Partitionable kernels for mapping kernels

Kilho Shin
Graduate School of Applied Informatics
University of Hyogo
Kobe, Japan
yshin@ai.u-hyogo.ac.jp

***Abstract*—Many of tree kernels in the literature are designed tanking advantage of the mapping kernel framework. The most important advantage of using this framework is that we have a strong theorem to examine positive definiteness of the resulting tree kernels. In the mapping kernel framework, each data object is viewed as a collection of components, and a mapping kernel for a pair of data objects is determined as a sum of kernel values of component pairs over a certain range determined according to the purpose of use of the resulting mapping kernel. For those tree kernels known to belong to the mapping kernel category, the string kernel of the product type is commonly used to compute the kernel values of component pairs. This is because it is known that use of the product-type string kernel together with the mapping kernel framework allows us to have recursive formulas to calculate the resulting tree kernels efficiently. We significantly generalizes this result. In fact, we show that we can use *partitionable kernels*, a new class of string kernels instead of the product-type string kernel to enjoy the same advantage, that is, efficient computation based on recursive formulas. The class of partitionable kernels is abundant, and contains the product-type string kernels just as an instance. Also, this result, not limited to tree kernels, can be applied to general mapping kernels after we formalize the decomposition properties of trees as the new notion of *pretty decomposability*.**

I. Introduction

The remarkable success of support vector machine in the recent ten years emphasizes the importance of designing good kernels. In fact, the efficiency and the predictive performance of SVM are greatly affected by the *quality* of kernels used, and we have two fundamental requirements for good kernels: (1) positive definiteness and (2) efficiency of computation ([1]). These two requirements are not always easy to support, and hence, an ad-hoc design of kernels could result in miserable failure. In this regard, a theoretical approach to design good kernels is important. For example, Haussler's convolution kernel ([2]) is an effective tool to design positive definite kernels. Recently, Shin et al. ([3]) generalized Haussler's convolution kernel, and introduced the notion of *mapping kernel*. Moreover, they applied the framework of mapping kernel to tree structures, and revealed that 13 of 19 tree kernels known in the literature are defined according to the single template formula of

$$K(X,Y) = \sum_{(X',Y')\in M_{X,Y}} \prod_{i=1}^{|X'|} \varphi(x_i,y_i).$$

In the formula, X and Y denote trees; $M_{X,Y}$ represents a set of *isomorphic* pairs (X',Y') of substructures of X and Y; $\{x_1,\ldots,x_{|X'|}\}$ and $\{y_1,\ldots,y_{|Y'|}\}$ are the vertex sets of X' and Y' (the isomorphism of X' and Y' determines a bijective correspondence between the vertex sets); and φ is a primitive kernel defined over an alphabet of (labels of) vertices. Shin et al. showed a general condition of $M_{X,Y}$ for $K(X,Y)$ to become positive definite.

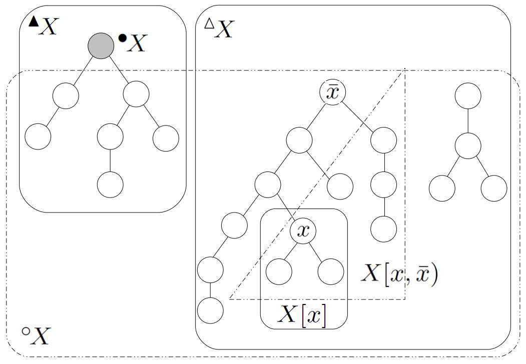
Furthermore, by taking advantage of certain good decomposability of the structure of trees and the form of the *inner kernel* $\prod_{i=1}^{|X'|}\varphi(x_i,y_i)$, we can derive a recursive formula to compute $K(X,Y)$ efficiently.

The main contribution of this paper is to generalize this result in two ways. On one hand, we formalize the structural features of trees useful for efficient computation of $K(X,Y)$, and introduce the notion of *pretty decomposability* of $M_{X,Y}$. By this notion, we can extend the range of application of the method from trees to more general structures. On the other hand, we define the class of *partitionable kernels*, which includes the kernel $\prod_{i=1}^{|X'|}\varphi(x_i,y_i)$ as an instance, and extend the range of inner kernels to which the aforementioned efficient method to compute $K(X,Y)$ is applicable. Hence, we will show that a mapping kernel

$$K(x,y) = \sum_{(x',y')\in M_{x,y}} \kappa(x',y')$$

has a set of recursive formulas to compute itself, if $M_{x,y}$ is pretty decomposable and κ is partitionable. Moreover, we define an invariant *hidden degree* for partitionable kernels, and show that the hidden degree one exactly characterizes the set of kernels of the form of $\prod_{i=1}^{|X'|}\varphi(x_i,y_i)$. For an arbitrary positive hidden degree, we have instances of the partitionable kernel with the hidden degree.

In the remainder of this introductory section, we take the theory of tree edit distance and tree kernel as an example, and clarify the problem that we are going to struggle with



Symbol	Definition
${}^{\bullet}X$	The root of ${}^{\blacktriangle}X$
${}^{\circ}X$	The remaining sub-forest when eliminating ${}^{\bullet}X$ from X
${}^{\blacktriangle}X$	The leftmost subtree of X
${}^{\triangle}X$	The remaining sub-forest when eliminating ${}^{\blacktriangle}X$ from X

Figure 1. Definition of ${}^{\bullet}X, {}^{\circ}X, {}^{\blacktriangle}X$ and ${}^{\triangle}X$

in this paper. Throughout in this paper, by trees, we always mean rooted, ordered and labeled trees.

Introduced by Taï et al. ([4]), Taï edit distance is widely used for comparison of trees, and is defined as follows. An edit script σ that converts a tree X into another Y is a finite sequence of edit operations, where each edit operation can either be:

1) deletion of a vertex x from X, written as $\langle x \to \bullet \rangle$,
2) insertion of a vertex y of Y into X, $\langle \bullet \to y \rangle$,
3) substitution of a vertex y of Y for x of X, $\langle x \to y \rangle$.

To each operation $\langle a \to b \rangle$, is associated a cost $\gamma\langle a \to b \rangle$ which is symmetric in the sense that $\gamma\langle a \to b \rangle = \gamma\langle b \to a \rangle$ holds. The cost $\gamma(\sigma)$ of a script σ is the sum of the costs of all edit operations in σ. Then, Taï edit distance $d(X, Y)$ between X and Y is defined as the minimum over the costs of all scripts that convert X into Y, which can be formulated as follows.

$$d(X, Y) = \min\{\gamma(\sigma) \mid \sigma : X \to Y\}$$

Taï et al. also showed that their tree edit distance can be computed by an efficient dynamic programming algorithm determined by the following recursive expression.

$$d(X, Y) = \min \left\{ \begin{array}{c} \gamma\langle {}^{\bullet}X \to \bullet \rangle + d({}^{\circ}X, Y), \\ \gamma\langle \bullet \to {}^{\bullet}Y \rangle + d(X, {}^{\circ}Y), \\ \gamma\langle {}^{\bullet}X \to {}^{\bullet}Y \rangle + d({}^{\circ\blacktriangle}X, {}^{\circ\blacktriangle}Y) + d({}^{\triangle}X, {}^{\triangle}Y) \end{array} \right\} \quad (1)$$

The range of X and Y is extended to *forests*, ordered sequences of one or more trees, and ${}^{\bullet}X, {}^{\circ}X, {}^{\blacktriangle}X$ and ${}^{\triangle}X$ are operations on forests defined as depicted by Figure 1.

On the other hand, since Collins and Duffy ([5]) introduced the first instance of tree kernels, many tree kernels have been proposed. In relation to tree edit distance, Shin and Kuboyama ([6]) introduced a kernel that is defined as the *exponentiated soft minimum* of costs of edit scripts.

$$K'(X, Y) = \sum_{\sigma: X \to Y} e^{-\lambda\gamma(\sigma)}. \quad (2)$$

This is a clear contrast with Taï's edit distance, which is defined as the hard minimum. The exponentiated soft minimum gives an analytic approximation of the hard minimum, and the following asymptotic property holds.

$$\lim_{\lambda \to \infty} \frac{1}{-\lambda} \log \left(\sum_{i}^{n} e^{-\lambda a_i} \right) = \min\{a_1, \ldots, a_n\}$$

Although the kernel is not always positive definiteness, it is positive definite for the practical setting of $\gamma\langle a \to b \rangle = 1 - \delta_{a,b}$, where $\delta_{a,b}$ is Kronecker's delta, that is, $\delta_{a,b}$ is 1 if $a = b$, and is 0 otherwise.

Moreover, the normalized form of this kernel, that is

$$K(X, Y) = g(X)g(Y)K'(X, Y), \quad (3)$$

where $g(X) = \prod_{x \in X} e^{\lambda\gamma\langle x \to \bullet \rangle}$, turns out to have an efficient method of calculation. In fact, the following recursive formulas yield an efficient dynamic programing algorithm.

$$K(X, Y) = {}^{\bullet}K(X, Y) + K(X, {}^{\circ}Y) + K({}^{\circ}X, Y) - K({}^{\circ}X, {}^{\circ}Y)$$

$${}^{\bullet}K(X, Y) = \left(\frac{e^{-\lambda\gamma\langle {}^{\bullet}X \to {}^{\bullet}Y \rangle}}{e^{-\lambda\gamma\langle {}^{\bullet}X \to \bullet \rangle} e^{-\lambda\gamma\langle \bullet \to {}^{\bullet}Y \rangle}} \right) \cdot \big(1 + K({}^{\circ\blacktriangle}X, {}^{\circ\blacktriangle}Y)\big) \cdot \big(1 + K({}^{\triangle}X, {}^{\triangle}Y)\big) \quad (4)$$

The reader may notice a similarity between the formulas (1) and (4). Both include $\gamma\langle {}^{\bullet}X \to {}^{\bullet}Y \rangle$, $\gamma\langle {}^{\bullet}X \to \bullet \rangle$, $\gamma\langle \bullet \to {}^{\bullet}Y \rangle$, $(X, {}^{\circ}Y)$, $({}^{\circ}X, Y)$, $({}^{\circ\blacktriangle}X, {}^{\circ\blacktriangle}X)$ and $({}^{\triangle}X, {}^{\triangle}Y)$, and the resulting dynamic programming algorithms for edit distance and kernel have the same time complexity $O(|X|^{3/2} |Y|^{3/2})$ by taking advantage of Demaine's strategy ([7]). This similarity is not just a coincidence, but is because these recursive expressions are designed based on the same structural decomposition rules of trees.

In addition, this method to calculate Shin and Kuboyama's kernel is extended to arbitrary tree kernels of the form of

$$K(X, Y) = \sum_{(X', Y') \in M_{X,Y}} \prod_{i=1}^{|X'|} \varphi(x_i, y_i). \quad (5)$$

In the above, $M_{X,Y}$ is the entire set of isomorphic pairs of substructures (X', Y'), that is, for $X' \subsetneqq X$ and $Y' \subsetneqq Y$, there exists a unique bijection between the vertex set of X' and that of Y' that preserves the parent and sibling orders at the same time. Moreover, (x_i, y_i) are the corresponding pairs of vertices of X' and Y' under this bijection, and φ is a kernel determined over the set of vertices. To obtain the tree kernel (3) from (5), we have only to let

$$\varphi(x, y) = \frac{e^{-\lambda\gamma\langle x \to y \rangle}}{e^{-\lambda\gamma\langle x \to \bullet \rangle} e^{-\lambda\gamma\langle y \to \bullet \rangle}}. \quad (6)$$

On the other hand, the recursive formulas to calculate the kernel (5) are obtained by simply replacing $\frac{e^{-\lambda\gamma\langle x\to y\rangle}}{e^{-\lambda\gamma\langle x\to\bullet\rangle}e^{-\lambda\gamma\langle y\to\bullet\rangle}}$ in (4) with $\varphi({}^\bullet X, {}^\bullet Y)$.

In order to investigate positive definiteness of the kernel (5), we can take advantage of a general result for mapping kernels. A *mapping kernel* is abstractly defined by

$$K(X,Y) = \sum_{(X',Y')\in M_{X,Y}} \kappa(X',Y'),$$

and the set $\{M_{X,Y} \mid X,Y\}$ is called a *mapping system*. Theorem 1 of [6] asserts that K is positive definite for any positive definite κ, if, and only if, the mapping system is *transitive* in the following sense: $(X',Y')\in M_{X,Y}$ implies $(Y',X')\in M_{Y,X}$, and $(X',Y')\in M_{X,Y}\wedge(Y',Z')\in M_{Y,Z}$ implies $(X',Z')\in M_{X,Z}$. The mapping kernel has a wide range of application. Shin and Kuboyama ([3]) reported that 18 of 19 tree kernels known in the literature can be defined as mapping kernels, and in particular, 13 of 18 are of the form (5) for different $M_{X,Y}$.

The aim of this paper is to extend the aforementioned technique for efficient computation of tree kernels to general mapping kernels. In fact, we investigate conditions of $M_{x,y}$ and κ such that a mapping kernel determined by

$$K(X,Y) = \sum_{(X',Y')\in M_{X,Y}} \kappa(x_1\dots x_{|X'|}, y_1\dots y_{|Y'|})$$

has recursive formulas to calculate itself efficiently. Here and throughout in this paper, we assume $M_{x,y}$ is a subset of $\bigcup_{i=0}^{\infty}\left(\Sigma^i\times\Sigma^i\right)$, where Σ denotes an alphabet, and hence, an element of $M_{x,y}$ is a pair of strings of the same length. This setting is not too restrictive in practice. For example, a substructure of a tree can be viewed as a sequence of vertices by aligning the vertices in the order of any of in-order, post-order and pre-order traversals. With this setting, we introduce two conditions, *pretty decomposability* for mapping systems $M_{X,Y}$ and *partionability* for κ, and show that, if a mapping system $M_{x,y}$ is *pretty decomposable* and κ is *partitionable kernel*, such recursive formulas exist.

The organization of this paper is as follows. After introducing the definition of partitionable kernels in Section II-A, we show several examples of partitionable kernels in Section II-B. In Section II-C, we see several important properties of partitionable kernels. The properties include closedness of partitionable kernels for addition, multiplication and scholar multiplication. In Section II-D, we first see the reason why partitionable kernels can provide recursive formulas to compute the resulting mapping kernels by taking tree kernels as an instance, and then, introduce the notion of pretty decomposability. In Section III, we see time efficiency and predictive performance of partitionable-kernel-based mapping tree kernels through experiments with four tree kernels and three tree datasets.

II. Partitionable string kernels

A. Definitions

In order to introduce the definition of partitionable kernels in Definition 3, we first see a couple of preliminary definitions.

Definition 1. An *integral* kernel $\kappa^{[*]}$ over Σ is a family of string kernels $\{\kappa^{[i]} : \Sigma^i\times\Sigma^i \longrightarrow \mathbb{R} \mid i\in\{0\}\cup\mathbb{N}\}$. In particular, Σ^0 is a singleton set of the null string $\{\emptyset\}$, and a real value is assigned to $\kappa^{[0]}(\emptyset,\emptyset)$. □

Definition 2. Let $\boldsymbol{x} = x_1\dots x_k$ and $\boldsymbol{y} = y_1\dots y_k$ be strings in Σ^k and ℓ be a positive integer. When a sequence of indices $0 = j_0 \le j_1 \le j_2 \le \cdots \le j_{\ell-1} \le j_\ell = k$ is given, the corresponding *ℓ-partition* of the pair $(\boldsymbol{x},\boldsymbol{y})$ is defined as a sequence of ℓ pairs of strings $(\boldsymbol{x}^{\mathsf{P}_1},\boldsymbol{y}^{\mathsf{P}_1}),\dots,(\boldsymbol{x}^{\mathsf{P}_\ell},\boldsymbol{x}^{\mathsf{P}_\ell})$ such that $\boldsymbol{x}^{\mathsf{P}_i} = x_{j_{i-1}+1}\dots x_{j_i}$ and $\boldsymbol{y}^{\mathsf{P}_i} = y_{j_{i-1}+1}\dots y_{j_i}$.

For $j_i = j_{i+1}$, we define that $\boldsymbol{x}^{\mathsf{P}_i}$ and $\boldsymbol{y}^{\mathsf{P}_i}$ represent the null string $\emptyset$.

For simplicity of description, we denote a 2-partition of $(\boldsymbol{x},\boldsymbol{y})$ by $((\boldsymbol{x}^{\mathsf{L}},\boldsymbol{y}^{\mathsf{L}}),(\boldsymbol{x}^{\mathsf{R}},\boldsymbol{y}^{\mathsf{R}}))$. □

Now, we are ready to introduce partitionable kernels.

Definition 3. A family of integral kernels $\boldsymbol{\kappa} = (\kappa_1^{[*]},\dots\kappa_n^{[*]})$ is said to be *partitionable*, when there exist n-dimensional square matrices $\mathbf{Q}_1,\dots,\mathbf{Q}_n$ such that

$$\kappa_i^{[k]}(\boldsymbol{x},\boldsymbol{y}) = \boldsymbol{\kappa}(\boldsymbol{x}^{\mathsf{L}},\boldsymbol{y}^{\mathsf{L}})\mathbf{Q}_i\,{}^{\mathsf{t}}\boldsymbol{\kappa}(\boldsymbol{x}^{\mathsf{R}},\boldsymbol{y}^{\mathsf{R}})$$

holds for any non-negative integer k, any strings $\boldsymbol{x}$ and $\boldsymbol{y}$ of length k, and any 2-partition $((\boldsymbol{x}^{\mathsf{L}},\boldsymbol{y}^{\mathsf{L}}),(\boldsymbol{x}^{\mathsf{R}},\boldsymbol{y}^{\mathsf{R}}))$ of $(\boldsymbol{x},\boldsymbol{y})$. $\boldsymbol{\kappa}(\boldsymbol{x}^{\mathsf{L}},\boldsymbol{y}^{\mathsf{L}})$ and ${}^{\mathsf{t}}\boldsymbol{\kappa}(\boldsymbol{x}^{\mathsf{R}},\boldsymbol{y}^{\mathsf{R}})$ are the row and column vectors determined by $\left[\kappa_1^{[j_1]}(\boldsymbol{x}^{\mathsf{L}},\boldsymbol{y}^{\mathsf{L}}),\dots,\kappa_n^{[j_1]}(\boldsymbol{x}^{\mathsf{L}},\boldsymbol{y}^{\mathsf{L}})\right]$ and ${}^{\mathsf{t}}\left[\kappa_1^{[k-j_1]}(\boldsymbol{x}^{\mathsf{R}},\boldsymbol{y}^{\mathsf{R}}),\dots,\kappa_n^{[k-j_1]}(\boldsymbol{x}^{\mathsf{R}},\boldsymbol{y}^{\mathsf{R}})\right]$, where the index sequence $0 = j_0 \le j_1 \le j_2 = k$ determines the relevant 2-partition. □

The constraint of 2-partition in Definition 3 is not essential. When we define *ℓ-degree form* p as a homogeneous polynomial of the form

$$p(\boldsymbol{X}_1,\dots,\boldsymbol{X}_\ell) = \sum_{(j_1,\dots,j_\ell)\in\{1,\dots,n\}^\ell} c_{j_1,\dots,j_\ell}\prod_{i=1}^{\ell} X_{(i,j_i)},$$

where $\boldsymbol{X}_i = (X_{(i,1)},\dots,X_{(i,n)})$ is a vector of independent variables, Theorem 1 indicates that the same concept can be defined using ℓ-degree form with $\ell\neq 2$.

Theorem 1. *For a family of kernels $\boldsymbol{\kappa} = (\kappa_1^{[*]},\dots,\kappa_n^{[*]})$, the following are equivalent to each other.*

1) *$\boldsymbol{\kappa}$ is partitionable.*
2) *For some $\ell \geqq 2$, there exist ℓ-degree forms $p_1,\dots,p_n$ such that $\kappa_i^{[*]}(\boldsymbol{x},\boldsymbol{y}) = p_i\big(\boldsymbol{\kappa}(\boldsymbol{x}^{\mathsf{P}_1},\boldsymbol{x}^{\mathsf{P}_1}),\dots,\boldsymbol{\kappa}(\boldsymbol{x}^{\mathsf{P}_\ell},\boldsymbol{y}^{\mathsf{P}_\ell})\big)$ holds for any ℓ-partition.*

3) *For any $\ell \geqq 2$, there exist ℓ-degree forms $p_1^\ell, \dots, p_n^\ell$ such that $\kappa_i^{[*]}(\boldsymbol{x}, \boldsymbol{y}) = p_i\big(\boldsymbol{\kappa}(\boldsymbol{x}^{\mathsf{P}_1}, \boldsymbol{x}^{\mathsf{P}_1}), \dots, \boldsymbol{\kappa}(\boldsymbol{x}^{\mathsf{P}_\ell}, \boldsymbol{y}^{\mathsf{P}_\ell})\big)$ holds for any ℓ-partition.*

Proof: We prove that 1 implies 3 by the mathematical induction on ℓ. Given an ℓ-partition $(\boldsymbol{x}^{\mathsf{P}_1}, \dots, \boldsymbol{x}^{\mathsf{P}_\ell})$, we let $\boldsymbol{x}^{\mathsf{L}} = \boldsymbol{x}^{\mathsf{P}_1}$ and $\boldsymbol{x}^{\mathsf{R}} = \boldsymbol{x}^{\mathsf{P}_2} \| \dots \| \boldsymbol{x}^{\mathsf{P}_\ell}$, where $\|$ indicates the concatenation of strings (vectors). Due to $\kappa_i^{[*]}(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{\kappa}(\boldsymbol{x}^{\mathsf{L}}, \boldsymbol{y}^{\mathsf{L}}) \mathbf{Q}_i {}^{\mathbf{t}}\boldsymbol{\kappa}(\boldsymbol{x}^{\mathsf{R}}, \boldsymbol{y}^{\mathsf{R}})$ and the hypothesis of induction, we have

$$\kappa_i^{[*]}(\boldsymbol{x}) = \boldsymbol{\kappa}(\boldsymbol{x}^{\mathsf{P}_1}, \boldsymbol{y}^{\mathsf{P}_i}) \mathbf{Q}_i \, {}^{\mathbf{t}}\Big[p_j^{\ell-1}\big(\boldsymbol{\kappa}(\boldsymbol{x}^{\mathsf{P}_2}, \boldsymbol{y}^{\mathsf{P}_2}), \dots, \boldsymbol{\kappa}(\boldsymbol{x}^{\mathsf{P}_\ell}, \boldsymbol{y}^{\mathsf{P}_\ell})\big)\Big]_{j=1,\dots,n}.$$

We can define the ℓ-degree form p_i^ℓ by

$$p_i^\ell(\boldsymbol{X}_1, \dots, \boldsymbol{X}_\ell) = \boldsymbol{X}_1 \mathbf{Q}_i {}^{\mathbf{t}}\Big[p_j^{\ell-1}(\boldsymbol{X}_2, \dots, \boldsymbol{X}_\ell)\Big]_{j=1,\dots,n}.$$

It is evident that 3 implies 2. To prove that 2 implies 1, we let $(\boldsymbol{x}^{\mathsf{P}_1}, \dots, \boldsymbol{x}^{\mathsf{P}_\ell}) = (\boldsymbol{x}^{\mathsf{L}}, \boldsymbol{x}^{\mathsf{R}}, \emptyset, \dots, \emptyset)$ and $(\boldsymbol{y}^{\mathsf{P}_1}, \dots, \boldsymbol{y}^{\mathsf{P}_\ell}) = (\boldsymbol{y}^{\mathsf{L}}, \boldsymbol{y}^{\mathsf{R}}, \emptyset, \dots, \emptyset)$. Then, we have

$$f_i(\boldsymbol{x}, \boldsymbol{y}) = p_i\big(\boldsymbol{\kappa}(\boldsymbol{x}^{\mathsf{L}}, \boldsymbol{y}^{\mathsf{L}}), \boldsymbol{\kappa}(\boldsymbol{x}^{\mathsf{R}}, \boldsymbol{y}^{\mathsf{R}}), \boldsymbol{\kappa}(\emptyset, \emptyset), \dots, \boldsymbol{\kappa}(\emptyset, \emptyset))\big).$$

$\mathbf{Q}_i$ is determined by

$$\boldsymbol{X}_1 \mathbf{Q}_i {}^{\mathbf{t}}\boldsymbol{X}_2 = p_i(\boldsymbol{X}_1, \boldsymbol{X}_2, (\emptyset, \emptyset), \dots, f(\emptyset, \emptyset)). \quad \blacksquare$$

Definition 4. An integral kernel $\kappa^{[*]}$ is said to be *partitionable*, if there exists a partitionable kernel family $(\kappa_1^{[*]}, \dots, \kappa_n^{[*]})$ that contains $\kappa^{[*]}$. □

Definition 5. The *hidden degree* of a partitionable kernel $\kappa^{[*]}$ that is not identically 0 is defined as the minimum n of the partitionable kernel families $(\kappa_1^{[*]}, \kappa_2^{[*]} \dots, \kappa_n^{[*]})$ that includes $\kappa^{[*]}$. We denote the hidden degree of $\kappa^{[*]}$ by $\mathsf{hd}(\kappa^{[*]})$, and define $\mathsf{hd}(\kappa^{[*]}) = 0$ for $\kappa^{[*]} \equiv 0$. □

B. Examples

We show a couple of examples of partitionable kernels.

Example 1. For an arbitrary kernel κ over Σ, $\sin_\kappa^{[*]}$ and $\cos_\kappa^{[*]}$ defined below are partitionable.

$$\begin{aligned}
&\sin_\kappa^{[0]}(\emptyset, \emptyset) = 0, \quad \sin_\kappa^{[k]}(\boldsymbol{x}, \boldsymbol{y}) = \sin\left(\sum_{i=1}^k \kappa(x_i, y_i)\right) \\
&\cos_\kappa^{[0]}(\emptyset, \emptyset) = 1, \quad \cos_\kappa^{[k]}(\boldsymbol{x}, \boldsymbol{y}) = \cos\left(\sum_{i=1}^k \kappa(x_i, y_i)\right) \\
&\sin_\kappa^{[*]}(\boldsymbol{x}, \boldsymbol{y}) = \sin_\kappa^{[*]}(\boldsymbol{x}^{\mathsf{L}}, \boldsymbol{y}^{\mathsf{L}}) \cos_\kappa^{[*]}(\boldsymbol{x}^{\mathsf{R}}, \boldsymbol{y}^{\mathsf{R}}) \\
&\qquad\qquad + \cos_\kappa^{[*]}(\boldsymbol{x}^{\mathsf{L}}, \boldsymbol{y}^{\mathsf{L}}) \sin_\kappa^{[*]}(\boldsymbol{x}^{\mathsf{R}}, \boldsymbol{y}^{\mathsf{R}}) \\
&\cos_\kappa^{[*]}(\boldsymbol{x}, \boldsymbol{y}) = \cos_\kappa^{[*]}(\boldsymbol{x}^{\mathsf{L}}, \boldsymbol{y}^{\mathsf{L}}) \cos_\kappa^{[*]}(\boldsymbol{x}^{\mathsf{R}}, \boldsymbol{y}^{\mathsf{R}}) \\
&\qquad\qquad - \sin_\kappa^{[*]}(\boldsymbol{x}^{\mathsf{L}}, \boldsymbol{y}^{\mathsf{L}}) \sin_\kappa^{[*]}(\boldsymbol{x}^{\mathsf{R}}, \boldsymbol{y}^{\mathsf{R}})
\end{aligned}$$

Example 2. For an arbitrary kernel κ over Σ, $(e_0^{[*]}, e_1^{[*]}, e_2^{[*]}, \dots)$ defined below are partitionable.

$$e_d^{[k]}(\boldsymbol{x}, \boldsymbol{y}) = \begin{cases} 1, & \text{if } d = 0, \\ 0, & \text{if } d > k, \\ \displaystyle\sum_{1 \le i_1 < \dots < i_d \le k} \prod_{j=1}^d \kappa(x_{i_j}, y_{i_j}) & \text{if } 0 < d \le k. \end{cases}$$

In fact, $e_d^{[*]}(\boldsymbol{x}, \boldsymbol{y}) = \displaystyle\sum_{i+j=d} e_i^{[*]}(\boldsymbol{x}^{\mathsf{L}}, \boldsymbol{y}^{\mathsf{L}}) \cdot e_j^{[*]}(\boldsymbol{x}^{\mathsf{R}}, \boldsymbol{y}^{\mathsf{R}})$ holds. In addition, $e_\infty^{[k]}(\boldsymbol{x}, \boldsymbol{y}) = \prod_{i=1}^k \kappa(x_i, y_i)$ satisfies $e_\infty^{[*]}(\boldsymbol{x}, \boldsymbol{y}) = e_\infty^{[*]}(\boldsymbol{x}^{\mathsf{L}}, \boldsymbol{y}^{\mathsf{L}}) \cdot e_\infty^{[*]}(\boldsymbol{x}^{\mathsf{R}}, \boldsymbol{y}^{\mathsf{R}})$.

C. Important properties

Partitionable string kernels have several good properties.

Proposition 1. *Let integral kernels $\kappa^{[*]}$ and $\lambda^{[*]}$ be partitionable. Then, $\kappa^{[*]} + \lambda^{[*]}$, $c\kappa^{[*]}$ and $\kappa^{[*]} \cdot \lambda^{[*]}$ are also partitionable. In addition, $\mathsf{hd}(\kappa^{[*]} + \lambda^{[*]}) \le \mathsf{hd}(\kappa^{[*]}) + \mathsf{hd}(\lambda^{[*]})$, $\mathsf{hd}(c\kappa^{[*]}) = \mathsf{hd}(\kappa^{[*]})$ and $\mathsf{hd}(\kappa^{[*]} \cdot \lambda^{[*]}) \le \mathsf{hd}(\kappa^{[*]}) \cdot \mathsf{hd}(\lambda^{[*]})$ hold.*

Due to these properties, we can derive an indefinitely abundant pool of partitionable kernels from a small number of seeds. Regarding positive definiteness, we have the following property.

Proposition 2. *Let $\boldsymbol{\kappa} = (\kappa_1^{[*]}, \dots, \kappa_n^{[*]})$ be a partitionable kernel family. If $\kappa_1^{[1]}, \dots, \kappa_n^{[1]}$ are all positive definite and $\mathbf{Q}_i$ are all non-negative, $\kappa_1^{[*]}, \dots, \kappa_n^{[*]}$ are positive definite.*

Of course, the converse does not necessarily hold.

Example 3. Alghough $\cos_\kappa^{[*]}$ is positive definite for $\kappa(x, y) = x - y$, $\mathbf{Q}_{\cos_\kappa^{[*]}} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$ holds.

We say that $\kappa^{[*]}$ is *character-wise symmetric*, iff,

$$\kappa^{[k]}(x_1 \dots x_k, y_1 \dots y_k) = \kappa^{[k]}(x_{\sigma(1)} \dots x_{\sigma(k)}, y_{\sigma(1)} \dots y_{\sigma(k)})$$

holds for arbitrary k and permutation $\sigma \in \mathfrak{S}_k$. Although whether all partitionable kernels are character-wise symmetric is an open problem, we have the following properties.

Proposition 3. *Let $(\kappa_1^{[*]}, \dots, \kappa_n^{[*]})$ be a partitionable kernel family. If all of $\mathbf{Q}_i$ are symmetric, all of $\kappa_i^{[*]}$ are character-wise symmetric. For $n = 2$, if $\kappa_1^{[1]}$ and $\kappa_2^{[1]}$ are linearly independent, all of $\mathbf{Q}_i$ are symmetric.*

Proof: 1. For a permutation $\sigma \in \mathfrak{S}_k$, we let ${}^\sigma x_1 \dots x_k = x_{\sigma(1)} \dots x_{\sigma(k)}$, and show $\kappa_i^{[k]}(\boldsymbol{x}, \boldsymbol{y}) = \kappa_i^{[k]}({}^\sigma\boldsymbol{x}, {}^\sigma\boldsymbol{y})$. We prove the claim by mathematical induction on k. When $k = 2$, the claim follows from

$$\begin{aligned}
\kappa_i^{[2]}(x_1 x_2, y_1 y_2) &= \boldsymbol{\kappa}(x_1, y_1) \mathbf{Q}_i {}^{\mathbf{t}}\boldsymbol{\kappa}(x_2, y_2) \\
&= \boldsymbol{\kappa}(x_2, y_2) \mathbf{Q}_i {}^{\mathbf{t}}\boldsymbol{\kappa}(x_1, y_1) = \kappa_i^{[2]}(x_2 x_1, y_2 y_1).
\end{aligned}$$

When $k > 2$, it suffices to prove the claim for the case of $\sigma = (i, i+1)$. First, we assume that $i < k-1$, and take the 2-partition corresponding to $0 = j_0 \le j_1 = i+1 < j_2 = k$. By the hypothesis of induction, we have

$$\begin{aligned}\kappa_i^{[k]}({}^\sigma\boldsymbol{x}, {}^\sigma\boldsymbol{y}) &= \boldsymbol{\kappa}({}^\sigma\boldsymbol{x}^{\mathsf{L}}, {}^\sigma\boldsymbol{y}^{\mathsf{L}})\mathbf{Q}_i{}^{\mathbf{t}}\boldsymbol{\kappa}({}^\sigma\boldsymbol{x}^{\mathsf{R}}, {}^\sigma\boldsymbol{y}^{\mathsf{R}}) \\ &= \boldsymbol{\kappa}({}^\sigma\boldsymbol{x}^{\mathsf{L}}, {}^\sigma\boldsymbol{y}^{\mathsf{L}})\mathbf{Q}_i{}^{\mathbf{t}}\boldsymbol{\kappa}(\boldsymbol{x}^{\mathsf{R}}, \boldsymbol{y}^{\mathsf{R}}) \\ &= \boldsymbol{\kappa}(\boldsymbol{x}^{\mathsf{L}}, \boldsymbol{y}^{\mathsf{L}})\mathbf{Q}_i{}^{\mathbf{t}}\boldsymbol{\kappa}(\boldsymbol{x}^{\mathsf{R}}, \boldsymbol{y}^{\mathsf{R}}) = \kappa_i^{[k]}(\boldsymbol{x}, \boldsymbol{y}).\end{aligned}$$

In the case of $i = k-1$, we have onlyt to let $j_1 = k-2$, and exchange the roles of $(\boldsymbol{x}^{\mathsf{L}}, \boldsymbol{y}^{\mathsf{L}})$ and $(\boldsymbol{x}^{\mathsf{R}}, \boldsymbol{y}^{\mathsf{R}})$.

2. We will see that $\mathbf{Q}_1$ is symmetric. Since $\kappa_1^{[*]}$ and $\kappa_2^{[*]}$ are linearly independent, we have

$$\mathbf{Q}_1 \begin{bmatrix} \kappa_1^{[0]} \\ \kappa_2^{[0]} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad \text{and} \quad [\kappa_1^{[0]}, \kappa_2^{[0]}]\mathbf{Q}_1 = [1, 0].$$

The claim follows. ■

Even if all $\kappa_i^{[*]}$ are character-wise symmetric, $\mathbf{Q}_i$ are not necessarily symmetric. For example, let $\kappa_i^{[*]}(\boldsymbol{x}, \boldsymbol{y}) \equiv i$ for $i = 1, 2$. When we define $\mathbf{Q}_i = \begin{bmatrix} a_i & b_i \\ c_i & d_i \end{bmatrix}$ to satisfy

$$i = \kappa_i^{[k]} = [1, 2] \begin{bmatrix} a_i & b_i \\ c_i & d_i \end{bmatrix} \begin{bmatrix} 1 \\ 2 \end{bmatrix} = a_i + 2b_i + 2c_i + 4d_i,$$

$\mathbf{Q}_i$ is not necessarily symmetric.

For partitionable kernels of $\mathsf{hd}(\kappa^{[*]}) = 1$, we have the following beautiful theorem.

Theorem 2. *If $\kappa^{[*]}$ is partitionable with $\mathsf{hd}(\kappa^{[*]}) = 1$, $\kappa^{[k]}(\boldsymbol{x}, \boldsymbol{y}) = (\kappa^{[0]})^{1-k} \prod_{i=1}^{k} \kappa^{[1]}(x_i, y_i)$ holds.*

D. Kernel computation

Lemma 1 shows the key property of partitionable kernels with respect to computational feasibility of mapping kernels derived from them. In the remainder of this paper, $\mathfrak{I}$ denotes $\mathfrak{P}_0\left(\bigcup_{i=0}^{\infty}(\Sigma^i \times \Sigma^i)\right)$, that is, the entire set of finite subsets of $\bigcup_{i=0}^{\infty}(\Sigma^i \times \Sigma^i)$.

Lemma 1. *Let $(\kappa_1^{[*]}, \ldots, \kappa_n^{[*]})$ be a partitionable kernel family. Hence, there exists Q_i such that*

$$\begin{aligned}\kappa_i^{[*]}(\boldsymbol{x}, \boldsymbol{y}) =& (\kappa_1^{[*]}(\boldsymbol{x}^{\mathsf{L}}, \boldsymbol{y}^{\mathsf{L}}), \ldots, \kappa_1^{[*]}(\boldsymbol{x}^{\mathsf{L}}, \boldsymbol{y}^{\mathsf{L}}))Q_i \\ & {}^{\mathbf{t}}(\kappa_1^{[*]}(\boldsymbol{x}^{\mathsf{R}}, \boldsymbol{y}^{\mathsf{R}}), \ldots, \kappa_1^{[*]}(\boldsymbol{x}^{\mathsf{R}}, \boldsymbol{y}^{\mathsf{R}}))\end{aligned}$$

holds for all i. Then, if M_1 and M_2 are in $\mathfrak{I}$,

$$\sum_{(\boldsymbol{x},\boldsymbol{y}) \in M_1 \| M_2} \kappa_i^{[*]}(\boldsymbol{x}, \boldsymbol{y}) = (K_{1,1}, \ldots, K_{1,n})\mathbf{Q}_i{}^{\mathbf{t}}(K_{2,1}, \ldots, K_{2,n})$$

holds for $K_{i,j} = \sum_{(\boldsymbol{x},\boldsymbol{y}) \in M_i} \kappa_j^{[*]}(\boldsymbol{x}, \boldsymbol{y})$.

We let $\boldsymbol{a} \| \boldsymbol{b}$ denote the concatenation of strings $\boldsymbol{a}$ and $\boldsymbol{b}$, and define $M_1 \| M_2 \in \mathfrak{I}$ by

$$\left\{(\boldsymbol{x}_1 \| \boldsymbol{x}_2, \boldsymbol{y}_1 \| \boldsymbol{y}_2) \mid (\boldsymbol{x}_1, \boldsymbol{y}_1) \in M_1, (\boldsymbol{x}_2, \boldsymbol{y}_2) \in M_2\right\}.$$

Since proving Lemma 1 is easy, we see how this property is useful to calculate partitionable-kernel-based mapping kernels using an example. In the example, we let x and y be ordered forests, and consider the entire set of isomorphic pairs of substructures of X and Y. Converting a substructure z of x or y into a string of vertices by aligning the vertices of z in the order of in-order traversal, $M_{x,y} \subseteqq \mathfrak{I}$ is derived from the set of isomorphic substructure pairs, where Σ denotes the alphabet composed of vertices. Letting ${}^\bullet M_{x,y} = \{(\boldsymbol{x}', \boldsymbol{y}') \in M_{x,y} \mid {}^\bullet x \in \boldsymbol{x}', {}^\bullet y \in \boldsymbol{y}'\}$, it is easy to see that the following decomposition formulas hold.

$$M_{x,y} = {}^\bullet M_{x,y} \sqcup \left(M_{x,{}^\circ y} \cup M_{{}^\circ x,y}\right) \tag{7}$$

$$M_{x,{}^\circ y} \cap M_{{}^\circ x,y} = M_{{}^\circ x,{}^\circ y} \tag{8}$$

$$\begin{aligned}{}^\bullet M_{x,y} = M_{{}^\bullet x,{}^\bullet y} &\sqcup \left(M_{{}^\bullet x,{}^\bullet y} \| M_{{}^{\circ\blacktriangle}x,{}^{\circ\blacktriangle}y}\right) \sqcup \left(M_{{}^\bullet x,{}^\bullet y} \| M_{{}^\vartriangle x,{}^\vartriangle y}\right) \\ &\sqcup \left(M_{{}^\bullet x,{}^\bullet y} \| M_{{}^{\circ\blacktriangle}x,{}^{\circ\blacktriangle}y} \| M_{{}^\vartriangle x,{}^\vartriangle y}\right)\end{aligned} \tag{9}$$

In the above, $\sqcup$ denotes the disjoint union.

When we let $K_i(x,y) = \sum_{(\boldsymbol{x}',\boldsymbol{y}') \in M_{x,y}} \kappa_i^{[*]}(\boldsymbol{x}', \boldsymbol{y}')$, the formulas of (7) and (8) imply

$$\begin{aligned}K_i(x,y) = &\sum_{(\boldsymbol{x}',\boldsymbol{y}') \in {}^\bullet M_{x,y}} \kappa_i^{[*]}(\boldsymbol{x}', \boldsymbol{y}') \\ &+ K_i(x, {}^\circ y) + K_i({}^\circ x, y) - K_i({}^\circ x, {}^\circ y).\end{aligned}$$

The first term of the right-hand side can be further decomposed by the formula (9) as

$$\begin{aligned}\sum_{(\boldsymbol{x}',\boldsymbol{y}') \in {}^\bullet M_{x,y}} \kappa_i^{[*]}(\boldsymbol{x}', \boldsymbol{y}') &= \kappa_i^{[1]}({}^\bullet x, {}^\bullet y) \\ &+ \sum_{(\boldsymbol{x}',\boldsymbol{y}') \in M_{{}^\bullet x,{}^\bullet y} \| M_{{}^{\circ\blacktriangle}x,{}^{\circ\blacktriangle}y}} \kappa_i^{[*]}(\boldsymbol{x}', \boldsymbol{y}') + \sum_{(\boldsymbol{x}',\boldsymbol{y}') \in M_{{}^\bullet x,{}^\bullet y} \| M_{{}^\vartriangle x,{}^\vartriangle y}} \kappa_i^{[*]}(\boldsymbol{x}', \boldsymbol{y}') \\ &+ \sum_{(\boldsymbol{x}',\boldsymbol{y}') \in M_{{}^\bullet x,{}^\bullet y} \| M_{{}^{\circ\blacktriangle}x,{}^{\circ\blacktriangle}y} \| M_{{}^\vartriangle x,{}^\vartriangle y}} \kappa_i^{[*]}(\boldsymbol{x}', \boldsymbol{y}'),\end{aligned}$$

and we can apply Lemma 1 to the last three terms of the right-hand side. For example, by Lemma 1, the second term turns out to be

$$\begin{aligned}&\sum_{(\boldsymbol{x}',\boldsymbol{y}') \in M_{{}^\bullet x,{}^\bullet y} \| M_{{}^{\circ\blacktriangle}x,{}^{\circ\blacktriangle}y}} \kappa_i^{[*]}(\boldsymbol{x}', \boldsymbol{y}') \\ &= \left(K_1({}^\bullet x, {}^\bullet y), \ldots, K_n({}^\bullet x, {}^\bullet y)\right)\mathbf{Q}_i \\ &\qquad {}^{\mathbf{t}}\left(K_1({}^{\circ\blacktriangle}x, {}^{\circ\blacktriangle}y), \ldots, K_n({}^{\circ\blacktriangle}x, {}^{\circ\blacktriangle}y)\right) \\ &= \left(\kappa_1^{[1]}({}^\bullet x, {}^\bullet y), \ldots, \kappa_n^{[1]}({}^\bullet x, {}^\bullet y)\right)\mathbf{Q}_i \\ &\qquad {}^{\mathbf{t}}\left(K_1({}^{\circ\blacktriangle}x, {}^{\circ\blacktriangle}y), \ldots, K_n({}^{\circ\blacktriangle}x, {}^{\circ\blacktriangle}y)\right).\end{aligned}$$

Thus, the formulas of (7), (8) and (9) together with Lemma 1 provide us with a method to reduce calculation of $K_1(x,y), \ldots, K_n(x,y)$ to that of $K_1, \ldots, K_n$ for smaller forests. By applying this method iteratively, we can finally reach the concrete values of $K_1(x,y), \ldots, K_n(x,y)$.

In [8], in addition to the formulas of (7), (8) and (9), seven sets of decomposition formulas are presented in order to calculate seven different classes of tree kernels. These tree kernels are of the form of $\sum_{(\boldsymbol{x}',\boldsymbol{y}')\in M_{x,y}} \prod_{i=1}^{|\boldsymbol{x}'|} \varphi(x'_i, y'_i)$, and hence, are derived from partitionable kernels with hidden degree 1.

In the same way as mentioned above, Lemma 1 enables us to take advantage of these decomposition formulas in order to compute novel mapping tree kernels derived by replacing $\prod_{i=1}^{|\boldsymbol{x}'|} \varphi(x'_i, y'_i)$ with partitionable kernels of higher hidden degree.

In the following, we will generalize what we saw in the above on decomposition formulas of tree mapping systems, and will introduce the notion of *pretty decomposability* as a formalized condition for mapping systems that allows application of Lemma 1 in order to calculate the resulting mapping kernels.

We let $(\mathcal{X}, <)$ be an (infinite) ordered set of objects such that, for any $x \in \mathcal{X}$, there exists a positive integer p and, if $x \geq x_1 \geq \cdots \geq x_q$ for $p < q$, then $x_p = x_{p+1} = \cdots = x_q$ holds. In addition, we assume that a mapping system $M : \mathcal{X} \times \mathcal{X} \to \mathfrak{I}$ and a finite set of *reduction operations* $\pi_1, \ldots, \pi_\ell$ are given. A reduction operation is a mapping $\pi : \mathcal{X} \to \mathcal{X}$ such that $\pi(x) < x$ holds for $\forall x \in \mathcal{X}$ unless x is minimal.

We define $\mathcal{E}$ as the minimum subset of $(\mathcal{X} \times \mathcal{X})^{\mathfrak{I}}$ that satisfies the following conditions. We let X and Y be variables that move over $\mathcal{X}$.

1) The constant mapping $(X, Y) \mapsto \emptyset$ is in $\mathcal{E}$.
2) $M(X, \pi_i(Y))$ and $M(\pi_i(X), Y)$ are in $\mathcal{E}$.
3) If $e(X, Y)$ is in $\mathcal{E}$, $e(\pi_i(X), Y)$ and $e(X, \pi_i(Y))$ are in $\mathcal{E}$.
4) If $e(X, Y)$ and $e'(X, Y)$ are in $\mathcal{E}$, $e(X, Y) \cup e'(X, Y)$ and $e(X, Y) \| e'(X, Y)$ are in $\mathcal{E}$.

We can view $e(X, Y) \in \mathcal{E}$ as an expression with respect to X and Y composed of symbols $\emptyset, M(\cdot, \cdot), \pi_i, \cup$ and $\|$.

Pretty decomposable defined below is a condition of a mapping system M such that the resulting mapping kernels have recursive decomposition formulas when the sub-kernels are partitionable.

Definition 6. A mapping system M is said to be *pretty decomposable*, if, and only if, there exists a finite set of expressions $\{e_1(X, Y), \ldots, e_m(X, Y)\} \subset \mathcal{E}$ that satisfies the following.

1) M is identical to $e_1(X, Y)$ as a mapping from $\mathcal{X} \times \mathcal{X}$ to $\mathfrak{I}$.
2) If a sub-expression $e'(X, Y) \cup e''(X, Y)$ appears in some $e_i(X, Y)$, $e'(X, Y) \cap e''(X, Y)$ determines the same mapping as some $e_j(X, Y)$. □

When $\mathcal{X}$ is the set of ordered forests, we can define an order $>$ so that $x > y$, if, and only if, y is a proper substructure of x. Then, the operations ${}^{\bullet}x$, ${}^{\circ}x$, ${}^{\triangle}x$ and ${}^{\blacktriangle}x$ are all reduction operators. Moreover, under these settings, the mapping system $M(x, y) = M_{x,y}$ is pretty decomposable. In fact, the decomposition formulas of (7), (8) and (9) determines a set of expressions that satisfies the conditions described in Definition 6.

Theorem 3. *Let $M_{x,y}$ be pretty decomposable and $(\kappa_1^{[*]}, \ldots, \kappa_n^{[*]})$ be a partitionable kernel family. There exists a set of recursive decomposition formulas that reduces calculation of*

$$K_i(x, y) = \sum_{(\boldsymbol{x}',\boldsymbol{y}')\in M(x,y)} \kappa_i^{[*]}(\boldsymbol{x}', \boldsymbol{y}')$$

for $i = 1, \ldots, n$ to calculation of a finite set of $\{K_{i_j}(x_j, y_j) \mid j = 1, \ldots, N\}$ such that at least one of $x_j < x$ and $y_j < y$ holds for any $j = 1, \ldots, N$.

Proof: (Sketch) First, we introduce an order $\leq$ into $\mathcal{E}$. We denote $e \leq e'$ if, and only if, for any $(x, y) \in \mathcal{X} \times \mathcal{X}$, there exists an injective mapping $f : e(x, y) \to e'(x, y)$ such that each component of $f(\xi, \eta)$ contains the corresponding component of (ξ, η). Note that, if $(\xi, \eta) \in e(x, y)$, the components ξ and η are strings over Σ of the same length. Also $e < e'$ indicates the case that $e \leq e'$ and e is not identical to e' as a mapping from $\mathcal{X} \times \mathcal{X}$ to $\mathfrak{I}$.

We prove that the assertion of the theorem holds for $\sum_{(\boldsymbol{x}',\boldsymbol{y}')\in e(x,y)} \kappa_i^{[*]}(\boldsymbol{x}', \boldsymbol{y}')$ when e is a sub-expression of some of e_i by the mathematical induction on the order $<$ of $\mathcal{E}$.

We investigate two cases one by one: $e'(X, Y) = e''(X, Y) \cup e'''(X, Y)$ and $e'(X, Y) = e''(X, Y) \| e'''(X, Y)$.

First, we assume $e(X, Y) = e'(X, Y) \cup e''(X, Y)$. If $e'(x, y) \subsetneqq e''(x, y)$ holds for all (x, y), we can eliminate such e' without harming the result at all. Thus, we can assume that $e''(x, y) \backslash e'(x, y) \neq \emptyset$ holds for some (x, y), and hence, we can conclude $e > e'$. In the same way, we have $e > e''$. On the other hand, $e'(X, Y) \cap e''(X, Y) = e_i(X, Y)$ holds for some i (Definition 6), and hence $e > e_i$ holds. The claim follows from the hypothesis of induction and

$$\sum_{(\boldsymbol{x}',\boldsymbol{y}')\in e(x,y)} \kappa_i^{[*]}(\boldsymbol{x}', \boldsymbol{y}') = \sum_{(\boldsymbol{x}',\boldsymbol{y}')\in e'(x,y)} \kappa_i^{[*]}(\boldsymbol{x}', \boldsymbol{y}') + \sum_{(\boldsymbol{x}',\boldsymbol{y}')\in e''(x,y)} \kappa_i^{[*]}(\boldsymbol{x}', \boldsymbol{y}') \quad - \sum_{(\boldsymbol{x}',\boldsymbol{y}')\in e_i(x,y)} \kappa_i^{[*]}(\boldsymbol{x}', \boldsymbol{y}').$$

Secondly, we assume $e(X, Y) = e'(X, Y) \| e''(X, Y)$. If $e''(x, y) = \emptyset$ for all (x, y), we can eliminate such e'' without harming the result at all. Thus, we can assume that $e > e'$ and $e > e''$. By Lemma 1, $\sum_{(\boldsymbol{x}',\boldsymbol{y}')\in e(x,y)} \kappa_i^{[*]}(\boldsymbol{x}', \boldsymbol{y}')$ is a function of $\sum_{(\boldsymbol{x}',\boldsymbol{y}')\in e'(x,y)} \kappa_j^{[*]}(\boldsymbol{x}', \boldsymbol{y}')$ and $\sum_{(\boldsymbol{x}',\boldsymbol{y}')\in e''(x,y)} \kappa_j^{[*]}(\boldsymbol{x}', \boldsymbol{y}')$. The claim follows from the hypothesis of induction. ■

Given $x \in \mathcal{X}$, since the length of a path from x to a minimal object is bounded above by p, a program that iteratively evaluate the recursive decomposition formulas obtained by Theorem 3 stops within finite lengths of time. Furthermore, the time complexity of this program when applied to $\kappa^{[*]}$ is $\mathsf{hd}(\kappa^{[*]})$ times as large as that of the program when calculating

$$K(X,Y) = \sum_{(X',Y')\in M(X,Y)} 1.$$

This is because the program has to calculate $\mathsf{hd}(\kappa^{[*]})$ mapping kernels in parallel.

III. Empirical results

In this section, we will show some experimental results. In the experiments, we experimented with four different mapping tree kernels, which are all derived from partitionable kernels. Two of the kernels (K_0 and K_3) are selected from the literature, while the others (K_1 and K_2) are newly examined in this paper.

To begin with, we would like to describe the purpose of our experiments briefly. The main contribution of this paper is to propose an abundant class of mapping kernels that have practical computational complexity, and hence, we think that it is not so meaningful to focus on a small number of instances of partitionable kernels out of a large number of candidates, and to run experiments with them for the purpose of comparing them with kernels known in the literature. In fact, Proposition 1, for example, provides us with a method to explore novel kernels in a large space of partitionable kernels. Hence, evaluation of predictive performance of a few instances will say almost nothing about the property of the entire class. In addition, a kernel which is effective to a particular type of datasets is not always effective to another type of datasets. Also, the fact that a kernel is not useful for some type of datasets does not necessarily mean that it is always useless. When we understand that the purpose of our contribution is never to propose kernels that will be effective to particular types of datasets, experimental results with a small number of specific datasets can only show a potential of the kernel class. Thus, the purpose of the experiments to describe here is to show (1) the fundamental computational feasibility of partitionable-kernel-based mapping kernels and (2) their potential in terms of predictive performance when applied to practical problems.

All tree kernels K_i used in the experiments are mapping kernels derived from partitionable kernels $\kappa_i^{[*]}$. The mapping system $M_{X,Y}$ is determined to be the entire set of isomorphic substructures of trees X and Y. Hence,

$$K_i(X,Y) = \sum_{(X',Y')\in M_{X,Y}} \kappa_i^{[*]}(X',Y')$$

holds. The partitionable kernels $\kappa_0^{[*]}$ and $\kappa_3^{[*]}$ are of hidden degree 1, while $\kappa_1^{[*]}$ and $\kappa_2^{[*]}$ are of hidden degree 2 and 3, respectively. In fact, $\kappa_1^{[*]}$ forms a partitionable kernel family with $\kappa_0^{[*]}$, and $\kappa_2^{[*]}$ does with $\kappa_0^{[*]}$ and $\kappa_1^{[*]}$. Table I and Table II describe K_i and κ_i.

For the experiments, we use three datasets, named `colon-cancer`, `cystic` and `leukemia`, from the KEGG/GLYCAN database [9]. The total number of and averages of sizes and heights of the examples in those datasets are given as follows.

Dataset	# Examples	Ave. Size	Ave. Height
`colon-cancer`	134	8.4	5.6
`cystic`	160	8.3	5.0
`leukemia`	442	13.5	7.4

The following is the steps of the experiment with each dataset.

1) For each tree kernel, compute the corresponding Gram matrix. The matrix includes N^2 elements, when the relevant dataset includes N sample trees. An element of the matrix includes the decay factor λ, and is a function of λ.
2) Generate 10 pairs of a training dataset and a test dataset by splitting the relevant dataset independently at random. The training dataset is so generated that it is approximately four times larger than the test dataset.
3) For each combination of a dataset pair and a tree kernel, perform the following steps.
   a) Run a grid search to find optimal values for the decay factor λ and the regulation parameter C of SVM. Hence, repeat run of five-fold cross validation on the training data changing λ and C and select the parameter assignment that exhibits the greatest AUC-of-ROC-Curve value.
   b) Make SVM learn the entire training data with the optimal parameters obtained in 3a.
   c) Evaluate three measures, namely, AUC of ROC Curve, F-Score and Accuracy by applying the hypothesis (model) obtained in 3b to the test dataset.

   In these steps, SVM refers to the Gram matrix generated in Step 1 in one of the following ways. In one way, SVM uses values $K_i(X,Y)$ found in the matrix as they are. In the other way, it uses normalized values $\frac{K_i(X,Y)}{\sqrt{K_i(X,X)K_i(Y,Y)}}$.

In computing the Gram matrices in Step 1, we employed the accelerating technique reported in [8], which first extracts all of the substructures up to congruence that appear in a dataset, and then calculate kernel values for all possible substructure pairs in the incremental order of their sizes. The followings are the run-time in seconds of having calculated these Gram matrices with a laptop PC. Since K_0, K_1 and K_2 are calculated simultaneously, only the total run-time is displayed.

Kernel	`colon-cancer`	`cystic`	`leukemia`
K_0, K_1 and K_2	17.9 sec	31.6 sec	661.8 sec
K_3	3.8 sec	5.1 sec	261.8 sec

Table I
DEFINITION OF THE MAPPING TREE KERNELS: K_0, K_1, K_2 AND K_3

Kernel	Explanation
K_0	A weighted count of congruent substructure pairs. The weight is determined by λ^n with the decay factor λ and the size n of the substructure.
K_1	A weighted sum of the sizes of congruent substructure pairs. Hence, $\lambda^n n$ is summed up over the entire congruent substructure pairs.
K_2	A weighted sum of the squared sizes of congruent substructure pairs. Hence, $\lambda^n n^2$ is summed up over the entire congruent substructure pairs.
K_3	A tree kernel derived from Taï tree edit distance with the cost function $\gamma\langle x \to y\rangle = 1 - \delta_{x,y}$.

Table II
DEFINITION OF THE PARTITIONABLE KERNELS: $\kappa_0^{[*]}, \kappa_1^{[*]}, \kappa_2^{[*]}$ AND $\kappa_3^{[*]}$

Sub-kernel ($\lvert X'\rvert = \lvert Y'\rvert = n$)	$\kappa_i^{[0]}$	Quadratic Form
$\kappa_0^{[n]}(X', Y') = \prod_{i=1}^{n} \lambda\delta_{x_i,y_i}$	1	$\kappa_0^{[*]}(\boldsymbol{x}, \boldsymbol{y}) = \kappa_0^{[*]}(\boldsymbol{x}^{\mathsf{L}}, \boldsymbol{y}^{\mathsf{L}})\kappa_1^{[*]}(\boldsymbol{x}^{\mathsf{R}}, \boldsymbol{y}^{\mathsf{R}})$
$\kappa_1^{[n]}(X', Y') = \left(\prod_{i=1}^{n} \lambda\delta_{x_i,y_i}\right) \cdot \left(\sum_{i=1}^{n} \delta_{x_i,y_i}\right)$	0	$\kappa_1^{[*]}(\boldsymbol{x}, \boldsymbol{y}) = \kappa_0^{[*]}(\boldsymbol{x}^{\mathsf{L}}, \boldsymbol{y}^{\mathsf{L}})\kappa_1^{[*]}(\boldsymbol{x}^{\mathsf{R}}, \boldsymbol{y}^{\mathsf{R}}) + \kappa_1^{[*]}(\boldsymbol{x}^{\mathsf{L}}, \boldsymbol{y}^{\mathsf{L}})\kappa_0^{[*]}(\boldsymbol{x}^{\mathsf{R}}, \boldsymbol{y}^{\mathsf{R}})$
$\kappa_2^{[n]}(X', Y') = \left(\prod_{i=1}^{n} \lambda\delta_{x_i,y_i}\right) \cdot \left(\sum_{i=1}^{n} \delta_{x_i,y_i}\right)^2$	0	$\kappa_2^{[*]}(\boldsymbol{x}, \boldsymbol{y}) = \kappa_0^{[*]}(\boldsymbol{x}^{\mathsf{L}}, \boldsymbol{y}^{\mathsf{L}})\kappa_2^{[*]}(\boldsymbol{x}^{\mathsf{R}}, \boldsymbol{y}^{\mathsf{R}}) + \kappa_2^{[*]}(\boldsymbol{x}^{\mathsf{L}}, \boldsymbol{y}^{\mathsf{L}})\kappa_0^{[*]}(\boldsymbol{x}^{\mathsf{R}}, \boldsymbol{y}^{\mathsf{R}})$ $+2\kappa_1^{[*]}(\boldsymbol{x}^{\mathsf{L}}, \boldsymbol{y}^{\mathsf{L}})\kappa_1^{[*]}(\boldsymbol{x}^{\mathsf{R}}, \boldsymbol{y}^{\mathsf{R}})$
$\kappa_3^{[n]}(X', Y') = \prod_{i=1}^{n} e^{\lambda(\delta_{x_i,y_i}+1)}$	1	$\kappa_3^{[*]}(\boldsymbol{x}, \boldsymbol{y}) = \kappa_3^{[*]}(\boldsymbol{x}^{\mathsf{L}}, \boldsymbol{y}^{\mathsf{L}})\kappa_3^{[*]}(\boldsymbol{x}^{\mathsf{R}}, \boldsymbol{y}^{\mathsf{R}})$

As described in the above, the new mapping kernels K_0 and K_1, which are derived from partitionable kernels of higher hidden degree, can be efficiently computed.

With respect to the predictive performance of the kernels, Table III shows the averages of AUC-of-ROC-Curve, F-Score and Accuracy values obtained from the test over 10 training-test dataset pairs. "K_i (norm)" indicates that we ran the test with normalized kernel values. The figures in bold font face indicate the greatest values in their columns.

At a glance of the table, K_0, K_1 and K_2 appear more appropriate than the others for `colon-cancer`, `leukemia` and `cystic`, respectively. Also, K_3 is effective to `leukemia`, although it is not the best.

Table IV, V and VI are for further investigation of this observation, and show p-values of the paired T test of AUC-of-ROC-Curve, F-Score and Accuracy values over the ten runs of test performed for each combination of a tree kernel and a dataset. In the tables, the p-values smaller than 0.01 are highlighted.

Table IV and VI exhibit clear patterns.

`colon-cancer`: Superiority of the method of using normalized kernel values (that is, K_i (normal) for $i = 0, 1, 2, 3$) over the method of using kernel values as they are is evident. We can deny the corresponding null hypotheses with the sufficiently small significance level of 1%. By contrast, we cannot conclude that any of K_i (normal) is better than the others without committing risk of error.

`leukemia`: K_2, K_2 (norm) and K_3 show evident inferiority to the other kernels. Also, the p-values in Table VI indicate that the evidences we have obtained from the experiments are insufficient to deny the null hypotheses that the other five are mutually comparable.

On the other hand, the pattern that we can observe in Table V is less clear.

`cystic`: Without too much risk of error, for $i = 0, 1, 2$, we could conclude that K_3 is inferior to K_0, K_1 and K_2, no matter whether it is used with normalization. On the other hand, although we can perceive that the use of normalized K_i values always yields better results than the use of K_i values as they are, we have to use a large significance level to draw this conclusion.

Although the datasets and the mapping kernel examples used in the experiments were limited, at least, we can conclude that partitionable kernels of higher hidden degrees have potential to yield practically efficient mapping kernels with good predictive performance.

IV. CONCLUSION

We have introduced the notions of pretty decomposability and partitionable string kernels, and have proved that, if the mapping system is pretty decomposable and the sub-kernel is partitionable, there exists a set of recursive formulas to compute the resulting mapping kernel efficiently. Also, we have showed we could introduce an invariant called

Table III
AVERAGES OF MEASUREMENTS OVER 10 PAIRS OF TRAINING AND TEST DATASETS

	colon-cancer			cystic			leukemia		
Kernels	AUC	F Score	Accuracy	AUC	F Score	Accuracy	AUC	F Score	Accuracy
K_0	0.88977	0.87245	0.82090	0.78194	0.78008	0.735	0.95999	0.87345	0.93258
K_0 (norm)	0.93375	**0.91573**	**0.88955**	0.76622	0.73308	0.7025	0.97168	**0.88397**	0.93370
K_1	0.88003	0.83706	0.75373	**0.79329**	0.78226	0.74125	0.97090	0.87083	0.93034
K_1 (norm)	**0.93504**	0.91482	0.88806	0.77438	0.74043	0.7075	**0.97216**	0.88215	**0.93483**
K_2	0.88586	0.82541	0.72985	0.79322	**0.78700**	**0.74375**	0.93242	0.71954	0.87087
K_2 (norm)	0.93193	0.90653	0.87612	0.77846	0.74155	0.70875	0.83973	0.74488	0.85843
K_3	0.79819	0.77757	0.63731	0.71489	0.63478	0.530	0.90198	0.19990	0.74719
K_3 (norm)	0.91241	0.87466	0.82836	0.73285	0.70893	0.66625	0.96323	0.88284	**0.93483**

Table IV
P-VALUES OF PAIRED T-TEST FOR COLON-CANCER DATASET

Upper triangle: AUC, Lower triangle: F-Score

	K_0	K_0 (norm)	K_1	K_1 (norm)	K_2	K_2 (norm)	K_3	K_3 (norm)
K_0	–	**0.00035**	0.25434	**0.00017**	0.39651	**0.00111**	0.02296	0.38339
K_0 (norm)	0.04518	–	**0.00100**	0.47732	**0.00047**	0.61081	**0.00276**	0.29781
K_1	0.04480	**0.00957**	–	**0.00067**	0.56539	**0.00166**	0.03143	0.24360
K_1 (norm)	0.04449	0.80264	**0.00845**	–	**0.00025**	0.26707	**0.00291**	0.29424
K_2	0.06331	**0.00771**	0.63102	0.01003	–	**0.00169**	0.03872	0.32506
K_2 (norm)	0.04897	0.17572	**0.00647**	0.10286	**0.00903**	–	**0.00304**	0.36786
K_3	**0.00008**	**0.00001**	**0.00651**	**0.00001**	0.01892	**0.00000**	–	**0.00420**
K_3 (norm)	0.81095	0.02136	0.03258	0.02062	0.04436	0.01589	**0.00001**	–

Accuracy

	K_0	K_0 (norm)	K_1	K_1 (norm)	K_2	K_2 (norm)	K_3	K_3 (norm)
K_0	–	0.05912	0.04648	0.06063	0.05177	0.07338	**0.00005**	0.64072
K_0 (norm)	–	–	**0.00997**	0.72632	**0.00627**	0.12123	**0.00000**	0.02070
K_1	–	–	–	**0.00918**	0.60768	**0.00821**	**0.00602**	0.03157
K_1 (norm)	–	–	–	–	**0.00801**	0.08684	**0.00000**	0.02132
K_2	–	–	–	–	–	**0.00737**	0.01742	0.03049
K_2 (norm)	–	–	–	–	–	–	**0.00000**	0.01993
K_3	–	–	–	–	–	–	–	**0.00000**
K_3 (norm)	–	–	–	–	–	–	–	–

Table V
P-VALUES OF PAIRED T-TEST FOR CYSTIC DATASET

Upper Triangle: AUC; Lower Triangle: F-Score

	K_0	K_0 (norm)	K_1	K_1 (norm)	K_2	K_2 (norm)	K_3	K_3 (norm)
K_0	–	0.08847	0.05479	0.37188	0.09695	0.67457	0.11480	**0.00343**
K_0 (norm)	**0.00393**	–	0.02422	0.01846	0.04169	0.01002	0.22090	0.01934
K_1	0.73600	0.00937	–	0.08078	0.97035	0.14178	0.06449	**0.00123**
K_1 (norm)	0.02739	0.30529	0.3861	–	0.12245	0.11927	0.14794	0.00712
K_2	0.47997	0.01294	0.48571	0.04057	–	0.19847	0.06975	**0.00197**
K_2 (norm)	0.02797	0.30766	0.04290	0.75020	0.04035	–	0.11749	**0.00266**
K_3	0.05528	0.20194	0.05661	0.20022	0.04582	0.19149	–	0.59184
K_3 (norm)	**0.00155**	0.12043	**0.00299**	0.01360	**0.00333**	**0.00382**	0.36126	–

Accuracy

	K_0	K_0 (norm)	K_1	K_1 (norm)	K_2	K_2 (norm)	K_3	K_3 (norm)
K_0	–	0.03606	0.13818	0.09095	0.38156	0.11320	**0.00000**	**0.00164**
K_0 (norm)	–	–	0.03312	0.39936	0.05908	0.44016	**0.00005**	0.01619
K_1	–	–	–	0.07060	0.71634	0.08492	**0.00000**	**0.00155**
K_1 (norm)	–	–	–	–	0.10107	0.72631	**0.00009**	**0.00294**
K_2	–	–	–	–	–	0.11076	**0.00000**	**0.00372**
K_2 (norm)	–	–	–	–	–	–	**0.00006**	**0.00056**
K_3	–	–	–	–	–	–	–	**0.00039**
K_3 (norm)	–	–	–	–	–	–	–	–

Table VI
P-VALUES OF PAIRED T-TEST FOR LEUKEMIA DATASET

Upper Triangle: AUC; Lower Triangle: F-Score

	K_0	K_0 (norm)	K_1	K_1 (norm)	K_2	K_2 (norm)	K_3	K_3 (norm)
K_0	–	0.03938	0.02851	0.09467	**0.00033**	**0.00007**	**0.00002**	0.48853
K_0 (norm)	0.23557	–	0.89573	0.86997	**0.00000**	**0.00001**	**0.00000**	0.01608
K_1	0.64166	0.04423	–	0.84300	**0.00010**	**0.00009**	**0.00001**	0.20138
K_1 (norm)	0.29866	0.84837	0.20154	–	**0.00002**	**0.00002**	**0.00000**	0.12343
K_2	**0.00010**	**0.00002**	**0.00004**	**0.00009**	–	**0.00023**	**0.00024**	**0.00001**
K_2 (norm)	**0.00015**	**0.00002**	**0.00010**	**0.00011**	0.31038	–	**0.00481**	**0.00003**
K_3	**0.00004**	**0.00004**	**0.00005**	**0.00005**	**0.00039**	**0.00027**	–	**0.00000**
K_3 (norm)	0.47065	0.92085	0.27984	0.95203	**0.00005**	**0.00029**	**0.00005**	–

Accuracy

	K_0	K_0 (norm)	K_1	K_1 (norm)	K_2	K_2 (norm)	K_3	K_3 (norm)
K_0	–	0.81137	0.44333	0.59105	**0.00004**	**0.00008**	**0.00001**	0.72631
K_0 (norm)	–	–	0.34344	0.82265	**0.00000**	**0.00001**	**0.00001**	0.85892
K_1	–	–	–	0.30923	**0.00001**	**0.00003**	**0.00001**	0.39936
K_1 (norm)	–	–	–	–	**0.00006**	**0.00003**	**0.00002**	1.00000
K_2	–	–	–	–	–	0.27021	0.00045	**0.00001**
K_2 (norm)	–	–	–	–	–	–	**0.00199**	**0.00010**
K_3	–	–	–	–	–	–	–	**0.00002**
K_3 (norm)	–	–	–	–	–	–	–	–

hidden degree to partitionable kernels. The hidden degree of a partitionable kernel linearly affects the time complexity of the mapping kernel derived from the partitionable kernel. In addition, the string kernels of the product type turn out to be characterized as partitionable kernels of hidden degree one. Although we know that any partitionable kernel of hidden degree two can be reduced one of the three normalized forms, we know only little about partitionable kernels of hidden degree higher than two. Also, whether any partitionable kernel is character-wise symmetric is an open problem. If this were proved affirmatively, we would not have to pay much attention to converting mappings from structures to strings any more.

REFERENCES

[1] T. Gärtner, "A survey of kernels for structured data." *SIGKDD Explorations*, vol. 5, no. 1, pp. 49–58, 2003.

[2] D. Haussler, "Convolution kernels on discrete structures," Dept. of Computer Science, University of California at Santa Cruz, UCSC-CRL 99-10, 1999.

[3] K. Shin and T. Kuboyama, "Generalization of haussler's convolution kernel - mapping kernel and its application to tree kernels," *J. Comput. Sci. Technol*, vol. 25(5):, pp. 1040–1054, 2010.

[4] K. C. Taï, "The tree-to-tree correction problem," *JACM*, vol. 26, no. 3, pp. 422–433, Jul. 1979.

[5] M. Collins and N. Duffy, "Convolution kernels for natural language," in *Advances in Neural Information Processing Systems 14 [Neural Information Processing Systems: Natural and Synthetic, NIPS 2001].* MIT Press, 2001, pp. 625–632.

[6] K. Shin and T. Kuboyama, "A generalization of Haussler's convolution kernel - mapping kernel," in *ICML 2008*, 2008.

[7] E. D. Demaine, S. Mozes, B. Rossman, and O. Weimann, "An optimal decomposition algorithm for tree edit distance," in *The 34th International Colloquium on Automata, languages and Programming (ICALP)*, 2007.

[8] K. Shin, M. Cuturi, and T. Kuboyama, "Mapping kernels for trees," in *ICML 2011*, 2011.

[9] K. Hashimoto, S. Goto, S. Kawano, K. F. Aoki-Kinoshita, and N. Ueda, "Kegg as a glycome informatics resource," *Glycobiology*, vol. 16, pp. 63R – 70R, 2006.