Applied Soft Computing 49 (2016) 259–268

Contents lists available at ScienceDirect

Applied Soft Computing

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

Naïve Bayes ant colony optimization for designing high dimensional experiments

M. Borrotti [a,b,*], G. Minervini [d], D. De Lucrezia [e], I. Poli [b,c]

[a] Institute of Applied Mathematics and Information Technologies, CNR-IMATI, via Bassini 15, 20133 Milan, Italy
[b] European Centre for Living Technology, Ca' Foscari University of Venice, San Marco 2940, 30124 Venice, Italy
[c] Department of Environmental Science, Informatics and Statistics, Ca' Foscari University of Venice, Dorsoduro 2137, 30123 Venice, Italy
[d] Department of Biology, University of Padua, Via U. Bassi 58, 35121 Padua, Italy
[e] Explora Biotech S.r.l., Via della Libertá 9, 30175 Venice, Italy

ARTICLE INFO

Article history:
Received 15 October 2015
Received in revised form 8 July 2016
Accepted 10 August 2016
Available online 22 August 2016

Keywords:
Ant colony optimization
Naïve Bayes classifier
Experimental design
Enzyme engineering
High dimensionality

ABSTRACT

In a large number of experimental problems, high dimensionality of the search area and economical constraints can severely limit the number of experimental points that can be tested. Within these constraints, classical optimization techniques perform poorly, in particular, when little a priori knowledge is available. In this work we investigate the possibility of combining approaches from statistical modeling and bio-inspired algorithms to effectively explore a huge search space, sampling only a limited number of experimental points. To this purpose, we introduce a novel approach, combining ant colony optimization (ACO) and naïve Bayes classifier (NBC) that is, the naïve Bayes ant colony optimization (NACO) procedure. We compare NACO with other similar approaches developing a simulation study. We then derive the NACO procedure with the goal to design artificial enzymes with no sequence homology to the extant one. Our final aim is to mimic the natural fold of 200 amino acids 1AGY serine esterase from *Fusarium solani*.

1. Introduction

Enzymes are biological molecules that catalyze thousands of chemical reactions that sustain life. Each enzyme is a polymer generated from a series of up to 20 different amino acids. Polymers can be represented as words formed according to the alphabet $\mathbf{a} = \{a_1, a_2, a_3, \ldots, a_{20}\}$. They may differ in length, sequence and amino acid order. A secondary and tertiary structure, which defines enzyme activity, is associated to each sequence [1]. Enzyme activities can be associated to each of these combinations of amino acids through experimentation. Biological experimentation aims to optimize the sequence of amino acids in order to find enzymes with the best activities. The number of possible combinations of amino acids rapidly increase leading to a combinatorial explosion of the search space. Designing artificial enzymes with desired properties is known in computational biology as *Enzyme Engineering* [2,3]. Enzyme Engineering is an important task in numerous fields such as chemicals, pharmaceuticals, fuel, food or agricultural additives [4].

In this context it is necessary to develop new enzyme engineering procedures based on computational approaches able to identify best solutions with the lowest number of costly and time consuming experimentations. Different optimization approaches have been proposed in the literature to tackle problems with high dimensional and complex search spaces [5–11]. Some approaches are based on Evolutionary Computation Algorithms [12]. Genetic Algorithms (GAs) [13,14], for example, use techniques inspired by natural evolution, i.e. mutation, selection and crossover, in order to optimize complex functions. The application of these approaches is limited by several issues such as the need of solid *a priori* knowledge of the relationship between the structure and function of an enzyme, the vast number of possible modifications of an existing enzyme, the lack of efficient screening procedures able to reduce the number of modified enzymes to be tested and the computational effort needed to create new enzymes. A promising solution is combining metaheuristic algorithms and statistical models. This is a novel research area that addresses problems characterized by large design space, high-order interactions between variables, and complex and non-linear experimental surfaces [15]. Some examples of these hybridized methods can be found in [16–21]. A recent

[*] Corresponding author at: Institute of Applied Mathematics and Information Technologies, CNR-IMATI, via Bassini 15, 20133 Milan, Italy.
E-mail address: matteo.borrotti@mi.imati.cnr.it (M. Borrotti).

http://dx.doi.org/10.1016/j.asoc.2016.08.018
1568-4946/© 2016 Elsevier B.V. All rights reserved.

approach is proposed in [21] where an optimization algorithm, called Co-Information Composite Likelihood (COIL), based on the evolutionary paradigm is introduced by coupling cross-entropy sampling [22] and composite likelihood principles [23] to design novel enzymes with improved functionalities.

In this paper, we introduce a novel approach that combines ant colony optimization (ACO) [24–26] and the naïve Bayes Classifier (NBC) [27], called *naïve-Bayes ant colony optimization* (NACO). The ACO approach has been widely used in many optimization problems ranging from sequential ordering studies [28] to open shop scheduling [29]. ACO has been also applied to biological problems such as DNA sequencing problems [30,31].

Several theoretical and practical features of NBC have been studied that have shown its reliability in terms of classification accuracy [32–37]. One of the most powerful procedures proposed in the biological field is an algorithm for genetic biomarker selections and subject classifications from the simultaneous analysis of genome-wide Single-Nucleotide Polymorphism (SNP) data based on the naïve Bayes framework [37].

The key idea of NACO is to include information achieved via the NBC method in the path construction mechanism of ACO, in order to drive the search process towards the target region in a much more effective way. NACO was evaluated both in a simulation study and in a real case of building artificial enzymes. In particular, the NACO algorithm was constructed and a simulation comparison with a set of other procedures frequently used for this purpose was developed. The algorithm was then derived with the goal of designing artificial enzymes with no sequence homology to extant ones; the final aim being to mimic the natural fold of 200 amino acid long 1AGY serine esterase from *Fusarium solani* [38].

This paper is organized as follows. In Section 2 the optimization problem is formalized and in Section 3, we briefly describe the ACO and NBC algorithm. In Section 4, we introduce our approach, naïve-Bayes ant colony optimization (NACO). In Section 5, we evaluate NACO performance in a simulation study, comparing results with well-known techniques and, in Section 6, we apply NACO to the design of new enzymes. Finally, some conclusions are drawn in Section 7.

2. The optimization problem

Enzyme engineering can be described as a combinatorial optimization problem $P = (\mathcal{X}, f)$ in which:

- f is the objective function to be optimized and defined as $f : D_1 \times D_2 \times \cdots \times D_d \rightarrow \mathbb{R}^+$ where $D_1, \ldots, D_d$ are the domains of the following variables in $X = \{x_1, \ldots, x_d\}$;
- $\mathcal{X}$ is the set of all possible combinations $\mathbf{x} = (x_{1l_i}, \ldots, x_{dl_i})$ where $x_{il_i} \in D_i$ indicates the value of the variable x_i $\forall\, i$ where $i = 1, \ldots, d$.

The function f calculates a response value for each combination of variables, $y = f(x_{1l_i}, \ldots, x_{dl_i})$ with $l_i = 1, \ldots, k$. Within this setting, the final aim is to find the combination, or solution, $\mathbf{x}^* \in \mathcal{X}$ with maximum response value, that is, $f(\mathbf{x}^*) > f(\mathbf{x}) \forall \mathbf{x} \in \mathcal{X}$.

To address the enzyme engineering problem we propose that the variables represent the set of amino-acids in the enzyme sequence, the variable domain is the alphabet **a**, $\mathcal{X}$ is the set of all possible candidate enzymes to be tested and the objective function is the enzyme activity to be maximized.

3. Ant colony optimization and naïve Bayes classifier

In this section we briefly introduce ant colony optimization (ACO) and naïve Bayes classifier (NBC), which we intend to combine in an new approach to high dimensional optimization problems.

3.1. Ant colony optimization (ACO)

Ant colony optimization (ACO) [24] is a metaheuristic algorithm introduced for complex combinatorial problems. It is inspired by the behaviour of specific real ant colonies in search of food in their environment. In nature ants explore the surrounding environment for food source through random walks. Once found the food, ants return to their colony leaving behind a trail of pheromones. Pheromones are volatile compounds that quickly dry off so that a short path from colony to food source is more likely to preserve detectable amount of pheromones. This in turn will allow more ants to follow that path and thus reinforce the pheromone trails. This positive feedback eventually leads all ants to use a single path from the colony to the food.

The main idea of the ACO algorithm is to mimic this behaviour with *artificial ants* walking around a graph representing the environment; more specifically, the problem to solve is illustrated in Fig. 1.

ACO algorithm is applied to problems that can be described as a graph $G = (N, A)$ where N is the set of nodes and A the set of arcs that fully connect the nodes. A weight $\tau_{i,j}$, called pheromone value, is associated with each arc which connects node i with node j. The pheromone value represents the attractiveness of a specific arc for the ants: the higher the amount of pheromone on an arc, the higher the probability that ants will choose it when constructing solutions. In addition, a heuristic value $\eta_{i,j}$, which represents *a priori* information, is introduced to move from node i to node j.

The construction of the ACO algorithm includes the following phases [26,39]: (i) *Construct ant solutions*, (ii) *Daemon actions* and (iii) *Update pheromone*. The first phase, *Construct ant solutions*, represents the procedure needed for ants to construct solutions incrementally: starting from a first node, ants add a new node at each iteration of the procedure in order to build the entire solution. An ant decides where to go next in accordance with pheromone values $\tau_{i,j}$ and the heuristic values $\eta_{i,j}$. The relative influence of these two values is weighed in accordance to two parameters, called $\alpha > 0$ and $\beta \geq 0$, respectively. The second phase, *Daemon actions*, comprises all problem-specific operations that may be considered for boosting the performance of ACO algorithms. The main example of such operations is the introduction of local search techniques [40,41]. The third phase, *Update pheromone*, includes the procedure to updates pheromone values. This procedure is divided in two phases. First, pheromone evaporation is applied to decrease pheromone values. The degree of decrease depends on the parameter $\rho \in [0, 1]$, called the evaporation rate. The aim of pheromone

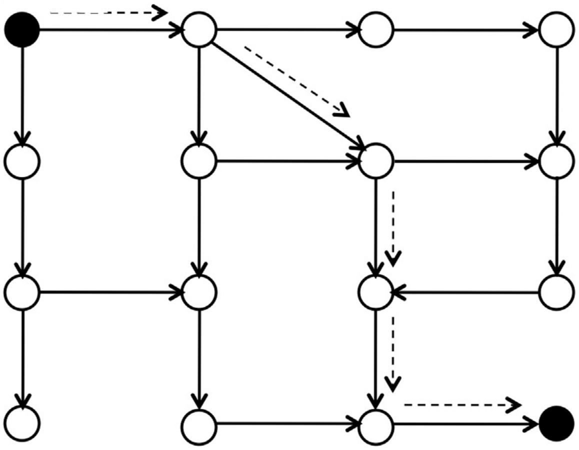

Fig. 1. Representation of a possible graph where ants move, from a starting node to a destination node.

evaporation is to avoid an unlimited increase in pheromone values and to allow the ant colony to forget poor choices made previously. A pheromone deposit is then applied to increase the pheromone values that belong to good solutions that ants have generated.

In the literature there are many different versions of this algorithm and one of the most powerful versions of ACO is the $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System ($\mathcal{MM}$AS) [42]. $\mathcal{MM}$AS is able to extensively exploit the search space by allowing only a single ant to deposit pheromone after each iteration, for example, the ant that reached the best solution in the current iteration, called *interation-best* solution s^{ib}. Moreover, it uses a simple method for limiting the strength of the pheromone trail that effectively avoids premature convergence of the search.

3.2. Naïve Bayes classifier (NBC)

The naïve Bayes classifier [27] is a probabilistic method, based on Bayes rule, which assumes that the variables $\{x_1, \ldots, x_d\}$ in the set X be all conditionally independent from each other, given the response value y. In other words, a naïve Bayes classifier assumes that the presence (or absence) of a particular variable in a candidate solution is unrelated to the presence (or absence) of any other variable and, further, assumes that interactions between variables do not influence the response value. More precisely, considering a candidate solution $\mathbf{x} = (x_{1l_i}, \ldots, x_{dl_i})$ where $x_{il_i} \in D_i$ indicates the value of the variable $x_i \, \forall \, i$ where $i = 1, \ldots, d$, we can calculate probability $P(x_{1l_i}, \ldots, x_{dl_i}|y)$ as follows:

$$P(x_{1l_i}, \ldots, x_{dl_i}|y) = \prod_{i=1}^{d} P(x_{il_i}|y). \tag{1}$$

The aim of the naïve Bayes classifier is to derive the probability distribution of y, for each new candidate solution not already tested. Starting from Eq. (1), it is possible to derive the naïve Bayes algorithm in the following way: y is assumed to be discrete with r possible values and variables $\{x_1, \ldots, x_d\}$ can assume any discrete or real values. The probability that y will take its rth possible value, according to Bayes rule, is given as:

$$P(y = y_r|x_{1l_i}, \ldots, x_{dl_i}) = \frac{P(y = y_r)P(x_{1l_i}, \ldots, x_{dl_i}|y_r)}{\sum_{j=1}^{r} P(y = y_j)P(x_{1l_i}, \ldots, x_{dl_i}|y_j)}, \tag{2}$$

where the sum is taken over all possible values y_j with $j = 1, \ldots, r$. Assuming then $\{x_1, \ldots, x_d\}$ to be conditionally independent given y, we can write the following equation:

$$P(y = y_r|x_{1l_i}, \ldots, x_{dl_i}) = \frac{P(y = y_r)\prod_{i=1}^{d} P(x_{il_i}|y_r)}{\sum_{j=1}^{r} P(y = y_j)\prod_{i=1}^{d} P(x_{il_i}|y_j)}. \tag{3}$$

Eq. (3) is the fundamental equation for the naïve Bayes classifier. Given a new combination of variables $\mathbf{x}^{new}$, this equation allows to derive the probability that y will take a possible value in its discrete domain, given $\mathbf{x}^{new}$ and the distributions $P(y)$ and $P(x_{il_i}|y)$ with $i = 1, \ldots, d$ estimated from the available set of candidate solutions and the associated set of response values.

4. Naïve-Bayes ant colony optimization (NACO)

In building the naïve-Bayes ant colony optimization (NACO), we assume that a colony of ants mutually collaborate to reach the optimum in a high dimensional and complex search space. This colony shares information and collaborates via pheromone values and information extraction by NBC. Pheromone update rules are applied once all ants have built their solutions based on the iteration-best solution, s^{ib}, and the probability distribution to obtain a certain response given a specific set o variables.

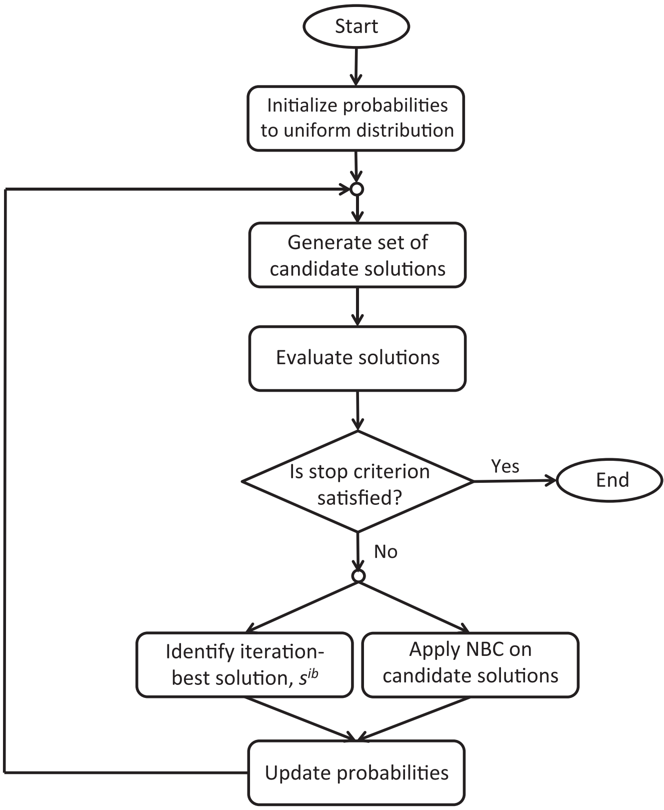
Start
Initialize probabilities to uniform distribution
Generate set of candidate solutions
Evaluate solutions
Is stop criterion satisfied?
Yes
End
No
Identify iteration-best solution, s^{ib}
Apply NBC on candidate solutions
Update probabilities

Fig. 2. Flow diagram of NACO approach.

The construction of NACO approach (see Fig. 2) can be describe as follows:

1. Initialization of the probability with which an ant w chooses t go from node i to node j. Each node has the same probability t be chosen.
2. Generation of a set of candidate solutions of size m (colony o ants).
3. Evaluation of the response y of each candidate solutions according to the objective function f.
4. If the optimal solution is found (or the maximum number o iterations is reached) stop the procedure. Otherwise, continue.
5. Identification of the iteration-best solution, s^{ib}.
6. Calculation of the NBC on the current set of candidate solutions At each iteration, NBC focuses on the values of the response greater than a problem-specific threshold γ, with $\gamma \in \mathbb{R}$.
7. Updating the probability with which an ant w chooses to go from node i to node j using the information extracted in points 3 an 4.
8. Go to point 2.

At each iteration, the candidate solution response is computed In order to update the probability with which an ant w chooses t go from node i to node j, NACO, first, identifies the iteration-bes solution, s^{ib} and then, in order to focus on the candidate solution that have reached the highest responses, candidate solutions are divided in two classes according to a certain constant threshol (γ, with $\gamma \in \mathbb{R}$) such as the specific quantile of the respons distribution. For instance, if we aim to minimize an objective function, γ can be equal to the first quartile. Consequently, the following rule will be applied:

$$y_r = \begin{cases} y_{r=1} = 1 & I(y \le \gamma) = 1 \\ y_{r=0} = 0 & I(y \le \gamma) = 0, \end{cases}$$

where $I()$ is the indicator function.

Given the set of candidate solutions and the assigned classes, the NBC derives the probabilities $P(y = y_r)$ and $P(x_{il_i}|y_r)$ with $r = \{0, 1\}$. Considering categorical variables these probability distributions are represented with conditional probabilities, which are estimated from the data by counting the occurrences of each variable and class label y_r. All the possible combinations obtained with the x_{il_i} in the candidate solutions with $y_{r=1} = 1$ are then considered and the corresponding $P(y = y_r|\mathbf{x}^{new})$ calculated. The possible solution with the highest probability is then selected and the related $P(x_{il_i}|y_r)$ are used to weigh on the arcs. These weights are named *Naïve Information* and denoted λ.

4.1. Probability update procedure

In NACO, each arc describes the probability of moving from node i to node j. This probability is usually calculated on the basis of pheromone quantity and heuristic information, i.e. a priori information concerning the problem. In our approach this probability also depends on Naïve Information λ.

Given the state of the graph at iteration $t - 1$ of the algorithm, the Naïve Information λ_{ij} is extracted from the set of solutions evaluated at iteration t and the set of $\{\lambda_{ij}\}$ is updated. Subsequently, the iteration-best solution s^{ib} is identified and the corresponding pheromone value is updated.

The probability at iteration t of going from node i to node j will be obtained by combining the pheromone values with the heuristic values (if available) and with Naïve Information as follows:

$$p_{ij}(t) = \frac{[\tau_{ij}(t)]^{\alpha}[\eta_{ij}]^{\beta}[\lambda_{ij}(t)]^{\delta}}{\sum_{p \in N_i}[\tau_{ip}(t)]^{\alpha}[\eta_{ip}]^{\beta}[\lambda_{ip}(t)]^{\delta}} \quad \forall p \in N_i, \tag{4}$$

where $\tau_{ij}(t)$ is the amount of pheromone value and $\lambda_{ij}(t)$ is the Naïve Information on arc (i, j) at iteration t and $\eta_{i,j}$ is the heuristic value on the same arc (i, j). N_i denotes the set of nodes that can be visited from node i and α, β and δ are the parameters that control the relative weight of the pheromone trail, heuristic information and Naïve Information. After all the ants have completed their tour, pheromone evaporation on all arcs is applied consistent with evaporation factor ρ, $\rho \in (0, 1]$.

ACO performance strongly depends on the availability of relevant *a priori* information of the problem. The introduction of Naïve Information aims to avoid the need for heuristic information in order to tackle high-dimensional design problems with computationally-expensive complex functions without a priori knowledge.

5. Simulation study

We evaluated the efficiency and robustness of NACO approach by developing a simulation study where we compared this approach with other optimization techniques. We considered two benchmark functions selected to test the performance of the techniques. These functions are frequently used to compare optimization procedures in the literature [21,43].

5.1. Benchmark functions

In our simulation study the response value of each solution was derived assuming the model $y = \varphi_h(\mathbf{x}) + \epsilon$, $\epsilon \sim N(0, \sigma_h^2)$, where $h = 1, 2$ denotes two types of response surface. The optimal solution *minimizes* the deterministic function φ_h, which was interpreted as the objective function. We computed Monte Carlo simulation composed of 50 runs and characterized by d variables with k levels each. A run was stopped if the global minimum was achieved before a maximum number of iterations $T = 30$.

The *Response Surface 1* takes the following form:

$$\varphi_1(x_1, \ldots, x_d) = 10d + \sum_{i=1}^{d}\sum_{l=1}^{k}\left[\beta_{il}^2 I_{il} - 10\cos(2\pi\beta_{il})I_{il}\right], \tag{5}$$

where $I_{il} = 1$, if $x_i = x_{il}$, and $I_{il} = 0$, otherwise. For all $i = 1, \ldots, d$, the constants β_{il}, $l = 1, \ldots, k$, formed an equally spaced grid of k elements on $(-5, 5]$. The function φ_1 has multiple local minima with similar values with respect to the global minimum.

The *Response Surface 2* takes the following form:

$$\varphi_2(x_1, \ldots, x_d) = \sum_{E \in \mathcal{P}:|E| \le d^*} \eta_E, \tag{6}$$

where η_E is the effect corresponding to the index set $E \in \mathcal{P}$, $\mathcal{P} = \mathcal{P}(\{1, \ldots, d\})$ the power set constructed from indexes $\{1, \ldots, d\}$ and the sum was taken over all variable combinations with at most d^* elements. The effects η_E are computed as $\eta_E = \prod_{i \in E} \beta_{il_i} I_{il_i}$, where I_{il_i} is an indicator variable taking value 1, if $x_i = x_{il_i}$, $(l_i = 1, \ldots, k)$ and the coefficients $\beta_{il_i} \in \{\beta_1, \ldots, \beta_k\}$ formed a equally spaced grid of k elements on the interval $(0, 1]$. In all the simulations presented here d^* is set equal to $\lceil d/2 \rceil$ in accordance to [21]. So, when using $d = 4$ variables, we compute only main effects and two-ways interactions. If $d = 20$, we take all main effects and all interactions with at most $d^* = 10$ variables.

5.2. Search approaches for comparison

In order to validate the performance of NACO we selected three approaches from the literature:

- $\mathcal{MAX}$-$\mathcal{MIN}$ Ant System ($\mathcal{MM}$AS) [42] is based on three key aspects. First of all, in the colony of ants only a single ant is used to update the pheromone trails after each iteration helping to exploit the best solutions found during an iteration or during the run of the algorithm. The second aspect restricts the range of possible values for the probability of choosing a specific arc which helps to avoid early stagnation. The final aspect requires that the pheromone trail is deliberately initialized to a maximum value, τ_{max}. Stützle et al. [42] show by numerical analysis that $\mathcal{MM}$AS achieves a strongly improved performance compared to other versions of ACO algorithm.

  We implemented an $\mathcal{MM}$AS with $\alpha = 1$, without any *a priori* information regarding the problem and with evaporation rate $\rho = 0.88$. The parameter ρ was used to avoid unlimited accumulation of the pheromone trails enabling the algorithm to forget bad decisions previously taken. In fact, if an arc is not chosen by the ants, its associated pheromone value decreases exponentially in the number of iterations [44]. Since evaporation rate ρ has an essential role in the algorithms performance we decided to perform a dedicated tuning analysis. The results are shown in Fig. 3 where the behaviour of $\mathcal{MM}$AS with different ρ values is reported. From the figure we notice that when $\rho = 0.88$ the algorithm is able to outperform the other versions of $\mathcal{MM}$AS in the two benchmark functions with $k = 13$ and $d = 2$.
- Genetic Algorithm (GA) [13,14] is considered one of the most powerful algorithms for optimization. GAs are adaptive heuristic search algorithms based on the evolutionary idea of natural selection. They are based on a population of solutions that undergo

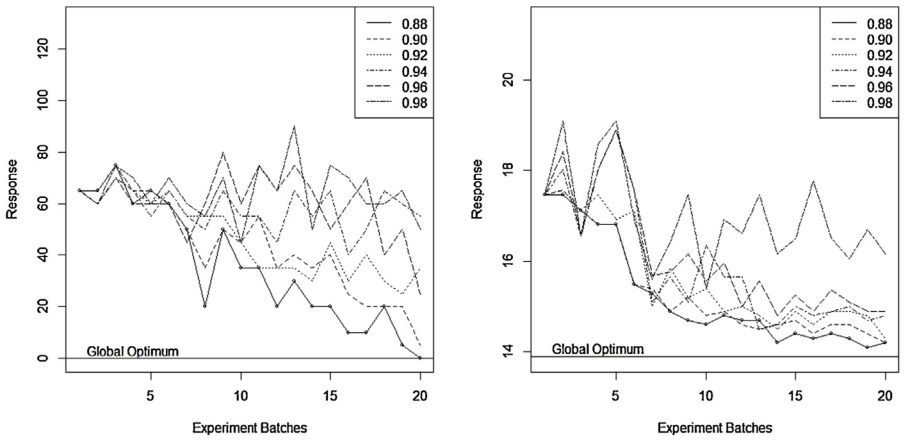

Fig. 3. Tuning analysis of $\mathcal{MM}$AS with different values of ρ with $k = 13$ and $d = 2$.

selection in the presence of variation-inducing operators such as mutation and recombination (crossover). Further, a fitness function is used to evaluate solutions, and the selection process is based on fitness values (responses).

We implemented a GA where the candidate solutions were generated by a one-point crossover with a 5% mutation rate and proportional selection as sampling schemes [21].

- Co-Information Composite Likelihood (COIL) [21] optimization is a novel approach for high dimensional problems. COIL is an adaptive procedure where new candidate solutions are chosen by minimizing the co-information composite likelihood objective function, derived from coupling importance sampling and composite likelihood principles.

  COIL depends on two parameters: (i) a fixed threshold γ that divides the response in two classes and (ii) a parameter η that controls the trade-off between exploration and exploitation of the search space. In these simulations, where the aim is to minimize the response, γ is the first quartile of the response distribution and $\eta_t = 1 - 1/(1 + t)$, $t \leq T$ where t is the current iteration and T the maximum number of iterations. This choice for η_t is motivated by the need to thoroughly explore $\mathcal{X}$ during the initial iterations and exploit the best solutions more efficiently later on [21].

NACO was tested with the following fixed parameters: (i) $\rho = 0.88$ and (ii) the constant threshold γ was equal to the first quartile as in COIL. α and δ were decided in accordance with preliminary results as shown in Fig. 4 and have been fixed to $\alpha = 1$ and $\delta = 2$. We did not report the results obtained with response surface φ_1 since the different settings gave similar performance.

In all the four approaches the number of candidate solutions to be tested at each iteration was set as $m = k \times 20$ in order to have enough candidate solutions and related responses to obtain reliable estimations at each iteration [21]. A tuning parameter procedure was implemented in [21] to optimize the performance of GA and COIL on the same benchmark functions used to optimize the NACO parameters.

5.3. Computational results

In order to compare the performance of NACO with the approaches presented in Paragraph 5.2, two indicators were derived to show the main difference between the four algorithms

- expected number of iterations to hit the optimal solutions, denoted by EHI;
- explored fraction of the search space, $\%\mathcal{X} = (EHI \times m \times 100)/dim(\mathcal{X})$ where $dim(\mathcal{X})$ is the dimension of the whole experimental search space.

To understand how NACO behaved when the dimension of the search space increased we considered various values for k and d, corresponding to $dim(\mathcal{X}) = k^d \approx 10^4$, 10^5 and 10^6. The results are reported in Table 1. Table 1(a) reports the results obtained for $dim(\mathcal{X}) \approx 10^4$. The benchmark function considered for the specific simulation is reported in the first column. In columns 2, 3, 4 we report the number of levels k, the number of variables d considered in the specific instance and the number of candidate solutions tested at each iteration. The value of EHI and the explored fraction of the search space was then calculated for each method. In Table 1(b) and (c) the same results are reported but considering larger search spaces, such as $dim(\mathcal{X}) \approx 10^5$ and $dim(\mathcal{X}) \approx 10^6$.

Our numerical studies show that NACO converges to the optimum in all the design space sizes and requires a very small number of tested solutions in all simulations. This is a very small number, in fact, at $dim(\mathcal{X}) \approx 10^4$ NACO needs just less than three iterations to reach the optimum, equal to 0, of response surface

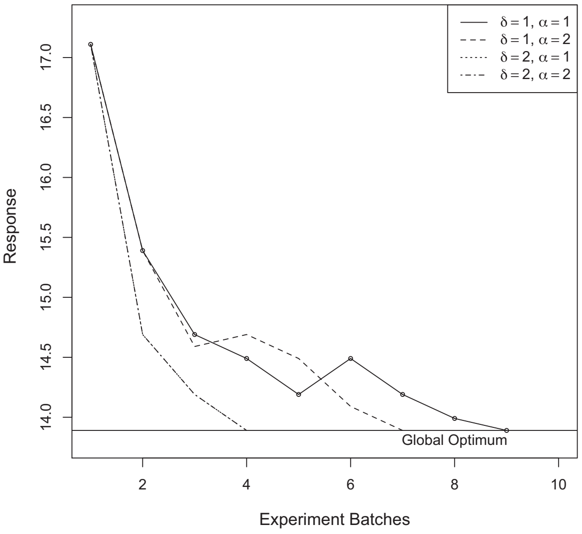

Fig. 4. Tuning analysis of NACO with different values of δ and α with $k = 13$ and $d = 2$ and response surface φ_2.

Table 1
Expected number of iterations needed to achieve the optimal solution (EHI) and the correspondent fraction of explored search space (% $\mathcal{X}$) for $\mathcal{MM}$AS, COIL and GA.

	k	d	m	NACO		$\mathcal{MM}$AS		COIL		GA	
				EHI	% $\mathcal{X}$	EHI	% $\mathcal{X}$	EHI	% $\mathcal{X}$	EHI	% $\mathcal{X}$
(a) $dim(\mathcal{X}) \approx 10^4$											
	2	13	40	**2.74**	1.34	16.26	7.94	3.42	1.67	8.30	4.05
φ_1	4	7	80	**2.22**	0.54	14.86	3.63	2.94	0.72	6.34	1.55
	10	4	200	**2.48**	0.99	18.66	7.46	3.30	1.32	8.60	3.44
	2	13	40	**3.48**	1.70	17.68	8.63	4.26	2.08	9.36	4.57
φ_2	4	7	80	**2.86**	0.70	23.10	5.64	4.12	1.01	9.16	2.24
	10	4	200	**3.20**	1.28	20.60	8.24	4.64	1.86	7.80	3.12
(b) $dim(\mathcal{X}) \approx 10^5$											
	2	17	40	**3.34**	0.20	23.70	1.45	4.26	0.26	10.80	0.66
φ_1	4	8	80	**2.54**	0.31	21.52	2.63	3.48	0.48	7.72	0.94
	10	5	200	**3.38**	0.27	29.04	2.32	4.60	0.37	12.24	0.98
	2	17	40	**5.14**	0.31	23.28	1.42	7.16	0.44	12.14	0.74
φ_2	4	8	80	**3.34**	0.41	26.44	3.23	4.66	0.57	10.36	1.26
	10	5	200	**4.56**	0.36	28.44	2.28	6.54	0.52	16.38	1.31
(c) $dim(\mathcal{X}) \approx 10^6$											
	2	20	40	**3.62**	0.07	27.70	0.52	4.52	0.09	13.28	0.25
φ_1	4	10	80	**2.96**	0.06	27.16	0.52	4.20	0.08	10.60	0.20
	10	6	200	**4.12**	0.08	28.42	0.57	6.06	0.10	16.08	0.32
	2	20	40	**6.12**	0.12	27.08	0.52	11.20	0.21	16.96	0.32
φ_2	4	10	80	**5.02**	0.10	29.60	0.56	6.98	0.13	16.74	0.32
	10	6	200	**5.96**	0.12	30.00	0.60	7.12	0.14	19.24	0.38

φ_1 and less than four iterations for response surface φ_2, with optimum value equal to 13.89. The method performs particularly well when $dim(\mathcal{X}) \approx 10^6$ considering response surface φ_1, since NACO needs less than 0.07% (in average) of the total space to achieve the optimum. $\mathcal{MM}$AS performs very badly in all the simulations because it does not benefit from the inclusion of heuristic information, an essential component in ACO algorithms. This result shows that the combination of the naïve Bayes classifier in the ACO framework boosts the performance of NACO outperforming all the other approaches.

If d and k increase, NACO is still able to reach the optimum in a reasonable time. In fact, considering response surface φ_2, $k = 10$ and $d = 6$, NACO requires 5.17 iterations (in average) to reach the optimum, which is 61.67. In the other case, response surface φ_1, NACO requires 3.57 iterations (in average) to reach the optimum, which is 0.

In Fig. 5, we compare the convergence of our method with COIL, $\mathcal{MM}$AS and GA methods. This analysis confirms that NACO outperforms $\mathcal{MM}$AS and GA, in fact NACO typically reaches the minimum faster than the other two approaches. NACO and COIL approaches have similar performance in almost all dimensions.

We further investigated the difference between NACO and all the other approaches performing a pairwise Student's t-test. The null hypothesis, H_0, is that the average best value is equal between two approaches otherwise the two values are different (alternative hyphotesis, H_1). If the *p-value*, p, is less than (or equal to) α_p then the null hypothesis is rejected in favor of the alternative hypothesis. NACO is significantly different with $\alpha_p = 0.05$ in a pairwise comparison with $\mathcal{MM}$AS in all instances. NACO is also significantly different with $\alpha_p = 0.05$ with GA method considering $k = 2$ and $d = 20$ and $k = 10$ and $d = (5, 6)$ with response surface φ_1. If we consider response surface φ_2, NACO is significantly different with $k = 4$ and $d = 10$ and $k = 10$ and $d = (5, 6)$. If we set $\alpha_p = 0.10$, NACO is significantly different in comparison with GA method in all instances except for $k = 4$ and $d = (7, 10)$ with φ_1 and $k = 2$ and $d = (13, 17)$ with φ_2. The pairwise Student's t-test confirms that NACO and COIL have similar behaviour in all instances since we accepted the null hypothesis H_0 with $\alpha_p = 0.05$.

To further understand the main difference between NACO and COIL, we investigate the computational time as a function of the average number of seconds needed to reach the optimal value. We then ran the two algorithms assuming the same setting of the previous study. Also in this case, we computed a Monte Carlo simulation composed by 50 runs. A run was stopped if the global minimum was achieved before a maximum number of iterations $T = 15$. For this study, we decided to limit the number of iterations since we are interested in the computational cost and not in reaching the optimum. The results are reported in Table 2 in which the average number of seconds, μ_{sec}, and relative standard deviation, σ, are reported for both algorithms. In terms of computational time, NACO performs better than COIL in almost all dimensions. In some cases the difference is large. For instance, NACO requires less than 13 s to reach the optimum when $k = 2$ and $d = 20$ whit φ_1. COIL needs more than 38 min.

Also in this study, we applied a pairwise Student's t-test to assess if the average computational time is significantly different between the NACO and COIL methods, alternative hyphotesis H_1. The NACO *p-value*, p, is less than $\alpha_p = 0.05$, then the alternative hypothesis is accepted in almost all instances except for $k = 10$ and $d = (5, 6)$ with φ_2.

6. The artificial 1AGY serine esterase

The NACO approach was used to address the problem of designing an artificial enzyme which mimics the fold (i.e. three-dimensional shape) of the natural enzyme *1AGY* of fungus *Fusarium solani* [38] (see Fig. 6). Artificial enzymes composed of 4 subsequences ($d = 4$) linked together sequentially were built and a solution $\mathbf{x}$ was then made from a sequence of the four variables. Each subsequence was embodied by a specific string of amino acids called pseudo-domains chosen from a collection of 95 different pseudo-domains. Accordingly, the corresponding variable domain was composed of $k = \{1, \ldots, 95\}$ levels. The search space $\mathcal{X}$ comprised all the possible permutations repeating k pseudo-domains in d positions equalling $95^4 = 8.1 \times 10^7$ possible solutions. For example, feasible solutions were $\mathbf{x}_i = (66, 37, 66, 14)$ and $\mathbf{x}_j = (37, 14, 66,$

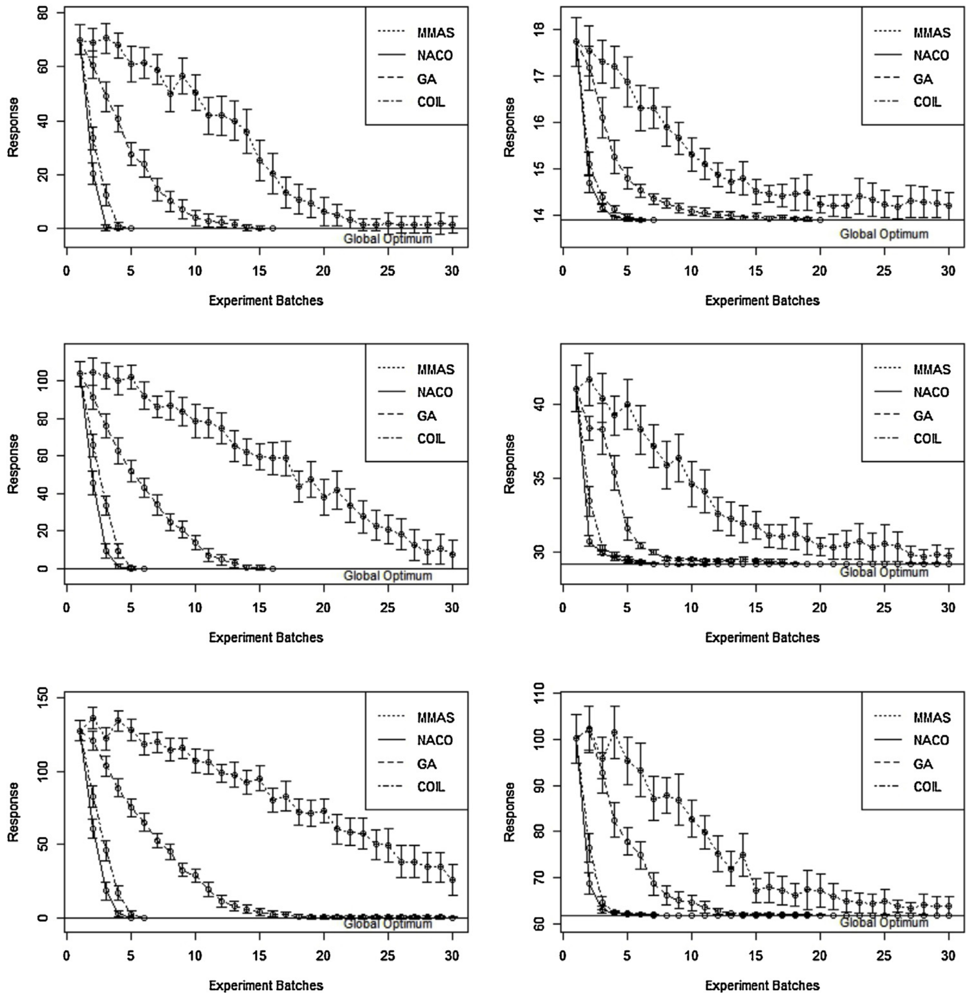

Fig. 5. Convergence behavior for Experiment 1 (φ_1) and Experiment 2 (φ_2) for the NACO (solid lines), COIL (dashed lines with dots), $\mathcal{MM}$AS (dotted lines) and GA (dashed lines) methods. The first, second and third rows of the display show Monte Carlo averages of the best response value for d = 13, 17, 20, respectively, and k = 2. The vertical bars are 95% Monte Carlo confidence intervals.

66). In this problem, the objective function was to maximize the structural similarity between a given candidate artificial enzyme and the natural enzyme 1AGY [45]. The similarity score is calculated by measuring the 2D structure similarity to the target enzyme by pairwise alignment of 2D structures predicted using PSIPRED algorithm [46,47]. The score ranges from 0 to 1000, the higher the score the higher is the structure similarity between the candidate artificial enzyme and the target enzyme.

Generally, in enzyme engineering problems, direct experimental evaluation of potential solutions is the only means for knowing how an enzyme performs. Experimentation, however, is costly and time consuming. In our study, real experimentation was coupled with the NACO algorithm. Solutions were evaluated in the real world by conducting physical experiments and by creating new candidate solutions used in a computational setting. At this point, evaluations were fed back to the computational phase of the approach and the generation of subsequent solutions was a function of these.

The maximum number of iterations was estimated in 5 cycles considering current lab procedures and corresponding costs. Furthermore, at each iteration, the number of candidate solutions was 96, which corresponded to the technology used for experimentations (96 microwell plate).

6.1. Graph representation

To address the *Enzyme Engineering* problem of designing a particular enzyme which mimics the natural enzyme 1AGY, we

Table 2
Average number of seconds, μ_{sec}, needed to reach the optimum. The maximum number of iterations was fixed to T = 15. The standard deviation, σ, is also reported.

			2^{13}	2^{17}	2^{20}	4^{7}	4^{8}	4^{10}	10^{4}	10^{5}	10^{6}
φ_1	NACO	μ_{sec}	**9.57**	**10.46**	**12.25**	**12.95**	**10.75**	**16.79**	67.08	63.03	**124.45**
		σ	1.13	5.69	3.19	7.12	6.14	6.15	21.44	27.79	42.20
	COIL	μ_{sec}	12.33	230.70	2288.67	6.66	33.75	6303.60	**3.39**	**40.83**	679.23
		σ	1.73	65.05	821.85	1.29	5.18	15908.38	1.28	6.55	88.28
φ_2	NACO	μ_{sec}	**9.21**	**27.07**	**55.19**	17.74	**16.80**	**35.96**	47.04	92.62	**1765.89**
		σ	3.09	20.89	59.32	7.73	6.18	16.50	31.11	74.57	5035.15
	COIL	μ_{sec}	13.49	468.05	17549.74	**7.86**	38.86	1772.35	**4.04**	**56.37**	8639.33
		σ	3.81	256.62	23912.79	2.27	12.97	1095.41	1.81	11.78	18928.35

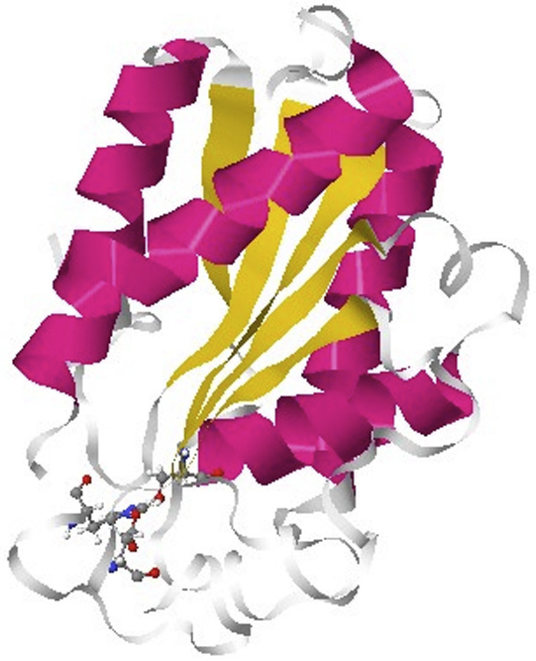

Fig. 6. The crystal structure of target protein 1AGY. The purple elements represent alpha helices, the blue ones turns and the yellow ones a beta strand. Gray lines are predicted coil with no defined secondary structure. Catalytic residues are represented in ball-and-stick configuration.

designed a network where a node corresponds to a pseudo-domain and an arc corresponds to the connection between variable *i* in a position and variable *j* in the next position of the enzyme sequence (see Fig. 7). The total number of nodes was 95 × 4 = 380 since an enzyme is composed of 4 subsequences ($d = 4$) and each subsequence can be chosen among $k = 95$ different pseudo-domains. Therefore, a candidate solution is a path composed of 4 variables (4 nodes); where this path represents a full-length enzyme. We assumed that no heuristic information was available with regards to the problem.

6.2. Results

To address the problem of discovering the optimal combination that mimics the fold (i.e. three-dimensional shape) of the natural enzyme *1AGY*, we developed NACO and we compared the results with the COIL approach. The main descriptive statistics of the response values are reported in Table 3. NACO identified the higher response value after four iterations reaching the response value 845.0 and sampling 384 points in the search space. COIL needed 480 points to reach the same value. This result shows that in an experimental setting NACO, compared to COIL, is more efficient in terms of reduced costs. Further, the third quantile in Table 3, shows that NACO has the capacity to move faster towards the optimal region with respect to COIL, incrementing the value of the third quantile by about 50% from the first to the second iteration. This is also confirmed applying a Student's *t*-test between values achieved by NACO and COIL methods. We performed a pairwise Student's *t*-test for each iteration with $H_1 : \mu_c^{NACO} \neq \mu_c^{COIL}$ with $c = 1, \ldots, 5$. Considering iteration $c = 1$, we accepted H_0 since the two approaches start from the same set of artificial enzymes. Considering iterations $c = 2$ and 3, the average biological fitness score responses are significantly different between NACO and COIL, which confirms a different behaviour towards the optimal region. In the last two iterations, $c = 4$ and 5, we accepted H_0. If we consider all the artificial enzymes tested by the two approaches, the average biological fitness score responses are significantly different.

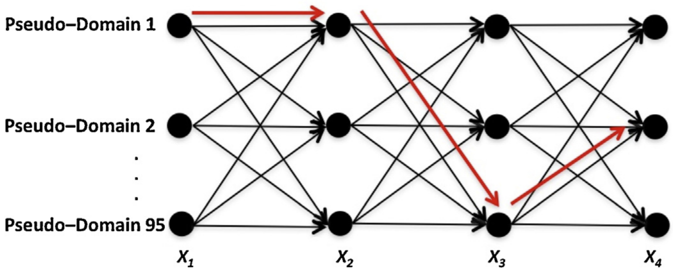

Fig. 7. Ant colony graph representation where ants move from x_i to x_j. A solution is a path composed by *d* nodes and $d - 1$ arcs.

In Fig. 8, we reported the best enzyme sequence (see [5,11] for more details about amino acids sequences), found by NACO, that mimics the tertiary structure of the natural enzyme 1AGY. 1AGY is characterized by bilaterally symmetric with a beta-sheet core and a set of alpha helices connected by unstructured loops. The best artificial enzyme found by NACO in the fourth iteration presents a solid structure around a beta-sheet core and bilateral symmetry similar to the natural enzyme 1AGY. Although the best artificial enzyme identified in the fourth iteration lacks some of

001 - 050	E A C I Q G E S P I W A V G R G H A T P A L R L T N V D T P Q K P D A Q K D G R R L L K Q E L T G L
051 - 100	D D S Y A V P R K A D C E T D G Q T R S E D S S K K Y K A S L F S A T M E V M Q Q W C Y K K T Q W D
101 - 150	Q P H V Q Y Q E E I A E S Q G D L P N P G K R F N N K D S G G E K L G D N N P T G K S V P C V L A N
151 - 200	T G C D I S V N S H T Y A P P E T R A F D C S F K L S L E D I L A N K I R K L F A H I V S K C G W G

Fig. 8. Amino acids sequence of the best artificial enzyme found by NACO. Each row is a subsequence of 50 amino acids (i.e. pseudo-domain).

Table 3
Main descriptive statistics of NACO and COIL for the 5 iterations. In each iteration 96 solutions are tested.

Iter.	Min.	1st Qu.	Median	Mean	3rd Qu.	Max
(a) NACO performance						
1	189.0	327.5	379.0	384.0	432.0	596.0
2	306.0	568.8	601.5	602.7	647.2	792.0
3	436.0	534.2	589.0	598.3	650.2	830.0
4	**241.0**	**476.5**	**579.4**	**579.4**	**712.8**	**845.0**
5	402.0	509.5	582.0	602.4	751.2	841.0
(b) COIL performance						
1	189.0	327.5	379.0	384.0	432.0	596.0
2	230.0	338.8	399.5	424.8	498.2	759.0
3	217.0	412.0	493.0	482.4	549.8	754.0
4	219.0	491.0	562.5	552.8	615.2	780.0
5	400.0	533.5	585.0	590.1	643.0	845.0

the structural elements of the target enzyme (e.g. misplacement of the alpha helices with respect to beta core), it clearly shows an overall similarity to the target enzyme.

7. Conclusion

In this work, we have presented a novel approach for high dimensional problems with computationally-expensive complex functions without a priori knowledge. The main idea behind the new method is to combine the naïve Bayes classifier to ant colony optimization to create the resulting naïve Bayes ant colony optimization (NACO) algorithm.

Numerical results point out quick convergence to the global optimum. One interesting finding is the remarkable scalability of the method to the overall size of the design space $\mathcal{X}$. As the number of variables/variable interactions increases, NACO shows growing efficiency, measured in terms of the total number of experiments needed to reach the optimum. This aspect makes NACO a valid choice when dealing with high-dimensional settings characterized by large design spaces.

NACO has also been tested on a complex enzyme engineering problem with the aim of identifying an artificial enzyme which mimics the natural enzyme 1AGY. The approach represents an interactive process where the dialogue between design and laboratory experimentation at each generation creates a path in the combinatorial search space that may lead toward a region of optimality. The evolving design requires a small number of experimental points to test from generation to generation and, consequently, a limited investment in terms of resources. Furthermore, the small number of tests make each experimental phase extremely rapid.

Acknowledgements

This work was supported by the EU integrated project Programmable Artificial Cell Evolution (PACE) in FP6-IST-FET-002035, Complex Systems Initiative and by the Fondazione di Venezia – DICE project. The authors would like to acknowledge the European Centre for Living Technology (www.ecltech.org) for providing opportunities of presentation and fruitful discussion of the research. Furthermore, the authors are grateful to Thomas Stützle, Mauro Birattari and Alessandra Giovagnoli for their very useful suggestions. The authors gratefully acknowledge the editor and the referees for their comments and suggestions.

References

[1] C.-I. Branden, J. Tooze, Introduction to Protein Structure, Garland Pub., 1999.

[2] A. Zanghellini, De novo computational enzyme design, Curr. Opin. Biotechnol. 29 (C) (2014) 132–138.

[3] J. Damborsky, J. Brezovsky, Computational tools for designing and engineering enzymes, Curr. Opin. Chem. Biol. 19 (C) (2014) 8–16.

[4] B.M. Nestl, B.A. Nebel, B. Hauer, Recent progress in industrial biocatalysis, Curr. Opin. Chem. Biol. 15 (2) (2011) 187–193.

[5] A.M.M.S. Ullah, A DNA-based computing method for solving control chart pattern recognition problems, CIRP J. Manuf. Sci. Technol. 3 (4) (2010) 293–303.

[6] V.K. Garlapati, P.R. Vundavilli, R. Banerjee, Evaluation of lipase production by genetic algorithm and particle swarm optimization and their comparative study, Appl. Biochem. Biotechnol. 162 (5) (2010) 1350–1361.

[7] A.M. Mohsen, A.T. Khader, D. Ramachandram, An optimization algorithm based on harmony search for RNA secondary structure prediction, in: Z.W. Geem (Ed.), Recent Advances in Harmony Search Algorithm, Studies in Computational Intelligence, vol. 270, Springer Berlin Heidelberg, 2010, pp. 163–174.

[8] Y. Zhang, J. Xu, Z. Yuan, H. Xu, Q. Yu, Artificial neural network-genetic algorithm based optimization for the immobilization of cellulase on the smart polymer Eudragit L-100, Bioresour. Technol. 101 (9) (2010) 3153–3158.

[9] B. Blum, M.I. Jordan, D. Baker, Feature space resampling for protein conformational search, Proteins 78 (6) (2010) 1583–1593.

[10] M. Borrotti, D. De Lucrezia, G. Minervini, I. Poli, A model based ant colony design for the protein engineering problem, in: M. Dorigo, M. Birattari, G. Di Caro, R. Doursat, A. Engelbrecht, D. Floreano, L. Gambardella, R. Groß, E. Sahin, H. Sayama, T. Stützle (Eds.), Swarm Intelligence, Lecture Notes in Computer Science, vol. 6234, Springer Berlin Heidelberg, 2010, pp. 352–359.

[11] A.M.M.S. Ullah, D. D'Addona, N. Arai, DNA based computing for understandin complex shapes, BioSystems 117 (1) (2014) 40–53.

[12] K.A. De Jong, Evolutionary Computation, MIT Press, 2006.

[13] D.E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley Longman Publishing Co., 1989.

[14] J.H. Holland, Adaptation in Natural and Artificial Systems, MIT Press, 1992.

[15] R. Baragona, F. Battaglia, I. Poli, Evolutionary Statistical Procedures, Springer-Verlag, 2011.

[16] R. Berni, D. De March, F.M. Stefanini, T-optimality and neural networks: a comparison of approaches for building experimental designs, Appl. Stoch. Models Bus. Ind. 29 (5) (2013) 454–467.

[17] F. Caschera, M.A. Bedau, A. Buchanan, J. Cawse, D. de Lucrezia, G. Gazzola, M.M. Hanczyc, N.H. Packard, Coping with complexity: machine learning optimization of cell-free protein synthesis, Biotechnol. Bioeng. 108 (9) (2011) 2218–2228.

[18] J. Ji, R. Hu, H. Zhang, C. Liu, A hybrid method for learning Bayesian networks based on ant colony optimization, Appl. Soft Comput. 11 (4) (2011) 3373–3384.

[19] R. Tang, S. Fong, X.S. Yang, S. Deb, Integrating nature-inspired optimization algorithms to k-means clustering, in: Seventh International Conference on Digital Information Management, 2012, pp. 116–123.

[20] M. Forlin, D. Slanzi, I. Poli, Combining probabilistic dependency models and particle swarm optimization for parameter inference in stochastic biological systems, in: F.L. Gaol, Q.V. Nguyen (Eds.), Proceedings of the 2011 2nd International Congress on Computer Applications and Computational Science Advances in Intelligent and Soft Computing, vol. 145, Springer Berlin Heidelberg, 2012, pp. 437–443.

[21] D. Ferrari, M. Borrotti, D.D. March, Response improvement in complex experiments by co-information composite likelihood optimization, Stat. Comput. 24 (3) (2014) 351–363.

[22] Y. Rubinstein, D.P. Kroese, The Cross-Entropy Method: A Unified Approach to Monte Carlo Simulation, Randomized Optimization and Machine Learning, Springer Verlag, 2004.

[23] B.G. Lindsay, Composite likelihood methods, in: N.U. Prabhu (Ed.), Statistical Inference from Stochastic Processes, 1988, pp. 221–239.

[24] M. Dorigo, Optimization, learning and natural algorithms (Ph.D. thesis), Politecnico di Milano – DEI, 1992.

[25] M. Dorigo, V. Maniezzo, A. Colorni, Ant system: optimization by a colony of cooperating agents, IEEE Trans. Syst. Man Cybern. B: Cybern. 26 (1) (1996) 29–41.

[26] M. Dorigo, G. Di Caro, The ant colony optimization metaheuristic, in: D. Corne M. Dorigo, F. Glover, D. Dasgupta, P. Moscato, R. Poli, K.V. Price (Eds.), New Ideas in Optimization, McGraw-Hill, 1999, pp. 11–32.

[27] T.M. Mitchell, Machine Learning, McGraw-Hill, 1997.

[28] R. Montemanni, D.H. Smith, L.M. Gambardella, A heuristic manipulation technique for the sequential ordering problem, Comput. Oper. Res. 35 (2008) 3931–3944.

[29] C. Blum, Beam-ACO – hybridizing ant colony optimization with beam search: an application to open shop scheduling, Comput. Oper. Res. 32 (6) (2005) 1565–1591.

[30] C. Blum, M.Y. Vallès, M.J. Blesa, An ant colony optimization algorithm for DNA sequencing by hybridization, Comput. Oper. Res. 35 (11) (2008) 3620–3635.

[31] S.J. Shyu, C.Y. Tsai, Finding the longest common subsequence for multiple biological sequences by ant colony optimization, Comput. Oper. Res. 36 (1) (2009) 73–91.

[32] P.J. Bickel, E. Levina, Some theory for Fisher's linear discriminant function, naïve Bayes, and some alternatives when there are many variables than observations, Bernoulli 10 (6) (2004) 989–1010.

[33] I. Rish, An empirical study of the Naïve Bayes classifier, in: The IJCAI-01 Workshop on Empirical Methods in Artificial Intelligent, 2001, pp. 41–46.

[34] M. Yousef, M. Nebozhyn, H. Shatkay, S. Kanterakis, L.C. Showe, M.K. Showe, Combining multi-species genomic data for microRNA identification using a Naïve Bayes classifier, Bioinformatics 22 (11) (2006) 1325–1334.

[35] G.L. Rosen, E.R. Reichenberger, A.M. Rosenfeld, NBC: the Naïve Bayes classification tool webserver for taxonomic classification of metagenomic reads, Bioinformatics 27 (1) (2011) 127–129.

[36] M. Sahami, S. Dumais, D. Heckerman, E. Horvitz, A Bayesian approach to filtering junk e-mail, in: Learning for Text Categorization: Papers from the 1998 Workshop, 1998, pp. 1–8.

[37] F. Sambo, E. Trifoglio, B.D. Camillo, G.M. Toffolo, C. Cobelli, Bag of Naïve Bayes biomarker selection and classification from genome-wide SNP data, BMC Bioinf. 13 (14) (2012) 1–10.

[38] S. Longhi, M. Czjzek, V. Lamzin, A. Nicolas, C. Cambillau, Atomic resolution (1.0 a) crystal structure of *Fusarium solani* cutinase: stereochemical analysis, J Mol. Biol. 268 (4) (1997) 779–799.

[39] C. Blum, A. Roli, Metaheuristic in combinatorial optimization: overview and conceptual comparison, ACM Comput. Surv. 35 (3) (2003) 268–308.

[40] L. Gambardella, M. Dorigo, An ant colony system hybridized with a new local search for sequential ordering problem, INFORMS J. Comput. 12 (3) (2000) 237–255.

[41] P. Pellegrini, E. Moretti, A computational analysis on a hybrid approach: quick-and-dirty ant colony optimization, Appl. Math. Sci. 3 (23) (2009) 1127–1140.

[42] T. Stützle, H. Hoos, $\mathcal{MAX}$–$\mathcal{MIN}$ ant system, Future Gener. Comput. Syst. 16 (8) (2000) 889–914.

[43] X.-S. Yang, Engineering Optimization: An Introduction with Metaheuristic Applications, John Wiley and Sons, 2010.

[44] M. Dorigo, T. Stützle, Ant Colony Optimization, Bradford Company, Scituate, MA, USA, 2004.

[45] G. Minervini, G. Evangelista, L. Villanova, D. Slanzi, D.D. Lucrezia, I. Poli, P.L. Luisi, F. Polticelli, Massive non-natural proteins structure prediction using grid technologies, BMC Bioinf. 10 (6) (2009) 1–9.

[46] D.T. Jones, Protein secondary structure prediction based on position-specific scoring matrices, J. Mol. Biol. 292 (2) (1999) 195–202.

[47] D. Slanzi, D. De Lucrezia, I. Poli, Querying Bayesian networks to design experiments with application to 1AGY serine esterase protein engineering, Chemom. Intell. Lab. Syst. 149 (A) (2015) 28–38.