



Deposited via The University of York.

White Rose Research Online URL for this paper:

<https://eprints.whiterose.ac.uk/id/eprint/88205/>

Version: Submitted Version

---

**Proceedings Paper:**

Bai, Lu, Zhang, Zhihong and Hancock, Edwin R (2015) A Graph Kernel based on Jensen-Shannon Representation. In: International Joint Conference on Artificial Intelligence (IJCAI).

---

**Reuse**

Items deposited in White Rose Research Online are protected by copyright, with all rights reserved unless indicated otherwise. They may be downloaded and/or printed for private study, or other acts as permitted by national copyright laws. The publisher or other rights holders may allow further reproduction and re-use of the full text version. This is indicated by the licence information on the White Rose Research Online record for the item.

**Takedown**

If you consider content in White Rose Research Online to be in breach of UK law, please notify us by emailing [eprints@whiterose.ac.uk](mailto:eprints@whiterose.ac.uk) including the URL of the record and the reason for the withdrawal request.

# A Graph Kernel Based on the Jensen-Shannon Representation Alignment\*

Lu Bai<sup>1,2†</sup>, Zhihong Zhang<sup>3‡</sup>, Chaoyan Wang<sup>4</sup>, Xiao Bai<sup>5</sup>, Edwin R. Hancock<sup>2</sup>

<sup>1</sup>School of Information, Central University of Finance and Economics, Beijing, China

<sup>2</sup>Department of Computer Science, University of York, York, UK

<sup>3</sup>Software School, Xiamen University, Xiamen, Fujian, China

<sup>4</sup>School of Contemporary Chinese Studies, University of Nottingham, Nottingham, UK

<sup>5</sup>School of Computer Science and Engineering, Beihang University, Beijing, China

## Abstract

In this paper, we develop a novel graph kernel by aligning the Jensen-Shannon (JS) representations of vertices. We commence by describing how to compute the JS representation of a vertex by measuring the JS divergence (JSD) between the corresponding  $h$ -layer depth-based (DB) representations developed in [Bai *et al.*, 2014a]. By aligning JS representations of vertices, we identify the correspondence between the vertices of two graphs and this allows us to construct a matching-based graph kernel. Unlike existing R-convolution kernels [Haussler, 1999] that roughly record the isomorphism information between any pair of substructures under a type of graph decomposition, the new kernel can be seen as an **aligned subgraph kernel** that incorporates explicit local correspondences of substructures (*i.e.*, *the local information graphs* [Dehmer and Mowshowitz, 2011]) into the process of kernelization through the JS representation alignment. The new kernel thus addresses the drawback of neglecting the relative locations between substructures that arises in the R-convolution kernels. Experiments demonstrate that our kernel can easily outperform state-of-the-art graph kernels in terms of the classification accuracies.

## 1 Introduction

There have been many successful attempts to classify or cluster graphs using graph kernels [Gärtner *et al.*, 2003; Jebara *et al.*, 2004; Barra and Biasotti, 2013; Harchaoui and Bach, 2007]. The main advantage of using graph kernels is that they characterize graph features in a high dimensional space and thus better preserve graph structures. A graph kernel is usually defined in terms of a similarity measure between graphs. Most of the recently developed graph kernels are instances of the family of R-convolution kernels proposed

by Haussler [Haussler, 1999], which provide a generic way of defining graph kernels by comparing all pairs of isomorphic substructures under decomposition, and a new decomposition will result in a new graph kernel. Generally speaking, the R-convolution kernels can be categorized into the following classes, namely graph kernels based on comparing all pairs of a) walks (e.g., the random walk kernel [Jebara *et al.*, 2004]), b) paths (e.g., the shortest path kernel [Borgwardt and Kriegel, 2005]), c) cycles (e.g., the backtracless kernel from the cycles identified by the Ihara zeta function [Aziz *et al.*, 2013]), and d) subgraph or subtree structures (e.g., the subgraph or subtree kernel [Costa and Grave, 2010; Shervashidze *et al.*, 2011]).

One drawback arising in the R-convolution kernels is that they neglect the relative locations of substructures. This occurs when an R-convolution kernel adds an unit value to the kernel function by roughly identifying a pair of isomorphic substructures. As a result, the R-convolution kernels cannot establish reliable structural correspondences between the substructures. This drawback limits the precise kernel-based similarity measure for graphs.

To overcome the shortcomings of existing R-convolution kernels, we propose a novel matching kernel by aligning JS representations of vertices, which are computed based on the JSD measure [Bai *et al.*, 2012; Bai and Hancock, 2013] between DB representations [Bai and Hancock, 2014] of graphs. The main advantages of using DB representations and the JSD measure are twofold. First, in the literature [Crutchfield and Shalizi, 1999; Escolano *et al.*, 2012; Bai and Hancock, 2014], DB representations of graphs are powerful tools for characterizing graphs in terms of complexity measures and reflect rich depth characteristics of graphs. Second, in the literature [Bai *et al.*, 2012; Bai, 2014], the JSD measure of graphs can not only reflect the information theoretic (dis)similarities between entropies of graphs but can also be efficiently computed for a pair of large graphs. As a result, the DB representation and the JSD measure provide us an elegant way of defining new effective graph kernels.

To compute the new matching kernel, for each graph under comparison, we commence by computing the  $h$ -layer DB representation of each vertex, that has been previously introduced in [Bai *et al.*, 2014a]. Moreover, we determine a  $m$ -sphere (*i.e.*, a vertex set) for each vertex by selecting the vertices that have the shortest path length  $m$  to the vertex.

\*This work is supported by the National Natural Science Foundation of China (61402389 and 61272398). Edwin R. Hancock is supported by a Royal Society Wolfson Research Merit Award.

†Primary Author: bailu69@hotmail.com; lu@cs.york.ac.uk

‡Corresponding Author: zhihong@xmu.edu.cn.

We compute a new  $m$ -layer JS representation for each vertex by measuring the JSD between the  $h$ -layer DB representations of the vertex and the vertices from its  $m$ -sphere. The  $m$ -layer JS representation rooted at a vertex not only encapsulates a high-dimensional entropy-based depth information for the vertex, but also reflects the co-relation between the vertex and its  $m$ -sphere vertices (i.e., the JS representation reflects richer characteristics than the original DB representation). Based on the new JS representations for two graphs we develop a new vertex matching strategy by aligning the JS representations. Finally, we compute the new kernel, namely the JS matching kernel, for graphs by counting the number of matched vertex pairs. We theoretically show the relationship between our kernel and the classical all subgraph kernel and explain the reason for the effectiveness of our kernel. Our JS matching kernel can be seen as an **aligned subgraph kernel** that counts the number of aligned isomorphic subgraph (i.e., the local information graph [Dehmer and Mowshowitz, 2011]) pairs which are correspond by the corresponding aligned JS representations. Our kernel thus overcomes the mentioned shortcoming of neglecting the structural correspondence information between substructures that arises in R-convolution kernels. Furthermore, compared to the DB matching kernel that is computed by aligning the  $h$ -layer DB representations [Bai, 2014; Bai *et al.*, 2014a], our kernel not only reflects richer characteristics but also identifies more pairs of isomorphic subgraphs that encapsulate the structural correspondence information. We empirically demonstrate the effectiveness of our new kernel on graphs from computer vision datasets.

## 2 Vertex Matching using JS Representations

In this section, we define a JS representation for a vertex and a vertex matching method by aligning the representations.

### 2.1 The JSD Measure for Graphs

In this work, we require the JSD measure for graphs. In mutual information, the JSD is a dissimilarity measure for probability distributions in terms of the entropy difference associated with the distributions [Martins *et al.*, 2009]. In [Bai and Hancock, 2013; Bai *et al.*, 2012], Bai *et al.* have extended the JSD measure to graphs for the purpose of computing the JSD based information theoretic graph kernels. In their work, the JSD between a pair of graphs is computed by measuring the entropy difference between the entropies of the graphs and those of a composite graph (e.g., the disjoint union graph [Bai *et al.*, 2012]) formed by the graphs. In this subsection, we generalize their work in [Bai *et al.*, 2012] and give the concept of measuring the JSD for a set of graphs. Let  $\mathbf{G} = \{G_n | n = 1, 2, \dots, N\}$  denote a set of  $N$  graphs, and  $G_n(V_n, E_n)$  is a sample graph in  $\mathbf{G}$  with vertex set  $V_n$  and edge set  $E_n$ . The JSD measure  $\mathcal{D}$  for the set of graphs  $\mathbf{G}$  is

$$\mathcal{D}(\mathbf{G}) = H_S(G_{DU}) - \frac{1}{N} \sum_{n=1}^N H_S(G_n), \quad (1)$$

where  $H_S(G_n)$  is the Shannon entropy for  $G_n$  associated with steady state random walks [Bai and Hancock, 2013] and

is defined as

$$H_S(G_n) = - \sum_{v \in V_n} P_{G_n}(v) \log P_{G_n}(v), \quad (2)$$

where  $P_{G_n}(v) = D_n(v, v) / \sum_{u \in V} D_n(u, u)$  is the probability of the steady state random walk visiting the vertex  $v \in V_n$ , and  $D$  is the diagonal degree matrix of  $G_n$ . Moreover, in Eq.(1)  $G_{DU}$  is the disjoint union graph formed by all the graphs in  $\mathbf{G}$ , and  $H_S(G_{DU})$  is the Shannon entropy of the union graph. Based on the definition in [Köner, 1971], the entropy of the disjoint union graph is defined as

$$H_S(G_{DU}) = \frac{\sum_{n=1}^N |V_n| H_S(G_n)}{\sum_{n=1}^N |V_n|}, \quad (3)$$

where  $|V_n|$  is the number of vertices of  $G_n$ . Eq.(1) and Eq.(3) indicate that the JSD measure for a pair of graphs can be directly computed from their vertex numbers and entropy values. Thus, the JSD measure can be efficiently computed.

### 2.2 The JS Representation through the JSD

In this subsection, we compute a  $m$ -layer JS representation around a vertex for a graph. To commence, we first review the concept of the  $h$ -layer DB representation around a vertex. This has been previously introduced by Bai *et al.* [Bai *et al.*, 2014a], by generalizing the DB complexity trace around the centroid vertex [Bai and Hancock, 2014]. For an undirected graph  $G(V, E)$  and a vertex  $v \in V$ , let a vertex set  $N_v^K$  be defined as  $N_v^K = \{u \in V \mid S_G(v, u) \leq K\}$ , where  $S_G$  is the shortest path matrix of  $G$  and  $S_G(v, u)$  is the shortest path length between  $v$  and  $u$ . For  $G$ , the  $K$ -layer expansion subgraph  $\mathcal{G}_v^K(\mathcal{V}_v^K; \mathcal{E}_v^K)$  around  $v$  is

$$\begin{cases} \mathcal{V}_v^K = \{u \in N_v^K\}; \\ \mathcal{E}_v^K = \{u, v \in N_v^K, (u, v) \in E\}. \end{cases} \quad (4)$$

For the graph  $G$ , the  $h$ -layer DB representation around  $v$  is

$$DB_G^h(v) = [H_S(\mathcal{G}_v^1), \dots, H_S(\mathcal{G}_v^K), \dots, H_S(\mathcal{G}_v^h)]^T, \quad (5)$$

where ( $K \leq h$ ),  $\mathcal{G}_v^K$  is the  $K$ -layer expansion subgraph around  $v$ , and  $H_S(\mathcal{G}_v^K)$  is the Shannon entropy of  $\mathcal{G}_v^K$  and is defined in Eq.(2). Note that, if  $L_{max}$  is the greatest length of the shortest paths from  $v$  to the remaining vertices and  $K \geq L_{max}$ , the  $K$ -layer expansion subgraph is  $G$  itself.

Clearly, the  $h$ -layer DB representation  $DB_G^h(v)$  reflects an entropy-based information content flow through the family of  $K$ -layer expansion subgraphs rooted at  $v$ , and thus can be seen as a vectorial representation of  $v$ .

**The  $m$ -layer JS representation:** For the graph  $G(V, E)$  and the vertex  $v \in V$ , we define the  $m$ -sphere (i.e., a vertex set) around  $v$  as  $\hat{N}_v^m = \{u \in V \mid S_G(v, u) = m\}$  (note that,  $\hat{N}_v^m$  is different from  $N_v^K$ , even if  $m = K$ ), based on the definition in [Dehmer and Mowshowitz, 2011]. The JSD for the  $h$ -layer DB representations of  $v$  and the vertices in  $\hat{N}_v^K$  is

$$\mathcal{D}_G^{(m,h)}(v) = [\mathcal{D}(\mathbf{G}_{\hat{N}_v^m}^1), \dots, \mathcal{D}(\mathbf{G}_{\hat{N}_v^m}^K), \dots, \mathcal{D}(\mathbf{G}_{\hat{N}_v^m}^h)]^T, \quad (6)$$

where  $\mathbf{G}_{\hat{N}_v^m}^K$  is a graph set and consists of the  $K$ -layer expansion subgraphs around  $v$  and the vertices in  $\hat{N}_v^m$ . The  $m$ -layer JS representation around the vertex  $v$  is

$$J_G^{(m;h)}(v) = \mathcal{D}_G^{(m;h)}(v) + DB_G^h(v). \quad (7)$$

Note that, the dimension of  $J_G^m(v)$  is equal to that of  $DB_G^h(v)$ , i.e.,  $J_G^m(v)$  is a  $h$  dimensional vector. This can be observed from Eq.(5), Eq.(6) and Eq.(7).  $\square$

**Discussions:** Compared to the  $h$ -layer DB representation  $DB_G^h(v)$ , the  $m$ -layer JS representation  $J_G^{(m;h)}(v)$  not only reflects the information content (i.e., the entropy) flow rooted at the vertex  $v$  relying on  $DB_G^h(v)$ , but also encapsulates the entropy-based dissimilarity information (or the relationship) between the  $h$ -layer DB representation of  $v$  and that of the vertices in  $\hat{N}_v^m$  in terms of the JSD measure  $\mathcal{D}_G^{(m;h)}(v)$ . In other word, for each vertex the  $m$ -layer JS representation reflects richer characteristics than its original  $h$ -layer DB representation. This can be observed from Eq.(7). Moreover, based on the definition proposed by Dehmer and Mowshowitz in [Dehmer and Mowshowitz, 2011], the shortest paths departing from  $v$  to the vertices in  $\hat{N}_v^m$  can be used to form a local information graph  $L_G(v, m)$  rooted at  $v$ . Since  $L_G(v, m)$  is encompassed or determined by the  $m$ -sphere  $\hat{N}_v^m$  around  $v$ , the vertex  $v$  and its  $m$ -sphere play a crucial role of generating  $L_G(v, m)$ . As a result, the  $m$ -layer JS representation  $J_G^{(m;h)}(v)$  can also be seen as a vectorial signature for the local information graph  $L_G(v, m)$ . Details of local information graphs can be found in [Dehmer and Mowshowitz, 2011].

### 2.3 Vertex Matching from the JS Representations

We propose a new vertex matching method, namely the JS matching method, for a pair of graphs by aligning their JS representations of vertices. Our matching method is similar to that previously introduced by Scott et al. in [Scott and Longuet-Higgins, 1991] for point set matching, that computes an affinity matrix in terms of the distances between points. For a pair of graphs  $G_p(V_p, E_p)$  and  $G_q(V_q, E_q)$ , we use the  $m$ -layer JS representations  $J_{G_p}^{(m;h)}(v_i)$  and  $J_{G_q}^{(m;h)}(u_j)$ , that are computed from the corresponding  $h$ -layer DB representations, as the point coordinates for the vertices  $v_i \in V_p$  and  $u_j \in V_q$ , respectively. For  $G_p$  and  $G_q$ , we compute the Euclidean distance between  $J_{G_p}^{(m;h)}(v_i)$  and  $J_{G_q}^{(m;h)}(u_j)$  as the element  $R^{(m;h)}(i, j)$  of their affinity matrix  $R^{(m;h)}$ , and  $R^{(m;h)}(i, j)$  is defined as

$$R^{(m;h)}(i, j) = \|J_{G_p}^{(m;h)}(v_i) - J_{G_q}^{(m;h)}(u_j)\|_2. \quad (8)$$

where  $R^{(m;h)}$  is a  $|V_p| \times |V_q|$  matrix, and the parameters  $m$  and  $h$  indicate that the affinity matrix is computed from the  $m$ -layer JS representation over the  $h$ -layer DB representation.  $R^{(m;h)}(i, j)$  represents the distance or dissimilarity between the vertex  $v_i$  in  $G_p$  and the vertex  $u_j$  in  $G_q$ . Furthermore, for the affinity matrix  $R^{(m;h)}$ , the rows index the vertices of  $G_p$ , and the columns index the vertices of  $G_q$ .

If  $R^{(m;h)}(i, j)$  is the smallest element simultaneously in row  $i$  and in column  $j$ , moreover  $|\hat{N}_{v_i}^m| = |\hat{N}_{u_j}^m| \neq 0$  (i.e., the vertex number of the  $m$ -sphere around  $v_i \in V_p$  is equal to that around  $u_j \in V_q$ , and these vertex numbers also cannot be 0), there should be a one-to-one correspondence between the vertex  $v_i$  of  $G_p$  and the vertex  $u_j$  of  $G_q$ , i.e.,  $v_i$  and  $u_j$  are matched. We record the state of correspondence using the correspondence matrix  $C^{(m;h)} \in \{0, 1\}^{|V_p| \times |V_q|}$  that satisfies

$$C^{(m;h)}(i, j) = \begin{cases} 1 & \text{if } R(i, j) \text{ is the smallest element,} \\ & \text{both in row } i \text{ and in column } j, \\ & \text{and } |\hat{N}_{v_i}^m| = |\hat{N}_{u_j}^m| \neq 0; \\ 0 & \text{otherwise.} \end{cases} \quad (9)$$

Eq.(9) indicates that if  $C^{(m;h)}(i, j) = 1$ , the vertices  $v_i$  and  $u_j$  are matched. Moreover, the condition  $|\hat{N}_{v_i}^m| = |\hat{N}_{u_j}^m| \neq 0$  for Eq.(9) guarantees that both of the  $m$ -spheres  $|\hat{N}_{v_i}^m|$  and  $|\hat{N}_{u_j}^m|$  are not empty vertex sets.

Note that, similar to the DB matching previously introduced by Bai et al. [Bai et al., 2014a], for a pair of graphs a vertex from a graph may have more than one matched vertices from the other graph. In our work, we propose to assign a vertex at most one matched vertex. One way to achieve this is to update the matrix  $C^{(m;h)}$  by adopting the Hungarian algorithm [Munkres, 1957] that can solve the assignment problem, following the strategy proposed in [Bai et al., 2014a]. Here, the matrix  $C^{(m;h)} \in \{0, 1\}^{|V_p| \times |V_q|}$  can be seen as the incidence matrix of a bipartite graph  $G_{pq}(V_p, V_q, E_{pq})$ , where  $V_p$  and  $V_q$  are the two sets of partition parts and  $E_{pq}$  is the edge set. By performing the Hungarian algorithm on the matrix  $C^{(m;h)}$ , we can assign each vertex from the partition part  $V_p$  or  $V_q$  at most one matched vertex from the other partition part  $V_q$  or  $V_p$ . Unfortunately, the Hungarian algorithm usually requires extra expansive computation and thus may lead to computational inefficiency for the JS matching. To address the inefficiency, an alternative way or strategy is to randomly assign each vertex an unique matched vertex through the correspondence matrix  $C^{(m;h)}$ . In other words, for the correspondence matrix  $C^{(m;h)}$ , from the first row and the first column, we will set each evaluating element of  $C^{(m;h)}$  as 0 if there has been an existing element that is 1 either in the same row or the same column. Based on our evaluations, this strategy will not influence the effectiveness of our resulting kernel in Section 3, and the kernel using the strategy will be more efficient than that using the Hungarian algorithm.

## 3 A Graph Kernel from the JS Matching

In this section, we propose a new graph kernel from the JS vertex matching by aligning the JS representations of vertices.

### 3.1 The Jensen-Shannon Matching Kernel

**Definition (The Jensen-Shannon matching kernel):** Consider a pair of graphs as  $G_p(V_p, E_p)$  and  $G_q(V_q, E_q)$ . Based on the definition of JS matching introduced in Section 2.3, we commence by computing the correspondence matrix  $C^{(m;h)}$ . The JS matching kernel  $k_{JSM}^{(M;h)}$ , which aligns the  $m$ -layer JS

representations, for the graphs is

$$k_{JSM}^{(M;h)}(G_p, G_q) = \sum_{m=1}^M \sum_{i=1}^{|V_p|} \sum_{j=1}^{|V_q|} C^{(m;h)}(i, j), \quad (10)$$

where  $M$  is the greatest value of the parameter  $m$  (i.e.,  $m$  varies from 1 to  $M$ ). Eq.(10) indicates that  $k_{JSM}^{(M;h)}(G_p, G_q)$  counts the number of matched vertex pairs between  $G_p$  and  $G_q$  over the  $M$  correspondence matrices  $C^{(m;h)}$ .  $\square$

**Lemma.** *The matching kernel  $k_{JSM}^{(M;h)}$  is positive definite (pd).*

**Proof.** Intuitively, the proposed JS matching kernel is **pd** because it counts pairs of matched vertices (i.e., the smallest subgraphs) over the  $M$  correspondence matrices  $C^{(m;h)}$ . More formally, let the base counting kernel  $k^m$  be a function counting pairs of matched vertices between the graphs  $G_p$  and  $G_q$  from the correspondence matrix  $C^{(m;h)}$ , and

$$k^m(G_p, G_q) = \sum_{v_i \in V_p} \sum_{u_j \in V_q} \delta(v_i, u_j), \quad (11)$$

where

$$\delta(v_i, u_j) = \begin{cases} 1 & \text{if } C(i, j) = 1; \\ 0 & \text{otherwise.} \end{cases} \quad (12)$$

where  $\delta$  is the Dirac kernel, that is, it is 1 if the arguments are equal and 0 otherwise (i.e., it is 1 if a pair of vertices are matched and 0 otherwise). From the base counting kernel  $k^m$ , the JS matching kernel  $k_{JSM}^{(M;h)}$  can be re-written as

$$k_{JSM}^{(M;h)}(G_p, G_q) = \sum_{m=1}^M k^m(G_p, G_q). \quad (13)$$

Eq.(11) indicates that the base counting kernel  $k^m$  is the sum of **pd** Dirac kernels and is thus **pd**. As a result, the kernel  $k_{JSM}^{(M;h)}$  summing the positive definite kernel  $k^m$  is **pd**.  $\blacksquare$

### 3.2 Relation to the All Subgraph Kernel

The JS matching kernel can be re-defined in another manner that elucidates its advantage and effectiveness, compared to the all subgraph (AS) kernel. To commence, we review the definition of the AS kernel introduced in [Borgwardts, 2007]. Let  $G_p(V_p, E_p)$  and  $G_q(V_q, E_q)$  be a pair of graphs for the kernel computation. The AS kernel is defined as

$$k_{AS}(G_p, G_q) = \sum_{S_p \subseteq G_p} \sum_{S_q \subseteq G_q} k_I(S_p, S_q), \quad (14)$$

where

$$k_I(S_p, S_q) = \begin{cases} 1 & \text{if } S_p \simeq S_q, \\ 0 & \text{otherwise.} \end{cases} \quad (15)$$

Below, we re-define the JS matching kernel in a manner that is similar to the AS kernel. Based on the definition in Section 2.2, the  $m$ -layer JS representations around a vertex  $v \in V_p$  of  $G_p(V_p, E_p)$  and a vertex  $u \in V_q$  of  $G_q(V_q, E_q)$  are  $J_{G_p}^{(m;h)}(v) = \mathcal{D}_{G_p}^m(v) + DB_{G_p}^h(v)$  and  $J_{G_q}^{(m;h)}(u) = \mathcal{D}_{G_q}^m(u) + DB_{G_q}^h(u)$ , respectively. According to the discussions in Section 2.2, each  $m$ -layer JS representation around

a vertex can be seen as a vectorial signature of a local information graph, which is determined by the vertex and the  $m$ -sphere around the vertex. Based on the JS matching defined in Section 2.3, if the vertices  $v$  and  $u$  are matched, the two  $m$ -layer JS representations around the two vertices are close together in a corresponding principle space. Thus, the corresponding local information graphs  $L_{G_p}(v, m)$  around  $v \in V_p$  and  $L_{G_q}(u, m)$  around  $u \in V_q$  can be seen as approximate isomorphism, i.e.,  $L_{G_p}(v, m) \simeq L_{G_q}(u, m)$ . As a result, the JS matching kernel  $k_{JSM}^{(M;h)}$  can be re-written as

$$k_{JSM}^{(M;h)}(G_p, G_q) = \sum_{S_p \subseteq G_p} \sum_{S_q \subseteq G_q} k_I(S_p, S_q), \quad (16)$$

where

$$k_I(S_p, S_q) = \begin{cases} 1 & \text{if } S_p = L_{G_p}(v, m) \text{ and } L_{G_q}(u, m), \\ & v \text{ and } u \text{ are matched,} \\ & \text{and } |\hat{N}_v^m| = |\hat{N}_u^m| \neq 0; \\ 0 & \text{otherwise.} \end{cases} \quad (17)$$

Here, the condition  $|\hat{N}_v^m| = |\hat{N}_u^m| \neq 0$  of Eq.(17) guarantees that the  $m$ -spheres  $\hat{N}_v^m$  and  $\hat{N}_u^m$  exist, i.e., the local information graphs  $L_{G_p}(v, m)$  and  $L_{G_q}(u, m)$  that are determined by the  $m$ -spheres exist. This is because an empty  $m$ -sphere cannot form a local information, more details can be found in [Dehmer and Mowshowitz, 2011].

**Discussions:** Through Eq.(14) and Eq.(17), we observe that both the kernels  $k_{AS}$  and  $k_{JSM}^{(M;h)}$  need to identify any pair of isomorphic subgraphs. For  $k_{AS}$  and  $k_{JSM}^{(M;h)}$ , each pair of isomorphic subgraphs pair will add an unit value to the kernel function, i.e., both the AS kernel and our JS matching kernel count the number of isomorphic subgraph pairs. However, we also observe that there are two significant differences between the AS kernel and the JS matching kernel. First, for the JS matching kernel, only the subgraphs (i.e., the local information graphs) around a pair of matched vertices that are determined by the vertices and the  $m$ -spheres around the vertices are evaluated with respect to be isomorphic. By contrast, for the AS kernel, any pair of subgraphs are evaluated for identifying the isomorphism. As a result, the JS matching kernel overcomes the NP-hard problem of measuring all possible pairs of subgraphs that arises in the AS kernel. Second, for the JS matching kernel, any pair of isomorphic local information graphs are identified by a pair of matched vertices and the  $m$ -spheres around the vertices. Thus, there is a locational correspondence between the isomorphic local information graphs (i.e., the corresponding subgraphs) with respect to the global graphs. By contrast, a pair of subgraphs having no location correspondence may also be seen as isomorphism by the AS kernel. The above observations indicate that our JS matching kernel is essentially an **aligned subgraph kernel** that counts the number of aligned isomorphic local information graph pairs, which are corresponded by the corresponding JS representations. Our kernel thus overcomes the shortcoming of neglecting the relative locations between substructures arising in the R-convolution kernels.

### 3.3 Discussions and Related Work

Clearly, the JS matching kernel is related to the DB representation defined in [Bai and Hancock, 2014]. However, there are two significant differences. First, the DB representation is computed by measuring the entropies of subgraphs rooted at the centroid vertex. The centroid vertex is identified by evaluating the minimum shortest path length variance to the remaining vertices. By contrast, in our work, we first compute the  $h$ -layer DB representation rooted at each vertex, and then compute the resulting  $m$ -layer JS representation. For a vertex, its  $m$ -layer JS representation is computed by summing its DB representation and the JSD measure between its DB representation and that of the vertices from its  $m$ -sphere. Second, in [Bai and Hancock, 2014] the DB representation from the centroid vertex is a vectorial signature of a graph, i.e., it can be seen as an embedding vector for the graph. Embedding a graph into a vector tends to approximate the structural correlations in a low dimensional space, and thus leads to information loss. By contrast, the JS matching kernel aligning the  $m$ -layer JS representation represents graphs in a high dimensional space and thus better preserves graph structures.

On the other hand, the DB matching kernel developed in [Bai *et al.*, 2014a] is also related to the DB representation in [Bai and Hancock, 2014]. Moreover, similar to our JS matching kernel, the DB matching kernel can also better preserve graph structures by kernelizing the DB representation. However, unlike the JS matching kernel, this kernel is computed by aligning the  $h$ -layer DB representation rather than aligning the  $m$ -layer JS representation. According to the statement in Section 2.2, the  $m$ -layer JS representation for a vertex reflects richer characteristics than its original  $h$ -layer DB representation. As a result, the JS matching kernel can capture more information for graphs than the DB matching kernel. Moreover, Bai [Bai, 2014] also demonstrates the relationship between the DB matching kernel and the AS kernel. The DB matching kernel can also be re-defined as a manner that is similar to the AS kernel, i.e., the DB matching kernel can be seen as a subgraph kernel that counts the number of approximately isomorphic  $h$ -layer (i.e.,  $K = h$ ) expansion subgraph pairs. Each pair of isomorphic  $h$ -layer expansion subgraphs are identified by a pair of matched vertices from the DB matching. Thus, similar to our JS matching kernel, the DB matching kernel can also reflect the locational correspondence information between each pair of identified isomorphic subgraphs. Unfortunately, the identified isomorphic subgraph pair number of the DB matching kernel is less than that of the JS matching kernel. For a pair of graphs each having  $x$  vertices, the DB matching kernel can only identify  $x$  pairs of isomorphic subgraphs at most. By contrast, the JS matching kernel can identify  $Mx$  pairs of isomorphic subgraphs at most. This again demonstrates that our JS matching kernel captures more information than the DB matching kernel.

Finally, like the JS graph kernel [Bai and Hancock, 2013], our kernel is also related to the JSD. However, the JS graph kernel only reflects the global similarity of graphs in terms of the JSD measure between a pair of global graphs, and lacks the interior topological information of graphs. By contrast, our JS matching kernel can reflect rich vertex correspondence information from the JS representations computed by measur-

ing the JSD between corresponding DB representations.

## 4 Experimental Results

We demonstrate the performance of our new kernel on three standard graph datasets from computer vision databases. The reason of using computer vision datasets is that many computer vision applications usually require the correspondence information between pairwise feature points that are abstracted from images or 3D shapes, for the objective of similarity measure. For an instance, one has two graphs abstracted from two digital images both containing the same object, based on different viewpoints. Here, each vertex represents a feature point. Identifying the correspondence information between pairwise vertices or substructures from the identical region is our concern, and can provide us an elegant way of reflecting precise similarity between the images or shapes. As a result, the new matching kernel can easily indicate its main advantage of identifying the correspondence information, on computer vision datasets. The advantage is unavailable for most existing graph kernels from R-convolution.

**BAR31, BSPHERE31 and GEOD31:** The SHREC 3D Shape database consists of 15 classes and 20 individuals per class, that is 300 shapes [Biasotti *et al.*, 2003]. This is a usual benchmark in 3D shape recognition. From the SHREC 3D Shape database, we establish three graph datasets named BAR31, BSPHERE31 and GEOD31 datasets through three mapping functions. These functions are a) ERG barycenter: distance from the center of mass/barycenter, b) ERG bsphere: distance from the center of the sphere that circumscribes the object, and c) ERG integral geodesic: the average of the geodesic distances to the all other points. The number of maximum, minimum and average vertices for the three datasets are a) 220, 41 and 95.42 (for BAR31), b) 227, 43 and 99.83 (for BSPHERE31), and c) 380, 29 and 57.42 (for GEOD31), respectively.

### 4.1 Experiments on Graph Datasets

**Experimental Setup:** First, we evaluate the performance of our JS matching kernel (JSMK) on graph classification problems. We also compare our kernel with several alternative state-of-the-art graph kernels. These graph kernels include 1) the DB matching kernel (DBMK) [Bai, 2014; Bai *et al.*, 2014a], 2) the Weisfeiler-Lehman subtree kernel (WLSK) [Shervashidze *et al.*, 2011], 3) the shortest path graph kernel (SPGK) [Borgwardt and Kriegel, 2005], 4) the graphlet count graph kernel [Shervashidze *et al.*, 2009] with graphlet of size 4 (GCGK) [Shervashidze *et al.*, 2009], 5) the un-aligned quantum Jensen-Shannon kernel (UQJS) [Bai *et al.*, 2015], 6) the Jensen-Shannon graph kernel (JSGK) [Bai and Hancock, 2013], and 7) the Jensen-Tsallis  $q$ -difference kernel (JTQK) [Bai *et al.*, 2014b] associated with  $q = 2$ . For our JS matching kernel  $k_{JSM}^{(M:h)}$ , we set the  $h$  as 10 and the greatest value of  $m$  as 40 (i.e.,  $M = 40$ ). For the WLSK kernel and the JTQK kernel, we set the highest dimension (i.e., the highest height of subtrees) of the Weisfeiler-Lehman isomorphism (for the WLSK kernel) and the tree-index method (for the JTQK kernel) as 10. For the DBMK kernel, we set the highest layer of the required DB representation as 10. For

each kernel, we compute the kernel matrix on each graph dataset. We perform 10-fold cross-validation using the C-Support Vector Machine (C-SVM) Classification to compute the classification accuracies, using LIBSVM [Chang and Lin, 2011]. We use nine samples for training and one for testing. All the C-SVMs were performed along with their parameters optimized on each dataset. We repeat the experiment 10 times. We report the average classification accuracies and standard errors for each kernel in Table.1.

**Experimental Results:** In terms of the classification accuracies, our JSMK kernel can easily outperform all the alternative graph kernels on any dataset. The classification accuracies of our JSMK kernel are obviously higher than those of all the alternative kernels. The reasons for the effectiveness are threefold. First, compared to the WLSK, SPGK, GCGK and JTQK kernels that require decomposing graphs into substructures, our JSMK kernel can establish the substructure location correspondence which is not considered in these kernels. Second, compared to the JSGK and QJSK kernels that rely on the similarity measure between global graphs in terms of the classical or quantum JSD, our JSMK kernel can identify the correspondence information between both the vertices and the substructures, and can thus reflect richer interior topological characteristics of graphs. By contrast, the JSGK and QJSK kernels can only reflect the global similarity information between graphs. Third, compared to the DBMK kernel that can also reflect the correspondence information between substructures, our JSMK kernel can identify more pairs of aligned isomorphic substructures. Moreover, as we have stated in Section 3.3, the  $m$ -layer JS representation can reflect richer characteristics than the  $h$ -layer DB representation. As a result, the JSMK kernel using the JS representation can capture more information for graphs than the DBMK kernel using the DB representation.

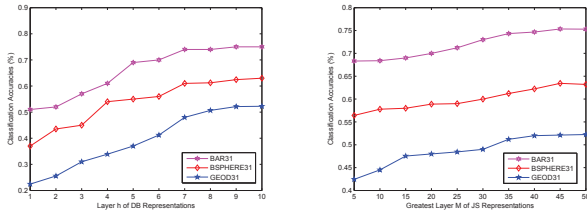


Figure 1: The Accuracy with Different Parameters  $h$  and  $M$ .

**Comparisons with Increasing  $h$  and  $M$ :** We explore the performance of our JSMK kernel on graph datasets with increasing  $h$  (i.e.,  $h = 1, 2, \dots, 10$  when  $M = 40$ ) and  $M$  (i.e.,  $M = 5, 10, 15 \dots, 50$  when  $h = 10$ ). We report the results in Fig.1. In each subfigure, the x-axis gives the varying of  $h$  or  $M$ , and the y-axis gives the classification accuracies of our JSMK kernel. The lines of different colours represent the results on different datasets. The classification accuracies tend to become greater with increasing  $h$  or  $M$ . The reasons are twofold. First, for the parameter  $h$ , the greater the  $h$ , the higher dimensional DB complexity information of our kernel can be captured. Second, for the parameter  $M$ , the greater the  $M$ , the more pairs of aligned isomorphic local information graphs can be identified by our kernel.

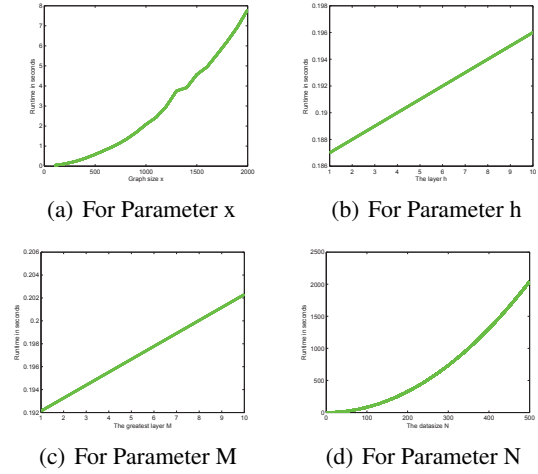


Figure 2: Runtime Evaluations.

## 4.2 Computational Evaluations

We explore the computational efficiency (i.e., the CPU runtime) of our JSMK kernel on randomly generated graphs with respect four parameters: the graph size  $n$ , the layer  $h$  of the DB representations of graphs, the greatest value of the JS representation layer  $M$ , and the graph dataset size  $N$ . We vary  $x = \{100, 200, \dots, 2000\}$ ,  $h = \{1, 2, \dots, 10\}$ ,  $M = \{1, 2, \dots, 10\}$ , and  $N = \{5, 10, \dots, 500\}$ , separately. a) For the parameter  $x$ , we generate 20 pairs of graphs with increasing number of vertices. We report the runtime for computing the kernel values between pairwise graphs ( $h = 10$  and  $M = 10$ ). b) For the parameter  $h$ , we generate a pair of graphs each of which has 200 vertices. We report the runtime for computing the kernel values of the pair of graphs as a function of  $h$  ( $M = 10$ ). c) For the parameter  $M$ , we use the pair of graphs from step b. We report the runtime for computing the kernel values of the pair of graphs as a function of  $M$  ( $h = 10$ ). d) For the parameter  $N$ , we generate 500 graph datasets with an increasing number of test graphs. In each dataset, one graph has 200 vertices. We report the runtime for computing the kernel matrices for each graph dataset ( $h = 10$  and  $M = 10$ ). **Note that, since  $M = 10$ , the runtime is for computing the 10 kernel matrices for each dataset.** The CPU runtime is reported in Fig.2, as operated in Matlab R2011b on a 2.5GHz Intel 2-Core processor (i.e., i5-3210m). Fig.2 indicates that the runtime scales cubically with  $n$ , linearly with  $h$  and  $M$ , and quadratically with  $N$ . These verify that our kernel can be computed in a polynomial time.

## 5 Conclusions

We have developed a JS matching kernel by aligning the JS representations that are computed over the corresponding DB representations in terms of the JSD measure. Our new kernel can incorporate explicit local substructure correspondence into the process of kernelization, the new kernel thus addresses the drawback of neglecting the relative locations between substructures that arises in the R-convolution kernels. Experiments demonstrate that our kernel can easily outperforms start-of-the-art graph kernels.

Table 1: Classification Accuracy (In %  $\pm$  Standard Error) Using C-SVM and Runtime.

| Datasets  | JSMK            | DBMK            | WLSK            | SPGK            | GCGK            | UQIS            | JSGK            | JTQK            |
|-----------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|
| BAR31     | 75.93 $\pm$ .51 | 69.40 $\pm$ .56 | 58.53 $\pm$ .53 | 55.73 $\pm$ .44 | 23.40 $\pm$ .60 | 30.80 $\pm$ .61 | 24.10 $\pm$ .86 | 60.56 $\pm$ .35 |
| BSPHERE31 | 64.46 $\pm$ .66 | 56.43 $\pm$ .69 | 42.10 $\pm$ .68 | 48.20 $\pm$ .76 | 18.80 $\pm$ .50 | 24.80 $\pm$ .61 | 21.76 $\pm$ .53 | 46.93 $\pm$ .61 |
| GEOD31    | 50.46 $\pm$ .45 | 42.83 $\pm$ .50 | 38.20 $\pm$ .68 | 38.40 $\pm$ .65 | 22.36 $\pm$ .55 | 23.73 $\pm$ .66 | 18.93 $\pm$ .50 | 40.10 $\pm$ .46 |

## References

- [Aziz *et al.*, 2013] Furqan Aziz, Richard C. Wilson, and Edwin R. Hancock. Backtrackless walks on a graph. *IEEE Trans. Neural Netw. Learning Syst.*, 24(6):977–989, 2013.
- [Bai and Hancock, 2013] Lu Bai and Edwin R. Hancock. Graph kernels from the jensen-shannon divergence. *Journal of Mathematical Imaging and Vision*, 47(1-2):60–69, 2013.
- [Bai and Hancock, 2014] Lu Bai and Edwin R. Hancock. Depth-based complexity traces of graphs. *Pattern Recognition*, 47(3):1172–1186, 2014.
- [Bai *et al.*, 2012] Lu Bai, Edwin R. Hancock, and Peng Ren. Jensen-shannon graph kernel using information functionals. In *Proceedings of ICPR*, pages 2877–2880, 2012.
- [Bai *et al.*, 2014a] Lu Bai, Peng Ren, Xiao Bai, and Edwin R. Hancock. A graph kernel from the depth-based representation. In *Proceedings of S+SSPR*, pages 1–11, 2014.
- [Bai *et al.*, 2014b] Lu Bai, Luca Rossi, Horst Bunke, and Edwin R. Hancock. Attributed graph kernels using the jensen-tsallis q-differences. In *Proceedings of ECML-PKDD*, pages 1:99–114, 2014.
- [Bai *et al.*, 2015] Lu Bai, Luca Rossi, Andrea Torsello, and Edwin R. Hancock. A quantum jensen-shannon graph kernel for unattributed graphs. *Pattern Recognition*, 48(2):344–355, 2015.
- [Bai, 2014] Lu Bai. *Information Theoretic Graph Kernels*. University of York, UK, 2014.
- [Barra and Biasotti, 2013] Vincent Barra and Silvia Biasotti. 3d shape retrieval using kernels on extended reeb graphs. *Pattern Recognition*, 46(11):2985–2999, 2013.
- [Biasotti *et al.*, 2003] S. Biasotti, S. Marini, M. Mortara, G. Patanè, M. Spagnuolo, and B. Falcidieno. 3d shape matching through topological structures. In *Proceedings of DGCI*, pages 194–203, 2003.
- [Borgwardt and Kriegel, 2005] Karsten M. Borgwardt and Hans-Peter Kriegel. Shortest-path kernels on graphs. In *Proceedings of ICDM*, pages 74–81, 2005.
- [Borgwardts, 2007] K.M. Borgwardts. *Graph Kernels*. Munchen, 2007.
- [Chang and Lin, 2011] C.-C Chang and C.-J. Lin. Libsvm: A library for support vector machines. *Software available at <http://www.csie.ntu.edu.tw/~cjlin/libsvm>*, 2011.
- [Costa and Grave, 2010] Fabrizio Costa and Kurt De Grave. Fast neighborhood subgraph pairwise distance kernel. In *Proceedings of ICML*, pages 255–262, 2010.
- [Crutchfield and Shalizi, 1999] J.P. Crutchfield and C.R. Shalizi. Thermodynamic depth of causal states: Objective complexity via minimal representations. *Physical Review E*, 59:275283, 1999.
- [Dehmer and Mowshowitz, 2011] Matthias Dehmer and Abbe Mowshowitz. A history of graph entropy measures. *Inf. Sci.*, 181(1):57–78, 2011.
- [Escolano *et al.*, 2012] F. Escolano, E.R. Hancock, and M.A. Lozano. Heat diffusion: Thermodynamic depth complexity of networks. *Physical Review E*, 85:036206, 2012.
- [Gärtner *et al.*, 2003] T. Gärtner, P.A. Flach, and S. Wrobel. On graph kernels: hardness results and efficient alternatives. In *Proceedings of COLT*, pages 129–143, 2003.
- [Harchaoui and Bach, 2007] Z. Harchaoui and F. Bach. Image classification with segmentation graph kernels. In *Proceedings of CVPR*, 2007.
- [Haussler, 1999] D. Haussler. Convolution kernels on discrete structures. *Technical Report UCS-CRL-99-10*, UC Santa Cruz, 1999.
- [Jebara *et al.*, 2004] T. Jebara, R.I. Kondor, and A. Howard. Probability product kernels. *Journal of Machine Learning Research*, 5:819–844, 2004.
- [Köner, 1971] J. Köner. Coding of an information source having ambiguous alphabet and the entropy of graphs. In *Proceedings of the 6th Prague Conference on Information Theory, Statistical Decision Function, Random Processes*, pages 411–425, 1971.
- [Martins *et al.*, 2009] André F. T. Martins, Noah A. Smith, Eric P. Xing, Pedro M. Q. Aguiar, and Mário A. T. Figueiredo. Nonextensive information theoretic kernels on measures. *Journal of Machine Learning Research*, 10:935–975, 2009.
- [Munkres, 1957] J. Munkres. Algorithms for the assignment and transportation problems. *Journal of the Society for Industrial and Applied Mathematics*, 5(1), 1957.
- [Scott and Longuet-Higgins, 1991] G.L. Scott and H.C. Longuet-Higgins. An algorithm for associating the features of two images. In *Proceedings of the Royal Society of London B*, pages 244:313–320, 1991.
- [Shervashidze *et al.*, 2009] N. Shervashidze, S.V.N. Vishwanathan, T. Petri, K. Mehlhorn, and K.M. Borgwardt. Efficient graphlet kernels for large graph comparison. *Journal of Machine Learning Research*, 5:488–495, 2009.
- [Shervashidze *et al.*, 2011] Nino Shervashidze, Pascal Schweitzer, Erik Jan van Leeuwen, Kurt Mehlhorn, and Karsten M. Borgwardt. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research*, 12:2539–2561, 2011.