New Criteria for Comparing Global Stochastic Derivative-Free Optimization Algorithms

Jonathan McCart[1], Ahmad Almomani[2]
Department of Mathematics
State University of New York at Geneseo
Geneseo, New York 14454

***Abstract*—For many situations, the function that best models a situation or data set can have a derivative that may be difficult or impossible to find, leading to difficulties in obtaining information about the optimal values of the function. Thus, numerical methods for finding these important values without the direct involvement of the derivative have been developed, making the representation and interpretation of the results for these algorithms of importance to the researchers using them. This is the motivation to use and compare between derivative-free optimization (DFO) algorithms. The comparison methods developed in this paper were tested using three global solvers: Genetic Algorithm (GA), Particle Swarm Optimization (PSO), and Simulated Annealing (SA) on a set of 26 n-dimensional test problems of varying convexity, continuity, differentiability, separability, and modality. Each solver was run 100 times per problem at 2, 20, 50 and 100 dimensions. The formulation for each algorithm used comes from the MATLAB Optimization Toolbox, unedited or revised. New criteria for comparing DFO solver performance are introduced in terms defined as Speed, Accuracy, and Efficiency, taken at different levels of precision and dimensionality. The numerical results for these benchmark problems are analyzed using these methods.**

Keywords*—*Derivative-free optimization; algorithm comparison; test problem benchmarking.

I. Introduction

When deciding which algorithm is most appropriate to use on a given problem, it is crucial to have a detailed and encompassing description of how the algorithm will perform when applied in different contexts. Equally important is that the description of the algorithm is comprised of measures which can be universally applied to any other algorithm and thus, used as an objective basis for comparison between many algorithms. This paper aims to define a set of characteristics that begin to form this basis through providing information on the speed at which a solver completes a problem, the efficiency at which resources are used to determine solutions, and the accuracy/success rate of a solver on a problem. The testing each of these measures provided both intuitive and non-intuitive results.

Prior to this study, most of the benchmarking and performance information for GA, PSO and SA was either very specific to a limited number of circumstances or was only mentioned in passing while discussing another topic. The goal of this study is to expand upon what is known about the capabilities of these solvers, while using them as a means of applying and testing the comparison methods that are proposed in Section 3. The parameters we will be mainly focused on comparing across are dimension and problem type. These are two areas that substantially influence solver performance, which makes knowing the details on how exactly these areas impact performance important in the improvement of these solvers as well as the development of solvers with similar characteristics that are capable of circumventing the discovered weaknesses to these parameters.

This paper is meant as a preliminary study into what characteristics are necessary to describe DFO algorithm performance. Further study into what these characteristics may be is encouraged and could lead to a set of attributes beyond those initially proposed within this paper. We seek to provide useful measures of comparison specifically for derivative-free approaches to optimization because there is a wide variance in how derivative-free optimization is performed, which makes having a fair premise for comparison desirable.

A. Background

In [1], performance and data profiles are introduced and used on a set of three solvers applied to three problem types: smooth, piecewise smooth and noisy. Performance profiles are given as a measure for determining the relative differences in the proportion of problems solvable by one solver compared to another. Data profiles provide an independent comparison of what proportion of a problem set can be solved within a given budget of simplex gradients. Performance and data profiles derive their utility from their ability to consolidate the results of many test problems and the subsequent solver performance data from a common comparison criteria. Most often this criteria will be the computational budget: the number of function evaluations or the CPU time required to run the iterations. The information provided in a data profile will inform as to which solver is best when comparing across the problem's computational expense, while performance profiles provide information that relates to a fixed budget relative comparison. Currently, the most prominent method for comparing derivative-free optimization algorithms is the use of performance and data profiles, as discussed in [2].

In [2], 22 solvers were tested on over 500 problems in which the results were compared using performance and data profiles. The overarching conclusion of their comparison was that the solvers TOMLAB/MULTIMIN, TOMLAB/GLC-CLUSTER, MCS, and TOMLAB/LGO performed the best on average for the given problems. These are packages that come from the TOMLAB software, which is a very powerful modeling environment software used for solving optimization problems; TOMLAB is implemented in MATLAB. These implementations take advantage of clustering techniques, scatter search algorithms, as well as Lipschitzian optimization techniques such as branch-and-bound and DIRECT algorithm.

In [3], the authors give measures for comparing algorithms by way of defining characteristics in terms of raw performance data. These comparative measures are used in addition to the criteria used in [1] on performance and data profiles. A background, as well as the motivations and complications with benchmarking optimization algorithms are given and thereafter used to provide an example of a method for how researchers ought to approach the benchmarking process. Important considerations for future papers on the comparison of solvers are raised such as having a large enough set of problems that are also of a wide enough variety. The information presented in the data profiles is used in conjunction with other methods of analysis to properly illustrate a fair and unambiguous representation of solver performance, which demonstrates a better-informed process to follow when comparing algorithms.

In [4] “No Free Lunch Theorems for Optimization", David H. Wolpert and William G. Macready propose the No Free Lunch (NFL) Theorems, which state that no single algorithm can be the overall best algorithm for all problem types. This implies that an optimization algorithm that is strong on one set of problems will show weaknesses in other types. The ideas of what it means for an algorithm to be well suited for a problem are also presented. This theorem is central to the further development of future algorithms, which can be enhanced and informed by the information provided by a strong set of comparison criteria.

II. Algorithms Tested

For the experiments, testing consisted of three derivative-free solvers that were global and stochastic in nature: Genetic Algorithm (GA), Particle Swarm Optimization (PSO), and Simulated Annealing (SA).

A. Genetic Algorithm

Genetic Algorithm operates based on the ideas of evolution, where the sample points in the search space are seen as members of a population. As the number of iterations increases, the reported values by each of the individuals improves towards the optimal solution, analogous to the survival of the “most fit" individual in a population. GA also incorporates techniques inspired by other biological processes such as cross-over and mutation via binary encoding and permutation encoding in order to diversify the values obtained in the hope of finding numbers with higher fitness values [5], [6]. The general form of this process happens in the following manner:

Genetic Algorithm

1. Values are taken from the search space.

2. Fitness values calculated.

3. Lower fitness values go through crossover and mutation and higher fitness values are stored for future mutation/crossover with other individuals of comparably high fitness values.

4. Repeat until convergence.

B. Particle Swarm Optimization

Particle Swarm Optimization utilizes the idea of swarm intelligence to use a set of search particles randomly dispersed around the search space to improve the best iterate value by comparison between each particle’s value. PSO is built on vector equations of each particle’s position and velocity in the space, and these parameters are adjusted and updated after each iteration, relative to the best value found during that iteration. The values obtained by each individual particle in the space are considered their personal best for the given run, and the best value found by any particle during any iteration is defined as the global best value [7], [8], [9]. All particles update their position (x) and velocity (v) equations according to the global best. These equations are given by:

$$x^i_{k+1} = x^i_k + v^i_{k+1}$$

$$v^i_{k+1} = wv^i_k + c_1r_1(p^i_k - x^i_k) + c_2r_2(p^g_k - x^i_k)$$

Where:

x^i_k = the current position of particle i at iteration k.

x^i_{k+1} = the position of an individual particle for the subsequent iteration

v^i_k = the current velocity of particle i at iteration k.

v^i_{k+1} = the velocity of the particle for the subsequent iteration

p^i_k = the personal best value achieved by particle i

p^g_k = the best value achieved for any particle so far

c_1, c_2 = cognitive and social parameters

r_1, r_2 = random numbers between 0 and 1

w = the inertia weight

All of these components come together in ways that distinguish PSO in its approach to solving global optimization problems. The term in the velocity equation wv^i_k is the inertia term, which gives weighting to the current values obtained by the particle to resist extreme changes based on each individual random value obtained during the search. The term $c_1r_1(p^i_k - x^i_k)$ is the cognitive term, which is the personal supervising term that weights the data of the individual particles prior to inter-particle comparison as well as prior to updating the iterate value to ensure improvement. The term given by $c_2r_2(p^g_k - x^i_k)$ is the social term, which permits the influence of data returned by each particle to change the behavior of the swarm. This process of updating vector equations based on values gathered from the search space continues while other particles in the swarm continue traversing the space pseudorandomly to discover new points in an effort to avoid trapping with local minima. This continues until convergence on the solution, with parameter functionality further described in [10]. The general form of implementation for PSO typically runs in the following way:

Particle Swarm Optimization

1. Define and Initialize variables and particles.

2. Particles take random values from the space.

3. If value x improves PBest, set PBest = x, otherwise continue searching.

4. Update velocity and position vectors for the PBest of each particle relative to GBest.

5. Repeat until convergence of GBest to minimum.

C. Simulated Annealing

Simulated Annealing is an adaptation of a Monte Carlo Method [11] for solving global optimization problems, accomplished by simulating a thermodynamic system. In SA, points are randomly sampled from the search space according to the probability of a point improving the state of the system where points are seen as sample states of the thermodynamic system. A cooling schedule associated with the “temperatures" in the space tends to zero as the algorithm converges to a solution: lower values obtained progress this schedule as they represent lower energy states in the system [12], [13]. This is a probabilistic solver that runs according to the following general form:

Simulated Annealing

1. Select a data point randomly from the search space.

2. Calculate the the cost difference between current and prospective point.

3. If $cost_{new} < cost_{current}$, accept new.

4. If $cost_{new} > cost_{current}$, accept, but modify the probability of selecting future points that do not improve $cost_{current}$.

5. Repeat until Temperature (T) $\rightarrow$ 0 (convergence).

III. Attributes and Criteria for Comparison

In order to provide an objective basis for comparing global optimization algorithms, a delineation must be made between the aspects of the solution method that are innate to the solver’s metaheuristic process and the aspects that are general to all solvers. The aspects of the solution methods that are similar between the solvers will come through in the analysis of performance trends via comparing algorithms relatively, as well as tracking the changes in performance over problem type and dimensionality.

An example of what it means to investigate the qualities of algorithms can be seen by examining the differences between PSO and SA. Note that although both PSO and SA are global stochastic solvers, each is different in how it approaches a given problem. The main differences between PSO and SA are in their search methods. PSO is a population-based solver that uses multiple particles in communication with one another to collectively find the solution, whereas SA is a single point search algorithm that uses a probability function to assess points ability to improve a thermodynamic system. The idea being investigated here is whether there are descriptions of solver behavior that allow for comparing different solvers through the ways in which the functionalities of the algorithms are fundamentally the same. Section 3 discusses attributes that begin the formation of a basis for comparing these different approaches to DFO. Some algorithms may prove to be stronger in a particular attribute than others, and this difference stands to differentiate between where each algorithm can be effectively used in application, which would agree with [4]. In trying to determine the effectiveness of these new metrics for ascertaining key information on solver performance, the best indication that a metric is well defined and capable of providing useful information is if both a clear relationship is demonstrated and the relationship is consistent with the observations in the raw data. The metrics below are defined in terms of the resulting data obtained from running the solver on any problem.

Speed

Speed is defined by the following ratio:

$$\text{Speed} = \frac{\text{Average Number of Function Evaluations}}{\text{Average CPU time}}. \quad (1)$$

This ratio indicates the relative expense of an evaluation of the algorithm, expressed as related to CPU time, which is a known indicator of the computational expense of the algorithm [3]. Speed can be used to see how factors such as dimensionality and problem type affect the computational expense of a run of the solver, which has implications about how well an algorithm would scale with dimension.

Accuracy

The Accuracy of a solver is defined as the proportion of runs in which the global minimum was successfully found within a tolerance of ϵ. Here, ϵ represents the difference between the best value found by the solver and the optimal value, and was taken at ϵ = 10^{0}, ϵ = 10^{-2}, ϵ = 10^{-5} and ϵ = 10^{-10}. The success rate graphs represent the average proportion of successful runs for each solver on the total number of problems within a given group of problems. Accuracy will be represented by the variable A_ϵ, meaning the accuracy of a solver at the corresponding tolerance level ϵ.

Efficiency

The Efficiency of a solver is designed to give an indication as to how well a solver is performing across areas of Speed, Accuracy, ability to achieve the global minimum in as few runs as possible, and also the ability to obtain the minimum in a low number of runs on average:

$$\text{Efficiency} = M * \chi * S * A_\epsilon \quad (2)$$

Where :

$$M = \frac{\log_{10}(M^*)}{\log_{10}(M_i)}$$

$$\chi = \frac{\log_{10}(\chi^*)}{\log_{10}(\chi_i)}$$

$$S = \frac{\log_{10}(S_i)}{\log_{10}(S^*)}$$

Defined by:

A_ϵ = the accuracy value for the given solver within ϵ.

M_i = the mean number of evaluations taken by solver i for a given problem.

M^* = the lowest mean number of evaluations taken by any solver for a given problem.

χ_i = the mean lowest number of evaluations by solver i for a given problem.

χ^* =the absolute lowest number of evaluations by any solver for a given problem.

S_i = the speed of solver i for a given problem.

S^* =the speed of the fastest solver for a given problem.

For each of these three areas (M, χ, S) in which a algorithm performs the best, it will have a corresponding value of 1 and therefore will not receive any loss in Efficiency. However, the Efficiency values of the subsequently lower performing algorithms will be dampened by the M, χ, and S factors. Thus if a solver is the top performer, its Efficiency will equal its Accuracy. Efficiency is only calculated using runs in which the global minimum was obtained. If a solver had runs in which the global minimum was not obtained, the Efficiency of that solver was set to 0 for those runs.

Difficulty

When any solver is applied to a problem requiring a high accuracy, we expect difficulties for the solver resulting in more CPU time or more function evaluations. The chance of failure is very high for achieving a highly accurate solution. Difficulty is given by the following:

$$\text{Difficulty} = -ln(1 - \text{failure}) = -ln(\text{success}) \qquad (3)$$

This is the metric used in [14], which was used to measure the performance of PSO.

These metrics provide more detail in the information obtained on solver performance, as to highlight more specific areas in which a given algorithm may see strength or issue. Contrasting this with the information obtained through the use of performance and data profiles, the purpose of the results discussed in this paper are not intended to give a direct overview of the solver's capabilities with respect to some constraint such as computational budget, but rather to look into more detailed questions within the context of the whole performance. This is done by defining sub-categories of performance (Accuracy, Speed and Efficiency) which can assess the effort required on the part of the solver to obtain the optimal value. This then gives rise to a more generalized form of obtaining and representing the different levels of performance and success within these shared areas of aptitude.

IV. Numerical Experiments

All numerical experiments for GA, PSO, and SA were run on an iMac, version 10.13.6, Processor 4.2 GHz Intel Core i7, 64 GB 2400 MHz DDR4 memory. The implementation of each algorithm used the default settings in the MATLAB Optimization Toolbox [15], with the stopping criteria for all being within the tolerance ϵ = 10^{-10}. The budget here is 10^6 function evaluations.

Limitations

In an effort to maintain the reproducibility of this experiment, no implementations of GA, PSO or SA from outside of MATLAB were used. There are many adaptations of GA [16], many hybrids of PSO with other optimization techniques [17], [18], [19], and popular variations of SA [20] seen across different fields such as engineering and machine learning with these algorithms producing competitive results. These solvers and their advantages are not discussed within the scope of this paper. Given that this paper seeks to discuss manners of comparison and benchmarking along with the representations for these metrics, the algorithms used need only function as examples to demonstrate a comparison process.

Problem Types

In order to see the differences in performance across a variety of problem types as well as a range of dimensions, it is important to classify the set of test problems into 4 groups of the following types:

G1: non-convex, continuous, differentiable, non-separable, multimodal.

G2: convex, non-differentiable, continuous, separable, unimodal.

G3: non-convex, non-differentiable, non-separable, multimodal.

G4: All of the listed types.

Groups

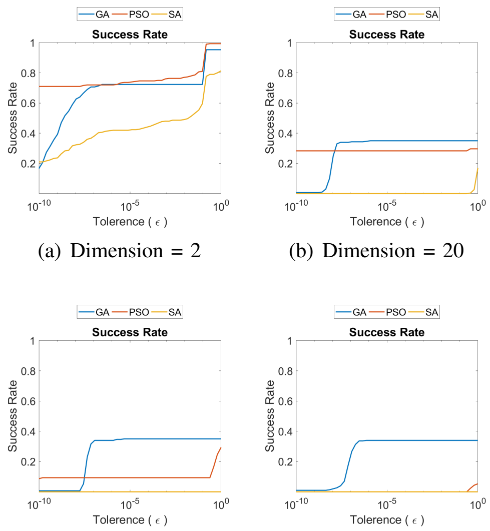
Problems within G1, G2, G3 and G4

G1	**G4**
alpinen2fcn	ackleyfcn
periodicfcn	alpinen1fcn
shubert4fcn	alpinen2fcn
	periodicfcn
G2	rosenbrockfcn
powellsumfcn	salomonfcn
schwefel220fcn	shubertfcn
schwefel223fcn	shubert3fcn
	shubert4fcn
	xinsheyangn3fcn
G3	powellsumfcn
schwefelfcn	schwefel220fcn
xinsheyangn2fcn	schwefel221fcn
xinsheyangn4fcn	schwefel222fcn
	schwefel223fcn
	exponentialfcn
	sumsquaresfcn
	xinsheyangn1fcn
	xinsheyangn2fcn
	xinsheyangn4fcn
	griewankfcn
	quarticfcn
	rastriginfcn
	styblinskitankfcn
	zakharovfcn

V. Results

A. Accuracy

In Fig. 1, all solvers display competitive performance at 2 dimensions. Yet as the dimension increased, each solver's performance decreased at different rates with GA having decreased the least among all solvers leaving GA capable of solving at least one of the problems in G1 within 10^{-7} for all dimensions. GA showed a gradual reduction in success from 20 to 100 dimensions, coming through as the algorithm which was most robust to changes in dimensionality for G1. For the problems in G1, PSO observed the most dramatic change in performance, demonstrating higher sensitivity to changes in dimensionality. SA dropped to near zero at just 20 dimensions, showing SA to be the least competitive on this set.

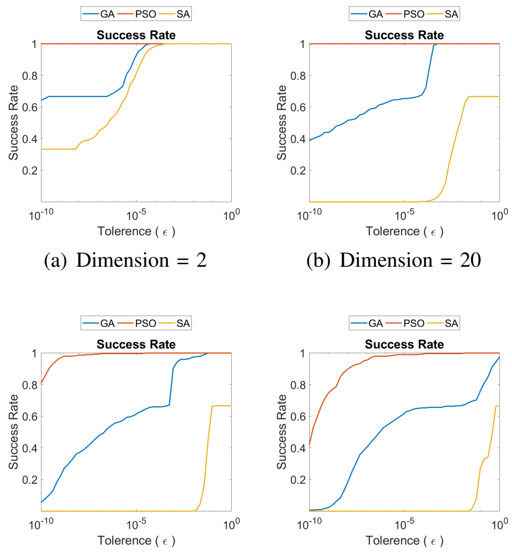


(a) Dimension = 2 (b) Dimension = 20

(c) Dimension = 50 (d) Dimension = 100

Fig. 1. Accuracy for G1

All solvers performed significantly better on G2 than on G1. This indicates that the differences in problem type had a sizable impact on performance, despite the problems of G2 being non-differentiable. Fig. 2 shows that the problems of G2 remained 100% solvable by all solvers at 2 dimensions for $\epsilon > 10^{-3}$. All solvers showed a decrease in success rate as ϵ decreased. In comparing the performance of PSO on G2 to G1, unlike in G1, there is less sensitivity to dimensionality within G2. Thus it seems likely that the characteristics of the problems in G2 are more favorable for PSO's heuristic. GA once again shows a resistance to a loss in success from 2 to 100 dimensions, yet does not outperform PSO on this set due to the amount by which PSO's performance increased from G1. SA is once again approaching a success rate of zero for most observed values of ϵ below about 10^{-2} for dimensions greater than 20, which makes SA a less competitive choice on a problem set similar to G2.

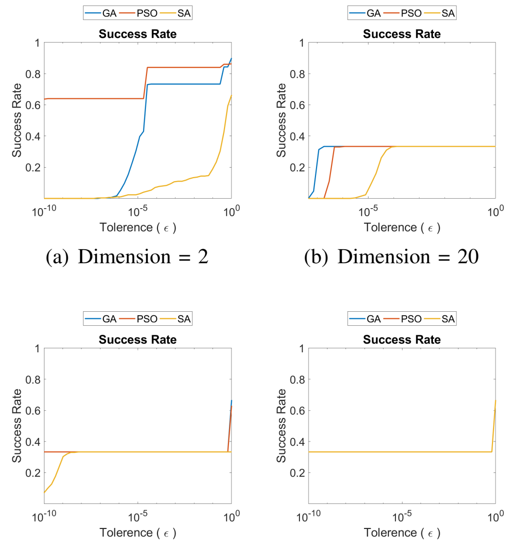


(a) Dimension = 2 (b) Dimension = 20

(c) Dimension = 50 (d) Dimension = 100

Fig. 2. Accuracy for G2

The results of G3 were quite unintuitive. The behavior of the solvers was not strongly correlated with dimension in the same way as in G1 and in G2. Behavior for this group was much more problem dependent, where in Fig. 3 going from 2 to 100 dimensions, each solver is shown to have had one problem they remained capable of solving, with an *increase* in the success rate of each algorithm as dimensionality increased and as ϵ decreased. Note that each algorithm achieved a success rate of 33% for G3, meaning that each solver failed two of the three problems every time, but succeeded on the remaining problem each time. There is no clear explanation for this behavior, making it a point of interest for further investigation.

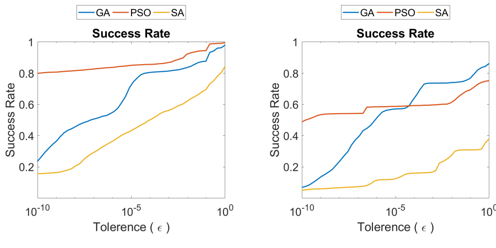


(a) Dimension = 2 (b) Dimension = 20

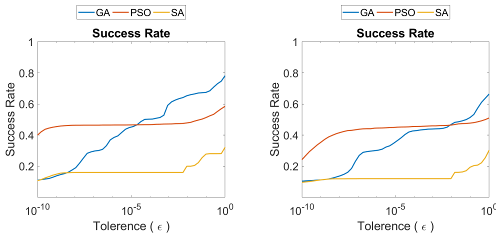
(c) Dimension = 50 (d) Dimension = 100

Fig. 3. Accuracy for G3

Fig. 4 shows a decreasing trend in solver ability to obtain the global minimum for small ϵ as well as for high dimensionality. We take G4 as a strong indication as to how solver performance will average out across a variety of problems of different or mixed type, showing the expected trends that would arise from general use of any of these algorithms. Algorithms which tend to have a shallow slope display a high level of consistency across the levels of ϵ. Notice that although PSO maintains this consistency, its Accuracy drops as dimension increases. Solvers like SA that cannot resist a loss in Accuracy for higher dimensions and higher ϵ will see their success rate go to zero much quicker than other algorithms. GA has a sharper slope than PSO or SA, but does not translate as far down the graph when dimension increases. This trend in the amount to which success rates decrease shows GA to be the most robust for ϵ level (10^{-4}, 10^{-4}, 10^{-2}) for dimensions 20, 50 and 100 respectively. PSO, however, is able to achieve the highest success rate for small ϵ as opposed to GA or SA.



(a) Dimension = 2 (b) Dimension = 20



(c) Dimension = 50 (d) Dimension = 100

Fig. 4. Accuracy for G4

Accuracy proved to be a very effective tool in analyzing the performance of the solvers in terms of their reliability for finding the global minimum on problems of varying type and difficulty. There was a large spread in performance across the groups, suggesting that Accuracy is able to identify changes in the solution capability of the solvers on these types of problems. Accuracy also showed that problem type greatly affected solver ability to produce highly accurate values for the minimum.

B. Speed

The trends observed in Speed for each group yielded similar results, including G3. When looking to Fig. 5, the graph of Speed for G4, a clear ordering from the fastest to slowest solver emerges and remains consistent from 2 to 100 dimensions. However, the response to the increase in dimension was different for each solver:

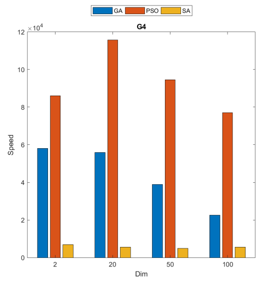
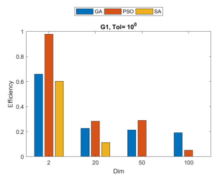


Fig. 5. Average Speed for G4

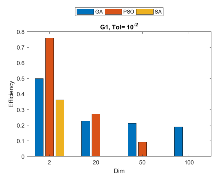
GA was virtually unaffected by dimensionality until after 20 dimensions where there is then a steady decrease going to 100 dimensions. For SA, there seemed to be little effect on Speed as a result of increasing dimension, however SA was consistently the slowest of the three. PSO's behavior between 2 and 50 dimensions was unanticipated. There was an increase in Speed from 2 to 20 dimensions, prior to an eventual and steady decline moving towards 100 dimensions. One possible account for this is a difference in the rate of change in evaluations taken by the solver compared to CPU time needed to take those evaluations of the function at different dimensions, meaning that the rate at which the number of evaluations grew with dimension outpaced the increase in CPU time taken until after 20 dimensions. The differences in the relationships of each solver's Speed value to the dimensionality of the problem shows how the strengths of the approach by each algorithm affect the use computational resources. Knowing how each algorithm uses these resources is important for assessing the utility of an algorithm for a task, especially one requiring performance that is both computationally inexpensive and fast.

C. Efficiency

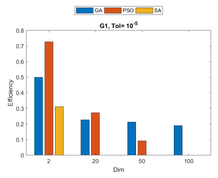
The Efficiencies for G1 show that the characteristics of the problems posed a significant challenge for the solvers, despite the fact that the problems in G1 were differentiable. This indicates a strong relationship between problem type and general difficulty, further supported in Fig. 10 and also in the comparison of Fig. 6 and Fig. 7. GA is shown to be not as efficient as PSO when ϵ decreases, but GA remains the most efficient for higher dimensional problems. This further supports the claim that GA has a resistance to loss in Accuracy for higher dimensional problems. Note that solvers that had no runs in which the global was obtained within ϵ have Efficiencies of 0. Overall, it seems that the performance for all solvers dropped significantly with dimension, which corroborates the results from Accuracy.



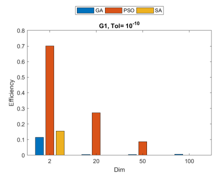
(a) $\epsilon = 10^0$



(b) $\epsilon = 10^{-2}$



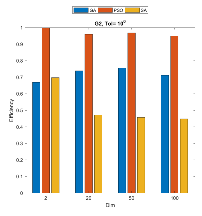
(c) $\epsilon = 10^{-5}$



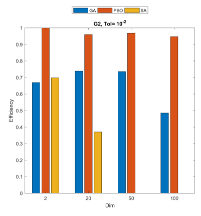
(d) $\epsilon = 10^{-10}$

Fig. 6. Efficiencies for G1

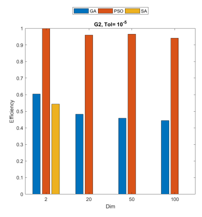
As opposed to G1, the characteristics of the problems in G2 were significantly easier for the solvers, despite being non-differentiable problems. Each algorithm performed noticeably better across dimension and tolerance. PSO is the most sensitive to the problem characteristics, as its performance across different groups displays a large variance. This sensitivity, however, has shown to benefit PSO, as PSO had the highest Efficiency value at every tolerance and dimension. GA was robust to changes in tolerance across each group, while SA was most affected by tolerance, as one may notice that the Efficiency value for SA dropped to zero for 50 and 100 dimensions after an increase in the tolerance from $\epsilon = 10^0$ to $\epsilon = 10^{-2}$. As the Accuracy plots for G3 suggested, there were some problems for which the solvers saw a slight increase in performance as dimension increased. Although not as extreme as in G3, subtle increases in Efficiency for GA can be seen across each dimension at $\epsilon = 10^0$ to $\epsilon = 10^{-2}$ in G2.



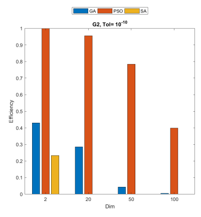
(a) $\epsilon = 10^0$



(b) $\epsilon = 10^{-2}$



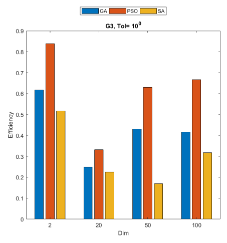
(c) $\epsilon = 10^{-5}$



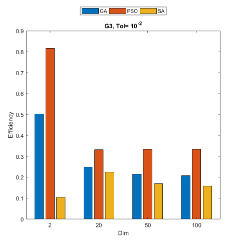
(d) $\epsilon = 10^{-10}$

Fig. 7. Efficiencies for G2

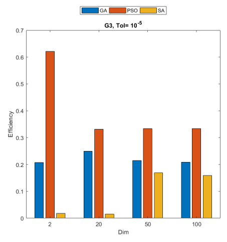
G3 exhibited trends in performance that did not follow the implication of the other groups. The Efficiency values for G3 did *not* consistently decrease with dimension. Instead, Fig. 8 shows a drop in performance from 2 to 20 dimensions, which becomes more and more magnified as ϵ decreases, followed by an increase in performance from 20 to 100 dimensions. Further investigation into this group shows that after 25 dimensions, the decreasing trend in performance turns around and the solvers show a resurgence in Efficiency for higher dimensions. This implies that there are qualities of these problems that elicit a difference in solver approach from within the algorithms themselves. The results for Efficiency of G3 also corroborate what was observed in the results for the Accuracy of G3. However, unlike in the Accuracy for G3, we do not see the same equilibrium phenomena where each solver approaches solving one out of three problems each time, but rather we see a score of zero for all solvers at 20 dimensions for $\epsilon = 10^{-10}$. This means that there were no values for either M_i, χ_i, or S_i for any solver at this level, and thus at 20 dimensions the Efficiency values for each solver were assigned a value of 0. The counter-intuitive behavior of the solvers on the problems within G3 will be the subject of further research, as understanding why this happens can more broadly and completely describe solver characteristics.



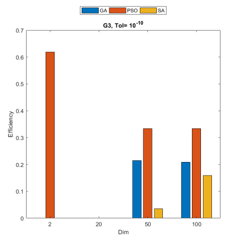
(a) $\epsilon = 10^0$



(b) $\epsilon = 10^{-2}$



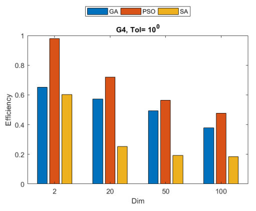
(c) $\epsilon = 10^{-5}$



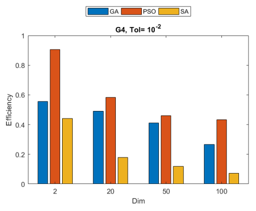
(d) $\epsilon = 10^{-10}$

Fig. 8. Efficiencies for G3

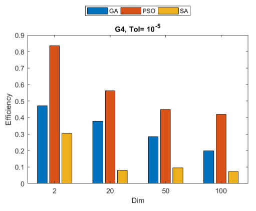
Just as we examined the Accuracy plots for G4 to gain insight into the average solution capability of each algorithm, we now look to Efficiency values for G4 shown in Fig. 9 to get an indication as to how well the successful attempts at finding the optimal were completed. PSO shows the strongest resistance to loss in Efficiency on solving problems in G4 at low dimensions, while the decreasing trend in performance as dimension increases shows a sharper decline as tolerance increases. GA, however, displays a rather consistent decreasing trend with a shallow change between dimensions. In terms of Efficiency, PSO surmounts GA and SA for all dimensions and tolerances in the long run. Thus, PSO is likely a good choice for when the problem set is mixed.



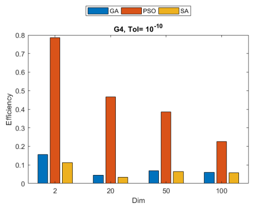
(a) ϵ = 10^0



(b) ϵ = 10^{-2}



(c) ϵ = 10^{-5}



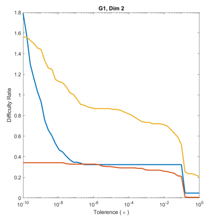
(d) ϵ = 10^{-10}

Fig. 9. Efficiencies for G4

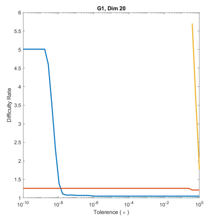
Efficiency proved to be able to effectively measure the quality of algorithm performance across each parameter we sought to measure. The formulations for the factors of M, χ, and S give proper weighting to each solver's level of ability and balanced for factors that Accuracy alone could not. The consolidation of the information regarding the magnitudes of the best values obtained by a solver into Efficiency makes a solver's Efficiency capable of giving weight to the relative sizes and differences these values in a way that the values themselves do not convey alone, making Efficiency a valuable tool for starting to look into trends within a solvers own performance across dimension and tolerance.

D. Difficulty

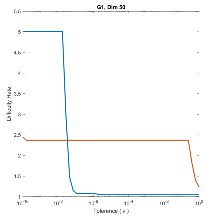
The Difficulty values for G1 displayed in Fig. 10 corroborate the comparison of the Efficiency values seen in G1 and G2, showing that the problems in G1 posed a greater challenge to the solvers than did the problems of G2. This further substantiates the idea that it is likely that attributes outside of differentiability must also be considered in determining which solver to choose for a problem set. As it pertains to dimensionality, GA manages the change in dimensions the best, being roughly consistent in its Difficulty rating. PSO's Difficulty rapidly increases from 50 to 100 dimensions, showing yet again a sensitivity to dimension.



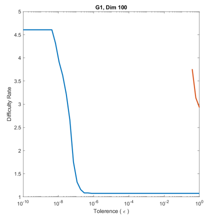
(a) Dimension = 2



(b) Dimension = 20



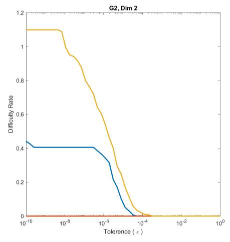
(c) Dimension = 50



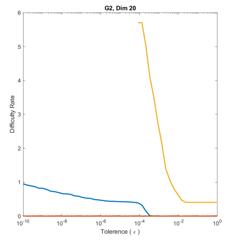
(d) Dimension = 100

Fig. 10. Difficulties for G1

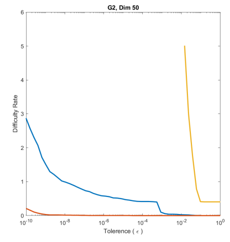
The results shown in Fig. 11 also agree with the comparison of the Efficiency values for G1 and G2, but the Difficulty values for G2 give more information about relative solver performance. PSO remains the best solver in keeping the lowest difficulty for all dimensions. This indicates that although there is a demonstrated sensitivity for PSO to dimension, it seems quite possible based on the G2 difficulty graphs that given the optimal set of problem characteristics, PSO's increase in performance for these types of problems outweigh the effects of dimensionality. GA does no share the same strength in awareness to problem type, as its performance is similar to G1. SA shows some improvement on G2, yet still struggled as dimension increased.



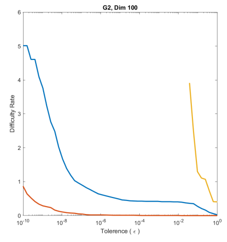
(a) Dimension = 2



(b) Dimension = 20



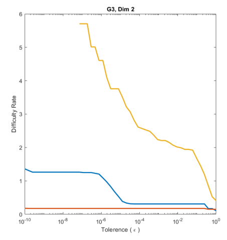
(c) Dimension = 50



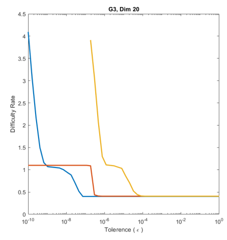
(d) Dimension = 100

Fig. 11. Difficulties for G2

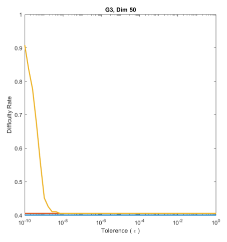
The Difficulty values for G3 shown in Fig. 12 display a similarly odd trend to what we saw in Fig. 3 and Fig. 8. We see that Difficulty *decreases* for higher dimensions after a spike at 20 dimensions. The reasons for this remain unclear. The entire set was run multiple times and similar results were seen. It is possible that problems of the same type as G3 would elicit similar behavior from the solvers that is not only atypical of the general trends we have observed, but also does not fit into what we saw or expected for PSO's sensitivity to dimension or GA's consistency in performance. The tendency to move towards an equilibrium value is seen in the Difficulty of G3 as was seen in the Accuracy for G3.



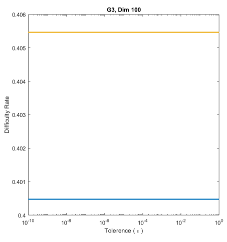
(a) Dimension = 2



(b) Dimension = 20



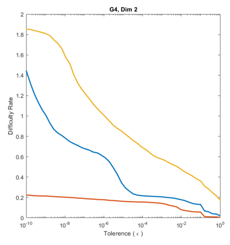
(c) Dimension = 50



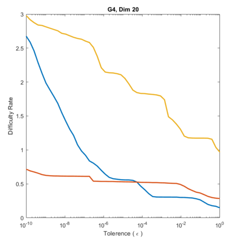
(d) Dimension = 100

Fig. 12. Difficulties for G3

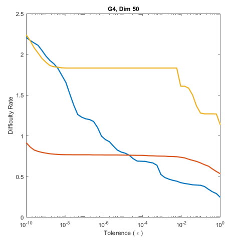
The Difficulty values for G4 seen in Fig. 13 show that, overall, the trends in Difficulty for solvers maintained a consistent placement of the solvers relative to one another. Note also that the expected trend that Difficulty increases with dimension. Seeing that all solvers did well on G4 gives an indication that certain problems may have attributes that worked well with each algorithms heuristic, which improved their Difficulty rating overall. SA's improvement on these types of problems could mean that the particular attributes of the problems in G1 and G2 were especially challenging for SA compared to other problem types.



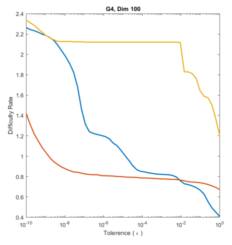
(a) Dimension = 2



(b) Dimension = 20



(c) Dimension = 50



(d) Dimension = 100

Fig. 13. Difficulties for G4

Difficulty supports the proposed methods for identifying pertinent trends in performance that can be objectively measured between solvers. Difficulty also allows for more depth in searching for explanations of trends seen in Efficiency, Accuracy and Speed that may not be superficially obvious, such as the relatively small differences between GA and SA in Difficulty seen in Fig. 13, despite the larger differences in Accuracy and Efficiency. This indicates that the work required for GA to obtain optimal values led more often to success than that of SA. This demonstrates even further that looking within the details of benchmarking methods is crucial to fully describing solver performance, and that the decision between applying one solver versus another can be more informed by the knowledge of what exactly goes into solving each problem for each solver.

VI. Conclusions

It was seen in the analysis of each solver's performance across G1, G2, G3 and G4 that problem type and dimension play a key role in determining which algorithm will be most successful in a given context. There were a variety of outcomes among the solvers as to how often the global minimum could be obtained, as well as how precise that value could be. In looking across each group, with the exception of G3, it is shown that the data for each characteristic corroborated the trends displayed in one another. Describing algorithmic behavior in terms of Accuracy, Efficiency and Speed demonstrates the strengths and weaknesses of these algorithms in a way that formalizes performance into a comprehensive qualitative and quantitative description.

One of the main strengths that GA exhibited on the groups was a resistance to changes in dimension, making GA a strong choice for higher dimensional problems. Also, GA maintained a competitive Accuracy for large ϵ, outperforming PSO in some cases, as seen in Fig. 1 and Fig. 4. A weakness observed for GA was a sensitivity to tolerance, meaning that as ϵ

decreased, GA observed significant drops in its attribute values. Note also that GA had a rather large average number of evaluations, which in turn reduced GA's Speed.

PSO was, for the most part, the strongest algorithm on each group. PSO showed not only an impressive resistance to loss in Accuracy across tolerance, but also consistently had the highest Speed value of all solvers. PSO's consistency across tolerance makes it a strong choice for problems requiring knowledge of the minimum to within small ϵ. PSO would also be the best choice for low dimensional problems, as PSO enjoys a high success rate for most problems below 20 dimensions. PSO's performance was only significantly affected during changes in dimension. For although the minimum found by PSO is highly accurate and precise, PSO still saw a large drop in Accuracy as dimension increased, seen specifically in Fig. 1, were PSO translated down the graph for each observed dimension. It must be said, however, that this weakness for PSO that was also seen in Fig. 2 that PSO's sensitivity to dimension had very little, if any, negative effects on PSO's performance. This suggests that PSO has a greater sensitivity to problem type than to dimension, and PSO is more problem conscious than the other solvers, something that both helped and hurt PSO's performance during these experiments.

The performance seen for SA was the least competitive of the three solvers. SA demonstrated an extreme sensitivity to dimension, dropping in Accuracy as much as from about 80% to less than 20% for G1 going from 2 to 20 dimensions in Fig. 1. SA also had the lowest Speed for all solvers, which was on average less than about 15% the Speed of PSO and even less than about 20% that of GA, as seen in Fig. 5. Potential strengths for SA may lie in the further analysis of G3, as this is the only area in which SA observed competitive performance. SA also never observed a difficulty below 1 outside of G3, indicating that even the performance seen on G1, G2 and G4 was a struggle. This indicates that problem type has perhaps the biggest effect on SA.

The characteristics of Speed, Accuracy and Efficiency provide useful information about solver performance and give a clear description of solver capabilities within these areas. Certain problems produced solver behavior that defied intuition and expected trends in the data, which shows the current descriptions of solver behavior to be, although useful, incomplete. This emphasizes the need for further research into more criteria which can help describe such phenomena as seen in G3.

VII. Closing Remarks

These experiments were an attempt to begin the formalization of solver performance comparison metrics into a concise and usable format. A formalized set of criteria will hopefully establish a medium for communication between researchers where coherence and consistency in representation of performance will be established. Further testing and study of these criteria is encouraged so that the definitions may improve and new criteria may emerge that will be universally beneficial. The characteristics proposed in this paper were defined using stochastic, global algorithms. Results regarding deterministic solvers and/or local solvers are not observed in this paper and remain a topic for future research. Future research endeavors may also include seeing the effects of parallelizing these algorithms for use on a GPU so that more problems can be run at even higher dimensions, as well as more solvers, as to increase the variety of algorithms considered, including deterministic and local solvers. Looking further into G3 is also a topic of importance as it may lead to progress in determining the cause for the observed behavior, possibly leading also into establishing more characteristics which can help broaden the description of algorithmic trends.

Acknowledgment

This work was partially supported by funds made available under a State University of New York Expanded Investment and Performance Award to the State University of New York at Geneseo.

References

[1] J. J. Moré and S. M. Wild, "Benchmarking derivative-free optimization algorithms," *SIAM J. on Optimization*, vol. 20, no. 1, pp. 172–191, Mar. 2009. [Online]. Available: http://dx.doi.org/10.1137/080724083

[2] L. M. Rios and N. V. Sahinidis, "Derivative-free optimization: a review of algorithms and comparison of software implementations," *Journal of Global Optimization*, vol. 56, no. 3, pp. 1247–1293, Jul 2013. [Online]. Available: https://doi.org/10.1007/s10898-012-9951-y

[3] V. Beiranvand, W. Hare, and Y. Lucet, "Best practices for comparing optimization algorithms," *Optimization and Engineering*, vol. 18, no. 4, pp. 815–848, Dec 2017. [Online]. Available: https://doi.org/10.1007/s11081-017-9366-1

[4] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *Trans. Evol. Comp*, vol. 1, no. 1, pp. 67–82, Apr. 1997. [Online]. Available: https://doi.org/10.1109/4235.585893

[5] S. N. Sivanandam and S. N. Deepa, *Introduction to Genetic Algorithms*, 1st ed. Springer Publishing Company, Incorporated, 2007.

[6] P. A. Estévez, R. C. Eberhart, and Y. Shi., "Computational intelligence: Concepts to implementation," *Genetic Programming and Evolvable Machines*, vol. 9, no. 4, pp. 367–369, Dec 2008. [Online]. Available: https://doi.org/10.1007/s10710-008-9064-z

[7] A. P. Engelbrecht, *Fundamentals of Computational Swarm Intelligence*. USA: John Wiley &#38; Sons, Inc., 2006.

[8] T. M. Blackwell, "Particle swarms and population diversity," *Soft Computing*, vol. 9, no. 11, pp. 793–802, Nov 2005. [Online]. Available: https://doi.org/10.1007/s00500-004-0420-5

[9] J. E. Hicken and J. J. Alonso, "Introduction to multidisciplinary design optimization, chapter 6: Gradient-free optimization," 2012. [Online]. Available: http://adl.stanford.edu/aa222/Home.html,Accessed: 2019-04-25

[10] A. P. Engelbrecht, *Computational Intelligence: An Introduction*, 2nd ed. Wiley Publishing, 2007.

[11] R. W. Shonkwiler and F. Mendivil, *Explorations in Monte Carlo Methods*, 1st ed. Springer Publishing Company, Incorporated, 2009.

[12] J. Heaton, *Introduction to Neural Networks for Java, 2Nd Edition*, 2nd ed. Heaton Research, Inc., 2008.

[13] L. Ingber, "Simulated annealing: Practice versus theory," *Mathl. Comput. Modelling*, vol. 18, pp. 29–57, 1993.

[14] M. Clerc, *Particle Swarm Optimization*. ISTE, 2006.

[15] "Matlab," MATLAB2017b, The MathWorks, Natick, MA, USA.

[16] F. P. Such, V. Madhavan, E. Conti, J. Lehman, K. O. Stanley, and J. Clune, "Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning," *CoRR*, vol. abs/1712.06567, 2017. [Online]. Available: http://arxiv.org/abs/1712.06567

[17] S. Li, M. Tan, I. W.-H. Tsang, and J. T. Kwok, "A hybrid pso-bfgs strategy for global optimization of multimodal functions," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 41, pp. 1003–1014, 2011.

[18] W.-J. Zhang and X.-F. Xie, "Depso: Hybrid particle swarm with differential evolution operator," 2003.

[19] S. Sengupta, S. Basak, and R. A. Peters, "Particle swarm optimization: A survey of historical and recent developments with hybridization perspectives," *ArXiv*, vol. abs/1804.05319, 2018.

[20] H. Djidjev, G. Chapuis, G. Hahn, and G. Rizk, "Efficient combinatorial optimization using quantum annealing," *arXiv*, 01 2018.

[21] B. Functions, "http://benchmarkfcns.xyz/fcns, accessed: 2018-06-23," accessed: 2018-06-23.

Test Problems

All the test functions [21] were sorted by the following characteristics:

- S = separable
- S' = not separable
- D = differentiable
- D' = non-differentiable
- C = convex
- C' = non-convex
- U = unimodal
- M = multimodal
- c = continuous
- c' = not continuous
- n = dimension

*xinsheyangn3fcn minimum occurs for m = 5 and β = 15

Function	Characteristics	Range	Optimal	Formulation
ackleyfcn	C′ c D S′ M	[-32,32]	0	$-a \cdot exp(-b\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}) - exp(\frac{1}{n}\sum_{i=1}^{n} cos(cx_i)) + a + exp(1)$
alpinen1fcn	“ "	[0,10]	0	$\sum_{i=1}^{n} \lvert x_i sin(x_i) + 0.1x_i \rvert$
alpinen2fcn	“ "	[0,10]	2.808^n	$\prod_{i=1}^{n} \sqrt{x_i} sin(x_i)$
periodicfcn	“ "	[-10,10]	0.9	1 + $\sum_{i=1}^{n} sin^2(x_i) - 0.1exp(\sum_{i=1}^{n} x_i^2)$
rosenbrockfcn	“ "	[-5,10]	0	$\sum_{i=1}^{n} [b(x_{i+1} - x_i^2)^2 + (a - x_i)^2]$
salomonfcn	“ "	[-100,100]	0	1-$cos(2\pi\sqrt{\sum_{i=1}^{D} x_i^2}) + 0.1\sqrt{\sum_{i=1}^{D} x_i^2}$
shubertfcn	“ "	[-10,10]	-186.73	$\prod_{i=1}^{n}\left(\sum_{j=1}^{5} cos((j+1)x_i + j)\right)$
shubert3fcn	“ "	[-10,10]	-29.673	$\sum_{i=1}^{n}\sum_{j=1}^{5} j sin((j+1)x_i + j)$
shubert4fcn	“ "	[-10,10]	-25.7408	$\sum_{i=1}^{n}\sum_{j=1}^{5} j cos((j+1)x_i + j)$
xinsheyangn3fcn	“ "	$[-2\pi, 2\pi]$	-1 *	$exp(-\sum_{i=1}^{n}(x_i/\beta)^{2m})$ - 2$exp($-$\sum_{i=1}^{n} x_i^2) \prod_{i=1}^{n} cos^2(x_i)$
powellsumfcn	C c D′ S U	[-1,1]	0	$\sum_{i=1}^{n} \lvert x_i \rvert^{i+1}$
schwefel220fcn	“ "	[-100,100]	0	$\sum_{i=1}^{n} \lvert x_i \rvert$
schwefel221fcn	“ "	[-100,100]	0	$\max_{i=1,\dots,n} \lvert x_i \rvert$
schwefel222fcn	“ "	[-100,100]	0	$\sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$
schwefel223fcn	“ "	[-10,10]	0	$\sum_{i=1}^{n} x_i^{10}$
exponentialfcn	C c D S′ U	[-1,1]	0	$-exp(-0.5\sum_{i=1}^{n} x_i^2)$
sumsquaresfcn	C c D S U	[-10,10]	0	$\sum_{i=1}^{n} ix_i^2$
xinsheyangn1fcn	C′ c′ D′ S′ M	[-5,5]	0	$\sum_{i=1}^{n} \epsilon_i \lvert x_i \rvert^i$
xinsheyangn2fcn	C′ c D′ S′ M	$[-2\pi, 2\pi]$	0	$\sum_{i=1}^{n}(\lvert x_i \rvert)\ exp(-\sum_{i=1}^{n}$ sin(x_i^2))
xinsheyangn4fcn	C′ c D′ S′ M	[-10,10]	-1	$\left(\sum_{i=1}^{n} sin^2(x_i) - exp(-\sum_{i=1}^{n} x_i^2)\right) exp($-$\sum_{i=1}^{n} sin^2\sqrt{\lvert x_i \rvert})$
griewankfcn	C′ c D S U	[-600,600]	0	1 + $\sum_{i=1}^{n}\frac{x_i^2}{4000} - \prod_{i=1}^{n} cos(\frac{x_i}{\sqrt{i}})$
quarticfcn	C′c D S M	[-1.28,1.28]	0	$\sum_{i=1}^{n} ix_i^4 +$ random$[0, 1)$
rastriginfcn	C′ c D S M	[-5.12,5.12]	0	$10n + \sum_{i=1}^{n}(x_i^2 - 10cos(2\pi x_i))$
styblinskitankfcn	C′ c D S M	[-5,5]	-39.1659n	$\frac{1}{2}\sum_{i=1}^{n}(x_i^4 - 16x_i^2 + 5x_i)$
zakharovfcn	C c D S M	[-5,10]	0	$\sum_{i=1}^{n} x_i^2 + (\sum_{i=1}^{n} 0.5ix_i)^2 + (\sum_{i=1}^{n} 0.5ix_i)^4$
schwefelfcn	C’ c D’ S’ M	[-500,500]	0	418.9829d $-\sum_{i=1}^{n} x_i sin(\sqrt{\lvert x_i \rvert})$