Proceedings of the 5th WSEAS/IASME Int. Conf. on SYSTEMS THEORY and SCIENTIFIC COMPUTATION, Malta, September 15-17, 2005 (pp330-335)

A Parameterless Differential Evolution Optimizer

JASON TEO and MOHD. YUNUS HAMID
Artificial Intelligence Research Group
School of Engineering and Information Technology
Universiti Malaysia Sabah
Kota Kinabalu, Sabah
MALAYSIA

Abstract: - Although the Differential Evolution (DE) algorithm has been shown to be a simple yet powerful evolutionary algorithm for optimizing continuous functions, users are still faced with the problem of preliminary testing and hand-tuning of the evolutionary parameters prior to commencing the actual optimization process. As a solution, self-adaptation has been found to be highly beneficial in automatically and dynamically adjusting evolutionary parameters such as crossover rates and mutation rates. In this paper, we present a first attempt at self-adapting the population size parameter in addition to self-adapting crossover and mutation rates. Firstly, our main objective is to demonstrate the feasibility of self-adapting the population size parameter in DE. Using the F1-F5 benchmark test problems proposed by De Jong, we showed that DE with self-adaptive populations produced highly competitive results compared to a conventional DE algorithm with static populations. In addition to reducing the number of parameters used in DE, the proposed algorithm actually outperformed the conventional DE algorithm for one of the test problems. It was also found that that an absolute encoding methodology for self-adapting population size in DE produced results with greater optimization reliability compared to a relative encoding methodology.

Keywords: - Soft computing, Evolutionary computation**,** Differential evolution, Parameter encoding, Self-adaptation, Evolutionary dynamics, Population dynamics.

1 Introduction

One of the current state-of-the-art evolutionary algorithms is the *Differential Evolution* (DE) algorithm [10]. It is a very simple population-based stochastic optimizer for continuous domains. The crucial idea behind DE is in the generation of trial solutions where the weighted difference between two population vectors is added to a third vector. The DE algorithm has been successfully applied to a whole host of engineering problems including the design of digital filters, mechanical design optimization, aerodynamic design and multiprocessor synthesis.

There are three critical steps when using evolutionary algorithms to solve any problem: choosing appropriate (i) representations, (ii) selection processes, and (iii) parameter settings [3, 8]. This paper focuses on the third issue, that is the parameter settings required to run an evolutionary algorithm. In this regard, while the use of evolutionary algorithms for solving design and optimization problems is both widespread and diverse, there is still the common complaint voiced by end-users that there is the need to first find a good combination of parameter settings for running the evolutionary algorithms before their actual optimization work can begin in earnest [4] and the DE algorithm is no exception to this requirement.

The parameter settings in evolutionary algorithms most commonly comprise of the (i) crossover rate, (ii) mutation rate, (iii) population size, and (iv) number of generations [3]. The number of generations can usually be eliminated as a user-defined parameter by simply determining an acceptable stopping or convergence criteria and building this metric directly into the evolutionary algorithm. In terms eliminating crossover and mutation rates as user-defined parameters, significant progress has been made through adaptation and self-adaptation of crossover and mutation rates [9, 12]. In fact, it has been found to actually improve the quality of solutions resulting from the search process [3]. Here, we make the distinction between adaptation and self-adaptation using Eiben's definition [3]: (i) adaptation — where changes are made to parameters during evolution based on some feedback from the search and (ii) self-adaptation — where changes are made to parameters during evolution through inclusion of parameters into the genetic encoding that is itself subjected to evolutionary processes and pressures. In fact, it has been pointed out that the usage of static parameters that do not change over the course of the artificial evolution may not give the best results [3].

However, significantly less work has been done in terms of adapting population size so as to render it as a non-user defined parameter. It has been highlighted that an appropriate population size is critical to both run-time efficiency and optimization effectiveness [5, 7]. As such, the main objective of this research work is to implement and investigate a dynamic approach to the population sizing problem through self-adaptation, thereby eliminating population size as a user-defined parameter in working towards a parameterless evolutionary algorithm.

This remainder of this paper is organized as follows. Section 2 surveys the work that has been previously conducted in the area of adaptation of population size. Section 3 outlines our proposed methodology of a dynamic population size through self-adaptation. Section 4 presents the setup of the experiments conducted. Section 5 presents the results obtained and analysis conducted. Finally, Section 6 summarizes the main conclusions arising from this work.

2 Adapting Population Sizes

The task of selecting an appropriate population size for solving particular classes of problems has been known to be a challenging and often puzzling question in and of itself. A number of different approaches have been used in adapting the sizes of dynamic populations. Adaptation based on chromosomal age [1], probability of selection error [7], competition among subpopulations [6], generational improvement [13] and exponential population growth [4] have been used. However in all of the methods mentioned above, the approach is based on adaptation and not self-adaptation, that is the changes made to the population size parameter are based on some form of feedback from the search. Population size is not included as an element of the genetic encoding that is subjected to evolutionary processes and pressures as in self-adaptation. Since self-adaptation has been shown to be a very effective method for dynamically adjusting evolutionary parameters [9, 12], we thus propose a similar treatment for dynamically adjusting the population size. In the next section, we outline our methodology for implementing a dynamic population size for the DE algorithm based on self-adaptation using firstly an absolute encoding methodology and secondly a relative encoding methodology.

3 Proposed Algorithms

Our proposed algorithm is called **D**ifferential **E**volution with **S**elf-**A**dapting **P**opulations (DESAP). The main contribution of this proposed algorithm is that the population size is automatically determined from initialization right through the evolutionary search process to completion. Therefore, the user does not need to be concerned at all with regards to determining the population size to be used for the evolution. A self-adaptive strategy is also used for controlling the crossover and mutation rates for both our proposed DESAP as well as conventional DE algorithms where the rates are encoded into the chromose and subjected to evolution similar to the search variables. We propose two different versions of DESAP, where the first version utilizes an absolute encoding methodology for population size, which we shall refer to as DESAP-Abs, and the second version utilizes a relative encoding methodology for population size, which we shall refer to as DESAP-Rel. A real-number representation is used in the chromosome for each of the search variables as well as the self-adapted parameters.

The initial population size for DESAP-Abs and DESAP-Rel are both set at 10 times the number of design variables as recommended by the authors of the original DE algorithm [10]. For DESAP-Abs, the population size of subsequent generations is taken as the average of the population size attribute from all individuals in the current population whereas for DESAP-Rel, it is taken to be the current population size plus a percentage increase or decrease according to the population growth rate [11]. The formulation for calculating the new population size parameter is as denoted below, where π represents the evolvable population size parameter that is encoded into the chromosome:

DESAP-Abs:

$$M_{new} = round(\sum_{1}^{M} \pi / M)$$

DESAP-Rel:

$$M_{new} = round(M + (\pi * M))$$

It should be pointed out also that elitism and culling are achieved naturally in both DESAP-Abs and DESAP-Rel. Elitism is only executed when the population size of the next generation exceeds the population size of the current generation ($M_{new} > M$). In this case, all of the current population will survive into the next generation and on top of that, the best solution in the current population is copied into $M_{new} - M$ individuals to make up the required number of individuals in the next generation. In the case where the population size of the next generation is less than the population size of the current

generation ($M_{new} < M$), then only the best M_{new} individuals will survive into the next generation and the remainder of the current population will be culled. No elitism or culling will occur if the population size of the next generation equals the population size of the current population ($M_{new} = M$).

4 Experimental Setup

To investigate the effects of a dynamic self-adaptive population, both versions of our proposed DESAP algorithm are compared against a conventional DE algorithm, which is similar to DESAP in all aspects except that the population size is static and non-adaptive. Five different population sizes of 10, 20, 30, 50, and 100 were used for the conventional DE algorithm. These algorithms were compared using the widely used F1-F5 test functions proposed by De Jong [2]. We treat these benchmark problems in their original form, that is as function minimization problems. Each evolutionary setup was run for 100 generations and repeated 50 times using different seeds.

5 Results and Discussion

5.1 DESAP vs. Conventional DE

We present below the results obtained using both versions of our proposed DESAP algorithm with dynamic self-adapting population sizes against the conventional DE algorithm with static population sizes of 10, 20, 30, 50, and 100, which are denoted as DE-10P, DE-20P, DE-30P, DE-50P, and DE-100P respectively.

Algorithm	Test Function				
	F1	F2	F3	F4	F5
DESAP-Abs	8.8E-06	9.2E-05	0	2.7598	0.9980
DESAP-Rel	8.4E-06	1.1E-04	0	3.1400	0.9980
DE-10P	1.7E-05	5.4E-05	0	3.3556	0.9980
DE-20P	7.0E-05	3.6E-05	0	4.0818	0.9980
DE-30P	6.9E-05	1.5E-04	0	4.4530	0.9980
DE-50P	1.5E-05	1.7E-06	0	5.2456	0.9980
DE-100P	1.6E-06	1.6E-05	0	4.8230	0.9980

Table 1: Overall best solution found.

Table 1 lists the overall best solution found over 50 runs. Both versions of our proposed algorithm DESAP produced highly similar results as the conventional DE algorithm. In fact, the proposed DESAP algorithm obtained the best overall solution for F4. It converged to the best possible solution for F3 and F5 as did the DE algorithm with different population sizes. DESAP obtained slightly better solutions than DE for F1 except when DE was set to largest population size (DE-100P). For F2, DESAP performed marginally better than DE-30P and marginally worse against the remaining setups for DE. Comparing between DESAP-Abs and DESAP-Rel, the results were highly similar between both versions except that the absolute encoding produced marginally better results for F2 and F4 and vice-versa for F1. Thus, in terms of the overall best solution found, DESAP produced highly competitive results compared to the conventional DE algorithm.

Algorithm	Test Function				
	F1	F2	F3	F4	F5
DESAP-Abs	0.0005	0.0113	0.48	7.7899	5.2351
DESAP-Rel	0.0005	0.0156	1.14	7.0239	5.8397
DE-10P	0.0017	0.0271	3.74	21.6080	7.9802
DE-20P	0.0009	0.0129	2.36	16.7103	5.5710
DE-30P	0.0005	0.0098	1.52	14.3968	3.2273
DE-50P	0.0003	0.0034	0.54	12.9318	1.8220
DE-100P	0.0001	0.0018	0.16	10.4214	0.9980

Table 2: Average of best solutions found.

Table 2 lists the average of the best solutions found over 50 runs. The results again indicate that across the 50 different evolutionary runs, DESAP was able to perform better on average compared to DE with smaller population sizes of 10, 20, and 30 for F1, F3 and F4 and better than DE with population sizes of 10 for F2 and F5. Again, DESAP produced the most favorable average best solution for F4. DE with the larger population sizes of 50 and 100 were able to produce slightly better solutions on average compared to DESAP with the exception of F4 where DESAP outperformed all configurations of the conventional DE. Comparing between DESAP-Abs and DESAP-Rel, both versions had the same average of the best solutions for F1 but was better using the absolute encoding for F2, F3 and F5, whereas the opposite held true for F4.

Algorithm	Test Function				
	F1	F2	F3	F4	F5
DESAP-Abs	0.0004	0.0133	0.81	2.1918	4.6760
DESAP-Rel	0.0006	0.0153	1.41	2.1012	5.0260
DE-10P	0.0020	0.0325	2.52	8.6251	4.9501
DE-20P	0.0012	0.0198	2.22	7.0218	4.7626
DE-30P	0.0003	0.0114	1.59	4.9793	3.9585
DE-50P	0.0003	0.0037	0.88	5.0388	2.3722
DE-100P	0.0001	0.0018	0.42	3.0276	0.0000

Table 3: Standard deviation of best solutions found.

Table 3 lists the standard deviation of the best solutions found over 50 runs. These results indicate

that the performance of DESAP is quite stable compared to the conventional DE algorithm. The dynamic nature of the population size does not appear to adversely affect the stability of the evolutionary search process and was in fact more stable than DE when DE was set to use the smaller population sizes. For DE with the larger population sizes, the standard deviation of the best solutions of DESAP were not very different compared to DE. Comparing between the two different methodologies of self-adapting the population size parameter, the absolute encoding appeared to generate less variation in terms of the best solutions found in all the test functions except for F4. This suggests that the evolutionary search process is more stable using the absolute encoding compared to the relative encoding.

Algorithm	Test Function				
	F1	F2	F3	F4	F5
DESAP-Abs	0.0016	0.0537	3	12.7752	12.6705
DESAP-Rel	0.0035	0.0691	5	11.4348	12.6705
DE-10P	0.0093	0.1306	8	45.8672	12.6705
DE-20P	0.0075	0.1219	7	30.7957	12.6705
DE-30P	0.0017	0.0509	5	29.4706	12.6705
DE-50P	0.0014	0.0164	3	21.9949	10.7632
DE-100P	0.0007	0.0068	2	17.1941	0.9980

Table 4: Worst of the best solutions found.

Table 4 lists the worst of the best solutions found over 50 runs. Except for F5 where DE-100P was able to find the optimal solution in all runs, the worst of the best solutions generated by DESAP were highly similar to those generated by DE. Again what this shows is that the dynamic self-adapted population sizes in DESAP does not negatively affect the optimization process. Moreover, the dynamic sizing of the population was able to generate better solutions in the run which produced the least optimal results compared to static populations with small sizes.

Comparing between DESAP-Abs and DESAP-Rel, the absolute encoding again appeared to have slightly better results in terms of the worst of the best solutions found for three of the five test functions. However, the relative encoding generated a better result for F4. Both algorithms produced the same worst of the best solutions of 12.6705 for F5.

DESAP-Abs	Test Function				
	F1	F2	F3	F4	F5
No. of Variables	3	2	5	30	2
Minimum P.S.	22	11	42	297	13
Maximum P.S.	54	25	52	300	42
Average P.S.	28.92	18.72	49.08	299.16	21.52
S.D. of P.S.	4.44	2.53	1.95	0.71	6.33

Table 5: Final population sizes for absolute encoding.

DESAP-Rel	Test Function				
	F1	F2	F3	F4	F5
No. of Variables	3	2	5	30	2
Minimum P.S.	14	9	37	251	20
Maximum P.S.	210	24	120	355	26
Average P.S.	36	19.62	53.18	303.46	20.2
S.D. of P.S.	27.20	2.11	13.034	17.17	1.01

Table 6: Final population sizes for relative encoding.

Table 5 and 6 provides an analysis of the dynamic population size reached after 1000 generations for the 50 evolutionary runs of DESAP (P.S. refers to population size and S.D. refers to standard deviation). One clear observation that can be made here is that there is a fair amount of dynamics present in DESAP when attempting to solve these five problems. The population size analysis for DESAP-Abs is discussed first. There is a noticeable difference between runs that achieved a smaller population size at the end of the search process and those that achieved a higher population size. For F2-F4, the final population sizes appear to be limited within a much narrower range as compared to F1 and especially F5.

Surprisingly, the problem with the most number of design variables, F4, produced the least amount of variation in terms of final population sizes with only a standard deviation of 0.71 and a range difference of only 3 individuals between the minimum and maximum population size. As this was the problem in which DESAP-Abs obtained the best solution compared to the conventional DE algorithm for all population settings, it may be an indication that DESAP-Abs was able to self-adapt to the appropriate population size for solving this problem. This figure is very close to the recommended population size setting of ten times the number of design variables advocated by the authors of the original DE algorithm as explained earlier [10]. It should also be pointed out that although DESAP-Abs utilized a higher population size than those used by the conventional DE algorithm in the F4 experiment, in which it obtained superior results, this supports our motivation in proposing a parameterless evolutionary algorithm — an evolutionary algorithm should be able to automatically determine, adjust and self-adapt its population size appropriately to the problem at hand, which is exactly that achieved by our proposed DESAP-Abs algorithm.

Secondly for DESAP-Rel, it can be seen from the standard deviation that there is a much larger variation in terms of population size compared to DESAP-Abs

except for F5. This observation corresponds to the larger standard deviation that was also detected for the best solutions found over 50 runs (refer to Table 3). Therefore, running DESAP using the absolute population size encoding provided more stability than using the relative population size encoding. Coupled with the fact that the best solutions obtained did not differ significantly between the two types of encoding, it would appear that using the absolute population size encoding may be preferable from a stability point of view.

5.2 Population Dynamics: DESAP-Abs vs. DESAP-Rel

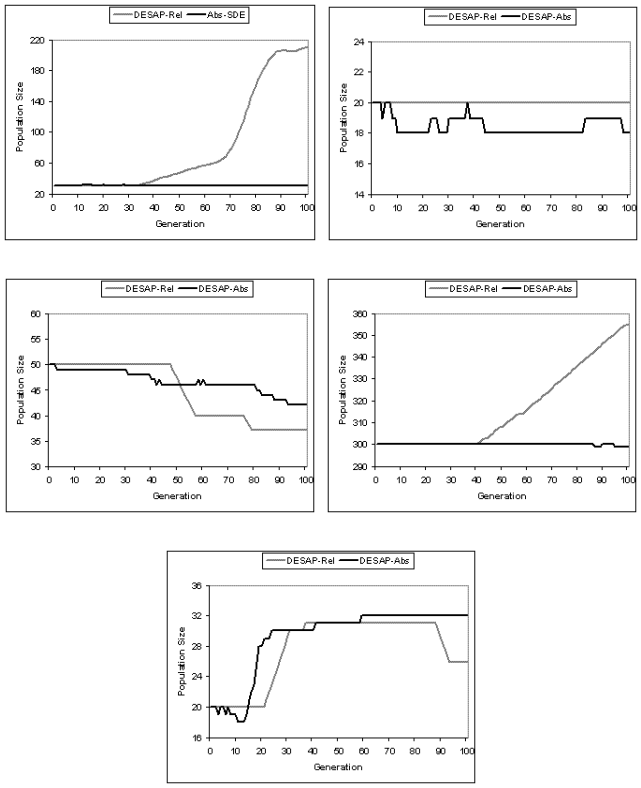

Fig. 1: DESAP-Abs vs. DESAP-Rel: Population dynamics of run generating the best solution for F1 (top left), F2 (top right), F3(middle left), F4 (middle right) and F5 (bottom).

We now compare the population dynamics of DESAP-Abs and DESAP-Rel for F1-F5 in terms of the overall best solution found. Figure 1 plots the progression of the dynamically self-adapting population size for every generation in the run that found the overall best solution for the five test problems. Dark solid lines indicate the population size of DESAP using absolute encoding while the gray solid lines indicate the population size of DESAP using relative encoding.

From these graphs which compare the population dynamics of the absolute encoding against the relative encoding, it can be seen that for some of the problems, the self-adaptation of population size appear to follow quite similar trends over time in both approaches. For F2, the relative encoding did not change its population size from initialization to termination and the absolute encoding also remained very narrowly range-bound between 18 and 20 individuals.

For F3, both versions of encoding could be seen to decrease its population size over time whereas for F5, both encodings could be seen to increase its population size very rapidly in the early stages of evolution and then stabilize somewhat after generations 40–50 with a slight decrease towards the end for the relative encoding. However in F1 and F4, the population size remained quite stable in the absolute version compared to the relative version where the population size increasingly became larger over time. Moreover, the overall fluctuations in population size over time was greater in DESAP-Rel for F1, F3, F4 and F5 compared to DESAP-Abs.

Perhaps the most interesting and revealing problem for DESAP is the F5 test function, which has many deep local minima surrounding the global minimum and is known to be a challenging problem for function optimizers to find the optimum solution. As such, it may have presented the most difficult evolutionary landscape in terms of finding the solution for DESAP. For both DESAP-Abs and DESAP-Rel, the population size which can be seen to increase from around 18 individuals to more than 30 individuals may have been caused by the fact that a much larger population size was required to solve this problem. Therefore, the DESAP-Abs self-adapted the population size to higher numbers of individuals as evolution progressed due to the advantage provided by this fact. The largest change occurred before generation 25 and 30 for DESAP-Abs and DESAP-Rel respectively. This observation again supports the fact that the algorithm is able to home in on a suitably sized population while at the same time solving the problem at hand.

Overall, the population dynamics analysis shows that the preconception of larger population sizes tending to improve the search results to be untrue, which corroborates the findings of [5]. If it were true then all the graphs would show an increasing trend in terms of the growth of the population size over time. Furthermore, as the results are highly similar to the conventional DE and clear dynamics are occurring in the population of the proposed algorithms, the elitism and culling procedures inherent in the self-adapted DE

optimizer do not appear to encourage premature convergence and may have actually benefited the search process for F4 where superior results were obtained.

6 Conclusion

We have proposed a DE algorithm with a dynamic population sizing strategy called DESAP based on self-adaptation. Two versions of DESAP were implemented using absolute and relative encoding methodologies respectively for dynamically self-adapting the population size parameter. The algorithm was tested against a conventional DE algorithm with different but fixed, static population sizes on five benchmark test problems. The results show that DESAP with dynamic self-adapting population sizes performed similarly well when compared against the conventional DE algorithm. It also outperformed the conventional DE algorithm for one of the test problems. Additionally, it was found that both types of parameter encoding yielded highly similar results in terms of the best solutions found. However, there was significantly more variation in both the best solutions found and final population sizes using the relative encoding. Since no significant advantages could be observed using the relative encoding, it is concluded that the use of an absolute encoding for the self-adaptation of population size may be more favorable by virtue of providing more stability over the course of the evolutionary search process compared to the relative encoding. A population dynamics analysis also showed that the overall trends in self-adapting the population size parameter were quite similar for some of the problems although there was significantly more variation in population size over time using the relative encoding methodology.

References:

[1] J. Arabas, Z. Michalewicz, and J. Mulawka. GAVaPS - A genetic algorithm with varying population size. In *Proceedings of the 1st IEEE Conference on Evolutionary Computation*, pages 73–78, Piscataway, NJ, 1994. IEEE Press.

[2] K.A. De Jong. *An Analysis of the Behaviour of a Class of Genetic Adaptive Systems*. PhD thesis, Univesity of Michigan, 1975.

[3] A.E. Eiben, R. Hinterding, and Z. Michalewicz. Parameter control in evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 3(2):124–141, 1999.

[4] G.R. Harik and F.G. Lobo. A parameter-less genetic algorithm. In W. Banzhaf, et al., editors, *Proceedings of the 1999 Genetic and Evolutionary Computation Conference*, volume 1, pages 258–265, Orlando, Florida, 1999. Morgan Kaufmann.

[5] R. Sarker and M.F.A. Kazi. Population size, search space and quality of solution: An experimental study. In *Proceedings of the 2003 Congress on Evolutionary Computation (CEC2003)*, volume 3, pages 2011–2018. IEEE Press, Piscataway, NJ, 2003.

[6] D. Schlierkamp-Voosen and H. Muhlenbein. Strategy adaptation by competing subpopulations. In Y. Davidor, H.-P. Schwefel, and R. Manner, editors, *Proceedings of the 3rd Conference of Parallel Problem Solving from Nature*, number 866 in Lecture Notes in Computer Science, pages 199–208. Springer-Verlag, Berlin, 1994.

[7] R. Smith. Population size. In T. Back, D. Fogel, and Z. Michalewicz, editors, *Handbook of Evolutionary Computation*, chapter E1.1, pages 1–5. Oxford University Press, New York, 1997.

[8] W. M. Spears, K. A. De Jong, T. Back, D.B. Fogel, and H. de Garis. An overview of evolutionary computation. In P.B. Brazdil, editor, *Proceedings of the 1993 European Conference on Machine Learning*, volume 667, pages 442–459, Vienna, Austria, 1993. Springer-Verlag.

[9] W.M. Spears. Adapting crossover in evolutionary algorithms. In J.R. McDonnell, R.G. Reynolds, and D.B. Fogel, editors, *Proceedings of the 4th Annual Conference on Evolutionary Programming*, pages 367–384. MIT Press, Cambridge, MA, 1995.

[10] R. Storn and K. Price. Differential evolution: A simple and efficient adaptive scheme for global optimization over continuous spaces. Technical Report TR-95-012, International Computer Science Institute, Berkeley, 1995.

[11] J. Teo. Differential evolution with self-adaptive populations. In *9th International Conference on Knowledge-Based Intelligent Information and Engineering Systems (accepted)*, Melbourne, Australia, 2005.

[12] J. Teo and H.A. Abbass. Elucidating the benefits of a self-adaptive Pareto EMO approach for evolving legged locomotion in artificial creatures. In *Proceedings of the 2003 IEEE Conference on Evolutionary Computation*, volume 2, pages 755–762, Pistacaway, NJ, 2003. IEEE Press.

[13] N. Zhuang, M. S. Benten, and P.Y. Cheung. Improved variable ordering of BDDS with novel genetic algorithm. In *Proceedomgs of the IEEE International Symposium on Circuits and Systems*, volume 3, pages 414–417, Piscataway, NJ, 1996. IEEE Press.