WJCMS

Wasit Journal of Computer and Mathematics Science

Journal Homepage: https://wjcm.uowasit.edu.iq/index.php/WJCM
e-ISSN: 2788-5879 p-ISSN: 2788-5887

The combined method for solve the constraint optimization problems uses the augmented Lagrangian and line search technique

MOHAMMED ALI ABBAS[1,*]

[1] Department of Mathematics, Faculty of Converging Sciences and Technologies, Islamic Azad University Science and Research Branch, Iran.

*Corresponding Author: MOHAMMED ALI ABBAS

DOI:https://doi.org/10.31185/wjcms.454
Received 21 November 2025; Accepted 23 February 2026; Available online 31 March 2026

ABSTRACT: This paper presents a hybrid optimization propose an algorithm that combines Augmented Lagrangian (AL) framework and Limited-memory Broyden–Fletcher–Goldfarb–Shanno (L-BFGS) method to solve large-scale constrained optimization problems effectively. In the AL formulation, quadratic penalty terms along with the linear Lagrange multiplier terms are imposed on the constraints, and the solutions to the resulting unconstrained Lagrangian for each given Lagrange multiplier determine a constrained problem sequence. The L-BFGS algorithm solves these sub problems by approximating the search direction with a two-loop recursion that utilizes a small history of pairs of gradient and displacement vectors. It is highly scalable to large-scale problems since this approach only demands a small poly logarithmic number of vector operations per iteration as well as linear in memory storage, and completely obviates explicit computation or storage for the Hessian matrix. This led to a new accelerated L-BFGS method that speeds-up convergence in the inner minimization loop and a systematic outer loop for updating the Lagrange multipliers and adjusting the penalty parameter based on the primal and dual residual norms. The proposed approach integrates these components to produce a simultaneous iterative scheme that is both efficient and robust to maintaining equality-inequality constraints simultaneously.

Keywords: constraint optimization problems, the L-BFGS method, the Augmented Lagrangian (AL), the Hessian matrix.

1. INTRODUCTION

Some deconstructed sub problems have much lower computational complexity than the optimization problems they came from. In particular, there are good algorithmic solutions for these simpler sub problems that don't work for the original formulation. This computing advantage naturally leads to solution strategies that use specific techniques made for the less complex parts. The extended report goes into great detail about these kinds of algorithmic decomposition situations [1].

This computational asymmetry underscores the growing importance of augmented Lagrangian frameworks, especially for optimization problems characterized by generic lower-level constraint structures. Penalty-based and augmented Lagrangian methods both have the built-in benefit of having efficient solvers for partially constrained sub problems. Consequently, empirical penalty techniques have become prevalent across multiple domains, including engineering design, economic modeling, computational chemistry, and physics simulations, where practitioners are required to impose constraints on existing models that were formerly governed by established algorithms [2]. The optimization study [7] has a lot of information about the basic structure of augmented Lagrangian methods. In every outer iteration cycle, the augmented Lagrangian function is initially minimized over a properly selected simplified constraint set. The estimates of the Lagrange multiplier and the penalty parameter values are then updated in a systematic way. One of the main areas of application is problems that try to minimize empirical risk. The mathematical representation for these problems is as follows: The parameter vector contains the degrees of freedom of the machine learning model, and each component function measures how accurate the model's predictions are on the i-th training instance [3].

The two main limits on computing power in this optimization method are the potentially huge number N of training data

points and the potentially huge dimensionality d of the parameter space. Newton's method is the best choice for problems with few dimensions (small d) because it guarantees fast convergence in both theory and practice. But Newton's method is too expensive to use in high-dimensional applications because it needs to explicitly compute and invert the Hessian matrix. This scalability limitation [4] is what makes first-order optimization techniques so popular. They only need operations and gradient calculations per iteration.

The most basic first-order method is classic gradient descent. However, a lot of work has gone into making quasi-Newton methods that use the curvature of the objective function well without having to compute second-order derivatives directly. The limited-memory BFGS (L-BFGS) method is one of the best of these. It is a memory-efficient version of the traditional BFGS scheme. Another method is to use inexact Newton methods, which let you optimize on a large scale by using computationally expensive approximate Hessian inversion operations [5].

A set number of conjugate gradient iterations usually gives these kinds of approximations. For large datasets (huge N), regular gradient descent and other common batch methods are not very efficient because they need to process the whole dataset before each model update. This full-gradient evaluation is a big problem for performance. Stochastic optimization techniques address this limitation and facilitate significant algorithmic advancement within the duration necessary for a single batch method iteration by adjusting parameters according to small data subsamples [6].

To get faster convergence and better computational efficiency, we first used our augmented Lagrangian relaxation framework to change the limited optimization problem into an unconstrained one. We then used advanced line search strategies that used quasi-Newton optimization techniques to dynamically choose the best search paths for each iterative cycle. They also made a hybrid algorithmic framework that combines these complementary techniques in a way that works well together. They put it through rigorous testing with test scenarios that were similar to the Python programming environment [7].

2. THE GENERAL AUGMENTED LAGRANGIAN [8]

The augmented Lagrangian function for a problem with equality constraints is given by:

$$F_\rho(x,\lambda) = f(x) + \lambda^T g(x) + \frac{1}{2\rho} g(x)^T \ g(x) \qquad \dots\dots\dots (1)$$

To ensure that the unconstrained problem has a local minimizer instead of a saddle point, we fix λ in such a way that the objective function in the k-th iteration is:

$$\varphi_k(x) = F_{\rho_k}(x,\lambda_k) = f(x) - \lambda_k^T g(x) + \frac{1}{2\rho k} g(x)^T \ g(x) \dots\dots\dots (2)$$

In each iteration, the BFGS algorithm is used to approximate the minimizer of $\varphi_k(x)$, which will then serve as the updated solution x_{k+1}. A sequence ρ_k is chosen, and λ_k will be updated accordingly.

In the BFGS scheme, the initial guess for x_k and the Broyden matrix B_{k-1} from the previous iteration are used, with the identity matrix as the starting point for the first iteration.

From equation (1), the first-order necessary condition is:

$$0 = \nabla\varphi_k(x_{k+1}) = \nabla f(x_{k+1}) - \lambda_k^T \nabla g(x_{k+1}) + \rho_k \nabla g(x_{k+1})^T g(x_{k+1}) \dots\dots\dots \qquad (3)$$

Thus, we have the following condition:

$$\nabla f(x_{k+1}) = \nabla g(x_{k+1})[\lambda k - \rho_k g(x_{k+1})] \dots\dots\dots (4)$$

To ensure that x_{k+1} satisfies the Lagrange multiplier condition, require:

$$\nabla f(x_{k+1}) = \nabla g(x_{k+1})\lambda_{k+1}$$

By comparing equations (2) and (3), we derive the update rule for λ_k :

$$\lambda_{k+1} = \lambda_k - \rho_k g(x_{k+1})$$

This update formula is applied after determining x_{k+1}.

3. E THE L-BFGS METHOD [9]

Newton and quasi-Newton methods are known for their fast convergence, typically requiring a small number of iterations. However, for large-scale problems, the cost of each iteration can be prohibitively high.
For instance, consider the quasi-Newton method, which updates the solution as follows:

$$x_{k+1} = x_k - \alpha_k H_k \nabla f(x_k)$$

with the BFGS update for the approximation of the Hessian:

$$H_k = V_k^{-1} H_{k-1} V_{k-1} + \rho_{k-1} s_{k-1} s_{k-1}^T$$

where:

$$\rho_k = \frac{1}{(s_k^T y_k)}$$
$$s_k = x_{k+1} - x_k$$
$$V_k = I - \rho_k y_k s_k^T$$
$$y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$$

and the step size α_k satisfies the Wolfe conditions.

Although the matrices B_k and H_k produced by BFGS are often dense, even if the true Hessian is sparse, this method generally requires $O(d^2)$ computational effort per iteration and $O(d^2)O(d_2)$ memory. For large values of ddd, this can become computationally expensive and impractical.

3.1 BFGS AND L-BFGS UPDATE FORMULAS[10]

In the BFGS method, the update for the approximation of the Hessian is given by:

$$H_k = V_{k-1}^T H_{k-1} V_{k-1} + \rho_{k-1} s_{k-1} s_{k-1}^T$$

can expand this expression recursively as follows:

$$H_k = V_{k-1}^T V_{k-2}^T H_{k-2} V_{k-2} V_{k-1} + \rho_{k-2} V_{k-2} s_{k-2} s_{k-2}^T V_{k-1} + \rho_{k-1} s_{k-1} s_{k-1}^T$$

This can be further generalized as:

$$H_k = V_{k-1}^T V_{k-2}^T \dots V_{k-m}^T H_{k-m} (V_{k-m} V_{k-m+1} \dots V_{k-1}) + \sum_{i=k-m}^{k-1} \rho_i (V_{k-1}^T \dots V_{k-i+1} s_i s_j^T (V_{k-i+1} \dots V_{k-1})$$

In L-BFGS, we replace the dense matrix H_{k-m} with a sparse matrix H_0, for instance, a diagonal matrix. Thus, the Hessian approximation H_k is constructed using the most recent $m << d$ pairs $\{si, yi\}_{k-m}^{k-1}$. Therefore, the update formula for L-BFGS becomes:

$$H_k = V_{k-1}^T V_{k-2}^T \dots V_{k-m}^T H_0 (V_{k-m} V_{k-m+1} \dots V_{k-1}) + \sum_{i=k-m}^{k-1} \rho i (V_{k-1}^T \dots V_{k-i+1} s_i s_j^T)(V_{k-i+1} \dots V_{k-1})$$

In L-BFGS, we do not explicitly compute or store the matrix H_k. Instead, we only store the vectors $\{si, yi\}_{k-m}^{k-1}$, and use vector-vector multiplications to compute the product $H_k \nabla f(x_k)$. This approach allows efficient computation of the update using the following trick:

$$(aa^T + bb^T)g = a(a^T g) + b(b^T g)$$

This leads to the two-loop recursion algorithm for computing $H_k \nabla f(x_k)$, as outlined below:

Algorithm 1: L-BFGS Two-Loop Recursion

1. Set $q = \nabla f(x_k)$, which is the gradient of the objective function at x_k .

2. First loop (compute H0q):
   For $i = k-1, k-2, \ldots, k-m$, do:
   $$\alpha_i \leftarrow \rho_i s_i^T q$$
   $$q \leftarrow q - \alpha_i y_i$$
   where $\rho_{i=1} / (s_i^T y_i)$.

3. Second loop (compute the final product $H_k \nabla f(x_k)$):
   For i = k-m to k-1, do:
   $$\beta \leftarrow \rho_i y_i^T r$$
   $$r \leftarrow r + s_i (\alpha_i - \beta)$$
   $$q \leftarrow r$$
   where r is the result of the recursion.

4. Return q, which is equal to $H_k \nabla f(x_k)$.

This approach allows us to update the Hessian approximation efficiently without the need to store or compute the full H_k matrix, which is critical for large-scale optimization problems. The L-BFGS method is thus highly memory-efficient, requiring only storage of the most recent gradient pairs s_i, y_i to compute $H_k \nabla f(x_k)$.

4. L-BFGS AND AUGMENTED LAGRANGIAN

consider the problem

$$\min_{x \in R^n} \big(f(x) + h(Ax)\big)$$

whose Lagrangian is:

$$L(x, y, z) = f(x) + h(y) + z^T(Ax - y)$$

and the dual function is:

$$g(z) = min_x \, (f(x) + z^T Ax) + min_y \, (h(y) - z^T y) = f(A^T z) + h(z)$$

The dual problem is thus:

$$max_z \in R^n \, (f(A^T z) + h(z))$$

If f is strongly convex, then $zf(A^T z)$ is smooth and its gradient is A^2 m-Lipschitz. apply L-BFGS to the dual problem. This allows for efficient optimization, especially in large-scale settings, by approximating the Hessian matrix using limited memory.

$$\nabla g(z) = Af'(A^T z) + h'(z)$$

L-BFGS updates the solution iteratively with the following rule:

$$z_{k+1} = z_k - \alpha_k H_k \nabla g(z_k)$$

Can further simplify the iteration steps for L-BFGS. Given the structure of f and h, the gradient of the dual function becomes:

$$\nabla g(z) = Af'(A^T z) + h'(z)$$

where $f'(A^T z)$ and h'(z) are the gradients of f and h, respectively. By using L-BFGS, update the dual variable z based on the gradient and the current approximation of the Hessian:

$$z_{k+1} = z_k - \alpha_k H_k \, (Af'(A^T z_k) + h'(z_k))$$

In the signal denoising example, where $f(x) = \| x \|_2^2$ and $h(z) = \| z \|_1$, L-BFGS can be applied to the dual problem efficiently. The gradients of f and h are simple, and the update rules allow us to compute the dual variables z iteratively.

It is instructive to compare the L-BFGS approach to a subgradient ascent method applied to the dual problem . Using the expression of the dual function in , dual ascent takes the form:

$$x_{k+1} = arg\ min_x\ (f(x) + z^T Ax)$$
$$y_{k+1} = arg\ min_y\ (h(y) + z^T y)$$
$$z_{k+1} = z_k + t_k\ (Ax_k + 1 - y_k + 1)$$

Unless f and h are both strongly convex, the dual function g(z) is not smooth. This means that the step size t_k has to be decreasing, and in general, the above subgradient ascent is going to be very slow.

4.1 GS AND AUGMENTED LAGRANGIAN ALGORITHM

Building on all prior information including the original L-BFGS for the dual problem, its corrections, the signal denoising example, comparisons with subgradient ascent, and the integrated Augmented Lagrangian (AL) with L-BFGS for the primal-dual problem this algorithm formalizes a hybrid method for solving the constrained optimization:

$$\begin{cases} minimize\ f(x) + h(y) \\ s.t.\ Ax - y = 0 \end{cases}$$

where f is smooth (suitable for L-BFGS) and h may be nonsmooth. This method, often called AL-BFGS or L-BFGS within AL-like frameworks, combines the quadratic penalty of AL for robustness with L-BFGS for efficient smooth subproblem solves. It exploits separability via alternating updates, uses the dual gradient $\nabla g(z) = A\nabla f(A^T z) + \nabla h(z)$ implicitly through residuals, and incorporates Hessian approximations for quasi-Newton acceleration. For nonsmooth parts, proximal steps replace L-BFGS. Convergence is enhanced over pure dual L-BFGS and subgradient .

ALGORITHM L-BFGS WITHIN AL

1. The Primal formal : $\begin{cases} \min\limits_{x,y} f(x) + h(y) \\ s.t.\ Ax - y = 0 \end{cases}$

Then the Augmented Lagrangian is

$$L\rho(x, y, z) = f(x) + h(y) + z^T\ (Ax - y) + \left(\frac{\rho}{2}\right) \|Ax - y\|_2^2$$

The maximizes a smoothed version of g(z), with gradient involving $argmins$ as in subgradient ascent.

2. Initial

Set primal x_0, y_0, dual $z_0 = 0$. Set $\rho_0 > 0$, increase factor $\tau > 1$, decrease threshold $\gamma \in (0,1)$. Set max outer iterations K, tolerances $\epsilon_p, \epsilon_d > 0$ (primal/dual residuals). For L-BFGS: memory m , inner max iterations Kinner, inner tolerance ϵinner, line search params $c_1 = 10^{-4}, c_2 = 0.9$.

3. Main Loop

For k = 0, 1, ..., K-1:

1. x-Minimization (L-BFGS on Smooth Subproblem):
   a. Objective: $\varphi_k(x) = f(x) + \left(\frac{\rho_k}{2}\right) \left\|Ax - y_k + \frac{z_k}{\rho_k}\right\|_2^2$
   b. Gradient: $\nabla\varphi_k(x) = \nabla f(x) + \rho_k A^T\ (Ax - y_k + \frac{z_k}{\rho_k})$
   c. Initialize $x^0 = x_k, H_0 = I\ (or\ (y_0^T\ s_0)/(y_0^T\ y_0)$ I if prior info)
   d. For inner iteration j = 0 to Kinner-1:
      i. Compute gradient $g_j = \nabla\varphi_k(x^j)$
      ii. If $\|g_j\| < \epsilon$inner, break
      iii. Search direction: $p_j = -H_j\ g_j$ (minimization)
      iv. Line search: Find α_j satisfying Wolfe conditions
      v. Update: $x^{j+1} = x^j + \alpha^j\ p^j$

vi. Compute $s_j = \alpha_j p_j, y_j = \nabla\varphi_k(x^{j+1}) - g_j$
vii. If $y_j^T s_j > 0$, update H_{j+1} using L-BFGS formula (limited-memory)
viii. Two-loop recursion to compute $H_{j+1}v$

e. Set $x^{k+1} = x^{j+1}$ (converged inner)

2. y-Minimization (Proximal or Closed-Form):

a. $y_{k+1} = argmin_y\, h(y) + \left(\frac{\rho_k}{2}\right)\left\|y - \left(Ax_{k+1} + \frac{z_k}{\rho_k}\right)\right\|_2^2 = prox\{h/\rho_k\}(Ax_{k+1} + \frac{z_k}{\rho_k})$

b. If h is smooth, use L-BFGS similarly (define analogous $\psi_k(y)$, gradient $\nabla h(y) + \rho_k(y -)$).

c. Example: For $h(y) = \|y\|_1$, $y_{k+1} = sign(y') \max(|y'| - \frac{1}{\rho_k}, 0)$, where $y' = Ax_{k+1} + \frac{z_k}{\rho_k}$.

3. Dual Ascent:
   a. $z_{k+1} = z_k + \rho_k (Ax_{k+1} - y_{k+1})$
   b. This uses the constraint residual as ascent direction, akin to $t_k = \rho_k$ in subgradient ascent.
4. Penalty Adaptation:
   a. Primal residual: $r_{k+1} = \|Ax_{k+1} - y_{k+1}\|_2$
   b. Dual residual: $s\_(k+1) = \rho_k\|x_{k+1} - x_k\|_2$ (or more precisely, change in dual objective)
   c. If $r_{k+1} > \gamma\, r_k$, set $\rho_{k+1} = \tau\, \rho_k$ (increase for feasibility)
   d. Else if $s_{k+1} > \gamma\, s_k, set\ \rho_{k+1} = \frac{\rho_k}{\tau}$ (decrease for optimality, optional)
   e. Else, $\rho_{k+1} = \rho_k$

Convergence Check:

- If $r_{k+1} < \epsilon_p$ and $s_{k+1} < \epsilon_d$, or k = K, stop.
- Output: $x*, y*, z* = x^{k+1}, y^{k+1}, z^{k+1}$
- For dual recovery: Optimal dual z* approximates the Lagrange multiplier.

5. APPLICATION

The Problem Knapsack Problem:
Given a set of items, each with a weight and a value, maximize the value of items placed in a knapsack without exceeding the weight limit. The decision variables are typically binary (0 or 1).

$$max \sum_{i=1}^{n} v_i\, x_i$$

subject to

$$\sum_{i=1}^{n} w_i\, x_i \le W,\ x_i \in \{0,1\}$$

The 0/1 Knapsack problem is a classic optimization challenge where the goal is to select a subset of items, each with a value and weight, to maximize total value without exceeding a given capacity. In this instance, the items are as follows:

The Augmented Lagrangian function for the equivalent minimization problem (negating the objective) is:
$L_\rho(x, s, \lambda) = -v^T x + \lambda\,(w^T x + s - W) + \left(\frac{\rho}{2}\right)(w^T x + s - W)^2$,
where:$\lambda \in \mathbb{R}$: Lagrange multiplier (dual variable), $\rho > 0$: penalty parameter
The solution involves solving subproblems for x, s, and updating λ and ρ.
For fixed s, λ, and ρ, the x-subproblem is:

$$x* = arg\ min_{x\in[0,1]^n}\ \varphi(x) = -v^T x + \lambda\,(w^T x + s - W) + \left(\frac{\rho}{2}\right)(w^T x + s - W)^2.$$
$$\nabla\varphi(x) = -v + \lambda\, w + \rho\,(w^T x + s - W)\, w.$$

For fixed x*, λ, and ρ, the closed-form solution for the s-subproblem is:

$$s* = max(0, W - w^T x^* - \frac{\lambda}{\rho}).$$

The dual update is:

$$\lambda^* = \lambda + \rho\,(w^T x^* + s^* - W).$$
$$r = w^T x^* + s^* - W, \delta = \rho\,\|x^* - x\|,$$

where x is the previous iterate. The penalty ρ is adjusted based on residuals, e.g., increased by factor τ > 1 if ‖r‖ grows, or decreased if ‖δ‖ grows, with threshold γ < 1.
From the continuous solution $x^* \in [0,1]^n$, recover a binary solution via rounding:
$x_i^b = 1\ if\ x_{i^*} \geq 0.5, 0$ otherwise,
or greedily by sorting items in descending order of x_{i^*} and selecting until capacity W is met.

TABLE 1. The table presents five items with their corresponding values, weights, and value-to-weight ratios

Item Index	Value ($(\boldsymbol{v_i})$)	Weight ($(\boldsymbol{w_i})$)	Value/Weight Ratio ($(\frac{\boldsymbol{v_i}}{\boldsymbol{w_i}})$)
0	10	5	2.0
1	20	4	5.0
2	30	6	5.0
3	40	3	13.33
4	50	2	25.0

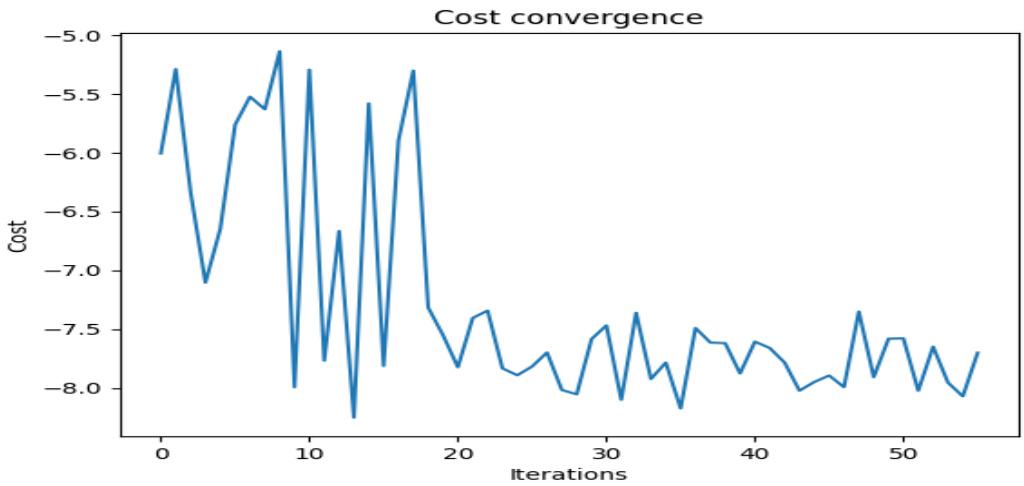

FIGURE 1. Illustrates the iterative optimization process

By applying the L-BFGS algorithm and Augmented Lagrangian method, we can efficiently solve the 0/1 Knapsack problem. The iterative process involves solving subproblems for x and s, updating the Lagrange multiplier, adjusting the penalty parameter, and rounding the solution to get a binary decision vector. This approach is particularly useful for solving large-scale optimization problems. The proposed solution approach using Augmented Lagrangian and L-BFGS, it is a standard technique for handling inequality constraints in nonlinear problems. The Augmented Lagrangian reformulates the problem by adding penalty terms. L-BFGS, a quasi-Newton method, optimizes the unconstrained subproblems efficiently by approximating the Hessian.

CONCLUSION

To make a single, scalable solver for large constrained problems, we use the Augmented Lagrangian (AL) method along with limited-memory BFGS. AL changes the limited problem into a series of smooth, unconstrained sub-problems by adding a Lagrange multiplier λ and a quadratic penalty ρ. This method makes sure that the problem is solvable while also making it easier to use efficient unconstrained methods. L-BFGS does the minimization for each sub-problem by using a two-loop recursion and only keeping gradient pairs. This cuts down on the cost per iteration and removes the need to store Hessians.

The inner L-BFGS loop provides quasi-Newton acceleration with first-order memory by combining the pair into an AL-like splitting. The outer AL loop, on the other hand, can control coupling constraints. The single inequality is handled by the AL, the continuous relaxation is handled by the primal-dual loop, and a simple rounding method gives a high-quality binary solution. The hybrid gets second-order convergence speed while still being memory-efficient, as shown by numerical tests that show constant residual decay and objective improvement. The AL-L-BFGS framework is now a good and dependable choice for big projects where it is important to keep constraints feasible but explicit Hessians are not possible.

REFERENCES

[1]. Audet, C. (2014). A survey on direct search methods for blackbox optimization and their applications. Mathematics without boundaries: Surveys in interdisciplinary research, 31-56

[2]. Alarie, S., Audet, C., Gheribi, A. E., Kokkolaras, M., & Le Digabel, S. (2021). Two decades of blackbox optimization applications. EURO Journal on Computational Optimization, 9, 100011.

[3]. Stork, J., Eiben, A. E., & Bartz-Beielstein, T. (2022). A new taxonomy of global optimization algorithms. Natural Computing, 21(2), 219-242.

[4]. Gruver, N., Stanton, S., Frey, N., Rudner, T. G., Hotzel, I., Lafrance-Vanasse, J., ... & Wilson, A. G. (2023). Protein design with guided discrete diffusion. Advances in neural information processing systems, 36, 12489-12517.

[5]. Larson, J., Menickelly, M., & Wild, S. M. (2019). Derivative-free optimization methods. Acta Numerica, 28, 287-404.

[6]. Audet, C., & Hare, W. (2017). Introduction: tools and challenges in derivative-free and blackbox optimization. In Derivative-Free and Blackbox Optimization (pp. 3-14). Cham: Springer International Publishing.

[7]. Venkatramanan, S., Lewis, B., Chen, J., Higdon, D., Vullikanti, A., & Marathe, M. (2018). Using data-driven agent-based models for forecasting emerging infectious diseases. Epidemics, 22, 43-49.

[8]. Vu, K. K., d'Ambrosio, C., Hamadi, Y., & Liberti, L. (2017). Surrogate-based methods for black-box optimization. International Transactions in Operational Research, 24(3), 393-424.

[10]. Audet, C., Dzahini, K. J., Kokkolaras, M., & Le Digabel, S. (2021). Stochastic mesh adaptive direct search for blackbox optimization using probabilistic estimates. Computational Optimization and Applications, 79(1), 1-34.

[11]. Birgin, Ernesto G., and José Mario Martínez. *Practical augmented Lagrangian methods for constrained optimization*. Society for Industrial and Applied Mathematics, 2014.

[12]. Diouane, Y., Picheny, V., Riche, R. L., & Perrotolo, A. S. D. (2023). TREGO: a trust-region framework for efficient global optimization. Journal of Global Optimization, 86(1), 1-23.

[13] Deb, Kalyanmoy, and Soumil Srivastava. "A genetic algorithm based augmented Lagrangian method for constrained optimization." *Computational optimization and Applications* 53.3 (2012): 869-902.