IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.Doi Number

Biomedical Classification Problems Automatically Solved by Computational Intelligence Methods

Luis Carlos Padierna[1]✉, Carlos Villaseñor-Mora[1], Silvia Alejandra Lopez Juarez[1]

[1] Universidad de Guanajuato – División de Ciencias e Ingenierías, Campus León, Loma del Bosque 103, Lomas del Campestre, 37150 León, México

Corresponding author: Luis Carlos Padierna (e-mail: lc.padierna@ugto.mx).

This work was supported by the Universidad de Guanajuato, Campus León with the research project CIIC-232/2019; the Secretaría de Innovación, Ciencia y Educación Superior (SIECES) del Estado de Guanajuato with the project IJ-19-36 and the Consejo Nacional de Ciencia y Tecnología (CONACYT) with the project CB-2016-288605.

ABSTRACT Biomedical classification problems are of great interest to both medical practitioners and computer scientists. Due to the harmful consequences of a wrong decision in this ambit, computational methods must be carefully designed to provide a reliable tool for helping physicians to obtain accurate predictions on unseen cases. Computational Intelligence (CI) provides robust models to perform optimization, classification and regression tasks. These models have been previously designed, mainly based on the expertise of computer scientists, to solve a vast number of biomedical problems. As the number of both CI algorithms and biomedical problems continues to grow, selecting the right method to solve a given problem becomes more challenging. To deal with this complexity, a systematic methodology for selecting a suitable model for a given classification problem is required. In this work, we review the more promising classification and optimization algorithms and reformulate them into a synergic framework to automatically design and optimize pattern classifiers. Our proposal, including state-of-the-art evolutionary algorithms and support vector machines, is tested on a variety of biomedical problems. Experimental results on benchmark datasets allow us to conclude that the automatically designed classifiers reach higher or equal performance than those designed by computer specialists.

INDEX TERMS biomedical classification problems, estimation of distribution algorithm, evolutionary algorithms, genetic programming, orthogonal polynomial kernels, support vector machines

I. INTRODUCTION

Biomedical engineering is a concept that bridges medicine and technology influencing fields of specialization such as: bioimaging, bioinstrumentation, biomolecular analysis, biomechanics and biomaterials [1]. Research conducted in these fields frequently faces classification problems, for instance when deciding: if a gene can be a candidate of risk to develop Autism [2], if a patient presents a Parkinson disease based on the information obtained by acoustic biomarkers [3], if a microscopic image of breast tumor tissue provides evidence that it is benign or malignant [4] or if thermal patterns can help on the diagnosis of diabetic foot [5].

Computational Intelligence has provided methods to solve biomedical classification problems for years. However, the task of selecting an appropriate set of algorithms for a given problem has been mainly delegated to an expert. Among classification methods, Support Vector Machines (SVMs) are one of the most successful approaches due to its generalization capability by conducting structural risk minimization, absence of local minima by solving a quadratic programming problem and representation based on few parameters [6] [7] [8].

The performance of SVMs is highly dependent on two of its components: kernel function and hyper-parameters [9] [10]. Concerning the kernel functions, different scenarios have emerged through the years to ensure that the best kernel function is used for a classification task. These scenarios include kernel generation from primary operations [11] [12] [13], and multiple kernel combination from existing kernels [14] [15] [16] [17] [18] [19]. Out of these, the latter has proved to be more convenient, since it combines the flexibility of kernel generation with the effectiveness of pre-designed kernels. Regarding the hyper-parameter tuning, several optimization methods have been applied, from the simple Grid Search and Random Search [20], to the more sophisticated Evolutionary Strategies [21], Genetic Algorithms [22], Bio-inspired Metaheuristics [23] [24] [25], and Estimation of Distribution Algorithms (EDAs) [26], among others. Recently, it has been shown that EDAs perform SVM hyper-parameter tuning more efficiently than other methods when solving biomedical classification tasks [27].

Some studies have described strategies for simultaneous kernel selection (or kernel generation) and hyper-parameter optimization [12] [18]; however, it can be argued that they lack convergence efficiency, explore a limited search space, and are prone to overfitting. In this work, a novel method is formulated to solve the problems of previous studies by

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2998749, IEEE Access

combining the advantages of evolutionary programming with the guided exploration of the estimation of distribution algorithms. Our method, hereafter referred as Smart Evolution of Ensemble Kernel for Support Vector Machines (SEEKS), consists of a Genetic Programming (GP) mechanism [28] able to build new multiple kernels based on different kernel families (or to select the best single kernel) and an EDA [29] adapted to the GP mechanism and aimed to build probability models based on estimations of the hyper-parameter distributions. Thus, our method performs hyper-parameter tuning of kernels as these are being evolved without adding any significant overhead.

Through robust experimentation, it is shown that SEEKS automatically achieves simultaneous kernel design and hyper-parameter tuning for SVM classifiers as successfully as previous methods designed by specialists, with higher computational efficiency.

The next section provides the background theory to understand the tasks of SVM-kernel design and hyper-parameter tuning. In Sect. III, our SEEKS method is justified and described. Sect. IV presents the experimental methodology followed and the results obtained in terms of the performance of the generated SVMs. In Sect. V the effectiveness and efficiency of our proposal are analyzed. Conclusions and further directions are offered in Sect. VI.

II. BACKGROUND

A. KERNEL FUNCTIONS FOR SUPPORT VECTOR MACHINES

Given a set of m training data points $\{\mathbf{x}_i, y_i\}_{i=1}^{m}$, where $\mathbf{x}_i \in R^d$ is the i-th input vector and $y_i \in \{+1, -1\}$ its corresponding class label; an SVM classifier in dual form can be formulated, introducing the Lagrange multipliers $\boldsymbol{\alpha}$ and following the Karush-Kuhn-Tucker conditions, as [30]:

$$Max\ L(\boldsymbol{\alpha}) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} y_i y_j \alpha_i \alpha_j K(\mathbf{x}_i, \mathbf{z}_j)$$
$$s.t.\ 0 \le \alpha_i \le \mathrm{C},\ \forall i = 1, \ldots, m\ \ \textstyle\sum_{i=1}^{m} \alpha_i y_i = 0 \quad (1)$$

where C is called a penalty factor, $\mathbf{x}_i, \mathbf{z}_j \in R^d$ are the i-th and j-th input vector, respectively; and the function $K(\mathbf{x}, \mathbf{z})$ defined on $R^d \times R^d$ is called a kernel if there exists a map ϕ from the space R^d to the Hilbert space, $\phi: R^d \to H$ such that $K(\mathbf{x}, \mathbf{z}) = \langle \phi(\mathbf{x}), \phi(\mathbf{z}) \rangle$ [10]. The kernel function can take different forms, like those in TABLE I. Hyper-parameters are those parameters of a kernel plus the parameters of a specific SVM formulation (e.g. the penalty factor C). Choosing among different kernels is equivalent to choosing among different SVM models [9]. A deterministic way to select the most appropriate kernel for a given classification problem is still unknown. Moreover, kernel selection is becoming more challenging because the number of valid kernels continues to grow as new kernel families are proposed. These families include: wavelet kernels [31], non-parametric kernels [32], and orthogonal polynomial kernels [33] [34] [35] [36]. Kernels in TABLE I have proved to be valid kernels since they satisfy the necessary and sufficient conditions established in Mercer's theorem [37]. Briefly stated, this theorem affirms that the series $\sum_{i=1}^{\infty} \lambda_i \phi_i(\mathbf{x}) \phi_i(\mathbf{z})$, in terms of eigenfunctions $\phi_i \in L_2(X \subseteq R^d)$ and positive associated eigenvalues λ_i, converges absolutely and uniformly to $K(\mathbf{x}, \mathbf{z})$ when the latter is symmetric and positive semidefinite. To be positive semidefinite a kernel function must comply with $\iint K(\mathbf{x}, \mathbf{z}) g(\mathbf{x}) g(\mathbf{z}) d\mathbf{x} d\mathbf{z} \ge 0$.

TABLE I
CLASSIC, WAVELET, NON-PARAMETRIC, AND ORTHOGONAL POLYNOMIAL KERNELS

Kernel (short label)	Expression	Eq.
Linear (L)	$K_{Lin}(\mathbf{x}, \mathbf{z}) = \mathbf{x}^T \mathbf{z}$	(2)
Polynomial (P)	$K_{Pol}(\mathbf{x}, \mathbf{z}) = (\mathrm{a}\mathbf{x}^T \mathbf{z} + \mathrm{b})^{\mathrm{n}}$	(3)
Radial (R)	$K_{RBF}(\mathbf{x}, \mathbf{z}) = exp(-\gamma \|\mathbf{x} - \mathbf{z}\|^2)$	(4)
Wavelet (W)	$K_{Wav}(\mathbf{x}, \mathbf{z}) = \prod_{j=1}^{d} \left(h \times e^{\frac{-\|(x_j - z_j)\|^2}{2a^2}} \right)$ with $h = cos(1.75 \times (\mathbf{x}_j - \mathbf{z}_j)/a)$	(5)
Non-param. (K_{11})	$K_{11}(\mathbf{x}, \mathbf{z}) = 1 + \sum_{i=1}^{3} (-\|\mathbf{x} - \mathbf{z}\|/\tau)^i / i$	(6)
Non-param. (K_{13})	$K_{13}(\mathbf{x}, \mathbf{z}) = 1 - sin(\pi \|\mathbf{x} - \mathbf{z}\|/2\,\tau)$	(7)
Legendre (E)	$K_{Leg}(\mathbf{x}, \mathbf{z}) = \prod_{j=1}^{d} \sum_{i=0}^{n} P_i(x_j) P_i(z_j)$	(8)
s-Hermite (H)	$K_{Her}(\mathbf{x}, \mathbf{z}) = \prod_{j=1}^{d} \sum_{i=0}^{n} He_i(x_j) He_i(z_j) (2^{-2n})$	(9)
Gegenbauer(G) cf. [36] for details	$K_{Geg}(\mathbf{x}, \mathbf{z}) = \prod_{j=1}^{d} \sum_{i=0}^{n} C_i^{\alpha}(x_j) C_i^{\alpha}(z_j) w_{\alpha}(x_j, z_j) u(C_i^{\alpha})^2$ With $w_{\alpha}(x_j, z_j) = \left((1 - x^2)(1 - z^2)\right)^{\alpha} + \epsilon, \alpha > -1$ $u(C_i^{\alpha}) = \left(\sqrt{n+1} \times \|C_i^{\alpha}(1)\|\right)^{-1}$	(10)

A current research trend consists of combining two or more kernels to increase the accuracy rate and generalization capability of SVMs. A combination of Mercer kernels is also valid under the closures in TABLE II (for a formal proof of these closures, cf. [38]). This approach has been followed in several relevant works [14] [18] [19] [32] [34] [35] [36], and has introduced the concept of a Multiple Kernel, denoted: $K_{\eta}(\mathbf{x}, \mathbf{z}) = f_{\eta}(\{K_i(\mathbf{x}^i, \mathbf{z}^i)\}_{i=1}^{P} \mid \eta)$ where $\mathbf{x}^i, \mathbf{z}^i \in R^{d_i}$; the kernel functions, $\{K_i: R^{d_i} \times R^{d_i} \to R\}_{i=1}^{P}$ take P feature representations (not necessarily different) of data instances; and the combination function $f_{\eta}: R^p \to R$ can be linear or nonlinear. The parameter η indicates that a certain set of predefined kernels is used (i.e., the kernels and their parameters are known before training) [39].

TABLE II
CLOSURES ALLOWING THE VALID COMBINATION OF KERNELS

$K(x, z)$	Description
$K(x, z) = K_1(x, z) + K_2(x, z)$	Closure under the sum.
$K(x, z) = aK_1(x, z)$	Multiplication by a scalar, $a \in R^+$.
$K(x, z) = K_1(x, z) K_2(x, z)$	Closure under product.
$K(x, z) = f(x) f(z)$	$f(\cdot)$ is a real-valued function on X.
$K(x, z) = K_3(\phi(x), \phi(z))$	Kernel composition.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2998749, IEEE Access

B. METAHEURISTIC ALGORITHMS

Metaheuristics refer to a family of approximate optimization techniques that provide acceptable solutions in a reasonable time for solving complex problems [40]. There exist several types of metaheuristics; however, only Genetic Programming and Estimation of Distribution algorithms will be considered in the implementation of SEEKS, since these are the more promising methods found in the literature related to kernel evolution [12] [18] [19] and hyper-parameter tuning of SVMs [23] [21] [20] [24] [25], respectively. The selection of an EDA among other metaheuristics is based on two reasons: first, its iterative mechanism naturally adapts to the GP algorithm; and second, it was proved to be the optimal hyper-parameter tuner of SVM classifiers when solving biomedical problems [27].

EDAs explore a solution space by iteratively building and sampling explicit probabilistic models of candidate solutions that guide the search [29]. The procedure is shown in LISTING 1.

LISTING 1. General Estimation of Distribution Algorithm

1 Generate an initial population of N solutions: $S^{(0)} = \{\mathbf{s}_i\}_{i=1}^{N}$ at iteration $t = 0$, uniformly
2 **while** (stop criteria are not met)
3 Compute objective function of each solution $g\left(S^{(t)}\right)$
4 Select a subset of the best solutions, $S_M^{(t)}$ from $S^{(t)}$
5 Build a probabilistic model $\boldsymbol{P}^{(t)}$ based on $S_M^{(t)}$
6 Sample $\boldsymbol{P}^{(t)}$ to generate new solutions $S'^{(t)}$
7 Substitute / Incorporate $S'^{(t)}$ into $S^{(t)}$
8 Update $t = t + 1$
9 **end while**
10 Output the best solution found, $\mathbf{s}^*$, as a result

By taking a specific probabilistic model, an EDA receives a certain name. Two of these particular cases of probabilistic models are considered in this work: the Univariate Marginal Distribution Algorithm (UMDA) [41], and the Boltzmann Univariate Marginal Distribution Algorithm (BUMDA) [42].

The UMDA builds a Gaussian model $\boldsymbol{P}$ from a set of solutions $S = \{\mathbf{s}_i\}_{i=1}^{M}$, with parameters given by:

$$\mu_\kappa = \frac{1}{M}\sum_{i=1}^{M} s_{i,\kappa} \quad \text{and} \quad \sigma_k = \left(\frac{1}{M-1}\sum_{i=1}^{M}\left(s_{i,\kappa} - \mu_\kappa\right)^2\right)^{1/2} \tag{11}$$

where M is the number of solutions considered to compute these parameters and $s_{i,\kappa}$ represents the κ-th component of a solution $\mathbf{s}_i$, which is a D-dimensional vector in $\Re^D$.

The BUMDA modifies UMDA by employing a model based on the Boltzmann distribution and incorporating a truncation selection method to accelerate convergence, as well as to free the user from having to set the number of samples that are selected for parameter estimation. The parameters of the Gaussian distribution used by the BUMDA are:

$$\mu_\kappa = \frac{1}{\tilde{g}}\sum_{i=1}^{M} s_{i,\kappa}\, g(\mathbf{s}_i) \qquad \sigma_\kappa = \left(\frac{1}{\tilde{g}+1}\sum_{i=1}^{M}\left(s_{i,\kappa} - \mu_\kappa\right)^2 g(\mathbf{s}_i)\right)^{\frac{1}{2}} \tag{12}$$

where M is adjusted by the automatic truncation method, $g(\mathbf{s}_i)$ is the objective function value of the i-th solution, and $\tilde{g}$ represents the sum of the objective function values over the M selected solutions.

The GP algorithm arose as an extension of the conventional genetic algorithm, and it is useful for discovering computer programs using the expressiveness of symbolic representation. The search space for GP is the space of all possible expressions that can be recursively created by compositions of the available functions and variables for a given problem [28]. In the case of the kernel evolution problem, functions can be any operator in TABLE II, and variables can be any kernel as those in TABLE I. The major difference of GP, to other evolutionary algorithms, is that the solutions are stored in variable-sized structures, commonly in parse trees where the operations are internal nodes and the variables are leaves [43]. These tree-based structures are used to combine kernels in our proposal. The pseudocode of the GP method is shown in LISTING 2.

LISTING 2. General Genetic Programming Algorithm

1 Generate an initial population of solutions: $S^{(0)} = \{\mathbf{s}_i\}_{i=1}^{N}$ at iteration $t = 0$, uniformly
2 **while** (stop criteria are not met)
3 Compute objective function of each solution $g\left(S^{(t)}\right)$
4 Select m individuals for reproduction $S_r^{(t)} \subset S^{(t)}$
5 Apply variation operators to $S_r^{(t)}$ and keep the offspring $S_o^{(t)}$
6 Compute objective function of each new candidate solution $g\left(S_o^{(t)}\right)$
7 Integrate candidates $S_o^{(t)}$ to the new population according to replacement operators
8 Update $t = t + 1$
9 **end while**
10 Output the best solution found, $\mathbf{s}^*$, as a result.

III. SMART EVOLUTION OF KERNELS FOR SVMs

Previous works have constructed single and multiple kernels by using Evolutionary Algorithms [12] [13] [14] [18] [19]. The first attempt to evolve kernels appears to be [14], where genetic algorithms were employed to explore possible kernel combinations. Subsequent studies [12] [18] [19] utilized GP with tree data structures to encode multiple kernels. Those works using GP adopted two different approaches: the first one focuses on combining vector operators to discover new kernel functions; this strategy introduces several problems (from numerical errors to poor results), mainly because it is prone to generate invalid kernels. The second alternative approach combines predefined kernels under some of the closures in TABLE II. Although this latter strategy has shown to be more consistent and it is adopted as part of our proposed method, it presents three major limitations:

(i) The hyper-parameters of evolved kernels were assigned unsystematically, leaving the burden of this task to the guided search of the GP algorithm. This raises two problems: the search space is extended dramatically, and the GP method is

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2998749, IEEE Access

overloaded, since it should control the convergence of both hyper-parameters and kernel shape.

(ii) Evolved kernels obtained from the GP mechanism are prone to overfitting, since the search process is guided just by the accuracy index (without considering if datasets are unbalanced or the amount of data required to build the decision function).

(iii) The terminal set and consequently the basis of the GP search space was limited to combinations of classic kernels (Linear, Sigmoid, Polynomial, and RBF).

Our current proposal removes these limitations of previous works through the following improvements:

(i) The GP search space is reduced and the kernel shape convergence is increased, by delegating the hyper-parameter tuning task to a synergic EDA mechanism.

(ii) One advantage of SVMs, over other classifiers, is the property to estimate its generalization capability as a function of the Proportion of Support Vectors (PSV) that defines the SVM hyperplane, here PSV = SV/N, where SV is the number of essential support vectors and N is the number of instances in the training dataset [9]. In our SEEKS method, the overfitting problem is considered by integrating the PSV as a complementary quality measure of evolved kernels.

(iii) Kernels recently developed have shown advantages when combined with classical kernels. In references [34], [35] and [44] the mixtures of classical, wavelet, and orthogonal polynomial kernels were shown to reach a better performance than classic kernels. As a natural next step, SEEKS enhances this expertise-based way of combining kernels by adapting a systematic tool for automatically exploring combinations of kernels from different families. To date, no single work has been found that reports the hyper-parameter tuning or the evolution of kernels based on different kernel families. Thus, another contribution is the potentially first analysis of these kernels in an evolutionary methodology.

A. THE SEEKS ALGORITHM

SEEKS is formulated to take advantage of the GP and EDA mechanisms so that, on each iteration, kernel evolution and hyper-parameter tuning are performed simultaneous and independently. The algorithm is overviewed in LISTING 3, its time complexity is presented in APPENDIX A, and the sequence of steps is detailed below.

1. In the first step, N kernel combinations are codified as binary trees, randomly populated from the lists of kernels and operators described in TABLE I and TABLE II, respectively. Hyper-parameters vectors are all of cardinality $\kappa = 6$, since $\mathbf{h} = (C, \gamma, a, b, n, \alpha)^T$. FIGURE 1 illustrates two possible initial kernel trees with corresponding hyper-parameter vectors.

2. In step two, each pair of kernel tree $\mathbf{k}$ and hyper-parameter vector $\mathbf{h}$ is evaluated as an SVM classifier on the same k-fold cross-validation for a given dataset. The objective function $g(\mathbf{k}, \mathbf{h})$ is the k-fold classification average performance. Then, the kernel population is sorted by this average.

3. Common stop criteria include: The max number of iterations reached, the tolerance between the best solution and possible objective function value was achieved, it has achieved a small deviation on the population performance, etc. If criteria are not satisfied, then, at iteration t perform steps 4 to 7.

LISTING 3. Pseudocode of the SEEKS Algorithm

1 Initialize a population of kernel trees $K^{(0)} = \{\mathbf{k}_i\}_{i=1}^{N}$ and a set of hyper-parameter vectors $H^{(0)} = \{\mathbf{h}_i\}_{i=1}^{N}$ following a uniform distribution. Set iteration $t = 0$
2 Compute objective function $g\left(K^{(0)}, H^{(0)}\right)$ for each kernel with its corresponding parameters on the same validation data.
3 **while** (stop criteria are not met)
4 Select the best M parameter vectors to update a probability model $\boldsymbol{P}^{(t)}$ by using (11) or (12). Also select kernel trees for reproduction $K_r^{(t)} \subset K^{(t)}$ with a crossover method.
5 Apply variation operators to $K_r^{(t)}$ and keep the offspring $K_o^{(t)}$
6 Sample new hyper-parameter vectors from $\boldsymbol{P}^{(t)}$ and compute $g\left(K_o^{(t)}, H^{(t)}\right)$ for each new kernel tree.
7 Integrate candidates $K_o^{(t)}$ to a new population according to replacement operators and update $t = t + 1$
8 **end while**
9 Output the best solution found, $(\mathbf{k}^*, \mathbf{h}^*)$, as a result.

4. The indexes of the best M kernel trees are used to update $\boldsymbol{P}^{(t)}$ for all continuous parameters. In the case of the discrete parameter n, a multinomial model is estimated based on the distribution of the M best degrees. After this, kernel trees are again selected from the population $K^{(t)}$ by a method such as tournament, reward-based or binary selection, etc.

5. Crossover and one-point mutation are used as variation operators. To avoid uncontrolled growth a bloat-control method, such as Tarpeian or others [45] is recommended.

6. Sample the value of the required hyper-parameters for the new individuals from the current probability model $\boldsymbol{P}^{(t)}$ and train the SVMs corresponding to the new individuals. Again, following the same k-fold cross-validation scheme using the partition obtained in step 2.

7. Update the population $K^{(t)}$ by replacing the worst-performing individuals with the top-performing new individuals from $K_o^{(t)}$. Increment the iteration counter and go to step 2.

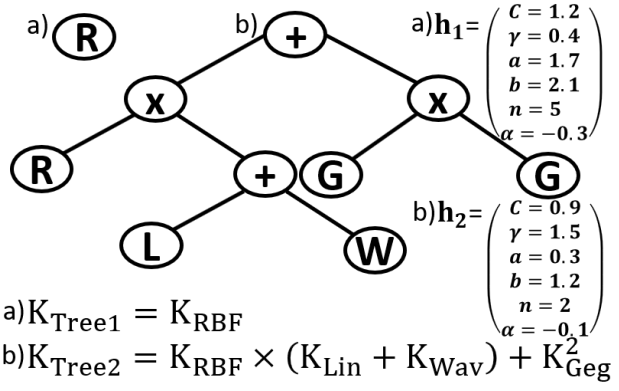

FIGURE 1. Kernel tree decoding. a) Kernel selection is possible when the root node is a kernel, irrelevant hyper-parameters are ignored. b) Kernel construction combining four basic kernels, hyper-parameters are shared.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2998749, IEEE Access

IV. EXPERIMENTAL DESIGN

A. BIOMEDICAL PROBLEMS

Fifteen benchmark biomedical classification problems were selected from related works to test the SEEKS method. Datasets (and short labels) corresponding to these problems are: breast cancer prediction (*breast*), chronic kidney disease prediction (*chronic*), vertebral column orthopedic normality (*column_2C*), wart treatment results by using cryotherapy (*cryotherapy*), type 2 diabetes diagnosis (*diabetes*), identification of altered sperm concentration (*fertility*), survival prediction after surgery of breast cancer (*haberman*), determination of heart disease (*heart*), wart treatment results by using immunotherapy (*immuno*), liver disorders caused by alcohol (*liver*), discrimination of benign and malignant mammographic masses (*mammo*), Parkinson prediction based on voice measurements (*parkinsons*), post-operative life expectancy in lung cancer patients after thoracic surgery (*thoracic*), prediction on the donation of blood (*transfusion*) and prognostic on breast cancer (*wpbc*).

All datasets are publicly available at the UCI Machine Learning Repository [46]. Their characteristics and results of previous studies are summarized in TABLE III. The best-reported rates, in classification accuracy by using the RBF kernel or Multiple kernels, are also included as a base-reference for comparison. All datasets were scaled to the range $[-1,1]$ to prevent the influence of attributes with dominating values and to preserve the conditions required by orthogonal kernels. For the sake of reproducibility, our pre-processed data can be found in the *datasets* folder of the public repository in [47] or [48].

TABLE III
SUMMARY OF BIOMEDICAL CLASSIFICATION PROBLEMS

	Dataset short label	Total cases (positive - negative)	Fts[1]	Best Accuracy[2] reported with RBF or *Multiple Kernels*	References
1	breast	683 (239-444)	10	*98.03* | 97.31 | 97.18	[18] [49] [27]
2	chronic	400 (150-250)	24	99.60	[27]
3	column_2C	310 (100-210)	7	87.00 | 86.02	[20] [27]
4	cryotherapy	90 (43-47)	6	91.00	[50]
5	diabetes	768 (268-500)	8	*81.25* | 77.73 | 76.83	[18] [49] [51]
6	fertility	100 (12-88)	9	88.00 | 89.19 | 88.04	[20] [27] [8]
7	haberman	306 (81-225)	3	73.55 | 75.77 | 75.91	[49] [52] [35]
8	heart	270 (120-150)	13	*86.98* | 83.70 | 84.67	[18] [51] [27]
9	immuno	90 (19-71)	7	88.00 | 85.46	[50] [53]
10	liver	345 (145-200)	7	72.45 | 74.20 | 74.78	[51] [54] [8]
11	mammo	961(445-516)	5	86.44	[52]
12	parkinsons	195 (48-147)	22	95.30 | 95.98 | 98.88	[24] [27] [55]
13	thoracic	470 (70-400)	17	85.30 | 85.15	[20] [27]
14	transfusion	748 (178-570)	4	75.00 | 80.53	[20] [52]
15	wpbc	194 (46-148)	33	80.09 | 81.22	[51] [52]

[1] Fts is the number of attributes (features) that model a case.
[2] Variations on accuracy are due to either the SVM solver algorithm applied or the validation scheme followed on each study (dataset pre-processing and partition, hyper-parameter tuning strategy, etc.)

B. KERNEL EVOLUTION SETTINGS

For training SVMs with kernel trees, the LIBSVM [56] solver was selected to achieve a direct comparison against previous works. Training a standard SVM has an algorithmic complexity between $O(N^2)$ and $O(N^3)$, with N the number of input vectors [57]. Furthermore, the associated Gram matrix of size $N \times N$ must be allocated. Thus, the computational cost of evolving SVMs is high for each evaluation of the fitness function (classification accuracy obtained by 5-fold cross-validation). Taking into account this cost and the fact that only small improvements of candidate solutions have been observed after the 15th generation [18], in our experiments the number of generations was set to 15 as a stop criterion.

The performance index that guides the search of SEEKS is the rate in classification accuracy. However, high classification rates may hide overfitting to the training data. One way to observe this case is through an estimation of the generalization capability of an SVM, such as the PSV. Therefore, producing a good solution with the minimum PSV is a desirable goal when evolving kernels.

For the sake of a direct comparison against previous works: the PSV is not used to guide the evolution of SVMs, but it is adopted as a complementary performance measure; parameters including tree depth, mutation, and crossover rates, as well as methods for initialization, selection, variation, and replacement were set to the values used in [18]. The function set was defined so that results could be directly compared with those reported in [34] and [35]. The terminal set includes the identifiers of kernels presented in TABLE I. This configuration is provided in TABLE IV.

TABLE IV
CONFIGURATION OF SEEKS FOR KERNEL EVOLUTION

Parameter	Koch et al. 2012	Diosan et al. 2012	Souza et al. 2017	**SEEKS**
Population size	20	50	200	**100**
Max Tree depth	---	10	100[1]	**2-5**
Generations	2500[2]	50	100	**15**
Mutation rate	---[3]	30%	30%	**30%**
Crossover rate	---	80%	90%	**90%**
Function set (FS)	$+,-,\times,$ %, exp, norm	$\times,+,$ exp	$\times,+,exp$ log, tanh	$\times,+$
Terminal set (TS)	x,z $c_1,c_2,$ c_3,c_4	$P,R,S,$ c_1,c_2	$(x^T z),$ $\|\|x-z\|\|^2$ others	**L, P, R, W, K_{11},K_{13}, E, G, H**
Initialization	Grow	Grow	GE	**Grow**
Selection method	Tourn-8	Tourn-2	Tourn-2	**Tourn-2**

[1] As GE does not use a tree structure, a max depth tree is not required. Instead, the max size of expression is indicated.
[2] Instead of generations, Koch et al used 2500 kernel tree evaluations.
[3] Values marked with "---" were not reported by the authors of that experiment.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2998749, IEEE Access

C. PARAMETER SETTING AND EXPERIMENTAL GOALS

The standard C-SVM considers few hyper-parameters, namely: the penalty factor C, and the kernel parameters, such as the decaying parameter γ, the degree n, and the dilation factor a for the RBF, orthogonal and Wavelet kernels, respectively. The work of Sun et al. [51] is the only one that we have identified that addresses the hyper-parameter tuning of orthogonal kernels (through Grid Search). No studies have been found about the influence of the Wavelet kernel parameters, so the values of the dilation parameter a are taken from the original work [31]. The non-parametric and Linear kernels do not possess parameters. From the analysis of these works, the hyper-parameter setting used to perform tuning is summarized in TABLE V.

TABLE V
CONFIGURATION OF SEEKS FOR HYPER-PARAMETER TUNING

Parameter	Experimental Setting
Previous Works	
RBF kernel decaying γ [18]	$\gamma_{qt} = q \cdot 10^t$, $q = \{1,2,\dots,9\}$, $t = \{-5,-4,\dots,-1\}$
Polynomial order n [18]	$n \in \{1,2,\dots,15\}$
Regularization C, Kernel decaying γ [51]	$C \in \{2^{-1}, 1,2^1,\dots 2^5\}$ $\gamma \in \{2^{-6}, 2^{-5},\dots,1\}$
Polynomial order n Kernel decaying γ, [51]	$n \in \{2,3,\dots,6\}$ $\gamma \in \{0.1,0.25,0.5,1,1.5,2,2.5,3\}$
SEEKS	
Regularization C	$C \in (0,32]$
RBF kernel decaying γ	$\gamma \in (0,4]$
Polynomial order n	$n \in \{1,2,\dots,6\}$
Wavelet dilation factor and classic polynomial scale a	$a \in (0,2]$
Classic polynomial offset b	$b \in [0,5]$
Gegenbauer α	$\alpha \in (-1,1]$
UMDA-Sample size M	25% of the population

Two experiments were performed with the settings described in TABLES III-V. The objective of the first experiment is to analyze the effect (in terms of effectiveness and efficiency) of reformulating an EDA to synchronize with the GP mechanism, which is the main idea of the SEEKS algorithm. Both EDAs detailed in Sect. II, UMDA and BUMDA, were implemented so that three different kernel evolutionary algorithms were compared. These algorithms are referred to as GP, GP-U, and GP-B as short names for the standard Genetic Programming with totally random hyper-parameter setting, GP coupled with UMDA and GP coupled with BUMDA, respectively.

The second experiment is aimed to observe if there exist improvements in classification performance when varying the terminal set of kernels being evolved. Three different terminal sets were employed: the set of classic kernels ($clas = \{L,R,P\}$), the set of modern kernels ($mod = \{K\_11, K\_13, H, E, W, G\}$), and the set of all kernels ($all = \{L,R,P,K\_11,K\_13,H,E,W,G\}$).

Both experiments were carried out on the same framework, following the methodology described in LISTING 4. Also, they were performed on an Intel® Core™ i9-9980XE CPU (18 cores) running at 3.0 GHz and with 16 GB of RAM. The algorithms were implemented in the Java programming language using multithreading, where each thread execute an independent call of the SEEKS algorithm. Experimental results were analyzed with the Python programming language and are reported in the following section. Java and Python codes will be updated in the *codes* folder in [47].

LISTING 4. Experimental Methodology

INPUT: User-defined parameters for kernel evolution, hyper-parameter ranges, number of trials, performance metrics, etc.
OUTPUT: Performance Indexes (PI: Accuracy, PSV, G-mean, etc)

```
FOR EACH experimental trial (i) DO:                                   i = 1, ..., 5
    (K_i^(0), H_i^(0)) ← Generate new initial population              Same initial
    of kernel trees and hyper-parameters vectors                      population for
                                                                      all algorithms
    FOR EACH D ∈ Datasets DO:
        Partition D into F = {Train_j, Test_j}_{j=1}^5                5-fold cross-validation
        FOR EACH algorithm-terminal_set (A, T)                        A ∈ {GP, GPU, GPB}
        pair RUN IN PARALLEL:                                         T ∈ {clas, mod, all}
            (K_i^*, H_i^*) ← SEEKS_A(K_i^(t), H_i^(t), F, T)          Listing 3
            PI_{i,A,T} ← save_file_{A,T}(K_i^*, H_i^*)                9 files per dataset
```

V. EXPERIMENTAL RESULTS AND DISCUSSION

A. EFFECTIVENESS AND EFFICIENCY OF THE SEEKS

The results of all the algorithms evaluated are summarized in TABLE VI. Concerning the first experiment, on TABLE VI it is possible to observe that, in almost all cases, both versions of the SEEKS algorithm (GP-U and GP-B) reached higher accuracies than the GP with a random hyper-parameter search. This is an important finding, because it means that the effect of an EDA coupled to the GP mechanism improves the performance on the classification rate of evolved SVMs.

To draw stronger conclusions regarding this finding, three statistical tests were applied: *Friedman*, *Aligned Friedman*, *and Quade*. These tests differ in the way of ranking each algorithm. In Friedman test, differences between the accuracy index values are equally weighted without considering their magnitude; in Aligned Friedman test, the biomedical problems are equally weighted independently of their difficulty, thus evaluating a general behavior of all datasets as a group; and in Quade test, rankings are weighted based on the difficulty of each dataset [58]. The three tests are helpful to determine if there exist differences in at least one of the algorithms under comparison. Once a difference is found, *post hoc* tests (such as Bonferroni-Dunn, Holm, Holland, Rom or Finner) are used to conclude which algorithms are statistically different from a pre-selected control method [59]. A significance level of 0.05 was used for all statistical tests. The average ranks and *p-values* of these tests are reported in TABLES VII-IX.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2998749, IEEE Access

TABLE VI
AVERAGE ACCURACIES AND STANDARD DEVIATIONS OF NINE PAIRS (ALGORITHM, TERMINAL SET). STATISTICS WERE COMPUTED FROM AROUND 2500 KERNELS FOR EACH PAIR (100 KERNELS EVOLVED DURING THE LAST FIVE GENERATIONS OF FIVE INDEPENDENT RUNS).

	algorithm	GP_clas		GP-U_clas		GP-B_clas		GP_mod		GP-U_mod		GP-B_mod		GP_all		GP-U_all		GP-B_all	
	dataset	**mean**	**std**	**mean**	**std**	**mean**	**std**	**mean**	**std**	**mean**	**std**	**mean**	**std**	**Mean**	**std**	**mean**	**std**	**mean**	**std**
1	breast	97.45	0.2	**97.60**	0.2	97.51	0.2	97.51	0.2	97.59	0.2	**97.62**	0.2	97.49	0.1	97.52	0.1	**97.55**	0.1
2	chronic	**99.64**	0.4	99.46	0.3	99.63	0.4	**99.95**	0.1	99.88	0.1	**99.95**	0.1	99.87	0.2	**99.92**	0.1	99.88	0.1
3	column_2C	85.91	0.4	86.17	0.4	**86.20**	0.4	86.39	0.4	86.47	0.3	**86.92**	0.4	86.44	0.5	86.59	0.6	**86.72**	0.6
4	cryotherapy	92.55	1.6	**93.48**	1.8	93.46	1.0	95.04	1.5	95.56	1.2	**95.81**	1.3	95.19	1.5	97.24	1.2	**97.36**	0.7
5	diabetes	77.60	0.4	77.69	0.4	**77.78**	0.6	77.90	0.5	78.11	0.4	**78.12**	0.5	77.86	0.6	78.01	0.6	**78.05**	0.6
6	fertility	88.82	0.4	**88.87**	0.4	88.86	0.4	89.30	1.1	**90.14**	1.8	89.90	1.0	89.15	0.3	**89.52**	0.5	89.36	0.5
7	haberman	75.49	0.7	**75.83**	0.9	75.72	0.9	75.79	0.7	75.76	0.7	**76.38**	0.7	75.71	0.7	75.49	0.6	**76.05**	0.8
8	heart	84.23	0.7	84.25	0.7	**84.36**	0.5	85.27	0.7	85.59	0.5	**85.91**	0.6	85.04	0.6	85.43	0.8	**85.67**	0.7
9	immuno	82.10	1.3	**82.39**	1.2	**82.39**	1.1	87.08	1.2	**87.80**	1.2	87.48	1.2	85.13	0.9	86.13	0.8	**86.30**	0.5
10	liver	74.18	0.6	**74.67**	1.0	74.56	0.7	74.17	0.7	74.61	1.1	**75.09**	0.8	74.32	0.9	74.51	1.0	**75.17**	1.1
11	mammo	82.97	0.5	**83.07**	0.6	83.00	0.6	82.56	0.4	**83.02**	0.5	**83.02**	0.4	82.67	0.3	83.10	0.5	**83.33**	0.6
12	parkinsons	96.02	0.4	96.17	0.3	**96.23**	0.5	96.25	0.3	96.37	0.4	**96.52**	0.4	96.79	0.5	**96.89**	0.5	96.79	0.5
13	thoracic	85.16	0.2	85.15	0.1	**85.17**	0.2	85.25	0.3	85.33	0.3	**85.41**	0.4	85.33	0.5	**85.34**	0.4	85.33	0.4
14	transfusion	79.37	0.3	79.48	0.3	**79.58**	0.3	79.07	0.4	79.33	0.4	**79.43**	0.5	79.55	0.5	**79.78**	0.6	79.77	0.6
15	wpbc	82.31	0.7	82.46	0.7	**82.66**	0.7	81.54	1.2	81.66	1.0	**82.19**	1.5	81.71	0.7	82.11	0.9	**82.30**	1.0
Ranking by terminal set		3		2		1		3		2		1		3		2		1	
Best solutions		0		0		1		0.5		2		6.5		0		2		3	
Friedman Ranking		9		6		8		7		4		**1**		5		3		2	

TABLE VII
AVERAGE RANKS AND P-VALUES OF STATISTICAL TESTS ON THE NINE ALGORITHMS FOR KERNEL EVOLUTION

Test Algorithm	Friedman	Aligned Friedman	Quade
GP_clas	8.033	106.033	8.092
GP-U_clas	5.967	90.367	6.079
GP-B_clas	6.267	92.800	6.317
GP_mod	6.000	80.133	5.917
GP-U_mod	4.067	55.267	3.838
GP-B_mod	**2.400**	32.533	**2.471**
GP_all	5.833	78.033	5.829
GP-U_all	3.700	45.267	3.892
GP-B_all	2.733	**31.567**	2.567
p-value	**3.14E-09**	0.093	**8.41E-09**

Results in TABLE VII indicate that the nine algorithms are not statistically equivalent under the Friedman and Quade tests. In both cases, the SEEKS version GP-B_mod is suggested as the control method for post hoc tests. In TABLE VIII and TABLE IX the hypothesis of equal performance to the control method is rejected with a *p*-value smaller or equal to that reported in the last row. Rejections are highlighted in boldface. From TABLE VIII it can be concluded that GP-B_mod is statistically better in finding higher accuracies than the other methods, except for GP-U_all, GP-B_all, and GP-U_mod. Thus, indicating that the introduction of both the new set of kernel functions and the EDA mechanism helps to improve the performance of the previous GP-clas scheme.

TABLE VIII
POST HOC TEST FOR THE CONTROL METHOD (GP-B_mod) SUGGESTED BY THE FRIEDMAN TEST. ALGORITHMS ARE SORTED p-VALUE

Test Algorithm	Bonferroni – Dunn	Holm	Holland	Rom	Finner
GP_clas	**0.0000**	**0.0063**	**0.0064**	**0.0066**	**0.0064**
GP-B_clas	**0.0001**	**0.0071**	**0.0073**	**0.0075**	**0.0127**
GP_mod	**0.0003**	**0.0083**	**0.0085**	**0.0088**	**0.0191**
GP-U_clas	**0.0004**	**0.0100**	**0.0102**	**0.0105**	**0.0253**
GP_all	**0.0006**	**0.0125**	**0.0127**	**0.0131**	**0.0315**
GP-U_mod	0.0956	**0.0167**	**0.0170**	0.0167	**0.0377**
GP-U_all	0.1936	0.0250	0.0253	0.0250	0.0439
GP-B_all	0.7389	0.0500	0.0500	0.0500	0.0500
p-threshold	*0.0063*	*0.0167*	*0.0170*	*0.0131*	*0.0377*

TABLE IX
POST HOC TEST FOR THE CONTROL METHOD (GP-B_mod) SUGGESTED BY THE QUADE TEST. ALGORITHMS ARE SORTED p-VALUE

Test Algorithm	Bonferroni – Dunn	Holm	Holland	Rom	Finner
GP_clas	0.013	**0.0063**	**0.0064**	0.0066	**0.0064**
GP-B_clas	0.091	0.0071	0.0073	0.0075	0.0127
GP-U_clas	0.112	0.0083	0.0085	0.0088	0.0191
GP_mod	0.130	0.0100	0.0102	0.0105	0.0253
GP_all	0.140	0.0125	0.0127	0.0131	0.0315
GP-U_all	0.532	0.0167	0.0170	0.0167	0.0377
GP-U_mod	0.548	0.0250	0.0253	0.0250	0.0439
GP-B_all	0.966	0.0500	0.0500	0.0500	0.0500
p-threshold	0.0063	0.0063	0.0063	0.0064	0.0064

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2998749, IEEE Access

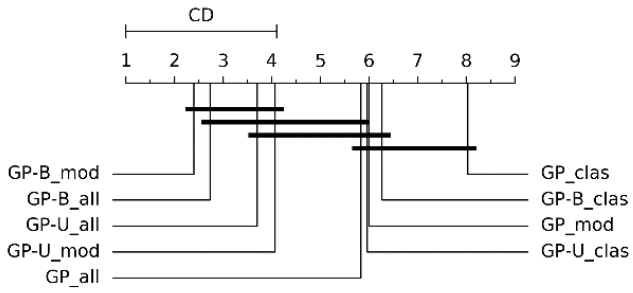

FIGURE 2. Comparison among methods for kernel evolution. Groups of methods that are not significantly different are connected. CD is the critical difference that determines the cutoff range of each group.

From TABLE IX it can be observed that the GP-B_mod method is not statistically better to the rest of algorithms, with exception to the state-of-the-art GP-clas. Since Quade test weights rank based on the difficulty of each dataset, its results indicate that the GP-B_mod is the only SEEKS version that can reach a statistically better performance than GP-clas for kernel evolution when solving biomedical classification problems.

The critical difference diagram, described in FIGURE 2, is based on the Friedman and post hoc Nemenyi tests [60], it helps to visualize and further understand the conclusions provided by the Friedman and Quade tests, since it presents how the kernel evolution methods are grouped.

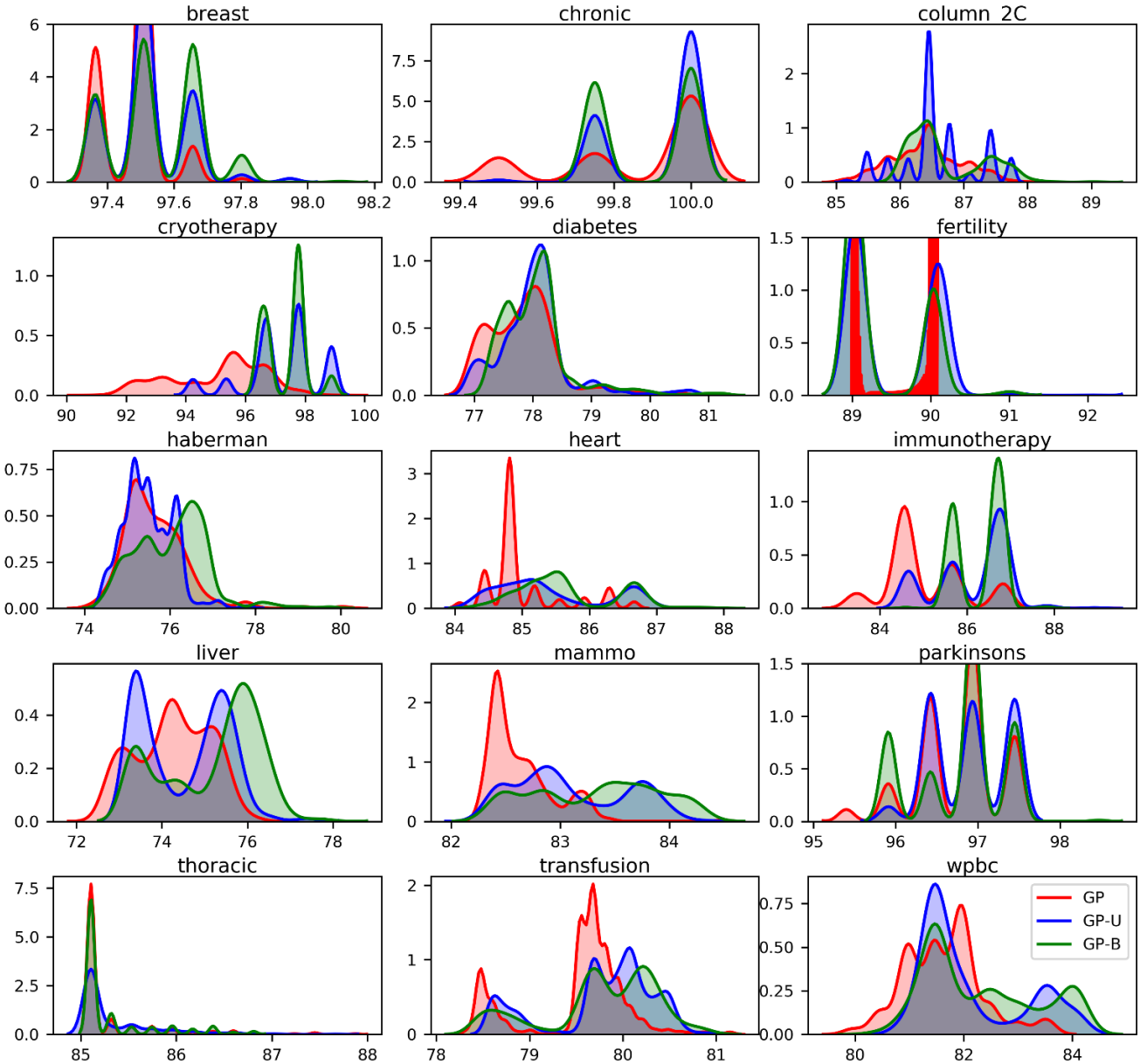

FIGURE 3. Density estimations based on the distribution of all SVMs with kernel trees evaluated during the last 5 generations (around 2500 SVMs minus duplicates for each dataset). The terminal set of all kernels was used. Densities illustrate convergence to the best performance on accuracy (x-axis). Higher peaks to the right indicate better kernels were found.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2998749, IEEE Access

The information presented in TABLES VI-IX is useful to determine improvements in performance; however, it is not enough to verify if there exists any gain in efficiency, and it is insufficient to understand why the three evolutionary methods (GP, GP-U, and GP-B) differ. To understand how the efficiency of the kernel evolution process was performed, a density estimation [61], based on the distribution of all kernel trees (without duplicates) evaluated during the last five generations of the SEEKS algorithm and the baseline GP, is presented for each dataset in FIGURE 3. For the three algorithms shown in this figure, the terminal set with all kernels was used. Due to space restrictions, Figures showing densities corresponding to classic and modern sets of kernels are not presented but are available in the *figures* folders in [47] [48].

As can be observed from FIGURE 3, most densities for the GP algorithm (red curves) present a large area to the left side, thus indicating that most of the evaluated kernels reached lower performance than those generated by GP-U (blue curves) or GP-B (green curves). An interesting case is the mammographic dataset, which obtained similar average accuracy on the nine algorithms reported in TABLE VI. For the mammographic dataset, it could be hard to determine which algorithm is preferable, but after revealing the notable skewness of the GP method, it is possible to justify that the SEEKS algorithm is a preferable option because it found a larger amount of kernels with higher accuracy rate. FIGURE 3 also helps to estimate how hard is to solve a given dataset. For instance, datasets like breast, chronic or cryotherapy are easily solvable, whereas diabetes, haberman or liver do not. Also, it is worth noting that although the three methods can find the kernel with higher accuracy, the GP reaches it due to randomness more than by a proper convergence.

B. EVALUATING DIFFERENT TERMINAL SETS

Regarding the second experiment, TABLE VI shows that, independently of the evolutionary mechanism employed, algorithms taking the modern or full set of kernels reach higher performance than those using the classic set. To ease the comparison of terminal sets, FIGURE 4 illustrates the max, average, and min performance over each dataset. In this figure it can be observed for which datasets, the terminal set of modern (blue dots) or all (green dots) kernels obtained better or equal performance than the classic (solid red line) set of kernels. Statistical tests of Friedman, Aligned Friedman, and Quade were applied to the results presented in TABLE VI. The results of these tests, summarized in TABLE X, indicate that there are differences among the three terminal sets for each one of the evolutionary algorithms. GP_all, GP-U_all, and GP-B_mod were suggested as control methods by the tests of Friedman, Aligned Friedman, and Quade, respectively. To determine which algorithms present statistical differences, the Bonferroni-Dunn post hoc test was applied to the control methods and their results are summarized in TABLE XI

TABLE X
RANKS AND *p*-VALUES OF STATISTICAL TESTS APPLIED TO THE ALGORITHMS FOR KERNEL EVOLUTION

Test	Friedman	Aligned Friedman	Quade
GP_clas	2.60	32.20	2.61
GP_mod	1.80	20.07	1.80
GP_all	**1.60**	**16.73**	**1.59**
p-value	**0.0149**	**0.0038**	**0.0274**
GP-U_clas	2.33	32.10	2.58
GP-U_mod	1.93	19.87	1.78
GP-U_all	1.73	**17.03**	**1.64**
p-value	0.2465	**0.0039**	**0.0448**
GP-B_clas	2.80	35.63	2.84
GP-B_mod	**1.53**	**16.57**	**1.49**
GP-B_all	1.67	16.80	1.67
p-value	**6.98E-04**	**0.0047**	**5.21E-04**

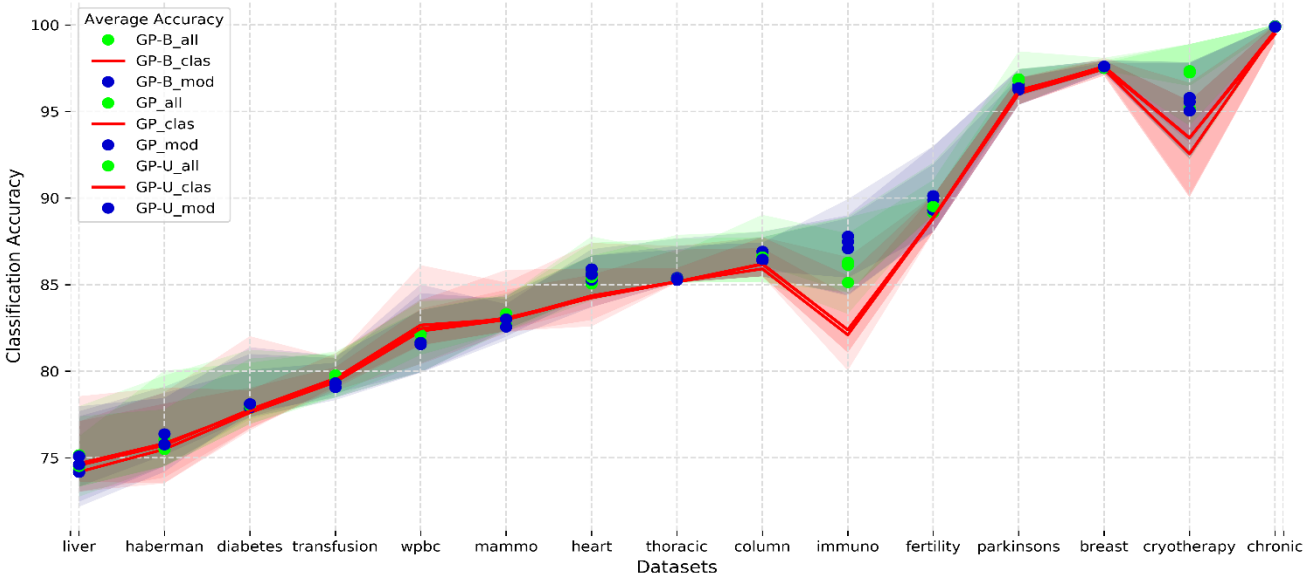

FIGURE 4. Performance of all algorithms sorted by average accuracy and grouped by terminal set. The red solid line represents the GP with random hyper-parameter search. Shadowed areas are limited with the max and min values. The same data that in Fig. 2 and Table VI were used

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2998749, IEEE Access

TABLE XI
BONFERRONI-DUNN *POST HOC* TESTS TO DETERMINE WHICH TERMINAL SETS OF KERNELS ARE STATISTICALLY DIFFERENT.

Algorithm/Test	Friedman	Aligned Friedman	Quade
GP_clas	**0.006**	**0.001**	**0.013**
GP_mod	0.584	0.487	0.608
p-treshold	**0.025**	**0.025**	**0.025**
GP-U_clas	-*	**0.0017**	**0.023**
GP-U_mod	-	0.554	0.725
p-treshold	-	**0.025**	**0.025**
GP-B_clas	**5.23E-04**	**7.02E-05**	**0.0011**
GP-B_all	0.715	0.961	0.673
p-treshold	**0.025**	**0.025**	**0.025**

* No difference was found; therefore, *post hoc* test was not applied. The GP_all, GP-U_all, and GP-B_mod were taken as the control methods

The conclusion from *post hoc* tests is that, with statistical significance, GP_all is better than GP_clas, but not than GP_mod; GP-U_all is better than GP-U_clas, but not than GP-U_mod, and GP-B_mod is better than GP-B_clas, but not than GP-B_all. Therefore, the introduction of the modern set of kernels (either as an independent terminal set or as in combination with the classic kernels) improves the performance of the evolutionary algorithms that use only the classic set of kernels functions.

C. SELECTING THE BEST EVOLVED KERNEL

For each of the 15 biomedical problems considered in this work, different evolved kernels were found to reach the highest performance in terms of accuracy. This is an important finding because it means that there are several single kernels or kernel combinations that can effectively solve a given dataset. At the same time, it raises the problem of determining which kernel should be selected since, due to the non-deterministic nature of the evolutionary algorithms, the final form of the best kernel found can be different from one run of the algorithm to another. Furthermore, as evolutionary algorithms are targeted to find the best possible accuracy without considering other aspects such as the PSV, the kernel complexity, the training time, the balance of the dataset, etc.; it is necessary to verify that the selected kernel is not overfitted. TABLE XII illustrates both cases, the existence of different kernels with equal performance and kernels that tend to overfit (highlighted in boldface). For the sake of brevity, only the information of the first three independent runs is provided, the rest of evolved kernels and all experimental results given in section V are available in the *results* folder of [47] or [48]. From TABLE XII it may be noted that overfitted kernels were found for the fertility and wpbc datasets, and in contrast, for the chronic dataset desirable kernels were obtained.

To alleviate the overfitting problem and being able to select the most convenient kernel for each dataset, we propose to consider the PSV as a complementary quality index, because it measures the amount of data required to build the decision function of SVMs. A PSV value equals to 1 indicates that all training data points were used, so the model is overfitted. Thus, kernels with similar accuracy and lower PSV should be preferred since they are expected to have better generalization on unseen cases. In order to conduct a systematic evaluation of SVMs by measuring the accuracy and PSV at the same time, the following *combined index*, previously reported in [27], is used. The results of applying this selection strategy are provided in TABLE XIII

$$cmbIdx = \exp\left(\frac{-0.1\,(100 \times PSV + (100 - Acc)^2)}{200 - 100 \times PSV - Acc + .001}\right) \quad (13)$$

TABLE XII
BEST KERNELS FOUND ON DIFFERENT RUNS OF THE EVOLUTIONARY METHODS REPORTED IN TABLE VI (HIGHLIGHTED IN GRAY). KERNEL EXPRESSIONS ARE FACTORED WHEN POSSIBLE

Dataset	Acc.	PSV	Kernels with highest Acc.	Run
breast	97.81	0.254	$W + K_{11}$	1
	97.95	0.230	$W + K_{11} + 2K_{13}$	2
	97.95	0.287	$W + K_{11}$	3
chronic	100	0.091	G	1
	100	0.089	H	2
	100	0.089	G	3
column_2C	88.06	0.470	H	1
	87.42	0.346	$G \times H$	2
	87.41	0.367	$K_{11} + G$	3
cryotherapy	97.78	0.230	$(H \times G) + L$	1
	96.72	0.369	$H + G$	2
	97.78	0.328	$(H \times G) + H$	3
diabetes	80.99	0.516	H	1
	80.46	0.551	H	2
	79.55	0.520	H	3
fertility	92.03	**0.945**	$(E \times K_{13}) \times (2 + K_{13})$	1
	91.03	0.364	G	2
	92.99	0.392	$G + K_{11}$	3
haberman	77.79	0.523	$2H^2 + E$	1
	78.11	0.522	$E + H$	2
	79.10	0.534	$E + G$	3
heart	86.67	0.498	G	1
	86.67	0.425	H	2
	86.67	0.487	G	3
immuno	88.82	0.417	H	1
	89.05	0.430	$3G$	2
	88.99	0.491	G	3
liver	77.97	0.636	$3H + 2G + L$	1
	73.91	0.745	$E + H$	2
	76.52	0.631	$H \times (2L + 2H)$	3
mammo	83.36	0.433	$L + H$	1
	83.15	0.418	$L \times L$	2
	83.98	0.391	G	3
parkinsons	96.93	0.461	$W + (K_{11} \times L)$	1
	97.44	0.501	$K_{11} \times (2L + 3R + P)$	2
	96.42	0.495	$R + W$	3
thoracic	87.66	0.394	H	1
	87.45	0.419	H	2
	87.23	0.415	H	3
transfusion	80.08	0.484	$R^2 \times H$	1
	81.02	0.476	$L \times R \times G$	2
	80.74	0.478	$L + L \times (H + R)^2$	3
wpbc	82.52	0.660	$L \times (R + L)^3 + L^2 \times (R + L)$	1
	83.52	**0.838**	$R \times (L + P) + L$	2
	84.03	0.628	$6R + L$	3

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2998749, IEEE Access

TABLE XIII
EVOLVED KERNEL SELECTION BY THE HIGHEST COMBINED INDEX OF ACCURACY AND PSV

dataset	Previous Works Acc.[REF]	Previous Works PSV[REF]	Acc.	PSV	100 × CmbIdx	G_mean	Time (ms)	Kernel	Hyper-parameters
breast	**98.03**[1]	---[1]	97.22	0.087	98.27	97.07	28	$G+G$	$C=23.06, \alpha=-0.041, n=1$
chronic	99.57[2]	0.135[2]	**99.75**	**0.079**	99.14	99.80	28	H	$C=08.29, n=3$
column_2C	86.02[2]	0.416[2]	**87.42**	**0.346**	78.09	90.53	70	$G \times H$	$C=31.15, \alpha=-0.057, n=3$
cryotherapy	91.00[3]	---[3]	**97.78**	0.197	97.05	98.97	6	$3G+L+H$	$C=16.45, \alpha=-0.065, n=2$
diabetes	**81.00**[1]	---[1]	80.99	0.517	54.17	74.71	56	H	$C=19.76, n=2$
fertility	89.19[2]	0.943[2]	**92.99**	**0.383**	88.07	56.16	16	$G+K_{11}$	$C=03.99, \alpha=0.217, n=6, \tau=4.759$
haberman	75.77[4]	---[4]	79.10	0.534	48.34	0[1]	78	$G+E$	$C=28.23, \alpha=0.482, n=2$
heart	84.67[2]	0.459[2]	**86.67**	**0.425**	73.27	86.31	7878	H	$C=13.29, n=6$
immuno	88.00[3]	---[3]	**89.06**	0.411	79.43	97.14	28	$3G$	$C=17.97, \alpha=0.031, n=5$
liver	74.15[2]	0.712[2]	**77.97**	**0.636**	39.07	0[1]	249	$3H+2G+L$	$C=25.38, \alpha=0.429, n=1$
mammo	**86.44**[4]	---[4]	83.97	0.392	68.03	91.65	846	G	$C=26.77, \alpha=-0.102, n=5$
parkinsons	95.98[2]	0.561[2]	**96.42**	**0.353**	93.21	94.02	8	$L+R$	$C=19.30, \gamma=1.403$
thoracic	85.15[2]	0.588[2]	**87.23**	**0.337**	77.97	20	11350	H	$C=04.43, n=6$
transfusion	**80.53**[4]	---[4]	80.22	0.431	56.76	49.26	322	$G^2 \times R \times L$	$C=25.93, \alpha=0.464, n=1, \gamma=2.234$
wpbc	81.22[4]	---[4]	**81.38**	0.420	60.22	67.04	14	L	$C=16.28$

[1] Genetic Programing for kernel Evolution and hyper-parameter tuning of SVMs with classic kernels, PSV was not reported [18]. [2]SVMs hyper-parameter tuning by BUMDA [27]. [3] Grid Search for hyper-parameter tuning of the RBF kernel function, PSV was not reported [50]. [4] The Particle Swarm Optimization (PSO) was applied to optimize the RBF kernel hyper-parameters, PSV was not reported [52]. [5] A zero value in the G-mean index in combination with high accuracy indicates that all test data were assigned to the majority class.

Other aspects of the evolved kernels such as the kernel complexity, the training time or the bias to an unbalanced class, can be evaluated based on the user's requirements. For instance, a third criterion can be suggested by following the Occam's razor principle, so that evolved kernels with similar accuracy and PSV, but with smaller tree-depth are selected since they are simpler to implement and cheaper to evaluate. The integration of these ideas into an evolutionary method like SEEKS will conduct to a multi-objective optimization scheme, which is beyond the scope of the present work. However, in order to provide a baseline for future comparisons of the SEEKS method, TABLE XIII presents the accuracy, PSV, cmbIdx, and G-mean indexes [62] and also the training time and associated hyper-parameters for the evolved kernels with the highest cmbIdx during the first three runs reported in TABLE XII. The same information that in TABLE XIII is available for all the evolved kernels and can be consulted in the *results* directory of [47] or [48]. The accuracy and PSV obtained by the base techniques of SEEKS and by other methods are also presented in TABLE XIII for reference.

VI. CONCLUSIONS

The main conclusion of the present work is that the proposed SEEKS method was able to automatically design an effective SVM classifier for each of the considered biomedical classification problems. Effectiveness was evaluated by comparing the performance of SVMs designed by our method against the SVMs produced by the state-of-the-art GP_clas algorithm. Experimental results indicate that the GP-U_mod version of SEEKS was statistically better than GP_clas, under the Friedman and Quade tests, and that GP_clas was inefficient and sensible to randomness. Results from FIGURE 2 allow us to conclude that the introduction of both EDAs and new kernel functions to the previous evolutionary scheme is beneficial. As per the results recorded in TABLES VI and X, The improvements in effectiveness were found to be caused by the introduction of new families of kernel functions.

Regarding efficiency, the SEEKS method found more kernels with the highest accuracy than GP for most datasets. As both methods used the same computational resources, initial population and terminal sets, the improvement in efficiency can only be explained by the introduction of the EDA mechanisms. From this result, it can be concluded that delegating the hyper-parameter tuning of evolved kernels to EDAs leads to an increase in the convergence of the GP evolutionary algorithms for SVMs.

In addition to the accuracy index, the quality of the best kernels found by the evolutionary algorithms was measure through the PSV, which resulted to be useful for detecting classifiers prone to overfitting. Also, the G-mean index, the kernel-tree depth, and other valuable information can be obtained from the evolutionary process. Thus, a major direction to future improvements of this work consists in reformulating the SVM kernel evolution as a multi-objective optimization problem, in order to reduce the PSV or the kernel complexity while maintaining the optimal accuracy or G-mean indexes.

It is important to mention that evolving kernels during 15 generations, with a maximum depth of 2 levels in the tree-based chromosome, was enough to find multiple kernels with high performance. Further these thresholds, kernels presented numerical problems and the evolution was time-consuming. This increment in computational time as a function of the kernel-tree depth is a limitation that may be overcome with the introduction of more efficient and manageable evolutionary algorithms, such as grammatical evolution. Genetic operators and GP hyper-parameters values were fixed to those reported

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2998749, IEEE Access

in TABLE IV, a deeper analysis on this configuration is required to further improve the SEEKS evolutionary process. Another restriction is the maximum size of the datasets (around 2000 instances) that can be handled by SEEKS. In order to deal with larger datasets, an SVM solver different to the LIBSVM may be implemented, as long as it allows the introduction of kernel functions. Due to the stochastic nature of SEEKS, different runs of the algorithm can produce different kernel-trees; thus, a systematic strategy to identify which kernel-tree is the best for a given dataset is required.

Finally, since the SEEKS strategy is independent of the problem domain, applications to domains other than classification (such as regression or density estimation) can be easily conducted in future work.

APPENDIX

A. TIME COMPLEXITY OF SEEKS

The SEEKS algorithm integrates two metaheuristics into a single mechanism, namely EDAs (LISTING 1) and GP (LISTING 2). The time complexity of both EDAs, UMDA and BUMDA, was reported for the SVM hyper-parameter optimization problem to be $O(pN^3)$ [27]. Where p and N stand for the population size and the number of training samples, respectively. The evaluation of the fitness function (training an SVM with the LIBSVM solver) is the most expensive step that dominates the processing time with a complexity of $O(N^3)$ [57]. The GP time complexity has not been reported by previous studies on kernel evolution [12, 18, 19]; hence, an analysis of its mechanism is now provided.

The GP time complexity can be bounded only after some assumptions are made on the genetic operators of selection, crossover, and mutation; and also after hyper-parameters such as the population size (p), maximum number of generations (G), and max tree depth (T_{max}) are fixed. Changing these genetic operators and/or hyper-parameters directly impacts on the GP processing time, e.g., the time complexity of selection by tournament is $O(p)$, and by roulette wheel with binary search is $O(p \log p)$ [63]. If one-point crossover and one-point mutation operators are used with constant probabilities $p_c > 0$, and $p_m > 0$, respectively; then, the overall complexity of the genetic reproduction step is $O(pnG)$, where $n < 2^{T_{max}+1}$ is the total number of nodes [64]. Regarding the GP hyper-parameters, the complexity of depth-first traversals of binary trees (such as the kernel tree used by SEEKS) is in $O(n)$; and the sorting algorithm required to rank the evolved SVMs is $O(p \log(p))$ [65]. Therefore, the bound of the GP time complexity is given by the sum of the following steps: evaluation of the initial population (A), ranking and selection (B), reproduction and mutation (C), population updating (D).

The SEEKS algorithm takes advantage of the similarities between the GP and EDA mechanisms, so that the more expensive steps (A and D) are used for updating the probability model of EDAs and the kernel-tree population of GP at the same time. The only computational cost that SEEKS adds to GP is on the computation required to obtain the parameters of either the BUMDA or the UMDA probabilistic model, which is linear $O(dp)$ with d the number of SVM hyper-parameters [42]. Let be G, T_{max}, and p, $p_c > 0$, $p_m > 0$ constant values and assume tournament selection, one-point crossover and one-point mutation are the genetic operators employed; the time complexities of SEEKS and its base techniques is bounded as reported in TABLE XIV.

TABLE XIV
TIME COMPLEXITY OF SEEKS AND ITS BASE TECHNIQUES

Algorithm	Time Complexity by steps: A+B+C+D
BUMDA	$O(p\,N^3) + O(p \log p) + 0 + G\left[O(p\,N^3) + O\left(p\,log(p)\right)\right]$
UMDA	$O(p\,N^3) + O(p \log p) + 0 + G\left[O(p\,N^3) + O\left(p\,log(p)\right)\right]$
GP	$O(p\,N^3) + O(p \log p) + O(pnG) + G\left[O(p\,N^3) + O\left(p\,log(p)\right)\right]$
SEEKS	$O(p\,N^3) + O(p \log p) + O(pnG) + G\left[O(p\,N^3) + O\left(p\,log(p)\right)\right] + O(dp)$

B. SUPPLEMENTARY MATERIAL

All datasets, files with performance indexes, evolved kernel trees, complementary figures, and codes are available at: *https://github.com/padiernacarlos/SEEKS*. Or at IEEE Dataport: *http://dx.doi.org/10.21227/32ab-9884*. For copyright reasons, the source code will be available once this paper is published.

ACKNOWLEDGMENT

The authors want to acknowledge the support provided by the *División de Ciencias e Ingenierías, Universidad de Guanajuato, Campus León* during the research and preparation of the manuscript. Luis Carlos Padierna also wants to thank to Dr. Arturo González-Vega for his valuable comments about experimental results.

REFERENCES

[1] W. M. Saltzman, Biomedical Engineering: Bridging Medicine and Technology, Cambridge: Cambridge University Press, 2009.

[2] S. Cogil and L. Wang, "Support vector machine model of developmental brain gene expression data for priorization of Autism risk gene candidates," *Bioinformatics*, vol. 32, no. 23, pp. 3611-3618, 2016.

[3] L. Naranjo, C. J. Pérez, J. Martín and Y. Campos-Roca, "A two-stage variable selection and classification approach for Parkinson's disease detection by using voice recording replications," *Computer Methods and Programs in Biomedicine*, vol. 142, pp. 147-156, 2017.

[4] F. A. Spanhol, L. S. Oliveira, C. Petitjean and H. Laurent, "A Dataset for Breast Cancer Histopathological Image

Classification," *IEEE Transactions on Biomedical Engineering*, vol. 63, no. 7, pp. 1455-1462, 2016.

[5] M. Adam, E. Y. Ng, S. L. Oh, M. L. Heng, Y. Hagiware, J. H. Tan, J. W. Tong and U. R. Acharya, "Automated characterization of diabetic foot using nonlinear features extracted from thermograms," *Infrared Physics & Technology*, vol. 89, pp. 325-337, 2018.

[6] I. Yoo, P. Alafaireet, M. Marinov, K. Pena-Hernandez, R. Gopidi, J.-F. Chang and L. Hua, "Data Mining in Healthcare and Biomedicine: A Survey of the Literature," *Journal of Medical Systems*, vol. 36, pp. 2431-2448, 2012.

[7] S. Maldonado and J. López, "Alternative second-order cone programming formulations for support vector classification," *Information Sciences*, vol. 268, pp. 328-341, 2014.

[8] Y. Xu, Z. Yang and X. Pan, "A Novel Twin Support-Vector Machine With Pinball Loss," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 2, pp. 359-370, 2017.

[9] V. Vapnik, Statistical Learning Theory, New York: John Wiley and Sons, 1998.

[10] N. Deng, Y. Tian and C. Zhang, Support Vector Machines, Boca Ratón: CRC Press, 2013.

[11] C. M. S. M. S. M. T. Gagné, "Genetic Programming for Kernel-based Learning with Co-evolving Subsets Selection.," in *Proceedings of Parallel Problem Solving*, vol. 4193, T. Runarsson, H. Beyer, E. Burke, J. Merelo-Guervós, L. Whitley and X. Yao, Eds., Reykjavik, Reykjavik, Springer Verlag, 4193 (4193), 2006, pp. 1008-1017.

[12] P. Koch, B. Bischl, O. Flasch, T. Bartz-Beielstein, C. Weihs and W. Konen, "Tuning and Evolution of Least-Squares Support Vector Machines.," *Evolutionary Intelligence*, pp. 1-30, 2011.

[13] A. Sousa, A. Lorena and M. Basgalupp, "GEEK: Grammatical Evolution for Automatically Evolving Kernel Functions," *Trustcom/BigDataSE/ICESS*, pp. 941-948, 2017.

[14] T. Howley and M. Madden, "The genetic kernel support vector machine: Description and evaluation," *Artificial Intelligence Review*, pp. 379-395, 2005.

[15] K. Sullivan and S. Luke, "Evolving kernels for support vector machine classification," in *Proceedings of the 9th annual Conference on Genetic and Evolutionary Computation*, London, 2007.

[16] A. Majid, A. Khan and A. M. Mirza, "Combination of support vector machines using genetic programming," *International Journal of Hybrid Intelligent Systems*, vol. 3, no. 2, pp. 109-125, 2006.

[17] A. Gijsberts, G. Metta and L. Rothkrantz, "Evolutionary optimization of least-squares support vector machines," in *Data Mining*, New York, Springer, 2010, pp. 277-297.

[18] L. Dioşan, A. Rogozan and J. Pecuchet, "Improving classification performance of support vector machine by genetically optimising kernel shape and hyper-parameters," *Applied Intelligence*, vol. 36 , no. 2, pp. 280-294, 2012.

[19] B. Zamani, A. Akbari and B. Nasersharif, "Evolutionary combination of kernels for nonlinear feature transformation," *Information Sciences*, pp. 95-107, 2014.

[20] R. Mantovani, A. Rossi, J. Vanschoren and B. d.-C. A. Bischl, "Effectiveness of Random Search in SVM hyper-parameter tuning," in *International Joint Conference on Neural Networks (IJCNN)*, 2015.

[21] T. Phienthrakul and B. Kijsirikul, "Evolutionary strategies for hyperparameters of support vector machines based on multi-scale radial basis function kernels," *Soft Computing*, vol. 14, pp. 681-699, 2010.

[22] M. Zhao, C. Fu, L. Ji, K. Tang and M. Zhou, "Feature selection and parameter optimization for support vector machines: A new approach based on genetic algorithm with feature chromosomes," *Expert Systems with Applications*, vol. 38, no. 5, pp. 5197-5204, 2011.

[23] S.-W. Lin, K.-C. Ying, S.-C. Chen and Z.-J. Lee, "Particle swarm optimization for parameter determination and feature selection of support vector machines," *Expert Systems with Applications*, vol. 35, no. 4, pp. 1817-1824, 2008.

[24] L. Shen, H. Chen, Yu, W. Kang, B. Zhang, H. Li, Y. Bo and D. Liu, "Evolving support vector machines using fruit fly optimization for medical data classification," *Knowledge-Based Systems*, vol. 96, no. 15, pp. 61-75, March 2016.

[25] A. Tharwat, A. E. Hassanien and B. E. Elnaghi, "A BA-based algorithm for parameter optimization of Support Vector Machine," *Pattern Recognition Letters*, October 2016.

[26] L. C. Padierna, C. Martin, A. Rojas, H. Puga, R. Baltazar and F. Héctor, "Hyper-Parameter Tuning for Support Vector Machines by Estimation of Distribution Algorithms," in *Nature-Inspired Design of Hybrid Intelligent Systems*, Vols. Studies in Computational Intelligence, 667, P. Melin, O. Castillo and J. Kacprzyk, Eds., Springer, 2017, pp. 787-800.

[27] A. Rojas-Domínguez, L. C. Padierna, J. M. Carpio, H. J. Puga and H. Fraire, "Optimal Hyper-parameter Tuning of SVM Classifiers with Application to Medical Diagnosis," *IEEE Access*, vol. 6, pp. 7164-7176, 2017.

[28] J. R. Koza, Genetic programming: on the programming of computers by means of natural selection, Massachusetts: MIT press, 1992.

[29] M. Hauschild and M. Pelikan, "An introduction and survey of estimation of distribution algorithms," *Swam and Evolutionary Computation*, vol. 1, pp. 111-128, 2011.

[30] J. Shawe-Taylor and N. Cristianini, Kernel Methods for Pattern Analysis, New York: Cambridge University Press, 2004.

[31] L. Zhang, W. Zhou and L. Jiao, "Wavelet Support Vector Machine," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 34, no. 1, pp. 34-39, 2004.

[32] A. D. Essam and T. Hamza, "New empirical nonparametric kernels for support vector machines classification," *Applied Soft Computing*, no. 13, pp. 1759-1765, 2013.

[33] Z. Pan, H. Chen and X. You, "Support vector machine with orthogonal Legendre kernel," in *International Conference on Wavelet Analysis and Pattern Recognition*, Xian, 2012.

[34] S. Ozer, C. Chen and H. Cirpan, "A set of new Chebyshev kernel functions for support vector machine pattern classification," *Pattern Recognition*, vol. 44, no. 7, pp. 1435-1447, 2011.

[35] V. H. Moghaddam and J. Hamidzadeh, "New Hermite orthogonal polynomial kernel and combined kernels in Support Vector Machine classifier," *Pattern Recognition*, vol. 60, pp. 921-935, 2016.

[36] L. C. Padierna, M. Carpio, A. Rojas-Domínguez, H. Puga and H. Fraire, "A novel formulation of orthogonal polynomial kernel functions for SVM classifiers: The Gegenbauer family," *Pattern Recognition*, vol. 84, pp. 211-225, 2018.

[37] J. Mercer, "Functions of Positive and Negative Type, and their Connection with the Theory of Integral Equations," *Philosophical transactions of the royal society of London. Series A*, pp. 415-446, 1909.

[38] N. Christianini and J. Shawe-Taylor, An Introduction to SVM and other Kernel Based Methods., Cambridge, U.K.: Cambridge University Press, 2000.

[39] M. Gönen and E. Alpaydin, "Multiple Kernel Learning Algorithms," *Journal of Machine Learning Research*, pp. 2211-2268, 2011.

[40] E. Talbi, Metaheuristics: from design to implementation, New Jersey: John Wiley., 2009.

[41] H. Mühlenbein, "The equation for response to selection and its use for prediction. Evol. Comput.," *Evolutionary Computation*, vol. 5, no. 3, pp. 303-346, 1997.

[42] S. I. Valdez, A. Hernández and S. Botello, "A Boltzmann based estimation of distribution algorithm," *Information Sciences*, vol. 236, pp. 126-137, 2013.

[43] D. Ashlock, Evolutionary Computation for Modeling and Optimization, New York: Springer, 2006.

[44] M. Tian and W. Wang, "Some sets of orthogonal polynomial kernel functions," *Applied Soft Computing*, vol. 61, pp. 742-756, 2017.

[45] S. Luke and L. Panait, "A Comparison of Bloat Control Methods for Genetic Programming," *Evolutionary Computation*, vol. 14, no. 3, pp. 309-344, 2006.

[46] D. Dua and C. Graff, *UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]*, Irvine, CA: University of California, School of Information and Computer Science, 2019.

[47] L. C. Padierna, "SEEKS Repository," University of Guanajuato, 29 12 2019. [Online]. Available: https://github.com/padiernacarlos/SEEKS. [Accessed 26 02 2020].

[48] L. C. Padierna, "Datasets, Results and Figures about Biomedical Classification Problems," IEEE Dataport, 15 March 2020. [Online]. Available: https://ieee-dataport.org/documents/datasets-results-and-figures-about-biomedical-classification-problems. [Accessed 2020 March 15].

[49] A. López, X. Li and W. Yu, "Support Vector Machine Classification for Large Datasets Using Decision Tree and Fisher Linear Discriminant," *Future Generation Computer Systems (36) 57-65*, vol. 36, pp. 57-65, 2014.

[50] A. Cüvitoglu and Z. Isik, "Evaluation Machine-Learning Approaches for Classification of Cryotherapy and Immunotherapy Datasets," *International Journal of Machine Learning and Computing*, vol. 8, no. 4, pp. 331-335, 2018.

[51] L. Sun, K.-A. Toh and Z. Lin, "A center sliding Bayesian binary classifier adopting orthogonal polynomials," *Pattern Recognition*, vol. 48, no. 6, pp. 2013-2028, 2015.

[52] H. l. Chen, B. Yang, S. j. Wang, G. Wang, D. y. Liu, H. z. Li and W. b. Liu, "Towards an optimal suppport vector machine classifier using a parallel particle swarm optimization strategy," *Applied Mathematics and Computation*, vol. 239, pp. 180-197, 2014.

[53] Y. F. Hernández-Julio, M. J. Prieto-Guevara, W. Nieto-Bernal, I. Meriño-Fuentes and A. Guerrero-Avendaño, "Framework for the Development of Data-Driven Mamdani-Type Fuzzy Clinical Decision Support Systems," *Diagnostics*, vol. 9, no. 2, p. 52, 2019.

[54] J. Zhao, Z. Yang and X. Yitian, "Nonparallel least square support vector machine for classification," *Applied Intelligence*, pp. 1-10, 2016.

[55] M. Li, X. Lu, X. Wang, S. Lu and N. Zhong, "Biomedical classification application and parameters optimization of mixed kernel SVM based on the information entropy particle swarm optimization," *Computer Assited Surgery*, vol. 21, no. 1, pp. 132-141, 2016.

[56] C.-C. Chang and L. Chih-Jen, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, 2011.

[57] I. Tsang, J. Kwok and P.-M. Cheung, "Core Vector Machines: Fast SVM Training on Very Large Data Sets," *Journal of Machine Learning Research*, pp. 363-392, 2005.

[58] S. García, A. Fernández, J. Luengo and F. Herrera, "Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power," *Information Sciences*, vol. 180, pp. 2044-2064, 2010.

[59] T. Eftimov, G. Petelin and P. Korosec, "DSCTool: A web-service-based framework for statistical comparison of stochastic optimization algorithms," *Applied Soft Computing Journal*, vol. 87, p. 105977, 2020.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2998749, IEEE Access

[60] J. Demsar, "Statistical Comparisons of Classifiers over Multiple Data Sets," *Journal of Machine Learning Research*, vol. 7, pp. 1-30, 2006.

[61] B. Silverman, Density Estimation for Statistics and Data Analysis, Boca Raton: Routledge, 2018.

[62] Y. Tang, Y.-Q. Zhang, N. Chawla and S. Krasser, "SVMs Modeling for Highly Imbalanced Classification," *IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics*, vol. 39, no. 1, pp. 281-288, 2009.

[63] D. E. Goldberg and D. Kalyanmoy, "A Comparative Analysis of Selection Schemes Used in Genetic Algorithms," in *Goundations of Genetic Algorithms*, San Mateo, California, Morgan Kaufmann Publishers, 1991, pp. 69-93.

[64] C.-W. Tsai, S.-P. Tseng, M.-C. Chiang, C.-S. Yang and T.-P. Hong, "A High-Performance Genetic Algorithm: Using Traveling Salesman Problem as a Case," *The Scientific World Journal*, vol. 2014, pp. 1-14, 2014.

[65] S. S. Skiena, The Algorithm Design Manual, Second Edition ed., New York: Springer, 2008.

LUIS CARLOS PADIERNA was born in Dolores Hidalgo, Guanajuato, México, in 1985. He received the B.Eng. degree in computer systems engineering from the Celaya Institute of Technology in 2009; the M.Sc. degree in computer science in 2011 from the León Institute of Technology and the Ph.D. in computer science from the Tijuana Institute of Technology in 2018.

Dr. Padierna is now an associate professor at the University of Guanajuato and has authored research works on the designing of support vector machines. His main research interests are: machine learning, pattern recognition, evolutionary algorithms, and deep learning.

CARLOS VILLASEÑOR-MORA was born in Acuitzio del Canje, Michoacán, Mëxico, in 1979. He received the B.Eng. degree in electronics engineering from the Morelia Institute of Technology in 2001; the M.Sc. in electronics in 2004 from the Celaya Institute of Technology and the Ph.D. in optical science from the Optical Research Center (CIO) in 2009. He has authored research works on infrared, biomedical and optics applications. His main research interests are on non-invasive biomedical techniques for diagnostics and infrared thermal applications such as diabetic retinopathy and diabetic foot.

SILVIA ALEJANDRA LOPEZ JUAREZ was born in Irapuato, Guanajuato, Mexico in 1980. She received the Bachelor degree in Chemical Pharmaceutical Biology from the Universidad de Guanajuato in 2004; M.Sc. in Neurobiology in 2006 from Universidad Nacional Autonoma de México and the PhD degree from Museum Nationale d'Histoire Naturelle a Paris in 2011.She has authored research Works on Neurobiology and Cellular therapy by using Cell Imaging Analysis. She is particularly interested to apply computational intelligence to analyze cellular markers in both neurodegenerative (Parkinson's disease) and neuroregenerative (adult neurogenesis) cellular processes.