

Authors' Instructions

Preparation of Camera-Ready Contributions to SCITEPRESS Proceedings

First Author Name¹, Second Author Name¹ and Third Author Name²

¹*Institute of Problem Solving, XYZ University, My Street, MyTown, MyCountry*

²*Department of Computing, Main University, MySecondTown, MyCountry*
{f_author, s_author}@ips.xyz.edu, t_author@dc.mu.edu

Keywords: The paper must have at least one keyword. The text must be set to 9-point font size and without the use of bold or italic font style. For more than one keyword, please use a comma as a separator. Keywords must be titlecased.

Abstract: Operational maturity of biological control systems have fuelled the inspiration for a large number of mathematical and logical models for control, automation and optimisation. The human brain represents the most sophisticated control architecture known to us and is a central motivation for several research attempts across various domains. In the present work, we introduce an algorithm for mathematical optimisation that derives its intuition from the hierarchical and distributed operations of the human motor system. The system comprises global leaders, local leaders and an effector population that adapt dynamically to attain global optimisation via a feedback mechanism coupled with the structural hierarchy. The hierarchical system operation is distributed into local control for movement and global controllers that facilitate gross motion and decision making. We present our algorithm as a variant of the classical Differential Evolution algorithm, introducing a hierarchical crossover operation. The discussed approach is tested exhaustively on standard test functions as well as the CEC 2017 benchmark. Our algorithm significantly outperforms various standard algorithms as well as their popular variants as discussed in the results.

1 INTRODUCTION

2 Introduction

Evolutionary algorithms are classified as meta-heuristic search algorithms, where possible solution elements span the n-dimensional search space to find the global optimum solution. Over the years, natural phenomena and biological processes have laid the foundation for several algorithms for control and optimization that have highlighted their applicability in solving intricate optimization problems. For instance, at the cellular level in the E.Coli Bacterium, there is sensing and locomotion involved in seeking nourishment and avoiding harmful chemicals. These behavioral characteristics fuelled the inspiration for the Bacterial Foraging Optimization algorithm (Passino, 2002)(Onwubolu and Babu, 2013). Ant Colony Optimization (Dorigo and Stützle, 2010) deals with behavior of ants and has been a successful model for solving complex problems. Particle Swarm Optimization (Kennedy and Eberhart, 1995) is a swarm intelligence algorithm based on behavior of birds and

fishes that models these particles as they traverse an n-dimensional search space and share information in order to obtain global optimum. From a biological control point, the human brain represents one of the most advanced architectures and several research attempts seek to mimic its functional accuracy, precision and efficiency. The brain function activities can be broadly classified into 2 categories: sensory and motor operations. Sensory cortical functions inspired the concept of neural networks that are being scaled successfully in deep learning to solve vast amount of problems.

The human motor function represents a distributed neural and hierarchical control system. It can be classified as having local control functions for movement as well as higher level controllers for gross motion and decision making. The execution of motor operation involves distributed brain structures at different levels of hierarchy. These include the pre-frontal cortex, motor cortex, spinal cord, anterior horn cells etc (Shaw et al., 1982). For executing an action sequence, a sequence of actions is implemented by a string of subsequences of actions each implemented

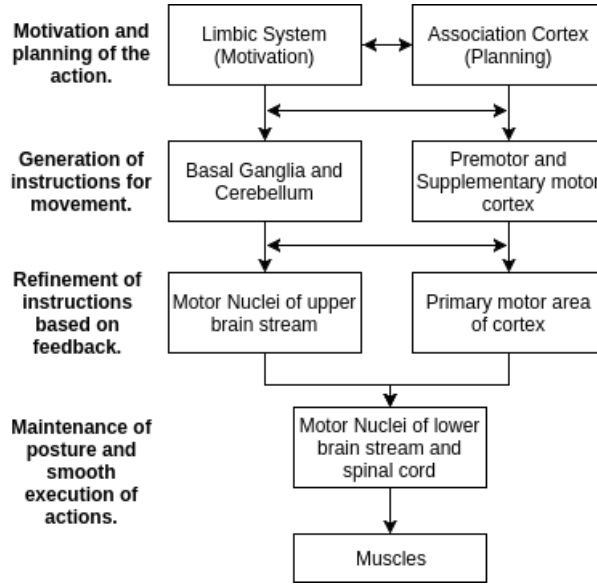


Figure 1: Hierarchy of Motor Control in Humans

in a different part of the body. The operational structure has been depicted in Figure 1(Passino, 2005). For optimality of actions, neurons act in unison. The neurons in the motor cortex act like global leaders and send inhibitory or facilitatory influence over anterior horn cells, the local leaders, located in the spinal cord(Shaw et al., 1982). These local leaders are connected to muscle fibers, the effectors, through a peripheral nerve and neuromuscular junction. Efficient execution of task requires feedback based facilitation and inhibition of the effectors over the anterior horn cells. These sequence of operations realise the optimal convergence of the system leading to smooth motor execution.

The present work introduces an algorithm modelled intuitively on the distributed and hierarchical operation of the brain motor function.

The Classical DE Algorithm (Storn and Price, 1995), proposed by Storn and Price has been hailed as one of the premier evolutionary algorithms, owing to its simple yet effective structure(Das and Suganthan, 2011). However, in recent times, it has been criticized for its slow convergence rate and inability to effectively optimize multimodal composite functions(Das and Suganthan, 2011). This work focusses on supplementing the algorithm's performance through the introduction of hierarchical influence in the pipeline. The architecture enables the algorithm to control the flow of agents through the cumulative effect of global and local leaders in the hierarchy.

The proposed approach, Hierarchy Influenced Differential Evolution (HIDE), has been subjected to exhaustive analysis on the hybrid and composite ob-

jective functions of the CEC 2017 benchmark(Awad et al., 2016). Comparison with the classical DE algorithm and its other popular variants including JADE and PSODE (Zhang and Sanderson, 2009) highlights the particular viability of the schemed approach in solving complex optimization tasks. We show that even with fixed parameters, HIDE is able to outperform adaptive architectures such as JADE by a respectable margin, as discussed in the result sections.

3 Classical Differential Evolution

The classical Differential Evolution (DE) algorithm is a population-based global optimization algorithm, utilizing a crossover and mutation approach to generate new individuals in the population for achieving optimum solutions(Das and Suganthan, 2011). For each individual x_i that belongs to the population for generation G , DE randomly samples three individuals from the population namely $x_{r1,G}$, $x_{r2,G}$ and $x_{r3,G}$. Employing these randomly chosen points, a new individual trial vector, v_i , is generated using equation (1):

$$v_i = x_{r1,G} + F(x_{r2,G} - x_{r3,G}) \quad (1)$$

Where, F is called the differential weight (Usually lies between $[0, 1]$).

To obtain the updated position of the individual, a crossover operation is implemented between $x_{i,G}$ and v_i , controlled by the parameter CR called the crossover probability. The value for CR always lies between $[0, 1]$.

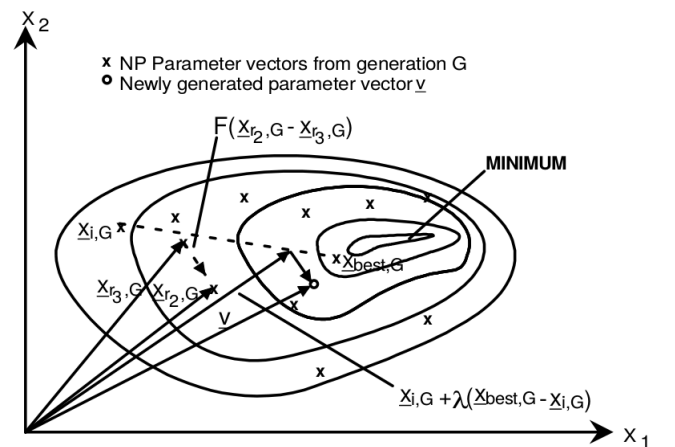


Figure 2: Motion planning of individuals in DE on two dimensional example of objective function.

4 Hierarchy Influenced Differential Evolution

Taking inspiration from the human motor system, we model the hierarchical motor operations in our optimization agents, where we define a global leader which influences the action of several distributed local leaders and the particle agents which act as the effectors. The global leader is analogous to the decision making and planning section in the motor system hierarchy whilst, the local leaders correspond to motion generators acting under the influence of the global leader.

The position of each particle in the population is affected by the influence of global leader and local leaders, while also being affected by a randomly chosen particle from the population to induce some stochasticity in the optimization pipeline. We first model the influence of the global leader on the local leaders and the influences of the local leaders on each population element using equation (3) and (4). We introduce a hierarchical crossover between the two influencing equations governed by a hierarchical crossover parameter HC .

Analogously to the brain motor operation as depicted in Figure 1, the update of particle positions requires generating feedback for the leaders as a part of the optimization procedure, and hence the local leaders and the global leader are updated based on their objective function value generated from the perturbations in population particles. This series of events comprise of one optimization pass (one generation step). On execution of several optimization passes as described, the system is able to converge to an optimal configuration, analogous to the successful execution of the required task as shown in the final steps of Figure 1.

For each particle $x_{i,G}$, $i = 0, 1, 2, \dots, NP - 1$ for generation G , the trial vector x'_i of the particle, is governed by the hierarchical crossover operation and a mutation operation as follows :

$$u_i = \begin{cases} E_g, & \text{if } G < HC * G_t \\ E_l, & \text{otherwise} \end{cases} \quad (2)$$

$$E_g = g_L + F(x_{L_i,G} - x_{r,G}) \quad (3)$$

$$E_l = x_{L_i,G} + F(x_{i,G} - x_{r,G}) \quad (4)$$

for each dimension j of $x_{j,i,G}$:

$$x'_{j,i} = \begin{cases} x_{j,i,G} & \text{if } \text{rand}(0,1) < HC \\ u_{j,i} & \text{otherwise} \end{cases} \quad (5)$$

$$x_{i,G+1} = \begin{cases} x'_{i,G}, & \text{if } f(x'_{i,G}) < f(x_{i,G}) \\ x_{i,G}, & \text{otherwise} \end{cases} \quad (6)$$

where,

G_t is the total number of generations,

$x_{i,G+1}$ is the vector position of $x_{i,G}$ for next generation

F is factor responsible for amplification of differential variation,

f is the objective function,

$x_{i,G}$ is the current position of the individual for generation G ,

u_i is the intermediate trial vector of the current

Algorithm 1 Hierarchy Influenced Differential Evolution

```

1: procedure START
2:   Initialize parameters ( $HC, F, P, N_l, NP$ ).
3:   Generate initial global leader  $g_L$  as a random point.
4:   Generate  $N_l$  local leader points around  $g_L$  global leader.
5:   Using a Normal distribution, generate  $NP$  points for population  $P$  around the local leaders.
6:   while termination criteria is not met do
7:     for each individual  $x_{i,G}$  in  $P$  do
8:       Determine the corresponding local leader  $x_{L_i,G}$  from the set of all local leader based on nearest position.
9:       Let  $u = 0$  be an empty vector.
10:      Let  $G$  and  $G_t$  be the current generation and total generations of the procedure.
11:      if  $G == (HC * G_t)$  then
12:        Increase the population :  $G_t = 2 * G_t$ 
13:      end if
14:      if  $G < (HC * G_t)$  then
15:         $u_i = E_g$  from (3).
16:      else
17:         $u_i = E_l$  from (4).
18:      end if
19:       $x'_i = \text{BinomialCrossover}(u_i, x_{i,G}, CR)$ 
20:      if  $f(x'_i) < f(x_{i,G})$  then
21:        Replace  $x_{i,G}$  with  $x'_i$  in the next generation.
22:      end if
23:    end for
24:    Alter local leaders in each population cluster based on objective function value.
25:    Compute updated global leader  $g_L$ .
26:  end while
27: end procedure

```

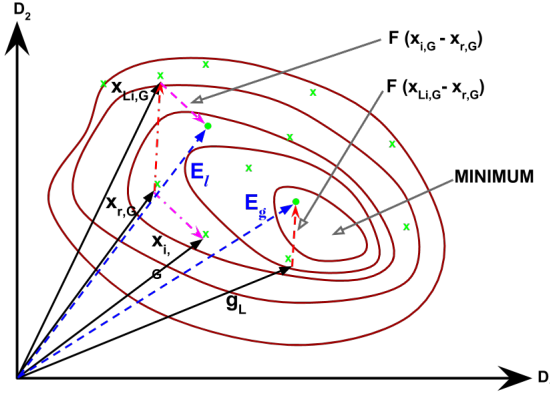


Figure 3: Hierarchical Decisive Motion planning of individuals in HIDE on two dimensional example of objective function. The position vectors resulting from the influence of global leader and local leaders are both represented as E_g and E_l on the contour of a two dimensional objective function.

individual,

E_g represents the global and local leader interaction,
 E_l represents the local leader and effector interaction,
 g_L is the global leader for generation G ,
 $x_{L_i,G}$ is the position of the local leader for current individual,
 $x_{r,G} \in P$; $r \in [0,1,\dots, NP-1]$
 $x'_{j,i}$ is the trial vector

$x_{r,G}$ is randomly chosen particle from the population to induce stochasticity. The hierarchical operation is affected by the global leader g_L and the local leader $x_{L_i,G}$ through the parametric equations (3) and (4). Switching between the two is governed by the hierarchical crossover parameter HC .

Algorithm 2 Binomial.Crossover(u, x, CR)

```

1: procedure START
2:   Let  $x' = 0$  be an empty vector.
3:   Select a random integer  $k = irand(\{1,2,\dots,d\})$ ;
   where  $d$  = number of dimensions
4:   for each dimension  $j$  do
5:     if  $random(0,1) < CR$  or  $j == k$  then
6:       Set  $x'_j = u_j$ 
7:     else
8:       Set  $x'_j = x_j$ 
9:     end if
10:  end for
11: end procedure

```

4.1 Hierarchical Crossover

Convergence trend in HIDE is largely pivoted about (3) and (4), which in unison, lend a hierarchical structure to the algorithm. A successful optimization algorithm involves establishing a trade-off between exploration and exploitation. Achieving global optimization can be visualized as collaboration of two forces, exploration over a larger subspace followed by intensive exploitation over the resulting search space governed by clusters. Phase 1, involving (3) is marked by the interaction between the global and local leaders representing decision planning and facilitation of gross motion. This is followed by phase 2, involving (4) wherein the local leaders interact with and guide their effector population to control intricate motion over the constraint subspace to achieve smooth convergence. Robust convergence necessitates an optimal transition from phase 1 to phase 2 in the hierarchy. This hierarchical transition is characterized by our proposed parameter, HC . **The value of HC belongs to $[0,1]$. An optimal value for HC was observed experimentally to lie about one-quarter. For the purpose of our experiment, we have deterministic fixed HC to be 0.27. The choice for HC was made after successful experimental observation on more than 50% of the test functions with HC set as 27% of the total generation budget.**

The HIDE algorithm achieves a performance improvement in the early optimization phase ($G < HC * G_t$) by replacing clusters of the initially generated candidate solutions with the locally best. This strategy rules out a number of mutation vectors that are more unfavorable in terms of performance gain. Additionally, by focusing on mutants of the globally best candidate solution the search space is explored rather quickly during this phase. After the population advances to $HC * G_t$ generations, the algorithm changes its reference point (the trial vector) to the locally best candidate solutions of a certain cluster. That is, having approached a closer distance from the optimal, the algorithm is able to exploit the search space. Our proposition is complemented by the observations in our results section wherein we significantly outperform several popular algorithms on involved multimodal hybrid and composite functions in higher dimensions.

5 Results and Discussions

All evaluations were performed using Python 2.7.12 with Scipy(Oliphant, 2007) and Numpy(Van Der Walt et al., 2011) for numerical computations and

Table 1: CEC 2017 Test Functions

F_{id}	Problem Function	F^*
f_1	Shifted and Rotated Bent Cigar Function	100
f_2	Shifted and Rotated Sum of Different Power Function	200
f_3	Shifted and Rotated Zakharov Function	300
f_4	Shifted and Rotated Rosenbrock's Function	400
f_5	Shifted and Rotated Rastrigin's Function	500
f_6	Shifted and Rotated Expanded Scafer's F6 Function	600
f_7	Shifted and Rotated Lunacek Bi_Rastrigin Function	700
f_8	Shifted and Rotated Non-Continuous Rastrigin's Function	800
f_9	Shifted and Rotated Levy Function	900
f_{10}	Shifted and Rotated Schwefel's Function	1000
f_{11}	Hybrid Function 1 (N=3)	1100
f_{12}	Hybrid Function 2 (N=3)	1200
f_{13}	Hybrid Function 3 (N=3)	1300
f_{14}	Hybrid Function 4 (N=4)	1400
f_{15}	Hybrid Function 5 (N=4)	1500
f_{16}	Hybrid Function 6 (N=4)	1600
f_{17}	Hybrid Function 7 (N=5)	1700
f_{18}	Hybrid Function 8 (N=5)	1800
f_{19}	Hybrid Function 9 (N=5)	1900
f_{20}	Hybrid Function 10 (N=6)	2000
f_{21}	Composition Function 1 (N=3)	2100
f_{22}	Composition Function 2 (N=3)	2200
f_{23}	Composition Function 3 (N=4)	2300
f_{24}	Composition Function 4 (N=4)	2400
f_{25}	Composition Function 5 (N=5)	2500
f_{26}	Composition Function 6 (N=5)	2600
f_{27}	Composition Function 7 (N=6)	2700
f_{28}	Composition Function 8 (N=6)	2800
f_{29}	Composition Function 9 (N=3)	2900
f_{30}	Composition Function 10 (N=3)	3000
Search Range: $[-100,100]^D$		

Matplotlib (Hunter, 2007) package for graphical representation of the result data. This section is divided into two sub-sections: Section A provides description about the problem set used for analysis of algorithmic efficiency and accuracy, and section B comprises of tabular and graphical data to reinforce the claim of superiority of the proposed approach.

5.1 Problem Set Description

The set of objective functions considered for testing the proposed algorithm and compare its performance against classical DE and its variants PSODE and JADE have been taken from the CEC 2017 set of benchmark functions. Exhaustive comparisons and analysis have been depicted on dimensions $D = 10, 30, 50$ and 100 for a clear understanding of the strengths of the proposed algorithm. Objective functions $f_1 - f_3$ are simple unimodal functions and $f_4 - f_{10}$ are multimodal functions with a high number of local optima values. Functions $f_{11} - f_{20}$ are all hybrid functions using a combination of functions from $f_1 - f_{10}$. The set of composite function range from $f_{21} - f_{30}$ and merges the properties of the sub-functions better while incorporating the basic functions as well as hybrid functions to increase complexity while maintaining continuity around the global optima.

Summarized in Table 1 are the 30 objective functions from the CEC 2017 dataset and the global optimum value for each function denoted by F^* . In all simulation runs, we set the population size NP to a fixed value of 100 , and the results are shown in a tabular structure depicting the best and average values of the population individuals for the simulations. Additionally, several graphical results have been discussed to observe the convergence rate and efficiency of the algorithms used in the simulation. These graphs were plotted based on the numerical results obtained from the simulation runs used to build the tables.

5.2 Parameter Settings

For fair comparisons, the parameters for all algorithms are fixed to the values depicted in table 2. As

Table 2: Algorithm Parameter Settings used for comparison

Algorithm
DE (Storn and Price, 1995),(Mezura-Montes et al., 2006),(Brest et al.,
PSODE ()
JADE (Zhang and Sanderson, 2009)
HIDE

clear from the table, we set the parameters F and CR as 0.5 and 0.9 for DE across all experiments, as recommended in (Storn and Price, 1995), (Mezura-Montes et al., 2006), (Brest et al., 2006). The parameters for JADE were selected as suggested in the original work (Zhang and Sanderson, 2009). These parameter settings allow transparency in results and a base for fair and clear comparisons in the analysis of the algorithms.

5.3 Numerical and Graphical Results

In tables 3-6, the best and mean values obtained for the population agents in the simulation runs have been reported, and the optimum values for each objective function have been highlighted in **bold**. For the sake of clarity, the comparison results in each table have been summarized in "w/t/l" format wherein w represents the number of objective functions where the algorithm outperforms all other algorithms, t specifies the number of objective functions where it is tied as the best algorithm for the objective function and l represents the number of test functions where it does not finish first. The utilization of the evaluation metric facilitates a definitive comparison of the different algorithms under consideration.

Tables 3 and 4 highlight the performance of the algorithms for $D = 10$ and $D = 30$ respectively. For $D = 10$, HIDE depicts impressive performance. It registered a "w/t/l" score of 12/7/11 in the best case and 14/4/12 in the mean of population case. In both these test functions, JADE achieved the second best performance registering a w/t/l of 5/7/18 on the best optimal value case and 9/4/17 on the mean of optimal value case. On $D = 30$, HIDE achieved maximum number of wins in both best and mean case (17 and 18 respectively). JADE achieved second position with 8 and 9 wins in the best and mean case. The decent performance of JADE can be attributed to the adaptive nature of its parameter selection which enables enhancement of its convergence rate.

The results for $D = 50$ and $D = 100$ (higher dimensions) have been summarized in tables 5 and 6. On $D = 50$, HIDE depicted exceptional performance, outperforming all other algorithms. It registered 17 wins in the best case and 18 wins in the mean case. Classical DE shows no wins in any case in high dimensional settings owing to its slow convergence rate and inability to attain global optimum thus highlighting the usefulness of the modifications introduced in the variants including HIDE. Similarly for $D = 100$, HIDE again outperforms all other algorithms by an appreciable margin. From a functional standpoint, It would be worthwhile to highlight that HIDE out-

performed the other 3 compared algorithms on majority on the composite and hybrid functions, particularly on the higher dimensional settings. The efficiency of HIDE can be attributed to the hierarchical nature of crossover selection and concurrency in vector configurations at the higher hierarchy levels. The tabular results reinforce the fact that HIDE outperforms JADE, PSODE and DE. On close analysis, it can be witnessed that HIDE falls behind the other algorithms on a small fraction of unimodal functions such as f_5, f_7 on lower dimensions due to fast convergence during early stages of execution. However, the performance of higher dimensions, particularly on the more involved functions highlights utility for real world problems.

The tabular results are complemented through the graphical representations in Figure 4. For the sake of clarity, representations of higher dimensional problems span more number of iterations than those for lower dimensional settings. Analysis of the plots clearly depicts that HIDE shows better convergence rate as compared to other algorithms. As the analysis transcends to higher dimensional settings, the proposed approach outperforms the other algorithms on majority of the objective functions with respect to both convergence rate and optimality. the superiority of our algorithm in higher dimensions (50 and 100) is clearly evident from Figure 4. (m,n,q,r,s,t). Figure 4. (e,f,g,h) depict that for functions where HIDE and the other variants may depict similar trends on lower dimensions, HIDE eventually excels and surpasses them in higher dimensions in most scenarios. Almost all figures are representative of a faster convergence rate for HIDE on higher dimensions. This remarkable trait in HIDE enhances its utility for high dimensional problems where fast convergence to global optimum value is required, hence making it superior to the other considered algorithms and several variants of the DE algorithm.

6 Conclusion

Differential Evolution has been regarded as one of the most successful optimization algorithms and over the years, several variants have been proposed to enhance its convergence rate and performance. In the present work, we introduced a hierarchy influenced variant of the classical DE algorithm and modeled the same on the brain motor operation. The algorithm was characterized by global leader, local leaders and an effector population. The global leader and distributed local leaders interacted to facilitate gross motion via a greedy exploration strategy. The local

Table 3: Objective Function Value for Dimension: 10

ID	DE		JADE		PSO-DE		HIDE	
	best	mean	best	mean	best	mean	best	mean
f_1	100.000051	100.011085	100.0	100.0	100.000712	185.975885	100.0	100.0
f_2	200.0	200.1	200.0	200.0	200.0	200.0	200.0	200.0
f_3	300.00134	300.214502	300.0	300.0	300.000006	300.000985	300.0	300.0
f_4	400.042617	403.674837	400.0	400.409399	400.064644	404.307763	400.0	400.000003
f_5	566.661791	604.867489	523.908977	541.521084	525.868824	575.61616	533.803201	579.483815
f_6	621.914237	634.807962	620.878276	636.034759	603.187964	635.865001	613.730565	629.293758
f_7	724.831278	739.129935	717.016542	723.983312	725.44788	733.15638	720.345706	725.233785
f_8	818.904202	829.749207	821.914433	826.321588	820.8941	830.246691	821.064763	828.160987
f_9	900.0	908.104383	900.0	1084.478253	900.0	1124.102561	900.0	903.454324
f_{10}	1911.51009	2447.44375	1760.95686	2162.64858	2049.64472	2518.24109	1694.43759	2049.07426
f_{11}	1102.98570	1113.42310	1105.66167	1117.50974	1105.97013	1120.19297	1101.76974	1108.86359
f_{12}	2531.74630	6509.74307	1438.60571	5430.67468	4089.00635	10810.3876	1308.43834	1327.40588
f_{13}	1313.13022	1404.90360	1304.68156	1328.75526	1319.83919	1453.34078	1306.68204	1344.28224
f_{14}	1409.94961	1426.57193	1412.93443	1428.16943	1420.91065	1434.11288	1404.92899	1410.00077
f_{15}	1504.13139	1521.44661	1502.49618	1508.31154	1501.38951	1518.31035	1500.08137	1503.16926
f_{16}	1958.42062	2104.55573	1958.85799	2094.63082	1958.41153	2048.15688	1958.43351	2062.38595
f_{17}	1728.19497	1743.15524	1730.71532	1748.12987	1727.80039	1791.60774	1723.85397	1747.58908
f_{18}	1801.58601	1838.84055	1804.29854	1825.09164	1817.15464	1840.54692	1800.23551	1804.01430
f_{19}	1901.19548	1903.60476	1900.39977	1902.15296	1902.71174	1906.25233	1900.00563	1901.01412
f_{20}	2204.55412	2289.22658	2148.53894	2178.31317	2140.56131	2261.03877	2139.91553	2172.81652
f_{21}	2337.77299	2387.23036	2314.42114	2338.68872	2337.20734	2351.89886	2320.49621	2344.61612
f_{22}	2300.80585	2304.13288	2300.0	2300.09348	2300.68418	2301.71048	2300.00002	2301.09598
f_{23}	3070.17708	3145.77229	3003.67856	3091.22041	2773.37286	3060.02252	2867.02004	3047.98231
f_{24}	2500.0	2500.0	2500.0	2500.0	2500.0	2500.0	2500.0	2500.0
f_{25}	2899.58497	2933.24981	2899.58497	2930.26651	2897.74287	2921.27479	2897.83339	2927.97651
f_{26}	2800.0	4117.597033	2800.0	2956.06417	2800.0	3367.60765	2800.0	3161.54808
f_{27}	3113.15765	3358.80643	3072.43902	3178.50964	3078.87313	3240.50181	3071.20357	3107.26854
f_{28}	3184.75565	3230.921422	3184.75565	3195.113042	3184.755652	3198.370691	3100.0	3195.411961
f_{29}	3148.587115	3266.979786	3172.400194	3233.707677	3191.348193	3244.892638	3189.211417	3292.420474
f_{30}	3442.555095	11927.404685	3207.766942	4615.591316	4573.358512	16415.162901	3205.740954	3249.710975
w/t/l	2/4/24	1/1/28	5/7/18	9/4/17	4/4/22	2/2/26	12/7/11	14/4/12

leaders and their effectors interacted to control intricate motion for smooth convergence. A hierarchical crossover parameter was introduced to characterize the hierarchical transition between the two interactions. The influence of the vector configurations at the higher levels of hierarchy enabled the algorithm to avoid local minima in most objective functions. The same is complemented through our result observations wherein we significantly outperform several popular algorithm on complex multimodal functions in higher dimensional settings. Our proposed approach has sought to establish a viable tradeoff between fast optimization, robust convergence and low number of control parameters. The performance analysis of the algorithm highlights the particular effectiveness of the proposed approach on high dimensional hybrid and composite functions. The observed results provide sufficient motivation to extend the scope of the work to complex high dimensional real life problems including image enhancement, traveling salesman problem and flexible job-shop scheduling.

REFERENCES

- Awad, N., Ali, M., Liang, J., Qu, B., and Suganthan, P. (2016). Problem definitions and evaluation criteria for the cec 2017 special session and competition on single objective real-parameter numerical optimization. *Technical Report*.
- Brest, J., Greiner, S., Boskovic, B., Mernik, M., and Zumer, V. (2006). Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems. *IEEE transactions on evolutionary computation*, 10(6):646–657.
- Das, S. and Suganthan, P. N. (2011). Differential evolution: A survey of the state-of-the-art. *Evolutionary Computation*, *IEEE Transactions on*, 15(99):4–31.
- Dorigo, M. and Stützle, T. (2010). Ant colony optimization: overview and recent advances. In *Handbook of metaheuristics*, pages 227–263. Springer.
- Hunter, J. D. (2007). Matplotlib: A 2d graphics environment. *Computing In Science & Engineering*, 9(3):90–95.
- Kennedy, J. and Eberhart, R. (1995). Particle swarm optimization. In *Neural Networks, 1995. Proceedings., IEEE International Conference on*, volume 4, pages 1942–1948 vol.4.
- Mezura-Montes, E. n., Velázquez-Reyes, J., and Coello Coello, C. A. (2006). A comparative study of differential evolution variants for global

Table 4: Objective Function Value for Dimension: 30

f_{id}	DE		JADE		PSO-DE		HIDE	
	best	mean	best	mean	best	mean	best	mean
f_1	100.001508	4334.438478	100.001338	100.056201	364.295574	4236.363207	100.0	100.0
f_2	40412441.0	5.129601e+19	9200.0	1535352368	6332899.0	9.590679e+11	1200.0	159855.5
f_3	17926.87287	322131.54271	1969304.92609	174080.70037	215792.54757	521683.20909	23679.811599	8999.947269
f_4	481.255055	519.422652	403.633939	442.206911	468.341175	479.341966	400.004163	443.016156
f_5	689.041352	737.79326	667.50756	735.204027	715.904429	746.548906	685.40454	738.842184
f_6	643.626307	652.582714	651.39169	655.142819	642.724237	655.106996	644.701241	652.002395
f_7	883.347367	962.591129	779.907693	818.344111	790.014281	854.285524	812.923573	856.90477
f_8	923.37426	967.251501	931.500175	957.362003	915.414882	960.486239	930.288539	964.11663
f_9	5652.483961	7878.781444	4953.05469	5146.600953	6018.417197	9042.410178	4003.118072	4734.984364
f_{10}	3596.63104	4536.989761	4012.723292	4204.18969	3934.606704	4863.741107	3793.781776	4346.741344
f_{11}	1162.405965	1184.634006	1152.748529	1174.58813	1165.144993	1189.171787	1149.748499	1171.130409
f_{12}	56679.43509	2317650.6134	924821.17176	558930.09024	210221.07746	5161046.0554	09208.289246	41947.22269
f_{13}	3002.029489	18794.83599	14276.907742	13775.81623	93871.279833	10612.26359	1664.06241	2453.606969
f_{14}	1773.180798	5502.160382	1496.219858	42868.9158	1555.452763	4029.808535	1462.926848	1504.191515
f_{15}	1860.435669	2484.689969	1688.05046	2222.674323	1651.747476	2223.060524	1611.074402	1852.66177
f_{16}	2517.439623	2827.004968	2344.19818	2621.618684	2239.242719	2664.114667	2298.041965	2691.674809
f_{17}	2321.175936	2604.529778	2062.898023	2546.995596	2107.43677	2457.34021	1820.806639	2418.723829
f_{18}	38987.28245	694156.328505	51841.60813	184888.16218	862294.85325	118430.28912	212578.00378	423024.11193
f_{19}	2043.469888	3010.235379	1959.71819	2156.957875	3049.52231	6840.408394	1949.271714	1987.866761
f_{20}	2625.539158	2864.832611	2706.314441	2805.600064	2619.996493	2895.107238	2753.806213	2966.035793
f_{21}	2412.081757	2504.777775	2414.52134	2456.718982	2431.740293	2478.841357	2200.0	2442.734316
f_{22}	2300.481796	5655.569322	2300.0	4157.698784	2307.721358	6811.069162	2300.009985	6795.24842
f_{23}	3050.654508	3572.965066	2772.002023	2946.749322	2764.922461	3199.874364	2883.276891	3543.839343
f_{24}	3104.623692	3290.698756	2891.557648	2965.225566	2911.63347	2983.772932	2500.0	2940.75997
f_{25}	2916.180657	2946.711753	2875.106846	2881.091389	2875.498843	2889.943671	2874.171109	2877.484904
f_{26}	4043.691403	6756.3724	2900.0	3266.510982	2800.007809	3273.128769	2900.0	3298.490539
f_{27}	3200.005857	3998.876498	3145.810354	3189.82261	3145.425231	3639.634132	3132.816283	3284.28897
f_{28}	3290.744025	3326.263983	3100.0	3131.027315	3195.486838	3225.594053	3100.0	3115.505829
f_{29}	3720.314598	4115.185803	3305.310139	3626.887552	3535.952295	3867.593068	3352.845055	3709.102375
f_{30}	3359.030768	3900.826662	3263.496536	3749.610722	3312.635025	3524.714477	3298.704645	3421.715322
w/t/l	2/0/28	0/0/30	8/2/20	11/0/19	4/0/26	1/0/29	15/2/13	17/0/13

optimization. In *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation, GECCO '06*, pages 485–492, New York, NY, USA. ACM.

Oliphant, T. E. (2007). Python for scientific computing. *Computing in Science & Engineering*, 9(3):10–20.

Onwubolu, G. C. and Babu, B. (2013). *New optimization techniques in engineering*, volume 141. Springer.

Passino, K. M. (2002). Biomimicry of bacterial foraging for distributed optimization and control. *IEEE control systems*, 22(3):52–67.

Passino, K. M. (2005). *Biomimicry for optimization, control, and automation*. Springer Science & Business Media.

Shaw, D. M., Kellam, A., and Mottram, R. (1982). Brain sciences in psychiatry. In Shaw, D. M., Kellam, A., and Mottram, R., editors, *Brain Sciences in Psychiatry*. Butterworth-Heinemann.

Storn, R. and Price, K. (1995). *Differential evolution—a simple and efficient adaptive scheme for global optimization over continuous spaces*, volume 3. ICSI Berkeley.

Van Der Walt, S., Colbert, S. C., and Varoquaux, G. (2011). The numpy array: a structure for efficient numerical computation. *Computing in Science & Engineering*, 13(2):22–30.

Zhang, J. and Sanderson, A. C. (2009). Jade: adaptive differential evolution with optional external archive.

IEEE Transactions on evolutionary computation, 13(5):945–958.

Table 5: Objective Function Value for Dimension: 50

f_{id}	DE		JADE		PSO-DE		HIDE	
	best	mean	best	mean	best	mean	best	mean
f_1	5884574.8731	1467294248.52	2136.072384	3708.75086	5811.218992	154233.6467	4406.072862	3665.419272
f_2	4.718137e+24	43.364977e+44	2635725.0	5.02374e+26	2.212101e+19	2.544543e+23	32.279950e+17	71.00729e+31
f_3	45520.96637	662237.29681	9143481.79314	1756166.7623	562308.42743	64435.24063	44613.29993	58182.83733
f_4	574.400328	801.384952	418.580378	470.113207	477.080964	574.528479	400.005049	447.775413
f_5	816.394775	843.258843	809.899483	834.131266	778.59312	831.066954	791.405194	830.218472
f_6	652.541914	655.794152	633.217881	654.893828	653.291336	658.183613	645.25633	656.060597
f_7	1109.02123	1263.038487	889.036574	944.90319	915.153525	1047.43879	989.957862	1186.248741
f_8	1139.278925	1175.893113	1118.339103	1144.604745	1092.62639	1159.032351	1100.476077	1168.529946
f_9	22196.38781	729218.77598	211958.280061	13174.66236	24753.040541	32233.95451	10251.47631	14752.7168
f_{10}	6228.49289	7289.183679	6054.707691	6833.306317	6207.795302	7055.595231	6050.434374	6609.804567
f_{11}	1170.858603	1258.517635	1202.694857	1232.204268	1206.154564	1252.939541	1156.439606	1205.254497
f_{12}	677263.0799	16987989.98	174784.6159	530814.6481	584300.6983	3448448.7906	126908.2157	494471.0756
f_{13}	6005.535308	16893.949921	12041.488125	4332.5945	1572.252973	4301.829606	1484.761799	7760.056137
f_{14}	38490.532315	174367.45065	52466.047056	238838.47005	516327.42317	67939.000264	2967.818485	26290.31618
f_{15}	2278.141229	26989.255509	13553.041864	425636.76961	13443.587343	9167.267098	1938.200405	14976.72189
f_{16}	2722.026011	3176.916902	2345.400708	2916.561016	2521.93881	3146.04527	2436.449338	2978.37746
f_{17}	2799.949776	3289.61565	2568.383575	2907.869272	2887.281107	3236.957928	2561.370306	2874.965038
f_{18}	264037.1257	0872072.4773	936176.58677	113941.3176	26965.28512	114846.12136	6060540.7818	1936454.326476
f_{19}	10051.91240	720380.25713	2089.172253	7763.17234	9905.850822	16555.756926	2013.126904	3609.258962
f_{20}	2950.923195	3274.334015	3041.81309	3113.289461	2991.589293	3361.823946	2495.031774	3080.137478
f_{21}	2596.725663	2689.688363	2526.190898	2597.677199	2555.8788	2642.381597	2447.758274	2570.911014
f_{22}	9713.993241	10803.65373	210759.59674	11032.88095	38918.436264	10465.022457	8181.446081	9755.070369
f_{23}	3451.104943	4200.174424	2971.160647	3237.778662	2977.554961	3490.639751	2851.650254	3162.313622
f_{24}	3434.465028	3682.846708	3103.955173	3185.382676	3036.799607	3158.330504	3136.927747	3284.656095
f_{25}	3141.144886	3292.303449	2931.162959	2962.471758	2931.926959	3008.895353	2931.142314	2954.767839
f_{26}	4906.132848	7989.490966	2900.0	3346.874039	2900.441895	3653.757741	2900.0	3262.668498
f_{27}	3200.010703	3792.645588	3143.038057	3184.646353	3158.178238	3397.130323	3141.010872	3176.011524
f_{28}	3300.010827	3431.570911	3240.725865	3288.253039	3263.207144	3300.257609	3243.631996	3294.373237
f_{29}	3812.475517	4605.349537	3533.945743	3956.835243	3955.324537	4364.18129	3653.675553	3966.471956
f_{30}	3673.711968	5813.173755	3916.725719	4869.089335	3730.309354	5143.078706	3346.483679	4747.88675
w/t/l	0/0/30	0/0/30	8/1/21	9/0/21	4/0/26	3/0/27	17/1/12	18/0/12

Table 6: Objective Function Value for Dimension: 100

f_{id}	DE		JADE		PSO-DE		HIDE	
	best	mean	best	mean	best	mean	best	mean
f_1	3427212811.7	93807281895.7	141.263356	13516.69893	36067123.521	089751976.509	122.398748	11708.82360
f_2	4.19617e+84	1.54741e+11	28.73752e+74	2.54362e+87	6.1536e+66	3.2118e+73	3.8835e+80	8.8914e+114
f_3	228808.96909	462699.68763	912244.36094	332179.29069	341427.72366	6257462.97788	20765.0838	251901.1093
f_4	1975.651157	2752.246068	539.386275	677.054657	777.314462	836.965399	531.169819	621.219143
f_5	1223.536503	1286.153332	1249.195036	1307.110127	1248.410134	1310.887657	1068.11742	1272.47682
f_6	651.650133	657.84974	654.709342	659.421427	656.877048	662.318417	642.33355	654.132758
f_7	1614.003864	1920.797726	1367.066537	1536.357878	1311.849757	1534.207764	1562.379772	2076.702502
f_8	1595.418732	1736.367379	1672.567849	1768.082435	1678.127263	1761.94051	1293.552115	1592.162983
f_9	59726.51462	171986.04390	528906.90908	30336.74533	563640.33135	174961.220998	23466.57501	27067.02959
f_{10}	12005.88972	114725.34833	414227.80190	915355.62189	112937.02785	714972.950738	11153.58683	13298.09210
f_{11}	7540.617987	11481.26014	540447.54868	857228.68366	63521.901521	4544.804011	5380.432052	9916.347692
f_{12}	529993877.3	25881773956	29893556.272	26415173.6097	26105108.93	741876679.08	63680108.181	10059039.63
f_{13}	7943.9249	508209.56266	68622.698553	8892.775994	8246.515295	12675.845535	2976.841354	11376.986338
f_{14}	728122.8332	53329183.172	2432194.7952	365560.8816	548410.33828	8041547.52476	6234045.9401	6867160.306892
f_{15}	2660.465784	181957.0601	31799.506503	3362.509604	1899.073444	2914.44348	1976.789124	4485.415275
f_{16}	4749.254663	5847.826738	4817.483738	5632.3022	3852.700054	5228.663526	3519.494945	4796.802728
f_{17}	4397.496352	4958.418182	3842.206015	4450.177422	3790.72056	4730.994585	3582.785882	5463.216947
f_{18}	1357845.3930	5938893.279	7146426.2736	763318.8226	1004224.2038	3315010.2986	6831040.1463	51335739.59138
f_{19}	2482.170159	26455.70695	42098.9496	4767.529535	2263.725158	3927.459947	2071.077067	3664.159878
f_{20}	4968.497438	5436.604051	5231.026486	5690.748998	5109.460563	5781.300835	3627.777893	5228.430669
f_{21}	3180.746656	3355.4783	2921.900122	3085.692252	2885.574085	3127.356835	2926.350399	3199.986183
f_{22}	17808.89774	419562.98664	619213.375668	20278.92909	318695.52231	220167.41374	17548.33905	19547.15124
f_{23}	4907.519646	5819.207866	3352.556985	4222.436894	3582.043556	4779.921248	3418.983204	3609.098575
f_{24}	5173.249408	5946.12042	4060.951302	4095.429519	3801.368588	4042.426859	3998.054028	4216.824895
f_{25}	4089.118918	4548.285768	3153.485413	3236.61784	3348.382262	3407.526581	3176.3038	3264.318532
f_{26}	8557.498566	20159.11458	2900.077371	11924.79947	33021.136025	8682.035439	2900.000382	7867.5518
f_{27}	3200.023355	3772.409153	3194.809213	3201.670732	3200.024171	3494.618132	3200.023542	3200.023953
f_{28}	4947.745152	5948.213156	3295.122914	3340.280383	3456.828432	3542.571307	3300.807691	3354.717338
f_{29}	6004.774424	7090.642544	5208.711727	5970.628689	5462.328635	6178.559061	4541.195471	5739.291549
f_{30}	7798.106217	202435555.5	93584.974771	10674.21733	13920.327039	7139.460728	3850.317099	15318.554601
w/t/l	0/0/30	0/0/30	8/0/22	8/0/22	5/0/25	6/0/24	17/0/13	16/0/14

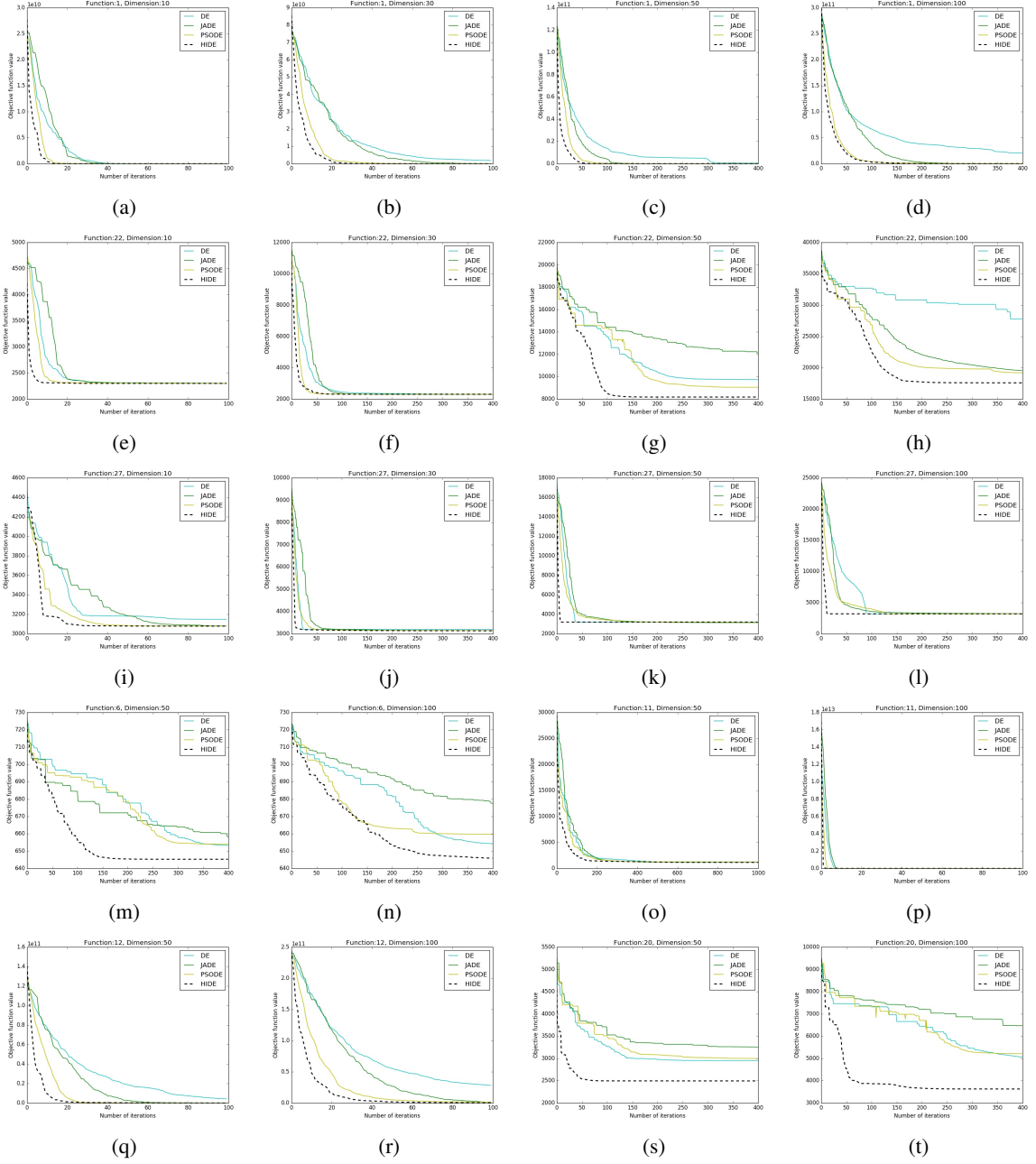


Figure 4: Comparative convergence profiles for test functions from CEC 2017 Benchmark over $D = 10, 30, 50, 100$