Appl Intell (2015) 42:514–526
DOI 10.1007/s10489-014-0613-2

Measuring the curse of dimensionality and its effects on particle swarm optimization and differential evolution

Stephen Chen · James Montgomery · Antonio Bolufé-Röhler

Published online: 14 November 2014

Abstract The existence of the curse of dimensionality is well known, and its general effects are well acknowledged. However, and perhaps due to this colloquial understanding, specific measurements on the curse of dimensionality and its effects are not as extensive. In continuous domains, the volume of the search space grows exponentially with dimensionality. Conversely, the number of function evaluations budgeted to explore this search space usually grows only linearly. The divergence of these growth rates has important effects on the parameters used in particle swarm optimization and differential evolution as dimensionality increases. New experiments focus on the effects of population size and key changes to the search characteristics of these popular metaheuristics when population size is less than the dimensionality of the search space. Results show how design guidelines developed for low-dimensional implementations can become unsuitable for high-dimensional search spaces.

Keywords Curse of dimensionality · Large scale global optimization · Particle swarm optimization · Differential evolution · Exploration · Exploitation

S. Chen (✉)
School of Information Technology, York University, Toronto, Canada
e-mail: sychen@yorku.ca

J. Montgomery
School of Engineering and ICT, University of Tasmania, Hobart, Australia
e-mail: james.montgomery@utas.edu.au

A. Bolufé-Röhler
School of Mathematics and Computer Science, University of Havana, Havana, Cuba
e-mail: bolufe@matcom.uh.cu

1 Introduction

Our intuitive understanding of population-based heuristic search techniques is often based on examples formed in two dimensions. Of note, particle swarm optimization (PSO) starts with an image of birds circling in towards a food source in a corn field [1]. As we picture the 15–30 birds involved with this original conceptualization, we can imagine how the proposed search technique will achieve excellent coverage of the search space. However, these intuitive ideas can be inaccurate and even misleading in higher dimensional search spaces.

A simple guideline from differential evolution (DE) is to use a population size (n) that is ten times the dimensionality (d) of the search space [2]. However, going from $d = 2$ to $d = 42$ can increase the size of the search space by a factor of about $2^{40} \approx 10^{12}$. Whereas we might imagine 20 birds being able to fully explore a 10 m $\times$ 10 m courtyard, we should have a different visualization of 420 birds in an area similar to the size of the Atlantic Ocean.

The general effects of the "curse of dimensionality" have been well known since 1957 when Bellman first coined the term [3]. Nonetheless, despite the knowledge that search spaces grow exponentially, recommendations for population sizes include linear growth (e.g. [2]) and constant size (e.g. [4]). As the fundamental parameter of a population-based search technique, it is important to understand how the curse of dimensionality can cause a population size (guideline) that is capable of adequate search space coverage in low dimensional search spaces to degrade into sparse coverage in high dimensional search spaces. The necessity of this sparse coverage occurs because exponential increases in computational resources are not viable. Further, increases in search space dimensionality also increase the cost to converge to any local optimum, and this causes a rapid decrease

in the ability of a heuristic search technique to dedicate any effort to exploration.

Chu, Gao, and Sorooshian have argued that in high dimensional search spaces, metaheuristics must focus almost exclusively on gradient exploitation [5]. Randomized search (which is required for "deceptive" problems), has a negligible chance to succeed. Their specific example is making the global optimum an entire "quadrant" of the search space, and there are then 2^d quadrants [5]. In $d = 100$ dimensions, the chance to find the global optimum by randomly sampling the search space is 1 in 2^{100}. In a deceptive problem, it would thus take $O(2^{100})$ function evaluations to find the attraction basin of the global optimum before an exploitive (e.g. gradient based) mechanism could be deployed to find the actual global optimum. Thus, the cost of search/exploration becomes prohibitive in very high dimensions.

In agreement with this assessment, a series of experiments have been designed to explicitly measure the effects of the curse of dimensionality. Focusing on PSO and DE (the two most popular population-based search techniques for continuous-domain search spaces), the first set of experiments demonstrates why a constant population size is a popular guideline for the design of metaheuristics. The second set of experiments exposes an important limitation of population-based metaheuristics that use a population size that is less than the dimensionality of the search space. Together, these experiments lead to new insights on the effects of increasing dimensionality on the performance and behaviour of particle swarm optimization and differential evolution.

After a review of population size recommendations with $n > d$ for problems in low dimensions, new experiments which study the convergence properties (as opposed to the performance) for PSO and DE in low and high dimensions are presented in Section 3. These experiments find that $n < d$ is often required in high dimensions, so the effects of $n < d$ is studied in Section 4. Finally, a broad discussion of the new results and insights is conducted in Section 5 before concluding remarks are given in Section 6.

2 Background

It has long been known that the size of the population is an important control parameter in the performance of population-based metaheuristics [6]. Almost all research involving population-based metaheuristics involves tuning of the population size, and a large amount of research has been dedicated solely to the optimization of this parameter. Several samples of this research that specifically deal with particle swarm optimization and differential evolution are highlighted below.

One of the most popular standards for particle swarm optimization [4] uses a constant population size of $n = 50$ on a series of benchmark functions with dimensions of $d = 2$, 4, and 30. The authors state that "While it may certainly be beneficial to tune this parameter (n) based on the problem at hand, generally speaking it is of minor importance." This claim is based on the evidence that "no swarm size between 20–100 particles produced results that were clearly superior or inferior to any other value for a majority of the tested problems."

Other experiments with PSO have also concluded that "the performance of PSO is not sensitive to the population size, and PSO scales well." [7] This and another experiment [8] studied PSO for dimensions of $d = 10 - 30$ and populations sizes of $n = 10 - 160$. In general, these and other experiments have focused on "performance" (in terms of the error from the known optimum). The experiments in this paper will focus on "convergence time", and the presented results offer new and different insights into the effects of increasing dimensionality on the behaviour of PSO and DE.

In DE, a linear growth in population size with dimension is a popular guideline. This guideline was first formalized as $n = 10d$ in [2], and supported by a subsequent study which focused on $n = 2d$, $4d$, $6d$, $8d$, and $10d$ [9]. Although it was noted that "the convergence speed can be slower" for larger population sizes [9], this factor is not yet dominant for low dimensional problems of $d = 10$ and 30. The following experiments consider problem sizes of $d = 5 - 1000$.

When other researchers have studied DE in $d > 50$ dimensions, they have tended to get better results with population sizes smaller than those considered in the previous studies. Dragoi et al. (e.g. [10]) have used $n \approx d$ for $d = 250 - 1000$, and $n = 100$ for $d = 100$ has been used in [11]. However, these studies have not explicitly focused on the relationship between n and d as d becomes larger.

3 Linear growth and constant sized populations

Popular recommendations for population size as a function of dimensionality include linear growth (e.g. [2]) and constant size (e.g. [4]). To evaluate these recommendations, two sets of experiments are conducted – one in lower dimensions ($d = 5 - 50$) and one in higher dimensions ($d = 50 - 500$). Current benchmark sets often focus on (lower) dimensions of $d \leq 50$ (e.g. [12, 13]), so the two ranges selected for testing are useful and able to show different behaviours at different search scales. The experiments measure the computational effort required for PSO and DE to converge "close" to the optimum on sphere. The sphere function is chosen because it is the simplest (unimodal) baseline function used in many benchmark sets (e.g. [12, 13]). If a search technique is unable to reach the optimum

on sphere, the error on sphere suggests a minimum error that will exist for any bowl-shaped (local) optimum on a deceptive (multi-modal) problem.

The current implementation of sphere uses the range of ± 100 and the limit of $10{,}000d$ function evaluations from [13]. Starting with uniform random solutions on this range, we record the number of function evaluations required to first produce a solution within a ± 0.1 hypercube (for PSO) or a ± 1 hypercube (for DE) around the origin (where the un-shifted global optimum is located). The choice of a target hypercube as opposed to a target fitness should make it easier to generalize the results – if the search techniques cannot converge to the hypercube, they will produce a corresponding error for any fitness function. (In general, PSO converged much faster than DE, so the key trends were more visible when a smaller hypercube was used.) Thirty independent trials are run for each data point. Except where noted, either zero or all thirty trials successfully converged for most of the data points. Variations amongst the trials were also negligible (with the excess of trials mostly to confirm the key trends), so only averages of the converged trials are presented below.

3.1 Particle swarm optimization

A version of standard particle swarm optimization [4] with a ring topology has been implemented. The key parameters specified from this standardization are $\chi = 0.72984$, and $c_1 = c_2 = 2.05$ for the velocity updates given in (1). (See Fig. 7 in Section 4.1 for more insight into the effects of this update equation.) Additional implementation details are the use of zero initial velocities [14] and "Reflect-Z" for particles that exceed the boundaries of the search space (i.e. reflecting the position back into the search space and setting the velocity to zero) [15]. The source code for this implementation is available on-line [16].

$$v_{i+1,d} = \chi(v_{i,d} + c_1\epsilon_1(\text{pbest}_{i,d} - x_{i,d}) + c_2\epsilon_2(\text{lbest}_{i,d} - x_{i,d})) \quad (1)$$

The first set of experiments involves a 10 by 10 grid in parameter space – dimensions d from 5 to 50 in steps of 5 and population sizes n from 5 to 50 also in steps of 5. The average number of generations (i.e. function evaluations divided by population size) to converge close to the optimum (i.e. reach the target hypercube) on sphere is reported in Table 1. In general, most studies (e.g. [7–9]) and most benchmark functions (e.g. [12, 13]) focus on "performance" – the actual value of the fitness function and/or its difference from the known global optimum. By focusing on the given measure of "convergence time" the following observations can be made.

For extremely small population sizes (i.e. $n = 5$ and $d \geq 30$), insufficient diversity can cause PSO to stall – from 2 ($d = 30$) to 21 ($d = 50$) trials never reach the target hypercube around the optimum, and those that do can take a rapidly increasing number of generations. For larger population sizes (e.g. $n \geq 30$), the number of generations required for convergence is mostly constant with respect to population size with a slight benefit (i.e. fewer generations required) for larger values of n. This result shows that the "convergence rate" depends more on the generations (iterations per particle) than on the population size.

It is typical to use a fixed budget of function evaluations in comparisons of metaheuristics. Under this constraint, the number of allowed generations will vary inversely with the population size. If the population size increases too much, the performance of PSO will be adversely affected. These changes in performance may not be highly visible since the sphere function gets flatter and flatter near the optimum. Thus, the previous observation that "the performance of PSO is not sensitive to the population size, and PSO scales well" [7] misses the behavioural trends shown in Table 1.

To emphasize this trend from Table 1, Fig. 1 shows the convergence time (to the target hypercube) with respect to both function evaluations and generations. For $d = 50$ and $n = 5 - 50$, the range for function evaluations is quite small (i.e. the largest and smallest values vary by a factor of about 2) while the range for generations is quite large (i.e. varying by a factor of about 10). Focusing on this small range for function evaluations, it is easy to understand how previous researchers might have concluded that "the performance of PSO is not sensitive to the population size" [7]. However, the following experiments focusing on generations will help lead us to alternative conclusions.

The second set of experiments also involves a 10 by 10 grid in parameter space, but dimensions d now range from 50 to 500 in steps of 50 and population sizes n range from 50 to 500 in matching steps of 50. With these larger population sizes, the effects of losing diversity (e.g. with $n = 5$) become less, so two key trends become more easily visible in Table 2. First, the convergence time in generations is roughly constant with respect to population size (with only a slight downward trend as n increases). Second, the approximate growth rate in generations (and thus function evaluations for each population size n) required to converge as a function of dimensions d is faster than a linear function with a constant of 1. Since the allowed function evaluations increase linearly (i.e. function evaluations $= 10{,}000d$), the growth rate in generations to converge eventually eclipses the growth rate in function evaluations allowed. PSO with $n = 450$ cannot converge for $d \geq 450$ dimensions, and $n = 500$ fails for $d \geq 250$ dimensions.

Table 1 Generations required to converge to the ±0.1 hypercube for PSO with $n = 5 - 50$ and $d = 5 - 50$

	Dimensions (d)									
n	5	10	15	20	25	30	35	40	45	50
5	114.4	325.0	774.2	1610.8	2698.2	4359.3	6013.6	7263.9	11469.9	11178.8
10	100.7	195.3	306.2	429.5	580.2	749.7	906.2	1221.6	1413.2	1778.0
15	94.7	189.5	294.5	384.9	485.8	605.8	709.1	817.8	927.9	1080.8
20	93.2	180.9	284.8	380.3	480.3	575.6	683.3	792.9	885.5	996.5
25	92.9	181.0	282.1	376.9	480.5	572.7	673.3	779.7	868.6	966.7
30	88.0	182.2	275.6	375.5	466.9	569.5	662.1	765.0	873.3	960.3
35	86.8	180.3	269.2	371.9	468.8	563.0	663.0	763.8	861.3	968.0
40	87.6	175.3	269.0	368.2	463.1	555.1	652.5	764.0	853.6	957.5
45	84.6	177.9	267.2	364.0	458.8	559.4	647.5	761.0	869.5	958.8
50	85.8	176.2	269.2	365.5	461.9	555.8	662.2	752.7	854.9	957.2

This super-linear trend is more easily seen again if dimensions are increased to $d = 1000$. In Fig. 2, the actual generations required for PSO to converge with $n = 50$ and $d = 50 - 1000$ (in steps of 50) is shown against a linear extrapolation. Since the budget of function evaluations grows only linearly with dimension d, it becomes clear that for any population size n, there will eventually be a problem size d for which a linear budget of function evaluations (e.g. 10,000d) will become insufficient to achieve a desired level of convergence (e.g. $d = 250$ for $n = 500$ and a target hypercube of ±0.1).

3.2 Differential evolution

A DE/rand/1/bin version of DE has been implemented with crossover $Cr = 0.9$ and scale factor $F = 0.8$ [17]. Each population member (x_i) is considered as a target for replacement by a candidate solution that is constructed in two steps: creation of an intermediate solution and crossover with x_i. During the creation of an intermediate solution (y_i) from three distinct random solutions (r_1, r_2, and r_3) in (2), the scale factor F affects the "step size" from r_1 taken in the direction of the "difference vector" created with r_2 and r_3. (See Fig. 10 in Section 4.2 for more insight into the effects of this movement.)

$$y_i = r_1 + F(r_2 - r_3) \tag{2}$$

This intermediate solution is then crossed (term-by-term in each dimension of the search space) with the target solution (x_i) to produce a new candidate solution (x_i'). In (3), a low value of Cr leads to more terms from the target x_i, and this is often useful on separable functions such as sphere because changing fewer dimensions can exploit the separable nature of the objective function [18]. However, our intention in the following experiments is to observe the "typical" behaviour, and the most common value of Cr is 0.9 [18]. This high value of Cr is often more effective on

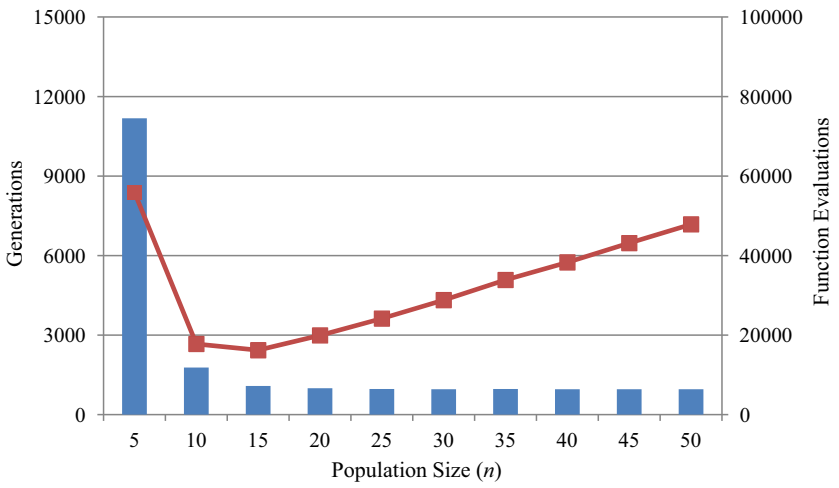

Fig. 1 Time for PSO to converge for $n = 5 - 50$ and $d = 50$ as measured in generations and function evaluations. Function evaluations vary by a factor of about 2 while generations vary by a factor of about 10

Table 2 Generations required to converge to the ± 0.1 hypercube for PSO with $n = 50 - 500$ and $d = 50 - 500$

	Dimensions (d)									
n	50	100	150	200	250	300	350	400	450	500
50	959.4	1982.9	3099.6	4251.3	5465.6	6816.1	8121.4	9346.4	10807.0	12361.0
100	935.9	1961.3	3047.1	4212.2	5416.2	6653.1	7905.1	9234.7	10579.8	11899.0
150	924.4	1963.1	3053.3	4174.4	5371.7	6611.6	7879.1	9180.5	10480.0	11870.7
200	920.1	1942.8	3021.8	4177.8	5348.0	6562.4	7830.4	9107.8	10436.3	11738.2
250	920.6	1944.1	3015.7	4148.9	5335.3	6559.6	7814.1	9078.4	10416.2	11776.3
300	918.2	1932.7	3016.9	4134.9	5299.7	6563.9	7763.8	9059.8	10439.3	11739.0
350	914.2	1938.3	3008.6	4129.6	5296.7	6557.4	7793.1	9045.3	10392.5	11692.9
400	900.5	1918.4	3003.2	4118.1	5307.1	6548.5	7805.5	9069.5	10312.4	11685.8
450	909.4	1919.1	3016.5	4121.0	5284.3	6504.5	7710.9	8821.3	–	–
500	913.3	1929.3	2971.6	3974.0	–	–	–	–	–	–

difficult large-scale problems, which are neither unimodal nor separable [18, 19].

$$x'_{i.d} = \begin{cases} y_{i.d} & u_d \leq Cr \\ x_{i.d} & u_d > Cr \end{cases} \tag{3}$$

The same two experiment sets performed with PSO were conducted with DE (but with a larger target hypercube of ± 1 to accommodate the slower convergence rate of DE). The average number of generations to converge close to the optimum/reach the target hypercube on sphere is reported in Table 3. As with PSO, smaller populations, in particular $n = 5$, can stall before reaching the optimum. With $n = 5$ and $d = 5$ only a single trial found the target hypercube around the optimum, while $n = 5$ was completely ineffective with values of $d > 5$.

For all other combinations of population size ($n = 10 - 50$) and problem size ($d = 5 - 50$), DE was able to find the target region in all 30 trials. However, the key performance trend first presented in PSO is still present – for each fixed population size n, the number of generations required to find the target region increases super-linearly with problem dimension d. This super-linear increase is most visible with $n = 10$. Although there is sufficient diversity to not stall out completely, the contraction of the population leads to very small difference vectors and thus very slow progress towards the optimal solution.

Inversely, too much diversity also leads to poor performance in DE. The fastest convergence for $d = 5 - 50$ occurs with $n = 20$. For $n > 20$, the number of generations required to reach the target hypercube also increases with n. (Additional analysis showed that the population upon reaching the target was also more diverse with larger values of n.) This negative effect of too much diversity is clearly shown in the second grid of experiments with $n = 50 - 500$ and $d = 50 - 500$ (see Table 4). For $n \geq 100$, none of the trials for $d = 50 - 500$ were able to reach the target hypercube in the allowed function evaluations. Larger population sizes maintain high levels of diversity and, consequently, large magnitude difference vectors, which produce ineffective exploratory moves once population members have

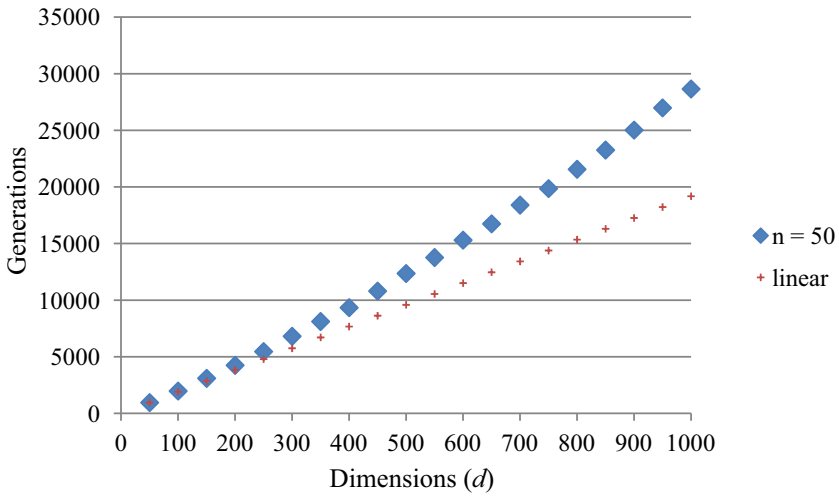

Fig. 2 Actual generations required for PSO to converge versus an extrapolation from the origin through $d = 50$. It can be seen that the actual generations increases at a super-linear rate

Table 3 Generations required to converge to the ± 1 hypercube for DE with $n = 5 - 50$ and $d = 5 - 50$

	Dimensions (d)									
n	5	10	15	20	25	30	35	40	45	50
5	77.0	–	–	–	–	–	–	–	–	–
10	65.9	182.7	386.7	833.2	1402.2	2355.5	2327.3	2793.4	4195.0	5045.8
15	68.7	172.1	280.8	407.2	606.3	760.1	955.0	1265.0	1535.3	1932.0
20	68.8	195.9	350.1	488.5	598.2	761.9	931.4	1102.3	1336.8	1545.0
25	68.8	225.3	419.2	573.0	738.0	889.9	1049.1	1224.8	1403.0	1634.2
30	68.2	240.6	478.0	703.4	902.6	1047.3	1215.4	1400.9	1618.4	1797.6
35	69.2	253.1	537.5	808.1	1082.8	1304.2	1510.9	1707.4	1895.3	2127.8
40	71.1	258.5	583.8	946.9	1278.5	1501.9	1713.9	1991.3	2200.8	2401.9
45	68.8	263.2	626.4	1061.0	1465.3	1702.7	2048.6	2290.9	2513.7	2798.1
50	68.8	265.4	656.4	1190.4	1663.3	2033.1	2311.4	2598.2	2832.9	3149.5

undergone some initial improvement [19]. Compared to experiments measuring performance, the binary indicator of a failure to converge shows more starkly these effects. These effects are not affected by the omitted standard deviations for the reported means.

3.3 Convergence and diversity

The typical parameters suggested for PSO and DE work well enough for small values of d (e.g. $d \leq 50$), but limitations begin to appear for $d > 50$. In PSO, larger populations tend to help – the increased diversity leads to fewer generations required to first enter the hypercube close to the optimum. In DE, a larger population appears to have a negative effect – the maintenance of too much diversity allows the difference vectors to stay too large to achieve convergence in the allotted budget of function evaluations.

Following the advice in [5] to focus on "gradient exploitation" in high dimensional search spaces, the following modifications have been made to PSO and DE. In PSO, exploitation is increased by increasing constriction. With less momentum, particles overshoot their personal and local best attractors by less, and they thus spend more time moving towards them which is essentially gradient exploitation. In DE, the parameter F serves a similar role to the constriction factor in PSO, with smaller values increasing exploitation. As shorter step sizes create a less diverse population which has smaller difference vectors amongst the population members, the more homogenous population can align more quickly to the gradients in the search space and exploit them more effectively.

3.3.1 More constriction in PSO

In the following experiments, constriction is increased by reducing the value of χ in (1). The reduction occurs by multiplying χ by factors of $w = 0.9$, 0.8, and 0.7 (i.e. simple reductions of 10 %). Similar to experiments

Table 4 Generations required to converge to the ± 1 hypercube for DE with $n = 50 - 500$ and $d = 50 - 500$

	Dimensions (d)									
n	50	100	150	200	250	300	350	400	450	500
50	3150	6560	11979	19793	35033	43796	59930	–	–	–
100	–	–	–	–	–	–	–	–	–	–
150	–	–		–	–	–	–	–	–	–
200	–	–	–	–	–	–	–	–	–	–
250	–	–	–	–	–	–	–	–	–	–
300	–	–	–	–	–	–	–	–	–	–
350	–	–	–	–	–	–	–	–	–	–
400	–	–	–	–	–	–	–	–	–	–
450	–	–	–	–	–	–	–	–	–	–
500	–	–	–	–	–	–	–	–	–	–

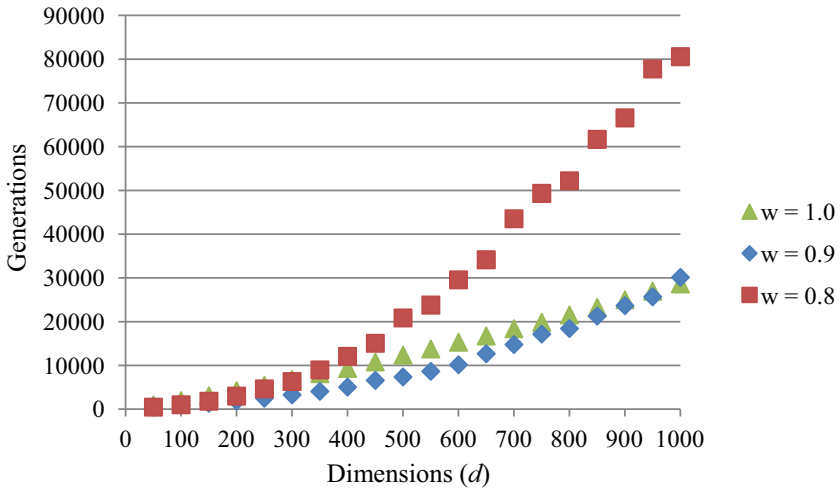

Fig. 3 The effect of increased constriction on the generations required for PSO to converge. Less momentum can increase the convergence rate, but too little momentum can cause PSO to stall ($n = 50$, $\chi = w * 0.72984$, and $c_1 = c_2 = 2.05$)

presented in Fig. 2 (where the key trends are most visible for $d > 500$), PSO is run with $n = 50$ on $d = 50 - 1000$ for 30 independent trials. The new results are presented in Fig. 3.

The number of generations required to converge generally improves with $w = 0.9$. With $w = 0.8$, a similar improvement occurs until about $d = 300$, and then the premature loss of diversity causes convergence to the target hypercube to slow noticeably for $d > 300$. The results for $w = 0.7$ are not shown – the loss of diversity is so severe that the performance is never better than the original result (i.e. $w = 1.0$) and most of the trials never reach the target hypercube. With the faster convergence caused by increased constriction, larger populations may be viable, but a full parametric study is beyond the scope of this paper. Nonetheless, the current results suggest that the recommended constriction factor in [4] does not scale well to very high dimensions.

3.3.2 Smaller F in DE

In the following experiments, step sizes are reduced by using a scale factor of $F = 0.4$. To illustrate the algorithm's behavior, Fig. 4 shows the average generations until the target region was found for $n = 100$, 200, 300, 400, and 500. A population size of $n = 50$ was also investigated, but with the current scale factor of $F = 0.4$, DE converges prematurely with $n = 50$ for all $d > 50$.

When $n = 100$, the algorithm is able to find the target region for all problem sizes, but exhibits super-linear growth in the number of generations required as d increases. Thus, while $n = 100$ provides sufficient diversity to prevent premature convergence, the population still contracts to a small area, leading to small difference vectors and slow search progress. Within each problem size, the number of generations required diminishes as population size is increased, although the number of function evaluations used by the algorithm does not reduce at the same rate. Figure 5 shows the number of function evaluations represented by the generations shown in Fig. 4 and shows that, in this instance, the most efficient search is with a population of $n = 200$ individuals. Improved performance cannot therefore be guaranteed by continuing to increase DE's population size.

Given that a population smaller than d can produce the most efficient search, the common decision to use a relatively small population with high dimensional problems is

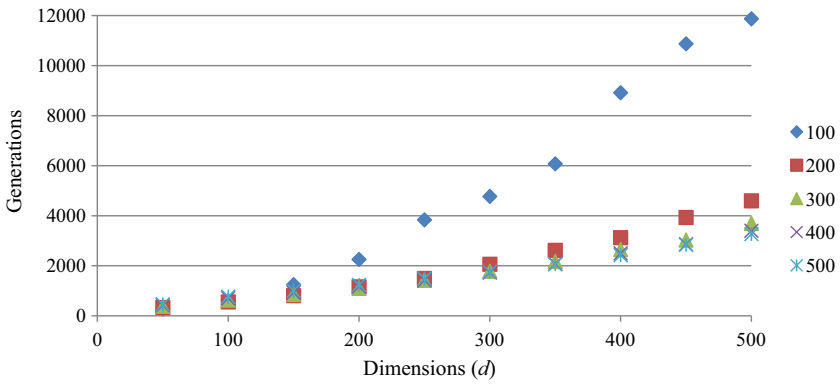

Fig. 4 Generations required for DE to converge when smaller F is used. For $n \geq 200$, generations becomes roughly constant with respect to n

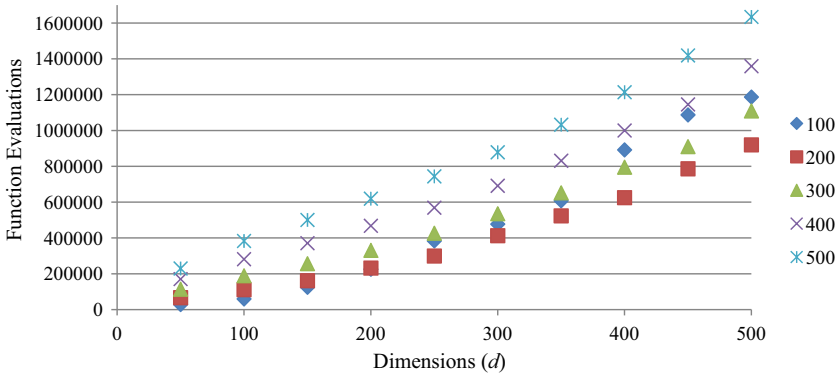

Fig. 5 Function evaluations required for DE to converge when smaller F is used. Population size can be seen here to affect performance, but the key features relating to generations become less visible here as well

somewhat justified. However, an effective value for F must still be identified. Figure 6 shows the average number of generations to find the target hypercube for $n = 50$ and a full range of F in 0.1 increments (i.e. $F = 0.5, 0.6, 0.7$, and 0.8). Note, noise in the observed trends when $F > 0.5$ is due to decreasing success rates as d increases. The shaded area represents generation counts above the maximum imposed by the FE limit. These results also indicate that lowering F can lead to improved search efficacy. However, while the trend in the plot suggests that continuing to reduce F would produce further improvements, experiments with $F = 0.4$ resulted in stalling for all $d > 50$. The balance between allowing the population to contract (so that it can make effective exploratory moves) while maintaining sufficient diversity is evidently critical to DE's success, but it can be difficult to achieve.

4 The effects of population size less than dimensionality

From basic geometry, two points define a line, three points define a plane, and n (non-degenerate) points define an n-1 dimensional hyperplane. The 50 population members used in several of the previous experiments will thus define a 49-dimensional hyperplane that is a partial subspace in the problems with $d = 50 - 1000$ dimensions. The primary (exploitative) search mechanisms of PSO (i.e. attraction vectors – see (1)) and DE (i.e. difference vectors – see (2)) will stay within the hyperplane of the current population. For the current problem, the global optimum is at the origin which is a part of every possible hyperplane. However, like a line-search in a plane, any search restricted to a hyperplane of lower dimensionality than the total search space will have important limitations.

4.1 Escaping the hyperplane in PSO

The effects of the velocity update in (1) are often visualized by a vector diagram such as that shown in Fig. 7. In this diagram, $a_i = \chi v_i$, $b_i = c_1\epsilon_1(\text{pbest}_i - x_i)$, and $c_i = c_2\epsilon_2(\text{lbest}_i - x_i)$. The vectors b_i and c_i are typically drawn as shortened (i.e. scaled) versions of the attraction vectors to their respective attractors. However, since "ϵ_1 and ϵ_2 are independent random numbers uniquely generated at every update for each individual dimension $d = 1$ to D" [4], the actual attraction vector could have up to a 45° angle with the vector defined by the attractor and x_i. The vector b_i' shows a possible example of this vector that is much less intuitive than the typical visualization.

Deterministic versions of PSO [20], such as the Canonical Deterministic PSO (CDPSO) are "derived from the stochastic PSO by omitting the stochastic factors" [21]. Instead of the standard PSO (1), the deterministic version generates new solutions using (4), where p is a desired fixed

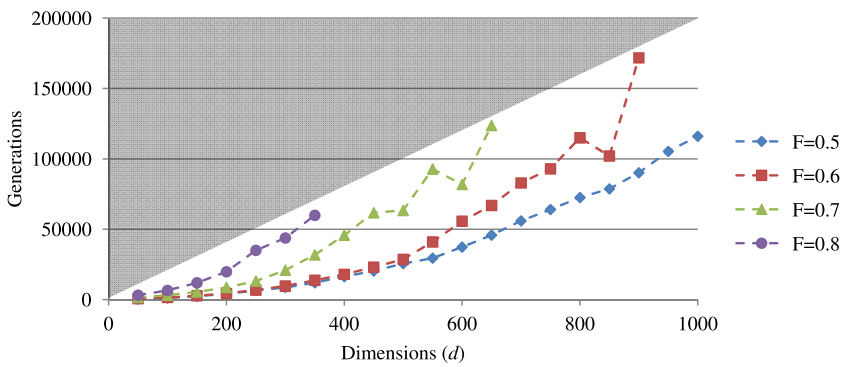

Fig. 6 The effect of F on the convergence of DE for $n = 50$ and $d = 50 - 1000$. The shaded area represents exceeding the maximum generations possible for the allotted function evaluations

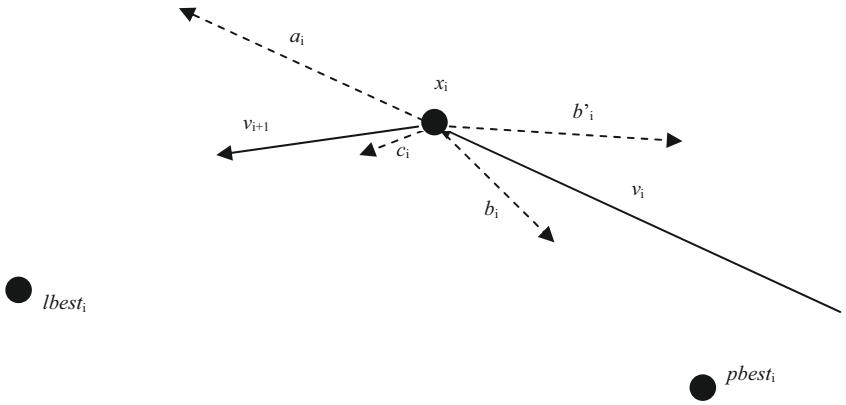

Fig. 7 Vector-based diagram to visualize velocity update (1) in PSO. The effects of non-scalar weights (e.g. b'_i) are usually omitted

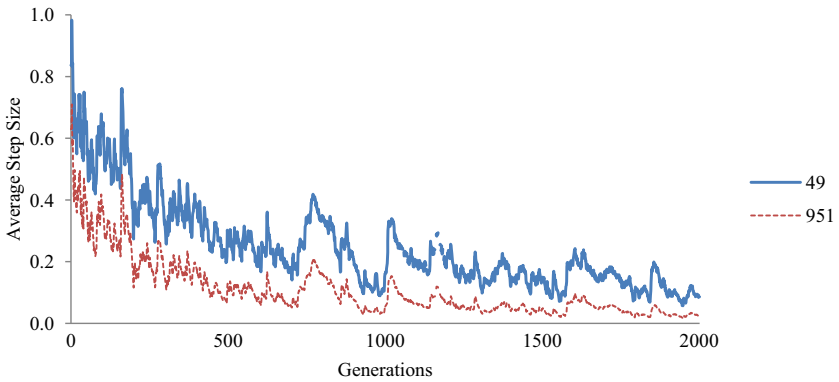

Fig. 8 Average change for each dimension for the 49 dimensions inside the current hyperplane (*top blue*) and the 951 dimensions outside the current hyperplane (*bottom red*) for PSO. Search outside the hyperplane is less than search inside the hyperplane

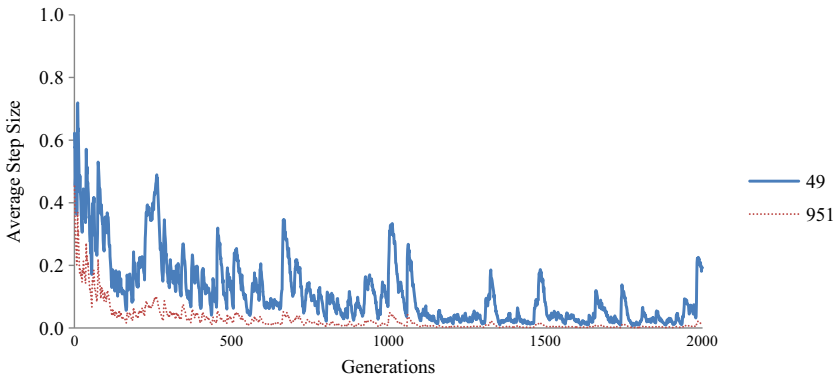

Fig. 9 Average change for each dimension for the 49 dimensions inside the current hyperplane (*top blue*) and the 951 dimensions outside the current hyperplane (*bottom red*) for CDPSO. Search outside the hyperplane is much less than search inside the hyperplane (especially in comparison to PSO – see Fig. 8)

Fig. 10 A difference vector formed by r_2 and r_3 is used to generate a search motion from r_1

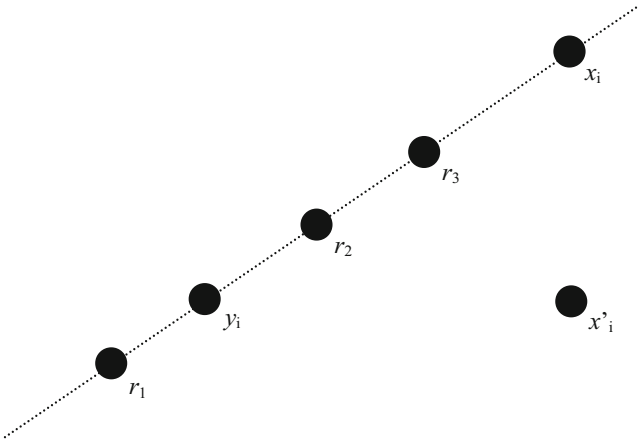

Fig. 11 The effects of crossover can produce an offspring x_i' outside of a hyperplane/subspace defined by r_1, r_2, r_3, and x_i

point and $\gamma = c_1/(c_1 + c_2)$ controls the mixture rate of the personal best and the local best. (Note: since $c_1 = c_2$, $\gamma = 0.5$ and the desired fixed point is the midpoint of attractions towards pbest and lbest.)

$$p = (\gamma)\text{pbest} + (1 - \gamma)\text{lbest} \tag{4}$$

In CDPSO, the absence of the ϵ_1 and ϵ_2 "random vectors" decreases the chances of generating new solutions outside the hyperplane defined by the existing population members. However, clamping effects when particles leave the boundaries of the search space can still cause new solutions to "escape" from the current hyperplane in deterministic versions of PSO. Further, after these initial escapes, the pbest and lbest attractors may become points outside the hyperplane defined by the current population members, so subsequent motions will no longer be restricted to the current hyperplane.

To determine how strong the search is inside and outside of the hyperplane, experiments comparing PSO and CDPSO have been performed. Using a population size of $n = 50$ for a version of sphere with $d = 1000$ (the extreme case for the current experimental range), the step sizes/search magnitudes inside the hyperplane and outside the hyperplane are recorded during a single trial. Specifically, for each generation, the 49-dimensional hyperplane for the $n = 50$ population members is determined. Each "step vector" (i.e. the difference between a new position and the previous position) is projected into two components: a component inside the hyperplane, and an (orthogonal) component outside of the hyperplane. These components are then normalized by the number of dimensions inside and outside of the hyperplane to provide insight into how much search/exploration is performed inside the hyperplane versus outside the hyperplane.

In Figs. 8 and 9, the top line (in red) shows the average change for each dimension within the hyperplane, and the bottom line (in blue) shows the average change for each dimension outside of the hyperplane (for PSO and CDPSO respectively). Considering that there are 951 dimensions outside of the hyperplane and only 49 dimensions within it, the total magnitude of the step component outside of the hyperplane is usually larger than the total magnitude of the step component inside the hyperplane. However, the figures demonstrate that search in PSO does favour the current hyperplane when $n << d$, and this bias is increased in CDPSO which explicitly aims to reduce the effects of randomness.

4.2 Escaping the hyperplane in DE

The effects of the difference vector in (2) are often shown in a diagram which highlights how the key motion of DE is based on difference vectors (see Fig. 10). However, after crossover, a motion not related to difference vectors is highly prevalent. In particular, the example in Fig. 11 shows how four co-linear solutions (r_1, r_2, r_3, and x_i) can produce an offspring solution x_i' that is not on the line. The effects of crossover which are dimension-by-dimension allow DE to escape the hyperplane of its current population.

Similar to the previous demonstrations for PSO and CDPSO, Fig. 12 shows the average change for each dimension within the hyperplane (top line in red) and the average change for each dimension outside of the hyperplane

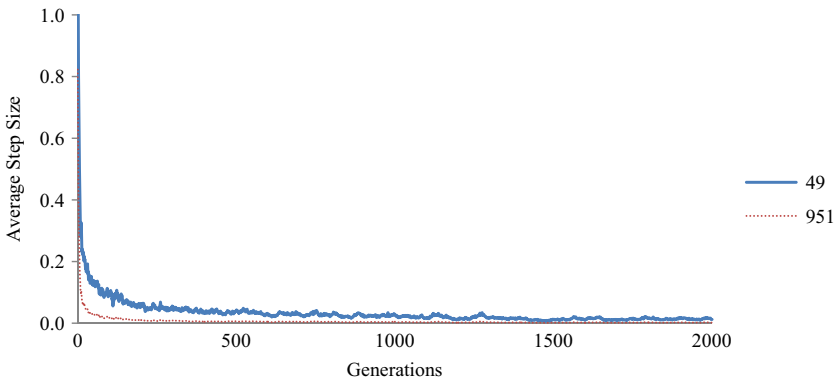

Fig. 12 Average change for each dimension for the 49 dimensions inside the current hyperplane (*top blue*) and the 951 dimensions outside the current hyperplane (*bottom red*) for DE. Search outside the hyperplane is less than search inside the hyperplane

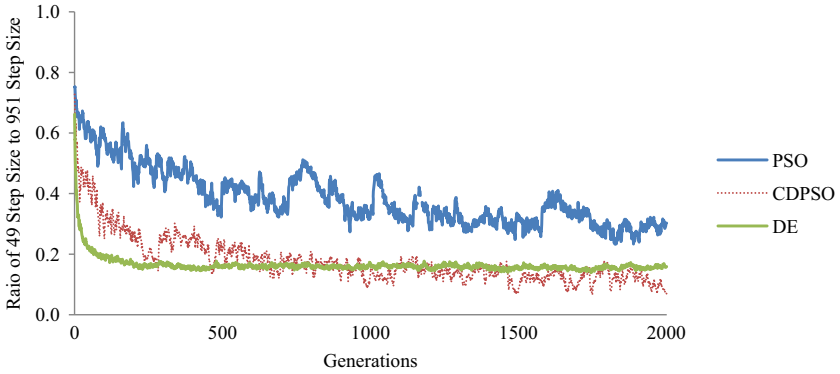

Fig. 13 Ratio of average change for each dimension outside the hyperplane to average change for each dimension inside the hyperplane for PSO, CDPSO, and DE. PSO is the least restricted to its current hyperplane

(bottom line in blue) for a single execution of DE on sphere with $d = 1000$ when using a population size of $n = 50$. When using a high value of Cr (i.e. 0.9), most of the final solution x_i' comes from the intermediate solution y_i constructed from r_1, r_2, and r_3. Thus, offspring solutions have only a small deviation from the current hyperplane.

4.3 Effects of escapes on search performance

Attraction vectors in PSO and difference vectors in DE are mechanisms for exploitation – information from existing solutions is used to create new solutions. The balance between exploration and exploitation in PSO, CDPSO, and DE is exaggerated by the previous analysis of step components inside and outside of the current hyperplane. In Fig. 13, the average ratio of the outside component to the inside component is shown at each generation for PSO, CDPSO, and DE. PSO with its "random vectors" ϵ_1 and ϵ_2 has the largest amount of exploration (as indicated by step components outside of the hyperplane).

While PSO is more explorative, CDPSO and DE are more exploitative. This greater focus on exploitation is shown by their faster convergence (see Fig. 14). However, less exploration eventually leads to a lack of diversity which affects the later performance of CDPSO and DE. Being less restricted to the current hyperplane helps PSO achieve better performance than CDPSO and DE – even when the global optimum at the origin is a part of all possible hyperplanes.

5 Discussion

The Nelder-Mead method is a well-established numerical method for finding (local) minima in multi-dimensional search spaces. It uses a series of specific moves (i.e. reflection, expansion, contraction, and reduction) that are based on difference vectors [22]. Since there are no random inputs to these moves, new search points are restricted to the $n-1$ dimensional hyperplane defined by the n points used in the simplex. Thus, a polytope of $n = d + 1$ dimensions is required for a d dimensional search space. However, the complexity of the Nelder-Mead method grows with the size of the polytope [23], so the standard algorithm can scale poorly to high dimensional search spaces (e.g. [24, 25]).

The primary search mechanism of differential evolution (2) is also restricted to the n-1 dimensional hyperplane defined by the n population members. Although the effects of dimension-by-dimension crossover can allow escapes from the hyperplane, the small ratio of activity in these exterior dimensions (see Fig.12) means that DE will adjust quite slowly to new hyperplanes and these adjustments can be important in the following of valleys and other search

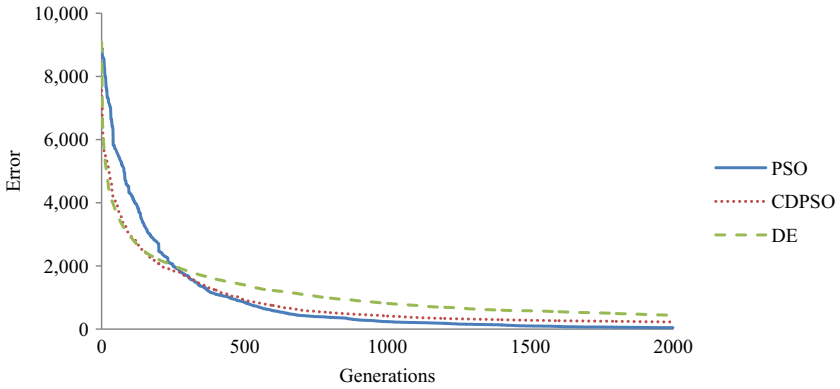

Fig. 14 Performance of PSO, CDPSO, and DE on sphere with $d = 1000$ when using a population size of $n = 50$. More diversity slows convergence at the beginning, but it also prevents stalling at the end

space contours. This inability to quickly leave a hyperplane defined by a population size n that is (much) less than the problem dimension may help explain why DE performs better in high dimensional search spaces when relatively large populations are used (e.g. [10]). These relatively large population sizes, however, are still less than the $n = 2d - 10d$ population sizes previously tested in lower dimensions [2, 9].

The use of a large population size (e.g. $n > d$) will meaningfully affect the computational complexity of a metaheuristic. The experiments in Section 3 demonstrate that convergence time measured in terms of generations grows as a super-linear function of problem dimensionality, so a population size with linear growth would lead to super-quadratic growth in function evaluations. Further, each function evaluation has a complexity of at least $O(d)$ for a simple function like sphere and likely $O(d^2)$ or much higher for any interesting real-world problem. To contain a computational complexity likely exceeding $O(d^3)$, the use of small/constant population sizes is desirable to achieve convergence in a reasonable time.

The use of small populations requires a compensatory mechanism to maintain activity in all d (non-axial) dimensions of the search space, and this is achieved with randomness in PSO. The "random vectors" ϵ_1 and ϵ_2 have their values in the range $[0, 1]$ which can alternately be viewed as a scalar value of 0.5 and a random vector in the range $[-0.5, 0.5]$. The maximum magnitude of this random vector (which causes the step component out of the hyperplane) is equal to the scalar applied to the attraction vector within the hyperplane. Thus, even with a large amount of randomness, there is still a larger attraction to the pbest and lbest attractors and thus an important bias that targets the hyperplane/subspace defined by the current population.

There is a computational cost associated with the generation of random numbers, so their removal from search techniques such as simulated annealing [26] to create modified forms such as threshold accepting [27] can lead to improved performance [28]. In addition to simpler analysis, deterministic versions of PSO [20, 21] also achieve this advantage. However, randomness increases motion, and almost any injection of motion which prevents the search process from stalling can improve the performance of PSO (e.g. [29]). Due to the increased risk of stalling, the current results suggest that CDPSO should not be used in high dimensional search spaces.

In general, the primary mechanisms of attraction vectors in PSO and difference vectors in DE act within the n-1 dimensional hyperplane defined by the n population members. When this hyperplane is less than the dimensionality of the problem, secondary mechanisms play a greatly increased role in the behaviour and performance of these search techniques. In PSO, secondary effects include the "random vectors" of ϵ_1 and ϵ_2 and boundary effects. In DE, the key secondary effect is the use of a crossover operator which acts dimension by dimension.

The smaller effect of the secondary mechanisms in DE means that population size n cannot become too much smaller than the problem dimensionality d. Fortunately, simple modifications (e.g. smaller scale factor F) allow DE to converge fast enough to support reasonably large population sizes. Conversely, a similarly simple modification (i.e. more constriction) has not worked for PSO, so smaller/constant population sizes seem to work best with PSO. However, these smaller population sizes greatly restrict the ability of PSO to conduct exploration in high dimensional search spaces.

The ability of PSO and DE to conduct exploration in high dimensional search spaces is also greatly affected by the super-linear growth in generations required for convergence. Even with a linear growth in budgeted function evaluations, this super-linear growth in generations begins to consume the entire budget of function evaluations to target a single local optimum. At the same time, the curse of dimensionality can cause the number of local optima to grow exponentially, so it becomes truly impossible to meaningfully explore a deceptive search space in high dimensions [5] (let alone to still have time to converge to a local optimum after said exploration). Techniques like PSO and DE which explore and exploit multiple optima may be suitable for lower dimensional search spaces, but their feasibility for very high dimensional search spaces will become strained as more and more of their overall search effort becomes consumed by convergence/exploitation.

6 Conclusions

Population size is a critical parameter affecting the performance of population-based search techniques. In low dimensional search spaces, it is common to use $n > d$. However, current experiments show that convergence time tends to grow super-linearly with dimensionality. In conjunction with the use of a linear increase in allotted function evaluations, this result means that high dimensional search spaces will likely require $n < d$ to allow sufficient generations for the search technique to converge.

To cover a d dimensional search space, it is necessary to have $d + 1$ (non-degenerate) points to define search vectors (e.g. attraction vectors in PSO and difference vectors in DE) that can cover the entire search space. The use of $n < d$ means that the primary search mechanisms of PSO and DE will be trapped within a lower dimensional hyperplane that is a subspace of the overall search space. Current experiments show that this entrapment can affect

the convergence properties on sphere – a function where the unshifted global optimum at the origin is within all possible subspaces. Effects on exploration in deceptive/multi-modal search spaces are likely larger and an important area for future research.

References

1. Kennedy J, Eberhart RC (1995) Particle swarm optimization. IEEE ICNN:1942–1948
2. Storn R (1996) On the usage of differential evolution for function optimization. EEE NAFIPS:519–523
3. Bellman RE (1957) Dynamic Programming. Princeton University Press, Princeton
4. Bratton D, Kennedy J (2007) Defining a standard for particle swarm optimization. IEEE SIS:120–127
5. Chu W, Gao X, Sorooshian S (2011) A new evolutionary search strategy for global optimization of high-dimensional problems. Inform Sci 181:4909–4927
6. Grefenstette JJ (1986) Optimization of control parameters for genetic algorithms. IEEE SMC 16(1):122–128
7. Shi Y, Eberhart RC (1999) Empirical study of particle swarm optimization. IEEE CEC:1945–1950
8. Zhang L-P, Yu H-J, Hu S-X (2005) Optimal choice of parameters for particle swarm optimization. J Zhejiang Univ Sci 2005 6A(6):528–534
9. Mallipeddi R, Suganthan PN (2008) Empirical study on the effect of population size on differential evolution algorithm. IEEE CEC:3663–3670
10. Dragoi E-N, Curteanu S, Leon F, Galaction A-I, Cascaval D (2011) Modeling of oxygen mass transfer in the presence of oxygen-vectors using neural networks developed by differential evolution algorithm. Eng Appl AI 24(7):1214–1226
11. Vafashoar R, Meybodi MR, Momeni Azandaryani AH (2012) CLA-DE: a hybrid model based on cellular learning automata for numerical optimization. Appl Intell 36(3):735–748
12. Hansen N, Finck S, Ros R, Auger A (2009) Real-parameter black-box optimization benchmarking 2009: noiseless functions definitions, INRIA Technical Report RR-6829
13. Liang JJ, Qu BY, Suganthan PN, Hernández-Díaz AG (2013) Problem definitions and evaluation criteria for the CEC 2013 special session on real-parameter optimization, Technical Report. Nanyang Technological University
14. Engelbrecht A (2012) Particle swarm optimization: velocity initialization. IEEE CEC:70–77
15. Helwig S, Branke J, Mostaghim S (2013) Experimental analysis of bound handling techniques in particle swarm optimization. IEEE TEC 17(2):259–271
16. https://www.researchgate.net/publication/259643342_Source_code_for_an_implementation_of_Standard_Particle_Swarm_Optimization_--_revised?ev=prf_pub
17. Storn R, Price K (1997) Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. J Glob Optim 11:341–359
18. Montgomery J, Chen S (2010) An analysis of the operation of differential evolution at high and low crossover rates. IEEE CEC:881–888
19. Montgomery J (2009) Differential evolution: Difference vectors and movement in solution space. IEEE CEC:2833–2840
20. Clerc M (1999) The swarm and the queen: towards a deterministic and adaptive particle swarm optimization. IEEE CEC:1951–1957
21. Jin'no K, Shindo T. (2010) Analysis of dynamical characteristic of canonical deterministic PSO. IEEE CEC:1105–1110
22. Nelder JA, Mead R (1965) A simplex method for function minimization. Comput J 7(4):308–313
23. Singer S, Singer S (1999) Complexity analysis of Nelder-Mead search iterations. Proc Appl Math Comput:185–196
24. Pham N, Wilamowski BM (2011) Improved Nelder Mead's simplex method and applications. J Comput 3(3):55–63
25. Gao F., Han L (2012) Implementing the Nelder-Mead simplex algorithm with adaptive parameters. Comput Optim Appl 51(1):259–277
26. Kirkpatrick S, Gelatt CD Jr, Vecchi MP (1983) Optimization by simulated annealing. Science 220:671–680
27. Dueck G, Scheuer T (1990) Threshold accepting: A general purpose optimization algorithm appearing superior to simulated annealing. J Comp Phys 90(1):161–175
28. Talbi E-G (2009) Metaheuristics. Wiley Publishing, From Design to Implementation
29. (2009) Particle swarm optimisation and high dimensional problem spaces. IEEE CEC:1988–1994