www.editada.org

International Journal of Combinatorial Optimization Problems and Informatics, 17(3), 2026, 261-286. ISSN: 2007-1558. https://doi.org/10.61467/2007.1558.2026.v17i3.1377

A Hybrid Bio-Inspired Roach Infestation Algorithm with Interval Type-2 Fuzzy Adapter for Large-Scale Optimization: An extensive Evaluation

Enrique Lizárraga, Fevrier Valdez, Oscar Castillo and Patricia Melin
[1] Tijuana Institute of Technology TecNM, Division of Graduate Studies and Research, Tijuana, 22414, México.
Email address(es): arcenio.lizarraga201@tectijuana.mx, fevrier@tectijuana.mx, ocastillo@tectijuana.mx, pmelin@tectijuana.mx
✉Corresponding author:

Abstract. The Swarm-based metaheuristic algorithms have been widely adopted for solving large-scale optimization problems due to their flexibility and population-based search mechanisms. Within this family, Roach Infestation Optimization (RIO) leverages collective interactions and hunger-related dynamics to promote global exploration. Despite these advantages, its search behavior is highly sensitive to parameter settings, especially in high-dimensional spaces, where an improper balance between exploration and exploitation can reduce convergence efficiency. This study presents an adaptive strategy based on Interval Type-2 Fuzzy Logic (IT2FL) to regulate the key RIO parameters, C_maxand C_o, according to the evolutionary state of the search process. By explicitly modeling uncertainty in the parameter adaptation mechanism, the proposed framework enables smoother and more robust adjustments compared to static configurations and Type-1 fuzzy approaches. The proposed method is evaluated on large-scale benchmark optimization problems with dimensionalities ranging from 50 to 500. Its performance is systematically compared to the original RIO algorithm, a Type-1 fuzzy-enhanced RIO variant, Cuckoo Search, and Particle Swarm Optimization (PSO). Experimental results demonstrate consistent improvements in solution quality, convergence stability, and robustness across different problem instances. Furthermore, the statistical significance of the observed improvements is validated through statistical analysis based on Z-tests, confirming the effectiveness of the proposed IT2FL adaptation in high-dimensional optimization scenarios.

Keywords: Roach Infestation Optimization; parameter adaptation; interval type-2 fuzzy logic; Adaptive hybrid metaheuristics

Article information
Received: March 20, 2026
Accepted: April 14, 2026

1 Introduction

The performance of metaheuristic algorithms depends heavily on the configuration of their control parameters, which are usually static. While static parameters may provide good performance for a specific set of problems, they do not adapt well to changes, multimodal problems, or noisy problems, compromising the balance between exploration and exploitation necessary for efficient search (Brindha & Joe Amali, 2021).

To reduce this dependency, parameter adaptation has become a necessary strategy. Fuzzy parameter adaptation has been widely used to formalize heuristic rules based on behavioral indicators such as population diversity, improvement rate, or current iteration, allowing for smooth adjustments to the algorithm's parameters. This approach has shown improvements in stability and robustness in several metaheuristics (Barraza et al., 2024).

However, Type-1 fuzzy systems have limitations in modeling the uncertainty associated with membership functions (MFs). Some examples include uncertainty arising from noise in the evaluation of the objective function or imprecise estimates of population

indicators. Interval Type-2 fuzzy logic (IT2FLS) extends the representation of uncertainty by allowing a region of uncertainty in the MFs, thus increasing the capacity to handle imprecise information (Almaraashi, 2024).

Despite their superior modeling capabilities, the use of IT2FLS has traditionally been limited by their higher computational cost and the complexity associated with their design. However, recent research has proposed practical improvements and learning schemes that mitigate these limitations, showing that IT2FLS can be efficiently applied to optimization and control problems when properly designed and implemented (Koklu et al., 2024; Miramontes & Melin, 2022; Nishanth et al., 2024).

On the other hand, the RIO algorithm has demonstrated high exploratory capacity in complex search spaces; however, its performance is sensitive to parameter settings, which can lead to performance losses during fine-tuning stages. These characteristics make RIO a suitable candidate for incorporating an IT2FLS based parameter tuning scheme (Havens et al., 2008), since this type of system allows for the explicit handling of uncertainty associated with noisy or imprecise evolutionary indicators. In this way, some of the limitations of Type-1 fuzzy systems in dynamic and stochastic environments are overcome (Cara et al., 2013; Yuan et al., 2021), also favoring an adaptive balance between exploration and exploitation (Hepworth et al., 2022; Liu et al., 2022).

This paper proposes a dynamic parameter adaptation scheme for the RIO algorithm based on a Type-2 interval fuzzy system, an approach widely used to modify parameters in metaheuristic algorithms and to manage the uncertainty inherent in dynamic optimization processes (Jahanshahi et al., 2022; Lizarraga et al., 2025). The fuzzy system is activated periodically every five iterations to reduce computational overhead while maintaining its direct integration within the RIO algorithm's evolutionary cycle (Tsai, 2015). This mechanism allows the c_{max} and c_0 parameters to be adjusted according to the search process state and the population diversity level, a criterion commonly used to balance exploration and exploitation processes in bio-inspired algorithms (Lizarraga et al., 2025).

This article is organized as follows: Section 2 reviews related work, including parameter adaptation in metaheuristics, fuzzy systems in evolutionary algorithms, and Type-2 fuzzy logic in optimization. Section 3 describes the fundamentals of RIO algorithm, its biologically inspired behaviors, key parameters, and limitations. Section 4 presents interval-based Type-2 fuzzy logic systems for dynamic parameter adaptation, detailing the motivation, architecture, input and output variables, MFs, rule base, and inference mechanism. Section 5 introduces the proposed IT2F-RIO algorithm, describing its general flow, pseudocode, and integration into the RIO framework. Section 6 presents the experimental setup, reference functions, evaluation metrics, and comparison schemes. Section 7 present the computational complexity and scalability analysis, Section 8 presents the results, while Section 9 discusses the findings and performance analysis. Finally, Section 10 concludes the paper and suggests directions for future research.

2 Related Work

This section reviews the main approaches reported in the literature related to improving the performance of metaheuristic algorithms. In particular, emphasis is placed on parameter tuning strategies, given their direct influence on the balance between exploration and exploitation throughout the search process. The review initially focuses on parameter tuning methods and subsequently addresses more advanced approaches, such as hybridization and the use of fuzzy systems.

2.1. Parameter adaptation in metaheuristics

Parameter adaptation in metaheuristic algorithms has become a key strategy for improving their performance in complex, dynamic, or large-scale optimization problems. In many classical approaches, control parameters remain fixed throughout execution, which may be suitable for certain types of problems, but limits the algorithm's ability to respond to changes in the dynamics of the search space. Several studies have indicated that this lack of flexibility can negatively affect the balance between exploration and exploitation, even favoring premature convergence (Johnvictor et al., 2022; Selvarajan, 2024).

To overcome these limitations, the literature has proposed different parameter adaptation schemes that allow for the modification of critical values during algorithm execution based on the state and progress of the search. Generally, these schemes can be grouped into deterministic, adaptive, and self-adaptive methods. Among these, adaptive approaches have shown a favorable relationship between simplicity and effectiveness, as they adjust parameters using information from the search process itself without significantly increasing the algorithm's structural complexity (Dragoi & Dafinescu, 2021; Parouha & Verma, 2021a).

Parameter adaptation has been explored in various bio-inspired metaheuristics, with particular emphasis on genetic algorithms and particle swarm optimization. Comparative studies report that methods with adaptive parameters often offer consistent improvements in solution quality, process stability, and convergence speed compared to their fixed parameter versions (Chen et al., 2021; Gad, 2022; Wang et al., 2021). These improvements are especially noticeable in multimodal and high dimensional problems, where exploration and exploitation needs vary throughout the evolutionary process.

Another relevant line of research within this field focuses on the use of population state indicators to guide adjustment mechanisms. In particular, metrics such as population diversity have been widely used to characterize the collective behavior of the algorithm and identify transitions between predominantly exploratory and exploitative phases. Using this type of information allows for adjustments that are more consistent with the algorithm's internal dynamics, contributing to a more balanced and robust search (Milner et al., 2023; Oliveira et al., 2020).

Overall, the reviewed works demonstrate that dynamic parameter adaptation is a key component for improving the performance and robustness of metaheuristic algorithms. However, much of the literature has focused on classical algorithms and general adaptation schemes, which opens the possibility of exploring more specialized adjustment mechanisms adapted to the particular characteristics of less studied metaheuristics (Sutikno, 2023).

2.2 Fuzzy systems in evolutionary algorithms

A relevant adaptive approach is the use of dynamic search process metrics, such as population diversity, the rate of improvement of the objective function, or convergence-related indicators. These metrics allow the identification of different phases of the evolutionary process and the adjustment of parameters to favor a dynamic balance between exploration and exploitation (Johnvictor et al., 2022). Several studies have shown that this type of adaptation significantly improves the performance of algorithms in multimodal and high dimensional problems, where the optimal parameter values vary throughout execution (Ibrahim et al., 2024; Parouha & Verma, 2021b).

In this context, fuzzy systems have been proposed as efficient mechanisms for parameter adjustment due to their ability to model nonlinear relationships and handle imprecise information. Initially, Type-1 fuzzy systems were used to adjust control parameters based on search process indicators, showing improvements in stability and solution quality (Lizarraga et al., 2025). However, these approaches can be limited in highly dynamic or noisy scenarios. To overcome these limitations, Type-2 interval fuzzy systems have been introduced, which provide a more flexible representation of uncertainty and have demonstrated better performance in complex optimization problems (Paz et al., 2021).

Despite the advances reported in the literature, many adaptation schemes make adjustments in each iteration or incorporate mechanisms external to the evolutionary cycle, which increases the computational cost of the algorithm. This has motivated the development of more efficient strategies, where the adaptation process is integrated directly into the dynamics of the algorithm and is activated periodically or conditionally, allowing for a suitable compromise between performance and computational efficiency (Valdez et al., 2021).

2.3 Hybridization of evolutionary algorithms

The hybridization of metaheuristic algorithms has emerged as an effective strategy to overcome the inherent limitations of individual approaches, particularly in complex, high dimensional optimization problems. In general terms, hybridization consists of combining two or more algorithms, or incorporating external mechanisms within a base algorithm, to leverage the complementary strengths of each component and compensate for their limitations (Chen & Shang, 2021). This approach has proven particularly effective in improving convergence, reducing the risk of premature stagnation, and enhancing the quality of the resulting solutions.

Broadly speaking, hybridization schemes can be classified into sequential, parallel, and hierarchical approaches. In sequential hybridization, the algorithms are executed consecutively, using the solution resulting from one phase as the starting point for the next. Parallel approaches, on the other hand, allow the simultaneous execution of multiple algorithms, facilitating information exchange between them, while hierarchical schemes integrate different strategies within the same control structure. These types of configurations have been successfully employed to achieve a more effective balance between exploration and exploitation compared to individual methods (Xiang et al., 2021).

The most relevant hybridizations typically focus on combining algorithms with complementary search capabilities. In particular, hybrid approaches have been proposed that integrate algorithms with a strong exploratory capacity alongside others geared towards local exploitation, allowing for a more balanced exploration of the solution space. These types of combinations have shown significant improvements in both convergence speed and the stability of results, especially in multimodal problems, where population diversity tends to decrease rapidly (Vidal-Martínez et al., 2025).

More recently, hybridization research has evolved towards more flexible and dynamic schemes, in which the interaction between the different components is not fixed but adapts according to the state of the optimization process. These adaptive approaches allow hybrid mechanisms to be activated or deactivated based on indicators of algorithm performance, reducing computational overhead and improving overall efficiency. These types of strategies have proven to be particularly useful when combined with parameter adaptation mechanisms (Torres-Salinas et al., 2022).

3 Roach Infestation Optimization Algorithm

The RIO algorithm belongs to the family of bio-inspired metaheuristics and is based on modeling the social behavior of cockroaches, particularly their tendency to group together, explore their environment, and respond to stimuli related to shelter and survival. Since its introduction, RIO has been applied to various optimization and control problems, demonstrating a strong exploratory capacity derived from the collective dynamics of the agents and the interaction between individual and social behaviors (Pradhan et al., 2022).

3.1 Basic Concepts

RIO's operation is based on the simulation of a cockroach population interacting with each other and with the search environment. Each individual represents a candidate solution whose movement is governed, firstly, by the velocity update, described in Equation (1), incorporating local and global system information. Subsequently, the position update of each agent is performed based on this velocity, as shown in Equation (2). The algorithm integrates behaviors inspired by cockroach biology, such as shelter seeking, social aggregation, and hunger response, which directly influence the dynamics of solution space exploration (Le et al., 2021).

$$V_{i+1} = C_0 V_i + C_{max} R_1 (p_i - x_i) + C_{max} R_2 (l_i - x_i) \quad (1)$$

$$x_{i+1} = \begin{cases} x_i + V_{i+1}, & if\left(hunger_i < T_{hunger}\right) \\ b & if\ hunger_i = T_{hunger} \end{cases} \quad (2)$$

Where V_i is the agents velocity, x_i is the current location found by agent i, p_i is the best location found by swarm i, l_i is the darkest location known to agent i after socializing, C_0 is the momentum terms adjustment, C_{max} is the cognitive and social adjustment factor, R_1 and R_2 are uniformly distributed random number vectors, $hunger_i$ is an incrementing hunger counter that identifies a cockroachs current hunger level, T_{hunger} is the hunger threshold and b is a random position within the solution space.

During the evolutionary process, cockroaches tend to cluster in promising regions of the search space, favoring the exploitation of high-quality solutions, while dispersal mechanisms allow them to maintain exploration and avoid premature stagnation. This combination of behaviors gives RIO a natural balance between exploration and exploitation, differentiating it from other metaheuristics based solely on mathematical or probabilistic rules (Le et al., 2021; Obagbuwa & Adewumi, 2014).

3.2 Key parameters

The performance of the RIO algorithm depends largely on the proper configuration of its control parameters, which regulate both the intensity of social interaction and the individual behavior of the cockroaches during the search process. Among the most relevant parameters are those associated with social influence, movement dynamics, and hunger related mechanisms, commonly represented by constants such as c_0 and c_{max}. The random term, modulated by c_0, encourages the exploration of new regions of the search space by introducing variability in the movement of individuals, especially in the initial stages of the algorithm. Meanwhile, the social component, regulated by c_{max}, drives the cockroaches to group around promising solutions, promoting aggregation behavior that emulates their natural behavior of concentrating in safe or favorable areas. This mechanism primarily contributes to the exploitation of the solution space (Asianuba & Precious, 2023).

Taken together, these parameters determine the balance between the tendency of individuals to follow the best agents in the group and the algorithm's ability to explore unvisited regions of the search space. An inappropriate configuration can lead to premature convergence or, at the opposite extreme, to excessive exploration that slows down the optimization process. Several studies have indicated that manually adjusting these parameters is highly dependent on the problem and the application environment, which limits the algorithm's robustness in dynamic or multimodal scenarios (Obagbuwa & Adewumi, 2014; Pradhan et al., 2022).

3.3 Limitations

Despite its advantages, the RIO algorithm has some inherent design limitations. One of the most significant is the use of fixed control parameters during execution, which reduces its ability to adapt to changes that occur throughout the search process. In complex, dynamic, or multimodal optimization problems, this lack of flexibility often translates into a gradual loss of population diversity and a greater risk of premature convergence to local optima (Obagbuwa & Adewumi, 2014).

Additionally, RIO exhibits high sensitivity to parameter settings, which considerably increases the effort required during the tuning stage. This situation becomes even more critical when the algorithm is applied to problem domains different from those for which it was originally designed, limiting its generalizability and applicability in practical scenarios. Consequently, RIO's performance can vary significantly across different problems, making it difficult to predict its behavior without resorting to extensive experimentation.

These limitations have motivated the development of modified variants, hybrid approaches and adaptive mechanisms designed to dynamically adjust the control parameters based on the behavior of the algorithm and the characteristics of the search environment (Cheng et al., 2023).

4 Interval Type-2 Fuzzy Adaptation

The increasing complexity of optimization problems has highlighted the limitations of static parameterization schemes in metaheuristic algorithms. As the search process evolves, exploration and exploitation need change, necessitating the incorporation of adaptation mechanisms capable of responding to these variations. In this context, fuzzy logic has become established as an effective tool for modeling uncertainty and heuristic knowledge in parameter tuning schemes, especially in dynamic and nonlinear environments (Hu et al., 2024).

4.1 Motivation for Fuzzy-Based Parameter Adaptation in Metaheuristics

Fuzzy logic based parameter adaptation has been addressed in the literature as a mechanism for translating qualitative information about the algorithm's state into quantifiable control actions. In this approach, indicators such as search progress, solution stability, and agent interaction are expressed using linguistic variables, allowing for the capture of nonlinear relationships that are difficult to model using traditional analytical techniques (Wang et al., 2022).

Several studies have demonstrated that fuzzy logic facilitates the incorporation of expert heuristic knowledge into the optimization process, enabling the definition of rules that relate the observed behavior of the algorithm to gradual adjustments in its control parameters. This strategy has been successfully applied in evolutionary and collective intelligence algorithms, where the parameters directly influence population dynamics and convergence speed (Hasan et al., 2021; Too & Abdullah, 2021).

In particular, fuzzy adaptation has been used to regulate factors associated with exploration and exploitation, such as social coefficients, learning rates, and perturbation probabilities. These schemes have shown consistent improvements in terms of stability and performance, especially in complex problems where the search process requirements change during execution (Bensadok & Babar, 2024; Singh et al., 2011).

Additionally, recent work has incorporated fuzzy systems as external control modules, decoupled from the core of the metaheuristic algorithm. This modular architecture allows the fuzzy adapter to be reused in different algorithms and facilitates its extension to hybrid or multi adaptive schemes without modifying the optimizer's internal structure (Duan et al., 2023; Franklin et al., 2024).

4.2 Interval Type-2 Fuzzy Logic Systems

IT2FLS are an extension of Type-1 fuzzy systems, designed to explicitly handle the uncertainty associated with defining MFs and the inherent imprecision of expert knowledge. Unlike Type-1 systems, where the degree of membership is a single point value, in Type-2 systems it is represented by an interval, allowing for a more realistic modeling of the variability and noise present in dynamic and non-deterministic environments (Jafari, 2024).

A distinctive feature of IT2FLS is the incorporation of the Footprint of Uncertainty (FOU), which delimits the range of possible values for the MFs. This mechanism provides greater flexibility in representing linguistic concepts, reducing the system's sensitivity to disturbances or errors in rule definition. Several studies have demonstrated that the use of fuzzy unit arrays FOU's improves the robustness of fuzzy systems in the face of abrupt changes in the environment and incomplete or imprecise information (Too & Abdullah, 2021).

In the context of optimization and collective intelligence, IT2FLS have been employed as adaptive control mechanisms to regulate critical parameters of metaheuristic algorithms. Their ability to handle uncertainty makes them particularly suitable for scenarios where the algorithm's behavior varies throughout the search process and where the relationships between variables are not strictly linear. Recent applications have shown significant improvements in stability, performance, and generalizability when integrating Type-2 fuzzy systems into parameter adaptation schemes (Guerrero et al., 2022; Takahashi & Takahashi, 2021).

Another relevant aspect of IT2FLS is the type reduction process, whereby the Type-2 fuzzy output is transformed into a Type-1 set before defuzzification. This step allows for obtaining concrete control values without losing the information associated with the uncertainty modeled during inference. Various type reduction methods exist, which aim to balance accuracy and computational cost, the latter being a key factor when fuzzy systems are integrated into iterative optimization algorithms (Starczewski et al., 2022).

Taken together, interval Type-2 fuzzy logic systems offer a suitable framework for designing robust adaptive mechanisms capable of capturing the uncertain and changing nature of optimization processes. These properties make them a particularly relevant tool for the dynamic adaptation of parameters in population metaheuristics, laying the groundwork for their integration with bio-inspired algorithms such as RIO.

4.3 Advantages of Interval Type-2 Fuzzy Systems in Dynamic Optimization

IT2FL offer advantages when applied to dynamic optimization problems, where the algorithm's behavior and the characteristics of the search space can vary unpredictably over time. In these scenarios, IT2FLS's ability to explicitly model uncertainty allows for a more stable response to fluctuations in the search process indicators (Ebrahim, 2024).

One of the main advantages of IT2FLS is its greater robustness against noise and imprecision in the input variables. Compared to Type-1 systems, Type-2 systems reduce the sensitivity of the adaptive mechanism to small variations or erroneous measurements, which is particularly relevant in population metaheuristics where metrics such as diversity or improvement rate can exhibit highly nonlinear behavior (Yiğit, 2023).

Furthermore, IT2FLS allows for smoother transitions between exploration and exploitation strategies, preventing abrupt changes in control parameters. This property is essential for maintaining the stability of the optimization process and preventing phenomena such as premature convergence or abrupt loss of diversity. Recent studies have shown that this type of gradual adaptation improves both the quality of the final solutions and the consistency of performance across different runs (Valdez et al., 2021).

Finally, the structural flexibility of IT2FLS facilitates their integration with different metaheuristic algorithms without requiring deep modifications to their core. This has driven their adoption as external control modules or adapters, capable of operating periodically or continuously within the algorithm's evolutionary cycle, maintaining a suitable balance between performance and computational cost (Amador-Angulo et al., 2022).

4.4 General Architecture of an Interval Type-2 Fuzzy Adapter

IT2FLS extends the concept of Type-1 fuzzy systems by explicitly modeling the uncertainty associated with MFs which is particularly suitable for dynamic environments and evolutionary optimization processes. In this context, a Type-2 interval fuzzy set (Patel, 2024).

is formally defined in the Equation (3).

$$\tilde{A} = \{(x, \mu_{\tilde{A}}(x)) \mid x \in X, \mu_{\tilde{A}}(x) \in [\underset{\sim}{\mu}_{\tilde{A}}(x), \bar{\mu}_{\tilde{A}}(x)] \subseteq [0,1]\} \quad (3)$$

where X denotes the universe of discourse, $x \in X$ represents a specific element of that universe, and $\tilde{A}$ denotes an interval Type-2 Fuzzy set. The functions $\underset{\sim}{\mu}_{\tilde{A}}(x)$ and $\bar{\mu}_{\tilde{A}}(x)$ represent the lower and upper MFs, respectively. The interval defined between these two limits constitutes the FOU, which allows capturing variations, noise, and inaccuracies inherent in the decision-making process (Paik & Mondal, 2021).

Under this formulation, the general architecture of a Type-2 interval fuzzy adapter for parameter adaptation in metaheuristic algorithms consists of five main functional blocks: fuzzification, rule base, inference engine, type reduction, and defuzzification. Unlike Type-1 systems, fuzzification in an IT2FLS assigns each input a membership degree interval, increasing the system's robustness to incomplete or fluctuating information (Patel, 2024).

The fuzzy rule base is constructed using antecedents and consequents expressed as Type-2 interval fuzzy sets, generally under an IF THEN structure. During the inference process, these rules are activated by simultaneously considering the lower and upper bounds of the MFs, resulting in an output fuzzy set that is also Type-2 interval fuzzy. This mechanism allows for a more accurate reflection of the uncertainty present in the evaluation of the optimization algorithm's state (Cuevas et al., 2022; Kasmi & Hassam, 2021).

Subsequently, the output fuzzy set must undergo a type reduction process, the objective of which is to transform the Type-2 interval fuzzy set into an equivalent Type-1 set. This step is fundamental, as it allows for obtaining a manageable numerical representation from the FOU. Several methods have been proposed in the literature for this stage, the most common being those based on calculating the lower and upper centroids of the reduced set (Celik, 2024).

Finally, the defuzzification process converts the resulting Type-1 set into a crisp value, which is used directly to dynamically adjust the control parameters of the metaheuristic algorithm, such as C_{max} or exploration and exploitation factors. Thanks to this architecture, the interval Type-2 fuzzy adapter is able to respond more stably and flexibly to changes in the behavior of the search process, overcoming the limitations of schemes based on static parameters or Type-1 fuzzy systems (Vincent & Jidesh, 2023).

5 Proposed Method

An adaptive extension of the RIO algorithm is proposed by incorporating an IT2FLS that dynamically adjusts a control parameter of the algorithm in each iteration. The motivation for this integration is to provide RIO with an automatic parameter adjustment mechanism, responding in real time to quantitative information about the evolutionary state of the solution population.

In each iteration of the modified algorithm, after evaluating the current population, two normalized metrics are computed in the interval (0,1): Normalized Iteration, defined as the proportion of elapsed iterations relative to the total, and Population Diversity, understood as a quantitative measure of the dispersion of solutions in the search space.

These metrics feed into the IT2FLS, which is structured with fuzzy rules and MFs that include an uncertainty FOU to capture the uncertainty inherent in the search dynamics. The output of the fuzzy system is a continuous adjustment value that redefines the selected control parameter for the next iteration of the RIO, maintaining adaptability without manual intervention.

The inference process is based on a Type-2 interval Mamdani scheme, and type reduction produces a numerical output value directly applicable to the algorithm. In this sense, the use of IT2FLS for dynamic parameter adaptation has been reported as an effective strategy in other metaheuristic algorithms, improving the search without requiring prior adjustment of fixed parameters throughout execution. For example, IT2FLS approaches have been proposed for parameter adaptation in optimization algorithms such as the Whale Optimization Algorithm applied to benchmark mathematical functions (Ma et al., 2024), and reviews have recognized that Type-2 systems offer advantages in handling uncertainty in parameter adaptation compared to Type-1 systems (Mittal et al., 2020).

Fig. 1 shows the general flow of the RIO–IT2FLS algorithm, where an external fuzzy adapter is periodically activated to adjust the RIO parameters based on population diversity and iterative progress, returning the adapted values to the optimization process.

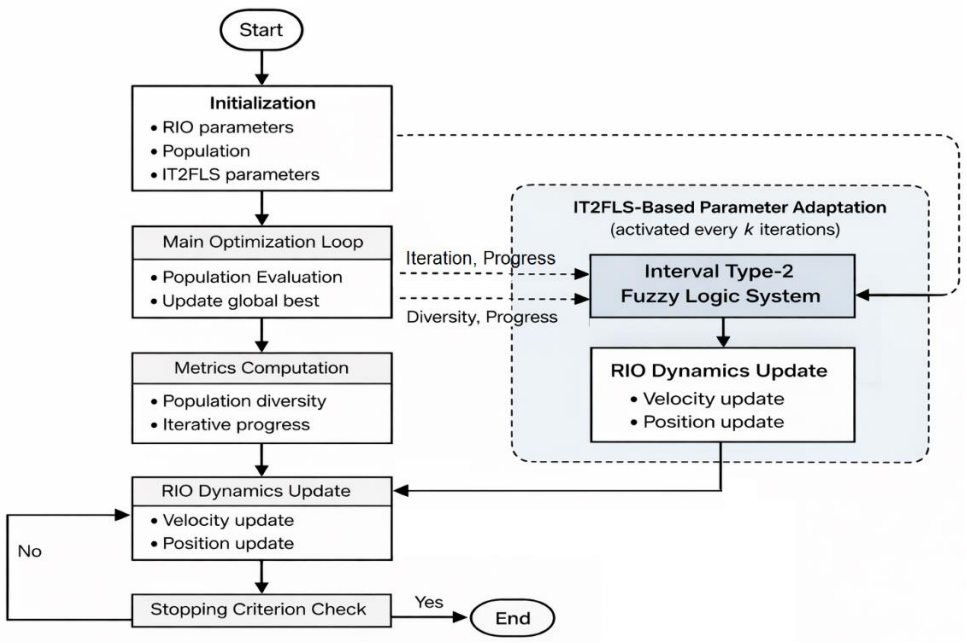

Fig. 1. Integration of the IT2FLS adapter into the RIO algorithm.

Fig. 1 illustrates the general workflow of the proposed RIO–IT2FLS algorithm. It shows how a IT2FLS is implemented as an independent adaptive module based on population diversity and iterative progress. Once the adaptation process is complete, the updated parameters are reintroduced into the main optimization loop, preserving the original structure of the RIO algorithm and improving its adaptive behavior. The adaptation mechanism is activated every five iterations within the main flow of the algorithm (Algorithm 1). In each activation, the algorithm's state metrics are evaluated by the fuzzy system, producing an adjustment factor that is applied directly to the algorithm's control parameters, immediately influencing its search dynamics without modifying its original structure.

Algorithm 1: Roach Infestation Optimization with Interval Type-2 Fuzzy Adapter
Input: Objective function f(x), search bounds, population size nPop,maximun iterations MaxIt
Output: Best solution found, final population.
1: Initialize roach population positions randomly within bounds
2: Evaluate fitness of each roach
3: Determine GlobalBest
4: diversityHistory ← ∅
5: Load interval Type-2 fuzzy system: IT2FIS
6: for it = 1 to MaxIt do
7: norm_it ← it / MaxIt
8: // Interval Type-2 fuzzy adaptation
9: if mod(it, 5) == 0 then
10: diversity ← CALCULATE_DIVERSITY(population)
11: norm_div, diversityHistory ← NORMALIZE_DIVERSITY(diversity, diversityHistory)
12: adapt_factor ← EVAL_IT2FIS([norm_it, norm_div], IT2FIS)
13: Update exploration/exploitation parameters using adapt_factor
14: end if
15: // Standard RIO behaviors
16: Apply chase and aggregation behavior
17: Apply dispersion behavior if required
18: Apply cannibalism behavior
19: Enforce boundary constraints
20: Evaluate fitness of updated population
21: Update GlobalBest
22: end for
23: return GlobalBest, population

6 Interval Type-2 Fuzzy Adapter

Population diversity is used as an indicator of the dynamic state of the search, as it reflects the degree of dispersion of candidate solutions in the search space. To capture its behavior more stably and reduce the influence of instantaneous fluctuations, diversity is calculated using a recent history of values and then normalized using a fixed size sliding window over 20 iterations. This scheme allows for a smoothed and consistent representation of the level of exploration or exploitation present in the population at each stage of the process (Mwaura et al., 2021).

Iterative progress, on the other hand, represents the relative advancement of the algorithm within the optimization process. This indicator is normalized to a range of 0 to 1, where values close to zero correspond to the initial stages of the search and values close to one indicate advanced stages of the process. In this way, the fuzzy adapter has explicit temporal information about the algorithm's operational stage, allowing for adjustments to the parameters according to the exploration or refinement phase. Additionally, population diversity is quantified using Equation (4), which measures the dispersion of solutions within the population and provides an estimate of the balance between exploration and exploitation by the algorithm.

$$d = \frac{1}{n_s D} \sum_{i=1}^{n_s} \sum_{j=1}^{D} \left| x_{ij}(t) - \overline{X}_j(t) \right| \tag{4}$$

Where n_s is the number of individual solutions in the population, D is the number of decision variables or dimensions, $x_{ij}(t)$ is the value of the j-th variable por the i individual at time t , $\overline{X}_j(t)$ is the average value of the j-th variable across the entire population at time t. In order to obtain a stable and bounded input signal for the fuzzy system, the calculated diversity is dynamically normalized using a sliding window of size W as shown in Equation (5).

$$D_{norm}(t) = \frac{D(t) - \min\left(h(t)\right)}{max\left(h(t)\right) - \min(h(t)) + \varepsilon} \tag{5}$$

Where D(t) is the population diversity at iteration t computed using Equation (4) ,$h(t)$ is the diversity history stored within a sliding window of size W,$min(h(t))$ and $max(h(t))$ are the minimum and maximum diversity values recorded in that window, ε is a small positive constant used to avoid division by zero, and $D_{norm}(t)$ is the normalized diversity value used as input to the fuzzy system.

6.1 Fuzzy system architecture

Trapezoidal MFs were used for the fuzzy sets due to their computational efficiency, robustness to small input variations, and ease of parameter tuning, making them ideal for adaptive mechanisms embedded in metaheuristic algorithms. Compared to triangular functions, trapezoidal shapes provide a complete membership plateau that reduces sensitivity to noise, while avoiding the higher computational cost and greater tuning complexity associated with Gaussian functions. This choice is consistent with recent studies indicating that simple shapes like trapezoidal ones are preferable in applications where efficiency and robustness are critical for the performance of the fuzzy system and its integration into optimization and adaptive inference environments (Anari et al., 2022). The general architecture of the ITFLS system is shown in Fig. 2.

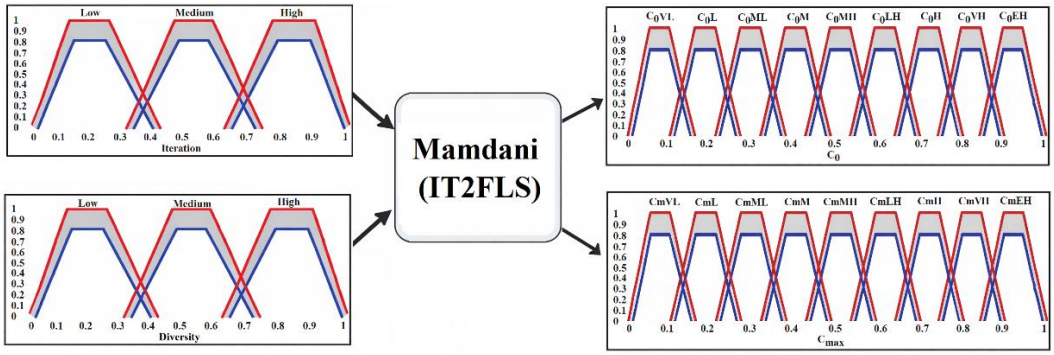

Fig. 2. IT2FLS Fuzzy Inference System Architecture.

Fig. 2 presents the general architecture of the Mamdani IT2FLS. The system considers two input variables, corresponding to iterative progress and population diversity, represented by fuzzy sets defined in a normalized domain. These inputs are processed by a Mamdani inference engine (Tang & Ahmad, 2024), which generates two output variables associated with the control parameters C_{max} and C_0. The relationship between the input variables, the inference mechanism, and the adapter's outputs is illustrated globally, as well as the general organization of the fuzzy sets used.

6.1.1. Input membership functions

Each input variable was modeled using three linguistic terms: Low, Medium, and High. The associated fuzzy sets were defined uniformly across the input domain, ensuring adequate overlap between adjacent regions and a smooth transition between the different linguistic states.

This configuration effectively captures the algorithm's temporal evolution and variations in population dispersion, providing the fuzzy system with relevant information for decision making in adapting control parameters (Dutta et al., 2025). The specific values of the MFs are presented in detail in Table 1.

Table 1. Membership values of the input variables

MFs	a	b	c	d	λ	ℓ_1	ℓ_2
LI (Low iteration)	0.10	0.10	0.30	0.47	0.80	0.20	0.20
MI (Medium iteration)	0.22	0.10	0.58	0.73	0.80	0.20	0.20
HI (High iteration)	0.53	0.72	0.88	1.10	0.80	0.20	0.20
LD (Low diversity)	0.00	0.10	0.30	0.47	0.80	0.20	0.20
MD (Medium diversity)	0.22	0.40	0.58	0.73	0.80	0.20	0.20
HD (High diversity)	0.53	0.72	0.58	1.10	0.80	0.20	0.20

Table 1 presents the numerical values assigned to the parameters that define the MFs of the input variables. A structural correspondence is maintained between the sets associated with iterative progress and population diversity, allowing for consistent system behavior under different optimization process conditions. The parameter λ, associated with the size of the FOU, was determined through exhaustive experimentation, evaluating increments of 0.1 within the allowed range, and selecting the value that showed the most stable system performance. The remaining parameters related to uncertainty representation were set uniformly for all sets, ensuring homogeneity in the fuzzy modeling.

6.1.2. Output membership functions

The output variables of the fuzzy adapter correspond to the control parameters C_{max} and C_0, which directly influence the dynamic behavior of the algorithm during the optimization process. These variables were defined within a normalized domain to ensure consistency in the inference process and stability in the response of the fuzzy system.

The MFs associated with the output variables were designed to allow gradual modulation of the parameter values, facilitating smooth transitions between different levels of adjustment and avoiding abrupt changes that could affect the convergence of the algorithm. The numerical values corresponding to these MFs are presented in detail in Table 2.

Table 2. Membership values of the output variables

Rule	a	b	c	d	λ	ℓ_1	ℓ_2
C_0 Very low	0.64	0.65	0.66	0.67	0.80	0.15	0.15
C_0 Low	0.66	0.67	0.68	0.69	0.80	0.15	0.15
C_0 Moderately low	0.68	0.69	0.70	0.71	0.80	0.15	0.15
C_0 Medium low	0.70	0.71	0.72	0.73	0.80	0.15	0.15
C_0 Medium	0.72	0.73	0.74	0.75	0.80	0.15	0.15
C_0 Medium high	0.73	0.74	0.75	0.76	0.80	0.15	0.15
C_0 Slightly high	0.75	0.76	0.77	0.78	0.80	0.15	0.15
C_0 High	0.77	0.78	0.79	0.80	0.80	0.15	0.15
C_0 Very high	0.79	0.80	0.81	0.82	0.80	0.15	0.15
C_{max} Very low	0.97	0.98	0.99	1.00	0.80	0.15	0.15
C_{max} Low	0.99	1.00	1.01	1.03	0.80	0.15	0.15
C_{max} Moderately low	1.01	1.03	1.04	1.05	0.80	0.15	0.15
C_{max} Medium low	1.04	1.05	1.06	1.07	0.80	0.15	0.15
C_{max} Medium	1.06	1.07	1.08	1.09	0.80	0.15	0.15
C_{max} Medium high	1.08	1.09	1.10	1.11	0.80	0.15	0.15
C_{max} Slightly high	1.10	1.11	1.12	1.14	0.80	0.15	0.15
C_{max} High	1.12	1.14	1.15	1.16	0.80	0.15	0.15
C_{max} Very high	1.15	1.16	1.17	1.18	0.80	0.15	0.15

Table 2 presents the numerical parameters used to define the MFs associated with the output variables C_0 and C_{max} of the fuzzy adapter. Nine linguistic terms were established for each parameter, allowing for gradual and precise modulation of the output values within the inference system. Once this parameter was determined, the parameter C_{max} was kept fixed while the values of C_0 were varied, and subsequently, C_0 was fixed to analyze the impact of C_{max} on the algorithm's behavior. Increments of 0.1 were also made, but in this case, they were distributed among the parameters a, b, c, and d. This procedure allowed the identification of configurations that favor the intensification of the exploratory or exploitative phase depending on the conditions of the optimization process, providing a solid basis for the design of the output MFs.

The parameter λ was determined through a systematic ablation study evaluating six discrete values ($\lambda \in \{0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$) across eight benchmark functions, with 30 independent runs per configuration. To assess overall performance differences, a Friedman test was conducted, and the resulting average ranks are reported in Table 3.

Table 3. Average ranks of λ configurations according to the Friedman test

λ	Average Rank
0.8	1.50
0.7	2.12
0.9	3.38
0.5	3.75
0.6	5.12
0.4	5.12

The Friedman test revealed statistically significant differences between the evaluated λ configurations ($\chi^2(5) = 25.71$, $p = 1.01 \times 10^{-4}$), indicating that their performance varies considerably among the benchmark functions. As shown in Table 3, the $\lambda = 0.8$ configuration obtained the lowest average rating, demonstrating the best overall performance among all candidates. Conversely, the lowest λ values, particularly 0.4 and 0.6, consistently obtained lower ratings, reflecting inferior performance. All the benchmark mathematical functions used in the study are summarized in Table 4.

Table 4. Benchmark functions

Function	Name	Search Space	Min
F1	Sphere	[-10,10]	0
F2	Rastrigin	[-5.12, 5.12]	0
F3	Griewank	[-600, 600]	0
F4	Powell	[-1, 1]	0
F5	Rosenbrock	[-30, 30]	0
F6	Alpine	[-10, 10]	0
F7	Zakharov	[-5, 10]	0
F8	Sum squares	[-10, 10]	0
F9	Wood	[-10, 10]	0
F10	Djeong1	[-5.12, 5.12]	0
F11	Levy	[-10, 10]	0
F12	Dixon price	[-10, 10]	0
F13	Hyper ellipsoid	[-100, 100]	0

Table 4 presents the set of benchmark mathematical functions employed across the different experiments in this study. For each function, its identifier, name, search space, and minimum theoretical value (zero in all cases) are specified. This collection includes unimodal and multimodal functions widely used in the literature, enabling an objective evaluation of the performance, robustness, and exploration–exploitation capabilities of the analyzed algorithms. To examine the statistical significance of the differences between pairs of configurations in greater detail, the Wilcoxon signed-rank test was applied to a representative subset of benchmark functions, selected to capture diverse landscape characteristics including unimodal, multimodal, and separable structures. The results are presented in Table 5.

Table 5. Wilcoxon signed-rank test p-values comparing $\lambda = 0.8$

Function	0.4	0.5	0.6	0.7	0.9
F1	0.00001	0.00001	0.00001	0.00001	0.00001
F2	0.02750	0.14000	0.04040	0.42800	0.09550
F3	0.69400	0.50800	0.09550	0.72200	0.19700
F5	0.00001	0.00010	0.00001	0.00001	0.00001
F7	0.00001	0.00010	0.00001	0.00001	0.00001
F8	0.04070	0.19100	0.01920	0.26500	0.02510
F11	0.00001	0.00001	0.00001	0.01640	0.00001
F12	0.00001	0.00001	0.00001	0.00001	0.00001

The Wilcoxon signed-rank test results confirm that $\lambda = 0.8$ significantly outperforms most competing configurations on the majority of benchmark functions. In particular, statistically significant improvements ($p < 0.05$) were observed in almost all comparisons for functions such as F1, F5, F7, F11, and F12. Conversely, for functions such as F3, no statistically significant differences were detected, suggesting comparable performance among the configurations. Overall, these results reinforce the robustness and superiority of $\lambda = 0.8$ identified by the Friedman test.

6.2 Fuzzy Rule Base

The C_{max} and C_0 parameters are dynamically regulated by nine fuzzy rules that automatically balance exploration and exploitation, which are shown in Table 6.

Table 6. Fuzzy rules for C_{max} and C_0

Rule	Iteration	Diversity	C_{max}	C_0
R_1	Low	Low	Very high	Very low
R_2	Low	Medium	High	Low
R_3	Low	High	Slightly	Moderately low
R_4	Medium	Low	Medium high	Medium low
R_5	Medium	Medium	Medium	Medium
R_6	Medium	High	Medium low	Medium high
R_7	High	Low	Moderately low	Slightly high
R_8	High	Medium	Low	High
R_9	High	High	Very low	Very high

Table 6 presents a set of nine fuzzy rules (R1–R9) used for the dynamic adaptation of the parameters C_{max} and C_0 based on iterative progress and population diversity. In the initial stage of optimization, high values of C_{max} and low values of C_0 are assigned to favor exploration. As the algorithm progresses and diversity reaches intermediate levels, both parameters are adjusted toward average values, allowing a balance between exploration and exploitation. In the final stage, with a high number of iterations, the rules reduce C_{max} and increase C_0, promoting predominantly exploitative behavior and the refinement of solutions

7 Computational complexity and scalability analysis

To evaluate the computational overhead introduced by the interval-2 type fuzzy logic adapter, empirical runtime measurements were collected for RIO and Interval Type-2 Fuzzy Roach Infestation Optimization (IT2FRIO). across three problem dimensions (D = 50, 500, and 2000). Each configuration was run 30 times independently per reference function, using 2000 iterations. Table 7 shows the average runtime per function and dimension, while Table 8 presents the IT2FRIO/RIO overhead ratio, quantifying the additional computational cost attributed to the fuzzy system across all dimensions.

Table 7. Mean execution time(s) for RIO and IT2FRIO

Function		RIO (50D)	IT2FRIO (50D)	RIO (500D)	IT2FRIO (500D)	RIO (2000D)	IT2FRIO (2000D)
F1	$\bar{x}$	1.3796×10^{0}	3.7298×10^{0}	8.2597×10^{0}	1.1380×10^{1}	4.3829×10^{1}	4.7234×10^{1}
	s	2.9570×10^{-1}	1.8026×10^{-1}	3.0164×10^{-1}	1.6071×10^{0}	3.6821×10^{0}	8.1379×10^{0}
F2	$\bar{x}$	1.5727×10^{0}	3.9303×10^{0}	1.0100×10^{1}	1.2713×10^{1}	5.2977×10^{1}	5.3020×10^{1}
	s	1.0340×10^{-1}	1.4399×10^{-1}	2.7433×10^{-1}	1.7001×10^{-1}	5.9702×10^{0}	5.2253×10^{0}
F3	$\bar{x}$	1.7003×10^{0}	4.0342×10^{0}	1.0405×10^{1}	1.2890×10^{1}	4.7382×10^{1}	4.9489×10^{1}
	s	1.3700×10^{-1}	1.5056×10^{-1}	2.8812×10^{-1}	1.1149×10^{-1}	3.7080×10^{0}	9.7179×10^{-1}
F4	$\bar{x}$	1.6498×10^{0}	4.1296×10^{0}	1.0416×10^{1}	1.2637×10^{1}	5.0445×10^{1}	5.1660×10^{1}
	s	1.3210×10^{-1}	1.1355×10^{-1}	5.1034×10^{-1}	1.3154×10^{-1}	4.3173×10^{0}	3.4051×10^{0}
F5	$\bar{x}$	1.9496×10^{0}	4.4331×10^{0}	1.1375×10^{1}	1.2080×10^{1}	5.1207×10^{1}	5.0413×10^{1}
	s	1.4050×10^{-1}	2.3675×10^{-1}	2.0142×10^{0}	4.4765×10^{-1}	4.8617×10^{0}	6.7552×10^{0}
F6	$\bar{x}$	1.4333×10^{0}	3.9240×10^{0}	8.2352×10^{0}	1.0944×10^{1}	3.9089×10^{1}	4.7145×10^{1}
	s	1.4820×10^{-1}	1.1898×10^{-1}	3.2584×10^{-1}	1.3237×10^{-1}	5.0201×10^{-1}	4.4603×10^{0}
F7	$\bar{x}$	1.5656×10^{0}	4.3049×10^{0}	9.7141×10^{0}	1.2461×10^{1}	4.3275×10^{1}	4.6618×10^{1}
	s	1.5090×10^{-1}	6.5293×10^{-1}	1.1711×10^{-1}	1.5002×10^{-1}	1.4361×10^{0}	2.1614×10^{-1}
F8	$\bar{x}$	1.4194×10^{0}	4.2615×10^{0}	8.6426×10^{0}	1.1222×10^{1}	4.7918×10^{1}	4.5515×10^{1}
	s	1.6170×10^{-1}	4.8228×10^{-1}	3.1879×10^{-1}	3.5622×10^{-1}	7.3163×10^{0}	1.5390×10^{0}
F9	$\bar{x}$	1.4136×10^{0}	4.3156×10^{0}	8.1717×10^{0}	1.0791×10^{1}	4.1162×10^{1}	4.2222×10^{1}
	s	1.4830×10^{-1}	5.0999×10^{-1}	1.8449×10^{-1}	1.2530×10^{-1}	6.2640×10^{0}	3.1080×10^{-1}
F10	$\bar{x}$	1.8862×10^{0}	4.4606×10^{0}	1.0674×10^{1}	1.2070×10^{1}	5.3697×10^{1}	4.4130×10^{1}
	s	1.2910×10^{-1}	3.0827×10^{-1}	6.7222×10^{-1}	4.1411×10^{-1}	6.5863×10^{0}	3.9997×10^{0}
F11	$\bar{x}$	1.4397×10^{0}	4.3754×10^{0}	8.2521×10^{0}	1.0959×10^{1}	4.7171×10^{1}	4.2530×10^{1}
	s	1.0520×10^{-1}	1.0669×10^{0}	1.4198×10^{-1}	2.0968×10^{-1}	3.3658×10^{0}	2.3350×10^{-1}
F12	$\bar{x}$	1.6190×10^{0}	4.1128×10^{0}	1.0594×10^{1}	1.2644×10^{1}	4.5060×10^{1}	4.6963×10^{1}
	s	1.1580×10^{-1}	1.2476×10^{-1}	1.2410×10^{0}	1.3829×10^{-1}	5.3661×10^{-1}	4.3507×10^{-1}
F13	$\bar{x}$	1.3971×10^{0}	3.9045×10^{1}	1.0915×10^{1}	1.0981×10^{1}	4.1544×10^{1}	4.2769×10^{1}
	s	1.3760×10^{-1}	1.2479×10^{-1}	3.2272×10^{0}	1.1019×10^{-1}	2.0421×10^{0}	3.4900×10^{-1}

As shown in Table 7, IT2FRIO consistently requires a longer runtime than RIO on all reference functions at D = 50. As the dimensionality increases to D = 500 and D = 2000, the absolute difference in runtime decreases markedly, and both algorithms converge to comparable runtimes at the highest tested dimensionality. This pattern is observed uniformly across all 13 functions.

Table 8. T2FRIO/RIO execution time ratio per function and dimension

Function	D = 50	D = 500	D = 2000
F1	2.70x	1.38x	1.08x
F2	2.50x	1.26x	1.00x
F3	2.37x	1.24x	1.04x
F4	2.50x	1.21x	1.02x
F5	2.27x	1.06x	0.98x
F6	2.74x	1.33x	1.21x
F7	2.75x	1.28x	1.08x
F8	3.00x	1.30x	0.95x
F9	3.05x	1.32x	1.03x
F10	2.36x	1.13x	0.82x
F11	3.04x	1.33x	0.96x
F12	2.54x	1.19x	1.04x
F13	2.79x	1.01x	1.03x

Table 8 further quantifies this behavior using the IT2FRIO/RIO overhead ratio. For D = 50, the average ratio is 2.68×, indicating that IT2FRIO requires approximately 2.7 times the execution time of RIO. This ratio decreases to 1.23× for D = 500 and reaches 1.01× for D = 2000, confirming that the additional cost introduced by the fuzzy adapter becomes negligible as the problem's dimensionality increases. This trend suggests that the overhead is primarily associated with fixed costs such as type reduction and defuzzification rather than dimension dependent operations, demonstrating that IT2FRIO scales favorably in large-scale optimization scenarios. Fig.2 illustrates the mean execution time of RIO and IT2FRIO across 13 benchmark functions at D = 50, 500, and 2000. Each panel corresponds to a dimensionality level, allowing a direct visual comparison of how runtime evolves for both algorithms as problem size increases.

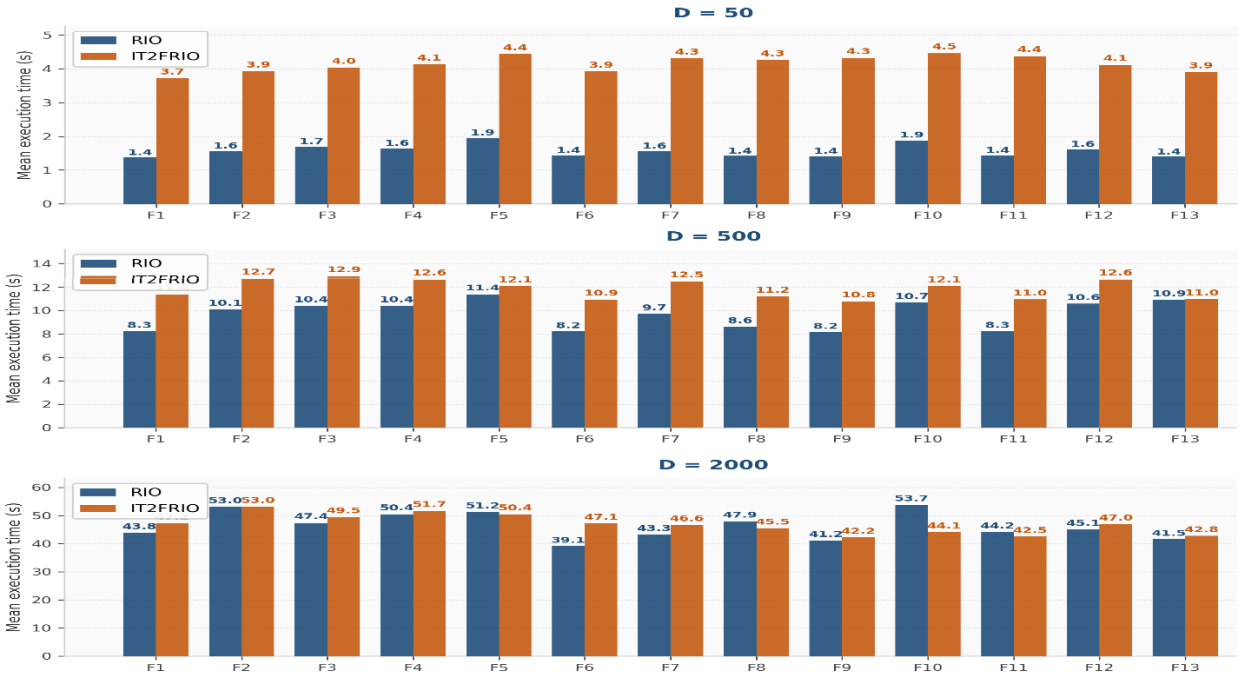

Fig. 2. Execution time scaling: RIO vs IT2FRIO

At D = 50, the difference between both algorithms is visually pronounced across all functions, with IT2FRIO bars reaching consistently higher values than RIO, reflecting the relative weight of the fuzzy initialization cost at low dimensionality. Notably, the gap is not uniform functions such as F9, F10, and F11 exhibit the largest overhead ratios, while F5 and F13 show comparatively

smaller differences, suggesting that the interaction between the fuzzy adapter and the function landscape has a modest influence at this scale. At D = 500, both algorithms show a substantial absolute increase in runtime, yet the relative gap between them narrows visibly the bars begin to approach comparable heights, particularly in functions such as F5, F6, and F13 where the ratios are closest to 1.0×. At D = 2000, the bars of RIO and IT2FRIO are nearly indistinguishable across all 14 functions, with values fluctuating within a narrow range of approximately 39–54 seconds for both algorithms. In several functions including F8, F10, and F11 IT2FRIO records marginally lower execution times than RIO, a result attributable to natural run to run variability rather than a systematic advantage, consistent with the near unity ratios reported in Table 8.

8 Results

This section employs benchmark mathematical functions to analyze the effectiveness of the parameter adaptation mechanism based on Interval Type-2 Fuzzy Logic, a strategy commonly used for the comparative evaluation of evolutionary and metaheuristic algorithms (Moloodpoor & Mortazavi, 2025; Mortazavi, 2024). The experimental study considers three configurations of the RIO algorithm: the original version, a variant incorporating Type-1 Fuzzy Logic (FRIO), and the proposed version based on Interval Type-2 Fuzzy Logic.

The experiments were conducted on a computer with an Intel® Core™ i7-5500U processor at 2.40 GHz, 8 GB of RAM, and a 1 TB solid state drive (SSD), running Windows 10. The programming environment used was MATLAB R2023b. A population of 20 agents and a maximum of 2000 iterations were used, maintaining homogeneous evaluation conditions to ensure the comparative validity of the results. The evaluations were conducted considering different levels of problem dimensionality, specifically 50, 100, 200, and 500 dimensions, in order to analyze the scalability and robustness of each approach. For statistical validation of the results, a two sample left tailed hypothesis test with a 95% confidence level was applied to determine whether IT2FRIO performance is statistically superior to that of the other RIO algorithm variants. Additionally, IT2FRIO performance was compared to that of two metaheuristics widely recognized in the literature: PSO and Cuckoo Search (CS). The comparative analysis of the fuzzy adapters began with 50-dimensional benchmark functions, the results of which are reported in Table 9.

Table 9. Performance comparison of metaheuristic algorithms with 50 dimensions

Function		RIO	FRIO	IT2FRIO	PSO	CS
F1	$\bar{x}$	8.8641×10^{-4}	1.1306×10^{-4}	1.1731×10^{-7}	5.7903×10^{-6}	1.0370×10^{-6}
	s	2.6670×10^{-3}	1.5617×10^{-4}	2.8434×10^{-6}	1.6795×10^{-5}	4.7133×10^{-7}
F2	$\bar{x}$	5.0689×10^{1}	3.3528×10^{1}	2.2461×10^{1}	9.2107×10^{1}	1.3372×10^{2}
	s	5.1204×10^{1}	9.9891×10^{0}	2.9276×10^{1}	2.1793×10^{1}	1.6397×10^{1}
F3	$\bar{x}$	3.2392×10^{0}	3.1833×10^{0}	3.1803×10^{0}	3.1803×10^{0}	2.2718×10^{1}
	s	1.2065×10^{-1}	2.5419×10^{-2}	3.1459×10^{-2}	3.1459×10^{-2}	3.6134×10^{-15}
F4	$\bar{x}$	1.1914×10^{-3}	5.5200×10^{-6}	1.0266×10^{-6}	1.0374×10^{-1}	3.9312×10^{-3}
	s	2.6760×10^{-3}	1.0653×10^{-5}	1.9235×10^{-5}	1.8848×10^{-1}	4.2383×10^{-3}
F5	$\bar{x}$	1.7445×10^{-7}	3.1058×10^{-6}	5.0111×10^{-8}	2.8983×10^{-9}	1.5834×10^{-37}
	s	2.0744×10^{-7}	3.8399×10^{-6}	2.4936×10^{-7}	3.7697×10^{-9}	6.2693×10^{-37}
F6	$\bar{x}$	1.6643×10^{2}	9.2536×10^{1}	4.0865×10^{1}	9.2089×10^{1}	5.7458×10^{1}
	s	6.0671×10^{1}	2.7438×10^{1}	4.1057×10^{1}	4.1607×10^{1}	2.4894×10^{1}
F7	$\bar{x}$	1.1034×10^{-1}	2.4903×10^{-3}	1.0087×10^{-3}	1.4205×10^{-1}	1.6694×10^{-6}
	s	2.2241×10^{-1}	6.1369×10^{-3}	6.5096×10^{-3}	3.0308×10^{-1}	7.9404×10^{-7}
F8	$\bar{x}$	3.0672×10^{2}	8.5019×10^{1}	1.4480×10^{1}	1.1894×10^{2}	3.6861×10^{1}
	s	2.3304×10^{2}	3.6167×10^{1}	7.6628×10^{1}	1.4481×10^{2}	1.3159×10^{0}
F9	$\bar{x}$	6.4372×10^{-3}	1.2271×10^{-3}	1.0699×10^{-3}	1.7984×10^{-3}	2.1183×10^{-2}
	s	2.1634×10^{-2}	1.8921×10^{-3}	3.3442×10^{-2}	9.3212×10^{-3}	3.8939×10^{-2}
F10	$\bar{x}$	3.5151×10^{-3}	1.7614×10^{-2}	4.5307×10^{-4}	2.6252×10^{-3}	2.3601×10^{-25}
	s	5.8481×10^{-3}	2.7115×10^{-2}	5.9543×10^{-3}	1.2308×10^{-3}	1.0207×10^{-25}
F11	$\bar{x}$	1.0212×10^{-2}	6.3665×10^{-3}	2.1960×10^{-4}	1.3925×10^{-2}	4.7380×10^{-3}
	s	2.0111×10^{-2}	4.0003×10^{-3}	3.7361×10^{-4}	4.3815×10^{-2}	1.5540×10^{-3}
F12	$\bar{x}$	7.5671×10^{-4}	3.1389×10^{-2}	1.8144×10^{-3}	5.1802×10^{0}	3.0532×10^{0}
	s	3.4808×10^{-3}	1.6324×10^{-2}	4.3290×10^{-3}	3.5483×10^{0}	2.2289×10^{0}
F13	$\bar{x}$	3.7673×10^{0}	4.6419×10^{0}	2.9848×10^{0}	1.9328×10^{1}	6.1317×10^{0}
	s	2.9722×10^{0}	2.7416×10^{0}	3.1183×10^{0}	1.5486×10^{1}	3.8649×10^{0}

Table 9 shows that the IT2FRIO variant performs better than the other versions of the RIO algorithm, as well as compared to the PSO and CS metaheuristics, when considering the reported average values and standard deviations. Table 10 reports the Z statistical tests corresponding to 50 dimensions.

Table 10. Statistical Z-test analysis for 50-dimensional functions

Function	RIO	FRIO	PSO	CS
F1	**-1.7791**	**-3.9605**	**-1.8241**	**-1.7477**
F2	**-2.6213**	**-1.9596**	**-10.4521**	**-18.1609**
F3	**-2.4364**	-1.1193	**-3.0147**	**-5.0790**
F4	**-2.0996**	**-4.3495**	1.0369	1.1007
F5	**-9.3881**	**-5.7312**	**-4.7998**	**-1.8928**
F6	**-2.5216**	-1.2841	**-7.5809**	3.6338
F7	**-6.5575**	**-4.7619**	**-3.5317**	**-1.6876**
F8	**-2.7092**	-0.0257	-0.1149	**-2.1463**
F9	-0.6386	**-3.3858**	**-1.9567**	**-1.7290**
F10	**-2.7210**	**-8.3799**	**-1.7132**	0.4168
F11	1.0429	**-9.5917**	**-7.9935**	**-7.4984**
F12	-0.9949	**-2.1859**	**-5.6667**	**-3.4709**
F13	-1.3089	1.1569	-0.8100	1.2189

Table 10 reports the results of the Z tests for 50-dimensional benchmark functions, used to evaluate the statistical significance of IT2FRIO performance against the compared algorithms. Table 11 presents the average values and standard deviations for 100 dimensions.

Table 11. Performance comparison of metaheuristic algorithms with 100 dimensions

Function		RIO	FRIO	IT2FRIO	PSO	CS
F1	$\bar{x}$	3.9719×10^{-1}	1.2624×10^{-1}	2.5343×10^{-2}	6.5986×10^{0}	1.5763×10^{-1}
	s	5.2440×10^{-1}	1.3661×10^{-1}	3.4018×10^{-2}	3.3605×10^{0}	1.5002×10^{-1}
F2	$\bar{x}$	2.1387×10^{2}	9.3443×10^{1}	6.5282×10^{1}	2.5479×10^{2}	2.8941×10^{2}
	s	1.4272×10^{2}	3.0338×10^{1}	5.4213×10^{1}	3.6922×10^{2}	3.6138×10^{2}
F3	$\bar{x}$	1.0581×10^{-2}	4.7441×10^{-3}	3.2268×10^{-3}	7.7200×10^{0}	1.1121×10^{0}
	s	2.4777×10^{-2}	8.9728×10^{-3}	3.9488×10^{-3}	3.2052×10^{0}	3.5100×10^{0}
F4	$\bar{x}$	1.9291×10^{-6}	4.7177×10^{-6}	3.3978×10^{-7}	8.7070×10^{-8}	3.1041×10^{-29}
	s	1.6350×10^{-6}	5.4339×10^{-6}	2.9988×10^{-7}	7.7033×10^{-7}	1.5415×10^{-28}
F5	$\bar{x}$	4.5017×10^{2}	4.1271×10^{2}	1.2515×10^{2}	6.3957×10^{4}	9.4299×10^{3}
	s	2.0739×10^{2}	8.6434×10^{1}	5.8839×10^{1}	7.8789×10^{4}	5.0019×10^{3}
F6	$\bar{x}$	5.0951×10^{-1}	1.4529×10^{-1}	2.2611×10^{-2}	1.1723×10^{1}	1.6346×10^{-2}
	s	5.5056×10^{-1}	3.3273×10^{-1}	2.9430×10^{-2}	3.0428×10^{1}	5.5515×10^{-3}
F7	$\bar{x}$	8.8690×10^{2}	4.2297×10^{2}	2.3931×10^{2}	2.0332×10^{3}	3.5493×10^{2}
	s	5.1997×10^{2}	2.1564×10^{2}	3.7351×10^{2}	8.2178×10^{2}	6.0140×10^{1}
F8	$\bar{x}$	7.2453×10^{0}	3.9300×10^{0}	2.6118×10^{0}	3.4439×10^{1}	1.3095×10^{3}
	s	7.8564×10^{0}	3.3267×10^{0}	2.6067×10^{0}	2.6734×10^{2}	2.0674×10^{3}
F9	$\bar{x}$	1.4695×10^{-2}	2.8735×10^{-2}	5.0003×10^{-3}	2.6215×10^{-1}	1.2747×10^{-23}
	s	3.1560×10^{-2}	2.3389×10^{-2}	5.6292×10^{-2}	3.4356×10^{-1}	2.9817×10^{-23}
F10	$\bar{x}$	4.6478×10^{-1}	1.2209×10^{-1}	2.0713×10^{-2}	1.7051×10^{0}	6.8027×10^{-2}
	s	9.2602×10^{-1}	8.7783×10^{-2}	1.8283×10^{-2}	2.6333×10^{0}	3.6126×10^{-2}
F11	$\bar{x}$	4.3806×10^{-1}	2.1768×10^{-1}	9.9592×10^{-2}	2.6230×10^{1}	9.8526×10^{0}
	s	6.0940×10^{-1}	3.5219×10^{-2}	9.0916×10^{-2}	6.7217×10^{1}	2.6314×10^{1}
F12	$\bar{x}$	3.5883×10^{1}	2.1614×10^{1}	2.6322×10^{1}	1.9448×10^{3}	2.3034×10^{2}
	s	3.4261×10^{1}	1.3871×10^{1}	1.2038×10^{1}	1.1511×10^{3}	1.3491×10^{2}
F13	$\bar{x}$	3.2589×10^{1}	6.8398×10^{0}	2.5076×10^{0}	4.1273×10^{1}	1.5343×10^{3}
	s	4.7513×10^{1}	5.4515×10^{0}	2.8606×10^{0}	2.8872×10^{2}	4.9508×10^{3}

Table 11 shows that IT2FRIO consistently achieves the lowest average values in almost all benchmark functions, both unimodal (F1, F3, F10) and multimodal (F2, F5, F7), accompanied by lower standard deviations, indicating greater stability. In contrast, PSO and CS significantly degrade their performance in more complex functions, demonstrating poor scalability. Table 12 reports the Z statistical tests corresponding to 100 dimensions.

Table 12. Statistical Z-test analysis for 100-dimensional functions

Function	RIO	FRIO	PSO	CS
F1	**-3.8757**	**-5.3261**	**-10.7131**	**-4.7102**
F2	**-5.3308**	**-2.4828**	**-2.7814**	**-3.3594**
F3	-1.6055	-0.8477	**-12.3324**	**-1.7304**
F4	**-5.2368**	**-4.4801**	1.6744	6.2060
F5	**-5.7172**	**-9.8250**	**-4.3610**	**-10.0788**
F6	**-4.8370**	**-2.0116**	**-2.1061**	1.1458
F7	**-5.5403**	**-2.2324**	**-10.8848**	**-1.6739**
F8	**-3.0660**	**-1.7084**	-0.6520	**-3.4624**
F9	-0.8228	**-2.1147**	**-4.0457**	**-8.0779**
F10	**-2.6261**	**-6.1925**	**-3.5034**	0.4865
F11	**-3.0088**	**-1.7782**	**-2.1293**	**-2.0301**
F12	-1.4421	1.4040	**-9.1281**	**-8.2502**
F13	**-3.4615**	**-3.8542**	-0.7354	**-1.7168**

Table 12 presents the Z-test results for 100-dimensional benchmark functions, used to evaluate the statistical significance of IT2FRIO performance compared to the benchmark algorithms, showing that the superiority demonstrated at 50 dimensions is maintained. Table 13 presents the average values and standard deviations for 200 dimensions.

Table 13. Performance comparison of metaheuristic algorithms with 200 dimensions

Function		RIO	FRIO	IT2FRIO	PSO	CS
F1	$\bar{x}$	1.0019×10^{1}	5.1514×10^{0}	2.4399×10^{0}	7.9173×10^{1}	4.5403×10^{0}
	s	5.9210×10^{0}	2.4054×10^{0}	2.9382×10^{0}	5.6304×10^{1}	1.7155×10^{0}
F2	$\bar{x}$	6.5212×10^{2}	4.1074×10^{2}	3.0281×10^{2}	7.9581×10^{2}	9.1931×10^{2}
	s	3.2819×10^{2}	2.0211×10^{2}	2.4891×10^{2}	1.5077×10^{2}	6.1935×10^{2}
F3	$\bar{x}$	7.9802×10^{-2}	5.1633×10^{-2}	3.7134×10^{-2}	7.0758×10^{1}	1.9571×10^{1}
	s	5.2847×10^{-2}	2.4814×10^{-2}	2.7590×10^{-2}	4.6928×10^{1}	5.1733×10^{1}
F4	$\bar{x}$	2.2114×10^{-5}	3.5689×10^{-6}	4.8165×10^{-7}	5.4342×10^{-9}	1.4128×10^{-7}
	s	1.9804×10^{-5}	4.1734×10^{-6}	3.7333×10^{-7}	4.9609×10^{-9}	1.7128×10^{-7}
F5	$\bar{x}$	1.4806×10^{3}	1.4766×10^{3}	1.1768×10^{3}	2.0263×10^{6}	4.1662×10^{5}
	s	2.2912×10^{2}	2.2034×10^{2}	9.5645×10^{2}	3.3086×10^{6}	9.0557×10^{5}
F6	$\bar{x}$	9.8343×10^{0}	4.7853×10^{0}	3.9278×10^{0}	6.0775×10^{0}	2.7864×10^{0}
	s	7.2350×10^{0}	2.3687×10^{0}	1.4627×10^{0}	8.1129×10^{0}	6.9658×10^{-1}
F7	$\bar{x}$	7.5487×10^{3}	3.6003×10^{3}	1.9253×10^{3}	6.9929×10^{3}	6.7294×10^{3}
	s	6.8189×10^{3}	7.7000×10^{2}	3.0875×10^{3}	1.3024×10^{3}	8.1699×10^{3}
F8	$\bar{x}$	5.3993×10^{2}	3.4472×10^{2}	2.4901×10^{2}	7.3462×10^{3}	3.7892×10^{3}
	s	3.2412×10^{2}	1.6675×10^{2}	2.5987×10^{2}	4.6973×10^{3}	4.4303×10^{3}
F9	$\bar{x}$	1.1738×10^{-1}	1.6931×10^{-2}	3.9532×10^{-3}	1.1697×10^{-2}	1.9842×10^{-23}
	s	1.2712×10^{-1}	2.1030×10^{-2}	4.2469×10^{-3}	3.3588×10^{-2}	6.5619×10^{-23}
F10	$\bar{x}$	8.9768×10^{0}	4.6348×10^{0}	3.1028×10^{0}	2.1259×10^{1}	4.4849×10^{0}
	s	4.9288×10^{0}	1.5758×10^{0}	2.5746×10^{0}	5.0893×10^{1}	1.2269×10^{0}
F11	$\bar{x}$	5.3613×10^{0}	3.2435×10^{0}	1.7535×10^{0}	7.6477×10^{1}	2.6180×10^{1}
	s	2.3681×10^{0}	5.0971×10^{0}	1.6995×10^{0}	7.9624×10^{1}	6.8861×10^{0}
F12	$\bar{x}$	6.4716×10^{2}	3.5258×10^{2}	3.7950×10^{2}	8.4867×10^{4}	3.1938×10^{4}
	s	8.9057×10^{2}	1.3182×10^{2}	1.6228×10^{2}	9.0916×10^{4}	2.7051×10^{4}
F13	$\bar{x}$	2.8544×10^{2}	4.6749×10^{2}	2.2476×10^{0}	6.6709×10^{5}	1.4824×10^{5}
	s	6.1499×10^{2}	2.3313×10^{2}	1.6069×10^{0}	3.89728×10^{5}	3.7710×10^{5}

The results in Table 13 support the previously observed trends. IT2FRIO achieves the lowest mean fitness values in the majority of benchmark functions, outperforming RIO, FRIO, PSO, and CS in most cases. This confirms that the Interval Type-2 Fuzzy adapter consistently enhances the search capability of the base algorithm at 200 dimensions. Table 14 validates these results through statistical tests.

Table 14. Statistical Z-test analysis for 200-dimensional functions

Function	RIO	FRIO	PSO	CS
F1	**-5.4517**	**-2.4673**	**-7.3572**	**-1.7715**
F2	**-4.6449**	**-1.8437**	**-9.2789**	**-5.0588**
F3	**-3.9202**	**-2.1401**	**-8.2542**	**-2.0681**
F4	**-5.9818**	**-4.0356**	6.9861	4.5388
F5	**-1.6919**	**-1.6730**	**-3.3525**	**-2.5128**
F6	**-4.3828**	**-1.6871**	-1.4283	3.8588
F7	**-4.1148**	**-2.8831**	**-8.2831**	**-3.0128**
F8	**-3.8356**	**-1.6978**	**-8.2630**	**-4.3693**
F9	**-4.8845**	**-3.3132**	-1.2528	5.0984
F10	**-5.7858**	**-2.7798**	**-1.9515**	**-2.6543**
F11	**-6.7794**	-1.5189	**-5.1350**	**-1.9423**
F12	-1.6195	0.7052	**-5.0899**	**-6.3898**
F13	**-7.8659**	**-7.6493**	**-9.3754**	**-2.6760**

The results in Table 14 support the previously observed trends, as comparisons of RIO and FRIO against IT2FRIO show a statistically significant advantage for IT2FRIO in most of the evaluated functions. This confirms that the proposed approach maintains a consistent advantage under the analyzed conditions. Table 15 presents the average values and standard deviations for 500 dimensions.

Table 15. Performance comparison of metaheuristic algorithms with 500 dimensions

Function		RIO	FRIO	IT2FRIO	PSO	CS
F1	$\bar{x}$	9.1052×10^{1}	6.0102×10^{1}	5.6442×10^{1}	6.7839×10^{2}	1.0038×10^{2}
	s	4.3861×10^{1}	9.8909×10^{0}	1.2267×10^{1}	8.7641×10^{2}	1.0525×10^{2}
F2	$\bar{x}$	3.6236×10^{3}	2.7518×10^{3}	2.1113×10^{3}	3.8804×10^{3}	3.1436×10^{3}
	s	6.5186×10^{2}	6.4771×10^{2}	7.9090×10^{2}	5.8126×10^{2}	2.0311×10^{2}
F3	$\bar{x}$	3.0624×10^{-1}	2.4314×10^{-1}	2.3797×10^{-1}	6.2059×10^{2}	8.9625×10^{1}
	s	5.9499×10^{-2}	5.1867×10^{-2}	4.2067×10^{-2}	8.1348×10^{2}	8.2277×10^{1}
F4	$\bar{x}$	2.3994×10^{-5}	6.1136×10^{-5}	3.9391×10^{-6}	6.3623×10^{-8}	2.0852×10^{-6}
	s	1.9986×10^{-5}	7.5975×10^{-5}	3.0865×10^{-6}	5.6746×10^{-5}	3.8597×10^{-6}
F5	$\bar{x}$	5.4174×10^{4}	4.1785×10^{4}	4.1448×10^{3}	3.2712×10^{7}	1.7603×10^{6}
	s	4.3606×10^{4}	2.8009×10^{4}	5.2333×10^{2}	8.8604×10^{7}	3.3719×10^{6}
F6	$\bar{x}$	8.8085×10^{1}	6.3974×10^{1}	5.7088×10^{1}	2.9444×10^{2}	1.0561×10^{2}
	s	2.7704×10^{1}	1.1540×10^{1}	1.4104×10^{1}	2.7558×10^{2}	1.4418×10^{2}
F7	$\bar{x}$	8.7255×10^{4}	1.0878×10^{4}	3.6666×10^{4}	2.2253×10^{17}	4.6331×10^{4}
	s	9.6628×10^{4}	7.1616×10^{3}	2.6549×10^{4}	6.5730×10^{17}	1.3085×10^{4}
F8	$\bar{x}$	1.9365×10^{4}	1.2253×10^{4}	1.0216×10^{4}	1.5604×10^{5}	1.1948×10^{4}
	s	4.6109×10^{3}	2.3356×10^{3}	3.1440×10^{3}	1.9238×10^{4}	1.5993×10^{3}
F9	$\bar{x}$	2.2067×10^{-2}	2.1156×10^{-2}	6.4036×10^{-3}	1.2363×10^{-4}	2.0993×10^{-24}
	s	2.6594×10^{-2}	2.5164×10^{-2}	1.0254×10^{-2}	1.9453×10^{-4}	2.5053×10^{-23}
F10	$\bar{x}$	8.8458×10^{1}	6.8376×10^{1}	6.2594×10^{1}	1.7808×10^{2}	9.8112×10^{1}
	s	1.8100×10^{1}	1.1396×10^{1}	1.2872×10^{1}	2.6943×10^{2}	4.0128×10^{1}
F11	$\bar{x}$	2.3555×10^{1}	1.9418×10^{1}	1.7425×10^{1}	3.7258×10^{2}	9.1740×10^{1}
	s	2.0593×10^{0}	3.7297×10^{0}	3.2039×10^{0}	4.9934×10^{2}	5.7198×10^{1}
F12	$\bar{x}$	1.7849×10^{4}	8.5860×10^{3}	9.9832×10^{3}	3.6060×10^{6}	1.8574×10^{5}
	s	3.9598×10^{3}	2.8743×10^{3}	3.9386×10^{3}	8.5828×10^{6}	4.5650×10^{5}
F13	$\bar{x}$	1.9255×10^{4}	9.9570×10^{4}	1.4057×10^{3}	1.4506×10^{7}	2.0678×10^{6}
	s	3.2782×10^{4}	6.2515×10^{4}	2.8458×10^{2}	1.9176×10^{7}	2.0412×10^{6}

Table 15 reports the average values and standard deviations obtained for 500 dimensions, which show that IT2FRIO consistently achieves the lowest average values and lower variability in most functions, especially in multimodal and non-separable problems, while PSO and CS show a degradation of performance, reflecting less stability in more complex scenarios. Table 16 validates these results through statistical tests.

Table 16. Z-tests for 500 dimensions

Function	RIO	FRIO	PSO	CS
F1	**-4.2825**	**-1.6785**	**-3.8928**	**-2.3229**
F2	**-8.0712**	**-3.4210**	**-9.8610**	**-6.9109**
F3	**-7.9486**	**-2.5565**	**-4.1771**	**-5.9523**
F4	**-5.4317**	**-4.1201**	6.8762	2.0547
F5	**-6.2836**	**-7.3593**	**-2.3001**	**-2.8527**
F6	**-5.4613**	**-2.0696**	**-4.7113**	**-1.8345**
F7	**-9.8074**	**5.1366**	**-1.8543**	**-1.7934**
F8	**-8.8792**	**-2.8487**	**-4.1512**	**-2.6894**
F9	**-4.3981**	**-2.9736**	3.3539	3.4205
F10	**-6.3782**	**-1.8421**	**-2.3597**	**-4.6163**
F11	**-8.8156**	**-2.2201**	**-3.8956**	**-7.1052**
F12	**-7.7140**	1.5695	**-2.2948**	**-2.1087**
F13	**-4.2825**	**-1.6785**	**-3.8928**	**-2.3229**

Table 16 presents the Z-test results obtained for functions F1–F13. In general, the Z-values are mostly negative, indicating a statistical advantage of the reference algorithm over the compared methods. Positive values are present in a few isolated cases, reflecting specific scenarios where this advantage does not hold. Overall, the results show a consistent trend that supports the behavior observed in the average performance tables, further demonstrating that IT2FRIO superiority increases with the problem's dimensionality. To validate the Z tests, Wilcoxon signed-rank tests were performed to compare RIO, FRIO, PSO, and CS with IT2FRIO across different dimensions, using a significance level of $\alpha = 0.05$. Table 17 shows the results obtained for 50 dimensions.

Table 17. Wilcoxon test 50 dimensions

Function	FRIO	FRIO	PSO	CS
F1	0.0000000019	0.0000000019	0.0000000019	0.0000000019
F2	0.0027663205	0.0006084088	0.0000000093	0.0000000019
F3	0.0004183967	0.1518869195	0.1579476744	0.0000000019
F4	0.0000000019	0.0000304952	0.0000000019	0.0000000019
F5	0.0000207983	0.0000000019	0.0006084088	0.0000000019
F6	0.0000000019	0.0000000130	0.0000079945	0.0025600903
F7	0.0000002049	0.0120476224	0.0000001639	0.0000000019
F8	0.0000014193	0.0000037905	0.0000051446	0.0000044219
F9	0.0000011921	0.0000182446	0.0001373943	0.0000000261
F10	0.0000000019	0.0000000019	0.0000000019	0.0000000019
F11	0.0000000130	0.0000000019	0.0000000019	0.0000000019
F12	0.0001373943	0.0000000019	0.0000000019	0.0000000019
F13	0.1190765686	0.0136632416	0.0000000261	0.0053827800

Table 17 presents the Wilcoxon signed-rank test results for the 50-dimensional case. The observed p-values reflect consistent performance differences. The analysis is extended to higher-dimensional scenarios, as shown in Table 18.

Table 18. Wilcoxon test 100 dimensions

Function	FRIO	FRIO	PSO	CS
F1	0.0000000615	0.0000002552	0.0000000019	0.0000003856
F2	0.0000000037	0.0003127549	0.0000440758	0.0000069179
F3	0.0046648048	0.0732440744	0.0000000019	0.0000000019
F4	0.0000002552	0.0000000130	0.0001698975	0.0000000019
F5	0.0000000019	0.0000000019	0.0000000019	0.0000000019
F6	0.0000000019	0.0002832636	0.0000000019	0.0961015727
F7	0.0000000019	0.0000236668	0.0000000019	0.0005548261
F8	0.0000059735	0.1518869195	0.0426517054	0.0000000019
F9	0.0029870011	0.0000000801	0.0000000019	0.0000000019
F10	0.0000000093	0.0000000019	0.0000000019	0.0000001304
F11	0.0000304952	0.0000037905	0.0000000019	0.0000000019
F12	0.2621222734	0.0128337331	0.0000000019	0.0000000019
F13	0.0000000019	0.0000009984	0.0000092201	0.0000000019

Table 18 presents the Wilcoxon signed-rank test results for the 100-dimensional case. The results are consistent with those obtained in lower dimensions, maintaining the observed performance trends across all benchmark functions. Table 19 shows the results obtained for 200 dimensions.

Table 19. Wilcoxon test 200 dimensions

Function	FRIO	FRIO	PSO	CS
F1	0.0000000019	0.0000000037	0.0000000019	0.0000016838
F2	0.0000003856	0.0840654783	0.0000000261	0.0000009984
F3	0.0000182446	0.1980757993	0.0000000019	0.0000000019
F4	0.0000000019	0.0000000130	0.0000000019	0.0000000354
F5	0.1347350832	0.0767212156	0.0000000019	0.0000000019
F6	0.0000079945	0.8235769439	0.3284698576	0.0007978976
F7	0.0000106096	0.0000079945	0.0000037905	0.0002316367
F8	0.0000628661	0.0071114562	0.0000000019	0.0000000037
F9	0.0000000019	0.0000000615	0.2621222734	0.0000000019
F10	0.0000006910	0.0040317941	0.0037444327	0.0076121371
F11	0.0000000019	0.0522633474	0.0000000019	0.0000000019
F12	0.5158484429	0.1094320584	0.0000000019	0.0000000019
F13	0.0000559259	0.0000000019	0.0000000019	0.0000000019

Table 19 presents the results of the statistical tests. The outcomes are consistent with the trends observed in earlier cases. Table 20 shows the results obtained for 500 dimensions.

Table 20. Wilcoxon test 500 dimensions

Function	FRIO	FRIO	PSO	CS
F1	0.0002091266	0.4399667662	0.0000000037	0.0961015727
F2	0.0000002552	0.0005548261	0.0000002049	0.0000003148
F3	0.0000000130	0.0120476224	0.0000000019	0.0000000019
F4	0.0000009984	0.0000000019	0.0000000019	0.0017185900
F5	0.0000000019	0.0000000019	0.0000000019	0.0000000019
F6	0.0000559259	0.1641840823	0.0000000037	0.0345367342
F7	0.0000705682	0.0000001304	0.0000000019	0.0003449582
F8	0.0000000019	0.0000001024	0.0000000019	0.0000092201
F9	0.0021877754	0.0012321044	0.0000000019	0.0000000019
F10	0.0000000093	0.0208498519	0.0635564886	0.0000159778
F11	0.0000023488	0.0154600032	0.0000000019	0.0000000019
F12	0.0000014193	0.2054096926	0.0000000019	0.0000000130
F13	0.0000000261	0.0000000019	0.0000000019	0.0000000019

The results reflect a similar statistical behavior as the problem complexity increases. Table 20 shows the corresponding results, completing the analysis

9 Discussion of Results

This section discusses the results obtained from the comparisons made between the evaluated algorithms. The analysis focuses on identifying general trends, differences in behavior, and effects associated with increased dimensionality, based on average values, standard deviations, and reported statistical tests.

Table 9 presents the results obtained for the benchmark functions using 50 dimensions. At this scale, IT2FRIO exhibits the best overall performance, achieving the lowest average value in 8 of the 13 functions (F1, F2, F4, F6, F8, F10, F11, and F13). FRIO shows competitive behavior in a few cases, such as F7, while CS achieves the best result in F5 due to its high exploitation capability on this specific function. In contrast, PSO does not consistently obtain the best performance in any function and shows higher variability in several cases.

Table 10 shows the Z-test results obtained when comparing IT2FRIO against RIO, FRIO, PSO, and CS. In the unimodal functions (F1, F4, and F5), IT2FRIO exhibits favorable Z-values in most cases, indicating a consistent improvement in the convergence and stability of the solution. This behavior is consistent with the ability of the Type-2 fuzzy mechanism to more precisely adjust the exploration-exploitation balance.

In the multimodal functions (F2, F3, F6, F7, and F8), IT2FRIO also maintains a general advantage over the compared algorithms, particularly in those functions with highly irregular landscapes, where adaptive parameter control is more relevant. However, in some specific functions, such as F11 and F13, IT2FRIO does not show clear superiority. In the case of F11, this behavior is linked to the strong variable interdependencies characteristic of non-separable functions: the incremental adjustments introduced by the Type-2 fuzzy mechanism provide less benefit when the correlation structure between variables is consistent, allowing algorithms with simpler but more coordinated update rules to match its performance at this scale.

Overall, the results suggest that IT2FRIO shows more robust behavior in both unimodal and multimodal functions, although its advantage is not uniform across all scenarios, which is to be expected in optimization problems with heterogeneous features.
Table 11 shows the results for 100 dimensions, where IT2FRIO exhibits the lowest average values in most functions, both unimodal and multimodal, and in several cases shows less variability. This behavior suggests greater stability of the algorithm as dimensionality increases.

In multimodal and non-separable functions, IT2FRIO maintains more robust performance, while PSO and CS show a more pronounced degradation, reflected in higher average values and greater dispersion. Overall, the results indicate that IT2FRIO maintains competitive and stable performance as the complexity of the problem increases.

The statistical analysis presented in Table 12 indicates that IT2FRIO maintains a significant advantage over RIO, FRIO, PSO, and CS in most of the functions evaluated for 100 dimensions, demonstrating that the behavior observed at lower dimensionality is preserved as the problem complexity increases.

In unimodal functions such as F1 and F3, IT2FRIO shows favorable differences compared to RIO and PSO, reflecting more efficient exploitation. In multimodal and non-separable functions, such as F2, F5, F7, and F12, the statistical differences are more pronounced, suggesting a better balance between exploration and exploitation in more complex scenarios.

Although in some specific functions, such as F3 and F12 versus FRIO, no significant differences are observed, these exceptions do not modify the general trend observed in Table 8, which supports the robustness and scalability of IT2FRIO when increasing dimensionality.

Analysis of Table 13, corresponding to 200-dimensional problems, shows that IT2FRIO maintains a favorable performance compared to RIO, FRIO, PSO, and CS as dimensionality increases. In most functions, IT2FRIO exhibits lower average values and more contained variability, suggesting more stable behavior in large scale search spaces.

This trend is most evident in multimodal and non-separable functions, such as F5, F6, and F7, where the other algorithms show a more marked decline. In contrast, PSO and CS demonstrate greater difficulty handling high dimensionality, reflected in considerably higher average values.

Although other methods achieve competitive results in some specific functions, the overall behavior observed in Table 9 indicates that IT2FRIO retains a practical advantage as dimensionality scales, although this difference has not yet reached statistical significance.

The statistical analysis of Table 14 for 200 dimensions shows that IT2FRIO exhibits statistically favorable differences compared to RIO, FRIO, PSO, and CS in most of the evaluated functions, confirming its competitive performance in high dimensionality scenarios.

In unimodal functions (F1–F3), IT2FRIO maintains consistent advantages, indicating effective exploitation even when the search space increases significantly. This result reflects adequate stability of the adaptive mechanism under conditions of greater complexity.

For multimodal functions (F4–F13), IT2FRIO dominates in most cases, particularly compared to PSO and CS. However, no clear statistical advantages are observed in F6, F9, and F12. This behavior in F6 and F9 can be attributed to the presence of flat regions and nearby optima, which favor algorithms with more aggressive exploration or dynamics less dependent on gradual adaptation.
In the case of F9, the regularly spaced local optima create a highly structured multimodal surface where IT2FRIO gradual adaptation can lead to over-exploitation of a non-optimal basin before the fuzzy mechanism accumulates sufficient information to redirect the search, giving more disruptive methods a situational advantage at this dimensionality. Even so, these cases do not alter the overall observed behavior, as most comparisons continue to support IT2FRIO superior performance in large scale problems.

Table 15 presents the results for 500 dimensions, where the scalability differences between the algorithms become more evident. In this scenario, IT2FRIO maintains lower average values and reduced variability in most functions, suggesting stable behavior even at high dimensionality. In unimodal functions, IT2FRIO maintains consistent performance, while its advantage is more pronounced in multimodal and non-separable functions. RIO and FRIO show acceptable performance in some cases, but their performance is progressively surpassed as dimensionality increases. PSO and CS, on the other hand, exhibit a considerable loss of effectiveness and greater instability.

Table 16 presents the results of the Z-test for 500 dimensions, evaluating the performance of IT2FRIO against the compared algorithms. Negative Z-values predominate in most functions (F1, F3, F5, F6, F8, F10–F13), confirming that IT2FRIO maintains a statistically significant advantage even in high dimensionality scenarios.
In F4 and F9, positive values are observed against some algorithms, indicating that IT2FRIO does not dominate in these particular cases. This behavior can be attributed to the specific landscape characteristics of these functions: F4 presents a deceptive structure where the global optimum is geometrically distant from the most promising local optima, strongly rewarding wide range exploration in early stages a dynamic that fixed, aggressive search strategies can exploit more directly than a gradually adapting

mechanism. F9, as discussed above, remains challenging due to its regularly structured multimodal surface. In F7, although IT2FRIO outperforms most approaches, there is a specific exception, associated with the high complexity and multimodality of the search space.

Overall, the statistical analysis confirms that IT2FRIO exhibits superior and consistent performance at a large scale, and that the observed isolated deviations do not alter the overall trend identified.

Table 17 presents the Wilcoxon signed-rank test results for 50 dimensions. IT2FRIO achieves statistically significant superiority in 49 of 52 comparisons (94.2%), with only three non-significant cases: F3 vs. RIO and PSO, and F13 vs. FRIO. These exceptions are consistent with the landscape characteristics of F3, whose nearly flat structure with a single global basin reduces the benefit of adaptive parameter control at low dimensionality.

Table 18 extends the analysis to 100 dimensions, where significance is maintained in 48/52 comparisons (92.3%), with isolated non-significant results for F12 vs. FRIO, F3 and F8 vs. RIO, and F6 vs. CS. The lack of significance for F8 may reflect its smooth, bowl-shaped landscape, where adaptive parameter control offers limited additional guidance over fixed configurations.

Table 19 presents the case of 200 dimensions, revealing the greatest variability across all evaluated dimensions, with 42/52 significant comparisons (80.8%). The most notable pattern emerges in the comparison with RIO, where only 7 of 13 functions reach statistical significance. Functions F2, F3, F5, F6, F11, and F12 do not reach significance in this dimension, suggesting that, at an intermediate scale, RIO becomes competitive in functions with smooth or separable landscapes, where the fuzzy adapter provides a minor directional benefit. In contrast, IT2FRIO maintains a perfect score of 13/13 against CS, confirming that its advantage over algorithms without adaptive control remains robust regardless of dimensionality.

Table 20 shows the results for D = 500, where statistical significance recovers to 47/52 comparisons (90.4%). IT2FRIO achieves a perfect score of 13/13 against FRIO, demonstrating that the type 2 interval extension offers a consistent advantage over its type 1 counterpart in high dimensionality. The three non-significant cases against RIO F1, F6, and F12 are consistent with the runtime convergence observed in the computational analysis, where both algorithms exhibit nearly identical runtimes at large scale. Against PSO, only F10 is non-significant, which can be attributed to the strong variable coupling structure of the Wood function, favoring PSO's social learning mechanism in high dimensions.

10 Conclusions

Experimental results and statistical analysis confirm that IT2FRIO exhibits robust, consistent, and scalable performance compared to benchmark algorithms, particularly as the problem's dimensionality increases. The results systematically demonstrate that incorporating an adaptation mechanism based on interval based fuzzy logic Type-2 effectively helps stabilize the balance between exploration and exploitation, a key aspect in solving complex, large scale optimization problems.

In 50 and 100 dimensional scenarios, IT2FRIO consistently achieves lower average values and, in many cases, less variability, reflecting not only improved solution quality but also greater stability in the search process. These observations are supported by statistical tests, which confirm that the differences favoring IT2FRIO are statistically significant in most of the evaluated functions. This behavior holds true for both unimodal functions, where efficient exploitation is crucial, and multimodal functions, where preserving diversity and avoiding local optima play a fundamental role.

As dimensionality increases, the performance differences between the algorithms become more pronounced. While PSO and CS show progressive degradation reflected in high average values and greater dispersion, IT2FRIO maintains competitive and stable performance. In this context, the fuzzy Type-2 adapter demonstrates a greater capacity to manage the uncertainty and variability inherent in high dimensional search spaces, allowing for more flexible and precise tuning of the algorithm's parameters compared to schemes based on Type-1 logic or static configurations.

Statistical analysis confirms that IT2FRIO superiority is not circumstantial but is consistently maintained across most of the functions considered. In particular, in multimodal functions, where the balance between global exploration and local exploitation is critical, the advantages of IT2FRIO are more pronounced, validating the central objective of the proposed adapter. The exceptions observed in specific functions do not alter the general trend and can be attributed to particular characteristics of the search landscape that favor more rigid or highly exploitative strategies.

Overall, the results confirm that IT2FRIO achieves a more effective balance between exploration and exploitation, exhibits greater robustness to increasing dimensionality, and demonstrates a superior capacity for handling highly complex functions. This supports the notion that using interval Type-2 fuzzy logic as a parameter adaptation mechanism constitutes a sound and effective strategy for improving the performance of metaheuristic algorithms in high dimensional optimization problems, fully meeting the objectives set forth in this work. As future work, we propose exploring more expressive adaptation schemes based on generalized Type-2 fuzzy logic, following approaches reported in complex parameter optimization (Güven & Kumbasar, 2025). This would allow for more precise modeling of uncertainty in membership degrees and boundaries, and improve robustness against noise and variability. Furthermore, we plan to extend this mechanism to other metaheuristic algorithms and evaluate it in high dimensional real-world problems, including genomic parameter optimization (Riza et al., 2024) and pharmacokinetic estimation (Kim et al., 2021), scenarios where similar approaches have shown promising results.

References

Almaraashi, M. S. (2024). A practical design of interval type-2 fuzzy logic systems with application to solar radiation prediction. *Cogent Engineering, 11*(1), Article 2395426. https://doi.org/10.1080/23311916.2024.2395426

Amador-Angulo, L., Castillo, O., Melin, P., & Castro, J. R. (2022). Interval type-3 fuzzy adaptation of the bee colony optimization algorithm for optimal fuzzy control of an autonomous mobile robot. *Micromachines, 13*(9), Article 1490. https://doi.org/10.3390/mi13091490

Anari, Z., Hatamlou, A., & Anari, B. (2022). Automatic finding trapezoidal membership functions in mining fuzzy association rules based on learning automata. *International Journal of Interactive Multimedia and Artificial Intelligence, 7*(4), 27–43. https://doi.org/10.9781/ijimai.2022.01.001

Asianuba, I., & Precious, D. (2023). Cockroach swarm optimization for side lobe level reduction in linear array antenna. *SSRG International Journal of Electronics and Communication Engineering, 10*(3), 23–29. https://doi.org/10.14445/23488549/IJECE-V10I3P104

Barraza, J., Rodríguez, L., Castillo, O., Melin, P., & Valdez, F. (2024). An enhanced fuzzy hybrid of fireworks and grey wolf metaheuristic algorithms. *Axioms, 13*(7), Article 424. https://doi.org/10.3390/axioms13070424

Bensadok, A., & Babar, M. Z. (2024). *Fuzzy hyperparameters update in a second-order optimization* [Preprint]. arXiv. https://arxiv.org/abs/2403.15416

Brindha, S., & Joe Amali, S. M. (2021). A robust and adaptive fuzzy logic based differential evolution algorithm using population diversity tuning for multi-objective optimization. *Engineering Applications of Artificial Intelligence, 102*, Article 104240. https://doi.org/10.1016/j.engappai.2021.104240

Cara, A. B., Wagner, C., Hagras, H., Pomares, H., & Rojas, I. (2013). Multiobjective optimization and comparison of nonsingleton type-1 and singleton interval type-2 fuzzy logic systems. *IEEE Transactions on Fuzzy Systems, 21*(3), 459–476. https://doi.org/10.1109/TFUZZ.2012.2236096

Celik, E. (2024). Analyzing the shelter site selection criteria for disaster preparedness using best–worst method under interval type-2 fuzzy sets. *Sustainability, 16*(5), Article 2127. https://doi.org/10.3390/su16052127

Chen, Y., & Shang, N. (2021). Comparison of GA, ACO algorithm, and PSO algorithm for path optimization on free-form surfaces using coordinate measuring machines. *Engineering Research Express, 3*(4), Article 045039. https://doi.org/10.1088/2631-8695/ac3e13

Cheng, L., Chang, L., Song, Y., Wang, H., & Bian, Y. (2023). A robot path planning method based on synergy behavior of cockroach colony. *The International Arab Journal of Information Technology, 20*(5), 717–726. https://doi.org/10.34028/iajit/20/5/4

Cheng, L., Lyu, C., Song, Y., Wang, H., Xu, Y., & Bian, Y. (2021). A bionic optimization technique with cockroach biological behavior. *Chinese Journal of Electronics, 30*, 644–651. https://doi.org/10.1049/cje.2021.05.006

Cuevas, F., Castillo, O., & Cortés-Antonio, P. (2022). Generalized type-2 fuzzy parameter adaptation in the marine predator algorithm for fuzzy controller parameterization in mobile robots. *Symmetry, 14*(5), Article 859. https://doi.org/10.3390/sym14050859

Dragoi, E. N., & Dafinescu, V. (2021). Review of metaheuristics inspired from the animal kingdom. *Mathematics, 9*(18), Article 2335. https://doi.org/10.3390/math9182335

Duan, S., Jiang, S., Dai, H., Wang, L., & He, Z. (2023). The applications of hybrid approach combining exact method and evolutionary algorithm in combinatorial optimization. *Journal of Computational Design and Engineering, 10*, 934–946. https://doi.org/10.1093/jcde/qwad029

Dutta, B., García-Zamora, D., Figueira, J. R., & Martínez, L. (2025). *Building interval type-2 fuzzy membership function: A deck of cards–based co-constructive approach* [Preprint]. arXiv. https://arxiv.org/abs/2503.01413

Ezzat, M., Hefny, H. A., & Mohmmed, A. (2024). Type 1 and Type 2 Fuzzy Logic for Enhance Wi-Fi Direct Performance in Vehicular Communication. In *Proceedings of the 56th Annual Conference on Data Sciences, Faculty of Graduate Studies for Statistical Research, Cairo University, 1*(1). https://doi.org/10.5281/zenodo.14551025

Gad, A. G. (2022). Particle swarm optimization algorithm and its applications: A systematic review. *Archives of Computational Methods in Engineering, 29*, 2531–2561. https://doi.org/10.1007/s11831-021-09694-4

Guerrero, M., Valdez, F., Castillo, O., & Melin, P. (2022). Comparative study between type-1 and interval type-2 fuzzy systems in parameter adaptation for the cuckoo search algorithm. *Symmetry, 14*(11), Article 2289. https://doi.org/10.3390/sym14112289

Güven, Y., & Kumbasar, T. (2025). Adapting GT2-FLS for uncertainty quantification: A blueprint calibration strategy. In *2025 IEEE International Conference on Fuzzy Systems (FUZZ)* (pp. 1–6). IEEE. https://doi.org/10.1109/FUZZ62266.2025.11152253

Hasan, R. A., Najim, S. S., & Ahmed, M. A. (2021). Correlation with the fundamental PSO and PSO modifications to be hybrid swarm optimization. *Iraqi Journal for Computer Science and Mathematics, 2*(2), 25–32. https://doi.org/10.52866/ijcsm.2021.02.02.004

Havens, T. C., Spain, C. J., Salmon, N. G., & Keller, J. M. (2008). Roach infestation optimization. In *Proceedings of the IEEE Swarm Intelligence Symposium* (pp. 1–7). IEEE. https://doi.org/10.1109/SIS.2008.4668317

Hepworth, A. J., Hussein, A., Reid, D. J., & Abbass, H. A. (2023). Swarm analytics: Designing information markers to characterise swarm systems in shepherding contexts. *Adaptive Behavior, 31*(4), 323–349. https://doi.org/10.1177/10597123221137090

Herrera-Franklin, J., Rosete, A., Sosa-Gómez, G., & Rojas, O. (2024). A metaheuristic approach for a two-dimensional fuzzy version of the variable size and cost bin packing problem. *International Journal of Computational Intelligence Systems, 17*, Article 281. https://doi.org/10.1007/s44196-024-00693-4

Hu, H., Fan, X., & Wang, C. (2024). Energy efficient clustering and routing protocol based on quantum particle swarm optimization and fuzzy logic for wireless sensor networks. *Scientific Reports, 14*, Article 18595. https://doi.org/10.1038/s41598-024-69360-0

Ibrahim, O., Mohd Junaidi, A. A., Ayop, R., Dahiru, A. T., Low, W. Y., Mohd Herwan, S., & Amosa, T. I. (2024). Fuzzy logic-based particle swarm optimization for integrated energy management system considering battery storage degradation. *Results in Engineering, 24*, Article 102816. https://doi.org/10.1016/j.rineng.2024.102816

Jafari, S. (2024). Research on fuzzy logic and mathematics with applications. *Symmetry, 16*(12), Article 1684. https://doi.org/10.3390/sym16121684

Jahanshahi, H., Yousefpour, A., Soradi-Zeid, S., & Castillo, O. (2026). A review on design and implementation of type-2 fuzzy controllers. *Mathematical Methods in the Applied Sciences, 49*(3), 1814–1835. https://doi.org/10.1002/mma.8492

Johnvictor, A. C., Amalanathan, A. J., Pariti Venkata, R. M., & Jethi, N. (2022). Critical review of bio-inspired data optimization techniques: An image steganalysis perspective. *WIREs Data Mining and Knowledge Discovery, 12*(4), Article e1460. https://doi.org/10.1002/widm.1460

Kasmi, B., & Hassam, A. (2021). Comparative study between fuzzy logic and interval type-2 fuzzy logic controllers for the trajectory planning of a mobile robot. *Engineering, Technology & Applied Science Research, 11*(2), 7011–7017. https://doi.org/10.48084/etasr.4031

Kim, S., Hooker, A. C., Shi, Y., Kim, G. H. J., & Wong, W. K. (2021). Metaheuristics for pharmacometrics. *CPT: Pharmacometrics & Systems Pharmacology, 10*(11), 1297–1309. https://doi.org/10.1002/psp4.12714

Koklu, A., Guven, Y., & Kumbasar, T. (2024). *Enhancing interval type-2 fuzzy logic systems: Learning for precision and prediction intervals* [Preprint]. arXiv. https://doi.org/10.48550/arXiv.2404.12802

Liu, R., Mo, Y., Lu, Y., Lyu, Y., Zhang, Y., & Guo, H. (2022). Swarm-intelligence optimization method for dynamic optimization problem. *Mathematics, 10*(11), Article 1803. https://doi.org/10.3390/math10111803

Lizarraga, E., Valdez, F., Melin, P., & Castillo, O. (2025). A hybrid enhanced mayfly optimization algorithm with improved performance through fuzzy-based automatic parameter adaptation. *Computación y Sistemas, 29*(2), 615–631. https://doi.org/10.13053/CyS-29-2-5709

Ma, Y., Wang, X., & Meng, W. (2024). A reinforced whale optimization algorithm for solving mathematical optimization problems. *Biomimetics, 9*(9), Article 576. https://doi.org/10.3390/biomimetics9090576

Milner, E., Sooriyabandara, M., & Hauert, S. (2023). *Swarm performance indicators: Metrics for robustness, fault tolerance, scalability and adaptability* [Preprint]. arXiv. https://doi.org/10.48550/arXiv.2311.01944

Miramontes, I., & Melin, P. (2022). Interval type-2 fuzzy approach for dynamic parameter adaptation in the bird swarm algorithm for the optimization of fuzzy medical classifier. *Axioms, 11*(9), Article 485. https://doi.org/10.3390/axioms11090485

Mittal, K., Jain, A., & Vaisla, K. S. (2020). A comprehensive review on type-2 fuzzy logic applications: Past, present and future. *Engineering Applications of Artificial Intelligence, 95*, Article 103916. https://doi.org/10.1016/j.engappai.2020.103916

Moloodpoor, M., & Mortazavi, A. (2025). A comparative review of fuzzy reinforced search algorithms: Methods and applications. *Archives of Computational Methods in Engineering, 32*, 3933–3977. https://doi.org/10.1007/s11831-025-10259-y

Mortazavi, A. (2024). A novel type-2 decision mechanism for dynamic parameter adaptation: Theory and application in mathematical and structural problems. *Neural Computing and Applications, 36*, 19729–19757. https://doi.org/10.1007/s00521-024-10176-4

Mwaura, J., Engelbrecht, A. P., & Nepomuceno, F. V. (2021). Diversity measures for niching algorithms. *Algorithms, 14*(2), Article 36. https://doi.org/10.3390/a14020036

Nishanth, F. P., Dash, S. K., & Mahapatro, S. R. (2024). Critical study of type-2 fuzzy logic control from theory to applications: A state-of-the-art comprehensive survey. *e-Prime – Advances in Electrical Engineering, Electronics and Energy, 10*, Article 100771. https://doi.org/10.1016/j.prime.2024.100771

Obagbuwa, I. C., & Adewumi, A. O. (2014). A modified roach infestation optimization. In *2014 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB)* (pp. 1–7). IEEE. https://doi.org/10.1109/CIBCB.2014.6845498

Oliveira, M., Pinheiro, D., Macedo, M., Bastos-Filho, C., & Menezes, R. (2020). Uncovering the social interaction network in swarm intelligence algorithms. *Applied Network Science, 5*, Article 24. https://doi.org/10.1007/s41109-020-00260-8

Paik, B., & Mondal, S. K. (2021). Representation and application of fuzzy soft sets in type-2 environment. *Complex & Intelligent Systems, 7*(3), 1597–1617. https://doi.org/10.1007/s40747-021-00286-0

Parouha, R. P., & Verma, P. (2021). Design and applications of an advanced hybrid meta-heuristic algorithm for optimization problems. *Artificial Intelligence Review, 54*, 5931–6010. https://doi.org/10.1007/s10462-021-09962-6

Parouha, R. P., & Verma, P. (2022). A systematic overview of developments in differential evolution and particle swarm optimization with their advanced suggestion. *Applied Intelligence, 52*, 10448–10492. https://doi.org/10.1007/s10489-021-02803-7

Patel, H. R. (2024). Lévy distribution meta-heuristic fuzzy-based optimization algorithm for optimal framework design of type-2 fuzzy controller subject to perturbations. *Proceedings, 105*(1), Article 29. https://doi.org/10.3390/proceedings2024105029

Paz, F., Leguizamón, G., & Montes, E. M. (2021). Cooperative coevolutionary particle swarms using fuzzy logic for large-scale optimization. *Journal of Computer Science & Technology, 21*, Article e11. https://doi.org/10.24215/16666038.21.e11

Pradhan, C., Senapati, M. K., Ntiakoh, N. K., & Calay, R. K. (2022). Roach infestation optimization MPPT algorithm for solar photovoltaic system. *Electronics, 11*(6), Article 927. https://doi.org/10.3390/electronics11060927

Riza, L. S., Prasetyo, Y., Zain, M. I., Siregar, H., Megasari, R., Hidayat, T., Kusumawaty, D., & Rosyda, M. (2024). Comparative study of population-based metaheuristic algorithms in case study of DNA sequence assembly. *International Journal of Bioautomation, 28*(3), 133–150. https://doi.org/10.7546/ijba.2024.28.3.000976

Selvarajan, S. (2024). A comprehensive study on modern optimization techniques for engineering applications. *Artificial Intelligence Review, 57*, Article 194. https://doi.org/10.1007/s10462-024-10829-9

Singh, R., Jain, K., & Pandit, M. (2011). Comparison of PSO variants with traditional solvers for large-scale multi-area economic dispatch. In *Proceedings of the International Conference on Sustainable Energy and Intelligent Systems (SEISCON)*. https://doi.org/10.1049/cp.2011.0379

Starczewski, J. T., Przybyszewski, K., Byrski, A., Szmidt, E., & Napoli, C. (2022). A novel approach to type-reduction and design of interval type-2 fuzzy logic systems. *Journal of Artificial Intelligence and Soft Computing Research, 12*, 197–206. https://doi.org/10.2478/jaiscr-2022-0013

Sutikno's article is listed in *Babylonian Journal of Mathematics*, pages 59–65, with the stated DOI. Sutikno, T. (2023). Fuzzy optimization and metaheuristic algorithms. *Babylonian Journal of Mathematics, 2023*, 59–65.

Takahashi, A., & Takahashi, S. (2021). A new interval type-2 fuzzy logic system under dynamic environment: Application to financial investment. *Engineering Applications of Artificial Intelligence, 100*, Article 104154. https://doi.org/10.1016/j.engappai.2021.104154

Tang, H. H., & Ahmad, N. S. (2024). Fuzzy logic approach for controlling uncertain and nonlinear systems: A comprehensive review of applications and advances. *Systems Science & Control Engineering, 12*(1), Article 2394429. https://doi.org/10.1080/21642583.2024.2394429

Too, J., & Abdullah, A. R. (2021). A new and fast rival genetic algorithm for feature selection. *The Journal of Supercomputing, 77*, 2844–2874. https://doi.org/10.1007/s11227-020-03378-9

Torres-Salinas, H., Rodríguez-Reséndiz, J., Cruz-Miguel, E. E., & Ángeles-Hurtado, L. A. (2022). Fuzzy logic and genetic-based algorithm for a servo control system. *Micromachines, 13*(4), Article 586. https://doi.org/10.3390/mi13040586

Tsai, H. C. (2015). Roach infestation optimization with friendship centers. *Engineering Applications of Artificial Intelligence, 39*, 109–119. https://doi.org/10.1016/j.engappai.2014.12.003

Valdez, F., Castillo, O., & Melin, P. (2021). Bio-inspired algorithms and its applications for optimization in fuzzy clustering. *Algorithms, 14*(4), Article 122. https://doi.org/10.3390/a14040122

Vidal-Martínez, R., García-Martínez, J. R., Rojas-Galván, R., Álvarez-Alvarado, J. M., González-Lee, M., & Rodríguez-Reséndiz, J. (2025). A review of Mamdani, Takagi–Sugeno, and type-2 fuzzy controllers for MPPT and power management in photovoltaic systems. *Technologies, 13*(9), Article 422. https://doi.org/10.3390/technologies13090422

Vincent, A. M., & Jidesh, P. (2023). An improved hyperparameter optimization framework for AutoML systems using evolutionary algorithms. *Scientific Reports, 13*, Article 4737. https://doi.org/10.1038/s41598-023-32027-3

Wang, J., Wang, K., Yan, X., & Wang, C. (2022). A hybrid learning particle swarm optimization with fuzzy logic for sentiment classification problems. *International Journal of Cognitive Informatics and Natural Intelligence, 16*(1), 1–23. https://doi.org/10.4018/IJCINI.314782

Wang, Z., Qin, C., Wan, B., & Song, W. W. (2021). A comparative study of common nature-inspired algorithms for continuous function optimization. *Entropy, 23*(7), Article 874. https://doi.org/10.3390/e23070874

Xiang, L., Sang, H., & Qu, F. (2021). A type-2 fuzzy logic–based maintenance solution for power system in renewable energy applications. *Frontiers in Energy Research, 9*, Article 762360. https://doi.org/10.3389/fenrg.2021.762360

Yiğit, F. (2023). A novel type-2 hexagonal fuzzy logic approach for predictive safety stock management for a distribution business. *Scientific Reports, 13*, Article 19835. https://doi.org/10.1038/s41598-023-46649-0

Yuan, K., Li, W., Xu, W., Zhan, T., Zhang, L., & Liu, S. (2021). A comparative experimental evaluation on performance of type-1 and interval type-2 Takagi–Sugeno fuzzy models. *International Journal of Machine Learning and Cybernetics, 12*, 2135–2150. https://doi.org/10.1007/s13042-021-01298-5