Swarm and Evolutionary Computation 50 (2019) 100493

Contents lists available at ScienceDirect

Swarm and Evolutionary Computation

journal homepage: www.elsevier.com/locate/swevo

Speed up differential evolution for computationally expensive protein structure prediction problems

Hojjat Rakhshani *, Lhassane Idoumghar, Julien Lepagnot, Mathieu Brévilliers

IRIMAS, Universié de Haute-Alsace (UHA), 12 rue des Frères Lumière, 68093, Mulhouse, France

ARTICLE INFO

Keywords:
Evolutionary computation
Surrogate-assisted optimization
Machine learning
Protein structure prediction
Meta-heuristics

ABSTRACT

Protein structure prediction (PSP) plays an important role in the field of computational molecular biology. Although powerful optimization algorithms have been proven effective to tackle the PSP, researchers are faced with the challenge of time consuming simulations. This paper introduces a new modification of differential evolution (DE) which makes use of the computationally cheap surrogate models and gene expression programming (GEP) in order to address the aforementioned issue. The incorporated GEP is used to generate a diversified set of configurations, while radial basis function (RBF) surrogate model helps DE to find the best set of configurations. In addition to this, covariance matrix adaptation evolution strategy (CMAES) is also adopted to explore the search space more efficiently. The introduced algorithm, called SGDE, is tested on real-world proteins from the Protein data bank (PDB) using both a simplified and an all-atom model. The experiments show that SGDE performs better than the state-of-the-art algorithms on the PSP problems in both terms of the convergence rate and accuracy. In the case of run time complexity, SGDE significantly outperforms the other competitive algorithms for the adopted all-atom model.

1. Introduction

The PSP represents the optimization problem of how to determine the 3D structure of proteins from their primary sequence. Generally speaking, it is characterized by arranging a sequence of basic elements a-helix, b-strand and coil. Determining 3D structures of a protein can affect its functions and is vitally important for rational drug design. The early works on PSP subject, i.e., X-ray crystallography and Nuclear Magnetic Resonance (NMR) were based on experimental techniques. In Ref. [1], authors adopted NMR to determine the structure of the tetrameric M2 (M2 is of influenza virus and forms pH-gated proton channels in the viral lipid envelope channel) in complex with rimantadine. In another research [2], authors put forwarded a method for structural characterization of mitochondrial uncoupling protein 2 (UCP2) based on NMR. Furthermore, Oxenoid et al. [3] reported the structure of the pore domain of MCU from Caenorhabditis elegans, determined using NMR and electron microscopy. In Ref. [4], authors applied NMR to determine an atomic structure of the transmembrane domain of HIV-1 Env reconstituted in bicelles that mimic a lipid bilayer.

The aforementioned methods are very expensive and time-consuming. In the case of X-ray diffraction, not all proteins can be successfully crystallized. Besides, most of the membrane proteins are difficult to crystallize and they will not dissolve in normal solvents [5]. Moreover, the question arises as to whether structure in single crystals adequately characterizes the protein conformation in a complex and dynamic environment of living cells [6]. NMR is indeed a very powerful tool in determining the 3D structures of membrane proteins, but the interpretation of NMR spectra is very complex, and the assignment of interproton distances is not always feasible. Also, a major issue is purifying the protein and reconstituting it in a model membrane medium that supports protein solubility and stability [2]. These practical limitations stimulate continual progress in the development of various structural bioinformatics tools for the PSP problem.

The PSP problem becomes easier if similar proteins (also referred to as templates) are found in the PDB. In this case, a database of known structures could be used to find high-resolution models by aligning target sequences to the solved similar structures. Otherwise, we need to build a model from scratch without the explicit use of templates (i.e. ab initio PSP). In this regard, Chou et al. [7] proposed to use the three-dimensional structure of the catalytic domain of caspase-9 and its interaction with the inhibitor acetyl-Asp-Val-Ala-Asp fluoromethyl ketone (Ac-DVAD-fmk) have been predicted by a segment matching modeling procedure. In

* Corresponding author.
E-mail address: hojjatrakhshani@gmail.com (H. Rakhshani).

https://doi.org/10.1016/j.swevo.2019.01.009
Received 1 March 2018; Received in revised form 24 January 2019; Accepted 25 January 2019
Available online 28 January 2019

Ref. [7], the tertiary structure of the protease domain of caspase-8, also called FLICE, has been predicted by a segment match modeling procedure (caspase-3 structure was used as a template for modeling the protease domain of caspase-8).

The 3D structure obtained by homology modeling is very sensitive to the sequence alignment of the query protein with the structure-known protein (template) [8]. In addition, not all proteins with unknown structures have ideal templates. Furthermore, recent CASP10 experiment reveals the fact that the prediction process for the hard targets with unknown templates has not been improved [9]. Up to September 2014, there had been only 100,000 proteins in PDB compared to 80 million unknown protein sequences in UniprotKB/TrEMBL. This encourages the need for developing ab initio PSP methods.

A successful ab initio model depends on two components: a powerful search algorithm and an accurate potential energy function. The later consideration is helpful to distinguish between the native and non-native structures of the proteins. Despite significant advances in computational capabilities, the simulation cost of such energy functions is still too expensive (approximately 150 CPU days for a small protein $<$100 residues) [10]. So, researchers have adopted simplified models in order to develop and test their new search algorithms. This includes instances of very simplified models [11], as well as for more complex all-atom models where the energy functions is computed based on experiments in physics, chemistry and calculations in quantum mechanics [12,13]. However, it should be noted that even the simplest models are still too complex from the computational point of view.

Although these models give rise to a large number of studies for proposing more effective and rapid algorithms, there is still room for further improvement due to the growing complexity of the PSP problem. A potential solution might be found by adapting the machine learning (ML) techniques. In Ref. [14], Zhang et al. presented a taxonomy of the ML enhanced meta-heuristics in order to study the interest of such integration. The most important insight which has resulted from this research is that insertion of ML techniques usually leads to: 1) speeding up the search process and 2) improving the quality of the solutions. Subsequently, this study takes a more detailed look on the importance of the ML techniques for the PSP. The implementation steps are based on surrogate modeling, genetic programming (GP) and the CMAES algorithm which are incorporated into the DE. The standard DE algorithm and its variants are among noteworthy approaches for the PSP problem [11] and are considered in this study.

The first component known as the approximation model, meta-model or response surface model is computationally cheap mathematic model which has been successfully employed to replace expensive fitness functions in time-demanding real-world applications. The Gaussian process/Kriging, polynomial regression (PR), RBF, radial basis neural network (RBNN) and support vector regression (SVR) are well-known meta-models which have been reported in literature [15]. They have exhibited superior performance in solving computationally expensive dynamic, multi-objective and single-objective optimization problems. The advantages of the cheap surrogate models become clear where they could be used to explore a large area of the configuration space using far less computation time. This is due to the fact that conventional optimization algorithms are highly sensitive to their search strategies and associated control parameters and it is necessary to perform a trial-and-error search to find the best combination for the problem at hand. Consequently, fitness functions need to be frequently evaluated which results in costly computational expenses. Motivated by this aspect, we employed a surrogate model based on RBF in order to evaluate the quality of the different configurations using less number of fitness evaluations. The proposed algorithm in this research is applied on the simplified AB off-lattice and on an all-atom model under a classical force field.

As a novel feature in DE, the second consideration integrates mutation strategies of DE by the generated ones using a GP technique. The introduced GEP [16] enables us to have a diversified pool of mutation search operators for the individuals in the population. The GEP produces linear and non-linear extrapolation of the individuals which results in different combinations of solutions. This differs from previous studies that use a predefined set of configurations to generate competitive trial vectors.

As the final modification, the standard DE is equipped with the CMAES [17] algorithm in order to learn and takes into account dependencies of the optimization parameters. On large scale problems with high dependencies, CMA-ES becomes a valuable alternative and could reduce the number of required function evaluations.

The rest of the paper is organized as follows. Section 2 provides us with a literature survey regarding the application of different parameter adaptation in DE. Section 3 briefly reviews the adopted protein models. In Section 4, we introduce the RBF surrogate. In Section 5, we present the conventional DE algorithm. Section 6 elaborates technical details of our proposed algorithm. In Section 7, performance of the SGDE is examined by performing experiments on several real-world PSP problems using both a simplified and an all-atom model, followed by a conclusion in the last section.

2. Related works

It has been frequently shown that choosing appropriate parameter settings and mutation strategy for DE is not a trivial task [18]. This is due to the complex interactions between the decision variables and characteristics of the problem at hand. An inappropriate mutation strategy may lead to the so-called premature convergence or stagnation problems which result in a time consuming and costly optimization process. Therefore, DE variants based on parameter adaptations and ensembles of mutation strategies attract increasing attention. For example, SaDE [19] uses its previous successful experiences for gradually self-adapting both trial vector generation strategies and their associated parameter settings. Interestingly, jDE [20] adds F and CR at the individual level and accordingly more promising solutions will propagate also better parameter settings. In CoDE [21], trial vector generation is based on a pool of three control parameter settings and mutation strategies. At each generation, this configuration pool is used to create three trial vectors and the best solutions goes for the selection step. In a similar way, EPSDE [22] employs an ensemble of mutations and parameter settings to improve the results of the standard DE. In another study, JADE [23] introduces a new mutation strategy based on an archive and a parameter adaptation strategy. More precisely, the JADE keeps recent successful crossover and amplification values for each individual in order to generate new parameter settings. Similarly, SHADE [24] adopts a historical memory of good settings to adjust F and CR values for each individual. Remarkably, L-SHADE [25] further improves the SHADE by incorporating a linear population size reduction strategy.

The aforementioned algorithms share a common characteristic: they attempt to find the best DE's configurations for the problem at hand. Consequently, recent works tried to further improve the results by applying cheap surrogate models to adapt the best parameters using less evaluations. The SA-DE-DPS [26] incorporated the surrogates to dynamically select the best set of the amplification factor, crossover rate and population size. In another work [27], authors put forward LES-CDE which adopts an ensemble of several adjacent local surrogates to have more promising trial vectors in the crowding DE (CDE). Mallipeddi et al. [28] introduced ESMDE to enhance the performance of the EPSDE by means of a Kriging model.

To further continue the research in this direction, we developed a novel approach by utilizing the surrogate models. As illustrated in Fig. 1, the main point to distinguish the proposed approach from the previously surrogate assisted works lies in the aim of the integration.

3. The adopted models

The search for an appropriate computational method for PSP and the

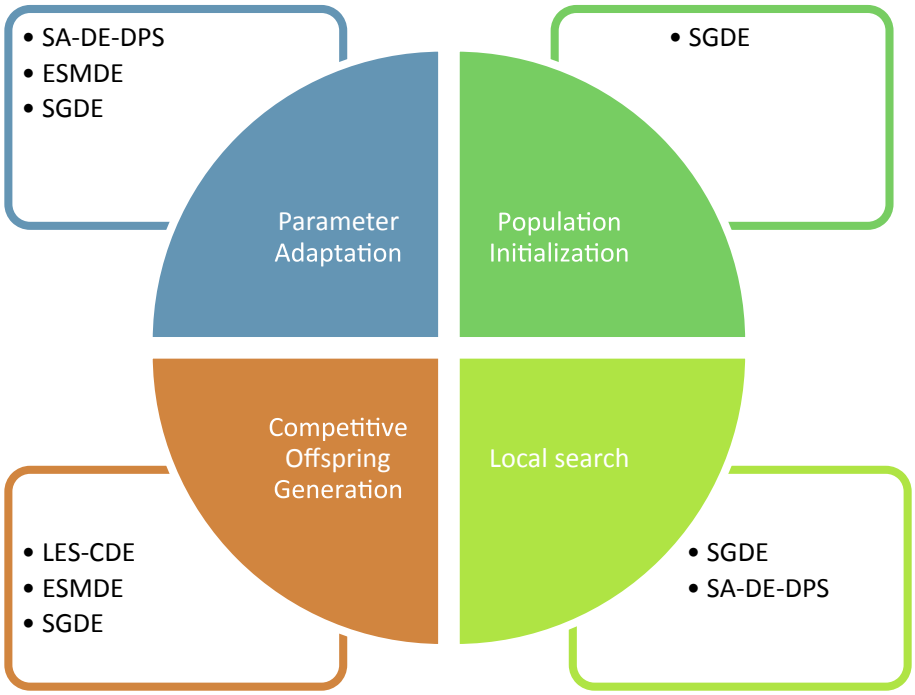


Fig. 1. Comparing the aim of integrating surrogates into the DE algorithm based on the SA-DE-DPS, ESMDE, LES-CDE and SGDE.

protein-ligand docking has led to the development of models which make use of interaction between the atoms in order to assess the quality of a given solution. This study compares performance of the SGDE and other methods from the literature using two models. The first one is 3D AB off-lattice model which takes less computational time and is useful to analyze and benchmark the behavior of the SGDE during the development process. The second model is a more complicated all-atom model under a classical force field energy function.

3.1. The 3D AB off-lattice model

This section explains how we can define three-dimensional structure of a protein by a set of bond/torsional angles, and unit-length bonds between two amino-acids. The AB off-lattice model is among the most popular models for the PSP problem [29]. This model computes the free-energy function value of a protein from its hydrophobic interactions. It assumes that all types of amino acid residues fall into two hydrophobic (labeled as 'A') and hydrophilic (labeled as 'B') species of monomers. More precisely, Aspartic acid, Glutamic acid, Phenylalanine, Histidine, Lysine, Asparagine, Glutamine, Arginine, Serine, Threonine, Tryptophan and Tyrosine are supposed to be hydrophilic; and Isoleucine, Tyrosine, Leucine, Proline, Cysteine, Methionine, Alanine and Glycine are classified as hydrophobic. These monomers are linked up by unit-length chemical bonds to form a structural chain in the three dimensional space. Conformation of such a chain with n amino acids will be determined by horizontal bond angles $\boldsymbol{\theta} = [\theta_1, \theta_2, \ldots, \theta_{n-2}]$ and torsional angles $\boldsymbol{\beta} = [\beta_1, \beta_2, \ldots, \beta_{n-3}]$. Here, $\boldsymbol{\theta}$ is used to consider the angles between adjacent amino acids in XOY plane while $\boldsymbol{\beta}$ takes into account the angles for corresponding amino acids and XOY plane. Thus, the position of amino acids is determined by (1) [29]:

$$(x_i, y_i, z_i) = \begin{cases} (0,0,0) & i = 1 \\ (0,1,0) & i = 2 \\ (\cos(\theta_1), \sin(\theta_1) + 1, 0) & i = 3 \\ \begin{cases} x_{i-1} + cos(\theta_{i-2}) \times cos(\beta_{i-3}) \\ y_{i-1} + sin(\theta_{i-2}) \times cos(\beta_{i-3}) \\ z_{i-1} + \sin(\beta_{i-3}) \end{cases} & 4 \le i \le n \end{cases} \quad (1)$$

After defining a unique structure with the above mentioned properties, the free energy of a protein can be computed. To do so, the AB off-lattice model considers the intra-molecular bending potential energy of the backbone v_1 and non-bonded interactions v_2; as given in (2):

$$\text{energy\_func} = \sum_{i=2}^{n-1} v_1(\theta_i) + \sum_{i=1}^{n-2} \sum_{j=i+2}^{n} v_2(r_{i,j}, \xi_i, \xi_j) \quad (2)$$

Here, v_1 is defined as follow:

$$v_1(\theta_i) = \frac{1}{4}(1 - \cos(\theta_i)) \quad (3)$$

and the non-bonded interactions v_2 is computed using Eqs. (4) and (5), where $r_{i,j}$ represents the Euclidean distance between two residues i and j.

$$v_2(r_{i,j}, \xi_i, \xi_j) = 4 \times \left[r_{i,j}^{-12} - C(\xi_i, \xi_j) r_{i,j}^{-6} \right] \quad (4)$$

$$C(\xi_i, \xi_j) = \frac{1}{8}(1 + \xi_i + \xi_j + 5\xi_i\xi_j) \quad (5)$$

Thereby, the main task is to find parameter settings $[\theta_1, \theta_2, \ldots, \theta_{n-2}; \beta_1, \beta_2, \ldots, \beta_{n-3}]$ with a minimum free energy state.

3.2. The AMBER force field

Force fields are a class of computational methods which measure the interaction energy of the atoms with each other to determine whether a model is native or not. They do so by means of the purely physics-based principles, quantum-based calculations or empirical-based fitting approaches. As the energy value reduces, native conformation is more likely to be detected. The force fields get coordinates of the atoms in 3D dimensional space and output its energy value. The geometry of chemical bonds is fixed during the PSP process and the coordinates of the proteins can be modified by changing the dihedral and torsion angles. Thus, a solution representation approach commonly used by PSP methods is based on backbone and side-chain angles. There are three backbone dihedral angles ϕ, ψ and ω. As shown in Fig. 2, ϕ is the angle of right-handed rotation around N-CA bond, ψ defined around CA-C bond and ω rotates about C—N bond. Depending on their amino acid sequence,

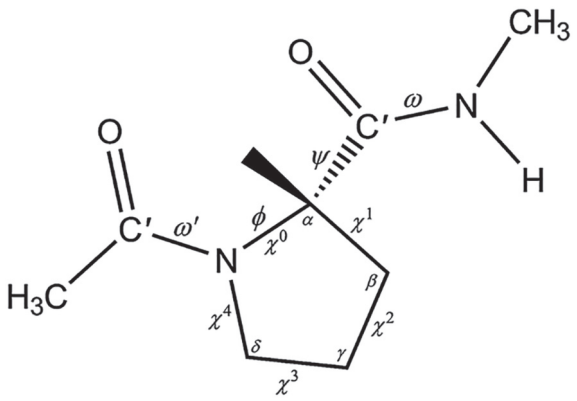


Fig. 2. A representation of the dihedral angles ϕ (C-N-CA-C), ψ (N-CA-C-N) and ω (CA-C-N-CA).

proteins also have different number of side-chain angles namely χ1, χ2, χ3 and χ4. Having these in mind, solution representation for a conformation has been presented in Fig. 3.

This study used the well-known Assisted Model Building with Energy Refinement (AMBER) force field [12]. It is also worth mentioning that the amber force fields ff99SB is used to provide the protein parameters. The functional form of the AMBER force field computes the energy between covalently over bonded atoms, the energy of electron orbitals involved in covalent bonding over the bond angles, the energy for twisting a bond over torsion angles and the non-bonded energy between all atom pairs *i* and *j*; which can be decomposed into van der Waals and electrostatic energies. A detailed study of the other parameters can be found in Ref. [12].

4. The radial basis function

In the optimization context, surrogate models (or meta-models) are a class of mathematical techniques that simulate the behavior of a computationally expensive fitness function. Given solution vector $\mathbf{x}$ and exact fitness function f, a surrogate model is represented as in (6), where ε is the approximation error.

$$\widehat{f}(\mathbf{x}) = f(\mathbf{x}) + \varepsilon \tag{6}$$

For an m-dimensional problem, suppose we have n observations $\mathbf{S}$ and $\mathbf{f}_S$ for a known function f: $\mathbb{R}^m \to \mathbb{R}$ as follows:

$$\mathbf{S} = \left[\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(n)}\right]^T \varepsilon\, \mathbb{R}^{n\times m}, \quad \mathbf{x}^{(i)} = \left\{x_1{}^{(i)}, \ldots, x_m{}^{(i)}\right\} \varepsilon\, \mathbb{R}^m \tag{7}$$

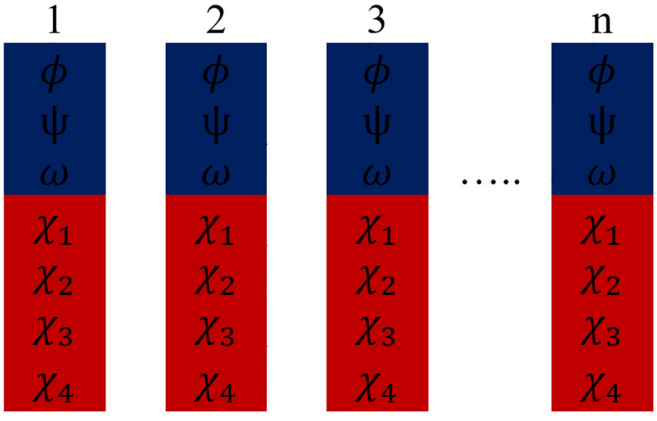


Fig. 3. The proposed encoding representation schema for a protein with *n* residues using the backbone and side-chain dihedral angles.

$$\mathbf{f}_S = \left[f^{(1)}, \ldots, f^{(n)}\right]^T = \left[f\left(\mathbf{x}^{(1)}\right), \ldots, f\left(\mathbf{x}^{(n)}\right)\right]^T \varepsilon\, \mathbb{R}^n \tag{8}$$

In particular, a surrogate model tries to predict the fitness function f for any unseen input vector $\widehat{\mathbf{x}}$ according to the date sets $(\mathbf{S}, \mathbf{f}_S)$.

Among different surrogate models, we utilized the RBF which is a good model for high dimensional problems [30,31]. The RBF model is an interpolation method for scattered multivariate data which takes into account all the sample points. To do so, it adopts linear combinations of a RBF $h(\mathbf{x})$ to approximate a response function $\widehat{f}(\widehat{\mathbf{x}})$ as:

$$\widehat{f}\left(\widehat{\mathbf{x}}\right) = \sum_{i=1}^{n} w_i h\left(\left\|\widehat{\mathbf{x}} - \mathbf{x}^{(i)}\right\|\right) + \mathbf{b}^T\mathbf{x} + a \tag{9}$$

where w_i represents the ith unknown weight coefficient, $\widehat{\mathbf{x}}\, \varepsilon\, \mathbb{R}^m$ is an unseen point, h denotes a RBF and ε are independent errors with variance σ^2. A function h: $\mathbb{R}^m \to \mathbb{R}$ is called a radial function if it satisfies the property $h(\mathbf{x}) = h(\|\mathbf{x}\|)$. Typical radial basis functions are presented in Table 1. Given a suitable kernel h; the unknown parameters a, $\mathbf{b}$, and $\mathbf{w}$ could be obtained by solving the following system of linear equations:

$$\begin{pmatrix} \phi & \mathbf{P} \\ \mathbf{P}^T & 0 \end{pmatrix} \begin{pmatrix} \mathbf{w} \\ \mathbf{c} \end{pmatrix} = \begin{pmatrix} \mathbf{f}_S \\ 0 \end{pmatrix} \tag{10}$$

Here, ϕ is a n × k matrix with $\phi_{ij} = h(\mathbf{x}^{(i)} - \mathbf{x}^{(j)}{}_2)$, $\mathbf{c} = (\mathbf{b}^T, a)^T$ and

$$\mathbf{P}^T = \begin{pmatrix} \mathbf{x}^{(1)} & \mathbf{x}^{(2)} \ldots & \mathbf{x}^{(n)} \\ 1 & 1 \ldots & 1 \end{pmatrix} \tag{11}$$

5. Differential evolution

In evolutionary computation, DE finds a solution by iterative improvement of a candidate solution with regard to a given measure of quality. It is one of the most powerful optimization tools that operate on the basis of the same developmental process in evolutionary algorithms. Nevertheless, different from traditional evolutionary algorithms, DE uses the scaled differences of vectors to produce new candidate solutions in the population. Hence, no separate probability distribution should be used to perturb the population members. The DE is also characterized by the advantages of having few parameters and ease of implementation. The application of DE on engineering and biomedical studies has attracted a high level of interest, concerning its potential [32].

Basically, DE algorithm works through a particular sequence of stages. First, it creates an initial population sampled uniformly at random within the search bounds. Thereafter, three components namely mutation, crossover and selection are adopted to evolve the initial population. The mutation and crossover are used to create new solutions, while selection determines the solutions that will breed a new generation. The algorithm remains inside a loop until stopping criteria are met. In the following, we explain each stage separately in details.

Like other optimization algorithms, DE starts with a randomly initialized population of parameter vectors, the so-called individuals. Each such individual represents an m-dimensional vector of decision

Table 1
Radial basis kernels.

Kernel	Formulation
Linear splines	$\left\|\widehat{\mathbf{x}} - \mathbf{x}^{(i)}\right\|$
Thin-plate splines	$\left\|\widehat{\mathbf{x}} - \mathbf{x}^{(i)}\right\|^k \ln\left\|\widehat{\mathbf{x}} - \mathbf{x}^{(i)}\right\|^k$
Cubic splines	$\left\|\widehat{\mathbf{x}} - \mathbf{x}^{(i)}\right\|^3$
Gaussian	$\exp(-(\left\|\widehat{\mathbf{x}} - \mathbf{x}^{(i)}\right\|^2)/\beta_i)$
Multiquadrics	$\sqrt{1 + (\left\|\widehat{\mathbf{x}} - \mathbf{x}^{(i)}\right\|^2)/\beta_i}$
Inverse multiquadrics	$1 + (\left\|\widehat{\mathbf{x}} - \mathbf{x}^{(i)}\right\|^2)/\beta_i{}^{-1/2}$

variables. The i^{th} individual of the population at generation G can be denoted as follows:

$$\mathbf{x}_{G,i} = \left[x_{G,i,1},\ x_{G,i,2},\ \ldots,\ x_{G,i,j}\right];\ j = 1,\ 2,\ \ldots,\ m \tag{12}$$

For each individual, the values of the decision variables should be restricted to their lower bounds $\mathbf{lb} = [lb_1,\ lb_2,\ \ldots,\ lb_m]$ and upper bounds $\mathbf{ub} = [ub_1,\ ub_2,\ \ldots,\ ub_m]$. Once initialization search ranges have been determined, DE assigns each individual a value from within the specified range as follows [33]:

$$x_{0,i,j} = lb_j + r \times \left(ub_j - lb_j\right);\ i = 1,\ 2, \ldots,\ NP; j = 1,\ 2,\ \ldots, m \tag{13}$$

where $r \in [0,\ 1]$ represents a uniformly distributed random number and NP denotes the population size. After initialization, mutation operator produces new solutions by forming a mutant vector with respect to each parent individual (target vector). For each target vector, its corresponding mutant vector can be generated by different mutation strategies. Each strategy employs different approaches to make a balance between the exploration and exploitation tendencies. For the i^{th} target vector at the G generation, the five most well-known mutation strategies are presented as follows:

$$\mathbf{v}_{G,i} = \mathbf{x}_{G,r1} + F \times (\mathbf{x}_{G,r2} - \mathbf{x}_{G,r3}) \tag{14}$$

$$\mathbf{v}_{G,i} = \mathbf{x}_{G,best} + F \times (\mathbf{x}_{G,r1} - \mathbf{x}_{G,r2}) \tag{15}$$

$$\mathbf{v}_{G,i} = \mathbf{x}_{G,i} + F \times (\mathbf{x}_{G,best} - \mathbf{x}_{G,i}) + F \times (\mathbf{x}_{G,r1} - \mathbf{x}_{G,r2}) \tag{16}$$

$$\mathbf{v}_{G,i} = \mathbf{x}_{G,best} + F \times (\mathbf{x}_{G,r1} - \mathbf{x}_{G,r2}) + F \times (\mathbf{x}_{G,r3} - \mathbf{x}_{G,r4}) \tag{17}$$

$$\mathbf{v}_{G,i} = \mathbf{x}_{G,r1} + F \times (\mathbf{x}_{G,r2} - \mathbf{x}_{G,r3}) + F \times (\mathbf{x}_{G,r4} - \mathbf{x}_{G,r5}) \tag{18}$$

Here $r1, r2, r3, r4, r5 \in \{1, \ldots, NP\}$ are five different randomly generated integer numbers. Furthermore, $F \in [0,\ 2]$ is a scaling factor affecting the difference vector and $best \in \{1, \ldots, NP\}$ is the index of the best individual vector at generation G. The presented strategies in (14)–(18) are called DE/best/1, DE/rand/1, DE/rand-to-best/1, DE/best/2 and DE/rand/2, respectively.

In the next step, DE applies a discrete crossover approach to each pair of the target vector and its corresponding mutant vector. The basic version of DE incorporates the binomial crossover defined as follows [33]:

$$u_{G,i,j} = \begin{cases} v_{G,i,j} & \text{if } \left(r_j \le CR\right) \text{or} \left(j = rand_j\right) \\ x_{G,i,j} & \text{otherwise} \end{cases} \tag{19}$$

In (19), $CR \in [0,\ 1]$ is the user-specified crossover rate which determines the probability of mixing between parent and mutant vectors. Also, $r_j \in [0,\ 1]$ and $rand_j \in [0, m]$ are two randomly picked float and integer numbers, respectively.

Finally, DE adopts a selection mechanism to choose the best individuals according to their fitness for producing the next generation of population. To this end, it compares performance of the trial and target vectors and copies the best one into the next generation; as presented in (20). Here, f is the objective function that should be minimized. To sum up, a detailed pseudocode of the aforementioned DE steps is presented in Fig. 4.

$$\mathbf{x}_{G+1,i} = \begin{cases} \mathbf{u}_{G,i} & \text{if } f(\mathbf{u}_{G,i}) \le f(\mathbf{x}_{G,i}) \\ \mathbf{x}_{G,i} & \text{otherwise} \end{cases} \tag{20}$$

6. The proposed approach

This section provides a comprehensive discussion on the details of the implementation steps in SGDE. In practice, there are two issues that should be addressed when designing a surrogate-assisted algorithm like SGDE. The first one is how to combine both approximated and original fitness values to prevent the algorithm from being misled by a false minimum introduced by the surrogates (i.e., model management) [15]. Whether building a local or global surrogate model is another issue to be addressed regarding the high dimension of the search space. We tried to tackle the aforementioned problems by incorporating both the local and global surrogate models in different stages of the evolution including population initialization, offspring reproduction and parameter adaptation. A generic framework for the surrogate-assisted SGDE algorithm includes some basic steps:

1 Initial sampling using design of experiments (DoE)
2 Population initialization using SRS
3 Offspring reproduction by means of GEP
4 Configuration selection using local surrogate models
5 Parameter adjustment using a global surrogate model
6 Training set update for the surrogate model
7 DE termination if some stopping criteria are satisfied and going to Step 8; otherwise going to Step 3
8 CMAES executing on the best found solution

The first step samples a population of individuals using the Latin hypercube sampling (LHS) design method [34]. Next, the solutions are evaluated in terms of expensive objective functions. These will be archived for training and building an initial surrogate model. The model and the SRS methods are then used to initialize a population for DE. Thereafter, the population is evolved using a new introduced offspring reproduction strategy in SGDE, followed by a parameter adaptation strategy. Next, the training set will be updated. In the next phase, CMAES is also applied to the best known solution by DE. A detailed description of the above components is presented as follows.

6.1. Initial sampling

Generally speaking, optimization algorithms are subjected to *curse of dimensionality* which makes them unsuitable for high dimensional problems. To this fact, DoE methods are used to maximize the amount of information across the design space. This enables us to build a global model of fitness landscapes with a minimum number of samples. Currently, orthogonal array design (OAD), uniform design (UD) and LHS are among the most widely applied fractional factorial DoE methods. The proposed SGDE adopts LHS according to which projections of the generated points onto each variable axis should be uniform. The main advantage of the LHS is that it does not need more samples for more dimensions. Assume that $\mathbf{x}_1 \in [0,1]$, the LHS divides the range of input variable $x_{1,j},\ j = 1, .., d$ into n equally probable intervals $\frac{1}{n}$. Accordingly, the LHS is defined by combing n values obtained for $\mathbf{x}_1$ with the n values of $\mathbf{x}_2,\ \mathbf{x}_3, \ldots, \mathbf{x}_n$. An example of LHS design with $n = 3$ and $d = 2$ might look like Fig. 5.

Researchers introduced different criteria for LHS to address poor space filling properties. In this work, we adopt a Point-distance criterion. The so-called mindist criterion $\varnothing_{Mm}$ should be maximized during G_{LHs} generations and is defined as in (21). A larger value of $\varnothing_{Mm}$ allows a better cover of the search space.

$$\varnothing_{Mm}\left(X_d^N\right) = \min_{\substack{i,j=1\ldots N, \\ i \ne j}} x^{(i)} - x^{(j)} \tag{21}$$

All the solutions obtained by LHS will be evaluated using the computationally expensive function. These solutions and their fitness values will be archived for training the surrogate model. This is necessary as the RBF model needs to be trained before it can be used for the approximation purpose.

DE Algorithm

Procedure Differential - Evolution
  Set the generation number $G \leftarrow 0$
  Initialize a population of NP individuals as presented in (13)
  Compute fitness values for all the individuals
  Rank the solutions and find the current best
  repeat
    $G \leftarrow G + 1$
    for $i = 1: NP$ **do**
      Randomly pick different individuals $r1, r2, r3, r4, r5 \in \{1,..., NP\}$
      for $j = 1: m$ **do**
        Pick a random index $j_{rand} \in \{1,\ldots, m\}$
        Pick a uniformly distributed number $r_j \in [0,1]$
        Generate $v_{G,i,j}$ using a mutation strategy
        if $(r_j \leq CR)$ or $(j = j_{rand})$
          $u_{G,i,j} \leftarrow v_{G,i,j}$
        else
          $u_{G,i,j} \leftarrow x_{G,i,j}$
        end
      end
    end
    for $i = 1: NP$ **do**
      Compute fitness value for trial solutions $\mathbf{u}_{G,i}$
      if solution $f(\mathbf{u}_{G,i})$ is better than $f(\mathbf{x}_{G,i})$
        $\mathbf{x}_{G+1,i} \leftarrow \mathbf{u}_{G,i}$
      else
        $\mathbf{x}_{G+1,i} \leftarrow \mathbf{x}_{G,i}$
      end
    end
    Rank the solutions and find the current best
  until (stopping criteria are not met)
  Post process results and visualization
end

Fig. 4. Pseudocode of the standard DE algorithm.

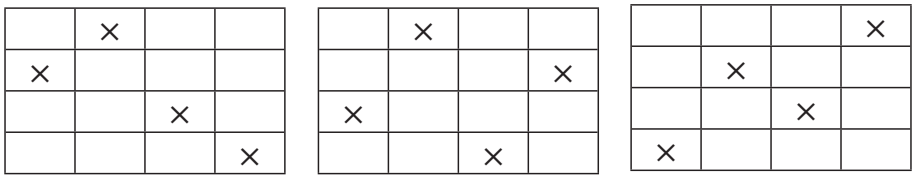


Fig. 5. Three examples of LHS of size N = 4 over [0, 1].

6.2. Population initialization

As a novel and effective component, SGDE adopts the SRS optimization method for population initialization. The main idea is to put forward a computationally cheap initialization method which is able to preserve diversity in more than one promising region. By adopting SRS, only a small part of the computational budget is used to find the promising regions and in most cases we employ the surrogate model. The SRS employs an optimal response for a surface (output variable) which is influenced by several explanatory variables (input variables). To do so, it uses a collection of mathematical and statistical information from surrogates and exact fitness function [35]. The SRS implementation consists of several steps; as presented in Fig. 6. Here, f is a continuous function defined on a compact hypercube $\mathscr{D} = [a, b] \subseteq R^d$, d is the number of decision variables, n_0 is the initial number of generated points and $\mathcal{A}_n$ contains the obtained solutions by the SRS. Also, a and b are the values of lower and upper bounds. The obtained solutions by the SRS are then considered as the initial population for the DE algorithm.

Considering the above explanations and Fig. 6, a few remarks are presented as follows. First, it should be noticed that the initial points in Step 2 come from the LHS. Second, the surrogate model in Step 8 can be RBF, Kriging, RBNN, SVR or any other type of function approximation models. Moreover, the meaning of the word "random" in Step 9 is very specifically related to the original study [35]. Also, "information" word in Step 14 refer to the estimated function value obtained from the surrogate model, and minimum distance from the previously evaluated points; as described in Ref. [35].

6.3. Offspring reproduction

It has been frequently testified that incorporating multiple search strategies can greatly improve the performance of DE [32]. Depending on the characteristics of the problem, they could perform better during different stages of evolution than a single strategy. Based on this finding,

SRS Method

1: **Procedure** Stochastic-Response-Surface
2: Consider a set of initial points $\mathcal{T} \leftarrow \{x_1, x_2, ..., x_{n_0}\} \subseteq \mathcal{D}$
3: $n \leftarrow n_0$
4: $\mathcal{A}_n \leftarrow \mathcal{T}$
5: Compute fitness value using the costly function for the generated points in τ
6: set $x_n^* \in \mathcal{A}_n$ to the point with the best function value
7: **repeat**
8: Fit surrogate model $\mathcal{S}_n$ using the data points $\mathcal{B}_n \leftarrow \{(x_i, f(x_i)) : i \leftarrow 1, ..., n\}$
9: Randomly generate a set of new solutions σ_n
10: Apply the bound constraints on each solution $\mathcal{C} \in \sigma_n$ (called as candidates)
11: Use $\mathcal{S}_n$ and $\mathcal{B}_n$ to evaluate candidate solutions $\mathcal{C}$
12: Set $x_{n+1} \in \mathcal{C}$ to the point with the best approximated function value
13: Use the costly function to evaluate at the x_{n+1}
14: Update information $\mathcal{A}_{n+1} \leftarrow \mathcal{A}_n \cup x_{n+1}$, $\mathcal{B}_{n+1} \leftarrow \mathcal{B}_n \cup \{(x_{n+1}, f(x_{n+1}))\}$
15: Set $x_{n+1}^* \in \mathcal{A}_n$ to the point with the best function value
16: $n \leftarrow n + 1$
17: **until** (stopping criteria are met)
18: Post process results and visualization
19: **end**

Fig. 6. Pseudocode of the SRS method.

SGDE generates competitive offspring for each individual in the population by means of different mutations. Maintaining diversity can help DE to improve the performance of its search operators. Hence, the SGDE is extended to the case of an arbitrary number of mutation strategies. This differs from previous studies that use a predefined set of search operators. The proposed diversification process begins with an initializing step in which the SGDE generates a random population of the chromosomes using the GEP for each individual. These chromosomes denote our mutation strategies and are generated using heuristic information of the DE. Considering the search operators of DE presented by (14)–(18), the chromosomes are based on a mathematic combination of $\mathbf{x}_{G,r1}$, $\mathbf{x}_{G,r2}$, $\mathbf{x}_{G,r3}$, $\mathbf{x}_{G,r4}$, $\mathbf{x}_{G,r5}$, $\mathbf{x}_{G,best}$, $\mathbf{x}_{G,i}$, and F. Indeed, the introduced diversification schema only changes the way that search operators process information. The new generated chromosomes are then decoded into tree based programs and we perform a breadth-first search method to obtain the mathematic formula. During this process, a specific part of each chromosome might be useless (the useful part is called open reading frame and the other part is referred to as non-coding region). An example of such schema is illustrated in Fig. 7. The mentioned mutation strategies are then employed to generate trial vectors for each individual. In this approach, efficiency of the generated configurations for each individual is computed based on the approximated objective value for the generated trial vectors. The GEP takes advantages of these cheap approximated values in order to adaptively evolve mutation strategies.

Subsequently, an effective model management procedure is required to build the surrogate models when the *curse of dimensionality* emerges. To alleviate this difficulty, we involve local surrogate models which are capable of generating more accurate fitness values. Furthermore, they are relatively fast and take into account the most important information of the closest neighbors. To this fact, SGDE builds separate local surrogate models for each trial vector. It builds the local surrogates on the basis of the idea that training set for each trial vector should not lie too far from it. The SGDE selects *k*-nearest neighbors from the archive in order to build the local surrogate models. The generated surrogate models are then used to evaluate the corresponding trial vectors.

In SGDE, the estimated objective value is not the only one considered, and several other criteria are used to find the best set of generated trial vectors for the individuals. The rules of the SGDE scheme are given as follows: between any two generated trial vectors for an individual, the solution with better approximated objective value which is evaluated

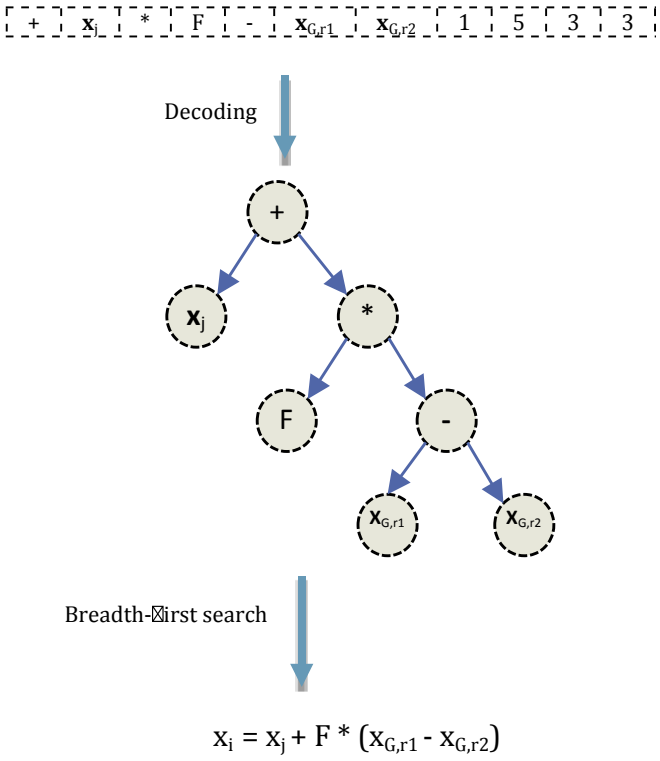


Fig. 7. A schematic of the proposed GEP based search operators generation.

with more accurate surrogate model, and which is also closer to its corresponding individual, is preferred. The accuracy of the models here is calculated using the cross-validation error, while the Euclidean distance is used to measure the distance. To measure the accuracy, the suggested training data set is randomly split into K equal subsets with n data points. Then, each subset is removed in turn from the training data set and the surrogate model is updated using the $K-1$ remaining data. Each time, all the solutions in the removed subset are predicted using the model. Consequently, for each subset s_j we have n predictions $(\hat{f}_1^{(j)}, \hat{f}_2^{(j)}, \ldots, \hat{f}_n^{(j)})$

of the n data points $(\mathbf{x}_1^{(j)}, \mathbf{x}_2^{(j)}, \ldots, \mathbf{x}_n^{(j)})$. Accordingly, the cross-validation error ε_{cv} is calculated as in (22)–(23) [36]. To sum up, the proposed component is depicted in Fig. 8. In this figure, PSR is a randomly initialized population of NSP search operators and i denotes the i^{th} individuals.

$$\varepsilon_{cv} = \frac{1}{k}\sum_{j=1}^{k}\varepsilon_j \tag{22}$$

$$\varepsilon_j = \frac{1}{n}\sum_{l=1}^{n}\left(f_l^{(j)}\left(\mathbf{x}_l^{(j)}\right) - \hat{f}_l^{(j)}\left(\mathbf{x}_l^{(j)}\right)\right)^2 \tag{23}$$

6.4. Parameter adaptation

Besides the search operators, success rate of DE depends on appropriately tuning its control parameters. Although using a trial-and-error scheme seems to be a good solution, the main challenge lies in high computational costs. We use the feedback from the surrogate model to overcome such inconvenience and to guide the parameter adaptation. The proposed approach adaptively optimizes the F and CR parameters by incorporating an embedded standard DE algorithm. First, we train a global RBF kernel based on all the collected data in the archive. Next, a population of solutions is directly optimized using this surrogate for n generations without requiring any expensive function evaluations. Particularly, the optimization process runs multiple times with different configuration. We consider different lower and upper bounds for each individual in the population. So, the DE algorithm here tries to explore the search space around inside only a small interval. More precisely, for each individual a confidence interval $[-\vartheta, +\vartheta]$ is adopted. The introduced strategy tries to pre-screen the most promising configurations using a full factorial experiment.

Such an experiment allows us to take into the account all possible combinations of the parameters. We constructed a design table for a three-level full factorial in two factors (i.e., F and CR), as presented in Table 2. The configuration which leads to the best results will survive. In Table 2, the minus sign $-\delta$ indicates that the factor is decreased by δ, the plus sign $+\delta$ indicates the factor increased δ and $=$ show that the factor does not change.

The adapted parameters have a *lifetime* which represents the number of generation that a configuration can survive. The *lifetime* property depends on the quality of the global surrogate model; the higher the accuracy, the longer the parameter will be used. Similar to the previous section, the accuracy here is computed by the cross-validation error. Whenever the *lifetime* of the parameters ends, they will be re-configured.

Table 2
The full factorial experiment design matrix for two factors and three levels.

F	CR
$=$	$=$
$=$	$+\delta$
$=$	$-\delta$
$+\boldsymbol{\delta}$	$=$
$+\boldsymbol{\delta}$	$+\delta$
$+\boldsymbol{\delta}$	$-\delta$
$-\boldsymbol{\delta}$	$=$
$-\boldsymbol{\delta}$	$+\delta$
$-\boldsymbol{\delta}$	$-\delta$

Offspring reproduction

Procedure Offspring - reproduction
  Randomly pick different individuals $r_1, r_2, r_3, r_4, r_5 \in [1,NP]$
  for $l = 1: NSP$ **do**
    Program$\leftarrow$ Decode the PSR_l into a tree based program
    $GEP_{search\_operator} \leftarrow$ perform the breath first search on generated Program PSR_l
    for $j = 1: D$ **do**
      Pick a random index $j_{rand} \in [1,D]$
      Pick a uniformly distributed number $r_j \in [0,1]$
      Produce mutant vector $V_{l,i,d}$ via $GEP_{search\_operator}$
      if $(r_j \le CR)$ or $(j = j_{rand})$
        $U_{l,i,j} \leftarrow V_{l,i,j}$
      else
        $U_{l,i,j} \leftarrow X_{l,i,j}$
      end
    end
    NB$\leftarrow$ Select k nearest neighbors from archive for $U_{l,i}$ using k-NN algorithm
    Build a local surrogate model $\hat{f}$ using the RBF
    $\hat{f}_l \leftarrow$ Evaluate the $U_{l,i}$ using $\hat{f}$
    $\varepsilon_l \leftarrow$ Compute the cross-validation error for $\hat{f}$ as shown in (22)-(23)
    $d_l \leftarrow 0$
    for $L = 1:k$
      $d_l \leftarrow d_l +$ Euclidean_distance$(NB_L, U_{l,i})$
    end
    $E_l = \hat{f}_l + \varepsilon_l * d_l$
  end
  T$\leftarrow$ Find the best trial vector based on the computed values E
  if solution $f(T)$ is better than $f(X_{G,i})$
    $X_{G+1,i} \leftarrow T$
  else
    $X_{G+1,i} \leftarrow X_{G,i}$
  end
  Apply the GEP on PSR_l
end

Fig. 8. Pseudocode of the offspring reproduction strategy in SGDE for the ith individual.

Accordingly, the *lifetime* for a given configuration is determined as the multiplication of the cross-validation error ε_{cv} and a constant $\gamma > 1$. The elaborated parameter adaptation strategy is illustrated in Fig. 9.

6.5. Updating the training set size

The SGDE use a training data set DB to archive the solutions evaluated by the exact fitness function. The DB is supposed to be employed for both the global and local surrogate models. For this reason, the training set must be properly update to ensure that the created models can cover a slightly larger region, but still be most relevant to the current solutions. Furthermore, using all the evaluated solutions will increase the computational time. To tackle this, we adopted a linear decreasing rule which reduces the archive size linearly according to the number of fitness evaluations (i.e., FEs). After each generation G, the archive size a_size_{G+1} is updated as follows:

$$a\_size_{G+1} = \left(\frac{min\_v - max\_v}{max\ number\ of\ FEs}\right) \times current\ FEs + max\_v \tag{24}$$

The min_v and max_v are the minimum and maximum values for the archive size, respectively. Whenever $a\_size_{G+1} < a\_size_G$, the ($a\_size_G - a\_size_{G+1}$) worst-ranking individuals will be deleted from the archive.

6.6. Applying CMAES

Considering SGDE, it should be remarked that the generated mutation strategies by GEP focus on multimodal rugged search landscapes which is actually a limitation of the reliability of the SGDE. So, we employed CMAES approach to make the SGDE also computationally viable for large scale smooth problems. To do so, it is applied to the best known solution provided by DE. If the solution found by CMAES is better than the solution obtained in the exploration phase by DE, then it replaces the old one; otherwise, it is discarded. This algorithmic choice generalizes the behavior of the algorithm; regarding properties of the fitness landscape.

7. Experimental results

In this section, performance of the SGDE on protein sequences from PDB using the elaborated models is investigated. The detailed information about these proteins is presented in Table 3. To provide a fair comparison procedure, we select amino-acid sequences which have been frequently used to benchmark new algorithms for the PSP. The free energy state of the presented sequences should be minimized. For the AB off-lattice model, the hydrophobic and hydrophilic characteristics of the amino-acids are denoted as follows: D, E, F, H, K, N, Q, R, S, T, W, Y fall into B residues and I, V, L, P, C, M, A, G belong to A residues. The SGDE is implemented in Matlab under Windows 7 operating system. To implement the RBF, we used the SURROGATES toolbox which is a general-purpose library for the surrogate models [37].

7.1. Experimental setup

First, the proposed algorithm is compared with the basic ABC, DE, PSO and CMAES optimization algorithms. In DE, we set F to 0.5, population size to 100 and CR to 0.9. In PSO, population size is 100, C1 and C2 coefficients are 1.8, and inertia weight is 0.6. The parameter configurations of ABC and CMAES are set according to Refs. [29,38]. To reduce stochastic behavior of the random population initialization, the results of the algorithms over 30 runs are considered. The mean, best and standard deviation of the results are reported in Table 5. Thereafter, a comparison

Parameter Adaptation

Procedure Parameter- Adaptation
  Build a global RBF surrogate model $\hat{f}$ using all the archived data
  for $i = 1: NP$ **do**
    for $j = 1: D$ **do**
      $lb_{i,j} = X_{i,j} - \vartheta$
      $ub_{i,j} = X_{i,j} + \vartheta$
    end
  end
  for each configuration C in Table 2
    Execute the standard DE with C and $\hat{f}$ as presented in Algorithm 1 for $\check{G} = 100$ generation
  end
  Select the configurations with the best obtained value $\hat{f}$
  $\varepsilon_{cv} \leftarrow$ Compute the cross-validation error for $\hat{f}$ as shown in (22)-(23)
  lifetime $\leftarrow \varepsilon_{cv} \times \gamma$
End

Fig. 9. Pseudocode of the parameter adaptation strategy in SGDE.

Table 3
The employed protein sequences along their properties.

PDB ID	Dim	Sequences
1CB3	21	BABBBAABBAAAB
1BXL	27	ABAABBAAAAABBABB
1EDP	29	ABABBAABBBAABBABA
2H3S	45	AABBAABBBBBABBBABAABBBBBB
2KGU	63	ABAABBAABABBABAABAABABABABABAAABBB
1TZ4	69	BABBABBAABBAAABBAABBAABABBBABAABBBBBB
1TZ5	69	AAABAABAABBABABBAABBBBAABBBABAABBABBB
1AGT	71	AAAABABABABABAABAABBAAABBABAABBBABABAB
1CRN	87	BBAAABAAABBBBBAABAAABABAAAABBBAAAAAAAABAAABBAB
1HVV	145	BAABBABBBBBBAABABBBABBABBABABAAAAABBBABAABBABBBABBAABBABBAABBBBBAABBBBBABBB
1GK4	163	ABABAABABBBBABBBABBABBBBAABAABBBBBAABABBBABBABBBAABBABBBBBAABABAAABABAABBBBAABABBBBA
2EWH	191	AABABAAAAAAABBBAAAAAABAABAABBAABABAAABBBAAAABABAAABABBAAABAAA BAAABAABBAABAAAAABAAABABBBABBAAABAABA

study between the SGDE and state-of-the-art algorithms from the literature is conducted. In this comparison, the following algorithms are included: DE_{pfo}, jDE (DE/best/1), L-SHADE, SaDE, CoDE, JADE and EPSDE. The results for the DE_{pfo}, jDE (DE/best/1) and L-SHADE are directly taken from Ref. [11]. The parameters of the other competitive algorithms are set according to their original works. Furthermore, we adopt the same implementation and parameter configuration for the SRS and RBF as suggested in Ref. [15]. The rest of the parameters are tuned and presented in Table 4.

7.2. Analysis of the incorporated GEP

In this subsection, we analyzed performance of the SGDE in order to give an insightful view for the influence of the introduced GEP schema on the algorithm's efficiency. To do so, we conducted 30 runs based on 1CB3 and 1BXL amino-acid sequences using the AB off-lattice model. The stopping condition is determined as 50,000. The configurations of SGDE are equal to those in Section 7.1. The obtained results are shown in Fig. 10. In this figure, the SGDE* is a version of SGDE which only adopts the mutation strategies of the DE presented by (14)–(18), while SGDE uses a diversified pool of mutation strategies for each individual. The depicted results show that SGDE performs better and obtained more stable results. In contrast to SGDE*, the introduced algorithm evolves a pool of effective trial vector mutation strategies for individuals based on their previous experience. Accordingly, unfavorable mutation strategies of less effective results will be discarded in later generations.

7.3. Computational results

The number of function evaluations is set to 200,000 for the standard algorithms, improved versions and for the SGDE. In SGDE, 0.01% of the computational budget is used by SRS, 0.5% by the DE and 0.49% by CMAES. The experiments are performed through 30 independent runs. The obtained results are reported in Tables 5 and 6. Also, Table 7 presents the best obtained results by SGDE for the problem at hand.

In Table 6, DE^{*}_{pfo} and $SGDE^{\dagger}$ denote the results with 100,000 function evaluations for DE_{pfo} and SGDE, respectively. As previously mentioned, major contribution of the proposed SGDE approach is using the GEP and surrogate modeling to replace in part the original computationally

Table 4
The tuned parameters for the proposed SGDE algorithm.

Parameter	Value	Parameter	Value
Population size	100	Head length (GEP)	50
$\mathbf{G_{LHs}}$	1000	Selection rate (GEP)	0.8
$\mathbf{n_0}$	2*(D+1)	Mutation rate (GEP)	0.05
$\boldsymbol{k}$	15	Function Set (GEP)	{*,+,-}
K	4	γ	1.5
NSP	10	min_v	100
ϑ	0.1	max_v	200

Table 5
Comparison of the results between the standard algorithms and the SGDE (Std.: standard deviation).

Protein sequence	Algorithm	Mean	Std.	Best	PI
1CB3	ABC	−4.296455	0.439499	−5.268011	29.30%
	DE	−1.866129	0.845961	−4.659330	69.29%
	PSO	−3.203742	2.084507	−5.776234	47.28%
	CMAES	−2.503826	2.056343	−7.778162	58.80%
	SGDE	−6.077217	1.754122	−8.369052	
1BXL	ABC	−9.446039	0.655434	−11.268678	35.69%
	DE	−11.294667	1.028064	−12.648686	23.11%
	PSO	−9.710913	2.753267	−11.933259	33.89%
	CMAES	−7.734668	3.440012	−15.362006	47.35%
	SGDE	−14.689404	1.839411	−16.478776	
1EDP	ABC	−6.121525	0.880439	−8.253825	32.59%
	DE	−7.233354	2.695314	−11.316626	20.34%
	PSO	−4.307987	1.113684	−5.711125	52.56%
	CMAES	−4.609754	2.726812	−9.337707	49.24%
	SGDE	−9.080840	1.857780	−13.145400	
2H3S	ABC	−7.341647	0.893442	−10.199716	41.91%
	DE	−7.443750	1.227711	−9.292054	41.10%
	PSO	−4.618970	1.955581	−6.897306	63.45%
	CMAES	−7.368380	4.120026	−14.856206	41.70%
	SGDE	−12.637971	2.861910	−17.303680	
2KGU	ABC	−19.223037	1.562318	−22.130853	50.38%
	DE	−18.511796	6.859623	−25.403276	52.21%
	PSO	−12.471775	5.963385	−21.131556	67.81%
	CMAES	−19.605122	8.453813	−29.880254	49.39%
	SGDE	−38.738349	4.606122	−46.091724	
1TZ4	ABC	−12.582645	1.410564	−15.867712	47.84%
	DE	−8.448939	2.895693	−11.970046	64.97%
	PSO	−5.843409	3.655283	−9.194886	75.78%
	CMAES	−11.486689	4.331165	−18.983567	52.38%
	SGDE	−24.122570	4.058022	−31.503100	
1TZ5	ABC	−15.313003	1.316238	−18.879996	48.56%
	DE	−12.045163	6.590958	−18.610951	59.53%
	PSO	−8.755463	4.135149	−14.293049	70.59%
	CMAES	−15.702593	5.638691	−29.306050	47.25%
	SGDE	−29.766762	4.581026	−39.053634	
1AGT	ABC	−22.568844	1.942112	−25.652326	44.46%
	DE	−13.522272	5.734693	−21.687050	66.72%
	PSO	−18.835705	7.774478	−26.380818	53.65%
	CMAES	−23.411478	8.961818	−34.457463	42.39%
	SGDE	−40.634420	4.219346	−46.229500	
1CRN	ABC	−38.261975	2.804349	−41.290523	40.46%
	DE	−18.963015	9.991839	−35.898240	70.49%
	PSO	−23.902511	3.355806	−27.545842	62.80%
	CMAES	−40.975829	13.444372	−60.939240	36.23%
	SGDE	−64.258945	7.687089	−78.245070	
1HVV	ABC	−21.609320	2.569676	−27.353049	43.76%
	DE	−3.241457	13.398311	−11.795201	91.56%
	PSO	4.497784	6.747108	−1.060249	111.71%
	CMAES	−18.988722	3.968934	−27.915393	50.58%
	SGDE	−38.422210	5.975514	−52.558824	
1GK4	ABC	−27.836994	2.267989	−32.729077	40.75%
	DE	11.778927	8.120520	−1.4981720	125.61%
	PSO	−6.571952	9.798663	−20.691791	86.01%
	CMAES	−25.297559	3.207333	−41.108156	46.16%
	SGDE	−46.984386	3.969936	−57.965434	

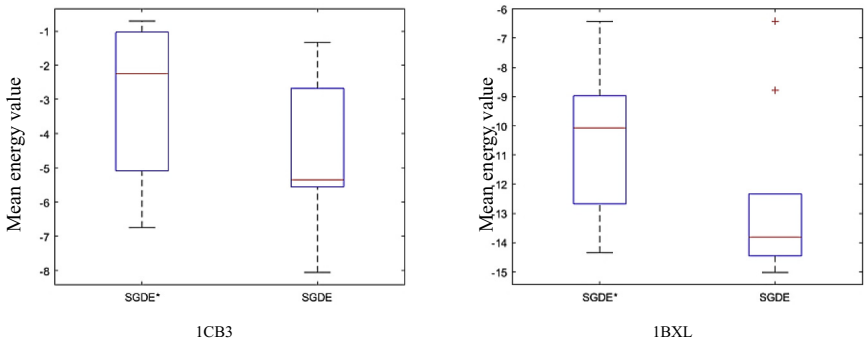


Fig. 10. The obtained results over 30 runs for protein sequences 1CB3 and 1BXL using SGDE and SGDE*.

Table 6
Comparison of the results between the improved algorithms and the SGDE (Std.: standard deviation).

Protein sequence	Algorithm	Mean	Std.	Best	PI
1CB3	DE_{pfo}	−5.588400	1.960300	−8.369000	8.04%
	DE^{*}_{pfo}	−4.539159	2.314210	−8.116150	25.31%
	jDE (DE/best/1)	−3.898800	2.443700	−8.198300	35.85%
	L-SHADE	−2.791600	2.106800	−8.115100	54.06%
	SaDE	−3.487625	2.306001	−5.933400	42.61%
	CoDE	−5.585840	0.677920	−6.774100	8.09%
	JADE	−5.391650	0.423160	−6.394700	11.28%
	EPSDE	−4.918070	0.383574	−5.822700	19.07%
	SGDE	−6.077217	1.754122	**−8.369052**	
	SGDE†	−5.765060	1.081309	−8.116200	5.14%
1BXL	DE_{pfo}	−12.610400	2.530600	−16.344300	14.15%
	DE^{*}_{pfo}	−11.862654	2.510254	−15.833700	19.24%
	jDE (DE/best/1)	−12.404700	2.491300	−16.010100	15.55%
	L-SHADE	−10.542800	2.871200	−14.201500	28.23%
	SaDE	−11.208780	0.809479	−12.345200	23.69%
	CoDE	−11.907830	2.605866	−15.447600	18.94%
	JADE	−11.384120	0.765585	−12.732600	22.50%
	EPSDE	−10.530370	0.877880	−12.178400	28.31%
	SGDE	−14.689404	1.839411	**−16.478776**	
	SGDE†	−14.436030	1.174347	−16.223800	1.72%
1EDP	DE_{pfo}	−8.666600	2.560300	−13.562000	13.03%
	DE^{*}_{pfo}	−8.495471	3.339964	−13.428000	14.75%
	jDE (DE/best/1)	−7.466700	2.937600	−11.988000	25.07%
	L-SHADE	−4.590000	3.217800	−11.697700	53.94%
	SaDE	−5.392990	2.666704	−9.946400	45.88%
	CoDE	−8.087580	3.861207	−13.527000	18.84%
	JADE	−7.120270	0.892819	−8.673800	28.55%
	EPSDE	−6.200950	0.585151	−7.307400	37.77%
	SGDE	−9.964899	2.623943	**−14.292856**	
	SGDE†	−9.080840	1.857780	−13.145400	8.87%
2H3S	DE_{pfo}	−10.676700	2.751800	−16.503000	15.52%
	DE^{*}_{pfo}	−12.111503	2.982058	−17.287400	4.17%
	jDE (DE/best/1)	−10.793100	2.786400	−16.692000	14.60%
	L-SHADE	−10.383000	2.627300	−15.668700	17.84%
	SaDE	−6.265260	1.385776	−8.454000	50.43%
	CoDE	−9.679260	2.438849	−14.432800	23.41%
	JADE	−8.240400	1.049730	−9.997900	34.80%
	EPSDE	−5.108480	0.924547	−6.755500	59.58%
	SGDE	−12.637971	2.861910	**−17.303680**	
	SGDE†	−12.539310	2.823182	−17.046000	0.78%
2KGU	DE_{pfo}	−35.385000	4.701300	−44.336900	8.66%
	DE^{*}_{pfo}	−28.521353	4.771998	−37.74590	26.37%
	jDE (DE/best/1)	−29.551100	5.374000	−40.503500	23.72%
	L-SHADE	−26.628200	2.907100	−35.070700	31.26%
	SaDE	−13.514080	1.496921	−16.036900	65.11%
	CoDE	−24.925000	4.086078	−30.124600	35.66%
	JADE	17.218310	0.993447	−19.218100	144.45%
	EPSDE	−8.360970	0.666336	−9.341700	78.42%
	SGDE	−38.738349	4.606122	**−46.091724**	
	SGDE†	−38.201660	6.536553	−44.284300	1.39%
1TZ4	DE_{pfo}	−20.436100	5.279800	−30.921100	15.35%
	DE^{*}_{pfo}	−19.293730	4.088801	−25.517600	20.09%
	jDE (DE/best/1)	−16.913500	3.885100	−24.300000	29.94%
	L-SHADE	−16.469300	2.896300	−20.221600	31.78%
	SaDE	−8.634780	1.771702	−11.188000	64.23%
	CoDE	−11.740680	4.679727	−18.023000	51.37%
	JADE	−10.376180	1.692243	−13.829800	57.02%
	EPSDE	−1.814920	0.893272	−3.3096000	92.48%
	SGDE	−24.142960	6.107640	**−31.503100**	
	SGDE†	−24.122570	4.058022	−31.503100	0.08%
1TZ5	DE_{pfo}	−27.341200	4.084700	−38.186800	8.15%
	DE^{*}_{pfo}	−23.652170	3.674546	−31.400700	20.54%
	jDE (DE/best/1)	−20.365500	3.837800	−30.127900	31.58%
	L-SHADE	20.640300	3.116300	−34.311500	169.34%
	SaDE	−14.617780	2.796681	−19.886800	50.89%
	CoDE	−19.165230	3.451774	−24.744500	35.62%
	JADE	−13.169800	1.505524	−15.702900	55.76%
	EPSDE	−4.124440	0.638126	−5.020000	86.14%
	SGDE	−29.766762	4.581026	**−39.053634**	
	SGDE†	−29.486118	4.296542	−38.626900	0.94%
1AGT	DE_{pfo}	−39.026800	5.344600	−50.631100	5.78%
	DE^{*}_{pfo}	−36.684787	6.323074	−48.854200	11.44%
	jDE (DE/best/1)	−30.777000	6.309000	−42.992600	25.70%
	L-SHADE	−29.356400	2.684600	−39.316800	29.13%
	SaDE	−14.187640	1.487899	−15.680800	65.75%
	CoDE	−28.921060	4.867545	−35.709000	30.18%
	JADE	−19.432300	0.622762	−20.246100	53.09%
	EPSDE	−8.914320	1.135966	−10.824500	78.48%
	SGDE	−41.422991	6.285417	**−54.362306**	
	SGDE†	−40.634420	4.219346	−46.229500	1.90%
1CRN	DE_{pfo}	−60.244400	7.577200	−74.406800	6.25%
	DE^{*}_{pfo}	−53.743007	6.253436	−68.836100	16.36%
	jDE (DE/best/1)	−46.903000	7.424300	−63.713800	27.01%
	L-SHADE	−46.960400	3.768300	−60.237100	26.92%
	SaDE	−22.044860	1.553122	−24.751700	65.69%
	CoDE	−44.234280	4.972664	−49.020300	31.16%
	JADE	−30.053940	1.572330	−32.762100	53.23%
	EPSDE	−9.730750	1.139082	−12.381900	84.86%
	SGDE	−64.258945	7.687089	**−78.245070**	
	SGDE†	−60.634070	6.462495	−70.030400	5.64%
1HVV	DE_{pfo}	−34.805900	5.292600	−44.726400	9.41%
	DE^{*}_{pfo}	−32.351140	4.849457	−41.921100	15.80%
	jDE (DE/best/1)	−20.954100	7.642400	−31.587800	45.46%
	L-SHADE	−25.491000	1.709000	−28.778700	33.66%
	SaDE	−13.775810	6.625543	−21.356300	64.15%
	CoDE	−21.148700	4.558295	−26.106000	44.96%
	JADE	−8.771650	1.124914	−10.729700	77.17%
	EPSDE	18.788040	1.614203	15.689900	148.90%
	SGDE	−38.422210	5.975514	**−52.558824**	
	SGDE†	−38.201660	6.536553	−44.284300	0.57%
1GK4	DE_{pfo}	−44.859100	4.722700	−52.065100	4.52%
	DE^{*}_{pfo}	−40.455554	5.350739	−47.983600	13.90%
	jDE (DE/best/1)	−22.321800	7.416900	−35.677900	52.49%
	L-SHADE	−32.908200	2.210800	−40.265500	29.96%
	SaDE	−19.701000	5.829114	−25.855100	58.07%
	CoDE	−27.696075	6.414348	−35.723300	41.05%
	JADE	−9.812420	1.007794	−11.348200	79.12%
	EPSDE	12.030714	9.152999	1.614203	125.61%
	SGDE	−46.984386	3.969936	**−57.965434**	
	SGDE†	−46.859100	5.142473	−52.024740	0.27%

expensive solver. To this fact, results after 100,000 evaluations are also reported in Table 6 to investigate whether SGDE algorithm is able to take advantages of the introduced surrogate-based schema. We considered DE_{pfo} and SGDE because they yield best performances in terms of solution accuracy and convergence rate.

The values of best, worst, mean, median and standard deviation of the energy functions are presented for the purpose of the comparison. Furthermore, the percentage of improvement (PI) for the SGDE in comparison to each of the other algorithms based on the mean value is also collected. The corresponding results are given in Tables 5 and 6. In Table 5, we considered ABC and PSO as two algorithms with different properties from that of SGDE, and CMAES and DE as the basic components of the proposed algorithm. Regardless of SGDE, it can be seen that ABC provides better mean values for 1CB3, 1TZ4, 1HVV, 1GK4; CMAES for 2KGU, 1TZ5, 1AGT, 1CRN; and finally DE for 1BXL, 1EDP, 2H3S. These results indicate how algorithm scalability affects their performances. For example, it is clear that the performance of DE decreased by increasing the dimension of the problem at hand. Conversely, CMAES shows a better performance in the case of high dimensional problems. From Table 5, however, we can see that the enhanced algorithm

Table 7
The best obtained solution vectors by SGDE.

Sequences	Solution vector in degrees
1CB3	−13.041, 20.817, −39.505, −14.249, 28.612, 3.133, −6.776, 6.401, 33.293, −30.798, −10.004, −30.594, 204.909, 165.391, 183.932, 188.005, 101.259, 14.773, 2.565, 221.216, 190.832
1BXL	−23.058, −101.926, −16.652, 46.170, 8.710, −0.418, 75.287, 24.366, 43.752, 1.303, −2.988, −2.432, −38.211, −0.777, 51.683, 98.770, 5.898, −76.227, −63.225, 162.538, −148.684, −33.216, 278.209, −167.932, −255.219, −154.179, −94.389
1EDP	−22.739, 1.910, −104.101, 21.195, −125.910, 26.263, −3.295, 23.916, 25.536, 21.164, 9.161, −44.040, 31.740, −46.338, −4.520, 15.290, 49.322, −157.821, −25.050, −189.757, −151.354, −49.747, 1.413, 20.335, −15.761, 55.586, −187.339, −53.988, −176.481
2H3S	59.655, −26.849, −113.330, 3.700, −56.051, −49.854, 71.446, −27.260, 1.044, −123.514, −5.514, 90.039, −15.741, −26.559, 50.759, −23.017, 45.851, 48.831, 9.401, 50.956, −23.219, 3.534, 35.316, −0.199, −5.270, −124.885, −65.149, −30.528, 192.078, −198.127, 63.566, 127.694, −39.422, 200.616, 71.077, 133.549, 45.635, 196.027, −159.076, 222.841, −108.331, −202.128, 369.663, 269.937, 13.612
2KGU	−159.139, −71.147, −26.228, 5.416, −27.889, 37.277, 85.860, 29.850, 66.600, −220.387, 73.223, 224.554, −81.451, 55.263, −3.473, 43.010, −46.396, 45.797, −288.564, −20.181, −32.074, 14.577, 6.068, −151.986, −0.326, −79.239, −11.343, −258.643, 35.753, −144.949, −5.784, 12.601, −53.979, −56.730, 212.901, 131.871, 96.724, 160.449, 17.666, −231.835, 20.939, −41.439, 125.357, −131.312, −63.064, −55.975, −180.921, −182.716, −62.213, 182.275, −146.784, −256.896, 148.335, 35.609, 51.117, −342.816, 168.226, −52.008, 197.307, −124.738, −10.302, −164.867, −63.083
1TZ4	−165.334, 0.199, −308.832, −19.564, 57.088, −22.242, −33.621, 13.795, 54.533, −181.066, 8.051, −74.348, −43.488, 2.648, −42.501, −236.707, −11.640, −288.063, −13.457, −153.326, 2.644, −58.256, −155.085, 8.710, −24.551, 19.462, −59.251, 81.531, −3.096, −17.812, −255.897, 8.354, −69.865, −2.716, 48.152, 89.646, −2.270, −160.095, −39.007, 6.903, −28.411, −28.702, 50.580, 28.463, −226.413, −23.231, 17.755, −248.482, −170.342, 9.488, 197.418, 162.935, 28.759, −204.977, 5.143, −127.868, −190.456, 77.663, −177.556, 203.181, −242.256, −179.994, −126.389, −166.967, −40.466, 211.163, 7.530, −92.841, 5.068
1TZ5	144.424, 54.747, 102.739, 92.689, −2.814, 94.396, 2.146, 21.189, 69.594, 13.985, −66.838, 50.379, 48.012, 2.767, −126.631, 18.757, −11.099, 50.994, 31.931, −27.937, 3.396, 35.722, −21.505, −63.870, 47.882, −33.406, 16.053, 99.876, 279.718, −19.088, −294.202, −27.874, 62.764, −24.135, −23.670, 126.114, −127.681, −146.080, −52.368, 24.685, 4.347, 306.027, 23.096, 120.475, 203.812, 185.362, 283.888, 187.446, 66.357, −225.057, 126.390, −242.125, −146.168, −16.040, 302.675, −56.468, −65.507, −23.753, −223.905, −216.439, 243.711, 21.638, −228.963, 141.464, −176.279, −22.492, −232.266, 100.986, 202.063
1AGT	−156.030, −82.798, −95.070, −0.464, −112.604, −13.003, 36.340, −124.184, −39.982, 229.694, −3.162, −267.961, −33.234, 2.010, −93.592, 81.454, 331.001, 32.176, −9.160, 35.237, −1.374, −25.395, −128.790, −10.705, −56.745, −48.426, −12.472, 0.918, 66.112, −15.636, 12.279, 23.290, −76.885, −32.544, 184.096, −132.908, −47.248, −127.123, 33.011, 211.996, 59.836, 39.474, 26.187, 112.558, −21.629, 153.042, 223.658, 4.476, 33.030, −125.145, −74.598, −45.447, −149.345, 139.283, 128.786, 5.417, −24.633, −46.156, −207.543, 33.837, 146.291, 96.303, 49.327, 32.750, −172.228, −126.106, 137.126, −43.327, −147.187, 206.192, −35.189
1CRN	145.300, 36.488, −27.693, 165.534, 77.902, 354.837, 7.136, 48.594, −80.512, −14.057, 99.159, 112.074, 46.753, 256.470, 90.789, 22.948, 85.355, 83.407, −43.953, −96.154, 11.458, 128.099, 43.359, −72.034, 237.431, −33.780, −47.403, −90.489, 19.382, 18.790, 337.112, 177.606, −106.985, −35.294, 171.970, 50.324, −52.184, 192.523, −188.396, 63.694, 29.464, −81.714, −31.965, 342.749, −181.666, −136.103, −133.733, 28.622, 112.944, 155.587, 133.711, 34.217, −194.437, −6.297, −32.510, −59.951, −55.728, −180.195, −8.319, 23.287, 15.044, −165.647, −27.660, −94.603, 158.104, 21.053, 2.937, −46.729, −59.605, −119.248, 151.592, 76.387, 36.969, 64.091, 23.301, 195.288, 50.213, 19.561, 180.144, 55.116, 224.443, 202.698, 163.036, 128.596, 71.251, 137.036, 239.625
1HVV	82.557, −4.648, 49.800, −23.404, −28.628, 41.943, 7.134, −166.340, −56.528, −337.274, −31.374, −93.214, 37.106, −117.152, 42.778, 100.133, −14.835, 76.012, −321.270, −72.339, 42.745, −37.322, 45.282, 37.796, −11.162, 54.482, −117.711, −5.714, −143.808, −1.347, 79.931, −10.487, −11.978, −17.393, 44.289, 29.164, 15.536, 23.316, −24.147, 163.034, −1.999, −6.900, 40.658, 42.585, 1.649, −77.651, 26.304, −60.850, −43.554, 9.506, 358.792, 27.782, −17.848, 13.687, 20.478, 6.313, −267.353, 2.239, 7.522, −78.233, −32.929, −7.866, 16.250, 30.633, 2.260, −5.060, −75.798, 11.519, 2.689, 11.681, 13.585, 1.897, 27.862, −113.754, −16.753, 43.384, −66.091, −161.383, −96.891, −187.561, 46.594, 114.072, 121.035, 138.510, 104.586, −37.781, 119.307, 40.934, −132.005, −52.130, −159.752, −124.843, 15.955, 260.259, −5.885, 81.430, 6.415, 161.376, 124.015, 193.957, 52.734, 235.418, 21.524, 278.142, −110.396, −79.301, −184.381, −135.690, −213.179, −100.948, −26.899, −129.316, −81.018, −81.180, −187.666, 108.727, 132.176, −201.880, −61.588, −176.929, 93.506, 21.892, −53.959, −106.365, −72.412, −181.517, −165.334, 197.385, −128.767, −13.453, −323.226, −163.631, −59.022, 37.137, 43.509, −299.102, 90.037, 89.780, −14.997, 53.471, −203.178, 143.063, 40.360, −24.016, −16.890
1GK4	−21.350, 7.289, −98.280, −44.417, −29.790, 32.793, −45.553, −6.752, 56.575, −8.413, 10.111, −8.803, 14.148, 8.485, 3.998, −45.350, −24.909, −12.958, 1.046, −10.167, 33.426, 9.071, 13.890, −40.705, 69.725, 17.744, −20.340, 18.051, 16.650, −90.021, 7.510, −13.634, 32.259, −0.115, 19.477, −25.760, 3.567, −7.441, 4.887, −5.443, 2.298, −7.960, 8.788, −9.155, 9.726, 34.049, −0.973, −17.697, −2.901, −0.428, 22.487, −52.917, 58.330, −19.208, −1.853, −6.544, −17.736, −11.070, 5.067, 39.871, 89.058, 16.728, 80.819, 5.917, −15.588, −26.382, −34.298, 1.734, −73.843, 12.014, −0.598, −44.611, −13.265, −1.520, 0.664, 4.390, −9.704, 10.029, 3.965, −0.721, 0.382, −3.113, 85.934, −19.991, −78.439, −7.090, 128.539, −9.184, −62.805, −12.132, 91.381, 35.076, −68.442, −24.881, 72.582, 121.197, 6.538, 21.101, −43.487, −148.246, −148.722, −40.254, 24.561, 47.359, 43.872, 31.372, −78.728, −108.732, −124.579, −154.733, 12.213, −26.988, 76.124, 58.749, 43.400, −17.033, −145.120, −38.315, 22.566, −78.594, −16.745, 85.985, −0.592, −85.335, 16.567, 76.919, −22.160, −24.485, 27.512, 63.799, 83.927, 63.474, −25.497, −187.504, −126.813, −21.404, −123.582, −121.014, −180.669, −167.495, −46.515, −159.078, 17.356, 16.533, −104.856, −41.776, 70.995, −11.817, 105.383, 162.064, 58.494, 79.789, −21.554, 76.475, 35.576, 44.884, −15.714, −128.459, −23.287, 35.668, 85.378, −16.573, −11.516

Table 7 (*continued*)

outperforms all the standard algorithms on both the low and high dimensional problems. The reported PI values also confirm the superiority of the SGDE. It can be explained by the adopted surrogate models and GEP technique which reduce the probability of the exploration around already explored search areas.

Thereafter, we collect the results of the recently proposed algorithms for the PSP in order to provide a more comprehensive study. The best, mean and standard deviation of the algorithms are presented for the purpose of the comparison. Table 6 shows the performance of the algorithms for all the sequences over 30 runs. This table denotes the superior accuracy of the SGDE for all the presented test sequences in terms of mean, standard deviation, and the best results; as reported in Table 7. The 3D representations for some sequences which have a near optimum structure are illustrated in Fig. 11.

Interestingly, we can also see that SGDE† outperforms other propositions for all the problems when considering the mean values. It is also evident that the introduced cheap algorithms fairly provide comparable results when considering the best obtained solutions. Surprisingly, SGDE† outperforms the other algorithms on the 2H3S, 1TZ4 and 1TZ5 protein sequences by considering the best obtained value. This is due to the fact that the incorporated local and global surrogate models help the SGDE to walk around the potentially promising regions and avoid unnecessary computation cost in non-promising regions. Furthermore, the GEP algorithm enables the SGDE to learn the utility of adopting the best mutation strategies which leads to a proper balance between the exploration and exploitation.

In Table 6, PI of the different algorithms is also presented. From this table, we can see that the introduced method needs lower number of fitness evaluations to attain a similar or better result. More precisely, if we compute the average improvement of the algorithms over 11

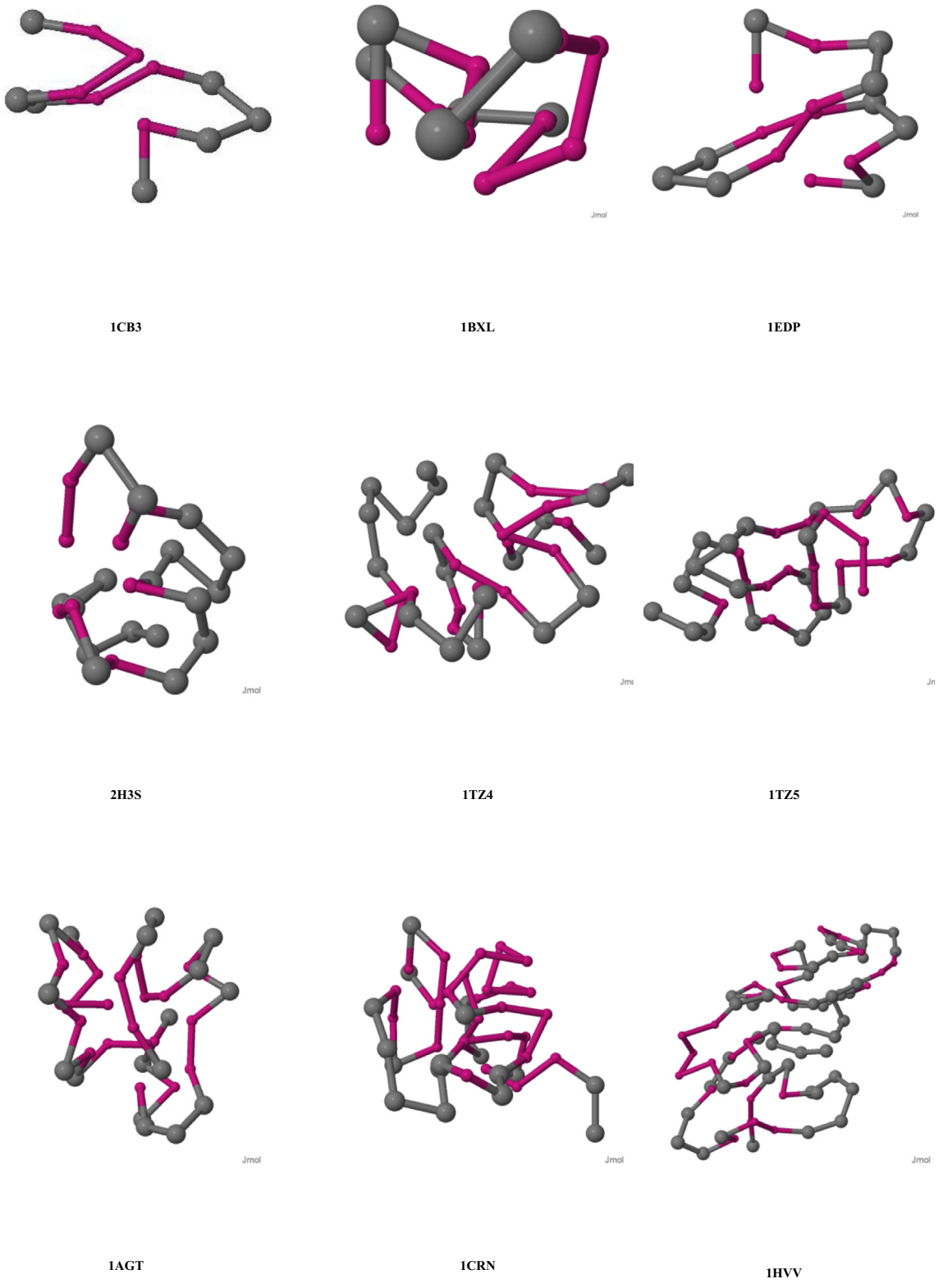


Fig. 11. The folded structure by SGDE.

problems as depicted in Fig. 12, we can see that the overall percentage improvement for the SGDE is always positive.

To evaluate stability between the algorithms, a rank based analysis is presented. This procedure determines the most effective algorithm for each protein sequence by ranking them from the best to the worst. The rank procedure assigns rank 1 to the algorithm with the best fitness; rank 2 to the second best, and rank N to the Nth best. Given this procedure, the average ranks based on the mean fitness values are calculated and summarized in Fig. 13. According to this figure, the algorithms can be sorted in the following order: SGDE, SGDE†, DEpfo, DEpfo*, jDE, L-SHADE, JADE, SaDE, CoDE and EPSDE. From this figure, we can say that both the SGDE and SGDE† are less sensitive to the dimensionality of the problem at hand and thus, have proved to be more scalable compared to the competitive algorithms.

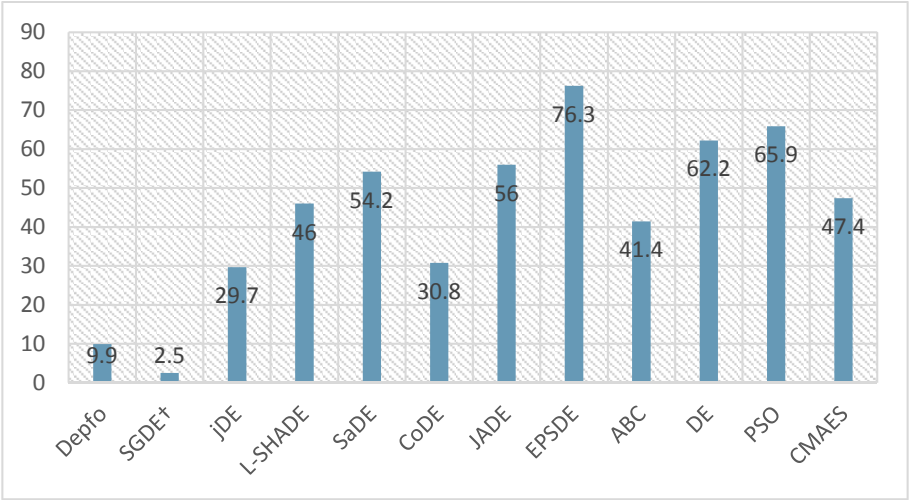
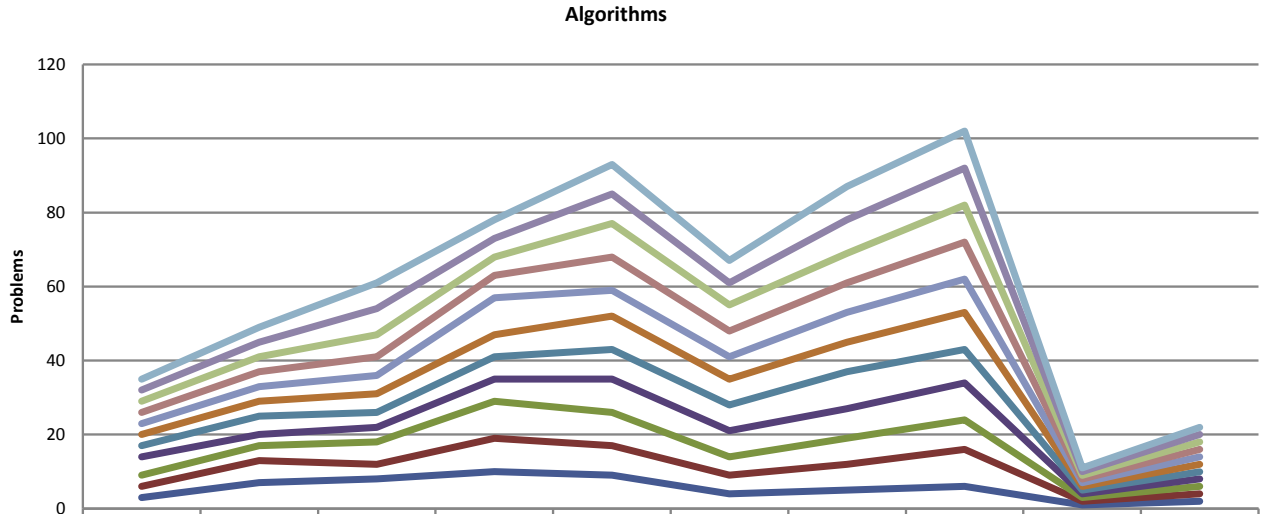


Fig. 12. The percentage of improvement in terms of the mean values for the enhanced SGDE algorithm.



	DEpfo	DE_pfo*	jDE (DE/best/1)	L-SHADE	SaDE	CoDE	JADE	EPSDE	SGDE	SGDE†
1GK4	3	4	7	5	8	6	9	10	1	2
1HVV	3	4	7	5	8	6	9	10	1	2
1CRN	3	4	6	5	9	7	8	10	1	2
1AGT	3	4	5	6	9	7	8	10	1	2
1ZT5	3	4	5	10	7	6	8	9	1	2
1ZT4	3	4	5	6	9	7	8	10	1	2
2KGU	3	5	4	6	8	7	10	9	1	2
2H3S	5	3	4	6	9	7	8	10	1	2
1EDP	3	4	6	10	9	5	7	8	1	2
1BXL	3	6	4	9	8	5	7	10	1	2
1CB3	3	7	8	10	9	4	5	6	1	2

Fig. 13. Ranking results of algorithms based on mean values.

Besides the above mentioned performance measures in Tables 5 and 6, a nonparametric test, called Friedman, followed by a Conover method, is also conducted to determine whether the obtained results by SGDE are significantly different from the other approaches. To do so, the mean values obtained from 12 algorithms over the 11 protein sequences are subjected to this test with 0.05 as the level of significance. The results indicate that with 95% certainty SGDE provided significantly better results for these problems.

The convergence curves of the improved algorithm over 30 runs per sequence are presented in Fig. 14. The x-axis presents the function evaluations consumed, and the y-axis is the logarithmic value of the mean energy value. Here, a bias value of 100 is added to make the energy values positive. From this figure we can see the proposed SGDE algorithm shows continuously fast convergence rate for the presented problems. The advantage of combining machine learning techniques and DE is revealed in this convergence plot. Take 1EDP as an example, all the

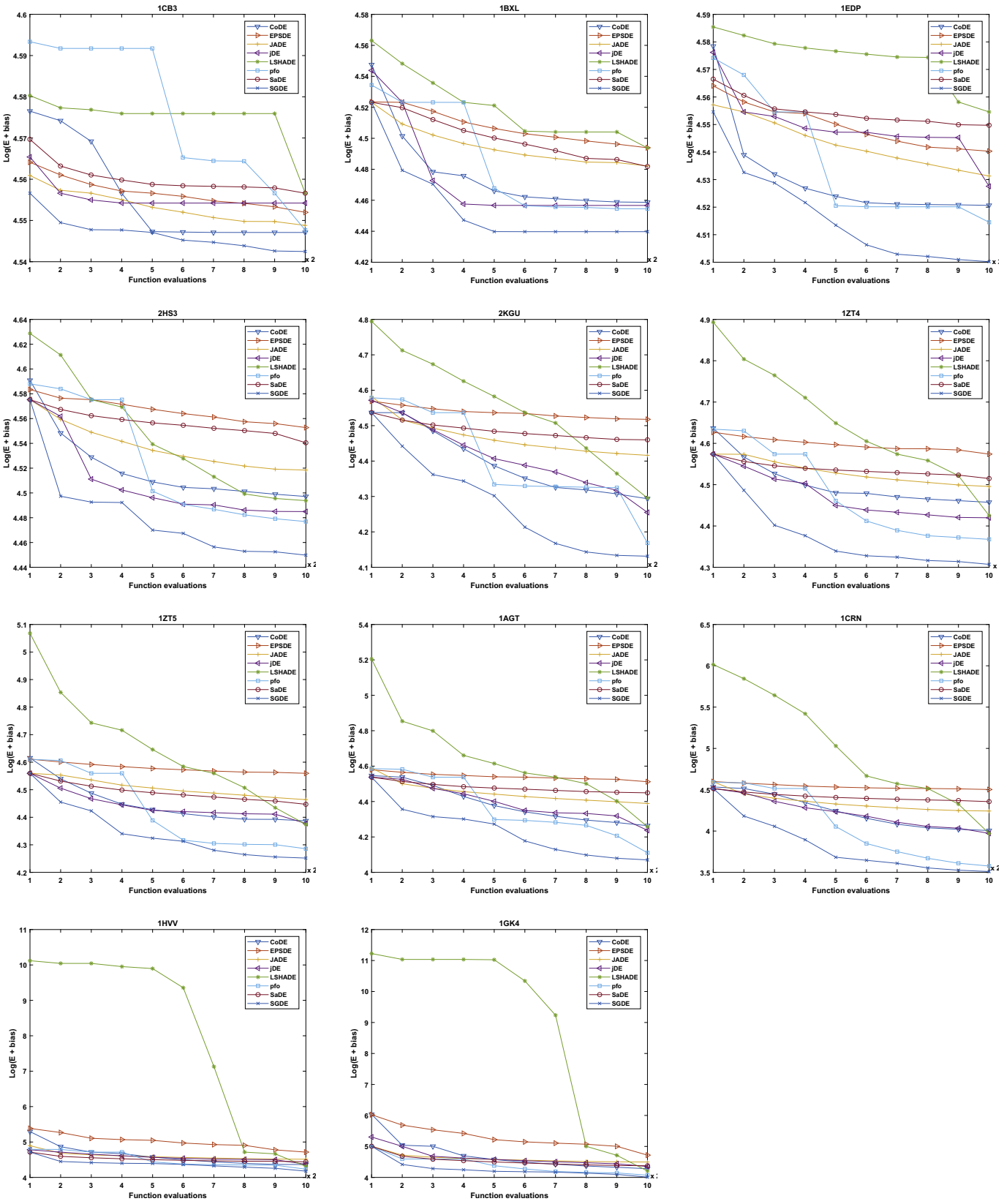


Fig. 14. The convergence curves of the improved algorithm over 30 runs per sequence. A bias value of 100 is added to make the energy values positive.

algorithms are trapped in local optima while SGDE successfully escaped from the local optima and found the global optimum. The main point is that the introduced surrogate models provide a measure of uncertainty associated with the DE configurations. This uncertainty can effectively be used to construct more promising strategies during the optimization process.

Thereafter, we compared the results by the DE_{pfo}, jDE and L-SHADE on the high dimensional sequence 2EWH after 800,000 function

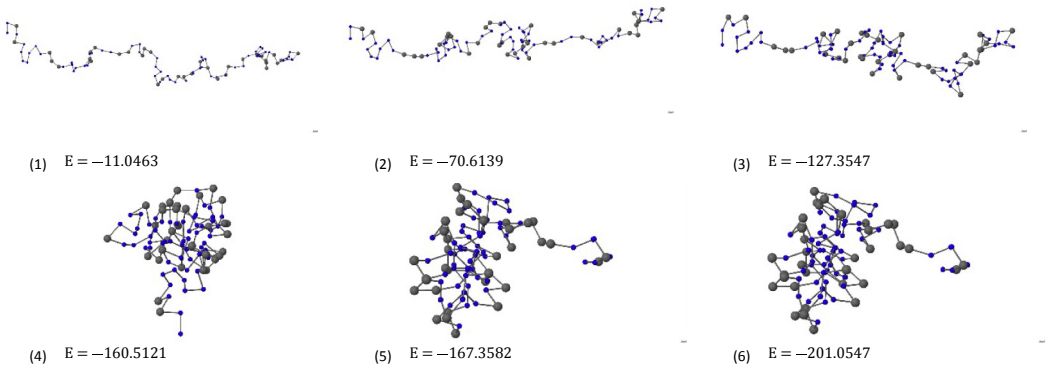


Fig. 15. Folding process for 2EWH using the proposed SGDE algorithm.

evaluations, and the best offered solution by the SGDE after 400,000 evaluations to further validate its approximation strategy. According to our experiments, SGDE was capable of obtaining the best energy value compared to the other competitive algorithms. The folding process for the 2EWH using the SGDE is depicted in Fig. 15 and the results are reported in Table 8.

Finally, we benchmark the performance of the SGDE for the real-world protein sequences. The results for 3 runs each with 1,000,000 function evaluations are collected in Table 9. The SGDE† uses only 500,000 evaluations. This simulation is computationally expensive. In this experiment, the total energy value (kJ/mol) of the obtained structure is used to assess the quality of the solutions from the optimization aspect, while the Root Mean Square Deviation (RMSD) is adopted to measure the similarity between the equivalent atoms in the obtained model and the native structure. We used the well-known BLAST server to provide the secondary structure of the target protein.

7.4. Analysis of the computation times

This study investigates how approximation strategy of surrogate models can be used to replace in part the original computationally expensive PSP solver which may take from several minutes to several hours. For such problem, time overhead of training and building the surrogate models is insignificant compared to evaluating the exact fitness function which can happen when you are working on a new fold. In this case, a high proportion of the processing time involved in running is spent in function evaluation and surrogate models can be used to ease the computational burden. The surrogate models themselves are often expensive and are not recommended for the simple problems. Hence, this section takes a more detailed look on the importance of surrogate models for the PSP in regard to execution time. Generally speaking, the main purpose of the surrogate based methods is to find optimal solutions within very few expensive evaluations. So, we are interested in the obtained computation times for the SGDE†.

The most direct of this section is to show that the surrogate models can successfully reduce the computational costs for the PSP problems. Toward this goal, the execution (CPU) times for the *ab-initio* and all-atom models are presented in Tables 10 and 11, respectively. We did not report the results for the DEpfo because it uses different setup, coding language, compiler and computational architecture. In Tables 10 and 11, the self-time value show the total time in seconds spent inside an algorithm, excluding time spent for the fitness evaluations. In the case of *ab-initio* model, we record the time for two low-dimension problems (1CB3, 1AGT) and a high-dimension problem (1GK4). The results of 1AGT using

Table 8
Comparison of the results between the improved algorithms and the SGDE (Std.: standard deviation).

Protein sequence	Algorithm	Mean	Std.	Best
2EWH	DE_{pfo}	−144.90	12.84	−171.63
	jDE (DE/best/1)	−88.830	20.29	−129.88
	L-SHADE	−104.96	4.930	−118.15
	SGDE	−149.15	10.75	−201.05

Table 9
The obtained results for the 1GK4 protein sequence based on the all-atom model.

Protein sequence	Algorithm	Run 1	Run 2	Run 3	RMSD
1GK4	DE_{pfo}	−769.36	−874.33	−657.32	0.05
	jDE (DE/best/1)	−468.36	−569.33	−496.32	0.07
	L-SHADE	−154.65	−256.32	−365.36	0.09
	SaDE	150.14	130.15	94.31	0.18
	CoDE	987.32	658. 12	498.32	0.21
	JADE	21.63	12.36	−0.65	0.17
	EPSDE	1001.12	987.65	698.78	0.23
	SGDE	−6036.14	−5964.15	−7973.82	0.03
	SGDE†	−2338.02	−1834.36	−1920.99	0.02

Table 10
The reported computational times based on AB off lattice model.

Protein sequence	Algorithm	Evaluation time (s)	Self-time (s)	Total time (s)
1CB3	jDE (DE/best/1)	7.86	0.72	8.58
	L-SHADE		1.21	9.07
	SaDE		8.04	15.90
	CoDE		0.88	8.74
	JADE		0.71	8.57
	EPSDE		4.37	12.23
	SGDE		59.1	66.80
	SGDE†	3.93	24.1	28.05
1AGT	jDE (DE/best/1)	65.73	0.92	66,65
	L-SHADE		1.56	67,29
	SaDE		8.39	74,12
	CoDE		0.86	66,59
	JADE		0.91	66,64
	EPSDE		4.92	70,65
	SGDE		80.03	145,8
	SGDE†	32.87	43.46	76.33
1GK4	jDE (DE/best/1)	325.60	0.93	326,53
	L-SHADE		3.61	329,21
	SaDE		10.25	335,85
	CoDE		0.85	326,45
	JADE		2.16	327,76
	EPSDE		5.38	330,98
	SGDE		100.76	426,36
	SGDE†	162.8	74.16	236.96

Table 11
The reported computational times based on the all-atom model for 3 runs.

Protein sequence	Algorithm	Evaluation time (m)	Self-time (m)	Total time (m)
1GK4	jDE (DE/best/1)	3142	219	3361
	L-SHADE		234	3376
	SaDE		281	3423
	CoDE		227	3369
	JADE		221	3363
	EPSDE		249	3391
	SGDE		321	3463
	SGDE†	1571	291	1862

the all-atom model are also collected in Table 11. In this table, the values are rounded down. The simulations performed under Windows 7 operating system on an Intel(R) Core i7-6700HQ CPU and 8 GB of RAM.

As can be seen from the results in Table 10, SGDE† needs a huge computational times compared to the other algorithms. However, it is not a big surprise due the fact that surrogate based methods such as SGDE are not developed for computationally cheap problems. Having this in mind, one can see that SGDE† significantly reduced the computational times for the 1GK4 problem. The effectiveness of this approach is further demonstrated in Table 11, where SGDE† convergence about 26 h faster for the 1GK4. The results also reveal the fact that even by considering SGDE, computational complexity of the algorithm become negligible compared to time overhead resulting from the expensive evaluations.

8. Conclusion and discussion

This study proposed a new extension of DE to accelerate the convergence rate of the standard algorithm for the computationally expensive PSP problem. The introduced SGDE verifies convergence conditions by adopting the surrogate modeling and GEP techniques. Moreover, it provides an improved search process which guides solutions by using CMAES. We evaluate the performance of SGDE as a specific algorithm for solving high dimensional real-world PSP problems using both *ab-initio* and an all-atom model. The SGDE is compared with ABC, DE, PSO, CMAES, DEpfo, jDE (DE/best/1), L-SHADE, SaDE, CoDE, JADE and EPSDE. In this regard, six performance metrics are used: best fitness value, mean value of solutions, standard deviation, percentage of improvements, convergence rate and runtime complexity of the algorithms. Tables 5 and 6 presented the results of competitive algorithms based on the *ab-initio* model. These results clearly showed that SGDE significantly outperformed other algorithms in terms of the solution accuracy, robustness and PI. The Friedman statistical test followed by a Conover method is in agreement with previous observations. From Fig. 13, we also see that SGDE is less sensitive to the increases in dimensionality of the sequences and thus, has proved to be scalable. Moreover, it has been illustrated that SGDE provides a highly enhancement in term of the convergence speed as presented in Fig. 14. The same conclusions can be drawn for the all-atom model when considering the results in Table 9. As we tried to show how approximation strategy of surrogate models can be used to speed up the PSP process, the results of the proposed SGDE with only 100,000 function evaluations are also taken into account during all the experiments. Experimental results suggest that the adopted surrogate models lead to a high convergence rate using a limited computational budget.

As a different approach, we tried to incorporate the surrogate models during the PSP process. The main direction was to investigate how surrogate models can be used to evaluate more conformational search by means of cheap surrogate models. As the future work, we want to introduce the surrogate models into the well-known Rosetta [39] predication tools in order to provide more details on the use of surrogate models for the PSP. Furthermore, we should provide a user-friendly web interfaces in order to generate reliable models using the proposed approach. They allow non-expert users to generate 3D models without the need to install and learn complex molecular modeling software and have increasing impacts on both basic research and drug development. As pointed out and demonstrated in a series of recent publications [40], an established web-server gives a step-by-step guide on how to use the proposed algorithms to get the desired results without the need to follow the complicated mathematic equations. Particularly, it would be even more useful if the users can testify their new proposition through this web interface. The fully automated server for PSP problem has been continuously developed since 20 years ago [40]. Actually, many practically useful web-servers have increasing impacts on medical science, driving medicinal chemistry into an unprecedented revolution [5]. Hence, we shall make efforts in our future work to provide a web-server for the new structure prediction method presented in this paper.

References

[1] J.R. Schnell, J.J. Chou, Structure and mechanism of the M2 proton channel of influenza A virus, Nature 451 (2008) 591.
[2] M.J. Berardi, W.M. Shih, S.C. Harrison, J.J. Chou, Mitochondrial uncoupling protein 2 structure determined by NMR molecular fragment searching, Nature 476 (2011) 109.
[3] K. Oxenoid, Y. Dong, C. Cao, T. Cui, Y. Sancak, A.L. Markhard, Z. Grabarek, L. Kong, Z. Liu, B. Ouyang, Architecture of the mitochondrial calcium uniporter, Nature 533 (2016) 269.
[4] J. Dev, D. Park, Q. Fu, J. Chen, H.J. Ha, F. Ghantous, T. Herrmann, W. Chang, Z. Liu, G. Frey, Structural basis for membrane anchoring of HIV-1 envelope spike, Science (2016), aaf7066.
[5] K.-C. Chou, An unprecedented revolution in medicinal chemistry driven by the progress of biological science, Curr. Top. Med. Chem. 17 (2017) 2337–2358.
[6] W.K. Surewicz, H.H. Mantsch, D. Chapman, Determination of protein secondary structure by Fourier transform infrared spectroscopy: a critical assessment, Biochemistry 32 (1993) 389–394.
[7] K.-C. Chou, A.G. Tomasselli, R.L. Heinrikson, Prediction of the tertiary structure of a caspase-9/inhibitor complex, FEBS Lett. 470 (2000) 249–256.
[8] T. Schwede, J. Kopp, N. Guex, M.C. Peitsch, SWISS-MODEL: an automated protein homology-modeling server, Nucleic Acids Res. 31 (2003) 3381–3385.
[9] A. Kryshtafovych, K. Fidelis, J. Moult, CASP10 results compared to those of previous CASP experiments, Proteins: Struct. Funct. Bioinf. 82 (2014) 164–174.
[10] S.H. de Oliveira, E.C. Law, J. Shi, C.M. Deane, Sequential search leads to faster, more efficient fragment-based de novo protein structure prediction, Bioinformatics 1 (2017) 9.
[11] B. Bošković, J. Brest, Differential evolution for protein folding optimization based on a three-dimensional AB off-lattice model, J. Mol. Model. 22 (2016) 252.
[12] K. Lindorff-Larsen, S. Piana, K. Palmo, P. Maragakis, J.L. Klepeis, R.O. Dror, D.E. Shaw, Improved side-chain torsion potentials for the Amber ff99SB protein force field, Proteins: Struct. Funct. Bioinf. 78 (2010) 1950–1958.
[13] R.B. Best, X. Zhu, J. Shim, P.E. Lopes, J. Mittal, M. Feig, A.D. MacKerell Jr., Optimization of the additive CHARMM all-atom protein force field targeting improved sampling of the backbone ϕ, ψ and side-chain χ1 and χ2 dihedral angles, J. Chem. Theor. Comput. 8 (2012) 3257–3273.
[14] J. Zhang, Z.-h. Zhan, Y. Lin, N. Chen, Y.-j. Gong, J.-h. Zhong, H.S. Chung, Y. Li, Y.-h. Shi, Evolutionary computation meets machine learning: a survey, IEEE Comput. Intell. Mag. 6 (2011) 68–75.
[15] Y. Jin, Surrogate-assisted evolutionary computation: recent advances and future challenges, Swarm Evol. Comput. 1 (2011) 61–70.
[16] J.R. Koza, Genetic programming as a means for programming computers by natural selection, Stat. Comput. 4 (1994) 87–112.
[17] N. Hansen, The CMA evolution strategy: a comparing review, in: Towards a New Evolutionary Computation, Springer, 2006, pp. 75–102.
[18] A.K. Qin, V.L. Huang, P.N. Suganthan, Differential evolution algorithm with strategy adaptation for global numerical optimization, IEEE Trans. Evol. Comput. 13 (2009) 398–417.
[19] A.K. Qin, P.N. Suganthan, Self-adaptive differential evolution algorithm for numerical optimization, in: Evolutionary Computation, 2005. The 2005 IEEE Congress on, IEEE, 2005, pp. 1785–1791.
[20] J. Brest, S. Greiner, B. Boskovic, M. Mernik, V. Zumer, Self-adapting Control Parameters in Differential Evolution: A Comparative Study on Numerical Benchmark Problems, vol. 10, 2006, pp. 646–657.
[21] Y. Wang, Z. Cai, Q. Zhang, Differential Evolution with Composite Trial Vector Generation Strategies and Control Parameters, vol. 15, 2011, pp. 55–66.
[22] R. Mallipeddi, P.N. Suganthan, Differential evolution algorithm with ensemble of parameters and mutation and crossover strategies, in: International Conference on Swarm, Evolutionary, and Memetic Computing, Springer, 2010, pp. 71–78.
[23] J. Zhang, A.C. Sanderson, JADE: Adaptive differential evolution with optional external archive, IEEE Trans. Evol. Comput. 13 (2009) 945–958.
[24] R. Tanabe, A. Fukunaga, Success-history based parameter adaptation for differential evolution, in: Evolutionary Computation (CEC), 2013 IEEE Congress on, IEEE, 2013, pp. 71–78.

[25] R. Tanabe, A.S. Fukunaga, Improving the search performance of SHADE using linear population size reduction, in: Evolutionary Computation (CEC), 2014 IEEE Congress on, IEEE, 2014, pp. 1658–1665.
[26] S.M. Elsayed, T. Ray, R.A. Sarker, A surrogate-assisted differential evolution algorithm with dynamic parameters selection for solving expensive optimization problems, in: Evolutionary Computation (CEC), 2014 IEEE Congress on, IEEE, 2014, pp. 1062–1068.
[27] C. Jin, A.K. Qin, K. Tang, Local ensemble surrogate assisted crowding differential evolution, in: Evolutionary Computation (CEC), 2015 IEEE Congress on, IEEE, 2015, pp. 433–440.
[28] R. Mallipeddi, M. Lee, An evolving surrogate model-based differential evolution algorithm, Appl. Soft Comput. 34 (2015) 770–787.
[29] N.D. Jana, J. Sil, S.J.I.S. Das, Selection of appropriate metaheuristic algorithms for protein structure prediction in AB off-lattice model: a perspective from fitness landscape analysis 391 (2017) 28–64.
[30] A. Díaz-Manríquez, G. Toscano-Pulido, W. Gómez-Flores, On the selection of surrogate models in evolutionary optimization algorithms, in: Evolutionary Computation (CEC), 2011 IEEE Congress on, IEEE, 2011, pp. 2155–2162.
[31] J. Park, I.W. Sandberg, Universal approximation using radial-basis-function networks, Neural Comput. 3 (1991) 246–257.
[32] S. Das, S.S. Mullick, P.N. Suganthan, Recent advances in differential evolution–an updated survey, Swarm Evol. Comput. 27 (2016) 1–30.
[33] R. Storn, K. Price, Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces, J. Global Optim. 11 (1997) 341–359.
[34] M.J.T. Stein, Large sample properties of simulations using, Latin Hypercube Sampling 29 (1987) 143–151.
[35] R.G. Regis, C.A. Shoemaker, A stochastic radial basis function method for the global optimization of expensive functions, Inf. J. Comput. 19 (2007) 497–509.
[36] F.A. Viana, R.T. Haftka, V. Steffen, Multiple surrogates: how cross-validation errors can help us to obtain the best predictor, Struct. Multidiscip. Optim. 39 (2009) 439–457.
[37] J.J.a.p.a. Müller, MATSuMoTo: the MATLAB Surrogate Model Toolbox for Computationally Expensive Black-Box Global Optimization Problems, 2014.
[38] B. Li, Y. Li, L. Gong, Protein secondary structure optimization using an improved artificial bee colony algorithm based onAB off-lattice model, Eng. Appl. Artif. Intell. 27 (2014) 70–79.
[39] C.A. Rohl, C.E. Strauss, K.M. Misura, D. Baker, Protein structure prediction using Rosetta, in: Methods in Enzymology, Elsevier, 2004, pp. 66–93.
[40] K.-C. Chou, H.-B. Shen, Recent advances in developing web-servers for predicting protein attributes, Nat. Sci. 1 (2009) 63.