Review of Optimization Techniques: Applications and Comparative Analysis*, Iraqi Journal for Computer Science and Mathematics 5, 122-134, (2024). DOI: 10.52866/ijcsm.2024.05.02.011.

15. Sulaiman, H. K., Hassan, A. M., Hammood, H. K., Alridha, A. H., Al-Khafaji, Z., *An evaluation of the reliability optimization problem for the electromagnetic system of an airplane using a comparison of PSO and GA*, AIP Conf. Proc. 3264, 050010, (2025). DOI: 10.1063/5.0259100.

16. Al-Jilawi, A. S., Alsharify, F. H., *Review of Mathematical Modelling Techniques with Applications in Biosciences*, Iraqi Journal for Computer Science and Mathematics 3, 135-144, (2022). DOI: 10.52866/ijcsm.2022.01.01.015.

17. Alridha, A. H., Alsharify, F. H., Al-Khafaji, Z., *Maximizing Reliability in the Age of Complexity: A Novel Optimization Approach*, Baghdad Science Journal 22, (2025). DOI: 10.21123/bsj.2024.9894.

18. Hiba A. Ahmed, A., Hasan, A. H.,*Mathematical Optimization Strategies for address Production Mixing Challenges*, Iraqi Journal of Science 66, 2446-2454, (2025). DOI: 10.24996/ijs.2025.66.6.21.

19. Alridha, A., Al-Jilawi, A. S., *K-cluster combinatorial optimization problems as NP-Hardness problems in graph clustering*, AIP Conf. Proc. 2398, 060034, (2022).

20. Kadhim, M. K., Wahbi, F. A., Alridha, A. H., *Mathematical optimization modeling for estimating the incidence of clinical diseases*, International Journal of Nonlinear Analysis and Applications 13, 185-195, (2022). DOI: 10.22075/ijnaa.2022.5470.

21. Ahmed, A., Wahbi, F. A., Kadhim, M. K., *Training analysis of optimization models in machine learning*, International Journal of Nonlinear Analysis and Applications 12, 1453-1461, (2021). DOI: 10.22075/ijnaa.2021.5261.

Abbas Musleh Salman,

Department of Mathematics,

Ministry of Education, General Directorate of Education in Babylon,

Iraq.

E-mail address: abbas.mmm2019@gmail.com

and

Asmaa N. Al-Janabi,

Department of Computer Engineering,

College of Engineering, Al-Nahrain University, Baghdad,

Iraq.

E-mail address: asm_no_2006@ced.nahrainuniv.edu.iq

and

Ahmed Hadi Hussain,
Department of Air conditioning and Refrigeration Engineering,
University of Babylon, College of Engineering Al-Musayab,
Iraq.
E-mail address: met.ahmed.hadi@uobabylon.edu.iq