An Ancestor based Extension to Differential Evolution (AncDE) for Single-Objective Computationally Expensive Numerical Optimization

Rushikesh Sawant, Donagh Hatton, Diarmuid P. O'Donoghue
Department of Computer Science,
Maynooth University,
Co. Kildare, Ireland.
diarmuid.odonoghue@nuim.ie

***Abstract*—This paper presents the Ancestral Differential Evolution (AncDE) algorithm, which extends the standard Differential Evolution (DE) algorithm by adding an archive of recently discarded ancestors. AncDE adds the ability to occasionally compute difference vectors between current and archived solutions, using these *inter-generational difference vectors* in place of traditional difference vectors. Results for AncDE are presented for the *CEC2015 Bound Constrained Single-Objective Computationally Expensive Numerical Optimization Problems* using *AncDE/best/1/bin*. Summary results are included for standard DE for comparison purposes and these show that AncDE generally outperforms standard DE. These results suggest that the inter-generational difference vectors can help overcome some local optima, leading to faster convergence towards the global optimum. AncDE involves the very small overhead of storing and updating the ancestral cache. This paper introduces two empirically determined stochastic rates; one for updating the ancestral cache and the other for using an ancestral difference vector in place of the normal difference vector.**

Keywords—differential evolution; ancestror archive; inter-generation difference vector

I. INTRODUCTION

Evolutionary algorithms apply the general Darwinian strategy of "Survival of the Fittest" to the task of generating high-quality solutions for challenging and sometimes ill-defined problems. Among the attractive features of evolutionary algorithms are their ability to generate solutions across a very wide range of problem types and their ability to generate solutions when alternative approaches fail. [1]

But an often overlooked aspect of evolutionary computing concerns the role played by the laws of Mendelian inheritance. In this paper we focus on Mendel's Second law - "The Law of Independent Assortment". This asserts that separate genes for separate traits are passed independently - and that this genetic information is reliably transferred from the parents directly to the offspring. It is this second part of the "Law of Independent Assortment" that is challenged by our particular extension to the (otherwise) standard of Differential Evolution (DE) [*1*] algorithm. That is, we add an extra-Mendelian inheritance pathway that stochastically allows genetic information from a cache of recent ancestors to influence the current population. We can think of this as an infrequent "second chance" for some genetic information, helping broaden a search space that perhaps became stuck in a local optimum.

Evolutionary algorithms that maintain not just the current population, but that also maintain a population of recent solutions have been called *archive algorithms* [*2*]. However, very few of these archive algorithms have been developed. One notable exception has been the Non-Dominated Sorting Genetic Algorithm II (NSGA-II) [*3*]. NSGA-II combines the parent and offspring populations together forming a larger combined population from

The authors would like to thank the Irish Research Council for Science, Engineering and Technology (IRCSET), the John Pat Hume scholarship and the Erasmus Mundus DESEM program for part funding this project and R. Storn and K Price for the DE code, extended to create AncDE.

[1] **Pre-Print Version of:**
An Ancestor based Extension to Differential Evolution (AncDE) for Single-Objective Computationally Expensive Numerical Optimization, IEEE Congress on Evolutionary Computation (***IEEE CEC***), Sawant R, Hatton D, O'Donoghue D.P, pp 3228-3234, Sendia, Japan, May 2015. DOI: 10.1109/CEC.2015.7257293.

which solutions are selected – and from which the subsequent generation is generated. In contrast to NSGA-II, the cache of ancestors introduced in this paper allows significantly older ancestors to conference the current population.

A few other algorithms have explored the use of old ancestral archives. References [*4*] [*5*] [*6*] explored the use of old ancestor archives to support a template-driven genetic repair process, used to generate solutions to combinatorial optimization problems. This work explored ancestors that were up to three generations older than the parent population – up to great-great-grandparents of a newly generated population of solutions.

The use of a significantly older archive of solutions has also been explored [*7*] [*8*], using a stochastic process to update the ancestor archive - this work again focused on combinatorial optimization problems. Some of the surprising outcomes of this work were that best results were often produced using archives that were tens of generations older than the current population. While evolutionary algorithms can be slow to converge for large combinatorial optimization problems, producing the best results using such ancient archives was considered very surprising.

This paper describes the AncDE algorithm that incorporates some of the lessons learned from [*7*] [*8*] and attempts to apply these to the (otherwise standard) Differential Evolution (DE) algorithm [*1*]. The main modifications made to the DE algorithm concern firstly, the introduction and maintenance of an archive of recent solutions. The second modification allowed the (occasional) calculation of difference vectors between a solution from the population and the ancestral cache. It should be pointed out that much of the operation of AncDE is identical to DE – what we have introduced is an additional (supplemental) extension to the algorithm that is used to calculate a modified difference vector. The implementation of AncDE described in this paper was built on a standard DE implementation – introducing as few modifications to the algorithm as possible.

II. DIFFERENTIAL EVOLUTION (DE)

We first describe the DE algorithm itself, before we describe the modifications required for the ancestral extension required for the AncDE algorithm.

A. Differential Evolution

Differential evolution (DE) is an evolutionary based optimization algorithm that generates high-quality solutions to numeric optimization problems. DE begins by creating a population of random solutions (called agents) to the given problem. After this initialization step DE applies a series of update steps to help improve the quality of these solution agents, as measured by the objective function being optimized. In effect DE then moves these agents around the problem space, in such a way that their new positions result in an improvement in the objective function.

For simplicity this paper will focus on the *best/1/bin* variant of DE, noting that this has been found to be a particularly effective version of the algorithm [*9*]. The algorithm begins by creating a population of random solutions (called agents) to the given problem. An agent is selected from the population, called the base. Then two other distinct vectors are selected from the population and a difference vector is calculated between them, with the result being called the *difference vector*. This difference vector is then multiplied by a scalar mutation factor F (parameter), allowing some tuning of the impact of these difference vectors upon the population. This difference vector represents the difference between two solution agents located within the problem space. The main novelty associated with AncDE is that the trial vector is stochastically generated from differential mutation and binomial crossover using the same one base vector from the ancestral cache – as shall be discussed in more detail below.

The difference vector is next added to the base vector and the result is called the *donor* vector. The "*best/1*" allows for variation in the selection of the base vector and variation in the number of difference vectors used by variants of the DE algorithm.

The next step takes the existing agents and combines them with the donor vector to create a set of trial vectors for possible inclusion in the population. Binomial *"/bin"* crossover selects elements from the parents according to a probability *CR*, with elements being selected from one or other parent according to this *CR* value.

The final step of DE is *selection* that determines whether a new trial vector will enter the population. Selection occurs by replacing a target vector with the corresponding trial vector if the trial vector has a lower objective-function value (for minimization problems). DE typically terminates when the algorithm converges to a solution, when a maximum number of function evaluations have been performed or when some known solution has been reached.

We note that each of the different variations in the DE algorithm can, in principle, also make use of our ancestral archive of solutions. However, his paper focuses on extending and adapting the *best/1/bin* variant of the DE algorithm.

III. Re-Visting Ancient Solutions

While ancestral solutions are not normally retained by evolutionary algorithms, we make a brief case for the potential benefits that ancestor might introduce. EA are a form of stochastic beam search, where the breadth of this beam is determined by the population size. However, for problems involving very large members of local optima, the breadth of this beam may be lower than required for reliable convergence to the global optimum. In some of these situations it might be considered beneficial to augment the search space using a cache of recent ancestors. We point out that while recent ancestors might introduce some much needed diversity, they do so without resorting to the potentially damaging impact of "true" randomness.

When viewed from the perspective of the search space, it may appear less surprising that a cache of ancestor may have a beneficial impact on the final solution. We note that some other search procedures retain and re-use past solutions or return to previous search state. The process of backtracking used in depth-first search returns the search process to a previous state, while *tabu* search that maintains a finite list of recent states that cannot be revisited [*10*]. Backtracking and tabu search can also been seen as techniques to avoid local minima.

This paper adds to the evolutionary algorithm a small population-sized cache of recent ancestors. This allows calculation of difference vectors that are derived from a longer temporal base-line – between a current vector and one that was generated and superseded in previous generations. That is, we suggest that in situations where AncDE provides improved results when compared to DE, that some of this improvement may originate from the fact that AncDE uses some ancestors that might overcome the current (perhaps local) optimum in the population and help AncDE to reach the global optimum. Careful tuning of the age of these ancestors might even allow the AncDE population to skip past some of the local optima and converge faster upon the global optimum.

We briefly look at some other work in non-standard and non-Mendelian inheritance theories. Recent advances in biology have been exploring inheritance mechanisms beyond those covered by Mendel's inheritance laws. Among those attracting most attention have been the Horizontal Gene Transfer (HGT) strategies, also called lateral gene transfer (LGT) associated with simple life forms such as viruses and the prokaryotes (such as bacteria). Several specific mechanisms for HGT have been proposed that transfer genetic information by means other than sexual reproduction. A HGT mechanism called *conjugation*, has previously been used to investigate how the cost and benefit affect evolutionary outcomes [*11*]. Other evolutionary schemes like Lamarckian evolution and the Baldwin effect have long been explored [*12*] [*13*].

A more controversial proposal has been for the vertical, extra-Mendelian transfer of genetic information [*14*] [*15*]. This proposal states that genetic information appears to have been detected in the offspring – even though it has not been detectable (by traditional means) in either of the parents. However, there does appear to be a link in that the genetic information in the offspring has been detected in a grand-parent of that offspring. While this inheritance theory has been subject to some criticisms information [*14*] is also the top rated paper by the prestigious Faculty 1000. This paper was also the inspiration for AncDE.

IV. Ancestral Differential Evolution (AncDE)

As stated earlier, much of AncDE is identical to DE. The only modification involves the difference vector and the information that is used to form that vector. The major architectural change is the introduction of a second "ghost" population of recently "deceased" ancestors. This adjunct population is referred to as the *ancestral cache*.

The ancestral cache is a repository of genetic information that is normally discarded from the current population when new trial vectors enter the population. This cache contains exact copies of agents that have been replaced in the original population. The contents of this ancestral cache do not undergo any form of modification or learning, but are stochastically replaced by newer solutions from the main population.

As with DE the base vector is always selected from the current population. As stated previously, the main difference between DE and AncDE concerns calculation of the difference vectors. The difference is still calculated between two vectors (for *AncDE/best/1/bin*) – one vector being source from the current population. AncDE makes a stochastic choice for the origin of the second target vector that is used to calculate the difference vector, as controlled by the *aup* parameter discussed below. We refer to the result produced by this new strategy as an *ancestral difference vector* – to distinguish it from the normal DE difference vectors generated between two agents from the current population. Thus, AncDE extends DE by allowing this additional type of difference vector to be calculated.

In the variant of AncDE presented in this paper, one of the two vectors used to calculate the difference vector is sourced from the current population – and additionally, this vector also used as the base vector.

$$donor\ vector = base\ vector + F^* (ancestral\ vector - base\ vector)$$

We point out that AncDE retains the ability to calculate the normal difference vector (described in section II above), but adds the possibility of calculating this new difference vector. Figure 1 outlines the most significant changes to the DE algorithm, with the most significant new features highlighted by the dotted lines. AncDE adds two significant new parameters into the differential evolution algorithm.

A. *Additional Parameters for AncDE*

- *arp ancestor replacement probability* This parameter controls the relative age of the ancestor archive. A value of 1.0 will keep this archive as a duplicate of the current population. Lower values of *arp* (eg 10^{-2}) will update the archive very infrequently, maintaining a large ancestral (and temporal) distance between the current population and the contents of the ancestor archive. We generally expect that lower *arp* values will result in greater distance between the ancestor cache and the current population.

In our most recently work, we have been experimenting with different ancestor replacement strategies. The results presented in this paper were produced with a strategy that only checks the ancestor replacement rate when some vector in the population is being replaced. (An alternate strategy we have explored is to replace the ancestor stochastically even when the current vector is *not* being replaced). The strategy presented in this paper tends to result in older ancestors, especially during late convergence.

- *aup ancestor usage probability* This controls the frequency with which an ancestral difference vector is used in place of the traditional DE difference vector – between agents from the current population. Setting *aup = 0* removes these ancestral difference vectors from consideration, while higher rates increase the impact that archived agents have on the current population.

These are the two parameters introduced by AncDE and suitable values for them will be discussed in sub-section C below.

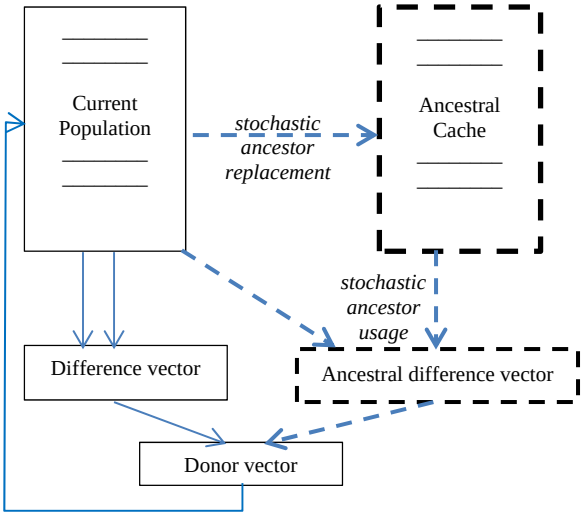

Fig. 1. Ancestral DE (AncDE) modifies DE by adding a cache of recent ancestors and the ability to derive difference vector between an agent from the population and one from the cache. The main modifications are highlithed by the dotted lines above. New ancestrally-based features have obvious parallels in (non-ancestral) normal DE.

B. *Initializing the Population*

No special efforts were made to initialize the population for the CEC 2015 problems. The standard initialization procedure for DE was adopted. Thus, the population was initialized using random values generated uniformly from within the valid range.

The ancestral population was initialized by taking copy of the standard initial population. We don't give any special treatment to the ancestral population at this point. And this is the only point where there is a direct resemblance between current population and ancestral population. Typically the ancestor population is changed when new values enter the main population, moving the older less-it values into the ancestral cache.

C. *Setting Parameters*

Standard DE also offers a number of other parameters that are also used by AncDE. Many of these parameters were assigned values after performing preliminary *ad hoc* testing.

Firstly, we look at the parameters that originate with the standard DE algorithm. These parameters were used to generate the results for the 10D and 30D problems presented in Table I and Table II below. For the 10D problems NP was set to 12, F=0.6, CR=0.75, Range =75.

For the 30D problems CR the crossover probability was set to 0.6, NP the population size was set to 25. The differential weight F was set to 0.6. One parameter was introduced for the CEC 2015 competition itself, limiting

the range of the input variable to -+75 and was used for all input parameters. These parameters we used to produce all results presented in this paper.

The two remaining parameters relate specifically to the ancestor cache and its use. Firstly the ancestor replacement probability (*arp*) that controls the relative age of the ancestor cache was set to 0.15. In this paper this factor is significantly moderated by the fact that this stochastic parameter was only checked when a solution from the main population is begin superseded. (Note: we are currently refining the best strategy to use for ancestor replacement). Thus, during early evolution when solutions are regularly updated, the ancestors will also be updated frequently. However, during late convergence the ancestor will tend to become even more ancient due to the infrequent replacement of member of the population.

Secondly the ancestor usage probability (*aup*) controls the probability that a standard difference vector will not be used, but that the ancestral difference vector strategy will be used to generate the trial vector instead. The results presented in Table 1 were generated using *aup* = 0.3, indicating that approximately 30% of difference vectors are derived using an ancient ancestor.

The best results produced for DE were generated with the following parameters (the parameter set described above produced weaker results). CR=0.95, NP the population size was set to 55, the differential weight F was set to 0.55 and range =75.

D. Algorithmic Variants of DE and AncDE

A number of standard algorithmic variants of standard DE can also be applied to AncDE. However for the purposes of this paper, only a small subset of these possibilities has been explored. All results presented herein were generated only using the following variants of *DE/best/1/bin* and *DE/rand/1/bin.*

Firstly, two selection mechanisms were investigated, as indicated by the “Best” part of Best/1/bin. Best selection ensures that the best agent in the current population. An alternate selection strategy was also investigated. Random (rand) selection chooses agents with uniform probability from within the current population. The results presented in Table 1 used only the “best” selection strategy.

The number of difference vectors that are used during the mutation step provides one of the normal variations in the standard DE algorithm. DE often uses use either 1 or 2 difference vectors to drive this mutation process. We note that a difference vector is derived from the difference between 2 agents selected from the current population. The implementation of AncDE presented in this paper uses only 1 difference vector for mutation. AncDE either uses the normal difference vector, or it uses a difference vector that takes one agent from the current population and one agent (only) from the ancestral population. It is interesting to note that we conducted a control experiment that selected both agents of a difference vector from the ancestral population, but this did not result in any measured improvement in performance or solution quality. All results presented in this paper used just 1 difference vector that involved either; a) both agents being selected from the current (main) population or b) one agent from the current population and one agent from the ancestral cache.

Finally, we look at the recombination step that generates the new solution as a combination of the target vector and the mutant vector. Binomial (*bin*) crossover is used for all results presented in this paper. Because AncDE only changes the evaluation of the difference vector, the process of crossover remains unaffected – except for the influence that the ancestral difference vector may have on the donor vector and the trial vectors.

V. RESULTS

Numerical experiments were performed on a computer with Intel® Core™ i7-3520M CPU @ 2.90GHz × 4 and 16 GB RAM, under Ubuntu 14.04 LTS, 64-bit OS. The implementation of AncDE was done in Java, using JDK 1.7. Note that the results presented in this paper were generated using Java 1.7 and its native random number generator. Because of changes to the PRNG (pseudo-random number generator) algorithm in Java 1.8, the results generated using AncDE on newer versions of Java will be different (and generally not as good). Hence, the implementation of PRNG from Java 1.7 is explicitly included in AncDE.

Tables I and II details the results produced by the AncDE algorithm on the 15 problems of the IEEE CEC expensive Numeric Optimisation problem set. These were all produced using the same settings and parameters for all 15 problem sets, as discussed in Section IV C above. These tables also include a brief overview of results for the standard DE algorithm, detailing the mean and median results for comparison purposes. Other tests not shown in this paper suggest that the *AncDE/best/1/bin* variant produced better results than *AncDE/Rand/1/bin*. However, this conclusion was reached after preliminary *ad hoc* testing.

Numbers displayed in boldface in Tables I and II indicate a victory for one of the two strategies: DE and AncDE. AncDE outperformed DE on 13 of the 15 10D problems as measured by the mean results of each. This is echoed by the median results, where AncDE outperforms DE on 12 of the 15 problems. Thus we conclude that AncDE performs better overall than standard DE on these problems.

TABLE I. RESULTS FOR 10D PROBLEMS

Standard DE			AncDE				
Func	**Median**	**Mean**	**Best**	**Worst**	**Median**	**Mean**	**St.Dev.**
1	3.29E+08	4.33E+08	2.16E+06	8.87E+07	**8.84E+06**	**1.78E+07**	2.25E+07
2	**1.93E+04**	**2.82E+04**	1.41E+04	5.91E+04	3.78E+04	3.53E+04	1.49E+04
3	7.77E+00	7.64E+00	2.48E+00	1.06E+01	**5.84E+00**	**5.73E+00**	2.11E+00
4	1.95E+03	1.99E+03	1.03E+03	2.12E+03	**1.66E+03**	**1.63E+03**	3.21E+02
5	**2.58E+00**	2.75E+00	1.50E+00	4.05E+00	2.66E+00	**2.60E+00**	6.18E-01
6	1.02E+00	9.93E-01	2.85E-01	8.20E-01	**5.55E-01**	**5.50E-01**	1.43E-01
7	1.81E+00	2.27E+00	2.94E-01	1.29E+00	**5.34E-01**	**6.35E-01**	2.91E-01
8	1.23E+01	5.25E+01	4.46E+00	9.02E+00	**6.32E+00**	**6.26E+00**	1.12E+00
9	4.08E+00	4.05E+00	3.64E+00	4.35E+00	**3.93E+00**	**3.97E+00**	2.02E-01
10	**8.02E+04**	**1.94E+05**	3.10E+04	6.99E+05	1.98E+05	2.62E+05	2.09E+05
11	6.78E+00	7.76E+00	3.58E+00	1.05E+01	**6.39E+00**	**6.65E+00**	2.00E+00
12	2.80E+02	2.79E+02	9.10E+01	4.44E+02	**2.18E+02**	**2.24E+02**	9.90E+01
13	3.29E+02	3.32E+02	3.13E+02	3.39E+02	**3.24E+02**	**3.25E+02**	6.91E+00
14	2.06E+02	2.07E+02	1.96E+02	2.15E+02	**2.03E+02**	**2.04E+02**	5.33E+00
15	4.09E+02	3.68E+02	9.93E+00	5.03E+02	**4.06E+02**	**3.59E+02**	1.50E+02

TABLE II. RESULTS FOR 30D PROBLEMS

Standard DE			AncDE				
Fnc.	**Median**	**Mean**	**Best**	**Worst**	**Median**	**Mean**	**St.Dev.**
1	1.11E+10	1.24E+10	1.12E+08	1.49E+09	**4.45E+08**	**5.21E+08**	3.54E+08
2	**6.25E+04**	**6.37E+04**	7.64E+04	1.76E+05	1.15E+05	1.18E+05	2.49E+04
3	2.74E+01	2.80E+01	2.05E+01	3.96E+01	**2.59E+01**	**2.67E+01**	4.48E+00
4	7.68E+03	7.77E+03	5.90E+03	7.62E+03	**6.81E+03**	**6.78E+03**	4.94E+02
5	4.27E+00	4.30E+00	3.18E+00	5.34E+00	**4.27E+00**	**4.24E+00**	5.91E-01
6	2.78E+00	2.76E+00	3.30E-01	9.31E-01	**5.58E-01**	**5.89E-01**	1.39E-01
7	2.21E+01	2.15E+01	3.31E-01	1.15E+00	**5.43E-01**	**5.77E-01**	2.08E-01
8	3.32E+04	1.07E+05	3.93E+01	2.16E+03	**2.43E+02**	**4.48E+02**	5.51E+02
9	**1.38E+01**	1.38E+01	1.33E+01	1.41E+01	1.38E+01	**1.38E+01**	2.03E-01
10	**1.44E+06**	**2.45E+06**	6.97E+06	3.83E+07	1.83E+07	1.81E+07	7.95E+06
11	3.84E+01	5.37E+01	2.79E+01	1.86E+02	**3.42E+01**	**4.24E+01**	3.44E+01
12	**1.23E+03**	**1.14E+03**	1.09E+03	1.62E+03	1.33E+03	1.37E+03	1.51E+02
13	4.51E+02	4.69E+02	3.53E+02	3.87E+02	**3.66E+02**	**3.68E+02**	9.11E+00
14	**2.45E+02**	**2.51E+02**	2.42E+02	3.20E+02	2.66E+02	2.71E+02	2.24E+01
15	1.05E+03	1.04E+03	7.80E+02	1.19E+03	**9.53E+02**	**9.71E+02**	1.23E+02

AncDE also produces better results on the more challenging 30D problems. AncDE outperforms DE on 11 of the 15 30D problems as measured by the mean results of each. This is echoed by the median results, where AncDE outperforms DE on 10 of the 15 problems. We again highlight that this performance improvement was achieved without additional computation, but merely involved re-visiting some recently discarded solutions form the main population.

A. AncDE Run times

The run times for AncDE should not differ significantly from those of DE itself. The first overhead associated with AncDE is storage of the ancestral population, which could in theory become exceptionally large. However, in our work we have limited the size of this ancestral population to be the same size as the main population. This small ancestral population contains a sample of the ancestors of the solutions contained in the current population. Thus, the additional memory footprint of AncDE is quite small.

The most notable overhead associated with performance of AncDE is the act of occasionally storing an ancestor, by copying it from the current population into the ancestor cache. Because updating the cache is stochastically controlled, this can be a very infrequent operation.

Another overhead of AncDE concerns the two stochastic parameters, controlling the update and use of the ancestral vectors. Calculating and checking these stochastic values is an overhead for AncDE. Computing the inter-generational difference vector itself should not involve an additional overhead, compared with the standard intra-generation difference vectors.

The recorded runtimes for the 10D and 30D problems are listed in Table III, with the times measured in milliseconds. The runtime for the CEC 2015 test function T0 (involving a *for* loop containing a simple function evaluation) on our test machine was 71.

TABLE III. RUN TMES FOR THE ANCDE ALGORITHM ON THE 10D AND 30D PROBLEMS

	10D Problems		**30D Problems**	
Func	**Runtime**	**T1/T0**	**Runtime**	**T1/T0**
1	48	0.676056338	149.5	2.105633803
2	48	0.676056338	141.5	1.992957746
3	105	0.676056338	581	1.992957746
4	48.5	0.683098592	153	2.154929577
5	68	0.957746479	239.5	3.373239437
6	48	0.676056338	146.5	2.063380282
7	48.5	0.683098592	140.5	1.978873239
8	50.5	0.711267606	155.5	2.190140845
9	48.5	0.683098592	145.5	2.049295775
10	51	0.718309859	163	2.295774648
11	74	1.042253521	252	3.549295775
12	64	0.901408451	200	2.816901408
13	105.5	1.485915493	272	3.830985915
14	88	1.23943662	245.5	3.457746479
15	166	2.338028169	718.5	10.11971831

VI. FUTURE WORK

The results presented in this paper suggest that any performance advantage conferred by AncDE is more noticeable on more complex problems. One investigation that arises from this is to compare DE and AncDE on larger problems.

One extension to AncDE is a newer variant currently being investigated (called *2AndDE*) that uses two related caches containing two chronologically related ancestors for each solution in the current population. These caches contain an older ancestor

and a newer ancestor. The presence of two ancestors may allow even better performance to be achieved, where the ancestors serve to scale the difference vector sourced from within the current population.

VII. Conclusion

This paper presented an ancestral extension to the DE algorithm (called AncDE) that added an archive of solutions discarded from the main population, stochastically updated from the main population. AncDE added to DE the ability to calculate difference vectors between a current and an ancient solution. The resulting *inter-generational difference* vectors are controlled stochastically in terms of both their relative age and the frequency that they impact on the current population.

Results were presented for the "CEC 2015 Special Session & Competition on Real-Parameter Single Objective Optimization". Summary results are included for standard DE and these show that AncDE outperforms standard DE on most of these problems. Thus, the performance of DE has (surprisingly) been improved using a cache of solutions that were discarded from the main population. We see this ancestral information as an extension to Mendel's Law of Inheritance, acting as an additional inheritance pathway. As an explanation for these results we suggest that AncDE's occasional use of the difference vectors originating from a long-temporal baseline – leads to difference vectors of larger magnitude and occasionally large jumps across the solutions space that avoid some local minima.

Acknowledgment

The authors would like to thank the Irish Research Council for Science, Engineering and Technology (IRCSET) for part funding this project. We would also like to thank the John and Pat Hume scholarship for part funding this project. The authors would also like to thank the Erasmus Mundus DESEM program for part funding this project. We would also like to thank R. Storn and K. Price for the DE code, which was extended to create the Ancestral DE (AncDE) implementation.

References

[1] R. Storn and K. Price, "Differential Evolution - a Simple and Efficient Heuristic for Global Optimization over Continuous Spaces," *Journal of Global Optimization*, vol. 11, pp. 341-359, 1996.

[2] S. Luke, *Essentials of Metaheuristics*, 2nd ed.: Lulu, 2013.

[3] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182-197, 2002.

[4] A. FitzGerald and D.P. O'Donoghue, "Genetic Repair for Optimization under Constraints inspired by Arabidopsis thaliana," in *Parallel Problem Solving from Nature (PPSN)*, 2008, pp. 399-408.

[5] A. FitzGerald and O'Donoghue D.P., "Biologically Inspired Non-Mendelian Repair for Constraint Handling in Evolutionary Algorithms," in *The Genetic and Evolutionary Computation Conference (GECCO) - Constraint Handling Workshop*, Portland, Oregon, 2010, pp. 7-11.

[6] A. FitzGerald, D. P. O'Donoghue, and X. Liu, "Genetic Repair Strategies inspired by Arabidopsis thaliana," in *Lecture Notes in Artificial Intelligence (LNAI 6206)*., 2010, pp. pp 61-71.

[7] D. Hatton and D.P. O'Donoghue, "Explorations on Template-Directed Genetic Repair Using Ancient Ancestors and Other Templates," in *The Genetic and Evolutionary Computation Conference (GECCO) - Constraint Handling Workshop*, 2011.

[8] D. Hatton and D.P. O'Donoghue, "Arabidopsis thaliana Inspired Genetic Restoration Strategies," *International Journal of Biometrics and Bioinformatics*, vol. 7, no. 1, pp. 35-48, 2013.

[9] E. Mezura-Montes, J. Velazquez-Reyes, and C. A. Coello Coello, "A comparative study of differential evolution variants for global," in *Proc. Genet. Evol. Comput. Conf.*, 2006, pp. 485–492.

[10] S. Russell and P. Norvig, *AI: A Modern Approach.*: Prentice Hall, 2009.

[11] B. D. Connelly, L. Zaman, P.K. McKinley, and C. Ofria, "Modeling the Evolutionary Dynamics of Plasmids in Spatial Populations," in *Proceedings ACM GECCO Conference*, Dublin, Irealnd., 2011.

[12] Darrell Whitley, V. Scott Gordon, and Keith Mathias, "Lamarckian Evolution, The Baldwin Effect and Function Optimization," in *PPSN*, Jerusalem, Israel, 1994, pp. 6--15.

[13] A.E. Qin, Ke Tang, Hong Pan, and Siyu Xia, "Self-adaptive Differential Evolution with Local Search Chains for Real-Parameter Single-Objective Optimization," in *IEEE Congress on Evolutionary Computation (CEC)*, Beijing , China, 2014, pp. 467 - 474.

[14] S.J. Lolle, J.L. Victor, J.M. Young, and R.E. Pruitt, "Genome-wide non-mendelian inheritance of extra-genomic information in Arabidopsis," *Nature*, vol. 434, no. 7032, pp. 505-509, March 2005.

[15] M.T. Hopkins et al., "De novo genetic variation revealed in somatic sectors of single Arabidopsis plants (v2)," *F1000 Research*, vol. 2, no. 5, July 2013.