Highlights

Voronoi Tessellations based Simple Optimizer

Prathu Bajpai, Jagdish Chand Bansal

- This paper proposed a computational geometry approach for improving population diversity in evolutionary algorithms.
- This includes a novel Voronoi Tessellations-based population generation method.
- A levy flight and Differential Evolution assisted approach for updating population in the search space.
- In addition, a population diameter-based switch is proposed to check the stagnation or premature convergence.

Voronoi Tessellations based Simple Optimizer

Prathu Bajpai[a], Jagdish Chand Bansal[a]

[a]*Faculty of Mathematics and Computer Sciences, South Asian University, Rajpur road, New Delhi, 110068, Delhi, India*

Abstract

Population diversity holds significant importance in determining the success of any evolutionary algorithm. It helps the algorithm in efficiently exploring the search space, and identifying the promising region(s) containing global optimal solution(s). However, during the optimization procedure the population may lose its diversity, causing premature convergence in the algorithm and resulting in approximations to the sub-optimal solution(s). This paper proposes a Voronoi Tessellations-based Simple Optimizer (VTSO) algorithm that utilizes a niche concept of Voronoi Tessellations (VTs) from the field of computational geometry to ensure a well-distributed population throughout the optimization procedure. It proposed an elite sampling mechanism that utilizes Lévy flights to aggressively explore the search space for locating potential optimal region(s), and the Differential Evolution (DE) algorithm to exploit these regions in order to approximate the global optimal solution(s). In addition, a population diameter-based switch is devised which activates itself when the algorithm detects premature convergence in the algorithm. Experiments are conducted on CEC14 and CEC17 benchmark test suit, and the proposed algorithm is compared with the existing state-of-the-art evolutionary algorithms. The results are competitive to recommend the VTSO algorithm as a new efficient and accurate optimizer for handling complex optimization problems.

Keywords: Computational Geometry, Voronoi Tessellations, Evolutionary Computing, Differential Evolution

1. Introduction

Evolutionary algorithms (EAs) are gaining popularity in various branches of science and engineering due to their remarkable problem-solving capabilities. These algorithms have been proving their success in solving complex real-world optimization problems ranging from machine learning applications to robotic systems [1, 2, 3]. These real-world optimization problems are defined over high-dimensional search

spaces and are predominantly characterized by non-linearity, non-convexity, and non-differentiability. Due to the unavailability of gradient-specific information and high computational cost, the traditional optimization methods are usually unable to solve these complex optimization problems. In contrast, evolutionary algorithms offer fast, computationally inexpensive, and gradient-free approaches for handling such complex optimization problems.

Evolutionary algorithms follow a multiple solution or population-based approach in which a population of solutions is evolved using stochastic operators such as mutation, crossover, and selection [4]. Several such algorithms have been proposed in the EAs literature, out of which, some popular algorithms are Genetic Algorithms (GA) [5], Differential Evolution (DE) [6], Particle population Optimizer (PSO) [7], Artificial Bee Colony (ABC) [8], Covariance Matrix Adaption Evolutionary Strategy (CMA-ES) [9], and many more. The performance of evolutionary algorithms is highly dependent on their ability to generate a well-distributed population during the optimization process [10]. The population distribution in the evolutionary algorithms is managed by pseudo-random number generation (PRNG) techniques. These PRNG techniques are responsible for ensuring that solutions are diverse, and are placed across the search space to facilitate thorough exploration. However, when the size of the population is small or the number of dimensions is large, these PRNG techniques fail to distribute solutions uniformly, resulting in different types of biases (like center bias, diagonal bias, or edge bias) in the algorithm [11].

Moreover, a less diverse population in the search space is more prone to being stuck in the local optimal regions, and it also limits the scope of the crossover operator [12]. For instance, when solutions from a less diverse population are crossed, the newly generated solutions share similarities with the original (or, parent) population. This might trap the population in the basin of local optimal solution(s) causing premature convergence or stagnation in the algorithm [13]. Whereas, a well-distributed population is beneficial for global exploration. It assists the algorithm in escaping local optimal regions, and also benefits the crossover operator in producing more diverse solutions [14]. In this research, one of the major focuses is to construct a novel population distribution method using the concept of Voronoi Tessellations (VTs), that are integral to the field of computational geometry.

VTs are known to generate more uniform samples/points in the search space by dividing the space into smaller regions known as Voronoi regions (see section 3). Although, VTs provide wide coverage of the underlying space, computing them becomes very expensive when the space is high-dimensional or the number of input seed points is large [15]. Since the design philosophy of evolutionary algorithms promotes simplicity and inexpensive computational methodologies, using VTs in their

original form may be computationally expensive. Motivated by this fact, a slicing-concatenation mechanism is proposed that uses a divide and conquer approach for utilizing 2− dimensional Voronoi Tessellations to generate D-dimensional samples. This reduces the computational cost of calculating Voronoi Tessellations for high-dimensional spaces and offers a novel way of increasing population diversity in EAs. The details of the proposed Voronoi Tessellations-based population distribution technique are given in the sub-section 5.2.1.

Other than this, the proposed VTSO algorithm incorporates an elite sampling mechanism (ESM) in which the sample population generated by VTs is perturbated using Lévy flights. It helps in aggressively exploring the search space and locating the locally optimal regions. To exploit the potential local regions flagged by ESM, the canonical Differential Evolution (DE) algorithm is used. In addition, a population diameter-based switch is proposed which activates the reinitialization of the population when the algorithm detects premature convergence (or, stagnation). This process is iteratively repeated till the termination criterion is not satisfied. The proposed algorithm is named, *Voronoi Tessellations-based Simple Optimizer (VTSO)* due to the presence of Voronoi Tessellations, and it is abbreviated as simple because of its easy design implementation. The experiments are conducted on 30D and 50D problems of CEC14 and CEC17 benchmark test suits, and experimental results support the competitive performance of the proposed VTSO algorithm over other compared state-of-the-art evolutionary algorithms.

The rest of the paper is organized as follows: Section 2 is the related work. Section 3 discusses the concept of Voronoi Tessellations. Section 4 explains the canonical Differential Evolution (DE) algorithm. Section 5 is the proposed Voronoi Tessellations-based Simple Optimizer (VTSO). Experimental results are reported in Section 6, and the discussion of results is given in Section 7. The conclusion and further research directions are added in Section 8 of the paper.

2. Related Work

Voronoi Tessellations based methods have been implemented in various kinds of optimization problems ranging from space decomposition, spatial pattern analysis, clustering, proximity resolutions, and motion planning [16, 17]. For instance, Chi et al.[18] utilized the concept of a generalized Voronoi diagram to propose a path-planning method for mobile robots. Wei et al.[19] proposed a bi-level Voronoi diagram-based meta-heuristic for solving large-scale multi-depot vehicle routing problems. Shatnawi et al.[20] initialized the population of Cuckoo Search(CS) using Centroid Voronoi Tessellations and reported improved performance. Zhang et al.[21]

proposed a Voronoi neighborhood-based evolutionary approach for solving multi-modal optimization problems. The authors proposed an approximation algorithm for finding Voronoi neighbors in the population. Individuals in the population were labeled into three different niches, and their positions were updated using the tailored search strategies of the Differential Evolution (DE) algorithm. Tong et al.[22] utilized the concept of Voronoi Tessellations in solving large-scale expensive optimization problems using a Voronoi-based local search mechanism. Okabe et al.[23] used a Voronoi Tessellations-based approach to construct stochastic models for generating offspring while solving multi-objective optimization problems. Similarly, Huang et al. utilized the Voronoi neighborhood-based Differential Evolution algorithm for solving multi-modal problems [24]. Pan et al.[25] integrated the Voronoi Tessellations based method for multi-UAV (Unmanned Ariel Vehicles) relay deployment. The success of Voronoi Tessellations-based methods is due to their capacity to generate well-distributed solutions in the search space. However, the applications of Voronoi Tessellations-based methods in the area of evolutionary algorithms require further investigation and attention. An overview of Voronoi Tessellations is presented in the next section 3.

3. Voronoi Tessellations or Voronoi Diagram

Computational Geometry methods play an important role in many practical applications like robotics, artificial intelligence, machine learning, and computer graphics [26]. The Voronoi Tessellations is one of the most useful data structures in computational geometry [27]. It is a well-recognized concept because of its distinguished topological and geometric properties, and known for its ability to produce more distributed sample points in the search space [15, 28]. The Voronoi Tessellations or Voronoi diagram, is a geometric framework that partitions a euclidean space into regions based on a set of points, known as sites or seed points. Mathematically it is defined as follows [29]:

Suppose $P = \{p_1, p_2, \ldots, p_N\}$ denotes a set of seed points or sites, and $dist(p_i, p_j)$ denotes the Euclidean distance between sites p_i and p_j. The line segment connecting p_i and p_j is denoted by $\overline{p_i p_j}$. A bisector or perpendicular line segment is drawn from the center of line segment $\overline{p_i p_j}$ to form half-plane $H(p_i, p_j)$ containing site p_i, and half-plane $H(p_j, p_i)$ containing site p_j. The half-plane $H(p_i, p_j)$ is a collection of points that are lying nearer to the site p_i. Mathematically this can be represented by equation (1).

$$H(p_i, p_j) = \{x | dist(x, p_i) < dist(x, p_j)\} \tag{1}$$

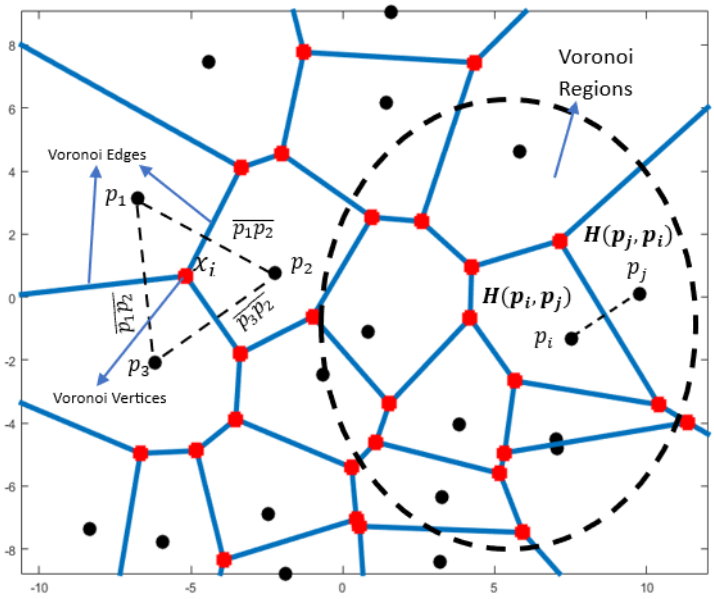


Figure 1: Voronoi Diagram of 20 randomly generated points in 2-dimensional space

The intersection of the half-planes corresponding to a specific site, say p_i produces the Voronoi region (or Voronoi cell), with respect to P denoted by $VR(\boldsymbol{p}_i, P)$, and defined in equation (2).

$$VR(\boldsymbol{p}_i, P) = \bigcap_{\boldsymbol{p}_j \in P, i \neq j} H(\boldsymbol{p}_i, \boldsymbol{p}_j) \tag{2}$$

Subsequently, the Voronoi diagram of the set P, denoted by $V(P)$, is defined using equation (3).

$$V(P) = \bigcup_{\boldsymbol{p}_i, \boldsymbol{p}_j \in P, i \neq j} \overline{VR(\boldsymbol{p}_i, P)} \cap \overline{VR(\boldsymbol{p}_j, P)} \tag{3}$$

Where, $\overline{VR(\boldsymbol{p}_i, P)}$ denotes the closure set of $VR(\boldsymbol{p}_i, P)$. It can be seen from equation (3) that, the Voronoi diagram of set P is the union of intersecting half-planes containing points of P. The corresponding Voronoi regions $VR(\boldsymbol{p}_i, P)$ $(\forall i = 1, 2, \ldots, N)$ are disjoint convex polygons. The shared boundary between two Voronoi regions is called the Voronoi edge and the endpoints of Voronoi edges are referred to as Voronoi vertices. The Voronoi vertices are formed by shared boundaries of three or more Voronoi regions. For given N site points, the number of Voronoi vertices and

Voronoi edges is at most $2N - 5$ and $3N - 6$, respectively [30]. A Voronoi diagram of a set containing 20 randomly generated points is illustrated in Figure 1.

It is worth noting that the position of Voronoi vertices is different from the seed points or sites in the search space. The population formed by combining the seed points with their corresponding Voronoi vertices is more distributed and covers the search space more efficiently. Now to evolve this population, an evolutionary approach is required and for that purpose, the DE algorithm is integrated into the optimization procedure of the proposed VTSO algorithm. For completeness, a brief overview of the DE algorithm is presented, in the next section 4.

4. Canonical Differential Evolution (DE) Algorithm

The DE algorithm is considered one of the popular evolutionary algorithms. It exploits the difference between the vectors in the population to guide the search process. There are three major operators associated with the DE algorithm which are mutation, crossover, and selection [31]. The mutation operation is performed using a mutation scheme referred to as mutation strategy. For each vector X_i in the current (or, parent) population, a mutant vector V_i is generated by the underlying mutation strategy. One popular mutation strategy known as 'DE/rand/1' is given in equation(4) below.

$$V_{i,G} = X_{r_1,G} + F \cdot (X_{r_2,G} - X_{r_3,G}) \tag{4}$$

Where, $V_{i,G}$ is a mutant vector, $X_{r_1,G}, X_{r_2,G}, X_{r_3,G}$ are randomly selected vectors known as target vectors and G is the generation counter. F is a control parameter known as the scaling factor or differential weight. Various mutation strategies have been proposed in the DE literature. Some widely accepted mutation strategies are 'DE/best/1', 'DE/rand-to-best/1', 'DE/best/2', and 'DE/rand-to-pbest/2'. After generating mutant vector $V_i = \{v_{i,1}, v_{i,2}, \ldots, v_{i,D}\}$, it is crossed with corresponding parent vector X_i to produce a trial vector $U_i = \{u_{i,1}, u_{i,2}, \ldots, u_{i,D}\}$ (D represents the dimension). One widely used crossover operator is binary crossover given in equation (5).

$$u_{i,j} = \begin{cases} v_{i,j} & \text{if } (rand(0,1) \leq CR) \\ x_{i,j} & \text{otherwise} \end{cases} \tag{5}$$

Where, $j = 1, 2, \ldots, D$, and $rand(0,1)$ is a random number between $(0,1)$. CR is a control parameter known as the crossover rate. Its value lies in the range $[0,1]$. The generated trial vectors are evaluated and their objective function values are

compared with their corresponding parent vectors. A greedy selection mechanism is employed in which vectors with the best objective values survive for the next generations. More details on the DE algorithm and its variants can be found in [32]. In this study, the DE algorithm is employed to enhance the exploiting capabilities of the VTSO algorithm. A discussion of the VTSO algorithm is presented in the subsequent sections, offering a detailed analysis of the proposed approach.

5. Voronoi Tessellations based Simple Optimizer

The Voronoi Tessellations-based Simple Optimizer (VTSO) follows a simple design principle that emphasizes on maintaining a healthy population diversity throughout the optimization procedure. In this section, first motivation for the proposed approach is discussed and then a comprehensive analysis on the working procedure of the proposed VTSO algorithm is presented.

5.1. Motivation

For successful development or implementation of the evolutionary algorithms, solution distribution plays a crucial role in determining its success. Properly distributed solutions enhance the chances of approximating position(s) of global optimal solution(s) or, locating basin(s) containing global optimal solution(s) [33]. This motivates us to integrate Voronoi Tessellations (VTs) in the population distribution of an evolutionary algorithm. VTs are important data structures in the field of computational geometry and are known for their exceptional distributive properties [17]. However, computing VTs for high dimensional spaces or, for a larger number of input seed points becomes very computationally expensive, contradicting the basic design principle of evolutionary algorithms. Therefore, a novel Voronoi Tessellations-based population distribution method that utilizes $2-$ dimensional Voronoi Tessellations to produce $D-$ dimensional solutions in the search space is proposed. This approach reduces the computational burden of calculating higher dimensional Voronoi Tessellations and placing solutions in well-distributed manner. The working procedure of the proposed VTSO algorithm is discussed in the following subsequent subsections.

5.2. Working of VTSO Algorithm

The VTSO algorithm follows a simple design principle. First, a population of N solutions referred to as seed population is randomly initialized in the given D-dimensional search space. After initializing this seed population, Voronoi vertices are calculated and combined with the seed population to form a sample population. This sample population is perturbated using Lévy flights to locate the potential

local regions of the search space (see sub-section 5.2.2). Then, these local regions are exploited using the DE algorithm till the population diameter becomes smaller than a fixed threshold (see, sub-section 5.2.4). This implies solutions in the population have come very close to each other and may result in premature convergence (or, stagnation) in the algorithm. For this purpose, a population-diameter based switch is proposed which activates itself and redistributes the seed population when it detects the above case. This process is continued till the termination criteria are satisfied. In the subsequent subsections, distinct phases of the proposed VTSO algorithm are discussed.

5.2.1. Voronoi Tessellations based Population Generation

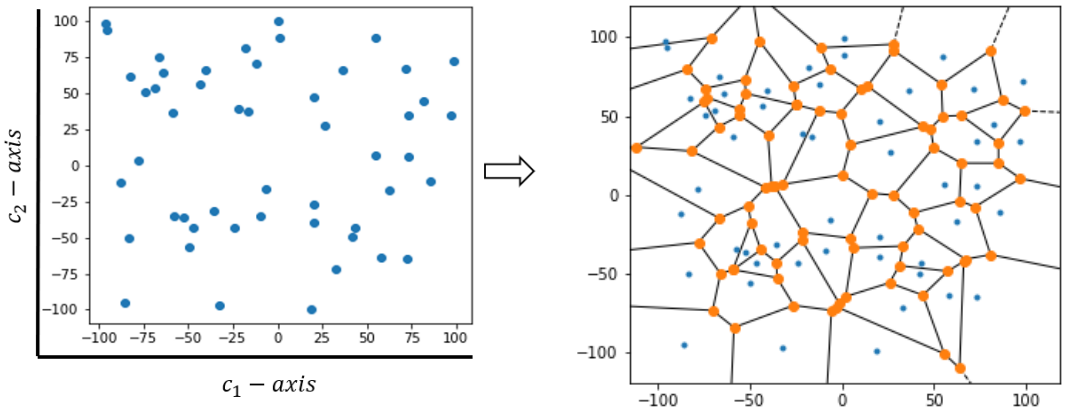
Suppose a population of N solutions is randomly initialized in a $D-$ dimensional search space. This creates a $N \times D$ data matrix for the population (see Figure 2). Row vectors $s_1, s_2, \cdots, s_N$ represent distinct solutions and column vectors $c_1, c_2, \cdots, c_D$ represent allele or locus sites of the solutions. The proposed Voronoi Tessellations based population distribution method incorporates a divide and conquer approach. First, a pairwise column selection is done to form $2-$ dimensional sites in a random order, and then Voronoi Tessellations are computed for these sites to produce $2-$ dimensional Voronoi vertices. The process of pairwise column selection is referred to as the slicing process, and it is performed till all the dimensions of the population data matrix are exhausted.

	c_1	c_2	$\cdots$	c_{D-1}	c_D
s_1	$x_{1,1}$	$x_{1,2}$	$\cdots$	$x_{1,D-1}$	$x_{1,D}$
s_2	x_{21}	x_{22}	$\cdots$	$x_{2,D-1}$	$x_{2,D}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
s_N	$x_{N,1}$	$x_{N,2}$	$\cdots$	$x_{N,D-1}$	$x_{N,D}$

Figure 2: A representative population data matrix

It is important to note that, the process of pair-wise column selection or slicing is dependent on the dimension D. The value of D can be even or odd. When D is even, all the dimensions of the population data matrix can be exhausted completely using pair-wise column selection. However, pair-wise column selection can not be performed when D is odd. In this case, an arbitrarily selected dimension is treated

as a unidirectional line, and pairwise column selection is performed on the remaining $D-1$ dimensions, which is similar to the case for even dimensions.



(a) Points lying in the $c_1 - c_2$ plane

(b) Corresponding Voronoi diagram

Figure 3: Voronoi vertices corresponding to the sites in $c_1 - c_2$ plane

For illustrating this process, suppose two column vectors $c_1 = (x_{1,1}, x_{2,1}, \cdots, x_{N,1})^t$ and $c_2 = (x_{2,1}, x_{2,2}, \cdots, x_{N,2})^t$ are selected using the pair-wise column selection mechanism. The points in the $c_1 - c_2$ plane can be considered as sites for computing $2-$dimensional Voronoi vertices (see Figure 3). This process is continued till all the dimensions of the population data matrix are exhausted. The generated Voronoi vertices are concatenated together to form a new population called the Voronoi population. The pair-wise concatenation mechanism is depicted in Figure 4.

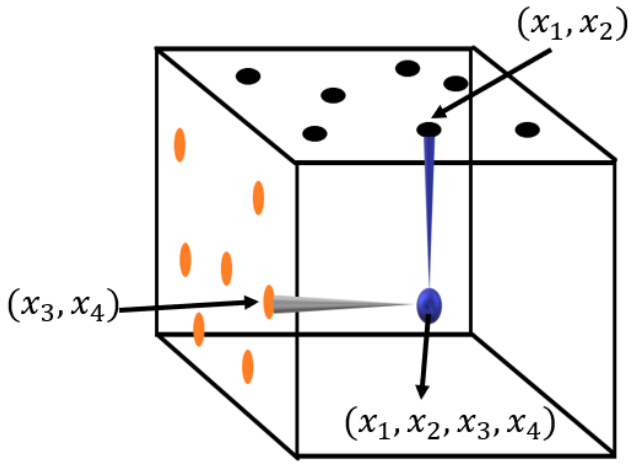


Figure 4: Pair-wise Vector Concatenation Mechanism

This slicing and concatenation-based approach helps the VTSO algorithm in efficiently populating the search space. The initial seed population and the Voronoi population are combined to create a sample population. It is interesting to note that, if the size of the initial population is N, then the upper bound for the number of Voronoi vertices is $2N - 5$ (see Section 3). This implies that the upper bound for the size of the sample population would be less than or equal to $3N - 5$. However, carrying a large population load throughout the optimization process increases the computational burden on the algorithm. Considering this factor, an elitist sampling mechanism (ESM) is proposed for filtering out the elite solutions from the sample population, and it is presented in the next subsection.

5.2.2. Elitist Sampling Mechanism (ESM)

Combining the Voronoi population with the seed population increases the size of the sample population and provides wide coverage in the search space. However, increasing the number of solutions also means an additional computational burden on the algorithm. Since carrying a large population load is not a recommended choice, an elite sampling mechanism is proposed. In this approach, an elite population is filtered out from the sample population. First, the positions of the solutions in the sample population are perturbated using Lévy flights derived using a high-tailed Lévy distribution. Mathematically, Lévy flights are characterized as random walks in which step lengths are drawn from the Lévy distribution using the simple power-

law formula $L(s) \sim |s|^{-1-\beta}$, where $0 < \beta < 2$. Lévy flights illustrate superior efficacy than Brownian random walks when exploring unfamiliar large-scale search spaces. This efficiency can be attributed to their higher variance. For instance, variance $\sigma^2(t) \sim t^{3-\beta}$, $1 \leq \beta \leq 2$ of Lévy flights rises at an accelerated rate when compared to the linear relationship (i.e, $\sigma^2(t) \sim t$) of Brownian random walks [34]. This ensures better coverage of unknown regions of the search space enhancing the exploration capabilities of the algorithm. Due to this reason, Lévy flights are used for providing an exploration boost to the sample population. The fitness values of the sample population and the perturbated population are evaluated, and a greedy selection mechanism is employed. Solutions with better fitness values are stored and sorted in ascending order according to their fitness value. The corresponding top N solutions are stored in elite samples and the rest of the $2N-5$ particles are discarded from the search space. The local regions around elite solutions can be identified as the promising regions of the search space and can be exploited further in order to approximate the global optimal solution(s). For this reason, the proposed VTSO algorithm uses the DE algorithm as a local search mechanism. The details of the proposed method are presented in the next subsection.

5.2.3. Local Search Mechanism

For exploiting regions around the elite particles, the Differential Evolution (DE) algorithm with different mutation operators is employed, and its effect on optimization performance of the proposed VTSO algorithm is studied. Five different mutation operators, namely, 'DE/current-to-pbest/ with archive (ST1)', 'DE/rand/1 (ST2)', 'DE/best/1 (ST3)', 'DE/rand/2 (ST4)', and 'DE/best/2 (ST5)' are considered for performance evaluation. Since, the mutation operators with two vector differences are more robust when compared to mutation operators with single vector difference [35]. The effect of all five mutation operators with single vector difference or two vector difference are studied. The incorporation of the DE algorithm as a local search technique helps the VTSO algorithm to effectively exploit the region around the positions of elite particles and enhances the robustness of the algorithm. However, at this stage, particles are prone to stuck in the local optimal regions causing premature convergence (or, stagnation) in the algorithm. To overcome this issue, a population diameter-based switch is proposed which gets activated when the population diameter (defined in the next subsection 5.2.4) reaches below a predefined threshold value. Implementation details of the proposed population diameter-based switch are given in the next subsection.

5.2.4. Population Diameter based Switch

Population diameter is the maximum distance between any two solutions in the population. The distance between two solutions is measured using the Euclidean metric. The diameter of the population is calculated using the equation (6) given below.

$$P_d = \max_{(i \neq j) \in [1,N]} \left(\sqrt{\sum_{k=1}^{D} (x_{i,k} - x_{j,k})^2} \right) \tag{6}$$

Here, P_d is the population diameter, N is the population size, and D is the dimension. The $x_{i,k}$ represents locus of the i^{th} solution at k^{th} dimension. The larger values of population diameter represent high particle dispersion, while smaller values indicate convergence [36]. However, a smaller value of population diameter might also indicate premature convergence or stagnation. That means that a diversity boost can be supplied to the population in case of small values of population diameter. For this purpose, a threshold value $D_T = 0.5$ is introduced in the proposed VTSO algorithm. After every $0.01 \cdot MAX_{nfes}$ or 1% of maximum function evaluations (MAX_{nfes}), population diameter S_d is calculated and compared with threshold value D_T. Suppose the value of S_d is less than D_T, then the population is reinitialized and the Voronoi population generation phase is invoked to redistribute the solutions in the search space. If the population diameter S_d is not below the threshold value D_T, the local search mechanism continues till the termination is satisfied. The detailed algorithmic steps of the proposed VTSO algorithm are given in the Algorithm 1.

6. Experiment Results

The performance of the proposed VTSO algorithm is tested on CEC14 and CEC17 benchmark test suits. These benchmark test sets contain different class of optimization problems. The objective functions of these problems are highly non-linear, non-differentiable, non-separable, and asymmetric [37] [38]. These problems pose difficulties of various levels and are well-suited for testing the performance of evolutionary algorithms. The optimization problems lying in these test beds are categorized into four types: unimodal, multi-modal, hybrid, and composite problems. For instance, in the CEC14 benchmark test suit, problems $F_1 - F_3$ are unimodal, $F_4 - F_{17}$ are multi-modal, $F_{18} - F_{22}$ are hybrid, and $F_{23} - F_{30}$ are composite [37]. Similarly, in the CEC17 benchmark test suit, problems $F_1 - F_3$ are unimodal, $F_4 - F_{11}$ are multi-modal, $F_{12} - F_{20}$ are hybrid, and $F_{21} - F_{30}$ are composite [38]. In this work, experiments are conducted on 30-dimensional and 50-dimensional problems.

To compare the performance of the VTSO algorithm, five standard meta-heuristic algorithms namely GA [5], PSO [7], DE [6], ABC [8], CMA-ES [9], and three DE variants, SaDE [39], jDE [40], and JADE [41] are considered. The setting of parameters is the same as mentioned in their original papers. The maximum function evaluation criteria MAX_{nfes} is set as $10^4 \times D$, D is the dimension of the underlying problems [37], [38]. Experiments are conducted on a personal computer with Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz, 16GB RAM with Windows 11 Operating System, and codes of the algorithms are implemented in MATLAB R2015a. For testing the statistical reliability of the experiments, as per the criterion given for CEC problems, 51 independent runs are conducted, and the mean and standard deviation of the errors of the underlying algorithms are reported for all the benchmark problems of CEC14 and CEC17 (See Table 3, Table 4, Table 5, Table 6 of the supplementary data file 8). The error is defined as the difference of $|F(x^*) - F(\hat{x})|$, where $F(x^*)$ is the true optimal value, $F(\hat{x})$ is the approximated optimal value, x^* is the true optimal solution, and $\hat{x}$ is the approximation of the optimal solution.

Algorithm 1 Pseudo Code for the VTSO algorithm

Input: Population size N, Dimension D, Search Bounds $[x_{min_j}, x_{max_j}]$, total function evaluations MAX_{nfes}, Threshold D_T, Objective Function $f(X)$.
Output: Global Optimal value $f(X_{Gbest})$, Global Optimal Solution X_{Gbest} .

1: Initialize Seed Population randomly within the search bounds
2: **while** $nfes < MAX_{nfes}$ **do** // nfes is the current function evaluation counter //
3: Apply Voronoi Population Generation
4: Merge Seed population and Voronoi population to form sample population
5: Apply Elitist Sampling Mechanism to create an elite sample population
6: Apply Local Search Mechanism on the elite sample population
7: **if** $nfes == 1\%\ of\ MAX_{nfes}\ and\ nfes < MAX_{nfes}$ **then**
8: Calculate population Diameter S_d
9: **if** $S_d < D_T$ **then**
10: Reinitialize seed population
11: Go to step 3
12: **end if**
13: Apply step 4 - step 6
14: **else**
15: Continue to step 6
16: **end if**
17: **end while**

6.1. Non-Parametric Tests

The performance of the VTSO algorithm is compared with other considered algorithms using the Wilcoxon signed rank test at 5% level of significance. The null hypothesis "H_0 : There is no significant difference between the VTSO algorithm and other underlying algorithms" is tested on the median values of the mean error of $30D$ and $50D$ benchmark problems. The corresponding p-values at the significance level 5%, and their effect size metric Cohens-d are reported in Table 1. The effect size metric Cohens-d estimates the magnitude of the difference between the means of two populations [42]. This allows to compare the impact of different treatments under study. The Cohens-d can be employed in four different situations: (a) simple group design, (b) two-group design, (c) single group-two repeated measures, (d) designs with baselines to compare. A detailed methodology for computing Cohen's-d values can be found in [42]. For our study, we have employed a two-group design for which Cohens-d is calculated using the following equation (7).

$$Cohens - d = \frac{M2 - M1}{S_p} \tag{7}$$

Where, $M1$, and $M2$ are mean values of group 1 and group 2, respectively. And, S_p is called pooled standard deviation and calculated using the equation (8).

$$S_p = \sqrt{\frac{(n_1 - 1) \cdot S_1^2 + (n_2 - 1) \cdot S_2^2}{(n_1 + n_2) - 2}} \tag{8}$$

Where, S_1 and S_2 are the standard deviation of the group 1 and group 2, respectively. And, n_1 and n_2 are a sample size of group 1 and group 2, respectively. The value of Cohens-d is subjective and depends on the experimental specifications, however, values near 0.2 are considered as 'merely statistical', and values between 0.2-0.5 are considered as 'subtle', and values above 0.5 are considered 'obvious' [42]. The proposed VTSO algorithm is considered as group 1 and other algorithms are treated as group 2 for calculating cohens-d values. These values are reported in Table 1. It can be observed from Table 1 that, VTSO obtained a Cohens-d value higher than 0.2 for all other compared algorithms on 30D and 50D CEC14 benchmark problems. Similarly, for CEC17 30D benchmark problems, the VTSO algorithm obtained a Cohens-d value higher than 0.2 when compared with all other underlying algorithms. For 50D CEC17 benchmark problems, the VTSO algorithm obtains a Cohens-d value higher than 0.2 for all other compared algorithms, except, the jDE algorithm for which the value of Cohens-d is near 0.2. Considering the corresponding significance values and effect size metric values, the performance of the proposed

Table 1: p-values at significance level 0.05 and effect size metric Cohens-d

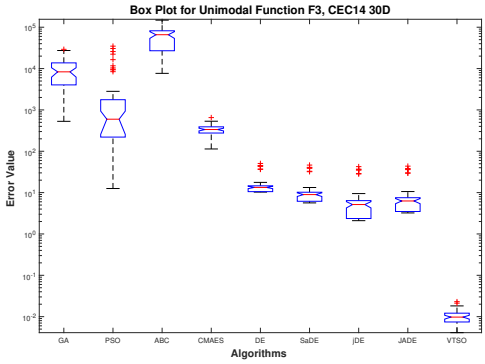
Benchmarks	CEC14				CEC17			
	30D		50D		30D		50D	
Algorithms	p-values	Cohens-d	p-values	Cohens-d	p-values	Cohens-d	p-values	Cohens-d
GA	0.00318	0.4349	0.00215	0.3805	0.00166	0.7900	0.00281	0.3336
PSO	0.00000383	0.3738	0.000132	0.3985	0.00015	0.5882	0.0261	0.2394
ABC	0.0000139	0.3453	0.000769	0.3665	0.00138	0.6278	0.00421	0.2598
CMA-ES	0.000484	0.3745	0.00436	0.2772	0.00871	0.3764	0.0103	0.5460
DE	0.000531	0.3684	0.00511	0.3367	0.00231	0.5698	0.00677	0.3083
SaDE	0.0317	0.3108	0.0713	0.3112	0.0319	0.3481	0.0661	0.0841
jDE	0.0188	0.2717	0.0599	0.3194	0.0237	0.2448	0.0689	0.1953
JADE	0.0533	0.2479	0.0611	0.2900	0.0241	0.3255	0.0459	0.2350

VTSO algorithm is superior when compared with the state-of-the-art evolutionary algorithms and some of their recent variants.

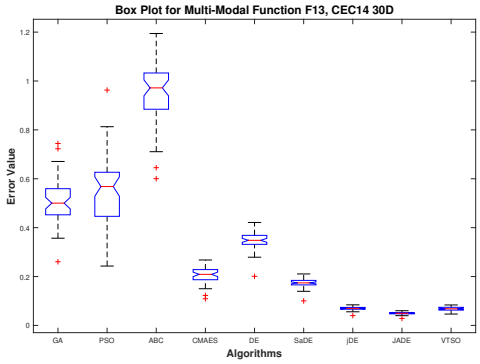
Further, for assessing the relative performance of the compared algorithms and determining the order of their success, the Friedman ranking test is employed to calculate the rankings of compared algorithms. It can be observed from Table 2 that the proposed VTSO algorithm outperforms all other underlying algorithms and secures the first rank. Further, to indicate the performance of the VTSO algorithm on individual benchmark problems, their corresponding ranks are reported in the result tables available in the supplementary data file 8. To visualize the results on different classes of benchmark problems, box plots for candidate unimodal, multi-modal, hybrid, and composite functions of CEC14 and CEC17 benchmarks are depicted in Figure 5, and the Figure 6, respectively.

Table 2: Average Ranks in Friedman Test

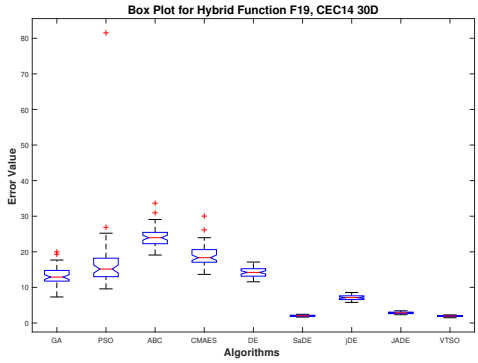
Algorithms	CEC14		CEC17		Avg	Final
	30D	50D	30D	50D	Rank	Rank
GA	5.2	5.7	5.66	5.93	5.62	6
PSO	7.56	7.13	6.86	5.56	6.48	8
ABC	7.5	7.56	7.7	7.93	7.16	9
CMA-ES	5.66	4.86	6.41	5.18	5.34	5
DE	6.3	3.86	6.43	7.16	6.39	7
SaDE	3.8	4.38	3.35	4.53	4.59	4
jDE	3.1	3.51	2.60	3.46	3.93	3
JADE	3.0	3.28	2.63	3.13	3.77	2
VTSO	2.8	2.11	2.16	2.08	3.16	1



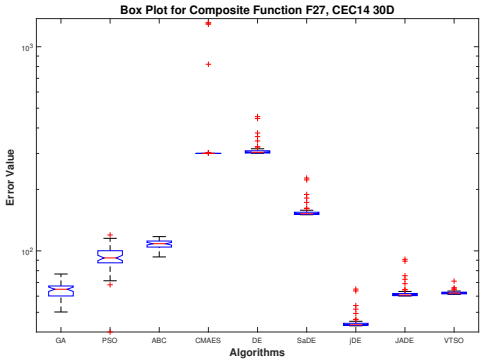
(a) Unimodal F_3



(b) Multi-Modal F_{13}



(c) Hybrid F_{19}

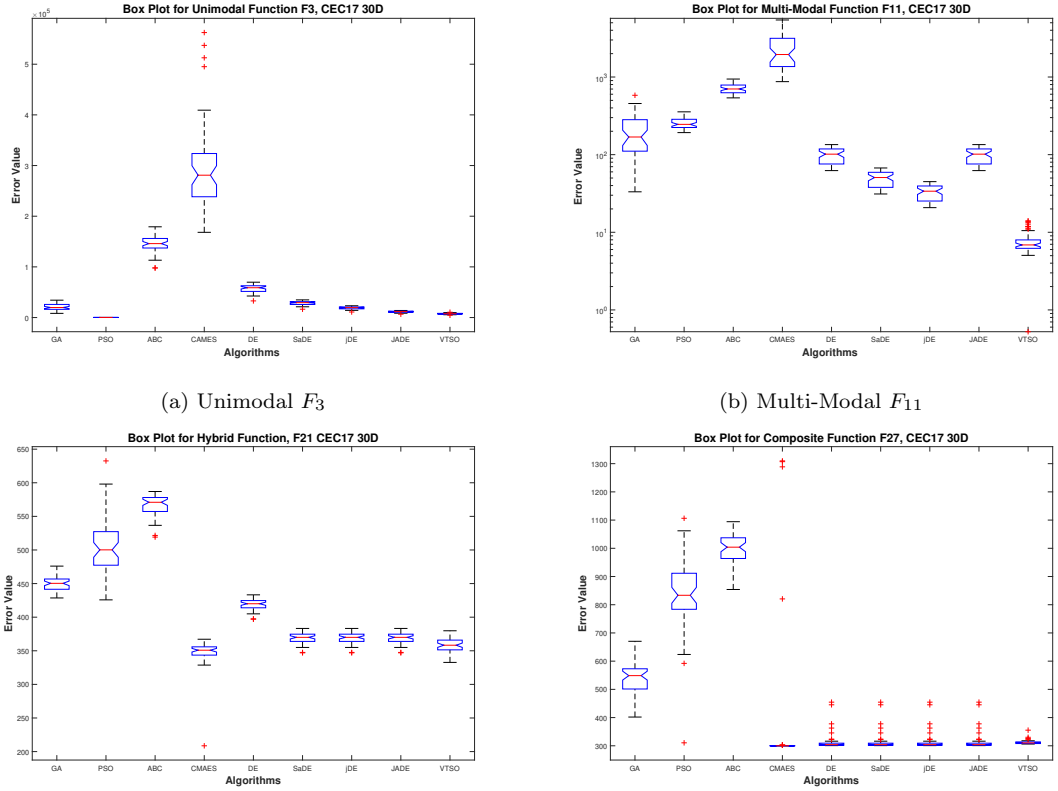


(d) Composite F_{27}

Figure 5: Box Plot Analysis for CEC14 Functions, 30D

6.2. Analysis of Population Diversity

Population diversity is one of the significant factors in determining the performance of any evolutionary algorithm [43]. The fundamental strength of the proposed



(a) Unimodal F_3

(b) Multi-Modal F_{11}

(c) Hybrid F_{19}

(d) Composite F_{27}

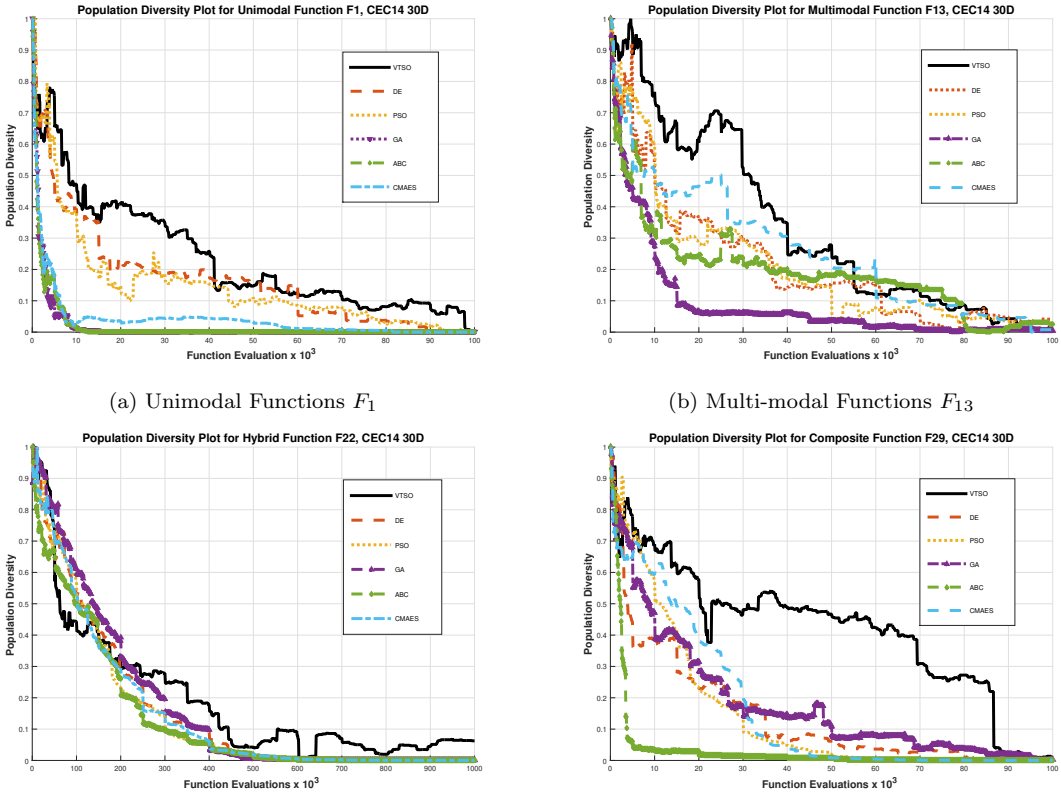
Figure 6: Box Plot Analysis for CEC17 Functions, 30D

VTSO algorithm is that it offers sufficient population diversity to reduce the shortcomings of premature convergence (or, stagnation) in the local optimal regions of the search space. The population diversity is calculated using the following equation (9) as mentioned in [43]. Here, $\bar{X}_j$ is the average position of the population in j^{th} dimension, and $Dist_{max}$ is the maximum distance calculated between each pair of solutions calculated using the equation (10) given below.

$$Population\ Diversity = \frac{1}{N \times Dist_{max}} \sum_{i=1}^{N} \sqrt{\sum_{j=1}^{D} \left(X_{i,j} - \bar{X}_j\right)^2} \tag{9}$$

$$Dist_{max} = \max_{(i \neq k) \in [1,N]} \left(\sqrt{\sum_{j=1}^{D} \left(x_{i,j} - x_{k,j}\right)^2} \right) \tag{10}$$

To examine the diversity managing capabilities of the proposed VTSO algorithm, population diversity is calculated for some representative functions, viz., unimodal functions F_1, multi-modal functions F_{13}, hybrid functions F_{22}, and composite functions F_{29} of CEC14 30 dimensional benchmark test suit and their population diversity plots are depicted in the Figure 7. It can be observed from Figure 7 that, the VTSO algorithm maintains a higher population diversity compared to other algorithms maintaining a well distributed solutions in the search space.



(a) Unimodal Functions F_1

(b) Multi-modal Functions F_{13}

(c) Hybrid Functions F_{22}

(d) Composite Function F_{29}

Figure 7: Population Diversity Plot, CEC14 30D Unimodal, Multi-modal, Hybrid, and Composite Functions

Further, to investigate the exploration vs exploitation trade-off of the proposed VTSO algorithm, a dimension-wise diversity-based analysis is incorporated [43]. The percentage of exploitation and exploration is calculated using the equation (11) given below.

$$
\begin{aligned}
Exploitation\% &= \left(\frac{\mathrm{Div} - \mathrm{Div}_{\max} |}{\mathrm{Div}_{\max}} \right) \times 100 \\
Exploration\% &= \left(\frac{\mathrm{Div}}{\mathrm{Div}_{\ \max}} \right) \times 100
\end{aligned} \tag{11}
$$

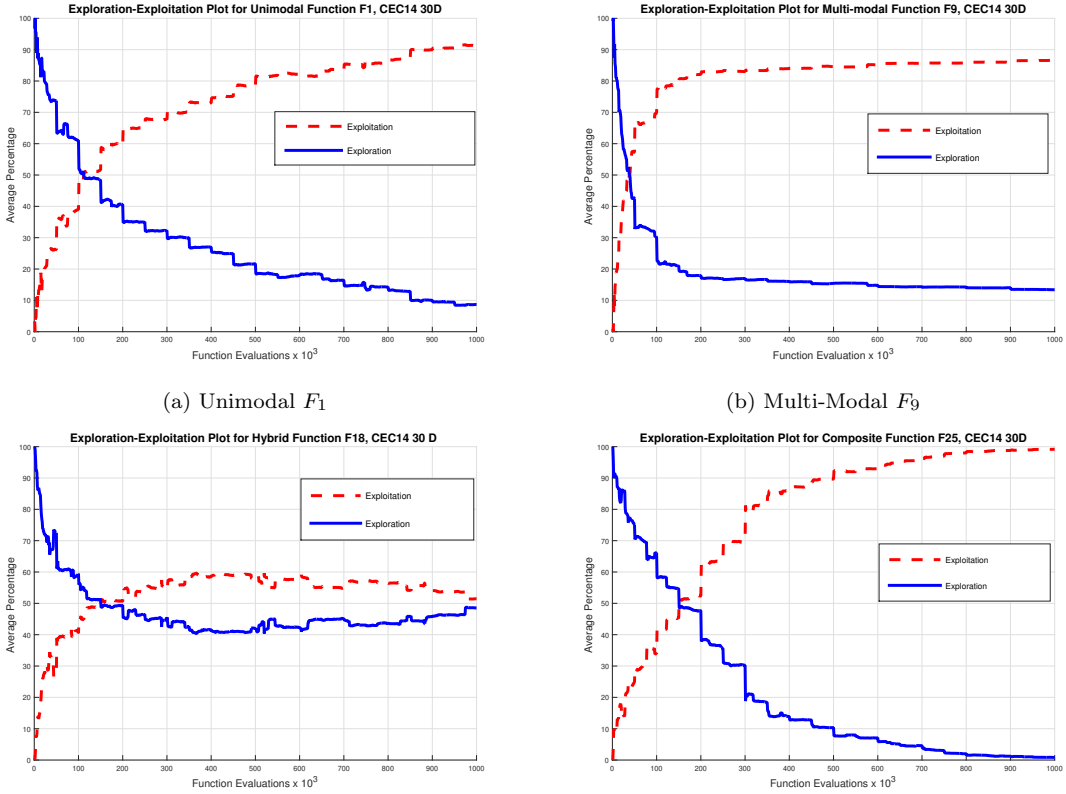
Where, the diversity Div is calculated using the equation (12) given below:

$$
\begin{aligned}
\mathrm{Div} &= \frac{1}{D} \sum_{j=1}^{D} Div_j \\
\mathrm{Div}_j &= \frac{1}{N} \sum_{i=1}^{N} |med_j(X) - x_{i,j}|
\end{aligned} \tag{12}
$$

Div_{max} is the maximum diversity recorded during the optimization process. Here, $med_j(X)$ denotes the median of the j^{th} dimension in the population. Exploration-exploitation plots for unimodal function F_1, multi-modal function F_9, hybrid function F_{18}, and composite function F_{25} of CEC14 30 dimensional benchmark test suits are presented in Figure 8. It can be observed from Figure 8 that, the proposed VTSO algorithm maintains a proper balance in the transition from the exploration phase to the exploitation phase.

6.3. Associated Parameters

The proposed VTSO algorithm incorporates five major control parameters, which are population size (N), scaling factor (F), crossover rate (CR), and population diameter switch threshold (D_T), and β introduced in the elite sampling mechanism (ESM). ESM utilizes Lévy flights whose step lengths are drawn from the power law formula $L(s) \sim |s|^{-1-\beta}$, where $0 < \beta < 2$. For the underlying experiments, the values of associated parameters are as follows, the size of the population (N) is taken to be 60, the value of the scaling factor F is taken to be 0.4, the value of CR is taken to be 0.5, the value of population diameter switch threshold (D_T) is taken to be 0.5, and the value of β is taken to be 1.7. These parameter values are suggested based on the empirical study following the multiple experiments on the CEC14 and CEC17 benchmark problems. In the next subsection, the computational complexity of the proposed algorithm is discussed.



(a) Unimodal F_1

(b) Multi-Modal F_9

(c) Hybrid F_{18}

(d) Composite F_{25}

Figure 8: Exploration-Exploitation Plot, CEC14 30D Unimodal, Multi-modal, Hybrid, and Composite Functions

6.4. Analysis of Computational Complexity

The computational complexity of all the compared algorithms is computed as per the criteria given in the technical reports of CEC14 and CEC17 benchmark problem [37],[38]. Table 3 reports the computational complexity of all compared algorithms on 30D and 50D CEC14 benchmark problems. T_0 is the computing time for evaluating the test problem given in [37].

T_1 represents the computing time needed for performing 200000 function evaluation of the problem $F18$. $\hat{T}_2$ is the average computational time observed by the algorithm across 5 independent runs to compute $F18$ with 200000 function evaluations as mentioned in the technical report of CEC14 benchmark problems, refer [37]. $(\hat{T}_2 - T_1)/T_0$ represents the performance of each algorithm in computing complexity on CEC14 benchmark problems. Following Table 3, the computational complexity of

Table 3: Computational complexity of the underlying algorithms

CEC14 Benchmarks Problems							
		30D			50D		
Algorithms	T_0	T_1	$\hat{T2}$	$(\hat{T2}-T1)/T0$	T_1	$\hat{T2}$	$(\hat{T2}-T1)/T0$
GA	0.09	1.05	3.79	30.44	1.34	4.17	31.44
PSO	0.09	1.05	2.59	17.11	1.34	3.90	28.44
DE	0.09	1.05	4.38	25.88	1.34	4.71	37.44
ABC	0.09	1.05	5.12	45.22	1.34	5.68	48.22
CMA-ES	0.09	1.05	9.21	90.66	1.34	14.02	140.88
SaDE	0.09	1.05	4.21	35.11	1.34	4.89	39.44
jDE	0.09	1.05	4.07	33.55	1.34	4.43	34.33
JADE	0.09	1.05	4.29	36.00	1.34	4.97	40.33
VTSO	0.09	1.05	4.72	40.77	1.34	5.07	42.55

Table 4: Computational complexity of the underlying algorithms

CEC17 Benchmarks Problems						
	30D			50D		
Algorithms	T_1	T_2	$(T_2-T1)/T_1$	T_1	T_2	$(T_2-T_1)/T_1$
GA	0.018	0.73	39.55	0.035	0.96	26.42
PSO	0.018	0.42	22.33	0.035	0.64	17.28
DE	0.018	0.49	26.22	0.035	0.98	27.00
ABC	0.018	1.02	55.66	0.035	1.46	40.42
CMA-ES	0.018	1.74	95.66	0.035	3.13	88.85
SaDE	0.018	0.53	28.44	0.035	1.13	31.28
jDE	0.018	0.51	27.33	0.035	1.07	29.57
JADE	0.018	0.57	30.66	0.035	1.13	31.28
VTSO	0.018	0.77	41.77	0.035	1.29	35.85

the VTSO algorithm is lower than ABC and CMA-ES algorithms, while greater than other compared algorithms. Similarly, Table 4 reports the computational complexity of all compared algorithms on CEC17 benchmark problems. T_1 is the average computing time for performing 10000 function evaluations for each problem in CEC17, and T_2 is the average computing time of an algorithm in performing 10000 function evaluations for each problem as mentioned in the technical report for CEC17 benchmark problems, refer [38]. $(T_2-T_1)/T_1$ represents the performance of each algorithm in computing complexity on CEC17 benchmark problems. It can be observed from Table 4, the computational complexity of the VTSO algorithm is lower than ABC and CMA-ES algorithms, while the complexity is greater than other compared algorithms. The reason for this is due to the time associated with computing Voronoi vertices in the optimization procedure. Further, for assessing the performance of the proposed VTSO algorithm on real-world problems, engineering design problems are solved and discussed in the next subsection.

Table 5: The performance of VTSO algorithm on constraint engineering problems

Algorithm	Metric	EP1	EP2	EP3	EP4
DE	Mean	1.76673	0.01281	6067.7532	2996.4712
	Std	0.01557	0.000085	8.9238	0.03329
SaDE	Mean	1.72697	**0.012688**	6059.7112	2996.4011
	Std	0.0000919	0.0000195	0.02664	0.017389
jDE	Mean	1.73175	0.012705	6060.0235	2996.6667
	Std	0.003335	0.0000309	0.2319	0.05602
JADE	Mean	1.72817	0.012684	6059.9193	**2996.3713**
	Std	0.001724	0.0000574	0.3211	0.00562
VTSO	Mean	**1.72518**	0.012749	**6059.6777**	2996.4119
	Std	0.0001743	0.0000691	0.00397	0.00000167

6.5. Engineering Design Problems

The performance of the proposed VTSO algorithm is also tested on the constraint engineering design problems. The details of these problems can be found in [44] [45]. In this work, four engineering design problems, namely, welded beam design (EP1), tension-compression spring design (EP2), pressure vessel design (EP3), and speed reducer design (EP4) are taken for experimentation. For each problem, 30,000 function evaluations are performed, and the mean and the standard deviation of the errors are reported in Table 5. VTSO algorithm achieves minimum mean error on the problems EP1 and EP3, while the performance is competitive with SaDE and JADE on problems EP2 and EP4, respectively.

7. Discussion

The experimental findings indicate that the VTSO algorithm demonstrates competitive performance compared to other state-of-the-art algorithms. The experimental results are reported in Table 3, Table 4, Table 5, and Table 6 of the supplementary data file 8. The comparative assessment of the VTSO algorithm with other underlying algorithms on CEC14 and CEC17 benchmark problems is discussed in the following subsections.

7.1. Effect of Different Mutation Strategies

In the local search mechanism of the proposed VTSO algorithm, the DE algorithm has been utilized to change the position of the elite particles in the search space. The performance of the DE algorithm is highly dependent on the underlying mutation strategies. Hence, to assess the impact of the different mutation strategies on the performance of the VTSO algorithm, five different mutation strategies are studied, which are 'DE/current-to-pbest/1 (ST1)', 'DE/rand/1 (ST2)', 'DE/best/1 (ST1)', 'DE/rand/2 (ST4)', and 'DE/best/2 (ST5)'. Here, ST is used as shorthand for

strategy. The experimental results on 30D CEC14, and CEC17 benchmark problems are reported in Table 1, and Table 2 of the supplementary data file 8, respectively. It is observed that strategies using 'pbest' and 'best' performed better compared to other strategies under consideration. Further, the performance of the VTSO algorithm with other algorithms is compared and discussed below in the following subsections.

7.2. Comparison on CEC14 Benchmark Problems

For CEC14 benchmark problems the performance of the proposed VTSO algorithm is compared with other underlying algorithms. For 30D and 50D problems, the VTSO algorithm secures an average rank of 2.8 and 2.11, respectively, in the Friedman test (see Table 2). For three unimodal functions $F1 - F3$, VTSO algorithm secures the lowest rank on two functions $F1$ and $F3$, for both 30D and 50D problems. For thirteen multi-modal functions $F4 - F17$, VTSO algorithm secures the lowest rank on seven functions ($F4, F5, F7$, and $F13 - F17$) for 30D problems, and on six functions $F6, F7, F13, F14, F16, F17$ for 50D problems. For five hybrid problems ($F18 - F22$), VTSO algorithm secures the lowest rank on three functions ($F19, F20, F22$) for 30D problems, and secures the lowest rank on four functions ($F18 - F21$) for 50D problems. Similarly, for the last seven composite functions ($F23 - F30$), VTSO algorithm secures the lowest rank on three functions ($F25, F27, F30$) for 30D problems, and secures the lowest rank on five functions ($F23 - F27$) for 50D problems. For the rest of the functions, the performance VTSO is similar or competitive to other algorithms (see, Table 3 and Table 4 of the supplementary data file 8). Following this, the performance of the VTSO algorithm on CEC17 benchmark problems is discussed in the next subsection.

7.3. Comparison on CEC17 Benchmark Problems

The experimental findings demonstrate the superior performance of the VTSO algorithm as it attains the lowest average friedman rank of 2.08 and 3.16 for 30D and 50D benchmark problems of the CEC17 benchmark test suit, respectively (see Table 2). For two unimodal functions $F1 - F3$, VTSO algorithm secures the lowest rank on the function $F3$ for both 30D and 50D problems. For thirteen multi-modal functions $F4 - F17$, VTSO algorithm secures the lowest rank on five functions ($F4, F11 - F15$) for 30D problems, and on seven functions $F4, F7, F11 - F15, F17$ for 50D problems. For five hybrid problems ($F18 - F22$), VTSO algorithm secures the lowest rank on three functions ($F18, F21, F22$) for 30D problems, and secures the lowest rank on three functions ($F18, F19, F21$) for 50D problems. Similarly, for the last seven composite functions ($F23 - F30$), VTSO algorithm secures the lowest rank

on five functions ($F23 - F28$) for 30D problems and secures the lowest rank on five functions ($F23, F25, F26, F28, F29$) for 50D problems. For the rest of the functions, the performance of VTSO is similar or competitive to other algorithms (see, Table 5 and Table 6 of the supplementary data file 8).

Experimental analysis indicates that the performance of the VTSO algorithm is competitive when compared with other state-of-the-art evolutionary algorithms and their variants. The VTSO algorithm secures the lowest average Friedman ranking score of 3.16, hence obtaining a first rank in the overall ranking (see Table 2). The proposed algorithm demonstrates balanced exploration-exploitation capabilities, along with maintaining a healthy population diversity. Accounting for these factors, the VTSO algorithm can be considered a good candidate for solving intricate optimization problems. However, there are some limitations to the implementation of the VTSO algorithm which are discussed in the next subsection.

7.4. Limitations

The inherent limitation of the VTSO algorithm lies in the calculation of the Voronoi Tessellations (VTs) of the underlying search space. Calculating VTs for points distributed in the search space might cause difficulties if the distributed points are very close to each other, and in extreme cases are co-linear to each other. For instance, when the points are distributed on a straight line or very close[1] to each other, computing voronoi diagram results in parallel lines, making the number of voronoi vertices infinite and resulting a case of degeneracy, a limitation of the quickhull algorithm [46]. This occurs with the working procedure of the VTSO algorithm, when the size of the seed population is large and computing voronoi vertices in this case may produce degeneracy and can hamper the optimization procedure. The other limitations can be attributed to the involvement of the algorithmic parameters which is a common challenge associated with all evolutionary algorithms. These parameters need to be tuned optimally and should not cause high computational complexity to the algorithm. However, despite having these limitations, the concept of generating well-distributed solutions using the concept of Voronoi Tessellations offers a novel approach to enrich the robustness of the evolutionary optimizers.

8. Conclusion and Future Research Directions

The proposed VTSO algorithm is an effective optimizer based on the principles of Voronoi Tessellations and Differential Evolution. It uses a novel solution distribution

[1]Closeness here is subjective, depending on the precision of the underlying computing machine.

method that relies on the concept of Voronoi Tessellations. The distributive properties of Voronoi Tessellations, help the VTSO algorithm in exploring the search space in a more efficient manner, while employing the DE algorithm as a local search mechanism assists the VTSO algorithm in exploiting the potential regions of the search space. For handling the issue of premature convergence or stagnation, the population diameter-based switch activates itself when solutions come nearer to each other, and the population loses its diversity. Experimental findings support the reliability of the VTSO algorithm in locating regions around the position of true optimal solution(s).

However, the inherent limitations require more attention and need to be analyzed carefully. For instance, the deterministic methods for computing Voronoi Tessellations in higher dimensions are computationally expensive, which makes these methods contradictory to the basic design principle of evolutionary algorithms. In this work, a slicing and concatenating-based approach is proposed to utilize lower dimensional Voronoi Tessellations, however slicing and concatenating of higher dimensional spaces in an optimal manner require careful analysis. The associated parameters, like population diameter threshold, can be tuned using adaptive techniques which would be a matter of future research.

Supplementary Data

The performance results in terms of *mean, std* and *best* for 30D and 50D problems of CEC14 and CEC17 benchmark sets are available in Mendeley Data Repository: Bajpai, Prathu; Bansal , Jagdish (2024), "dataset-Voronoi Tessellations based Simple Optimizer", Mendeley Data, V1, doi: 10.17632/cvvyytydsj.1.

Declaration of Competing Interest

The authors declare that they have no competing financial or personal interests that could influence the present work.

Declaration of generative AI and AI-assisted technologies in the writing process

During the preparation of this work, the author(s) have not used any generative AI or AI-assisted technologies.

Acknowledgement

The authors acknowledge the support of South Asian University in preparing this research work. Both Authors acknowledge the support of Alvin Alexander of Curtin University, Australia, for improving the linguistic quality of the manuscript. In addition, author one acknowledges the financial support under the JRF/SRF scheme of the University Grant Commission (UGC), India.

References

[1] R. Espinosa, F. Jiménez, J. Palma, Multi-surrogate assisted multi-objective evolutionary algorithms for feature selection in regression and classification problems with time series data, Information Sciences 622 (2023) 1064–1091.

[2] N. Li, L. Ma, G. Yu, B. Xue, M. Zhang, Y. Jin, Survey on evolutionary deep learning: Principles, algorithms, applications, and open issues, ACM Computing Surveys 56 (2) (2023) 1–34.

[3] J. Huizinga, J. Clune, Evolving multimodal robot behavior via many stepping stones with the combinatorial multiobjective evolutionary algorithm, Evolutionary Computation 30 (2) (2022) 131–164.

[4] Z.-H. Zhan, L. Shi, K. C. Tan, J. Zhang, A survey on evolutionary computation for complex continuous optimization, Artificial Intelligence Review (2022) 1–52.

[5] J. H. Holland, Genetic algorithm, Scientific American 267 (1992) 66–73.

[6] R. Storn, K. Price, A simple and efficient heuristic for global optimization over continuous spaces, Journal of Global Optimization 11 (1997) 341–359.

[7] J. Kennedy, R. Eberhart, Particle swarm optimization, Proceedings of IEEE International Conference on Neural Networks 4 (1995) 1942–1948.

[8] D. Karaboga, B. Akay, A comparative study of artificial bee colony algorithm, applied mathematics and computation, Applied Mathematics and Computation 214 (2009) 108–132.

[9] N. Hansen, A. Ostermeier, Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation, in: Proceedings of IEEE international conference on evolutionary computation, IEEE, 1996, pp. 312–317.

[10] Q. Li, S.-Y. Liu, X.-S. Yang, Influence of initialization on the performance of metaheuristic optimizers, Applied Soft Computing 91 (2020) 106193.

[11] L. Swiler, R. Slepoy, A. Giunta, Evaluation of sampling methods in constructing response surface approximations, in: 47th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference 14th AIAA/ASME/AHS Adaptive Structures Conference 7th, 2006, p. 1827.

[12] O. M. Shir, T. Bäck, Niching in evolution strategies, in: Proceedings of the 7th annual conference on Genetic and evolutionary computation, 2005, pp. 915–916.

[13] D.-C. Dang, T. Friedrich, T. Kötzing, M. S. Krejca, P. K. Lehre, P. S. Oliveto, D. Sudholt, A. M. Sutton, Escaping local optima with diversity mechanisms and crossover, in: Proceedings of the Genetic and Evolutionary Computation Conference 2016, 2016, pp. 645–652.

[14] D. Sudholt, The benefits of population diversity in evolutionary algorithms: a survey of rigorous runtime analyses, Theory of evolutionary computation: Recent developments in discrete optimization (2020) 359–404.

[15] d. B. Mark, C. Otfried, v. K. Marc, O. Mark, Computational geometry algorithms and applications, Springer (2008).

[16] R. A. Apu, M. L. Gavrilova, Efficient swarm neighborhood management using the layered delaunay triangulation, Generalized Voronoi Diagram: A Geometry-Based Approach to Computational Intelligence (2008) 109–129.

[17] M. Pedergnana, S. G. García, et al., Smart sampling and incremental function learning for very large high dimensional data, Neural Networks 78 (2016) 75–87.

[18] W. Chi, Z. Ding, J. Wang, G. Chen, L. Sun, A generalized voronoi diagram-based efficient heuristic path planning method for rrts in mobile robots, IEEE Transactions on Industrial Electronics 69 (5) (2021) 4926–4937.

[19] W. Tu, Z. Fang, Q. Li, S.-L. Shaw, B. Chen, A bi-level voronoi diagram-based metaheuristic for a large-scale multi-depot vehicle routing problem, Transportation Research Part E: Logistics and Transportation Review 61 (2014) 84–97.

[20] M. Shatnawi, M. F. Nasrudin, Starting configuration of cuckoo search algorithm using centroidal voronoi tessellations, in: 2011 11th International Conference on Hybrid Intelligent Systems (HIS), IEEE, 2011, pp. 40–45.

[21] Y.-H. Zhang, Y.-J. Gong, Y. Gao, H. Wang, J. Zhang, Parameter-free voronoi neighborhood for evolutionary multimodal optimization, IEEE Transactions on Evolutionary Computation 24 (2) (2019) 335–349.

[22] H. Tong, C. Huang, J. Liu, X. Yao, Voronoi-based efficient surrogate-assisted evolutionary algorithm for very expensive problems, in: 2019 IEEE Congress on Evolutionary Computation (CEC), IEEE, 2019, pp. 1996–2003.

[23] T. Okabe, Y. Jin, B. Sendoff, M. Olhofer, Voronoi-based estimation of distribution algorithm for multi-objective optimization, in: Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No. 04TH8753), Vol. 2, IEEE, 2004, pp. 1594–1601.

[24] T. Huang, W. Gao, H. Li, J. Xie, A voronoi neighborhood based differential evolution algorithm for multimodal multi-objective optimization, in: 2021 IEEE 7th International Conference on Cloud Computing and Intelligent Systems (CCIS), IEEE, 2021, pp. 128–133.

[25] W. Pan, N. Lv, K. Chen, Y. Pan, X. Hong, X. Zhang, Multi-uav relay deployment algorithm based on voronoi diagram division, in: 2023 6th International Conference on Electronics Technology (ICET), IEEE, 2023, pp. 624–628.

[26] Q. Du, V. Faber, M. Gunzburger, Centroidal voronoi tessellations: Applications and algorithms, SIAM review 41 (4) (1999) 637–676.

[27] R. Klein, Concrete and abstract voronoi diagrams, Springer Science and Business Media 400 (1989).

[28] H. Hiyoshi, Intelligent solutions for curve reconstruction problem, Generalized Voronoi Diagram: A Geometry-Based Approach to Computational Intelligence (2008) 131–158.

[29] K. Mulmuley, Computational geometry. an introduction through randomized algorithms, Prentice Hall Inc. (1994).

[30] M. De Berg, O. Cheong, M. Van Kreveld, M. Overmars, Computational geometry: Algorithms and applications. springer-verlag berlin heidelberg (2008).

[31] K. Price, R. M. Storn, J. A. Lampinen, Differential evolution: a practical approach to global optimization, Springer Science and Business Media (2006).

[32] M. Pant, H. Zaheer, L. Garcia-Hernandez, A. Abraham, et al., Differential evolution: A review of more than two decades of research, Engineering Applications of Artificial Intelligence 90 (2020) 103479.

[33] B. Kazimipour, X. Li, A. K. Qin, A review of population initialization techniques for evolutionary algorithms, in: 2014 IEEE congress on evolutionary computation (CEC), IEEE, 2014, pp. 2585–2592.

[34] X.-S. Yang, Nature-inspired optimization algorithms, Academic Press (2020).

[35] R. Gämperle, S. D. Müller, P. Koumoutsakos, A parameter study for differential evolution, Advances in intelligent systems, fuzzy systems, evolutionary computation 10 (10) (2002) 293–298.

[36] O. Olorunda, A. P. Engelbrecht, Measuring exploration/exploitation in particle swarms using swarm diversity, in: 2008 IEEE congress on evolutionary computation (IEEE world congress on computational intelligence), IEEE, 2008, pp. 1128–1134.

[37] J. J. Liang, B. Y. Qu, P. N. Suganthan, Problem definitions and evaluation criteria for the cec 2014 special session and competition on single objective real-parameter numerical optimization, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Technical Report, Nanyang Technological University, Singapore 635 (2) (2013).

[38] G. Wu, R. Mallipeddi, P. N. Suganthan, Problem definitions and evaluation criteria for the cec 2017 competition on constrained real-parameter optimization, National University of Defense Technology, Changsha, Hunan, PR China and Kyungpook National University, Daegu, South Korea and Nanyang Technological University, Singapore, Technical Report (2017).

[39] A. K. Qin, P. N. Suganthan, Self-adaptive differential evolution algorithm for numerical optimization, in: 2005 IEEE congress on evolutionary computation, Vol. 2, IEEE, 2005, pp. 1785–1791.

[40] J. Brest, S. Greiner, B. Boskovic, M. Mernik, V. Zumer, Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems, IEEE transactions on evolutionary computation 10 (6) (2006) 646–657.

[41] J. Zhang, A. C. Sanderson, Jade: adaptive differential evolution with optional external archive, IEEE Transactions on evolutionary computation 13 (5) (2009) 945–958.

[42] J.-C. Goulet-Pelletier, D. Cousineau, A review of effect sizes and their confidence intervals, part i: The cohen'sd family, The Quantitative Methods for Psychology 14 (4) (2018) 242–265.

[43] B. Morales-Castañeda, D. Zaldivar, E. Cuevas, F. Fausto, A. Rodríguez, A better balance in metaheuristic algorithms: Does it exist?, Swarm and Evolutionary Computation 54 (2020) 100671.

[44] A. H. Gandomi, X.-S. Yang, A. H. Alavi, Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems, Engineering with computers 29 (2013) 17–35.

[45] L. C. Cagnina, S. C. Esquivel, C. A. C. Coello, Solving engineering optimization problems with the simple constrained particle swarm optimizer, Informatica 32 (3) (2008).

[46] C. B. Barber, D. P. Dobkin, H. Huhdanpaa, The quickhull algorithm for convex hulls, ACM Transactions on Mathematical Software (TOMS) 22 (4) (1996) 469–483.