Analyzing the Effects of Memetic Variations on Convergence in Overlapping Swarm Intelligence

Nathan Patera and John Sheppard(✉)

Gianforte School of Computing, Montana State University, Bozeman, MT 59717, USA
john.sheppard@montana.edu

Abstract. Evolutionary algorithms often struggle when there exists a high degree of interdependence between the variables (epistasis), non-separability, discontinuity, high dimensionality, and sparsity. These complexities can lead to hitchhiking, where poor parameters are associated with good schemata, and "two steps forward and one step back" where near-optimal parameters are lost in favor of lower-quality parameters that immediately improve fitness; phenomena that contribute to premature convergence. Overlapping Swarm Intelligence (OSI) has been introduced as a cooperative coevolutionary algorithm that utilizes overlap of variables between subswarms to encourage sharing and competing among variables across the subswarms. OSI has shown success in handling epistasis, however it can still suffer from premature convergence when subswarms get stuck in "pseudo-optima," i.e., when a subset of variables is at a minimum with respect to the reduced search space but not the entire space. We investigate convergence on memetic variations of OSI, CPSO (OSI with no overlap), and Particle Swarm Optimization using block coordinate descent applied to the CEC2010 Benchmark problems. The memetic algorithms show some success in assisting subpopulations in finding more optimal solutions compared to their non-memetic counterparts. We also use the Gini coefficient on a solution's derivative to estimate the proportion of variables stuck in pseudo-optima to help understand what contributes to premature convergence.

Keywords: Overlapping Swarm Intelligence · Cooperative co-evolution · pseudo-optima · Block coordinate descent

1 Introduction

Evolutionary Algorithms (EAs) are frequently used in solving complex optimization problems in lieu of other methods when the search space is particularly difficult to navigate. Multiple characteristics can contribute to the difficulty of this task: high dimensionality and the relatively sparse and uninformative

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/978-3-031-90065-5_11.

P. García-Sánchez et al. (Eds.): EvoApplications 2025, LNCS 15613, pp. 176–191, 2025.
https://doi.org/10.1007/978-3-031-90065-5_11

points within the search space; complexity such as non-linearity or non-convexity of the objective functions or constraints; and or non-separable and discontinuous regions within the search space. EAs are often selected to navigate these types of space due to their stochastic nature.

Commonly used algorithms such as Genetic Algorithm (GA), Differential Evolution (DE), and Particle Swarm Optimization (PSO) all have different approaches in exploring and exploiting the search space. Although this general class of algorithms has shown great success, it is well known these issues are exacerbated in higher dimensions [28], and when the relationship between variables become highly interdependent (epistasis) [5,14]. Interdependent variables and complex search-spaces can contribute to *hitchhiking*, where poor parameters are associated with good schemata [21], and "*two steps forward and one step back*," where near-optimal values are lost in favor of lower-quality values that immediately appear to have better fitness [1].

1.1 Cooperative Coevolutionary Algorithms

To address these issues, a class of EAs called Cooperative Coevolutionary Evolutionary Algorithms (CCEAs) emerged to deal with issues caused by epistasis, hitchhiking, "two steps forward and one step back," and non-separability [20]. These algorithms divide the population into subpopulations or "subspecies" so that each evolves and optimizes over a disjoint subset of the variables. By partitioning the set of variables, CCEAs have been shown to be able to navigate high-dimensional search spaces more effectively [1,20]. For most CCEAs, a global "context" vector is maintained to represent the best total solution seen so far and updated based on the performance of the individual subpopulation.

1.2 Factored Evolutionary Algorithms

Strasser *et al.* introduced Factored Evolutionary Algorithms (FEA) in [25] as a new class of CCEAs, but instead of variables being subdivided into disjoint factors, FEA allows subpopulations to optimize over overlapping factors. Rather than treating each subpopulation as independent, FEA takes into account that there may exist interaction between subpopulations and allows each overlapping subpopulation to independently solve over the factor overlap. It was shown in [26] that FEA is able to outperform CCEA in settings where there was a high degree of epistasis between the variables. Overlapping Swarm Intelligence (OSI) in [19] was the earliest known implementation of FEA, which used PSO to optimize on overlapping subswarms. It was then in [25] that OSI was generalized to FEA by allowing any stochastic search algorithm to optimize over the subpopulations, not only PSO. Throughout this paper, we focus on the OSI instance of FEA.

1.3 Convergence in CCEAs and FEAs

Throughout this work, we refer to the structure of the set of subswarms in OSI as a *factor architecture*. Previous works [24,25] validated that factor architecture

matters in FEA/OSI and CCEA across problems. In their work, for example, in the Bayesian network inference setting, an architecture of factors based on each variable's Markov blanket performed significantly better than other architectures, and a "neighborhood architecture" in the NK Landscapes outperformed other architectures.

Van den Bergh and Engelbrecht were likely the first to realize one convergence issue was caused by factors or variables getting caught in *pseudominima*. If a subpopulation was in a pseudominimum with respect to its own factor, it could not improve because it was effectively in a local optimum for that factor. Although it was generally shown OSI (FEA+PSO) outperformed CPSO (CCEA+PSO) in multiple settings, including under random architectures, both algorithms remained susceptible getting stuck in pseudominima.

In this work, motivated to prevent premature convergence issues and improve performance of CPSO and OSI, we will investigate the relationship between "pseudominima" and convergence by incorporating memetic additions into the OSI algorithm. Memetic algorithms are a class of algorithms that incorporate an additional heuristic process into the search on top of the underlying optimization algorithm. We incorporate two variations of gradient descent into OSI, CPSO, and PSO: full gradient descent (FGD) on the context vector (global solution), and block coordinate descent (BCD) on every individual in each subpopulation.

Previous works have successfully incorporated local search methods, including gradient descent, into EAs to improve optimization [4,17,18]. It is hypothesized that, like previous works, incorporating gradient descent into FEA will help avoid premature convergence by preventing subpopulations getting stuck in pseudominima or assisting subpopulations to escape their pseudominima via the variables that overlap between subpopulations. It is further hypothesized that the memetic variations applied to the underlying algorithms will help improve the convergence rate of these algorithms while also enabling a comparison of "pseudominima" and convergence across algorithms.

2 Background

2.1 Factored Evolutionary Algorithms

In this work, we make use of a specialization of Factored Evolutionary Algorithms [25] called Overlapping Swarm Intelligence (OSI) [19]. As background on OSI, we focus on the more general FEA approach, emphasizing that the only difference is the use of PSO as the underlying optimizer.

In Cooperative Coevolutionary Algorithms (CCEAs), the set of variables $\mathbf{X}$ is subdivided into partitions $\boldsymbol{\mathcal{S}} = \{\mathbf{S}_i\}_{i=1}^{F}$ such that $\mathbf{S}_i \subset \mathbf{X}$ and $\mathbf{S}_i \cap \mathbf{S}_j = \varnothing$ for all $i, j \in [1, F]$, where $i \neq j$. In contrast to CCEAs, the class of Factored Evolutionary Algorithms (FEAs) allow for the possibility of one more $\mathbf{S}_i$ overlapping, i.e., $\mathbf{S}_i \cap \mathbf{S}_j \neq \varnothing$. Note that CCEA can be regarded as a special case of FEA where there is no such overlap. In the FEA context, these subsets of variables $\boldsymbol{\mathcal{S}}$ are referred to as *factors*. In FEA and CCEA, the set of subpopulations $\boldsymbol{\mathcal{P}} = (\mathcal{P})_{i=1}^{F}$ optimizes over the factors such that $\mathcal{P}_i$ optimizes over factor $\mathbf{S}_i$.

Algorithm 1. FEA

Input: Function f, base algorithm $\mathcal{A}$
Output: Full solution $\mathbf{G}$

1: $\boldsymbol{\mathcal{S}} \leftarrow$ Initialize factors and subpopulations from f, $\mathbf{X}$, and algorithm A.
2: $\mathbf{G} \leftarrow$ Initialize full global from $\boldsymbol{\mathcal{S}}$.
3: **repeat**
4: **for all** $\mathcal{P}_i \in \boldsymbol{\mathcal{S}}$ **do**
5: Optimize $\mathcal{P}_i$ using algorithm A
6: $\mathbf{G} \leftarrow \text{Compete}(f, \boldsymbol{\mathcal{S}})$
7: $\boldsymbol{\mathcal{S}} \leftarrow \text{Share}(f, \mathbf{G}, \boldsymbol{\mathcal{S}})$
8: **until** Termination criterion is met
9: **return** $\mathbf{G}$

Evaluating the fitness of an individual using only the variables of factor $\mathbf{S}_i$ is insufficient, and requires the remaining variables outside of the factor to evaluate the objective function. Let $\mathbf{s} = \mathbf{x}|_{\mathbf{S}}$ be a restriction of variables in a full solution $\mathbf{x}$ to a subset of variables $\mathbf{S} \subseteq \mathbf{X}$ such that $\mathbf{x}|_{\mathbf{S}} = \{x_i \mid X_i \in \mathbf{S}\}$. Let $\mathbf{R}_j = \mathbf{X} \setminus \mathbf{S}_j$ be the set of variables not included by the subspace factor $\mathbf{S}_j$. For a full solution $\mathbf{x}$, let $\mathbf{s}_j = \mathbf{x}|_{\mathbf{S}_j} = \{x_i \mid X_i \in \mathbf{S}_j\}$ and $\mathbf{r}_j = \mathbf{x}|_{\mathbf{R}_j} = \{\mathbf{x}_i \mid X_i \in \mathbf{R}_j\}$ be a partition on $\mathbf{x}$. The context of a factor $\mathbf{S}_i$ is the remaining variables in the global solution $\mathbf{G}$, denoted $\mathbf{r}_j(\mathbf{G}) = \{g_i \in \mathbf{G} \mid X_i \in \mathbf{R}_j\}$. Let $\mathbf{s} \cup \mathbf{r}$ denote the recomposition of a full solution $\mathbf{x} = \{x_i \in \mathbf{s} \cup \mathbf{r} \mid X_i \in \mathbf{X}\}$.

The FEA algorithm, as shown in Algorithm 1, has 3 primary steps.

1. **Optimize:** In the first step (lines 4–5), each of the individuals in subpopulations $\mathcal{P}_i$ are optimized. For OSI, PSO is the optimizer chosen. This is done by optimizing subspecies on factor $\mathbf{S}_i$ while holding $\mathbf{R}_i$ constant.
2. **Compete:** In the second step (line 6), each individual in the subpopulation competes and contributes to a new or updated global solution $\mathbf{G}$ (also referred to as the global context vector). Competition occurs at the points of overlap between the factors, and the global solution is updated with the best values from the overlapped subpopulations.
3. **Share:** In the third step (line 7), information exchange occurs between the subpopulations. Each subpopulation's local context vector $\mathbf{R}_i$ is updated from the global context $\mathbf{G}$, and the worst solution $\mathbf{s}_j \in \mathcal{P}_i$ is replaced with the global context vector, $\mathbf{s}_j \leftarrow \mathbf{G}|_S$.

CCEA and its factor architecture is able to run on the FEA framework, and if $\boldsymbol{\mathcal{S}}$ contains a single factor $\mathbf{S}$ then $\mathbf{S} = \mathbf{X}$ and $\mathbf{R} = \varnothing$. Thus any single-population evolutionary algorithm can also be run using the FEA framework.

2.2 Minima and Pseudominima

To analyze convergence and the issue of pseudominima realized by [1], Strasser and Sheppard introduced a metric to measure the proportion of factors in pseudominima relative to the number of factors in local minima [26]. The metric,

here called *pseudominima ratio*, is calculated as

$$R(t) = \frac{Pm}{Pm + Lm},$$

where Pm is the number of factors that are pseudominimum, and Lm is the number of factors that are (exclusively) at a local minima (yet are not a pseudominimum as well) at iteration t. If $R(t) = 1$ then all subpopulations are at pseudominima, and if $R(t) = 0$ then they are all at local minima. This is the inspiration for a metric we propose in Sect. 5.2, except it is equipped to handle first-order derivable functions and analyze convergence of subpopulations.

3 Related Work

Cooperative Coevolutionary algorithms (CCEA) began with works proposed by [20] with the purpose of partitioning a solution-space into subsets of variables called "subspecies" using GA, "Cooperative Coevolutionary GA" (CCGA). Treating each "subspecies" as a separate sub-problem made it potentially more effective and feasible to find a solution by solving for simpler components and recombining them into a final solution.

Van den Bergh and Engelbrecht brought cooperative coevolution to PSO [1]. In CPSO, the GA optimizer is replaced with a PSO optimizer. In this work, consideration was also given to problems of premature convergence where there was an issue with subpopulations getting trapped in "pseudominima". One approach to handling this problem was through the definition of the hybrid CPSO where optimization alternated between focusing on individual factors and running full PSO updates.

As an early attempt to extend CPSO, Pillai and Sheppard looked at the possibility of training neural networks where the factors corresponded to paths through the network [19]. This necessitated being able to handle overlap in the subswarms. Subsequently, Fortier *et al.* explored applying the overlapping swarm idea in OSI to solving the abductive inference problem with Bayesian networks [10]. This led to a further generalization of OSI where Fortier introduced distributed OSI (DOSI) to distribute the context vector across subpopulations, each maintaining a locally-best solution rather than maintaining a global solution vector, and sharing the subpopulation's most optimal variable(s) with other subpopulations [11]. DOSI increased diversity, improved competition, and showed improved performance compared to OSI as applied to Bayesian Network abductive inference. Subsequent work evaluated the extent to which consensus needed to be reached on the context vector and also showed that there should be equivalent performance with DOSI when consensus is reached [2].

Strasser and Sheppard developed a variant of FEA called Hybrid Factored Evolutionary Algorithms [26], similar to the Hybrid CPSO [1]. These hybrid algorithms ran an iteration of regular CPSO or FEA steps, then ran some number of iterations of the base algorithm on an unfactored subpopulation, that is,

optimizing over the full set of variables $\mathbf{X}$. Section 4 introduces a framework that generalizes Hybrid FEA for any particular optimizer $\mathcal{M}$, a memetic variant.

This work incorporates gradient descent (GD) methods as memetic variations to PSO, CPSOs, and OSI. In [26] it was noted an algorithm such as "Sequential Subspace Optimization" (SESOP) in [9] that utilizes gradient descent (effectively a block coordinate descent) might be less susceptible to becoming trapped in pseudominima, as it is able to update any subset of variables at once rather than just the variables of $\mathbf{S}_i$. Gradient descent methods are usually always able to escape saddle-points (a form of pseudominima) in the limit [12,13]. Additionally, there has been success applying GD to escape saddle-points, as in [6] with Deep Neural Networks.

Du *et al.* found that, while gradient descent methods based on random subsets generally are able to escape in the limit, perturbation-based GD methods are considerably better at escaping saddle points [7]. Many stochastic-based algorithms, including PSO, could be considered perturbation-based methods, and as such reap benefits of using a gradient descent method. Empirically, there has been success in memetic algorithms incorporating GD into EAs such as "Evolutionary Stochastic Gradient Descent (ESGD)" [17] and "Improved Particle Swarm Optimization Based on Gradient Descent Method" (GDPSO) [4].

There are a number of CCEA-based memetic algorithms. Smith introduced a memetic framework for GA, adding memetic recombination and mutation operator steps after the basic mutation operations [22]. Later, Cao *et al.* introduced a generalized memetic framework, called "CC-GLS" for CCEA, by introducing a local search component to the global search [3]. Their framework optimized each subpopulation, first using the chosen global search method and then using the local search method, before moving on to the next subpopulation. Their implementation utilized a DE algorithm, and Solis Wets method as the local search method. Notably, another memetic algorithm incorporating a Solis Wets method [23] as a local search operator using a GA, named "MA-SW-Chains," won the CEC 2010 Benchmark Competition for its ability to handle the highly non-separable benchmarks [18]. Besides these, no work besides in Hybrid-FEA, has been done on memetic algorithms for FEAs and OSI.

4 Memetic-OSI

Formally, we define a framework for which we are able to introduce any memetic optimizer into OSI. The Memetic-OSI algorithm is mostly identical to OSI, except it alternates between optimization algorithms for the Compete step. That is, Memetic-OSI alternates between using PSO for the the underlying subpopulation optimizer and a memetic optimizer $\mathcal{M}$. We allow $\mathcal{M}$ to be either a population optimizer or a solution optimizer. If $\mathcal{M}$ is a solution optimizer, then a full solution $\mathbf{x}$ is updated to improve fitness. If $\mathcal{M}$ is a population optimizer, then the individual members of $\mathbf{P}_j \in \mathcal{P}$, $\mathbf{x}_j$, are updated. The Memetic-OSI framework is given in Algorithm 2. Here, Memetic-OSI iterates through rounds of Optimize, Compete and Share steps, once for each of the base algorithm $\mathcal{A}$ and

Algorithm 2. MEMETIC-OSI

Input: Function f, base algorithm $\mathcal{A}$ = PSO, memetic optimizer $\mathcal{M}$
Output: Full solution $\mathbf{G}$

$\boldsymbol{\mathcal{S}} \leftarrow$ Initialize factors and subpopulations from f, $\mathbf{X}$, and A.
$\mathbf{G} \leftarrow$ Initialize full global from $\boldsymbol{\mathcal{S}}$.
while true do
    ▷ `FEA Steps using base algorithm` $\mathcal{A}$
    for all $\mathcal{P}_i \in \boldsymbol{\mathcal{S}}$ **do**
        Optimize $\mathcal{P}_i$ using $\mathcal{A}$
    $\mathbf{G} \leftarrow \text{Compete}(f, \boldsymbol{\mathcal{S}})$
    $\boldsymbol{\mathcal{S}} \leftarrow \text{Share}(f, \mathbf{G}, \boldsymbol{\mathcal{S}})$
    if $A = \mathcal{A}$ and Termination criterion is met **then**
        return G
    ▷ `FEA Steps using memetic algorithm` $\mathcal{M}$
    $\boldsymbol{\mathcal{S}}, \mathbf{G} \leftarrow \mathcal{M}(f, \boldsymbol{\mathcal{S}}, \mathbf{G})$
    $\mathbf{G} \leftarrow \text{Compete}(f, \boldsymbol{\mathcal{S}})$
    $\boldsymbol{\mathcal{S}} \leftarrow \text{Share}(f, \mathbf{G}, \boldsymbol{\mathcal{S}})$

the memetic optimizer $\mathcal{M}$. Due to the possible stochastic nature of the memetic optimizer, we require that Memetic-OSI terminate only upon meeting a criterion *after* a round using $\mathcal{A}$.

In this work, two memetic optimizers are suggested to improve convergence in OSI, CPSO and PSO. The first memetic variation uses Gradient Descent (GD) on the full global context vector $\mathbf{G}$ to improve the local context $\mathbf{S}_i$ for all subpopulations $\mathcal{P}_i$. The second memetic variation uses a Block Coordinate Descent (BCD) method to optimize every individual $\mathbf{P}_j$ in $\mathcal{P}$.

The computational cost of estimating the gradient for a solution $\mathbf{x}$ requires estimating the partial for every variable $X_i \in \mathbf{X}$. Let $\nabla f(\mathbf{x})$ be the gradient on $\mathbf{x}$ composed of partials $f'_{X_i}(\mathbf{x})$ for all $x_i \in \mathbf{x}$. Estimating the gradient for a factor $\mathbf{S}$ of size M requires calculating M partials for every $X_j \in S$. The expense is M function evaluations for every individual in each subpopulation, which is a total of $M \cdot P \cdot F$ function evaluations (where $M \cdot F \approx N$ in the CCEA setting) as compared to approximately $P \cdot F$ function evaluations per iteration for every individual in PSO. Due to the cost of calculating this, we devised a heuristic similar to directional direct-search methods [8,15,16] to find a subgradient $\mathbf{d}^*$ that is close to the steepest gradient, requiring fewer function evaluations. This method generates a set of direction vectors $\boldsymbol{\xi}^S = \{\mathbf{d}_i \in \mathbb{S}^N\}_{i=1}^S$ sampled from the unit ball, and chooses a direction $\mathbf{d}^*$ from $\boldsymbol{\xi}$ that minimizes an objective function at an arbitrarily small distance $\epsilon \ll 1$ from $\mathbf{x}$. The direction is found as

$$\mathbf{d}^*(\mathbf{x}) = \arg\max_{\mathbf{d} \in \boldsymbol{\xi}^S} \{|f(\mathbf{x} + \epsilon\mathbf{d}) - f(\mathbf{x})|\}.$$

Algorithm 3. BCD-Optimizer

Input: Function f, population $\mathcal{P}_i$

for $Iter = 30$ iterations **do**
  for all $\mathbf{P}_j \in \mathcal{P}_i$ **do**
    $\mathbf{d} \leftarrow \text{gradsample}(f, \mathbf{s}_j \cup \mathbf{r}_i(\mathbf{G}), \mathbf{S}_i, S = \lceil\sqrt{M}\rceil)$
    $\mathbf{d} \leftarrow \mathbf{d}/\max|\mathbf{d}|$
    $\mathbf{s}_j \leftarrow \mathbf{s}_j - \alpha \cdot \mathbf{d}$

Algorithm 4. FGD-Optimzer

Input: Function f, global solution $\mathbf{G}$

for $Iter = 30$ iterations **do**
  $\mathbf{d} \leftarrow \text{gradsample}(f, \mathbf{G}, \mathbf{X}, S = \lceil\sqrt{M}\rceil)$
  $\mathbf{d} \leftarrow \mathbf{d}/\max|\mathbf{d}|$
  $\mathbf{G} \leftarrow \mathbf{G} - \alpha \cdot \mathbf{d}$

To obtain a gradient estimate for a factor $\mathbf{S}$ of size M on a solution $\mathbf{x}$, points are only sampled around $\mathbf{s}$ to obtain a direction

$$\mathbf{d}_S^*(\mathbf{x}) = \arg\min_{\mathbf{d}} \{f(\mathbf{r} \cup (\mathbf{s} + \epsilon\mathbf{d})) \mid \mathbf{d} \in \boldsymbol{\xi}^S \subset \mathbb{S}^M\}.$$

We defined a function

$$\text{gradsample}(f, \mathbf{x}, \mathbf{S}, S) := \mathbf{d}_S^*(\mathbf{x})$$

to obtain a direction of descent $\mathbf{d}$ on $\mathbf{s} \subseteq \mathbf{x}$ from S samples $\xi^S \subset \mathbb{S}^M$ where $M = |\mathbf{S}|$. This function generates a direction g that will be used as a gradient descent direction in our memetic optimizers.

The Block Coordinate Descent (BCD) optimizer is given in Algorithm 3 as a population optimizer, and as applied in Memetic-OSI, we refer to them as OSI+BCD, CPSO+BCD, PSO+BCD for OSI, CCEA, and PSO. Likewise, the Full Global Descent (FGD) optimizer is presented in Algorithm 4 as a single solution optimizer and are referred to similarly in Memetic-OSI as OSI+FGD, CPSO+FGD, and PSO+FGD.

In both BCD and FGD optimizers, $\mathbf{d}$ is chosen as a descent direction from a heuristic number of sample directions ($S = \lceil\sqrt{M}\rceil$) that best minimizes the objective function, so the descent direction estimates a directional derivative in proportion. Note that $\mathbf{d}$ is normalized using an infinity norm, as a step size $\alpha\mathbf{d}$ under ℓ_2 may be too small for any learning rate $\alpha \leq 1$ in high dimensional settings like $N = 1000$ and large domains. In smaller domains or low dimensional settings, the maximum step size may be too large, so the learning rate α will need to be tuned appropriately. A descent step is then taken to minimize the objective. In FGD, a descent step on the global solution $\mathbf{G}$ occurs for $Iter$ iterations, updating all variables $\mathbf{X}$. On the other hand, in BCD, $Iter$ iterations of descent steps occurs for each individual in the subpopulation. In both algorithms, the descent step is taken regardless of the fitness improvement.

Table 1. Separability and modality for the CEC 2010 benchmark functions, grouped by their basis functions.

$\mathcal{F}$	Degree of Separability					Modality
	S	1	$N/2m$	N/m	NS	
Elliptic	F1	F4	F9	F14	—	Unimodal
Rastrigin	F2	F5	F10	F15	—	Multimodal
Ackley	F3	F6	F11	F16	—	Unimodal
Schwefel	—	F7	F12	F17	F19	Unimodal
Rosenbrock	—	F8	F13	F18	F20	Multimodal

5 Methods

5.1 Experimental Design

In this work, we wish to compare the convergence properties of OSI, CPSO, and PSO with their memetic counterparts using the Memetic-FEA framework with Block Coordinate Descent and Full Gradient Descent as memetic optimizers.

We have elected to use the CEC 2010 Benchmark Problems [27] for varying degrees of separability and difficulty. Table 1 summarizes the characteristics of the 20 benchmark problems, showing the degrees of modality and separability in the problems. In all cases, we seek to minimize the functions. The functions are grouped based on their basis functions, consisting of Elliptic, Rastrigin, Ackley, Schwefel's Problem 1.2 (Schwefel), and Rosenbrock respectively. The first three are naturally separable functions, F1-F3, which we label simply as "S," while the last two, F19 and F20, are naturally non-separable functions and are labeled as "NS." These functions, F1-F3, F19, F20, are the straight forms of their basis functions. The remaining degrees of separability ($g = 1, N/2m, N/m$) specify the number of non-separable groups of size m variables in the objective function. These groups are separable from each other, and separable from the ungrouped variables. The exception is Schwefel and Rosenbrock, which use the Sphere function as the basis function over the ungrouped variables. The non-separable groups of variables, are made non-separable by rotating the m-sized group of variables using a rotation matrix $\mathbf{R}$. For g-groups of size m, each of the benchmarks is of the form,

$$\mathrm{F}(\mathbf{x}) = \sum_{i=1}^{g} f_b(\mathbf{R}(\mathbf{x}[m(i-1)+1 : mi])) + f_b(\mathbf{x}[gm+1 : N]),$$

where f_b is a basis function, and the colon operator specifies a range of variables on $\mathbf{x}$. The benchmarks are derivable for an exact gradient, which is used to analyze convergence of solutions.

The experimental design is a block design generated by {OSI,CPSO,PSO} × {None, BCD, FGD} × $\{F_1, \cdots, F_{20}\}$. We did not experiment with PSO+FGD on the global solution, and only had a full gradient descent step on each particle,

denoted PSO+BCD, with all variables being a block. A total of 25 trials were performed for each of the blocks. Each experiment had randomized starting positions, and a sequence of overlapping factors based on a random permutation of the variable indices. For example, variables $[\mathbf{x_1}, \ldots, \mathbf{x}_9]$ might be factored as

$$[[\mathbf{x}_5, \mathbf{x}_1, \mathbf{x}_8], [\mathbf{x}_8, \mathbf{x}_9, \mathbf{x}_2], [\mathbf{x}_2, \mathbf{x}_4, \mathbf{x}_3], [\mathbf{x}_3, \mathbf{x}_7, \mathbf{x}_6]].$$

For each of the 20 benchmarks and all algorithm variations, the learning rate α, PSO parameters ϕ and ω were tuned using grid-search. Termination criteria were either finding the solution or detecting a diminishing rate of improvement after a minimum of 25M FEs. For OSI, the number of variables M in each factor and the maximum number of overlapping variables O was tuned; while for CPSO, only M variables in each factor were tuned since there was no overlap. To analyze performance of the algorithms, the global solution $\mathbf{G}$, its benchmark value $f(\mathbf{G})$, and the number of function evaluations (FEs), were recorded after every Compete step.

5.2 Gini Pseudominimum Measure

Since the *pseudominimum* ratio metric from [26] requires knowing if a factor $\mathbf{S}_i$ is at a pseudo-optimum to calculate the proportion of factors in pseudominima, we introduce what appears to be a novel usage of the Gini coefficient to assess the proportion of variables at a critical point as a proxy to estimating how close a factor is to a pseudominimum. To estimate such a minimum, we use a metric $R(\mathbf{x})$ that has the following property,

$$R(\mathbf{x}) = \frac{1}{N} \sum_{i=1}^{N} I(\nabla_{x_i} f(\mathbf{x}) = 0),$$

where I is the indicator function that returns one when the partial is zero.

A few issues make this approach difficult in practice. The first issue is determining if x_i is at a critical point, which requires choosing an ϵ such that $\nabla_{x_i} f(\mathbf{x}) < \epsilon$. Another issue is how to determine if x_i is "stuck." The ideal characteristic of a metric would be to provide the proportion of x_i derivatives near-zero in a solution such that x_i contributes less to the proportion as it is further from zero. A dispersion metric such as the Gini coefficient can capture the proportion of values in a distribution that are the same or all near-zero.

$$G(\mathbf{x}) = \frac{\sum_{i=1}^{n} \sum_{j=1}^{n} |x_i - x_j|}{2n \sum_{i=1}^{n} x_i}$$

We estimate the ratio of variables in pseudominima with the Gini coefficient on the derivative of the global solution $\mathbf{G}$ to take the distance from 0 into account.

$$R_G(\mathbf{x}) = G(\nabla f(\mathbf{x}) \cup \{0\})$$

Furthermore, as variables reach a minimum, the distribution of partials should be dispersed evenly, so the Gini measure should approach one.

5.3 Hypotheses

Previous work has established that CCEA and CPSO are successful in solving problems with a high degree of epistasis and low separability [1,20]. Additional work on the FEA and OSI have demonstrated improvement over CPSO and PSO in with non-separable landscapes [24,25]. In the memetic setting, PSO has had varied success in incorporating gradient descent [4], and CPSO has had similar improvement over their regular counterparts [3,18]. With FEA and OSI, Hybrid-FEA like Hybrid-CPSO, has shown improved convergence in problems with low separability [26]. It is hypothesized that memetic variations of OSI will show similar improved convergence as separability decreases in the benchmark problems compared to PSO and CPSO counterparts.

Table 2. Final objective values for each algorithm variant $\mathcal{M}$ and algorithm class $\mathcal{C}$ on naturally separable basis functions, across degrees of separation (Sep.) for each family $\mathcal{F}$ of basis functions.

Average Final Objective Values										
Sep. \ $\mathcal{F}$	Elliptic-based			Rastrigin-based			Ackley-based			$\mathcal{F}$ / $\mathcal{C}$
S	$1.44{\cdot}10^{9}$	—	0	$1.99{\cdot}10^{3}$	—	$4.4{\cdot}10^{3}$	19.8	—	19.8	PSO
	17.7	0	**0** *	**48.1** *	581	279 *	0	**0** *	1.69 *	CPSO
	0	0	0	73.3	279 *	314	0	0	2.01	OSI
1	$7.56{\cdot}10^{12}$	—	$2.92{\cdot}10^{11}$ *	$2.22{\cdot}10^{8}$ *	—	$2.17{\cdot}10^{8}$ *	$6.5{\cdot}10^{6}$ *	—	$\mathbf{5.29{\cdot}10^{6}}$ *	PSO
	$6.38{\cdot}10^{14}$	$\mathbf{4.73{\cdot}10^{10}}$ *	$4.18{\cdot}10^{14}$	$4.73{\cdot}10^{8}$	$3.11{\cdot}10^{8}$	$7.8{\cdot}10^{8}$	$2.03{\cdot}10^{7}$	$2{\cdot}10^{7}$	$2.08{\cdot}10^{7}$	CPSO
	$4.4{\cdot}10^{14}$	$2.66{\cdot}10^{11}$	$2.17{\cdot}10^{14}$	$5.62{\cdot}10^{8}$	$\mathbf{1.92{\cdot}10^{8}}$ *	$7.6{\cdot}10^{8}$	$2.06{\cdot}10^{7}$	$1.95{\cdot}10^{7}$ *	$2.05{\cdot}10^{7}$	OSI
$N/2m$	$9.03{\cdot}10^{7}$ *	—	$5.7{\cdot}10^{6}$ *	$3.1{\cdot}10^{4}$	—	$2.89{\cdot}10^{4}$	84.1	—	69.2	PSO
	$1.11{\cdot}10^{9}$	$1.97{\cdot}10^{7}$	$1.06{\cdot}10^{9}$	$7.56{\cdot}10^{3}$	$\mathbf{5.82{\cdot}10^{3}}$ *	$9.02{\cdot}10^{3}$	55	40	62.8 *	CPSO
	$6.54{\cdot}10^{8}$	$\mathbf{9.04{\cdot}10^{5}}$ *	$9.38{\cdot}10^{8}$	$6.97{\cdot}10^{3}$ *	$6.18{\cdot}10^{3}$	$7.72{\cdot}10^{3}$ *	54.8	**3.79** *	67.7	OSI
N/m	$2.42{\cdot}10^{8}$	—	$\mathbf{3.12{\cdot}10^{5}}$ *	$4.14{\cdot}10^{4}$	—	$5.12{\cdot}10^{4}$	403 *	—	400 *	PSO
	$3.23{\cdot}10^{7}$	$1.96{\cdot}10^{7}$	$2.34{\cdot}10^{7}$	$1.63{\cdot}10^{4}$	$\mathbf{1.08{\cdot}10^{4}}$	$1.79{\cdot}10^{4}$	429	**400** *	429	CPSO
	$3.28{\cdot}10^{7}$	$1.94{\cdot}10^{7}$	$2.76{\cdot}10^{7}$	$1.44{\cdot}10^{4}$ *	$1.1{\cdot}10^{4}$	$1.83{\cdot}10^{4}$	429	400	429	OSI
Sep. \ $\mathcal{M}$	∅	FGD	BCD	∅	FGD	BCD	∅	FGD	BCD	$\mathcal{C}$ / $\mathcal{M}$

Van den Bergh and Engelbrecht claimed that Hybrid-CPSO had improved success in non-separable problems because alternation to full PSO, which optimized over all variables at once, bridged gaps between the subspaces of the subpopulations, improving cooperation after competition [1]. Similar improvement may result with Hybrid-FEA to offset too extreme competition at the points of overlap [26]. As additional hypotheses, the FGD memetic variant optimizes over all variables of $\mathbf{G}$ in alternation, so improvement of the context vector may improve the performance of the subpopulations by effectively improving cooperation with other subpopulations. The BCD variant optimizes only within the subpopulation, so it is not expected BCD will have the same success as FGD,

The BCD and FGD variations choose a descent direction, resulting in being similar to a perturbation-based method. Perturbation-based gradient descent methods [6,7] are known to be able to escape saddle-points; however, they may not be able to escape saddle-points when only optimizing over a factor at a pseudominimum. Nonetheless, BCD may see improvement from the stochastic and

random nature of the descent direction, so it may be able to escape anyways. PSO should be able to escape similarly, but marginal improvement is expected with BCD. An improved context vector $\mathbf{G}$ with FGD may permit a subpopulation to escape pseudominima while also having the benefit of improved search.

6 Results

Tables 2 and 3 show the final objective values obtained for each of the methods based on an average of 25 trials. The results are grouped by the family of basis functions $\mathcal{F}$, where each row group is the underlying algorithm class $\mathcal{C}$ of PSO, CPSO or OSI, a column represents the memetic variant $\mathcal{M}$ applied ($\varnothing$ indicates no memetic variation), and within each algorithm block are the results for each degree of separability (See Table 1). The best algorithm for each function, by family and separability, is bolded. An asterisk $*$ and triangle ◤ denote the significantly best algorithm class $\mathcal{C}$ for each memetic variant $\mathcal{M}$, and the best memetic variant $\mathcal{M}$ for each class $\mathcal{C}$ respectively. A p-value of 0.05 was used to test significance with Kruskal–Wallis and Dunn's test. Finally, the results are visualized in Fig. 1, with the average objective value of the global solution $\mathbf{G}$ and the Gini coefficient of the global solution gradient $R_G(\nabla f(\mathbf{G}))$ plotted over the number of function evaluations (FEs) for each benchmark.[1]

Table 3. Final objective values for each algorithm variant $\mathcal{M}$ and algorithm class $\mathcal{C}$ on naturally non-separable basis functions, across degrees of separation (Sep.) for each family $\mathcal{F}$ of basis functions.

Average Final Objective Values							
Sep. \ $\mathcal{F}$	Schwefel-based			Rosenbrock-based			$\mathcal{F}$ \ $\mathcal{C}$
1	$1.19{\cdot}10^{9}$ $*$	—	**0** $*$	$2.15{\cdot}10^{9}$	—	$6.15{\cdot}10^{7}$ $*$	PSO
	$2.06{\cdot}10^{11}$	$2.03{\cdot}10^{7}$	$1.66{\cdot}10^{11}$	$2.83{\cdot}10^{8}$	$8.75{\cdot}10^{7}$	$1.58{\cdot}10^{8}$	CPSO
	$2.12{\cdot}10^{11}$	$2.04{\cdot}10^{7}$	$1.58{\cdot}10^{11}$	$1.5{\cdot}10^{8}$	**$5.64{\cdot}10^{7}$** $*$	$1.5{\cdot}10^{9}$	OSI
$N/2m$	$2.17{\cdot}10^{4}$ $*$	—	**3.38** $*$	$4.66{\cdot}10^{10}$	—	**298** $*$	PSO
	$3.28{\cdot}10^{5}$	944 $*$	$1.95{\cdot}10^{5}$	$1.2{\cdot}10^{5}$	$1.07{\cdot}10^{5}$	$2.71{\cdot}10^{4}$	CPSO
	$2.29{\cdot}10^{5}$	$1.78{\cdot}10^{3}$	$3.06{\cdot}10^{5}$	$3.87{\cdot}10^{4}$	$9.61{\cdot}10^{4}$	$3.79{\cdot}10^{5}$	OSI
N/m	$1.59{\cdot}10^{3}$ $*$	—	**0.723** $*$	$1.45{\cdot}10^{8}$	—	**$1.4{\cdot}10^{3}$** $*$	PSO
	$1.62{\cdot}10^{4}$	$8.5{\cdot}10^{3}$	$1.14{\cdot}10^{4}$	$1.39{\cdot}10^{4}$	$4.66{\cdot}10^{3}$	$2.41{\cdot}10^{4}$	CPSO
	$1.27{\cdot}10^{4}$	$8.03{\cdot}10^{3}$ $*$	$1.56{\cdot}10^{4}$	$1{\cdot}10^{4}$	$5.49{\cdot}10^{3}$	$6.22{\cdot}10^{5}$	OSI
NS	$2.35{\cdot}10^{6}$ $*$	—	**$3.72{\cdot}10^{5}$** $*$	$6.57{\cdot}10^{4}$	—	$2.2{\cdot}10^{3}$ $*$	PSO
	$2.04{\cdot}10^{7}$	$2.12{\cdot}10^{6}$ $*$	$2.56{\cdot}10^{7}$	$2.46{\cdot}10^{3}$	$1.21{\cdot}10^{3}$	$3.02{\cdot}10^{3}$	CPSO
	$1.5{\cdot}10^{7}$	$2.66{\cdot}10^{6}$	$1.51{\cdot}10^{7}$	$2.41{\cdot}10^{3}$	**$1.1{\cdot}10^{3}$**	$3.75{\cdot}10^{3}$	OSI
Sep. \ $\mathcal{M}$	$\varnothing$	FGD	BCD	$\varnothing$	FGD	BCD	$\mathcal{C}$ \ $\mathcal{M}$

The general trend of best memetic variant was FGD for CPSO and OSI, and BCD for PSO. However, it appeared OSI+FGD and CPSO+FGD had the most consistent performance, often outperforming other methods and having similar convergence across the functions.

[1] Our supplementary document provides plots for all benchmarks.

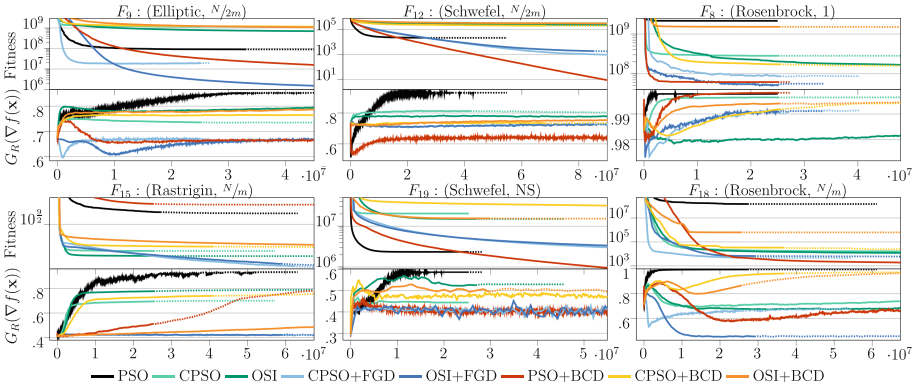

Fig. 1. Six representative plots of average objective value and derivative Gini measure across FEs. Columns are by separability properties. Dotted lines show at least one trial has ceased, and the line stops when all trials cease running.

In the Schwefel-family, PSO+BCD outperformed all other algorithms. Interestingly, plain PSO outperformed all OSI and CPSO variations in the two most non-separable Schwefel problems. We suspect this behavior is from the high epistasis of the Schwefel basis problem, which unlike the other basis families, has all variables $\mathbf{X}$ contribute non-separably to a sum simultaneously. This high degree of epistasis may require all variables to be optimized simultaneously, which is best-suited in the PSO setting. The inability of CPSO and OSI to outperform PSO, especially in the most non-separable problems, may be related to having suboptimal factor architectures.

Consider the Gini measure of variables at critical points in Fig. 1. When performance of an algorithm stagnates, the measure appears to plateau or flatten, but in the case of CPSO, OSI, and their variants, the measure plateaus and sticks to lower values than PSO or BCD variants. This appears particularly true for CPSO and OSI in most experiments. This supports previous results that subpopulations are sensitive to pseudominima, affecting the global solution.

Across the majority of benchmark problems, the global solutions for OSI and CPSO appear to be unable to reach local minima, unlike PSO and its memetic variants. The most likely reason for the inability to reach a local minimum is related to how solutions from the subpopulations are incorporated into the global solution. In our competition step implementation, a hill climbing-like improvement is made for each variable; it tries every subpopulation individual on each variable, and tests if fitness is improved if that individual variable is incorporated into the global solution. This hill-climbing behavior may make it difficult to incorporate improved solutions from the subpopulations, because multiple variables may need to be incorporated at once to have an improved solution.

With the FGD and BCD variants, the global solution seldom converges to a local optimum, although it improved in the case of CPSO and OSI. In cases where the solution reaches a full local optimum, performance stagnates, and improve-

ment dramatically slows and stops. If during the beginning and throughout the algorithm, the measure of critical points remains low and does not plateau into a local optimum, the algorithms appear to have better performance and are generally able to find better solutions.

The Gini measure of the global derivative has the property of gauging how many variables are minimum, while also representing the exploration of a variable. Consider then, the Gini measure when fitness is improving actively, and has not stagnated. Lower values indicate regions of active descent to a minimum, effectively providing a measure of how effective exploration is. Consider the measure for PSO+BCD; in several cases, the algorithm achieves a low Gini measure and outperforms on multiple benchmarks. In F_{18}, PSO+BCD starts at a high Gini measure, and partial derivatives are more evenly dispersed because most variables are improving. Later, the measure falls to lower values, implying less uniform variable improvement but also indicating effective exploration with relatively few variables are at minimum. Finally, as fitness flattens, the measure increases to a stagnation level, but not to a local minimum. In the case of OSI+FGD in F_{18}, a lower Gini measure does not necessarily mean improvement; it may mean optimization is happening on too few variables simultaneously because there is too much exploitation.

7 Conclusion

After 30 years, the original PSO framework continues to show its resiliency in settings of very high epistasis and non-separability, although still unfortunately susceptible to premature convergence. CPSO was designed to improve optimization by solving separable subproblems individually, and OSI was intended to address epistasis and non-separability via the overlapping regions. One of the issues with CCEA and OSI methods is that of getting caught in pseudominima. This work attempted to address the pseudominimum and premature convergence problem originally described in CPSO and FEA works [1,25].

In this paper, we demonstrated benefit of incorporating memetic variants in the OSI framework. We considered options whereby block coordinate descent was used with each factor. We also applied full gradient descent with the global context vector. BCD was shown to be more effective than their non-memetic counterparts, and had the best performance with PSO. Gradient descent on the individual particles in the population tended to cause subpopulations to reach pseudominima sooner, thus preventing improvement in the global context vector. On the other hand, full gradient descent on the global context yielded the best results overall in terms of consistency for CPSO and OSI.

Our use of the Gini coefficient on a solution's gradient gives insight on convergence of the subpopulations, by measuring the number of variables at critical points. Results showed that during periods of stagnation, minimizing the number of variables at critical points helps exploration, but should simultaneously avoid getting stuck in those pseudominima by not being too greedy with exploitation.

Future work will investigate improved OSI competition to prevent pseudominima. Such methods as dynamic factor architectures and OSI/full-PSO

hybrids can be adapted to the memetic case. We will address issues of potential blockades in the fitness landscape, by improving communication between the subpopulations and context vector. The approach will consider complete subset competition among subpopulations and overlap regions rather than element-wise competition of these architecture components.

References

1. Van den Bergh, F., Engelbrecht, A.P.: A cooperative approach to particle swarm optimization. IEEE Trans. Evol. Comput. **8**(3), 225–239 (2004)
2. Butcher, S.G.W., Strasser, S., Hoole, J., Demeo, B., Sheppard, J.W.: Relaxing consensus in distributed factored evolutionary algorithms. In: ACM Genetic and Evolutionary Computation Conference (GECCO), pp. 5–12 (2016)
3. Cao, Z., Wang, L., Shi, Y., Hei, X., Rong, X., Jiang, Q., Li, H.: An effective cooperative coevolution framework integrating global and local search for large scale optimization problems. In: IEEE Congress on Evolutionary Computation (CEC), pp. 1986–1993 (2015)
4. Cui, X., Zhang, W., Tüske, Z., Picheny, M.: Evolutionary stochastic gradient descent for optimization of deep neural networks. In: Advances in Neural Information Processing Systems (2018)
5. Davidor, Y.: Epistasis variance: suitability of a representation to genetic algorithms. Complex Syst. **4**(4), 369–383 (1990)
6. Du, S., Lee, J., Li, H., Wang, L., Zhai, X.: Gradient descent finds global minima of deep neural networks. In: International Conference on Machine Learning, pp. 1675–1685. PMLR (2019)
7. Du, S.S., Jin, C., Lee, J.D., Jordan, M.I., Singh, A., Poczos, B.: Gradient descent can take exponential time to escape saddle points. In: Advances in Neural Information Processing Systems (2017)
8. Dzahini, K.J.: Expected complexity analysis of stochastic direct-search. Comput. Optim. Appl. **81**, 179–200 (2022)
9. Elad, M., Matalon, B., Zibulevsky, M.: Coordinate and subspace optimization methods for linear least squares with non-quadratic regularization. Appl. Comput. Harmon. Anal. **23**(3), 346–367 (2007)
10. Fortier, N., Sheppard, J., Strasser, S.: Abductive inference in bayesian networks using distributed overlapping swarm intelligence. Soft. Comput. **19**(4), 981–1001 (2015)
11. Fortier, N.L.: Inference and learning in Bayesian networks using overlapping swarm intelligence. Ph.D. thesis, Department of Computer Science, Montana State University (2015)
12. Ge, R., Huang, F., Jin, C., Yuan, Y.: Escaping from saddle points-online stochastic gradient for tensor decomposition. In: Conference on Learning Theory, pp. 797–842. PMLR (2015)
13. Ge, R., Lee, J.D., Ma, T.: Matrix completion has no spurious local minimum. In: Advances in Neural Information Processing Systems (2016)
14. Jafari, S., Kapitaniak, T., Rajagopal, K., Pham, V.-T., Alsaadi, F.E.: J. Zhejiang Univ.-Sci. A **20**(2), 109–116 (2018). https://doi.org/10.1631/jzus.A1800399
15. Kolda, T.G., Lewis, R.M., Torczon, V.: Optimization by direct search: new perspectives on some classical and modern methods. SIAM Rev. **45**(3), 385–482 (2003)

16. Larson, J., Menickelly, M., Wild, S.M.: Derivative-free optimization methods. Acta Numer. **28**, 287–404 (2019)
17. Lu, W., Cai, B., Gu, R.: Improved particle swarm optimization based on gradient descent method. In: ACM International Conference on Computer Science and Application Engineering (2020)
18. Molina, D., Lozano, M., Herrera, F.: MA-SW-chains: memetic algorithm based on local search chains for large scale continuous global optimization. In: IEEE Congress on Evolutionary Computation, pp. 1–8 (2010)
19. Pillai, K.G., Sheppard, J.W.: Overlapping swarm intelligence for training artificial neural networks. In: IEEE Swarm Intelligence Symposium (SIS), pp. 1–8 (2011)
20. Potter, M.A., De Jong, K.A.: A cooperative coevolutionary approach to function optimization. In: Davidor, Y., Schwefel, H.-P., Männer, R. (eds.) PPSN 1994. LNCS, vol. 866, pp. 249–257. Springer, Heidelberg (1994). https://doi.org/10.1007/3-540-58484-6_269
21. Shi, Y., Teng, H., Li, Z.: Cooperative co-evolutionary differential evolution for function optimization. In: Wang, L., Chen, K., Ong, Y.S. (eds.) ICNC 2005. LNCS, vol. 3611, pp. 1080–1088. Springer, Heidelberg (2005). https://doi.org/10.1007/11539117_147
22. Smith, J.E.: Self-adaptative and coevolving memetic algorithms. In: Handbook of Memetic Algorithms, pp. 167–188. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-23247-3_11
23. Solis, F.J., Wets, R.: Minimization by random search techniques. Math. Oper. Res. **6**(1), 19–30 (1981)
24. Strasser, S., Sheppard, J., Butcher, S.: A formal approach to deriving factored evolutionary algorithm architectures. In: IEEE Symposium Series on Computational Intelligence (SSCI), pp. 1–8 (2017)
25. Strasser, S., Sheppard, J., Fortier, N., Goodman, R.: Factored evolutionary algorithms. IEEE Trans. Evol. Comput. **21**(2), 281–293 (2016)
26. Strasser, S., Sheppard, J.W.: Convergence of factored evolutionary algorithms. In: ACM/SIGEVO Conference on Foundations of Genetic Algorithms, pp. 81–94 (2017)
27. Tang, K., Li, X., Suganthan, P.N., Yang, Z., Weise, T.: Benchmark functions for the CEC'2010 special session and competition on large-scale global optimization. Nat. Insp. Comput. Appl. Lab. USTC, China **24**, 1–18 (2007)
28. Zhan, Z.H., Shi, L., Tan, K.C., Zhang, J.: A survey on evolutionary computation for complex continuous optimization. Artif. Intell. Rev. **55**(1), 59–110 (2022)