PAPER

# A Family-Based Evolutional Approach for Kernel Tree Selection in SVMs

**Ithipan METHASATE**[†a)], ***Student Member and*** **Thanaruk THEERAMUNKONG**[†], ***Member***

**SUMMARY** Finding a kernel mapping function for support vector machines (SVMs) is a key step towards construction of a high-performanced SVM-based classifier. While some recent methods exploited an evolutional approach to construct a suitable multifunction kernel, most of them searched randomly and diversely. In this paper, the concept of a family of identical-structured kernel trees is proposed to enable exploration of structure space using genetic programming whereas to pursue investigation of parameter space on a certain tree using evolution strategy. To control balance between structure and parameter search towards an optimal kernel, simulated annealing is introduced. By experiments on a number of benchmark datasets in the UCI and text classification collection, the proposed method is shown to be able to find a better optimal solution than other search methods, including grid search and gradient search.

***key words:*** *support vector machines, multifunction kernel, kernel tree, optimal search, simulated annealing*

## 1. Introduction

Recently kernel-based learning has been recognized as a powerful technique to solve nonlinearity in several supervised, unsupervised and semi-supervised tasks, including classification, regression, clustering, and eigenvector analysis. Some popular kernel-based methods are Support Vector Machines (SVM) [1], [2], Kernel Fisher Discriminant (KFD) [3], Semi-Supervised Kernel Machine (SSKM) [4], and Kernel Principle Analysis (KPCA) [5]. For classification tasks, an SVM is a well-known kernel-based method that is successfully applied in many applications such as text categorization [6], information retrieval [7], DNA detection [8], intrusion detection [9] and image/speech recognition [10], [11]. Being a binary classifier, an SVM utilizes support vectors to maximize both classification accuracy and margin between the classes. Like other kernel-based methods, an SVM requires a kernel mapping function to transform original data in a linear space to a non-linear high-dimensional space by means of dot product between a pair of mapped data points, known as kernel trick.

However, a common problem in SVM-based classification is how to select a suitable kernel function and its hyperparameters (kernel parameters and SVM tradeoff parameter) that lead to satisfactory performance. While it is possible to perform trial-and-error in choosing a kernel function blindly, several works [12]–[14] applied numerical analysis techniques to search for an optimal solution. As an early work based on numerical analysis, Chapelle and Vapnik [12] presented a gradient descent algorithm, that guides searching process to an optimal solution by adjusting kernel parameters under consideration of their gradients in order to improve time complexity of searching and also still maintain classification accuracy. Their work introduced either a span bound or a radius-margin bound as the objective function for optimization, instead of empirical errors, and showed a general framework of optimization for any kernel type. Later, with inspiration of Chapelle's work, Keerthi [14] pointed out an analytical exploration on several different kernel functions with implementation of radius-margin bound using the gradient descent approach. In contrast with the previous methods which utilizes L2 SVMs, Chung et al. [13] originally proposed a method to tune kernel parameters using L1 SVM with the BFGS Quasi-Newton method. While most of the mentioned works used the Radius Basis Function (RBF) kernel, some recent works have applied a gradient-based method on other kernel functions. As a more complicated approach, Glasmachers and Igel [15] extended the gradient method to a general form of Gaussian kernels.

To overcome the limitation of local optimal problem as well as differentiability and smoothness of the objective function used in the gradient method, more recent works have exploited on the evolutional method. In 2005, Friedrichs and Igel [16] initiated a so-called covariance matrix adaptation evolution strategy (CMA-ES) to extend the RBF kernel with scaling and rotation in order to realize invariance against linear transformation in the space of SVMs parameters. In [17], Howley and Madden proposed a new approach to use a so-called genetic kernel, which is represented by a tree each leaf node of which expresses a feature vector (i.e., any data point or instance). Although this work sounded the first attempt to apply genetic programming to select an optimal kernel in SVM, it is not formulated well and the experiments were limited. Moreover, unfortunately in some situations, their genetic kernel may not satisfy the Mercer's condition. Recently, there have been a series of works on a more flexible kernel so-called multiple kernel learning (MKL). Originally Lanckriet et al. [18], [19] proposed the framework of a linear combination of multiple kernels that resulted in a quadratically-constrained quadratic program (QCQP). Later, Sonnenberg et al. [20] reformulated the MKL problem as a semi-infinite linear program (SILP) and applied it on large-scale data. These MKL-based approaches were designed to find a set of weights in a fixed

Manuscript received July 13, 2009.
Manuscript revised November 24, 2009.

[†]The authors are with Information and Computer Technology School, Sirindhorn International Institute of Technology, Thammasat University, Thailand.
a) E-mail: ithipan.methasate@nectec.or.th

DOI: 10.1587/transinf.E93.D.909

form of a multi-RBF kernel using numerical analysis. As an alternative, an evolutional method on multi-RBF kernels using evolutionary strategy was proposed by Phienthrakul and Kijsirikul [21]. In their experiments, the more RBF terms are added in a multi-RBF kernel, the higher accuracy the method can achieve.

More recently, Sullivan and Luke's work [22] applied genetic programming to search an optimal function in a more general form, where operators and kernels can be arbitrary under the Mercer's condition. In the same year, Methasate and Theeramunkong [23], [24] combined genetic programming and gradient search to find an optimal kernel function. These works originally introduced the concept of a so-called kernel tree to systematically represent a kernel in the form of a tree under the Mercer's conditions. Compared with gradient search and grid search, the method was shown to improve classification accuracy on various datasets in the UCI repository [25] and text classification collection. A hybrid of genetic programming and evolutionary strategy was presented in [26] to find an optimal for both kernel structure and kernel parameters. However, the existing methods have explored both structure space and parameter space in a random manner.

Towards this problem, we propose an adaptive control mechanism based on simulated annealing, into the evolutional method to find an optimal kernel more efficiently. The concept of a family of identical-structured kernel trees is introduced to explore structure space (a global space) using genetic programming and investigate parameter space (a local subspace) on a specific tree. In the rest of this paper, Sect. 2 describes the kernel selection problem in SVM-based classification. Section 3 presents the definition of kernel trees and kernel tree family. In Sect. 4, global search and local search are illustrated. Then the control mechanism between these two search methods using simulated annealing are proposed. In Sect. 5, experimental results using various datasets in the UCI repository and the text classification collection and the diversity measurement are displayed. Finally, the conclusion is made in Sect. 6.

## 2. Kernel Selection for Support Vector Machines

In the SVM-based classification, the performance of a classifier is mainly affected by kernel function and its parameters. The rest of this section describes how to select a single-kernel function and a multi-kernel function for SVMs.

### 2.1 Single-Kernel Selection

The naive method for kernel selection in SVMs is to choose one standard kernel function and then do trial-and-error on parameter values in the kernel function. For each individual kernel function, the set of parameters that yields the highest accuracy is selected. Table 1 illustrates the four commonly used kernel functions and their parameter sets. In addition to the parameters in the functions, a tradeoff (usually, denoted by C) is another factor on determination of classification accuracy when some errors are allowed for soft margin in an SVM. Therefore the total numbers of parameters for linear, RBF, polynomial and sigmoid kernels are 1, 2, 3 and 3, respectively. As a simple approach, grid search can be applied by fully enumerating all combinations of parameter values. This method does not work well with a large number of parameters. In this work, the number of parameters in these four common kernels is not large, i.e. at most 3 parameters. Besides the grid search, gradient search can find the optimal solution more efficiently with gradient information. However, as a drawback the gradient search may lead to a local optima, instead of global one, if the initialization is not good enough. In contrast with gradient search, an evolutional method can be applied to avoid trapping into this local optimum.

**Table 1** Some basis kernel functions $K(x, y)$.

| Kernel Name | Function | #Parameters |
|---|---|---|
| linear | $(x \cdot y)$ | 0 |
| RBF | $\exp\left(\frac{-\|\|x-y\|\|^2}{c}\right)$ | 1 |
| polynomial | $((x \cdot y) + \theta)^d$ | 2 |
| sigmoid | $\tanh(\kappa(x \cdot y) + \theta)$ | 2 |

### 2.2 Multi-Kernel Selection

A natural extension of a single-kernel function is to allow several basic kernels to form a kernel function. For this point, the well-known Mercer's theory [27] was introduced to create a complex kernel function, that has positive-definite property, from several kernel functions, called multi-kernel function. Intuitively the multi-kernel selection problem is much more difficult than the single-kernel selection problem since, in addition to parameters, the form of the function can be varied with some operators such as addition, multiplication and exponential operator. This extension gives us a chance to get a more suitable mapping kernel for a classification problem. To deal with multiple kernel tuning, a method namely multiple kernel learning (MKL) was proposed [18], [19] to find an optimal weighting on a fixed superposition form of a single RBF. For more flexibility, genetic programming can be applied to form a more general multi-kernel function. With this technique, the kernel can be constructed freely over the Mercer's criteria. However, some control mechanisms are needed for tuning in multi-kernel learning.

## 3. A Kernel Tree as a Multi-Kernel in SVM

In this section, the definitions of a kernel tree and a tree family are given. A kernel tree enables the representation of a multi-kernel function in the form of a tree structure. Originally proposed in [24], a kernel tree was shown to satisfy the Mercer's theory [27] which is a representation of a symmetric positive-definite function on a square as a sum of a convergent sequence of product functions. In this work, we introduce the concept of a kernel tree family as a group of

structurally identical trees (or isostructural trees). Any two trees are considered as structurally identical when they have the same tree structure with the same node type for each node position but may not occupy the same parameter values for any corresponding edge and any corresponding leaf node. Moreover, we also propose a mechanism to control the population generation in generic programming, based on this kernel tree family. In the rest, the formal definition of kernel trees is given and then a family of kernel tree is explained.

### 3.1 Definition of Kernel Trees

A kernel function $K(x_i, y_j)$ have been used to implement a kernel trick which is a method to exploit a linear classifier to solve a non-linear problem by mapping the original non-linear observations into a higher-dimensional space by a dot product between two points $x_i$, $y_j$, as follow.

$$K(x_i, y_j) = K(x_i)K(y_j) \tag{1}$$

**Definition 1** (A kernel tree): A kernel tree $T(\cdot)$ encodes a kernel function $K(\cdot)$ by a tree structure $T = (N_I, N_L, E)$, where an intermediate node $n_i \in N_I$ represents an operator, and a leaf node $n_l \in N_L$ specifies a single kernel function and an edge $e \in E$ expresses a link between node $n_i$ and node $n_j$ with a coefficient.

In other words, a kernel tree is characterized by the way to assign an operator, a coefficient and a kernel function at each intermediate node, each edge and each leaf node, respectively, in a kernel tree. In addition to these three components, the number of branches, called branching factor, for each intermediate node is another factor in determining the property of the kernel tree. In this work, an operator in consideration is one of two types: addition and multiplication, a coefficient is a real number, a kernel function is one of the three types: linear, RBF, polynomial function and branching factor is set to be two. Figure 1 displays a kernel tree $T\left(x_i, x_j\right)$ of a kernel function with four sub-kernels, $K\left(x_i, x_j\right) = \alpha_1\left(\alpha_3 K_1\left(x_i, x_j\right) + \alpha_4 K_2\left(x_i, x_j\right)\right) + \alpha_2\left(K_3\left(x_i, x_j\right)^{\beta_1} K_4\left(x_i, x_j\right)^{\beta_2}\right)$. Here, $\alpha_1$, $\alpha_2$, $\alpha_3$ and $\alpha_4$ are coefficients and $\beta_1$, $\beta_2$ are exponential, and $K_1$, $K_2$, $K_3$ and $K_4$ are basic kernel functions.

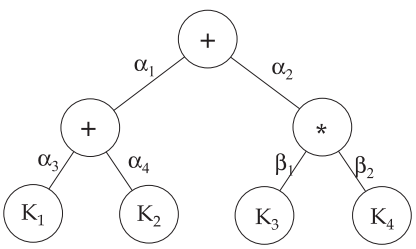



**Fig. 1** A kernel tree $T\left(x_i, x_j\right)$ of a kernel function with four sub-kernels $K\left(x_i, x_j\right) = \alpha_1\left(\alpha_3 K_1\left(x_i, x_j\right) + \alpha_4 K_2\left(x_i, x_j\right)\right) + \alpha_2\left(K_3\left(x_i, x_j\right)^{\beta_1} K_4\left(x_i, x_j\right)^{\beta_2}\right)$.

### 3.2 A Family of Kernel Trees

In this section, first we introduce a normalized structure of a kernel tree and isostructural kernel trees. Thereafter, we describe a family of kernel trees based on these two concepts.

**Definition 2** (Normalized Structure of a Kernel Tree): The normalized structure of a kernel tree is a tree the branches of which are reordered according to their weights calculated from the substructures under these branches, in order to preserve structural regularity. The structural regularity includes the order among leaf node types, the order among intermediate node types, and the order between leaf node types and intermediate node types.

Technically, since a branch in any tree will have a unique corresponding node (the node under the branch), a weight given to a branch implies the weight given to its corresponding node. In this work, the weight of a node $n$, denoted by $V(n)$, is calculated with the following formula. This formula enables the preservation of structural regularity.

$$V(n) = W(n)(B \cdot M)^d + \sum_{c \in N_c} V(c) \tag{2}$$

where $B$ is the branching factor in the tree, $d$ is the depth level of the node $n$, and $N_c$ is the set of the child nodes of the node $n$, $W(n)$ is an index representing each type of the node $n$ and $M$ is the maximum index value. In this task, two different indexing systems ($W(n)$) will be set for leaf nodes and intermediate nodes. Given $T1$ leaf node types and $T2$ intermediate node types, leaf node types will be assigned the values of 1, $(B+1)$, $(B^2+B+1)$,..., and $(B^{T1}+B^{T1-1}+\ldots+1)$ in order, while intermediate node types will be given the values of 1, $(B+1)$, $(B^2+B+1)$,..., and $(B^{T2}+B^{T2-1}+\ldots+1)$ in order. That is, the first node has the value of 1, the second node type takes the value of $(B+1)$, and so on. $M$ is the maximum value from two indexing systems, i.e. $M = MAX(B^{T1}+B^{T1-1}+\ldots+1, B^{T2}+B^{T2-1}+\ldots+1)$.

Figure 2 illustrates a kernel tree, its normalized structure and the weight of each node in the tree. In this figure, $B$ is set to 2 (i.e, a binary tree). The index ($W(n)$) equals to 1, 3 and 7 when the node $n$ expresses a linear, RBF and polynomial function respectively while it is set to 1 and 3 for an addition operator and a multiplication operator, respectively. Moreover, $M$ is 7 (i.e., $MAX(7, 3)$).

To clarify the calculation, some weighting examples are given as follows. The node $n_2$ at the depth of 0 has a weight of 3 ($V(n_2) = 3$) since this node represents a RBF ($W(n_2) = 3$). The node $n_7$ is given a weight of 52 since it has two child nodes with the weights of 7 and 3, and it is located at the depth of 1 with the operator '$*$' ($W(n_7) = 3$). Therefore, $V(n_7)$ equals to $(3 \times (2 \times 7)^1) + (7 + 3) = 52$. The node $n_{12}$ is given a weight of 221 since its depth is 2, its operator is '+' ($W(n_{12}) = 1$), and its left and right nodes get the weights of 22 and 3, respectively. Thus, $V(n_{12})$ obtains the value of $(1 \times (2 \times 7)^2) + (22 + 3) = 221$.

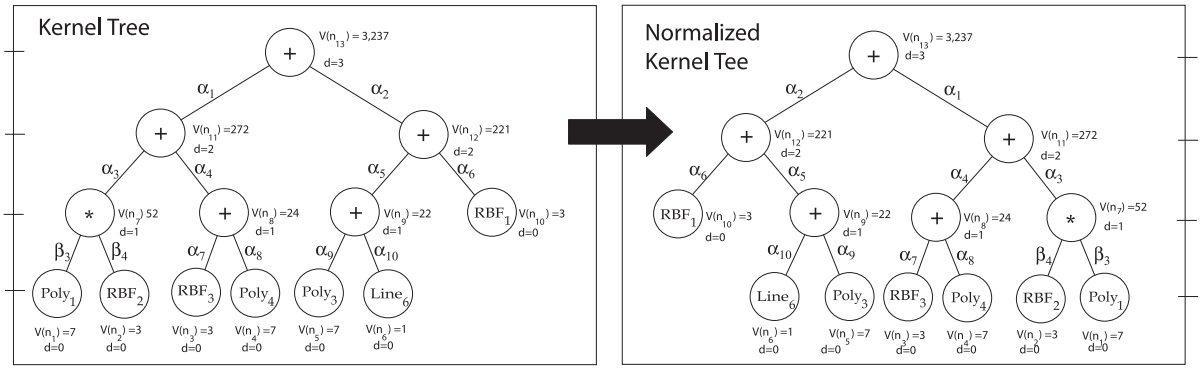



**Fig. 2** Normalized structure of a kernel tree.

Note that the child nodes of each node in the normalized tree will be reordered in the manner that a child node with a lower weight will be located on the left of a child node with a higher weight. Any child nodes with a same weight will have the same structure. Therefore, there is no need to swap their order.

### 3.2.1 Isostructural Kernel Trees

**Definition 3** (Isostructural Kernel Trees): Two kernel trees, $T_1$ and $T_2$, are isostructural if and only if the normalized structure of $T_1$ has the identical structure with the normalized form of $T_2$.

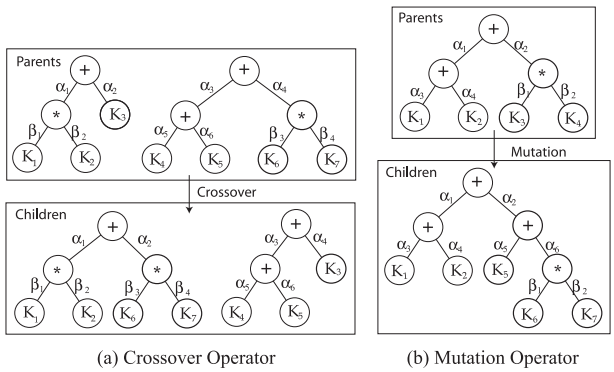
It is noted that two isostructural kernel trees will have an identical structure but may have different parameter values at the edges and leaf nodes. Moreover, the weight value of the top nodes of any two identical structures are always equal according to Eq. (2). Therefore, it is a trivial task to find identical structures.

**Definition 4** (A Family of Kernel Trees): A family of kernel trees is defined as a set of isostructural kernel trees.

A family of kernel trees consists of the normalized trees which have the same weight value. The computational cost for finding the families of kernel trees mostly come from the tree normalization step. Anyway it is obvious that the calculation cost in this step is relatively small compared to SVMs calculation. The next section illustrates how to apply the concept of a family of kernel trees to control local and global searching. Later for simplicity, we will call a family of kernel tree as a tree family.

## 4. Searching for an Optimal Kernel Tree

In this section, we propose an adaptive control mechanism, where simulated annealing is incorporated into the evolutional-based method to find an optimal kernel efficiently. This approach involves two scales of searching; global search on structure space and local search on parameter space. As a modification of genetic programming (GP), a structure space is explored as global search while a parameter space is investigated as local search. With the concept of




(a) Crossover Operator (b) Mutation Operator

**Fig. 3** GP operators on kernel trees: crossover operator and mutation operator.

a tree family stated in the previous section, a structure space is formulated from varied tree families while a parameter space can be defined among kernel trees existing in a tree family. Although it is possible to apply these two searching methods blindly, a suitable control may make global search and local search efficiently cooperate with each other to gain a higher accuracy for a specific problem. For this purpose, simulated annealing is introduced to balance structure search and parameter search under the concept of a tree family, i.e., a family of kernel trees. In the rest, we describe, in turn, structure search as global search, parameter search as local search, and simulated annealing as search control.

### 4.1 Structure Evolution as Global Search

In our work, genetic programming (GP) is applied to search for optimal structure of a kernel function as global search. Like the conventional GP, two common operators, namely crossover and mutation, are used to generate new trees as a part of searching process on the structure space as shown in Fig. 3. The crossover operator selects two parents as targets for swapping their substructures while the mutation operator replaces a substructure of the targeted kernel tree with a new substructure. When crossover and mutation are applied, the structure of the newly constructed kernel tree is controlled to be different from its parents. Same as done in the conventional method, a crossover operator selects two

parents (kernel trees), and randomly exchanges a part of a selected tree with a part of the other. On the other hand, a mutation operator chooses a parent and then replaces it with a randomly created structure. As described in Sect. 3, all trees need to be in the normalized form. Therefore, after the crossover and mutation process, we need to transform the result tree into a normalized tree. For crossover, the nodes that we exchanged between two trees and their upper nodes need to be recalculated the weight and rearranged to the normalized form. Similar to crossover, we need to rearrange the mutated trees from the nodes in their mutated part and their upper nodes in the tree.

### 4.2 Parameter Evolution as Local Search

In the past, evolution strategy (ES) was applied to tune hyperparameters among several SVMs with the fixed form of the kernel function [16], [21]. The task was to find suitable settings of parameters in such function. As an modification to this conventional usage of ES, our method locally searches for a set of suitable parameter values of kernel trees in each tree family selected by GP in the stage of global search. In our method, a parameter vector is constructed for a kernel tree, with each element corresponding to a weight (parameter) at a branch in the kernel tree or a value of a parameter in the kernel function, which is located at the leaf node of the kernel tree. Moreover, in this stage, the vectors formulated from the isostructural kernel trees will be gathered together. The mean and the variance of each element of the vectors are calculated and then used for producing a set of vectors which represent the kernel trees in the next generation.

Suppose that a kernel tree $T_i$ is a member of a tree family $F$ (i.e., $T_i \in F$), and the parameters of $T_i$ is denoted by a vector of $v_{i,j}$, where $j \in [1, n]$ is the ordinal index of the $j$-th parameter, and $n$ is the number of parameters in $T_i$. Note that all trees in the same family have the same number of parameters. The mean $\mu_j$ and variance $\sigma_j^2$ are calculated from the values of the $j$-th parameter in the tree family as follows.

$$\mu_j = \frac{1}{|F|} \sum_{i=1}^{|F|} v_{i,j} \tag{3}$$

$$\sigma_j^2 = \frac{1}{|F|} \sum_{i=1}^{|F|} \left(v_{i,j} - \mu_j\right)^2 \tag{4}$$

The family mutation is a process to generate a new kernel tree from a set of isostructural kernel trees. Here, let the $j$-th element of the newly generated kernel tree denoted by $v'_j$. Its value can be derived by

$$v'_j = \mu_j + \sigma_j \cdot N(0, 1) \tag{5}$$

where $N(0, 1)$ is a normally distributed random scalar.

Figure 4 displays our framework structure search and parameter search are combined to perform two different levels of search. As the first step, an initial population is generated by using a common random technique, such as Ramped

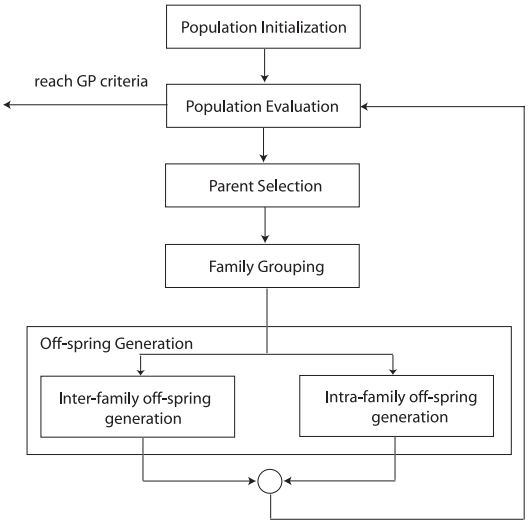



**Fig. 4** Family-based genetic programming.

Half-n-Half method [28]. Second, the population is evaluated by averaging accuracy of a cross validation. Third, the n-best parents are selected as seeds for generating the next population. Among the selected parents, isostructural trees are grouped together to form a set of tree families in the fourth. At the fifth step, trees in the next population are produced by two offspring generation methods; inter-family and intra-family method. Here, the inter-family method (i.e., global search) applies GP to produce a new tree by exchanging subtrees of any two trees or mutating the tree while the intra-family method (i.e., local search) utilizes parameter search to generate a new tree (with the preserved structure) by considering the mean and variance of parameter values among trees in the same family. Iteratively the offsprings will be evaluated and selected as parents for generating the next population. The repetition is terminated when the number of generations reach a predefined threshold.

### 4.3 Controlled Search Using Simulated Annealing

In this work, the concept of simulated annealing is introduced to control offspring generation by inter-family and intra-family methods. As a well-known technique towards better approximation of global optimal, the simulated annealing analogizes the process of increasing the size of metal crystal, to the process of finding an optimal solution in large search space. With this technique, a temperature is analogous to the ratio of offspring generated by the inter-family method to those produced by the intra-family method. Following the traditional annealing approach, our proposed method reduces the temperature gradually from high to low. With a higher temperature, we randomly wander through states with more dynamic exploration of global search space. In contrast, a lower temperature enables us to find better configurations with more stable exploration of local search space.

Algorithm 1 illustrates our proposed genetic program-

**Algorithm 1**: Genetic Programming with Simulated Annealing (GPSA)

```
begin
    MnT = τ1; MxT = τ2;
    TempIRate = α; TempDRate = β;
    AccThres = μ; CountThres = ψ;
    T = MxT; Count = 0;
    OldAccIntra = 0; Gen = 0; GenMax = σ;
    POP = Initiate();
    foreach Gen < GenMax do
        POPACC = Evaluation(POP);
        PARENT = Select(POPACC);
        FAM = Group(PARENT);
        INTEROFF = GenInterFam(T,FAM);
        INTRAOFF = GenIntraFam(MxT-T,FAM);
        AccIntra = EvaluationIntra(INTRAOFF);
        DiffAccIntra = AccIntra - OldAccIntra;
        if DiffAccIntra < AccThres then
            Count = Count +1;
        else
            Count = 0;
        if Count > CountThres then
            T = MIN(MxT,T+TempIRate) ;
            Count = 0;
        else
            T = MAX(MnT,T−TempDRate) ;
        Gen = Gen + 1;
        OldAccIntra = AccIntra;
        POP = INTEROFF ∪ INTRAOFF ;
end
```

ming with simulated annealing (GPSA) to control the different offspring generation in genetic programming. In the algorithm, $T$ expresses the temperature for managing the ratio of offspring generation between inter-family and intra-family methods. The temperature $T$ is controlled to be a value between minimum temerature (*MnT*) and maximum temperature (*MxT*), with decrement of *TempDRate*. *OldAccIntra* and *AccIntra* are the accuracy that is gained from the intra-family method in the previous generation and that in the current generation, respectively. *DiffAccIntra* is the difference between *OldAccIntra* and *AccIntra*. *Gen* is the index of the current generation and *GenMax* is the maximum number of generations. The **Initiate** function randomly generates a population (POP). The **Evaluation** function evaluates each element in the population (POP) and then returns the element with its accuracy attached (POPACC). The **Select** function selects out a number of best elements from the prioritized population (POPACC) as the parents (PARENT) for the next generation. The **Group** function clusters the parents (PARENT) into a set of groups (FAM) based on their structures. The trees in a group are isostructural. The **GenInterFam** function generates offsprings among families (INTEROFF) while the **GenIntraFam** function creates offspring within a family (INTRAOFF). The **EvaluationIntra** function returns accuracy performance (AccIntra) of the offsprings produced within a family. At the last line in the algorithm, the next generation population (POP) is formed by the union of both types of the offsprings. Moreover, in this work, the **Evaluation** and **EvaluationIntra** are performed to find the accuracy of the ten-fold cross-validation.

Unlike conventional simulated annealing, our proposed method allows *Temp* to be increased with a step of *TempIRate* if the intra-offsprings do not give a sufficient improvement on average accuracy, say *AccThres*, for a certain number of generations (*CountThres*). Moreover, the loop of GPSA terminates when the number of iterations reaches the preset maximum (*GenMax*). With this mechanism, it is possible to search for a new structure even the intra-family search stuck with a local optima.

While it is possible to have several settings for the parameters in the algorithm, the two major classes can be defined by setting *AccThres* to either a non-negative value or a negative value. A non-negative *AccThres* allows us to stimulate the temperature to be higher while the negative one will permanently set *Count* to zero and then the temperature will never increase. In this paper, we investigate both classes to compare their performance.

## 5. Experimental Evaluation

To evaluate our proposed method, two experiments have been conducted using two different types of classification datasets, the UCI repository and a text classification collection. In the experiments, classification accuracy and training complexity are investigated. Kernel tree structures that obtain the best result for each dataset is also discussed. The rest constitutes three subsections. The first subsection describes properties of our datasets and experimental settings. The second subsection presents performance of our proposed family-based evolutional method compared to the non-family-based evolutional method, as well as two naive methods; namely grid search and gradient search. Moreover, the computational consumption of these five approaches is also discussed. The last subsection illustrates an analysis of optimal kernel tree structures.

### 5.1 Datasets and Experimental Settings

For benchmarking, two different types of datasets are utilized. The first type is composed of ten datasets selected from the UCI repository [25], i.e., *Adult, Balance-Scale (BS), Breast-Cancer-Wisconsin (BCW), German, Heart, Image, Iris, Sonar, Waveform (WF)*, and *Wine*. Table 2 shows the detailed characteristics of each dataset in terms of the number of attributes (#Attrs), the number of dimensions (#Dims), the number of classes (#Cls), the number of samples (#Samps), and the number of samples per class (#Samps/#Cls). Note that the number of dimensions is not equal to that of attributes since a discrete-valued (opposed to numeric-valued) attribute can be broken down to more than one dimension. In addition, values in each dimension are normalized to zero-mean and unit standard deviation.

As the second type, five text classification datasets; WebKB1, WebKB2 [29], 20-Newsgroup (20NG) [30], Thai-Medical [31] and Thai-News [32], are used to explore per-

formance of the proposed method in real application with high dimensions. Some common properties of these datasets are illustrated in Table 3. For WebKB1 and WebKB2, HTML tags are removed to form a plain-text version. In order to reduce the variety of text features in the English datasets, WebKB1, WebKB2 and 20-Newsgroup are preprocessed with stopword removal and word stemming. Since a Thai text has no word boundary, the texts in Thai-Medical and Thai-News datasets are segmented with an open-source word segmentation program, namely cttex [33].

In this work, we investigate the performance of five kernel selection methods, i.e., grid search (GR), grid with gradient search (gGD), genetic programming with gradient search (gGP), family-based GP with non-stimulation (fGP) and family-based GP with stimulation (sfGP). Toward a reasonable comparison, the parameters of each method in our experiments are set in the manner that its SVMs training number is close to each other. The settings of the experiments are as follows. As the baselines, GR occupies a single RBF and polynomial as their kernels. In practical, the computational complexity of GR depends on the sampling size in parameter space. For our GR experiments on the UCI datasets, the sampling in a logarithmic scale with eight (GR-8) and thirty two portions (GR-32) applied on RBF searching space. In the text classification datasets, in addition to RBF, polynomial kernel is also investigated with sampling size of thirty two. Same as GR, gGD occupies a single RBF for all datasets. For gGD, the near optimal solution is first roughly found by GR-32 and then is further refined by the BFGS Quasi-Newton method to adjust the parameters of an RBF kernel in order to achieve the optimal result. Unlike GR and gGD with a single RBF, gGP initially finds a set of near optimal multi-kernel functions by genetic programming, and then fine-tunes the parameters (not the structures) of these multi-kernel functions with gradient search.

Table 4 illustrates our settings of genetic programming in gGP, which is the same as the gGP originally proposed in [24] (even a different name called GP+Gr). The maximum depth and branching factor control the complexity of generated trees during the genetic programming phase. To limit the complexity, our work limits the depth and the branching factor of kernel trees to five and two, respectively. This condition allows us to generate a kernel tree with up to maximum 32 kernels as its leaves. A set of functions and terminals defined determine the set of possible kernel trees. The more operators in the function set as well as the more kernel functions in the terminal set we have, the more varieties of kernel tree structures we can generate. As stated in Sect. 3.1, two operators (addition and multiplication) and three basic kernels (linear, RBF and polynomial) are used in this work. As the fitness function (Fitness Func.), the average classification accuracy of a ten-fold cross-validation is targeted. Same as the gGD, the BFGS Quasi-Newton method is applied as the gradient descent method in the gGP.

Introducing the family concept into the gGP, our proposed two family-based GPs, i.e. fGP and sfGP, perform global search by GP for inter-family crossover and mutation as well as local search using evolution strategy for intra-family crossover and mutation. In the experiment, the ratio of global and local searches is controlled by setting a stimulated temperature during simulated annealing as discussed in Sect. 4.3. To investigate performance of our proposed method, five settings of genetic programming with simulated annealing (GPSA) in Table 5 are used. These settings are applied in Algorithm 1 which is described in Sect. 4.3. The fGP, the version with non-stimulation (NS-SA), allows only decrement of temperature. On the other hand, the sfGP permits both temperature increment and decrement. This mechanism is controlled by setting a negative or a non-negative value for accuracy threshold in the table, accordingly. For the experiment settings of both fGP and sfGPs in Table 5 the range of temperature $[\tau_1, \tau_2]$ is set to $[0, 1000]$.

**Table 2** Characteristics of datasets in the UCI repository.

| Dataset | #Attrs | #Dims | #Cls | #Samps | #Samps/Cls |
|---|---|---|---|---|---|
| Adult | 14 | 105 | 2 | 32561 | 7841;24720 |
| BS | 4 | 20 | 3 | 625 | 49;288;288 |
| BCW | 10 | 10 | 2 | 699 | 241;458 |
| German | 20 | 24 | 2 | 1000 | 300;700 |
| Heart | 13 | 20 | 2 | 270 | 120;150 |
| Image | 19 | 19 | 7 | 210 | 30 each |
| Iris | 4 | 4 | 3 | 150 | 50 each |
| Sonar | 60 | 60 | 2 | 208 | 97;111 |
| WF | 40 | 40 | 3 | 5000 | 1667 each |
| Wine | 13 | 13 | 3 | 178 | 48;59;71 |

**Table 4** Settings of genetic programming (GP).

| Parameter | Setting |
|---|---|
| Maximum Depth | 5 |
| Function Set | Addition and Multiplication (details in [24]) |
| Terminal Set | Linear, RBF, Polynomial |
| Maximum Gen. | 50 |
| #Populations | 30 |
| #Parents | 10 |
| Initialize Method | Ramped Half and Half |
| Fitness Func. | Avg. Acc. over ten-fold CV |
| Operators | Reproduction, Crossover and Mutation |

**Table 3** Characteristics of datasets in the text classification collection.

| Dataset | Type | Lang | #Dims | #Cls | #Docs | #Docs/Cls |
|---|---|---|---|---|---|---|
| 20NG | Plain Text | English | 130151 | 20 | 19997 | $\approx 1000$ |
| WebKB1 | HTML | English | 54492 | 5 | 8282 | 827–4120 |
| WebKB2 | HTML | English | 54492 | 7 | 8282 | 137–3764 |
| Thai Medical | Plain Text | Thai | 297628 | 5 | 3599 | 34–774 |
| Thai News | Plain Text | Thai | 185041 | 5 | 1492 | 252–310 |

**Table 5** Settings of simulated annealing with genetic programming. (NS-SA:non-stimulated simulated annealing, S-SA:stimulated simulated annealing)

| Parameter | NS-SA (fGP) | S-SA-I (sfGP-I) | S-SA-II (sfGP-II) | S-SA-III (sfGP-III) | S-SA-IV (sfGP-IV) |
|---|---|---|---|---|---|
| Temp. Range ($[\tau_1, \tau_2]$) | [0,1000] | [0,1000] | [0,1000] | [0,1000] | [0,1000] |
| Temp. Increasing Rate ($\alpha$) | NA | 10 | 20 | 50 | 100 |
| Temp. Decreasing Rate ($\beta$) | 10 | 10 | 10 | 10 | 20 |
| Accuracy Threshold($\mu$) | NA | 0 | 0 | 0 | 0 |
| Count Threshold ($\psi$) | NA | 5 | 5 | 5 | 5 |
| Maximum Generation ($\sigma$) | 50 | 50 | 50 | 50 | 50 |

**Table 6** Average number of SVM trainings for GR, gGD, gGP, fGP and sfGP in our experimental settings.

| Methods | UCI | TC |
|---|---|---|
| GR-8(RBF) | 64 | - |
| GR-32(Poly) | - | 64 |
| GR-32(RBF) | 1024 | 1024 |
| gGD(RBF) | 1027 | 1024 |
| gGP(Multi) | 1069 | 1080 |
| fGP(Multi) | 1010 | 1010 |
| sfGP(Multi) | 1010 | 1010 |

Intuitively, the range does not have a large effect on accuracy. Therefore the range is fixed for all settings. The temperature starts from 1000, i.e. full encouragement of inter-family operators. In the case of non-stimulated simulated annealing (fGP), the temperature decreasing rate $\beta$ is set to 1% (i.e., 10 out of 1000). This means the percentage of inter-family operators against intra-family operators decreases gradually. Finally the temperature is set to 500 (half of 1000) since the maximum generation is set to 50. At this temperature, the ratio of inter-family and intra-family operators will be the same and the offspring are generated from both intra-family and inter-family operators equally. For the stimulated simulated annealing (sfGP), four settings of parameters are investigated. In sfGP-I, sfGP-II an sfGP-III, the stimulation may be activated every $\psi$ iterations with $\alpha$ at the rate of 10, 20 and 50, respectively. The higher value of $\alpha$ is, the higher rate of inter-family operators against intra-family increases. In the case of sfGP-IV, the decreasing rate $\beta$ is set to 20. That is the intra-family operator increases faster than the case of $\beta = 10$ in sfGP-I, sfGP-II and sfGP-III. In an extreme case of this condition, if the stimulation process is not activated until reaching the maximum generations, there are only intra-family operators available since the temperature will reduce to 0.

Table 6 summarizes the list of the methods we explore, as well as their complexity in the form of the average number of SVM training. The table illustrates average number of invoking an SVM trainer in either of GR, gGD, gGP, fGP or sfGP, for the UCI and text classification datasets. However it is possible to vary the number of SVM calling, by setting different parameters. In the experiment, in order to set the comparable complexity for all methods we set the parameters that make the number of SVM calling as shown in the table. As a practical method, GR-8 consists of 64 SVM trainings since there are two possible parameters, $log(C)$ and $log(\gamma)$. For GR, the number of invoking trainers equals to the multiplication of sampling numbers in every parameter (dimension). For the RBF-based GR-32, the sampling number for each parameter is set equally to 32, the number of calling an SVM trainer in the RBF-based GR is $32 \times 32 = 1024$. In contrast with RBF, the polynomial-based GR varies $log(C)$ and $d$ parameters. The sampling number of $log(C)$ and $d$ are 32 and 2 ($d = 2, 3$), respectively. The total number of trainings with polynomial-based GR is $32 \times 2 = 64$. In general, more parameters with a higher sampling rate will trigger a larger number of SVM trainings. The number of SVM trainings in the gGD method is set to be equal to the summation of the numbers of SVM trainings in the GR method and that in the gradient search (GD) method. As stated previously, the number of SVM trainings in the RBF-based GR corresponds to 1024. Moreover, the number of SVM trainings for the GD depends on the gradient of the objective surface. The more steep the gradient is, the more quickly the GD converges to the optimal value. There are 1024 GR steps and 5 GD steps on average for gGD in our experiment. For gGP, fGP and sfGP, the number of SVM trainings depends on the population size, the number of parents and the number of iterations. In this work, the setting is 30 populations each with 10 parents over 50 iterations. In the first iteration, 30 offsprings are generated as the first population. In each later iteration, only 20 offsprings are generated since 10 excellent parents are remained from the current population to the next population. Therefore, the number of SVM learnings is $30 + (20 \times 49) = 1010$). This number is applicable to the cases of fGP and sfGP. However, unlike fGP and sfGP, the gGP occupies an additional step of gradient search (GD), which contains on average 63 GD steps for the selected 10 offsprings as the parents.

### 5.2 Evaluation on UCI and Text Classification Collection

This section presents the results of investigating our approach in two different tasks: UCI machine learning and text classification. All experiments are done with ten times of ten-fold cross-validation. The experimental results of UCI repository are shown in Table 7. Intuitively GR-32 performs a better result over GR-8 since the GR-32 evaluates with a higher sampling size of parameters in the search space than in GR-8. Unsurprisingly the gGD always outperforms GR-32 since the gGD performs the same process with GR-32, but also followed by gradient search. The gGP overcomes gGD in almost all datasets, excepts Heart, Image and Wine. The fGP and all sfGPs give better performance over gGP in nine out of the ten datasets, except Iris. Anyway, for the exceptional datasets, there is no significance difference on

**Table 7** Average accuracy ± standard deviation of the ten UCI datasets, with significance testing.

| Dataset | GR-8(RBF) | GR-32(RBF) | gGD(RBF) | gGP(Multi) | fGP(Multi) | sfGP-I | sfGP-II | sfGP-III | sfGP-IV |
|---|---|---|---|---|---|---|---|---|---|
| Adult | 82.11 ± 1.37 | 85.63 ± 0.52 | 86.95 ± 1.54 | 87.32 ± 1.14 | **87.65 ± 0.98** | 86.29 ± 1.19 | 86.84 ± 1.26 | 86.97 ± 1.38 | 86.78 ± 1.21 |
| BS | 79.03 ± 2.84 | 80.05 ± 3.16 | 80.47 ± 1.92 | 80.74 ± 1.69 | *82.48 ± 1.42 | 80.76 ± 2.17 | *82.52 ± 2.31 | ***82.61 ± 2.45** | 81.87 ± 3.32 |
| BCW | 91.94 ± 0.32 | 96.35 ± 0.42 | 96.73 ± 1.44 | 97.24 ± 1.42 | 98.15 ± 1.85 | 96.63 ± 2.74 | **98.75 ± 1.39** | 98.13 ± 1.76 | 98.57 ± 1.26 |
| German | 70.55 ± 1.79 | 74.91 ± 1.57 | 75.71 ± 1.89 | 77.31 ± 2.13 | ***79.18 ± 1.91** | *78.83 ± 1.93 | 78.95 ± 2.53 | *79.07 ± 2.20 | *78.84 ± 1.78 |
| Heart | 83.47 ± 3.07 | 86.98 ± 2.68 | 87.32 ± 3.61 | 87.25 ± 1.91 | 86.73 ± 2.34 | 87.14 ± 2.49 | **88.29 ± 3.34** | 88.21 ± 3.75 | 87.71 ± 3.16 |
| Image | 92.83 ± 0.85 | 96.83 ± 0.51 | 97.18 ± 0.87 | 97.03 ± 0.55 | ***98.28 ± 0.79** | *98.02 ± 0.96 | *98.13 ± 0.75 | *98.05 ± 0.65 | *98.22 ± 0.64 |
| Iris | 93.78 ± 3.41 | 96.57 ± 1.09 | 97.27 ± 1.04 | **98.54 ± 0.74** | 97.81 ± 1.63 | 97.52 ± 1.72 | 98.16 ± 0.59 | 98.11 ± 0.72 | 98.36 ± 1.02 |
| Sonar | 83.25 ± 2.93 | 87.84 ± 2.68 | 88.53 ± 2.19 | 90.78 ± 1.76 | 92.37 ± 1.93 | 92.61 ± 1.57 | ***93.19 ± 1.87** | *93.15 ± 1.74 | *92.79 ± 1.65 |
| WF | 84.36 ± 3.96 | 86.11 ± 3.24 | 86.39 ± 3.75 | 88.61 ± 2.73 | 87.74 ± 2.47 | 88.43 ± 1.68 | 88.72 ± 2.83 | **88.98 ± 2.62** | 88.61 ± 2.06 |
| Wine | 96.91 ± 1.44 | 98.77 ± 0.35 | 98.98 ± 0.62 | 98.86 ± 0.70 | 99.31 ± 0.62 | 98.81 ± 0.59 | ***99.24 ± 0.37** | 99.18 ± 0.71 | 99.05 ± 0.31 |

*denotes accuracy improvement when comparing with gGP(Multi) at $P < 0.1$

**Table 8** Average accuracy (%) ± standard deviation of 20NG, WebKB1, WebKB2, Thai Medical and Thai News dataset with significance testing.

| Dataset | GR-32(Poly) | GR-32(RBF) | gGD(RBF) | gGP(Multi) | fGP(Multi) | sfGP-II(Multi) |
|---|---|---|---|---|---|---|
| 20NG | 87.31 ± 2.15 | 87.45 ± 3.86 | 88.64 ± 2.88 | 91.36 ± 3.16 | **92.78 ± 2.76** | 92.76 ± 3.16 |
| WebKB1 | 86.32 ± 3.07 | 86.11 ± 3.47 | 88.39 ± 3.87 | 90.58 ± 3.24 | 90.30 ± 3.43 | **91.54 ± 2.93** |
| WebKB2 | 75.89 ± 2.59 | 77.26 ± 2.13 | 78.18 ± 2.72 | 78.59 ± 2.72 | 82.94 ± 3.08 | ***83.87 ± 1.48** |
| Thai Medical | 83.54 ± 1.84 | 83.23 ± 2.07 | 85.01 ± 1.04 | 86.72 ± 1.43 | *89.72 ± 1.68 | ***89.92 ± 1.45** |
| Thai News | 87.61 ± 1.93 | 88.98 ± 1.88 | 89.12 ± 1.66 | 91.54 ± 1.51 | 91.73 ± 1.21 | ***93.71 ± 2.06** |

*denotes accuracy improvement when comparing with gGP(Multi) at $P < 0.1$

accuracy. In the table, sfGP-II yields the highest average of accuracy for three datasets (i.e., BCW, heart and sonar) and sfGP-III gives the highest for two datasets (i.e. BS and WF) while fGP is superior for three datasets (Adult, German and Image).

As statistical significance testing, the paired t-test with significance level of $P < 0.1$ is performed to compare the proposed methods (i.e., fGP sfGP-I, sfGP-II, sfGP-III and sfGP-IV) with the baseline (i.e., gGP). The significant improvement will be marked with $*$ in Table 7 and Table 8. The sfGP-I, sfGP-II, sfGP-III and sfGP-IV, a family-based kernel tree with stimulated SA, significantly improves the classification over the gGP in two (i.e. German and Image), four (i.e. BS, Image, Sonar and Wine), four (i.e. BS, German, Image and Sonar) datasets, and three (i.e. German, Image and Sonar), respectively. The fGP, a family-based kernel tree with non-stimulated SA, gains significant improvement over the gGP in three datasets (i.e., BS, german and image). Although the gGP acquires the highest accuracy for the Iris dataset, there is no significance difference between gGP and fGP as well as sfGP. For the settings in Table 5, with a higher degree of stimulations (larger $\alpha$), sfGP-II and sfGP-III afford a better performance than sfGP-I. In case of sfGP-IV, both $\alpha$ and $\beta$ is set to a higher value to make more change in the ratio of operation on a higher adjustment inter-family and intra-family. The sfGP-IV gives the performance in the same level with sfGP-II and sfGP-III. As a conclusion, both fGP and sfGPs tend to achieve the higher accuracy compared to gGP while sfGP performs better than fGP in several cases. The sfGP-II seems to be the best setting. Therefore, later we focus on sfGP-II in text datasets.

Besides the toy datasets, we also explore our methods on a number of datasets on real application, i.e. text classification. It is well-known that polynomial kernel frequently gives a good result in text classification task so we investigate GR with polynomial as well as RBF kernel. In this experiment, the performances of fGP and sfGP-II are compared to those of GR(poly), GR(RBF), gGD(RBF) and gGP. For each of five text-classification datasets, the average accuracy and standard deviation of ten times of ten-fold cross validation are figured out as shown in Table 8. It is obvious that gGP seems to obtain a better accuracy than GR(Poly), GR(RBF) and gGD. As another observation, both fGP and sfGP-II outperform gGP for all datasets. In details, sfGP-II performs the best on four datasets (i.e., WebKB1, WebKB2, Thai Medical and Thai News) while fGP yields the highest accuracy for one dataset (20-NewsGroups). However, for the 20-NewsGroup dataset, accuracy difference between fGP and sfGP-II are very trivial (i.e., 92.78% and 92.76%). Compared to gGP (the baseline) using the paired t-test with significance level of $P < 0.1$, sfGP-II significantly improves the classification over gGP in three datasets (i.e., WebKB2, Thai Medical, and Thai News) while fGP gains significant improvement over gGP in only one dataset (i.e., Thai Medical).

As a conclusion, both fGP and sfGP outperform gGP to some extent while sfGP performs better than fGP in most cases.

### 5.3 Analysis on Optimal Kernel Trees

This section provides an analysis of the structural characteristics of the optimal kernel trees generated by fGP and sfGP-II which achieves a best performance among other sfGPs. To do this, given a certain dataset, ten testing trials of algorithm evaluation are performed with ten best trees selected from each trial. By this setting, 100 best trees are considered for each dataset in the UCI repository and the text classification collection. Since the ten datasets shown in Table 2 are selected from the UCI in our experiment, at most 1,000 best

**Table 9** Number of families (#F), minimum depth (MnD), maximum depth (MxD), average depth (AvD), the ratio of the number of addition nodes to that of multiplication nodes (#A/#M), the ratio of the number of radial basis functions to that of linear functions (#R/#P), the ratio of the number of linear functions to that of polynomial functions (#L/#P), of 100 best trees (the UCI repository).

| Dataset | Method | #F | MnD | MxD | AvD | #A/#M | #R/#P | #L/#P |
|---|---|---|---|---|---|---|---|---|
| Adult | fGP | 8 | 2 | 4 | 2.8 | 22.8 | 20.8 | 4.0 |
| | sfGP | 7 | 3 | 5 | 3.8 | 26.6 | 11.7 | 2.3 |
| BS | fGP | 6 | 2 | 5 | 3.9 | 18.5 | 37.5 | 3.5 |
| | sfGP | 5 | 2 | 5 | 3.6 | 34.5 | 22.0 | 2.3 |
| BCW | fGP | 8 | 1 | 4 | 2.8 | 4.3 | 4.8 | 3.2 |
| | sfGP | 9 | 0 | 4 | 2.6 | 11.8 | 7.4 | 3.6 |
| German | fGP | 5 | 2 | 4 | 3.4 | 18.0 | 18.0 | 2.5 |
| | sfGP | 7 | 2 | 4 | 3.4 | 25.5 | 17.0 | 2.0 |
| Heart | fGP | 5 | 2 | 5 | 3.2 | 14.0 | 14.3 | 1.7 |
| | sfGP | 6 | 0 | 4 | 3.3 | 16.5 | 17.0 | 3.7 |
| Image | fGP | 6 | 1 | 4 | 2.1 | 11.5 | 8.0 | 1.3 |
| | sfGP | 8 | 0 | 5 | 2.7 | 18.3 | 17.3 | 3.7 |
| Iris | fGP | 6 | 1 | 4 | 2.7 | 6.75 | 15.5 | 2.0 |
| | sfGP | 5 | 1 | 4 | 2.3 | 30.0 | 32.0 | 3.0 |
| Sonar | fGP | 7 | 2 | 4 | 3.2 | 19.0 | 21.0 | 1.5 |
| | sfGP | 6 | 2 | 4 | 3.1 | 19.0 | 39.0 | 5.0 |
| WF | fGP | 8 | 0 | 4 | 3.6 | 17.5 | 24.7 | 1.7 |
| | sfGP | 7 | 2 | 5 | 3.3 | 22.0 | 43.5 | 5.0 |
| Wine | fGP | 6 | 1 | 4 | 1.9 | 9.7 | 5.4 | 1.2 |
| | sfGP | 7 | 2 | 4 | 3.2 | 18.7 | 19.0 | 2.0 |
| All | fGP | 37 | 0 | 5 | 3.16 | 13.1 | 14.8 | 2.3 |
| | sfGP | 41 | 0 | 5 | 3.14 | 21.0 | 18.1 | 2.9 |

**Table 10** Number of families (#F), minimum depth (MnD), maximum depth (MxD), average depth (AvD), the ratio of the number of addition nodes to that of multiplication nodes (#A/#M), the ratio of the number of radial basis functions to that of polynomial functions (#R/#P), the ratio of the number of linear functions to that of polynomial functions (#L/#P), of 100 best trees (the text classification collection).

| Dataset | Method | #F | MnD | MxD | AvD | #A/#M | #R/#P | #L/#P |
|---|---|---|---|---|---|---|---|---|
| 20NG | fGP | 11 | 2 | 5 | 4.1 | 22.2 | 13.1 | 3.7 |
| | sfGP | 8 | 3 | 4 | 3.4 | 12.3 | 15.5 | 5.5 |
| WebKB1 | fGP | 7 | 3 | 5 | 3.8 | 19.3 | 13.6 | 3.0 |
| | sfGP | 7 | 2 | 5 | 3.6 | 17.7 | 22.7 | 3.7 |
| WebKB2 | fGP | 10 | 2 | 4 | 3.6 | 27.0 | 13.9 | 2.4 |
| | sfGP | 9 | 2 | 4 | 3.9 | 21.0 | 24.0 | 3.2 |
| Thai-Med | fGP | 7 | 1 | 4 | 2.7 | 10.3 | 8.0 | 1.3 |
| | sfGP | 7 | 2 | 5 | 3.5 | 23.0 | 16.8 | 2.0 |
| Thai-News | fGP | 8 | 1 | 5 | 3.8 | 21.8 | 17.6 | 1.2 |
| | sfGP | 7 | 1 | 5 | 3.4 | 16.3 | 10.5 | 1.2 |
| All | fGP | 30 | 1 | 5 | 3.6 | 20.7 | 17.6 | 1.2 |
| | sfGP | 27 | 1 | 5 | 3.6 | 17.6 | 17.3 | 2.9 |

trees, in total, are chosen and their structures are analyzed. From the five text datasets shown in Table 3, five hundred best trees are chosen for structure analysis.

Tables 9 and 10 show the number of families, the minimum tree depth, and the maximum tree depth, average depth for each dataset in the UCI repository and text classification collection, respectively. Moreover, the ratio of addition and multiplication operators and the ratio of three functions are shown. Since both collections obtain relatively similar results, if not specified, we conclude the result of both cases together. For both collections, fGP and sfGP produce approximately 6-10 families for each dataset, even though there are up to 100 trees for each dataset in consideration. The optimal trees for fGP and sfGP hold the average depth of 3.14-3.6. The average depth result shows that a complicated structure may not always yield a best performance. The more complicate the structure is, the more number of parameters are required for adjustment. This property makes it difficult to find the optimal set of parameters. Moreover, when we consider families of the optimal kernel trees in all datasets, we find out that the number of distinct families equals to 37 and 41 for fGP and sfGP in the UCI, respectively. Therefore, of the ten datasets, the average is 3.7 and 4.1. This implies that there are several common kernel trees among datasets since there are around 6-9 families for each dataset. For the text classification collection, they are 30 and 27, respectively. The average is 6 for five datasets. This number means there are few common kernel trees. Since there are around 7-11 families in Table 10. Two additional observations can be made on the portion of operations and the portion of basic kernel functions as shown in the three rightmost columns of both tables (Table 9 and 10). For both fGP and sfGP, it is observed that the number of addition nodes is much larger than that of multiplication nodes. As a possible explanation, the addition node is a natural way to combine many different items together. There

have been several works [18], [19], [21] indicating the advantage of the addition. In contrast with an addition node, a multiplication node may change the properties of the functions. This may trigger a complicate situation and then it is hard to achieve a high accuracy. For an investigation of kernel type, the RBF is the most used function in the optimal kernel tree while the linear is the second frequent one, for both fGP and sfGP. While the range of the value of a single RBF is controlled between 0 and 1, linear and polynomial kernels have a chance to give a bigger value. The proposed method is developed based on evolutional method where the offsprings are gradually evolved for each generation. The kernels and operators that make the overall functions change rapidly may not be suitable to find a proper set of parameters and to survive in a selection process. The last row in the tables (Table 9 and 10) summarizes characteristics when all datasets are considered simultaneously.

In an analysis of structural popularity in the 100 best trees, plotted are two graphs (fGP and sfGP) for the UCI repository in Fig. 5 and the other two graphs (fGP and sfGP) for the text classification collection in Fig. 6. For all graphs, x-axis represents the index of each tree structure in the order of its popularity to be an optimal tree structure, the left y-axis expresses occurrence frequency that the corresponding kernel tree structure becomes the optimal tree structure, and the right y-axis describes the number of datasets that have the corresponding tree as one of its optimal structures. Moreover, each of these four graphs includes two plots; one for the occurrence frequency (related to the left y-axis) and the other for the number of datasets (related to the right y-axis)

In Figs. 5 and 6, it is observed that some most popular trees structure (indexed by a small number, such as 1, 2 and 3, in the x-axis) of sfGP have higher frequency to be the optimal tree structure, comparing to those of fGP. This tendency appears in both the UCI and text classification collection. Moreover, the other plot (related to the right y-axis) states that the most popular tree structure is an optimal tree in 4 (fGP) or 5 (sfGP) out of 10 datasets for UCI. In our assumption we expect that there are some good kernel trees that are common among many problem. This experiment shows some evidences.

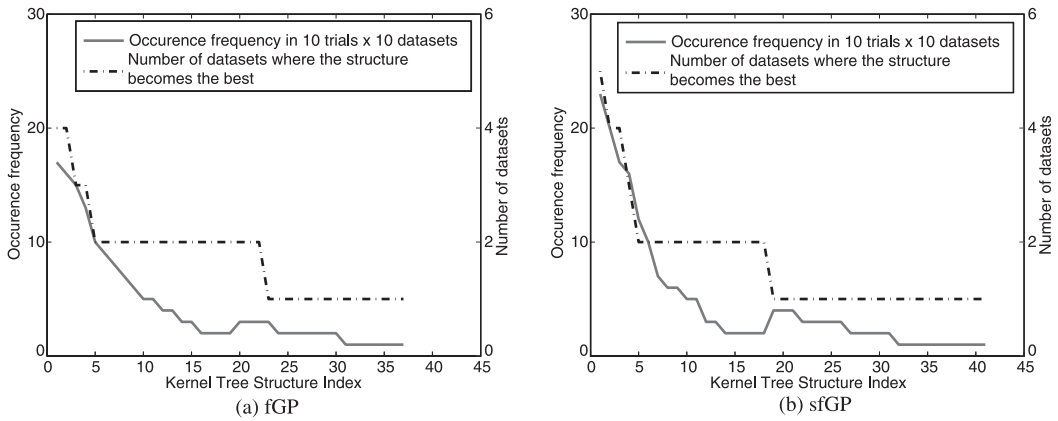



**Fig. 5** The number of datasets where an indexed tree structure remains as one of the final optimal kernel trees (left) and occurrence frequency that an indexed tree structure appears as the optimal kernel tree in ten trials of ten datasets for UCI repository (right: fGP, left: sfGP).

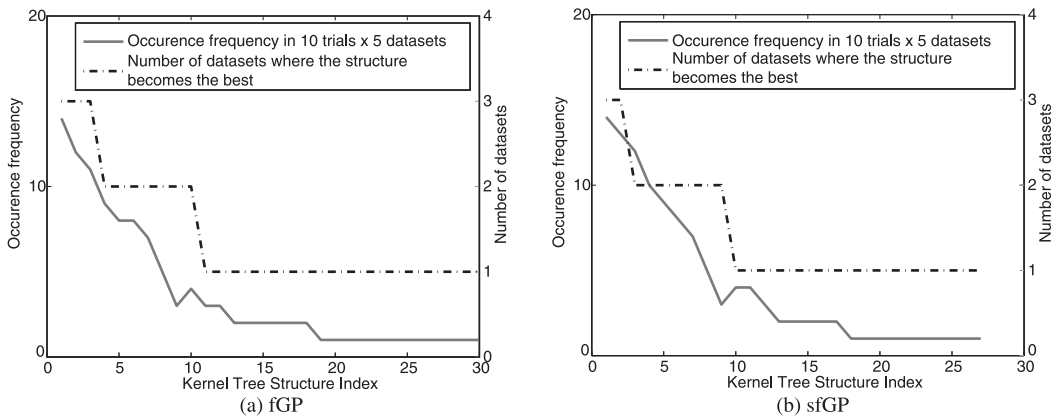



**Fig. 6** The number of datasets where an indexed tree structure remains as one of the final optimal kernel trees (left) and occurrence frequency that an indexed tree structure appears as the optimal kernel tree in ten trials of ten datasets for text classification collection (right: fGP, left: sfGP).

In the text classification task, the most popular tree structure becomes optimal for 3 out of 5 datasets. This result implies that some optimal kernel tree structures are common in several datasets. To be more concrete, the most popular tree structure in the UCI with sfGP is $(((L + R) + (P + R)) + ((P + R) + (R + R)))$, where $R$ is RBF, $P$ is polynomial function and $L$ is linear function. It is the optimal tree structure in four datasets. For UCI with fGP, the most popular tree that appears in five datasets is $(R + (((L + R) + (P + R)) + (L + (R + R))))$. For text classification with sfGP and fGP, the most popular trees structure that occurs in three datasets are $(R + (((L + R) + (L + R)) + ((R + R) + (R + R))))$ and $((R + R) + (R + (L + R)))$, respectively. Although they are various among the conditions, we observed that a popular optimal kernel tends to be a simple addition among multiple basic kernels.

## 6. Conclusion

This paper presented a novel mechanism to find an optimal kernel mapping function for support vector machines (SVMs) towards construction of a high-performance SVM-based classifier. Unlike the conventional approach which searched randomly and diversely, the proposed mechanism utilized the concept of a family of identical-structured kernel trees to enable exploration of structure space using genetic programming whereas to pursue investigation of parameter space on a certain tree. Moreover, simulated annealing is applied to control balance between structure and parameter search towards an optimal kernel. Two proposed versions, namely fGP and sfGP, were proved to be effective using two benchmark collections, the UCI and text classification collections, compared to a naive grid search, a powered gradient search as well as the bare version of GP with gradient search. The experimental results, with significance testing, evidence us the improvement obtained by the proposed methods over the conventional GP in nine out of ten datasets for UCI and in three from five datasets for text classification. By a detailed analysis, some optimal kernel structures were found to be commonly superior in several datasets. It was also observed that an addition node tends to be used in the optimal tree structures rather than a multiplication node. Moreover, RBF is the most frequently used in the optimal kernel tree, compared to linear and polynomial. In summary, the proposed method is shown to be able to find a better optimal solution than other search methods. As our future work, it is worth studying the effectiveness of the proposed methods when they are applied to standard tasks such as regression and traveling salesman, as well as unsupervised tasks such as clustering.

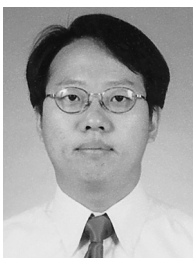
## Acknowledgement

This work has been supported by Golden Jubilee Ph.D. Scholarship under Thailand Research Fund (TRF) PHD/0040/2547 and TRF grant BRG50800013.

## References

[1] C.J.C. Burges, "A tutorial on support vector machines for pattern recognition," Data Mining and Knowledge Discovery, vol.2, no.2, pp.121–167, 1998.
[2] N. Cristianini and J. Shawe-Taylor, An Introduction to Support Vector Machines, Cambridge University Press, 2000.
[3] S. Mika, G. Rätsch, J. Weston, B. Schölkopf, and K.R. Müller, "Fisher discriminant analysis with kernels," Proc. Neural Network for Signal Processing IX, pp.41–48, IEEE, 1999.
[4] G. Dai and D.Y. Yeung, "Kernel selection forl semi-supervised kernel machines," ICML '07: Proc. 24th International Conference on Machine Learning, pp.185–192, New York, USA, ACM, 2007.
[5] B. Schölkopf, A. Smola, and K.R. Müller, "Nonlinear component analysis as a kernel eigenvalue problem," Neurocomputing, vol.10, pp.1299–1319, 1998.
[6] T. Joachims, "Text categorization with support vector machines: Learning with many relevant features," Proc. ECML-98, pp.137–142, 1998.
[7] H. Drucker, B. Shahrary, and D.C. Gibbon, "Support vector machines: relevance feedback and information retrieval," Inf. Process. Manage., vol.38, no.3, pp.305–323, 2002.
[8] Y.D. Cai and S.L. Lin, "Support vector machines for predicting rrna-, rna-, and dna-binging proteins from amino acid sequence," Biochimica et Biophysica Acta - Proteins and Proteomics, vol.1648, no.1-2, pp.127–133, 2003.
[9] A.H. Sung and S. Mukkamala, "Identifying important features for intrusion detection using support vector machines and neural networks," SAINT, pp.209–217, IEEE Computer Society, 2003.
[10] M. Gordan, C. Kotropoulos, and I. Pitas, "A support vector machine-based dynamic network for visual speech recognition applications," EURASIP J. Appl. Signal Process., vol.2002, no.1, pp.1248–1259, 2002.
[11] I. Kotsia and I. Pitas, "Facial expression recognition in image sequences using geometric deformation features and support vector machines," IEEE Trans. Image Process., vol.16, no.1, pp.172–187, 2007.
[12] O. Chapelle, V. Vapnik, O. Bousquet, and S. Mukherjee, "Choosing multiple parameters for support vector machines," Mach. Learn., vol.46, no.1-3, pp.131–159, 2002.
[13] K.M. Chung, W.C. Kao, C.L. Sun, L.L. Wang, and C.J. Lin, "Radius margin bounds for support vector machines with the rbf kernel," Neural Comput., vol.15, no.11, pp.2643–2681, 2003.
[14] S. Keerthi, "Efficient tuning of svm hyperparameters using radius/margin bound and iterative algorithms," IEEE Trans. Neural Netw., vol.13, no.5, pp.1225–1229, 2002.
[15] T. Glasmachers and C. Igel, "Gradient-based adaptation of general gaussian kernels," Neural Comput., vol.17, no.10, pp.2099–2105, 2005.
[16] F. Friedrichs and C. Igel, "Evolutionary tuning of multiple svm parameters," Neurocomputing, vol.64, pp.107–117, 2005.
[17] T. Howley and M.G. Madden, "The genetic kernel support vector machine: Description and evaluation," Artif. Intell. Rev., vol.24, no.3-4, pp.379–395, 2005.
[18] F.R. Bach, G.R.G. Lanckriet, and M.I. Jordan, "Multiple kernel learning, conic duality, and the smo algorithm," ICML '04: Proc. twenty-first International Conference on Machine Learning, p.6, ACM Press, New York, 2004.
[19] G. Lanckriet, N. Cristianini, P. Bartlett, L.E. Ghaoui, and M.I. Jordan, "Learning the kernel matrix with semidefinite programming," J. Mach. Learn. Res., vol.5, pp.27–72, 2004.
[20] S. Sonnenburg, G. Rätsch, C. Schäfer, and B. Schölkopf, "Large scale multiple kernel learning," J. Machine Learning Research, vol.7, pp.1531–1565, July 2006.
[21] T. Phienthrakul and B. Kijsirikul, "Evolutionary strategies for multi-scale radial basis function kernels in support vector machines,"

Proc. 2005 Conference on Genetic and Evolutionary Computation, pp.905–911, ACM Press, New York, USA, 2005.
[22] K.M. Sullivan and S. Luke, "Evolving kernels for support vector machine classification," GECCO '07: Proc. 9th Annual Conference on Genetic and Evolutionary Computation, pp.1702–1707, ACM, New York, USA, 2007.
[23] I. Methasate and T. Theeramunkong, "Experiments on kernel tree support vector machines for text categorization," PAKDD, Lect. Notes Comput. Sci., vol.4426, pp.720–727, Springer, 2007.
[24] I. Methasate and T. Theeramunkong, "Kernel trees for support vector machines," IEICE Trans. Inf. & Syst., vol.E90-D, no.10, pp.1550–1556, Oct. 2007.
[25] D. Newman, S. Hettich, C. Blake, and C. Merz, "UCI repository of machine learning databases," 1998.
[26] T. Phienthrakul and B. Kijsirikul, "GPES: An algorithm for evolving hybrid kernel functions of support vector machines," IEEE Congress on Evolutionary Computation, Singapore, pp.2636–2643, IEEE Press, Sept. 2007.
[27] J. Mercer, "Functions of positive and negative type and their connection with the theory of integral equations," Philosophical Trans. of Royal Society, pp.415–446, 1909.
[28] J.R. Koza, "A genetic approach to the truck backer upper problem and the inter-twined spiral problem," Proc. IJCNN International Joint Conference on Neural Networks, pp.310–318, IEEE Press, 1992.
[29] R. Ghani, "Cmu world wide knowledge base (webkb) project," 2001.
[30] K. Lang, "Newsweeder: Learning to filter netnews," Proc. Twelfth International Conference on Machine Learning, pp.331–339, 1995.
[31] T. Theeramunkong, P. Iamtana-anan, C. Nattee, A. Suriyawongkul, E. Nantajeewarawat, and P. Aimmanee, "A framework for constructing a thai medical knowledge base," Proc. 8th International Symposium on Knowledge and Systems Sciences, pp.45–50, JAIST Press, 2007.
[32] T. Theeramunkong, "Research on automatic relationship discovery in news articles.," Tech. Rep. BRG5080013, Thailand Research Fund, Bangkok, Thai, June 2008.
[33] V. Ampornaramveth, "cttex," Hui's Blog. 2004, Blogspot, 1 Aug. 2009. http://vuthi.blogspot.com/2004/07/cttex.html



**Ithipan Methasate** received his B.Eng. and M.Eng. in Electrical Engineering from Chulalongkorn University, Thailand. He is currently a student in Ph.D. program, Information and Computer Technology School, Sirindhorn International Institute of Technology, Thailand. His research interests are pattern recognition, machine learning and character recognition.

**Thanaruk Theeramunkong** received a bachelor degree in Electric and Electronics, and master and doctoral degrees in Computer Science from Tokyo Institute of Technology in 1990, 1992 and 1995, respectively. His current research interests include data mining, machine learning, natural language processing, and information retrieval.