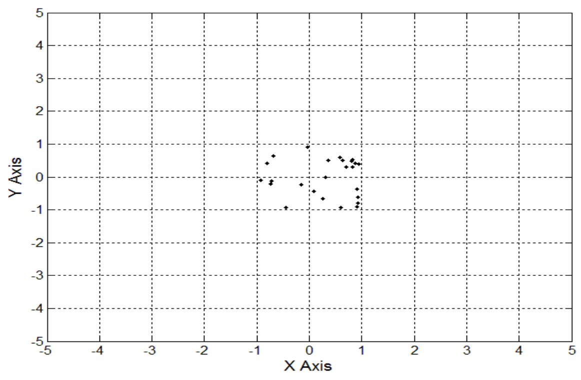
Supplementary Material for:

Swagatam Das, Ankush Mandal, and Rohan Mukkherjee, "An adaptive differential evolution algorithm for global optimization in dynamic environments", *IEEE Transaction of Cybernetics*, Accepted, 2013.

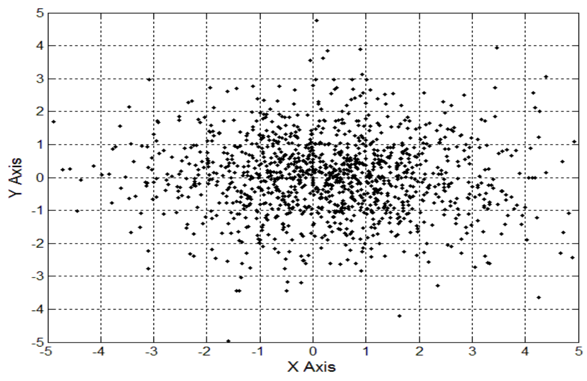
Different sections of this document are numbered as A1, A2, ….etc. In what follows, references of the main paper will be called by their actual number as in the main paper and references added in the suppl. material only will be called as R1, R2,…etc.

A1: Preserving Population Diversity with DE, Brownian and Quantum Individuals - A Comparative View

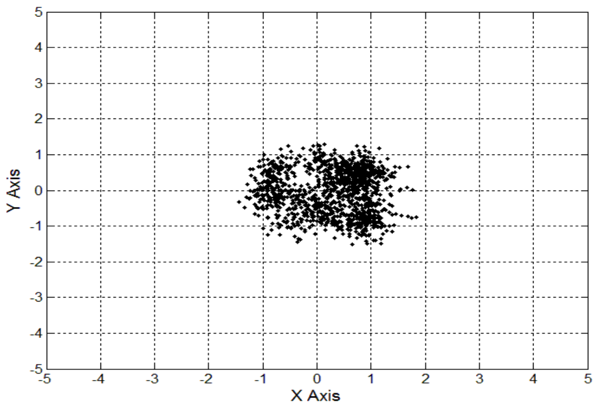
In [23], Wong *et al*. proposed a niching algorithm where, in the exploration stage, random individuals are generated to maintain the diversity of the population for detecting multiple peaks on a static landscape. The algorithm described in [23] generates the random individuals from a uniform distribution and not about any specific member of the current population. During the species-specific explosion stage of this algorithm, multiple copies for each species seed are created and conditionally mutated with variable step-sizes. However, in our proposal, to meet the demands of a dynamically changing functional landscape, we use two kinds of random individuals that are generated around the locally best position in different ways. For quantum individuals, the direction of the perturbation is randomly sampled from a Gaussian distribution but the magnitude is randomly sampled from a uniform distribution. In case of Brownian individuals, both the magnitude and direction are randomly sampled from Gaussian distribution. The experiments undertaken in this study indicate that DE individuals usually explore greater region than both the quantum and Brownian individuals, especially during the early stages of search. Thus, they act like global explorers. On the other hand, initially Brownian individuals are likely to explore a region smaller than DE individuals but greater than the adaptive quantum individuals. Also, each of such explored regions is centered on a local best position. Therefore, Brownian individuals are expected to act like local explorers. To further illustrate this point, Figure S1 depicts how the same initial population spreads in a two-dimensional search space when the individuals are perturbed by DE operators (the double mutation described in the next subsection and the binomial crossover, but no selection), adaptive quantum individual generation process, and Brownian individual generation process. Figure S1 shows the positions of all the intermediate population-members generated over 25-th generation. Section 6.1 experimentally illustrates how these two kinds of individuals, together with the DE-vectors, enhances the efficiency of search on a dynamic fitness landscape.



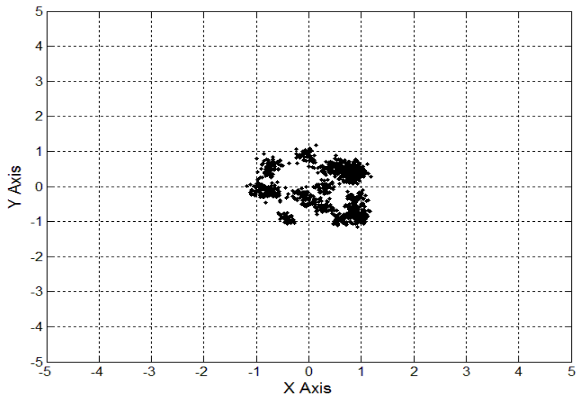
(a) Initial population



(b) Spread of DE population (selection disabled) after 25 generations



(c) Spread of Brownian population after 25 generations



(d) Spread of adaptive quantum population after 25 generations

Fig. S1: A comparative view of the spreads of DE, Brownian, and adaptive quantum populations over 25 generations.

A2: Complete Pseudo-code of the DDEBQ Algorithm

A complete pseudo-code of the proposed algorithm has been provided below as Algorithm S1.

Algorithm S1 : Proposed DDEBQ algorithm

```
1. Set FEs=0, G=1 (Generation counter), C= 0, Change = 0, R = 1, Dis_marginal = 0.8, Cr = 0.9. Set number of
   subpopulation Nsub = 10 and number of individuals in each subpopulation m = 6.
   //set parameters of DDEBQ as per Table 1 of the main paper.
    Set  Age_best = zeros(Nsub,m) and Age_worst = zeros(Nsub,m)
    Set the Update Interval (UI) as discussed in Section 3.5.
// Initialization of population and memory archive //
2. Generate the individuals for all the subpopulations randomly from a uniform distribution within the search range.
3. In the same way, create the memory archive with 10 randomly generated individuals.
// Evolutionary search procedure//
4.    for i = 1 to Nsub
5.     Evaluate all the individuals of the i-th subpopulation and check whether a dimensional change has occurred or not. If a
        dimensional change is detected then perform the required process as described in Section 3.6.
6.    if C = 0 then
7.        Randomly choose two individuals to perform quantum and Brownian individual generation procedures respectively.
8.    else if C = 1 then
9.        Randomly choose one individual to perform Brownian individual generation.
10.   else if C = 2 then
11.       Randomly choose one individual to perform quantum individual generation.
13.   if the best individual within the subpopulation is chosen for Brownian or quantum individual generation then
          discard that choice and randomly choose another individual for the purpose.
15.    for j = 1 to m
16.   if the j-th individual is best individual then increase Age_best(i, j) by 1 and if the j-th individual is the worst  individual
         then increase Age_worst (i, j) by 1.
19.   for ind = 1 to m
20.          if the current individual is chosen for quantum individual generation then perform that and use Repairing rule
```

```
(Section 3.8 of main paper) as necessary.
21.    else if the current individual is chosen for Brownian individual generation then perform that and use the repairing rule
           (Section 3.8 of main paper) as necessary.
22.      else generate a mutant vector V_mut using double mutation strategy (eqns. (8), (9), and (10)).
23.        Generate a trial vector U_trial from V_mut by binomial crossover operation (eqn. (5)). Use Repairing rule (Section
3.8) as necessary.
24.           if the trial vector is better than the current individual then replace the current individual with trial vector. (During
              evaluation of trial vector, check whether a dimensional change has occurred or not. If a dimensional change is
              detected then perform the required process as described in Section 3.6 of main paper).
29.               Replace the memory individuals with the best individuals of the previous subpopulations.
30.               Check whether a dynamic change has occurred or not (by checking whether the fitness value of any memory
                        individual has degraded or not).
31. if a dynamic change has occurred, then set Change to 1.
32. else Change=0.
34. if Change=1 then
35. Set C, Change, Dis_marginal, R, and entries of the Age_best and Age_worst matrices to their initial values.
37. if G is divisible by UI  then
38.  Depending upon the difference between the current global best value and the global best value achieved UI generations
     ago, set the appropriate values for C (Algorithm 2), R, and Dis_marginal.
40. Perform the aging mechanism to all the subpopulations except the one that contains the global best individual as per
     Algorithm 1. Perform re-initialization of any individual or the whole subpopulation as required.
41. Apply exclusion rule to all the subpopulations and reinitialize any subpopulation if required.
42. G = G+1.
43. Check whether the stopping criterion is reached or not. If the stopping criterion is not reached then go to step 4.
```

A3: Parameter Settings for the DDEBQ Algorithm

This section presents results of the experiments carried out to set the parameter values for DDEBQ. Results are shown for six benchmark functions from the GDBG system considering only one change instance T1, although in each case the experiments are conducted with all the 49 test instances of the system. The experiments indicate that the parameters that provide best results for this change type also provide best results for most of the other change types. Note that while investigating the effect of one parameter, values of the other parameters are kept fixed to their best values obtained through the series of empirical experiments.

A3.1 Population Size

The Population size (*Np*) is kept fixed at 60 throughout the search process. The whole population is partitioned into *Nsub* = 10 subpopulations. As mentioned in [7], dimensionality is fixed to 10 for all the test instances with fixed dimensions and is varied in the range 5 to 15 for the test instances with dimensional changes. If the population size is kept very small, then the probability of convergence to local optima may increase and thus the performance of the algorithm may degrade considerably. Hence, a large population size can ensure that the algorithm is performing at its best. However, a large population size needs huge number of FEs at each generation and this in turn degrades the efficiency of the algorithm. In practice, increasing the population size beyond a certain limit degrades the performance of the algorithm. The performance of DDEBQ for different population sizes is presented in Table S1 in terms of mean error values obtained in 20 independent runs. The errors are recorded after same number of FEs for every population size and the frequency of dynamic changes were kept same for all the population sizes. In every test case, numbers of subpopulations are kept fixed at experimentally obtained best value of 10 except in cases of population sizes 10, 20, and 30 where the whole population is divided into 2, 4, and 6 subpopulations respectively to keep the number of individuals per subpopulation reasonable. The population size and number of subpopulations are indicated in the first two columns of Table S1.

Table S1. Performance variation of DDEBQ for different population sizes. Wilcoxon's rank sum test results are indicated in brackets.

Population size	No. of Subpopulations	Mean error value					
		T1 of F1 with No. of peaks=10	T1 of F2	T1 of F3	T1 of F4	T1 of F5	T1 of F6
10	2	4.5023(+)	7.5083(+)	24.3416(+)	2.8873(+)	0.4264(+)	8.1012(+)
20	4	1.3078(+)	3.9779(+)	20.8491(+)	1.9166(+)	0.2649(+)	6.8426(+)
30	6	3.1054e-03(+)	1.5973(+)	16.6021(+)	1.3043(+)	0.1175(+)	6.2112(+)
40	10	4.2264e-05(+)	1.0007(+)	15.0027(+)	1.0759(+)	0.0938(+)	6.0003(+)
50	10	2.8021e-08(+)	0.8725(+)	14.0013(+)	0.9946(+)	0.0761(≈)	5.4062(≈)
60	10	**2.0314e-09**	**0.6679**	**12.7332**	**0.8611**	**0.0611**	**5.3914**
70	10	2.3052e-09(≈)	0.6694(≈)	13.0548(+)	0.8658(≈)	0.0878(+)	5.5448(+)
80	10	1.0305e-08(+)	0.8152(+)	13.9901(+)	0.9274(+)	0.0993(+)	6.0175(+)
90	10	3.1193e-08(+)	0.9365(+)	14.8254(+)	1.2557(+)	0.1329(+)	6.7316(+)
100	10	2.4021e-07(+)	1.0853(+)	16.3195(+)	1.6233(+)	0.2235(+)	7.3051(+)

Increase in population size causes escalation of the number of FEs required per generation. Table S1 indicates that the performance of the algorithm is best at population size 60. Hence, the population size is fixed to Np = 60. In general if the population is divided into a large number of subpopulations, the performance deteriorates because each subpopulation will have very small number of individuals. If the number of individuals in a subpopulation is very small, then the performance of the subpopulation becomes very poor due to lack of diversity of the nearly identical difference vectors. This is same as the result of small population size in DE family of algorithms. However, if the number of subpopulations is reduced, then the effectiveness of the multiple populations becomes negligible and the performance of the algorithm deteriorates. Here the whole population is partitioned into 10 subpopulations as this gives considerably good results on the tested benchmark problems.

A3.2 Control Parameter Adaptation

The value of C is adapted as described in Section 3.5 of the main paper. Table S2 indicates the performance of the algorithm if the values of C were chosen for different ranges. In Table S3, we provide sample results from an experimental setup in which the update interval UI was varied linearly in steps of 5 and 10 to explore how it causes variation in the performance of DDEBQ for a few test configurations of GDBG benchmark set. Experimental results in terms of mean error as well as statistical analysis show that visibly an UI=20 is best suited for DDEBQ and as expected it yields best result for all test instances considered. The results do not seriously degrade with UI set at 15 to 35 but performance hampers with UI too small or too large.

Table S2. Performance of the algorithm for different ranges of the control parameter. The Wilcoxon's rank sum test results are shown in brackets.

C	Range of the difference between the global best fitness value between every UI generations	Mean error value					
		T1 of F1[Number of peaks=10]	T1 of F2	T1 of F3	T1 of F4	T1 of F5	T1 of F6
0	$\geq(PR/5)$	2.8517e-09($\approx$)	0.9912(+)	14.6742(+)	1.4973(+)	0.1254(+)	6.5903(+)
	$\geq(PR/7)$	2.3109e-09($\approx$)	0.7923(+)	13.0964(+)	0.9976(+)	0.0952(+)	5.8249(+)
	$\geq(PR/10)$	**2.0314e-09**	**0.6679**	**12.7332**	**0.8611**	**0.0611**	**5.3914**
	$\geq(PR/15)$	2.2011e-09($\approx$)	0.8230(+)	12.9947(+)	1.0325(+)	0.0983(+)	5.9331(+)
	$\geq(PR/20)$	3.2926e-09($\approx$)	1.6357(+)	13.8723(+)	1.4576(+)	0.1376(+)	6.7982(+)
1	$<(PR/10)$and $\geq(PR/35)$	2.8754e-09($\approx$)	1.0814(+)	13.6423(+)	1.5876(+)	0.1598(+)	6.0136(+)
	$<(PR/10)$and $\geq(PR/45)$	2.1045e-09($\approx$)	0.8106(+)	12.9886(+)	0.9662(+)	0.0854(+)	5.6420(+)
	$<(PR/10)$and $\geq(PR/50)$	**2.0314e-09**	**0.6679**	**12.7332**	**0.8611**	**0.0611**	**5.3914**
	$<(PR/10)$and $\geq(PR/55)$	2.1176e-09($\approx$)	0.8572(+)	13.1297(+)	1.2125(+)	0.0923(+)	5.6604(+)
	$<(PR/10)$and $\geq(PR/65)$	2.8986e-09($\approx$)	1.1034(+)	14.9866(+)	1.6574(+)	0.1428(+)	6.2139(+)
2	$<(PR/40)$	2.2253e-09($\approx$)	0.7898(+)	13.2712(+)	1.0346(+)	0.09235(+)	5.8462(+)
	$<(PR/50)$	**2.0314e-09**	**0.6679**	**12.7332**	**0.8611**	**0.0611**	**5.3914**
	$<(PR/55)$	2.0653e-09($\approx$)	0.8556(+)	13.1986(+)	1.1876(+)	0.0967(+)	5.9761(+)
	$<(PR/60)$	2.3191e-09($\approx$)	1.0465(+)	14.6246(+)	1.2987(+)	0.1287(+)	6.4653(+)
	$<(PR/70)$	3.1085e-09($\approx$)	1.9563(+)	15.9098(+)	1.5876(+)	0.1463(+)	6.7203(+)

Table S3: Performance variation of DDEBQ for different update intervals. Wilcoxon's rank sum test results are indicated in brackets

Update Interval(UI)	Mean error value					
	T1 of F1 with 10 peaks	T1 of F2	T1 of F3	T1 of F4	T1 of F5	T1 of F6
5	5.6712e-03(+)	4.6592(+)	21.4712(+)	5.2306(+)	3.1565(+)	9.2414(+)
10	8.7724e-08(+)	0.8027(+)	13.3456(+)	0.9569(+)	0.2334(+)	5.7058(+)
15	5.4012e-09($\approx$)	0.6825($\approx$)	12.9745($\approx$)	1.0512(+)	0.1023($\approx$)	5.3914($\approx$)
20	**2.0314e-09**	**0.6679**	**12.7332**	**0.8611**	**0.0611**	**5.2093**
25	4.2702e-09($\approx$)	0. 6834($\approx$)	12.7541($\approx$)	0.8874($\approx$)	0.0770($\approx$)	5.4912($\approx$)
30	8.1503e-09($\approx$)	0.6860($\approx$)	12.9837($\approx$)	0.8972($\approx$)	0.0849($\approx$)	5.9424(+)
35	9.3423e-09($\approx$)	0.7231(+)	12.8735($\approx$)	0.9472($\approx$)	0.0768($\approx$)	5.5624($\approx$)
40	8.1503e-06(+)	3.6860(+)	16.9837(+)	2.9261(+)	0.1849(+)	7.6723(+)
50	7.9243e-02(+)	2.9433(+)	21.4782(+)	3.2019(+)	5.7038(+)	12.9013(+)

A3.3 Quantum Individual Adaptation

The parameter R is the radius within which the quantum individuals are generated. In DDEBQ, R is made adaptive in nature. If C is 0, then R is set to 1. If C is 2, then R changes according to the following rule depending upon the difference (*Diff*) of global best objective function values before and after *UI* generations.

$$R = Diff \cdot \log_{10}\left(10 + \frac{PR}{50 \cdot Diff}\right).$$

Here *PR* has the same meaning as mentioned previously. It is clear that as the difference decreases, R also decreases. But to prevent R from decreasing rapidly, *Diff* is multiplied with a log term. As *Diff* decreases, the value of the log term increases. When C becomes 2, value of the radius R is updated according to the above relation. For $C = 2$, the difference *Diff* falls below (*PR*/50). The reason behind choosing this value is that when the difference is much larger than (*PR* /50), the quantum individuals generated within unity radius are good for maintaining the diversity within the subpopulation. But when the difference becomes small, i.e. when the search process converges, the radius should be gradually contracted so that the gradually decreasing diversity, maintained by the quantum individuals, can be used for locating the global optimum with a high precision. If the diversity is too high, it can be detrimental for locating the global optimum precisely. On the other hand, if the diversity becomes too low, the search process may terminate prematurely and this in turn can result in high error values. Figure S2 shows how R varies with number of FEs for function F1 (number of peaks = 10) with change type T1. After every 100,000 FEs, due to the occurrence of a dynamic change, R is reinitialized to 1. However, in order to show that $Diff$ = (PR /50) can be an optimal setting for starting this adaptation procedure; the performance of DDEBQ is summarized in Table S4 for starting the adaptation of quantum individuals from different values of $Diff$.

Table S4. Performance of the algorithm for starting the adaptation of quantum individuals from different values of *Diff*. The Wilcoxon's rank sum test results are shown in brackets.

Value of *Diff* from which the adaptation of quantum individuals starts	Mean error value					
	T1 of F1[Number of peaks=10]	T1 of F2	T1 of F3	T1 of F4	T1 of F5	T1 of F6
(PR /30)	2.9912e-09($\approx$)	1.3873(+)	14.1826(+)	1.6298(+)	1.1120(+)	6.2127(+)
(PR /40)	2.2681e-09($\approx$)	0.8923(+)	13.0236(+)	0.9923(+)	0.0845(+)	5.6193(+)
(PR /50)	**2.0314e-09**	**0.6679**	**12.7332**	**0.8611**	**0.0611**	**5.3914**
(PR /60)	2.3638e-09($\approx$)	0.9013(+)	13.0365(+)	1.0452(+)	0.9702(+)	5.7072(+)
(PR /70)	2.7264e-09($\approx$)	1.4985(+)	14.4721(+)	1.7265(+)	1.2764(+)	6.3816(+)

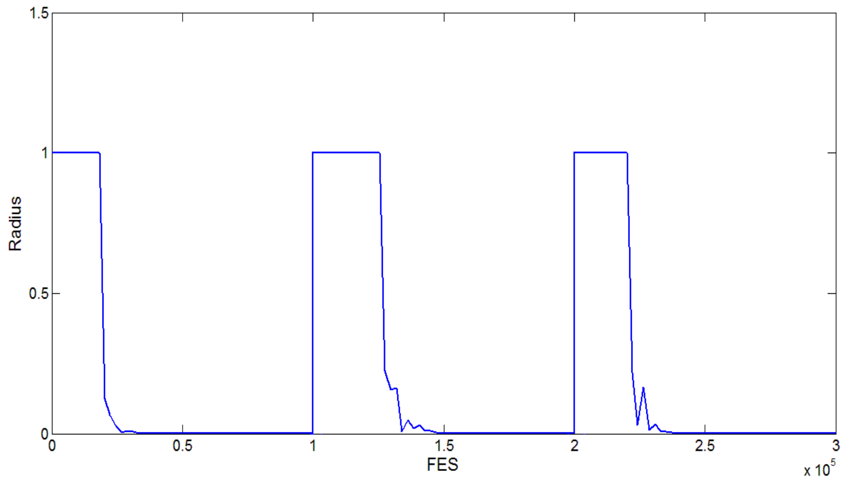


Fig S2: Variation of radius R with number of FEs for F1 (with number of peaks = 10) with change type T1

A3.4 Marginal Value of Distance between Best Individuals

As mentioned earlier, the marginal value of the distance between the best individuals of two different subpopulations is calculated according to the following rule of thumb: $Dis\_marginal = SR / (Nsub \cdot D)$, where SR (search range)=10, $Nsub$(number of subpopulations)=10 and D(dimension)=10. Thus, the maximum margin of the distance becomes 0.1. In DDEBQ, when C is 0, the marginal value for the distance between two best individuals of different subpopulations is set to 0.08. However, if C becomes 1 or 2, then the marginal value is set to 0.03. Actually, the search process is converging when C becomes 1 or 2. Hence, in order to help the convergence procedure, the algorithm reduces the marginal value for distance between the best individuals of the different subpopulations. Table S5 lists the results of DDEBQ for different values of $Dis\_marginal$.

Table S5. Performance of the algorithm for different marginal values of distance between best individuals. The Wilcoxon's rank sum test results are shown in brackets.

Control Parameter	*Dis_marginal*	Mean error value					
		T1 of F1[No. of peaks=10]	T1 of F2	T1 of F3	T1 of F4	T1 of F5	T1 of F6
0	0.1	2.2896e-09(≈)	0.8420(+)	12.9634(+)	0.9936(+)	0.0846(+)	5.6105(+)
	0.09	2.1237e-09(≈)	0.7523(+)	12.8037(+)	0.9021(+)	0.0745(+)	5.5021(+)
	0.08	**2.0314e-09**	**0.6679**	**12.7332**	**0.8611**	**0.0611**	**5.3914**
	0.07	2.0642e-09(≈)	0.7013(≈)	12.7990(+)	0.8936(+)	0.0729(+)	5.5264(+)
	0.06	2.1383e-09(≈)	0.8349(+)	12.8936(+)	0.9945(+)	0.0863(+)	5.6319(+)
1 or 2	0.05	2.2336e-09(≈)	0.8531(+)	12.8153(+)	0.9925(+)	0.0862(+)	5.7235(+)
	0.04	2.1873e-09(≈)	0.7130(+)	12.7561(≈)	0.9013(+)	0.0723(+)	5.5105(+)
	0.03	**2.0314e-09**	**0.6679**	**12.7332**	**0.8611**	**0.0611**	**5.3914**
	0.02	2.1723e-09(≈)	0.7537(+)	12.8015(+)	0.8902(+)	0.0750(+)	5.6021(+)
	0.01	2.2901e-09(≈)	0.8343(+)	12.9346(+)	0.9846(+)	0.087(+)	5.6933(+)

A3.5 Weight Factor for Double Mutation Strategy

The weight factor used in the second stage of double mutation strategy is set to 0.1. Here the performance of the algorithm for different values of the weight factor is investigated. In Table S6, the results are presented in terms of the mean error values.

According to the rule mentioned in CEC 2009 DOP competition technical report [7], the parameter values should remain same for all test instances. Therefore, finally the value of the weight factor is fixed to 0.1. Actually, when the weight factor is too low, the perturbation becomes very small and the subpopulations lack in diversity. This, in turn hinders the search process. When the weight factor is too high the perturbation becomes large and the exploration process is hindered.

Table S6. Performance of the algorithm for different values of the weight factor. The Wilcoxon's rank sum test results are shown in brackets.

Weight Factor (ω)	Mean Error Value					
	T1 of F1[No. of peaks=10]	T1 of F2	T1 of F3	T1 of F4	T1 of F5	T1 of F6
0.025	**1.7356e-09**	0.7421(+)	13.7863(+)	0.9735(+)	0.0895(+)	5.8754(+)
0.05	1.8459e-09(≈)	0.6991(+)	13.2978(+)	0.9467(+)	0.0725(≈)	5.6859(+)
0.1	2.0314e-09(≈)	**0.6679**	**12.7332**	**0.8611**	**0.0611**	**5.3914**
0.2	2,0523-09(≈)	0.6796(≈)	13.5638(+)	0.8746(≈)	0.0887(+)	5.6873 (+)
0.3	2.0803e-09(≈)	0.7134(+)	13.8084(+)	0.9026(+)	0.0957(+)	5.8976(+)

A3.6 Crossover Rate *Cr*

The Crossover Rate *Cr* is kept fixed at 0.9 throughout the search process. Some experiments are conducted in order to see how the performance of the algorithm varies as *Cr* takes different constant values. The results are tabulated in Table S7. From Table S7, it is clear that keeping *Cr* constant at 0.9 yields the best results.

Table S7. Performance of the algorithm for different *CR* values. The Wilcoxon's rank sum test results are shown in brackets.

Values of *Cr*	Mean Error Value					
	T1 of F1[Number of peaks=10]	T1 of F2	T1 of F3	T1 of F4	T1 of F5	T1 of F6
0.7	2.9735e-09(≈)	1.2024(+)	14.3106(+)	1.4649(+)	0.1538(+)	6.3820(+)
0.8	2.4574e-09(≈)	0.9012(+)	13.5932(+)	1.2764(+)	0.1027(+)	5.9769(+)
0.85	2.1158e-09(≈)	0.7354(+)	13.1843(+)	0.9953(+)	0.0834(+)	5.5665(+)
0.9	**2.0314e-09**	**0.6679**	**12.7332**	**0.8611**	**0.0611**	**5.3914**
0.95	2.2153e-09(≈)	0.8012(+)	13.1011(+)	0.9656(+)	0.0786(+)	5.5242(+)

A4: Experimental Validation of Different Components of DDEBQ

DDEBQ incorporates several algorithmic strategies like generation of the Brownian and quantum individuals, double mutation strategy, adaptation rule for the scale factor *F*, exclusion, and aging mechanism. This section illustrates, through experimental results, how the performance of the algorithm varies on removing such strategies and/or replacing them with some alternative strategies. Such study at least empirically investigates the need for these modifications and also demonstrates how when taken together in the complete framework of DDEBQ, they lead to superior performance of the algorithm on the tested DOP instances. Note that for all experimental results reported here, when a particular modification is considered, rests of the strategies are kept unaltered. Also the best suited set of other parameters are employed as discussed in Section A4.

A4.1 The Brownian and Quantum Individuals

Table S8 compares the results of the original DDEBQ algorithm with the results of the dynamic DE scheme without any Brownian or quantum individuals in any subpopulations. The same table also presents the performance of the dynamic DE framework when it uses either Brownian or adaptive quantum individuals with the DE individuals but not both. It can be clearly observed that the simultaneous incorporation of the Brownian and quantum individuals in subpopulations can significantly improve the performance of the algorithm.

Table S8. Performance of the proposed algorithm with and without Brownian and quantum individuals. The Wilcoxon's rank sum test results are shown in brackets.

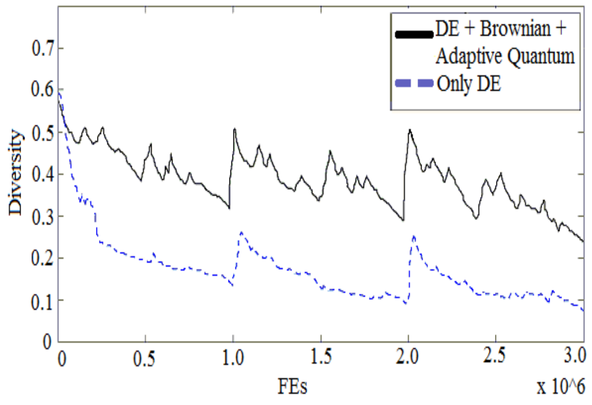
Schemes	Mean Error Value					
	F1 with T1[No. of peaks=10]	F2 with T1	F3 with T1	F4 with T1	F5 with T1	F6 with T1
Dynamic DE without the Brownian and quantum individuals	0.0294(+)	0.9639(+)	15.3741(+)	1.2592(+)	0.1438(+)	7.0271(+)
DDEBQ with Brownian and quantum individuals	**2.0314e-09**	**0.6679**	**12.7332**	**0.8611**	**0.0611**	**5.3914**
Dynamic DE with Brownian individuals	3.0635e-04(+)	0.7879(+)	13.8231(+)	0.9674(+)	0.07801(+)	5.5503(+)
Dynamic DE with adaptive quantum individuals	6.0284e-03(+)	0.8203(+)	13.9947(+)	1.0043(+)	0.07882(+)	5.6049(+)

Brownian and Quantum individuals play a key role in maintaining sufficient diversity in the population while responding to the environmental changes. In order to investigate DDEBQ's ability to maintain the diversity, two sample diversity plots are provided in Figure S3 for functions F3 and F5. During the search process, the diversity is calculated at each generation by the following formula [R1]:

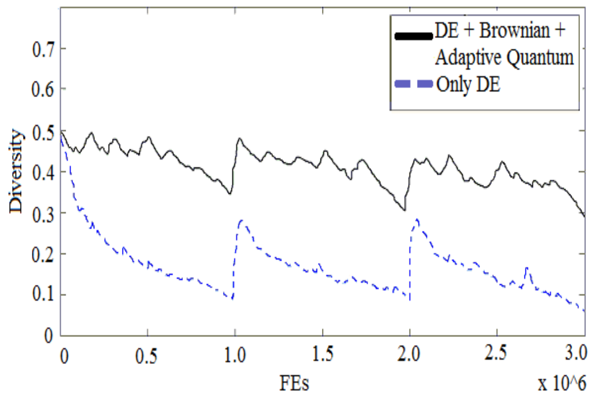
$$Diversity = \frac{1}{Np \times L} \sum_{i=1}^{Np} \sqrt{\sum_{d=1}^{D} (x_{i,d} - \bar{x}_d)^2},$$

where L is the length of the longest diagonal in search space and $\bar{x}_d$ is the d-th dimensional value of the average point. The above diversity measurement method is different from the entropy based diversity measurement. It is a "distance-to-average-point" measurement. As can be seen from the figures, DDEBQ maintains the diversity to a better level than the dynamic DE scheme without quantum and Brownian individuals. It can be also seen that the diversity reduces to a lower level before each dynamic change and this is mainly due to the action of the control parameter *C* (as it controls the diversity and helps us to achieve better convergence characteristics). There are many sharp ups and downs in the diversity variations plotted. Such high level of diversity variations within the interval of two dynamic changes are due to re-initializations performed by the aging mechanism.

Performance of the algorithm is also monitored by replacing the adaptive quantum individuals with the general quantum individuals. The comparative results given in Table S9 indicate that the adaptive nature of the quantum individuals further improves the accuracy levels of the algorithm.



(a) For function F3 with change type T1



(b) For function F5 with change type T1

Fig. S3. Sample diversity plots for two functions

Table S9. Performance of the algorithm with general quantum individuals and adaptive quantum individuals. The Wilcoxon's rank sum test results are shown in brackets.

Schemes	Mean Error Value					
	F1 with T1[No. of peaks=10]	F2 with T1	F3 with T1	F4 with T1	F5 with T1	F6 with T1
With general quantum individuals	0.0037(+)	0.6952(+)	13.8247(+)	0.9673(+)	0.0962(+)	5.7734(+)
With adaptive quantum individuals	**2.0314e-09**	**0.6679**	**12.7332**	**0.8611**	**0.0611**	**5.3914**

A4.2 Performance without using Double Mutation Strategy

Table S10 presents the comparative performance of DDEBQ with four distinct mutation strategies including the proposed double mutation strategy. It is clear from the entries of Table S10 that the double mutation strategy yields statistically superior results as compared to three other widely used DE-type mutation strategies.

Table S10. Performance of the algorithm with different mutation schemes. The Wilcoxon's rank sum test results are shown in brackets.

Mutation Strategy	Mean Error Value					
	F1 with T1 [10 peaks]	F2 with T1	F3 with T1	F4 with T1	F5 with T1	F6 with T1
DE/rand/1	0.0201(+)	0.7972(+)	13.4706(+)	1.0008(+)	0.0803(+)	5.7739(+)
DE/best/1	0.0262(+)	0.8265(+)	13.8452(+)	1.2043(+)	0.1077(+)	5.9153(+)
DE/current-to-best/2	0.0189(+)	0.7742(+)	13.4071(+)	0.9973(+)	0.0819(+)	5.7366(+)
Double mutation	**2.0314e-09**	**0.6679**	**12.7332**	**0.8611**	**0.0611**	**5.3914**

A4.3 Adaptation Strategy for Scaling Factor *F*

Table S11 contrasts the performance of DDEBQ with the suggested adaptation rule for the scaling factor *F* against two constant and one randomized settings of *F*. From the results of Table S11, it is clear that our proposed adaptation of *F* yields much better result than the other three settings.

Table S11. Performance of the algorithm for different settings of scaling factor *F*. The Wilcoxon's rank sum test results are shown in brackets.

Setting of *F*	Mean Error Value					
	F1 with T1[No. of peaks=10]	F2 with T1	F3 with T1	F4 with T1	F5 with T1	F6 with T1
Constant (0.5)	0.0203(+)	0.7623(+)	13.9932(+)	1.1044(+)	0.0935(+)	5.7487(+)
Constant (0.8)	0.0258(+)	0.7988(+)	14.3375(+)	1.2428(+)	0.0989(+)	5.8346(+)
(0.3+rand*0.7)	0.0114(+)	0.7345(+)	13.8423(+)	0.9992(+)	0.0873(+)	5.6953(+)
Proposed setting (eqns. 10a and 10b)	**2.0314e-09**	**0.6679**	**12.7332**	**0.8611**	**0.0611**	**5.3914**

A4.4 Exclusion Rule

The necessity of the exclusion rule is experimentally investigated by presenting the simulation results obtained by running DDEBQ with and without the exclusion rule. Table S12 shows the corresponding simulation results. In case of multi-population approaches of solving DOPs, the aim is to have each subpopulation around a different peak and the exclusion rule helps the population to spread each subpopulation around different basins of attraction. As can be seen from the entries of Table S12, by covering all the peaks in the search space, the exclusion rule enhances the quality of the solutions and thereby decreases the mean error value.

Table S12. Performance of DDEBQ with and without Exclusion Rule. The Wilcoxon's rank sum test results are shown in brackets.

Schemes	Mean Error Value					
	F1 with T1[10 peaks]	F2 with T1	F3 with T1	F4 with T1	F5 with T1	F6 with T1
DDEBQ without Exclusion Rule	1.4592e-03(+)	0.8922(+)	13.7742(+)	0.9336(+)	0.0971(+)	5.7862(+)
DDEBQ with Exclusion Rule	**2.0314e-09**	**0.6679**	**12.7332**	**0.8611**	**0.0611**	**5.3914**

A4.5 Aging Mechanism

Actually, the effect of applying the aging mechanism is more visible in case of functions having very rough search space, such as Composition of Rastrigin's function (F3), Composition of Griewank's function (F4), Composition of Ackley's function (F5) etc. From the previous discussion about the aging mechanism in Section 3.4 of the main paper, it can be perceived that the aging mechanism helps the search process to get rid of stagnating at a local optimum. If the search space is very rough, then the chance of stagnating at any local optimum is also high. Thus, the aging mechanism is more effective in case of rugged functional landscapes. A comparison of the results is provided in Table S13 and the results indicate how the aging mechanism greatly improves the performance of the algorithm especially in case of optimizing dynamic and multimodal objective functions.

Table S13. Performance of the algorithm with and without Aging Mechanism. The Wilcoxon's rank sum test results are shown in brackets.

Schemes	Mean Error Value					
	F1 with T1[10 peaks]	F2 with T1	F3 with T1	F4 with T1	F5 with T1	F6 with T1
Without Aging	1.8974e-08(+)	0.8152(+)	17.3419(+)	1.6027(+)	0.1261(+)	6.0182(+)
With Aging	**2.0314e-09**	**0.6679**	**12.7332**	**0.8611**	**0.0611**	**5.3914**

A5: Comparative Performance on MPB Problems

The MPB (Moving Peaks Benchmarks) problem set [29] has been widely used for benchmarking dynamic optimizers. Within the MPB framework, the optima can change with respect to three features: location, height, and width of the peaks. Three standardized sets of parameter settings or scenarios have been defined for better comparisons of different algorithms. Details of the parametric setups used in these three scenarios can be found in http://people.aifb.kit.edu/jbr/MovPeaks/. The details about the scenario 2 parametric settings used in the experiments are shown in Table S14.

Table S14. Default Settings for the MPB Problem Scenario 2

Parameters	**Value**
Number of peaks, *p*	10 , 50
Change frequency, *U*	5000
Height severity	7.0
Width severity	1.0
Peak shape	Cone

Basic function	No
Shift length, s	1.0
Number of dimensions, D	5 , 50
Correlation coefficient, λ	0
S	[0, 100]
H	[30.0, 70.0]
W	[1, 12]
I	50.0

Table S15 presents the comparison of the offline errors and the adaptability metric values achieved over scenario 2 of MPB problems and 2 other variations of it. The mean offline errors and standard deviations achieved by 50 independent runs of DDEBQ are compared with the Evolutionary Swarm Cooperative Algorithm (ESCA) [R2], the Hooke-Jeeves based memetic algorithm [R3], and the Multi-phase Multi-individual External Optimization (MMEO) Algorithm [R4]. The table clearly indicates that DDEBQ achieved statistically better result than the other three algorithms tested.

Table S15: Mean error values achieved by DDEBQ and three peer algorithms on MPB problem of scenario 2 and some variants of scenario 2. The Wilcoxon's rank sum test results of comparing DDEBQ with contender algorithms are indicated in parentheses. Adaptability metric values are indicated in the third brackets in each cell.

Algorithms	Offline errors for scenario 2	Offline errors for scenario 2 with 50 peaks	Offline errors for scenario 2 with 50 dimensions
DDEBQ	**0.0903 [4.9781]**	**0.2462 [3.3178]**	**2.5861 [8.4819]**
ESCA	1.5369 (+) [8.9618]	1.6437 (+) [7.4598]	10.5653 (+)[20.9823]
HJMA	0.2305 (+) [6.8302]	0.5714 (+) [5.8714]	5.6755 (+) [10.9430]
MMEO	0.6591 (+) [8.2106]	1.2459 (+) [9.3085]	49.8564 (+) [14.9834]

SUPPLEMENTARY REFFERENCES

[R1] J. Hu, J. Zeng, and Y. Tan, "A diversity-guided particle swarm optimizer for dynamic environments", in Bio-*Inspired Comput. Intell. And Applications*, Lecture Notes in Computer Science, Vol. 9, No. 3, pp. 303-317, June 2005.

[R2] R. I. Lung and D. Dumitrescu, "Evolutionary swarm cooperative optimization in dynamic environments", *Nature Inspired Cooperative Strategies for Optimization* (NICSO 2007), Vol. 129/2008, pp. 105-114, 2008.

[R3] I. Moser and R. Chiong, "A Hook-Jeeves Based Memetic Algorithm for Solving Dynamic Optimisation Problems", *in the Proceedings of the 4th International Conference on Hybrid Artificial Intelligence Systems* (*HAIS*), LNAI 5572, pp. 301-309, 2009.

[R4] I. Moser and T. Hendtlass, "A simple and efficient multi-component algorithm for solving dynamic function optimization problems", *IEEE Congress on Evolutionary Computation* (*CEC* 2007), pp. 252 – 259, Singapore, Sept. 2007.