

Received May 16, 2019, accepted June 4, 2019, date of publication June 14, 2019, date of current version July 1, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2923092

Benchmarks for Evaluating Optimization Algorithms and Benchmarking MATLAB Derivative-Free Optimizers for Practitioners' Rapid Access

LIN LI¹, (Member, IEEE), ALFREDO ALAN FLORES SALDIVAR¹, YUN BAI¹, (Member, IEEE), YI CHEN¹, (Senior Member, IEEE), QUNFENG LIU¹, AND YUN LI^{1,2}, (Senior Member, IEEE)

¹Industry 4.0 Artificial Intelligence Laboratory, Dongguan University of Technology, Dongguan 523808, China

²Department of Design, Manufacturing and Engineering Management, Faculty of Engineering, University of Strathclyde, Glasgow G1 1XJ, U.K.

Corresponding author: Yun Li (yun.li@ieee.org)

This work was supported in part by the Guangdong Higher Education Research Program under Grant 2017KQNCX193, in part by the Dongguan University of Technology under Research Grant 'Industry 4.0 Smart Design and Innovation Platform' KCYKYQD2017014 and under Postdoctoral Research Start-Up Grant GC300501-18, and in part by the National Natural Science Foundation of China under Grant 71801044 and Grant 61773119.

ABSTRACT MATLAB[®] has built in five derivative-free optimizers (DFOs), including two direct search algorithms (simplex search, pattern search) and three heuristic algorithms (simulated annealing, particle swarm optimization, and genetic algorithm), plus a few in the official user repository, such as Powell's conjugate (PC) direct search recommended by MathWorks[®]. To help a practicing engineer or scientist to choose a MATLAB DFO most suitable for their application at hand, this paper presents a set of five benchmarking criteria for optimization algorithms and then uses four widely adopted benchmark problems to evaluate the DFOs systematically. Comprehensive tests recommend that the PC be most suitable for a unimodal or relatively simple problem, whilst the genetic algorithm (with elitism in MATLAB, GAe) for a relatively complex, multimodal or unknown problem. This paper also provides an amalgamated scoring system and a decision tree for specific objectives, in addition to recommending the GAe for optimizing structures and categories as well as for offline global search together with PC for local parameter tuning or online adaptation. To verify these recommendations, all the six DFOs are further tested in a case study optimizing a popular nonlinear filter. The results corroborate the benchmarking results. It is expected that the benchmarking system would help select optimizers for practical applications.

INDEX TERMS Optimization methods, heuristic algorithms, evolutionary computation, benchmark testing, particle filters.

I. INTRODUCTION

As a high-level technical computing language and development environment, MATLAB[®] has over 4,000,000 registered users and over 525,000 contributors to its official repository [1]. It builds in five 'derivative-free optimizers' (DFOs) in the form of heuristic and direct search algorithms. These DFOs have provided powerful tools suitable for a broad range of real-world optimization applications and have hence been widely used by optimization practitioners [2]–[4].

The associate editor coordinating the review of this manuscript and approving it for publication was Wei Wei.

However, with the development of optimizers beyond conventional means, it has become challenging for a practicing engineer or scientist to select the most suitable one for their application at hand, especially without a common ground or a full set of criteria available for assessment. At present, there exists no widely applicable theoretical or analytical means to compare the performance of numerical optimizers. Further, benchmarks for comparing optimizers through tests are incomplete, mainly relying on fitness and convergence as benchmarks [5], despite a large number of benchmarking tests have been reported [6]. It is thus still difficult to assert quickly whether one particular algorithm is indeed 'better' or 'more suitable' than the others for a practitioner's application.

This paper aims to help practitioners rapidly to select a numerical optimizer, and in particular a DFO, suitable for their application at hand without needing to become an expert in these algorithms first. Therefore, we establish a full set of benchmarks for comparing DFOs, extending beyond those benchmarks currently available in the literature. Further, a scoring system for benchmarking DFOs and a decision tree are to be developed to facilitate their selection and use without the need for deep knowledge of the DFOs on the user’s part. We shall carry out complete benchmark tests against four Congress on Evolution Computation (CEC) benchmarking problems [7]–[11], and then verify the conclusion against a practical application - the particle filtering (PF) problem [12].

Included in the benchmarking tests are five MATLAB built-in DFO functions [2]–[4], i.e. simulated annealing (SA), particle swarm optimization (PSO), the genetic algorithm (with elitism, GAe), simplex search (SS), and pattern search (PS), plus one third-party implementation of the widely used Powell’s conjugate (PC) method (as an open-source m-file available from MATLAB’s official user repository [13]) recommended by MathWorks® [7]. Among these six DFO algorithms, the SS, PS and PC are direct search algorithms and the other three are heuristics, where the GAe uses encoding.

The following section analyzes the four CEC benchmark problems to be adopted in benchmarking. Then, Section III develops the full set of benchmarks for comparing numerical optimization algorithms. Section IV undertakes benchmarking tests, and then discusses the results with recommendations. Given the recommendations, Section V tests the DFOs further, in a case study of a nonlinear particle filter application. Section VI summarizes the paper and draws conclusions.

II. BENCHMARKING ANALYSIS FOR MATLAB DFOs

A. OPTIMIZATION OBJECTIVES AND SOLUTIONS

For evaluating the performance of optimization, search or machine learning algorithms, consider a benchmarking optimization problem. Suppose that the *objective function* or ‘performance index’ of the optimization, subject to direct and/or indirect constraints, is represented by:

$$f(\mathbf{x}) : X \rightarrow F \tag{1}$$

which is also called a *fitness function* in the context of maximization, or a *cost function* in that of minimization. Here, $\mathbf{f} \in \mathbf{F} \subseteq \mathbf{R}^m$ represents m objective elements, and $\mathbf{x} \in \mathbf{X} \subseteq \mathbf{R}^D$ represents D *decision variables* or parameters to be optimized. The benchmark function \mathbf{f} may only be evaluated numerically through computer simulations within \mathbf{X} .

As MATLAB DFOs are usually for a single objective, which would nonetheless include all elements of \mathbf{f} through preference weighting, we consider the case $m = 1$ here for simplicity. For benchmark testing, a benchmark problem or test function usually has a known *theoretical optimum*:

$$f_0 = \min_{\mathbf{x} \in X} f(\mathbf{x}) \tag{2}$$

Without loss of generality, minimization is typically considered, as maximization is simply an inverse case of minimization.

Note that a non-numerical decision variable, including a ‘logic’ variable (such as True/On or False/Off) and a ‘category’ variable (such as R, L or C in an electronic circuit) may only take a discrete value in \mathbf{R}^1 , in the form of an encoded \mathbf{x} as accommodated in the genetic algorithm or genetic programming. For an $\mathbf{x}_0 \in \mathbf{X}$ that satisfies:

$$f(\mathbf{x}_0) = f_0 \tag{3}$$

it is said to be a *theoretical solution* corresponding to f_0 to the optimization problem.

Let $\hat{\mathbf{x}}_0 \in \mathbf{X}$ represent a *found solution* by an optimization, search or machine learning algorithm, such as a DFO. Then, the corresponding *found optimum* is defined as

$$\hat{f}_0 = f(\hat{\mathbf{x}}_0) \tag{4}$$

which is usually an approximate optimum or a sub-optimum with respect to nondeterministic algorithms. Since DFOs are often nondeterministic, benchmarking should use mathematical means of results over multiple test runs.

According to evolutionary algorithm competitions in IEEE CEC’14 and CEC’15, many real-world problems are multimodal and can be represented by a set of benchmark problems or, analytically, test functions [8], [9]. Local optimization is concerned with unimodality, and global with multimodality. Given possible modal differences in different dimensions, four benchmark problems of unimodal, multimodal and hybrid types are used in this paper to illustrate and analyze benchmarking with thorough comparison. Further, all the functions are tested in 10 and 30 dimensions for 30 times [8], [9].

B. UNIMODAL PROBLEM

One of the basic test functions is a unimodal function, such as the Quartic test function f_1 [14], [15]:

$$f_1(\mathbf{x}) = \sum_{i=1}^D ix_i^4 + \text{random}[0, 1) \tag{5}$$

where $\mathbf{x} \in [-1.28, 1.28]^D$, $\text{random}[0, 1)$ is a Gaussian noise that helps assess the robustness of the tests, and $f_1(\mathbf{x}) \in [0, 147.64]$ and $f_1(\mathbf{x}) \in [0, 1248.2]$ for $D=10$ and $D=30$, respectively. A 3-D map for a 2-D function is shown in Fig. 1.

C. MULTIMODAL PROBLEMS

In addition to multidimensionality, multimodality is also a common feature seen in practical applications. Two example benchmark problems are given below.

1) VARYING LANDSCAPE MULTIMODAL PROBLEM

Consider the D -dimensional maximization problem introduced by Michalewicz [16] and further studied by Renders and Bersini [15]:

$$f_2(\mathbf{x}) = \sum_{i=1}^D f_i(x_i) = \sum_{i=1}^D \sin(x_i) \sin^{2m} \left(\frac{ix_i^2}{\pi} \right) \tag{6}$$

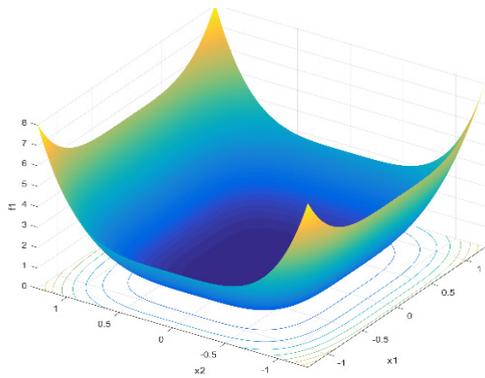


FIGURE 1. Contour plot to visualize a 2-D case of f_1 .

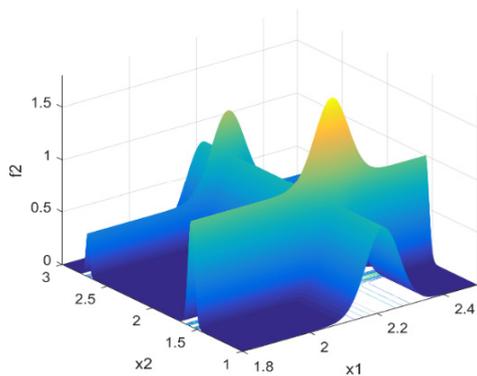


FIGURE 2. Visualizing a 2-D case of f_2 .

where $\mathbf{x} \in [0, \pi]^D$. It is composed of a family of amplitude-modulated sinewaves whose frequencies are linearly modulated. A 3-D map of a 2-D function is shown in Fig. 2.

The objective function $f_2(\mathbf{x})$ is, in effect, de-coupled in every dimension represented by $f_i(x_i)$, for the ease of verifying optimality. This characteristic yields the following properties:

- 1) The larger the product mD is the sharper the landscape becomes, and hence harder for the optimizer.
- 2) There are $D! = 2.6525 \times 10^{32}$ local maxima within the search space $[0, \pi]^{30}$.
- 3) The theoretical benchmark solution to this D -dimensional optimization problem may be obtained by maximizing D independent uni-dimensional functions, $f_i, \forall i \in \{1, \dots, D\}$, a fact that is however unknown to the optimizer under test. The results for $D \in \{10, 30\}$ and for $m=100$ are $f_2(\mathbf{x}) \in [0, 9.56]$ and $f_2(\mathbf{x}) \in [0, 29.63]$, respectively. The corresponding theoretical solutions are shown in Tables 1 and 2.
- 4) The ease of obtaining theoretical benchmarks regardless of D makes it ideal for studying nondeterministic polynomial (NP) characteristics of the algorithms being tested.

2) EXPANDED SCHAFFER'S MULTIMODAL PROBLEM

Consider a third test function f_3 of the expanded Schaffer's F6 function [8], [9], which includes many peaks and is difficult

TABLE 1. Theoretical solutions of benchmark function 2 for $D=10$.

i	1	2	3	4	5
x_{i0}	2.2195	1.5708	1.2828	1.9238	1.7207
i	6	7	8	9	10
x_{i0}	1.5708	1.4543	1.7562	1.6558	1.5708

TABLE 2. Theoretical solutions of benchmark function 2 for $D=30$.

i	1	2	3	4	5	6
x_{i0}	2.2195	1.5708	1.2828	1.9238	1.7207	1.5708
i	7	8	9	10	11	12
x_{i0}	1.4543	1.7562	1.6558	1.5708	1.4977	1.6967
i	13	14	15	16	17	18
x_{i0}	1.6301	1.5708	1.5176	1.6661	1.6164	1.5708
i	19	20	21	22	23	24
x_{i0}	1.5289	1.6475	1.6078	1.5708	1.5363	1.6350
i	25	26	27	28	29	30
x_{i0}	1.6019	1.5708	1.5415	1.5415	1.5977	1.5708

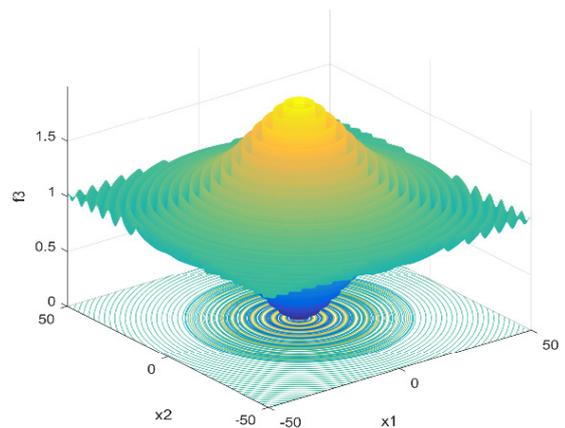


FIGURE 3. Visualizing a 2-D case of f_3 .

for an algorithm to converge to:

$$\begin{aligned}
 f_3(\mathbf{x}) &= g(x_1, x_2) + g(x_2, x_3) + \dots + g(x_{D-1}, x_D) \\
 &\quad + g(x_D, x_1) \\
 g(x, y) &= 0.5 + \frac{(\sin^2(\sqrt{x^2 + y^2}) - 0.5)}{(1 + 0.001(x^2 + y^2))^2} \tag{7}
 \end{aligned}$$

where $\mathbf{x} \in [-100, 100]^D$, $f_3(\mathbf{x}) \in [0, 9.98]$ and $f_3(\mathbf{x}) \in [0, 29.93]$ for $D = 10$ and $D = 30$, respectively. A 3-D map for a 2-D function is shown in Fig. 3.

D. HYBRID PROBLEM

In real-world optimization, subsets of decision variables may have diverse properties, similar to a hybrid test function, where the variables may be in subsets. Such a function is used to test diverse properties of the problem that may be encountered in practice. Hence, the fourth test function f_4 is a hybrid of Schwefel's function f_{41} , Rastrigin's function f_{42}

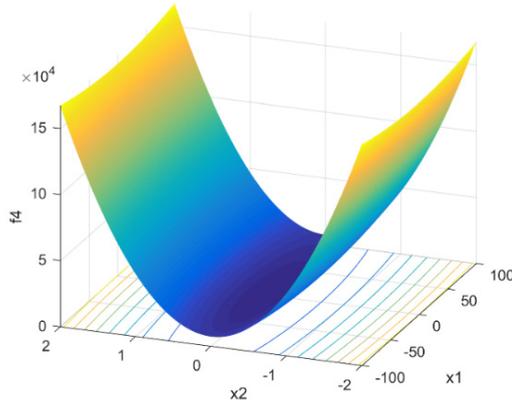


FIGURE 4. Visualizing a 2-D case of f_4 .

and the High Conditioned Elliptic Function f_{43} [8], [9]:

$$f_4(\mathbf{x}) = 0.3 f_{41}(\mathbf{x}) + 0.3 f_{42}(\mathbf{x}) + 0.4 f_{43}(\mathbf{x})$$

$$f_{41}(\mathbf{x}) = 418.9829 D - \sum_{i=1}^D g(z_i), \quad (8)$$

$$z_i = x_i + 4.209687462275036 \times 10^2 \quad (9)$$

$$g(z_i) = \begin{cases} z_i \sin(|z_i|^{1/2}) & \text{if } |z_i| \leq 500 \\ (500 - \text{mod}(z_i, 500)) \\ \times \sin(\sqrt{|500 - \text{mod}(z_i, 500)|}) \\ - \frac{(z_i - 500)^2}{10000D} & \text{if } z_i > 500 \\ (\text{mod}(|z_i|, 500) - 500) \\ \times \sin(\sqrt{|\text{mod}(|z_i|, 500) - 500|}) \\ - \frac{(z_i + 500)^2}{10000D} & \text{if } z_i < -500 \end{cases} \quad (10)$$

$$f_{42}(\mathbf{x}) = \sum_{i=1}^D (x_i^2 - 10 \cos(2\pi x_i) + 10) \quad (11)$$

$$f_{43}(\mathbf{x}) = \sum_{i=1}^D (10^6)^{\frac{i-1}{D-1}} x_i^2 \quad (12)$$

where $\mathbf{x} \in [-100, 100]^D$ and the hybrid rates used are 0.3, 0.3 and 0.4 for $f_{41}(\mathbf{x})$, $f_{42}(\mathbf{x})$ and $f_{43}(\mathbf{x})$, respectively. For $D=10$ and $D=30$, $f_4(\mathbf{x}) \in [0, 5.54 \times 10^8]$ and $f_4(\mathbf{x}) \in [0, 1.22 \times 10^9]$, respectively. The objective of this benchmark problem is to minimize $f_4(\mathbf{x})$, which is like a unimodal problem as depicted in Fig. 4 when $D=2$, which is relatively simple in global optimization.

In summary, f_1 is unimodal for testing local DFOs, f_4 is relatively simple in terms of modality, and f_3 and f_4 are multimodal for testing global DFOs.

III. FULL SET OF BENCHMARKS FOR EVALUATING NUMERICAL OPTIMIZERS

A. OPTIMALITY BENCHMARK

Optimality represents the relative closeness of a found optimum, \hat{f}_0 , to the theoretical optimum, f_0 , which is defined

here as:

$$Optimality = 1 - \frac{\|f_0 - \hat{f}_0\|}{\|\bar{f} - \underline{f}\|} \in [0, 1] \quad (13)$$

where \bar{f} and \underline{f} are the lower and upper bounds of f , respectively, which are known from proper benchmark problems or test functions.

For a single objective problem (i.e. $m=1$), consider the minimization problem with an objective bound of $[f_{\min}, f_{\max}]$ as an example. Then, by (13), the optimality benchmark can be simplified to:

$$Optimality_{\min} = 1 - \frac{|f_{\min} - \hat{f}_0|}{|f_{\max} - f_{\min}|} = \frac{f_{\max} - \hat{f}_0}{f_{\max} - f_{\min}} \quad (14)$$

Similarly, the optimality for the corresponding maximization problem can be simplified to:

$$Optimality_{\max} = 1 - \frac{|f_{\max} - \hat{f}_0|}{|f_{\max} - f_{\min}|} = \frac{\hat{f}_0 - f_{\min}}{f_{\max} - f_{\min}} \quad (15)$$

B. ACCURACY BENCHMARK

Is it necessary to introduce an accuracy benchmark in the control space in addition to the Optimality benchmark that already reflects a ‘performance index’ in the objective space? The necessity lies in that we are assessing global optimization that involves multiple optima and is approached stochastically often by a nondeterministic algorithm. As the highest value of optimality is one of the local optima, the quality of a found solution $\hat{\mathbf{x}}_0$ obtained by an algorithm cannot be assessed by the optimality benchmark alone if $\hat{\mathbf{x}}_0$ does not lie in the neighborhood of the global optimum.

Here, the accuracy benchmark is defined as the relative closeness of the found solution, $\hat{\mathbf{x}}_0$, to the theoretical solution, \mathbf{x}_0 :

$$Accuracy = 1 - \frac{\|\mathbf{x}_0 - \hat{\mathbf{x}}_0\|}{\|\bar{\mathbf{x}} - \underline{\mathbf{x}}\|} \in [0, 1] \quad (16)$$

where $\underline{\mathbf{x}}$ is the lower bound of \mathbf{x} and $\bar{\mathbf{x}}$ is the upper bound, and $[\underline{\mathbf{x}}, \bar{\mathbf{x}}]$ represents the search range. This benchmark may be particularly useful if the solution space is noisy, there exist multiple optima, or ‘niching’ is used.

By (16), the accuracy with respect to a single-parameter minimization problem within $[x_{\min}, x_{\max}]$ is measured by:

$$Accuracy_{D=1} = 1 - \frac{\hat{x}_0 - x_0}{x_{\max} - x_{\min}} \quad (17)$$

C. CONVERGENCE BENCHMARKS

To date, convergence of population-based algorithms is benchmarked only qualitatively using fitness or cost traces graphically against the number of iterations, generations or function evaluations (FEs). With this paper, convergence can now be measured both qualitatively and quantitatively.

Qualitatively, we adopt the method of graphical tracing, but on both benchmarks of optimality and accuracy, instead of on fitness alone, i.e.:

- The highest ‘optimality’ or fitness in every generation;
- The highest ‘accuracy’ or the parameter values of the individual solution that has the highest optimality fitness in every generation.

1) REACH TIME

Quantitatively, we define a ‘reach time’ as:

$$\text{Reach time}|_b = C^b \quad (18)$$

where C is the count of ‘function evaluations’ performed when the optimality of the best individual first reaches a predefined value b , where $b \in [0, 1]$, i.e., when the distance to the theoretical objective first drops to $(1 - b)$. For instance, $C^{0.999}$ indicates the generation number needed for an optimizer to reach the optimality with an error no greater than 0.1%, and $C^{0.632}$ indicates a convergence ‘time constant’ by which time optimality of 63.2% is first reached (in a manner similar to a first-order dynamic behavior).

2) NP-TIME

The power of a heuristic DFO is that it reduces an otherwise exponential time of an exhaustive search algorithm to a nondeterministic polynomial time. To estimate the order of the polynomial, $C^{0.999}$ may be plotted against the number of parameters being optimized, D . We can hence also propose a revised reach time, of D components, as:

$$\text{NP-time}(D) = C^{0.999}(D) \quad (19)$$

3) TOTAL NUMBER OF FUNCTION EVALUATIONS

The optimality of 99.9% may not be reached by certain algorithms under test. The total number of function evaluations is the number of evaluations, search trials or simulations performed until the optimization process is terminated. For benchmarking, the total number of evaluations is kept the same for all algorithms, such as $400mD^2$, which is more informatively defined as:

$$N = \min \{ C^{0.999}, 400mD^2 \} \quad (20)$$

Namely, with $C^{0.999}$, the benchmark test will terminate when the goal of the optimality reaching an error no greater than 0.1% is achieved or $20D$ generations with a population size of $(20D \times m)$ have been iterated.

D. OPTIMIZER OVERHEADS BENCHMARK

The ‘total CPU time’, T in seconds, of the entire optimization process can also be used in a benchmark test when assessing how long an optimization process would take in the real world to single out the amount of program overheads due to optimization maneuvers. Quantitatively, the optimizer overheads benchmark is defined as:

$$\text{Optimizer overheads} = \frac{T - T_{FE}}{T_{FE}} \quad (21)$$

where T_{FE} is defined as the CPU time taken (in seconds) for completing N function evaluations.

E. SENSITIVITY BENCHMARK

When the values of the optimal parameters found are perturbed, the optimality may well change, which will affect the reliability or robustness of the solution found and can hence impact on the quality of real-world applications. This is critical, for example, in an industrial design, where manufacturing tolerance may cause suddenly deteriorated quality of the end product as a result of the sensitive optimality or polarized optimization.

In this paper, ‘sensitivity’ is hence used to indicate the robustness of the optimizer by measuring how much relative change in the optimality will result from a ‘small’ relative change in the corresponding solution found. Quantitatively, the sensitivity is hence defined as:

$$\text{Sensitivity} = \frac{d(\text{Optimality})}{d(\text{Accuracy})} = \left\| \nabla \hat{f} \right\| \frac{\| \bar{\mathbf{x}} - \mathbf{x} \|}{\| \bar{\mathbf{f}} - \mathbf{f} \|} \quad (22)$$

In other words, sensitivity is a scaled ‘relative gradient’, which can be estimated to:

$$\text{Sensitivity} \approx \frac{\| \Delta \hat{f} \|}{\| \Delta \hat{\mathbf{x}} \|} \frac{\| \bar{\mathbf{x}} - \mathbf{x} \|}{\| \bar{\mathbf{f}} - \mathbf{f} \|} \quad (23)$$

where $\Delta \hat{f}$ is a neighborhood of \hat{f}_0 and $\Delta \hat{\mathbf{x}}$ is the corresponding neighborhood of the solution found.

It is convenient to set $\Delta \hat{\mathbf{x}} = 1\%$, which means that the prevailing sensitivity measures how many percent the optimality will degrade if the accuracy is 1% off. Note that the trend of sensitivity is rather dependent on the nature of the problem (e.g., the test function), and not mainly on the optimizer.

In addition to sensitivity, the two-tailed ‘t-test’ could be used to assess the reliability in comparing two algorithms to a certain degree [14], [18]. This is a statistical hypothesis test, which is usually used to determine if two sets of data are significantly different from each other [19] and used as a complimentary approach to further validation of a comparison when necessary.

IV. BENCHMARKING MATLAB DFOs

A. TEST CONDITIONS

Using the defined benchmarks and the chosen benchmark problems, we investigate the performance of the MATLAB (R2016b) heuristics SA, PSO and GAe, and the direct-search optimizers, SS, PS, and PC, through extensive experimental tests. Since the main purpose of this work is to help MATLAB DFO users select an algorithm without the need for in-depth knowledge of the algorithms, MATLAB’s default settings for these algorithms are used in the tests (unless otherwise stated) and all the DFOs are tested on the same random set of starting points.

Therefore, for SS, the reflection coefficient is 1, the expansion coefficient 2, the contraction coefficient 0.5, and the shrink coefficient 0.5.

For SA, the initial temperature is 100, and the temperature is reduced to 95% consecutively at each iteration.

For PSO, the swarm size is $20D$, where D is the dimension number of the test function. The maximum iteration is $20D$; so the maximum number of function evaluations is $400D^2$.

MATLAB’s genetic algorithm uses elitism, hence named GAe, which guarantees the best 5% of a generation to survive to the next generation. For the GAe, the population size is $20D$. The maximum number of generations is also $20D$; so the maximum number of function evaluations is the same as PSO. The crossover rate was 0.8, and the mutation rate is 0.2.

Each experiment is repeated 30 times to obtain a mean value of each benchmark value. As SA, SS, PS and PC are unary (as opposed to population-based) optimizers, one starting point is generated randomly for each run, identically used by all of these optimizers for fair comparison. Note that, however, when displaying convergence traces for these unary optimizers, one point is plotted for every $20D$ FEs, so as to compare fairly with one ‘generation’ of the population-based optimizers. For each run of the latter, the initial generation of $20D$ individuals are generated randomly for use identically by both PSO and the GAe.

In order to compare the optimizer overheads later, the CPU time, albeit less accurate in MATLAB than in a real-time program, is recorded for N function evaluations. The average T_{FE} over 30 runs for every benchmark problem is shown in Table 3. It also reveals that, when D increases by two times from 10 to 30, T_{FE} increases by over 20 times, indicating a highly nonlinear challenge by the dimensionality. These FEs were performed on a PC of an Intel Core i7-4790K 4.00 GHz CPU with 8 GB RAM running a Windows 64-bit operating system.

TABLE 3. Time taken by NFEs and dimensionality challenge.

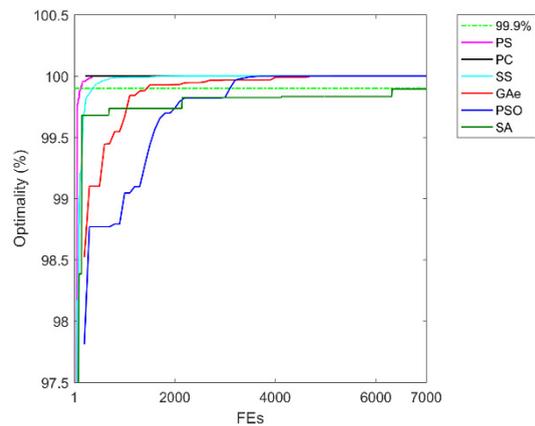
Functions	10-D function T_{FE} (s)	30-D function T_{FE} (s)	30-D / 10-D T_{FE} ratio
f_1	0.0487	1.2635	25.9
f_2	0.0595	1.5625	26.3
f_3	0.0215	0.5491	25.5
f_4	0.1002	2.2604	22.6

B. BENCHMARKING DFOs ON FOUR TEST PROBLEMS

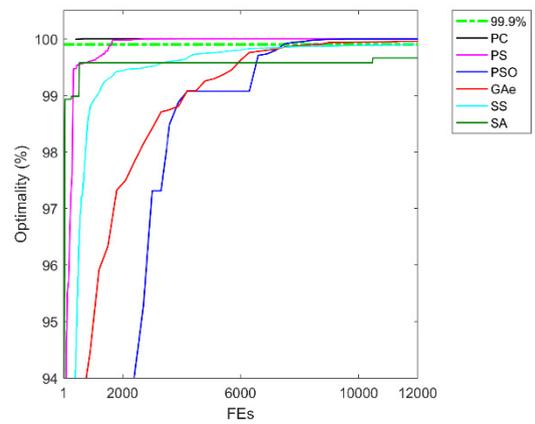
The performance of the six individual algorithms on each of the five benchmarks was first ranked and then assigned scores from 6 to 1, with 6 representing the best. For example, if the mean optimality of PSO is the highest among all the six algorithms, then it scores a 6 on Optimality. The scores of each algorithm on all the five benchmarks are summed up to give the algorithm a total score. Results from all the experiments are summarized in Tables 4-7, where the bold indicates the best at each benchmark and the red the best overall score and corresponding algorithm.

1) QUARTIC UNIMODAL PROBLEM f_1

It can be observed that, for 10-D, the algorithms except SS and SA offer optimality over 99.9995%. PC offers the



(a)



(b)

FIGURE 5. Typical optimality of each DFO in optimizing the unimodal quartic problem f_1 . (a) $D = 10$. (b) $D = 30$.

best performance overall, although PC, PS and PSO all win on three benchmarks. From both the 10-D and 30-D tests, it can be seen that PC offers the best overall performance, and should hence be recommended for uni-modal problems. Further, PC also offers the most ‘linear’ polynomial time (although NP), shown in the last column, whilst SS and SA are the worst. As expected, PSO and GAe are also relatively ‘polynomial’. On contrast, SS and SA offer the lowest performance with the highest overheads, and hence should be avoided for this type of problems. Fig. 5 depicts typical optimality convergence of the best point in a ‘generation’.

2) VARYING LANDSCAPE n-D MULTIMODAL PROBLEM f_2

This function has multiple extrema in $[0, \pi]^D$ and is a more challenging problem than the unimodal problem. Table 5 summarizes the test results and Fig. 6 shows the convergence traces of the optimizers. It is seen that none of the algorithms was able to reach the optimality of 99.9% in N function evaluations. SS and SA were observed stalled at a local maximum and hence delivered poor performance. The GAe, on contrast, showed steady performance and produced the best results in both 10-D and 30-D tests. PSO, PC and PS were able to approach the global optimum with relatively good results.

TABLE 4. Benchmarking results in solving the quartic unimodal problem (10-D and 30-D).

Algorithms Tested	10-D						30-D						NP time	
	Optimality (%)	Accuracy (%)	Reach-Time (FEs)	Optimizer Overheads (%)	Sensitivity (%)	Score	Optimality (%)	Accuracy (%)	Reach-Time (FEs)	Optimizer Overheads (%)	Sensitivity (%)	Score	Convergence 30-D / 10-D reach time ratio	
Direct search	SS	98.8197	85.6869	395	5714.20	6.3181	14	96.1318	78.1757	11816	1917.40	16.8683	13	29.9
	PS	100	99.9999	158	3988.00	0	26	100	99.9999	1593	884.1728	0	25	10.1
	PC	100	99.9991	218	626.50	0	27	100	100	638	207.4658	0	30	2.9
Heuristic search	SA	99.1124	86.7034	6516	84857.00	6.1442	13	99.4206	87.9779	360000	22278.00	4.7661	14	55.2
	PSO	100	100	1580	765.86	0	25	100	100	7560	338.618	0	27	4.8
	GAe	100	99.5390	1100	1808.50	7.36×10^{-4}	21	100	99.5236	9240	381.6034	9.67×10^{-4}	22	8.4

NB. The longest reach time was truncated at the maximum T_{FE} allowed, i.e., 40,000 and 360,000 for 10-D and 30-D tests, respectively.

TABLE 5. Benchmarking results in solving the varying landscape multimodal problem (10-D and 30-D).

Algorithms Tested	10-D						30-D						
	Optimality (%)	Accuracy (%)	Reach-Time (FEs)	Optimizer Overheads (%)	Sensitivity (%)	Score	Optimality (%)	Accuracy (%)	Reach-Time (FEs)	Optimizer Overheads (%)	Sensitivity (%)	Score	
Direct search	SS	24.3961	69.3935	40000	3683.00	252.1045	6	15.2886	70.1223	360000	1054.30	285.0725	7
	PS	75.6471	75.6478	40000	3441.40	97.8684	15	72.0822	74.7029	360000	612.52	96.3042	16
	PC	70.9946	73.4307	40000	565.84	107.1184	15	69.716	73.4067	360000	146.4539	94.7657	17
Heuristic search	SA	19.7309	73.3067	40000	59076	314.0178	6	13.9521	73.3699	360000	10812.00	333.0779	5
	PSO	80.3864	78.9212	40000	636.35	86.9948	20	69.9723	75.3317	360000	232.5636	118.8643	17
	GAe	91.6615	89.3889	40000	1562.00	82.4263	22	92.7308	90.5972	360000	259.1471	77.1631	22

TABLE 6. Benchmarking results in solving schaffer's F6 multimodal problem (10-D and 30-D).

Algorithms Tested	10-D						30-D						
	Optimality (%)	Accuracy (%)	Reach-Time (FEs)	Optimizer Overheads (%)	Sensitivity (%)	Score	Optimality (%)	Accuracy (%)	Reach-Time (FEs)	Optimizer Overheads (%)	Sensitivity (%)	Score	
Direct search	SS	52.7743	70.7192	40000	12781.00	165.0165	8	50.8777	66.7263	360000	9185.10	147.6313	8
	PS	82.5042	92.7385	40000	11403.00	259.8067	15	87.6359	95.4229	360000	3623.30	270.1302	15
	PC	52.9141	70.9131	40000	1429.10	165.6134	13	51.0547	66.7452	360000	473.3057	147.1825	15
Heuristic search	SA	56.1722	72.4828	40000	202110.00	161.9799	12	54.2523	69.6833	360000	100870	150.8992	10
	PSO	80.7691	85.8677	40000	1783.50	151.9992	19	71.339	79.3394	360000	1297.10	138.723	19
	GAe	90.9756	97.1353	40000	4991.60	311.3505	17	88.0715	96.4565	360000	1675.70	336.627	17

3) EXPANDED SCAFFER'S F6 MULTIMODAL PROBLEM f_3

Table 6 summarizes the test results on this problem. Fig. 7 shows the convergence traces of the algorithms. It can be seen that PSO delivered consistently good and the overall

best performance. However, the GAe again consistently demonstrated the highest optimality and highest accuracy in both 10-D and 30-D tests. Same as on f_2 , SS and SA showed poor optimality and accuracy, indicating low capability on

TABLE 7. Benchmarking results in solving the hybrid problem (10-D and 30-D).

Algorithms Tested	10-D						30-D						NP time	
	Optimality (%)	Accuracy (%)	Reach-Time (FEs)	Optimizer Overheads (%)	Sensitivity (%)	Score	Optimality (%)	Accuracy (%)	Reach-Time (FEs)	Optimizer Overheads (%)	Sensitivity (%)	Score	Convergence 30-D / 10-D reach time ratio	
Direct search	SS	99.5339	64.7221	622	2609.90	1.35	10	96.1921	67.9613	38174	1064.10	11.9876	8	61.4
	PS	100	99.9998	572	1808.50	7.09×10^{-4}	25	100	99.9999	5059	507.317	3.37×10^{-4}	24	8.8
	PC	100	99.8016	348	261.6911	1.90×10^{-3}	26	100	99.7659	997	106.6015	3.29×10^{-4}	28	2.9
Heuristic search	SA	99.8832	79.455	287	35662.00	0.556	15	99.7486	79.022	2443	12673.00	1.1786	14	8.5
	PSO	100	99.9792	1656	312.2917	3.38×10^{-2}	22	100	99.9249	9870	192.542	0.0257	23	6.0
	GAe	99.9977	99.0954	1843	779.1292	0.484	16	99.9984	99.5473	17130	208.3125	0.3501	17	9.3

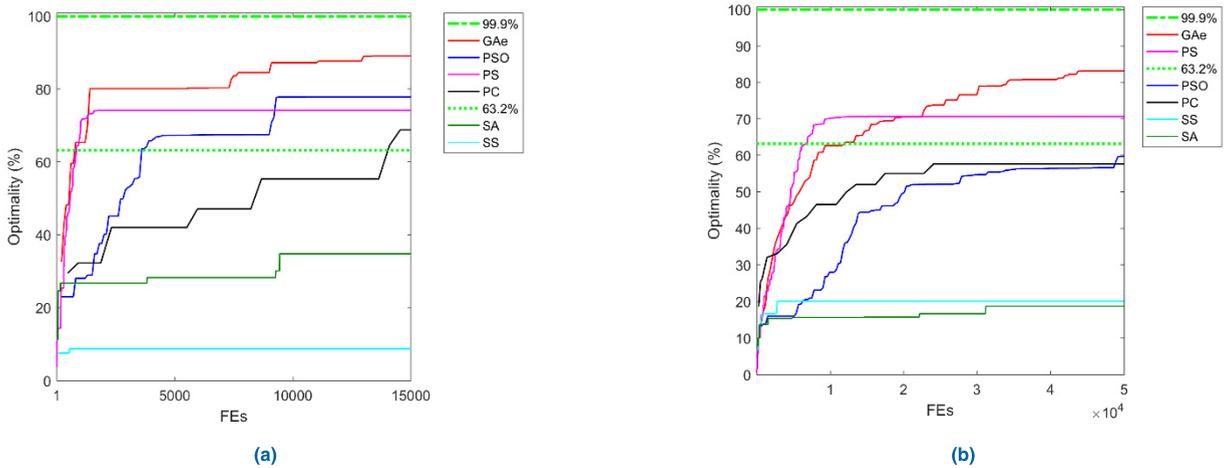


FIGURE 6. Typical optimality of each DFO in optimizing the varying landscape problem f_2 . (a) $D = 10$. (b) $D = 30$.

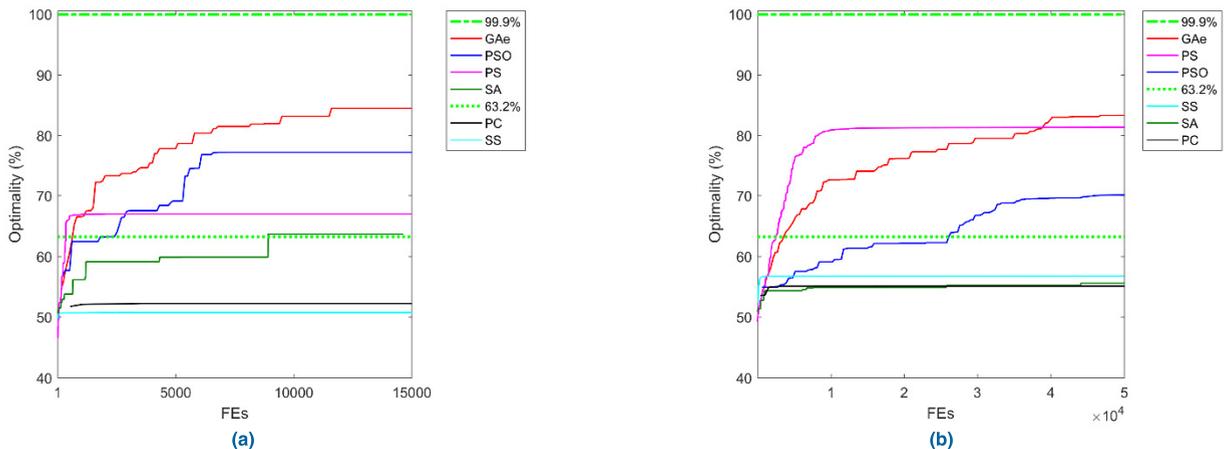


FIGURE 7. Typical optimality of each DFO in optimizing Schaffer's F6 problem f_3 . (a) $D = 30$. (b) $D = 30$.

multimodal problems. It is also seen that the optimality of PC was relatively inconsistent with that on the previous multimodal problem, indicating it is likely to be trapped in a local optimum when the complexity of the problem increases.

4) HYBRID PROBLEM f_4

Table 7 summarizes the test results on this function. More optimizer overheads were expected due to the complexity of this function. Except for SA and SS, the overheads of the algorithms are less than 2000. Results depicted

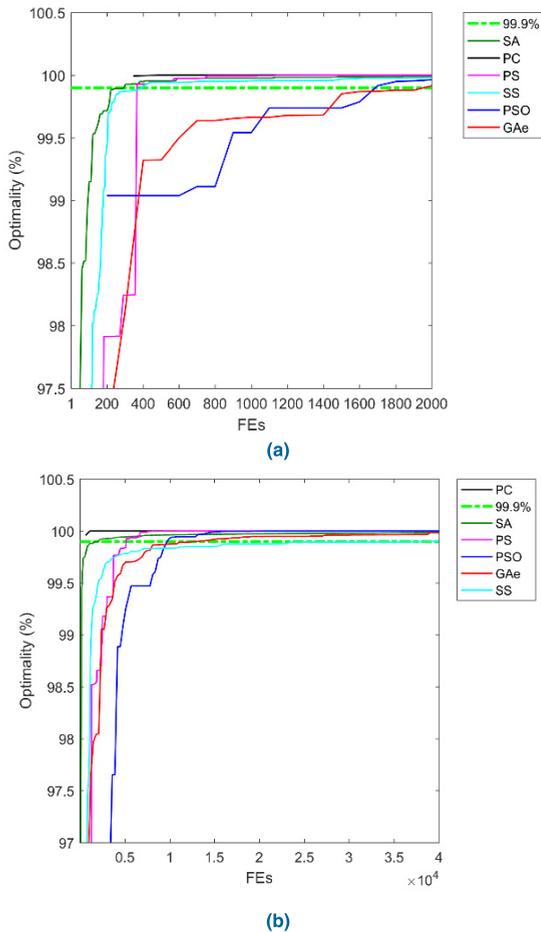


FIGURE 8. Typical optimality of each DFO in optimizing the hybrid problem f_4 . (a) $D = 10$. (b) $D = 30$.

in Fig. 8 shows that PC has similar convergence tendency to the case in f_3 . Combining Figs. 7 and 8, we can conclude that PC is suitable for simple or unimodal functions, but not for multimodal functions. As f_4 has relatively more straightforward troughs, PS and PC were able to exploit the search well.

C. COMPARISON ON INDIVIDUAL BENCHMARKS

1) OPTIMALITY

Table 8 presents the scores and ranking of the six algorithms on optimality in solving problems f_1 to f_4 in 10-D and 30-D, where a higher score is assigned to better optimality. The algorithms are then ranked from 1 to 6 based on their total scores in solving all the problems, where 1 being the top rank. Overall, when a good optimal solution is needed, GAe is seen the best method to use.

2) ACCURACY

Table 9 ranks the six algorithms on their accuracy. It can be seen that, when applied to f_2 , f_3 and f_4 , the algorithms did equally well in both 10-D and 30-D. Although different in 10-D and 30-D on f_1 , SS and SA did worse than the other four algorithms. For accuracy, GAe is seen the most suitable

TABLE 8. Scores and ranking on optimality.

Algorithms	Scores f_1		Scores f_2		Scores f_3		Scores f_4		Total rank
	10 D	30 D							
SS	4	4	2	2	1	1	3	3	6
PS	6	6	4	5	5	5	6	6	2
PC	6	6	3	3	2	2	6	6	4
SA	5	5	1	1	3	3	4	4	5
PSO	6	6	5	4	4	4	6	6	3
GAe	6	6	6	6	6	6	5	5	1

TABLE 9. Scores and ranking on accuracy.

Algorithms	Scores f_1		Scores f_2		Scores f_3		Scores f_4		Total Rank
	10 D	30 D							
SS	1	2	1	1	1	1	1	1	5
PS	5	5	4	4	5	5	6	6	1
PC	4	6	3	3	2	2	4	4	3
SA	2	3	2	2	3	3	2	2	4
PSO	6	6	5	5	4	4	5	5	1
GAe	3	4	6	6	6	6	3	3	2

TABLE 10. Scores and ranking on reach time.

Algorithms	Scores f_1		Scores f_2		Scores f_3		Scores f_4		Total Rank
	10 D	30 D							
SS	4	2	-	-	-	-	3	1	3
PS	6	5	-	-	-	-	4	4	2
PC	5	6	-	-	-	-	5	6	1
SA	1	1	-	-	-	-	6	5	4
PSO	2	4	-	-	-	-	2	3	5
GAe	3	3	-	-	-	-	1	2	6

for multimodal problems such as f_2 , f_3 , whilst PS and PSO for unimodal and simple problems such as f_1 and f_4 .

3) REACH TIME

Table 10 ranks on reach time, where f_2 and f_3 are missing, as none of the algorithms could reach 99.9% accuracy by the end of the predefined maximum number of FEs. When the purpose of optimization is to find a relatively fast converging solution, the PC is seen most suitable for unimodal and simple problems.

4) OPTIMIZER OVERHEADS

Table 11 ranks on optimizer overheads. Note that the ranking of each algorithm remains unchanged across the problems and dimensions, suggesting that optimizer overheads of the algorithms are not problem-dependent. It is therefore reasonable to conclude that, where reducing computational overheads is a priority, PC would be the most suitable, followed by PSO and GAe.

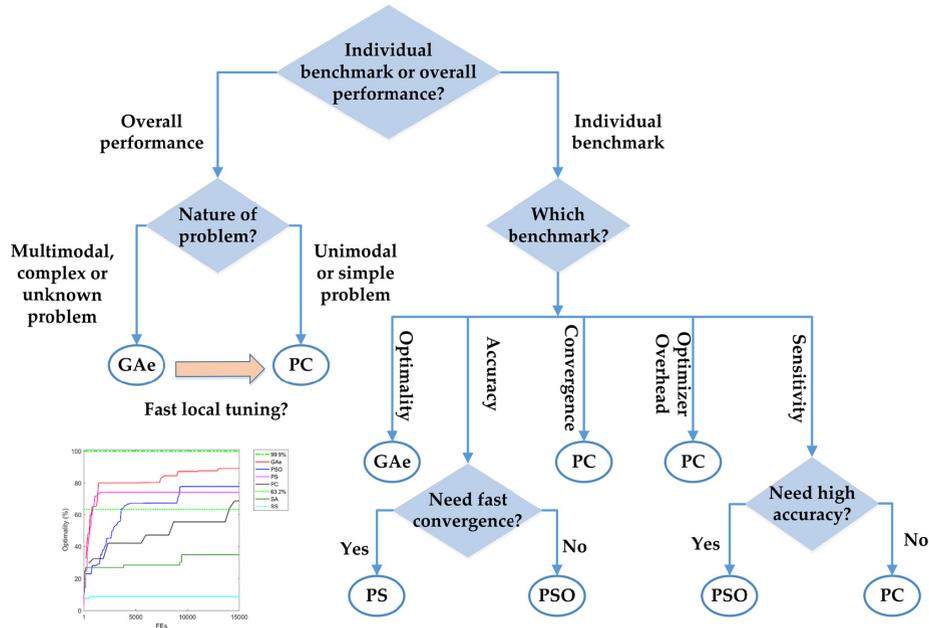


FIGURE 9. A simplified decision tree for selecting a DFO. Note that if an application requires optimizing its structure, the GAe is recommended, since its encoding capability permits optimization of structures and categories, in addition to numerical parameters of a structure. Relevantly, the GAe is recommended for global search offline together with PC for local tuning or rapid learning and adaptation online.

TABLE 11. Scores and ranking on optimizer overheads.

Algorithms	Scores f_1		Scores f_2		Scores f_3		Scores f_4		Total Rank
	10	30	10	30	10	30	10	30	
	D	D	D	D	D	D	D	D	
SS	2	2	2	2	2	2	2	2	5
PS	3	3	3	3	3	3	3	3	4
PC	6	6	6	6	6	6	6	6	1
SA	1	1	1	1	1	1	1	1	6
PSO	5	5	5	5	5	5	5	5	2
GAe	4	4	4	4	4	4	4	4	3

TABLE 12. Scores and ranking on sensitivity.

Algorithms	Scores f_1		Scores f_2		Scores f_3		Scores f_4		Total Rank
	10	30	10	30	10	30	10	30	
	D	D	D	D	D	D	D	D	
SS	3	3	1	2	4	4	1	1	5
PS	6	6	4	4	2	2	6	5	2
PC	6	6	3	5	3	5	5	6	1
SA	4	4	2	1	5	3	2	2	4
PSO	6	6	5	3	6	6	4	4	1
GAe	5	5	6	6	1	1	3	3	3

5) SENSITIVITY

Table 12 ranks on sensitivity. It can be seen that PC and PSO would be the choice, if low sensitivity is a major concern in practice.

D. QUICK GUIDE TO SELECTING A DFO

To suit their application at hand most, the user could select a DFO using its individual benchmark ranking or the overall ranking on amalgamated scores in Table 13, which is

summarized from Tables 8-12. For example, if the objective space or optimality is of a paramount importance for the application, then the GAe would be the No. 1 tool for it, as well as for a potentially multimodal or relatively complex problem.

If the decision space or accuracy is the goal, then PSO and PS would be the choice, with the former being more suitable for a multimodal problem and the latter unimodal. For unknown modality or a relatively simple problem, PC would provide a rapid assessment on possible effects of applying optimization to the problem at hand.

If the application is rather unknown, however, it is recommended that it could first be treated as a multimodal problem for global optimization using the GAe, which will take a longer time, and then as a unimodal problem for local optimization using PC, which will be a lot quicker. From the above analysis, a visual guide is provided as a decision tree in Fig. 9.

V. CASE STUDY - PARTICLE FILTERING

A. THE STOCHASTIC NONLINEAR FILTER

The particle filter (PF) uses a set of particles (i.e., samples) to represent the posterior distribution of a stochastic process given partial observations. It is a Monte Carlo algorithm to solve Bayesian estimation and signal processing problems. Second to the median filter, the PF is so far the most widely used nonlinear filter [20] following the Sequential Importance Resampling algorithm developed by Gordon [21].

A PF can be formalized in the generic form of a nonlinear system:

$$x_k = f(x_{k-1}, u_{k-1}) \tag{24}$$

TABLE 13. Individual and overall performance rankings of heuristic and direct search DFO for structural and numerical optimization.

Algorithms Tested		Optimality	Accuracy	Reach-Time	Overheads	Sensitivity	Multimodal Score & Rank		Unimodal Score & Rank		Hybrid Score & Rank	
Direct search	SS	6	5	3	5	5	29	6	27	5	18	6
	PS	2	1	2	4	2	61	3	51	2	49	2
	PC	4	3	1	1	1	60	4	57	1	54	1
Heuristic search	SA	5	4	4	6	4	33	5	27	5	29	5
	PSO	3	1	5	2	1	75	2	52	3	45	3
	GAe	1	2	6	3	3	78	1	43	4	33	4

$$\mathbf{z}_k = h(\mathbf{x}_k, \mathbf{v}_{k-1}) \quad (25)$$

where \mathbf{x}_k and \mathbf{z}_k are the state variables and observations at time k , respectively, $\mathbf{x}_k \in \mathbf{R}^{n_x}$, $\mathbf{z}_k \in \mathbf{R}^{n_z}$, n_x and n_z are the dimensions of \mathbf{x}_k and \mathbf{z}_k , respectively, $f(\cdot)$ and $h(\cdot)$ represent the system and observation functions, respectively, $\mathbf{u}_{k-1} \in \mathbf{R}^{n_u}$ is the system noise and $\mathbf{v}_k \in \mathbf{R}^{n_v}$ is the observation noise of given distributions, and \mathbf{u}_{k-1} and \mathbf{v}_{k-1} are independent of the past and current states. In addition, \mathbf{v}_k may be independent of \mathbf{u}_{k-1} . Denote the state and observation sequences by $\mathbf{x}_{0:k} = \{\mathbf{x}_0, \dots, \mathbf{x}_k\}$ and $\mathbf{z}_{1:k} = \{\mathbf{z}_1, \dots, \mathbf{z}_k\}$, respectively. The initial distribution $p(\mathbf{x}_0)$ is known a priori.

The objective of the PF is recursively to estimate the posterior density $p(\mathbf{x}_{0:k}|\mathbf{z}_{1:k})$ of the state $\mathbf{x}_{0:k}$ based on all available measurements $\mathbf{z}_{1:k}$. A recursive derivation of the posterior density is given by the Bayesian theorem when new observations arrive:

$$p(\mathbf{x}_{0:k}|\mathbf{z}_{1:k}) = p(\mathbf{x}_{0:k-1}|\mathbf{z}_{1:k-1}) \frac{p(\mathbf{z}_k|\mathbf{x}_k) p(\mathbf{x}_k|\mathbf{x}_{k-1})}{p(\mathbf{z}_k|\mathbf{z}_{1:k-1})} \quad (26)$$

where the conditional density $p(\mathbf{z}_k|\mathbf{z}_{1:k-1})$ is a normalizing constant. Eq. (26) can be simplified to:

$$p(\mathbf{x}_{0:k}|\mathbf{z}_{1:k}) \propto p(\mathbf{x}_{0:k-1}|\mathbf{z}_{1:k-1}) p(\mathbf{z}_k|\mathbf{x}_k) p(\mathbf{x}_k|\mathbf{x}_{k-1}) \quad (27)$$

where \propto means proportional to [19].

The above formulas are intractable integrals to an analytic solution for $p(\mathbf{x}_{0:k}|\mathbf{z}_{1:k})$. The PF hence uses the Monte Carlo methods to translate the integral problem into a cumulative particle probability transition. It approximates $p(\mathbf{x}_{0:k}|\mathbf{z}_{1:k})$ with a mass of particles $\mathbf{x}_{0:k}^i$ ($i = 1, \dots, N$), in which N is the particle number, \mathbf{x}_0^i is a set of initial particles drawn from $p(\mathbf{x}_0)$.

To estimate the posterior distribution of the transition, the PF uses an ‘importance sampling’ technique to sample particles first [22]:

$$q(\mathbf{x}_{0:k}|\mathbf{z}_{1:k}) = q(\mathbf{x}_{0:k-1}|\mathbf{z}_{1:k-1}) q(\mathbf{x}_k|\mathbf{x}_{0:k-1}, \mathbf{z}_{1:k}) \quad (28)$$

Accordingly, the weights of the particles are termed the importance weight. Un-normalized weights can then be written as

$$w_k^i = \frac{p(\mathbf{x}_{0:k}|\mathbf{z}_{1:k})}{q(\mathbf{x}_{0:k}|\mathbf{z}_{1:k})} \propto w_{k-1}^i \frac{p(\mathbf{z}_k|\mathbf{x}_k^i) p(\mathbf{x}_k^i|\mathbf{x}_{k-1}^i)}{q(\mathbf{x}_k^i|\mathbf{x}_{0:k-1}^i, \mathbf{z}_{1:k})} \quad (29)$$

where w_k^i is the importance weight. Denote the normalized w_k^i as \tilde{w}_k^i . If the importance distribution satisfies

$$q(\mathbf{x}_k|\mathbf{x}_{0:k-1}, \mathbf{z}_{1:k}) = q(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{z}_k) \quad (30)$$

Then

$$\tilde{w}_k^i \propto \tilde{w}_{k-1}^i \frac{p(\mathbf{z}_k|\mathbf{x}_k^i) p(\mathbf{x}_k^i|\mathbf{x}_{k-1}^i)}{q(\mathbf{x}_k^i|\mathbf{x}_{k-1}^i, \mathbf{z}_k)} \quad (31)$$

Consequently, the posterior density $p(\mathbf{x}_k|\mathbf{z}_{1:k})$ can be formulated as:

$$p(\mathbf{x}_k|\mathbf{z}_{1:k}) \approx \sum_{i=1}^N \tilde{w}_k^i \delta(\mathbf{x}_k - \mathbf{x}_k^i) \quad (32)$$

where $\delta(\cdot)$ is the Dirac delta [22].

Despite that the PF is the second most successful nonlinear filter, serious challenges exist. One is the particle impoverishment problem, which due to most particles sharing a few distinct values cause a loss of diversity and estimation.

To improve, a larger number of optimization algorithms have been applied to particle filtering.

B. TESTS OF MATLAB DFOs AND VERIFICATION

In this case study, tests of all the six DFO are conducted to select the most suitable DFO for the PF and to verify the benchmark guide. In the following, the SS, PS, PC, SA, PSO and GAe versions of the PF are termed SSPF, PSPF, PCPF, SAPF, PSOPF and GAePF, respectively.

A PF application is likely a multimodal or complex problem, the nature of which could be unknown to the user. The tests will be under the representative conditions described in [18], which are widely adopted owing to their strong nonlinearity [22]–[24].

The test formulation is given by:

$$x_k = 1 + \sin(w\pi k) + \phi_1 x_{k-1} + v_k \quad (33)$$

$$z_k = \begin{cases} \phi_2 x_k^2 + n_k & k \leq 30 \\ \phi_3 x_k - 2 + n_k & k > 30 \end{cases} \quad (34)$$

where v_k is a Gamma (3,2) random variable modeling the process noise, and $w = 4e - 2$, $\phi_1 = \phi_3 = 0.5$, and $\phi_2 = 0.2$ are scalar parameters. The observation noise n_k is drawn from a Gaussian distribution $N(0, 0.00001)$. Different filters were used to estimate the state sequences \mathbf{x}_k for $k = 1, 2, \dots, T$, with the total observation time set as $T = 70$. The maximum

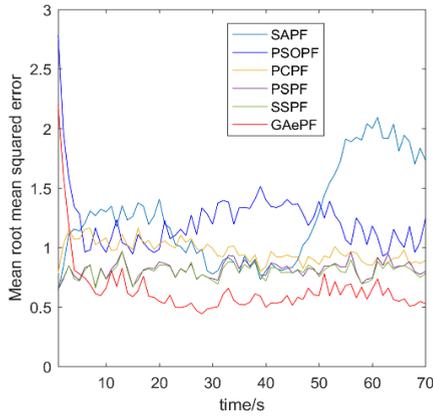


FIGURE 10. Mean RMSE errors with observation times for 100 simulations, comparing MATLAB DFOs.

number of generations is set as $G = 20$. In the GAePF, the crossover probability and the mutation probability are the usual 0.8 and 0.2, respectively. Other parameters of other four DFOs are the same as the above. All particle filters used $N = 100$ particles. The experiment was repeated for $M = 100$ Monte Carlo simulations to evaluate the performance of the six potential DFO PF algorithms.

1) ACCURACY ANALYSIS ON STATE ESTIMATIONS

The following three metrics were used to evaluate the performance of the six PF algorithms:

$$rmse = \sqrt{\frac{1}{T} \sum_{k=1}^T (x_k - \hat{x}_k)^2} \tag{35}$$

$$\overline{rmse} = \sqrt{\frac{1}{M} \sum_{m=1}^M \left[\frac{1}{T} \sum_{k=1}^T (x_k^m - \hat{x}_k^m)^2 \right]} \tag{36}$$

$$rmse' = \sqrt{\frac{1}{M} \sum_{m=1}^M (x_k^m - \hat{x}_k^m)^2} \tag{37}$$

where $rmse$ is the root mean squared error, its mean is \overline{rmse} , $rmse'$ is the $rmse$ over M simulations with a given observation time, x_k^m is the real state at time k for the m -th simulation, and \hat{x}_k^m is the estimated state.

Given the observation time, the mean $rmse$ values at every observation in 100 simulations of the six algorithms are shown in Fig. 10. As expected from the DFO guide of Fig. 9, the mean $rmse$ values of the GAePF are seen the lowest. Unexpectedly, however, the PSOPF has shown poor performance. Investigation into this has uncovered that the PSO in MATLAB does not make a set of the best population available for access after optimization, but the PF algorithm needs a set of particles.

To further verify the results, the mean estimation $rmse$ of the six algorithms in different numbers of simulations are shown in Fig. 11, where $N = 100$ and $M = 100$. This corroborates the observations from Fig. 10 above, confirming that the GAe is indeed the best DFO for such an unknown application as recommended in the guide.

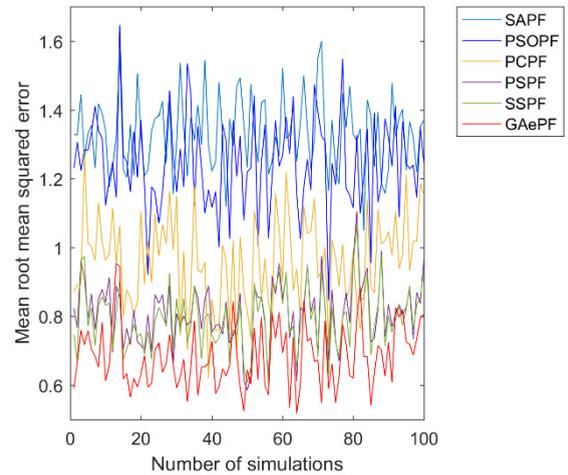


FIGURE 11. Mean RMSE errors vs. the number of simulations for 100 simulations, comparing MATLAB DFOs.

TABLE 14. Mean RMSE and runtime.

Algorithm	Performance		
	Mean root mean squared error	Runtime (s)	Particles needed (N)
GAePF	0.6832	4.3240	100
SSPF	0.8060	4.4941	300
PSPF	0.8088	20.3400	300
PCPF	0.9588	70.0731	300
PSOPF	1.0436	6.5433	300
SAPF	1.2410	1.3796	300

2) UTILIZATION OF PARTICLES AND RUNTIME

The means $rmse$ and average runtime with the total observation time T of the algorithms over 100 simulations are compared in Table 14 for both $N = 100$ and $N = 300$ at $G = 20$. It can be seen that the GAePF achieved a lower $rmse$ for just 100 particles than all the other algorithms even with 300 particles. As a result, the GAePF also demanded a shorter runtime. In other words, the use of the GAe needed fewer particles to achieve more accurate estimations for the PF application. These tests also confirm that the PF problem, looking relatively complex, is indeed a relatively multimodal problem.

VI. CONCLUSIONS

In this paper, a set of five criteria for benchmarking numerical optimization algorithms are first presented. These benchmarks reflect performance of DFOs on various aspects. Not only have the existing optimality, accuracy and convergence benchmarks been expanded in this paper, but also have the optimizer overheads and sensitivity benchmarks developed to enrich the existing evaluation criteria for practical numerical optimization applications.

Using four relatively representative and commonly adopted benchmarking problems, the six DFO algorithms available in MATLAB have been compared and ranked on the five benchmarks. The experiments have shown that PC is the

overall best for unimodal and simple hybrid problems, whilst the GAe is the overall best for multimodal and relatively complex problems.

To provide a guide for practical engineers to decide rapidly which of the DFOs would be the most suitable for their application at hand without the need for in-depth knowledge of the DFO algorithms themselves, a scoring system and a decision tree are provided in this paper. If optimality (the objective space) is of the paramount importance to the application, such as in a competition, the GAe would be the choice. If accuracy (the decision space) or sensitivity is the main concern, such as for tolerating manufacturing inaccuracies of a design-optimized product, then PSO or PS would be the choice. If the application is rather unknown, it is recommended that it could first be treated as a multimodal problem for global optimization using the GAe, which will take a longer time, and then, if desired, as a unimodal problem for local optimization using PC, which will be a lot quicker.

With the guide of the ranks and the decision tree, all the six optimizers have been applied to the nonlinear problem of particle filtering. The GAe is confirmed being better equipped to deal with such a relatively challenging optimization problem.

For future work, benchmarking and selection criteria will be extended to multi-target optimization problems.

REFERENCES

- [1] *The Mathworks, Company Overview*. Accessed: May 1, 2019. [Online]. Available: <https://www.mathworks.com/content/dam/mathworks/tag-team/Objects/c/company-fact-sheet-8282v18.pdf>
- [2] V. M. Lo, "Heuristic algorithms for task assignment in distributed systems," *IEEE Trans. Comput.*, vol. 37, no. 11, pp. 1384–1397, Nov. 1988.
- [3] A. R. Conn, K. Scheinberg, and L. N. Vicente, *Introduction to Derivative-Free Optimization*. Philadelphia, PA, USA: SIAM, 2009, p. 277.
- [4] M. J. D. Powell, "Direct search algorithms for optimization calculations," *Acta Numerica*, vol. 7, pp. 287–336, Nov. 1998.
- [5] Y. Sun, X. Wang, Y. Chen, and Z. Liu, "A modified whale optimization algorithm for large-scale global optimization problems," *Expert Syst. Appl.*, vol. 114, pp. 563–577, Dec. 2018.
- [6] O. Mersmann, M. Preuss, H. Trautmann, B. Bischl, and C. Weihs, "Analyzing the BBOB results by means of benchmarking concepts," *Evol. Comput.*, vol. 23, no. 1, pp. 161–185, 2015.
- [7] L. Li, Y. Chen, Q. Liu, J. Lazić, W. Luo, and Y. Li, "Benchmarking and evaluating MATLAB derivative-free optimisers for single-objective applications," in *Proc. ICIC*, Liverpool, U.K., 2017, pp. 75–88.
- [8] Q. Chen, B. Liu, Q. Zhang, J. Liang, P. Suganthan, and B. Qu, "Problem definitions and evaluation criteria for CEC 2015 special session on bound constrained single-objective computationally expensive numerical optimization," NTU, Singapore, Tech. Rep., 2014.
- [9] J. Liang, B. Qu, and P. Suganthan, "Problem definitions and evaluation criteria for the CEC 2014 special session and competition on single objective real-parameter numerical optimization," NTU, Singapore, Tech. Rep., 2013.
- [10] W. Feng, T. Brune, L. Chan, M. Chowdhury, C. Kuek, and Y. Li, "Benchmarks for testing evolutionary algorithms," in *Proc. Asia-Pacific Conf. Control Meas.*, GanSu, China, 1998, pp. 134–138.
- [11] K. C. Tan and Y. Li, "Performance-based control system design automation via evolutionary computing," *Eng. Appl. Artif. Intell.*, vol. 14, no. 4, pp. 473–486, 2001.
- [12] L. Li, C. Zhang, Z. Li, and Y. Li, "Particle filter with Lamarckian inheritance for nonlinear filtering," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Vancouver, BC, Canada, Jul. 2016, pp. 2852–2857.
- [13] *Powell on MathWorks*. Accessed: Mar. 29, 2017. [Online]. Available: <https://cn.mathworks.com/matlabcentral/fileexchange/15072-unconstrained-optimization-using-powell/content/powell.m>
- [14] Z.-H. Zhan, J. Zhang, Y. Li, and H. S.-H. Chung, "Adaptive particle swarm optimization," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 39, no. 6, pp. 1362–1381, Dec. 2009.
- [15] X. Yao, Y. Liu, and G. Lin, "Evolutionary programming made faster," *IEEE Trans. Evol. Comput.*, vol. 3, no. 2, pp. 82–102, Jul. 1999.
- [16] Z. Michalewicz, "Evolution strategies and other methods," in *Genetic Algorithms + Data Structures = Evolution Programs*, 1st ed. New York, NY, USA: Springer, 1996, pp. 159–177.
- [17] J.-M. Renders and H. Bersini, "Hybridizing genetic algorithms with hill-climbing methods for global optimization: Two possible ways," in *Proc. IEEE 1st Conf. Evol. Comput., IEEE World Congr. Comput. Intell.*, Orlando, FL, USA, Jun. 1994, pp. 312–317.
- [18] Q. Yang, W.-N. Chen, T. Gu, and H. Zhang, "Segment-based predominant learning swarm optimizer for large-scale optimization," *IEEE Trans. Cybern.*, vol. 47, no. 9, pp. 2896–2910, Sep. 2017.
- [19] G. E. P. Box, J. S. Hunter, and W. G. Hunter, *Statistics for Experimenters: Design, Innovation, and Discovery*, 2nd ed. New York, NY, USA: Wiley, 2005.
- [20] A. Banerjee and P. Burlina, "Efficient particle filtering via sparse kernel density estimation," *IEEE Trans. Image Process.*, vol. 19, no. 9, pp. 2480–2490, Sep. 2010.
- [21] N. J. Gordon, D. J. Salmond, and A. F. M. Smith, "Novel approach to nonlinear/non-Gaussian Bayesian state estimation," *IEE Proc. F (Radar Signal Process.)*, vol. 140, pp. 107–113, Apr. 1993.
- [22] C. Musso, N. Oudjane, and F. Le Gland, "Improving regularised particle filters," in *Sequential Monte Carlo Methods in Practice*. New York, NY, USA: Springer, 2001, pp. 247–271.
- [23] T. Higuchi, "Monte Carlo filter using the genetic algorithm operators," *J. Stat. Comput. Simul.*, vol. 59, no. 1, pp. 1–23, Feb. 1997.
- [24] N. M. Kwok, G. Fang, and W. Zhou, "Evolutionary particle filter: Resampling from the genetic algorithm perspective," in *Proc. IEEE/RSJ Int. Conf. IROS*, Edmonton, AB, Canada, Aug. 2005, pp. 2935–2940.



LIN LI (M'17) received the B.S. degree in electronic information engineering from Anhui Agriculture University, Hefei, China, in 2009, and the Ph.D. degree in signal information processing from Harbin Engineering University, Harbin, China, in 2016.

In 2015, she was a Postgraduate Visiting Researcher with the University of Glasgow, U.K. Since 2017, she has been a Postdoctoral Research Fellow with the Industry 4.0 Artificial Intelligence Laboratory, Dongguan University of Technology, Dongguan, China, and also with the South China University of Technology, Guangzhou, China. Her research interests include signal processing, target tracking, computational intelligence, optimization algorithms, and their applications.



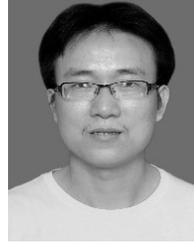
ALFREDO ALAN FLORES SALDIVAR was born in Saltillo, Coahuila, Mexico, in 1984. He received the B.Eng. degree in industrial and systems engineering from the Autonomous University of the North East, Coahuila, in 2009, the M.S. degree in advanced manufacturing systems from the Mexican Corporation of Materials Research, Saltillo, in 2013, and the Ph.D. degree in mechanical engineering from the University of Glasgow, Glasgow, U.K., in 2018.

He was with the Mahle Behr Group, Mexico, as a Logistic Department Assistant, in 2008. From 2009 to 2012, he was with the Daimler Trucks North America, Mexico, as a materials procurement of indirect materials for the facility in Saltillo. His research interests include predictive models for smart design and manufacture, machine learning applications on data analytics, optimization applied on manufacturing processes, statistical analysis, design of experiments, Industry 4.0 applications, and computer automated design.



YUN BAI (M'17) received the B.S. and M.S. degrees in environmental engineering from the Chongqing Technology and Business University, Chongqing, China, in 2008 and 2011, respectively, and the Ph.D. degree in civil engineering from Chongqing University, Chongqing, in 2014.

He is currently a Postdoctoral Research Fellow with the Dongguan University of Technology, Dongguan, China. His current research interests include intelligent computing and its applications to engineering system management, and reliability management.



QUNFENG LIU received the B.S. and M.S. degrees from the Huazhong University of Science and Technology, in 1999 and 2002, respectively, and the Ph.D. degree from Hunan University, China, in 2011.

He is currently an Associate Professor with the School of Computer Science and Network Security, Dongguan University of Technology, China. His current research interests include global optimization, evolutionary computation, and machine learning.



YUN LI (S'87–M'90–SM'17) received the B.S. degree in electronics science from Sichuan University, Chengdu, China, in 1984, the M.E. degree in electronic engineering from the Electronic Science and Technology of China (UESTC), Chengdu, China, in 1987, and the Ph.D. degree in parallel computing and control from the University of Strathclyde, Glasgow, U.K., in 1990.

In 1989, he was a Control Engineer with the U.K. National Engineering Laboratory. In 1990, he was a Postdoctoral Research Engineer with the Industrial Systems and Control Ltd. From 1991 to 2018, he was a Lecturer, Senior Lecturer, and Professor with the University of Glasgow and has served as its Founding Director of the University of Glasgow Singapore. He is currently a Professor with the Dongguan University of Technology, Dongguan, China, and also a part-time Professor with the University of Strathclyde, Glasgow, U.K. He has published over 260 papers, one of which is the most popular article every month in the IEEE TRANSACTIONS CONTROL SYSTEMS TECHNOLOGY and another the 3rd in the IEEE TRANSACTIONS SYSTEMS, MAN, AND CYBERNETICS - PART B: CYBERNETICS. He is the author of the popular interactive evolutionary algorithm (EA) courseware EA_demo (<http://14ai.org/EA-demo/>) published online first, in 1997. His current research interest includes artificial intelligence for Industry 4.0.

Dr. Li is an Associate Editor of the IEEE TRANSACTIONS EVOLUTIONARY COMPUTATIONAL INTELLIGENCE and of the IEEE TRANSACTIONS EMERGING TOPICS IN COMPUTATIONAL INTELLIGENCE. He is a Chartered Engineer and an FRSA in the U.K., and has chaired the EPSRC funded first Industrial Systems in the Digital Age Conference - "Looking Beyond Industry 4.0" held in Glasgow, in 2017.

...



YI CHEN (M'10–SM'17) received the B.Sc. degree from Chongqing Engineering University, China, the M.Sc. degree from Chongqing University, China, and the Ph.D. degree from the University of Glasgow, U.K.

His research interests are in artificial intelligence, high-performance computing, evolutionary computation, robotic systems, and automation. He is one of the editorial board members of a number of international journals and has been one of the guest editors for five special issues.

Dr. Chen is a member of the IMechE, AAAI, IET, AIAA, and ASME, and a fellow of the HEA. He is also a Chartered Engineer in the U.K.