DynDE: a Differential Evolution for Dynamic Optimization Problems

Rui Mendes
Centro de Ciências e Tecnologias de Computação
Universidade do Minho
Braga, Portugal
azuki@di.uminho.pt

Arvind S. Mohais
Department of Mathematics and Computer Science
The University of the West Indies
St. Augustine, Trinidad
amohais@fsa.uwi.tt

Abstract- This paper presents an approach of using Differential Evolution (DE) to solve dynamic optimization problems. Careful setting of parameters is necessary for DE algorithms to successfully solve optimization problems. This paper describes DynDE, a multi-population DE algorithm developed specifically to solve dynamic optimization problems that doesn't need any parameter control strategy for the F or CR parameters. Experimental evidence has been gathered to show that this new algorithm is capable of efficiently solving the moving peaks benchmark.

1 Introduction

We are often confronted with dynamic optimization problems in real life. Price fluctuations are one of the best known causes of changes in optimization problems. The good news is that, even though the situation doesn't remain static for a long time, the changes are slight. A very good solution continues to be a high quality one, even if it is no longer the best one. And if the best solution is superseded, it was by another good solution.

This paper is concerned with this class of problems: where the function landscape suffers slight changes as time progresses. The moving peaks benchmark (MPB) [3] was developed to simulate these problems. The landscape changes every fixed number of function evaluations by slightly moving the peaks and changing their height and width.

This problem was approached by developing a variant of Differential Evolution (DE) [7, 6, 4, 5]. To solve this specific problem, a multi-population version of the DE algorithm was developed, with the goal of maintaining each of the populations in one of the peaks. Thus, when the summit of the peak moves, the population moves with it.

To be able to detect changes in the fitness landscape, the best solution in each population is re-evaluated each iteration and, if its fitness changed, the entire population is re-evaluated.

Previous research indicated that, instead of reinitializing part of the population when the problem changes, it is preferable to have some individuals in each population following a diversity increasing scheme [1]. Thus, as a diversity preserving measure, instead of the normal DE rules some of the individuals in the population either update their positions stochastically around the best position of the population or introduce some form of entropy into the functioning of the schemes.

It is important to keep each of the populations on a different peak. For that, it is necessary to have a mechanism that will detect when two populations are on the same peak and that expels the less performing one [2, 1].

The DE algorithm makes use of three important parameters: F, the weighting coefficient that is used to generate new trial solutions; CR, the crossover probability that is used to determine how much of a trial solution should be adopted into a currently existing one; as well as the DE scheme that defines how a given individual will interact with others. It has been found [9] that the performance of the DE algorithm on a given function is sensitive to the values of F and CR, and furthermore that varying the values of F and CR during the course of a run can improve performance [8].

To begin the study, comprehensive experiments were performed to compare the behavior of the DE algorithm using fixed values for the F, CR and the scheme parameters. It was recognized that these three parameters could be tuned for the MPB to obtain highly efficient optimization performance. However, this task is time consuming. It was realized that using uniform random values from F and CR on a per-individual and per-dimension basis respectively produced equally good results.

The results obtained show that DynDE is competitive with some of the other approaches presented by other researchers in the area [2, 1].

In an attempt to design an algorithm that would adapt to the function being optimized, each individual was also assigned it's own DE scheme, chosen randomly at the beginning of the run. This approach did not produce results as good as using a specific scheme. However, there is no need to fine-tune any parameter whatsoever and the results are still of acceptable quality.

The organization of this paper is as follows. In section 2, an overview of the basic DE algorithm is given and various DE schemes are also described. Section 3 describes the MPB and presents some of the approaches for dynamic problems that were taken into consideration when creating the DynDE algorithm. Section 4 describes the details of DynDE. Section 5 presents the experimental results and analyzes the results and conclusions follow in section 6.

2 The Differential Evolution Algorithm

In this section we present the DE algorithm and the different schemes that are used.

2.1 Differential Evolution

Differential Evolution is a population-based approach to function optimization, whose main strategy is to generate a new position for an individual by calculating vector differences between other randomly selected members of the population.

Given a function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ to be minimized, a DE begins by randomly generating p n-dimensional vectors. These vectors (called individuals) form a population that will evolve over the course of the algorithm's run.

The algorithm then proceeds to manipulate the population until a termination criterion is met. In situations where the minimum value of the function being optimized is known, the termination criterion can be that a vector is generated whose evaluation under the objective function is sufficiently close to the known minimum. In other situations, the termination condition can be that a fixed number of function evaluations have elapsed or no sufficient improvement is achieved.

The following is an outline of a variant of the DE algorithm called DE/rand/1 (see subsection below) that uses a binomial crossover [5]. For clarity, the computation of the new trial vector has been shown separately from the crossover operation that selects only some of the dimensions of the trial vector.

1. Initialize the population.
2. Evaluate the population.
3. Generate a new population where each candidate individual is generated in parallel according to:
   (i) Randomly select 3 distinct individuals r_1, r_2, r_3 from the population that are different from i.
   (ii) Generate a trial vector based on the formula $\vec{t} = \vec{x}_{r_1} + F \cdot (\vec{x}_{r_2} - \vec{x}_{r_3})$
   (iii) Incorporate coordinates of this vector with probability CR, using at least one coordinate.
   (iv) Evaluate the candidate.
   (v) Use the candidate in the new generation if it is at least as good as the current individual.
4. Loop to 3 unless the termination criterion is met.

F and CR are the control parameters. F guides the amplitude of the influence of the difference vector and CR the amount of the candidate solution that is used.

2.2 DE Schemes

Various schemes are typically in use for DEs [4]. Each scheme varies with respect to the number of other random individuals that are used to construct a new trial vector, as well as with respect to whether or not the current individual or the global best individual will be used as part of that computation. Six schemes are considered in this paper. These are shown below along with the corresponding trial vector generation formula. x_j, $1 \leq j \leq 5$ represent distinct randomly selected individuals that are different from the current individual x_i and x_b is the best individual.

DE/rand/1	$\vec{x}_1 + F(\vec{x}_2 - \vec{x}_3)$
DE/rand/2	$\vec{x}_1 + F(\vec{x}_2 + \vec{x}_3 - \vec{x}_4 - \vec{x}_5)$
DE/best/1	$\vec{x}_b + F(\vec{x}_2 - \vec{x}_3)$
DE/best/2	$\vec{x}_b + F(\vec{x}_2 + \vec{x}_3 - \vec{x}_4 - \vec{x}_5)$
DE/rand-to-best/1	$\vec{x}_1 + \lambda(\vec{x}_b - \vec{x}_1) + F(\vec{x}_2 - \vec{x}_3)$
DE/current-to-rand/1	$\vec{x}_i + \lambda(\vec{x}_1 - \vec{x}_i) + F(\vec{x}_2 - \vec{x}_3)$
DE/current-to-best/1	$\vec{x}_i + \lambda(\vec{x}_b - \vec{x}_i) + F(\vec{x}_1 - \vec{x}_2)$

3 Dynamic Problems

It is believed that dynamic problems are characterized by small changes in the width, height and location of the peaks. The fitness landscape changes slowly over time and the algorithm is supposed to adapt to the new environment.

Th philosophy of the approach used in this paper is to use several populations and to try to keep each population on a different peak. Ideally, if one can have one population per peak, when a peak moves the population is able to keep up with the change. If another peak becomes the best then the population that resides on that peak will quickly find the best position and the online performance will not suffer greatly because of this change.

3.1 Exclusion

The philosophy of the approach of using multiple populations to solve dynamic optimization problems is to have each population on a different peak. There is the necessity of ensuring that no two sub-populations share the same peak.

A strategy called *exclusion* [2, 1] was developed to enforce this. At each iteration, the best individuals of each population are compared spatially. If any of them are too close to each other, i.e. if they have a separating distance less than some predefined amount r_{excl} then only the population with the highest performance is kept, and the other is marked for re-initialization.

The setting of the exclusion radius r_{excl} is performed according to a rule of thumb: assuming that all peaks are evenly distributed in the search space, the linear diameter of the basin of attraction will be used as an indicator. Assuming that we have d dimensions with ranges X and p peaks, r_{excl} is given by half of the diameter:

$$r_{excl} = \frac{X}{2p^{\frac{1}{d}}} \tag{1}$$

3.2 Increasing the Diversity

Maintaining diversity is especially important for dynamic optimization problems since the optimum of such a function changes over time and if the population is clustered in a tight region, the individuals may not be able to detect a change in the function landscape.

One of the commonly used procedures for maintaining diversity is to re-initialize the population once a change is detected. However, this procedure introduces a severe loss

of information. A more sensible approach would be to re-evaluate the individuals and to introduce more diversity so that each sub-population can be spread around the area that encompasses the possible change.

3.2.1 Quantum Individuals

Blackwell and Branke [1] suggest having individuals that do not follow the same rules as the others and that are simply generated in the general area of the best individual. They developed the idea of quantum particles, analogous to particles in quantum mechanics, whose positions are probabilistically defined. The individual is generated inside a ball centered on the global best position.

The stochastic generation of a point inside a ball centered at the global best position $\vec{p}_g$ of radius r_{cloud} is computed as follows:

(i) Generate $x_i \sim \mathcal{N}[0,1]$ for $1 \leq i \leq d$.

(ii) Compute the distance of the point to the origin $dist = \sqrt{\sum_{i=1}^{i=d} x_i^2}$.

(iii) Determine the actual radius that is going to be used $r = U[0, r_{cloud}]$.

(iv) The new point will be $\vec{p}_g + \frac{r\vec{x}_i}{dist}$.

It should be noted that this method has a higher probability of generating points near the best than at the limit of r_{cloud}. Besides, this probability increases with dimensionality d.

3.2.2 Brownian Individuals

This idea is quite similar to the one of Quantum individuals. The advantage is that it is simpler. Instead of generating the points inside a ball, we use a Gaussian hyper-parallelepiped. What we do is to generate the new point around the best individual in the sub-population by adding on each coordinate a random variable sampled from the normal distribution.

The stochastic generation of a Brownian individual centered at the global best position $\vec{p}_g$ with standard deviation σ is given by:

$$\vec{p}_g + \vec{\mathcal{N}}(0, \sigma) \tag{2}$$

3.2.3 Entropic Differential Evolution

This idea is inspired by the previous one. Instead of creating a completely different way of generating new solutions for some individuals in the population, the goal is to simply introduce some added entropy to the way the new solutions are generated by DE.

Once a change is detected, a vector $\vec{r} \sim \vec{\mathcal{N}}(0, \sigma)$ is added to the individual of the next generation. Thus, after the scheme has been applied and the individual $\vec{x}$ is about to be inserted in the next generation, the individual $\vec{x} + \vec{\mathcal{N}}(0, \sigma)$ is inserted instead.

4 DynDE

This paper proposes a new multi-population DE that can use dynamic F and CR parameters, as well as an individual-based DE scheme. One of the ideas presented in section 3.2 is included to deal with the dynamic nature of the problem.

F and CR may be set to fixed values, or may be randomly selected from the uniform distribution. A new strategy is to allow each individual to use its own scheme. When a new trial individual is being computed, the scheme of the individual is used to determine how many other individuals to select and how to interact with them.

4.1 Differential Evolution Individuals

When updating a particle i, this new DynDE algorithm operates as follows:

(i) If dynamic CR is being used, select $CR \sim U[0,1]$.

(ii) Randomly select $2 \leq m \leq 5$ distinct individuals $r_1, \ldots, r_m$ from the population that are different from i, according to i's scheme.

(iii) Generate a trial vector based on the appropriate formula for individual i's scheme. If dynamic F is being used, select $F \sim U[0,1]$ for use in the formula for each dimension of each individual.

(iv) Incorporate coordinates of the trial vector into the vector i from the original population with probability CR, using at least one coordinate.

(v) Evaluate the trial candidate.

(vi) Use the trial candidate in the new generation if it is at least as good as the current individual i. Otherwise copy i into the new population.

If using entropy, step (vi) adds $\vec{\mathcal{N}}(0, \sigma)$ to the individual (trial candidate or current individual) that is being copied into the new generation.

4.2 The Algorithm

The DynDE algorithm is the following:

1. Initialize the populations, assigning each individual a DE scheme, possibly at random.
2. Evaluate the populations.
3. Compare the distances between the best elements of each population. If any two are within r_{excl} of each other, mark the population with the worse solution for re-initialization.
4. For each population perform either step 5 or 6.
5. If the population was marked for re-initialization, perform it.
6. Generate a new population where each candidate individual i is generated in parallel according to its type: differential evolution, entropy differential evolution, Quantum or Brownian.

7. Loop to 2 unless the termination criterion is met.

5 Experiments

5.1 The Moving Peaks Benchmark

Branke [3] defined a set of benchmark dynamic optimization problems with real world characteristics that can be used to test the performance of evolutionary algorithms. Each problem consists of a set of peaks, of varying heights and widths that periodically move in a random direction by a fixed amount s (the change severity). The movement is correlated by a given $0 \leq \lambda \leq 1$ where 0 means uncorrelated and 1 correlated. The peaks change position every given number of iterations, called a time span.

In order to measure performance of an algorithm on these problems, 'offline error' is used. Offline error is defined as the average of the errors of the best points evaluated during each time span. Offline error is the measure that will be used to evaluate performance in this paper.

The suite consists of 3 functions, known as scenarios 1, 2 and 3. This paper uses scenario 2 exclusively for experimentation; the detailed parameter settings for this scenario are shown in Table 1. Software implementations of the benchmark have been made available [3].

Parameter	Scenario 2
Num. Peaks	10
Dimensions	5
Peak heights	[30,70]
Peak widths	[1.0,12.0]
Change cycle	5000
Change severity	1
Height severity	7.0
Width severity	1.0
Correlation coefficient λ	[0.0, 1.0]

Table 1: Dynamic Moving Peaks Function

5.2 Experiments and Results

As the scenario used has 10 peaks, a dimensionality of 5 and the dynamic range of 100, the exclusion radius was set to $r_{excl} = 31.5$ according to equation 1.

The radius used for the Quantum individuals was set to $r_{cloud} = 1$. This setting used information about the change severity s of the scenario. All experiments used exclusion. The experiments were conducted by allowing the algorithm to run for 500,000 function evaluations.

5.2.1 The Effect of F, CR and Scheme

A first experiment was conducted in order to see the effect of different settings of F, CR and scheme on the DynDE algorithm. Each different experimental condition involved selecting a DE scheme, and fixed values for F and CR.

The range for the choice of DE scheme and the fixed values used for F and CR are given below.

- scheme $\in$ {DE/rand/1, DE/rand/2, DE/best/1, DE/best/2, DE/rand-to-best/1, DE/current-to-rand/1}
- $F \in \{k/10 : k \in \mathbb{N} \wedge k \leq 10\}$
- $CR \in \{k/10 : k \in \mathbb{N} \wedge k \leq 10\}$

Each DynDE run was performed using 10 populations of 5 DE individuals and 5 quantum individuals (we will use the notation 5+5 from now on). Due to the large amount of time and computational power required to perform the experiments, each experimental condition was repeated only 50 times. The results of the 10 best parameter configurations are presented in table 2.

F	CR	Scheme	Error
0.4	0.6	Best/2	1.78 ± 0.100
0.3	0.5	Best/2	1.83 ± 0.130
0.3	0.3	Best/2	1.86 ± 0.127
0.3	0.4	Best/2	1.87 ± 0.144
0.2	0.2	Best/2	1.89 ± 0.124
0.4	0.5	Best/1	1.93 ± 0.125
0.3	0.7	Best/2	1.97 ± 0.163
0.5	0.5	Best/1	2.04 ± 0.166
0.4	0.4	Best/1	2.05 ± 0.153
0.4	0.4	Best/2	2.08 ± 0.146

Table 2: Top 10 results of experiments with fixed parameters for configuration 5+5. The error is the confidence interval of 50 trials.

Figure 1 helps with visualizing the effect of varying F and CR on offline performance when using the DE/best/2 scheme and a 5+5 configuration.

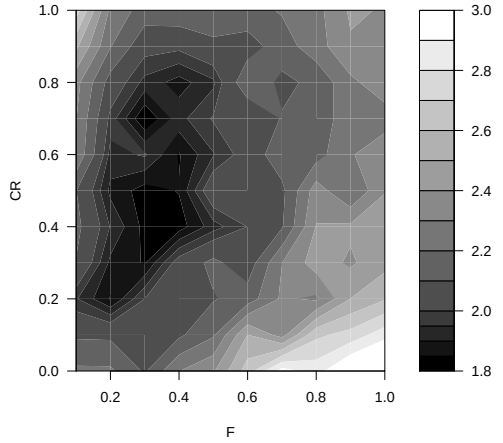

Figure 1: Effect of varying F and CR on offline performance (DE/best/2 scheme, 5+5 configuration).

5.2.2 The Effect of Population Make-up

In order to understand the implication of population sizes and the ratio of the DE and quantum individuals, a second

experiment, under the same conditions but with 10 populations of 4 DE individuals and 2 quantum individuals (denoted configuration 4+2) was performed. These are shown in table 3.

F	CR	Scheme	Error
0.5	0.6	Best/2	1.69 ± 0.149
0.4	0.5	Best/2	1.74 ± 0.112
0.3	0.3	Best/2	1.76 ± 0.124
0.6	0.4	Best/1	1.78 ± 0.152
0.3	0.4	Best/2	1.82 ± 0.117
0.9	0.6	RandToBest/1	1.84 ± 0.158
0.4	0.4	Best/2	1.86 ± 0.148
0.6	0.5	RandToBest/1	1.88 ± 0.146
0.6	0.5	Best/1	1.88 ± 0.162
0.7	0.3	RandToBest/1	1.99 ± 0.189

Table 3: Top 10 results of experiments with fixed parameters for configuration 4+2. The error is the confidence interval of 50 trials.

The results of 4+2 are much better than the ones presented with the 5+5 configuration. A Wilcoxon test performed on the data from table 2 and 3 corroborates this with a p-value of $p = 2.966 \cdot 10^{-5}$. The fact that each subpopulation has fewer individuals is one likely explanation for the better results. As the peaks move every 5,000 function evaluations, the smaller population is able to run more iterations with the same number of function evaluations. It seems that the bigger number of iterations (71 versus 45) helps the populations improve the quality of their positions. Another possible explanation could be that a larger number of quantum individuals introduces needless entropy into the system. We will return to this point on section 5.4.

Figure 2 helps with visualizing the effect of varying F and CR on offline performance when using the DE/best/2 scheme and a 4+2 configuration.

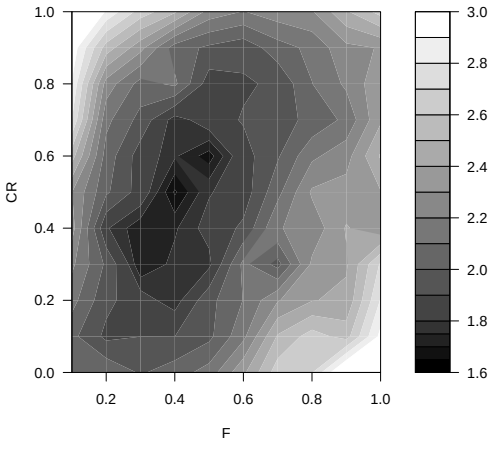

Figure 2: Effect of varying F and CR on offline performance (DE/best/2 scheme, 4+2 configuration).

5.2.3 The Vicissitudes of Experimentation

Neophyte researchers tend to only supply averages of their runs when reporting the result of empirical experimentation. However, a statistician will frown at those comparisons. Even when showing confidence intervals, one should observe extreme care when comparing the results of experimental data. To try to better validate our results, we decided to perform 1,000 trials with the best 5 entries of tables 2 and 3. These results are presented in tables 4 and 5.

F	CR	Scheme	Error Error
0.3	0.4	Best/2	1.87 ± 0.0303
0.4	0.6	Best/2	1.87 ± 0.0303
0.3	0.5	Best/2	1.87 ± 0.0313
0.3	0.3	Best/2	1.89 ± 0.0288
0.2	0.2	Best/2	1.95 ± 0.0316

Table 4: Results of the top 5 results presented in table 2 with 1,000 trials.

F	CR	Scheme	Error
0.4	0.5	Best/2	1.73 ± 0.0287
0.5	0.6	Best/2	1.76 ± 0.0294
0.3	0.4	Best/2	1.79 ± 0.0301
0.3	0.3	Best/2	1.81 ± 0.0320
0.6	0.4	Best/1	1.82 ± 0.0313

Table 5: Results of the top 5 results presented in table 3 with 1,000 trials.

As can be seen, the estimation of the average we obtained with 50 trials was always lower than the one with 1,000. Another obvious conclusion is that the ordering of the results is not the same. Once more, this indicates the fragility of the conclusions drawn from experimental data. This is due to the fact all the confidence intervals shown in tables 2 and 3 overlap to some extent. Thus, it is not surprising to see that the ordering changed.

There is a simple rule to see if the difference of two configurations is statistically significant: if the confidence intervals don't overlap.

We can only conclude that the results presented in these tables are among the best ones that may be obtained with a fixed F, CR and scheme are used. We have made no further attempts to achieve a more accurate ordering of the results. In light of the results gathered in this section, all subsequent results presented were based on 1,000 trials.

5.2.4 Using Random F, CR and Scheme

The experiments shown in the previous section were cumbersome: they needed a large amount of CPU time to arrive at good results. Fortunately, we were able to identify a region of the parameter space with good results. Even after presenting these results, we are not sure this setting will achieve a similar performance on other problems. Thus, it seems that it would be necessary to repeat this cumbersome task again.

In an attempt to suppress the need of parameter fine tuning, and inspired by the fact we are trying to solve dynamic optimization problems, we decided to experiment with setting F and CR to random values taken from the uniform distribution. Instead of using a fixed value, each time a value of either F or CR is needed a value is sampled from $U[0, 1]$.

We also decided to experiment with allowing each individual to choose its scheme. Two variations of this idea were considered. The first, called *RandomInit* assigns a random scheme to each individual at initialization time; the individual henceforth keeps that random scheme and uses it whenever it is being updated. In the second variation, called *RandomScheme*, each time an individual is being updated, a random scheme is selected. In both instances, the random schemes are selected from the set {rand/1, rand/2, best/1, best/2, rand-to-best/1}.

These results are presented in table 6 for the 5+5 configuration and those of the 4+2 population are shown in table 7. In these tables the 'error' column gives the confidence interval of the offline error taken from 1,000 trials.

Scheme	Error
Best/2	1.91 ± 0.0289
Best/1	1.95 ± 0.0339
RandToBest/1	2.00 ± 0.0319
RandomInit	2.09 ± 0.0336
RandomScheme	2.10 ± 0.0360
CurrentToBest/1	2.15 ± 0.0332
Rand/1	2.19 ± 0.0338
Rand/2	2.27 ± 0.0358
CurrentToRand/1	2.51 ± 0.0396

Table 6: Results of the experiments with random F and CR for configuration 5+5 with 1000 trials

In the 5+5 configuration, the best results are from DE/best/2 and DE/best/1. None of the random selection of schemes works as well as these results. This was corroborated by a Wilcoxon test with a p-value $p < 2.2 \cdot 10^{-16} \ll 0.05$[1]. However, it should be kept in mind that the advantage of the RandomInit and RandomScheme variations is that they eliminate the need for fine-tuning the parameters of the algorithm.

In the 4+2 configuration, we arrived at the same conclusions. Once again, a Wilcoxon test yielded a p-value $p \ll 0.05$. In this case, best/2 is clearly superior to the following two configurations. This was confirmed by a Wilcoxon test between best/2 and the set {best/1, rand-to-best/1} with a p-value $p = 3.646 \cdot 10^{-10}$.

One last question remains: did the quality of the solutions decrease with the use of random F and CR? We compared the first line of tables 2 and 6 using a Wilcoxon test and found them to be statistically different $p = 0.01087$. The same comparison was performed on tables 3 and 7 but this time the difference was not significant $p = 0.3307$. Even thought there was a statistical difference in the 5+5 configuration, the slight decrease of quality of the solution is a good compromise when compared to the titanic task of fine-tuning the parameters.

[1] In the following tests, whenever the p-value given is lower than this amount, we will use the convention of writing $p \ll 0.05$.

Scheme	Error
Best/2	1.76 ± 0.0307
RandToBest/1	1.85 ± 0.0295
Best/1	1.87 ± 0.0331
RandomInit	1.93 ± 0.0335
RandomScheme	1.96 ± 0.0333
CurrentToBest/1	1.97 ± 0.0330
Rand/1	2.08 ± 0.0331
Rand/2	2.11 ± 0.0355
CurrentToRand/1	2.31 ± 0.0376

Table 7: Results of the experiments with random F and CR for configuration 4+2 with 1000 trials.

5.3 Different Methods of Increasing Diversity

In this section we decided to compare the different ways of increasing diversity in the population. In the previous sections, the method used to introduce diversity was the Quantum individuals. This section explores the use of the other two methods presented in section 3.2.

Both of these methods have one parameter: the standard deviation σ. As the change severity $s = 1$, it makes sense to use values so that roughly 90% of the values are generated in the interval with a range of 1. If $\sigma = 0.3$, 90% of the values would fall in the interval $[-0.49, 0.49]$.

We decided to test this theory. Based on our previous experiments, we selected the following parameters:

Scheme In this case, we chose three setups:

- **s1** best/2;
- **s2** Random scheme from rand/1, rand/2, best/1, best/2, rand-to-best/1, current-to-rand/1 or current-to-best/1;
- **s3** Random scheme from best/1 or best/2.

σ One value from $\{0.01, 0.05, 0.1, 0.2, 0.3\}$

Diversity We tested two methods:

1. DE with regular and Brownian individuals
2. Entropic Differential Evolution

Population Size We tested two sizes: 6 and 10. The DE with regular and Brownian individuals used configurations 4+2 and 5+5 where the first number represents the number of regular individuals and the second represents the number of Brownian individuals. The Entropic DE simply used 6 and 10 entropic individuals.

The results are presented in table 8 for populations of size 10 and table 9 for populations of size 6. In these tables, error is the confidence interval of the offline error taken from 1,000 trials.

Scheme	Diversity	σ	Error
s1	brownian	0.2	1.94 ± 0.029
s3	brownian	0.2	1.95 ± 0.032
s3	brownian	0.3	1.95 ± 0.030
s1	brownian	0.3	2.00 ± 0.030
s1	brownian	0.1	2.10 ± 0.032
s2	brownian	0.2	2.11 ± 0.033
s2	brownian	0.3	2.12 ± 0.031
s3	brownian	0.1	2.19 ± 0.033
s3	entropy	0.1	2.24 ± 0.034
s2	brownian	0.1	2.31 ± 0.038

Table 8: Top 10 results of varying σ with the Brownian and entropy differential evolution for populations of size 10.

The results seem to corroborate the hunch of using $\sigma = 0.3$. In fact, they indicate that 0.2 works as well, if not better. To verify this hunch, we conducted a Kruskal-Wallis test with the null hypothesis that the means of the offline error factored by σ are equal. The result of the test was a p-value $p \ll 0.05$ that disproves the null hypothesis. Thus, we can conclude that σ explains a difference in offline error. We performed a similar test with the results of table 9 and also obtained a p-value $p \ll 0.05$.

We decided to investigate the hypothesis that a σ of 0.2 or 0.3 yields different results from the other values. This time we used a Wilcoxon test as there were only two populations, one for $\sigma \in \{0.2, 0.3\}$ and the other for the other values of σ. The result of the tests for both tables 8 and 9 gave a p-value of $p \ll 0.05$ that disproves the null hypothesis that the mean of the two populations was equal. Thus, we conclude that the offline error for $\sigma \in \{0.2, 0.3\}$ is different from other values.

We decided to test our hunch that entropic DE does not work as well as using Brownian individuals. A Wilcoxon test again yielded a p-value $p \ll 0.05$ for tables 8 and 9. Thus we can conclude that using Brownian individuals is indeed superior to using entropic differential evolution.

Scheme	Diversity	σ	Error
s3	brownian	0.3	1.73 ± 0.029
s1	brownian	0.3	1.74 ± 0.029
s1	brownian	0.2	1.75 ± 0.032
s3	brownian	0.2	1.78 ± 0.033
s1	brownian	0.1	1.87 ± 0.033
s2	brownian	0.2	1.88 ± 0.033
s2	brownian	0.3	1.90 ± 0.033
s3	entropy	0.05	1.94 ± 0.029
s3	entropy	0.1	1.95 ± 0.031
s3	brownian	0.1	2.02 ± 0.035

Table 9: Top 10 results of varying σ with the Brownian and entropy differential evolution for populations of size 6.

The entropic DE did not work as well as using Brownian individuals. And the best results were with much lower σ values: around 0.1. Unlike Brownian individuals, where only a fraction of the population used that behavior, in entropy schemes the entire population used entropy. The reason for much lower σ is easily explained: as the entire population is using a perturbation, it's amplitude needs to be smaller.

It seems to make sense that using only a fraction of the population with this scheme would work well. However, given the fact that Brownian individuals work so well, it doesn't make sense to try a more complicated procedure to achieve the same result.

The results also indicate there is no significant difference between the Brownian individuals and the Quantum ones. To verify it, we conducted Wilcoxon tests between the corresponding configurations using Brownian individuals with $\sigma = 0.2$ and using Quantum individuals for scheme DE/best/2. The p-values were $p = 0.7143$ and $p = 0.07237$ for the 4+2 and 5+5 configurations respectively. Given the fact that the Brownian individuals are simpler, Occam's razor suggests using them instead of Quantum individuals.

We decided to test if the schemes could explain a difference in the mean of offline error. Again, Kruskal-Wallis tests for both tables yielded p-values $p \ll 0.05$. We also tested the hypothesis that scheme *s1* was different from *s3* when using Brownian individuals and $\sigma \in \{0.2, 0.3\}$. Wilcoxon tests returned $p = 0.04274$ for table 8 and $p = 0.8238$ for table 9. Thus, there is a valid statistical difference in table 8 but we could not prove one for table 9.

We conclude that the random scheme *s2* does not seem to produce as good results as using DE/best/2 (*s1*). *s3* provides results that are almost as good as *s1*. However, given Occam's razor, we recommend DE/best/2.

5.4 Population Setup

We decided to study the relationship between normal individuals and Brownian ones. Thus we setup an experiment with sub-populations of 10 individuals where we tested all the possible combinations of normal and Brownian individuals. We also used schemes *s1*, *s2* and *s3*. The value of $\sigma = 0.2$ was used. Table 10 shows the results of 100 trials.

Scheme	Normal	Brownian	Error
s3	6	4	1.88 ± 0.088
s3	7	3	1.89 ± 0.089
s3	5	5	1.93 ± 0.090
s3	8	2	1.94 ± 0.080
s1	7	3	1.94 ± 0.087
s1	6	4	1.95 ± 0.098
s1	5	5	1.98 ± 0.107
s1	4	6	2.07 ± 0.113
s3	4	6	2.09 ± 0.104
s2	5	5	2.11 ± 0.117

Table 10: Top 10 results of varying the number of normal and brownian individuals.

The results suggest that configurations 6+4 and 7+3 seem better both for schemes *s1* and *s3*. A Kruskal-Wallis test indicates that the configuration does matter ($p \ll 0.05$) and even that there is a statistically significant difference be-

tween configurations 5+5, 6+4, 7+3 and 8+2 ($p = 0.04556$). Again, we find no statistically significant difference between schemes *s1* and *s3* ($p = 0.07823$). We hypothesize that roughly 40% of the individuals in the population should be Brownian but that using more is detrimental.

5.5 Comparison with Other Approaches

To validate our work, we compared it to the approach of [1] and [2]. The results presented in [2] report an offline error of 4.01. Our results are clearly superior to those. As our approach is similar to the one in [1], we implemented it and were able to perform tests in the same conditions. Table 11 compares the results of multi-swarms to DynDE and shows the results with 1,000 trials. In this table, DynDE used normal and Brownian individuals with $\sigma = 0.2$ and scheme DE/best/2.

Scheme	Error
DynDE 5+5	1.94 ± 0.029
Swarm 5+5	1.93 ± 0.032
Swarm 4+2	1.73 ± 0.029
DynDE 4+2	1.75 ± 0.032

Table 11: Comparison between Multi-swarm and DynDE. Both algorithms were run 1,000 times.

We conducted a Wilcoxon test to try to establish if there was a statistically significant difference between Multi-Swarm and DynDE. The test yielded $p = 0.1198$ for the 5+5 configuration and $p = 0.4531$ for the 4+2 configuration. In both cases, we were unable to establish any statistically significant difference between both approaches. The approach we are using has the advantage of using Brownian Individuals, that are simpler than the Quantum ones.

6 Conclusions

This paper presented DynDE: a differential evolution algorithm for solving dynamic optimization problems. It started by showing the cumbersome task of fine-tuning the parameters and then introduced random F and CR. The use of random values for these parameters produced equal quality solutions and was thus deemed far superior to the use of fixed values as it does not need any fine-tuning.

The use of random schemes did not improve the results. In fact, unless the schemes used were selected from the greedier ones, the results seemed somewhat inferior. However, in this case, the results seemed a bit more robust. More research is needed in this area.

Several ways of increasing the diversity during the run were compared: Quantum individuals, Brownian individuals and entropic differential evolution. From these, entropic differential evolution was considered inferior. As the guess of the σ parameter was proved correct, the Brownian method seems to be easily configurable to other problems. We recommend it because it is less computationally demanding than Quantum individuals.

Our experiments with the ratio of normal and Brownian individuals seem to indicate a ratio of 60% normal individuals to 40% Brownian. The comparison to the approaches of other researchers allows us to conclude that this method presents good results and that more research is needed in this area to further improve it.

Bibliography

[1] Tim Blackwell and Jürgen Branke. Multi-swarm optimization in dynamic environments. In *Proceedings of Applications of Evolutionary Computing*, volume 3005 of *Lecture Notes in Computer Science*, pages 489–500, Coimbra, Portugal, 2004.

[2] Jürgen Branke. *Evolutionary Optimization in Dynamic Environments*, volume 3 of *Genetic Algorithms and Evolutionary Computation*. Kluwer Academic Publishers, 2001.

[3] Jürgen. Branke. The moving peaks benchmark. `http://www.aifb.uni-karlsruhe.de/~jbr/MovPeaks/`, 2005.

[4] Rainer Storn. On the usage of differential evolution for function optimization. In *1996 Biennial Conference of the North American Fuzzy Information Processing Society (NAFIPS 1996)*, pages 519–523. IEEE, 1996.

[5] Rainer Storn and Kenneth Price. Minimizing the real functions of the icec'96 contest by differential evolution. In *IEEE International Conference on Evolutionary Computation*, pages 842–844. IEEE, May 1996.

[6] Rainer Storn and Kenneth Price. Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11:341–359, 1997.

[7] Rainer Storn and Kennth Price. Differential evolution - a simple and efficient adaptive scheme for global optimization over continuous space. Technical report, The International Computer Science Institute, 1995.

[8] Tomislav Šmuc. Improving convergence properties of the differential evolution algorithm. In *Proceedings of MENDEL 2002, 8th International Mendel Conference on Soft Computing*, pages 80–86, 2002.

[9] Tomislav Šmuc. Sensitivity of differential evolution algorithm to value of control parameters. In *Proceedings of the International Conference on Artificial Intelligence*, pages 1087–1093, 2002.