

AN ADAPTIVE OPTIMIZATION STRATEGY APPLIED TO THE ARTIFICIAL BEE COLONY ALGORITHM

Oussama Ait Sahed¹, Mohamed Lamine Fas¹, Kamel Kara¹, and Mohamed Benrabah²

¹LabSET laboratory - Department of Automatics
University of Blida 1
Blida, Algeria

*Corresponding author's e-mail: aitsahed.oussama@univ-blida.dz

²Laboratory of Robotics - Parallelism and Embedded Systems
University of Science and Technology Houari Boumediene
Algiers, Algeria

Balancing exploration and exploitation remains a central challenge in metaheuristic optimization, particularly for engineering applications requiring both global search efficiency and reliable convergence. This paper introduces a simple and adaptive strategy that dynamically adjusts the behavior of an optimization algorithm based on real-time performance feedback. The method identifies key search operators and assigns each an explorative and exploitative variant, grouped into two optimization modes. An online mechanism switches between these modes according to the successful mutation rate, enabling the optimizer to respond to the evolving landscape of the search process. The strategy is integrated into the artificial bee colony algorithm and evaluated on 18 numerical benchmarks, a CEC2021 test suite, and a nonlinear model predictive control problem. Results demonstrate improved robustness, competitive accuracy, and faster convergence by reducing the number of evaluations by more than 4% against the established GABC, highlighting the method's suitability for complex industrial optimization tasks.

Keywords: Adaptive Algorithms, Computational Intelligence, Optimization Methods, Model Predictive Control.

(Received on January 7, 2026; Accepted on August 2, 2026)

1. INTRODUCTION

In the design of metaheuristic optimization algorithms, it is essential to achieve an effective integration of exploration and exploitation mechanisms. Exploration enables the algorithm to identify and locate the regions of interest (global search) where high-quality solutions may exist, while exploitation focuses on the discovered regions in order to refine and enhance the solutions. Insufficient exploration may cause the optimizer to overlook interesting regions in the search space where an optimum may reside and only locate suboptimal regions. Conversely, inadequate exploitation may make the optimizer overlook good solutions and converge towards inferior suboptimal solutions.

We do not aim to address the traditional concept of balancing exploration and exploitation capabilities where modifications are made during algorithm design or by tuning one or more Search Operators (SO). What we are proposing is real-time, performance-driven switching at the mode level, where entire sets of SO variants are swapped based on online performance feedback. This mode-level switching remains underexplored. The proposed method introduces a lightweight Successful Mutation Rate (SMR) based approach that monitors short-term improvement rates and switches between exploration and exploitation modes accordingly. This mode-level switching differs from per-parameter adaptive schemes and portfolio methods by emphasizing low computational overhead, interpretability, and suitability for time-constrained optimization tasks. A good survey that covers this topic within the Evolutionary Algorithms (EA) community could be found in (Črepinšek *et al.*, 2013; Karafotias *et al.*, 2015).

The Artificial Bee Colony (ABC) optimization algorithm (Karaboga & Basturk, 2007) is considered to be one of the most efficient optimizers (Hussain *et al.*, 2020). In fact, it is known to be easy to implement (Cui *et al.*, 2017), has fewer control parameters (Cui *et al.*, 2024) and has a simple structure with good performance (Zhu *et al.*, 2024). Nonetheless, several shortcomings have been cited, especially its slow convergence rate (Kong *et al.*, 2018) and poor exploitation capabilities (Gao *et al.*, 2019).

The SOs that significantly influence the behavior of the ABC algorithm are the update equation used to generate new solutions, the probability equation used by the onlookers to determine which regions of the search space to further exploit, and the scout bee strategy used to introduce random solutions into the population. Two variants are used for each of the

aforementioned SOs. One variant will help increase the exploratory capabilities of the optimization algorithm, while the other will enhance its exploitation capabilities. These variants will be grouped into two optimization modes. The variants that help increase exploration will be grouped under the exploration mode, while the others will be grouped under the exploitation mode. The online adaptive strategy will continuously select which mode to use based on the current optimization status.

Generally, an efficient optimizer requires more exploration (less exploitation) when starting the optimization process and more exploitation (less exploration) as this process discovers interesting regions within the search space. As such, the proposed approach will start in exploration mode, where the different SOs will use their variants that help increase the exploration capabilities of the overall optimization algorithm. After each given number of iterations, defined here as the Tuning Cycle (TC), the SMR will be computed. As long as the SMR continues to increase, the algorithm keeps the optimization mode as it is. However, if the SMR decreases, the optimization mode will be changed.

Adaptive metaheuristics are quite important within the industrial engineering framework, where repeating time-critical optimization problems with evolving process conditions must be solved as efficiently as possible with regards to the solution quality and computing requirements. These demands are particularly evident in Nonlinear Model Predictive Control (NMPC), a widely used control technique within the chemical, petrochemical and food processing industries (Badgwell & Qin, 2013), where an optimization problem needs to be solved at each sampling period.

In order to investigate the optimizer efficiency, an open source project was established in GitHub, where several experiments are carried out comparing the proposed approach against several established and relatively new ABC variants, in particular the basic ABC by Karaboga (2005), the well-known GABC (Zhu & Kwong, 2010), the MABC by Akay and Karaboga (2012), the Directed ABC (DABC) by Kiran and Findik (2015), the ranking-based adaptive ABC dubbed here ARABC by Cui *et al.* (2017) and the improved ABC with adaptive parameter dubbed here IABC by Zhao *et al.* (2022). These experiments are performed using a set of 18 numerical benchmark functions of various characteristics, commonly used for performance evaluation and comparison, and the ten benchmark functions proposed in the CEC2021 special session on single-objective bound-constrained numerical optimization (Mohamed *et al.*, 2020), in addition to the implementation of a nonlinear model predictive controller. We emphasize that all source code, benchmark functions, and parameter settings used in our experiments are publicly available at <https://github.com/Aitsahed/MHPlatTest>, where researchers can replicate and verify our findings.

The paper starts by providing a concise overview of the ABC algorithm and some of its related works. The motivation behind the work and the in-depth explanation of the proposed algorithm are presented in Section 3. The empirical studies covering the different investigated aspects are given in the next section. The conclusion of this paper can be found in Section 5.

2. ABC ALGORITHM AND RELATED WORK

2.1. ABC Algorithm

The ABC is a meta-heuristic optimization algorithm that was introduced by Karaboga in 2005 (Karaboga, 2005). The artificial bees are split into three subgroups: employed, onlooker and scout bees. The artificial bees within the employed and onlooker groups are used to both explore the search space and exploit the different existing food sources, whereas the scouts are used to explore completely new positions. Each food source corresponds to a potential solution to the optimization problem. The quality of a food source, which is determined by the amount of nectar it contains, directly corresponds to the associated solution's fitness. In other words, a food source with more nectar represents a better solution. The total population, represented by PN , is evenly distributed between the employed bees (denoted as SN), which actively search for food sources, and the onlooker bees (denoted as LN), which exploit the existing food sources. The solutions (positions of the food sources) are represented by Equation (1), where D represents the number of optimization parameters.

$$X_h = \{x_{h1}, \dots, x_{hD}\} (h=1, \dots, SN). \quad (1)$$

The first step is to randomly initialize the food sources using Equation (2), where $j = 1, \dots, D$ and $x_{\max j}, x_{\min j}$ are the upper and lower bounds of the search space for dimension j . α_{hj} is randomly chosen in $[0, 1]$.

$$x_{hj} = x_{\min j} + \alpha_{hj}(x_{\max j} - x_{\min j}). \quad (2)$$

Each employed bee will exploit a single solution X_h by producing a candidate solution V_h in its neighborhood by updating one randomly selected optimization parameter within X_h as shown in Equation (3), where $j \in \{1, \dots, D\}$, ϕ_{hj} is chosen randomly in $[-1, 1]$ and X_k is a randomly selected solution not equal to X_h .

$$v_{hj} = x_{hj} + \phi_{hj}(x_{hj} - x_{kj}). \quad (3)$$

The newly generated candidate solution V_h is evaluated by its employed bee and compared to the old solution X_h . The solution X_h is replaced by V_h if the new candidate food source is better; otherwise V_h is forgotten.

Once the employed bees have completed their update process, each one of them tries to attract as many onlooker bees as possible by advertising the quality of their food sources. Based on this information, the onlookers further exploit the best existing solutions using a probability p_h associated with each food source using Equation (4), where $f_{fitness}(X_h)$ represents the fitness value of the solution X_h . It is possible that some food sources attract more than one onlooker, whereas others attract none.

$$p_h = \frac{f_{fitness}(X_h)}{\sum_{h=1}^{SN} f_{fitness}(X_h)}. \quad (4)$$

As soon as a given onlooker bee reaches its assigned food source X_h it will attempt to improve the food source by creating a new candidate solution V_h within its vicinity using Equation (3), and then performs greedy selection to keep only the best solution.

The algorithm keeps track of the unsuccessful continuous attempts to enhance each food source. If this number exceeds the limit parameter, this food source is considered to be exhausted and is replaced by a random solution (Equation (2)) using a scout bee. The commonly used value for the limit parameter is 100.

The artificial bees store the location of the best food source found by any bee in X_{best} .

2.2. ABC Variants

Although the ABC algorithm is a good optimizer, it remains susceptible to many shortcomings like any other optimization algorithm, namely its slow convergence rate (Kong *et al.*, 2018) and poor exploitation capabilities (Gao *et al.*, 2019).

Several works have been proposed to overcome these shortcomings, Zhu and Kwong (2010) have modified the update equation (Equation (3)), based on the Particle Swarm Optimization (PSO) algorithm, to incorporate information about the global best solution in order to improve the exploitation capabilities of the algorithm, whereas Gao and Liu (2012) have modified it to further exploit the vicinity of the best solution within the previous iteration. Akay and Karaboga (2012) have introduced the Modification Rate (MR) parameter to update more than one optimization parameter, subsequently increasing the convergence speed of the algorithm. They have also introduced the Scaling Factor (SF) to control the magnitude of the perturbation. Das *et al.* (2013) have modified the update equation by incorporating information from multiple top solutions while updating more than one optimization parameter. The probability equation used by the onlookers to choose which food sources to exploit has also been modified. Kiran and Findik (2015) have added directional information to the update equation, whereas Brajevic (2015) has modified the number of optimization parameters to update based on a dynamically adjusted MR parameter. He has also modified the update equation used by the onlooker bees and the way the scout bees are generated via a crossover operator. Sensitivity analysis has been employed by Loubière *et al.* (2016) to determine which optimization parameters have the most significant impact on the objective function. Based on this analysis, the search is then directed along these influential dimensions. Xiao *et al.* (2019) have changed the update equation by including information from a maintained archive containing the best solutions found so far. Brajević (2021), on the other hand, has used different equations for the employed and onlooker bees while introducing a shuffle mutation to be applied to all newly generated solutions in certain iterations. This parameter promotes diversity within the population. Zhao *et al.* (2022) have adopted two different update equations for the employed and onlooker bees, while the search step size of the onlooker bees is adaptively adjusted. Zhang *et al.* (2024) have modified the employed bee phase by introducing a crossover operation, and the onlooker phase by using a local search method.

Wang *et al.* (2014) have used three different search strategies to generate new solutions. Each food source uses a random search strategy at first, and only keeps this strategy if the newly generated solution has improved. Otherwise, another search strategy is randomly adopted. Lin *et al.* (2018) introduced changes to the behavior of the employed bees, restricting their interactions to their immediate surroundings. Meanwhile, the onlooker bees choose between two search strategies based on a calculated probability.

Others, such as Kang *et al.* (2009), have focused on combining the ABC algorithm with other methods to overcome and enhance the overall performance of the optimization algorithm. They have combined the ABC optimization algorithm with the Nelder–Mead simplex method to solve inverse analysis problems. Duan *et al.* (2009) have used the ABC algorithm with the quantum evolutionary algorithm to enhance exploitation and overcome the premature convergence limitations encountered when solving optimization problems. Sundar and Singh (2012) have used the ABC with a local search strategy

to solve the non-unicost Set Covering Problem (SCP), whereas Ivona *et al.* (2020) have used the ABC to perform local search and the firefly algorithm for global search. Sedighzadeh and Mazaheripour (2018) have combined the ABC with the PSO algorithm to solve the vehicle routing problem. This problem was also tackled by Comert and Yazgan (2023), who have combined the ABC with a hybrid ant colony optimization algorithm. Mallala *et al.* (2022) have combined the ABC algorithm with the fruit fly optimization algorithm by including the employed phase from the ABC algorithm and the oosphere foraging and the vision foraging phases from the fruit fly algorithm.

3. PROPOSED ADAPTIVE ALGORITHM

3.1. Motivations

As discussed in the introduction, exploration and exploitation are derived from the behavior of the underlying search operators. Some SOs boost exploration, whereas others boost exploitation. However, it is not uncommon to identify a single SO that could simultaneously influence both of them.

As an example, let us consider the ABC update equation. It is one of the most important SOs within the ABC algorithm. It is responsible for generating new solutions based on local and global information. Keep in mind that only one randomly selected optimization parameter is mutated to generate the new candidate solutions. This fact could be argued to greatly help explore the search space, since the solutions require more moves to converge towards the most prominent regions of the search space.

However, Akay and Karaboga (2012) and Das *et al.* (2013) also claim that choosing a single parameter to update could significantly decrease the optimizer's convergence speed. For this reason, they have introduced a parameter called the modification rate to increase the number of optimization parameters to update as a way to increase the exploitation capabilities of the optimizer. This has the effect of increasing the convergence speed, but with the drawback of increasing the possibility of being trapped in local optima.

We cannot claim which of the previously mentioned approaches will be the most efficient way to solve a given optimization problem, or more broadly, that a certain alteration to the optimizer, albeit explorative in nature or exploitative, will surely enhance the algorithm's performance. In fact, the No Free Lunch (NFL) theorem, as stated by Wolpert and Macready (1997), establishes that no optimizer outperforms any other when averaged across all conceivable optimization problems. Each method has its pros and cons and cannot be just 'better'.

We do not intend to address the traditional concept of balancing exploration and exploitation capabilities, where modifications are made to one or more search operators (Shen *et al.*, 2025). What we are proposing is an adaptive concept where the optimization algorithm could be driven to be more explorative when the optimization process requires more exploration and to have more exploitation capabilities when exploitation is needed.

This concept could be used with any optimization process or as a part of a larger framework for the development of optimization algorithms.

3.2. Search Operators

Each one of the considered SOs will have two variants: one variant drives the optimization algorithm to be more explorative, and the other helps increase the exploitation capabilities of the overall optimization algorithm.

Let us consider the following SOs:

3.2.1. Update Equation

The first SO to consider is the update equation (Equation (3)) used to generate new solutions. Two aspects within this SO are considered. The first aspect concerns the nature of the update equation. The newly generated solutions using Equation (3) will follow a random vector based on the randomly selected solution X_h . No already collected information about prominent regions of the search space is exploited in order to drive the search toward more interesting regions of the search space. One prominent work that was conducted by Zhu and Kwong (2010) proposed a well-known variant in which the update equation (Equation 3) has been modified based on the PSO algorithm to have the expression in Equation (5), where $x_{best\ j}$ is the J th dimension of the global best solution and ψ_{hj} is a random number in $[0, C]$. C represents a nonnegative parameter to be chosen by the practitioner. A value of $C = 1.5$ is recommended by the GABC authors (Zhu & Kwong, 2010).

$$v_{hj} = x_{hj} + \phi_{hj}(x_{hj} - x_{kj}) + \psi_{hj}(x_{best\ j} - x_{hj}). \quad (5)$$

This equation will bias the search toward the global best solution found so far; thus, increasing the exploitation in its vicinity. To this end, Equation (3) will be used within the exploration mode, whereas Equation (5) will be adopted within the exploitation mode.

The second aspect is the number of optimization parameters N_{upd} to mutate when generating the new solutions via the update equation. As long as more exploitation is required, the value of this parameter will be increased by 1. The update equation will not be applied to only one optimization parameter to generate the new possible solution but rather to N_{upd} randomly selected parameters. On the other hand, if more exploration capabilities are required, its value will be set to 1. Equation (6) will be used as the update equation instead of (3) when the exploitation mode is active, where R_h is a list of N_{upd} randomly selected indices from $\{1, 2, \dots, D\}$, and $1 \leq N_{upd} \leq D$.

$$\begin{cases} v_{hj} = x_{hj} + \phi_{hj}(x_{hj} - x_{kj}) + \psi_{hj}(x_{best\ j} - x_{hj}) & \text{if } j \in R_h \\ v_{hj} = x_{hj} & \text{otherwise} \end{cases} \quad (6)$$

while Equation (7) will be used when in exploration mode.

$$\begin{cases} v_{hj} = x_{hj} + \phi_{hj}(x_{hj} - x_{kj}) & \text{if } j \in R_h \\ v_{hj} = x_{hj} & \text{otherwise} \end{cases} \quad (7)$$

3.2.2. Probability Equation

The probability equation (Equation (4)) encourages the onlooker bees to exploit the most prominent regions of the search space discovered thus far, increasing the exploitation capabilities of the algorithm. The proposed equation in (8), on the other hand, makes the onlooker bees exploit the newly discovered food sources, which are generally of poor quality. In fact, Equation (8) ensures that low-quality food sources attract more onlookers, increasing diversity within the population. It could be argued that Equation (8) has a somewhat opposite behavior compared to Equation (4).

$$p_h = \frac{\frac{1}{f_{fitness}(x_h)}}{\sum_{h=1}^{SN} \frac{1}{f_{fitness}(x_h)}} \quad (8)$$

When we are in the exploration mode, the algorithm will use Equation (8) as the probability equation, whereas Equation (4) will be used in the exploitation mode.

3.2.3. Scout Bee Generation

The last SO to consider is the scout bee concept, where the algorithm replaces an existing solution, regardless of its quality, with a new one if it is found to be exhausted. Using Equation (2) replaces the discarded solution with a random point from the search space, thereby injecting new information into the population and increasing the optimizer's exploratory capability. On the other hand, Equation (9) utilizes existing solutions to generate the new solution, where $j = 1, \dots, D$ and $i \neq h$.

$$x_{hj} = \frac{\sum_{i=1}^{SN} x_{ij}}{(SN-1)} \quad (9)$$

When we are in the exploration mode, the algorithm will use Equation (2). Otherwise, Equation (9) will be used (exploitation mode).

3.3. Update Mechanism

The algorithm always starts in the exploration mode since more exploration is needed at the start of the optimization process. After completing a specified number of iterations, known as the tuning cycle, the SMR for these iterations is calculated using the following formula:

$$SMR = \frac{\text{Number of successful mutations}}{\text{Number of mutated solutions}} \quad (10)$$

This newly calculated SMR is compared to the previously computed SMR (which equals zero at the start of the optimization process). If the SMR is increasing, then the current optimization mode will be maintained. On the other hand, if the value of the SMR decreases, then the optimization mode will be switched.

The TC parameter controls how frequently the algorithm evaluates the SMR and therefore governs the responsiveness of the adaptive switching mechanism. A smaller TC value makes the algorithm more reactive, allowing it to detect changes in short-term improvement trends quickly and switch between exploration and exploitation modes more frequently. Conversely, a larger TC value results in more stable but less reactive switching behavior. In practice, TC acts as a trade-off between sensitivity and stability: it determines how much evidence the algorithm requires before altering its search behavior. The proposed approach uses a fixed TC value that was empirically found to provide a good balance between these two effects while maintaining low computational overhead.

To better understand this process, let us consider the following case where the optimization algorithm is in the exploitation mode. In this state, the algorithm will be using Equation (6) for the update equation, Equation (4) for the probability equation and Equation (9) for the scout bees. Once TC iterations have been completed, the new SMR will be computed. Comparing its value to the previously computed SMR, two cases emerge:

3.3.1. SMR is Increasing

The optimization mode will be maintained, i.e., the exploitation mode and the N_{upd} parameter, which controls the number of optimization parameters to mutate, will be increased by 1 (never above the dimensionality of the problem D). In this situation, equations (6), (4) and (9) will be adopted as the update equation, the probability equation and the scout bee generation equation, respectively. These equations will be used throughout the next TC iterations.

3.3.2. SMR is Decreasing

The optimization mode will be switched in this case to the exploration mode, and the N_{upd} parameter will be set to 1. Within this state, the algorithm will use Equation (7) for the update equation, Equation (8) for the probability computation and Equation (2) for the scout bee generation throughout the next TC iterations.

This tuning process will be run every TC iteration until the optimization has been stopped. Table 1 contains the details of each optimization mode.

Table 1. Details of Optimization Modes

Search Operator	Exploration Mode	Exploitation Mode
Update equation	Use Equation (7)	Use Equation (6)
Update equation	Set N_{upd} to 1	Increase N_{upd} by 1
Probability equation	Use Equation (8)	Use Equation (4)
Scout Bee equation	Use Equation (2)	Use Equation (9)

3.4. Flowchart of the Proposed Algorithm

The proposed adaptive ABC optimization algorithm is presented in the flowchart of Figure 1.

4. EXPERIMENTAL RESULTS

4.1. Numerical Benchmark Functions

The performance of the proposed approach is evaluated using two sets of numerical benchmark functions. The first batch, described in Table 2, is composed of 18 scalable benchmark functions of different types ($f_1 \rightarrow f_{18}$). The scalable functions are used with dimensions of $D=10$, $D=30$ and $D=50$. The second batch (Table 3) is comprised of 10 benchmark functions ($F_1 \rightarrow F_{10}$). These functions are defined in the CEC2021 test suite for the competition on single-objective bound-constrained numerical optimization with a search range of $[-100, 100]^D$ a dimension of $D=10$, and a global optimum at '0'. More information about the CEC2021 benchmark functions can be found in (Mohamed *et al.*, 2020).

Table 2. Numerical Benchmark Functions

Name	Function Scalable Numerical Benchmark Functions	Search Limit	Min
Ackley	$f_1 = 20 + \exp(1) - 20 \exp\left(-0.2\sqrt{(1/D) \sum_{i=1}^D x_i^2}\right) - \exp\left((1/D) \sum_{i=1}^D \cos(2\pi x_i)\right)$	$[-32.768, 32.768]^D$	0
Alpine	$f_2 = \sum_{i=1}^D x_i \sin(x_i) + 0.1x_i $	$[-10, 10]^D$	0
Discus	$f_3 = 10^6 x_1^2 + \sum_{i=2}^D x_i^2$	$[-100, 100]^D$	0
Expanded Schaffer's	$f_4 = g(x_1, x_2) + g(x_2, x_3) + \dots + g(x_{D-1}, x_D) + g(x_D, x_1)$ where $g(x, y) = 0.5 + (\sin^2(\sqrt{x^2 + y^2}) - 0.5)/(1 + 0.001(x^2 + y^2))^2$	$[-100, 100]^D$	0
Exponential	$f_5 = -\exp(-0.5 \sum_{i=1}^D x_i^2) + 1$	$[-1, 1]^D$	0
Griewank	$f_6 = \sum_{i=1}^D (x_i^2/4000) - \prod_{i=1}^D \cos(x_i/\sqrt{i}) + 1$	$[-600, 600]^D$	0
Hartmann	$f_7 = \sum_{i=1}^D x_i^2 - D ^{1/4} + (0.5 \sum_{i=1}^D x_i^2 + \sum_{i=1}^D x_i)/D + 0.5$	$[-100, 100]^D$	0
HGBat	$f_8 = \sqrt{(\sum_{i=1}^D x_i^2)^2 - (\sum_{i=1}^D x_i)^2} + (0.5 \sum_{i=1}^D x_i^2 + \sum_{i=1}^D x_i)/D + 0.5$	$[-5, 5]^D$	0
High Conditioned Elliptic	$f_9 = \sum_{i=1}^D (10^6)^{\frac{i-1}{D-1}} x_i^2$	$[-100, 100]^D$	0
Levy	$f_{10} = \sin^2(\pi \omega_1) + \sum_{i=1}^{D-1} [(\omega_i - 1)^2 (1 + 10 \sin^2(\pi \omega_i + 1))] + (\omega_D - 1)^2 (1 + \sin^2(2\pi \omega_D))$ where $\omega_i = 1 + ((x_i - 1)/4)$	$[-10, 10]^D$	0
Quartic	$f_{11} = \sum_{i=1}^D (ix_i^4) + \text{random}(0, 1)$	$[-1.28, 1.28]^D$	0
Rastrigin	$f_{12} = \sum_{i=1}^D (x_i^2 - 10 \cos(2\pi x_i) + 10)$	$[-5.12, 5.12]^D$	0
Rosenbrock	$f_{13} = \sum_{i=1}^{D-1} (100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2)$	$[-2.048, 2.048]^D$	0
Sphere	$f_{14} = \sum_{i=1}^D x_i^2$	$[-100, 100]^D$	0
SumSquares	$f_{15} = \sum_{i=1}^D i \cdot x_i^2$	$[-10, 10]^D$	0
Weierstrass	$f_{16} = \sum_{i=1}^D [\sum_{k=0}^{20} (0.5^k \cdot \cos(2\pi 3^k (x_i + 0.5)))] - D \sum_{k=0}^{20} (0.5^k \cos(2\pi \cdot 3^k \cdot 0.5))$	$[-10.24, 10.24]^D$	0
Whitley	$f_{17} = \sum_{i=1}^D \sum_{j=1}^D \left[\left[\frac{1}{4000} (100(x_i^2 - x_j)^2 + (1 - x_j)^2)^2 \right] - \cos(100(x_i^2 - x_j)^2 + (1 - x_j)^2) + 1 \right]$	$[-10.24, 10.24]^D$	0
Zakharov	$f_{18} = \sum_{i=1}^D x_i^2 + (\sum_{i=1}^D 0.5 \cdot i \cdot x_i)^2 + (\sum_{i=1}^D 0.5 \cdot i \cdot x_i)^4$	$[-5, 10]^D$	0

All experiments were conducted using a custom C# application developed in Microsoft Visual Studio Community 2022. The exact source code used to generate the results in this paper is publicly available in the GitHub repository at <https://github.com/Aitsahed/MHPlatTest> under commit e04ac12. All benchmark runs were executed on a Windows 10 system equipped with an Intel i5-4440 processor and 32 GB of RAM. The Friedman test and the Wilcoxon signed-rank test with Holm correction were computed using MATLAB.

The performance of the adaptive ABC is evaluated against six ABC variants, namely the basic ABC (Karaboga, 2005), the well-known GABC (Zhu & Kwong, 2010), the MABC (Akay & Karaboga, 2012), the DABC (Kiran & Findik, 2015), the ranking-based adaptive ABC dubbed here ARABC by Cui *et al.* (2017) and the improved ABC with adaptive parameter dubbed here IABC by Zhao *et al.* (2022). All algorithms have a population of 40 artificial bees with a maximum function evaluation of 100,000. The 'limit' parameter is selected to be 100 for all algorithms except for the MABC and ARABC, for which the authors have recommended a value of 200 and $D * 40$, respectively. Furthermore, the modification rate, used within the MABC, is selected to be 0.4, whereas the C parameter within the GABC is taken to be 1.5 per the authors' recommendation. For the proposed algorithm, the numerical value for the TC parameter was selected to be 20. This value was found to provide good performance.

We are interested in the mean optimal function values and the standard deviation (STD) averaged over 100 independently executed runs. The best obtained result among all algorithms is highlighted in bold blue font, whereas a grey background is used when a given algorithm generates a better result than the proposed one. Please note that a function evaluation below '1E-16' is taken as '0'.

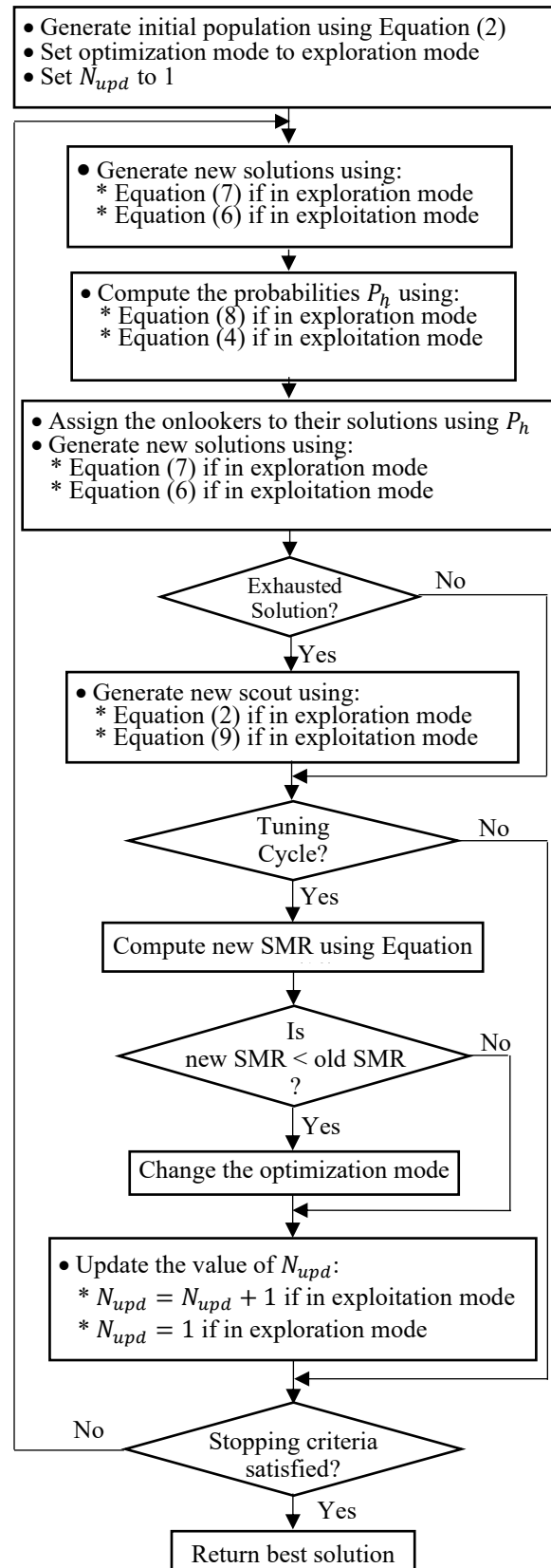


Figure 1. Flowchart of the Adaptive ABC Algorithm

The results are compiled in Table 4, where we can observe that the adaptive ABC has good performance compared to the combined results of the other six algorithms. In fact, it has the best results in 40 cases compared to only 15 for the GABC, 6 for the IABC, 4 for the DABC, 3 for the ARABC and only 2 for both the MABC and basic ABC. When comparing the proposed algorithm against the other six individually, we find that the adaptive ABC outperformed the MABC in 62 of 64 considered cases, the IABC in 58, the basic ABC in 57, the ARABC in 56, the DABC in 54 and the GABC in 45 cases.

We could also observe that the proposed adaptive ABC did not always generate the best results. In fact, it was significantly outperformed when dealing with f_2 for $D=30$ and 50 , where its obtained results are several orders of magnitude behind the best recorded results of the ARABC. For the remaining cases, such as with f_4 and f_7 , the adaptive ABC stays relatively close to the best solution.

For the CEC2021 benchmark functions ($F_1 \rightarrow F_{10}$) we could see that the proposed approach outperformed the other algorithms in 7 of the 10 considered cases, while the GABC did so in only 2 cases and the basic ABC in only one. The rest had none.

The Friedman test conducted on these results yielded a p-value of $2.21E-26$, providing overwhelming evidence against the null hypothesis and confirming that there is a statistically significant difference among the seven algorithms. The Friedman mean ranks further indicate that the proposed adaptive ABC achieves the best overall performance, with the lowest average rank of 1.76. To complement this global analysis, we also applied the Wilcoxon signed-rank test with Holm correction for pairwise comparisons. The corrected p-values range from $2.69E-10$ to $3.30E-02$, demonstrating that the proposed method significantly outperforms all six competitor algorithms. Five comparisons exhibit extremely strong statistical evidence ($p < 10^{-4}$) while the comparison with GABC ($p = 3.30E - 02$) indicates a moderate but still statistically significant advantage.

The following experiment assesses the convergence speed and success rate of the proposed approach in comparison to the other six algorithms. The experiment recorded the number of trials and the required number of function evaluations for the algorithm to converge to 0. Only cases where at least one optimization algorithm could converge within 100,000 function evaluations are displayed. The numerical values are obtained by averaging the results over 100 independent runs, whereas an obtained value of 100,000 indicates that the algorithm exhausted all possible evaluations without converging to 0.

The results for the mean number of function evaluations are compiled in Table 5, where we can clearly see that the proposed adaptive ABC converges more quickly in more than half of the cases. In fact, it outperforms the other six algorithms combined by requiring fewer function evaluations in 11 out of the 20 considered cases, whereas the GABC require less in 5, the IABC in 2 cases, the MABC and ARABC in only 1 case and the DABC and basic ABC have none. One-on-one, the proposed algorithm outperformed the basic ABC and DABC in all of the cases, the MABC in 19 cases, the ARABC in 18, the IABC in 15, and the GABC in 14 or 70% of the cases.

Table 6, on the other hand, contains the successful rate of convergence to 0. We can see a similarity between the results in Tables 5 and 6. If an algorithm requires fewer function evaluations to converge to 0, it will generally have a higher successful rate of convergence. In fact, the proposed adaptive ABC has the best successful rate in 12 of the considered cases, followed by the GABC in 7, the ARABC and IABC in 2, and the rest in only one case.

4.2. The Nonlinear Predictive Control Problem

In the second set of tests, the performance of the proposed algorithm is evaluated and compared considering the NMPC control of the Continuous Stirred Tank Reactor (CSTR) process.

The NMPC is an advanced control strategy used successfully in the industry (Bounkhel & Tadj, 2025). The purpose of the controller is to solve an optimization problem with the aim of reducing the tracking error between the process output and the reference trajectory, in addition to reducing its control effort.

The CSTR is a highly nonlinear chemical process where a continuous exothermic reaction is maintained in order to produce a chemical product. The objective is to control the concentration of the produced chemical product C_a (process output in mol/l) by adjusting how much cooling the tank reactor receives by controlling the flow rate of a coolant q_c (process input in l/min). The initial state of the process is defined by the following values: coolant flow rate $q_c = 97.227 l/min$, concentration of the produced product $C_a = 0.08 mol/l$ and the tank reactor temperature $T = 443.339K$. There is also a hard constraint applied to the input signal by forcing the value of the coolant flow rate q_c to be between $89 l/min$ and $111 l/min$. Additional details regarding this process can be found in Espinosa *et al.* (2005).

The objective of the control is to make the concentration of the produced product C_a follow the reference trajectory defined in Figure 2, while adopting the following design parameters for the NMPC controller: prediction horizon between $N_1 = 1$ and $N_2 = 10$, the control horizon $N_u = 2$, weighting matrices $Q = 1$ and $R = 2 \times 10^{-4}$ with a sampling period of $T_s = 6s$.

Table 3. CEC2021 Test Suite (Mohamed *et al.*, 2020)

Type	No	Id	Function Name	Min
Unimodal Function	1	F_1	Bent Cigar	0
Basic Functions	2	F_2	Schwefel's	0
	3	F_3	Lunacek bi-Rastrigin	0
	4	F_4	Expanded Rosenbrock's plus Griewank's	0
	5	F_5	Hybrid Function 1	0
Hybrid Functions	6	F_6	Hybrid Function 2	0
	7	F_7	Hybrid Function 3	0
Composition Functions	8	F_8	Composition Function 1	0
	9	F_9	Composition Function 2	0
	10	F_{10}	Composition Function 3	0

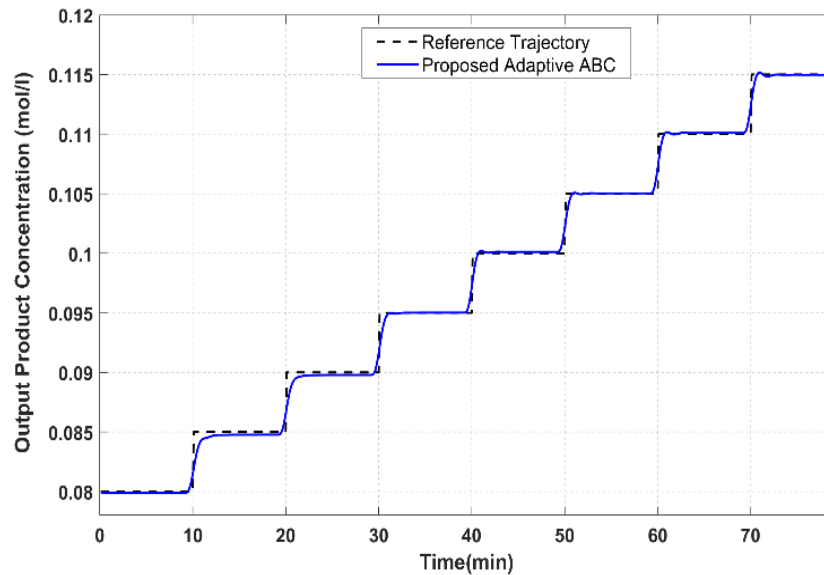


Figure 2. CSTR Output for the Proposed Adaptive ABC-Based Controller

To compare and evaluate the efficiency of the proposed approach against the six considered ABC-based algorithms, seven NMPC-based controllers were constructed for the different optimization algorithms. These controllers are identical, apart from the optimization algorithm they use. For complete details about the steps followed to construct these controllers, please consult (Sahed *et al.*, 2015).

A population of 10 artificial bees with 20 iterations is used by all of the optimization algorithms. Additionally, the best solution found in any given sampling period is always inserted into the population of the next sample. The numerical value for the TC parameter is selected to be 5. The rest of the optimization parameters use the values adopted in the previous section.

The CSTR output for the proposed adaptive ABC-based controller is presented in Figure 2, whereas a more detailed view between 49 and 69 min, containing all of the controller's outputs, is presented in Figure 3. The obtained results show that the proposed algorithm has adequate performance.

For a more in-depth comparison between the different controllers, we will be considering the Mean Cost Value (MCV) criteria computed over the whole simulation and averaged over 100 independently executed runs. The MCV expression is given in Equation (11), where N is the number of samples, $J(\cdot)$ is the cost function associated with the NMPC controller, and $U(t)$ the control sequence. The obtained results are gathered in Table 7, where we can clearly see that the proposed Algorithm has the lowest MCV value, meaning the best performance, followed by the GABC (almost 8% more), MABC (20% more), DABC (30% more), basic ABC (50%), ARABC (69% more), and lastly the IABC-based controller (139% more).

$$MCV = \frac{1}{N} \sum_{t=1}^N J(U(t)). \quad (11)$$

Table 4. Optimization Results for the Different Considered Algorithms

Id	D	DABC		Basic ABC		IABC		GABC		MABC		ARABC		Adaptive ABC	
		Mean	STD	Mean	STD	Mean	STD	Mean	STD	Mean	STD	Mean	STD	Mean	STD
f ₁	10	7.03E-15	1.01E-15	7.39E-15	1.40E-15	1.09E-06	5.18E-06	5.29E-15	1.78E-15	3.52E-15	3.55E-16	4.05E-15	1.24E-15	2.95E-15	1.34E-15
	30	5.61E-14	9.08E-15	6.04E-14	8.80E-15	1.07E-03	4.60E-03	4.00E-14	3.84E-15	1.51E-14	2.43E-15	2.95E-14	3.12E-15	9.17E-15	3.08E-15
	50	7.61E-11	4.61E-11	8.51E-09	5.69E-09	1.12E-01	2.62E-01	8.55E-14	7.34E-15	1.56E-09	5.25E-10	5.93E-14	6.46E-15	2.78E-14	1.05E-14
f ₂	10	3.31E-15	1.46E-14	8.44E-15	2.77E-14	7.98E-07	4.09E-06	3.07E-16	9.60E-17	6.75E-11	4.85E-10	3.79E-16	3.24E-16	4.20E-16	1.56E-15
	30	3.42E-07	1.44E-06	5.01E-07	1.31E-06	7.42E-06	5.66E-05	3.63E-07	2.82E-06	1.72E-02	3.92E-03	1.37E-15	7.72E-16	5.36E-08	1.13E-07
	50	2.76E-04	6.10E-04	4.82E-04	9.02E-04	7.32E-05	3.45E-04	3.92E-05	1.84E-04	1.25E+01	4.05E+00	2.24E-15	8.73E-16	3.28E-05	2.85E-05
f ₃	10	5.72E-17	5.31E-17	5.61E-17	5.85E-17	6.86E-11	6.84E-10	4.80E-17	5.34E-17	5.11E-17	5.35E-17	5.18E-17	5.32E-17	3.73E-17	5.12E-17
	30	1.28E-15	1.22E-15	5.53E-12	5.52E-11	1.05E-04	1.04E-03	5.43E-16	9.90E-17	3.79E-16	1.00E-16	5.20E-16	9.71E-17	6.73E-17	8.87E-17
	50	1.46E-14	5.13E-14	2.93E-07	6.13E-07	3.99E-04	2.73E-03	1.32E-15	2.24E-16	7.26E-16	1.18E-16	9.88E-16	1.55E-16	3.50E-16	2.81E-16
f ₄	10	4.15E-02	2.29E-02	4.27E-02	2.21E-02	5.68E-02	4.08E-02	5.00E-03	1.13E-02	2.65E-01	9.27E-02	2.92E-02	3.08E-02	1.67E-02	2.30E-02
	30	8.85E-01	2.08E-01	9.14E-01	1.95E-01	4.05E-01	8.75E-02	6.37E-01	1.36E-01	1.00E+01	3.09E-01	5.32E-01	1.25E-01	8.20E-01	1.66E-01
	50	2.26E+00	5.02E-01	2.32E+00	4.84E-01	1.02E+00	1.74E-01	1.89E+00	3.60E-01	1.99E+01	3.14E-01	1.43E+00	2.48E-01	2.53E+00	4.70E-01
f ₅	10	1.10E-16	1.11E-17	1.11E-16	0.00E+00	2.32E-13	2.31E-12	1.11E-16	0.00E+00	1.11E-16	0.00E+00	1.11E-16	0.00E+00	1.10E-16	1.11E-17
	30	8.60E-16	1.84E-16	8.08E-16	2.10E-16	1.57E-09	1.55E-08	5.45E-16	4.20E-17	5.30E-16	8.18E-17	5.47E-16	2.85E-17	1.10E-16	1.11E-17
	50	2.01E-15	4.33E-16	2.15E-15	5.88E-16	3.75E-09	2.91E-08	1.25E-15	2.16E-16	8.28E-16	2.01E-16	1.00E-15	1.01E-16	1.78E-16	1.20E-16
f ₆	10	9.12E-04	2.49E-03	1.90E-03	3.53E-03	3.79E-03	5.92E-03	4.24E-04	1.86E-03	4.30E-03	5.02E-03	3.95E-03	5.56E-03	1.48E-04	1.04E-03
	30	9.87E-05	9.87E-04	6.36E-04	2.81E-03	4.66E-03	1.59E-02	3.20E-04	1.84E-03	2.23E-13	2.15E-12	8.39E-04	3.47E-03	1.67E-16	1.64E-16
	50	2.68E-04	2.26E-03	1.27E-04	1.27E-03	1.15E-02	2.82E-02	5.98E-04	2.61E-03	2.81E-09	1.12E-08	7.38E-04	5.08E-03	1.54E-08	1.54E-07
f ₇	10	1.11E-01	2.44E-02	1.28E-01	2.56E-02	1.36E-02	9.27E-03	1.06E-01	2.39E-02	2.13E-01	3.78E-02	1.01E-01	2.50E-02	8.93E-02	1.74E-02
	30	2.19E-01	3.93E-02	2.34E-01	4.33E-02	2.07E-02	6.34E-03	2.10E-01	4.29E-02	4.81E-01	6.23E-02	1.38E-01	2.89E-02	1.99E-01	3.81E-02
	50	2.67E-01	4.55E-02	2.93E-01	4.79E-02	6.56E-02	2.45E-01	2.85E-01	3.51E-02	6.05E-01	6.10E-02	1.64E-01	3.49E-02	2.80E-01	4.11E-02
f ₈	10	1.77E-01	2.15E-02	1.81E-01	2.22E-02	1.87E-01	3.95E-02	1.19E-01	2.26E-02	2.17E-01	3.81E-02	2.80E-01	5.31E-02	1.70E-01	2.73E-02
	30	2.85E-01	3.04E-02	2.92E-01	3.10E-02	2.94E-01	4.22E-02	2.47E-01	2.88E-02	3.75E-01	6.22E-02	3.57E-01	5.23E-02	2.90E-01	2.80E-02
	50	3.46E-01	3.24E-02	3.50E-01	2.65E-02	3.39E-01	3.96E-02	2.94E-01	2.62E-02	4.54E-01	1.12E-01	3.77E-01	4.17E-02	3.39E-01	3.09E-02
f ₉	10	5.32E-17	5.99E-17	6.20E-17	6.18E-17	2.46E-04	2.44E-03	6.53E-17	5.25E-17	5.21E-17	5.35E-17	5.42E-17	5.34E-17	4.68E-17	5.31E-17
	30	7.71E-16	1.61E-16	8.15E-16	2.61E-16	4.56E-01	4.55E+00	5.16E-16	8.29E-17	3.84E-16	1.03E-16	4.80E-16	9.84E-17	8.57E-17	8.82E-17
	50	1.93E-12	6.43E-12	1.27E-05	3.07E-05	2.04E+01	1.02E+02	1.17E-15	2.20E-16	2.45E-14	1.34E-14	8.96E-16	1.36E-16	4.30E-16	3.65E-16
f ₁₀	10	6.46E-17	5.72E-17	6.93E-17	5.90E-17	7.85E-12	4.97E-11	4.26E-17	5.25E-17	4.94E-17	5.28E-17	6.07E-17	5.30E-17	4.26E-17	5.24E-17
	30	7.42E-16	1.18E-16	8.22E-16	1.57E-16	2.51E-06	2.50E-05	5.28E-16	8.16E-17	3.72E-16	9.57E-17	5.33E-16	8.55E-17	5.34E-17	5.95E-17
	50	1.76E-15	3.23E-16	1.91E-15	4.73E-16	5.73E-04	3.89E-03	1.28E-15	1.76E-16	9.18E-15	1.45E-14	1.01E-15	1.48E-16	2.19E-16	1.30E-16
f ₁₁	10	1.37E-02	3.94E-03	1.25E-02	5.38E-03	9.14E-04	4.10E-04	4.76E-03	1.78E-03	3.36E-03	1.29E-03	2.25E-03	1.00E-03	5.83E-04	2.79E-04
	30	2.00E-01	4.92E-02	2.05E-01	4.59E-02	2.00E-02	6.17E-03	8.83E-02	2.34E-02	4.43E-02	1.10E-02	1.84E-02	4.76E-03	3.65E-03	1.36E-03
	50	5.92E-01	9.58E-02	5.93E-01	1.07E-01	8.32E-02	2.15E-02	2.91E-01	5.07E-02	1.75E-01	3.63E-02	5.88E-02	1.10E-02	1.05E-02	3.45E-03
f ₁₂	10	0.00E+00	0.00E+00	0.00E+00	0.00E+00	3.04E-08	1.39E-07	0.00E+00	0.00E+00	4.06E-04	2.92E-03	0.00E+00	0.00E+00	0.00E+00	0.00E+00
	30	3.45E-11	1.69E-10	2.60E-10	2.48E-09	1.80E-03	1.55E-02	0.00E+00	0.00E+00	1.12E+02	7.62E+00	2.51E-01	5.33E-01	0.00E+00	0.00E+00
	50	9.99E-03	9.95E-02	3.01E-02	1.71E-01	2.63E-01	5.16E-01	2.07E-11	1.74E-10	3.03E+02	1.38E+01	1.07E+00	1.10E+00	2.13E-12	1.60E-11
f ₁₃	10	2.17E-01	3.17E-01	4.02E-01	4.83E-01	3.39E+00	2.44E+00	4.93E-04	6.54E-04	3.25E+00	8.69E-01	1.00E+00	1.72E+00	8.90E-01	7.22E-01
	30	5.91E+00	4.94E+00	1.06E+01	6.00E+00	2.49E+01	1.12E+01	1.42E+01	6.27E+00	2.39E+01	8.55E-01	1.88E+01	1.24E+01	1.62E+01	5.09E+00
	50	2.95E+01	1.56E+01	3.75E+01	9.59E+00	5.21E+01	1.87E+01	4.20E+01	1.52E+01	4.45E+01	6.84E-01	5.52E+01	2.25E+01	4.32E+01	1.29E+01
f ₁₄	10	6.43E-17	6.78E-17	5.15E-17	5.28E-17	1.98E-13	1.98E-12	5.49E-17	5.31E-17	4.56E-17	5.28E-17	5.45E-17	5.38E-17	3.83E-17	5.14E-17
	30	7.95E-16	1.48E-16	8.47E-16	1.67E-16	1.70E-03	1.62E-02	5.74E-16	1.07E-16	3.58E-16	9.05E-17	5.36E-16	8.72E-17	5.56E-17	5.76E-17
	50	1.78E-15	3.24E-16	7.64E-15	9.92E-15	4.49E-02	4.42E-01	1.29E-15	1.96E-16	6.83E-16	1.17E-16	9.68E-16	1.70E-16	3.40E-16	2.77E-16
f ₁₅	10	6.70E-17	7.07E-17	4.90E-17	5.35E-17	3.25E-10	1.94E-09	4.23E-17	5.21E-17	4.96E-17	5.30E-17	5.20E-17	5.34E-17	4.65E-17	5.76E-17
	30	7.45E-16	1.26E-16	8.44E-16	1.82E-16	2.62E-06	2.15E-05	5.55E-16	9.88E-17	3.50E-16	9.06E-17	5.17E-16	9.26E-17	6.36E-17	5.74E-17
	50	1.77E-15	3.47E-16	2.21E-15	9.04E-16	5.91E-04	2.98E-03	1.32E-15	1.46E-16	6.57E-16	1.09E-16	9.82E-16	1.34E-16	2.78E-16	2.04E-16
f ₁₆	10	1.15E-14	4.88E-15	1.67E-14	4.76E-15	3.30E-15	9.11E-16	3.55E-15	0.00E+00	1.43E-14	4.67E-15	6.32E-15	5.16E-15	3.55E-15	0.00E+00
	30	1.18E-13	1.73E-14	1.30E-13	1.60E-14	1.17E-02	5.09E-02	0.00E+00	0.00E+00	1.07E+00	5.06E-01	3.52E-14	1.95E-14	1.42E-16	1.42E-15
	50	4.40E-01	2.65E-01	5.62E-01	3.06E-01	5.68E-01	6.74E-01	2.97E-14	2.43E-14	1.32E+01	1.87E+00	7.32E-14	2.71E-14	2.29E-14	2.06E-14
f ₁₇	10	1.05E+00	2.00E+00	6.13E-01	1.53E+00	9.01E+01	9.79E+01	6.62E-02	2.26E-01	7.03E-01	1.49E+00	9.00E-01	3.33E+00	3.20E-02	3.20E-01
	30	1.19E+02	7.08E+01	2.07E+02	6.74E+01	7.68E+04	5.84E+04	8.30E-02	8.30E-01	6.15E+02	1.85E+01	8.75E+01	6.30E+01	3.02E-01	2.15E+00
	50	4.50E+02	2.44E+02	6.92E+02	2.80E+02	1.02E+06	7.93E+05	5.90E-01	4.13E+00	2.11E+03	4.32E+01	3.67E+02	1.64E+02	2.11E+01	7.18E+01
f ₁₈	10	1.56E-01	1.39E-01	1.57E-01	1.21E-01	2.61E+00	2.07E+00	3.95E-01	4.15E-01	1.52E-16	1.28E-16	6.12E+00	3.86E+00	2.74E-04	3.83E-04
	30	2.53E+02	3.29E+01	2.45E+02	3.91E+01	2.05E+02	2.74E+01	2.12E+02	3.73E+07						

Table 5. Mean Number of Function Evaluations

Id	D	Mean Number of Function Evaluations						
		DABC	Basic ABC	IABC	GABC	MABC	ARABC	Adaptive ABC
f_1	10	100,000	100,000	100,000	100,000	99,302	100,000	90,449
f_2	10	100,000	100,000	100,000	100,000	100,000	99,143	97,670
f_3	10	63,994	67,024	64,110	53,893	57,264	54,401	42,404
f_4	10	100,000	100,000	99,184	98,221	100,000	100,000	99,837
f_5	10	99,129	100,000	97,764	100,000	100,000	100,000	99,076
f_6	10	98,239	98,447	94,736	84,596	100,000	92,239	96,444
f_9	10	63,695	69,695	59,870	67,721	59,611	55,531	51,346
f_{10}	10	69,267	71,828	62,063	48,393	54,954	60,550	45,252
f_{12}	10	22,198	24,865	45,551	12,597	58,246	18,789	16,031
f_{14}	10	67,927	70,579	46,436	59,924	52,715	56,276	42,289
f_{15}	10	69,087	64,723	67,936	48,495	54,848	53,374	47,163
f_{16}	10	100,000	100,000	95,738	100,000	100,000	100,000	100,000
f_{17}	10	100,000	100,000	100,000	82,979	92,918	92,993	86,046
f_{18}	10	100,000	100,000	100,000	100,000	89,545	100,000	100,000
F_1	10	64,872	70,882	56,184	54,812	57,214	59,141	49,030
F_2	10	100,000	100,000	95,287	96,160	100,000	93,381	99,518
F_3	10	99,300	97,892	97,946	83,327	100,000	99,347	91,556
F_5	10	98,424	91,386	68,530	46,081	82,741	50,058	44,069
F_8	10	100,000	100,000	100,000	100,000	100,000	100,000	69,979
F_9	10	100,000	100,000	100,000	100,000	100,000	100,000	94,910
Best in		00	00	02	05	01	01	11

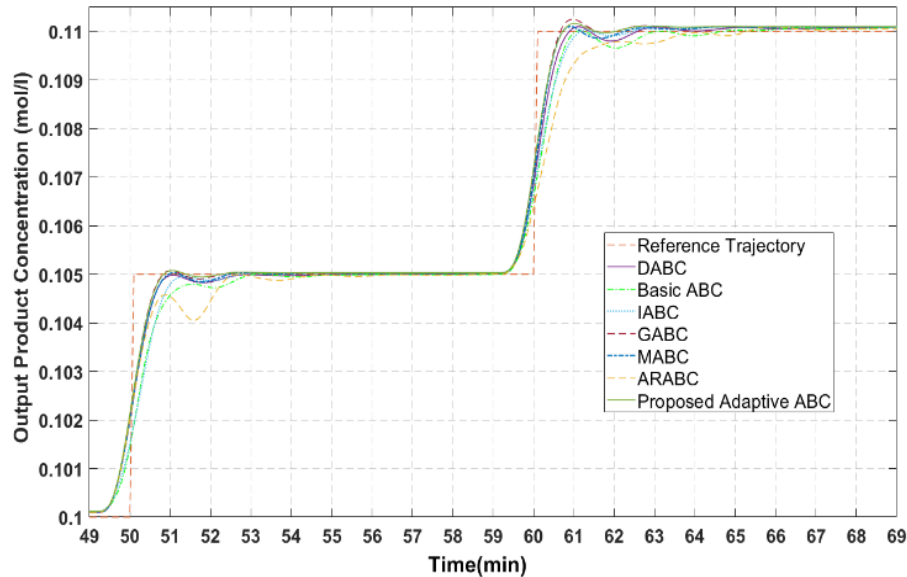


Figure 3. CSTR Outputs for All Controllers for Time Between 49 and 69 Min

Another important factor in evaluating a given optimization algorithm is its computing requirements. To this end, we have included the average time per sampling period required by each controller to compute the control sequence, averaged over 100 independently executed runs. The results, which are compiled in Table 7, indicate that the proposed algorithm requires the least computing time with only 2.67 *ms*, followed by the GABC, the IABC, the MABC, and the basic ABC, which are all around 2.76 *ms*, while the DABC and ARABC require the most time with 2.88 *ms* and 2.99 *ms*, respectively.

Table 6. Successful Rates of Convergence to 0

Id	D	Successful Rate of Convergence to 0						
		DABC	Basic ABC	IABC	GABC	MABC	ARABC	Adaptive ABC
f_1	10	0%	0%	0%	0%	1%	0%	17%
f_2	10	0%	0%	0%	0%	0%	1%	7%
f_3	10	46%	49%	39%	55%	52%	51%	65%
f_4	10	0%	0%	3%	9%	0%	0%	2%
f_5	10	1%	0%	3%	0%	0%	0%	1%
f_6	10	6%	3%	13%	28%	0%	17%	7%
f_9	10	52%	45%	44%	39%	51%	49%	56%
f_{10}	10	41%	38%	41%	60%	53%	43%	60%
f_{12}	10	100%	100%	64%	100%	88%	100%	100%
f_{14}	10	45%	51%	59%	48%	57%	49%	64%
f_{15}	10	45%	54%	35%	60%	53%	51%	58%
f_{16}	10	0%	0%	7%	0%	0%	0%	0%
f_{17}	10	0%	0%	0%	36%	28%	17%	40%
f_{18}	10	0%	0%	0%	0%	29%	0%	0%
F_1	10	54%	49%	49%	59%	55%	45%	59%
F_2	10	0%	0%	17%	17%	0%	20%	3%
F_3	10	1%	3%	3%	35%	0%	5%	17%
F_5	10	3%	18%	46%	80%	34%	72%	84%
F_8	10	0%	0%	0%	0%	0%	0%	75%
F_9	10	0%	0%	0%	0%	0%	0%	9%
Best in		01	01	02	07	01	02	12

Table 7. MCV Values & the Execution Times for the Different Considered NMPC Controllers

Algorithm	MCV	Execution Time (ms)
DABC	4.40E-06	2.88
Basic ABC	5.09E-06	2.77
IABC	8.08E-06	2.76
GABC	3.65E-06	2.75
MABC	4.08E-06	2.76
ARABC	5.73E-06	2.99
Proposed Adaptive ABC	3.38E-06	2.67

5. CONCLUSION

In this work, an adaptive optimization strategy has been introduced and applied to the artificial bee colony algorithm by dynamically switching between exploration and exploitation modes according to the successful mutation rate. The proposed approach has demonstrated good performance across a wide range of benchmark functions and a nonlinear predictive control problem. The results have confirmed that the adaptive mechanism can improve convergence behavior and enhance solution quality compared to several established ABC variants.

While the results demonstrate the promise of the proposed adaptive strategy within the ABC framework, more research is required to evaluate its effectiveness against other metaheuristic algorithms like particle swarm optimization, Differential Evolution and the Genetic Algorithm. What is more, further validation on additional real-world problems (constrained, noisy, and larger-scale industrial tasks) is also needed to assess practical effectiveness in diverse deployment environments. A thorough sensitivity analysis of the tuning cycle parameter and its impact on the optimizer's overall performance is also warranted.

REFERENCES

- Akay, B., & Karaboga, D. (2012). A modified Artificial Bee Colony algorithm for real-parameter optimization. *Information Sciences*, 192, 120-142. DOI: <https://doi.org/10.1016/j.ins.2010.07.015>
- Badgwell, T. A., & Qin, S. J. (2013). Model-Predictive Control in Practice. In J. Baillieul & T. Samad (Eds.), *Encyclopedia of Systems and Control* (pp. 1-6). Springer London. DOI: https://doi.org/10.1007/978-1-4471-5102-9_8-1
- Bounkhel, M., & Tadj, L. (2025). Optimizing Faculty Hiring in Higher Education Using Model Predictive Control. *International Journal of Industrial Engineering: Theory, Applications and Practice*, 32(2). DOI: <https://doi.org/10.23055/ijietap.2025.32.2.10099>
- Brajevic, I. (2015). Crossover-based artificial bee colony algorithm for constrained optimization problems. *Neural Computing and Applications*, 26(7), 1587-1601. DOI: <https://doi.org/10.1007/s00521-015-1826-y>
- Brajević, I. (2021). A Shuffle-Based Artificial Bee Colony Algorithm for Solving Integer Programming and Minimax Problems. *Mathematics*, 9(11), 1211. DOI: <https://doi.org/10.3390/math9111211>
- Comert, S. E., & Yazgan, H. R. (2023). A new approach based on hybrid ant colony optimization-artificial bee colony algorithm for multi-objective electric vehicle routing problems. *Engineering Applications of Artificial Intelligence*, 123, 106375. DOI: <https://doi.org/10.1016/j.engappai.2023.106375>
- Črepinšek, M., Liu, S.-H., & Mernik, M. (2013). Exploration and exploitation in evolutionary algorithms: A survey. *ACM Comput. Surv.*, 45(3), Article 35. DOI: <https://doi.org/10.1145/2480741.2480752>
- Cui, L., Li, G., Wang, X., Lin, Q., Chen, J., Lu, N., & Lu, J. (2017). A ranking-based adaptive artificial bee colony algorithm for global numerical optimization. *Information Sciences*, 417, 169-185. DOI: <https://doi.org/10.1016/j.ins.2017.07.011>
- Cui, Y., Hu, W., & Rahmani, A. (2024). Multi-robot path planning using learning-based Artificial Bee Colony algorithm. *Engineering Applications of Artificial Intelligence*, 129, 107579. DOI: <https://doi.org/10.1016/j.engappai.2023.107579>
- Das, S., Biswas, S., & Kundu, S. (2013). Synergizing fitness learning with proximity-based food source selection in artificial bee colony algorithm for numerical optimization. *Applied Soft Computing*, 13(12), 4676-4694. DOI: <https://doi.org/10.1016/j.asoc.2013.07.009>
- Duan, H., Xing, Z., & Xu, C. (2009). An Improved Quantum Evolutionary Algorithm Based on Artificial Bee Colony Optimization. *Advances in Computational Intelligence*, Berlin, Heidelberg.
- Espinosa, J., Vandewalle, J., & Wertz, V. (2005). *Fuzzy logic, identification and predictive control*. Springer-Verlag London Limited.
- Gao, W., & Liu, S. (2012). A modified artificial bee colony algorithm. *Computers & Operations Research*, 39, 687-697. DOI: <https://doi.org/10.1016/j.cor.2011.06.007>
- Gao, W., Sheng, H., Wang, J., & Wang, S. (2019). Artificial Bee Colony Algorithm Based on Novel Mechanism for Fuzzy Portfolio Selection. *IEEE Transactions on Fuzzy Systems*, 27(5), 966-978. DOI: <https://doi.org/10.1109/TFUZZ.2018.2856120>
- Hussain, K., Mohd Salleh, M. N., Cheng, S., Shi, Y., & Naseem, R. (2020). Artificial bee colony algorithm: A component-wise analysis using diversity measurement. *Journal of King Saud University - Computer and Information Sciences*, 32(7), 794-808. DOI: <https://doi.org/10.1016/j.jksuci.2018.09.017>
- Ivona, B., Predrag, S. S., Shuai, L., & Xinwei, C. (2020). A Hybrid Firefly and Multi-Strategy Artificial Bee Colony Algorithm. *International Journal of Computational Intelligence Systems*, 13(1), 810-821. DOI: <https://doi.org/10.2991/ijcis.d.200612.001>

- Kang, F., Li, J., & Xu, Q. (2009). Structural inverse analysis by hybrid simplex artificial bee colony algorithms. *Computers & Structures*, 87(13), 861-870. DOI: <https://doi.org/10.1016/j.compstruc.2009.03.001>
- Karaboga, D. (2005). *An Idea Based On Honey Bee Swarm for Numerical Optimization* (Technical Report TR06, Issue. Karaboga, D., & Basturk, B. (2007). A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. *Journal of Global Optimization*, 39(3), 459-471. DOI: <https://doi.org/10.1007/s10898-007-9149-x>
- Karafotias, G., Hoogendoorn, M., & Eiben, A. E. (2015). Parameter Control in Evolutionary Algorithms: Trends and Challenges. *IEEE Transactions on Evolutionary Computation*, 19(2), 167-187. DOI: <https://doi.org/10.1109/TEVC.2014.2308294>
- Kıran, M. S., & Findik, O. (2015). A directed artificial bee colony algorithm. *Applied Soft Computing*, 26, 454-462. DOI: <https://doi.org/10.1016/j.asoc.2014.10.020>
- Kong, D., Chang, T., Dai, W., Wang, Q., & Sun, H. (2018). An improved artificial bee colony algorithm based on elite group guidance and combined breadth-depth search strategy. *Information Sciences*, 442-443, 54-71. DOI: <https://doi.org/10.1016/j.ins.2018.02.025>
- Lin, Q., Zhu, M., Li, G., Wang, W., Cui, L., Chen, J., & Lu, J. (2018). A novel artificial bee colony algorithm with local and global information interaction. *Applied Soft Computing*, 62, 702-735. DOI: <https://doi.org/10.1016/j.asoc.2017.11.012>
- Loubière, P., Jourdan, A., Siarry, P., & Chelouah, R. (2016). A sensitivity analysis method for driving the Artificial Bee Colony algorithm's search process. *Applied Soft Computing*, 41, 515-531. DOI: <https://doi.org/10.1016/j.asoc.2015.12.044>
- Mallala, B., Papana, V. P., Sangu, R., Palle, K., & Chinthalachervu, V. K. R. (2022). Multi-Objective Optimal Power Flow Solution Using a Non-Dominated Sorting Hybrid Fruit Fly-Based Artificial Bee Colony. *Energies*, 15(11), 4063. DOI: <https://doi.org/10.3390/en15114063>
- Mohamed, A. W., Hadi, A. A., Mohamed, A. K., Agrawal, P., Kumar, A., & Suganthan, P. N. (2020). *Problem Definitions and Evaluation Criteria for the CEC 2021 Special Session and Competition on Single Objective Bound Constrained Numerical Optimization*.
- Sahed, O. A., Kara, K., & Benyoucef, A. (2015). Artificial bee colony-based predictive control for non-linear systems. *Transactions of the Institute of Measurement and Control*, 37(6), 780-792. DOI: <https://doi.org/10.1177/0142331214546796>
- Sedighzadeh, D., & Mazaheripour, H. (2018). Optimization of multi-objective vehicle routing problem using a new hybrid algorithm based on particle swarm optimization and artificial bee colony algorithm considering Precedence constraints. *Alexandria Engineering Journal*, 57(4), 2225-2239. DOI: <https://doi.org/10.1016/j.aej.2017.09.006>
- Shen, L., Zhu, H., & Liu, S. (2025). An Adaptive Multi-Swarm Gray Wolf Optimizer for Flexible Job Shop Scheduling Problem with Assembly/Disassembly and Lot Streaming. *International Journal of Industrial Engineering: Theory, Applications and Practice*, 32(6). DOI: <https://doi.org/10.23055/ijietap.2025.32.6.11097>
- Sundar, S., & Singh, A. (2012). A hybrid heuristic for the set covering problem. *Operational Research*, 12(3), 345-365. DOI: <https://doi.org/10.1007/s12351-010-0086-y>
- Wang, H., Wu, Z., Rahnamayan, S., Sun, H., Liu, Y., & Pan, J.-s. (2014). Multi-strategy ensemble artificial bee colony algorithm. *Information Sciences*, 279, 587-603. DOI: <https://doi.org/10.1016/j.ins.2014.04.013>
- Wolpert, D. H., & Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1), 67-82. DOI: <https://doi.org/10.1109/4235.585893>
- Xiao, S., Wang, W., Wang, H., Tan, D., Wang, Y., Yu, X., & Wu, R. (2019). An Improved Artificial Bee Colony Algorithm Based on Elite Strategy and Dimension Learning. *Mathematics*, 7(3), 289. DOI: <https://doi.org/10.3390/math7030289>

Zhang, Z., Fu, Y., Gao, K., Pan, Q., & Huang, M. (2024). A learning-driven multi-objective cooperative artificial bee colony algorithm for distributed flexible job shop scheduling problems with preventive maintenance and transportation operations. *Computers & Industrial Engineering*, 196, 110484. DOI: <https://doi.org/10.1016/j.cie.2024.110484>

Zhao, M., Song, X., & Xing, S. (2022). Improved Artificial Bee Colony Algorithm with Adaptive Parameter for Numerical Optimization. *Applied Artificial Intelligence*, 36(1), 2008147. DOI: <https://doi.org/10.1080/08839514.2021.2008147>

Zhu, F., Shuai, Z., Lu, Y., Su, H., Yu, R., Li, X., Zhao, Q., & Shuai, J. (2024). oBABC: A one-dimensional binary artificial bee colony algorithm for binary optimization. *Swarm and Evolutionary Computation*, 87, 101567. DOI: <https://doi.org/10.1016/j.swevo.2024.101567>

Zhu, G., & Kwong, S. (2010). Gbest-guided artificial bee colony algorithm for numerical function optimization. *Applied Mathematics and Computation*, 217(7), 3166-3173. DOI: <https://doi.org/10.1016/j.amc.2010.08.049>