0.1 Enhanced Opposition Differential Evolution Algorithm for Multimodal Optimization

INDIAN INSTITUTE OF TECHNOLOGY INDORE

CANDIDATE'S DECLARATION

I hereby certify that the work which is being presented in the thesis entitled **Enhanced Opposition Differential Evolution Algorithm for Multimodal Optimization** in the fulfillment of the requirements for the award of the degree of **Master of Science (Research)** and submitted in the **Department of Computer Science and Engineering, Indian Institute of Technology Indore**, is an authentic record of my own work carried out during the time period from July 2019 to August 2021 under the supervision of Dr. Aruna Tiwari, Associate Professor, Indian Institute of Technology Indore, Indore, India.

The matter presented in this thesis has not been submitted by me for the award of any other degree of this or any other institute.

Shatendra Singh
05/08/2021

Signature of the Student with date

(Shatendra Singh)

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Signature of the Thesis Supervisor with date

(Dr. Aruna Tiwari)

Shatendra Singh has successfully given her MS (Research) Oral Examination held on 05/11/2021.

Signature of Chairperson (OEB) 5/11/2021
Date:

Signature of Thesis Supervisor
Date:

Bodhisatwa Mazumdar

Signature of Convener DPGC
Date: 5/11/2021

Somnath Dey

Signature of Head of Department
Date: 05/11/2021

ACKNOWLEDGEMENTS

I would like to take this opportunity to express my gratitude to people who in one or the other way contributed by making this time as learnable, enjoyable, and bearable. At first, I would like to thank my supervisor **Dr. Aruna Tiwari**, who was a constant source of inspiration during my work. With her constant guidance and research directions, this research work has been completed. Her continuous support and encouragement has motivated me to remain streamlined in my research work.

I am thankful to **Mrs. Suchitra Agrawal**, PhD student for all her help and support. I am also grateful to **Dr. Somnath Dey**, HOD of Computer Science for all his help and support.

I am thankful to **Dr. Amod Umarikar** and **Dr. Kapil Ahuja**, my research progress committee members for taking out some valuable time to evaluate my progress all these years. Their valuable comments and suggestions helped me to improve my work at various stages.

My sincere acknowledgement and respect to **Prof. Neelesh Kumar Jain**, Director, Indian Institute of Technology Indore for providing me the opportunity to explore my research capabilities at Indian Institute of Technology Indore.

Lastly, I would like to thank my family and friends for all their love and support.

Shatendra Singh

To you as a reader

ABSTRACT

Most of the real-world problems are multimodal in nature that consists of multiple optimum values. Multimodal optimization amounts to finding multiple global and local optima (as opposed to a single solution) of a function so that the user can have a better knowledge about different optimal solutions in the search space and as and when needed, the current solution may be switched to another suitable one while still maintaining the optimal system performance. Classical gradient-based methods fail for optimization problems in which the objective functions are either discontinuous or non-differentiable. Evolutionary Algorithms (EAs), due to their population-based approaches, can detect multiple solutions within a population in a single simulation run and have a clear advantage over the classical optimization techniques, which need multiple restarts and multiple runs in the hope that a different solution may be discovered every run, with no guarantee. Hence, several EAs have been proposed to solve such kinds of problems. However, Differential Evolution (DE) algorithm is a population-based heuristic method that can solve such optimization problems, and it is simple to implement. The potential challenge in Multi-Modal Optimization Problems (MMOPs) is to search the function space efficiently to locate most of the peaks accurately. The optimization problem could be to minimize or maximize a given objective function and we aim to solve the maximization problems on multimodal functions in this study. Hence, we have proposed two algorithms known as Enhanced Opposition Differential Evolution (EODE) algorithm and Estimation of Distribution Algorithms based Differential Evolution (EDADE) algorithm to solve the MMOPs. The proposed algorithms have been tested on IEEE Congress on Evolutionary Computation (CEC) 2013 benchmark functions, and they achieve competitive results compared to the existing state-of-the-art approaches.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Evolutionary algorithms (EAs) have been widely used for solving optimization problems having a single global optimum. It is wise to find as many optimum points as possible for several reasons. First, an optimal solution currently favorable (say, due to availability of some critical resources, or others) may not remain to be so in the future. This would then demand the user to operate at a different solution when such a predicament occurs. With the knowledge of another optimal solution for the problem which is favorable to the changed scenario, the user can simply switch to this new optimal solution. Second, the sheer knowledge of multiple optimal solutions in the search space may provide useful insights into the properties of optimal solutions of the problem. Many real-world problems consist of multiple optima, such as holographic design [1], electro-magnetic design [2], protein structure prediction [3], and data mining [4]. Therefore, it is essential to find as many global optimizers as possible as it gives users the flexibility to choose alternate solutions if one solution cannot be chosen due to limited resource constraints. However, it is more challenging to find multiple global optima than to find a single global optimum. If a point-by-point classical optimization approach is used for multimodal optimization, the approach must have to be applied several times, every time with the expectation of finding a different optimal solution. Many researchers have used EAs [5] and swarm intelligence algorithms [6] to solve Multi-Modal Optimization Problems (MMOPs), such as the Genetic Algorithm (GA) [49], Ant Colony Optimization (ACO) [8], Estimation of

Distribution Algorithm (EDA) [9], Particle Swarm Optimization (PSO) [10], and Differential Evolution (DE) [11]. However, they tend to lose the effective balance between exploration and exploitation of the search space. "Exploration" here means increasing the diversity of the population in the hope to find better individuals while the term "Exploitation" here means refining the existing candidate solutions to reach the best one in a local region(sub-population). While solving the exploration vs exploitation dilemma, two major problems arise. First, if diversity is increased too much then it may lead to unnecessary fitness computations as there may be no optima in those areas of search space where the search is being performed. Second, if exploitation is performed in the undesired region of the search space, then it may get trapped in local optima. Therefore, a different mechanism, based on classical EAs, is required to locate multiple optima simultaneously. Niching [27] has been widely used in the literature to help an EA maintain population diversity in multimodal optimization. Some well-known niching techniques include crowding [12], clearing [13], fitness sharing [14], and speciation [15]. Niching divides the population into several sub-populations and each subpopulation is responsible to find the optima within that sub-population. Each sub-population can be considered as a species. Since the fitness landscape may be uneven and complex, the niches could be of different shapes and sizes. Therefore, it becomes a great challenge to form multiple sub-populations out of the global population in such a way that a maximum number of peaks are located by the sub-populations. Classical niching methods are sensitive to the parameters they use; for example, crowding is sensitive to the crowding size, and speciation is sensitive to the niching radius. Hence, parameter-free or parameter-insensitive techniques have been developed to improve niching, such as history-based topological speciation [31], and clustering [32]. Other adaptive learning strategies have been developed to enhance the diversity for EAs, such as GA [33], DE [35], and PSO [36].

1.1 Motivation

Since the complex multimodal landscapes consist of niches of different shapes and sizes, it becomes imperative to locate them. The existing methods are not efficient in locating most of the niches due to the improper segregation and balancing of the species. Therefore, there is a great need to design an efficient niching method that can form niches of uneven sizes and shapes. Also, assigning the static values to the parameters of the algorithm inhibits the search and convergence capabilities of the functional landscape. Low parameters value can lead to premature convergence and smaller exploration while high parameters value can lead to excessive perturbation and divergence from the optima which may lead to wastage of fitness evaluations. Hence, there is a need for an adaptive parameter control strategy that learns to adapt to the niche landscape in order to drive the population within the niche towards convergence (optima). To this aim, we have proposed an Enhanced Opposition Differential Evolution (EODE) algorithm to solve the above problems. ODE [48] helps in speeding up the convergence towards optima. It also helps in creating offsprings that enable efficient exploration of the sub-population as the current candidate and the opposite candidate squishes the function space from both ends in all dimensions. Since the opposite population does not have a strong enough capability to look up the search space out of the current species landscape, ODE has limited exploration capability. Hence, a hybrid DE method is proposed based on the Estimation of Distribution Algorithm (EDA) that produces diverse offsprings for better exploration.

1.2 Objectives

In this section we are going to cover the following objectives in our thesis:

1. To study and analyze the Differential Evolution (DE) algorithm for solving MMOPs.

2. To propose a method that divides the global population into local sub-populations efficiently such that each sub-population corresponds to a niche and

most of the niches are located.

3. To develop a better balancing technique for the newly created sub-populations in order to help enhance convergence speed towards optima within the sub-population.

4. To propose a method that performs an efficient local search within each sub-population to refine the candidate solutions.

5. To develop an adaptive technique to learn the parameters mutation factor (F) and crossover rate (CR) over generations.

6. To propose a hybrid approach based on Estimation Distribution Algorithm (EDA) and DE for solving MMOPs.

In this thesis, we propose algorithms for locating multiple global peaks for multimodal problems using Enhanced Opposition Differential Evolution (EODE) algorithm. Also, we have proposed a probabilistic version of DE which is based on EDA(EDADE) to solve MMOPs. Our proposed methods are adaptable to different functional landscapes and hence provide better results for multiple functions.

1.3 Thesis Contribution

In this thesis, we describe new algorithms for finding multiple optima of several multimodal problems. The contribution of our thesis are as follows:

I. Enhanced Opposition Differential Evolution (EODE) Algorithm This method aims to create a variant of standard Opposition Differential Evolution(ODE) [48] algorithm which is adaptive as it learns the parameters such as mutation factor(F) and crossover rate(CR) over the generations based on SHADE [41]. Two-level niching based on mNBC [39] is applied to locate a maximum number of near-peak regions in the function space. It also saves unnecessary resource (function evaluation) wastage. Existing methods have not used the concept of opposition-based learning to solve the multimodal optimization problems and rather they rely on

differential perturbation to create the next generation population. However, these methods lose effectiveness in locating a maximum number of peaks within the given cost. Thus as our first contribution, we propose a method to improve the effectiveness in locating a maximum number of peaks using adaptive guided search over the generations. The differential perturbation takes place dynamically depending upon the weightage of exploration and exploitation at that specific configuration of the evolution process. We have also proposed a species balance strategy that uses probability distribution methods to generate more candidates for the DE operations to take place effectively. We have proposed a local search method to improve the fitness value of the already found best member of a sub-population. We have also proposed a scheme to avoid the insertion of multiple peaks belonging to the same sub-population in the archive which essentially enables the algorithm capability to locate multiple different peaks.

II. Estimation of Distribution Algorithm based Differential Evolution(EDADE) Algorithm

To be able to locate multiple species in a global population, we have proposed a modified version of two-level speciation that is based on mNBC [39]. Also, a modified species balance strategy is proposed that is based on Gaussian distribution for balancing the oversized and undersized species. As discussed in [48], ODE is effective in the exploitation of the search space, it lacks better exploration. To be able to locate as many optima as possible, the method should explore the entire search space effectively which could be helpful for problems with a large number of optima. To include randomness and better exploration, a probabilistic approach has been introduced. The distribution estimation helps generate better offsprings by predicting the distribution parameters of the niches. Gaussian and Cauchy distributions are used to generate potentially better offspring. DE has a faster convergence rate as compared to the probabilistic approaches due to its vector based mutation operations. Hence, to this end, we introduced a hybrid method that is

based on DE and EDA to evolve the population effectively. We also proposed a probabilistic local search method to improve the already found candidate solutions. The distribution estimation helps the algorithm guide to produce offsprings of diverse nature that enables better exploration of the search space. Also, a simple strategy is introduced to avoid the population trap at local optima which could prevent wastage of significant fitness evaluations and help the algorithm locate global optima.

Thus, to assess the effectiveness of the above algorithms, we apply them to multimodal problems having dimensions upto 20. We compare the results of our proposed algorithms with that of several state-of-the-art algorithms.

1.4 Organization of Thesis

We organize the rest of this thesis as follows:

Chapter 2: In this chapter we discuss the related work on multimodal optimization. Overview of Non-DE approaches for solving MMOPs are discussed. Detailed survey of DE based approaches are covered in this chapter.

Chapter 3: In this chapter we describe our proposed algorithm EODE for multimodal optimization. We apply the proposed algorithm on IEEE CEC 2013 benchmark suite described in Appendix A. We present the results and comparison with the state-of-the-art algorithms.

Chapter 4: In this chapter we describe our proposed algorithm EDADE for multimodal optimization. We apply the proposed algorithm on IEEE CEC 2013 benchmark suite described in Appendix A. We present the results and comparison with the state-of-the-art algorithms.

Chapter 5: In this chapter we discuss the results obtained using EODE and EDADE. We also conclude the study presented in the thesis and provide directions for future work in this area.

Chapter 2

Literature Survey

Multimodal optimization is defined as a problem of finding multiple global and local optima (as opposed to a single solution) of a function so that the user can have a better knowledge about different optimal solutions in the search space and as and when needed, the current solution may be switched to another suitable one while still maintaining the optimal system performance. EAs, due to their population-based approaches, can detect multiple solutions within a population in a single simulation run and have a clear advantage over the classical optimization techniques, which need multiple restarts and multiple runs in the hope that a different solution may be discovered every run, with no guarantee, however. Numerous evolutionary optimization techniques have been developed since the late 1970s for locating multiple optima (global or local). These techniques are commonly referred to as "niching" methods. Niching [27] can be incorporated into a standard EA to promote and maintain the formation of multiple stable subpopulations within a single population, to locate multiple globally optimal or suboptimal solutions simultaneously. Niching is the technique of finding and preserving multiple stable niches, or favorable parts of the solution space possibly around multiple solutions, to prevent convergence to a single solution.

2.1 Niching Methods

Primarily niching techniques have been developed to reduce the effect of genetic drift resulting from the selection operator of the classical GAs. Thus they also aim at maintaining genetic diversity in the population and making GAs able to find multiple optima in parallel. A niching method must have to form and maintain multiple and diverse final solutions within the search and it should also be able to maintain these multiple solutions for a large enough number of iterations. In what follows, we discuss some of the most prominent niching techniques available in the existing literature:

2.1.1 Clearing

Clearing [13] removes the bad individuals and keeps only the best individual (or a few top individuals) within each niche. The algorithm first sorts the population in descending order according to the fitness values. Then it picks one individual at a time from the top and removes all the individuals with worse fitness than the selected one and falling within the specified clearing radius. This step will be repeated until all the individuals in the population are either selected or removed. Clearing eliminates similar individuals and maintains the diversity among the selected individuals. Similar to sharing, clearing also needs a user-specified parameter σ_{clear} called clearing radius. This parameter is used as a dissimilarity threshold. The complexity of clearing is O(cN), where c is the number of niches maintained during the generations.

2.1.2 Crowding

The crowding method introduced by De Jong in 1975 [12] allows competition for limited resources among similar individuals in the population. Hence, the competition is within each niche. This approach will maintain the diversity of the whole population. Generally, the similarity is measured using the distance between individuals. The algorithm compares an offspring with some randomly sampled individuals from the current population. The most similar individual will be replaced if the offspring is a superior solution. A parameter CF called the crowding factor is used to control the

size of the sample. CF is generally set to 2 or 3. The computational complexity of crowding is equal to O(N), where N is the population size. A major advantage of crowding is its simplicity. However, replacement error is the main disadvantage of crowding. Deterministic Crowding [27], Probabilistic Crowding [29] are other variants of crowding technique.

2.1.3 Sharing

The fitness sharing was introduced by Holland [14] and extended by Goldberg and Richardson [16]. The concept is to divide the population into different subgroups according to the similarity of the individuals. An individual must share information with other individuals within the same niche. The shared fitness for the i^{th} individual can be represented as follows:

$$f_{\text{shared}}(i) = \frac{f_{\text{original}}(i)}{\sum_{j=1}^{N} sh(d_{ij})} \tag{2.1}$$

where the sharing function is calculated as

$$\text{sh}(d_{ij}) = \begin{cases} 1 - \left(\frac{d_{ij}}{\sigma_{\text{share}}}\right)^{\alpha}, & \text{if } d_{ij} < \sigma_{\text{share}} \\ 0, & \text{otherwise.} \end{cases} \tag{2.2}$$

d_{ij} is the distance between individuals i and j, σ_{share} is the sharing radius, N is the population size, and α is a constant called sharing level. The complexity of fitness sharing is $O(N^2)$. The advantage of sharing is its ability to form and maintain stable subpopulation/niches. Sharing also encourages the search in unexplored regions of the space by increasing the diversity of the population. One of the drawbacks of sharing is the usage of the niching parameter σ_{share}. Specifying this parameter requires prior knowledge of how far apart the optima lies.

2.1.4 Speciation

The idea of speciation is commonly used in multimodal optimization [30]. This method also depends on a radius parameter rs, which measures the Euclidean distance from the center of a species to its boundary. The center of a species is called species

seed. Each of the species is built around the dominating species' seed. All individuals that fall within the radius from the species seed are identified as the same species. In this way, the whole population is classified into different groups according to their similarity. The complexity of speciation is O(N) in the best case and O(N^2) in the worst case. The main advantage of speciation is its ability to maintain high diversity and stable niches over generations while the main disadvantage is the selection of the radius parameter r_s.

In addition to the methods listed above, there are other niching methods such as restricted tournament selection (RTS) [17],multi-population [19], clustering [20], and localized niching [21].

2.2 Non-DE methods for multimodal optimization

There have been extensive amount of evolutionary algorithms discussed in literature for multimodal optimization. The multimodal optimization problems can be solved either by differential evolution based approaches or non-differential evolution based approaches. The non-differential evolution based approaches consist of genetic algorithms [49], swarm algorithms [53], evolution strategies [57], clustering based algorithms [74], ensemble methodologies [76] and other adhoc evolutionary algorithms [9]. In this section, we have discussed non-differential evolution based methodologies to solve MMOPs.

2.2.1 Genetic Algorithms(GAs) for multimodal op-timization

Genetic Algorithms (GAs) [49] are adaptive, randomized search techniques founded on the simulation of the Darwinian evolution and natural genetics. They are efficient, adaptive, and robust search processes, producing near-optimal solutions and offer a large amount of implicit parallelism. In our ecosystem natural evolutionary process basically maintains a variety of species, each occupying a different ecological niche, whereas classical GAs rapidly push the artificial population toward convergence i.e., all individual populations soon gather and become more or less identical. Even when

multiple optima exist in a problem, classical GAs are used to locate only one optimum. Niching methods are embedded in a GA [50] to make it capable of detecting multiple optimal solutions by using a single population. The basic motivation of including niching methods was to promote diversity in classical GAs. In [27] it has been concluded that three important factors that reasonably take part in the loss of diversity are selection pressure, selection noise, and operator disruption. Four niching techniques were tested in [27] with various difficulty levels on GA. Parallel hillclimbing is the best for the easiest problem and has some success on the problems with intermediate difficulty which often fails on complex optimization problems. Sequential niching is the weakest method to handle most of the difficult multimodal optimization problems available in the literature. Ursem proposed a Multinational Evolutionary Algorithm (MEA) [51] for detecting global and local optima on a function landscape. The method tends to adapt itself to the problem by catching some topological features of the fitness landscape under consideration. The main concept is to employ the topological information for grouping the whole population into subpopulations each of which will cover a part of the fitness landscape. Ursem further extended the work in [52] by proposing Multinational GAs (MGAs) to tackle multimodal optimization problems in dynamic environments.

2.2.2 Swarm intelligence algorithms for multimodal optimization

Swarm Intelligence (SI) has attracted interest from many researchers in various fields. Bonabeau defined SI as "The emergent collective intelligence of groups of simple agents" [81]. SI is the collective intelligence behavior of self-organized and decentralized systems, e.g., artificial groups of simple agents. Examples of SI include the group foraging of social insects, cooperative transportation, nest-building of social insects, and collective sorting and clustering. Two fundamental concepts that are considered as necessary properties of SI are self-organization and division of labor. Self-organization is defined as the capability of a system to evolve its agents or components

into a suitable form without any external help.

The concept of particle swarm, although initially introduced for simulating social behaviors commonly observed in the animal kingdom has become very popular these days, as an efficient means for intelligent search and optimization. Since its advent in 1995, the Particle Swarm Optimization (PSO) [53,54] algorithm has attracted the attention of a lot of researchers all over the world resulting in a huge number of variants of the basic algorithm as well as many parameter selection/control strategies, comprehensive surveys of which can be found in [55,56]. In PSO, the particles are conceptual mathematical entities, which accelerate simultaneously along with two directions – the best positions of the search space individually experienced by each of them at some point of time and the globally best position found by a neighborhood (geographical or social) of the current particle so far. Thus the particles have a tendency to fly toward the better and better regions of the search space over time, which results in the fast convergence of the search. PSO requires no gradient information of the function to be optimized is very easy to implement in any standard programming language and uses only primitive mathematical operators throughout.

2.2.3 Evolution strategies for multimodal optimization

Evolution Strategies (ESs) [57,58] are nature-inspired optimization techniques built around the concept of the evolution of evolution practically biological processes are optimized by evolution, and on the other hand, evolution is itself a biological process, thus it may be concluded that evolution optimizes itself. ES emphasizes the phenotypic behaviors of individuals or search agents that constitute a population. Each individual consists of a set of decision (search) variables and strategy parameters. Evolution then amounts to changing both the decision variables and strategy parameters and the evolution of decision variables is controlled by the strategy parameters. As far as real-valued search spaces are concerned, the mutation is normally performed by adding a normally distributed random value to each vector component. The step size or mutation strength (i.e. the standard deviation of the normal distribution) is often governed by self-adaptation. Individual step sizes for each coordinate or correlations

between coordinates are either governed by self-adaptation or by covariance matrix adaptation (as done in CMA-ES [59]. The canonical versions of the ES are denoted by $(\mu/\rho, \lambda) - ES$ and $(\mu/\rho + \lambda) - ES$ respectively. Here μ denotes the number of parents, $\rho \leqslant \mu$ the mixing number (i.e., the number of parents involved in the procreation of an offspring), and λ the number of offspring. The parents are deterministically selected (i.e., deterministic survivor selection) from the (multi-)set of either the offspring, referred to as comma-selection ($\mu < \lambda$ must hold), or both the parents and offspring, referred to as plus-selection.

2.2.4 Clustering-based multimodal evolutionary approaches

Several research works have been reported for solving multimodal problems by incorporating clustering methods in some classical EAs. Different clustering techniques have been suitably used to overcome the inability of the traditional EAs to locate multiple solutions and to enhance their efficiency, comprehensiveness, and robustness. Ling et al. [74] proposed Crowding Clustering Genetic Algorithm (CCGA) where they utilize clustering strategy to eliminate the genetic drift that is introduced by the crowding strategy. The authors introduced a peak detection concept to combine the clustering and crowding techniques. Basically to create multiple niches in the given landscape the standard crowding strategy is employed in CCGA. Clusters formed by the crowding model can coexist in the same niche and lead to the same optimal solution. As both standard and deterministic crowding tend to converge to numerous potential solutions and to create genetic drift, clustering operation is used to remove this genetic drift by introducing an inter-cluster competition and stimulating exploration in the entire search space. Felix et al. proposed a Clustering-Based Niching (CBN) [75] method for Evolutionary Algorithms (EAs) to identify multiple global and local optima's in a multimodal environment. The principle behind the CBN implementation is to apply the biological concept of species in separate ecological niches to EA for preserving diversity that results advantageous to find out niches of arbitrary size, shape, and spacing.

2.2.5 Ensemble method for solving multimodal optimization problems

Recently, ensemble idea was adopted in evolutionary algorithms to solve multimodal optimization problems. Yu and Suganthan [76] proposed an ensemble of niching algorithms (ENA) which use four parallel populations. Each population is associated with one niching method. All four populations use a genetic algorithm as the search method. The offspring produced by all four populations are combined and subsequently added to the four populations separately. Each population will select parents for the next generation from each combined population according to the niching method used. In this way, each algorithm always keeps the best offspring according to the selection rules of the associated niching method. Wu et al. [109] investigated the high-level ensemble of multiple existing DE variants. A multi-population-based framework (MPF) is proposed to realize the ensemble of multiple DE variants to derive a new algorithm named EDEV for short. EDEV consists of three highly popular and efficient DE variants, namely JADE [93] (adaptive differential evolution with optional external archive), CoDE (differential evolution with composite trial vector generation strategies and control parameters) and EPSDE (differential evolution algorithm with ensemble of parameters and mutation strategies) [108]. Wu et al. [109] provide an extensive survey about the ensemble strategies.

There are other approaches for solving MMOPs that cannot be categorised in the above mentioned sections 2.2.1-2.2.5. Hence, those approaches are discussed below.

Yang et al. [9] studied the application of Estimation of Distribution Algorithms (EDA) for solving MMOPs. They aimed at taking advantage of Gaussian and Cauchy distributions to generate the offspring at the niche level by alternatively using these two distributions. Such utilization can also potentially offer a balance between exploration and exploitation. Two popular niching strategies, crowding, and speciation, were incorporated in Multimodal EDA (MEDA), leading to MCEDA and MSEDA, respectively. Multi-objective Optimization (MO) problems involve multiple objectives, which should be optimized simultaneously and that often are in conflict with each

other. This results in a group of alternative solutions (Pareto optimal set) that must be considered equivalent in the absence of information concerning the relevance of the others. Many EAs were formulated by the researchers to tackle multi-objective problems in recent past [77,78]. Deb and Saha [79,80] took an MO approach for solving the multimodal optimization problems. They converted the single-objective multimodal problem into a suitable bi-objective optimization problem so that all optimal solutions become members of the resulting weak Pareto optimal set. One of the objectives was the objective function of the multimodal optimization problem and the authors made a number of suggestions for choosing the other objective. Starting with the gradient-based approaches (demonstrating the foundation of the bi-objective approach), more pragmatic neighborhood count based approaches were developed for this purpose.

2.3 Differential evolution for multimodal optimization

Differential Evolution (DE) [60–62] is arguably one of the most powerful stochastic real-parameter optimization algorithms of current interest. DE has been frequently adopted to tackle multi-objective, constrained, dynamic, large scale, and multi- modal optimization problems and the resulting variants have been achieving top ranks in various competitions held under the IEEE CEC (Congress on Evolutionary Computation) conference series. In DE community, the individual trial solutions (which constitute a population) are called parameter vectors or genomes. DE operates through the same computational steps as employed by a standard EA. However, unlike traditional EAs, DE employs difference of the parameter vectors to explore the objective function landscape. Like other population-based search techniques, DE generates new points (trial solutions) that are perturbations of existing points, but these deviations are not samples from a predefined probability density function, like those in ESs. Instead, DE perturbs current generation vectors with the scaled difference of two randomly selected population vectors. In its simplest form, DE adds the scaled, random vector difference

to a third randomly selected population vector to create a donor vector cor- responding to each population vector (also known as target vector). Next the components of the target and donor vectors are mixed through a crossover operation to produce a trial vector. In the selection stage, the trial (or offspring) vector competes against the population vector of the same index, i.e. the parent vector. Once the last trial vector has been tested the survivors of all the pair wise competitions become parents for the next generation in the evolutionary cycle. A detailed survey on the state-of-the-art research on DE as well as its applications to different kinds of optimization problems can be found in [62]. Thomsen integrated the fitness sharing concept with DE to form the sharing DE [63]. Sharing DE utilizes the classical sharing technique described in Eqs. (2.1) and (2.2), using the Euclidean distance as the distance metric. In each generation the number of offspring generated is equal to the parent population size. Thus after Np trial vectors have been generated from Np parents, the sharing function is used to calculate the fitness for each individual and the worst half of the population is purged. The algorithm provides elitism by always preserving the individual with best un-scaled fitness. The algorithm requires defining σ_{share} that represents the threshold of dissimilarity or niche radius. Thomsen also proposed to extend DE with a crowding Scheme (Crowding DE) [63] to allow it to tackle multimodal optimization problems. In Crowding DE (CDE) when an offspring is generated by using he standard DE, it competes only with the most similar (measured by Euclidean distance) individual in the current population. The offspring will replace this individual if it has a better fitness value. To avoid replacement error in CDE, the crowding factor CF is taken equal to the population size Np. Li proposed the Species-based DE (SDE) algorithm, built around the notion of speciation [64], for solving multimodal optimization problems. The Species-based DE (SDE) is capable of locating multiple global optima simultaneously through adaptive formation of multiple species (or subpopulations) in a DE population at each iteration step. The DE population is partitioned into species according to an Euclidean distance based similarity metric. Opposition DE [48] is one of the variants of DE based on the concept of opposition based learning. Generally speaking, evolutionary optimization methods start with some initial solutions (initial

population) and try to improve them toward some optimal solution(s). The process of searching terminates when some predefined criteria are satisfied. In the absence of a priori information about the solution, it usually start with random guesses. The computation time, among others, is related to the distance of these initial guesses from the optimal solution. It can improve the chance of starting with a closer (fitter) solution by simultaneously checking the opposite solution. By doing this, the fitter one (guess or opposite guess) can be chosen as an initial solution. In fact, according to probability theory, 50 percent of the time a guess is further from the solution than its opposite guess. Therefore, starting with the closer of the two guesses (as judged by its fitness) has the potential to accelerate convergence. The same approach can be applied not only to initial solutions but also continuously to each solution in the current population. The oppositon DE majorily involves two stages namely opposition-based population initialization and opposition based generation jumping. The pseudocode for opposition DE is described in algorithm 1. Notations used in algorithm 1 are P_0: Initial population,OP_0: Opposite of initial population, N_p: Population Size,P:Current Population, OP:Opposite of current population, V: Noise vector,U: Trial vector, D : Problem dimension,: $[a_j, b_j]$Range of the j^{th} variable, BFV: Best fitness value so far, VTR: value to reach, NFC: Number of function calls,MAX_{NFC}: Maximum number of function calls, F: Mutation Factor, $rand(0, 1)$: Uniform random variable, C_r:crossover rate, $f(.)$: objective function, P›: population of next generation, J_r : Jumping rate, min_j^p: minimum value of the j_{th} variable in current population,: maximum value of the j^{th} variable in the current poplulation.

A first approach that employs DE to evolve subpopulations for achieving simultaneous convergence to multiple optima of a multimodal function can be traced in [65]. The proposed algorithm implements a mating restriction, so that the variation operations are performed only inside each subpopulation. Additionally, a penalty is applied to members of each subpopulation that are too close to members of different subpopulations to drive each subpopulation toward a different optimum. The method requires definition of the number of subpopulations, a penalty term, and the minimum spanning distance to be maintained among the subpopulations, which are all problem-dependent

Algorithm 1 Opposition Differential Evolution

/*Opposition-Based Population Initialization*/

1. Generate uniformly distributed random population P_0
2. $for(i = 0; i<N_p; i++)$
3. $for(j = 0; j<D; j++)$
4. $OP_{0i,j} = a_j + b_j - P_{0i,j}$
5. Select N_p fittest individuals from the set P_0, OP_0 as initial population P_0

/*End of Opposition-Based Population Initialization*/

6. While($BFV>VTR$ and $NFC<MAX_{NFC}$)
7. $for(i = 0; i<N_p; i++)$
8. Select three parents P_{i1}, P_{i2}, P_{i3} randomly from the current population where $i \neq i_1 \neq i_2 \neq i_3$
9. $V_i = P_{i_1} + F * (P_{i_2} - P_{i_3})$
10. for(j=0;j<D;j++)
11. if $(rand(0,1)<C_r)$
12. $U_{i,j} = V_{i,j}$
13. else
14. $U_{i,j} = P_{i,j}$
15. Evaluate U_i
16. if $f(U_i) \leqslant f(P_i)$
17. $P_i^{'} = U_i$
18. else
19. $P_i^{'} = P_i$
20. $P = P^{'}$

/*Opposition-Based Generation Jumping*/

21. if $(rand(0,1)<J_r)$
22. $for(i = 0; i<N_p; i++)$
23. $for(j = 0; j<D; j++)$
24. $OP_{i,j} = MIN_j^p + MAX_j^p - P_{i,j}$
25. Select N_p fittest individuals from the set P, OP as population P

/*End of Opposition-Based Generation Jumping*/

26. End While

parameters and thus form a major difficulty for the use of the method. Rumbler and Moore [66] attempted to overcome these limitations by suggesting a NewEDE method for determining the optimal values for these parameters. The idea is to simply run the algorithm repeatedly with different parametric setups to determine suitable values and at the same time keep record of the found solutions. Of course the repeated runs increase the computational complexity of the whole process.

Zaharie [67] proposed a Multi-resolution Multipopulation DE (MMDE), which divides the population to c equally sized subpopulations. The search is divided into epochs, between which the subpopulations are reinitialized using a finer separation of the domain, so that the number of sub-domains increases by c after each epoch. The best solution in each subpopulation is stored in an archive after each epoch. To prevent redundant solutions from entering the archive, the Euclidean distance between each new entree is calculated for each existing solution in the archive, and too similar solutions are discarded. A hill–valley detection method [51] is used to exclude solutions belonging to the same peak. MMDE does not require the definition of the niche radius parameter, but introduces a set of new parameters, the number and size of the subpopulations, as well as the number and length of the epochs. Hendershot took a similar approach in his MultiDE algorithm [68] by considering equal-sized subpopulations. However, in MultiDE the number of subpopulations is kept variable i.e. subpopulations can appear and disappear. MultiDE uses a structure similar to archive in MMDE, called population 0. When an element from a subpopulation is similar to an element from population 0, the former is no longer considered for further evolutions. The similarity is based on a precision parameter to be controlled by the user. MultiDE employs a minimum spanning distance to encourage the search for different optima. It also introduces another time delay regarding the tunable parameter operation i.e. the number of generations after which a subpopulation is eliminated if it fails to discover a new optimum. Zaharie devised a multipopulation crowding DE [69] by integrating a crowding based niching technique along with the multipopulation DE algorithm. Under this scheme subpopulation reinitialization is no more necessary as each subpopulation is capable of locating multiple optima. The crowding computa-

tion is kept limited to subpopulations so that a global processing step can be avoided. DE with Local Selection (DELS) [70] is a variant of DE where the target vector and the base vector for mutation are kept same. In the selection phase each population member is always compared to its own mutant. When the selection is local, evolution of a vector only depends on the current set of vector differences, and not directly on the parameter values of vectors other than those of the target. In effect, local selection partitions the population into Np niches, each of which is inhabited by a single vector that evolves in isolation. In order to employ the potential of DELS for solving multimodal problems, Rönkkönen and Lampinen divided the mutation operation of DELS into two parts [71]: local mutation and global mutation. The resulting algorithm, DELS using local mutation (DELL) [71, 72], also adopts the "either/or" concept [61](p. 117), which uses only one variation operator for generating each trial. The selection between two possible operators is done probabilistically for each trial using the PX parameter to control the probabilities for using either. While it would be possible to use the traditional uniform crossover operation also with DELS, it would destroy the rotational invariance of the approach, and thus the crossover has been removed from the proposed algorithm. More recently, Qu and Suganthan [73] proposed a neighborhood-based DE mutation for multimodal optimization. The experimental results suggested that the performances of the DE-niching algorithms are greatly improved with the introduction of neighborhood mutation. Qu et al. [89] proposed a neighborhood mutation strategy and integrated with various niching DEs to solve MMOPs. The proposed neighborhood concept allows a higher exploitation of the areas piloting the moves, thereby facilitating multiple convergences. In neighborhood mutation, difference vector generation is limited to a number (parameter m) of similar individuals as measured by Euclidean distance. Each individual is evolved toward its nearest optimal point and the possibility of between niche difference vector generation is reduced. Basak et al. [90] proposed biobjective formulation of the multimodal optimization problem and used differential evolution (DE) with nondominated sorting followed by hypervolume measure-based sorting to finally detect a set of solutions corresponding to multiple global and local optima of the function under test. Unlike the

two earlier multiobjective approaches (biobjective multipopulation genetic algorithm and niching-based nondominated sorting genetic algorithm II), the proposed multimodal optimization with biobjective DE (MOBiDE) algorithm does not require the actual or estimated gradient of the multimodal function to form its second objective. Niching DE with indexed based neighborhood is studied in [91]. Epitropakis et al. [91] presented a niching DE algorithm that attempts to overcome the population size influence and produces good performance almost independently of its population size. Taking DE/nrand/1 [92] as a baseline model, they proposed a parameter independent algorithm by incorporating two additional mechanisms into its structure: a well known control parameter adaptation technique [93] and an external dynamic archive along with a reinitialization mechanism [94]. The adaptive control parameter technique will alleviate the problem of having to fine-tune the standard control parameters required by Differential Evolution, i.e. the mutation and recombination factor. On the other hand, the dynamic archive along with the reinitialization mechanism will be responsible for keeping the best potential solutions found by the algorithm and it will simultaneously re-initialize some individuals to allow the algorithm to search unexplored regions of the problem space. As a result, the algorithm is able to continue its search for additional good global solutions, without being bound by an initial small population size, resulting in a niching algorithm with its performance being almost independent from the population size parameter, i.e. small populations should be sufficient to tackle complex multimodal problems. Liang et al. [95] proposed DE based on fitness Euclidean-distance ratio for multimodal optimization. In the standard differential evolution, vectors are randomly chosen from the whole population to generate the differential vectors. While since the individuals which are far from given individual cannot represent the properties of the local landscape of the given individual, this operator is not suitable for the multi-modal optimization. Thus in the proposed differential evolution based on fitness Euclidean-distance ratio (FERDE), the fittest-and-closest individuals are chosen as vectors to generate the offspring. And considering that the scaling factor α is a constant value for the current population and does not affect the sorting of FER values, it is omitted in the proposed algorithm

to decrease the computational complexity. Zhang et al. [96] introduced a variant of DE by combining the Composite DE(CoDE) [97] with Queue Selection (QS) mechanism. CoDE randomly combines three trial vector generation strategies and three control parameter settings. The trial vector generation strategies and control parameter settings of CoDE are chosen in a way that they have distinct advantages. Therefore, their combination can be effective in solving different kind of problems. The three trial vector generation strategies are: 1) rand/1/bin 2) rand/2/bin 3) current-to-rand/1. The proposed queueing selection operator is an adaptation of the clearing procedure which is described in [96]. Huang et al. [99] proposed a hypercube-based partition of neighborhoods, and then apply it to DE with neighborhood mutation for multimodal optimization. The hypercube-based partition strategy is more computationally inexpensive but can still help DE to achieve a good solution accuracy and convergence speed. Although hypercube is a geometrical concept rarely adopted in DEs, it can indeed be used to form niches in DEs. Similar to distance- based neighborhood partitions, the hypercube-based partition divides a population into niches so as to maintain the diversity of the whole population. Besides, it is computationally inexpensive since neighborhoods are divided based on subtraction rather than Euclidean distance between any two individuals. Moreover, every individual has its own hypercube in the proposed hypercube-based DE, which serves as a niche or subpopulation in DE. And it can change the radius vector of its hypercube adaptively so as to adjust the number of members in each niche. Thus, the diversity and convergence of DE are adjustable. Based on hyper-cube-based neighborhoods, neighborhood mutation is adopted due to its high efficiency, and comparative selection takes place between an offspring and the closest individual in its hypercube. Zhao et al. [100] proposed a differential evolution (DE) algorithm based on the local binary pattern (LBP). The LBP makes use of the neighbors' information for extracting relevant pattern information, so as to identify the multiple regions of interests, which is similar to finding multiple peaks in MMOP. Inspired by the principle of LBP, this paper proposes an LBP-based adaptive DE (LBPADE) algorithm. It enables the LBP operator to form multiple niches, and further to locate multiple peak regions in MMOP. Moreover, based on the LBP niching

information, authors developed a niching and global interaction (NGI) mutation strategy and an adaptive parameter strategy (APS) to fully search the niching areas and maintain multiple peak regions. The proposed NGI mutation strategy incorporates information from both the niching and the global areas for effective exploration, while APS adjusts the parameters of each individual based on its own LBP information and guides the individual to the promising direction. Wang et al. [101] proposed a new automatic niching technique based on the affinity propagation clustering (APC) and design a novel niching differential evolution (DE) algorithm, termed as automatic niching DE (ANDE), for solving MMOPs. In the proposed ANDE algorithm, APC acts as a parameter-free automatic niching method that does not need to predefine the number of clusters or the cluster size. Also, it can facilitate locating multiple peaks without extra FEs. Furthermore, the ANDE algorithm is enhanced by a contour prediction approach (CPA) and a two-level local search (TLLS) strategy. Firstly, the CPA is a predictive search strategy. It exploits the individual distribution information in each niche to estimate the contour landscape, and then predicts the rough position of the potential peak to help accelerate the convergence speed. Secondly, the TLLS is a solution refine strategy to further increase the solution accuracy after the CPA roughly predicting the peaks. Wang et al. [103] developed a parameter-free niching method based on the adaptive estimation distribution (AED). Specifically, the AED first utilizes several close individuals to estimate an approximate distribution for each individual. Then, the AED will adaptively determine the appropriate niche size based on the distribution information for each individual. By using the AED, each individual will find its own appropriate niche size to form a niche and will act as an independent unit to find a global optimum, which avoids the difficulty of population partition and the sensitivity of niching parameters. After using AED to find the appropriate niche size, each individual will form a niche by finding its several close individuals according to its own niche size. Then, these niches co- evolve based on the master–slave multiniche distributed model. This distributed model is applied to DE, where the operators of DE are executed within each niche on the corresponding slave, forming a distributed DE (DDE). The multiniche co-evolution mechanism can fully

exchange the evolutionary information among different niches to enhance the population diversity for fully exploring the search space and finding more global optima. After that, to refine the solution accuracy and locate multiple global optima more precisely, a probabilistic local search (PLS) is further proposed.

2.4 Performance Measures

Our objective is to compare the capabilities of different niching algorithms to locate all global optima. To achieve this, first we need to specify a level of accuracy (e.g., $0 < \epsilon \leqslant 1$), a threshold value under which we would consider a global optimum is found. Second, we assume that for each test function, the following information is available:

1. The number of global optima.
2. The objective function value of the global optima (or peak height), which is known or can be estimated.
3. A niche radius value that can sufficiently distinguish two closest global optima.

We use peak ratio (PR) [28] and success rate (SR) as two performance measures, to evaluate the performance of a niching algorithm over multiple runs. Given a fixed maximum number of function evaluations (MaxFEs) and a required accuracy level ϵ, PR measures the average percentage of all known global optima found over multiple runs:

$$PR = \frac{\sum_{run=1}^{NR} NPF_i}{NKP * NR}$$

where NPF_i denotes the number of global optima found in the end of the i^{th} run, NKP the number of known global optima, and NR the number of runs. SR measures the percentage of successful runs (a successful run is defined as a run where all known global optima are found) out of all runs:

$$SR = \frac{NSR}{NR}$$

where NSR denotes the number of successful runs.

Chapter 3

Enhanced Opposition Differential Evolution Algorithm for Multimodal Optimization

3.1 Introduction

In this chapter, we are going to propose an opposition DE-based methodology for solving MMOPs. We follow the multi-species framework and hence the global population must be divided into local sub-populations. The key idea is to generate a standard DE population as well as an opposition DE population and select the best members for the next generation. Also, apart from the evolution process, the parameters of the algorithm need to be set adaptively to adapt to the different regions in the overall function landscape. Since the population size impacts the evolution process, hence a minimum size must be maintained. Also, there must be a mechanism to improve the candidate solutions by searching in the vicinity of the best candidate in the species. Considering all these above points, the contribution of this chapter is as follows:

1. To propose a method that divides the global population into local sub-populations efficiently such that each sub-population corresponds to a niche and

most of the niches are located.

2. To develop a better balancing technique for the newly created sub-populations to help enhance convergence speed towards optima within the sub-population.

3. To propose a modified version of ODE that acts as an efficient evolution process.

4. To develop an adaptive technique to learn the parameters, mutation factor (F) and crossover rate (CR) over the generations.

5. To propose a method that performs an efficient local search within each sub-population to refine the candidate solutions.

6. To introduce a method to deal with multiple peak values corresponding to the same species.

The remainder of this chapter is organized as follows. In Section 3.2 we describe the basic concepts for our contribution. In Section 3.3 we elaborate on the proposed algorithm and its components. Section 3.4 describes the experiments and results obtained. Finally, this chapter is concluded in Section 3.5.

3.2 Preliminaries

In order to achieve objective 1 mentioned in section 3.1, we have proposed a modified version of NBC-Minsize [39]. NBC-Minsize is an enhanced version of NBC. Nearest Better Clustering(NBC) [23] is a technique used in MMOPs to divide the population into several species and each species tries to find an optimal solution. The parameter φ controls the number of species in NBC. When we set φ to a relatively small value, the population could be divided into more species. Thus, more promising areas could be located by different species. However, a small φ value results in excessive segments. Moreover, the number of species containing only a few individuals (e.g., one or two individuals) would greatly increase, making the species incapable of evolving with the mutation operators of DE. Therefore, the parameter *minsize* is used to limit the minimal number of individuals in the species. It is noted that the value of *minsize*

must be carefully handled. If it is too small, the minimum size limitation mechanism is ineffective; if it is too large, two species locating at different peaks would likely be still linked together. To obtain better results, *minsize* should take a small value in the early stage and relatively large values in the middle and later stages. In the early stage of the evolution process, since the exploration of search space is desirable over exploitation, more niches needs to be located that apparently target more peaks. In the middle and later stages of the evolution, the population tends to converge near the region of optima, a larger *minsize* helps merge the species having global optima with the nearby species having local optima. Thus, the species converge to the global optima more quickly. Algorithm 2 refers to the NBC-Minsize.

First, the value of *minsize* of species is set by equations (3.1) and (3.2) which are as follows:

$$minsize(g) = 5 + g/2 \tag{3.1}$$

$$bound = max(10, 3 * D) \tag{3.2}$$

Second, a spanning tree of population is constructed identical to the standard NBC and the mean distance μdist is calculated. Following this, we calculate the $follow$ vector. Each element of $follow$ represents the number of nodes in the subtree rooted at the corresponding individual; specifically, each value of $follow$ is initially set to 1. Next, the edges are sorted in descending order according to the fitness values of the follower individuals. Finally, for each edge, the follow value of each follower individual is added to that of a leader individual. Following this, the edges in T are sorted by their Euclidean distance in descending order. Subsequently, edges satisfying the conditions are cut off to form two species with the end points of the edge as species' seeds. In addition to the condition of standard NBC, that the distance exceeds the weighted mean distance, the other condition is that the sizes of the two species after the cutoff must be all greater than or equal to *minsize* (illustrated in lines 8–16 of Algorithm 2). Finally, the partition scheme is returned and the species are obtained.

Algorithm 2 NBC-Minsize

Set $minsize$ by equation (3.1), (3.2);
Construct the spanning tree T ;
Calculate the mean distance μ_{dist} ;
Calculate the $follow$ vector;
Sort the edges in T from the longest to the shortest;
for each $e \in T$ **do**
    if $dist(e) > \varphi * \mu_{dist}$ **then**
        Set e_f to the follower individual of e;
        Set e_r to the root of the subtree containing e_f ;
        if $follow(e_f) \geqslant minsize$ **and**
        $follow(e_r) - follow(e_f) \geqslant minsize$ **and**
        f($\frac{e_f+e_r}{2}$)<$f(e_r)$ **and** f($\frac{e_f+e_r}{2}$)<$f(e_f)$ **then**
            Cut off e;
            Set e_l to the leader individual of e;
            for each x on the path from e_l to e_r **do**
                $follow(x) = follow(x) - follow(e_f)$
            end for
        end if
    end if
end for

3.3 Enhanced Opposition Differential Evolution Algorithm(EODE) for Multimodal Optimization

In this section, we present the opposition DE based method for solving MMOPs. The proposed method consists of five different components which are presented in the form of Algorithms 3-5, 8-9 where Algorithm 3 represents the generic framework used for solving MMOPs. Algorithms 4-5, 8-9 are further components of Algorithm 3. Algorithms 6-7 are components of Algorithm 5. The proposed method is tested on the IEEE CEC 2013 benchmark functions.

In the next sub-section, we describe the multi-species framework used for solving MMOPs.

3.3.1 Multi-Species Framework

In multi-species framework, the idea is to divide the randomly distributed population into sub-populations called species and perform the evolutionary process on each species independently [19]. The block diagram represented by Figure 3.1 broadly depicts the components of EODE Algorithm. In Figure 3.1, initialization is the process of randomly assigning values to the population within bounds and it corresponds to step 2 of Algorithm 3. Two-level speciation is used to divide the global population into local sub-populations or species. Two-level speciation corresponds to section 3.3.1.1 and step 5 of Algorithm 3. Species balance strategy is used to balance the oversized and undersized species. It corresponds to section 3.3.1.2 and step 6 of Algorithm 3. Evolutionary process is employed to evolve the hybrid population that has been generated by DE and ODE operations. Evolutionary process corresponds to section 3.3.1.3 and step 8 of Algorithm 3 while local search corresponds to section 3.3.1.5 and step 9 of the Algorithm 3. Merge Archive procedure checks for the peaks belonging to the same species in order to locate multiple different peaks. It corresponds to section 3.3.1.6 and step 10 of Algorithm 3. Fes represents the current count of fitness evalua-

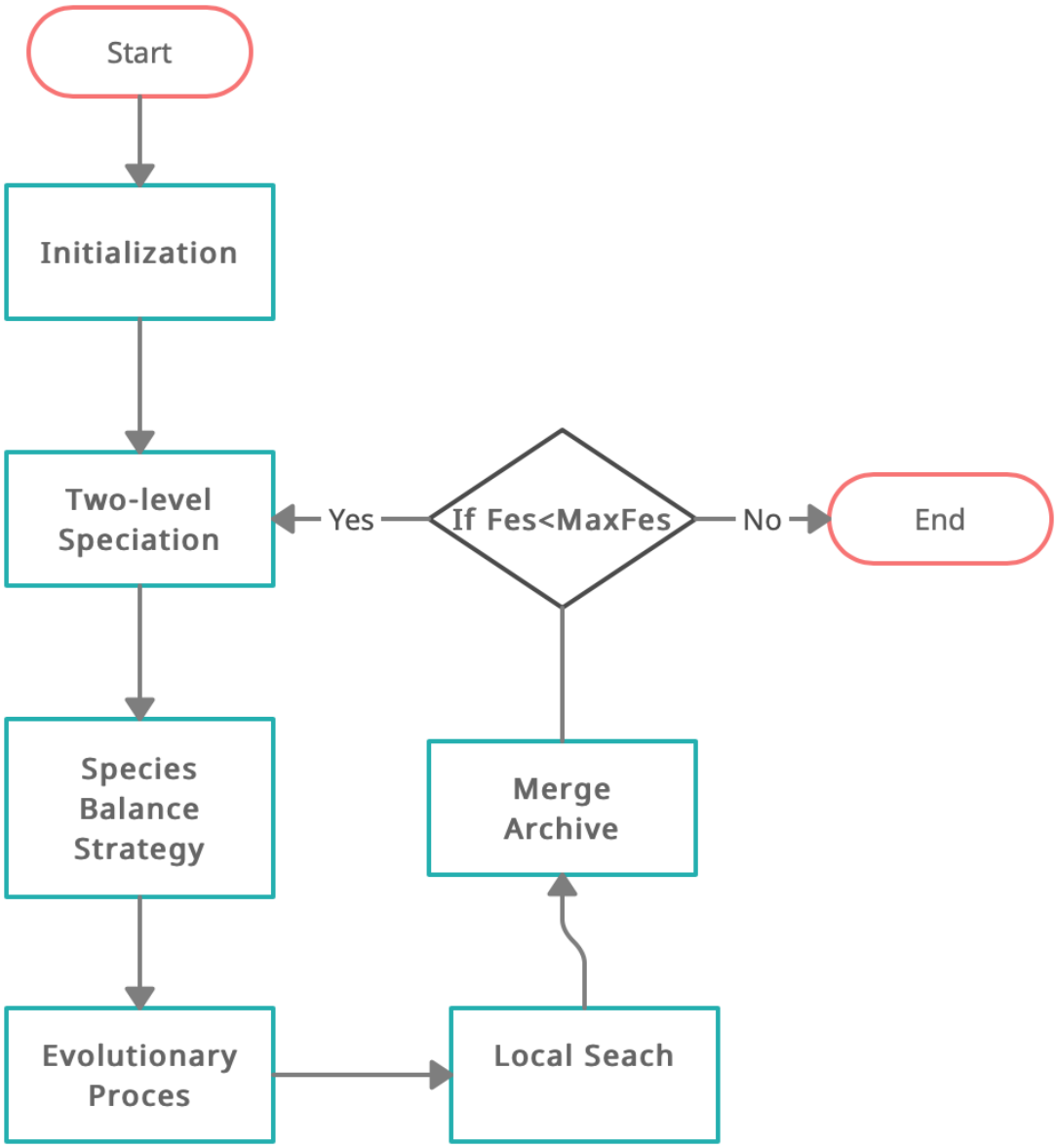


Figure 3.1: Block Diagram of EODE Algorithm

tions while MaxFes represents the count of maximum fitness evaluations allowed. The procedures such as two-level speciation, species balance strategy, evolutionary process, local search and merge archive keep on running until the current fitness count becomes greater than or equal to maximum count of fitness evaluations allowed. Algorithm 3 represents the generic framework which further consists of five other algorithms. In algorithm 4, the population is initialized randomly within the bounds specified for each

Algorithm 3 EODE

1: **Input**: Function ($f(\vec{X})$),Population Size (NP), $MaxFes$
2: Initialize the population randomly within the bounds of the dimensions
3: $Gen = 0$,$Fes = 0$,$archive = [\,]$
4: **while** $Fes \leqslant MaxFes$ **do**
5: Obtain multiple species by two-level application of Algorithm 2
6: balanceSpecies(*multi-species*)
7: **for** each $species \in multi\text{-}species$ **do**
8: $localbest$=Modified Opposition DE($species, speciesfitness$)
9: $bestfit$=localSearch($localbest$,$species$)
10: mergeArchive($archive$,$bestfit$)
11: **end for**
12: Gen+=1
13: **end while**

dimension. The parameters Gen, Fes, $archive$ represents generation number, number of fitness evaluations, and archive, respectively. The archive is used to store optimum values. Steps 5-10 are performed until maximum fitness evaluations are reached. Algorithm 3 is invoked for creating multiple species out of the global population in step 5. After the formation of multiple species, there is a need to balance the species as each species could be representing different niches in the function landscape. To this aim, step 6 is introduced which balances the species based on the shape and size of niches. Now, for each species that is present in multi-species, steps 8-10 are carried out. In step 8, modified ODE (section 3.3.1.2) is applied to each species and returns the local best for that species. Local search (section 3.3.1.4) is a method to refine the accuracy of the best solution obtained so far. Step 9 tries to find the better solution nearby the *localbest* achieved. It may so happen that the *localbest* obtained is very very close to one of the optima already present in the archive and both are part of the same niche. Hence, there is a need to check the redundancy of the optimizers obtained. To this aim, Step 10 is introduced.

In the next section we describe the two-level speciation procedure using the application of Algorithm 2 on the global population.

3.3.1.1 Two-level speciation

This section corresponds to the step-5 of the multi-species framework described in Algorithm 3. In this section we describe how the global population is divided into local sub-populations using Algorithm 2. In Algorithm 2, while using mNBC [39] for speciation, it may so happen that the root node (e_r) and follower node (e_f) belong to the same niche. In such a case, the existing mNBC will cut off the edge between e_r and e_f resulting in the formation of two different species even though they are part of the same species. This will lead to a wastage of fitness evaluations. Hence, to improve the efficacy of mNBC, an additional check is applied in Algorithm 1. Step 12 Algorithm 2 represents the check which checks for the presence of valley between e_r and e_f by using an extra fitness evaluation. Even though an extra fitness evaluation is used, it will save many unnecessary fitness evaluations. We have applied two-level speciation to capture the narrow regions of a peak. First-level speciation is performed with higher $minsize$ and hence it may associate multiple regions of a peak in a single species. To identify those close species, we employ a second level of speciation within the large species to segregate the nearby species. Second-level speciation is performed with lower $minsize$. Hence, it introduces four parameters i.e. $\varphi_1, \varphi_2, minsize_1, minsize_2$. The parameters $\varphi_1, minsize_1$ are associated with first-level speciation and $\varphi_2, minsize_2$ are associated with second-level speciation.

In the next section we describe the procedure to balance the species' that is obtained after two-level speciation.

3.3.1.2 Species Balance Strategy

In this section, we describe a method to balance the the species' by redistributing members across the different species'. Step-6 in Algorithm 3 corresponds to the species balancing strategy. After multiple species are obtained, some species with a narrow and small basin of attraction could have many individuals and some species with a

wide and large basin of attraction could have fewer individuals, hence there is a need to balance such unbalanced species. Since the niches could be of different shapes and sizes, the species balance strategy should be dynamic. In algorithm 4, initially, a minimum threshold of 10 individuals has to be present in the species which is taken care of by steps 3-4. It is done to be able to compute the covariance matrix of t best individuals in the species indicated in step 6. The variance or spread of a species is calculated by summing up the eigenvalues of the covariance matrix. The eigen values are represented using $eigen_i$ (step 8) where i runs from 1 to k and k represents the dimensionality of the problems. The average species size is computed in step 9 which considers the global population size(NP). Step 10 checks if the size of the species($speciessize$) is greater than the product of δ and the average size of the species($avgsize$) then the individuals with the least fitness values are removed from the species indicated by step 11. δ is a hyperparameter here that controls the number of new individuals generated in a species. The underbalanced species with large variance is balanced first and hence step 12. For each unbalanced species, the new individuals are generated in the vicinity of species seed using the Gaussian distribution. To this aim, Steps 13-15 are introduced.

In the next section we present the modified opposition DE for evolving the species that we obtained after balancing them.

3.3.1.3 Modified Opposition DE

In this section, we introduce the modified opposition DE to evolve the sub-population. This procedure corresponds to the step 8 mentioned in Algorithm 3 (section 3.3.1) . Algorithm 6 consists of the application of modified opposition DE to each species.

The notations used in algorithm 5 are as below:

F_1set: Set to store successful $\vec{F_1}$ values, F_2set:Set to store successful $\vec{F_2}$ values, $CRset$: Set to store successful $\vec{CR}$ values, Max_FES: Maximum Fitness Evaluations, FES: Current fitness evaluations, Gen: Current Generation Number, $MaxGen$: Maximum Generation Number, $\vec{FB}$: First Best Member, $\vec{SB}$: Second Best Member, $\vec{TB}$: Third

Algorithm 4 balanceSpecies

1: **Input**: *multi-species*

2: For each $species \in multi\text{-}species$

3: if $speciessize \leqslant dim$ or $speciessize \leqslant 10$

4: $c = max(dim - popsize, 10)$

5: Initialize c individuals around the species seed

6: $t = max(speciessize/Gen, 10)$

7: Compute covariance matrix of t best members of species

8: Compute variance of species using: $var = \sum_{i=1}^{k} eigen_i$

9: Average species size($avgsize$) = $\frac{NP}{\text{No of Species}}$

10: if $speciessize > \delta * avgsize$

11: Remove the $(speciessize - \delta * avgsize)$ worst individuals from the species

12: $sortedspecies$ =Sort the species in descending order of variances.

13: for each $species \in sortedspecies$

14: if($species < \delta * avgsize$)

15: Generate ($\delta * avgsize$-$speciessize$) number of individuals around the species seed using covariance matrix computed in step 7 and Gaussian distribution with standard deviation given by equation (3.8).

Best Member, $\vec{X_{rk}}$: Population Member, pr: Probability, NP: Population size, $\vec{U}$: Trial vector, F_1: Mutation factor, F_2: Mutation factor, CR: Crossover rate, JR: Jumping Rate, D: Dimensions,MIN_j: Minimum in j^{th} dimension , MAX_j: Maximum in j^{th} dimension, OP_{ij}: Opposite member, P_{ij}: Current member , P: Current Population, OP: Opposite Population

The initial population is generated by using the math formulation mentioned in 1.1.2. Three sets are taken to store the successful values of $\vec{F_1}$, $\vec{F_2}$, and $\vec{CR}$. The successful values here mean the values which lead to the production of better offspring. pr represents the probability with which the exploitation needs to be done. In early generations, more exploration of search space needs to be done while in the later phases more exploitation needs to be done. Hence, to this aim steps 8-17 are introduced. Step 10 represents the perturbation of lower degree as there is only one mutation component while step 12 represents the perturbation to be of higher degree as two mutation components are introduced. The moderate and strong exploitation are introduced by Step 15 and 17 respectively.

Step 15 uses the candidate with highest fitness value within that species represented by $\vec{FB}$ with random differential vector to create the target vector. The step 17 uses the best 3 different candidates of the species. The bounds are checked using the following equation.

$$P_{i,d} = \begin{cases} min(ub[d], 2 * lb[d] - P_{i,d}) & \text{if}(P_{i,d}<lb[d]) \\ max(lb[d], 2 * ub[d] - P_{i,d}) & \text{if}(P_{i,d}>ub[d]) \end{cases} \tag{3.3}$$

where 'd' represents dimension, ub[d] and lb[d] represents upper and lower bounds in dimension 'd'. $P_{i,d}$ represents the value of d^{th} dimension of i^{th} member in the population. The successful parameters such as mutation factor and crossover rate are stored in their respective sets using step 23. The jumping rate JR determines the degree of opposition applied to the species to create the opposite population. The opposite population is generated during the 33% of the evolution process to enable better exploitation. The key point to note here is that the mutation factors $(\vec{F}1, \vec{F}2)$ and crossover rate $(\vec{CR})$ are calculated for each dimension for every generation in the population. We have used the vector representations of mutation factors and crossover

Algorithm 5 Modified Opposition DE

Input : $species, speciesfitness$

$F_1set = \phi$, $F_2set = \phi$, $CRset = \phi$

While $FES<MaxFES$ and $Gen<MaxGen$

  For each member$\in$ species

    Find first($\vec{FB}$), second($\vec{SB}$) and third($\vec{TB}$) best members in the species.

    Randomly sample 5 members($\vec{X_{r1}}, \vec{X_{r2}}, \vec{X_{r3}}, \vec{X_{r4}}, \vec{X_{r5}}$) from the species

    $pr = Gen/MaxGen$

    if $pr \leqslant 0.33$

      if random(0,1)$\leqslant$ 0.75

        $\vec{V} = \vec{X}_{r1} + \vec{F}_1 * (\vec{X}_{r2} - \vec{X}_{r3})$

      else

        $\vec{V} = \vec{X}_{r1} + \vec{F}_1 * (\vec{X}_{r2} - \vec{X}_{r3}) + \vec{F}_2 * (\vec{X}_{r3} - \vec{X}_{r4})$

    else if $pr \leqslant 0.67$

      Randomly select 2 members ($\vec{X_{k1}}, \vec{X_{k2}}$) out of $\frac{NP}{2}$ best members

      $\vec{V} = \vec{FB} + \vec{F}_1 * (\vec{X}_{k1} - \vec{X}_{K2})$

    else

      $\vec{V} = \vec{FB} + \vec{F}_1 * (\vec{SB} - \vec{TB})$

    Apply binomial crossover operation to get $\vec{U}$

    Check for the bounds of $\vec{U}$ in each dimension using equation (3.3)

    Evaluate the child fitness.

    FES+=1

    If childfitness>parentfitness

      $F_1set \cup \vec{F}_1, F_2set \cup \vec{F}_2, CRset \cup \vec{CR}$,

  $JR = Gen/MaxGen$

  if $JR \leqslant 1$ and $JR>0.67$

    Find the minimum, maximum bounds of the current species in each dimension.

    $for(i = 0; i<NP; i++)$

      if random(0,1)$<$ 0.33

        $for(j = 0; j<D; j++)$

          $OP_{i,j} = MIN_j + MAX_j - P_{i,j}$

      else

        $for(j = 0; j<D; j++)$

33: $OP_{i,j} = MIN_j + random(0,1) * (MAX_j - MIN_j)$

34: Evaluate the opposite population

35: FES+=NP

36: Restart the candidates randomly that are stuck for 10 successive generations.

37: Select the NP candidates with best fitness from $\{P \cup OP\}$

38: Update $\vec{F_1}, \vec{F_2}$, and $\vec{CR}$

39: $Gen+ = 1$

40: Return the best member of the species

rate to be able to capture the degree of perturbation in every generation individually. This helps in creating diverse offsprings for better exploration.

While creating the opposite population, we have introduced a tweak while creating the offspring. Steps 20 - 22 indicate the modification. Instead of creating the opposite population in the shrunken space, some amount of randomization is introduced to help in the exploration of the function space. For the rest of the 50% of time, the target vector is produced by the differential perturbation of randomly selected vectors of the sub-population. During the evolution process, the population may get trapped in local optima. To avoid the trap, the population members are checked for improvement in fitness values, if they do not improve over some k (let say 10) successive generations as compared to global best member, then it is an indication of potential trap at local optima. To this end, step 36 is introduced. Step 38 is introduced at the end of every generation to update the algorithm parameters using adaptive parameter strategy described in the next section.

3.3.1.4 Adaptive Parameter Strategy

In this section, we describe the adaptive parameter strategy to update the mutation factors and crossover rate in such a way that it adapts to the landscapes of different shapes and sizes and helps in efficient evolution of the species. The evolution process i.e. modified ODE is discussed in section 3.3.1.3. The adaptive parameter strategy is used by modified ODE and it indirectly corresponds to the step 8 of the

Algorithm 3 (section 3.3.1). To update the values of $\vec{F_1}, \vec{F_2}, \vec{CR}$, we have used a mechanism inspired from SHADE [41]. The complete pseudocode is given by Algorithm 7 and Algorithm 8. We have used two mutation factors ($\vec{F_1}$ and $\vec{F_2}$) to guide the population towards the optima based on the stage of evolution. Weighted power mean is used in the calculation of $\vec{F}1$ and $\vec{CR}$ whereas weighted Lehmer mean is used in the calculation of the $\vec{F}2$. Power mean is used to interpolate between the minimum and maximum values using arithmetic mean and harmonic mean. Lehmer mean is used to capture the non-linearity of the moving averages of the parameter values. Weights are used to capture the improvement in influencing parameter adaptation. Weighted Lehmer mean and weighted power mean are computed using equation (3.5) and (3.4) respectively. The $S_{\vec{F}}$ represents the set of mutation factors that produces a better offspring. w_k, computed using equation (3.6), represents the weight, and δf_k represents the fitness difference between parent and offspring. The consideration of all the dimensions while calculating the $\vec{F}1$, $\vec{F}2$, $\vec{CR}$ for next-generation helps predict the degree of perturbation to be introduced in each dimension for better exploration. $random(0, 1)$ represents the random number between 0 and 1 including them. wlm, wpm represents weighted Lehmer mean and weighted power mean respectively. $\vec{F_{old}}$ represents the $\vec{F}$ of the previous generation. $\vec{max}$, $\vec{min}$ represents the boundaries of the current species (sub-population) to which DE operations are applied. $\vec{U}$, $\vec{L}$ represents the upper and lower bounds of the objective function respectively. So, essentially the calculation of $\vec{F}$ considers the older values of $\vec{F}$, the shape and size of the basin of attraction, and the information about the stage of evolution to estimate the parameter values of the algorithm for the next generation.

$$Mean_{W,Power}(\vec{F}_{success}) = (\frac{1}{|S_{\vec{F}}|}\sum_{k=1}^{|S_{\vec{F}}|} w_k * S_{\vec{F},k})^{\frac{1}{1.5}} \tag{3.4}$$

$$\text{Mean}_{W,Lehmer}\left(\vec{F}_{success}\right) = \frac{\sum_{k=1}^{|S_{\vec{F}}|} w_k \cdot S_{\vec{F},k}^2}{\sum_{k=1}^{S_{\vec{F}}|} w_k \cdot S_{\vec{F},k}} \tag{3.5}$$

$$w_k = \frac{\Delta f_k}{\sum_{k=1}^{|S_{CR}|} \Delta f_k} \tag{3.6}$$

where $Mean_{W,Power}(\vec{F}_{success})$, $\text{Mean}_{W,Lehmer}\left(\vec{F}_{success}\right)$ represents weighted power mean and weighted Lehmer mean. $S_{\vec{F},k}$ represents set of successful mutation factors while $|S_{\vec{F}}|$ represents the size of the set which contains successful mutation factors. Δf_k represents the fitness difference between the parent and successful offspring.

Algorithm 6 Adaptive Parameter Strategy($\vec{F}$)

1: $wf = 0.8 + 0.2 * random(0, 1)$

2: Compute weighted power mean for $\vec{F}1$ and weighted Lehmer mean for $\vec{F}2$.

3: $\vec{F} = 0.25 * \vec{F}_{old} + 0.25 * (\vec{max} - \vec{min})/(\vec{U} - \vec{L}) + 0.5 * (1 - \frac{FES}{MaxFes})$

4: $\vec{F} = wf * \vec{F}_{old} + (1 - wf) * wpm(\text{or } wlm)$

Algorithm 7 Adaptive Parameter Strategy($\vec{CR}$)

1: $wf = 0.9 + 0.1 * random(0, 1)$

2: Compute weighted power mean.

3: $\vec{CR} = wf * \vec{CR}_{old} + (1 - wf) * wpm$

The adaptive parameter strategy for mutation factors is described in Algorithm 6. Algorithm 7 introduces adaptive parameter strategy for crossover rate. In the next section, we define the local search method to improve the best solution obtained after applying modified ODE in hope to further improve its fitness.

3.3.1.5 Local Search

Local search methods are generally used to refine the accuracies of the obtained solutions. It is being used in step 9 of Algorithm 3. Our proposed local search method is described in Algorithm 8. Essentially, we generate some members in the vicinity of the best candidate obtained so far in the species in a hope to find further better candidate. We initialize variances ($vars$) using step 4 described in algorithm 8. The key idea is to keep incrementing the variances ($vars$) if the better offsprings are not found in the near by region of the best individual of the species. dim represents dimensionality of the problem. *dirvec*, computed in step 16, can be visualized as a direction vector and it is used to perform a guided search nearby the *localbest*

candidate. The degree of the shift from the *localbest* is continuously updated in the hope of finding a better solution by incrementing the variances. In Algorithm 8, Gaussian distribution has been used to generate offsprings around the *localbest* in the current species. For efficient search in the nearby region of the *localbest* candidate, we generate offspring by differential vector perturbation and estimating the distribution of the handful of better candidates in the species. Mean and standard deviation are computed using (3.7), (3.8).

$$\mu_i^d = \frac{1}{M}\sum_{j=1}^{M} X_j^d \tag{3.7}$$

$$\delta_i^d = \sqrt{\frac{1}{M-1}\sum_{j=1}^{M}\left(X_j^d - \mu_i^d\right)^2} \tag{3.8}$$

where $\mu_i = \left[\mu_i^1, \ldots, \mu_i^d, \ldots, \mu_i^D\right]$ and $\delta_i = \left[\delta_i^1, \ldots, \delta_i^d, \ldots, \delta_i^D\right] (1 \leqslant i \leqslant s)$ are, respectively, the mean and standard deviation (std) vectors of the i^{th} niche, $\boldsymbol{X}_j = \left[X_j^1, \ldots, X_j^d, \ldots, X_j^D\right]$ is the j^{th} individual in the i^{th} niche and D is the dimension size of the multimodal problem.

The covariance matrix is used to capture the behavior of variances of each dimension with respect to the other dimensions. Covariance between two variables of dimension D is computed as follows:

$$\mathrm{COV}_{\mathrm{X,Y}} = \frac{\sum_{i=1}^{D}\left(X_i - \bar{x}\right)\left(Y_i - \bar{y}\right)}{D-1} \tag{3.9}$$

where $\bar{x}, \bar{y}$ represents X mean and Y mean. $\vec{O}_{fitness}, \vec{X}_{best\,fitness}$ represents the offspring fitness and member with best fitness respectively. In the next section, mergeArchive procedure is discussed that aims at removing the peaks belonging to the same niche.

3.3.1.6 Merge Archive

In this section, we describe the procedure to deal with duplicate peaks in the *archive*. This section corresponds to the step 10 of the Algorithm 3. Since we are storing the peaks in an *archive*, it is important to store the peaks only. Otherwise, we may end up storing the entire candidates of a sub-population (species) and

Algorithm 8 localSearch

1: Input :*localbest*,*species*
2: *dirvec*=None
3: k=10
4: $vars = random(0.001, 0.01)$
5: While(k)
6: $mean = localbest$
7: if *dirvec* is not None and $random(0, 1) \leqslant 0.5$
8: offspring[k]=*localbest*+*vars***dirvec*
9: else
10: *mbest*=max(speciessize/4,10)
11: Find the covariance matrix of the *mbest* candidates of species using equation (3.9).
12: if dim> 1
13: offspring[k]=Apply multivariate normal distribution with mean as *mean* and covariance matrix as computed in step 11.
14: else
15: offspring[k]=Apply univariate normal distribution with mean as *mean* and standard deviation is as computed using equation (3.7)
16: if $\vec{O}_{fitness} > \vec{X}_{best\,fitness}$
17: *dirvec*=offspring[k]-*localbest*
18: *localbest*=offspring[k]
19: else
20: for d in dim:
21: $vars[d] += random(0.001, 0.01)$
22: k=k-1
23: Return the *localbest*

subsequently the entire population. This will lead to the consumption of more computational resources since the *archive* will contain huge points. Hence, given two individuals, it is essential to check whether they belong to similar species or different species. It is critical to algorithm performance in locating all the global optima. In algorithm 4, the input is *localbest*, and it needs to be checked whether this is a new species seed or an old species seed. The nearest peak to the given local best is found out, and the midpoint between them is calculated. If the midpoint has lower fitness than the nearest peak and *localbest*, it indicates the presence of a valley between them. Hence, local best can be considered species seed of new species, and it is added to the *archive*. In the next section, we describe the experiments performed and the results obtained.

Algorithm 9 mergeArchive

1: **Input** :$archive, localbest$
2: Find the nearest peak to *localbest* present in the *archive*
3: Find the mid-point between *localbest* and nearest peak
4: if (midpointfitness >*localbest* and midpointfitness >nearest peak)
5: Add *localbest* to the *archive*
6: else
7: Replace the better fit individual between *localbest* and nearest peak with midpoint.

3.3.2 Illustrative Example

In order to explain the components of EODE, an example is presented in this section. We have considered low dimensional (2-D) composition function F11 described in Table A.1 of Appendix section for demonstration purpose.

As it can be seen from Figure 3.1, the population is randomly initialized in the function space. Now we will try to map the steps given in Algorithm 3 to the process that happens in Figures 3.2-3.3.

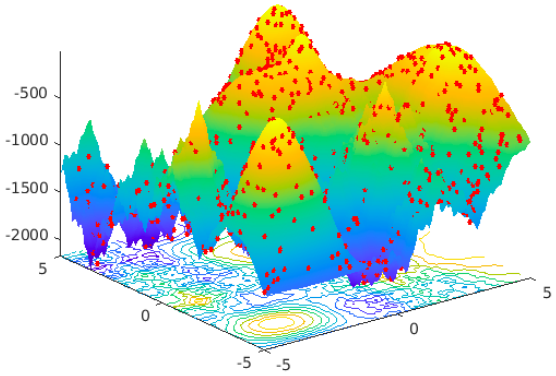


Figure 3.2: Initialization

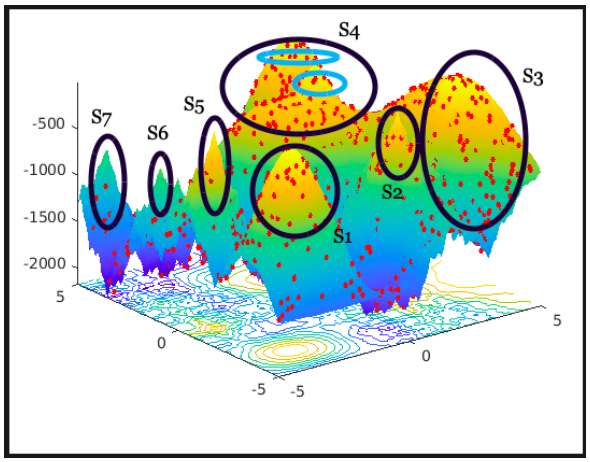


Figure 3.3: Niching and Balancing

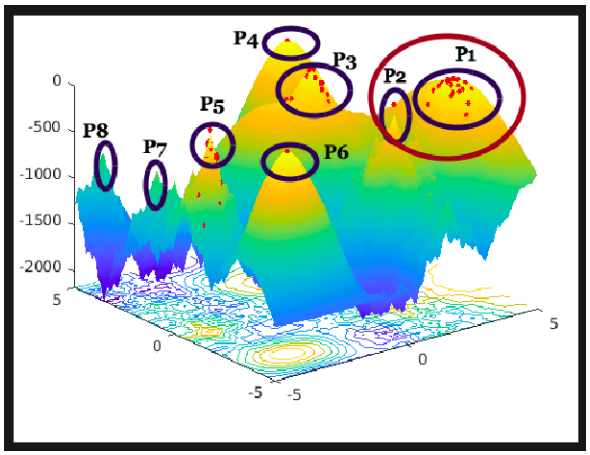


Figure 3.4: ODE and Local Search

1. Initially, the randomly distributed population needs to be divided into sub-populations that represent regions around the peaks. The multiple species marked as black circles in Figure 3.2 are shown as S1-S7. In species S4, the blue circles represent two species that got located as a single species S4. To avoid such merging, two-level speciation is introduced. To this aim step, 5 (Algorithm 3) is introduced.

2. As the multiple species are formed, some species may have huge population size and some may have small population size as can be seen from Figure 3.2. Hence, the species balance strategy is introduced that aims to generate offsprings using Gaussian distribution. Also, the species balance strategy is discussed in

Algorithm 4 which maps to step 6 in Algorithm 3.

3. Modified ODE (section 3.3.1.3) acts as an evolutionary process that tries to further converge the sub-populations towards the peak they represent. The species are represented in Figure 3.3 as P1-P8. Further, it is more effective to converge towards global optima rather than local optima. It can be seen in Figure 3.3 in which P7 and P8 are local optima. This process maps to step 8 in Algorithm 3.

4. After the evolution process completes, the population gets converged near the optima but may not be able to reach the optima accurately as can be seen in Figure 3.3. Hence, we proposed a local search method to search in the vicinity of the best individual of the species as the actual peak would be nearby the best value in the species. The local search method is described in Algorithm 8 and it maps to step 9 of Algorithm 3.

5. Since the speciation process is repeated after every certain number of generations, the best solutions returned by the local search method could be part of the same species that has been explored in the previous generations. Hence, there is a need to detect the best individuals that belong to the same species and they should not be added to the archive. Therefore, Algorithm 9 is introduced that maps to step 10 of Algorithm 3.

3.4 Experiments and Results

We have performed the experiments on a computer system with RAM 8GB, 1.8GHz CPU and MacOS 11 operating system. In this section, EODE is independently run 50 times for each function. The algorithm calculated the results in five levels of accuracy $\epsilon = \{1e{-}1, 1e{-}2, 1e{-}3, 1e{-}4, 1e{-}5\}$. For different problems, the value of NP is shown in Table 3.1. The parameters of EODE are listed in Table 3.2. φ_1, φ_2 are used for creating multiple species out of the global population by applying mNBC twice and

Index	Function	NKP	Peak height	r	MaxFEs	NP
1	$F_1(1D)$	2	200.0	0.01	5.0E + 4	250
2	$F_2(1D)$	5	1.0	0.01	5.0E + 4	250
3	$F_3(1D)$	1	1.0	0.01	5.0E + 4	250
4	$F_4(2D)$	4	200.0	0.01	5.0E + 4	250
5	$F_5(2D)$	2	1.03163	0.5	5.0E + 4	250
6	$F_6(2D)$	18	186.731	0.5	2.0E + 5	2000
7	$F_7(2D)$	36	1.0	0.2	2.0E + 5	2000
8	$F_6(3D)$	81	2709.0935	0.5	4.0E + 5	3000
9	$F_7(3D)$	216	1.0	0.2	4.0E + 5	4000
10	$F_8(2D)$	12	-2.0	0.01	2.0E + 5	1000
11	$F_9(2D)$	6	0	0.01	2.0E + 5	1000
12	$F_{10}(2D)$	8	0	0.01	2.0E + 5	1000
13	$F_{11}(2D)$	6	0	0.01	2.0E + 5	1000
14	$F_{11}(3D)$	6	0	0.01	4.0E + 5	1000
15	$F_{12}(3D)$	8	0	0.01	4.0E + 5	1000
16	$F_{11}(5D)$	6	0	0.01	4.0E + 5	1000
17	$F_{12}(5D)$	8	0	0.01	4.0E + 5	2000
18	$F_{11}(10D)$	6	0	0.01	4.0E + 5	1000
19	$F_{12}(10D)$	8	0	0.01	4.0E + 5	1000
20	$F_{12}(20D)$	8	0	0.01	4.0E + 5	800

Table 3.1: Information of the benchmark problems and the population size

Parameters	Values
φ_1	1
φ_2	1
$minsize_1$	-1
$minsize_2$	5
δ	1.0
$F1$	$(0, 1)$
$F2$	$(0, 1)$
CR	$(0, 1)$
$MaxGen$	$40(D <= 10)$
	$60(D>10)$

Table 3.2: Parameters in EODE

their values are set as 1, 1 respectively. $minsize_1$, $minsize_2$ are parameters used in two-level application of mNBC and their values are set as -1, 5 respectively. We define the $minsize_1$ as -1 to adapt the first level species as per the dimension and generation of the population. $minsize_2$ is set to be 5 to identify the peaks that have narrow basins of attraction as the species size would be very low in such regions. δ is used in the species balance strategy and it is set to 1. The mutation factors $(\vec{F}1, \vec{F}2)$ and crossover rate $(\vec{CR})$ are initially chosen from the closed interval [0,1] after which algorithm learns to adapt their values. $MaxGen$ represents the number of generations for which the ODE needs to run for the given species. The experimental results of EODE for all benchmark problems are listed in Table 3.3 at all five accuracy levels. These results show that EODE is very stable. In addition, EODE finds all peaks on the simple functions and most of the peaks in the low-dimensional composition problems. EODE is able to find more than 80% of all peaks on the functions containing a large number of global peaks, except for F7(3D).

3.4.1 Comparison with other algorithms

In this section, the results of various algorithms (including EODE) are compared. For simplicity, we have compared the results at ϵ = 1e-4, which are commonly adopted in [110] and [111]. In order to better evaluate the performance of EODE, 15 popular comparison algorithms are selected, such as CDE [28], SDE [64], NCDE, NSDE [89], MOMMOP [112], LoICDE, LoISDE [35], PNPCDE [46], LIPS [82], and the recently proposed algorithms of DE_{cl} [113], LMCEDA, LMSEDA [9], FBK-DE [39], LB-PADE [100], MaHDE [114]. Table 3.4, 3.5, 3.6 shows the different PRs and SRs at the accuracy level ϵ = 1e-4, where the row "bprs" represents the number of the best PR results achieved by these algorithms. The algorithm(s) with the best PR value for a given function is marked in bold. All the algorithms used their own default population sizes. Most of the recently compared results are from their corresponding papers, and the results of other algorithms are still from these papers. From Table 3.4, 3.5, 3.6, it is clear that EODE obtains most of the best PR results among the compared algorithms. The detailed analyses are given below.

1) For the first five problems, most algorithms including EODE can find all global optimal solutions.

2) For the 6^{th}–9^{th} problems (with a large number of global peaks), EODE does not perform as well; however, the results still exceed most of the compared algorithms. For problem 6, the difference is of order 0.005 which is very small. In addition, for the 10^{th} problem, EODE can find all the optimal solutions.

3) For problems 11,12,13 and 15 that are low-dimensional composition functions, EODE obtained the best results. Although on the 14^{th} function EODE does not produce optimal results, the difference between EODE and the best algorithm is small (around 10%). It is worth noting that EODE and FBK-DE find all global peaks on the 13^{th} problem while the other algorithms do not.

4) For the 16^{th}–20^{th} composition functions (in the relatively high dimensions), EODE achieves the best results for all except the 20^{th} function. Especially on the 5-D problem, the best result obtained by EODE exceeds 5% of the best result. For the 18^{th}

problem, EODE performed a little better than the best algorithm while for the 19^{th} problem it performed equivalently well.

3.4.2 Different Components of EODE

3.4.2.1 Different Mutation Operators

First, we compare different mutation operators of EODE. In order to show the balancing ability for exploration and exploitation, four algorithms denoted as EODE, EODE-r, EODE-b, EODE-rb are compared. For mutation operators, EODE-r uses only DE/rand/1 and DE/rand/2 to evolve the species. In addition, EODE-b uses both DE/best/1 and DE/best/2, EODE-rb uses DE/rand/1, DE/rand/2, DE/best/1 and DE/best/2 for mutation, which can be referred from [40]. The two mutation operators in the above algorithms are selected with an equal probability. Except for the mutation operators, the other components are identical. The results are shown in Table 3.8 at $\epsilon = 1e{-}4$. From Table 3.8, it is clear that EODE produces the best PR results for all benchmark problems except for the 6^{th} and 12^{th} functions. The performances of EODE-r and EODE-b are not so good as no peaks are found on the relatively high-dimensional functions. In addition, Table 3.8 shows that most compared algorithms perform well on 1st–5th functions, while EODE-b and EODE-rb are unable to find all the global peaks for function 2 and 4 respectively. EODE-r finds all peaks in the first five functions. Here, we specifically compare the results of EODE and EODE-rb to illustrate the effect of three stage-wise mutation operators. Similar to the previous experiment, the comparisons on four type of problems are discussed.

1) On the 1^{st}–5^{th} problems, considered as simple functions, both EODE and EODE-rb can achieve the desired results for all global optima except for function 4.

2) On the many global peaks problems (the 6^{th}–10^{th} optimization functions), it is clear that the performances of EODE are significantly better than that of EODE-rb except for function 6, which shows that the evolution stage-wise mutation operators are highly effective.

3) On the low-dimensional composition problems (the 11^{th}–15^{th} functions), the PR

results for EODE are superior to those of EODE-rb except for function 12. For functions 11 and 12, EODE-rb either performs better or equivalently well. For problems 13-15, EODE-rb does not perform well as compared to EODE.
4) On the relatively high-dimensional composition problems (the $16th$–20^{th} problems), EODE performs better than EODE-rb, indicating that the evolution stage-wise mutation operations can improve the overall performance of the algorithm.

3.4.2.2 Different values of φ_1, φ_2

Here, the different values of φ_1, φ_2 are compared. In Section 3.3.1.1, φ_1, φ_2 are used to form local sub-populations from global population. Thus, we set (φ_1, φ_2) to three different values (1,1), (0.6,0.6), and (2,1) respectively. From Table 3.7 it can be seen that for $\varphi_1 = 1, \varphi_2 = 1$, EODE achieves the best overall results. For $\varphi_1 = 0.6, \varphi_2 = 0.6$, EODE achieves best result for function 15. For functions 1-5, $\varphi_1 = 2, \varphi_2 = 1$ achieves better results as compared to that of $\varphi_1 = 0.6, \varphi_2 = 0.6$. For functions 6-9 having large number of optima, $\varphi_1 = 0.6, \varphi_2 = 0.6$ achieves better results than $\varphi_1 = 2, \varphi_2 = 1$. However, for higher dimensional functions 16-20, EODE with $\varphi_1 = 2, \varphi_2 = 1$ seems to perform better except for functions 16 and 20. The behaviour is attributed to the number of species formed out of the global population. If the number of optima is large, then lower value of φ_1, φ_2 is effective while if the number of optima is small, then higher value of φ_1, φ_2 is effective.

3.4.3 Different values of Jumping Rate (JR)

Here, different values of JR are compared and it can be referred to from Table 3.9. In section 3.3.1.3, JR is used to form the opposite population. We modified the JR values to see if the opposite population in the exploration stage helps or they help in the exploitation stage. JR is dependent upon the current generation and the maximum generation allowed using equation (3.10):

$$JR = \frac{Gen}{MaxGen} \tag{3.10}$$

$0 \leqslant JR \leqslant 0.5$ indicates that the opposite population is generated in the early stage of evolution i.e. exploration while $0.67 \leqslant JR \leqslant 1$ indicates that the opposite population is generated in the later stage of evolution i.e. exploitation. $0 \leqslant JR \leqslant 1$ indicates that the opposite population is generated across the evolution stages. It can be seen that the opposite population generated in the later phase of evolution turns out to be helpful in locating more peaks. For the functions 1-5, EODE with $0.67 \leqslant JR \leqslant 1$ and $0 \leqslant JR \leqslant 1$ performs equivalently well. For functions 6-9, EODE with $0 \leqslant JR \leqslant 1$ performs better than EODE with $0 \leqslant JR \leqslant 0.5$ except the function 8. For functions 17 and 20, EODE with $0 \leqslant JR \leqslant 1$ achieves best results similar to that of EODE with $0.67 \leqslant JR \leqslant 1$.

	1E-1	1E-2	1E-3	1E-4	1E-5
Functions	(PR,SR)	(PR,SR)	(PR,SR)	(PR,SR)	(PR,SR)
F1(1D)	(1,1)	(1,1)	(1,1)	(1,1)	(1,1)
F2(1D)	(1,1)	(1,1)	(1,1)	(1,1)	(1,1)
F3(1D)	(1,1)	(1,1)	(1,1)	(1,1)	(1,1)
F4(2D)	(1,1)	(1,1)	(1,1)	(1,1)	(1,1)
F5(2D)	(1,1)	(1,1)	(1,1)	(1,1)	(1,1)
F6(2D)	(1,1)	(0.995,0.9)	(0.995,0.9)	(0.995,0.9)	(0.9,0.822)
F7(2D)	(0.805,0)	(0.805,0)	(0.805,0)	(0.805,0)	(0.805,0)
F6(3D)	(0.886,0)	(0.852,0)	(0.852,0)	(0.845,0)	(0.832,0)
F7(3D)	(0.509,0)	(0.505,0)	(0.505,0)	(0.505,0)	(0.442,0)
F8(2D)	(1,1)	(1,1)	(1,1)	(1,1)	(1,1)
F9(2D)	(1,1)	(1,1)	(1,1)	(1,1)	(1,1)
F10(2D)	(0.975,0.8)	(0.975,0.8)	(0.975,0.8)	(0.975,0.8)	(0.975,0.8)
F11(2D)	(1,1)	(1,1)	(1,1)	(1,1)	(1,1)
F11(3D)	(0.8,0)	(0.8,0)	(0.8,0)	(0.8,0)	(0.8,0)
F12(3D)	(0.8,0)	(0.8,0)	(0.8,0)	(0.8,0)	(0.77,0)
F11(5D)	(0.733,0)	(0.730,0)	(0.730,0)	(0.730,0)	(0.728,0)
F12(5D)	(0.7,0)	(0.7,0)	(0.7,0)	(0.684,0)	(0.684,0)
F11(10D)	(0.7,0)	(0.7,0)	(0.7,0)	(0.684,0)	(0.684,0)
F12(10D)	(0.525,0)	(0.525,0)	(0.525,0)	(0.520,0)	(0.505,0)
F12(20D)	(0.25,0)	(0.25,0)	(0.25,0)	(0.25,0)	(0.25,0)

Table 3.3: Results on accuracy levels 1e-1,1e-2,1e-3,1e-4,1e-5

Function	EODE		CDE		SDE		NCDE		NSDE		MOMMOP	
Index	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR
1	**1**	1	**1**	1	0.657	0.373	**1**	1	**1**	1	**1**	1
2	**1**	1	**1**	1	0.737	0.529	**1**	1	0.776	0.667	**1**	1
3	**1**	1	**1**	1	**1**	1	**1**	1	**1**	1	**1**	1
4	**1**	1	**1**	1	0.284	0	**1**	1	0.240	0	**1**	1
5	**1**	1	**1**	1	0.922	0.843	**1**	1	0.745	0.490	**1**	1
6	0.995	0.9	**1**	1	0.056	0	0.305	0	0.056	0	**1**	1
7	0.805	0	0.861	0	0.054	0	0.873	0	0.053	0	**1**	1
8	0.845	0	0	0	0.015	0	0.001	0	0.013	0	**1**	1
9	0.505	0	0.474	0	0.011	0	0.461	0	0.006	0	**1**	1
10	**1**	1	**1**	1	0.147	0	0.989	0.863	0.098	0	**1**	1
11	**1**	1	0.330	0	0.314	0	0.729	0.059	0.248	0	0.716	0.020
12	**0.975**	0.8	0.002	0	0.208	0	0.252	0	0.135	0	0.939	0.549
13	**1**	1	0.141	0	0.297	0	0.667	0	0.225	0	0.667	0
14	0.8	0	0.026	0	0.216	0	0.667	0	0.190	0	0.667	0
15	**0.8**	0	0.005	0	0.108	0	0.319	0	0.125	0	0.618	0
16	**0.730**	0	0	0	0.108	0	0.667	0	0.170	0	0.650	0
17	**0.684**	0	0	0	0.076	0	0.250	0	0.108	0	0.505	0
18	**0.684**	0	0.167	0	0.026	0	0.500	0	0.163	0	0.497	0
19	**0.520**	0	0	0	0.105	0	0.348	0	0.098	0	0.223	0
20	0.250	0	0	0	0	0	0.250	0	0.123	0	0.125	0
bprs	**14**		7		1		5		2		10	

Table 3.4: Comparison with other algorithms on accuracy level 1e-4

Function	EODE		LoICDE		LoISDE		PNPCDE		LIPS		DE_{cl}	
Index	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR
1	**1**	1	**1**	1	**1**	1	**1**	1	0.833	0.686	**1**	1
2	**1**	1	**1**	1	0.235	0.039	**1**	1	**1**	1	**1**	1
3	**1**	1	**1**	1	**1**	1	**1**	1	0.961	0.961	**1**	1
4	**1**	1	0.975	0.902	0.250	0	**1**	1	0.990	0.961	**1**	1
5	**1**	1	**1**	1	0.667	0.333	**1**	1	**1**	1	**1**	1
6	0.995	0.9	**1**	1	0.056	0	0.537	0	0.246	0	0.942	0.340
7	0.805	0	0.705	0.02	0.029	0	0.874	0	0.4	0	0.986	0.64
8	0.845	0	0	0	0.012	0	0	0	0.084	0	0.999	0.9
9	0.505	0	0.187	0	0.005	0	0.472	0	0.104	0	0.726	0
10	**1**	1	**1**	1	0.083	0	**1**	1	0.748	0	**1**	1
11	**1**	1	0.66	0	0.167	0	0.66	0	0.974	0.843	0.667	0
12	**0.975**	0.8	0.495	0	0.125	0	0	0	0.574	0	0.943	0.58
13	**1**	1	0.51	0	0.167	0	0.461	0	0.794	0.176	0.667	0
14	0.8	0	0.657	0	0.167	0	0.592	0	0.644	0	0.667	0
15	**0.8**	0	0.299	0	0.125	0	0.258	0	0.336	0	0.623	0
16	**0.730**	0	0.559	0	0.167	0	0	0	0.304	0	0.667	0
17	**0.684**	0	0.223	0	0.076	0	0	0	0.162	0	0.42	0
18	**0.684**	0	0.219	0	0.157	0	0.147	0	0.098	0	0.667	0
19	**0.520**	0	0.037	0	0.027	0	0	0	0	0	0.357	0
20	0.250	0	0.123	0	0.088	0	0	0	0	0	0.212	0
bprs	**14**		6		2		6		2		6	

Table 3.5: Comparison with other algorithms on accuracy level 1e-4

Function	EODE		LMCEDA		LMSEDA		FBK-DE		LBPADE		MaHDE	
Index	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR
1	**1**	1	**1**	1	**1**	1	**1**	1	**1**	1	**1**	1
2	**1**	1	**1**	1	**1**	1	**1**	1	**1**	1	**1**	1
3	**1**	1	**1**	1	**1**	1	**1**	1	**1**	1	**1**	1
4	**1**	1	**1**	1	**1**	1	**1**	1	**1**	1	**1**	1
5	**1**	1	**1**	1	**1**	1	**1**	1	**1**	1	**1**	1
6	0.995	0.9	0.99	0.843	0.972	0.588	0.990	0.820	**1**	1	**1**	1
7	0.805	0	0.734	0	0.673	0	0.813	0	0.889	0	0.804	0
8	0.845	0	0.367	0	0.613	0	0.824	0	0.575	0	0.983	0.291
9	0.505	0	0.284	0	0.248	0	0.425	0	0.476	0	0.351	0
10	**1**	1	**1**	1	0.998	0.98	**1**	1	**1**	1	**1**	1
11	**1**	1	0.667	0	0.892	0.392	**1**	1	0.674	0	0.725	0.078
12	**0.975**	0.8	0.75	0	0.99	0.922	0.935	0.480	0.750	0	0.650	0
13	**1**	1	0.667	0	0.667	0	1	1	0.667	0	0.667	0
14	0.8	0	0.667	0	0.667	0	**0.907**	0.460	0.667	0	0.667	0
15	**0.8**	0	0.696	0	0.738	0	0.730	0	0.654	0	0.648	0
16	**0.730**	0	0.667	0	0.667	0	0.707	0	0.667	0	0.667	0
17	**0.684**	0	0.456	0	0.62	0	0.630	0	0.532	0	0.352	0
18	**0.684**	0	0.657	0	0.66	0	0.667	0	0.667	0	0.663	0
19	**0.520**	0	0.451	0	0.458	0	**0.520**	0	0.475	0	0.455	0
20	0.250	0	0.059	0	0.248	0	**0.450**	0	0.275	0	0.250	0
bprs	**14**		6		5		10		7		7	

Table 3.6: Comparison with other algorithms on accuracy level 1e-4

Function	$\varphi_1 = 1, \varphi_2 = 1$		$\varphi_1 = 0.6, \varphi_2 = 0.6$		$\varphi_1 = 2, \varphi_2 = 1$	
Index	PR	SR	PR	SR	PR	SR
1	**1**	1	**1**	1	**1**	1
2	**1**	1	0.68	0.2	0.92	0.6
3	**1**	1	**1**	1	**1**	1
4	**1**	1	0.5	1	0.9	0.6
5	**1**	1	**1**	1	**1**	1
6	0.995	0.9	**1**	1	**1**	1
7	**0.805**	0	0.744	0	0.688	0
8	**0.845**	0	0.723	0	0.650	0
9	**0.505**	0	0.476	0	0.420	0
10	**1**	1	**1**	1	**1**	1
11	**1**	1	**1**	1	0.95	0.9
12	**0.975**	0.8	0.95	0.4	**1**	1
13	**1**	1	0.733	0	0.95	0.9
14	**0.8**	0	0.733	0	0.667	0
15	**0.8**	0	**1**	1	**0.8**	0
16	**0.730**	0	0.709	0	0.667	0
17	**0.684**	0	0.515	0	0.580	0
18	**0.684**	0	0.342	0	0.620	0
19	**0.520**	0	0.125	0	0.418	0
20	**0.250**	0	0.175	0	0.125	0
bprs	19		7		7	

Table 3.7: Experimental results of different values of φ_1, φ_2 on the benchmark problems at the accuracy level $\epsilon = 1\text{e}{-4}$

Function	EODE		EODE-r		EODE-b		EODE-rb	
Index	PR	SR	PR	SR	PR	SR	PR	SR
1	**1**	1	**1**	1	**1**	1	**1**	1
2	**1**	1	**1**	1	0.88	0.4	**1**	1
3	**1**	1	**1**	1	**1**	1	**1**	1
4	**1**	1	**1**	1	**1**	1	0.75	0.4
5	**1**	1	**1**	1	**1**	1	**1**	1
6	0.995	0.9	**1**	**1**	0.997	0	**1**	1
7	**0.805**	0	0.734	0	0.733	0	0.788	0
8	**0.845**	0	0.567	0	0.306	0	0.809	0
9	**0.505**	0	0.314	0	0.452	0	0.425	0
10	**1**	1	**1**	1	**1**	1	**1**	1
11	**1**	1	**1**	1	0.933	0.4	**1**	1
12	0.975	0.8	0.925	0	0.8	0	**1**	1
13	**1**	1	0.735	0	0.55	0	0.667	0
14	**0.8**	0	0.667	0	0.505	0	0.667	0
15	**0.8**	0	0.580	0	0.612	0	0.785	0
16	**0.730**	0	0.409	0	0.441	0	0.549	0
17	**0.684**	0	0.186	0	0.279	0	0.667	0
18	**0.684**	0	0.147	0	0	0	0.558	0
19	**0.520**	0	0	0	0	0	0.395	0
20	**0.250**	0	0	0	0	0	0.125	0
bprs	**18**		8		5		8	

Table 3.8: Experimental results of different mutation operators on the benchmark problems at the accuracy level $\epsilon = 1e{-}4$

Function	$0.67 \leqslant JR \leqslant 1$		$0 \leqslant JR \leqslant 0.5$		$0 \leqslant JR \leqslant 1$	
Index	PR	SR	PR	SR	PR	SR
1	**1**	1	**1**	1	**1**	1
2	**1**	1	0.84	0.6	**1**	1
3	**1**	1	**1**	1	**1**	1
4	**1**	1	0.95	0.9	**1**	1
5	**1**	1	**1**	1	**1**	1
6	0.995	0.9	**1**	1	**1**	1
7	**0.805**	0	0.722	0	0.768	0
8	**0.845**	0	0.701	0	0.682	0
9	**0.505**	0	0.433	0	0.488	0
10	**1**	1	**1**	1	0.9	0.4
11	**1**	1	**1**	1	0.933	0.6
12	**0.975**	0.8	0.95	0.4	0.925	0.4
13	**1**	1	0.866	0.2	0.9	0.4
14	**0.8**	0	0.667	0	0.667	0
15	**0.8**	0	0.667	0	0.6	0
16	**0.730**	0	0.667	0	0.670	0
17	**0.684**	0	0.480	0	**0.684**	0
18	**0.684**	0	0.486	0	0.470	0
19	**0.520**	0	0.225	0	0.345	0
20	**0.250**	0	0.150	0	**0.250**	0
bprs	19		6		8	

Table 3.9: Experimental results of different values of JR on the benchmark problems at the accuracy level ϵ = 1e−4

Below figures 3.1-3.11 depict the convergence behaviour of the algorithm on some of the functions. The red dot is a marker which represent the coordinate where the optimum was achieved (or the species got converged).

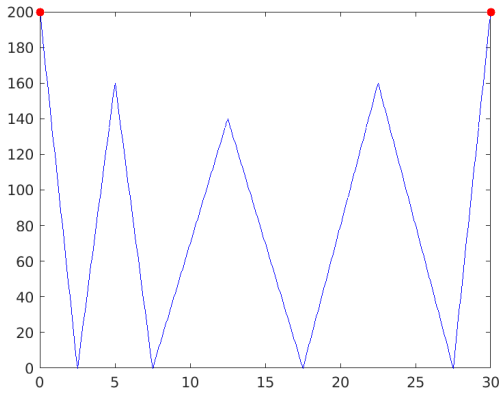


Figure 3.5: F1

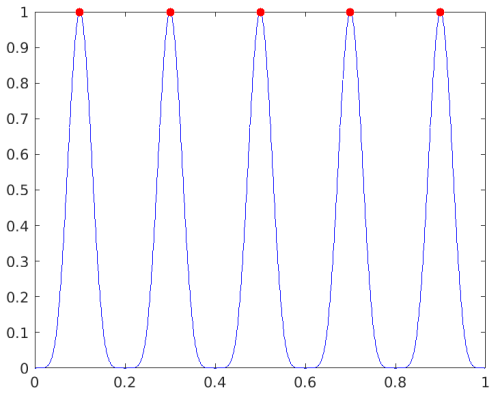


Figure 3.6: F2

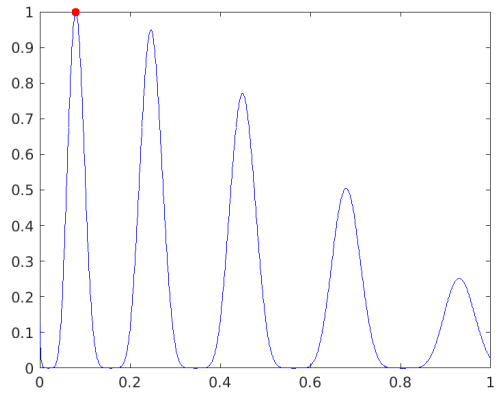


Figure 3.7: F3

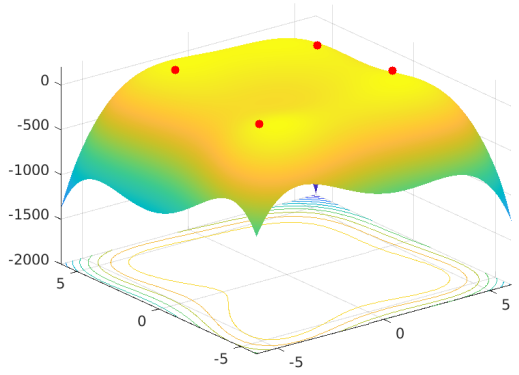


Figure 3.8: F4

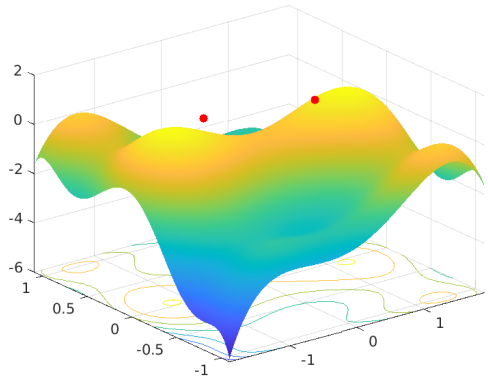


Figure 3.9: F5

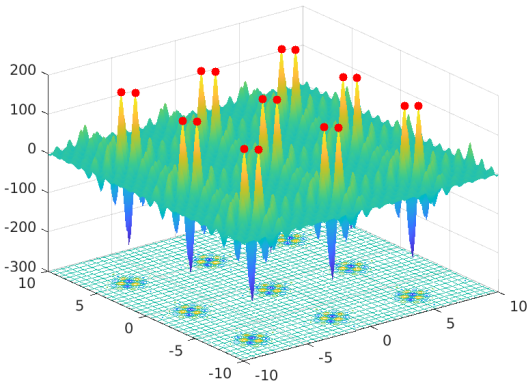


Figure 3.10: F6

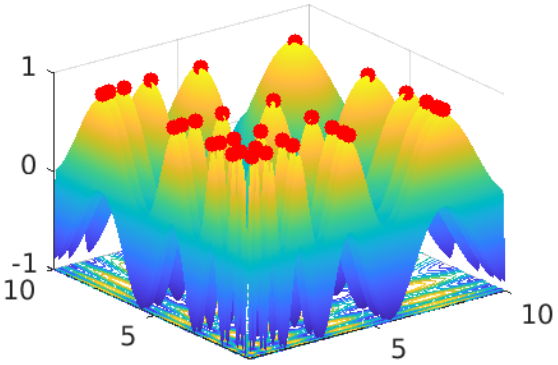


Figure 3.11: F7

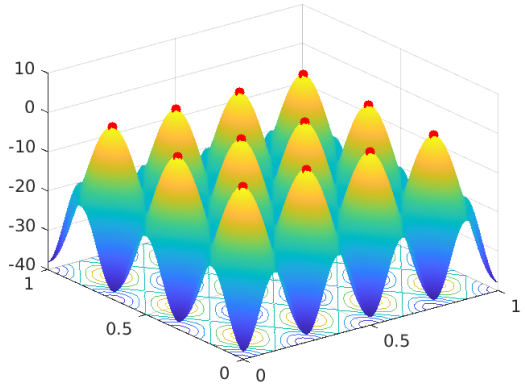


Figure 3.12: F10

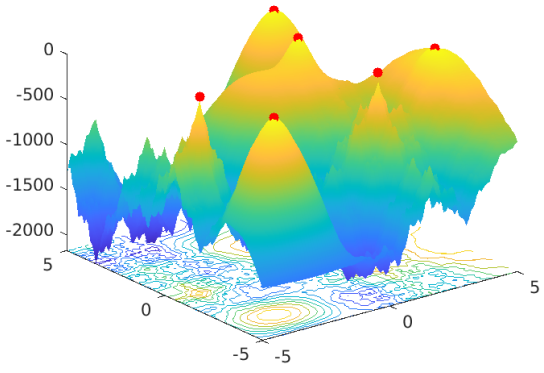


Figure 3.13: F11

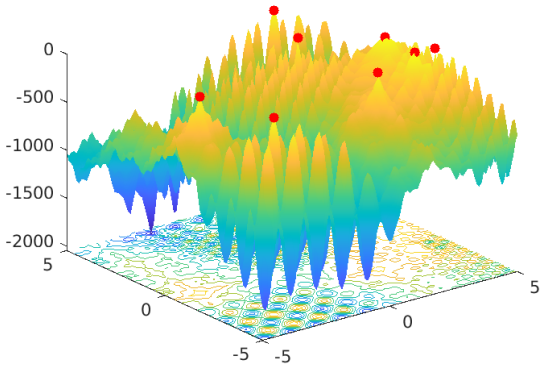


Figure 3.14: F12

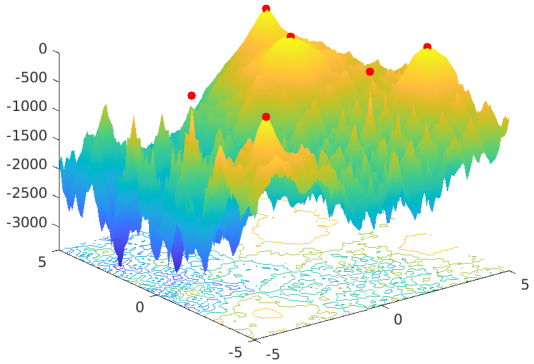


Figure 3.15: F13

3.5 Summary

In this chapter, we have proposed EODE to handle MMOPs, where several species techniques have been adopted to simultaneously find multiple optima. In EODE, two-level NBC-minsize has been embedded to divide the population into several species to locate the optima. To overcome the disadvantages of NBC-minsize an additional check is also performed to improve its efficacy. In addition, a species balance strategy has been adopted to balance the number of individuals generated by the species. We have also proposed modified opposition differential evolution algorithm that acts as the evolutionary process. The evolutionary process is further divided virtually into three stages and mutation operations for each of the stages are embedded individually. Besides, an adaptive parameter strategy has also been proposed to learn the parameters of the algorithm dynamically. We have also proposed a local search method to improve the quality of candidate solutions. Finally, EODE has been compared with several state-of-the-art algorithms. The experimental results shows that EODE performed better than other algorithms on most multimodal benchmark problems. However, in our experiments, we have achieved better results for problems having a moderate number of global optima, but it still needs to improve on highly multimodal functions. Hence, we aim to improve EODE in our future work and test it on highly multimodal problems.

Chapter 4

Estimation of Distribution Algorithm based Differential Evolution Algorithm for Multimodal Optimization

4.1 Introduction

The multimodal optimization technique proposed in the previous chapter uses opposition DE as well as standard DE for creating mutants of the population. However, it is not efficient at handling the problems with a huge number of optima. In this chapter, we extend the idea of the DE algorithm to be able to handle problems with massive multimodality. To this aim, we have proposed a hybrid DE algorithm (EDADE) which uses Estimation of Distribution Algorithms (EDA) as well as standard DE operations to create mutants. Our contributions in this chapter are

- We propose EDA and DE-based hybrid method for generating better offspring in the population and evolving the hybrid species.
- We propose a probabilistic local search method to refine the obtained candidate solutions.

- We introduce a simple strategy embedded in the evolutionary process of EDADE to avoid the population trap at local optima.

The remainder of this chapter is organized as follows. In Section 4.2 we describe the basic concepts for our contribution in this chapter. In Section 4.3 we elaborate on the proposed algorithm and its components. Section 4.4 describes the experiments and results obtained. Finally, this chapter is summarised in Section 4.5.

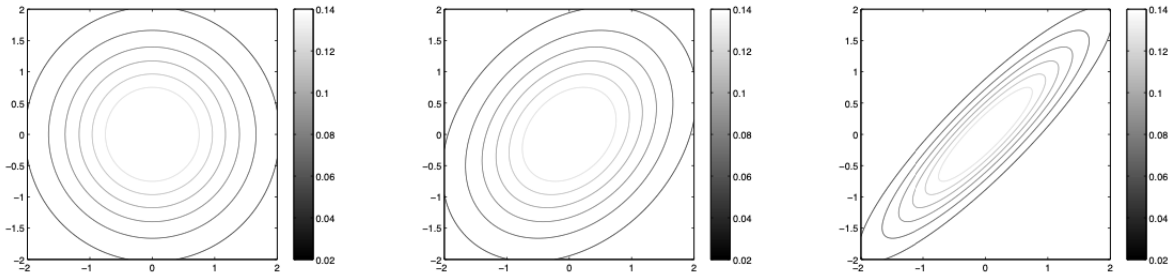
4.2 Preliminaries

EDAs [116], [117] form a new family of EAs, which generate offspring according to a probability distribution and have been intensively studied in the context of single optimization. A general framework of EDAs is outlined in Algorithm 10.

Algorithm 10 EDA

1: Input: population size NP, the number of selected individuals K
2: Randomly initialize the population
3: While the termination criteria is not satisfied
4: Select K best individuals from the population
5: Estimate the probability distribution of the population according to the selected individuals
6: Sample new individuals according to the estimated distribution
7: Combine the sampled individuals and the old population to create a new population with NP individuals
8: Output: the best individual and its fitness

EDAs are capable of capturing the correlation between the dimensions of the population members via covariance matrix. Figure 4.1 shows density plot for the different correlation values for 2D Gaussian distribution. In every sub-figure of Figure 4.1, each of the two variables has a standard deviation equals to 1, so here the correlation coefficient equals to the covariance. It can be seen that if the dimensions are not correlated the contour plot is circular (Figure 4.1(a)). As, the correlation between the



(a) correlation = 0 (b) correlation = 0.3 (c) correlation = 0.9

Figure 4.1: Demonstrations of 2D Gaussian distributions with different correlation coefficients.

The contours denote the Gaussian densities.

dimensions increases, the contour plot becomes more elliptical (Figure 4.1(c)). EDAs have achieved success in both combinatorial optimizations [55], [56] and continuous optimization domains [118], [119], [120] [121] [122].

4.3 EDADE for solving MMOPs

In this section, we present the EDA based DE method for solving MMOPs. The proposed method consists of five different components which are presented in the form of Algorithms. Algorithm 11 represents the generic framework used for solving MMOPs. Algorithm 2 (chapter 3), Algorithm 4 (chapter 3), Algorithm 9 (chapter 3), Algorithms 12 and 13 are further components of Algorithm 11. The proposed method is tested on the IEEE CEC 2013 benchmark functions. In the next section, we describe the multi-species framework used for solving MMOPs.

4.3.1 Multi-Species Framework

As discussed in section section 3.3.1, in multi-species framework, the idea is to divide the randomly distributed population into sub-populations called species and perform the evolutionary process on each species independently. The block diagram represented by Figure 4.2 broadly depicts the components of EDADE Algorithm. In

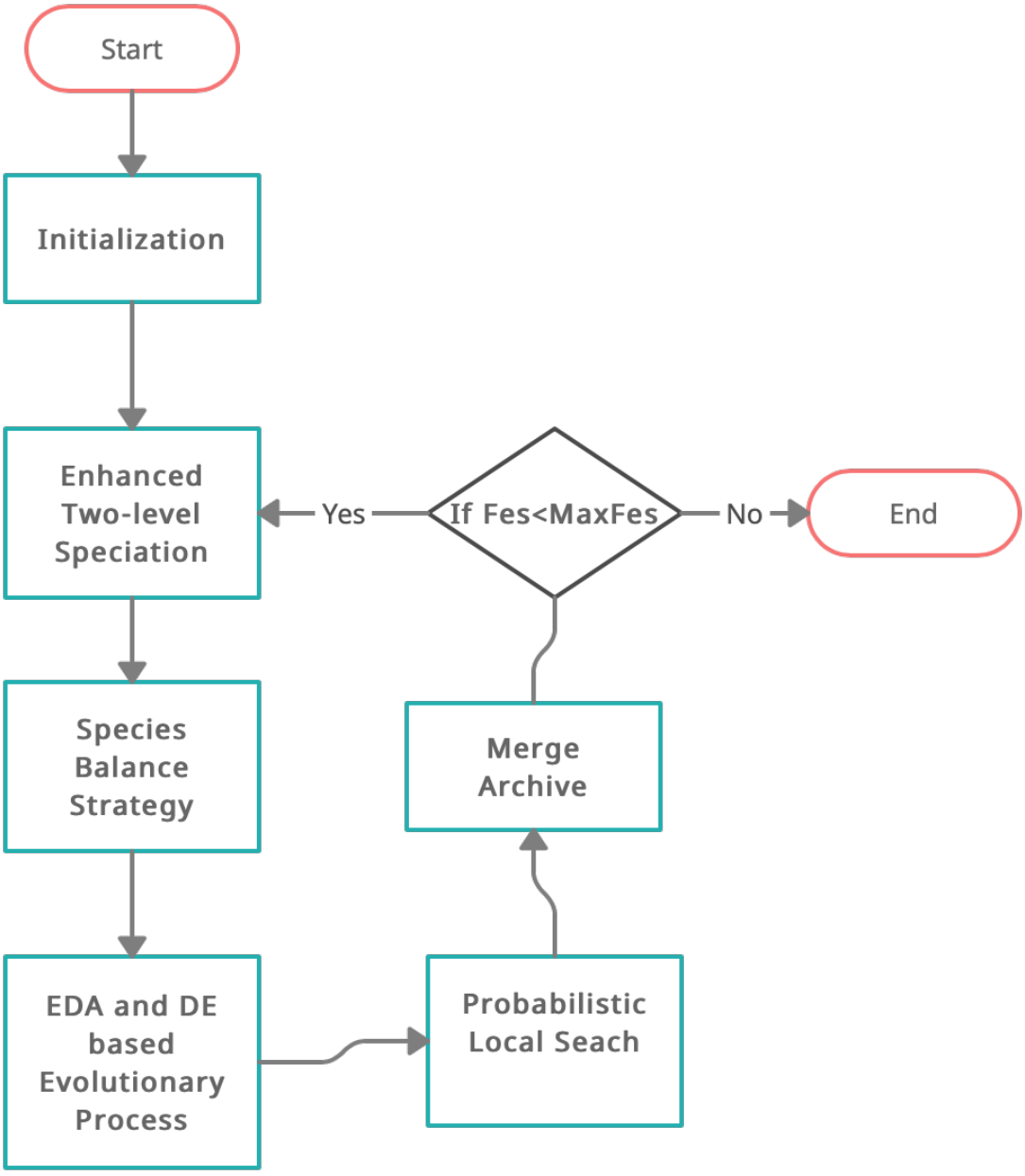


Figure 4.2: Block Diagram of EDADE Algorithm

Figure 4.2, initialization is the process of randomly assigning values to the population within bounds and it corresponds to step 2 of Algorithm 11. Enhanced two-level speciation is used to locate multiple niches in the landscape. Enhanced two-level speciation corresponds to section 4.3.1.1 and step 5 of Algorithm 11. Species balance strategy is employed to redistribute population members across multiple species. It corresponds

to section 3.3.1.2 and step 6 of Algorithm 11. We use evolutionary process to evolve the species that have population members generated by EDA and DE operations. It corresponds to section 4.3.1.3 and step 8 of Algorithm 11. Probabilistic local search is performed on the evolved species to improve the accuracy of the best member in the species. It corresponds to section 4.3.1.4 and step 9 of the Algorithm 11. Merge Archive is employed to avoid the insertion of duplicate peaks in the archive to be able to locate different peaks. It corresponds to section 4.3.1.5 and step 10 of Algorithm 11. Fes represents the current count of fitness evaluations while MaxFes represents the count of maximum fitness evaluations allowed. The procedures such as enhanced two-level speciation, species balance strategy, EDA and DE based evolutionary process, probabilistic local search and merge archive keep on running until the current fitness count becomes greater than or equal to maximum count of fitness evaluations allowed. Algorithm 11 represents the generic framework which further consists of five other algorithms. It is similar to the multi-species framework mentioned in chapter 3. However, some of the components are different in Algorithm 11. In algorithm 11,

Algorithm 11 EDADE

1: **Input**: Function ($f(\vec{X})$),Population Size (NP), $MaxFes$
2: Initialize the population randomly within the bounds of the dimensions
3: $Gen = 0$,$Fes = 0$,$archive = [\,]$
4: **while** $Fes \leqslant MaxFes$ **do**
5: Obtain multiple species by two-level application of Algorithm 2
6: balanceSpecies(*multi-species*)
7: **for** each $species \in multi\text{-}species$ **do**
8: $localbest$=Evolutionary Process of EDADE($species, speciesfitness$)
9: $bestfit$=probabilisticlocalSearch($localbest$,$species$)
10: mergeArchive($archive$,$bestfit$)
11: **end for**
12: Gen+=1
13: **end while**

the population is initialized randomly within the bounds specified for each dimension. The parameters *Gen*, *Fes*, *archive* represents generation number, number of fitness evaluations, and archive, respectively. The archive is used to store optimum values. Steps 5-10 are performed until maximum fitness evaluations are reached. Algorithm 2 (chapter 3) is invoked for creating multiple species out of the global population in step 5. The strategy for speciation is discussed in 4.3.1.1. After the formation of multiple species, there is a need to balance the species as each species could be representing different niches in the function landscape. To this aim, step 6 is introduced which balances the species based on the shape and size of niches. Now, for each species that is present in multi-species, steps 8-10 are carried out. In step 8, evolutionary process of EDADE (section 4.3.1.3) is applied to each species and returns the local best for that species. Probabilistic local search (section 4.3.1.4) is a method to refine the accuracy of the best solution obtained so far. Step 9 tries to find the better solution nearby the *localbest* achieved. It may so happen that the *localbest* obtained is very very close to one of the optima already present in the archive and both are part of the same niche. Such occurences inhibits the capability of the algorithm to find multiple optima. Hence, there is a need to check the redundancy of the optimizers obtained. To this aim, Step 10 is introduced. In the following subsection we describe the two-level speciation procedure using the application of Algorithm 2 on the global population.

4.3.1.1 Enhanced Two-level speciation

In this section we describe the the two-level speciation process which is mostly similar to that of chapter 3. This procedure maps to the step 5 of Algorithm 11. The population is initialized randomly across the function landscape initially. The speciation process is similar to that of EODE and it is described in chapter 3 section 3.2.1.1. However, we have proposed a small modification in the calculation of μ_{dist} parameter used in Algorithm 2. In the complex multimodal functions, the peaks could be unevenly located. The niches could be very close in some regions and very distant in other regions. Since μ_{dist} is a sensitive parameter in the creation of niches, it can be very large or very low if some niches are very distant or very close as it strongly

impacts the mean calculation. Hence, median distance is used to reduce the effect of the region of distant niches and densely located niches on speciation. In the next section, we discuss the species balance strategy used in EDADE.

4.3.1.2 Species balance strategy

The species balance strategy corresponds to the step 6 of Algorithm 11. The species balance strategy is mostly similar to that of EODE algorithm described in chapter 3. However, certain modifications are introduced with respect to the behaviour in EDADE environment. Species seed is the individual with the best fitness value in the sub-population. To estimate the distribution of the species a good enough species size is required since as the population size increases the estimation of probability distribution improves. Also, DE requires a minimum population size to be able to evolve the species efficiently. Hence, after multi-species formation, species balancing is required to maintain a minimal sub-population size in the species. However, covariance matrix can be approximated by univariate normal distribution which is computationally less intensive. Hence, there is a modification in step 15 of Algorithm 4 to enable it for use in EDADE environment. Instead of using covariance matrix for generating offspring, we have used Gaussian and Cauchy distribution alternatively with mean as species seed and variance (var) for each dimension is computed with the equation (4.1).

$$var[d] = random(0.1, 0.3); d \in dimensions \tag{4.1}$$

In the next section, we discuss the core evolutionary process behind the EDADE algorithm.

4.3.1.3 EDA and DE based Evolutionary Process of EDADE

In this section, we discuss the evolutionary process of the EDADE algorithm and it maps to the step 8 our main algorithm i.e. Algorithm 11. After balancing the species, the evolution process begins to exploit the niche and reach the optima within that niche. DE generates the offspring by first creating the mutants using differential perturbation and then applying crossover operation to the mutant. It does not

consider the correlation between the dimensions within the niche landscape and hence it is not effective in generating suitable offspring which could help reach optima fast. To deal with this drawback, we have proposed a hybrid methodology for generating offspring. The offsprings are generated using DE mutation operation as well as Gaussian distribution. The complete method is described in algorithm 12. The meaning of variables $FES, MaxFES, Gen, MaxGen, \vec{X}, random(a, b), \vec{U}$ can be referred from the section 3.2.3. Another parameter is pr that represents the probability that indicates the stage of evolution. During the early phase, exploration needs to be higher, and hence pr value will be low. For the 50% of the time algorithm focuses on exploration and for the rest 50%, it focuses on exploitation. This is done to maintain a balance between exploration and exploitation. The same can be seen from steps 7-17. The perturbation (trial vector generation) is done using "DE/rand/1", "DE/rand/2", a simple modification of "DE/current-to-rand/1" and "DE/current-to-rand/2" [40]. To induce convergence, four members with the best fitness values are chosen randomly from the k best members of the species. The value of k is determined by equation (4.2) where $size$ represents the species size.

$$k = size * (1 - \frac{\exp(\frac{Gen}{MaxGen})}{1 + \exp(\frac{Gen}{MaxGen})}) \tag{4.2}$$

While exploitation, if the absolute difference between fitness values of two members is lower than 0.00001 then random small perturbation is added to enable effective convergence. Besides, "DE/best/1" and "DE/best/2" are used probabilistically alternatively to generate trial vector generation. Steps 19-26 indicate the exploitation stage in DE. $\vec{lb}$ represents the member with the best fitness value in the species. Steps 27-31 represent crossover and selection operation. After applying DE operations on the population, EDA has been applied on the modified species obtained by DE. Essentially, we apply exploration operations for 50% of the time and exploitation for the rest of the time. The variance is required for offspring generation for 1-D(dimensional) functions while and covariance matrix is required for n-D(dimensional) functions. The covariance of two individuals (X and Y) of D dimensions is computed using equation

Algorithm 12 Evolutionary Process of EDADE

Input : $species, speciesfitness$

$F_1 = 1, F_2 = 1, CR = random(0.7, 1)$

While FES<$MaxFES$ and Gen<$MaxGen$

  $pr = FES/MaxFES$

  For each member-index (i)$\in$ species

    Randomly sample 5 members($\vec{X}_{r1}, \vec{X}_{r2}, \vec{X}_{r3}, \vec{X}_{r4}, \vec{X}_{r5}$) from the species

    if $pr \leqslant 0.5$

      if $random(0, 1) \leqslant 0.5$

        if $random(0, 1) \leqslant 0.5$

          $\vec{V} = \vec{X}_{r1} + F_1 * (\vec{X}_{r2} - \vec{X}_{r3})$

        else

          $\vec{V} = \vec{X}_{r1} + F_1 * (\vec{X}_{r2} - \vec{X}_{r3}) + F_2 * (\vec{X}_{r4} - \vec{X}_{r5})$

      else

        if $random(0, 1) \leqslant 0.5$

          $\vec{V} = \vec{X}_i + F_1 * (\vec{X}_{r2} - \vec{X}_{r3})$

        else

          $\vec{V} = \vec{X}_i + F_1 * (\vec{X}_{r2} - \vec{X}_{r3}) + F_2 * (\vec{X}_{r4} - \vec{X}_{r5})$

    else

      Select 4 members ($\vec{X}_{\{1,2,3,4\}}$) from the k best members randomly from the population.

      if $|fitness(\vec{X}_1) - fitness(\vec{X}_2)|$<0.00001 or $|fitness(\vec{X}_3) - fitness(\vec{X}_4)|$<0.00001

        if $random(0, 1)$< 0.4

          $\vec{X}_{\{1,2,3,4\}} = \vec{X}_{\{1,2,3,4\}} + random(0.1, 0.3)$

      if $random(0, 1) \leqslant 0.5$

        $\vec{V} = \vec{lb} + F_1 * (\vec{X}_1 - \vec{X}_2)$

      else

        $\vec{V} = \vec{lb} + F_1 * (\vec{X}_1 - \vec{X}_2) + F_2 * (\vec{X}_3 - \vec{X}_4)$

    Apply binomial crossover operation to get $\vec{U}$

    Check for the bounds of $\vec{U}$ in each dimension using equation (3.3)

    Evaluate the child fitness.

    FES+=1

    Perform selection operation between parent and child

32: if $random(0,1) \leqslant 0.5$

33: Compute the mean, variance (for 1-D functions), covariance matrix (for n-D functions) of the species.

34: for each member-index(i)$\in$ species

35: Generate offspring around $\vec{X}_i$ using univariate and multivariate Gaussian distribution for 1-D and n-D functions respectively.

36: else

37: Find the k best members of the species using equation (4.2)

38: Compute the mean, variance (for 1-D functions), covariance matrix (for n-D functions) of the k best members of the species using equation (3.7), (3.8) and (3.9)

39: Generate offspring around mean using univariate and multivariate gaussian distribution for 1-D and n-D functions respectively.

40: Perform selection operation between parent and offspring.

41: $c = 0$

42: for member-index(i) in species

43: if $|fitness(\vec{X}_i) - fitness(\vec{X}_{best})| < 0.00001$ and $|fitness(\vec{X}_{best})$-$fitness(\vec{X}_{gbest})| > 5$

44: $c+ = 1$

45: if $c > \frac{popsize}{2}$

46: break

47: if $c > \frac{popsize}{2}$

48: Add a random(0.1,0.5) perturbation to $\frac{popsize}{2}$ members.

49: break

50: $Gen+ = 1$

51: Return the best member of the species

(3.9). The covariance matrix of dimensions D*D is computed using equation (3.9). In the exploration stage, the offspring is generated in the vicinity of the parent to explore the space. Contrary to the exploration stage, mean, variance, and covariance matrix of best k members are calculated in the exploitation stage since algorithms tend to converge near the better candidate solutions. To this end, steps 32-39 are introduced in algorithm 12. The mean and variance are computed using equations (4.3) and (4.4):

$$\mu_i^d = \frac{1}{M}\sum_{j=1}^{M} X_j^d \tag{4.3}$$

$$\delta_i^d = \sqrt{\frac{1}{M-1}\sum_{j=1}^{M}\left(X_j^d - \mu_i^d\right)^2} \tag{4.4}$$

where $\mu_i = \left[\mu_i^1, \ldots, \mu_i^d, \ldots, \mu_i^D\right]$ and $\delta_i = \left[\delta_i^1, \ldots, \delta_i^d, \ldots, \delta_i^D\right] (1 \leqslant i \leqslant s)$ are, respectively, the mean and variance vectors of the i^{th} niche, $\boldsymbol{X}_j = \left[X_j^1, \ldots, X_j^d, \ldots, X_j^D\right]$ is the j^{th} individual in the i^{th} niche and D is the dimension size of the multimodal problem.

During the evolution process, the species is likely to stuck at the local optima and hence it becomes important to identify such scenarios to reduce the wastage of fitness evaluations and get effective exploration. To this aim, steps 42-49 are introduced. The key idea here is to keep checking the absolute difference between the fitness values of the best member and the given member and also to check the fitness difference between the species best member and the global best member of the entire population. $\vec{X}_{best}, \vec{X}_{gbest}$ represents the sub-population best fitness and global population best fitness value respectively. *popsize* represents the sub-population size.

In the next section we describe the local search mechanism to further improve the fitness of the best candidate.

4.3.1.4 Probabilistic Local search

A local search is performed to refine the obtained solutions. In this section a probabilistic approach is described and it maps to the step 9 of Algorithm 11. The probabilistic local search method is described in algorithm 13. We have computed

variance, covariance matrix for 1-D and n-D functions. Since the algorithm is in strong convergence phase, it is important to consider the members with better fitness in the population. Initially, the *mbest* members are calculated using step 3 where *speciessize* represents the species size. The *mbest* members estimate the parameters of the distribution in the region of exploitation that helps create better offsprings in the population. Steps 4-13 define the offspring generation process. If offspring is not better than the species seed then a random noise is added in the variance defined by step 18 in a hope to find better individual in the further broadened search region in the vicinity of the best candidate.

4.3.1.5 Merge Archive

To avoid the insertion of multiple best members from the same species into archive, mergeArchive procedure is used which is described in chapter 3 section 3.2.6.

To understand the EDADE on a broad level, the illustrative example discussed in chapter 3 section 3.3.2 can be referred. In the next section we describe the details of experiments and results.

4.3.2 Illustrative Example

In order to explain the components of EDADE, an example is presented in this section. We have considered low dimensional (2-D) composition function F11 described in Table A.1 of Appendix section for demonstration purpose.

As it can be seen from Figure 4.2, the population is randomly initialized in the function space. Now we will try to map the steps given in Algorithm 11 to the process that happens in Figures 4.2-4.4.

1. Initially, the randomly distributed population needs to be divided into sub-populations that represent regions around the peaks. The multiple species marked as black circles in Figure 4.3 are shown as S1-S7. In species S4, the blue circles represent two species that got located as a single species S4. To avoid such merging, two-level speciation is introduced. To this aim, step 5 of

Algorithm 13 probabilisticlocalSearch

1: Input :$localbest, species$
2: k=10
3: $mbest$=max($speciessize$/4,10)
4: Compute the variance (for 1-D functions), covariance matrix (for n-D functions) of the $mbest$ candidates of species.
5: While(k)
6: $mean$=$localbest$
7: if $uniform(0,1) \leqslant 0.5$
8: if $uniform(0,1) \leqslant 0.5$
9: Generate offspring using normal distribution around mean with variance computed in step 4.
10: else
11: Generate offspring using cauchy distribution around mean with variance computed in step 4..
12: else if dimension >1
13: Generate offspring using multivariate gaussian distribution around mean with covariance matrix computed in step 4.
14: if $fitness(offspring)$>$fitness(localbest)$
15: Replace the $localbest$ with the offspring.
16: else
17: for d in dim:
18: $vars[d]$+=$random(0.001, 0.01)$
19: k=k-1
20: Return the $localbest$

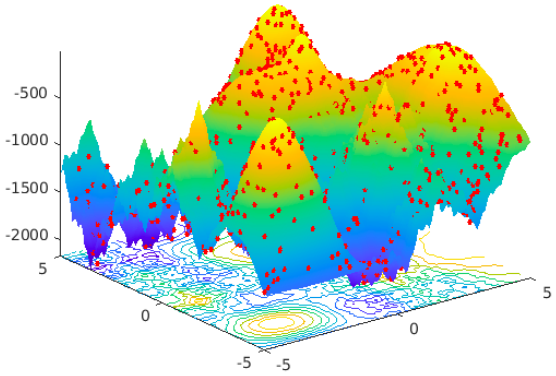


Figure 4.3: Initialization

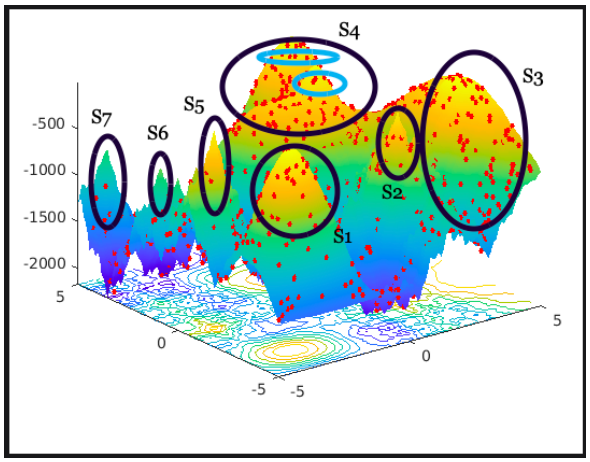


Figure 4.4: Niching and Balancing

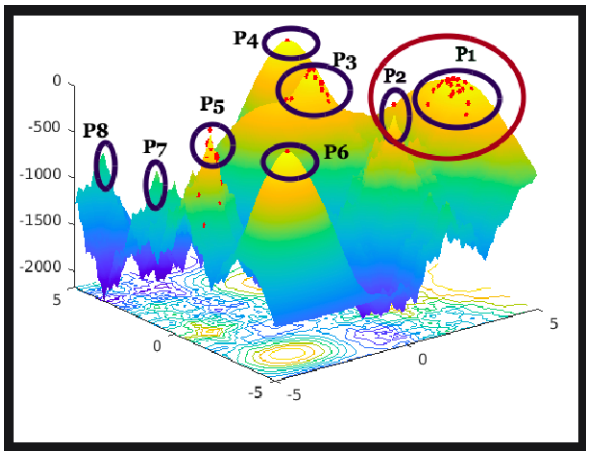


Figure 4.5: Evolutionary process and Local Search

Algorithm 11 is introduced that is further discussed in section 4.3.1.1.

2. As the multiple species are formed, some species may have huge population size and some may have small population size as can be seen from Figure 4.2. Hence, the species balance strategy introduced in section 3.3.1.2 is used with some modifications to it. We have used the Gaussian and Cauchy probability distributions to generate offsprings instead of covariance matrix as is used in section 3.3.1.2. The modification is further discussed in section 4.3.1.2.

3. The hybrid method based on EDA and DE operations acts as an evolutionary

process that tries to further converge the sub-populations towards the peak they represent. The species are represented in Figure 4.4 as P1-P8. Univariate and multivariate Gaussian distributions along with differential vector perturbation are used to generate offsprings.

4. After the evolution process completes, the population gets converged near the optima but may not be able to reach the optima accurately as can be seen in Figure 4.4. Hence, we proposed a local search method to search in the vicinity of the best individual of the species as the actual peak would be nearby the best value in the species. In this chapter, we have used a probabilistic local search method that aims to utilize Gaussian and Cauchy distributions with equal probability to generate offsprings near the species seed. The complete description is indicated by Algorithm 13.

5. Since the speciation process is repeated after every certain number of generations, the best solutions returned by the local search method could be part of the same species that has been explored in the previous generations. Hence, there is a need to detect the best individuals that belong to the same species and they should not be added to the archive. Therefore, Algorithm 9 from chapter 3 is referred to that maps to step 10 of Algorithm 11.

4.4 Experiments and Results

We have performed the experiments on a computer system with RAM 8GB, 1.8GHz CPU and MacOS 11 operating system. In this chapter, EDADE is independently run 50 times for each function. The algorithm calculated the results in the 1e-4 level of accuracy. For different problems, the value of NP is shown in Table 4.1. The parameters of EDADE are listed in Table 4.2. The parameters φ_1, φ_2, $minsize_1$, $minsize_2$ are used for creating multiple species out of the global population. δ is used in the species balance strategy and it is set to 1. $MaxGen$ represents the number of generations for which the EDADE needs to run for the given species and it is set to

40 for problems with dimension less than equal to 10 and 60 for higher-dimensional problems. F_1, F_2 are set to 1 while CR is chosen between 0.7 and 1. The PRs highlighted in bold represent the best PR value for that particular function. The bprs in the last row of the tables represent the number of functions for which the given algorithm performed the best. The algorithm is shown to perform better as compared to EODE for problems with a large number of optima. The experimental result and comparison with other algorithms at accuracy level 1e-4 are shown in tables 4.3-4.6.

4.4.1 Comparison with other algorithms

In this section, the results of various algorithms (including EDADE) are compared. For simplicity, we compare the results at $\epsilon = 1e{-}4$. In order to better evaluate the performance of EDADE, 16 popular comparison algorithms are selected, such as CDE [28], SDE [64], NCDE, NSDE [89], MOMMOP [112], LoICDE, LoISDE [35], PNPCDE [46], LIPS [82], and the recently proposed algorithms of DE_{cl} [113], LMCEDA, LMSEDA [9], FBK-DE [39], LBPADE [100], MaHDE [114], AED-DDE [123], EODE(proposed in chapter 3). Most of the recently compared results are from their corresponding papers, and the results of other algorithms are still from these papers. From Tables 4.3-4.6 it is clear that EDADE obtains most of the best PR results among the compared algorithms. The detailed analyses are given below.

1) For the first five problems, most algorithms including EDADE can find all global optimal solutions.

2) For the problems 6–9 (with a large number of global peaks), EDADE stands as the second-best algorithm and it finds most of the optima. In addition, for the problem 10, EDADE can find all the optimal solutions. For the 7^{th} problem having 36 optima, EDADE can find 92% of the optima. For the problems 8-9 having 81, 216 optima respectively, EDADE can find 95%, 65% optima.

3) For problems 11,12,13,14 and 15 that are low-dimensional composition functions, EDADE obtained the best results for problems 11,12, and 15. Although on the 13^{th}, function EDADE does not produce optimal results, the difference between EDADE and the best algorithm is small (around 7%).

Index	Function	NKP	Peak height	r	MaxFEs	NP
1	$F_1(1D)$	2	200.0	0.01	5.0E + 4	250
2	$F_2(1D)$	5	1.0	0.01	5.0E + 4	250
3	$F_3(1D)$	1	1.0	0.01	5.0E + 4	250
4	$F_4(2D)$	4	200.0	0.01	5.0E + 4	250
5	$F_5(2D)$	2	1.03163	0.5	5.0E + 4	250
6	$F_6(2D)$	18	186.731	0.5	2.0E + 5	2000
7	$F_7(2D)$	36	1.0	0.2	2.0E + 5	2000
8	$F_6(3D)$	81	2709.0935	0.5	4.0E + 5	2000
9	$F_7(3D)$	216	1.0	0.2	4.0E + 5	3000
10	$F_8(2D)$	12	-2.0	0.01	2.0E + 5	1000
11	$F_9(2D)$	6	0	0.01	2.0E + 5	1000
12	$F_{10}(2D)$	8	0	0.01	2.0E + 5	1000
13	$F_{11}(2D)$	6	0	0.01	2.0E + 5	1000
14	$F_{11}(3D)$	6	0	0.01	4.0E + 5	1000
15	$F_{12}(3D)$	8	0	0.01	4.0E + 5	1000
16	$F_{11}(5D)$	6	0	0.01	4.0E + 5	1000
17	$F_{12}(5D)$	8	0	0.01	4.0E + 5	2000
18	$F_{11}(10D)$	6	0	0.01	4.0E + 5	1000
19	$F_{12}(10D)$	8	0	0.01	4.0E + 5	1000
20	$F_{12}(20D)$	8	0	0.01	4.0E + 5	1000

Table 4.1: Information of the benchmark problems and the population size

Parameters	Values
φ_1	2
φ_2	1
$minsize_1$	-1
$minsize_2$	5
$F1$	1
$F2$	1
CR	$(0.7, 1)$
$MaxGen$	$40(D <= 10)$
	$60(D>10)$

Table 4.2: Parameters in EDADE

4) For the composition functions (in the relatively high dimensions) 16-20, EDADE achieves the best results for the 18^{th}, 19^{th} functions. However, for the 16^{th} function, the difference between EDADE and the best algorithm result is around 7%. For the 17^{th} problem, the difference between EDADE and the best algorithm is around 2%. For the 18^{th}, 19^{th} problems EDADE performs equivalent to the best algorithm.

4.4.2 Different Parameters of EDADE

4.4.2.1 Different values of F_1, F_2

Here, the different values of F_1, F_2 are compared. In Section 4.3.1.3, F_1, F_2 are used to generate offsprings via differential vector perturbation. Thus, we set (F_1, F_2) to three different values (1,1), (0.5,0.5), and (2,2) respectively. From Table 4.7 it can be seen that for $F_1 = 1, F_2 = 1$, EDADE achieves the best overall results. For functions 1-5, EDADE with $F_1 = 2, F_2 = 2$ locates all the optima as is the case with $F_1 = 1, F_2 = 1$. For $F_1 = 0.5, F_2 = 0.5$, EDADE achieves best result for function 7 and 9 which have 36, 216 optima respectively. The reason for better performance over massive multimodal functions could be a lower mutation factor that helps in better convergence. For functions 10-12 EDADE with $F_1 = 0.5, F_2 = 0.5$ and $F_1 = 1, F_2 = 1$

Function	EDADE		CDE		SDE		NCDE		NSDE		MOMMOP	
Index	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR
1	**1**	1	**1**	1	0.657	0.373	**1**	1	**1**	1	**1**	1
2	**1**	1	**1**	1	0.737	0.529	**1**	1	0.776	0.667	**1**	1
3	**1**	1	**1**	1	**1**	1	**1**	1	**1**	1	**1**	1
4	**1**	1	**1**	1	0.284	0	**1**	1	0.240	0	**1**	1
5	**1**	1	**1**	1	0.922	0.843	**1**	1	0.745	0.490	**1**	1
6	**1**	1	**1**	1	0.056	0	0.305	0	0.056	0	**1**	1
7	0.927	0.4	0.861	0	0.054	0	0.873	0	0.053	0	**1**	1
8	0.950	0.1	0	0	0.015	0	0.001	0	0.013	0	**1**	1
9	0.648	0	0.474	0	0.011	0	0.461	0	0.006	0	**1**	1
10	**1**	1	**1**	1	0.147	0	0.989	0.863	0.098	0	**1**	1
11	**1**	1	0.330	0	0.314	0	0.729	0.059	0.248	0	0.716	0.020
12	**1**	1	0.002	0	0.208	0	0.252	0	0.135	0	0.939	0.549
13	0.93	1	0.141	0	0.297	0	0.667	0	0.225	0	0.667	0
14	0.667	0	0.026	0	0.216	0	0.667	0	0.190	0	0.667	0
15	0.75	0	0.005	0	0.108	0	0.319	0	0.125	0	0.618	0
16	0.667	0	0	0	0.108	0	0.667	0	0.170	0	0.650	0
17	0.667	0	0	0	0.076	0	0.250	0	0.108	0	0.505	0
18	**0.667**	0	0.167	0	0.026	0	0.500	0	0.163	0	0.497	0
19	**0.520**	0	0	0	0.105	0	0.348	0	0.098	0	0.223	0
20	0.250	0	0	0	0	0	0.250	0	0.123	0	0.125	0
bprs	11		7		1		5		2		10	

Table 4.3: Comparison with other algorithms on accuracy level 1e-4

Function	EDADE		LoICDE		LoISDE		PNPCDE		LIPS		DE_{cl}	
Index	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR
1	**1**	1	**1**	1	**1**	1	**1**	1	0.833	0.686	**1**	1
2	**1**	1	**1**	1	0.235	0.039	**1**	1	**1**	1	**1**	1
3	**1**	1	**1**	1	**1**	1	**1**	1	0.961	0.961	**1**	1
4	**1**	1	0.975	0.902	0.250	0	**1**	1	0.990	0.961	**1**	1
5	**1**	1	**1**	1	0.667	0.333	**1**	1	**1**	1	**1**	1
6	1	1	**1**	1	0.056	0	0.537	0	0.246	0	0.942	0.340
7	0.927	0.4	0.705	0.02	0.029	0	0.874	0	0.4	0	0.986	0.64
8	0.950	0.1	0	0	0.012	0	0	0	0.084	0	0.999	0.9
9	0.648	0	0.187	0	0.005	0	0.472	0	0.104	0	0.726	0
10	**1**	1	**1**	1	0.083	0	**1**	1	0.748	0	**1**	1
11	**1**	1	0.66	0	0.167	0	0.66	0	0.974	0.843	0.667	0
12	**1**	1	0.495	0	0.125	0	0	0	0.574	0	0.943	0.58
13	0.93	1	0.51	0	0.167	0	0.461	0	0.794	0.176	0.667	0
14	0.667	0	0.657	0	0.167	0	0.592	0	0.644	0	0.667	0
15	0.75	0	0.299	0	0.125	0	0.258	0	0.336	0	0.623	0
16	0.667	0	0.559	0	0.167	0	0	0	0.304	0	0.667	0
17	0.667	0	0.223	0	0.076	0	0	0	0.162	0	0.42	0
18	**0.667**	0	0.219	0	0.157	0	0.147	0	0.098	0	**0.667**	0
19	**0.520**	0	0.037	0	0.027	0	0	0	0	0	0.357	0
20	0.250	0	0.123	0	0.088	0	0	0	0	0	0.212	0
bprs	11		6		2		6		2		6	

Table 4.4: Comparison with other algorithms on accuracy level 1e-4

Function	EDADE		LMCEDA		LMSEDA		FBK-DE		LBPADE		MaHDE	
Index	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR
1	**1**	1	**1**	1	**1**	1	**1**	1	**1**	1	**1**	1
2	**1**	1	**1**	1	**1**	1	**1**	1	**1**	1	**1**	1
3	**1**	1	**1**	1	**1**	1	**1**	1	**1**	1	**1**	1
4	**1**	1	**1**	1	**1**	1	**1**	1	**1**	1	**1**	1
5	**1**	1	**1**	1	**1**	1	**1**	1	**1**	1	**1**	1
6	**1**	1	0.99	0.843	0.972	0.588	0.990	0.820	**1**	1	**1**	1
7	0.927	0.4	0.734	0	0.673	0	0.813	0	0.889	0	0.804	0
8	0.950	0.1	0.367	0	0.613	0	0.824	0	0.575	0	0.983	0.291
9	0.648	0	0.284	0	0.248	0	0.425	0	0.476	0	0.351	0
10	**1**	1	**1**	1	0.998	0.98	**1**	1	**1**	1	**1**	1
11	**1**	1	0.667	0	0.892	0.392	**1**	1	0.674	0	0.725	0.078
12	**1**	1	0.75	0	0.99	0.922	0.935	0.480	0.750	0	0.650	0
13	0.93	1	0.667	0	0.667	0	**1**	1	0.667	0	0.667	0
14	0.667	0	0.667	0	0.667	0	**0.907**	0.460	0.667	0	0.667	0
15	0.75	0	0.696	0	0.738	0	0.730	0	0.654	0	0.648	0
16	0.667	0	0.667	0	0.667	0	**0.707**	0	0.667	0	0.667	0
17	0.667	0	0.456	0	0.62	0	0.630	0	0.532	0	0.352	0
18	**0.684**	0	0.657	0	0.66	0	**0.667**	0	**0.667**	0	0.663	0
19	**0.520**	0	0.451	0	0.458	0	**0.520**	0	0.475	0	0.455	0
20	0.250	0	0.059	0	0.248	0	**0.450**	0	0.275	0	0.250	0
bprs	11		6		5		13		7		7	

Table 4.5: Comparison with other algorithms on accuracy level 1e-4

Function	EDADE		AED-DDE		EODE	
Index	PR	SR	PR	SR	PR	SR
1	**1**	1	**1**	1	**1**	1
2	**1**	1	**1**	1	**1**	1
3	**1**	1	**1**	1	**1**	1
4	**1**	1	**1**	1	**1**	1
5	**1**	1	**1**	1	**1**	1
6	**1**	1	**1**	1	0.995	0.9
7	0.927	0.4	0.838	0.039	0.805	0
8	0.950	0.1	0.747	0	0.845	0
9	0.648	0	0.384	0	0.505	0
10	**1**	1	**1**	1	**1**	1
11	**1**	1	**1**	1	**1**	1
12	**1**	1	**1**	1	0.975	0.8
13	0.93	1	0.686	0	**1**	1
14	0.667	0	0.667	0	0.8	0
15	0.75	0	0.637	0	**0.8**	0
16	0.667	0	0.667	0	**0.730**	0
17	0.667	0	0.375	0	**0.684**	0
18	**0.684**	0	0.654	0	**0.684**	0
19	**0.520**	0	0.375	0	**0.520**	0
20	0.250	0	0.250	0	0.250	0
bprs	11		9		13	

Table 4.6: Comparison with other algorithms on accuracy level 1e-4

Function	$F_1 = 1, F_2 = 1$		$F_1 = 0.5, F_2 = 0.5$		$F_1 = 2, F_2 = 2$	
Index	PR	SR	PR	SR	PR	SR
1	**1**	1	**1**	1	**1**	1
2	**1**	1	0.88	0.4	**1**	1
3	**1**	1	**1**	1	**1**	1
4	**1**	1	**1**	1	**1**	1
5	**1**	1	**1**	1	**1**	1
6	**1**	1	**1**	1	0.78	0
7	0.927	0.4	**0.938**	0	0.822	0
8	**0.950**	0.1	0.944	0	0.92	0
9	0.648	0	**0.650**	0	0.425	0
10	**1**	1	**1**	1	**1**	1
11	**1**	1	**1**	1	0.76	0.2
12	**1**	1	**1**	1	0.925	0.4
13	0.93	1	**1**	1	0.667	0
14	**0.667**	0	**0.667**	0	**0.667**	0
15	**0.75**	0	0.680	0	0.586	0
16	**0.667**	0	**0.667**	0	**0.667**	0
17	**0.667**	0	0.455	0	0.635	0
18	**0.684**	0	0.668	0	0.645	0
19	**0.520**	0	0.465	0	0.419	0
20	**0.250**	0	0.125	0	**0.250**	0
bprs	17		13		9	

Table 4.7: Results on different values of F_1, F_2 at accuracy level 1e-4

Function	$\varphi_1 = 2, \varphi_2 = 1$		$\varphi_1 = 0.5, \varphi_2 = 0.5$		$\varphi_1 = 1, \varphi_2 = 1$	
Index	PR	SR	PR	SR	PR	SR
1	**1**	1	**1**	1	**1**	1
2	**1**	1	0.96	0.8	**1**	1
3	**1**	1	**1**	1	**1**	1
4	**1**	1	**1**	1	**1**	1
5	**1**	1	**1**	1	**1**	1
6	**1**	1	**1**	1	**1**	1
7	**0.927**	0.4	0.895	0	0.905	0
8	**0.950**	0.1	0.927	0	0.910	0
9	0.648	0	**0.655**	0	0.625	0
10	**1**	1	**1**	1	**1**	1
11	**1**	1	**1**	1	**1**	1
12	**1**	1	0.925	0.4	0.948	0.6
13	0.93	1	0.833	0	**1**	1
14	**0.667**	0	**0.667**	0	**0.667**	0
15	**0.75**	0	0.667	0	0.706	0
16	**0.667**	0	**0.667**	0	**0.667**	0
17	**0.667**	0	0.595	0	0.610	0
18	**0.684**	0	0.672	0	0.625	0
19	**0.520**	0	0.495	0	0.454	0
20	**0.250**	0	**0.250**	0	**0.250**	0
bprs	18		11		12	

Table 4.8: Results on different values of φ_1, φ_2 at accuracy level 1e-4

achieves best results. Higher values of mutation factors lead to more divergence. For function 13, EDADE with $F_1 = 0.5, F_2 = 0.5$ achieves the best result. It can be seen from Table 4.7 that for functions 14-20, EDADE with $F_1 = 1, F_2 = 1$ achieves the best result.

4.4.3 Different values of φ_1, φ_2

Here, the different values of φ_1, φ_2 are compared. In Section 4.3.1.1, φ_1, φ_2 are used to form local sub-populations from global population. Thus, we set (φ_1, φ_2) to three different values (1,1), (0.5,0.5), and (2,1) respectively. From Table 4.8 it can be seen that for $\varphi_1 = 2, \varphi_2 = 1$, EDADE achieves the best overall results. For functions 1-5, EDADE with $\varphi_1 = 1, \varphi_2 = 1$ and $\varphi_1 = 2, \varphi_2 = 1$ achieves the best results. For function 6, EDADE with $\varphi_1 = 1, \varphi_2 = 1$, $\varphi_1 = 2, \varphi_2 = 1$ and $\varphi_1 = 0.5, \varphi_2 = 0.5$ achieves the best results. For functions 7-8 EDADE with $\varphi_1 = 2, \varphi_2 = 1$ achieves best results. For function 9, lower value of φ_1, φ_2 (0.5) is effective in locating large number of peaks. For functions 10-12, EDADE with $\varphi_1 = 2, \varphi_2 = 1$ achieves best results. However, EDADE with $\varphi_1 = 0.5, \varphi_2 = 0.5$ and $\varphi_1 = 1, \varphi_2 = 1$ also achieves best results for functions 10-11. For functions 13-20, EDADE with $\varphi_1 = 2, \varphi_2 = 1$ achieves best results except for function 13 as can be seen from Table 4.8. In essence, lower values of φ_1, φ_2 are effective in locating large number of optima. To locate optima within wider basin of attraction larger values of φ_1, φ_2 are effective, since the species size is large if values of φ_1, φ_2 are large.

In the next section, we visually describe the convergence of population towards the peaks of the functions.

4.4.4 Convergence of population

During evolution, the MMOPs require that the algorithm can locate many global optima simultaneously. The algorithm with good performance not only can maintain the global optima that have been identified (found) but also can continue to search for the other global optima that have not been found. To investigate the performance of EDADE on maintaining the identified optima, the solution distribution of some functions (i.e., F2, F4, F6, F7, F10, F11, F12, and F13), described in Appendix A, is presented on some specific generations. Figures 4.6-4.8 shows the solution distribution of F2 with different generations (i.e., the generations are 1, 2, and 3).

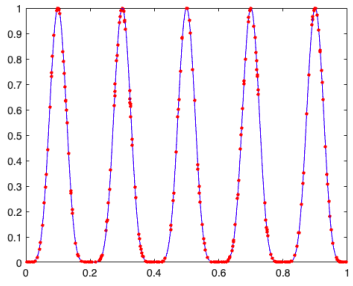


Figure 4.6: Gen-1 (F2)

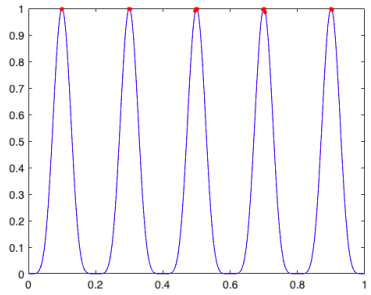


Figure 4.7: Gen-2 (F2)

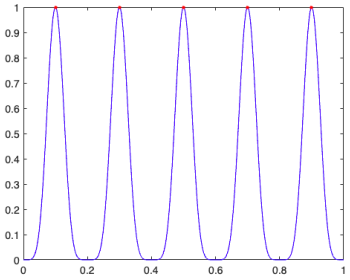


Figure 4.8: Gen-3 (F2)

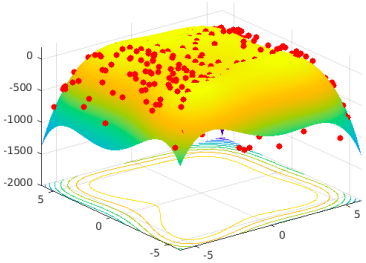


Figure 4.9: Gen-1 (F4)

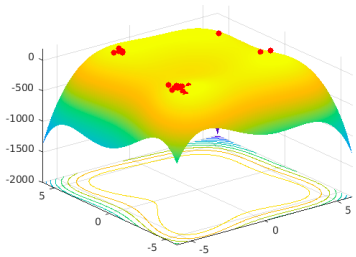


Figure 4.10: Gen-5 (F4)

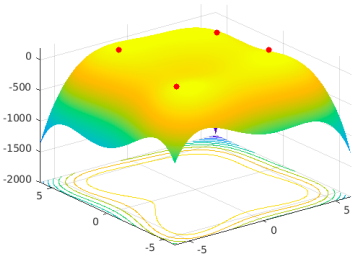


Figure 4.11: Gen-9 (F4)

We can find that EDADE has obtained all of the global optima when the generation is 3 on F2 [i.e., Figure 4.8]. It indicates that EDADE can locate the global optima quickly, namely, EDADE has a good ability of global search. It is important to note that the generation here refers to the number of times a multi-species framework is performed given in Algorithm 4. It does not refer to the generation of DE operations

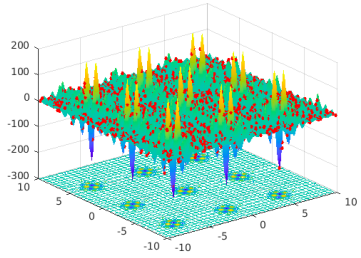


Figure 4.12: Gen-1 (F6)

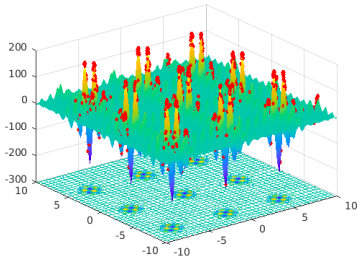


Figure 4.13: Gen-2 (F6)

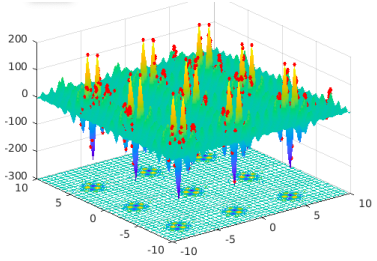


Figure 4.14: Gen-3 (F6)

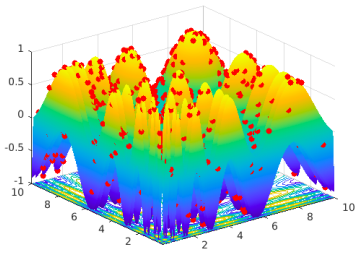


Figure 4.15: Gen-1 (F7)

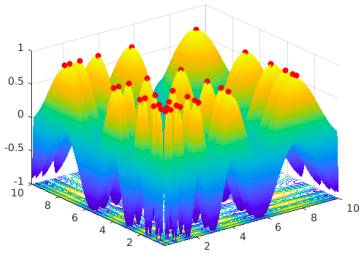


Figure 4.16: Gen-8 (F7)

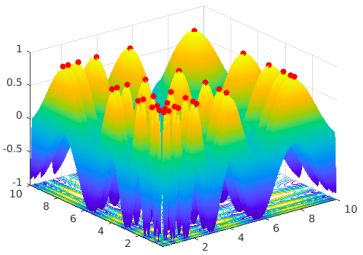


Figure 4.17: Gen-15 (F7)

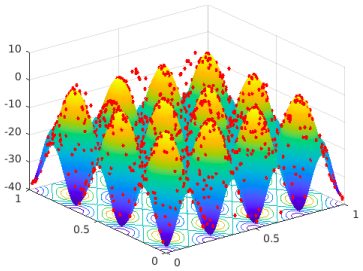


Figure 4.18: Gen-1 (F10)

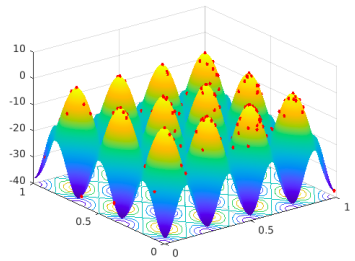


Figure 4.19: Gen-5 (F10)

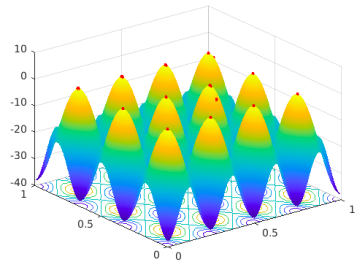


Figure 4.20: Gen-11 (F10)

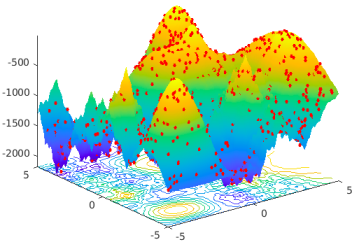


Figure 4.21: Gen-1 (F11)

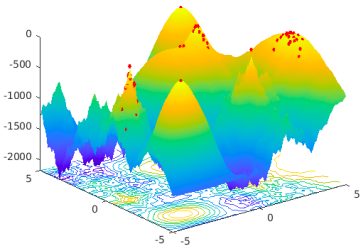


Figure 4.22: Gen-8 (F11)

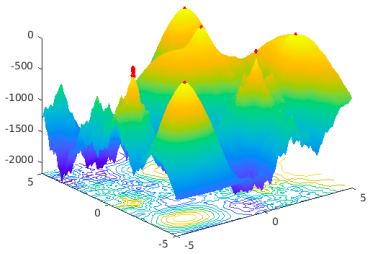


Figure 4.23: Gen-14 (F11)

directly. Figures 4.9-4.11 shows the solution distribution of F4 with different generations (i.e., 1, 5, and 9). From Figure 4.11, we can find that the population converges completely in the 9^{th} generation. Similarly, for F7, F10, EDADE obtained complete

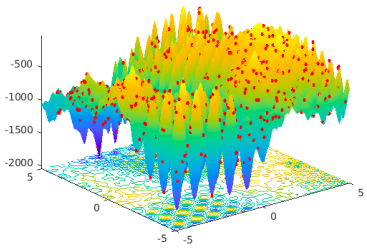


Figure 4.24: Gen-1 (F12)

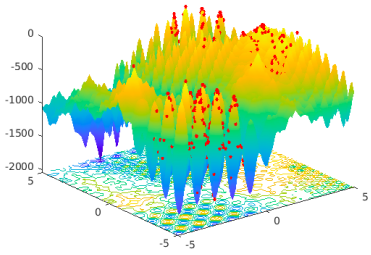


Figure 4.25: Gen-5 (F12)

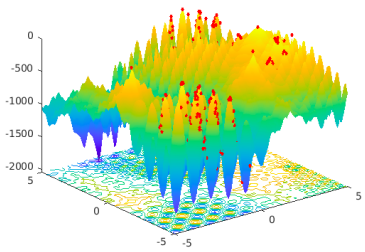


Figure 4.26: Gen-11 (F12)

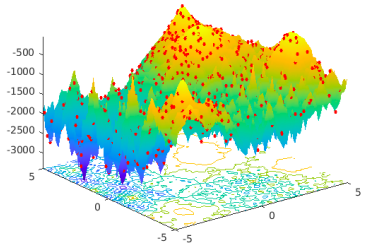


Figure 4.27: Gen-1 (F13)

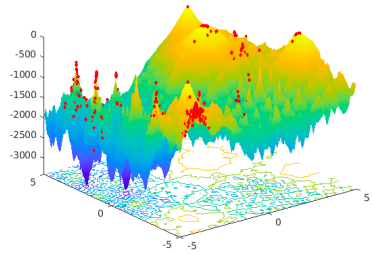


Figure 4.28: Gen-5 (F13)

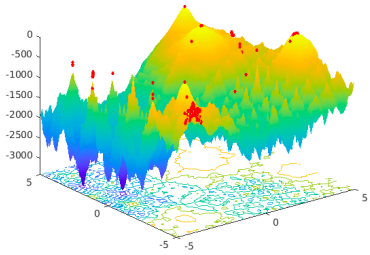


Figure 4.29: Gen-11 (F13)

convergence. However, for F6 (4.12-4.14), F11(4.21-4.23), F12(4.24-4.25), and F13 (4.27-4.29) there is some dispersion from the peak but still, the convergence is nearly complete. Hence, it can be concluded that with the increasing generation, all solutions gradually achieve the convergence state. Overall, it is clear that the found solutions are not dispersed with the increasing evolution in EDADE. Namely, our EDADE can maintain the global optima until the end of evolution.

In the next section we describe the summary of the chapter.

4.5 Summary

In this chapter, we have proposed EDADE to handle MMOPs, where multi-species technique has been adopted to simultaneously find multiple global optima. In EDADE, modified NBC-minsize has been used to divide the population into several species to locate the optima. Also, two-level speciation has been applied to identify the narrow niches. We have used the species balance strategy described in chapter 3 with some modifications to it. We have proposed an evolutionary method that evolves a hybrid population generated from DE operations and EDA operations. We have also proposed a probabilistic local search method to improve the accuracy of candidate solutions. To avoid the trap into local optima, we have proposed a simple strategy embedded in EDADE's evolutionary process. Finally, EDADE has been compared with several state-of-the-art algorithms to assess its effectiveness. The experimental results has shown that EDADE performed better on highly multimodal functions than other algorithms on most multimodal benchmark problems. However, in our experiments, we achieved better results for problems of lower dimensions but it still does not perform well for higher-dimensional problems. Hence, we aim to improve EDADE in our future work and test it on high-dimensional problems.

Chapter 5

Conclusion and Future Work

In this chapter, we summarize the techniques to solve MMOPs presented in the thesis and provide few directions for future work in this area. The goal of our work was to propose methods to solve MMOPs which can handle functions with the arbitrarily complex multimodal functional landscape.

MMOPs are widely seen in real-world scenarios, where the decision-making can be made based on multiple optimal solutions of a given optimization problem. Some of the real-world applications of MMOPs consist of virtual camera composition, metabolic network modeling, laser pulse shaping, job scheduling, data clustering, feature selection, and neural network ensembles. Hence, there is a need for efficient methodologies to solve MMOPs. We have described the problem of multimodal optimization. We have discussed DE and Non-DE-based approaches used in the literature to solve MMOPs. The existing methods are not efficient at handling the massive multimodality and complex functional landscapes. Hence, we have proposed two algorithms for solving problems with a large number of optima and uneven rugged landscapes.

5.1 Thesis Contributions

In this section, we summarize two methodologies proposed for solving MMOPs described in this thesis. In the first contribution, we have proposed an adaptive opposition DE-based approach to solve MMOPs while in the second contribution, we

have proposed EDA-based methodology to estimate the probability distribution of the species and generate better offsprings.

5.1.1 Enhanced Opposition Differential Evolution Algorithm for solving MMOPs

The proposed approach is an adaptive version of the DE algorithm to learn the parameters of the algorithm, explore and exploit the uneven complex multimodal landscapes efficiently. We have proposed a multi-species-based framework for solving MMOPs. Essentially, the core idea is to divide the global population into local sub-populations and evolve the sub-populations independently. To divide the global population, we have introduced two-level speciation to locate a maximum number of peak regions (niches or species) as accurately as possible. Since the species could be of uneven sizes based on the shape and size of the basin of attraction, it becomes important to balance the species to perform the evolution process effectively. Hence, to this end, we have introduced a dynamic species balance strategy. It is important to note that each sub-population (species) virtually represents an area near one of the peaks. The proposed algorithm tries to locate those areas first and then reach the optima represented by those regions. Since the landscapes of the functions could be rugged and uneven, it becomes a challenge to handle different landscapes using one specific methodology. Hence, an adaptive mechanism is needed to handle different landscapes differently. To this end, we introduced an adaptive parameter control strategy that computes the parameter values of next-generation (g+1) based on those parameter values of the previous generation (g) which produced better offsprings in the generation (g). The parameter values are used in the evolution process to introduce randomness in the population by creating diverse offspring. The modified opposition DE is used to perform the evolution process of the algorithm. Even after the evolution process, sometimes it is not guaranteed to reach the optima accurately. Hence, to deal with such a scenario local search-based methodology is proposed. Besides, a check is also performed to avoid considering the optima from the same species. The

proposed EODE is compared with the 15 state-of-the-art algorithms in the section 3.4.1. From the Tables 3.4-3.6, it can be concluded that EODE performs better than state-of-the-art algorithms.

5.1.2 Estimation of Distribution Algorithm (EDA) based Differential Evolution Algorithm for solving MMOPs

Our first Multi-Modal Optimization (MMO) method summarized in the previous sub-section is not able to handle the problems with massive multimodality. Hence, to improve the performance of problems with a large number of global optima without degrading the performance on complex composite functions, EDADE is proposed. The multi-species framework is followed as discussed in section 5.1.1 in this method. The massively multimodal functions have optima very closely spaced and hence most of the optima fall under a single species. This becomes a major reason for many optima to go undetected by the algorithm. Moreover, some functions may not be separable and hence it becomes difficult to generate offspring using DE operations as it consists of random perturbations only. However, a lower crossover rate is found to be useful for non-separable functions but it is difficult to capture the non-linear relationships amongst the dimensions of the target vector. Also, probability distribution helps in capturing the correlation between the dimensions that enables generation of the better offspring. Hence, there is a need to predict the parameters of the probability distribution. To this aim, we have proposed EDA based DE algorithm so that the best of both worlds are used in the evolution process to guide the sub-population to reach optima effectively. Gaussian and Cauchy distributions are used to generate offspring. Due to the presence of large number of local optima in the landscape, the population may get trapped at local optima due to high selection pressure. To deal with such scenarios, we have proposed a simple strategy that avoids the population from getting stuck at local optima. We have also proposed a probabilistic local search method to refine the obtained solutions. The proposed EDADE is compared with the 16 state-of-the-art algorithms in the section 4.4.1. It is also compared with the proposed

EODE (chapter 3) Algorithm. From the experiments and obtained results, it can be concluded that EDADE performs better at handling problems with a large number of optima without degrading the performance on complex composite functions.

5.2 Future Work

Our work on multimodal optimization can be extended in many ways. Following are the few possible extensions.

1. `Dynamic Population Size`: Since the population size impacts the evolution process, it would be an interesting exercise to dynamically increase or decrease the population size [34] based on the evolution stage and the shape and size of the species' landscape. If this exercise is implemented, then it would be interesting to study the convergence behavior of the algorithm.

2. `Application to Real-World Problems`: There are many real-world applications of multimodal optimization such as virtual camera composition [124], metabolic network modeling [125], laser pulse shaping [126], job scheduling [127], and neural network ensembles [128] e.t.c. It would be interesting work to apply the proposed algorithm to real-world datasets.

3. `Performance Improvement Over Higher-Dimensional Problems`: Since the proposed algorithm is not effective at locating optima in higher-dimensional problems, it becomes an open area of work to suggest modifications to the proposed algorithms to enhance their performance on high-dimensional multimodal functions.

Bibliography

[1] Q. Ling, G. Wu, and Q. Wang, "Restricted evolution based multimodal function optimization in holographic grating design," in Proc. IEEE Congr. Evol. Comput., Edinburgh, U.K., 2005, pp. 789–794.

[2] D.-K. Woo, J.-H. Choi, M. Ali, and H.-K. Jung, "A novel multimodal optimization algorithm applied to electromagnetic optimization," IEEE Trans. Magn., vol. 47, no. 6, pp. 1667–1673, Jun. 2011.

[3] K.-C. Wong, K.-S. Leung, and M.-H. Wong, "Protein structure predict- tion on a lattice model via multimodal optimization techniques," in Proc. Conf. Genet. Evol. Comput., Portland, OR, USA, 2010, pp. 155–162.

[4] M. Boughanem and L. Tamine, "A study on using genetic niching for query optimisation in document retrieval," in Advances in Information Retrieval. Heidelberg, Germany: Springer, 2002, pp. 135–149.

[5] E. C. Osuna and D. Sudholt, "Runtime analysis of crowding mechanisms for multimodal optimisation," IEEE Trans. Evol. Comput., to be published. doi: 10.1109/TEVC.2019.2914606.

[6] Y. H. Li, Z.-H. Zhan, S. J. Lin, J. Zhang, and X. N. Luo, "Competitive and cooperative particle swarm optimization with information sharing mechanism for global optimization problems," Inf. Sci., vol. 293, no. 1, pp. 370–382, Feb. 2015.

[7] J. Yao, N. Kharma, and P. Grogono, "Bi-objective multipopulation genetic algorithm for multimodal function optimization," IEEE Trans. Evol. Comput., vol. 14, no. 1, pp. 80–102, Feb. 2010.

[8] Q. Yang et al., "Adaptive multimodal continuous ant colony optimization," IEEE Trans. Evol. Comput., vol. 21, no. 2, pp. 191–205, Apr. 2017.

[9] Q. Yang, W.-N. Chen, Y. Li, C. L. P. Chen, X.-M. Hu, and J. Zhang, "Multimodal estimation of distribution algorithms," IEEE Trans. Cybern., vol. 47, no. 3, pp. 636–650, Mar. 2017.

[10] Y. L. Cao, H. Zhang, W. F. Li, M. C. Zhou, Y. Zhang, and W. A. Chaovalitwongse, "Comprehensive learning particle swarm optimization algorithm with local search for multimodal functions," IEEE Trans. Evol. Comput., to be published. doi: 10.1109/TEVC.2018.2885075.

[11] Z.-J. Wang et al., "Automatic niching differential evolution with contour prediction approach for multimodal optimization problems," IEEE Trans. Evol. Comput., to be published. doi: 10.1109/TEVC.2019.2910721.

[12] K. A. De Jong, "An analysis of the behavior of a class of genetic adaptive systems," Ph.D. thesis, Univ. Michigan, Ann Arbor, MI, USA, 1975.

[13] A. Petrowski, "A clearing procedure as a niching method for genetic algorithms," in Proc. 3rd IEEE Congr. Evol. Comput., Nagoya, Japan, 1996, pp. 798–803.

[14] J. Holland, Adaptation in Natural and Artificial Systems. Ann Arbor, MI, USA: Univ. of Michigan Press, 1975.

[15] A. D. Cioppa, C. D. Stefano, and A. Marcelli, "Where are the niches? Dynamic fitness sharing," IEEE Trans. Evol. Comput., vol. 11, no. 4, pp. 453–465, Aug. 2007.

[16] D. E. Goldberg and J. Richardson, "Genetic algorithms with sharing for multimodal function optimization," in Proc. 2nd Int. Conf. Genet. Algorithms, 1987, pp. 41–49.

[17] G. R. Harik, "Finding multimodal solutions using restricted tournament selection," in Proc. 6th Int. Conf. Genet. Algorithms, 1995, pp. 24–31

[18] J.-P. Li, M. E. Balazs, G. T. Parks, and P. J. Clarkson, "A species conserving genetic algorithm for multimodal function optimization," Evol. Comput., vol. 10, no. 3, pp. 207–234, 2002.

[19] D. Zaharie, "A multipopulation differential evolution algorithm for mul- timodal optimization," in Proc. 10th MENDEL Int. Conf. Soft Comput.,2004, pp. 17–24.

[20] X. Yin and N. Germay, "A fast genetic algorithm with sharing scheme using cluster analysis methods in multi-modal function optimization," in Proc. Int. Conf. Artif. Neural Nets Genet. Algorithms, 1993, pp. 450– 457.

[21] G. Dick and P. Whigham, "Spatially-structured sharing technique for multimodal problems," J. Comput. Sci. Technol., vol. 23, no. 1, pp. 64– 76, 2008.

[22] F. Caraffini, A. V. Kononova, and D. Corne, "Infeasibility and structural bias in differential evolution," Inf. Sci., vol. 496, pp. 161–179, Sep. 2019.

[23] M. Preuss, "Niching the CMA-ES via nearest-better clustering," in Proc. ACM 12th Annu. Conf. Companion Genet. Evol. Comput., 2010, pp. 1711–1718.

[24] X. Lin, W. Luo and P. Xu, "Differential Evolution for Multimodal Optimization With Species by Nearest-Better Clustering," in IEEE Transactions on Cybernetics, vol. 51, no. 2, pp. 970-983, Feb. 2021, doi: 10.1109/TCYB.2019.2907657.

[25] S. Das and P. N. Suganthan, "Differential evolution: A survey of the state-of-the-art," IEEE Trans. Evol. Comput., vol. 15, no. 1, pp. 4–31, Feb. 2011.

[26] M. Srinivas and L. M. Patnaik, "Genetic algorithms: A survey," Computer, vol. 27, no. 6, pp. 17–26, Jun. 1994.

[27] Mahfoud, Samir W, "Niching methods for genetic algorithms," Citeseer ,1995 .

[28] R. Thomsen, "Multimodal optimization using crowding-based differential evolution," in Proc. IEEE Congr. Evol. Comput., vol. 2. Portland, OR, USA, 2004, pp. 1382–1389.

[29] O. Mengsheol and D. Goldberg, "Probabilistic crowding: Deterministic crowding with probabilistic replacement," in Proc. GECCO, 1999, pp. 409–416.

[30] X. Li, "Efficient differential evolution using speciation for multi- modal function optimization," in Proc. Conf. Genet. Evol. Comput., Washington, DC, USA, 2005, pp. 873–880.

[31] C. Stoean, M. Preuss, R. Stoean, and D. Dumitrescu, "Multimodal optimization by means of a topological species conservation algorithm," IEEE Trans. Evol. Comput., vol. 14, no. 6, pp. 842–864, Dec. 2010.

[32] W. Gao, G. G. Yen, and S. Liu, "A cluster-based differential evolution with self-adaptive strategy for multimodal optimization," IEEE Trans. Cybern., vol. 44, no. 8, pp. 1314–1327, Aug. 2014.

[33] Y. Jie, N. Kharma, and P. Grogono, "Bi-objective multipopulation genetic algorithm for multimodal function optimization," IEEE Trans. Evol. Comput., vol. 14, no. 1, pp. 80–102, Feb. 2010.

[34] R. Tanabe, A. Fukunaga, "Improving the search performance of SHADE using linear population size reduction," 2014 IEEE Congress on Evolutionary Computation (CEC) (2014) 1658–1665.

[35] S. Biswas, S. Kundu, and S. Das, "Inducing niching behavior in differential evolution through local information sharing," IEEE Trans. Evol. Comput., vol. 19, no. 2, pp. 246–263, Apr. 2015.

[36] X. Li, "Niching without niching parameters: Particle swarm optimization using a ring topology," IEEE Trans. Evol. Comput., vol. 14, no. 1, pp. 150–169, Feb. 2010.

[37] S. Rahnamayan, H. R. Tizhoosh and M. M. A. Salama, "Opposition-Based Differential Evolution", in IEEE Transactions on Evolutionary Computation, vol. 12, no. 1, pp. 64-79, Feb. 2008, doi: 10.1109/TEVC.2007.894200.

[38] S. Guo and C. Yang, "Enhancing Differential Evolution Utilizing Eigenvector-Based Crossover Operator", in IEEE Transactions on Evolutionary Computation, vol. 19, no. 1, pp. 31-49, Feb. 2015, doi: 10.1109/TEVC.2013.2297160.

[39] X. Lin, W. Luo and P. Xu, "Differential Evolution for Multimodal Optimization With Species by Nearest-Better Clustering," in IEEE Transactions on Cybernetics, vol. 51, no. 2, pp. 970-983, Feb. 2021, doi: 10.1109/TCYB.2019.2907657.

[40] S. M. Islam, S. Das, S. Ghosh, S. Roy and P. N. Suganthan, "An Adaptive Differential Evolution Algorithm With Novel Mutation and Crossover Strategies for Global Numerical Optimization," in IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), vol. 42, no. 2, pp. 482-500, April 2012, doi: 10.1109/TSMCB.2011.2167966.

[41] R. Tanabe and A. Fukunaga, "Success-history based parameter adaptation for Differential Evolution," 2013 IEEE Congress on Evolutionary Computation, Cancun, Mexico, 2013, pp. 71-78, doi: 10.1109/CEC.2013.6557555.

[42] J. J. Liang, B. Y. Qu, P. N. Suganthan, and A. G. Hernández-Dıaz ,"Problem definitions and evaluation criteria for the cec 2013 special session on real-parameter optimization," Nanyang Technological University, Tech. Rep., 2013

[43] X. Li, A. Engelbrecht, and M. G. Epitropakis, "Benchmark functions for CEC'2013 special session and competition on niching methods for multimodal function optimization," Evol. Comput. Mach. Learn. Group, RMIT University, Rep., Melbourne, VIC, Australia, Rep., 2013.

[44] Q. Yang et al., "Multimodal estimation of distribution algorithms," IEEE Trans. Cybern., vol. 47, no. 3, pp. 636–650, Mar. 2017.

[45] Z.-J. Wang et al., "Dual-strategy differential evolution with affinity propagation clustering for multimodal optimization problems," IEEE Trans. Evol. Comput., vol. 22, no. 6, pp. 894–908, Dec. 2018.

[46] S. Biswas, S. Kundu, and S. Das, "An improved parent-centric mutation with normalized neighborhoods for inducing niching behavior in differential evolution," IEEE Trans. Cybern., vol. 44, no. 10, pp. 1726–1737, Oct. 2014.

[47] Zhu, L., Ma, Y. Bai, Y. "A self-adaptive multi-population differential evolution algorithm." Nat Comput 19, 211–235 (2020). https://doi.org/10.1007/s11047-019-09757-3

[48] S. Rahnamayan, H. R. Tizhoosh and M. M. A. Salama, "Opposition-Based Differential Evolution," in IEEE Transactions on Evolutionary Computation, vol. 12, no. 1, pp. 64-79, Feb. 2008, doi: 10.1109/TEVC.2007.894200.

[49] D.E. Goldberg, "Genetic Algorithms in Search, Optimization, and Machine Learning," Addison-Wesley, New York, 1989.

[50] O.J. Mengshoel, D.E. Goldberg, "The crowding approach to niching in genetic algorithms," Evolutionary Computation 16 (2008) 315–354.

[51] R.K. Ursem, "Multinational evolutionary algorithms," in: Proceedings of the Congress on Evolutionary Computation, vol. 3, 1999, pp. 1633–1640.

[52] R.K. Ursem, "Multinational GAs: multimodal optimization techniques in dynamic environments," in: Proceedings of the Second Genetic and Evolutionary Computation Conference, GECCO, Morgan Kaufmann, 2000.

[53] J. Kennedy, R.C. Eberhart, "Particle swarm optimization," in: Proc. IEEE Int. Conf. Neural Netw., ICNN, vol. 4, November 1995, pp. 1942–1948.

[54] J. Kennedy, R.C. Eberhart, Y. Shi, "Swarm Intelligence," Morgan Kaufmann, San Francisco, CA, 2001.

[55] A.P. Engelbrecht, "Fundamentals of Computational Swarm Intelligence," John Wiley Sons, 2006.

[56] Y. del Valle, G.K. Venayagamoorthy, S. Mohagheghi, J.-C. Hernandez, R.G. Harley, "Particle swarm optimization: basic concepts, variants and applications in power systems," IEEE Transactions on Evolutionary Computation 12 (2) (2008) 171–195.

[57] H.P. Schwefel: "Numerische optimierung von computer-modellen," Ph.D. Thesis. Reprinted by Birkhäuser, 1977.

[58] H.G. Beyer, H.-P. Schwefel, "Evolution strategies: a comprehensive introduction," Natural Computing 1 (1) (2002) 3–52.

[59] N. Hansen, A. Ostermeier, "Completely derandomized self-adaptation in evolution strategies," Evolutionary Computation 9 (2) (2001) 159–195.

[60] R. Storn, K.V. Price, "Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces," Journal of Global Optimization 11 (4) (1997) 341–359.

[61] K.V. Price, R. Storn, J. Lampinen, "Differential Evolution—A Practical Approach to Global Optimization," Springer, Berlin, 2005.

[62] S. Das, P.N. Suganthan, "Differential evolution—a survey of the state-of-the-art," IEEE Transactions on Evolutionary Computation 15 (1) (2011) 4–31.

[63] R. Thomsen, "Multimodal optimization using crowding-based differential evolution"" in: Proceedings of the Congress on Evolutionary Computation 2004, Portland, vol. 2, June 2004, pp. 1382–1389.

[64] X. Li, "Efficient differential evolution using speciation for multimodal function optimization," in: Proceedings of the Conference on Genetic and Evolutionary Computation, GECCO 2005, Washington DC, USA, 2005, pp. 873–880.

[65] B. Rigling, F. Moore, "Exploitation of subpopulations in evolutionary strategies for improved numerical optimization," in: Proceedings of the Eleventh Midwest Artificial Intelligence and Cognitive Science Conference, MAICS 1999, AAAI Press, 1999, pp. 80–88.

[66] J. Rumpler, F. Moore, "Automatic selection of subpopulations and minimal spanning distances for improved numerical optimization," in: Proceedings of the Congress on Evolutionary Computation, CEC 2001, vol. 1, 2001, pp. 38–43.

[67] D. Zaharie, "A multipopulation differential evolution algorithm for multimodal optimization," in: Proceedings of 10th MENDEL International Conference on Soft Computing, Brno, Czech Republic, June 2004, pp. 17–22.

[68] Z. Hendershot, "A differential evolution algorithm for automatically discovering multiple global optima in multidimensional discontinuous spaces," in: Proceedings of the Fifteenth Midwest Artificial Intelligence and Cognitive Sci- ences Conference, Chicago, April 2004, pp. 92–97.

[69] D. Zaharie, "Extensions of differential evolution algorithms for multimodal optimization," in: Proceedings of SYNASC'04, 6th International Symposium of Symbolic and Numeric Algorithms for Scientific Computing, 2004, pp. 523–534.

[70] K.V. Price, J. Rönkkönen, "Comparing the uni-modal scaling performanceee of global and local selection in mutation-only differential evolution algorithm," in: Proceedings of 2006 IEEE World Congress on Computational Intelligence, Vancouver, Canada, 16–21 July 2006, pp. 7387–7394.

[71] J. Rönkkönen, J. Lampinen, "On determining multiple global optima by differential evolution," in: Evolutionary and Deterministic Methods for Design, Optimization and Control, Proceedings of EUROGEN 2007, Jyyvaskyla, Finland, 11–13 June 2007, pp. 146–151.

[72] J. Rönkkönen, "Continuous multimodal global optimization with differential evolution-based methods," Ph.D. Thesis, Lappeenranta University of Technology, Lappeenranta, Finland, 2009.

[73] B.Y. Qu, P.N. Suganthan, "Differential evolution with neighborhood mutation for multi-modal optimization," IEEE Transactions on Evolutionary Computation (2011) (in press)

[74] L. Qing, W. Gang, Y. Zaiyue, W. Qiuping, "Crowding clustering genetic algorithm for multimodal function optimization," Applied Soft Computing 8 (2008) 88–95.

[75] F. Streichert, G. Stein, H. Ulmer, A. Zell, "A clustering based niching method for evolutionary algorithms," in: Proceedings of the Conference on Genetic and Evolutionary Computation, GECCO 2003, in: Lecture Notes in Computer Science, vol. 2723, Springer, 2003, pp. 644–645.

[76] E.L. Yu, P.N. Suganthan, "An ensemble of niching algorithms," Information Sciences 180 (15) (2010) 2815–2833. Elsevier.

[77] K. Deb, "Multi-Objective Optimization using Evolutionary Algorithms," John Wiley Sons, 2001.

[78] C.A. Coello Coello, G.B. Lamont, D.A. Van Veldhuizen, "Evolutionary Algorithms for Solving Multi-Objective Problems," Springer, 2007.

[79] K. Deb, A. Saha, "Multimodal optimization using a bi-objective evolutionary algorithm," KanGAL Report No. 2009006, IIT Kanpur, December 2009.

[80] K. Deb, A. Saha, "Finding multiple solutions for multimodal optimization problems using a multi-objective evolutionary approach," in: Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation, GECCO '10, ACM, New York, NY, Portland, Oregon, USA, 2010, pp. 447–454.

[81] Bonabeau E, Dorigo M, Theraulaz G. “Swarm Intelligence: From Natural to Artificial Systems,” Journal of Artificial Societies and Social Simulation. 1999;4: 320

[82] B. Y. Qu, P. N. Suganthan and S. Das, “A Distance-Based Locally Informed Particle Swarm Model for Multimodal Optimization,” in IEEE Transactions on Evolutionary Computation, vol. 17, no. 3, pp. 387-402, June 2013, doi: 10.1109/TEVC.2012.2203138.

[83] Sajjad Yazdani, Hossein Nezamabadi-pour, Shima Kamyab, “A gravitational search algorithm for multimodal optimization,Swarm and Evolutionary Computation,” Volume 14,2014,Pages 1-14, ISSN 2210-6502,https://doi.org/10.1016/j.swevo.2013.08.001.

[84] Cuevas, Erik Reyna Orta, Adolfo. “A Cuckoo Search Algorithm for Multimodal Optimization,” The Scientific World Journal. 2014. 27. 10.1155/2014/497514.

[85] Thirugnanasambandam, K., Prakash, S., Subramanian, V., “Reinforced cuckoo search algorithm-based multimodal optimization,” Appl Intell 49, 2059–2083 (2019). https://doi.org/10.1007/s10489-018-1355-3

[86] Y. Zhang, Y. Lin, Y. Gong and J. Zhang, “Particle Swarm Optimization with Minimum Spanning Tree Topology for Multimodal Optimization,” 2015 IEEE Symposium Series on Computational Intelligence, 2015, pp. 234-241, doi: 10.1109/SSCI.2015.43.

[87] J. Wang, “Enhancing Particle Swarm Algorithm for Multimodal Optimization Problems,” 2017 International Conference on Computing Intelligence and Information System (CIIS), 2017, pp. 1-6, doi: 10.1109/CIIS.2017.10.

[88] Beyer, Hans-Georg Sendhoff, Bernhard. “Covariance Matrix Adaptation Revisited – The CMSA Evolution Strategy,” 2008, –. 5199. 123-132. 10.1007/978-3-540-87700-4_13.

[89] Qu, B. Y., Suganthan, P. N., Liang, J. J., "Differential Evolution With Neighborhood Mutation for Multimodal Optimization," IEEE Transactions on Evolutionary Computation, 2012, 16(5), 601–614. doi:10.1109/tevc.2011.2161873

[90] Basak, A., Das, S., Tan, K. C. "Multimodal Optimization Using a Biobjective Differential Evolution Algorithm Enhanced With Mean Distance-Based Selection," IEEE Transactions on Evolutionary Computation, 2013, 17(5), 666–685. doi:10.1109/tevc.2012.2231685

[91] M. G. Epitropakis, V. P. Plagianakos and M. N. Vrahatis, "Multimodal optimization using niching differential evolution with index-based neighborhoods," 2012 IEEE Congress on Evolutionary Computation, 2012, pp. 1-8, doi: 10.1109/CEC.2012.6256480.

[92] M.G.Epitropakis, V.P.Plagianakos, and M.N.Vrahatis, "Finding multiple global optima exploiting differential evolution's niching capability", in 2011 IEEE Symposium on Differential Evolution (SDE), April 2011, pp. 1–8

[93] J. Zhang and A. Sanderson, "JADE: adaptive differential evolution with optional external archive," IEEE Transactions on Evolutionary Computation, vol. 13, no. 5, pp. 945–958, 2009.

[94] Z.Zhaiand, X.Li,"A dynamic archive based niching particle swarm optimizer using a small population size," in Proceedings of the Australian Computer Science Conference (ACSC 2011), M. Reynolds, Ed. Perth, Australia: ACM, 2011, pp. 1–7.

[95] Liang, J. J., Qu, B. Y., Mao, X. B., Niu, B., Wang, D. Y. "Differential evolution based on fitness Euclidean-distance ratio for multimodal optimization," Neurocomputing, 2014, 137, 252–260. doi:10.1016/j.neucom.2013.03.069

[96] Zhang, Y.-H., Gong, Y.-J., Chen, W.-N., Zhang, J. "Composite differential evolution with queueing selection for multimodal optimization," 2015 IEEE Congress on Evolutionary Computation (CEC), 2015, doi:10.1109/cec.2015.7256921

[97] Zhang, Y.-H., Gong, Y.-J., Chen, W.-N., Zhang, J. "Composite differential evolution with queueing selection for multimodal optimization," 2015 IEEE Congress on Evolutionary Computation (CEC), 2015, doi:10.1109/cec.2015.7256921

[98] Z. Wang et al., "Dual-Strategy Differential Evolution With Affinity Propagation Clustering for Multimodal Optimization Problems," in IEEE Transactions on Evolutionary Computation, vol. 22, no. 6, pp. 894-908, Dec. 2018, doi: 10.1109/TEVC.2017.2769108.

[99] Huang, H., Jiang, L., Yu, X., Xie, D. "Hypercube-Based Crowding Differential Evolution with Neighborhood Mutation for Multimodal Optimization," International Journal of Swarm Intelligence Research, 2018, 9(2), 15–27. doi:10.4018/ijsir.2018040102

[100] Zhao, H., Zhan, Z.-H., Lin, Y., Chen, X., Luo, X.-N., Zhang, J.,Zhang, J. "Local Binary Pattern-Based Adaptive Differential Evolution for Multimodal Optimization Problems," IEEE Transactions on Cybernetics, 2019, 1–15. doi:10.1109/tcyb.2019.2927780

[101] Wang, Z.-J., Zhan, Z.-H., Lin, Y., Yu, W.-J., Wang, H., Kwong, S., Zhang, J. "Automatic Niching Differential Evolution with Contour Prediction Approach for Multimodal Optimization Problems," IEEE Transactions on Evolutionary Computation, 2019, 1–1. doi:10.1109/tevc.2019.2910721

[102] B. J. Frey and D. Dueck, "Clustering by passing messages between data points," Science, vol. 315, no. 5814, pp. 972-976, 2007.

[103] Z. -J. Wang, Y. -R. Zhou and J. Zhang, "Adaptive Estimation Distribution Distributed Differential Evolution for Multimodal Optimization Problems," in IEEE Transactions on Cybernetics, doi: 10.1109/TCYB.2020.3038694.

[104] Gao, W., Yen, G. G., Liu, S. "A Cluster-Based Differential Evolution With Self-Adaptive Strategy for Multimodal Optimization," IEEE Transactions on Cybernetics, 2014, 44(8), 1314–1327. doi:10.1109/tcyb.2013.2282491

[105] Liu, Qingxue Du, Shengzhi Wyk, Barend Sun, Yanxia. "Niching particle swarm optimization based on Euclidean distance and hierarchical clustering for multimodal optimization," Nonlinear Dynamics., 2020, 99. 1-19. 10.1007/s11071-019-05414-7.

[106] S. Hui and P. N. Suganthan, "Ensemble and Arithmetic Recombination-Based Speciation Differential Evolution for Multimodal Optimization," in IEEE Transactions on Cybernetics, vol. 46, no. 1, pp. 64-74, Jan. 2016, doi: 10.1109/TCYB.2015.2394466.

[107] Wu, G., Mallipeddi, R., Suganthan, P. N., Wang, R., Chen, H. "Differential evolution with multi-population based ensemble of mutation strategies," Information Sciences, 2016, 329, 329–345. doi:10.1016/j.ins.2015.09.009

[108] Mallipeddi, Rammohan Suganthan, Ponnuthurai Pan, Quan-Ke Tasgetiren, Mehmet. "Differential evolution algorithm with ensemble of parameters and mutation strategies," Appl. Soft Comput., 2011, 11. 1679-1696.

[109] Wu, Guohua Mallipeddi, Rammohan Suganthan, Ponnuthurai. "Ensemble strategies for population-based optimization algorithms – A survey," Swarm and Evolutionary Computation., 2017, 44. 10.1016/j.swevo.2018.08.015.

[110] Z.-J. Wang et al., "Dual-strategy differential evolution with affinity propagation clustering for multimodal optimization problems," IEEE Trans. Evol. Comput., vol. 22, no. 6, pp. 894–908, Dec. 2018.

[111] Q. Yang et al., "Adaptive multimodal continuous ant colony optimization," IEEE Trans. Evol. Comput., vol. 21, no. 2, pp. 191–205, Apr. 2017.

[112] Y. Wang, H. Li, G. G. Yen, and W. Song, "MOMMOP: Multiobjective optimization for locating multiple optimal solutions of multimodal optimization problems," IEEE Trans. Cybern., vol. 45, no. 4, pp. 830–843, Apr. 2015.

[113] B. Boškovic and J. Brest, "Clustering and differential evolution for multimodal optimization," in Proc. IEEE Congr. Evol. Comput. (CEC), 2017, pp. 698–705.

[114] Z. Hong, Z. -G. Chen, D. Liu, Z. -H. Zhan and J. Zhang, "A Multi-Angle Hierarchical Differential Evolution Approach for Multimodal Optimization Problems," in IEEE Access, vol. 8, pp. 178322-178335, 2020, doi: 10.1109/ACCESS.2020.3027559.

[115] Z. -J. Wang, Y. -R. Zhou and J. Zhang, "Adaptive Estimation Distribution Distributed Differential Evolution for Multimodal Optimization Problems," in IEEE Transactions on Cybernetics, doi: 10.1109/TCYB.2020.3038694.

[116] M. Hauschild and M. Pelikan, "An introduction and survey of estimation of distribution algorithms," Swarm Evol. Comput., vol. 1, no. 3, pp. 111-128, 2011.

[117] P. Larrañaga and J. A. Lozano, "Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation," New York, NY, USA:Springer, 2002.

[118] M. Pelikan and H. Mühlenbein, "The bivariate marginal distribution algorithm" in Advances in Soft Computing, London, U.K.:Springer, pp. 521-535, 1999.

[119] J. S. De Bonet, C. L. Isbell and P. Viola, "MIMIC: Finding optima by estimating probability densities," Proc. Conf. Adv. Neural Inf. Process. Syst., pp. 424-430, 1997.

[120] A. Zhou, J. Sun and Q. Zhang, "An estimation of distribution algorithm with cheap and expensive local search methods," IEEE Trans. Evol. Comput., vol. 19, no. 6, pp. 807-822, Dec. 2015.

[121] C. W. Ahn, J. An and J.-C. Yoo, "Estimation of particle swarm distribution algorithms: Combining the benefits of PSO and EDAs," Inf. Sci., vol. 192, pp. 109-119, Jan. 2012.

[122] H. Karshenas, R. Santana, C. Bielza and P. Larranaga, "Multiobjective estimation of distribution algorithm based on joint modeling of objectives and variables," IEEE Trans. Evol. Comput., vol. 18, no. 4, pp. 519-542, Aug. 2014.

[123] Z. -J. Wang, Y. -R. Zhou and J. Zhang, "Adaptive Estimation Distribution Distributed Differential Evolution for Multimodal Optimization Problems," in IEEE Transactions on Cybernetics, doi: 10.1109/TCYB.2020.3038694.

[124] M. Preuss, P. Burelli, and G. N. Yannakakis, "Diversified virtual camera composition," in Proc. Eur. Conf. Appl. Evol. Comput., 2012, pp. 265–274.

[125] M. Kronfeld, A. Dräger, M. Aschoff, and A. Zell, "On the benefits of multimodal optimization for metabolic network modeling," in Proc. GCB, 2009, pp. 191–200.

[126] O. M. Shir, C. Siedschlag, T. Bäck, and M. J. J. Vrakking, "Niching in evolution strategies and its application to laser pulse shaping," in Proc. Int. Conf. Artif. Evol., 2005, pp. 85–96.

[127] E. Pérez, M. Posada, and F. Herrera, "Analysis of new niching genetic algorithms for finding multiple solutions in the job shop scheduling," J. Intell. Manuf., vol. 23, no. 3, pp. 341–356, Jun. 2012.

[128] C. Castillo, G. Nitschke, and A. Engelbrecht, "Niche particle swarm optimization for neural network ensembles," in Proc. Eur. Conf. Artif. Life, 2009, pp. 399–407.

Appendix A

Test Functions

In this Appendix, we describe the IEEE CEC 2013 benchmark functions that have been used as test functions in our thesis. There are 20 multimodal functions that are categorised in Table A.1 based on the dimensionality or complexity or no. of optima. Functions 1-3 are 1-D functions whereas functions 4-9 are 2-D functions with massive multimodality. Function 10 is modified rastrigin function that has no local optima but only global optima. Functions 11-15 are low dimensional (2-3 D) composition functions whereas functions 16-20 are high dimensional (5-20 D) composition functions.

Function-Index	Function Name	Dimensionality	No. Of Optima
1	Five-Uneven-Peak Trap	1	2
2	Equal Maxima	1	5
3	Uneven Decreasing Maxima	1	1
4	Himmelblau	2	4
5	Six-Hump Camel Back	2	2
6	Shubert	2	18
7	Shubert	3	36
8	Vincent	2	81
9	Vincent	3	216
10	Modified Rastrigin	2	12
11	Composition Function 1	2	6
12	Composition Function 2	2	8
13	Composition Function 3	2	6
14	Composition Function 3	3	6
15	Composition Function 4	3	8
16	Composition Function 3	5	6
17	Composition Function 4	5	8
18	Composition Function 3	10	6
19	Composition Function 4	10	8
20	Composition Function 4	20	8

Table A.1: IEEE CEC 2013 benchmark functions

Now, we present the mathematical formulations of the test functions that are defined in the Table A.1. 'D' represents the dimensionality of the problem.

1. **Five-Uneven-Peak Trap**

$$F_1(x) = \begin{cases} 80(2.5 - x) & \text{for } 0 \leqslant \text{x} < 2.5 \\ 64(x - 2.5) & \text{for } 2.5 \leqslant \text{x} < 5.0 \\ 64(7.5 - x) & \text{for } 5.0 \leqslant \text{x} < 7.5 \\ 28(x - 7.5) & \text{for } 7.5 \leqslant \text{x} < 12.5 \\ 28(17.5 - x) & \text{for } 12.5 \leqslant \text{x} < 17.5 \\ 32(x - 17.5) & \text{for } 17.5 \leqslant \text{x} < 22.5 \\ 32(27.5 - x) & \text{for } 22.5 \leqslant \text{x} < 27.5 \\ 80(x - 27.5) & \text{for } 27.5 \leqslant \text{x} \leqslant 30 \end{cases} \tag{A.1}$$

Properties:

Variable ranges: x ∈ [0,30]

No. of global optima: 2

No. of local optima: 3

2. **Equal Maxima**

$$F_2(x) = \sin^6(5\pi x) \tag{A.2}$$

Properties:

Variable ranges: x ∈ [0,1]

No. of global optima: 5

No. of local optima: 0

3. **Uneven Decreasing Maxima**

$$F_3(x) = \exp\left(-2\log(2)\left(\frac{x - 0.08}{0.854}\right)^2\right)\sin^6\left(5\pi\left(x^{3/4} - 0.05\right)\right) \tag{A.3}$$

Properties:

Variable ranges: x ∈ [0,1]

No. of global optima: 1

No. of local optima: 4

4. **Himmelblau**

$$F_4(x, y) = 200 - \left(x^2 + y - 11\right)^2 - \left(x + y^2 - 7\right)^2 \tag{A.4}$$

Properties:

Variable ranges: x, y ∈ [6,6]

No. of global optima: 4

5. **Six-Hump Camel Back**

$$F_5(x, y) = -4\left[\left(4 - 2.1x^2 + \frac{x^4}{3}\right)x^2 + xy + \left(4y^2 - 4\right)y^2\right] \tag{A.5}$$

Properties: Variable ranges: $x \in [-1.9, 1.9]; y \in [-1.1, 1.1];$

No. of global optima: 2

No. of local optima: 2

6. **Shubert**

$$F_6(\vec{\mathbf{x}}) = -\prod_{i=1}^{D}\sum_{j=1}^{5} j\cos\left[(j+1)x_i + j\right] \tag{A.6}$$

Properties:

Variable ranges: $x_i \in [-10, 10]^D, i = 1, 2, \ldots, D;$

No. of global optima: $D \cdot 3^D$

No. of local optima: many

7. **Vincent**

$$F_7(\vec{\mathbf{x}}) = \frac{1}{D}\sum_{i=1}^{D}\sin\left(10\log\left(x_i\right)\right) \tag{A.7}$$

Properties:

Variable range: $x_i \in [0.25, 10]^D, i = 1, 2, \ldots, D$

No. of global optima: 6^D

No. of local optima: 0

8. **Modified Rastrigin**

$$F_9(\overrightarrow{\mathbf{x}}) = -\sum_{i=1}^{D} (10 + 9\cos(2\pi k_i x_i)) \tag{A.8}$$

Properties:

Variable ranges: $x_i \in [0,1]^D$,i = 1,2,...,D;

No. of global optima: $\prod_{i=1}^{D} k_i$

No.of local optima:0;

Now we will describe the general framework for constructing multimodal composition functions with several global optima and then present the new composition functions.

More specifically, a D -dimensional, composition function $\mathrm{CF}_j : \mathcal{A}_D \subset \mathbb{R}^D \to \mathbb{R}$ can be generally constructed as a weighted aggregation of n basic functions $f_i : \mathcal{A}_D \subset \mathbb{R}^D \to \mathbb{R}$. Each basic function is shifted to a new position inside the optimization space $\mathcal{A}_D$ and can be either rotated through a linear transformation matrix or used as is. Thus, a composition function CF_j is calculated according the following equation:

$$\mathrm{CF}_j(\vec{x}) = \sum_{i=1}^{n} w_i \left(\hat{f}_i \left((\vec{x} - \overrightarrow{o_i}) / \lambda_i \cdot M_i \right) + \mathrm{bias}_i \right) + f^j_{\mathrm{bias}}$$

where n is the number of basic functions used to construct the composition function, $\hat{f}_i$ denotes a normalization of the i -th basic function, $i \in \{1, 2, \ldots, n\}$, w_i is the corresponding weight, $\overrightarrow{o_i}$ is the new shifted optimum of each $\hat{f}_i$, M_i is the linear transformation (rotation) matrix of each $\hat{f}_i$, and λ_i is a parameter which is used to stretch ($\lambda_i > 1$) or compress ($\lambda_i < 1$) each $\hat{f}_i$ function. The composition function includes two bias parameters bias $_i$ and f^j_{bias} . The former defines a function value bias for each basic function and denotes which optimum is the global optimum, while the latter defines a function value bias for the constructed composition function. Here, we set the bias $_i = 0, \forall i \in \{1, 2, \ldots, n\}$, thus the global optimum of each basic function is a

global optimum of the composition function. In addition, we set $f^j_{\text{bias}} = 0$, as such in each composition function, all global optima have fitness values equal to zero.

The weight w_i of each basic function can be easily calculated based on the following equations:

$$w_i = \exp\left(-\frac{\sum_{k=1}^{D}(x_k - o_{ik})^2}{2D\sigma_i^2}\right)$$

$$w_i = \begin{cases} w_i & w_i = \max(w_i) \\ w_i\left(1 - \max(w_i)^{10}\right) & \text{otherwise} \end{cases}$$

Finally, the weights are normalized according to $w_i = w_i / \sum_{i=1}^{n} w_i$. The parameter σ_i controls the coverage range of each basic function, with small values to produce a narrow coverage range to the corresponding $\hat{f}_i$.

The pool of basic functions may include functions with different properties, characteristics and heights. As such to have a better mixture of the basic functions a normalization procedure is incorporated. The normalized function $\hat{f}_i$, can be defined as: $\hat{f}_i(\cdot) = C f_i(\cdot)/|f^i_{\max}|$, where C is a predefined constant ($C = 2000$) and $f^i_{\max}$ is estimated using: $f^i_{\max} = f_i\left((x^\star/\lambda_i) M_i\right)$, with $x^\star = [5, 5, \ldots, 5]$ The pool of basic functions that we have used to construct the composition functions includes the following:

- Sphere function:

$$f_S(\vec{x}) = \sum_{i=1}^{D} x_i^2$$

- Grienwank's function:

$$f_G(\vec{x}) = \sum_{i=1}^{D} \frac{x_i^2}{4000} - \prod_{i=1}^{D} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$$

- Rastrigin's function:

$$f_R(\vec{x}) = \sum_{i=1}^{D} \left(x_i^2 - 10\cos(2\pi x_i) + 10\right).$$

- Weierstrass function:

$$f_W(\vec{x}) = \sum_{i=1}^{D} \left(\sum_{k=0}^{\text{kmax}} \alpha^k \cos\left(2\pi\beta^k (x_i + 0.5)\right) \right) - D \sum_{k=0}^{\text{kmax}} \alpha^k \cos\left(2\pi\beta^k (0.5)\right)$$

where $\alpha = 0.5, \beta = 3$, and kmax = 20. - Expanded Griewank's plus Rosenbrock's function (EF8F2):

$$
\begin{aligned}
F8(\vec{x}) &= \sum_{i=1}^{D} \frac{x_i^2}{4000} - \prod_{i=1}^{D} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1 \\
F2(\vec{x}) &= \sum_{i=1}^{D-1} \left(100\left(x_i^2 - x_{i+1}\right)^2 + \left(x_i - 1\right)^2\right) \\
EF8F2(\vec{x}) &= F8F2\left(x_1, x_2, \ldots, x_D\right) \\
&= F8\left(F2\left(x_1, x_2\right)\right) + F8\left(F2\left(x_2, x_3\right)\right) + \ldots \\
&+ F8\left(F2\left(x_{D-1}, x_D\right)\right) + F8\left(F2\left(x_D, x_1\right)\right)
\end{aligned}
$$

It is clear that the aforementioned basic functions do not incorporate either shifted positions, or linear transformations (rotations). Thus, in order to calculate for example $f_S\left(\left(\vec{x} - \overrightarrow{o_i}\right)/\lambda_i \cdot M_i\right)$, one can easily first calculate $\vec{z} = \left(\vec{x} - \overrightarrow{o_i}\right)/\lambda_i \cdot M_i$ and subsequently $f_S(\vec{z})$. It has to be noted that all composition functions are formulated as maximization problems.

1. **Composition Function 1** Composition Function 1 (CF_1) is constructed based on six basic functions ($n = 6$), thus it has six global optima in the optimization box $\mathcal{A}_D = [-5, 5]^D$. The basic functions used here include the following:
   $-f_1 - f_2$: Grienwank's function,
   $-f_3 - f_4$: Weierstrass function, and
   - $f_5 - f_6$: Sphere function.
   The composition function is constructed based on the following parameter settings:
   $-\sigma_i = 1, \forall i \in \{1, 2, \ldots, n\}$
   $-\vec{\lambda} = [1, 1, 8, 8, 1/5, 1/5]$
   - M_i are identity matrices $\forall i \in \{1, 2, \ldots, n\}$.
   Fig. 9 shows the 2D version of CF_1. Properties:
   - Multi-modal,
   - Shifted,
   - Non-Rotated,

- Non-symmetric,

- Separable near the global optima,

- Scalable,

- Numerous local optima,

- Different function's properties are mixed together,

- Sphere Functions give two flat areas for the function,

$-$ In the optimization box $\mathcal{A}_D = [-5, 5]^D$, there are six global optima $\overrightarrow{x_i^\star} = \overrightarrow{o_i}, i \in \{1, 2, \ldots, n\}$ with $\mathrm{CF}_1\left(\overrightarrow{x_i^\star}\right) = 0, \forall i \in \{1, 2, \ldots, n\}$

2. **Composition Function 2**

Composition Function 2 (CF_2) is constructed based on eight basic functions ($n = 8$), thus it has eight global optima in the optimization box $\mathcal{A}_D = [-5, 5]^D$. The basic functions used here include the following:

$-f_1 - f_2$: Rastrigin's function,

- $f_3 - f_4$: Weierstrass function,

$-f_5 - f_6$: Griewank's function, and

$-f_7 - f_8$: Sphere function.

The composition function is constructed based on the following parameter settings:

$-\sigma_i = 1, \forall i \in \{1, 2, \ldots, n\}$

$-\vec{\lambda} = [1, 1, 10, 10, 1/10, 1/10, 1/7, 1/7]$

$-M_i$ are identity matrices $\forall i \in \{1, 2, \ldots, n\}$.

Fig. 10 shows the 2D version of CF_2. Properties:

- Multi-modal,

- Shifted,

- Non-Rotated,

- Non-symmetric,

- Separable near the global optima,

- Scalable,

- Numerous local optima,

- Different function's properties are mixed together,

- In the optimization box $\mathcal{A}_D = [-5,5]^D$, there are eight global optima $\overrightarrow{x_i^\star} = \overrightarrow{o_i}, i \in \{1,2,\ldots,n\}$ with $\mathrm{CF}_2\left(\overrightarrow{x_i^\star}\right) = 0, \forall i \in \{1,2,\ldots,n\}$

3. **Composition Function 3** Composition Function 3 (CF_3) is constructed based on six basic functions ($n = 6$), thus it has six global optima in the optimization box $\mathcal{A}_D = [-5,5]^D$. The basic functions used here include the following:

   $-f_1 - f_2$: EF8F2 function,

   $-f_3 - f_4$: Weierstrass function, and

   - $f_5 - f_6$: Griewank's function. The composition function is constructed based on the following parameter settings:

   $-\vec{\sigma} = [1,1,2,2,2,2]$,

   $-\vec{\lambda} = [1/4, 1/10, 2, 1, 2, 5]$

   - M_i are different linear transformation (rotation) matrices with condition number one.

   Properties:

   – Multi-modal,

   – Shifted,

   - Rotated,

   - Non-symmetric,

   - Non-separable,

   - Scalable,

   - A huge number of local optima,

   - Different function's properties are mixed together,

   – In the optimization box $\mathcal{A}_D = [-5,5]^D$, there are six global optima $\overrightarrow{x_i^\star} = \overrightarrow{o_i}, i \in \{1,2,\ldots,n\}$ with $\mathrm{CF}_3\left(\overrightarrow{x_i^\star}\right) = 0, \forall i \in \{1,2,\ldots,n\}$

4. **Composition Function 4** Composition Function 4 (CF_4) is constructed based on eight basic functions ($n = 8$), thus it has eight global optima in the optimization box $\mathcal{A}_D = [-5,5]^D$. The basic functions used here include the following:

   $-f_1 - f_2$: Rastrigin's function,

$-f_3 - f_4$: EF8F2 function,

$-f_5 - f_6$: Weierstrass function, and

- $f_7 - f_8$: Griewank's function. The composition function is constructed based on the following parameter settings:

- $\vec{\sigma} = [1, 1, 1, 1, 1, 2, 2, 2]$

$-\vec{\lambda} = [4, 1, 4, 1, 1/10, 1/5, 1/10, 1/40]$

$-M_i$ are different linear transformation (rotation) matrices with condition number one. Properties:

- Multi-modal,

- Shifted,

- Rotated,

- Non-symmetric,

- Non-separable,

- Scalable,

- A huge number of local optima,

- Different function's properties are mixed together,

- In the optimization box $\mathcal{A}_D = [-5, 5]^D$, there are eight global optima $\overrightarrow{x_i^\star} = \overrightarrow{o_i}, i \in \{1, 2, \ldots, n\}$ with $\mathrm{CF}_4\left(\overrightarrow{x_i^\star}\right) = 0, \forall i \in \{1, 2, \ldots, n\}$

Publications

1. **Shatendra Singh**, Aruna Tiwari, and Suchitra Agrawal, "Differential Evolution Algorithm For Multimodal Optimization:A Short Survey", 10th International Conference on Soft Computing for Problem Solving - SocProS 2020, Indore, India, 2020 (**Accepted**)

2. **Shatendra Singh**, Aruna Tiwari, Enhanced Opposition Differential Evolution Algorithm for Multimodal Optimization. (To be submitted to Applied Intelligence Journal)