Article

JMA: Nature-Inspired Java Macaque Algorithm for Optimization Problem

Dinesh Karunanidhy ¹ , Subramanian Ramalingam ², Ankur Dumka ³, Rajesh Singh ⁴ , Mamoon Rashid ^{5,*} , Anita Gehlot ⁴, Sultan S. Alshamrani ⁶ and Ahmed Saeed AlGhamdi ⁷

- ¹ Department of Computer Science & Technology, Madanapalle Institute of Technology and Science, Madanapalle 517325, India; drdineshk@mits.ac.in
- ² Department of Computer Science & Engineering, Pondicherry University, Puducherry 605014, India; drrajakumarr@mits.ac.in
- ³ Department of Computer Science and Engineering, Women's Institute of Technology, Dehradun 248007, India; ankurdumka2@gmail.com
- ⁴ Department of Research and Development, Uttaranchal Institute of Technology, Uttaranchal University, Dehradun 248007, India; drrajeshsingh004@gmail.com (R.S.); dranitagehlot@gmail.com (A.G.)
- ⁵ Department of Computer Engineering, Faculty of Science and Technology, Vishwakarma University, Pune 411048, India
- ⁶ Department of Information Technology, College of Computer and Information Technology, Taif University, P.O. Box 11099, Taif 21944, Saudi Arabia; susamash@tu.edu.sa
- ⁷ Department of Computer Engineering, College of Computer and Information Technology, Taif University, P.O. Box 11099, Taif 21994, Saudi Arabia; asjannah@tu.edu.sa
- * Correspondence: mamoon.rashid@vupune.ac.in; Tel.: +91-7814346505

Citation: Karunanidhy, D.; Ramalingam, S.; Dumka, A.; Singh, R.; Rashid, M.; Gehlot, A.; Alshamrani, S.S.; AlGhamdi, A.S. JMA: Nature-Inspired Java Macaque Algorithm for Optimization Problem. *Mathematics* **2022**, *10*, 688. <https://doi.org/10.3390/math10050688>

Academic Editors: Alexander A Lazarev, Frank Werner and Bertrand M.T. Lin

Received: 25 December 2021

Accepted: 12 February 2022

Published: 23 February 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

Abstract: In recent years, optimization problems have been intriguing in the field of computation and engineering due to various conflicting objectives. The complexity of the optimization problem also dramatically increases with respect to a complex search space. Nature-Inspired Optimization Algorithms (NIOAs) are becoming dominant algorithms because of their flexibility and simplicity in solving the different kinds of optimization problems. Hence, the NIOAs may be struck with local optima due to an imbalance in selection strategy, and which is difficult when stabilizing exploration and exploitation in the search space. To tackle this problem, we propose a novel Java macaque algorithm that mimics the natural behavior of the Java macaque monkeys. The Java macaque algorithm uses a promising social hierarchy-based selection process and also achieves well-balanced exploration and exploitation by using multiple search agents with a multi-group population, male replacement, and learning processes. Then, the proposed algorithm extensively experimented with the benchmark function, including unimodal, multimodal, and fixed-dimension multimodal functions for the continuous optimization problem, and the Travelling Salesman Problem (TSP) was utilized for the discrete optimization problem. The experimental outcome depicts the efficiency of the proposed Java macaque algorithm over the existing dominant optimization algorithms.

Keywords: continuous optimization problem; discrete optimization problem; grey wolf optimizer; Java macaque algorithm; nature-inspired algorithm

MSC: 68U01

1. Introduction

Nature-Inspired Optimization Algorithms (NIOAs) are one of the dominant techniques used due to their simplicity and flexibility in solving large-scale optimization problems [1]. The search process of the nature-inspired optimization algorithms was developed based on the behavior or processes encountered from nature. Notably, the NIOAs have emphasised the significant collection of algorithms, like the evolutionary algorithm (EA), swarm intelligence (SI), physical algorithm, and bio-inspired algorithm. These algorithms have

illustrated their efficiency in solving wide variants of real-world problems [2]. A variety of nature-inspired optimization algorithms, such as the Genetic Algorithm (GA) [3], Differential Evolution (DE) [4], Ant Colony Optimization (ACO) [5], Artificial Bee Colony (ABC) [6], Particle Swarm Optimization (PSO) [7], Firefly Algorithm (FA) [8], Cuckoo Search (CO) [9], Bat Algorithm [10], Monkey Algorithm (MA) [11], Spider Monkey Algorithm (SMO) [12], Reptile Search Algorithm (RSA) [13], Membrane Computing (MC) [2], and whale optimization [14] use simple local searches for convoluted learning procedures to tackle complex real-world problems.

The complexity of real-world problems increases with the current scenario. Hence, the nature-inspired optimization algorithm has to find the best feasible solution with respect to the decision and objective space of the optimization problem. The increase in a decision variable may directly increase the size of the problem space. Then the complexity of the search space is also exponentially increased due to an increase in decision variables. Similarly, the search space will also increase due to an increase in the number of objectives [1]. Thus, the performance of the optimization algorithms will depend on two major components, as follows [15]: (i) the exploration used to generate the candidate solution to explore the search space globally; and (ii) the exploitation used to focus on exploiting the search space in the local region to find the optimal solution in the particular region. Thus, it is of essential importance for the optimization algorithm to acquire an equilibrium between the exploitation and exploration for solving different kinds of the optimization problem [16].

2. Literature Survey

Many optimization algorithms in the literature have been modified from the initial version to a hybrid model in order to tackle the balance between exploration and exploitation [17,18]. For example, we consider the most famous nature-inspired optimizations, such as GA, PSO, DE, ABC, and the most recent algorithms, such as the grey wolf-optimizer, reptile search algorithm, Spider Monkey Algorithm, and whale optimization algorithm. The main operators of the genetic algorithm are selection, crossover, and mutation [19]. Many authors [20] in the literature have proposed novel crossover operators in order to adjust the exploration capability of the genetic algorithm. Then, the selection process [21] and mutation operators [22,23] have been modified by authors in order to maintain a diverse and converged population. The author in the Ref. [24] proposed a special chromosome design based on a mixed-graph model to address the complex scheduling problem, and this technique enhances the heuristic behaviour of a genetic algorithm. Further, the genetic algorithm is combined with other techniques in order to improve its performance, such as a self-organization map [25,26], adaptive techniques [27], and other optimization algorithms [28]. These operators help to attain a balance over convergence and randomness to find the optimal results [29]. However, the convergence depends on the mutation operator because of its dual nature, that is, it either slows down the convergence or attains the global optima. Thus, fine-tuning the operators concerning the optimization problem is quite difficult [30].

The next feasible approach for solving the optimization problem is differential evolution. It also uses vector-based operators in the search process. However, the mutation operator shows its dominance over the search process by contributing to the weighted divergence between two individuals. The selection procedures of the DE have a crucial impact on the search process and also influences diversity among the parents and offspring [31]. However, the author in the Ref. [32] developed a new technique based on self-adaptive differential evolution with weighted strategies to address these large-scale problems. In the Particle Swarm Optimizer (PSO) [7], the selection process of a global best seems to play a vital role in finding the global optimal solution—that is, it can accelerate the convergence, or it can lead to premature convergence. ACO is based on the pheromone trails, such as trail-leaving and trail-after practices of ants, in which every ant sees synthetic pheromone fixations on the earth, and acts by probabilistic selecting bearings focused around the

accessible pheromone fixation. However, the search process of the ACO is well-suited for exploration, whereas the exploitation has considerably lower importance [33].

The Artificial Immune System (AIS) [34] is mimicked by the biological behavior of the immune system. The AIS uses the unsupervised learning technique obtained from the immune cell over the infected cells. Hence, the algorithm suits cluster-oriented optimization problems well, and strong exploitation and learning processes are intensified at the poor level [35]. The bacterial foraging algorithm (BFA) was developed by Passino [36], which works on the basic principle of the natural selection of bacteria. It is also a non-gradient optimization algorithm that mimicks the foraging behavior of the bacteria from the landscape towards the available nutrients [37]. However, the algorithm has different kinds of operators, where it depicts less convergence over complex optimization problems [38]. The Krill Herd (KH) is a bio-inspired optimization algorithm which imitates the behavior of krill [39]. The basic underlying concept of a krill herd is to reduce the distance between the food and each individual krill in the population. However, the author of the Ref. [40] stated that the KH algorithm does not maintain a steady process between exploration and exploitation.

The dominant author in the field of NIA, Yang [41], proposed the Firefly Algorithm (FA). The lighting behavior of fireflies during the food acquirement process is the primary ideology of the firefly algorithm [42]. Thus, the search process of the firefly algorithm is well-suited for solving multi-model optimization using a multi-population [8]. However, the cuckoo search well-utilises the random walk process that explores the search in an efficient manner, which may lead to slow convergence [43]. Further, the bat algorithm finds difficulty in the optimal adjustment of the frequency for an echo process, which helps to attain the optimal solution in the search process [9].

The various types of algorithms evolved in the literature based on the behavior of monkeys, such as the monkey algorithm (MA), monkey king evolution (MKO), and spider monkey algorithm (SMO). The initial solution was brought into existence via a random process, and it completes the local search process using the limb process and also performs a somersault process for the global search operation. Further, an improved version of the monkey algorithm was proposed in the Ref. [11], which includes the evolutionary speed factor to dynamically change the climb process and aggregation degree to dynamically assist somersault distance. The Spider Monkey Algorithm (SMO) was developed based on the swarm behavior of monkeys, and performs well with regard to the local search problem. The SMO also incorporates a fitness-based position update to enhance the convergence rate [44]. The Ageist Spider Monkey Algorithm (ASMO) includes features of agility and swiftness based on their age groups, which work based on the age differences present in the spider monkey population [12]. However, the algorithm performance decreases during the global search operation.

An empirical study from the literature [45] clearly shows the importance of a diverse population to attain the best solutions worldwide. Hence, a recent advancement in the field of an optimization algorithm incorporated multi-population methods where the population was subdivided into many sub-populations to avert the local optima and maintain diversity among the population [46]. Thus, multi-population methods guide the search process either in exploiting or exploring the search space. The significance of a multi-population has been incorporated in several dominant optimization algorithms [8,47].

Though multi-population-based algorithms have attained viable advantages in exploring and exploiting the search space, it has some consequences in designing the algorithm, such as the number of sub-populations, and communication and strategy among the sub-population [48]. Firstly, the number of sub-populations helps to spread the individuals over the search space. In fact, the small number of sub-populations leads to local optima, whereas the increase in the number of sub-populations leads to a wastage of computation resources and also extends the convergence [49]. The next important issue is the communication handling procedure between sub-populations, such as communication rates and policy [50]. The communication rate determines the number of individuals in a

sub-population who have to interact with other sub-populations, and a communication policy is subjected to the replacement of individuals with other sub-populations.

The above discussion clearly illustrates the importance of balanced exploration and exploitation in the search process. Most of the optimization algorithms have problems in tuning the search operators and with diversity among the individuals. On the other hand, the multi-population-based algorithm amply shows its efficiency in maintaining a diverse population. However, it also requires some attention towards communication strategies between the sub-population. Hence, this motivated our research to develop a novel optimization algorithm with well-balanced exploration and exploitation. In particular, the Java macaque is also the vital primitive among the monkeys family and is widely available in South-Asian countries. Thus, the Java macaques have suitable behavior for balancing the exploration and exploitation search process.

The novel contribution of the proposed Java macaque algorithm is described as follows:

- (1) We introduced the novel optimization algorithm based on the behavior of Java macaque monkeys for balancing the search operation. The balance is achieved by modeling the behavioral nature of Java macaque monkeys, such as multi-group behavior, multiple search agents, a social hierarchy-based selection strategy, mating, male replacement, and learning process.
- (2) The multi-group population with multiple search agents as male and female monkeys helps to explore the different search spaces and also maintain diversity.
- (3) This algorithm utilises the dominance hierarchy-based mating process to explore the complex search spaces.
- (4) In order to address the communication issues in a multi-group population, the Java monkeys have unique behavior, called male replacement.
- (5) The exploitation phase of the proposed algorithm is achieved by the learning process.
- (6) This algorithm utilises a multi-leader approach using male and female search agents, which consists of an alpha male and alpha female in each group, and also remains as the best solution globally. Thus, the multi-leader approach assists in a smooth transition from the exploration phase to exploitation phase [50]. Further, the social hierarchy-based selecting strategy helps in maintaining both an improved converged and diverse solution in each group, as well as in the population.

Further in this paper, Section 3 describes the brief behavior analysis of the Java macaque monkey. Section 4 formulates the algorithmic compounds using the Java macaque behavior model. Then, Section 3.1 explores the stability of Java macaque behavior, and Section 4 introduces the generic version of the proposed Java macaque algorithm for the optimization problem. Further, Section 5 provides a brief experimentation of the continuous optimization problem, and Section 6 illustrates the detailed experimentation of discrete optimization. Finally, the paper is concluded in Section 8.

3. Behavior Analysis of the Java Macaque Monkey

The Java macaque monkey is an essential native breed that lives with a social structure, and it has almost 95% of gene similarity when compared with human beings [51]. The basic search agent of the java monkey is divided into male and female, where males are identified with a moustache and females with a beard. The average number of male and female individuals is 5.7 and 9.9 in each group, respectively. Usually, the Java macaque lives (with a group community of 20–40 monkeys) in an environment where macaques try to dominate over others in their region of living. Age, size and fighting skills are the factors used to determine social hierarchy among macaques [52]. The structure of social hierarchy has been followed among the java monkey groups where the lower-ranking individuals are dominated by the higher-ranking individuals when accessing food and resources [51,53]. The higher-ranking adult male and female macaques of a group are called the “Alpha Male” and “Alpha Female”. Further, studies have shown that the alpha male has dominant access to females, which probably sires the offspring.

The number of female individuals is higher in comparison with the male individual for every group. The next essential stage of the java monkey culture is mating. The mating operator depends on the hierarchical structure of the macaques, that is, the selection of male and female macaques is based on the social rank [54]. The male macaque attracts the female by creating a special noise and gesture for the reproduction. The higher-ranking male individuals are typically attracted to the female individual due to their social power and fitness. The male individual reaches sexual maturity approximately at the age of 6 years, while the female matures at the age of 4 years [55]. The newborn offspring or juvenile's social status depends on the parent's social rank and matrilineal hierarchy, which traces their descent through the female line. In detail, the offspring of the dominant macaque have a higher level of security when compared with offspring of lower-ranking individuals [56].

The male offspring are forced to leave their natal group after reaching sexual maturity and become stray males, whereas the female can populate in the same group based on their matrilineal status [53]. Hence, stray males have to join another social group, otherwise they are subjected to risk in the form of predators, disease, and injury. Male replacement is a process in which stray males can reside in another group in two ways—they either have to dominate the existing alpha male or sexually attract a female, and that way the macaque can convince another member to let the stray male into the group. Hence, the behavior of the Java macaque shows excellent skill in solving real-world problems in order to protect their group from challenging circumstances. Thus, the stray male can continuously learn from the dominant behavior of other macaques. Thus, the improvement in the ability of the Java macaque enhances its social ranking and provides higher access to food and protection. Further, the monkey can improve its ranking via a learning process and can also become the alpha monkey, which leads to the attainment of higher power within the group.

The behavior of Java macaque monkeys illustrates the characteristics of an optimization algorithm, such as selection, mating, maintaining elitism, and male replacement, finding the best position via learning from nature. The Java macaque exhibits population-based behavior with multiple groups in it and also adapts a dominant hierarchy. Then, each group can be divided into male and female search agents, which can be further divided into an adult, sub-adult, juvenile, and infant based on age. Next, an important characteristic of a java monkey is mating. In particular, the selection procedure for mating predominantly depends on the fitness hierarchy of the individuals. Further, the dominant individuals in a male and female population are maintained in each group of the population, known as the alpha male and female. The ageing factor is also considered to be an important deciding factor. The male monkey which attains sexual maturity at the age of 4 is forced to leave its natal group, and becomes a stray male. Thus, the stray male has to find another suitable group based on fitness ranking, and this process is known as male replacement. In addition to the above behavior, the monkey efficiently utilises the learning behavior from dominant individuals in order to attain the dominant ranking. The behavior of java monkeys does well to attain global optimal solutions in real-world problems. According to the observations from the above discussion, the behavior of java monkeys exhibits the following features: (1) the group behavior of individuals with different search agents helps in exploring a complex search space; (2) exploration and exploitation is performed using selection, mating, male replacement, and learning behavior; and (3) dominant monkeys are maintained within the group using an alpha male and female.

3.1. State Space Model for Java Macaque

Over the centuries, Java macaque monkeys have survived in this world because of their well-balanced behavior. This amply shows that the behavior of Java macaque monkeys has stability within the population. Then, the state space model for the Java macaque monkey in Figure 1 was designed using the demographic data shown in Table 1, obtained from the Ref. [57]. The demographic composition of the monkey population over the period of 1978 to 1982 was used to develop the state space model of the java monkey.

Table 1. Demographic composition of the monkey population (1978–1982).

Class	Age Years	1978	1979	1980	1981	1982
Adult Male	≥ 7	19	25	30(1)	39(3)	63(2)
Adult Female	≥ 5	51(1)	57(1)	68(1)	76(1)	91(1)
Subadult Male	4–6	25	45	48(1)	54	43
Subadult Female	4	7	12	9	16	25
Juvenile Male	1–3	55	43(2)	47(1)	54	75
Juvenile Female	1–4	39	50(2)	71(2)	82(1)	82(2)
Infant Male	< 1	16(3)	15(3)	25(3)	36(2)	37(5)
Infant Unsexed	< 2	25(1)	32(2)	28(1)	27(1)	38(5)
Infant Female	< 3	0	0(2)	0(1)	0	0
Total		237	279	326	384	454
Infant Mortality 2		8.89	12.96	7.02	4.54	11.76
Noninfant Mortality 1		0.51	2.11	2.15	1.23	1.31

Figure 1. State space model of the Java macaque monkey.

The generalized equation for female individuals in the state space models are represented as:

$$AF^* = -\alpha AF + \gamma_1 SF \quad (1)$$

$$SF^* = \delta_1 JF - \gamma_1 SF - \alpha SF \quad (2)$$

$$JF^* = \varepsilon_1 IF - \delta_1 JF - \alpha JF \quad (3)$$

$$IF^* = \eta_1 AM + \mu_1 AF - \varepsilon_1 IF - \beta IF \quad (4)$$

$$\eta_1 = \frac{2}{3}\mu_1, \quad (5)$$

where β, α are the death rate of the Infant (Offspring) and Non-Infant. Input variables $u(t)$ are AF, SF, JF, IF , and the output variables $y(t)$ are AF^*, SF^*, JF^*, IF^* with respect to the state variable $\alpha, \beta, \gamma, \delta, \varepsilon, \mu, \eta$. Additionally, AF, SF, JF and IF indicates the adult, sub-adult, juvenile and infant females.

Similarly, the generalized equation for male individuals in the population are presented as:

$$AM^* = -\alpha AM + \gamma_2 SM \quad (6)$$

$$SM^* = \delta_2 JM - \gamma_2 SM - \alpha SM \quad (7)$$

$$JM^* = \varepsilon_2 IM - \delta_2 JM - \alpha JM \quad (8)$$

$$IM^* = \eta_2 AM + \mu_2 AF - \varepsilon_2 IM - \beta IM \quad (9)$$

$$\eta_2 = \frac{2}{3}\mu_2, \quad (10)$$

where β, α are the death rate of the Infant (Offspring) and Non-Infant. Input variables $u(t)$ are AM, SM, JM, IM and the output variable $y(t)$ are AM^*, SM^*, JM^*, IM^* with respect to state variable $\alpha_2, \beta_2, \gamma_2, \delta_2, \varepsilon_2, \mu_2, \eta_2$. Additionally, AM, SM, JM and IM indicate the adult, sub-adult, or stray, juvenile, and infant males.

By using the state space model, we generated the transition matrix, which helps in proving the stability of the population using the α -diagonally dominant method, which has been clearly explained in the supplementary documents.

4. Java Macaque Monkey Algorithm Modeling

The Java Macaque Algorithm (JMA) is a meta-heuristic algorithm based on the genetic and social behavior of the java monkey. Initially, it starts with a population of the random solution and explores the search space to find the optimal. The life cycle of the Java macaque consists of selecting, mating, male replacement, and learning. Hence, the intelligent behavior of the java monkey does well to solve large-scale optimization problems. Moreover, it also demonstrates the behavior of Java macaques, and exhibits their adaptive learning, genetic, and social behavior and how they often respond according to changes in their environment, which maintains the global order emerging from the interaction of the java monkey. Thus, the life cycle of the java monkey could be transformed as the algorithm model for solving the real-world optimization problem.

4.1. JMA—Preamble

In the primitive of the Java macaque monkey, the search agents are divided into two types, known as the 'Male' and 'Female' Java macaque. The nature of male and female search agents is to produce different cooperative behavior depending on its gender. Then, the Java macaque exhibits a multi-group population that is utilized to achieve the best performance.

The population of a Java macaque leaves in g -groups and N -individuals in the group.

$$\begin{aligned} POP &= \{G_i\}_{i=1,2,\dots,g} \\ &= \{M_j\}_{j=1,2,\dots,M_{size}} \cup \{F_k\}_{k=1,2,\dots,F_{size}} \\ &\quad s.t. \quad (G_i \supset M, F : M \notin F), \end{aligned} \quad (11)$$

where G_i presents the group i with M_{size} of male individuals and F_{size} of female individuals in the population. The initialization process starts with a minimum number of individuals in the group with respect to the problem size. Then the casual system is utilized to control the size of the population in every group. $|G_i|$, with a population size which consists of a number of male individuals M_{size} and female individuals F_{size} , is defined as:

$$M_{size} = \text{floor}[0.9 - \text{rand}(0.25, 0.4) * |G_i|] \quad (12)$$

$$F_{size} = \text{floor}[0.9 - \text{rand}(0.45, 0.6) * |G_i|] \quad (13)$$

$$M_{size} + F_{size} \Rightarrow Act_{size} : Act_{size} \in \mathbb{R}^*. \quad (14)$$

4.2. JMA—Fitness Evaluation

The Java macaque depends on the genetic and social behavior in which dominant individuals are likely to be a winner in a problem space. The fitness evaluation indicates how far the individual is converged concerning the problem space. Fitness evaluation predicts the probability and survival rate of the individual to be transferred to the next generation of the population and is defined as:

$$\begin{aligned} & \text{minimize} && F(\Psi) \\ & \text{s.t} && \Psi \in \mathbb{G}, \end{aligned} \quad (15)$$

where ψ denotes an individual of x -dimensional integer vector $\psi_i = (\psi_1, \psi_2, \dots, \psi_x)$, $\mathbb{G}$ represent the feasible region of the search space, and $F \in \mathbb{R}$, where F is the optimization value.

4.3. JMA—Categorization

The basic structure of the java monkey follows the dominant hierarchy where the higher-ranking individuals dominate the remaining lower-ranking individuals in the group. The dominant male of the group also shows its dominance in accessing food, places, and other resources. Likewise, the dominant female of the group receives more access to resources and protection from a dominant male. Thus, the dominance hierarchy is based on evaluations of fitness value. Hence, the individuals are subdivided into higher-ranking or non-dominated (dominant) individuals and lower-ranking or dominated (non-dominant) individuals. Therefore, the individual ranking is based on the fitness value. The best individual in the group is known as the “Alpha” individual. Then, the ‘Alpha Male’ is the best male, and the ‘Alpha Female’ is the best female individual in the group. These are represented as follows:

$$AM_i = \min \{F(\Psi_j) \mid j \in \{M\}_i, i \in \{G_i\}\} \quad (16)$$

$$AF_i = \min \{F(\Psi_j) \mid j \in \{F\}_i, i \in \{G_i\}\}, \quad (17)$$

where AM_i and AF_i are the local optimal solution for the problem $\mathbb{X}$ from the group $\{G_i\}$. Therefore, the $AM_i, AF_i \in \mathbb{X}$, and it is better than other individuals from the set $\{M\}_i$ and $\{F\}_i$, where $(\forall AM_i, AF_i \in \mathbb{G} \Rightarrow (\forall AM, AF \in \mathbb{X} : F(AM_i) < \{M\}_i, F(AF_i) < \{F\}_i))$.

The global best individual from the set $\{AM\}$ and $\{AF\}$ are selected using:

$$[GM, GF] = \min\{\{AM\}, \{AF\}\}, \quad (18)$$

where $GM, GF \in \mathbb{X}$ are globally the best solution for the individual from the *POP*.

Then the set of dominant and non-dominant (subordinate) male and female individuals in the group can also be formulated using the fitness value.

$$\Psi, \Psi^* \in \{M\} \text{ then } = \begin{cases} \Psi & \in \{DS\}, \text{ if } F(\Psi) > F(\Psi^*) \\ \Psi^* & \in \{DS\}, \Psi \in \{NDS\}, \text{ otherwise,} \end{cases} \quad (19)$$

$$\Psi, \Psi^* \in \{F\} \text{ then } = \begin{cases} \Psi & \in \{NDS\}_F, \text{ if } F(\Psi) > F(\Psi^*) \\ \Psi^* & \in \{NDS\}_F, \Psi \in \{DS\}_F, \text{ otherwise,} \end{cases} \quad (20)$$

where both Ψ and Ψ^* are either distinct male or female individuals from the G_i . The non-dominant set of male and female individuals are represented as $\{NDS\}_M$ and $\{NDS\}_F$. Similarly, the dominant male and female individuals are presented as $\{DS\}_M$ and $\{DS\}_F$.

4.4. JMA—Mating

Mating is an important operator in the Java macaque algorithm that ensures group survival and also enables the exchange of genetic information and social behavior between individuals. In particular, the dominant males have the privilege of mating with dominant females. Hence, non-dominant males may also have a lesser chance of mating due to the

special prerequisite of dominance. The selection of individuals in the mating process plays an important role in generating populations for the next generation. Specifically, if the selection process depends only on a dominant individual, this may lead to local optima and reduce population diversity. Thus, the non-dominant individuals are also selected to perform mating operations with the probability ratio, which maintains diversity among individuals. Mating is the search process which is used to exploit the problem space $\mathbb{X}$. Mating between males Ψ^m and females Ψ^f is either from the set {NDS} or {DS}. Then, new offspring Ψ_{off} are generated as follows:

$$\Psi^{off} = \text{Mating}(\Psi^m, \Psi^f) \quad s.t. \quad \Psi_m, \Psi_f \in G_i, \quad (21)$$

where the uniform crossover is used for the discrete optimization problem, and the simulated binary crossover for the continuous optimization problem.

The age of the Infant Male (IM) and Infant Female (IF) is set at 0, who then undergo a learning process to improve their fitness. The offspring generated in each $\{G_i\}$ from the mating process reaches sexual maturity $\mathbb{S}$. Then the SM, SF and AF represent the Stray Male (subadult male), Subadult Female, and Adult Female, defined as the following:

$$\Psi^{IM} = \begin{cases} \Psi^{IM} \in \{JM\}, & \text{if Age} = 1 \\ \Psi^{IM} \in \{IM\}, & \text{otherwise.} \end{cases} \quad (22)$$

$$\Psi^{JM} = \begin{cases} \Psi^{JM} \in \{SM\}, & \text{if Age} = 4 \\ \Psi^{JM} \in \{JM\}, & \text{otherwise.} \end{cases} \quad (23)$$

$$\Psi^{IF} = \begin{cases} \Psi^{IF} \in \{JF\}, & \text{if Age} = 1 \\ \Psi^{IF} \in \{IF\}, & \text{otherwise.} \end{cases} \quad (24)$$

$$\Psi^{JF} = \begin{cases} \Psi^{JF} \in \{SF\}, & \text{if Age} = 3 \\ \Psi^{JF} \in \{JF\}, & \text{otherwise;} \end{cases} \quad (25)$$

$$\Psi^{SF} = \begin{cases} \Psi^{SF} \in \{AF\}, & \text{if Age} = 5 \\ \Psi^{SF} \in \{SF\}, & \text{otherwise,} \end{cases} \quad (26)$$

where *Age* indicates the age of the individual. Similarly, if the *Age* = 5, then the Juvenile Female (JF) is moved to the set of Adult Females (AFs).

4.5. JMA—Male Replacement

The new offspring generated from the mating operation undergo a learning process from the dominant individual and the circumstances of the environment. Male replacement plays an important role in the Java macaque algorithm where the stray male chooses another group to reside in and replace the existing dominant male in the group. This is also considered as swarm behavior, where the stray male has to find a suitable position in the ruling hierarchy of the male. On the contrary, if a stray male cannot find a suitable position, then it is subjected to the risk of death. According to the fitness of the stray male, the replacement strategy variation plays out in a different manner. The male replacement can be defined as:

$$MR = \begin{cases} \text{REPLACE}(\Psi_{sm}, \Psi_m), & \text{if } F(\Psi_{sm}) > F(\Psi_m) \\ \Psi_{sm} \in \{ES\}, & \text{otherwise;} \end{cases} \quad (27)$$

$$\exists \Psi_{sm} \in G_i || \Psi_m \in \{DS\}_M : (\Psi_m \in \{G_j\}_{j=1,2,\dots,M}) | i \neq j,$$

where $\text{REPLACE}(\Psi_{sm}, \Psi_m)$ is the replacement process with the stray Ψ_{sm} and dominant Ψ_m male. Then $\{ES\}$ is the elimination set where the individuals are eliminated from the population.

4.6. JMA—Learning

In this stage, the individuals in the population set can improve their fitness to reach the dominant individual set. The efficiency of the individual is improved based on the environment and social behavior. However, the potential of this learning process is to increase efficiency when compared with the current dominant hierarchy. Exploration is achieved by the learning method which enhances the fitness of an individual and also attains the optimal solution. The learning method is applied to the Java macaque algorithm to attain the desired results in global optimum, and this can be represented as follows:

$$\text{Learning} = \{POP, \mathbb{G}, \delta, L(\Psi), F(\Psi), \mathbb{X}\}, \quad (28)$$

where POP represents the set of the individual, $\mathbb{G}$ is the feasible search space of the solution, $F(\Psi)$ is the fitness function, $\mathbb{X}$ is the problem space of the optimization problem, and δ is referred to as the learning rate of the individual which is uniformly distributed ($0 \leq \delta \leq 1$). Then, the learning process is $L(\Psi_k, i, j) : \forall \Psi \in \mathbb{G} | \exists x \in \mathbb{X}$.

5. Continuous Optimization Problem

In the continuous optimization problem, each decision variable may take any value between the range of constraints. Thus, the search space of continuous optimization may exponentially increase with the associated constraints. The initial difference between the continuous and discrete optimization is the representation of the individual with regard to a continuous search space. The java macaque for the continuous optimization algorithm consists of significant features selection, categorization, mating, male replacement and learning behavior. The Algorithm 1 clearly describes the computational methodology of java macaque optimization algorithm.

Definition 1. The individuals are mapped ($\text{map} : \mathbb{G} \rightarrow \mathbb{X}$) between the dimension vector of the search space $\mathbb{G}$ to a dimensional vector in the problem space $\mathbb{X}$. In the continuous optimization problem, the individuals are mapped to the number of decision variables (NOD) and represented as the dimension vector of the problem space.

$$\begin{aligned} \vec{\Psi} &= \{\psi_1, \psi_2, \dots, \psi_{NOD}\} \quad \vec{\Psi} \cdot \psi \in \mathbb{G}_i, \forall i \in 1, 2, \dots, NOD, \\ \exists \vec{\Psi} \cdot x \in \mathbb{X} : \text{map}(\vec{\Psi} \cdot \psi) &= \vec{\Psi} \cdot x \end{aligned} \quad (29)$$

where $\vec{\Psi}$ is a vector of decision variable $\{\psi_1, \psi_2, \dots, \psi_{NOD}\}$ in the search space $\mathbb{G}$. Each individual for the continuous function is represented as $\Psi \in \mathbb{R}$ and $\mathbb{G} = \mathbb{X} \subseteq \mathbb{R}$. Then the next level of difference in the continuous optimization problem is the evaluation of the fitness value of the individual.

Definition 2. The generic form of the fitness evaluation is presented as follows:

$$\begin{aligned} \text{Minimize} \quad & F(\vec{\Psi}) \\ \text{S.t} \quad & \forall \vec{\Psi} \in \mathbb{X}, \end{aligned} \quad (30)$$

where $F(\vec{\Psi})$ is the fitness value of individuals with respect to the continuous optimization problem, and $\mathbb{X}$ is the decision space $\{F : \mathbb{X} \rightarrow \mathbb{R}^{NOC} : \vec{lb}_i \leq \vec{\Psi}_i \leq \vec{ub}_i\}$ with the upper bound (ub) and the lower bound (lb) of individual $\vec{\Psi}$.

Then the remaining processes, like selecting the alpha male, alpha female, global best individual, and subdividing the population into dominated and non-dominated individuals, are completed using the fitness value.

Definition 3. Mating is an important search process of JMA which is used to exploit the continuous search space $\mathbb{G}$. Thus, the mating process is redefined with respect to the continuous search space. Mating between males $\vec{\Psi}_m$ and females $\vec{\Psi}_f$ is either from the set $\{DS\}$ or $\{NDS\}$. Therefore, A_i and

I_i are obtained from $\max(\psi_i^m, \psi_i^f)$ and $\min(\psi_i^m, \psi_i^f)$. Then, new offspring $\overrightarrow{\Psi}_{off}$ are generated as follows [58–60]:

$$\overrightarrow{\Psi}_{off} = \text{Mating}(\overrightarrow{\Psi}_m, \overrightarrow{\Psi}_f) \quad \text{s.t.} \quad \overrightarrow{\Psi}_m, \overrightarrow{\Psi}_f \in G_i \quad (31)$$

$$|off| = \{P(A) * (|G_i| - Act_{size})\}$$

$$\overrightarrow{\Psi}_{off}^0 = 0.5[(A_i + I_i) - \theta_0 \times (A_i - I_i)] \quad (32)$$

$$\overrightarrow{\Psi}_{off}^1 = 0.5[(A_i + I_i) + \theta_1 \times (A_i - I_i)], \quad (33)$$

where $\overrightarrow{\Psi}_{off}^0$ and $\overrightarrow{\Psi}_{off}^1$ are two new offspring generated from the crossover, and $\theta_j (j = 0, 1)$ can be incurred as

$$\theta_j = \begin{cases} (C_j \times D_j)^{\frac{1}{\zeta+1}}, & \text{if } C_j \leq 1/D_j \\ (\frac{1}{2-C_j \times D_j})^{\frac{1}{\zeta+1}}, & \text{otherwise,} \end{cases} \quad (34)$$

where C_j is a uniformly distributed random number between $[0, 1]$ and θ is the distributed crossover index of offspring related to the parent's natal code. Then $D_j (j = 0, 1)$ is generated by the assumption $A_i \neq I_i$:

$$D_j = \begin{cases} 2 - (\frac{A_i - I_i}{A_i + I_i - 2lb_i})^{(\zeta+1)} & j = 0 \\ 2 - (\frac{A_i - I_i}{2ub_i - A_i - I_i})^{(\zeta+1)} & j = 1, \end{cases} \quad (35)$$

where lb_i and ub_i indicate the upper and lower bound of the decision variable.

Definition 4. The learning process is adopted to improve the fitness value of the individual with respect to the problem space. Then, the learning process of the Java macaque may change with respect to the environment. Then the learning model for the continuous optimization problem is defined as follows:

$$\text{Learning} = \{\text{PoP}, \mathbb{G}, \overrightarrow{\delta}, L(\overrightarrow{\Psi}_k), F(\overrightarrow{\Psi}), \mathbb{X}\} \quad (36)$$

where POP represents the set of individuals, $\mathbb{G}$ is the feasible search space of solutions, $F(\overrightarrow{\Psi})$ is the fitness function, and $\overrightarrow{\delta}$ indicates the learning rate of the individual in linear decreasing order ($0 \leq \overrightarrow{\delta} \leq 1$). Then, the learning process of the individual is given by

$$\overrightarrow{L}_1(\overrightarrow{\Psi}_{GM}, \overrightarrow{\Psi}_k) := \overrightarrow{\Psi}_{GM} - (2 \cdot \overrightarrow{\delta} \cdot \overrightarrow{r}_1 - \overrightarrow{\delta}) | 2 \cdot \overrightarrow{r}_2 \cdot \overrightarrow{\Psi}_{GM} - \overrightarrow{\Psi}_k |, \quad (37)$$

where $\overrightarrow{r}_1, \overrightarrow{r}_2$ are the random vectors between $[0, 1]$. Similarly, the learning process of the individual $\overrightarrow{\Psi}_k$ is performed between the global best female $\overrightarrow{L}_2(\overrightarrow{\Psi}_{GF}, \overrightarrow{\Psi}_k)$, alpha male $\overrightarrow{L}_3(\overrightarrow{\Psi}_{AM}, \overrightarrow{\Psi}_k)$ and alpha female $\overrightarrow{L}_4(\overrightarrow{\Psi}_{AF}, \overrightarrow{\Psi}_k)$. Then, the individual $\overrightarrow{\Psi}_k$ is modified as follows:

$$\overrightarrow{\Psi}_k = \frac{\overrightarrow{L}_1 + \overrightarrow{L}_2 + \overrightarrow{L}_3 + \overrightarrow{L}_4}{4}, \quad (38)$$

where $\overrightarrow{\Psi}_i$ is the individual obtained from the learning process and replaced in the POP.

Algorithm 1 JMAC: Java Macaque Algorithm for Continuous Optimization Problem.

Input: N number of individuals, g number of groups, F is the fitness function.

Output: GB_{AM} , GB_{FM} , POP

Step 1: [Initialization] The initial Population is initialized using a random seeding technique using Equation (29).

Step 2: [Evaluation] Each individual in the population is evaluated using the fitness function, as represented in Equation (30).

Step 3: The various categories of search agents are classified as:

(a) [Alpha Individual] Determine the AlphaMale and AlphaFemale in each group using Equations (16) and (17).

(b) [GB-Alpha Individual] Select the best individual from the AlphaMale and AlphaFemale sets using Equation (18).

(c) [NDS and DS] Male and Female individual sets are further divided into dominant and non-dominant solutions using Equations (19) and (20).

Step 4: [Mating] Then the new offsprings are generated using the mating process, defined as in Equation (31).

Step 5: [Evaluation of Offspring] The fitness values of offspring are evaluated and the stray male and female are determined.

Step 6: [Male Replacement] The stray male finds a suitable group to replace the dominant male.

Step 7: [Learning] The fitness value of the individual is improved by moving the global best and alpha individuals by Equation (36).

Step 8: [Termination] The population is maintained by using Equation (11) and the above process is repeated from Step 2 until termination criteria are satisfied.

5.1. Experimentation and Result Analysis of Continuous Optimization Problem

In this section of experimentation, the benchmark function is taken from the popular literature [61,62] and used for analyzing the performance of the Java macaque algorithm. The performance of the proposed Java macaque algorithm is measured on 23 benchmark functions. The benchmark function utilized for the experimentation process consists of the 7 unimodal functions, 6 multimodal functions, and 10 fixed-dimension multimodal functions, where the dimension of the function is indicated in Dim, the upper and lower bound of the search space is referred to using Range, and finally, the optimal value of the benchmark function is indicated using the f_{min} .

In the experimentation process, the proposed JMA is compared with the dominant algorithm from the literature, such as the grey wolf optimization (GWO) technique [61] and spider monkey algorithm (SMO) [12]. For all the algorithms, the population size was fixed as 200, stopping criteria were set as 100 iterations, and the primary population was produced using the random population seeding techniques. Every algorithm executed the 30 independent runs for each benchmark function, and the evaluation of the algorithm was measured using statistical measures such as the mean and standard deviation, as referred to in the literature [61,62]. Further, the best result is mentioned in bold font for reference in the respective table.

5.1.1. Result Analysis for Unimodal Benchmark Functions

The unimodal benchmark functions are suitable for analyzing the exploitation capability of the algorithm because it contains only one global optimum. Table 2 clearly shows a list of unimodal benchmark functions which is used for the experimentation process, and the function has different properties, like convex and non-convex shaped, non-differentiable, discontinuous, scalable, separable, and non-separable properties. Table 3 clearly shows the results of SMO, GWO and JMA on the unimodal benchmark functions. The performance of the proposed Java macaque optimization algorithm has clear dominance over the other two algorithms. In particular, the JMA attains the best results in both means and standard deviation on all the unimodal instances. GWO typically performs well in F5 and F6 in terms of means value, but struggles to outperform JMA. The Figure 2 facilitates the clear

illustration that JMA dominates the other algorithm in terms of better convergence. The JMA has a dominant exploitation capability and can achieve a well-converged population. Further, it can clearly be depicted via standard deviations.

Table 2. Benchmark Function.

Function	Dim	Range	f_{min}
$f_1(x) = \sum_{i=1}^n x_i^2$	30	$[-100, 100]$	0
$f_2(x) = \sum_{i=1}^n x_i + \prod_{i=1}^n x_i $	30	$[-10, 10]$	0
$f_3(x) = \sum_{i=1}^n \left(\sum_{j=1}^i x_j \right)^2$	30	$[-100, 100]$	0
$f_4(x) = \max_{1 \leq i \leq n} \{ x_i \}$	30	$[-100, 100]$	0
$f_5(x) = \sum_{i=1}^n \left 100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2 \right $	30	$[-30, 30]$	0
$f_6(x) = \sum_{i=1}^n (x_i + 0.5)^2$	30	$[-100, 100]$	0
$f_7(x) = \sum_{i=1}^n ix_i^4 + U(0, 1)$	30	$[-1.28, 1.28]$	0

Table 3. Experimental results of unimodal benchmark function.

Function	Technique	Optimal Value	Best	SD
F1	SMO	0	9.64E-02	1.09E+04
	GWO		1.21E-07	7.92E+03
	JMA		1.06E-30	1.47E+03
F2	SMO	0	8.96E-01	2.45E+10
	GWO		3.18E-05	9.84E+08
	JMA		2.50E-17	1.40E+02
F3	SMO	0	1.28E+02	1.83E+04
	GWO		3.28E-01	1.67E+04
	JMA		5.43E-24	5.21E+03
F4	SMO	0	1.15E+00	1.93E+01
	GWO		3.31E-02	1.75E+01
	JMA		4.75E-14	6.63E+00
F5	SMO	0	8.85E+01	2.67E+07
	GWO		2.58E+01	2.20E+07
	JMA		2.81E+01	1.80E+06
F6	SMO	0	1.21E-01	1.23E+04
	GWO		3.89E-04	7.57E+03
	JMA		3.98E-04	2.09E+03
F7	SMO	0	3.15E-01	1.78E+01
	GWO		1.76E-03	7.61E+00
	JMA		1.15E-03	6.35E-01

Figure 2. Convergence Curve for Schwefel 2.21 Function (F4).

5.1.2. Multimodal Benchmark Function

The next important set of test functions used for the experimentation process is multimodal benchmark functions. The multimodal functions are utilized for analyzing the exploration potential of the algorithm because of its multiple local minima. Further, the benchmark function measures the global exploration capability of the algorithm, and an increase in dimension exponentially increases the number of local optima in the search space. Table 4 shows the list of multimodal benchmark functions for the experimentation.

Table 4. Multimodal benchmark function.

Function	Dim	Range	f_{min}
$f_8(x) = \sum_{i=1}^n -x_i \sin(\sqrt{ x_i })$	30	$[-500, 500]$	-418.9829×5
$f_9(x) = \sum_{i=1}^n x_i^2 - 10 \cos(2\pi x_i) + 10 $	30	$[-5.12, 5.12]$	0
$f_{10}(x) = -20 \exp\left(-0.2 \sqrt{\frac{1}{n} \sum_{i=1}^n x_i^2}\right) - \exp\left(\frac{1}{n} \sum_{i=1}^n \cos(2\pi x_i)\right) + 20 + e$	30	$[-32, 32]$	0
$f_{11}(x) = \frac{1}{4000} \sum_{i=1}^n x_i^2 - \prod_{i=1}^n \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	30	$[-600, 600]$	0
$f_{12}(x) = \frac{\pi}{n} \left\{ 10 \sin(\pi y_1) + \sum_{i=1}^{n-1} (y_i - 1)^2 [1 + 10 \sin^2(\pi y_{i+1})] + (y_n - 1)^2 \right\} + \sum_{i=1}^n u(x_i, 10, 100, 4)$ $y_i = 1 + \frac{x_i + 1}{4} u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m & x_i > a \\ 0 - a & x_i < a \\ k(-x_i - a)^m & x_i < -a \end{cases}$	30	$[-50, 50]$	0
$f_{13}(x) = 0.1 \left\{ \sin^2(3\pi x_1) + \sum_{i=1}^n \frac{(x_i - 1)^2 [1 + \sin^2(3\pi x_i + 1)]}{[1 + \sin^2(2\pi x_n)]} + (x_n - 1)^2 \right\} + \sum_{i=1}^n u(x_i, 5, 100, 4)$	30	$[-50, 50]$	0

The result illustrated in Table 5 and Figure 3 exhibits the performance of existing and proposed algorithms in terms of means and standard deviation. From the table observation, it is clearly shown that the performance of JMA rules out the other two algorithms. The JMA demonstrates its significance in mean value as 0.00E+00, 8.88E-16, 0.00E+00, 3.87E-05 and 3.21E-04 for the instance of F9 to F13, whereas the GWO achieves only 8.74E+00, 7.69E-05, 1.45E-07, 6.52E-03 and 4.83E-04. Similarly, the performance of standard deviation also demonstrates the dominance of the proposed algorithm over the GWO and SMO. Hence, this observation clearly provides the proof that the exploring capability of the proposed algorithm is superior to the existing algorithms.

Table 5. Experimental results of the multimodal benchmark function.

Function	Technique	Optimal Value	Best	SD
F8	SMO	−12569.5	−4706.47	2.60E+02
	GWO		−4374.35	1.18E+03
	JMA		−3384.65	1.26E+01
F9	SMO	0	7.81E+01	6.70E+01
	GWO		8.74E+00	9.46E+01
	JMA		0.00E+00	5.04E+01
F10	SMO	0	3.78E−01	3.77E+00
	GWO		7.69E−05	4.85E+00
	JMA		8.88E−16	2.94E+00
F11	SMO	0	2.78E+00	1.76E+02
	GWO		1.45E−07	6.94E+01
	JMA		0.00E+00	1.46E+01
F12	SMO	0	1.36E−03	5.15E+07
	GWO		6.52E−03	7.12E+07
	JMA		3.87E−05	2.61E+06
F13	SMO	0	5.45E−02	1.07E+08
	GWO		4.83E−04	9.37E+07
	JMA		3.21E−04	9.53E+06

Figure 3. Convergence curve for the Ackley function (F10).

5.1.3. Fixed-Dimension Multimodal Benchmark Function

The final experimentation for a continuous Java macaque algorithm is experimented with using the fixed-dimension multimodal benchmark functions. In this test function, the dimension of the benchmark function is fixed, which is used to analyze the performance of the algorithm in terms of exploration, exploitation, and also to avoid the local minima. The list of benchmark functions used for experimentation is shown in Table 6.

The results described in Table 7 clearly show that the JMA attains better values for the performance measures by comparing the GWO and SMO. The mean values of JMA and GWO are same as 1 for the instance F14, but the JMA shows its dominance in the SD value as 0.194221, whereas the GWO reaches only 1.737368. Similarly, Table 7 portrays how the performance of the three algorithms are almost the same as for the instance F18, but the proposed algorithm dominates in standard deviation. On the other hand, the performance of JMA lacks in terms of standard deviation for the instances F20 and F22 by GWO, and F17 and F21 by SMO. However, the JMA algorithm outperformed the GWO and SMO in terms of mean value for instances like f14, F16, F18, F19, F20, F21, F22 and F23. The fixed-dimension multi-modal benchmark function is utilized for analysis in the performance of the algorithm due to its multiple local minima, exploitation, and exploration capabilities.

Table 6. Fixed-dimension multimodal benchmark function.

Function	Dim	Range	f_{min}
$f_{14}(x) = \left(\frac{1}{500} + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^2 (x_i - a_{ij})^6} \right)^{-1}$	2	[−65, 65]	1
$f_{15}(x) = \sum_{i=1}^{11} \left[a_i - \frac{x_1(b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4} \right]^2$	4	[−5, 5]	0.00030
$f_{16}(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	2	[−5, 5]	−1.0316
$f_{17}(x) = \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6 \right)^2 + 10 \left(1 - \frac{1}{8\pi} \right) \cos x_1 + 10$	2	[−5, 5]	0.398
$f_{18}(x) = \left[1 + (x_1 + x_2 + 1)^2 (19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2) \right] \times [30 + (2x_1 - 3x_2)^2 \times (18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]$	2	[−2, 2]	3
$f_{19}(x) = -\sum_{i=1}^4 c_i \exp \left(-\sum_{j=1}^3 a_{ij} (x_j - p_{ij})^2 \right)$	3	[1, 3]	−3.86
$f_{20}(x) = -\sum_{i=1}^4 c_i \exp \left(-\sum_{j=1}^6 a_{ij} (x_j - p_{ij})^2 \right)$	6	[0, 1]	−3.32
$f_{21}(x) = -\sum_{i=1}^5 \left[(X - a_i)(X - a_i)^T + c_i \right]^{-1}$	4	[0, 10]	−10.1532
$f_{22}(x) = -\sum_{i=1}^7 \left[(X - a_i)(X - a_i)^T + c_i \right]^{-1}$	4	[0, 10]	−10.4028
$f_{23}(x) = -\sum_{i=1}^{10} \left[(X - a_i)(X - a_i)^T + c_i \right]^{-1}$	4	[0, 10]	−10.5363

Table 7. Experimental results of the fixed-dimension multimodal benchmark function.

Function	Technique	Optimal Value	Best	SD
F14	SMO	1	1.08E+00	1.13E+00
	GWO		1.00E+00	1.74E+00
	JMA		1.00E+00	1.94E-01
F15	SMO	0.0003	4.26E-04	4.18E-03
	GWO		3.08E-04	5.26E-04
	JMA		3.23E-04	9.30E-03
F16	SMO	−1.0316	−1.03E+00	1.87E-02
	GWO		−1.03E+00	6.11E-02
	JMA		−1.03E+00	7.90E-03
F17	SMO	0.398	3.98E-01	3.81E-02
	GWO		3.98E-01	7.42E-02
	JMA		3.98E-01	4.15E-02
F18	SMO	3	3.00E+00	8.54E-01
	GWO		3.00E+00	1.85E+00
	JMA		3.00E+00	6.43E-01
F19	SMO	−3.86	−3.86E+00	3.67E-02
	GWO		−3.86E+00	3.11E-02
	JMA		−3.86E+00	2.45E-02
F20	SMO	−3.22	−3.32E+00	1.28E-01
	GWO		−3.32E+00	5.63E-02
	JMA		−3.07E+00	1.43E-01
F21	SMO	−10.1532	−1.02E+01	2.13E+00
	GWO		−1.01E+01	2.38E+00
	JMA		−1.02E+01	2.65E+00
F22	SMO	−10.4028	−1.04E+01	3.40E+00
	GWO		−1.04E+01	2.42E+00
	JMA		−1.04E+01	3.03E+00
F23	SMO	−10.5363	−1.05E+01	2.71E+00
	GWO		−1.05E+01	2.87E+00
	JMA		−1.05E+01	2.52E+00

The result shown in Table 8 and sample Figure 4 helps to analyze the experimentation in terms of convergence. The means value of 30 independent runs for the best convergence

value in iterations 1, 10, and 25 illustrates the search ability of algorithms. The best value obtained from the iteration limit of 1 clearly demonstrated stronger convergence behavior of the JMA over the GWO and SMO in terms of all the unimodal instances. Further, the JMA was able to attain a near-optimal value in just 25 iterations for instances like F1, F2, F3, F4 and F7 with respective values 8.56E-04, 1.09E-02, 2.76E-01, 1.16E-01, and 7.76E-04, respectively. Additionally, in the case of the multimodal benchmark function, the proposed JMA led to the existing algorithm by all means. Correspondingly, in terms of the fixed dimensional multimodal function, the proposed JMA outperforms the existing algorithm in almost all instances, except F17, F21, and F22. The SMO algorithm shows better performance in terms of F17 and F21, whereas the GWO dominates only in F22. This observation shows that the potential behavior of the proposed JMA prevails over the existing algorithm in terms of avoiding local optima and achieving exploitation, exploration, and strong convergence in minimum iteration.

Table 8. Best values obtained by JMA, GWO and SMO in Iterations 1, 10, and 25.

Function	Technique	Iteration			Function	Technique	Iteration		
		1	10	25			1	10	25
F1	SMO	5.66E+04	1.28E+04	2.29E+02	F13	SMO	9.37E+08	3.10E+05	3.31E+01
	GWO	5.73E+04	2.23E+03	1.84E+00		GWO	8.34E+08	6.01E+04	2.27E+00
	JMA	1.11E+04	2.58E+01	8.56E-04		JMA	9.58E+07	7.49E+00	2.52E+00
F2	SMO	2.46E+11	6.32E+01	3.81E+01	F14	SMO	1.04E+01	1.00E+00	9.98E-01
	GWO	9.89E+09	1.34E+01	3.73E-01		GWO	1.09E+01	2.07E+00	9.99E-01
	JMA	1.41E+03	2.11E+00	1.09E-02		JMA	2.10E+00	1.20E+00	9.98E-01
F3	SMO	1.01E+05	2.45E+04	2.97E+03	F15	SMO	4.12E-02	4.70E-03	8.52E-04
	GWO	1.05E+05	2.27E+04	2.25E+03		GWO	3.47E-03	1.05E-03	7.26E-04
	JMA	3.93E+04	1.56E+03	2.76E-01		JMA	9.36E-02	7.02E-04	7.02E-04
F4	SMO	8.89E+01	4.00E+01	7.37E+00	F16	SMO	−8.52E-01	−1.03E+00	−1.03E+00
	GWO	7.72E+01	3.25E+01	4.02E+00		GWO	−4.18E-01	−1.03E+00	−1.03E+00
	JMA	4.18E+01	5.90E+00	1.16E-01		JMA	−9.76E-01	−1.03E+00	−1.03E+00
F5	SMO	2.51E+08	8.24E+05	7.85E+04	F17	SMO	7.74E-01	4.00E-01	3.98E-01
	GWO	1.90E+08	4.18E+05	1.34E+02		GWO	6.84E-01	5.56E-01	3.99E-01
	JMA	1.71E+07	1.44E+03	2.88E+01		JMA	6.49E-01	4.49E-01	4.02E-01
F6	SMO	6.48E+04	1.06E+04	1.59E+02	F18	SMO	1.16E+01	3.14E+00	3.00E+00
	GWO	5.74E+04	7.93E+02	2.20E+00		GWO	2.16E+01	3.01E+00	3.00E+00
	JMA	2.00E+04	4.51E+01	3.43E+00		JMA	9.42E+00	3.00E+00	3.00E+00
F7	SMO	8.82E+01	4.04E+01	4.04E+01	F19	SMO	−3.50E+00	−3.85E+00	−3.86E+00
	GWO	6.81E+01	5.55E-01	6.53E-03		GWO	−3.67E+00	−3.84E+00	−3.85E+00
	JMA	5.17E+00	1.10E-02	7.76E-04		JMA	−3.63E+00	−3.84E+00	−3.86E+00
F8	SMO	−3.25E+03	−3.70E+03	−4.15E+03	F20	SMO	−2.13E+00	−3.01E+00	−3.11E+00
	GWO	−2.85E+03	−3.83E+03	−3.83E+03		GWO	−2.92E+00	−3.11E+00	−3.18E+00
	JMA	−2.83E+03	−2.96E+03	−2.96E+03		JMA	−2.22E+00	−3.23E+00	−3.29E+00
F9	SMO	4.03E+02	3.25E+02	3.18E+02	F21	SMO	−7.42E-01	−4.67E+00	−9.36E+00
	GWO	4.11E+02	2.12E+02	6.19E+01		GWO	−7.32E-01	−2.91E+00	−6.09E+00
	JMA	3.08E+02	7.21E+01	5.99E-03		JMA	−5.13E-01	−3.82E+00	−4.25E+00
F10	SMO	2.04E+01	1.04E+01	6.81E+00	F22	SMO	−9.54E-01	−1.95E+00	−5.23E+00
	GWO	2.05E+01	1.04E+01	6.82E-01		GWO	−8.49E-01	−3.89E+00	−6.45E+00
	JMA	1.77E+01	2.73E+00	5.31E-03		JMA	−8.38E-01	−2.38E+00	−4.20E+00
F11	SMO	5.89E+02	4.79E+02	3.15E+02	F23	SMO	−1.11E+00	−4.74E+00	−7.82E+00
	GWO	5.56E+02	1.04E+01	9.28E-01		GWO	−1.49E+00	−6.06E+00	−8.27E+00
	JMA	1.25E+02	1.17E+00	7.83E-04		JMA	−1.39E+00	−2.96E+00	−3.93E+00
F12	SMO	4.65E+08	2.10E+01	4.39E+00					
	GWO	6.06E+08	2.46E+04	2.70E-01					
	JMA	2.62E+07	1.23E+00	5.28E-01

Figure 4. Convergence curve for Shekel 7 Function (F22).

6. Discrete Optimization Problem

The problem space of a discrete optimization problem is represented as the set of all feasible solutions that satisfy the constraint and the fitness function, which maps each element to the problem space. Thus, the discrete or combinatorial optimization problem searches for the optimal solution from the set of feasible solutions.

The generic form of the discrete optimization problem is represented as follows [63]:

$$\text{Minimize } F(\Psi) \quad (39)$$

$$\text{subject to } \forall \Psi \in \mathbb{G},$$

where $F : \mathbb{G} \rightarrow \mathbb{Z}$ is the objective function with discrete problem space, which maps each individual in the search space $\mathbb{G}$ to the problem space $\mathbb{X}$ and the set of feasible solutions is $\mathbb{G} \subset \mathbb{X}$.

The individuals were generated with the mapping ($map : \mathbb{G} \rightarrow \mathbb{X}$) between the dimensional vector of the search space $\mathbb{G}$ to the dimensional vector in the problem space $\mathbb{X}$. In the discrete optimization problem, the total number of elements (NOC) is represented as the dimensional vector of the problem space, and it must be mapped as the element of search spaces of individual.

$$\Psi = \{c_1, c_2, \dots, c_{NOC}\} \quad \forall \Psi.c \in \mathbb{G}, \exists \Psi.x \in \mathbb{X} : map(\Psi.c) = \Psi.x, \quad (40)$$

where Ψ is an individual, represented as a tuple $(\Psi.c, \Psi.x)$ of a dimensional vector $\Psi.c$ in the search space $\mathbb{G}$ and the corresponding dimensional vector $\Psi.x = \Psi.c$ in the problem space $\mathbb{X}$. G_i presents the group i with M_{size} of male individuals and F_{size} of female individuals in the population. The initialization process starts with a minimum number of individuals in the group with respect to the problem size.

The learning process is a mechanism which adapts the learning model for enhancing individual fitness. However, it increases individual fitness by exploiting the search space with respect to the problem space. The individual Ψ in the POP should improve its fitness value $F(\Psi)$ via learning and increase the probability of attaining a global optimum. Then, the learning for discrete optimization is defined as:

$$\text{Learning} = \{POP, \mathbb{G}, \delta, L(\Psi_k, i, j), F(\Psi), \mathbb{X}\}, \quad (41)$$

where POP represents the set of individuals, $\mathbb{G}$ is the feasible search space of solutions, $F(\Psi)$ is the fitness function, and δ is referred to as the learning rate of the individual between $(0 \leq \delta \leq 1)$.

Then, $L(\Psi_k, i, j)$ is the learning process $L(\Psi_k, i, j) : \Psi \rightarrow \mathbb{G}, \forall i, j \in \Psi$.

$$[i, j] = \text{sort}[\text{ceil}(x * \text{rand}(\delta, 2))], \quad (42)$$

where the two values are randomly generated for i and j . Thus, the x is in linear decreasing order, generated between $\max$ to 3 ($\max \leftarrow \mathbb{G}$).

$$(Case\ 1) : L_1(\Psi, [i, j]) = L(\Psi, [j, i]) \quad (43)$$

$$(Case\ 2) : L_2(\Psi, i, j) = L(\Psi, j : -1 : i) \quad (44)$$

$$(Case\ 3) : L_3(\Psi, i, j) = L(\Psi, [i + 1 : j]) \quad (45)$$

$$(Case\ 4) : L_4(\Psi, i, j) = L(\Psi, i, j) \quad (46)$$

$$\Psi^* = best\{F(L_1), F(L_2), F(L_3), F(L_4)\} \text{ where } \Psi^* \in POP, \quad (47)$$

where Ψ^* is the best individual obtained from the different learning process and replaced Ψ_k in the POP.

6.1. Experimentation and Result Analysis of Travelling Salesman Problem

The mathematical model for the travelling salesman problem is formulated as:

$$F = \min \sum_{i=1}^{NOC-1} D(C_i, C_{i+1}) + D(C_{NOC}, C_1), \quad (48)$$

where $D(C_i, C_{i+1})$ is the distance between two cities C_i and C_{i+1} and $D(C_{NOC}, C_1)$ indicate the tour between the last city C_{NOC} and first city C_1 .

The standard experimental setup for the proposed Java macaque algorithm is as follows: (i) the initial population was randomly generated with 60 individuals in each group ($m = 60$); (ii) the number of groups ($n = 5$); (iii) executed up to 1000 iterations. The performance of the proposed JMA is correlated with an Imperialist Competitive Algorithm with a particle swarm optimization (ICA-PSO) [64], Fast Opposite Gradient Search with Ant Colony Optimization (FOGS-ACO) (Saenphon et al., 2014 [65]), and effective hybrid genetic algorithm (ECOGA) (Li and Zhang, 2007 [66]). Each algorithm was run on each instance 25 times, and hence, the best among the 25 runs was considered for analysis and validation purposes.

6.2. Parameter for Performance Assessment

This section briefly explains the list of parameters which evaluates the performance of the proposed Java macaque algorithm with the existing algorithm. It also helps to explore the efficiency of the proposed algorithm in various aspects. The various types of investigation parameters are the convergence rate, error rate, convergence diversity, and average convergence from the population. Thus, the parameter for the performance assessment is summarized as follows [3,67]:

Best Convergence Rate: In the experimentation, the best convergence rate measures the quality of the best individual obtained from the population in terms of percentage with regard to the optimal value. It can be measured as follows:

$$BestConv.(%) = 1 - \frac{F(\Psi_{best}) - Opt.Fit.}{Opt.Fit.} \times 100, \quad (49)$$

where $F(\Psi_{best})$ is the fitness of the best individual in the population and $Opt.Fit.$ indicates the optimal fitness value of the instances.

Average Convergence Rate: The average percentage of the fitness value of the individual in the population with regard to the optimal fitness value is known as the average convergence rate. This can be calculated as:

$$AvgConv.(%) = 1 - \frac{F(POP_{avg}) - Opt.Fit.}{Opt.Fit.} \times 100, \quad (50)$$

where $F(POP_{avg})$ indicates the average fitness of all the individuals in the population.

Table 10. Medium-scale TSP instances.

S. No	TSP Instance	Technique	Optimum Value	Fitness			Convergence Rate (%)			Error Rate (%)			Convergence Diversity
				Best	Average	Worst	Best	Average	Worst	Best	Average	Worst	
5	tsp225	ICA-PSO	3919	4217.98	5084.07	5782.75	92.37	70.27	52.44	7.63	29.73	47.56	39.93
		FOGS-ACO		4164.41	4547.41	5199.74	93.74	83.97	67.32	6.26	16.03	32.68	26.42
		ECO-GA		4056.84	5015.69	5671.25	96.48	72.02	55.29	3.52	27.98	44.71	41.19
		JMA		3979.27	4477.03	5121.15	98.46	85.76	69.33	1.54	14.24	30.67	29.14
6	pr264	ICA-PSO	49201	59,574.21	64,837.84	73,339.96	78.92	68.22	50.94	21.08	31.78	49.06	27.98
		FOGS-ACO		51,675.05	57,959.70	66,105.45	94.97	82.20	65.64	5.03	17.80	34.36	29.33
		ECO-GA		49,254.00	63,845.90	72,011.25	99.89	70.23	53.64	0.11	29.77	46.36	46.25
		JMA		49,215.00	56,967.00	65,121.15	99.84	84.06	67.46	0.16	15.94	32.54	32.37
7	lin318	ICA-PSO	42,029	53,514.61	58,841.06	63,104.28	72.67	60.00	49.86	27.33	40.00	50.14	22.82
		FOGS-ACO		52,319.74	52,967.00	55,780.78	75.52	73.98	67.28	24.48	26.02	32.72	8.23
		ECO-GA		51,091.87	58,861.06	61,755.46	78.44	59.95	53.06	21.56	40.05	46.94	25.37
		JMA		49,797.00	52,957.00	54,785.00	81.52	74.00	69.65	18.48	26.00	30.35	11.87
8	fl417	ICA-PSO	11,861	14,331.81	15,644.21	17,201.57	79.17	68.10	54.97	20.83	31.90	45.03	24.19
		FOGS-ACO		12,448.05	13,973.67	15,458.00	95.05	82.19	69.67	4.95	17.81	30.33	25.38
		ECO-GA		11,861.00	15,535.60	17,118.54	100.00	69.02	55.67	0.00	30.98	44.33	44.33
		JMA		11,861.00	13,845.06	15,458.00	100.00	83.27	69.67	0.00	16.73	30.33	30.33
9	d493	ICA-PSO	35,002	44,419.18	49,475.36	54,273.86	73.10	58.65	44.94	26.90	41.35	55.06	28.15
		FOGS-ACO		43,449.12	44,565.08	48,190.98	75.87	72.68	62.32	24.13	27.32	37.68	13.55
		ECO-GA		42,389.06	48,025.28	53,152.86	78.90	62.79	48.14	21.10	37.21	51.86	30.75
		JMA		41,358.00	43,134.00	47,364.00	81.84	76.77	64.68	18.16	23.23	35.32	17.16

Table 11. Large-scale TSP instances.

S. No	TSP Instance	Technique	Optimum Value	Fitness			Convergence Rate (%)			Error Rate (%)			Convergence Diversity
				Best	Average	Worst	Best	Average	Worst	Best	Average	Worst	
10	rat575	ICA-PSO	6773	8294.57	9630.68	10,568.23	77.53	57.81	43.97	22.47	42.19	56.03	33.57
		FOGS-ACO		8083.38	8687.46	9436.00	80.65	71.73	60.68	19.35	28.27	39.32	19.97
		ECO-GA		7884.19	9489.22	10,511.34	83.59	59.90	44.81	16.41	40.10	55.19	38.79
		JMA		7686.00	8537.00	9436.00	86.52	73.96	60.68	13.48	26.04	39.32	25.84
11	u724	ICA-PSO	41910	45,757.90	56,315.70	62,157.88	90.82	65.63	51.69	9.18	34.37	48.31	39.13
		FOGS-ACO		44,512.60	50,428.30	55,927.52	93.79	79.67	66.55	6.21	20.33	33.45	27.24
		ECO-GA		43,251.30	55,138.40	60,126.32	96.80	68.44	56.53	3.20	31.56	43.47	40.26
		JMA		41,988.00	49,161.00	54,248.00	99.81	82.70	70.56	0.19	17.30	29.44	29.25
12	pr1002	ICA-PSO	259,045	313,554.45	358,324.65	413,357.62	78.96	61.67	40.43	21.04	38.33	59.57	38.53
		FOGS-ACO		272,002.25	322,458.35	375,264.00	95.00	75.52	55.14	5.00	24.48	44.86	39.86
		ECO-GA		259,121.00	350,743.30	411,544.30	99.97	64.60	41.13	0.03	35.40	58.87	58.84
		JMA		259,145	314,587.00	375,264.00	99.96	78.56	55.14	0.04	21.44	44.86	44.83
13	u1060	ICA-PSO	224,094	245,345.46	292,904.92	316,853.79	90.52	69.29	58.61	9.48	30.71	41.39	31.91
		FOGS-ACO		237,510.64	261,491.76	283,448.30	94.01	83.31	73.51	5.99	16.69	26.49	20.50
		ECO-GA		231,877.82	283,841.16	310,468.10	96.53	73.34	61.46	3.47	26.66	38.54	35.07
		JMA		225,165	252,428.00	278,945.00	99.52	87.36	75.52	0.48	12.64	24.48	24.00

7.1. Analyzes Based on Best Convergence Rate

The best convergence rate reflects the effectiveness of the population-generated by optimization algorithm. Figure 5 depicts a comparison of the proposed algorithm to the existing algorithm in terms of convergence rate. The JMA holds lesser value as 74.23% for pr76 in small-scale instances. For medium-scale instances, the JMA holds 100% for the instance fl417. Then, the ECO-GA has the highest value of 99.97%, and JMA achieves convergence of 99.96% for the instance pr1002. As a result, on a medium scale, the instance lin318 has the lowest value of a 72.67% convergence rate in the ICA-PSO and the highest of 81.52 % for the JMA and 75.52 % for the FOGS-ACO, while the ECO-GA comes in second with a value of 78.44 %. In terms of convergence rate, the proposed Java macaque algorithm outperforms the existing algorithm.

Figure 5. Best convergence rate.

7.2. Analyzes Based on Average Convergence Rate

Figure 6 depicts the results of an average convergence rate comparison between the proposed algorithm and existing algorithms. As an outcome, the investigation in terms of average convergence for JMA revealed that it outperformed the other algorithm. The average convergence rate of JMA has to attain a value above 65%, but the FOGS-ACO, ECO-GA, and ICA-PSO obtained values above 64%, 51%, and 50%. The JMA achieved a maximum value of 87.36% for u1060, while the FOGS-ACO achieved 88.74% for pr144, and ECO-GA reached an average convergence of 73.34% for u1060, and lastly, the ICA-PSO attained a maximum of 70.02%, for instance, tsp225, respectively. While comparing the performance of JMA and FOGS-ACO for the average convergence rate, both algorithms perform quite well, but JMA dominates in many cases.

Figure 6. Average convergence rate.

7.3. Analyzes Based on the Worst Convergence Rate

Figure 7 shows the performance of the experimentation with regard to the worst convergence rate. The proposed Java macaque algorithm dominated the existing algorithm by all means. For example, by acknowledging the occurrence of instance pr144, the ICA-PSO, FOGS-ACO, ECO-GA, and JMA obtained the worst convergence rates of 53.92%, 69.47%, 57.21%, and 72.62%, respectively. Then, by considering the pr1002 instance from a large-scale dataset, the proposed JMA and FOGS-ACO held an equal value of 55.14%, which was followed by an ECO-GA value of 41.13%, and the lowest value of 40.43% was attained by ICA-PSO. Further, in terms of medium-scale instances, the proposed Java macaque algorithm also shows its superiority over the existing algorithm, such as FOGS-ACO, ECO-GA, and ICA-PSO.

Figure 7. Worst convergence rate.

7.4. Analyzes Based on Error Rate

Figure 8 depicts the algorithm's performance in terms of error rate. The best error rate shows how far the best individual's convergence rate deviates from the optimal fitness value, whereas the worst error rate shows the difference between the worst individual's convergence rate and the optimal solution.

Figure 8. Error rate.

Consequently, the maximum error rate of 25.83 percent for particle swarm optimization, 24.14 percent for ant colony optimization, 22.37 percent for genetic algorithm, Java macaque algorithm is 20.52 percent, and in that order. Meanwhile, the least error rate value for JMA was 0% for instances such as eil51, eil76, tsp225, pr264, fl417, u724, pr1002 and u1060, whereas the genetic algorithm had 0% for instances eil51, fl417 and the particle swarm optimization had an approximate value of 8% for instances eil76, tsp225 and u724, and FOGS-ACO has less than 3% for instance eil51, respectively. JMA obtained better performance compared with the existing algorithm.

7.5. Analyzes Based on Convergence Diversity

One of the important assessment criteria that provides a concrete evidence of the optimal solution is convergence diversity. According to Figure 9, the performance of the existing ECO-GA surpassed the other algorithms in terms of diversity. For the sample instance u1060, the existing FOGS-ACO has a convergence diversity of 10.58%, the JMA has a convergence diversity for the instance as 12.19%, then the ICA-PSO has a value of 22%, whereas the ECO-GA achieved a highest value of 23.31%. The proposed JMA dominated the existing FOGS-ACO and was dominated by the other existing techniques,

such as ECO-GA and ICA-PSO. Thus, the proposed JMA maintained the satisfactory level of convergence diversity among the population, and also achieved the best performance in terms of convergence with the existing algorithm.

Figure 9. Convergence diversity.

8. Conclusions

The research proposed in this paper was motivated by the natural behavior of Java macaque monkeys and is suitable for solving real-world optimization problems. Thus, the Java macaque poses peculiar behavior with natural intelligence and a social hierarchy in an optimized way that well-suits the modeling of the novel Java macaque optimization algorithm. It was developed using the selection, categorisation, mating, male replacement, and learning behavior of the Java macaque. The important strategy exhibited by the population of the Java macaque is a dominance hierarchy, and that was incorporated with the JMA as a selection process. Hence, the individuals with higher social ranking dominated other individuals in the population and mostly gained preference in the mating process. Further, it helped the individuals with dominant status to gain a higher probability of generating new infants when considered with low-ranking individuals. Then, the algorithm also utilises the male replacement model, which increases the adaptive search capability of the JMA. Further, the JMA utilises the learning model that enhances the performance of the individual by increasing fitness, which proportionally upgrades social status. Hence, the JMA utilises the different search operations, which maintains well-balanced exploration and exploitation in finding the optimal solution. The performance of the proposed algorithm was analyzed with discrete and continuous optimization problems. The experimentation of continuous optimization was extensively conducted on 23 benchmark functions with unimodal, multimodal and fixed-dimension multimodal functions, and the proposed JMA depicts its dominant performance over the GWO and SMO. The outcomes of the experimentation were discussed as follows: (1) The unimodal functions are suitable for analyzing the exploitation capability of the algorithm because they contain only one global optimum. Thus, the result from Table 3 clearly depicts the dominance of JMA over other techniques. Hence, the JMA is well-suited for solving problems with exploitation behavior. (2) The multimodal functions are utilized for analyzing the exploration potential of the algorithm because of its multiple local minima, that is, the global exploration capability of the algorithm increases with an exponential increase in the number of local optima in the search space. Further, the performance of the experimentation illustrated from Table 5 also shows the dominance of the proposed system. (3) The final experimentation for the continuous Java macaque algorithm was experimented using the fixed-dimension multimodal benchmark functions. In this, the dimension of the benchmark function is fixed, which is used to analyze the performance of the algorithm in terms of exploration and exploitation. The results described in Table 7 clearly show that the JMA is well-suited

to maintain the balance between exploration and exploitation. Further, the experimentation conducted over a discrete optimization problem by using the travelling salesman problem has to maintain balanced exploration and exploitation with a quality selection process to achieve an optimal result. Thus, the results evidently show the performance of JMA over the existing dominant algorithm, such as FOGS-ACO, ECO-GA and ICA-PSO. Thus, the robustness of the JMA over the continuous and discrete search space clearly illustrates its potential over the optimization problem.

9. Future Enhancements

Future research will be focused on the development of the novel Java macaque algorithm with a hyper-optimization-based parameter-controlling casual system which efficiently explores and exploits the search space. Hence, the algorithm utilises social hierarchy-based selection, mating, male replacement, and a learning process as the operation of JMA. On the other hand, the grooming behavior of females and aggression behavior of males can also be incorporated with the JMA for enhancing the search process in the future. This algorithm enhancement may indeed improve the efficiency of the proposed algorithm for solving different kinds of real-world problems.

Author Contributions: Conceptualization, S.R.; methodology, A.D.; validation, R.S. and M.R.; formal analysis, D.K.; writing—original draft preparation, D.K.; writing—review and editing, M.R. and A.G.; supervision, S.S.A. and A.S.A.; funding acquisition, S.S.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by Taif University, Research Supporting Project Number (TURSP-2020/215), Taif University, Taif, Saudi Arabia.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data available on request due to restrictions.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

NIOAs	Nature-Inspired Optimization Algorithms
TSP	Travelling Salesman Problem
EA	Evolutionary Algorithm
SI	Swarm Intelligence
GA	Genetic Algorithm
DE	Differential Evolution
ACO	Ant Colony Optimization
ABC	Artificial Bee Colony
PSO	Particle Swarm Optimization
FA	Firefly Algorithm
CS	Cuckoo Search
MA	Monkey Algorithm
SMO	Spider Monkey Algorithm
ASMO	Ageist Spider Monkey Algorithm
GWO	Grey Wolf Optimization
JMA	Java macaque Algorithm
KH	Krill Herd
BFA	Bacterial foraging algorithm
AIS	Artificial Immune System
CA	Culture Algorithm

RA	Reptile Algorithm
ICA-PSO	Imperialist Competitive Algorithm with particle swarm optimization
FOGS-ACO	Fast Opposite Gradient Search with Ant Colony Optimization
ECOGA	SEffective hybrid genetic algorithm
AM	Adult Male
SM	Sub-Adult Male
JM	Juvenile Male
IM	Infant Male
AF	Adult Female
SF	Sub-Adult Female
JF	Juvenile Female
IF	Infant Female
α	Death rate of Infant
β	Death rate of Non-Infant
POP	Population of Java macaque
$\{G_i\}$	presents the Group i
M_{size}	Male individuals in the population
F_{size}	female individuals in the population
$F(\Psi)$	Fitness evaluation individuals
GM, GF	Global best male and female
$\{DS\}$	Dominant set of individuals
$\{NDS\}$	Non-Dominant set of individuals
δ	Learning rate of the individual

References

- Uniyal, N.; Pant, S.; Kumar, A.; Pant, P. Nature-inspired metaheuristic algorithms for optimization. *Meta-Heuristic Optim. Tech. Appl. Eng.* **2022**, *10*, 1. [\[CrossRef\]](#)
- Alsalihi, B.; Mirjalili, S.; Abualigah, L.; Gandomi, A.H. A Comprehensive Survey on the Recent Variants and Applications of Membrane-Inspired Evolutionary Algorithms. *Arch. Comput. Methods Eng.* **2022**, 1–17. [\[CrossRef\]](#)
- Dinesh, K.; Amudhavel, J.; Rajakumar, R.; Dhavachelvan, P.; Subramanian, R. A novel self-organisation model for improving the performance of permutation coded genetic algorithm. *Int. J. Adv. Intell. Paradig.* **2020**, *17*, 299–322. [\[CrossRef\]](#)
- Storn, R.; Price, K. Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces. *J. Glob. Optim.* **1997**, *11*, 341–359. [\[CrossRef\]](#)
- Liu, C.; Wu, L.; Huang, X.; Xiao, W. Improved dynamic adaptive ant colony optimization algorithm to solve pipe routing design. *Knowl.-Based Syst.* **2022**, *237*, 107846. [\[CrossRef\]](#)
- Karaboga, D.; Basturk, B. A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm. *J. Glob. Optim.* **2007**, *39*, 459–471. [\[CrossRef\]](#)
- Latchoumi, T.; Balamurugan, K.; Dinesh, K.; Ezhilarasi, T. Particle swarm optimization approach for waterjet cavitation peening. *Measurement* **2019**, *141*, 184–189. [\[CrossRef\]](#)
- Cao, L.; Ben, K.; Peng, H.; Zhang, X. Enhancing firefly algorithm with adaptive multi-group mechanism. *Appl. Intell.* **2022**, 1–21. [\[CrossRef\]](#)
- Tang, C.; Song, S.; Ji, J.; Tang, Y.; Tang, Z.; Todo, Y. A cuckoo search algorithm with scale-free population topology. *Expert Syst. Appl.* **2022**, *188*, 116049. [\[CrossRef\]](#)
- Yilmaz, S.; Küçüksille, E.U. A new modification approach on bat algorithm for solving optimization problems. *Appl. Soft Comput.* **2015**, *28*, 259–275. [\[CrossRef\]](#)
- Ahmed, O.; Hu, M.; Ren, F. PEDTARA: Priority-Based Energy Efficient, Delay and Temperature Aware Routing Algorithm Using Multi-Objective Genetic Chaotic Spider Monkey Optimization for Critical Data Transmission in WBANs. *Electronics* **2022**, *11*, 68. [\[CrossRef\]](#)
- Tarawneh, H.; Alhadid, I.; Khwaldeh, S.; Afaneh, S. An Intelligent Cloud Service Composition Optimization Using Spider Monkey and Multistage Forward Search Algorithms. *Symmetry* **2022**, *14*, 82. [\[CrossRef\]](#)
- Abualigah, L.; Abd Elaziz, M.; Sumari, P.; Geem, Z.W.; Gandomi, A.H. Reptile Search Algorithm (RSA): A nature-inspired meta-heuristic optimizer. *Expert Syst. Appl.* **2022**, *191*, 116158. [\[CrossRef\]](#)
- Jain, S. Mammals: Whale, Gray Wolf, and Bat Optimization. In *Nature-Inspired Optimization Algorithms with Java*; Apress: Berkeley, CA, USA, 2022; [\[CrossRef\]](#)
- Yang, X.S. Mathematical Analysis of Nature-Inspired Algorithms. In *Nature-Inspired Algorithms and Applied Optimization*; Springer: Cham, Switzerland, 2018; Volume 744. [\[CrossRef\]](#)
- Karunanidhy, D.; Ramalingam, R.; Dumka, A.; Singh, R.; Alsukayti, I.; Anand, D.; Hamam, H.; Ibrahim, M. An Intelligent Optimized Route-Discovery Model for IoT-Based VANETs. *Processes* **2021**, *9*, 2171. [\[CrossRef\]](#)

17. Qin, S.; Pi, D.; Shao, Z.; Xu, Y. Hybrid collaborative multi-objective fruit fly optimization algorithm for scheduling workflow in cloud environment. *Swarm Evol. Comput.* **2022**, *68*, 101008. [\[CrossRef\]](#)
18. dos Anjos, J.C.S.; Gross, J.L.G.; Matteussi, K.J.; González, G.V.; Leithardt, V.R.Q.; Geyer, C.F.R. An Algorithm to Minimize Energy Consumption and Elapsed Time for IoT Workloads in a Hybrid Architecture. *Sensors* **2021**, *21*, 2914. [\[CrossRef\]](#)
19. Dinesh, K.; Subramanian, R.; Dweib, I.; Nandhini, M.; Mohamed, M.Y.N.; Rajakumar, R. Bi-directional self-organization technique for enhancing the genetic algorithm. In Proceedings of the 6th International Conference on Information Technology: IoT and Smart City, Hong Kong, 29 December 2018–31 December 2019; pp. 251–255. [\[CrossRef\]](#)
20. Jana, S.; Dey, A.; Maji, A.K.; Pal, R.K. Solving Sudoku Using Neighbourhood-Based Mutation Approach of Genetic Algorithm. In *Advanced Computing and Systems for Security: Volume 13*; Springer: Singapore, 2022; pp. 153–167. [\[CrossRef\]](#)
21. Al-Sharhan, S.; Bimba, A. Adaptive multi-parent crossover GA for feature optimization in epileptic seizure identification. *Appl. Soft Comput.* **2019**, *75*, 575–587. [\[CrossRef\]](#)
22. Manicassamy, J.; Karunanidhi, D.; Pothula, S.; Thirumal, V.; Ponnurangam, D.; Ramalingam, S. GPS: A constraint-based gene position procurement in chromosome for solving large-scale multiobjective multiple knapsack problems. *Front. Comput. Sci.* **2018**, *12*, 101–121. [\[CrossRef\]](#)
23. Zhang, P.; Yao, H.; Li, M.; Liu, Y. Virtual network embedding based on modified genetic algorithm. *Peer-to-Peer Netw. Appl.* **2019**, *12*, 481–492. [\[CrossRef\]](#)
24. Gholami, O.; Sotskov, Y.N.; Werner, F. A genetic algorithm for hybrid job-shop scheduling problems with minimizing the makespan or mean flow time. *J. Adv. Manuf. Syst.* **2018**, *17*, 461–486. [\[CrossRef\]](#)
25. Karunanidhi, D.; Amudhavel, J.; Datchinamurthy, T.S.; Ramalingam, S. A Novel Java macaque Algorithm For Travelling Salesman Problem. *IIOAB J.* **2017**, *8*, 252–261.
26. Dinesh, K.; Rajakumar, R.; Subramanian, R. Self-organisation migration technique for enhancing the permutation coded genetic algorithm. *Int. J. Appl. Manag. Sci.* **2021**, *13*, 15–36. [\[CrossRef\]](#)
27. Santos, J.; Ferreira, A.; Flintsch, G. An adaptive hybrid genetic algorithm for pavement management. *Int. J. Pavement Eng.* **2019**, *20*, 266–286. [\[CrossRef\]](#)
28. Luan, J.; Yao, Z.; Zhao, F.; Song, X. A novel method to solve supplier selection problem: Hybrid algorithm of genetic algorithm and ant colony optimization. *Math. Comput. Simul.* **2019**, *156*, 294–309. [\[CrossRef\]](#)
29. Bujok, P.; Tvrdík, J.; Poláková, R. Comparison of nature-inspired population-based algorithms on continuous optimisation problems. *Swarm Evol. Comput.* **2019**, *50*, 100490. [\[CrossRef\]](#)
30. Ramos-Figueroa, O.; Quiroz-Castellanos, M.; Mezura-Montes, E.; Kharel, R. Variation operators for grouping genetic algorithms: A review. *Swarm Evol. Comput.* **2021**, *60*, 100796. [\[CrossRef\]](#)
31. Opara, K.R.; Arabas, J. Differential Evolution: A survey of theoretical analyzes. *Swarm Evol. Comput.* **2019**, *44*, 546–558. [\[CrossRef\]](#)
32. Wang, X.; Wang, Y.; Wong, K.C.; Li, X. A self-adaptive weighted differential evolution approach for large-scale feature selection. *Knowl.-Based Syst.* **2022**, *235*, 107633. [\[CrossRef\]](#)
33. Müller, F.M.; Bonilha, I.S. Hyper-Heuristic Based on ACO and Local Search for Dynamic Optimization Problems. *Algorithms* **2022**, *15*, 9. [\[CrossRef\]](#)
34. Mahmoodi, L.; Aliyari Shoorehdeli, M. Comments on “A Novel Fault Diagnostics and Prediction Scheme Using a Nonlinear Observer With Artificial Immune System as an Online Approximator”. *IEEE Trans. Control. Syst. Technol.* **2017**, *25*, 2243–2246. [\[CrossRef\]](#)
35. Coulter, N.; Moncayo, H. Artificial Immune System Optimized Support Vector Machine for Satellite Fault Detection. In Proceedings of the AIAA SCITECH 2022 Forum, San Diego, CA, USA, 3–7 January 2022; p. 1713. [\[CrossRef\]](#)
36. Passino, K.M. Biomimicry of bacterial foraging for distributed optimization and control. *IEEE Control. Syst.* **2002**, *22*, 52–67.
37. Hernández-Ocaña, B.; Chávez-Bosquez, O.; Hernández-Torruco, J.; Canul-Reich, J.; Pozos-Parra, P. Bacterial Foraging Optimization Algorithm for menu planning. *IEEE Access* **2018**, *6*, 8619–8629. [\[CrossRef\]](#)
38. Awad, H.; Hafez, A. Optimal operation of under-frequency load shedding relays by hybrid optimization of particle swarm and bacterial foraging algorithms. *Alex. Eng. J.* **2022**, *61*, 763–774. [\[CrossRef\]](#)
39. Wang, G.G.; Gandomi, A.H.; Alavi, A.H.; Deb, S. A hybrid method based on krill herd and quantum-behaved particle swarm optimization. *Neural Comput. Appl.* **2016**, *27*, 989–1006. [\[CrossRef\]](#)
40. Saravanan, D.; Janakiraman, S.; Harshavardhanan, P.; Kumar, S.A.; Sathian, D. Enhanced Binary Krill Herd Algorithm for Effective Data Propagation in VANET. In *Secure Communication for 5G and IoT Networks*; Springer: Cham, Switzerland, 2022; pp. 221–235. [\[CrossRef\]](#)
41. Yang, X.S.; He, X. Firefly algorithm: Recent advances and applications. *Int. J. Swarm Intell.* **2013**, *1*, 36–50. [\[CrossRef\]](#)
42. Cheng, Z.; Song, H.; Chang, T.; Wang, J. An improved mixed-coded hybrid firefly algorithm for the mixed-discrete SSCGR problem. *Expert Syst. Appl.* **2022**, *188*, 116050. [\[CrossRef\]](#)
43. Mohanty, P.P.; Nayak, S.K. A Modified Cuckoo Search Algorithm for Data Clustering. *Int. J. Appl. Metaheuristic Comput. (IJAMC)* **2022**, *13*, 1–32. [\[CrossRef\]](#)
44. Kumar, S.; Kumari, R. Modified position update in spider monkey optimization algorithm. *Int. J. Emerg. Technol. Comput. Appl. Sci. (IJETCAS)* **2014**, *2*, 198–204.
45. Wu, G.; Mallipeddi, R.; Suganthan, P.N.; Wang, R.; Chen, H. Differential evolution with multi-population based ensemble of mutation strategies. *Inf. Sci.* **2016**, *329*, 329–345. [\[CrossRef\]](#)

46. Ma, H.; Shen, S.; Yu, M.; Yang, Z.; Fei, M.; Zhou, H. Multi-population techniques in nature inspired optimization algorithms: A comprehensive survey. *Swarm Evol. Comput.* **2019**, *44*, 365–387. [\[CrossRef\]](#)
47. Xu, Z.; Liu, X.; Zhang, K.; He, J. Cultural transmission based multi-objective evolution strategy for evolutionary multitasking. *Inf. Sci.* **2022**, *582*, 215–242. [\[CrossRef\]](#)
48. Warwas, K.; Tengler, S. Multi-population Genetic Algorithm with the Actor Model Approach to Determine Optimal Braking Torques of the Articulated Vehicle. In *Intelligent Computing*; Springer: Cham, Switzerland, 2022; Volume 283, pp. 56–74. [\[CrossRef\]](#)
49. Hesar, A.S.; Kamel, S.R.; Houshmand, M. A quantum multi-objective optimization algorithm based on harmony search method. *Soft Comput.* **2021**, *25*, 9427–9439. [\[CrossRef\]](#)
50. Zhang, Y.; Li, J.; Li, L. A Reward Population-Based Differential Genetic Harmony Search Algorithm. *Algorithms* **2022**, *15*, 23. [\[CrossRef\]](#)
51. Van den Bercken, J.; Cools, A. Information-statistical analysis of factors determining ongoing behaviour and social interaction in Java monkeys (*Macaca fascicularis*). *Anim. Behav.* **1980**, *28*, 189–200. [\[CrossRef\]](#)
52. Veenema, H.C.; Spruijt, B.M.; Gispén, W.H.; Van Hooff, J. Aging, dominance history, and social behavior in Java-monkeys (*Macaca fascicularis*). *Neurobiol. Aging* **1997**, *18*, 509–515. [\[CrossRef\]](#)
53. Dewsbury, D.A. Dominance rank, copulatory behavior, and differential reproduction. *Q. Rev. Biol.* **1982**, *57*, 135–159. [\[CrossRef\]](#) [\[PubMed\]](#)
54. Engelhardt, A.; Pfeifer, J.B.; Heistermann, M.; Niemitz, C.; van Hooff, J.A.; Hodges, J.K. Assessment of female reproductive status by male longtailed macaques, *Macaca fascicularis*, under natural conditions. *Anim. Behav.* **2004**, *67*, 915–924. [\[CrossRef\]](#)
55. Sprague, D.S. Age, dominance rank, natal status, and tenure among male macaques. *Am. J. Phys. Anthropol.* **1998**, *105*, 511–521. [\[CrossRef\]](#)
56. Dasser, V. A social concept in Java monkeys. *Anim. Behav.* **1988**, *36*, 225–230. [\[CrossRef\]](#)
57. Paul, A.; Thommen, D. Timing of birth, female reproductive success and infant sex ratio in semifree-ranging Barbary macaques (*Macaca sylvanus*). *Folia Primatol.* **1984**, *42*, 2–16. [\[CrossRef\]](#)
58. Durillo, J.J.; Nebro, A.J. jMetal: A Java framework for multi-objective optimization. *Adv. Eng. Softw.* **2011**, *42*, 760–771. [\[CrossRef\]](#)
59. Deb, K.; Beyer, H.G. Self-Adaptive Genetic Algorithms with Simulated Binary Crossover. *Evol. Comput.* **2001**, *9*, 197–221. [\[CrossRef\]](#)
60. Lin, Q.; Chen, J.; Zhan, Z.H.; Chen, W.N.; Coello, C.A.C.; Yin, Y.; Lin, C.M.; Zhang, J. A hybrid evolutionary immune algorithm for multiobjective optimization problems. *IEEE Trans. Evol. Comput.* **2016**, *20*, 711–729. [\[CrossRef\]](#)
61. Meidani, K.; Hemmasian, A.; Mirjalili, S.; Barati Farimani, A. Adaptive grey wolf optimizer. *Neural Comput. Appl.* **2022**, 1–21. [\[CrossRef\]](#)
62. Zhang, L.; Liu, L.; Yang, X.S.; Dai, Y. A novel hybrid firefly algorithm for global optimization. *PLoS ONE* **2016**, *11*, e0163230. [\[CrossRef\]](#) [\[PubMed\]](#)
63. Di Gaspero, L.; Schaerf, A.; Cadoli, M.; Slany, W.; Falaschi, M. Local Search Techniques for Scheduling Problems: Algorithms and Software Tool. Ph.D. Thesis, Università degli Studi di Udine, Udine, Italy, 2003.
64. Idoumghar, L.; Chérin, N.; Siarry, P.; Roche, R.; Miraoui, A. Hybrid ICA–PSO algorithm for continuous optimization. *Appl. Math. Comput.* **2013**, *219*, 11149–11170. [\[CrossRef\]](#)
65. Saenphon, T.; Phimoltare, S.; Lursinsap, C. Combining new fast opposite gradient search with ant colony optimization for solving travelling salesman problem. *Eng. Appl. Artif. Intell.* **2014**, *35*, 324–334. [\[CrossRef\]](#)
66. Li, L.; Zhang, Y. An improved genetic algorithm for the traveling salesman problem. In *International Conference on Intelligent Computing*; Springer: Berlin/Heidelberg, Germany, 2007; Volume 2, pp. 208–216. [\[CrossRef\]](#)
67. Paul, P.V.; Moganarangan, N.; Kumar, S.S.; Raju, R.; Vengattaraman, T.; Dhavachelvan, P. Performance analyzes over population seeding techniques of the permutation-coded genetic algorithm: An empirical study based on traveling salesman problems. *Appl. Soft Comput.* **2015**, *32*, 383–402. [\[CrossRef\]](#)