How to Cite:
Aljawad, R. A., & Al-Jilawi, A. S. (2022). Solving multiobjective functions of dynamics optimization based on constraint and unconstraint non-linear programming. *International Journal of Health Sciences*, *6*(S1), 5236–5248. https://doi.org/10.53730/ijhs.v6nS1.6041

Solving multiobjective functions of dynamics optimization based on constraint and unconstraint non-linear programming

Rabab Abdulsattar Aljawad
Ministry of Education, General Directorate of Education in Karbala
Email: ababaljwad@gmail.com

Dr. Ahmed Sabah Al-Jilawi
College of Education for Pure Sciences, Department of Mathematics, University of Babylon, Iraq.
Email: ahmed.aljelawy@uobabylon.edu.iq

***Abstract*---**To better understand the connection between nonlinear optimization problems and differential equations, this study uses the mathematical models with one or more objective functions that are constrained by restrictions and therefore take the form of differential equations called dynamical optimization systems, which mean a decision-making method that uses differential and algebraic equation mathematical models to design-wise policies based on forecasts of future events. This paper focuses on solving mathematical models systems using the Python programming language.

***Keywords*---**Dynamic Optimization (DO), Mathematical model, Optimization techniques, and Python language.

Introduction

With differential and algebraic equation mathematical models, decision-makers may develop intelligent policies based on forecasts of future outcomes using a technique called "dynamic optimization." This form of analysis may use a wide variety of tools and methodologies.

Numerous real-world optimization problems have dynamic search spaces due to the objective function, the number of variables, and the presence of constraints [1]. Optimizing problems that change over time and must be handled online using an optimization approach are called dynamic optimization problems (DOPs) [2]. It is not enough for algorithms to discover excellent solutions when solving DOPs; they must also be capable of adapting to changing conditions to swiftly find a new key when the prior one is no longer good enough [3]. Therefore, an appropriate

benchmark generator is critical for thoroughly evaluating the efficacy of algorithms created for DOPs. Thus, the following vital qualities should be included in a DOP benchmark generator [1], [2]:

- Simple to execute and evaluate; Information on the location of the global optimum should be made available to scientists doing research and fitness value, as well as the features of the landscape's components, such as their evolving interaction structure, form, and intensity of change, among other things. It is recommended that this information be not used in the construction or tuning of algorithms and that the problem be treated as algorithms are hidden in a black box; It enables algorithmic researchers to explore and evaluate the behavior and performance of algorithms in a controlled environment. Also, a good benchmark should be easy to use [15].
- Flexibility: The benchmark generator should have a great deal of adjustability regarding the number of components, component form, size, change frequency, and degree of change. Additionally, each feature's setting should be self-contained. This independence enables researchers to examine the influence of each attribute on algorithm behavior. Additionally, the ability to generate examples of challenges of increasing complexity is essential for a solid benchmark. Numerous real-world situations are complicated and intricate [1], [2], and [4].
- Variety: The benchmark generator should generate problems instances with a range of characteristics, including modularity (completely separable, totally non-separable, and partly separable components), heterogeneity, balanced to very high unbalanced sub-functions, dimensionality ranges from low to high (largescale), and varying degrees of irregularity.

Numerous DOP benchmarks for various DOP domains have been suggested in the literature [2], [5], including combinatorial [6], [7], continuous [4], [8], multi-objective [9]- [11], and restricted DOPs [12] - [14]. This article discusses dynamic continuous unconstrained, and restricted optimization problems, including single- and multiobjective. The most extensively widely used and renowned benchmark generators in this subject are based on the notion of a landscape made up of multiple components. The width, height, and location of these components change over time in most published DOP benchmarks [5].

Optimization

Optimization has evolved into a flexible technique with industry and information technology applications to the social sciences. Methods for addressing optimization issues are many, and they constitute a vast pool of problem-solving technology. There are many different approaches that it is impossible to fully use [28], [29]. They are specified in several technical languages and implemented in various software programs. Many are never implemented. It is difficult to determine which one is best for a specific situation, and there are far too few opportunities to mix strategies with complementary qualities. The ideal situation would be to combine various techniques under one roof so that they and their combinations could all be used to address an issue. As it turns out, many of them have a similar problem-solving style on some level [16], [25], and [26]

Differential Equations

Differential equations may be used to model practically any system that is changing. They permeate all fields of study, including science and engineering, economics, sociology, biology, business, and healthcare services. Various mathematicians have studied the nature of these equations for hundreds of years, and there are numerous well-developed solution methodologies. However, the systems represented by differential equations are so sophisticated, or the scenarios described by them are so huge that a solution to the equations using just analytical methods is impracticable. In these complex systems, computer simulations and numerical techniques are beneficial. Before the advent of programmable computers, numerical approximation methods for solving differential equations were developed. Throughout World War II, in rooms filled with workers (typically women), mechanical calculators were often used to solve systems of differential equations numerically for military calculations. Before the advent of programmable computers, analog computers were often used to investigate mechanical, thermal, or chemical systems. As the speed and affordability of programmable computers have grown, it is possible to solve more complex differential equations on a typical personal computer using simple programs built for this purpose. Today, the computer on your desk can tackle issues that were previously unreachable to even the best supercomputers just five or ten years ago [17], [31].

Nonlinear programming

Problems involving nonlinear programming (NLP) have a nonlinear objective function or constraints [18]. No efficient approaches exist for tackling the nonlinear programming issue in its entirety. When it comes to solving problems with ten variables and hundreds of variables, even the most straightforward situations can be challenging. As a result, numerous techniques to solve the general nonlinear programming issue have been developed, which entails some degree of compromise [19].

Dynamic Optimization

Numerous optimization issues encountered in the real world are dynamic [20]. Additional jobs must should be included the timetable, the fundamental constituent composition may vary, and new orders may be received, contributing to the vehicle routing complexity. Due to the unpredictability inherent in dynamic issues, fixing them is often more difficult than solving their static counterparts. Additionally, solutions to dynamic issues must be discovered in real time as new information is received. Even the optimization objective shifts from finding the best solution to a static issue to monitoring the dynamic problem's shifting optimum in real time.

Dynamics Optimization Modeling:

The following is a representation of the dynamic optimization model:

$$minimize \qquad J(x, y, p)$$

$$subject\ to\ \ f\left(\frac{dx}{dt}, x, y, p\right) = 0$$
$$0 \leq g\left(\frac{dx}{dt}, x, y, p\right)$$

We note that $J(x, y, p)$ is the objective function of a dynamic optimization system; where p input values, y output values, and x are the system.

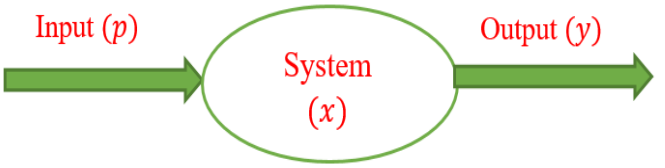

Figure (1): describe the input, output, and system

f, g are constraints of a dynamic optimization system where these constraints are either equality (f) or inequality (g), and the constraints contain differential equations $(\frac{dx}{dt})$

Dynamic optimization benchmarking

The development of benchmark problems is a key subject in the field of dynamic optimization, fitness landscapes that are varied and dynamic may be represented using this method. The literature has a large number for dynamic optimization in the form of test issues. Grefenstette [23] characterizes his dynamic environment as an assembly of components. The Natural Sciences and Engineering Research Council of Canada provided funding for this work. Each part is made of a single time-varying Gaussian peak in n dimensions. Three time-varying characteristics define each height: center, amplitude, and breadth. Brake [22] proposes a "Moving Peaks Function" in a similar paper. The author devises for the first time, a multimodal function with adjustable peak height, breadth, and center was introduced. In addition, he thoroughly analyzes the literature on standards for dynamic landscapes. Bear in mind, although the majority of these concerns involve real space, they are not all applicable to all combinatorial problems.

Basic concept of dynamic optimization

Python is a programming language used to aid with dynamic simulation, estimation, and control solutions. Numerous tools for modeling and optimizing dynamic systems are available. Additionally, there are other things to consider while picking the best instrument for a specific activity. For example, many factors should be considered when choosing an optimization tool for dynamic scalings, such as size (scaling with number equations and degrees of freedom), speed (use in real-time estimation or control), adaptability, extensibility, and support (commercial vs. open-source community) (ability to embed on a system or communicate with a particular platform).

*GEKKO combines machine learning and optimization strategies to solve complex mixed-integer and differential algebraic problems. It is used with solvers for large-

scale linear, quadratic, nonlinear, and mixed-integer programming (LP, QP, NLP, MILP, MINLP). All types of operation are available, including parametric regression, data reconciliation, real-time optimization, dynamic simulation, and nonlinear predictive control. GEKKO is a Python object-oriented library for executing AP Monitor on a local machine [26].

Algorithms of Dynamic Systems

Any computer procedure that receives an input value or set of values and outputs an output value or set of values is a formula. Thus, an algorithm is a sequence of computations that alter the data state. Additionally, an algorithm may be seen as a tool to solve a well-defined computing issue. The issue statement provides a comprehensive overview of the intended input/output relationship. The algorithm defines the computational technique for creating the suitable input/output connection. For instance, we may need to ascend a series of integers. This is a common occurrence in reality and gives an excellent opportunity to introduce a variety of conventional design methodologies and analytical tools [21] and [30].

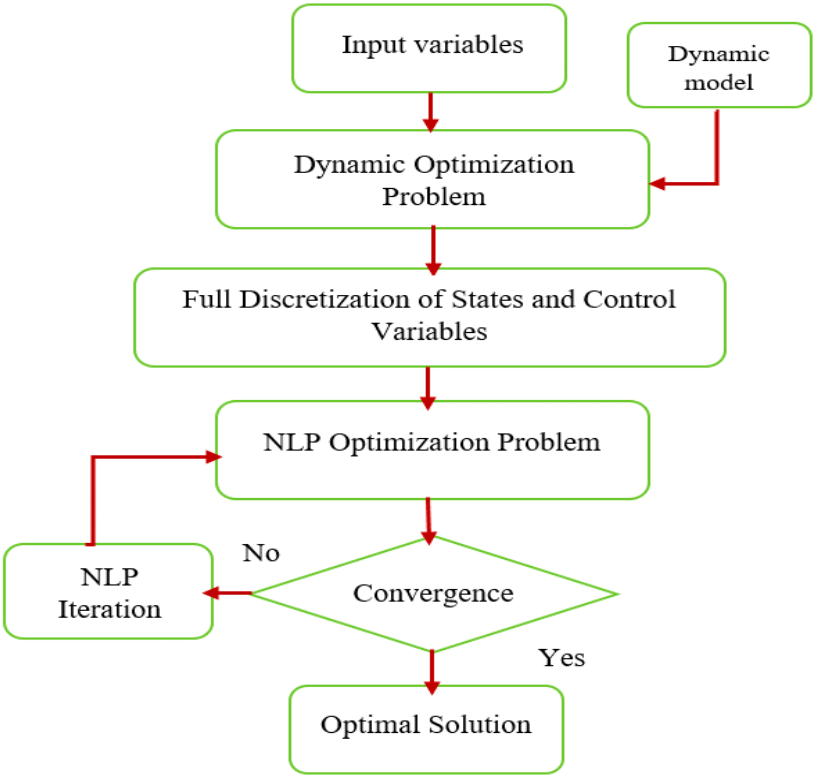

Figure (2): Dynamic Optimization Algorithm

Optimal control benchmarking

Case 1: This is a standard chemical dynamic optimization problem with an analytical solution, a mathematical system with no constraints. The state has an

analytical and global optimum solution due to two state variables. As an example, consider the following mathematical model:

$$\min_{u(t)} x_2(t_f)$$

subject to

$$\frac{dx_1}{dt} = u$$
$$\frac{dx_2}{dt} = x_1^2 + u^2$$
$$x(0) = [2 \quad 0]^T$$
$$t_f = 1$$

To solve this dynamic optimization system, we use GEKKO python code to find the optimal solution by minimizing the final state of the nonlinear system with an unconstraint dynamic such that the value of $u = -0.5$.

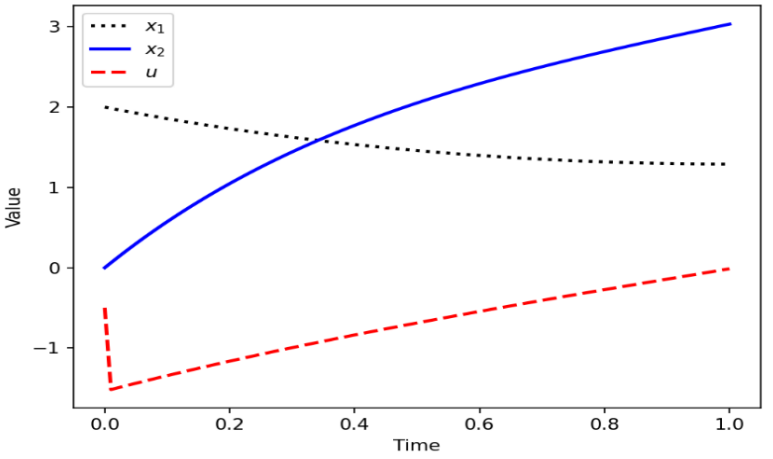

Figure (3): Dynamic optimization created the control and state profiles

Case 2: It's a typical chemical dynamic optimization issue with an analytical solution, a dynamic mathematical system. Including four state variables results in an analytical and global optimal solution for the state. For example, have a look at this mathematical formula:

$$\min_{u(t)} x_4(t_f)$$

subject to

$$\frac{dx_1}{dt} = x_2$$
$$\frac{dx_2}{dt} = -x_3 u + 16t - 8$$
$$\frac{dx_3}{dt} = u$$
$$\frac{dx_4}{dt} = x_1^2 + x_2^2 + 0.005(x_2 + 16t - 8 - 0.1x_3 u^2)^2$$
$$x(0) = [0 \quad -1 \quad -\sqrt{5} \quad 1]^T$$
$$-3 \le u \le 11$$

$$t_f = 1$$

$x_1(t)$ to $x_4(t)$ are state vectors, while $u(t)$ is the control vector.
For solving nonlinear systems that have the constraint to minimize the final state, we use code GEKKO of Python and get:

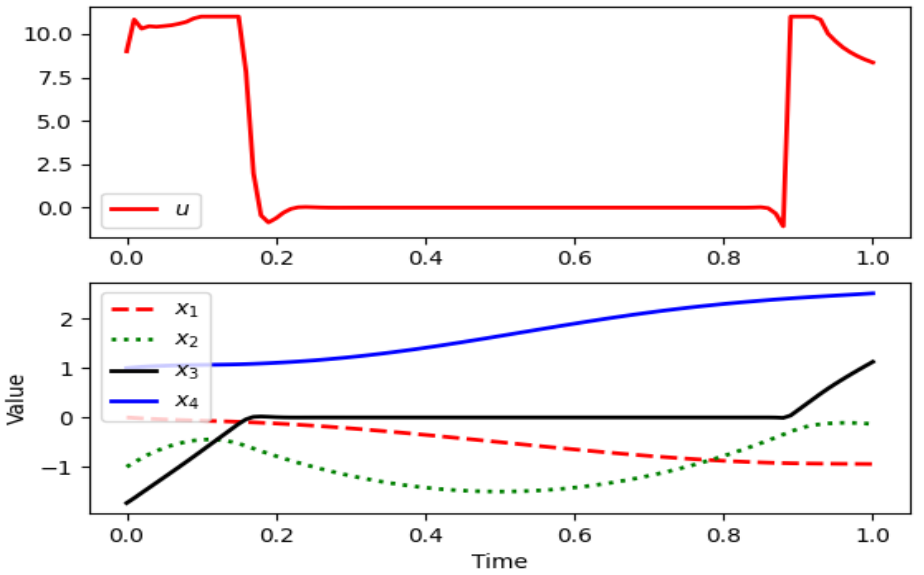

Figure (4): The control and state profiles were developed using dynamic optimization

Multiobjective Functions

We will look at optimization problems with a single objective function in this part, both with and without constraints. Cost reduction, efficiency optimization, and weight reduction are examples of conventional single-variable goal functions. Two or more objective functions must be optimized concurrently in multiobjective optimization situations. For example, the criteria may have cost reduction and efficiency maximization throughout the product's manufacturing process. The formal formulation of a multiobjective optimization problem is [24], [25], and [26]

$$\begin{aligned} \text{Minimize Or Maximize} \quad & f_k(\mathrm{x}) \qquad k = 1,2,\dots,K \\ \textit{subject to:} \quad & g_i(\mathrm{x}) \le 0 \quad i = 1,2,\dots,m < n \\ & h_j(\mathrm{x}) = 0 \qquad j = 1,2,\dots,r < n \\ & \mathrm{x}_l \le \mathrm{x} \le \mathrm{x}_u \end{aligned}$$

x is a vector that contains the n design variables defined by

$$x = \begin{bmatrix} x_1 \\ x_2 \\ . \\ . \\ . \\ x_n \end{bmatrix}$$

A solution to a multiobjective issue generates a set of Pareto optimum locations in the objective function space. Figure (5) illustrates a typical Pareto optimum front for a problem with multiple objectives and two objective functions (efficiency maximization and cost reduction). The x-axis of this picture includes the first goal (f1) function, "efficiency," and the second objective (f2) function, "cost," lies on the y-axis. The principle of dominance is used to produce the Pareto optimum front. Each answer is compared in this idea to see if it dominates another solution. For example, suppose the following requirements are met. A solution x^1 is considered to be superior to another x^2.

- x^1 is not worse than x^2 in all objectives.
- x^1 is superior to x^2 in a minimum of one goal.

Consider points A and C in terms of dominance. As expected, point C is superior to point A in both objective functions. However, at least one of the points on the Pareto optimum front is dominated by point C. Non-dominated solutions are the spots along the Pareto optimum front. For example, the Pareto optimum front is convex in Figure (5). This front, in contrast, may be concave, somewhat convex (concave), or discontinuous. Alternatively, the Pareto front's form is determined by the trade-off between the goal functions [24].

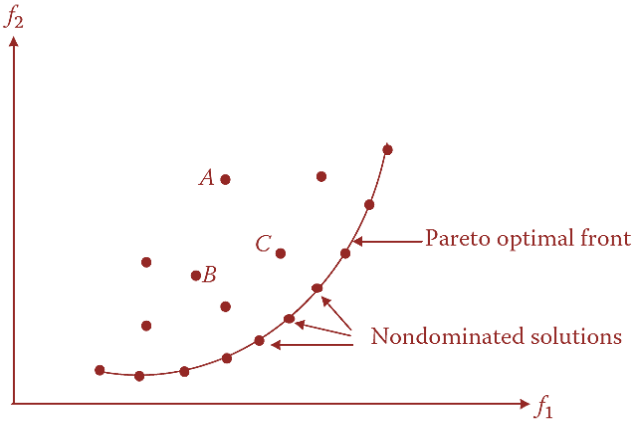

Figure (5): Pareto optimal front

Solving multiple objective function of dynamic optimization

Numerous optimization issues include conflicting goals. These conflicting aims are a component of the trade-off that characterizes the best solution. Occasionally, these opposing aims have distinct priorities, with one being accomplished before the other is considered. This is particularly true in model predictive control and other forms of dynamic optimization. With an ordered

hierarchy, there are competing aims. If extra degrees of freedom are available, the highest-level goals are met first, followed by lower-level objectives. The L1-norm objective is a natural approach to ranking goals explicitly and optimizing many priorities concurrently using a single optimization problem. For example, consider the limits or purposes associated with safety, the environment, and economics. Which are the most critical and why?

Case 3: Create a probable optimum trajectory for the following multiobjective optimization problem.

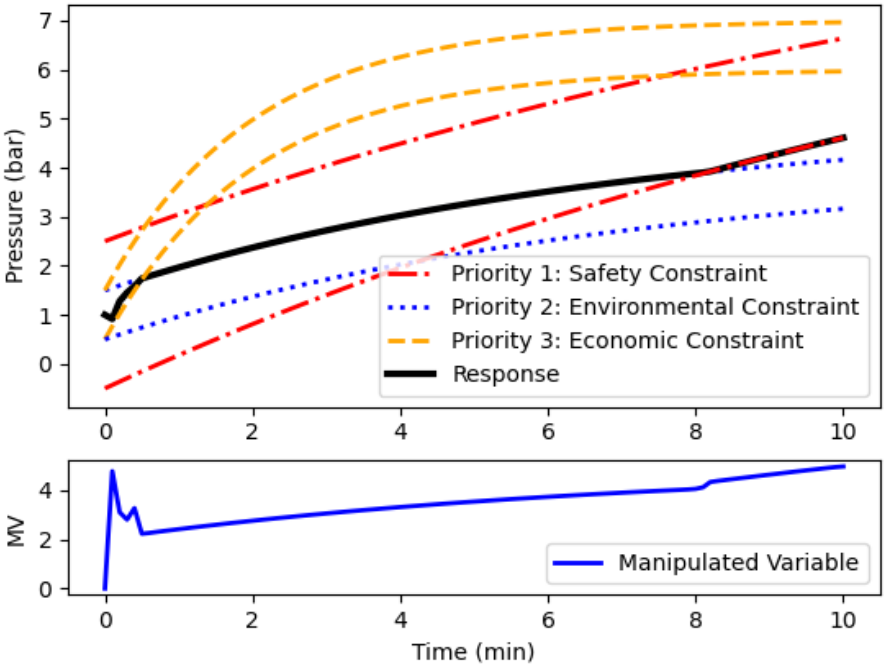

Figure (6): Describe the safety, environmental and economic constraints

Discussion of the Results

Each of the instances discussed in this study may demonstrate the final results after they have been numerically solved using Python scripts, particularly the library GEKKO. They can explain the final results of each case with a simple demonstration.

Case1: With a two-variable solution to the dynamic issue, we'll also minimize by altering the value of u, which we'll do by changing $x_1$and x_2. x_1 is equal to 2, and x_2 is equal to 0; therefore, we're going to compute the value until x_1 is equal to 2, and we'll do the same thing for x_2 until it is equal to 0, as you can see from our input or variables used to manipulate the X conditions, note Figure (3). Then we get:

Number of Iterations: 3

Table (1)
Final Results when Solve Case 1

	(scaled)	(unscaled)
Objective	3.0347753804326336e+00	3.0347753804326336e+00
Dual infeasibility	8.8817841970012523e-16	8.8817841970012523e-16
Constraint violation	8.8817841970012523e-16	8.8817841970012523e-16
Complementarity	0.0000000000000000e+00	0.0000000000000000e+00
Overall NLP error	8.8817841970012523e-16	8.8817841970012523e-16

The final value of the objective function is 3.03477538043263.

Case 2:
With the four-variable solution to the dynamic issue, we'll also minimize by altering the value of u, which we'll do by changing x_1, x_2, x_3 and x_4. x_1 is equal to 2, x_2= -1, x_3= $\sqrt{3}$ and x_4 is equal to 1; therefore, we're going to compute the value of x's, as you can see from our input or variables being used to manipulate the X conditions, see figure (4).

Number of Iterations: 19

Table (2):
Final Results when Solve Case 2

	(scaled)	**(unscaled)**
Objective	2.5121002922896420e+00	2.5121002922896420e+00
Dual infeasibility	6.2326704832482986e-07	6.2326704832482986e-07
Constraint violation	4.4408920985006262e-15	4.4408920985006262e-15
Complementarity	1.4447122070015801e-10	1.4447122070015801e-10
Overall NLP error	6.2326704832482986e-07	6.2326704832482986e-07

The final value of the objective function is 2.51210029228964.

Case 3: Figure (6) shows how to solve a multiobjective optimization problem in model predictive control or dynamic optimization. In many cases, we might have high-priority items like safety constraints. For example, priority two might be something like an environmental constraint or regulation, and priority three might be to make the process as profitable as possible.
When using python code to solve case 3, we get:

Number of Iterations: 28

Table (2)
Final Results when Solve Case 3

	(scaled)	(unscaled)
Objective	1.2277417296878346e+04	3.6832251890635038e+04
Dual infeasibility	7.1054273576010019e-13	2.1316282072803006e-12
Constraint violation	9.4368957093138306e-16	9.4368957093138306e-16
Complementarity	1.1738233691638086e-11	3.5214701074914258e-11
Overall NLP error	1.1738233691638086e-11	3.5214701074914258e-11

The final value of the objective function is 36832.2518906350.

Conclusion

This paper presented a technique for tackling single- and multiobjective, continuous, restricted, and unconstrained dynamic optimization problems. Please remember that many real-world problems may be optimized and solved using Python to arrive at the ideal answer.

References

[1] T. T. Nguyen, "Continuous dynamic optimization using evolutionary algorithms," Ph.D. dissertation, University of Birmingham, 2011.

[2] T. T. Nguyen, S. Yang, and J. Branke, "Evolutionary dynamic optimization: A survey of state of the art," Swarm and Evolutionary Computation, vol. 6, pp. 1–24, 2012.

[3] D. Yazdani, "Particle swarm optimization for dynamically changing environments with particular focus on scalability and switching cost," Ph.D. dissertation, Liverpool John Moores University, Liverpool, UK, 2018.

[4] C. Li, T. T. Nguyen, S. Zeng, M. Yang, and M. Wu, "An open framework for constructing continuous optimization problems," IEEE Transactions on Cybernetics, pp. 1–15, 2018.

[5] C. Cruz, J. R. Gonz'alez, and D. A. Pelta, "Optimization in dynamic environments: a survey on problems, methods, and measures," Soft Computing, vol. 15, no. 7, pp. 1427–1448, 2011.

[6] S. Yang, Y. Jiang, and T. T. Nguyen, "Metaheuristics for dynamic combinatorial optimization problems," IMA Journal of Management Mathematics, vol. 24, no. 4, pp. 451–480, 2013.

[7] M. Mavrovouniotis, F. M. Muller, and S. Yang, "Ant colony optimization with local search for dynamic traveling salesman problems," IEEE Transactions on Cybernetics, vol. 47, no. 7, pp. 1743–1756, 2017.

[8] J. Branke, "Memory enhanced evolutionary algorithms for changing optimization problems," in IEEE Congress on Evolutionary Computation. IEEE, 1999, pp. 1875–1882.

[9] S. B. Gee, K. C. Tan, and H. A. Abbass, "A benchmark test suite for dynamic evolutionary multiobjective optimization," IEEE Transactions on Cybernetics, vol. 47, no. 2, pp. 461–472, 2017.
[10] S. Jiang, M. Kaiser, S. Yang, S. Kollias, and N. Krasnogor, "A scalable test suite for continuous dynamic multiobjective optimization," IEEE Transactions on Cybernetics, pp. 1–13, 2019.
[11] S. Jiang and S. Yang, "Evolutionary dynamic multiobjective optimization: Benchmarks and algorithm comparisons," IEEE Transactions on Cybernetics, vol. 47, no. 1, pp. 198–211, 2017.
[12] T. T. Nguyen and X. Yao, "Continuous dynamic constrained optimization—the challenges," IEEE Transactions on Evolutionary Computation, vol. 16, no. 6, pp. 769–786, 2012.
[13] C. Bu, W. Luo, and L. Yue, "Continuous dynamic constrained optimization with ensemble of locating and tracking feasible regions strategies," IEEE Transactions on Evolutionary Computation, vol. 21, no. 1, pp. 14–33, 2017.
[14] Y. Wang, J. Yu, S. Yang, S. Jiang, and S. Zhao, "Evolutionary dynamic constrained optimization: Test suite construction and algorithm comparisons," Swarm and Evolutionary Computation, vol. 50, p. 100559, 2019.
[15] Yazdani, D, Omidvar, MN orcid.org/0000-0003-1944-4624, Cheng, R et al. (3 more authors) (2020) Benchmarking Continuous Dynamic Optimization: Survey and Generalized Test Suite. IEEE Transactions on Cybernetics. ISSN 2168-2267 https://doi.org/10.1109/TCYB.2020.3011828.
[16] INTEGRATED METHODS FOR OPTIMIZATION by JOHN N. HOOKER.
[17] introduction to numerical methods of differential equations.
[18] INTRODUCTION TO APPLIED OPTIMIZATION, Second Edition. By URMILA DIWEKAR, Vishwamitra Research Institute, Clarendon Hills, IL, USA.
[19] Convex Optimization, Stephen Boyd, Department of Electrical Engineering, Stanford University, Lieven Vandenberghe, Electrical Engineering Department, University of California, Los Angeles.
[20] L. Bianchi, Notes on dynamic vehicle routing - state of the art - Technical report idsia 05-01, Italy, 2000.
[21] Introduction to Algorithms. Third Edition, Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein.
[22] J. Branke, Evolutionary Optimization in Dynamic Environments. Kluwer Academic Publishers, 2002.
[23] J. Grefenstette, Evolvability in Dynamic Fitness Landscapes: A Genetic Algorithm Approach. In Proc. 1999 Congress on Evolutionary Computation (CEC 99), Washington, DC. IEEE Press, pp. 2031-2038.
[24] OPTIMIZATION Algorithms and Applications Rajesh Kumar Arora, Senior Engineer, Vikram Sarabhai Space Centre, Indian Space Research Organization, Trivandrum, India.
[25] Al-Jilawi, A. S., & Abd Alsharify, F. H. (2022). Review of Mathematical Modelling Techniques with Applications in Biosciences. Iraqi Journal For Computer Science and Mathematics, 3(1), 135-144.
[26] Alridha, A., Wahbi, F. A., & Kadhim, M. K. (2021). Training analysis of optimization models in machine learning. International Journal of Nonlinear Analysis and Applications, 12(2), 1453-1461.
[27] Kadhim, M. K., Wahbi, F. A., & Hasan Alridha, A. (2022). Mathematical optimization modeling for estimating the incidence of clinical diseases.

International Journal of Nonlinear Analysis and Applications, 13(1), 185-195.

[28] Alridha, A., Salman, A. M., & Al-Jilawi, A. S. (2021, March). The Applications of NP-hardness optimizations problem. In Journal of Physics: Conference Series (Vol. 1818, No. 1, p. 012179). IOP Publishing.

[29] Salman, A. M., Alridha, A., & Hussain, A. H. (2021, March). Some Topics on Convex Optimization. In Journal of Physics: Conference Series (Vol. 1818, No. 1, p. 012171). IOP Publishing.

[30] Alridha, A., & Al-Jilawi, A. S. (2021, March). Mathematical Programming Computational for Solving NP-Hardness Problem. In Journal of Physics: Conference Series (Vol. 1818, No. 1, p. 012137). IOP Publishing.

[31] Alridha, A. H., & Al-Jilawi, A. S. (2022). Solving NP-hard problems using a new relaxation of approximate methods. International Journal of Health Sciences, 6(S3), 523–536. https://doi.org/10.53730/ijhs.v6nS3.5375