IJE TRANSACTIONS B: Applications Vol. 39, No. 11, (November 2026) 2644-2658

International Journal of Engineering

Journal Homepage: www.ije.ir

Fig Tree Optimization Algorithm: A Case Study on Adaptive Cruise Control

M. Allahi Rudposhti, A. Bohlooli*, K. Jamshidi

Faculty of Computer Engineering, University of Isfahan, Isfahan, Iran

PAPER INFO

Paper history:
Received 29 August 2025
Received in revised form 21 December 2025
Accepted 30 January 2026

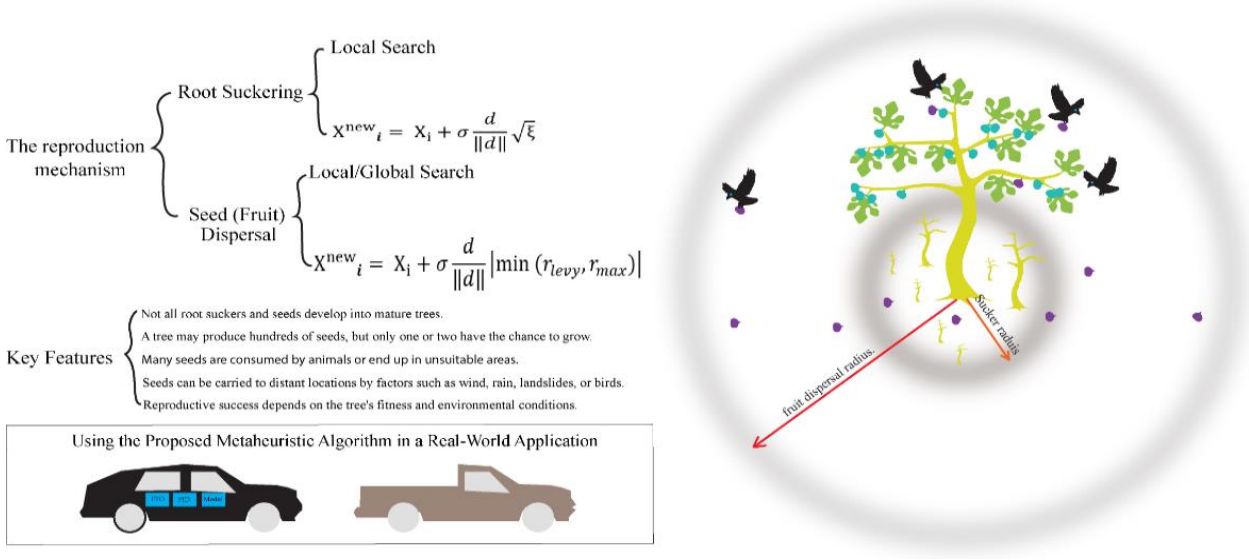
Keywords:
Stochastic Optimization
Cruise Control
Fig Tree Optimization
Meta-heuristic Algorithms

ABSTRACT

This paper presents the Fig Tree Optimization (FTO) algorithm as a novel nature-inspired metaheuristic for solving complex optimization problems, focusing on automotive applications. Inspired by root sucker propagation and seed dispersal in fig trees, FTO executes these two search strategies in a parallel and overlapping manner, resulting in rapid convergence toward solutions near the global optimum. The effectiveness of FTO is systematically validated through extensive experiments on 28 benchmark functions, including 16 classical benchmark functions and 12 standard benchmark functions. Its performance was compared against four state-of-the-art algorithms: Growth Optimizer (GO), Puma Optimizer (PO), Success-Based Optimization Algorithm (SBOA), and Gray Squirrel Food Search Algorithm (GSFA). The results consistently demonstrate the superiority of FTO in terms of accuracy, convergence speed, and solution quality. The practical application of FTO was evaluated by tuning a PID controller in an Adaptive Cruise Control (ACC) system and comparing its performance with the PO algorithm across three modes. In the first mode, with computational time similar to PO, the average best cost was 4-fold higher, making it suitable for energy-limited scenarios with lower accuracy requirements. In the second mode, a 33% longer runtime yielded a 1.5-fold performance improvement, fitting cases with constrained energy and time but requiring high accuracy. In the third mode, a 143% increase in runtime (reducible to 33% in parallel) enhanced overall efficiency by 8.15-fold. This mode is appropriate when ample resources are available and high accuracy is required. FTO is an efficient tool for both benchmark and real-world challenges.

doi: *10.5829/ije.2026.39.11b.01*

Graphical Abstract



*Corresponding Author Email: bohlooli@eng.ui.ac.ir (A. Bohlooli)

Please cite this article as: Allahi Rudposhti M, Bohlooli A, Jamshidi K. Fig Tree Optimization Algorithm: A Case Study on Adaptive Cruise Control. International Journal of Engineering, Transactions B: Applications. 2026;39(11):2644-58.

1. INTRODUCTION

Throughout history, humans have consistently sought to maximize the efficiency and optimal use of their limited resources across various domains. This drive for enhanced productivity has spurred the development of advanced optimization methods, which are now indispensable in fields such as engineering, economics, resource management, and modern technologies (1). Optimization is formally defined as the process of identifying the optimal value of an objective function subject to a set of constraints (2). These methods can be broadly categorized into two main types: Deterministic Methods and Stochastic Methods (3, 4). Deterministic methods rely on rigorous mathematical principles and guarantee precise solutions when the problem falls within their scope. Examples include Linear Programming (LP) (5) and Gradient-Based Optimizer (GBO) (6). In contrast, stochastic methods employ random search techniques to explore the solution space and typically converge to either the optimal or near-optimal solution after multiple iterations. These methods are particularly well-suited for addressing complex, nonlinear, and multifaceted problems that deterministic approaches often fail to solve effectively. Stochastic methods are further divided into two subcategories: Heuristic and Meta-Heuristic approaches. Heuristic methods adopt straightforward strategies based on rules of thumb or prior experience to derive solutions. On the other hand, meta-heuristic methods are more sophisticated and are specifically designed to tackle intricate, nonlinear, and high-dimensional problems that heuristic methods cannot efficiently resolve (3, 4). Many meta-heuristic algorithms draw inspiration from biological, physical, or natural phenomena, making them versatile tools for solving complex, challenging, and sometimes intractable real-world problems. Meta-heuristic algorithms are characterized by several key advantages, including ease of implementation, adaptability to external factors influencing the problem, effective exploration of the solution space, capability to identify global optima, independence from gradient information, and avoidance of local optima traps. Consequently, these methods have gained significant popularity in recent years due to their ability to address complex real-world challenges (7). Meta-heuristic algorithms can be further classified into two broad categories: Metaphor-Based and Non-Metaphor. Non-Metaphor algorithms are inspired by mathematical or computational concepts and do not mimic any natural or biological processes. These methods typically rely on mathematical principles or random search techniques Examples include Runge–Kutta method (RUN) (8) and Newton-Raphson-Based Optimizer (NRBO) (9). In contrast, Metaphor-Based algorithms are inspired by natural, physical, or biological phenomena (3, 4). Based on their sources of inspiration and design principles, meta-heuristic algorithms can be grouped into three primary categories: Human-Inspired, Laws-Inspired, and Nature-Inspired. Figure 1 shows the classification of metaheuristic algorithms. The Human-Inspired category encompasses algorithms inspired by human behavior, both at the individual and collective levels. Examples include social or group behaviors such as Hiking Optimization Algorithm (HOA) (10), Rock-Climbing Group (RCG) (11), Gold Seekers Algorithm (GSA) (12), Human Felicity Algorithm (HFA) (13), as well as individual behaviors like Deep Sleep Optimizer (DSO) (14) and Creative Thinking Algorithm (CTA) (15). The Laws-Inspired category consists of algorithms rooted in fundamental laws of nature and science. These methods leverage principles derived from various scientific domains and can be further subdivided into three subcategories: Physical Laws, Chemical Laws, and Biological Laws.

Physical Laws: Examples include gravity, particle motion, and energy dynamics. For instance, algorithms such as the Projectiles Optimization (PRO) (16), and Henry Gas Solubility Optimization (HGSO) (17) are inspired by these concepts.

Chemical Laws: Including chemical reactions, molecular transformations, and thermodynamic processes. For instance, some of these algorithms are Atomic Orbital Search (AOA) (18), and Homonuclear Molecules Optimization (HMO) (19).

Biological Laws: These algorithms are often inspired by natural processes, such as genetic evolution, cellular growth, and ecological interactions. For example, Cellular Genetic Algorithms (CGA) (20) represent one such approach.

The third category includes algorithms inspired by natural phenomena, living organisms, ecosystems, and natural structures. These inspirations manifest in diverse forms, which can be categorized as follows:

Symbiotic-Based: These algorithms model interactions between different organisms within an ecosystem, such as mutualism, parasitism, and commensalism. For example, the Symbiotic Organisms Search (SOS) (21) is inspired by such ecological relationships.

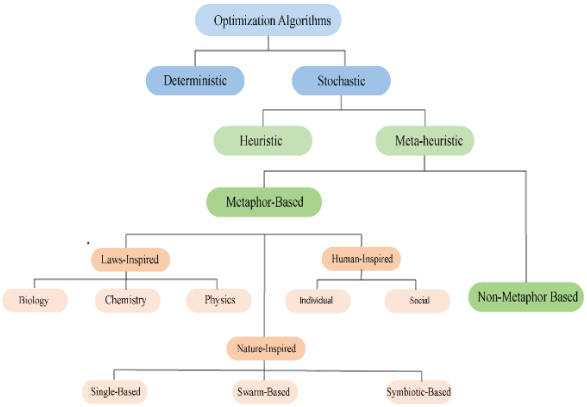


Figure 1. The classification of metaheuristic algorithms

Swarm-Based: Modelling collective movements and group dynamics, such as the coordinated behavior of animals (e.g., gray wolves), plant colonies (e.g., invasive weeds), or even natural structures (e.g., crystalline formations). These algorithms are commonly used for optimization and searching in large, complex spaces. Examples of Swarm-Based algorithms include Whale Optimization Algorithm (WOA) (7), Grasshopper Optimization Algorithm (GOA) (22), Crystal Structure Algorithm (CryStAl) (23), Dandelion Optimizer (DO) (2), Growth Optimizer (GO) (24), Walrus optimizer (WO) (25), Puma Optimizer (PO) (26), Starfish Optimization Algorithm (SFOA) (27), Success-Based Optimization Algorithm (SBOA) (28), and Gray Squirrel Foraging Algorithm (GSFA) (29).

Single-Based: Focusing on the actions of a single organism or entity in its natural environment, capturing localized decision-making and survival strategies. Example of Single-Based include Cuckoo Search (30).

In this paper, we proposed a new meta-heuristic algorithm called Fig Tree Optimization (FTO), which mimics the reproduction process of fig trees. To the authors' knowledge, no research has been conducted in this area to date. The paper is structured as follows: Section 2 delves into the inspiration, theoretical foundations, mathematical modeling, and procedural steps of the FTO algorithm. Section 3 evaluates the algorithm's performance through comparative analysis with four recent meta-heuristics (GO, PO, SBOA and GSFA) across 28 benchmark functions. Section 4 presents a practical case study, applying FTO to optimize PID controller coefficients in Adaptive Cruise Control (ACC) systems. The paper concludes with Section 5, which summarizes the findings and suggests directions for future research.

2. FIG TREE OPTIMIZATION (FTO)

In this section, we first discuss the inspiration behind the proposed method. Subsequently, we present the mathematical model.

2. 1. Inspiration The fig tree, scientifically designated as Ficus carica, belongs to the Mulberry family and is classified as a fruit-bearing plant. This species typically thrives in tropical and subtropical regions and is notable for its edible fruits, which can be found as either solitary or perennial plants in various parts of the world. Figs are generally recognized for their sweetness and nutritional benefits. The fig tree features large palmate leaves and produces fruits that contain small seeds. It is distinguished not only by its desirable qualities and flavor but also by its unique propagation methods compared to other plant species (31). The fig tree is commonly propagated using four primary methods:

Cutting: In this method, young and freshly cut fig branches are taken as cuttings, and under suitable conditions, they develop roots to become new trees.

Grafting: This involves connecting a part of one fig tree to another fig tree or another type of tree. This method is used to transfer desired traits or improve tree performance.

Root Suckering: After a certain growth stage, some of the roots of a fig tree emerge as new shoots from the soil around the mother plant until they grow into mature trees.

Seed Propagation: According to Figure 2, fig tree propagation through fig fruits is facilitated by tiny wasps from the Agaonidae family, which have a symbiotic relationship with the fig tree. This propagation cycle occurs as follows:

Step One: The female wasp, which has just exited a ripe fig and is carrying fig pollen (the pollen is stuck to the wasp's body), flies to an unripe fig (syconium).

Step Two: The female wasp enters the syconium through a small opening. Due to the narrowness of the entrance, part of the wasp's body or wings may be damaged, and the female wasp enters the fig without wings.

Step Three: The female wasp moves inside the fig, and as she moves, she spreads pollen, which fertilizes the female fig flowers, and some of them turn into seeds. The female wasp simultaneously lays eggs in some of the flowers and then dies. The eggs turn into larvae, forming galls inside the fig.

Step Four: The male wasps emerge from the galls and move to find the female inside the fig to fertilize them.

Step Five: The male wasps dig tunnels for the female wasps to exit the fig, and then they die inside the fig.

Step Six: The female wasps exit through the tunnels dug by the males, carrying fig pollen stuck to their bodies, which they transfer to the female unripe fig. This cycle continues as the female wasps lay eggs in new figs.

2. 2. Mathematical model of FTO The proposed algorithm is inspired by the natural reproductive behavior of fig trees, which utilize two parallel strategies for propagation: root suckering and seed dispersal. In the

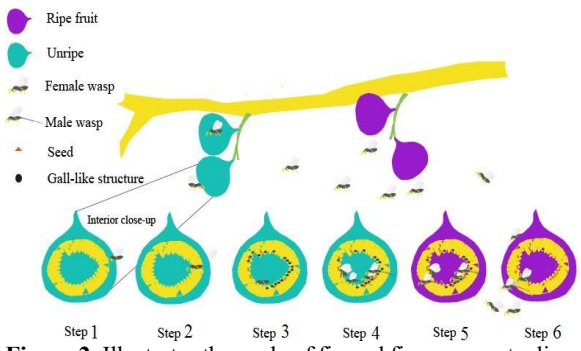


Figure 2. Illustrates the cycle of fig and fig wasp mutualism

initialization phase, a population of fig trees (candidate solutions) is randomly distributed across the search space. Each tree's growth and reproduction capability is proportional to its fitness value, representing the quality of the solution. Reproduction occurs through two distinct mechanisms:

Root suckering (exploitation): Root suckers develop in close proximity to the parent tree, representing local search around promising solutions. This process facilitates exploitation by intensifying the search in regions with high fitness values.

Seed dispersal (exploration): Seeds may be dispersed over varying distances—from nearby to remote locations—via environmental factors such as wind, water, or wildlife. This mechanism promotes exploration by enabling the algorithm to discover new, potentially optimal regions in the search space.

Not all reproduction attempts succeed. A fig tree may produce numerous seeds, yet only a small fraction germinate and develop into mature trees. Similarly, in FTO, the number of successful offspring is stochastic and depends on both the parent's fitness and environmental constraints. This selective survival mechanism ensures a balance between exploration and exploitation while maintaining population diversity.

2. 2. 1. Initialization Phase The initial population is defined as POP = $\{X_1, X_2,\ldots, X_{Pmax}\}$ consists of P_{max} individuals (the maximum number of fig trees in the search space) that are randomly distributed in the D-dimensional search space. Each individual X_i is represented as a vector $X_i = [x_{i,1}, x_{i,2}, \ldots, x_{i,D}]$, where $x_{i,j}$ denotes the coordinate of the i-th individual in the j-th dimension of the search space.

The initial values for each dimension are randomly generated within the permissible bounds. The individuals are then sorted in ascending order based on their objective function values $f(X_i)$:

$$\text{Sort(POP)} = \{X_1, X_2, \ldots, X_{Pmax}\}, \quad f(X_1) \le f(X_2) \le \ldots \le f(X_{Pmax}) \tag{1}$$

This sorted population is utilized in subsequent phases for selecting elite individuals and applying exploitation and exploration operations.

2. 2. 2. Exploitation Phase (Root Suckering) This phase simulates the root suckering process of fig trees, which performs local search around promising solutions. The following steps are executed:

1. Calculation of root sucker count: The number of root suckers (R) generated by each tree is calculated using Equation 2.

$$R_i = \left\lfloor \frac{f_{max} - f_i}{f_{max} - f_{min}} \times (r_{max} - r_{min}) + r_{min} \right\rfloor \tag{2}$$

where f_i represents the fitness value of the i-th fig tree, while f_{min} and f_{max} correspond to the lowest and highest fitness values of the population, respectively. Similarly, r_{min} and r_{max} define the minimum and maximum number of root suckers that a fig tree is permitted to generate. This equation indicates that fig tree with higher fitness values (f_i) will produce a greater number of offspring, facilitating a more extensive search in promising regions of the solution space.

2. Calculation of standard deviation: The standard deviation σ_k at the k-th iteration is computed using Equation 3.

$$\sigma_k = \left(\frac{iter_{max} - k}{iter_{max}}\right)^n \times (\sigma_{max} - \sigma_{min}) + \sigma_{min} \tag{3}$$

where $iter_{max}$ is the total number of iterations, n is a nonlinear modulation factor, and σ_{min} and σ_{max} are the minimum and maximum allowable standard deviation values.

3. Generation of root sucker positions: The position of each new root sucker is determined using Equation 4.

$$X^{new}_i = X_i + \sigma_k . S \tag{4}$$

According to Equation 4, the new position of each individual (X_i^{new}) is updated based on its current position (X_i), the σ_k calculated for every iteration, and S, a random vector defined by Equation 5.

$$S = \frac{d}{\|d\|}\sqrt{\xi} \tag{5}$$

Where d is a random vector with a standard normal distribution (i.e., d=(N(0,1),N(0,1),…,N(0,1)), ||d|| is the norm of this vector, ensuring directional normalization, and ξ is a uniform random variable in the range [0,1].

2. 2. 3. Exploration Phase (Seed Dispersal) This phase simulates seed dispersal, enabling global search in distant regions. The following steps are executed:

1. Calculation of seed count: The number of seeds produced by each tree is calculated as Equation 6.

$$S_i = \left\lfloor \left(\frac{f_{max} - f_i}{f_{max} - f_{min}}\right) . S_{max} . P_s \right\rfloor \tag{6}$$

Where S_{max} is the maximum allowable number of seeds and Ps is the probability of a seed growing into a tree (between 0 and 0.01).

2. Generation of seed positions: To simulate seed dispersal, the Lévy distribution (32) is employed due to its heavy tails and capacity for modeling long jumps. The position of each new seed (X_i^{new}) is computed using Equation 7.

$$X^{new}_i = X_i + \sigma_k . L \tag{7}$$

where X_i is the current position, and L is a random vector defined by Equation 8.

$$L = \frac{d}{\|d\|}\left|\min(N(0,1).\frac{|N(0,\sigma_{levy}{}^2)|}{|N(0,1)|^{\frac{1}{\beta}}}, r_{max})\right| \tag{8}$$

where σ_{levy} is the scale parameter of the Lévy distribution, as defined by Equation 9. β is the step-length control

parameter of the Lévy distribution ($0<\beta<2$), and r_{max} is the maximum allowable step size.

$$\sigma_{levy} = \left(\frac{\Gamma(1+\beta)\sin\left(\frac{\pi\beta}{2}\right)}{\Gamma\left(\frac{1+\beta}{2}\right)\beta 2^{\frac{\beta-1}{2}}} \right)^2 \quad (9)$$

2. 2. 4. Selection and Population Update After generating new individuals (root suckers and seeds), a combined population of old and new individuals is formed and sorted based on objective function values. Only the top P_{max} individuals are selected to form the new population. This process is repeated in each iteration, and the best solution is stored at every step. The termination condition can be defined as reaching the maximum number of iterations or achieving the desired accuracy.

2. 3. The Procedure of FTO The proposed algorithm consists of three main phases: initialization, exploration, and exploitation (Algorithm 1). Inspired by the biological behavior of fig trees—which are capable of simultaneous root suckering and seed dispersal after maturation—the algorithm performs root suckering (local exploitation) and seed dispersal (global exploration) in parallel. A distinctive feature of this algorithm is its integration of both local and global search during the exploration phase, mimicking the natural pattern of seed dispersal that can occur both near and far from the parent tree. As shown in the pseudocode of Algorithm 1, the FTO algorithm features independent exploration and exploitation loops with parallelization capability. It simultaneously performs global search through Lévy flight to escape local optima and local optimization using sub-Gaussian distributions. The algorithm iteratively merges the temporary and main populations while selecting the top P_{max} solutions. This balanced approach ensures an efficient search for the global optimum. The unique feature of this algorithm lies in its simultaneous utilization of overlapping search mechanisms, enabling efficient exploration of the search space. This approach offers two main advantages:

1. Accelerated optimization through parallel processing of exploration and exploitation processes.

2. Adaptive search that preserves high-quality solutions while simultaneously exploring new regions.

Algorithm 1. Pseudo-code of FTO

Begin
Initialize population POP randomly in D-dimensional space
k = 1
While k ≤ iter_max **Do**
Sort POP by fitness values (Equation 1)
f_min = minimum fitness, f_max = maximum fitness
For each individual X_i **in** POP **Do**
R_i = calculate using Equation 2
σ_k = calculate using Equation 3
S_i = calculate using Equation 6
For j = 1 **to** R_i **Do**
Generate random vector d with normal distribution
Generate uniform random variable ξ
Calculate S value using Equation 5.
X_new = X_i + σ_k * S // Equation 4
Add X_new **to** TEMP_POP
End For
For j = 1 **to** S_i **Do**
Generate random vector d with normal distribution
Calculate σ_levy using Equation 9.
Generate L value using Equation 8.
X_new = X_i + σ_k * L // Equation 7
Add X_new **to** TEMP_POP
End For
End For
Combine POP **and** TEMP_POP
Sort combined population by fitness
Select top Pmax individuals for new POP
Save Best Result
Update best_solution and best_fitness
k = k + 1
End While
Output best results

2. 4. Convergence to the Optimal Solution To prove the convergence of the proposed FTO algorithm, we employ a macroscopic model grounded in the Reaction-Diffusion Equation. This model characterizes the population density of individuals (fig trees) as a continuous field u(x, t) within the D-dimensional search space, where x denotes the spatial position and t represents the algorithm's iterations. The foundational equation is given by Equation 10.

$$\frac{\partial u}{\partial t} = D\nabla^2 u + R(u) \quad (10)$$

Where in the diffusion term $D\nabla^2 u$ encapsulates random displacements—encompassing root suckering via a sub-Gaussian distribution and seed dispersal via a Lévy distribution—while the reaction term R(u) delineates local reproduction and competitive selection predicated on the fitness function $f(X_i)$. In light of FTO's dual parallel strategies, we reformulate the equation in an extended form given by Equation 11.

$$\frac{\partial u(x,t)}{\partial t} = \sigma_k(t)\left(\alpha_L \nabla^2 u(x,t) + \alpha_N \nabla^2 u(x,t)\right) + \sum_{i=1}^{Pmax} \delta(x - X_i) \cdot R_i \cdot P_{accept}(X_i) + \sum_{i=1}^{Pmax} \delta(x - X_i) \cdot S_i \cdot P_s \cdot P_{accept}^{Levy}(X_i) \quad (11)$$

where $\sigma_k(t)$ (derived from Equation 3) serves as a monotonically decreasing temporal scaling factor, α_L and α_N denote the respective Lévy and sub-Gaussian diffusion coefficients (from Equations 5 and 8), and the reaction terms, informed by Equations 2 and 6, model the generation of root suckers and seeds alongside competitive acceptance probabilities (from the selection phase).

During the initial phase (small t, where $\sigma_k(t) \sim \sigma_{max}$), the diffusion term predominates, with the Lévy distribution's heavy-tailed structure ($P(r) \sim 1/r^{\beta+1}$ for $0 < \beta < 2$) ensuring robust global exploration. This elevates the likelihood of approaching the vicinity of the global optimum X* prior to σ_k diminishing to half its initial value, thereby mitigating entrapment in local optima. In the terminal phase (large t, as $\sigma_k(t) \sim \sigma_{min}$), the reaction terms dominate. Through elitist selection of the top P_{max} individuals, the density u converges to a sharp peak centered at X, corresponding to the global minimum $f(X^*) = f_{min}$. In conclusion, as described in Equation 11, the FTO algorithm effectively explores the search space in its initial stages through Lévy and sub-Gaussian distributions, gradually focusing on promising regions. The inherent competitive elimination mechanism of the algorithm directs the population toward convergence at a single point. Even if this point is not the global optimum—meaning regions with superior fitness remain unexplored—the probability of entrapment in local optima is low; since the Lévy distribution periodically generates long jumps and examines distant areas. Overall, these dynamics justify the convergence of the FTO algorithm to the global optimum, as the population density u(x,t) gradually concentrates around X*.

3. RESULTS AND DISCUSSION

In this section, we conduct several simulation studies to demonstrate the advantages of the proposed optimization algorithm. First, we evaluate the algorithm's ability to find the global minimum of 16 benchmark functions commonly used in the literature. To verify that the algorithm converges to the global solution, we compare its results with those of GO (24), PO (26), SBOA (28), and GSFA (29). Next, we assess the algorithm's performance using the 12 standard benchmark functions from CEC 2022, comparing the mean best cost with the five algorithms mentioned in this study. Finally, we apply the proposed algorithm to a practical problem: improving driving comfort in ACC systems.

3. 1. Evaluation on Well-Known Benchmarks In this section, the proposed FTO algorithm has been evaluated on 16 well-known benchmark functions in the field of optimization. These functions cover a diverse set of challenges, including unimodal problems (F1–F4), multimodal problems (F8–F10), and hybrid or penalized problems (F12–F13). Collectively, they test essential aspects of optimization algorithms such as local convergence, global exploration, and constraint handling. Table 1 presents the specifications of these functions, where: Dim represents the number of dimensions of the function, Range denotes the search space boundaries and f_{min} indicates the optimal value of the function. Additionally, in Figure 3, a 3D plot of the 2D benchmark functions is illustrated.

The experiments were conducted in MATLAB 2016b. five metaheuristic algorithms were implemented under identical conditions: GO (24), PO (26), SBOA (28), GSFA (29), and the proposed FTO. For FTO, the parameters were configured as follows: population size was set to a maximum of 100 individuals; however, for practical problems with higher complexity, this number was decreased to 30. The exponent (E) for sigma (σ) decay was set to 5 to optimally regulate the reduction rate, with a practical case value of 2. The initial sigma was 1, and its final value was set to 1.00E-13 to ensure the algorithm's convergence toward the optimal solution with high precision. The range of root suckers per tree was set between 0 and 5 to maintain diversity in reproduction and population expansion. On average, each tree produces 1000 fig fruits, of which only 1% (with a probability of Ps= 0.01) are transformed into new trees based on the fitness criterion of the parent tree. This mechanism ensures that higher-quality solutions have a greater chance of survival and participation in subsequent generations. The Lévy distribution parameter β was fixed at 0.1 to control step size, while the maximum search radius in the Lévy relation was set to 10σ to dynamically balance exploration and exploitation. The parameters of other implemented algorithms were also adjusted according to the information provided in the paper, ensuring fair comparison under identical conditions.

In unimodal benchmarks, FTO demonstrated highly competitive performance. Specifically for F1 and F2, while PO achieved the most accurate results, FTO secured the second-best position, outperforming older algorithms like GO and the newer GSFA approach. In F3 and F4, PO again delivered the best performance, with FTO achieving the second-closest results to PO, significantly outperforming SBOA, GO, and GSFA. Notably, in F6, FTO achieved the absolute best result among all algorithms. Overall, in the unimodal category, FTO consistently produced competitive results, in some cases surpassing the current state-of-the-art.

In multimodal benchmarks, the performance of FTO was more varied. For F7, FTO achieved the best solution, outperforming all competitors. In F8, with a theoretical optimum of approximately –1257, both PO and GO reached near-optimal values (≈ –1260), whereas FTO obtained –1120. Although weaker than PO and GO, this was substantially better than SBOA (–999). In F9 and F11, FTO underperformed compared to the competitors, falling behind even SBOA. However, in F10, FTO achieved results close to PO and significantly better than SBOA and GSFA. In F14, all algorithms produced nearly identical solutions close to zero, which is consistent with the problem's characteristics. Taken together, FTO performed strongly in F7 and F10, while in F9 and F11 its performance was less competitive.

TABLE 1. Unimodal and multimodal benchmark functions

Functions	Dim	Range	f_{min}
$F_1(x) = \sum_{i=1}^{n} x_i^2$	3	[-100,100]	0
$F_2(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	3	[-10,10]	0
$F_3(x) = \sum_{i=1}^{n} (\sum_{j=1}^{i} x_j)^2$	3	[-100,100]	0
$F_4(x) = \max_i \{\lvert x_i \rvert, 1 \le i \le n\}$	3	[-100,100]	0
$F_5(x) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	3	[-30,30]	0
$F_6(x) = \sum_{i=1}^{n} (\lvert x_i + 0.5 \rvert)^2$	3	[-100,100]	0
$F_7(x) = \sum_{i=1}^{n} i x_i^4 + random[0,1)$	3	[-1.28,1.28]	0
$F_8(x) = \sum_{i=1}^{n} -x_i \sin(\sqrt{\lvert x_i \rvert})$	3	[-500,500]	-418.9829×Dim
$F_9(x) = \sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i) + 10]$	3	[-5.12,5.12]	0
$F_{10}(x) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n} \cos(2\pi x_i)\right) + 20 + e$	3	[-32,32]	0
$F_{11}(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos(\frac{x_i}{\sqrt{i}}) + 1$	3	[-600,600]	0
$F_{12}(x) = \frac{\pi}{n}\{10\sin(\pi y_1) + \sum_{i=1}^{n-1}(y_i - 1)^2 [1 + 10\sin^2(\pi y_i)] + (y_n - 1)^2\} + \sum_{i=1}^{n} u(x_i, 10, 100, 4)$ $y_i = 1 + \frac{x_i + 1}{4}$ $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m & x_i > a \\ 0 & -a < x_i < a \\ k(-x_i - a)^m & x_i < -a \end{cases}$ *	3	[-50,50]	0
$F_{13}(x) = 0.1\{\sin^2(3\pi x_1) + \sum_{i=1}^{n}(x_i - 1)^2 [1 + \sin^2(3\pi x_i + 1)] + (x_n - 1)^2[1 + \sin^2(2\pi x_n)]\} + \sum_{i=1}^{n} u(x_i, 5, 100, 4)$	3	[-50,50]	0
$F_{14}(x) = -\sum_{i=1}^{n} \sin(x_i) . \left(\sin\left(\frac{ix_i^2}{\pi}\right)\right)^{2m}, m = 10$	3	[0,π]	-4.687
$F_{15}(x) = \left[\exp\left(-\sum_{i=1}^{n}\left(\frac{x_i}{\beta}\right)^{2m}\right) - 2\exp\left(e^{-\sum_{i=1}^{n} x_i^2}\right)\right] . \prod_{i=1}^{n} cos^2(x_i), m = 5$	3	[-20,20]	-1
$F_{16}(x) = \{[\sum_{i=1}^{n} \sin^2(x_i)] - \exp(-\sum_{i=1}^{n} x_i^2)\} . \exp[-\sum_{i=1}^{n} \sin^2(\sqrt{\lvert x_i \rvert})$	3	[-10,10]	-1

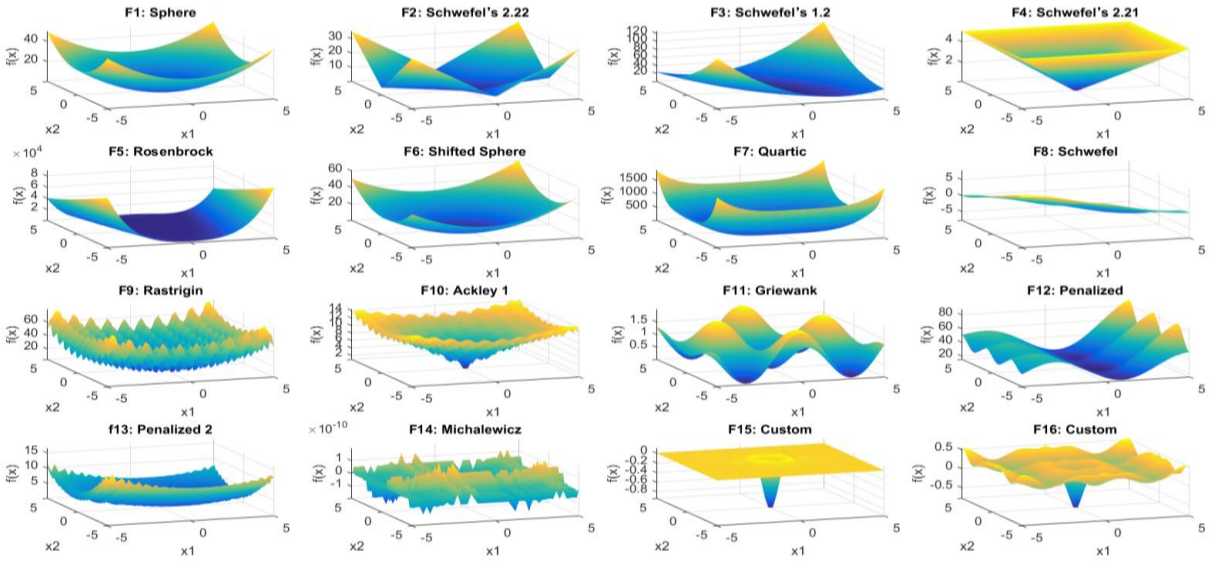


Figure 3. The 3D plots of the 2D benchmark functions

In hybrid and penalized benchmarks, FTO delivered particularly strong outcomes. In F12, almost all algorithms yielded similar results (≈ 19.8). In F13, FTO achieved the absolute best result, significantly outperforming all other methods. In F15, the theoretical optimum of –1 was reached precisely by FTO, along with PO, while GO and GSFA were close but slightly weaker, and SBOA produced a positive value far from the optimum. In F16, PO was closest to the optimum (–1), while FTO (–0.40) outperformed SBOA but was weaker than PO and GO. Overall, FTO exhibited absolute superiority in F13 and F15, while delivering competitive but not leading results in F16.

The evaluation results indicate that the proposed algorithm performs competitively with state-of-the-art methods and, particularly for complex functions, surpasses them. The heatmap in Figure 4 (green = optimal cost, red = suboptimal cost) demonstrates that the proposed method achieves the second-highest concentration of green after PO, affirming its competitiveness. Furthermore, as observed in Figure 5 and Table 2, the proposed algorithm converges to the desired solution within fewer than 20 iterations in most cases, while other methods generally require more iterations to achieve convergence. A key strength of the proposed algorithm is its inherent adaptability to resource constraints.

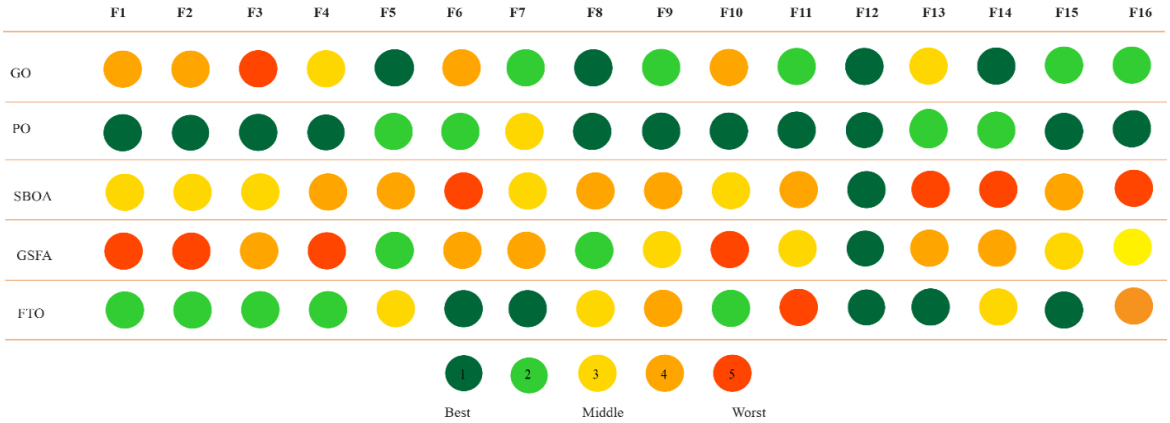


Figure 4. Heatmap comparison of algorithms

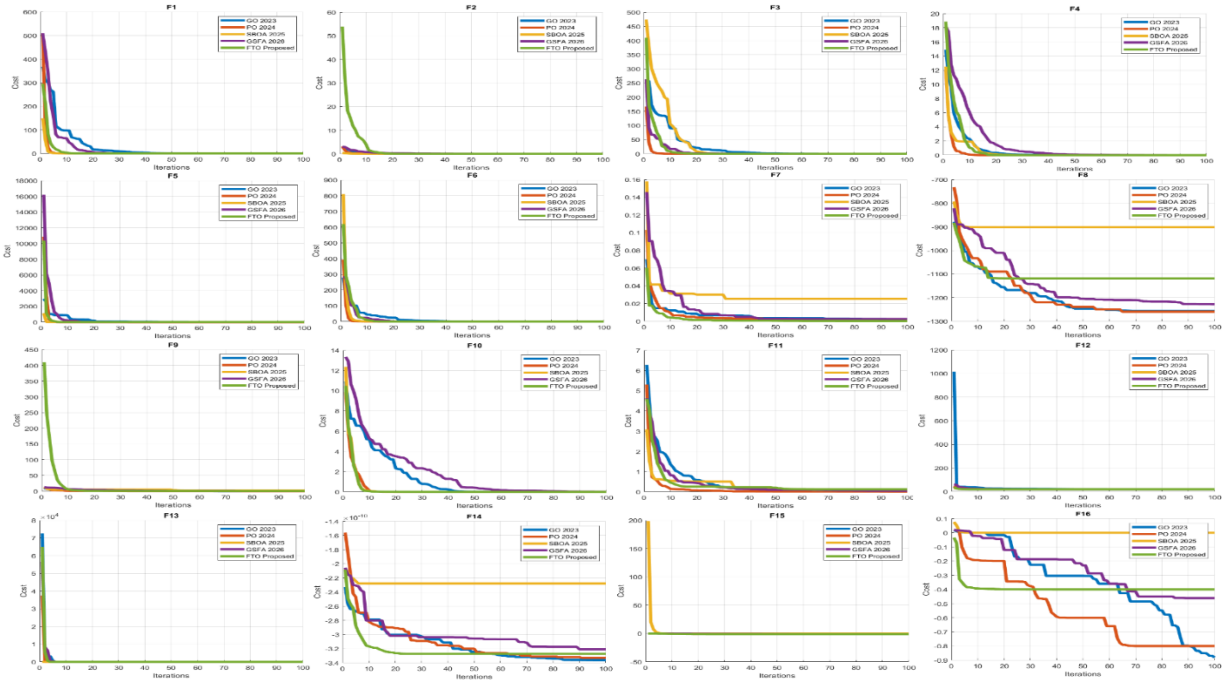


Figure 5. Convergence curves of optimization algorithms on well-known benchmarks (F1–F16)

TABLE 2. Average cost comparison of algorithms over 100 iterations and 5 runs on well-known benchmarks

Test	GO 2023 (24)	PO 2024 (26)	SBOA 2025 (28)	GSFA 2026 (29)	Proposed FTO
F1	5.26E-07	5.29E-48	2.55E-16	7.09E-07	1.06E-28
F2	1.29E-07	4.03E-22	1.15E-11	1.24E-04	8.80E-14
F3	3.74E-05	6.83E-46	2.45E-07	1.58E-06	1.99E-25
F4	1.47E-05	5.61E-25	6.00E-05	1.45E-03	3.69E-14
F5	3.00E+01	3.01E+01	3.20E+01	3.01E+01	3.19E+01
F6	2.60E-06	1.37E-07	4.21E-02	9.69E-07	2.77E-26
F7	1.71E-03	1.90E-03	5.19E-03	2.90E-03	1.53E-04
F8	-1.26E+03	-1.26E+03	-9.99E+02	-1.23E+03	-1.12E+03
F9	5.36E-05	0.00E+00	5.97E-01	5.85E-01	5.97E-01
F10	8.08E-06	8.88E-16	3.47E-08	1.76E-02	1.55E-13
F11	1.60E-02	1.28E-02	6.05E-02	4.15E-02	1.27E-01
F12	1.98E+01	1.98E+01	1.98E+01	1.98E+01	1.98E+01
F13	1.00E-06	4.81E-07	1.20E-04	3.97E-05	8.68E-25
F14	-3.36E-10	-3.33E-10	-2.86E-10	-3.18E-10	-3.27E-10
F15	-9.99E-01	-1.00E+00	2.21E-23	-9.27E-01	-1.00E+00
F16	-7.83E-01	-7.99E-01	3.51E-06	-4.87E-01	-4.00E-01

By parallelizing root sucker (sub-Gaussian) and seed dispersal (Lévy) mechanisms, the algorithm balances exploration and exploitation. It adaptively switches between single-mechanism (low-energy, fast convergence) and dual-mechanism (high-quality solutions) operation based on resource availability, making it particularly suitable for resource-constrained automotive applications.

3. 2. Evaluation on CEC 2022 We conducted a comprehensive evaluation of the proposed FTO algorithm over 30 iterations, with five independent runs on 12 standard benchmark functions. These benchmarks cover a broad range of function types, including simple, hybrid, and composite formulations. Table 3 summarizes their specifications, including names, classifications, mathematical descriptions, and known global minima. All evaluations were performed within a 10-dimensional search space of [-100, 100]. The comparative analysis focuses on the mean best cost achieved in the final iteration by FTO relative to four state-of-the-art optimization methods. As reported in Table 4, which presents the average best costs across all runs, FTO outperforms the previously best-performing PO algorithm in 8 out of 12 benchmark cases (66.7%) and maintains competitive performance in the remaining 4 cases (33.3%). The minor numerical differences observed in standard functions stem from the intrinsic bias of these functions. This inherent bias significantly assists optimizers in converging precisely toward global optimum values. The primary objective of this table is to highlight FTO's competitiveness with novel algorithms, rather than emphasizing minor numerical variations. By balancing local exploitation and global exploration mechanisms, FTO achieves rapid convergence with few iterations and delivers robust performance on complex functions. This makes it particularly well-suited for resource-constrained automotive applications.

TABLE 3. Details of CEC-2022 (standard) benchmark

Name	Type	Descriptions	f_{min}
F1	Unimodal Basic	Shifted and full rotated Zakharov	300
F2		Shifted and rotated Rosenbrock	400
F3		Shifted and rotated expanded Schaffer's F6	600
F4		Shifted and rotated non-continuous Rastrigin	800
F5		Shifted and rotated Levy function	900
F6	Hybrid	Hybrid function 1 (N=3)	1800
F7		Hybrid function 2 (N=6)	2000
F8		Hybrid function 3 (N=5)	2200
F9	Composite	Composition function 1 (N = 5)	2300
F10		Composition function 2 (N = 4)	2400
F11		Composition function 3 (N = 5)	2600
F12		Composition function 4 (N = 6)	2700

TABLE 4. Average cost comparison of algorithms over 30 iterations and 5 runs on standard benchmarks

Test	GO 2023 (24)	PO 2024 (26)	SBOA 2025 (28)	GSFA 2026 (29)	Proposed FTO
F1	300.000	300.000	300.000	300.000	300.000
F2	400.000	400.000	400.000	400.000	400.000
F3	600.000	600.002	600.000	600.001	600.000
F4	800.000	800.000	800.200	800.000	800.000
F5	900.0000	900.000	900.0000	900.000	900.000
F6	1800.002	1800.020	1800.000	1800.021	1800.000
F7	2424.196	2427.167	2424.101	2425.200	2423.727
F8	4819.083	4824.524	4824.402	4821.535	4819.840
F9	2494.968	2494.968	2494.968	2494.968	2494.968
F10	2400.000	2400.000	2400.000	2400.000	2400.000
F11	2600.002	2600.224	2600.000	2600.022	2600.000
F12	2700.199	2700.643	2700.000	2700.616	2700.006

4. PRACTICAL EXAMPLE

Driving requires significant mental focus and physical effort, resulting in fatigue. Activities such as adjusting the brakes, clutch, accelerator, changing gears, and steering contribute to this fatigue. According to a report (33), 94% of road accidents are due to human error. Automating parts of the driving process helps alleviate mental and physical fatigue, making driving more comfortable. Since the introduction of cruise control (CC) systems, driving has become more convenient by allowing drivers to maintain a steady speed without frequent braking or accelerating. CC has evolved significantly. Early systems only maintained a set speed, while ACC now also maintains a safe distance from the vehicle ahead. Initially designed for safety, ACC systems now address multiple objectives, including fuel efficiency and driving comfort, using optimal control methods like Model Predictive Control (MPC) (34). ACC systems often employ a hierarchical control structure: a high-level controller determines the vehicle's desired acceleration, while a low-level controller adjusts throttle and brake pressure to achieve this acceleration, as shown in Figure 6 (35).

The PID controller is commonly used in CC systems due to its simple theory, ease of implementation, and precise performance. The PID controller maintains the vehicle's speed by adjusting the throttle, improving comfort and safety. Its transfer function, as shown in Equation 12, uses proportional, derivative, and integral components to minimize output errors.

$$C(s) = K_p + K_d.S + K_i.S^{-1} \quad (12)$$

Where K_p is the proportional gain, K_d is the derivative gain, and K_i is the integral gain. In practice, tuning the PID controller's parameters is challenging, particularly for nonlinear systems like vehicle dynamics. Meta-heuristic algorithms are considered a suitable option for tuning PID parameters in ACC systems due to their compatibility with nonlinear problems and ability to escape local optima (36). Compared to other meta-heuristic methods, the FTO algorithm converges to a near-optimal solution with fewer iterations. Furthermore, its capability to operate in multiple modes represents a fundamental advantage for automotive systems with diverse operational requirements.

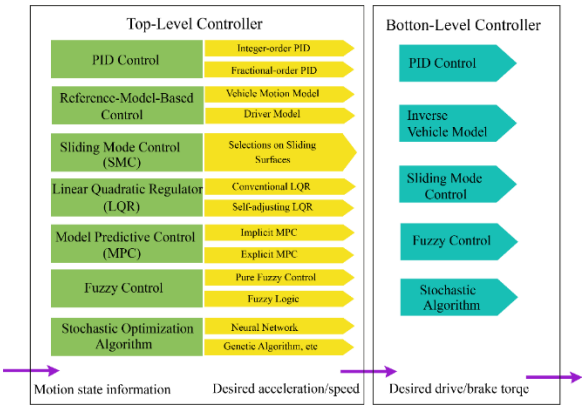


Figure 6. The structure of the hierarchical approach and the control units of each part (35)

4. 1. Vehicle Dynamic Model

The vehicle's longitudinal dynamics were modeled, incorporating forces such as rolling resistance, differential drag, aerodynamic drag, and gravity (37). The total road load force, denoted as F_{rl}, is defined by Equation 13.

$$F_{rl} = c_r mg\cos(\beta(x)) + c_d v(t) + c_{d0} + c_{gb} v(t) + c_{gb0} + \frac{1}{2}\rho_a A_f (ca + k) v_y^2 + \frac{1}{2}\rho_a A_f c_a (v(t) + \boldsymbol{v_x}(x)) \mid v(t) + v_x(x) \mid \quad (13)$$

In this equation, m represents the vehicle mass, and g denotes the gravitational acceleration. The rolling resistance coefficient is indicated by c_r, while the function $\beta(x)$ describes the road slope as a function of the traveled distance. For aerodynamic resistance, ρ_a is the air density, A_f is the vehicle's frontal area, and c_a is the aerodynamic drag coefficient. Wind velocity is decomposed into two components: v_x, representing the crosswind (perpendicular to vehicle motion), and v_y, representing the tailwind/headwind (aligned with the vehicle's direction). The coefficient k quantifies the effect of crosswind on aerodynamic resistance. c_{dc} and c_{d0} represent the differential loss coefficient and constant loss term, respectively. Similarly, c_{gb} and c_{gb0} represent the gearbox loss coefficient and constant loss component, respectively. This model is applicable only for forward vehicle velocities. Additionally, the vehicle experiences a gravitational force given by Equation 14.

$$F_g = mg\sin(\beta(x)) \quad (14)$$

The longitudinal dynamics of the vehicle are governed by Newton's second law of motion, considering the force equilibrium at the vehicle wheels by Equation 15.

$$m_e \frac{dv(t)}{dt} = \frac{[T_{ice} + T_{em} + T_{ser}] i_{gb} i_f}{r_e} - F_g - F_{rl} \quad (15)$$

Here, m_e is the equivalent mass of the vehicle, which accounts for the rotational inertia of the drivetrain. It is assumed constant, meaning inertia variations due to gear shifting are neglected. T_{ice} represents the net torque output from both the internal combustion engine and exhaust braking system. T_{em} indicates the torque generated or absorbed by the electric motor, while T_{ser} corresponds to the mechanical braking torque from the service brakes. The parameter r_e signifies the dynamic rolling radius of the tires, i_{gb} refers to the active transmission gear ratio determined by the vehicle's speed and torque requirements, and i_f represents the fixed ratio of the final drive unit. Since rolling resistance,

aerodynamic drag, and differential loss forces have the most significant impact on the total road load force, we consider only these forces and neglect the others for simplification. In this case, the final dynamic equation of the vehicle is obtained as Equation 16.

$$m_e \frac{dv(t)}{dt} = \frac{[T_{ice}+T_{em}+T_{ser}]i_{gb}i_f}{r_e} - mg\left(\sin(\beta(x)) + crcos(\beta(x))\right) - \frac{1}{2}\rho_a A_f c_a (v(t)+v_x(x)) \mid v(t)+v_x(x) \mid -c_d v(t) \quad (16)$$

4. 2. ACC Cost Function In this study, a hierarchical ACC system is designed, consisting of two cascaded PID controllers: a high-level controller responsible for speed tracking and distance maintenance, and a low-level controller tasked with acceleration control and jerk minimization. To optimize the parameters of both controllers simultaneously, a combined cost function is formulated that integrates multiple performance criteria from both control layers. The proposed cost function is defined in Equation 17.

$$J = W_1 \cdot ISE_{Speed} + W_2 . M_p^{speed} + W_3 \cdot T_S + W_4 \cdot ISE_{Distance} + W_5 . ISE_{Acceleration} + W_6 . \max|j(t)| \quad (17)$$

Where:

$ISE_{Speed} = \int_0^t (v_{ref} - v_{overshoot}(t))^2 dt$: Integral of Squared Error (ISE) for vehicle speed tracking.

$M_p^{speed} = \max(v_{overshoot}(t)) - v_{ref}$: Speed overshoot penalty to ensure passenger comfort and safety.

Ts: Settling time, defined as the time taken for the vehicle speed to stay within ±2% of the reference speed.

$ISE_{Distance} = \int_0^t (d_{safe}(t) - d_{lead}(t))^2 dt$: Distance error minimization term to maintain safe inter-vehicle spacing.

$ISE_{Acceleration} = \int_0^t (a_{desired}(t) - a_{actual}(t))^2 dt$: Acceleration tracking error to improve longitudinal control accuracy.

$\max|j(t)|$: Maximum absolute jerk over the simulation horizon, used to penalize abrupt accelerations and decelerations for ride comfort.

The weighting factors are set based on the relative importance of each objective in achieving stable and comfortable driving behavior. The values used in the simulations are W_1 = 1, W_2= 10, W_3= 3, W_4= 5, W_5=2, W_6=5. This cost function enables a unified optimization framework where both high-level and low-level PID parameters are tuned together to achieve an optimal trade-off between tracking accuracy, safety, passenger comfort, and control effort efficiency.

4. 3. Proposed ACC This study presents a hierarchical ACC system based on PID controllers, designed to ensure safe and comfortable vehicle operation under varying driving conditions. The control architecture is composed of two cascaded layers: a high-level PID controller responsible for speed tracking and inter-vehicle distance maintenance, and a low-level PID controller tasked with translating the desired acceleration into appropriate actuator commands while minimizing jerk and ensuring smooth longitudinal dynamics. Figure 7 illustrates the proposed hierarchical ACC system architecture based on the FTO metaheuristic algorithm. As shown in the figure, the model parameters are dynamically updated according to changing environmental conditions and vehicle dynamics. The FTO metaheuristic algorithm automatically adjusts these parameters to optimize PID coefficients based on the defined cost function. It should be noted that this study designed a cost function that simultaneously addresses the requirements of both high-level (speed/distance management) and low-level controller (acceleration control). The metaheuristic algorithm effectively solves a six-variable optimization problem (corresponding to the PID coefficients of both controllers). It is noteworthy that in the proposed ACC, a charge control signal is received from the vehicle's battery management system, which determines the operating mode of the proposed algorithm—a feature that distinguishes it from other metaheuristic algorithms. When the energy level is low, the proposed algorithm operates in a low-power search mode; whereas when sufficient energy is available, it performs optimization at full capacity. The optimization process occurs even in low iterations and for complex cost functions.

4. 4. Result from Practical Simulation In this section, the cost function of the proposed hierarchical ACC system is optimized using the PO (26) and FTO algorithms over 10 iterations. As mentioned, the proposed algorithm follows two mechanisms: root suckering with a sub-Gaussian distribution and seed scattering with a Lévy distribution. This feature serves as a fundamental advantage in automotive applications.

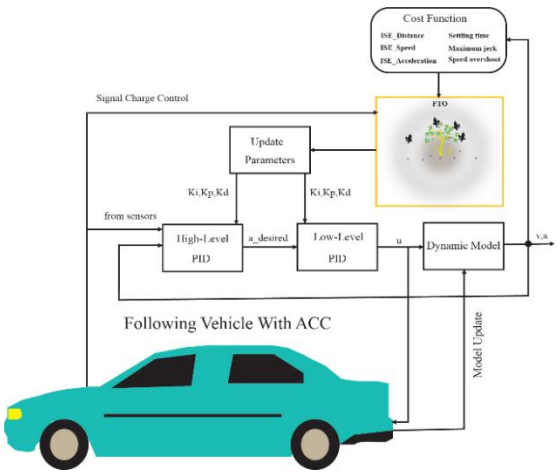


Figure 7. Architecture of the proposed hierarchical ACC system using the FTO algorithm

Accordingly, three operating modes are defined for the proposed algorithm. In the first operating mode, only the root suckering mechanism is activated. This mechanism is relatively simple and executes in less time, but it primarily conducts searches in promising points and examines new spaces to a lesser extent. In the second operating mode, only the seed scattering mechanism is active. This mechanism exhibits higher complexity and requires more time, but it yields better results since it adheres to a Lévy distribution pattern and performs both short and long jumps. In the third operating mode, both mechanisms are activated simultaneously, and with the overlap of the sub-Gaussian and Lévy distributions, the search space is monitored more effectively to discover additional optimal points. Furthermore, a signal is transmitted from the vehicle's battery management system to the algorithm to determine the operating mode based on the battery charge level.

The three operating modes of the proposed method, along with the PO algorithm, were simulated within the ACC proposed framework, and the average best cost obtained from 5 runs of each algorithm, along with their execution times, is presented in Table 5. As evident from the table, the average best cost for the PO algorithm is 3.66E+06 with an execution time of 20.34 seconds. In the first operating mode of the proposed algorithm, the average best cost is 1.52E+07 achieved in 21.31 seconds.

In this mode, although the execution time is nearly the same as PO, the optimization performance is significantly lower, achieving only about one-fourth of the improvement. This mode is suitable for scenarios where time and energy are limited and high accuracy is not critical.

In the second mode, the average best cost is 2.43E+06 with an execution time of 33.86 seconds, representing a 33% increase in runtime compared to PO and a 1.5-fold improvement in performance. This mode is suitable for situations where both time and energy are constrained, but higher accuracy is required.

Finally, in the third mode, the best cost is 4.49E+05 with an execution time of 56.84 seconds. This runtime corresponds to the sequential execution of two strategies; when executed in parallel, the runtime equals that of the longest serial loop segment. For the Lévy distribution, which requires more time, the parallel execution time becomes 33.86 seconds, a 33% increase relative to PO. This mode is suitable when sufficient computational resources are available and high accuracy is required. In the third mode, the proposed algorithm achieves an 8.15-fold improvement in optimization, with a 2.43-fold increase in serial computation time and a 1.5-fold increase in parallel computation time compared to PO, demonstrating its substantial superiority in practical applications.

TABLE 5. Average best cost and execution time: PO vs. proposed approach (10 Iterations, 5 Runs)

	PO [2024]	Proposed Mode 1	Proposed Mode 2	Proposed Mode 3
Best cost	3.66E+06	1.52E+07	2.43E+06	4.49E+05
Time (s)	20.34	21.31	33.86	56.84

Overall, the proposed algorithm, with a low number of iterations (for example, 10 iterations), provides highly efficient performance in solving real-world problems for complex functions such as vehicle dynamic equations, featuring three operational modes for energy management and utilizing sub-Gaussian and Lévy overlapping search.

5. CONCLUSION AND FUTURE WORK

This study introduces the FTO algorithm, a novel metaheuristic method inspired by the ecological adaptation and reproductive mechanisms of fig trees.

The algorithm effectively models biological processes such as root suckering and seed dispersal, leveraging them to achieve optimal performance in complex problems. Extensive evaluations on 16 well-known benchmark functions and 12 standard CEC 2022 test functions demonstrate that the proposed algorithm performs comparably on some functions and significantly outperforms reference methods on others.

The results indicate that FTO achieves excellent performance on complex functions even with a low number of iterations. The parallel implementation of the sucker growth mechanism with a sub-Gaussian distribution and the seed dispersal mechanism with a Lévy distribution distinguishes this algorithm for automotive applications. Based on this capability, a hierarchical ACC framework was developed, where the proposed algorithm operates in three modes to optimize the vehicle model’s cost function. In this structure, if the vehicle’s energy is limited, optimization is performed using only one of the mechanisms (either sucker growth or seed dispersal), while sufficient energy allows both mechanisms to operate simultaneously.

The algorithm effectively models biological processes such as root suckering and seed dispersal to achieve robust optimization performance. Extensive evaluations on 16 well-known benchmark functions and 12 standard CEC 2022 test functions demonstrate that the proposed algorithm not only converges rapidly to optimal solutions with fewer iterations but also delivers higher accuracy in solving complex problems. Simulation results of the proposed framework, compared with the best competing algorithm, show that FTO, with a 2.43-fold increase in computational time (1.5-fold increase in parallel computation time) compared to the PO algorithm, achieves 8.15 times more effective

optimization. This optimization enables precise tuning of vehicle parameters, resulting in smoother and safer driving.

In future works, we will focus on three main research directions: First, developing a cloud-based implementation of the FTO algorithm to reduce computational load on vehicle systems—particularly benefiting electric vehicles and systems with limited computational resources—aiming to facilitate broader adoption in intelligent transportation systems while maintaining optimization efficiency. Second, enhancing the algorithm's capability to operate in dynamic and complex environments by addressing variable communication delays and unpredictable traffic conditions. Third, integrating intelligent event detection and adaptive learning mechanisms to enable automatic parameter reconfiguration in response to critical scenarios such as road gradient changes or pedestrian presence. Additionally, practical implementation on hardware platforms and comprehensive field testing will be conducted to validate the algorithm's performance in real-world settings. Furthermore, a historical analysis module—illustrated in Figure 8—is incorporated into the proposed ACC system. This module recommends an desired vehicle speed by processing historical data, including time of day, day of the week, road conditions, and weather patterns, thereby enhancing driving safety.

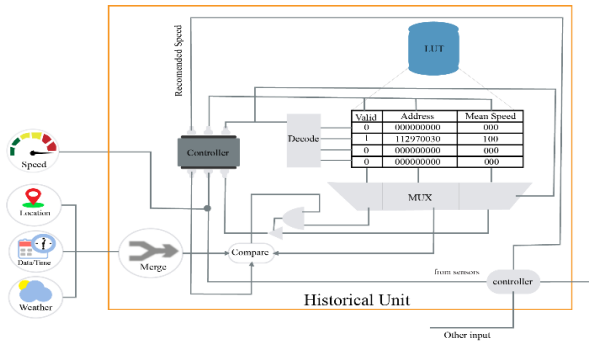


Figure 8. The proposed module for future work recommends a desired speed based on historical data analysis

Acknowledgements

The authors gratefully acknowledge the Cyber-Physical Systems Research Laboratory at the University of Isfahan for providing the facilities and resources essential to this research.

Funding

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Ethics Approval and Consent to Participate

This article does not involve any studies with human participants or animals performed by any of the authors. Therefore, ethics approval and consent to participate are not applicable.

Competing Interests

The authors declare no financial or organizational conflicts of interest.

Code Availability

The implementation codes for the proposed algorithm in this study are fully available via the following GitHub repository: [https://github.com/Cyber-Physical-Laboratory-FCE-UI/FTO].

Declaration of Generative AI and AI-assisted Technologies in the Writing Process

During the preparation of this manuscript, the authors used DeepSeek exclusively for minor language editing to improve readability. After using this tool, the authors carefully reviewed and edited the content as needed and take full responsibility for the content of the published article.

Authors Biosketch

Maytham Allahi Rudposhti received his B.Sc. and M.Sc. degrees in Computer Engineering from the University of Science and Technology of Mazandaran and Babol Noshirvani University of Technology, Iran, respectively. He is currently a Ph.D. candidate in Computer Engineering at the University of Isfahan, focusing on Autonomous Vehicles (AV) and Digital Twin (DT) technology.

Ali Bohlooli received the B.Sc. and M.Sc. degrees (Hons.) in computer engineering from the Department of Electrical and Computer Engineering, Isfahan University of Technology, Iran, in 2001 and 2003, respectively, and the Ph.D. degree in computer engineering from the University of Isfahan, Iran, in 2011. He is currently an Associate Professor with the Faculty of Computer Engineering, University of Isfahan. His research interests include cyber-physical systems, embedded artificial intelligence, and the Internet of Things (IoT).

Kamal Jamshidi received his Ph.D. in Electrical Engineering from the Indian Institute of Technology, India. He has been a faculty member of the Faculty of Computer Engineering at the University of Isfahan, Iran, since 1995. His academic background is in electrical engineering, and his teaching and research activities are carried out within the field of computer engineering.

REFERENCES

1. Pan JS, Hu P, Snášel V, Chu SC. A survey on binary metaheuristic algorithms and their engineering applications. Artificial Intelligence Review. 2023;56(7):6101-67. https://doi.org/10.1007/s10462-022-10328-9.
2. Zhao S, Zhang T, Ma S, Chen M. Dandelion optimizer: a nature-inspired metaheuristic algorithm for engineering applications. Engineering Applications of Artificial Intelligence. 2022;114:105075. https://doi.org/10.1016/j.engappai.2022.105075.
3. Kashani AR, Camp CV, Rostamian M, Azizi K, Gandomi AH. Population-based optimization in structural engineering: a review. Artificial Intelligence Review. 2022;55(1):345-452. https://doi.org/10.1007/s10462-021-10036-w.
4. Lameesa A, Hoque M, Alam MSB, Ahmed SF, Gandomi AH. Role of metaheuristic algorithms in healthcare: a comprehensive investigation across clinical diagnosis, medical imaging, operations management, and public health. Journal of Computational Design and Engineering. 2024;11(3):223-47. https://doi.org/10.1093/jcde/qwae046.
5. Bertsimas D, Tsitsiklis JN. Introduction to linear optimization. Athena Scientific; 1997.
6. Daoud MS, Shehab M, Al-Mimi HM, Abualigah L, Zitar RA, Shambour MKY. Gradient-based optimizer (GBO): a review, theory, variants, and applications. Archives of Computational Methods in Engineering. 2023;30(4):2431-49. https://doi.org/10.1007/s11831-022-09872-y.
7. Mirjalili S, Lewis A. The whale optimization algorithm. Advances in Engineering Software. 2016;95:51-67. https://doi.org/10.1016/j.advengsoft.2016.01.008.
8. Ahmadianfar I, Heidari AA, Gandomi AH, Chu X, Chen H. RUN beyond the metaphor: an efficient optimization algorithm based on Runge Kutta method. Expert Systems with Applications. 2021;181:115079. https://doi.org/10.1016/j.eswa.2021.115079.
9. Sowmya R, Premkumar M, Jangir P. Newton-Raphson-based optimizer: a new population-based metaheuristic algorithm for continuous optimization problems. Engineering Applications of Artificial Intelligence. 2024;128:107532. https://doi.org/10.1016/j.engappai.2023.107532.
10. Oladejo SO, Ekwe SO, Mirjalili S. The hiking optimization algorithm: a novel human-based metaheuristic approach. Knowledge-Based Systems. 2024;296:111880. https://doi.org/10.1016/j.knosys.2024.111880.
11. Ehsaeyan E. Rock-climbing group: an innovative meta-heuristic approach for efficiently tackling optimization problems. International Journal of Engineering Transactions B: Applications. 2025;38(11):2796-818. https://doi.org/10.5829/ije.2025.38.11b.24.
12. Ehsaeyan E. Gold seekers algorithm: an innovative metaheuristic approach for global optimization and its application in image segmentation. International Journal of Engineering Transactions C: Aspects. 2025;38(9):2114-29. https://doi.org/10.5829/ije.2025.38.09c.09.
13. Verij Kazemi M, Fazeli Veysari E. A new optimization algorithm inspired by the quest for the evolution of human society: human felicity algorithm. Expert Systems with Applications. 2022;193:116468. https://doi.org/10.1016/j.eswa.2021.116468.
14. Oladejo SO, Ekwe SO, Akinyemi LA, Mirjalili SA. The deep sleep optimizer: a human-based metaheuristic approach. IEEE Access. 2023;11:83639-65. https://doi.org/10.1109/ACCESS.2023.3298105.
15. Feng X, Zou R, Yu H. A novel optimization algorithm inspired by the creative thinking process. Soft Computing. 2015;19(10):2955-72. https://doi.org/10.1007/s00500-014-1459-6.
16. Kahrizi MR, Kabudian SJ. Projectiles optimization: a novel metaheuristic algorithm for global optimization. International Journal of Engineering Transactions A: Basics. 2020;33(10):1924-38. https://doi.org/10.5829/ije.2020.33.10a.11.
17. El-Shorbagy MA, Bouaouda A, Nabwey HA, Abualigah L, Hashim FA. Advances in Henry gas solubility optimization: a physics-inspired metaheuristic algorithm with its variants and applications. IEEE Access. 2024;12:26062-95. https://doi.org/10.1109/ACCESS.2024.3365700.
18. Azizi M. Atomic orbital search: a novel metaheuristic algorithm. Applied Mathematical Modelling. 2021;93:657-83. https://doi.org/10.1016/j.apm.2020.12.021.
19. Mahdavi-Meymand A, Zounemat-Kermani M. Homonuclear molecules optimization (HMO) meta-heuristic algorithm. Knowledge-Based Systems. 2022;258:110032. https://doi.org/10.1016/j.knosys.2022.110032.
20. Alba E, Dorronsoro B. Introduction to cellular genetic algorithms. Cellular Genetic Algorithms. 2008:3-20. https://doi.org/10.1007/978-0-387-77610-1_1.
21. Cheng MY, Prayogo D. Symbiotic organisms search: a new metaheuristic optimization algorithm. Computers & Structures. 2014;139:98-112. https://doi.org/10.1016/j.compstruc.2014.03.007.
22. Saremi S, Mirjalili S, Lewis A. Grasshopper optimisation algorithm: theory and application. Advances in Engineering Software. 2017;105:30-47. https://doi.org/10.1016/j.advengsoft.2017.01.004.
23. Talatahari S, Azizi M, Tolouei M, Talatahari B, Sareh P. Crystal structure algorithm (CryStAl): a metaheuristic optimization method. IEEE Access. 2021;9:71244-61. https://doi.org/10.1109/ACCESS.2021.3079161.
24. Zhang Q, Gao H, Zhan ZH, Li J, Zhang H. Growth optimizer: a powerful metaheuristic algorithm for solving continuous and discrete global optimization problems. Knowledge-Based Systems. 2023;261:110206. https://doi.org/10.1016/j.knosys.2022.110206.
25. Han M, Du Z, Yuen KF, Zhu H, Li Y, Yuan Q. Walrus optimizer: a novel nature-inspired metaheuristic algorithm. Expert Systems with Applications. 2024;239:122413. https://doi.org/10.1016/j.eswa.2023.122413.
26. Abdollahzadeh B, Khodadadi N, Barshandeh S, Trojovský P, Gharehchopogh FS, El-Kenawy ESM, et al. Puma optimizer (PO): a novel metaheuristic optimization algorithm and its application in machine learning. Cluster Computing. 2024;27(4):5235-83. https://doi.org/10.1007/s10586-023-04221-5.
27. Zhong C, Li G, Meng Z, Li H, Yildiz AR, Mirjalili S. Starfish optimization algorithm (SFOA): a bio-inspired metaheuristic algorithm for global optimization compared with 100 optimizers. Neural Computing and Applications. 2025;37(5):3641-83. https://doi.org/10.1007/s00521-024-10694-1.
28. Lara-Montaño OD, Gómez-Castro FI, Gutiérrez-Antonio C, Dragoi EN. Success-based optimization algorithm (SBOA): development and enhancement of a metaheuristic optimizer. Computers & Chemical Engineering. 2025;194:108987. https://doi.org/10.1016/j.compchemeng.2024.108987.
29. Amani B, Nouri M, Mousavi Ghasemi SA. Gray squirrel foraging algorithm for function optimization. International Journal of Engineering Transactions A: Basics. 2026;39(7):1644-56. https://doi.org/10.5829/ije.2026.39.07a.09.
30. Gandomi AH, Yang XS, Alavi AH. Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems.

Engineering with Computers. 2013;29(1):17-35. https://doi.org/10.1007/s00366-011-0241-y.

31. Falistocco E. The millenary history of the fig tree (Ficus carica L.). Advances in Agriculture Horticulture and Entomology. 2020;5:130. https://doi.org/10.37722/AAHAE.202051.
32. Houssein EH, Saad MR, Hashim FA, Shaban H, Hassaballah M. Lévy flight distribution: a new metaheuristic algorithm for solving engineering optimization problems. Engineering Applications of Artificial Intelligence. 2020;94:103731. https://doi.org/10.1016/j.engappai.2020.103731.
33. Singh S. Critical reasons for crashes investigated in the National Motor Vehicle Crash Causation Survey. National Highway Traffic Safety Administration; 2018. Report No.: DOT HS 812 506.
34. Liu Y, Liang Z, Zhong W, Xue Y, Wang Y, Tao N, et al. Multi-objective predictive cruise control for electric heavy-duty trucks considering fleet battery swapping under cyber-physical system. Energy. 2025;321:135462. https://doi.org/10.1016/j.energy.2025.135462.
35. Yu L, Wang R. Researches on adaptive cruise control system: a state of the art review. Proceedings of the Institution of Mechanical Engineers Part D: Journal of Automobile Engineering. 2021;236(2-3):211-40. https://doi.org/10.1177/09544070211019254.
36. Heybetli F, Danayiyen Y, Taşdemir AB, Şenyiğit Ş. Comparative analysis of metaheuristic algorithms in PID-based vehicle cruise control systems. Verus Journal. 2025;25(1):1-16. https://doi.org/10.5152/electrica.2025.25051.
37. Van Keulen T, Naus G, De Jager B, Van De Molengraft R, Steinbuch M, Aneke E. Predictive cruise control in hybrid electric vehicles. World Electric Vehicle Journal. 2009;3(1):494-504.

Persian Abstract

چکیده

این مقاله الگوریتم بهینه‌سازی درخت انجیر (FTO) را به عنوان یک فراابتکاری جدید الهام گرفته از طبیعت برای حل مسائل بهینه‌سازی پیچیده معرفی می‌کند که تمرکز آن بر کاربردهای خودرویی است. FTO با الهام از تکثیر پاجوش و پراکندگی بذر در درختان انجیر، این دو راهبرد جستجو را به صورت موازی و هم‌پوشان اجرا می‌کند که منجر به همگرایی سریع به سمت راه‌حل‌های نزدیک به بهینه‌ی جهانی می‌شود. کارایی FTO از طریق آزمایش‌های گسترده روی ۲۸ تابع محک، شامل ۱۶ تابع محک کلاسیک و ۱۲ تابع محک استاندارد، به طور سیستماتیک اعتبارسنجی شده است. عملکرد آن با چهار الگوریتم جدید مقایسه شده است: بهینه‌ساز رشد (GO)، بهینه‌ساز پوما (PO)، الگوریتم بهینه‌سازی مبتنی بر موفقیت (SBOA) و الگوریتم جستجوی غذای سنجاب خاکستری (GSFA). نتایج به طور مداوم برتری FTO را از نظر دقت، سرعت همگرایی و کیفیت راه‌حل نشان می‌دهند. کاربرد عملی FTO با تنظیم یک کنترل‌کننده PID دوسطحی در سیستم کنترل کروز تطبیقی (ACC) ارزیابی شد و عملکرد آن با الگوریتم PO در سه حالت مقایسه گردید. در حالت اول، با زمان محاسباتی مشابه PO، میانگین بهترین هزینه ٤ برابر بالاتر بود که آن را برای سناریوهای با محدودیت انرژی و نیازهای دقت پایین مناسب می‌سازد. در حالت دوم، زمان اجرای ۳۳ درصد طولانی‌تر منجر به بهبود عملکرد ۱.۵ برابری شد که برای موارد با انرژی و زمان محدود اما نیازمند دقت بالا مناسب است. در حالت سوم، افزایش ۱٤۳ درصدی زمان اجرا (که تحت اجرای موازی تا ۳۳ درصد قابل کاهش است)، بازده کلی را ۸.۱۵ برابر بهبود بخشید و برای زمانی مناسب است که منابع کافی در دسترس بوده و دقت بالا مورد نیاز است. FTO یک ابزار کارآمد برای چالش‌های شرایط محک و دنیای واقعی است.