An Analysis of the Operation of Differential Evolution at High and Low Crossover Rates

James Montgomery Stephen Chen

***Abstract*—A key parameter affecting the operation of differential evolution (DE) is the crossover rate $Cr \in [0, 1]$. While very low values are recommended for and used with separable problems, on non-separable problems, which include most real-world problems, $Cr = 0.9$ has become the *de facto* standard, working well across a large range of problem domains. Recent work on separable and non-separable problems has shown that lower-dimensional searches can play an important role in the performance of search techniques in higher-dimensional search spaces. However, the standard value of $Cr = 0.9$ implies a very high-dimensional search, which is not effective for other search techniques. An analysis of Cr across its range $[0, 1]$ provides insight into how its value affects the performance of DE and suggests how low values may be used to improve the performance of DE. This new understanding of the operation of DE at high and low crossover rates is useful for analysing how adaptive parameters affect DE performance and leads to new suggestions for how adaptive DE techniques might be developed.**

I. Introduction

Differential evolution (DE) [1] is a population-based search technique that has been successfully applied in many problem domains [2]. A key parameter that affects its performance is its crossover rate (Cr), and a value of $Cr = 0.9$ has been found to work well across a large range of problems domains. Consequently, $Cr = 0.9$ is often used as a default value in many DE implementations [2], [3], [4].

Although it is claimed that DE has some insensitivity to the value of its control parameters [2], an increasing body of work suggests that, especially on difficult or large problems, the effective range for these parameters is sometimes small [4]. Additionally, recent work [5], [6], [7], [8] on separable and non-separable problems has shown that lower-dimensional searches can play an important role in the performance of search techniques in higher-dimensional search spaces. However, $Cr = 0.9$ implies using very high-dimensional searches, which is not effective for other search techniques. An analysis of DE applied to the problems from the CEC2008 Large Scale Global Optimisation competition, with Cr varied across its range $[0, 1]$, provides insight into how its value affects the performance of DE. Explanations are presented why low and high values of Cr can be effective when values around 0.5 are not. The above analysis suggests that many existing adaptive DE techniques will likely be unable to find and exploit both low and high values of Cr while avoiding values which adversely affect algorithm performance. Suggestions on how to incorporate the new insights on Cr's effective ranges and their effects on search behaviour are presented.

James Montgomery is with the Complex Intelligent Systems Laboratory, Faculty of Information & Communication Technologies, Swinburne University of Technology, Melbourne, Australia (phone: +61 9214 5735; email: jmontgomery@swin.edu.au).

Stephen Chen is with the School of Information Technology, York University, Toronto, ON Canada (e-mail: sychen@yorku.ca).

A. Differential Evolution

The three main variants of DE are designated DE/rand/1/*, DE/best/1/* and DE/target-to-best/1/*, which add (respectively) a difference vector generated from two members of the population to a randomly chosen individual, the best individual or to some point between the target and best individual. The most common, and frequently effective [9], variant is DE/rand/1/*, which is the subject of the remainder of this work; details of the others are discussed in general in [2] and in relation to crossover by Zaharie [4]. The * may be either *bin* for a uniform crossover, where the probability of mutating a component follows a binomial distribution, or *exp*, where a sequence of vector components is taken, the length of which follows an inverse exponential distribution. Although the same value of Cr produces a different probability that a component is mutated in the *bin* and *exp* variants, when the actual probability of mutation is the same there are negligible differences between the performance of the two [4], so only the *bin* variant is considered here. Let $S = \{x_1, x_2, \ldots, x_{|S|}\}$ be the population of solutions. In each iteration, each solution in S is considered as a *target* for replacement by a new solution; denote the current *target* by x_i. A new point v_i is generated according to

$$v_i = x_{r1} + F \cdot (x_{r2} - x_{r3}) \tag{1}$$

where x_{r1} (also referred to as the *base*), x_{r2} and x_{r3} are distinct, randomly selected solutions from $S \setminus \{x_i\}$ and F is the scaling factor, typically in $(0, 1]$ although larger values are also possible. Uniform crossover is performed on v_i, controlled by the parameter $Cr \in [0, 1]$, to produce the candidate solution u_i, according to

$$u_i^j = \begin{cases} v_i^j & \text{if} R^j \leq Cr \text{ or } j = I_i, \\ x_i^j & \text{if} R^j > Cr \text{ and } j \neq I_i, \end{cases} \tag{2}$$

where $R_j \in [0, 1)$ is a uniform random number and I_i is the randomly selected index of a component that must be mutated, which ensures that $u_i \neq x_i$. The *target* is replaced if the new solution is as good or better.

Small values of Cr result in exploratory moves parallel to a small number of axes of the search space, while large values of Cr produce moves at angles to the search space's axes. Consequently, the general consensus, supported by some empirical studies [3], [4], is that small Cr is useful when

solving separable problems while large Cr is more useful when solving non-separable problems.

The following sections examine previous findings on the effect of Cr on DE's performance in problems up to 100 dimensions (Section II) and then describe new experiments on the CEC2008 Large Scale Global Optimisation (LSGO) competition problems [10] up to 1000 dimensions (Section III). Effects on DE's search behaviour, including convergence rate, and size and quality of moves made, is then examined (Section III-A). A broad range of adaptive (and self-adaptive) DE algorithms are considered in light of these results in Section IV before recommendations concerning DE's application and automatic adaptation are given in Section V.

II. Previous Findings

Despite the important role that crossover plays in the DE algorithm, there are relatively few studies that have examined the algorithm's performance as Cr is varied. Price, Storn and Lampinen [2] present, for a number of benchmark problems, phase portraits in which "successful" combinations of F and Cr are plotted across the 2D space of the values. The portraits reveal good combinations of the two parameters for the selected problems but do not illuminate the reasons for this good performance. Mezura-Montes, Velázquez-Reyes and Coello Coello [9] performed a sensitivity analysis to pick the "best" setting for Cr for another set of problems. They do not report DE's performance across all settings tested, but a low value of $Cr = 0.1$ was often chosen as the best performing. Rönkkönen, Kukkonen and Price [3] examined the relationship between Cr and DE's performance on different problems, concluding that $Cr \leq 0.2$ was appropriate for separable problems and $Cr > 0.9$ was best for non-separable problems. Zaharie [4] presents a detailed exploration of the differences between *bin* and *exp* crossover regimes, the actual mutation probability in each for a given value of Cr and performance results for different settings on 50- and 100-dimension separable and non-separable versions of the frequently-used Rastrigin and Griewangk problems. Cr is varied across the set $\{0, 0.01, 0.03, 0.1, 0.3, 0.5, 0.7, 0.9, 0.95, 0.99\}$. Considering just the *bin* variant those results may be summarised as follows:

- on the separable problems, performance is best when Cr is near zero, degrades as it increases towards 0.5–0.7 and then improves again as Cr approaches but does not reach 1;
- on the non-rotated Griewangk instance, a similar pattern is observed, but high values of Cr are as effective as smaller values;
- on the rotated Griewangk instance, performance is best when Cr is near the middle of its range; and
- as Cr gets very close to 1, performance degrades, even when values above 0.9 performed well.

III. Algorithm Performance with Varying Cr

This section presents results for DE/rand/1/bin applied to the CEC2008 LSGO competition problems [10]: Sphere (f_1), Schwefel's Problem 2.21 (f_2), Rosenbrock (f_3), Rastrigin (f_4), Griewangk (f_5), Ackley (f_6) and "FastFractal" (f_7). Problems f_1 and $f_3 - f_6$ are shifted versions of the classic functions of the same name, while the last consists of an irregular, fractal search landscape. The number of dimensions D of each problem is 100, 500 and 1000. As in the competition, the number of function evaluations was set to $5000 \times D$ and the algorithm was run on each problem using 25 random seeds.

As the effect of Cr is being studied, the population size and value of F must be held constant. There are broadly two options: a small population (e.g., 50) with $F \geq 0.8$ or a population of D individuals with smaller F [11]. Although it has been found that $F \in [0.3, 1]$ can be effective [12], the use of a large population in conjunction with large F will cause the algorithm to converge too slowly to produce a good result by the end of its run [13]. This is due to the interaction of the population's spread and the related size of difference vectors, an interaction that is explored in a companion work [11]. In this study, the population size was set to D, with $F = 0.5$ for 100D instances other than f_4 and f_7 and $F = 0.3$ for those two 100D instances and all 500D and 1000D instances.[1] Cr was varied across $\{0, 0.1, 0.25, 0.5, 0.75, 0.9, 1\}$.

Tables I through III present the average final result for different Cr settings for 100-, 500- and 1000-dimension instances, respectively. Results are expressed as the relative percentage deviation (RPD) from the optimal or, in the case of f_7, the best reported solutions (see [14] for best in 100D and 500D, and [8] for best in 1000D). RPD is calculated according to

$$RPD = \frac{|f(x) - f(x^*)|}{f(x^*)} \cdot 100$$

where $f(x)$ is the value of the solution being evaluated and $f(x^*)$ is the value of the optimal or best known solution.

Fig. 1 presents the relative performance of each Cr setting within each problem instance. Performance measures have been scaled such that 1 represents the maximum value (i.e., worst result) within Cr from $[0, 0.9]$; many results for $Cr = 1$ are too high to be meaningfully plotted. It should be noted that the most effective DE variants employ additional techniques to improve performance; the results here are to illustrate the kind of performance likely to result from a standard DE algorithm and so are, in some cases, poor.

On the 100D and 500D f_1, f_5 and f_6 instances, most Cr settings less than 1 performed well. However, on these same functions in 1000 dimensions, a pattern emerges in which $Cr = 0$ is less effective than 0.1, performance then degrades up to 0.5 before improving towards 0.75 or 0.9, finally degrading, sometimes quite severely, at 1. A similar pattern is also evident in the results for f_3, f_4 and f_7 across

[1] The performance across all Cr values was improved for the 100D f_4 and f_7 instances when the lower F value was used.

TABLE I
PERFORMANCE (RPD) ACROSS Cr VALUES ON 100D INSTANCES

Cr	f_1	f_2	f_3	f_4	f_5	f_6	f_7
0	0	9	32	0	0	0	6
0.1	0	0	26	48	0	0	23
0.25	0	1	27	109	0	0	32
0.5	0	4	210	156	0	0	38
0.75	0	3	31	139	0	0	40
0.9	0	10	40	45	0	0	9
1	5.1×10^4	18	1.1×10^{10}	417	1053	14	38

TABLE II
PERFORMANCE (RPD) ACROSS Cr VALUES ON 500D INSTANCES

Cr	f_1	f_2	f_3	f_4	f_5	f_6	f_7
0.00	14	35	8.9×10^4	177	1	3	17
0.10	0	9	285	1027	0	0	43
0.25	0	13	2572	1277	0	0	48
0.50	1	19	5.8×10^5	1465	0	0	50
0.75	0	15	313	1437	0	0	51
0.90	0	22	1248	239	0	2	51
1.00	3.4×10^5	22	1.3×10^{11}	2240	6618	15	43

all sizes, and to some extent in those for f_2 (500D and 1000D). Performance is generally poor across all values of Cr on f_2 and f_7, problems in which global structure and exploitable gradients are less prevalent, which suggests that on the other problems DE is exploiting global structure in its search. These results accord with those of Zaharie [4].

A related metric of algorithm behaviour is the proportion of new solutions that were improving, an indication of the utility of the search mechanism used. This is highly important in DE, as the particular new solutions it generates are a function of both the control parameters F and Cr and the set of solutions in the current population, from which difference vectors are drawn. If, after some time, the search has arranged its population in such a way that subsequent moves are unlikely to be improving, then it will stagnate. (Conceivably certain configurations could also start their runs this way.) Low overall acceptance rate can indicate situations where the search has stagnated. Fig. 2 presents the proportion of new solutions that were accepted by the algorithm for the runs whose performance is given above. There is generally an inverse relationship between the quality of final solutions produced and the acceptance rate.

TABLE III
PERFORMANCE (RPD) ACROSS Cr VALUES ON 1000D INSTANCES

Cr	f_1	f_2	f_3	f_4	f_5	f_6	f_7
0.00	1.4×10^4	39	1.5×10^9	1282	316	10	24
0.10	2	19	3.7×10^5	2587	0	0	48
0.25	246	24	1.6×10^8	3031	6	3	51
0.50	7135	30	2.2×10^9	3346	163	6	53
0.75	6	21	3.5×10^5	3131	1	1	54
0.90	10	23	1.4×10^6	688	1	3	54
1.00	6×10^5	23	2.8×10^{11}	4643	1.5×10^4	15	46

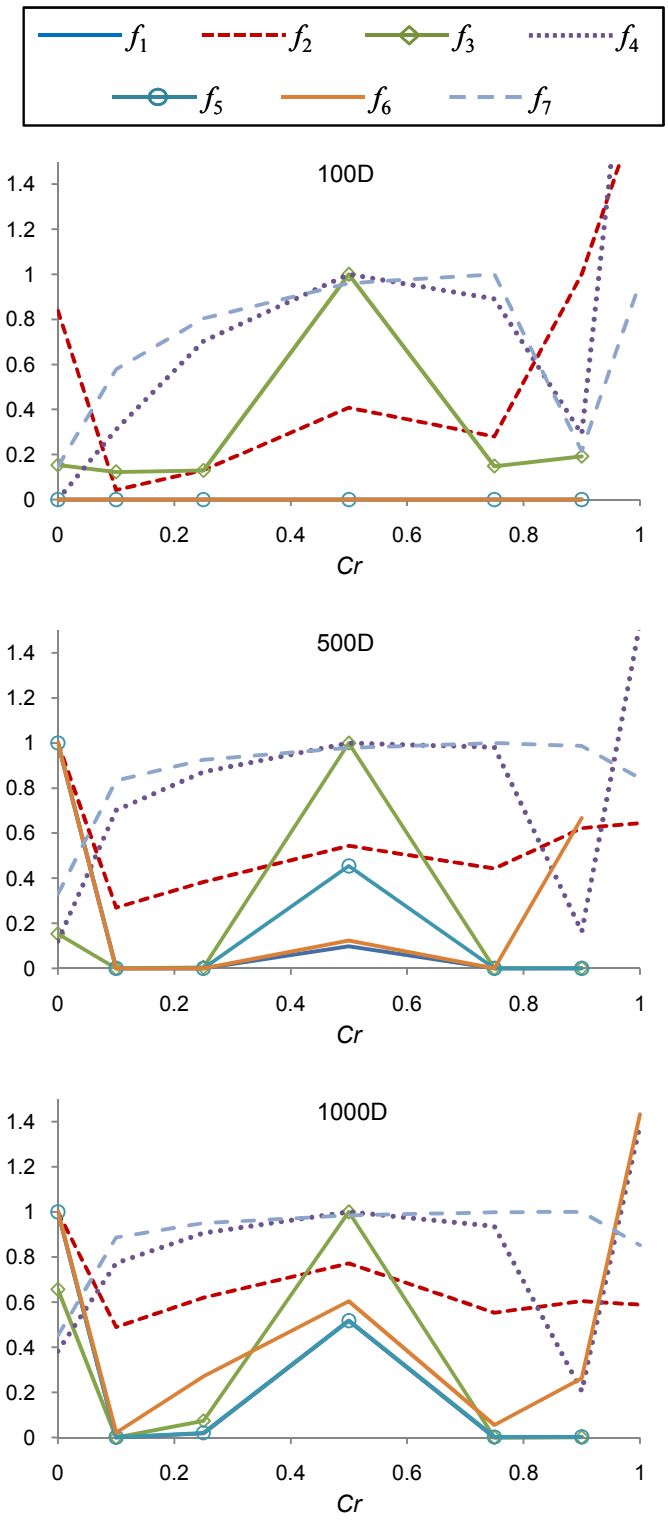

Fig. 1. Relative performance within problem instance. Results have been scaled within each problem such that the worst result for $Cr \in [0, 0.9]$ is 1. Values for $Cr = 1$ are not plotted when they exceed 2.5

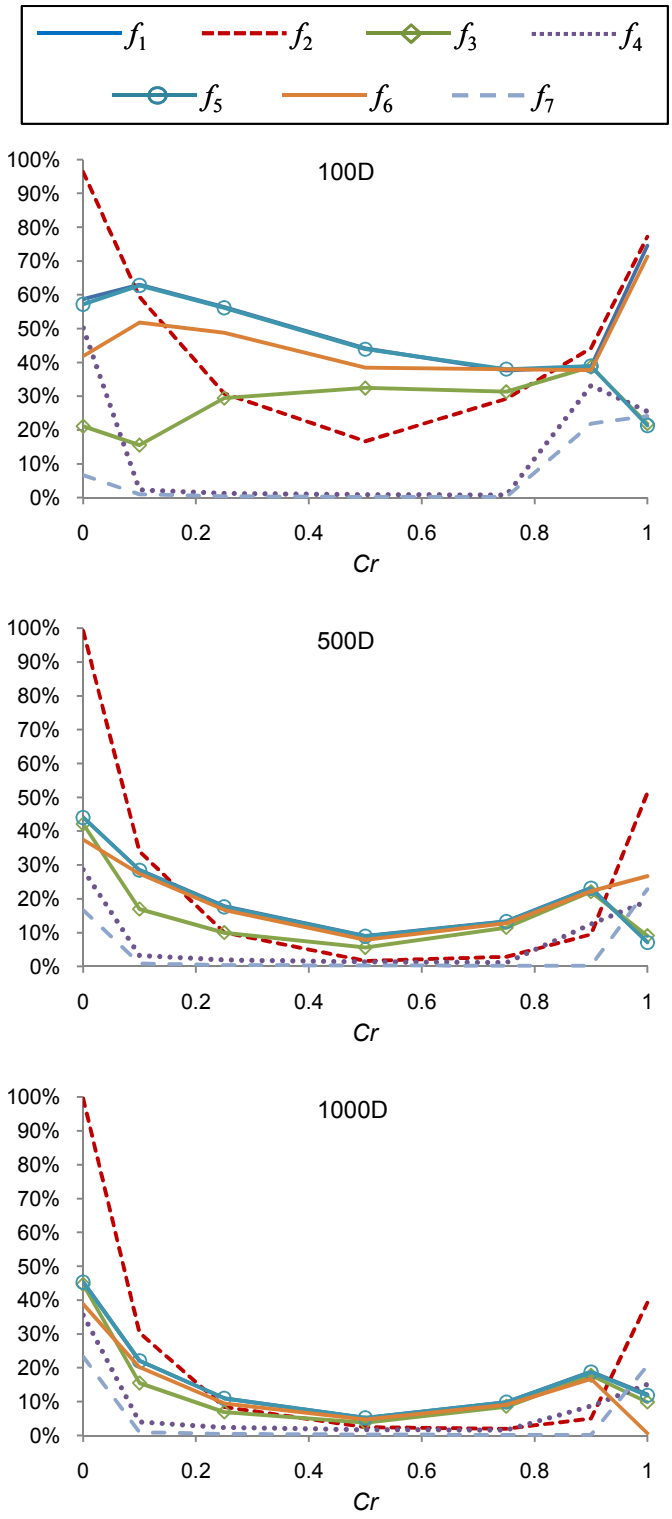

Fig. 2. Average proportion of accepted (improving) moves over entire run by problem instance and Cr value.

A. Rate of Convergence and Population Evolution

Typical explanations [2], [4] of DE's differing exploratory behaviour at the extreme values of Cr focus on the *directionality* of the search: low values search along axes and are good for separable problems, high values search at angles and are good for non-separable problems. While there are situations where this is the case, such explanations are insufficient to explain why *both* high and low Cr values can be effective on the *same* problem.

This section illustrates differences in the manner of search when Cr is set to 0, 0.5, 0.9 or 1. The measures used are the average magnitude of moves (both attempted and accepted, normalised to be a fraction of the main space diagonal), the average quality of population members, the average distance of population members from the population centroid (i.e., diversity) and the proportion of moves accepted. Two example runs are taken to illustrate the key differences: 1000D Sphere (f_1) in Fig. 3 and 100D FastFractal (f_7) in Fig. 4. The data for the average move magnitude and proportion of new solutions accepted (top left and bottom right, respectively, in Figs. 3 and 4) have been smoothed by taking a running average over 100 iterations.

On 1000D Sphere, the convergence rate is inversely related to the value of Cr, with $Cr = 1$ showing exceptionally fast convergence. Concomitantly, the mean move magnitude decreases as the run continues—this clearly demonstrates the self-scaling property of the algorithm that has long been lauded as a key benefit [2]. It is also evident that $Cr = 0$ makes very small moves during the entire run, whereas higher values of Cr lead to relatively large moves at the beginning. Related to the size of moves is the rate of change in the quality of solutions. For $Cr \in \{0.5, 0.9, 1\}$, rapid initial improvements are shown, while $Cr = 0$ shows gradual, but steady improvement over time. It is also worth noting the acceptance rate of new solutions. This is near 50% for $Cr = 0$ for the entire run, around 20% for 0.9 for much of the run and dropping to 5% for $Cr = 0.5$. The extremely low rate of improving solutions when $Cr = 0.5$ may indicate that the population has not converged sufficiently for smaller exploratory moves, which may be more successful, to be made [11]. $Cr = 1$ is a special case, with rapid convergence leading to a loss of diversity so early that it cannot make further progress. This is discussed further below.

On 100D FastFractal, for all but $Cr = 0.5$, the convergence rate is also inversely related to Cr. Again, the low value of Cr produces very small moves and a gradual, smooth decline in the value of the function. Very few moves with $Cr = 0.5$ are successful, so the search languishes, unable to make improvements. With $Cr = 0.9$ the search proceeds in a staggered fashion, with a relatively small number of successful moves producing good quality improvements in solutions; the acceptance rate rises once the population has converged, but subsequent improvements are slight. $Cr = 1$ again converges prematurely to an inferior solution.

The cause of DE's premature convergence when $Cr = 1$

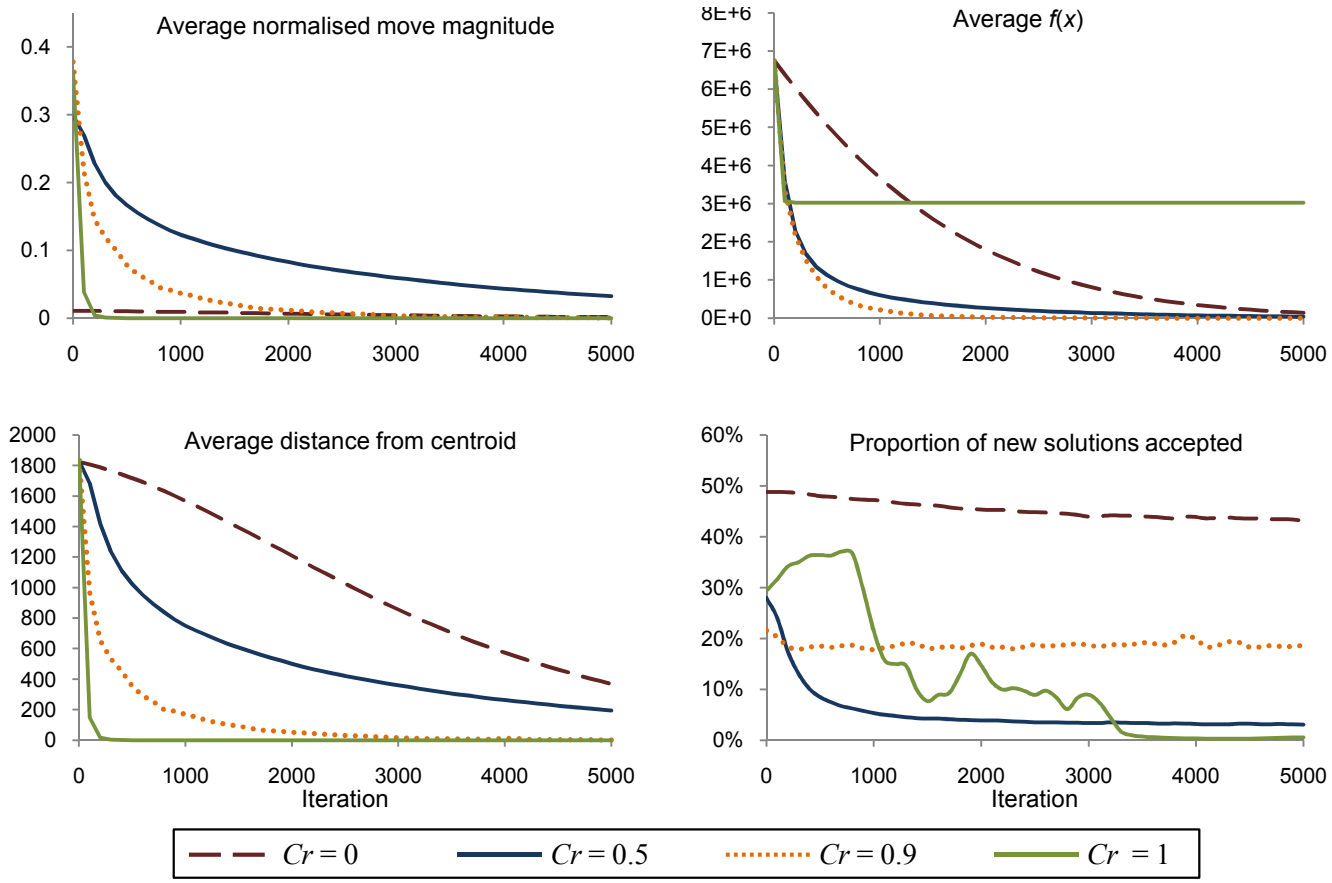

Fig. 3. From a sample run on 1000D f_1: normalised move magnitude, average $f(x)$, population spread and proportion of moves accepted, by iteration. Move magnitude and population spread are strongly related when Cr is high. A low acceptance rate after initial successes may indicate that the population has not converged sufficiently for smaller exploratory moves to be made [11]

likely lies in the nature of DE/rand/1's mutation mechanism, which generates a new point by displacing a *base* solution that is not the *target* for replacement. Previous work [13], [15] has found that relatively small difference vectors applied to *base* solutions at some distance from their respective *targets* can contribute to population convergence, as the new solutions, if accepted, are frequently nearer to another member of the population in the next iteration. In the case of $Cr = 1$, newly generated points are guaranteed to be within $F \times ||x_{r2} - x_{r3}||$ units of the *base*, while they are likely to be further away from the base if any value of Cr less than 1 is used (as the *target* keeps some of its own components).

With regards to the search behaviour when $Cr < 1$ the following general statements may be made:

- When $Cr \approx 0$ DE makes very small exploratory moves, aligned with a small number of axes. The search proceeds in a gradual but consistent fashion as the likelihood of making an improving move is higher when moves are small, even if the change in solution quality is not great [11]. It is worth noting that in results not included here, in which a population size of 50 was used, $Cr = 0$ outperformed $Cr = 0.1$. This is because, with a fixed number of function evaluations, a small population allows for a greater number of iterations over which DE with $Cr = 0$ may progress slowly but steadily.
- When $Cr \approx 0.9$ DE makes large exploratory moves that, while being less likely to be improving, can yield large improvements in solution quality. These large moves also reduce population diversity, which is a necessary step so that subsequent moves are scaled appropriately for performing a more fine-grained search of the solution space.
- When $Cr \approx 0.5$ DE behaves more similarly to when $Cr = 0.9$ than when $Cr = 0$. Large moves with large improvements also result in a reduction in population diversity, yet it appears plausible that the population is often still too spread for difference vectors to be scaled appropriately to continue the search. Thus, the exploratory moves made after initial improvements are neither gradual enough (as with $Cr \approx 0$) nor large enough (as with $Cr \approx 0.9$) to continue the search productively.

IV. Insights into Adaptive DE Algorithms

Given that selecting an appropriate value for Cr (and, indeed, F and the population size) is not necessarily straightforward, a range of approaches have been developed to allow

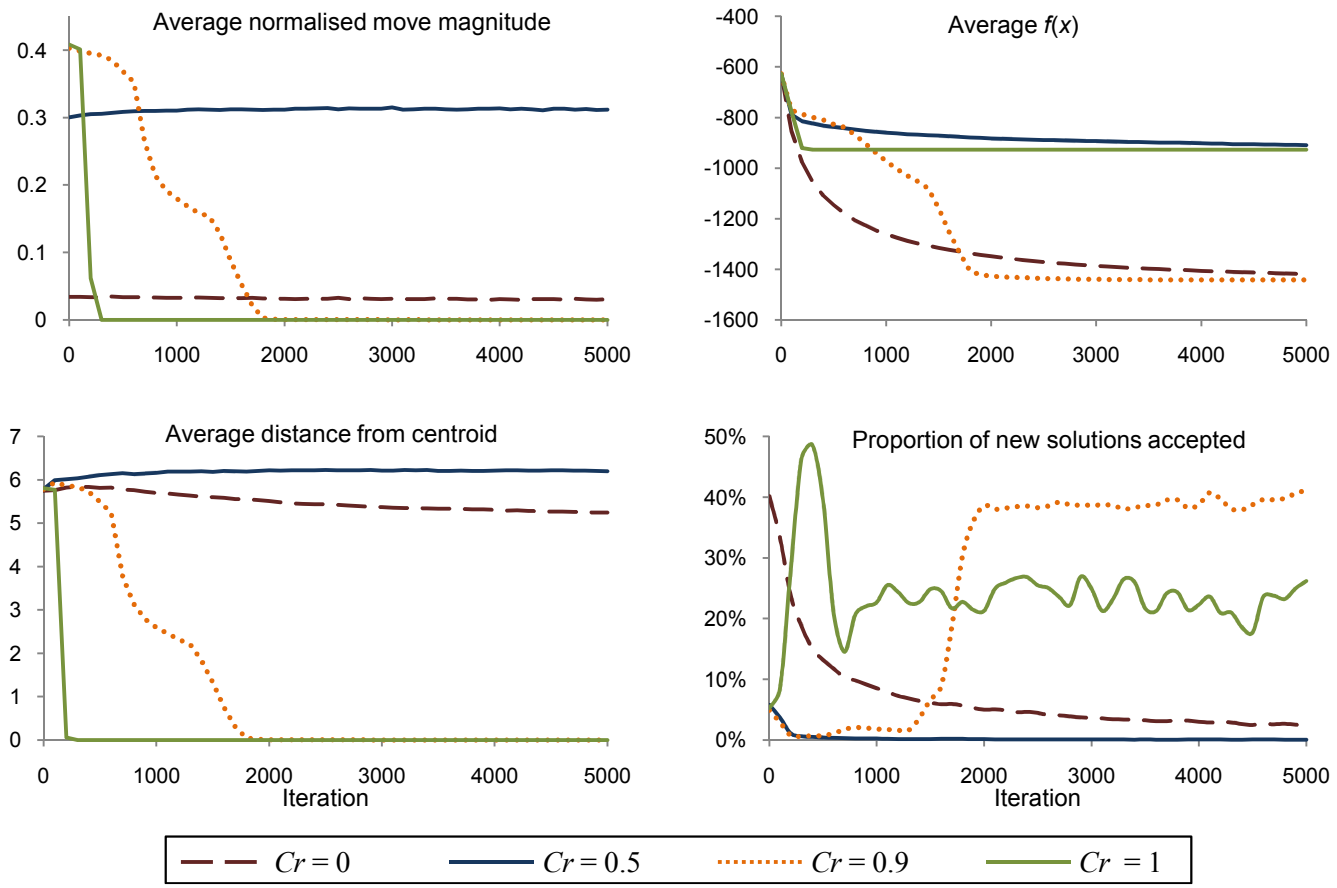

Fig. 4. From a sample run on 100D f_7: Normalised move magnitude, average $f(x)$, population spread and proportion of moves accepted, by iteration. Move magnitude and population spread are strongly related when Cr is high. A low acceptance rate after initial successes may indicate that the population has not converged sufficiently for smaller exploratory moves to be made [11]

the algorithm to select these values itself. Discussion of these adaptive and self-adaptive techniques has been left until this point so that they may be discussed in light of the results of the preceding sections.

Adaptive techniques include Fuzzy Adaptive DE (FADE) [16], which uses fuzzy logic controllers to change the values of F and Cr; while the decisions are based on fuzzy sets, the rules for changing their values are predetermined by the algorithm designer. Self-adaptive approaches are more numerous and employ a variety of techniques to adjust the values of the control parameters in response to changes in the algorithm's behaviour, or to allow the algorithm to manipulate their values in a similar manner to the way solutions are modified. Self-adaptive DE (SaDE) [17] keeps track of F and Cr values used in successful moves; at the end of each successive learning period their values are reset within a normal distribution centred on the average of these successful values. A similar approach is used by Tvrdik [18], whose algorithm selects probabilistically from a small set of $F - Cr$ combinations based on which have performed best most recently. After a number of successful moves (when the probability of selecting some of the combinations falls below a specified threshold) the probabilities are reset.

Other approaches adapt F and Cr separately for each individual. Brest *et al.*'s [19], [20] jDE periodically (on 10% of iterations) resets the values of F and Cr using a uniform random distribution. Unlike the adaptive algorithms described above, this technique does not learn the most effective values, instead exploring a wide range of values over the course of its run. Brest *et al.* [21] and Brest and Sepesy Maučec [22] also consider a deterministic population size reduction mechanism, which focuses the population on its most successful individuals (and, consequently, likely decreases move size due to accelerated population convergence).

In all of these it is unclear what values the algorithms settled on, if any. Indeed, techniques such as jDE's periodic resetting of values appear unlikely to select any particular values for a long period.

Salman, Engelbrecht and Omran [23] describe a self-adpative DE (SDE) that adapts F and Cr for individuals using the same mechanisms as used by the DE algorithm, by adding a weighted difference between two of the values in use to a randomly selected other. SDE does settle on fairly consistent values: $F \approx 0.35$ and $Cr \approx 0.5$. Zamuda *et*

al. [24]'s DE with self-adaptation and cooperative coevolution (DEwSAcc) is a derivative of jDE, but uses log-normal adaptation of F and Cr to make gradual adjustments to their values. Their final values are not given in the cited work, but correspondence with the author suggests that, on most of the CEC2008 LSGO instances the best solution's value of F drops to below 0.1 while Cr varies considerably around 0.5.[2] The algorithm's cooperative coevolution component implicitly further reduces the value of Cr, as each subpopulation explores a subset of the problem's dimensions. Although not reported in [24], the adaptation mechanism will tend to produce diminishing values over time, due to the hard upper bound imposed on F and Cr. Without a mechanism to preserve values associated with improving moves, this downward bias may be undesirable.

The results from Section III suggest that, on many problems, both low and high values of Cr can perform well, with low values frequently a robust choice as the algorithm makes gradual, small improvements to all solutions. They also suggest that, on some difficult problems, middling values of Cr prevent the algorithm from either making gradual improvements or achieving the necessary leaps in order to scale its subsequent moves appropriately for further successful exploration. Self-adaptive techniques that reset Cr within a large range are thus unlikely to spend much time in either of the two good positions.

Self-adaptive approaches that make gradual changes are potentially handicapped depending on the initial value of Cr: if the frequently-recommended value of $Cr = 0.9$ were used as an initial starting point, such techniques would be unlikely to explore the region around $Cr = 0.1$ unless no selection pressure is applied to which values are evolved and the algorithm is willing to endure a period of few successful solutions being produced. Interestingly, the two variants for which the evolved values of Cr have been reported [23] or discovered [24] appear to choose values of Cr around 0.5.[3] Both techniques appear to select small values of F, however, which may serve to attract solutions to the location of the best individuals when they are used as the *bases* for exploratory moves [13], [15].

Finally, adaptive algorithms that learn the best parameter settings by monitoring their relative success rates (e.g., [17], [18]) will likely be biased towards low values of Cr, as (small) improving moves are more likely to result with low Cr [11]. While results here and elsewhere [4], [9] show that low Cr can be effective, moves resulting from high Cr can produce large improvements in solution quality quickly, despite having a generally lower rate of success. Such implicit biases should be considered when developing a learning self-adaptive algorithm.

[2]A. Zamuda, personal communication, 22 January 2010.

[3]It is possible that individual solutions in DEwSAcc [24] actually explore a wide range of values for Cr, with its apparent preference for 0.5 an artefact of taking the average value used by the best individual across multiple runs.

V. Future Directions

Experiments with other search techniques [5], [6], [7] have demonstrated that lower-dimensional searches can improve performance on both separable and non-separable high-dimensional functions. The experiments conducted here demonstrate that lower-dimensional searches in DE (i.e., with $Cr \leq 0.1$) can lead to superior DE performance on separable functions. On the non-separable problems—Schwefel 2.21 (f_2), Rosenbrock (f_3), Griewangk (f_5), and FastFractal (f_7)—DE with $Cr \leq 0.1$ achieved results as good as or better than DE with $Cr = 0.9$, suggesting low-dimensional searches can also help improve the performance of DE on non-separable problems. Future tests are required, however, to determine how best to make use of low Cr values on such problems, since DE progress becomes slow.

DE with $Cr = 0.1$ is an almost entirely different algorithm to DE with $Cr = 0.9$. Population dynamics, move characteristics and convergence rates are all highly different. While DE with $Cr = 0.9$ relies on the population converging so that its moves may be scaled for finer-grained search, DE with $Cr \leq 0.1$ maintains a highly diverse population throughout its course, especially in complex landscapes, as individual solutions conduct largely independent searches of the solution space.

Given this new understanding of how DE generates successful moves, the development of an adaptive DE that can exploit both high and low values of Cr in a non-random fashion is a promising area for further research. Such an algorithm will not be able to rely solely on the number of successful moves produced, as high values of Cr will tend to produce fewer, but larger improvements in solution quality. Nor should it explore the (restricted) space of Cr values blindly: some selection pressure must exist so that effective values are identified and maintained. Finding an appropriate balance between these requirements will be a challenging and fruitful area of study.

VI. Conclusions

Of the three main parameters controlling DE's behaviour—population size, scaling factor F and crossover rate Cr—Cr is perhaps the most important. At its extremes Cr leads to vastly different search behaviours. Low values of Cr result in a search that is not just aligned with a small number of search space axes, but which is gradual, slow and robust. High values of Cr result in searches where fewer generated solutions may be improving, but the improvements can be large. Those searches rely on the population contracting in order to scale subsequent moves for finer-grained search. Both F and the population size have roles to play in controlling this convergence rate.

Finding the most appropriate settings for a problem can be time-consuming and error-prone, so adaptive and self-adaptive techniques are an attractive solution. However, due to the disjoint nature of DE's search behaviour as Cr is varied, techniques that sample values across Cr's entire range will use the best values only some of the time,

techniques that learn appropriate values based on success rate will likely select low values only, and techniques that make gradual changes will, if appropriate selection pressure is applied, be unable to move from one extreme to the other. With this new understanding of DE's operation at high and low crossover rates, future work will examine rules-based approaches to exploit and accommodate for the strengths and weaknesses of the two kinds of DE search.

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, 1997.

[2] K. Price, R. Storn, and J. Lampinen, *Differential Evolution: A Practical Approach to Global Optimization*. Berlin: Springer, 2005.

[3] J. Rönkköen, S. Kukkonen, and K. Price, "Real-parameter optimization with differential evolution," in *IEEE Congress on Evolutionary Computation, CEC 2005*. IEEE, 2005, pp. 567–574.

[4] D. Zaharie, "Influence of crossover on the behavior of differential evolution algorithms," *Applied Soft Comput.*, vol. 9, no. 3, pp. 1126–1138, 2009.

[5] S. Chen, "An analysis of locust swarms on large scale global optimization problems," in *4th Australian Conf. on Artificial Life (ACAL09)*, ser. LNAI, K. Korb, M. Randall, and T. Hendtlass, Eds., vol. 5865. Melbourne, Australia: Springer, 2009, pp. 211–220.

[6] S. Chen and Y. Noa Vargas, "Improving the performance of particle swarms through dimension reductions — a case study with locust swarms," submitted to *2010 IEEE Congress on Evolutionary Computation*.

[7] I. Moser and T. Hendtlass, "A simple and efficient multi-component algorithm for solving dynamic function optimisation problems," in *IEEE CEC 2007*, 2007, pp. 252–259.

[8] Y. Zhenyu, T. Ke, and Y. Xin, "Multilevel cooperative coevolution for large scale optimization," in *IEEE CEC 2008*, 2008, pp. 1663–1670.

[9] E. Mezura-Montes, J. Velázquez-Reyes, and C. A. Coello Coello, "A comparative study of differential evolution variants for global optimization," in *GECCO 2006*, Seattle, Washington, USA, 2006, pp. 485–492.

[10] K. Tang, X. Yao, P. N. Suganthan, C. MacNish, Y. Chen, C. Chen, and Z. Yang, "Benchmark functions for the cec 2008 special session on large scale globla optimization," Nature Inspired Computation and Applications Laboratory, USTC, China, Technical Report, 2007.

[11] J. Montgomery, "Crossover and the different faces of differential evolution searches," submitted to *2010 IEEE Congress on Evolutionary Computation*.

[12] D. Zaharie, "Critical values for the control parameters of differential evolution algorithms," in *MENDEL 2002, 8th International Conference on Soft Computing*, R. Matoušek and P. Ošmera, Eds., Brno, Czech Republic, 2002, pp. 62–67.

[13] J. Montgomery, "The effect of different kinds of move in differential evolution searches," in *4th Australian Conf. on Artificial Life*, ser. LNAI, K. Korb, M. Randall, and T. Hendtlass, Eds., vol. 5865. Melbourne, Australia: Springer, 2009, pp. 272–281.

[14] S. Chen and V. Lupien, "Opimtization in fractal and fractured landscapes using locust swarms," in *4th Australian Conf. on Artificial Life (ACAL09)*, ser. LNAI, K. Korb, M. Randall, and T. Hendtlass, Eds., vol. 5865. Melbourne, Australia: Springer, 2009, pp. 232–241.

[15] J. Montgomery, "Differential evolution: Difference vectors and movement in solution space," in *IEEE CEC 2009*. Trondheim, Norway: IEEE, 2009, pp. 2833–2840.

[16] J. Liu and J. Lampinen, "A fuzzy adaptive differential evolution algorithm," *Soft Comput.*, vol. 9, no. 6, pp. 448–462, 2005.

[17] A. K. Qin and P. N. Suganthan, "Self-adaptive differential evolution algorithm for numerical optimization," in *IEEE CEC 2005*, vol. 2, 2005, pp. 1785–1791.

[18] J. Tvrdik, "Differential evolution with competitive setting of control parameters," *Task Quarterly*, vol. 11, no. 1–2, pp. 169–179, 2007.

[19] J. Brest, B. Bošković, S. Greiner, V. Žumer, and M. Sepesy Maucec, "Performance comparison of self-adaptive and adaptive differential evolution algorithms," *Soft Comput.*, vol. 11, no. 7, pp. 617–629, 2007.

[20] J. Brest, S. Greiner, B. Bošković, M. Mernik, and V. Žumer, "Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems," *Evolutionary Computation, IEEE Transactions on*, vol. 10, no. 6, pp. 646–657, 2006.

[21] J. Brest, A. Zamuda, B. Boškovic, M. Sepesy Maučec, and V. Žumer, "High-dimensional real-parameter optimization using self-adaptive differential evolution algorithm with population size reduction," in *IEEE CEC 2008*, 2008, pp. 2032–2039.

[22] J. Brest and M. Sepesy Mauičec, "Population size reduction for the differential evolution algorithm," *Applied Intelligence*, vol. 29, no. 3, pp. 228–247, 2008.

[23] A. Salman, A. P. Engelbrecht, and M. G. H. Omran, "Empirical analysis of self-adaptive differential evolution," *Eur. J. Oper. Res.*, vol. 183, no. 2, pp. 785–804, 2007.

[24] A. Zamuda, J. Brest, B. Bošković, and V. Žumer, "Large scale global optimization using differential evolution with self-adaptation and cooperative co-evolution," in *IEEE CEC 2008*, 2008, pp. 3718–3725.