Soft Computing
https://doi.org/10.1007/s00500-024-09748-2

OPTIMIZATION

A critical take on the role of random and local search-oriented components of modern computational intelligence-based optimization algorithms

Babak Zolghadr-Asli[1,2]

Accepted: 5 February 2024

Abstract
The concept of computational intelligence (CI)-based optimization algorithms emerged in the early 1960s as a more practical approach to the contemporary derivate-based approaches. This paved the way for many modern algorithms to arise with an unprecedented growth rate in recent years, each claiming to have a novel and present a profound breakthrough in the field. That said, many have raised concerns about the performance of these algorithms and even identified fundamental flaws that could potentially undermine the integrity of their results. On that note, the premise of this study was to replicate some of the more prevalent, fundamental components of these algorithms in an abstract format as a measure to observe their behavior in an isolated environment. Six pseudo algorithms were designed to create a spectrum of intelligence behavior ranging from absolute randomness to local search-oriented computational architecture. These were then used to solve a set of centered and non-centered benchmark suites to see if statistically different patterns would emerge. The obtained result clearly highlighted that the algorithm's performance would suffer significantly as these benchmarks got more intricate. This is not just in terms of the number of dimensions in the search space but also the mathematical structure of the benchmark. The implication is that, in some cases, sheer processing resources can mask the algorithm's lack of sufficient intelligence. But as importantly, this study attempted to identify some mechanics and concepts that could potentially cause or amplify this problem. For instance, the excessive use of greedy strategy, a prevalent measure embedded in many modern CI-based algorithms, has been identified as potentially one of these reasons. The result, however, highlights a more fundamental problem in the CI-based optimization field. That is, these algorithms are often treated as a black box. This perception cultivated the culture of not exploring the underlying structure of these algorithms as long as they were deemed capable of generating acceptable results, which permits similar biases to go undetected.

Keywords Computation intelligence-based optimization · Metaheuristic optimization · Evolutionary algorithms · Swarm intelligence · Computational intelligence · Optimization

1 Introduction

In the early 1960s, the concept of computational intelligence (CI)-based methodologies arose as a more practical alternative to more traditional derivate-based approaches to mathematical programming, also known as optimization (Zolghadr-Asli 2023a). These were, in essence, providing an algorithmic guided sampling-based computational architecture that was, at least in theory, not restricted by problems rooted in high dimensionality, multimodality, epistasis, non-differentiability, and discontinuous search space imposed by constraints (Yang 2010; Du and Swamy 2016; Bozorg-Haddad et al. 2017). Though known by different names, such as metaheuristic optimization algorithms, a term coined in the 1980s (Glover 1986), the main idea behind these algorithms is direct intelligence-like sampling. In general, these algorithms would attempt to take samples from the search space, whether via one or

✉ Babak Zolghadr-Asli
b.zolghadrasli@uq.net.au; bz267@exeter.ac.uk

[1] Sustainable Minerals Institute, The University of Queensland, Brisbane, Australia

[2] The Centre for Water Systems, The University of Exeter, Exeter, UK

Published online: 05 July 2024

multiple search agents, evaluate the said sample against the objective function, and use this information within an intelligent-like computational structure to update the positions of the said agents. This process would be repeated until the algorithm reached a termination criterion, at which point the algorithm is expected to converge to the solution. The caveat is that the emerging result is often a near-optimum solution rather than a global one. In fact, there is even no guarantee that the algorithm has converged or that it has instead got trapped in a local optimum. These issues, however, can often be addressed by fine-tuning the parameters of the algorithms.

This concept has gained traction in recent years, inspiring many "novel" algorithms. Over 260 algorithms were introduced between 1990 and 2021, according to the Evolutionary Computation Bestiary dataset, a known repository that documents novel CI-based optimization algorithms (Campelo and Aranha 2021). What justifies this sheer number of supposedly novel algorithms is the widely known principle of the no-free-lunch theorem (NFLT) (Wolpert and Macready 1997). The NFLT is an impossibility theorem stating that a general-purpose, universal strategy can never exist in perpetuity (Gómez and Rojas 2016; Zolghadr-Asli 2023b).

Although the aim of this research is not to undermine the widely accepted nature of this field or its contemporary interpretations, and the abundance of seemingly novel algorithms is not inherently problematic, it prompts the inquiry into whether these purportedly "novel" and "inspired" algorithms yield tangible and meaningful breakthroughs. On that note, there is a school of thought that takes a solid dissenting stand against some of the modern algorithms by questioning the integrity of these new algorithms and their reliability in the face of actual problems (Tzanetos and Dounias 2021; Aranha et al. 2022; Camacho-Villalón et al. 2022; Velasco et al. 2024). As pointed out by such studies, a recognized documented recurring computational issue with some of these modern algorithms, for instance, is their tendency to gravitate toward the center of the search space, a problem often referred to as "central bias" (Kudela 2022). In certain cases, the skeptics even call into question the novelty of these ideas, going as far as to say that in some of these algorithms, the claimed "novelty" is primarily limited to the naming conventions rather than offering a genuinely new computational architecture (Sörensen 2015; Fong et al. 2016). However, evidence suggests this seems to be a more recent trending problem, as demonstrated by Kudela (2023), where 47 out of the 90 examined "novel" algorithms introduced between 1987 and 2022 have, to some degree, showcased this shortcoming. According to the findings of similar studies (i.e., Kudela 2022), some of the most extensively cited classical benchmark problems are built so that they are fundamentally incapable of flagging such issues.

A fundamental issue with the use of such algorithms is that, like many other data-driven models, it has become an unspoken standard to consider these as a *black box*. The implication of this black box label is that it permeates the user not to have an explicit understanding of how these algorithms work so long as the generated results fall within the conventional range of what is expected from such models. In turn, not only errors like central bias would go undiscovered, but no changes can be made toward a more robust algorithm because there is no reliable feedback on how the algorithm operates.

On that note, this paper tends to take a closer and more critical look at the underlying computational structure of these algorithms to identify the reason behind the documented bias. Ultimately this could equip us with a more in-depth understanding of why and, as importantly, how some of these algorithms perform virtually perfectly under certain conditions and yet fail to handle other computationally complex problems. In that spirit, five pseudo algorithms were developed to represent a continuum, ranging from absolute randomness to some variation of computational intelligence. The inspiration behind these is the components prevalent in some modern computational intelligence-based optimization algorithms that have displayed this bias. These were then used to solve two sets of optimization problems, one with an optimum solution set at the center of the search space and another set in which the optimum solutions have been relocated from the center. Each pseudo algorithm was then executed on numerous independent runs under different setups to see if a statistically significant pattern would emerge to signal a problem with any of the said mechanisms often incorporated in some modern computational intelligence-based optimization algorithms.

2 Materials and methods

To unveil how CI-based algorithms gravitate toward what they perceive to be the optimum solution, these algorithms were first stripped down to their core mechanisms, and, in turn, some of the more prevalent modules that can be found in these algorithms were selected for further evaluation. Each isolated module was then represented via a pseudo algorithm, where some were governed by pure randomness while the rest were geared toward a degree of intelligence that can be found in the computational architecture of the said algorithms. As such, this would create a spectrum of relative intelligence in the sense of how much information will be incorporated into how these algorithms navigate the search agents within the search space.

But the other critical factor suggested to have a heavy influence on the performance of these algorithms is the mathematical structure of the benchmark problems. On that note, two sets of benchmark problems were to be considered in this study, one having the solutions in the center of the search space, while the other tried to use rotation and shifting operators to separate the optimum solution from the central point of the search space. The following subsection will describe the architecture of these pseudo algorithms, the selected benchmark problems, and finally, the setup for this study.

2.1 Pseudo optimization algorithms

In this study, six pseudo optimization algorithms were designed to objectively measure how some of the modern CI-based algorithms converge to what they perceive to be the optimum solution. However, the architecture of these pseudo algorithms was designed to test the effect of two critical ideas that often manifest themselves in some modern CI-based optimization algorithms: (1) the greedy strategy and (2) the idea of a local search-derived module.

The greedy strategy is a scheme under which the algorithm permits the repositioning of a search agent if, and only if, the new position yields a better result in terms of the objective function. This strategy, frequently used to enhance the performance of established CI-based optimization (Yaghoubzadeh-Bavandpour et al. 2022), ensures that the performance of agents equipped with this strategy remains consistent or improves during the searching process. The caveat of leaning heavily on this strategy is that it would also increase the risk of being trapped in local optima (Zolghadr-Asli 2023a). That said, many modern CI-based optimization algorithms are initially embedded with this strategy. Notable examples would be the shuffled frog-leaping algorithm (Eusuff et al. 2006), the invasive weed optimization algorithm (Mehrabian and Lucas 2006), the plant propagation algorithm (Salhi and Fraga 2011), the teaching–learning-based optimization algorithm (Rao et al. 2011), the bat algorithm (Yang and Gandomi 2012), the flower pollination algorithm (Yang 2012), the water cycle algorithm (Eskandar et al. 2012), the symbiotic organisms search algorithm (Cheng and Prayogo 2014), to name a few.

The other idea that would be put to the test here is the effectiveness of local search-oriented modules. At its core, these types of modules use what has been identified as the best-encountered position until that point in time within the search process to realign all or a portion of the search agents. On paper, these should expedite the convergence rate of an algorithm, while leaning heavily on this mechanism would potentially exacerbate the risk of being trapped in local optima. Mathematically, the implementation of this idea can be expressed as follows:

$$R_j = \text{rand} \times vec \quad \forall j, \tag{1}$$

$$\text{diff}_j = xbest_j - x_j \quad \forall j, \tag{2}$$

$$x'_j = x_j + \left(\text{diff}_j \times R_j\right) \quad \forall j, \tag{3}$$

where rand is a randomly generated number between 0 and 1 under a uniform distribution, *vec* denotes the length of a vector that would be multiplied by the random number, R_j is the elongated random vector in the *j*th dimension of the search space, x_j represents the position of the search agent in the *j*th dimension, $xbest_j$ denotes the corresponding value of the best-encountered part during the search in the *j*th dimension, diff_j represent the component of the vector connecting the current position of the search space to the best-encountered point in the *j*th dimension, and finally, x'_j is the updated location of the search agent in the *j*th dimensions of the search space. It should be noted that here *vec*, from a computational standpoint, is performing as a parameter through which one can modify and control the behavior of the said module. Again, many modern CI-based optimization algorithms are equipped with similar mechanisms. The cat swarm optimization (Chu et al. 2006), the krill herd algorithm (Gandomi and Alavi 2012), the grey wolf optimizer (Mirjalili et al. 2014), and the moth-flame optimization algorithm (Mirjalili 2015) are just merely few examples in this category.

As for the pseudo algorithms, this study deliberately uses the idea of multi-search agent structures in appose to single search agent ones, as the former is considered far more effective from a computational standpoint (Zolghadr-Asli 2023a, b). And on that note, two general architectures are used here to construct these pseudo optimization algorithms. The first architecture is geared toward pure random search (Fig. 1), while the second one is designed with the idea of local search in mind (Fig. 2). In both cases, however, a set of search agents, the number of which is denoted by a parameter called *pop size*, would be placed

Begin
Position all the search agents
While the termination criterion is not met
Generate a new set of agents
Evaluate all the agents
If the *greedy strategy* is considered:
Compare both sets and select the best counterpart in the paired set
Else:
Create a pool based on previous and current agents
Select the new set out of the best agents in the current pool
End While
Report the best solution
End

Fig. 1 Pseudo code for the computational structure of the random-oriented architecture

```
Begin
Position all the search agents
While the termination criterion is not met
Identify the best position encountered thus far
Generate a new set of agents using the local optimum mechanism
Evaluate all the agent
If the greedy strategy is considered:
Compare both sets and select the best counterpart in the paired set
Else:
Create a pool based on previous and current agents
Select the new set out of the best agents in the current pool
End While
Report the best solution
End
```

Fig. 2 Pseudo code for the computational structure of the local search-oriented architecture

randomly within the feasible boundaries of the search space. Then the position of these agents would be upgraded repeatedly via the instructions dictated by each given pseudo algorithm. This iterative process would be terminated until the process has been repeated a certain number of times, another parameter here denoted by *max iteration*. Based on the described architectures, six pseudo optimization algorithms were defined, a description of which is summarized in Table 1.

2.2 Benchmark problems

As stated, one of the central premises of this study was to test the effect of the relative position of the optimum solution within the search space on some of the more prevalent and fundamental components of CI-based optimization algorithms. This subject is implicitly correlated to the number of decision variables denoted by n. From a mathematical standpoint, this represents the number of dimensions considered for the search space. To cover this matter, all the potential benchmark problems will be evaluated under two conditions, one with a more abstract search space ($n = 2$) and a more intricate one ($n = 10$). The sub-question that can be tested through this setup is to see whether the intricacy of the search space has a meaningful contribution to the performance of these pseudo algorithms.

As for the benchmarks, two sets of objective functions are to be considered in this study, one serving as the non-centered (NC) set and the other as the centered (C) one. The idea is to compare the obtained results to see if there are any statically significant changes in the performance of these pseudo algorithms as they try to tackle each set separately. For the NC benchmarks, a subset of the 2017 version of the IEEE Congress on Evolutionary Computation (CEC2017) suite has been selected (Wu et al. 2017). This is a well-regarded pre-defined packet of benchmarks designed deliberately to relocate the optimum solution from the center of the search space as a measure to put the CI-based optimization algorithm to the test. The selected subset includes 17 objective functions [$f_1, f_2, f_3, f_4, f_5, f_6, f_7, f_8, f_9, f_{10}, f_{21}, f_{23}, f_{24}, f_{25}, f_{26}, f_{27}$, and f_{28}]. The value of the optimum solution is directly correlated with the code number of the objective function in the suite (i.e., the number multiplied by 100 is the value of the optimum objective in the CEC2017 suite). As to why these specific functions were selected, it is simply because the rest were incompatible with the premise of this research, as they could not handle certain dimensions that were to be considered in this study. Readers are encouraged to visit Wu et al. (2017) for more information on this benchmark suite. For illustrative purposes, Fig. 3 depicts the shape of the NC benchmark suite's objective functions in a two-dimensional search space ($n = 2$). As for the C benchmark suite, five well-known objective functions have been selected, in which the optimums solutions are in the center of the search space. The definition of these functions is summarized in Table 2. For illustrative purposes, Fig. 4 depicts the shape of the C benchmark suite's objective functions in a two-dimensional search space ($n = 2$).

2.3 The premise of the research setup

Generally, it is well-understood that the performance of the CI-based optimization algorithm is strongly correlated to

Table 1 Description of devised pseudo optimization algorithms

Pseudo optimization algorithms		Acronym	Greedy strategy	Underlying architecture	Parameter of architecture
1	Pure Random	PR	✗	random-oriented	[N/A]
2	Pure Random Greedy	PRG	✔	random-oriented	[N/A]
3	Local search 0.5	LS.5	✗	local search-oriented	$vec = 0.5$
4	Local search 1.0	LS1	✗	local search-oriented	$vec = 1.0$
5	Local search greedy 0.5	LSG.5	✔	local search-oriented	$vec = 0.5$
6	Local search greedy 1.0	LSG1	✔	local search-oriented	$vec = 1.0$

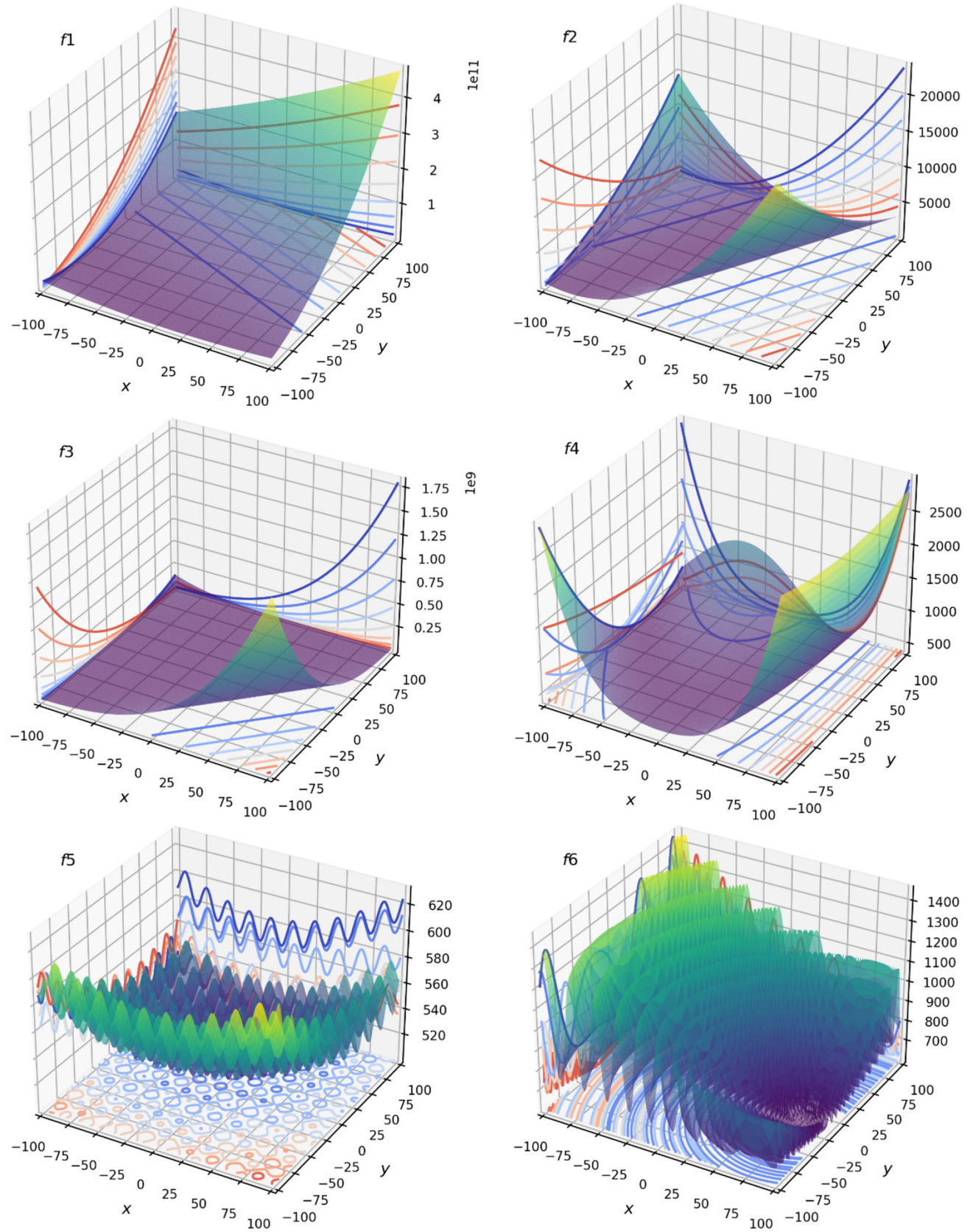

Fig. 3 Depiction of the NC benchmark suite search space in a two-dimensional space ($n = 2$)

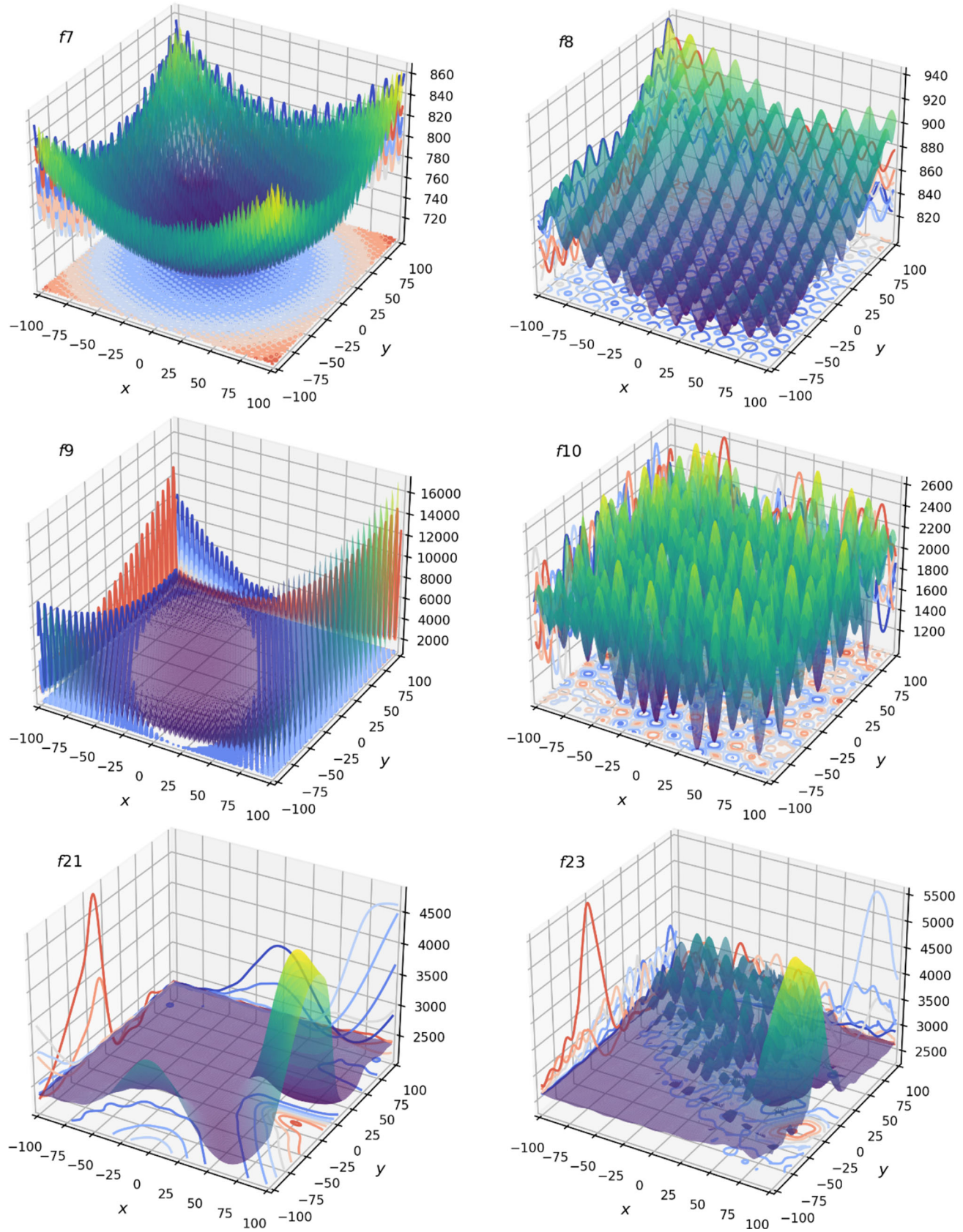

Fig. 3 continued

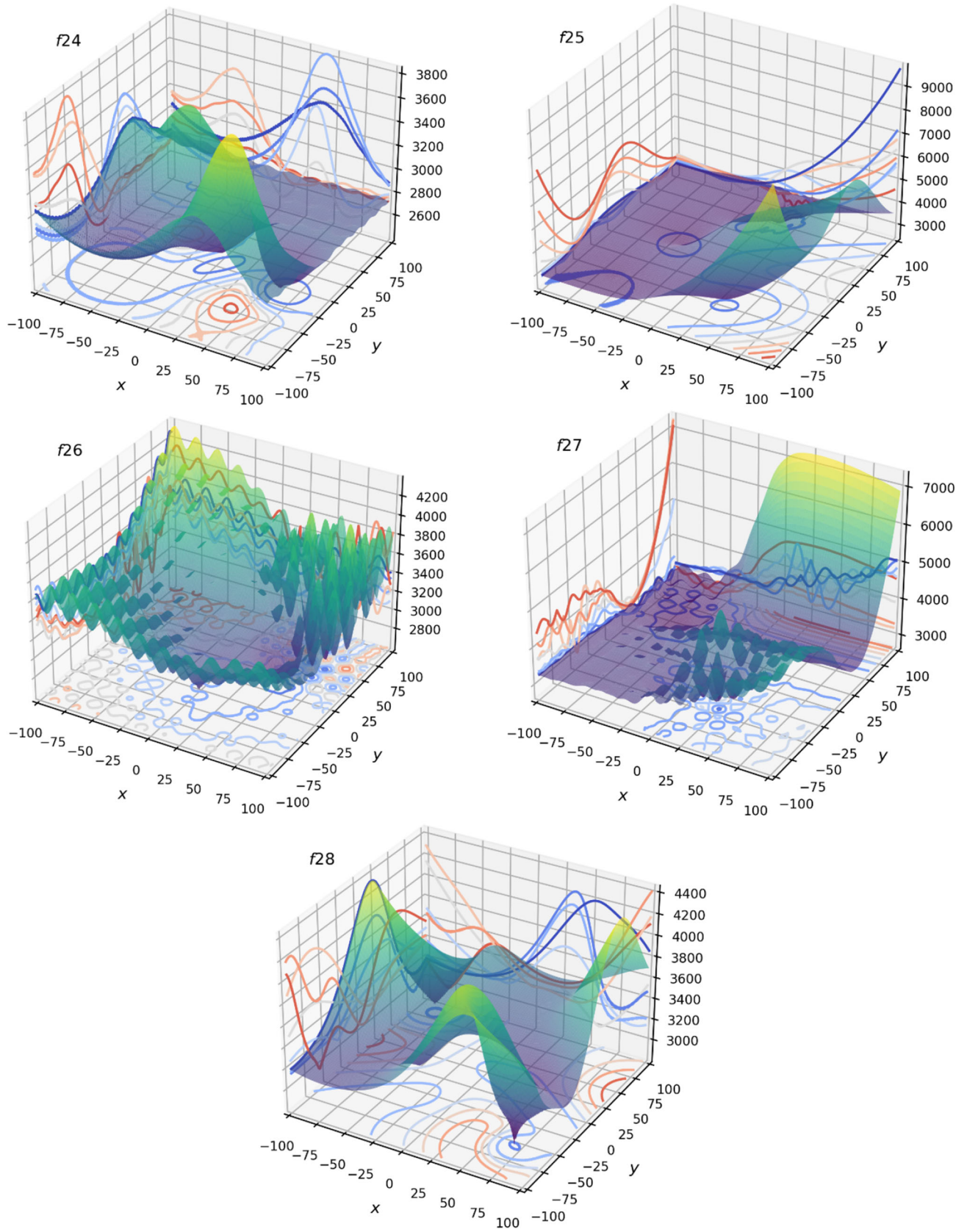

Fig. 3 continued

Table 2 A detailed mathematical description of the C benchmark suite

Function code	Function definition	Feasible domain	Global optima
f_1	$\sum_{i=1}^{N} x_i^2$	[− 100, 100]	0.00
f_2	$\sum_{i=1}^{N} \lvert x_i \rvert + \prod_{i=1}^{N} \lvert x_i \rvert$	[− 10, 10]	0.00
f_3	$\max_i \{\lvert x_i \rvert \lvert \forall i\}$	[− 100, 100]	0.00
f_4	$\sum_{i=1}^{N} \lfloor x_i + 0.5 \rfloor^2$	[− 100, 100]	0.00
f_5	$\sum_{i=1}^{N} \left(i \times x_i^4\right)$	[− 1.28, 1.28]	0.00

the number of function evaluations (NFE), which is partially reflected by the number of search agents and maximum permitted iteration. As the premise of this study is based upon multi-agent CI-based optimization algorithms, it was essential to ensure that the effect of NFE on the performance of these algorithms is monitored as well. On that note, as the NFE in all the designed pseudo algorithms is directly linked to the *pop size* and *max iteration*, a set of values was considered for each parameter in this study. It is worth noting a concept explored in recent studies, which highlights that certain parameters, such as the number of search agents, may possess greater potential to influence overall performance in comparison to other crucial factors, such as the number of iterations, in the implementation of CI-based algorithms (Velasco et al. 2022). On that note, for *pop size,* these values were selected from {50; 100; 200; 500; 750; 1000}, while for the *max iteration* parameter, the set of considered values was {500; 750; 1000; 2000; 2500}. As all the potential permutations of these two sets were considered, this would create 30 combinations with different NFE values, depicted in Fig. 5. This graph illustrates the spectrum of NFEs under examination in this study, along with the corresponding parameters for each setting. This analysis aims to elucidate the potential significance of each parameter in the subsequent discussion.

It should be noted that the devised pseudo algorithms were tested for each benchmark problem (i.e., both NC and C suites, each under $n = 2$ and $n = 10$ setups) under all these possible NFE combinations. Conventionally, as these are stochastic-based methods, each setup would be executed multiple times to get a better sense of the most likely behavior of the algorithm rather than potentially extreme cases. On that note, each setup is needed to be executed independently 100 times. All in all, this would result in 792,000 independent runs. The results would then be evaluated via statistical analysis and non-parametric statistical tests (e.g., Wilcoxon and Friedman). The Wilcoxon signed-rank test and Friedman test are both non-parametric statistical methods. The former is employed to compare the locations of two populations using two matched samples, while the latter is designed to establish a ranking for a set of matched samples. In this context, the Wilcoxon signed-rank test has traditionally been applied to assess the performance of two specific algorithms in relation to each other. On the other hand, the Friedman test is utilized to generate an overall ranking for the algorithms subjected to testing. The p value used in all tested in this study is 0.05. For more information on these standard procedures, the readers are referred to Derrac et al. (2011).

As a final note on the premise of this study and its objectives, it is essential to remember that the idea here is *not* to test whether the pseudo optimization algorithms can converge to the final solution on their own but rather to compare their relative performance, not just against one another, but as importantly under different setups. The point is that the aforementioned setups are designed deliberately to challenge the underlying core principles of these pseudo algorithms and, in turn, potentially unveil additional information about how, in effect, the CI-based optimization algorithms operate.

3 Result and discussion

As the first step to analyzing the results, it is worth exploring how these pseudo algorithms perform under different NFE values. For instance, Fig. 6 plots the progression of converged solution (NC—f_4) for LSG.5 ($n = 10$) against different NFE values. This graph shows NFE, as one would suspect, substantially impacts the algorithm's performance. But interestingly, the pattern observed in this graph also suggests that the *pop size* parameter may have a more decisive influence than the *max iteration* parameter in the algorithm's performance, most notably in lower NFE values. This implies that gaining access to certain information by the pseudo algorithm

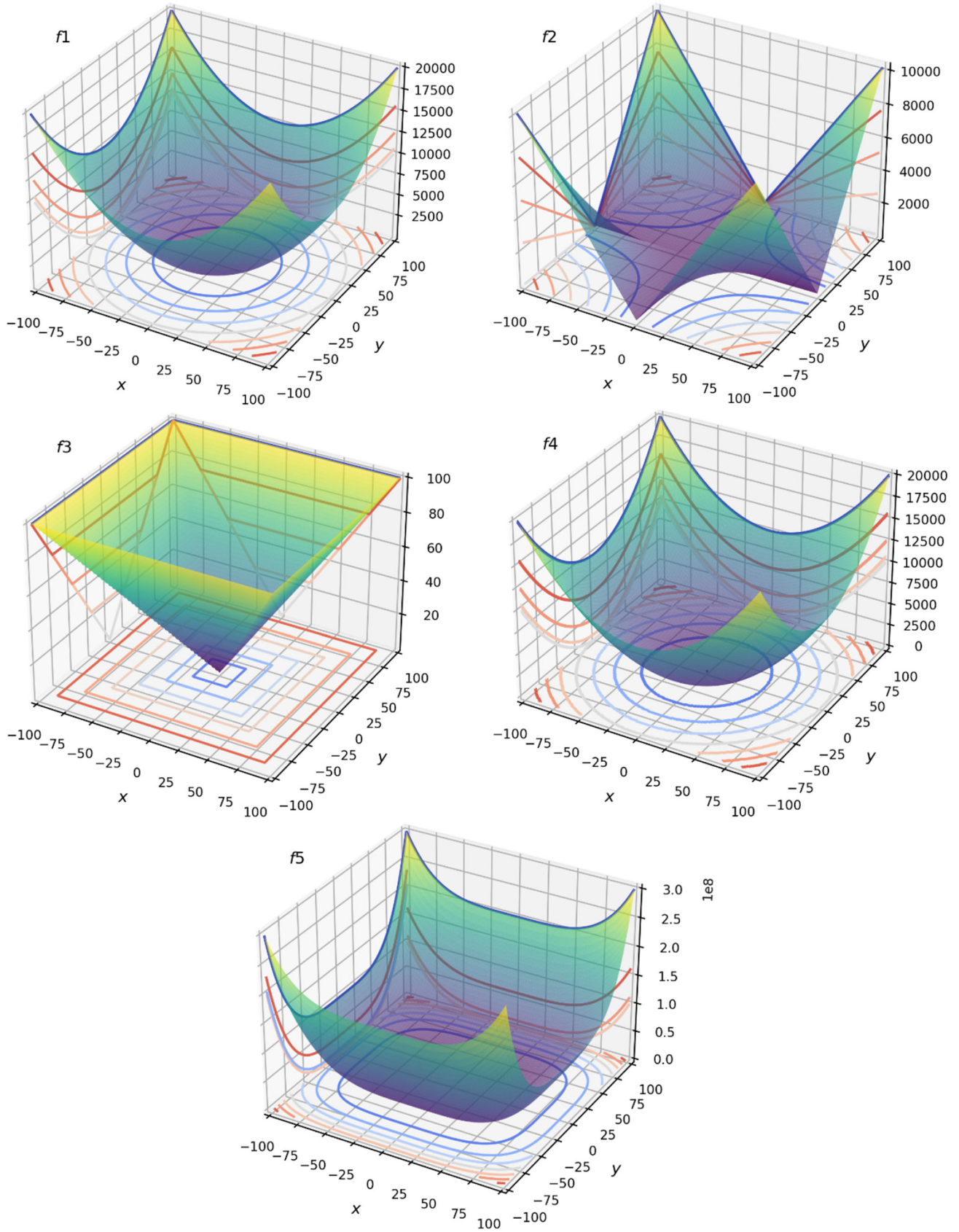

Fig. 4 Depiction of the C benchmark suite search space in a two-dimensional space ($n = 2$)

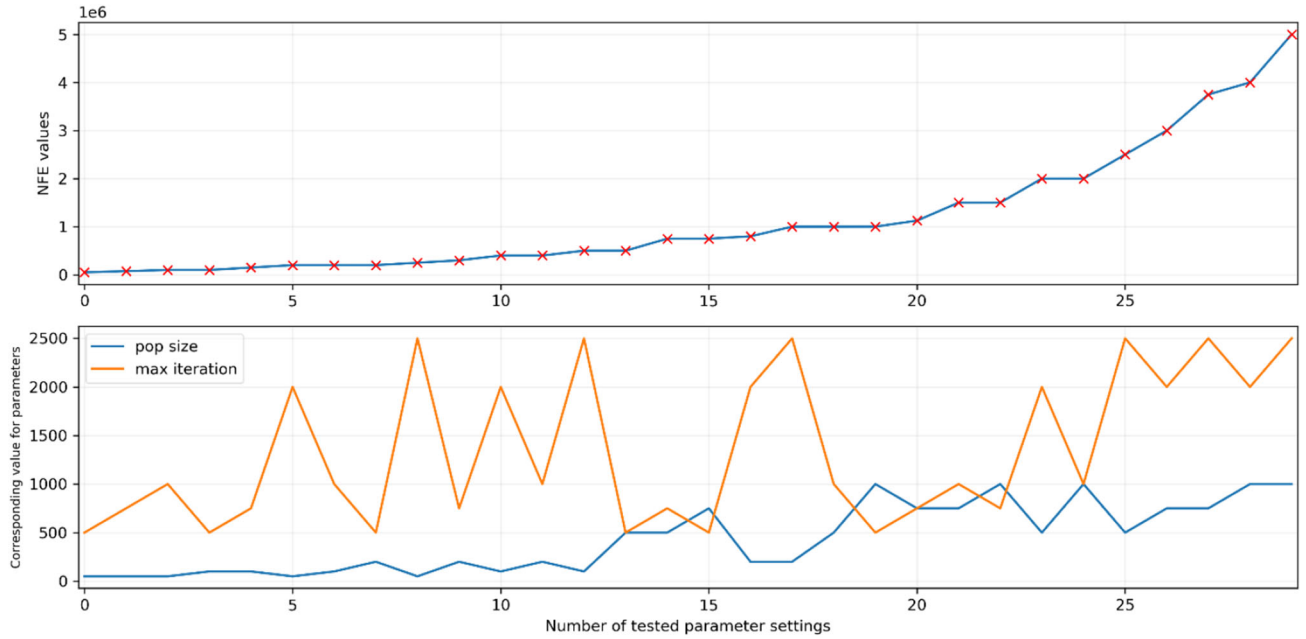

Fig. 5 The relationship between *pop size* and *max iteration* parameters and NFE

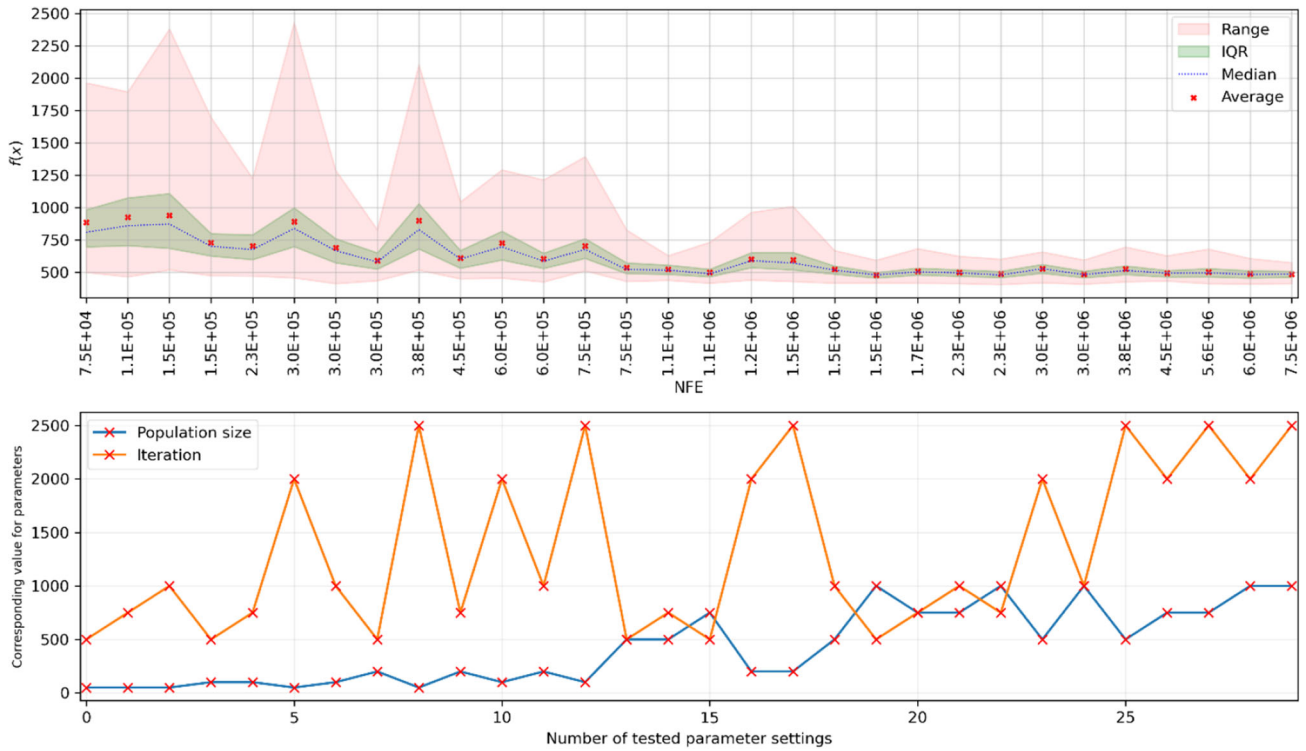

Fig. 6 The converged solution (NC—f_4) for LSG.5 (n = 10) under different NFE values

earlier in the searching process could be more beneficial, which is why weighing toward increasing the number of search agents than increasing the number of iterations is seemingly more advantageous for obtaining relatively better results. This is in line with what has been explored by Velasco et al. (2022). That said, it should be noted that generalizing the observed patterns beyond the tested abstract pseudo optimization algorithms would require further in-depth investigation to shed light on the

Table 3 Performance of different pseudo algorithms against C benchmark suite

		Best pseudo algorithms	Best obtained result	The mean of different pseudo algorithms	Standard deviation of different pseudo algorithms
$n = 2$	f_1	LSG1	0.00E+00	1.73E−03	2.68E−03
	f_2	LSG1	0.00E+00	2.70E−03	4.19E−03
	f_3	LSG1	0.00E+00	1.92E−02	2.97E−02
	f_4	*all*	0.00E+00	0.00E+00	0.00E+00
	f_5	LSG.5, LSG1	0.00E+00	6.02E−13	9.55E−13
$n = 10$	f_1	LSG.5	1.06E+00	5.66E+02	8.24E+02
	f_2	LSG.5	5.59E−01	4.35E+00	4.34E+00
	f_3	LSG.5	8.97E−01	9.33E+00	9.81E+00
	f_4	LSG.5	2.28E+00	5.57E+02	8.16E+02
	f_5	LSG.5	1.15E−06	2.41E−02	3.72E−02

importance and effect of these parameters on the overall performance of CI-based optimization algorithms.

As stated, the main research question here was to compare the performance of these pseudo algorithms as they handle C and NC benchmark problems. If this leads to statistically different patterns, this could potentially showcase how some modern CI-based algorithms may have some central bias tendencies. On that note, the performance of each pseudo algorithm was considered and compared under the highest NFE value (i.e., *pop size*: 1000 and *max iteration*: 2500 resulting in an NFE value of 7.5×10^6).

Table 3 summarizes the result of different pseudo algorithms against the C benchmark suite. If the search space consists of only two variables (i.e., $n = 2$), all pseudo algorithms, including the PR and PRG, which are random sampling, were able to converge to the optimum solution under the highest NFE value. That said, for the absolute best average performance under 100 independent runs, LSG1 showed a slight edge over the other alternatives (Table 3). These results can also undergo verification through a non-parametric Friedman test, the summary of which is available in Table 4. This table represents an overall ranking for the tested algorithms based on their paired samples. Interestingly, for f_4 of the C suite, the obtained result bears no statistically significant difference between the pseudo algorithms. A non-parametric Wilcoxon test can further validate this, the result of which are

Table 4 The results for the non-parametric Friedman test against the C benchmark suite

		f_1	f_2	f_3	f_4	f_5	Overall Rank
$n = 2$	PR	6	5	5	1	6	5
	PRG	5	6	6	1	5	5
	LS.5	4	4	4	1	4	4
	LS1	3	3	3	1	3	3
	LSG.5	2	2	2	1	1	2
	LSG1	1	1	1	1	1	1
$n = 10$	PR	5	6	5	6	6	6
	PRG	6	5	6	5	5	5
	LS.5	2	2	2	2	2	2
	LS1	4	4	4	4	4	4
	LSG.5	1	1	1	1	1	1
	LSG1	3	3	3	3	3	3

Table 5 The results for the non-parametric Wilcoxon test against the C benchmark suite

	Target	Function	PR	PRG	LS.5	LS1	LSG.5
$n = 2$	LSG1	f_1	−	−	−	−	−
		f_2	−	−	−	−	−
		f_3	−	−	−	−	−
		f_4	0	0	0	0	0
		f_5	−	−	−	−	0
			PR	PRG	LS.5	LS1	LSG1
$n = 10$	LSG.5	f_1	−	−	−	−	−
		f_2	−	−	−	−	−
		f_3	−	−	−	−	−
		f_4	−	−	−	−	−
		f_5	−	−	−	−	−

"−" signifies an inferior performance to the target; "+" means a superior performance to the target; and "0" identifies no statistically different performance to the target

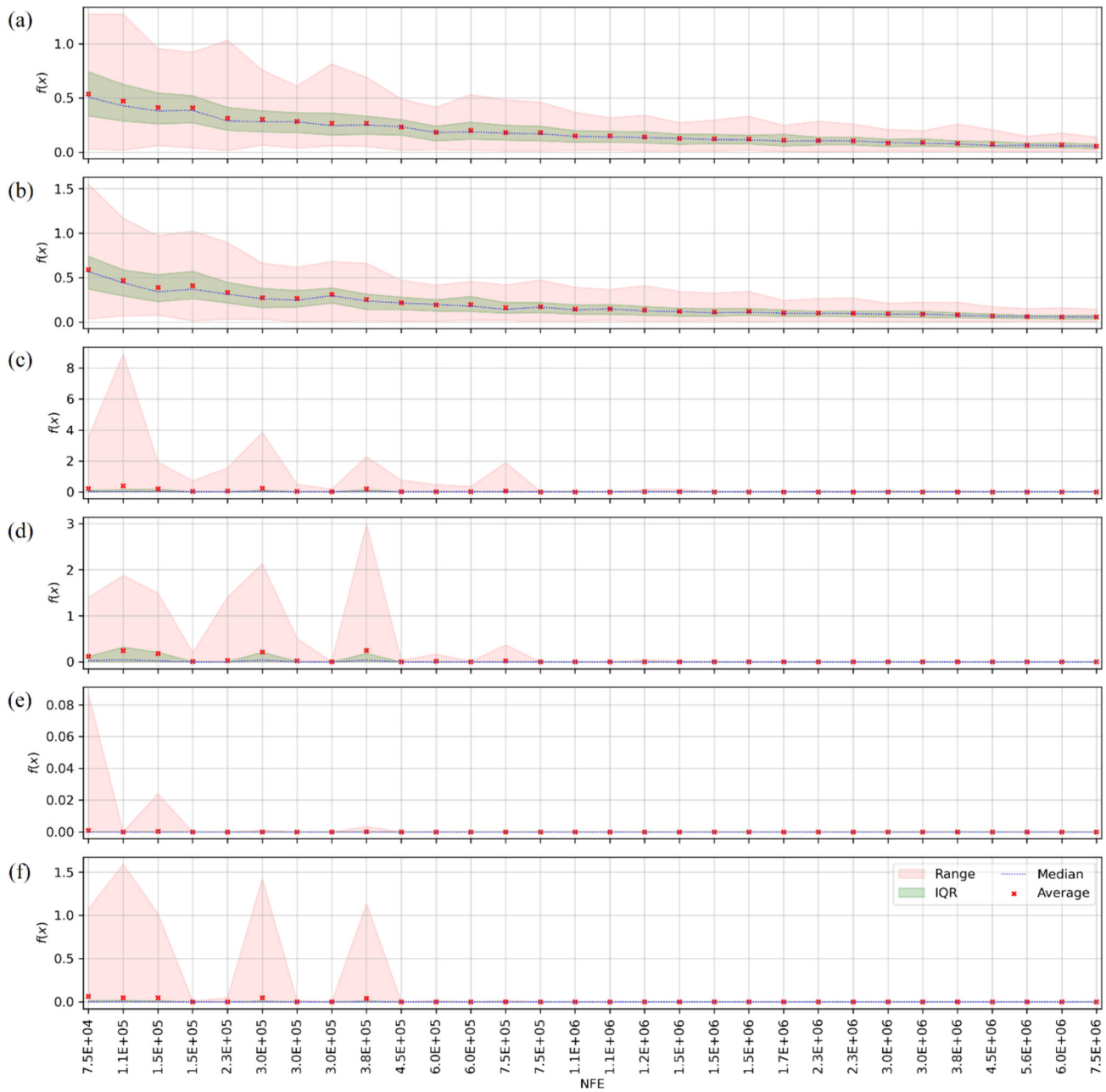

Fig. 7 Comparison of the average convergence rate of different pseudo algorithms for the f_3 function of the C benchmark suite ($n = 2$): **a** PR, **b** PRG, **c** LS.5, **d** LS1, **e** LSG.5, and **f** LSG1

summarized in Table 5. This would compare the relative performance of any two given algorithms based on their matched samples. These results highlight how insufficient intricacies of benchmark problems can easily mask the fallacies in the architecture of CI-based optimization algorithms by simply cranking up their processing power. In the ten-dimensional search space (i.e., $n = 10$), for the same benchmark suite, LSG.5 outperformed all other pseudo algorithms and, in some cases (e.g., f_5), nearly converged to the global optimum solution (Table 3). This finding can also be validated using a non-parametric Friedman test summarized in Table 4. Here LSG.5, followed by LS.5 showed the best relative performance, which considering the type of these pseudo algorithms and the value for the *vec* parameter, signifies that having a more prominent local search component may have played a role in favor of these selected algorithms. This is also supported by the results obtained from a non-parametric Wilcoxon test, with LSG.5 as the target performance reported in Table 5.

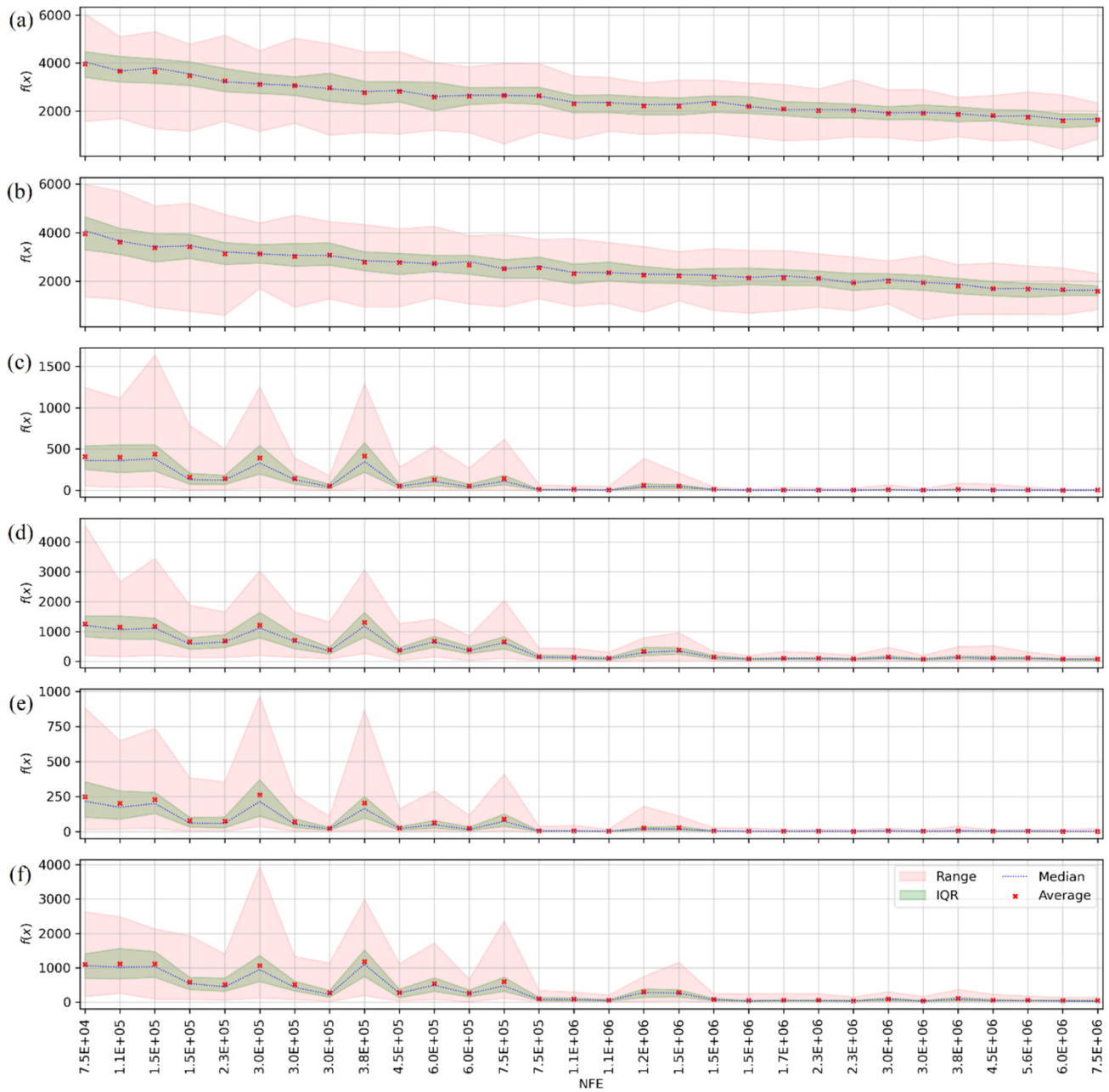

Fig. 8 Comparison of the average convergence rate of different pseudo algorithms for the f_4 function of the C benchmark suite (n = 10): **a** PR, **b** PRG, **c** LS.5, **d** LS1, **e** LSG.5, and **f** LSG1

Another approach to evaluate the performance of these pseudo algorithms is to compare their average convergence rate with different NFE values. Figure 7, for instance, depicts the average convergence rate of pseudo algorithms for the f_3 function of the C benchmark suite in a two-dimensional search space. Here at certain NFE values onward, all variations of LS and LSG pseudo algorithms seemed to be able to converge to the optimum solutions. Figure 8, on the other hand, shows the average convergence rate of pseudo algorithms for the f_4 function of the C benchmark suite in a ten-dimensional search space, which, as stated, seems to pose more computational challenges for the algorithms. Again, in addition to previously identified patterns for LS- and LSG-based pseudo algorithms, a prominent downward trend can be seen in PR and PRG graphs. This suggests that, on paper, by increasing NFE values, which in this particular setup means taking more samples, even these purely random-based pseudo algorithms could potentially converge to the optimum solution. However, as shown in these graphs, incorporating a slight

Table 6 Performance of different pseudo algorithms against the NC benchmark suite

		Best pseudo algorithms	Best obtained result	The mean of different pseudo algorithms	Standard deviation of different pseudo algorithms
$n = 2$	f_1	PR	1.15E+02	8.36E+02	5.76E+02
	f_2	LSG.5	2.00E+02	2.00E+02	1.57E−02
	f_3	LS1, LSG.5, LSG1	3.00E+02	3.00E+02	3.96E−03
	f_4	LS.5	4.00E+02	4.00E+02	1.19E−05
	f_5	LS.5	5.00E+02	5.00E+02	1.18E−02
	f_6	LSG.5	6.00E+02	6.00E+02	5.16E−02
	f_7	PRG	7.00E+02	7.01E+02	5.08E−01
	f_8	LS.5	8.00E+02	8.00E+02	2.37E−02
	f_9	LS1, LSG.5, LSG1	9.00E+02	9.00E+02	7.21E−04
	f_{10}	PR	1.00E+03	1.00E+03	7.17E−01
	f_{21}	LSG.5	2.10E+03	2.10E+03	5.32E−02
	f_{23}	LSG.5	2.30E+03	2.30E+03	3.82E−01
	f_{24}	LSG.5	2.40E+03	2.40E+03	6.63E−01
	f_{25}	PRG	2.50E+03	2.53E+03	3.11E+01
	f_{26}	LSG.5	2.60E+03	2.60E+03	4.83E−01
	f_{27}	LS1	2.70E+03	2.70E+03	9.72E−01
	f_{28}	PRG	2.81E+03	2.83E+03	2.29E+01
$n = 10$	f_1	LS.5	7.87E+09	1.21E+10	2.88E+09
	f_2	PR	2.75E+07	1.20E+08	1.38E+08
	f_3	LS.5	3.17E+03	4.62E+03	1.02E+03
	f_4	LS.5	4.73E+02	4.93E+02	1.47E+01
	f_5	LS.5	5.28E+02	5.40E+02	9.10E+00
	f_6	LS.5	6.14E+02	6.25E+02	6.44E+00
	f_7	LSG.5	7.31E+02	7.70E+02	4.43E+01
	f_8	LSG.5	8.18E+02	8.33E+02	1.33E+01
	f_9	LS.5	9.23E+02	1.10E+03	1.81E+02
	f_{10}	PR	2.08E+03	2.17E+03	7.85E+01
	f_{21}	PRG	2.21E+03	2.24E+03	1.84E+01
	f_{23}	PRG	2.58E+03	2.64E+03	4.48E+01
	f_{24}	PR	2.55E+03	2.66E+03	8.29E+01
	f_{25}	LSG.5	2.95E+03	2.98E+03	2.44E+01
	f_{26}	LS.5	3.12E+03	3.16E+03	4.68E+01
	f_{27}	PR	3.12E+03	3.16E+03	2.72E+01
	f_{28}	PRG	3.28E+03	3.34E+03	5.68E+01

hint of intelligent behavior in the algorithm's structure would substantially reduce the function evaluation needed to obtain a similar result. This is to the point that all considered variations of LS and LSG were able to converge to a closer approximation of the optimum solution with much lower NFE values. This may, however, be the result of two general yet opposing concepts. One is that these intelligence components have a notable effect on the performance of these pseudo algorithms. The alternative explanation would be that the C benchmark problems are innately incapable of truly testing the performance of CI-based optimization algorithms. On that note, the NC benchmark suite was tested and analyzed under a similar premise.

Table 6 summarizes the result of different pseudo algorithms against the NC benchmark suite. When the search space consists of only two variables (i.e., $n = 2$), different variations of LS and LSG seem to predominantly have the best average performance. PR and PRG pseudo algorithms also have appeared here as the outperforming

Table 7 The results for the non-parametric Friedman test against the NC benchmark suite

		f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{21}	f_{23}	f_{24}	f_{25}	f_{26}	f_{27}	f_{28}	Overall Rank
$n = 2$	PR	1	6	6	4	4	6	2	4	6	1	6	5	5	2	6	6	2	6
	PRG	2	5	5	5	3	5	1	3	5	2	5	6	6	1	5	5	1	4
	LS.5	3	4	4	1	1	3	4	1	4	3	3	4	4	4	4	3	4	2
	LS1	4	3	1	3	5	2	6	5	1	5	2	3	2	6	3	1	5	3
	LSG.5	6	1	1	2	2	1	3	2	1	4	1	1	1	3	1	2	3	1
	LSG1	5	2	1	6	6	4	5	6	1	6	4	2	3	5	2	4	6	5
$n = 10$	PR	5	1	5	4	6	6	5	6	6	1	2	2	1	5	2	1	2	4
	PRG	4	2	4	3	5	5	6	5	5	2	1	1	2	6	3	2	1	3
	LS.5	1	3	1	1	1	1	2	2	1	4	3	3	3	2	1	3	3	1
	LS1	3	5	3	5	4	3	4	4	3	5	5	5	5	3	5	6	6	5
	LSG.5	2	4	2	2	2	2	1	1	2	3	4	4	4	1	4	4	4	2
	LSG1	6	6	6	6	3	4	3	3	4	6	6	6	6	4	6	5	5	6

pseudo algorithms in some cases (e.g., f_1, f_7, f_{10}, f_{25}, and f_{28}). Again, the results obtained from the non-parametric Friedman test confirm these ideas, as LSG.5 followed by LS.5 have been selected as the best overall average performance for the NC benchmark suite (Table 7). A vital notion to note here is that LSG1 was outranked by the PRG pseudo algorithm, which contradicts what has been observed in the C benchmark suite, as, in that case, that was objectively the best option from a statistical standpoint. This can be further supported by the results of a non-parametric Wilcoxon test reported in Table 8. It is interesting to note that, based on the obtained results, the seemingly intelligent algorithm is being outperformed on occasion by PR (e.g., f_1, f_{10}) and PRG (e.g., f_1, f_7, f_{10}) pseudo algorithms, or in the case of f_{28} generate results that are statically at the same level as these random-based pseudo algorithms. In a ten-dimensional search space, which denotes the most complex search space considered in this study, the gap between the results of different pseudo algorithms in most cases gets close. For instance, under this setup, PR and PRG have, respectively, shown better average performance in 4 (i.e., f_2, f_{10}, f_{24}, f_{27}) and 3 (i.e., f_{21}, f_{23}, f_{28}) tested functions (Table 6). Based on these results, on average, LS.5 and LSG.5 still seem to produce better results than their considered counterparts. Again, these conclusions can be confirmed by the non-parametric Friedman test results, as these pseudo algorithms occupy the 1st and 2nd ranks as they get compared to other considered alternatives (Table 7). Interestingly, PRG and PR have been ranked 3rd and 4th under this setup, which opposes the results obtained from the C benchmark suite. All these conclusions are aligned with the results obtained from the non-parametric Wilcoxon test (Table 8).

Two additional sidenotes can be deduced from the obtained results thus far. First, it can be seen that while the greedy strategy often relatively helped the performance of the pseudo algorithms in the C benchmark suite, it had a less critical role while handling the NC benchmark suite. In fact, LSG1, a greedy embedded pseudo algorithm that was one of the top-tier options in the C benchmark suite, objectively had the worst performance under the NC benchmark suite. This shows that the greedy strategy, a prevalent mechanic in many modern CI-based optimizations, may have a role in creating the identified central bias tendencies. The other notable subject worth further investigation is the result of LS.5 and LSG.5 pseudo algorithms being selected as the best alternatives in the NC benchmark suite. The idea here is that these two, out of all tested setups, are primarily geared toward the local search process, as they are designed to permit more limited explorations than their alternative counterparts. While this could indicate that the idea of local search is an effective practice for a CI-based optimization algorithm, it could also indicate that the NC benchmark problems tested here (i.e., CEC2017) may unfairly reward mechanics that converge local optima. This also could be a potentially promising research idea.

Again, another method for assessing the performance of these pseudo algorithms is to compare their average convergence rate with various NFE values. Figures 9 and 10, for instance, depict the average convergence rate of pseudo algorithms for the f_{28} function of the NC benchmark suite in a two- and ten-dimensional search space, respectively. While there is a notable downward trend in both graphs, the range and the interquartile range (IQR) are more pronounced than the observed patterns in the C benchmark

Table 8 The results for the non-parametric Wilcoxon test against the NC benchmark suite

	Target	Function	PR	PRG	LS.5	LS1	LSG1
$n = 2$	LSG.5	f_1	+	+	+	0	0
		f_2	−	−	−	0	0
		f_3	−	−	0	0	0
		f_4	−	−	+	0	0
		f_5	−	−	+	+	+
		f_6	−	−	+	+	0
		f_7	0	+	0	−	−
		f_8	−	−	+	+	+
		f_9	−	−	0	0	0
		f_{10}	+	+	0	−	−
		f_{21}	−	−	−	−	−
		f_{23}	−	−	−	−	−
		f_{24}	−	−	−	−	−
		f_{25}	−	−	−	−	−
		f_{26}	−	−	−	−	−
		f_{27}	−	−	+	+	0
		f_{28}	0	0	0	−	−

			PR	PRG	LS1	LSG.5	LSG1
$n = 10$	LS.5	f_1	−	−	−	−	−
		f_2	0	−	−	0	−
		f_3	−	−	−	−	−
		f_4	−	−	−	−	−
		f_5	−	−	−	−	−
		f_6	−	−	−	−	−
		f_7	−	−	−	0	−
		f_8	−	−	−	+	−
		f_9	−	−	−	−	−
		f_{10}	+	+	0	0	0
		f_{21}	+	+	−	0	−
		f_{23}	+	+	−	−	−
		f_{24}	+	+	−	0	−
		f_{25}	−	−	−	0	−
		f_{26}	0	0	−	0	−
		f_{27}	+	+	−	0	−
		f_{28}	0	0	−	0	−

"−" signifies an inferior performance to the target; "+" means a superior performance to the target; and "0" identifies no statistically different performance to the target

suite results. This gap even tends to amplify with the increase in the number of dimensions. It is also worth noting that the graphs representing pseudo algorithms embedded with greedy strategy seem to have a smoother degression. However, in none of the cases, that, in and of itself, was not enough to converge the pseudo algorithm to the optimum solution.

4 Concluding remarks

As the number of CI-based optimization algorithms is increasing at an unprecedented rate, it is essential to examine the reliability of their underlying computational architecture. This becomes exceptionally more crucial in light of recent concerns raised about certain biases observed in the performance of some of these algorithms. As such, the premise of this study was to replicate some of the more prevalent, fundamental components of these algorithms in an abstract format as a measure to observe their behavior in an isolated environment. These were then used to solve a C and NC benchmark suite to showcase potential varying behavior.

First, the result hinted that while, in general, increasing the NFE would, as one would expect, improve the performance of the pseudo algorithms, it was also crucial to what these NFE values pertained. For instance, in this case, the results suggested that, in a relatively similar NFE value, *pop size* helped obtain better performance than the *max iteration* parameter. From a purely theoretical standpoint, this behavior stands to reason, as an increasing number of search agents would ultimately result in obtaining more information earlier in the search, which means that the course of the search may be adjusted via the information gathered from an earlier stage of the searching process. That said, this is merely a preliminary result as these are abstract and simplified pseudo algorithms, and further investigation is needed to generalize this beyond the scope of this research.

The second finding of this research was to showcase how the benchmark setup can have significant implications on how the algorithm works. For example, it was demonstrated here that insufficient intricacies of benchmark problems could easily disguise flaws in CI-based optimization algorithms' architecture if one increases the processing power of the algorithms. As for the underlying structure of these algorithms, certain features commonly utilized in many modern CI-based optimization algorithms can potentially generate central bias tendencies. The greedy strategy is an example of this. While this helped the pseudo algorithms obtain better results in the case of the C benchmark suite, the effect softened with the NC benchmark suite. That said, this helped smooth out the algorithm's performance in all cases, but it is crucial to create a balance in resorting to this strategy. This, in and of itself, could be a potentially promising research idea to understand the effect of this strategy.

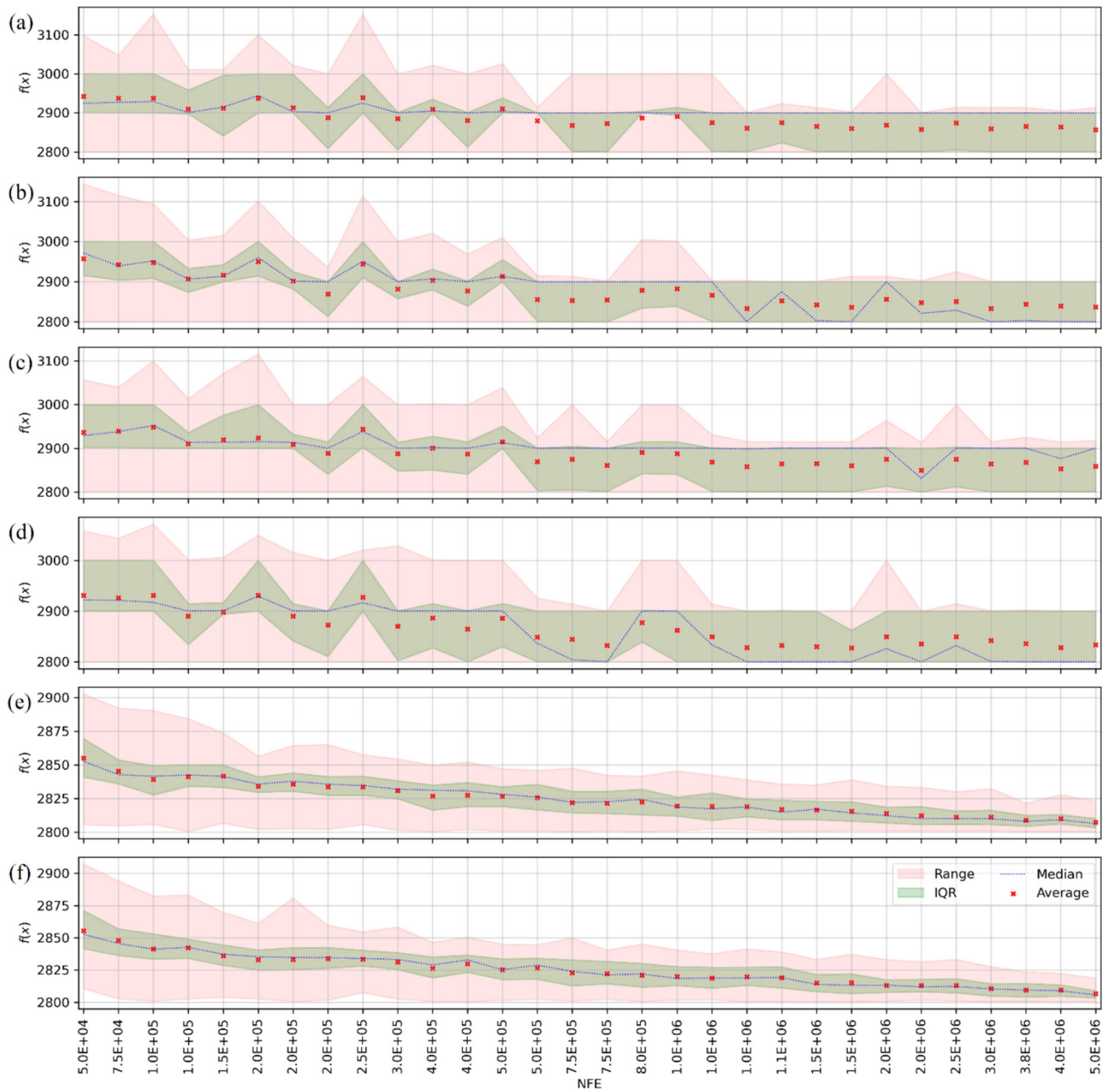

Fig. 9 Comparison of the average convergence rate of different pseudo algorithms for the f_{28} function of the NC benchmark suite ($n = 2$): **a** PR, **b** PRG, **c** LS.5, **d** LS1, **e** LSG.5, and **f** LSG1

But, perhaps the most fundamental issue when it comes to the practice of using CI-based optimization algorithms is that, often, like many other data-driven models, it became an unspoken standard to treat them as a *black box*. This black box perception implies that it is acceptable for the practitioner not to have explicitly understated how these algorithms work so long as the generated results fall within the conventional range of what is expected from such models. And the problem with such viewpoints is that not only does it permit oversights like central bias to go undetected, but one cannot make any amendment toward a more robust algorithm, as there would be no meaningful

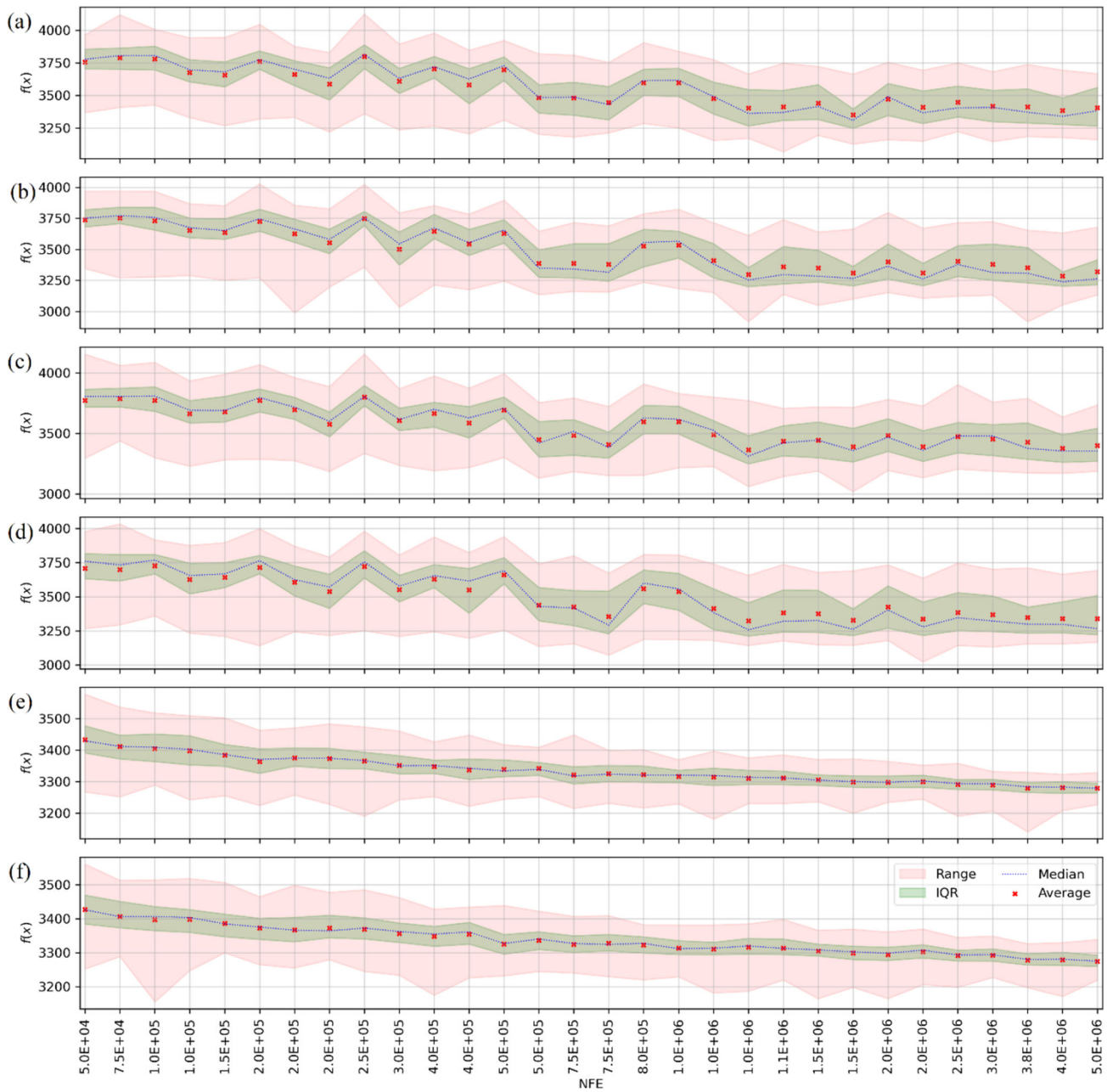

Fig. 10 Comparison of the average convergence rate of different pseudo algorithms for the f_{28} function of the NC benchmark suite (n = 10): **a** PR, **b** PRG, **c** LS.5, **d** LS1, **e** LSG.5, and **f** LSG1

feedback on how the algorithm works. That is why studies like this should aim to shed light on the underlying mechanics of CI-based optimization algorithms.

Authors contributions Not applicable.

Funding Open Access funding enabled and organized by CAUL and its Member Institutions.

Data availability All used data have been presented in the paper. The codes used to test these algorithms are available at the corresponding author's GitHub page: https://github.com/BabakZolghadrAsli/randomness_in_CI_optimization.

Declarations

Conflict of Interests The authors have no relevant financial or non-financial interests to disclose.

Ethical approval The authors declare all data and materials, as well as software applications or custom codes, are in line with published claims and comply with field standards.

Consent to participate Not applicable.

Consent to publish Not applicable.

References

Aranha C, Camacho Villalón CL, Campelo F, Dorigo M, Ruiz R, Sevaux M, Sörensen K, Stützle T (2022) Metaphor-based metaheuristics, a call for action: the elephant in the room. Swarm Intell 16(1):1–6

Bozorg-Haddad O, Solgi M, Loáiciga HA (2017) Meta-heuristic and evolutionary algorithms for engineering optimization. Wiley, New York

Camacho-Villalón CL, Dorigo M, Stützle T (2022) Exposing the grey wolf, moth-flame, whale, firefly, bat, and antlion algorithms: six misleading optimization techniques inspired by bestial metaphors. Int Trans Oper Res. https://doi.org/10.1111/itor.13176

Campelo F, Aranha C (2021) EC Bestiary: a bestiary of evolutionary, swarm and other metaphor-based algorithms. Accessed May 9, 2023. https://github.com/fcampelo/EC-Bestiary. https://doi.org/10.5281/zenodo.1293352

Cheng MY, Prayogo D (2014) Symbiotic organisms search: A new metaheuristic optimization algorithm. Comput Struct 139:98–112

Chu SC, Tsai PW, Pan JS (2006) Cat swarm optimization. In: PRICAI 2006: trends in artificial intelligence: 9th Pacific Rim international conference on artificial intelligence Guilin, China, August 7–11, 2006 Proceedings 9 (pp. 854–858). Springer, Berlin

Derrac J, García S, Molina D, Herrera F (2011) A practical tutorial on the use of non-parametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. Swarm Evol Comput 1(1):3–18

Du, K.L. & Swamy, M.N.S. (2016). Search and optimization by metaheuristics: techniques and algorithms by nature. Springer, Cham, ISBN: 9783319411910

Eskandar H, Sadollah A, Bahreininejad A, Hamdi M (2012) Water cycle algorithm—a novel metaheuristic optimization method for solving constrained engineering optimization problems. Comput Struct 110:151–166

Eusuff M, Lansey K, Pasha F (2006) Shuffled frog-leaping algorithm: a memetic meta-heuristic for discrete optimization. Eng Optim 38(2):129–154

Fong S, Wang X, Xu Q, Wong R, Fiaidhi J, Mohammed S (2016) Recent advances in metaheuristic algorithms: Does the Makara dragon exist? J Supercomput 72:3764–3786

Gandomi AH, Alavi AH (2012) Krill herd: a new bio-inspired optimization algorithm. Commun Nonlinear Sci Numer Simul 17(12):4831–4845

Glover F (1986) Future paths for integer programming and links to artificial intelligence. Comput Oper Res 13(5):533–549

Gómez D, Rojas A (2016) An empirical overview of the no free lunch theorem and its effect on real-world machine learning classification. Neural Comput 28(1):216–228

Kudela J (2022) A critical problem in benchmarking and analysis of evolutionary computation methods. Nat Mach Intell 4:1238–1245

Kudela J (2023) The evolutionary computation methods no one should use. arXiv preprint arXiv:2301.01984

Mehrabian AR, Lucas C (2006) A novel numerical optimization algorithm inspired from weed colonization. Eco Inform 1(4):355–366

Mirjalili S (2015) Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm. Knowl-Based Syst 89:228–249

Mirjalili S, Mirjalili SM, Lewis A (2014) Grey wolf optimizer. Adv Eng Softw 69:46–61

Rao RV, Savsani VJ, Vakharia DP (2011) Teaching–learning-based optimization: a novel method for constrained mechanical design optimization problems. Comput Aided Des 43(3):303–315

Salhi A, Fraga ES (2011) Nature-inspired optimisation approaches and the new plant propagation algorithm. In: Proceedings of the international conference on numerical analysis and optimization, Yogyakarta, Indonesia

Sörensen K (2015) Metaheuristics—the metaphor exposed. Int Trans Oper Res 22(1):3–18

Tzanetos A, Dounias G (2021) Nature inspired optimization algorithms or simply variations of metaheuristics? Artif Intell Rev 54:1841–1862

Velasco L, Guerrero H, Hospitaler A (2022) Can the global optimum of a combinatorial optimization problem be reliably estimated through extreme value theory? Swarm Evol Comput 75:101172

Velasco L, Guerrero H, Hospitaler A (2024) A literature review and critical analysis of metaheuristics recently developed. Arch Comput Methods Eng 31(1):125–146

Wolpert DH, Macready WG (1997) No free lunch theorems for optimization. IEEE Trans Evol Comput 1(1):67–82

Wu G, Mallipeddi R, Suganthan PN (2017) Problem definitions and evaluation criteria for the CEC 2017 competition on constrained real-parameter optimization, Technical Report. National University of Defense Technology, Changsha, Hunan, PR China and Kyungpook National University, Daegu, South Korea and Nanyang Technological University, Singapore

Yaghoubzadeh-Bavandpour A, Bozorg-Haddad O, Zolghadr-Asli B, Gandomi AH (2022) Improving approaches for meta-heuristic algorithms: a brief overview. In: Computational intelligence for water and environmental sciences, pp 35–61

Yang X, Gandomi AH (2012) Bat algorithm: A novel approach for global engineering optimization. Eng Comput 29(5):464–483

Yang, X. (2010). Nature-inspired metaheuristic algorithms. Luniver Press, ISBN: 9781905986286

Yang X (2012) Flower pollination algorithm for global optimization. In: Proceeding of the 11th International conference on unconventional computing and natural computation, Orléan, France

Zolghadr-Asli B (2023a) Computational intelligence-based optimization algorithms: from theory to practice, 1st edn. CRC Press, London

Zolghadr-Asli B (2023b) No-free-lunch-theorem: a page taken from the computational intelligence for water resources planning and management. Environ Sci Pollut Res 30:57212–57218