GRAPH KERNELS FOR DISEASE OUTCOME PREDICTION FROM PROTEIN-PROTEIN INTERACTION NETWORKS

KARSTEN M. BORGWARDT AND HANS-PETER KRIEGEL

Institute for Computer Science,
Ludwig-Maximilians-University Munich,
Oettingenstr. 67, 80538 Munich, Germany
E-mail: kb@dbs.ifi.lmu.de,
kriegel@dbs.ifi.lmu.de

S.V. N. VISHWANATHAN AND NICOL N. SCHRAUDOLPH

Statistical Machine Learning Program,
National ICT Australia,
Canberra, 0200 ACT, Australia
E-mail: SVN.Vishwanathan@nicta.com.au,
Nic.Schraudolph@nicta.com.au

It is widely believed that comparing discrepancies in the protein-protein interaction (PPI) networks of individuals will become an important tool in understanding and preventing diseases. Currently PPI networks for individuals are not available, but gene expression data is becoming easier to obtain and allows us to represent individuals by a co-integrated gene expression/protein interaction network. Two major problems hamper the application of graph kernels – state-of-the-art methods for whole-graph comparison – to compare PPI networks. First, these methods do not scale to graphs of the size of a PPI network. Second, missing edges in these interaction networks are biologically relevant for detecting discrepancies, yet, these methods do not take this into account. In this article we present graph kernels for biological network comparison that are fast to compute and take into account missing interactions. We evaluate their practical performance on two datasets of co-integrated gene expression/PPI networks.

1. Introduction

An important goal of research on protein interactions is to identify relevant interactions that are involved in disease outbreak and progression. Measuring discrepancies between protein-protein interaction (PPI) networks of healthy and ill patients is a promising approach to this problem. Unfor-

tunately, establishing individual networks is beyond the current scope of technology. Co-integrated gene expression/PPI networks, however, offer an attractive alternative to study the impact of protein interactions on disease. But, researchers in this area are often faced with a computationally challenging problem: how to measure similarity between large interaction networks? Moreover, biologically relevant information can be gleaned both from the presence and absence of interactions. How does one make use of this domain knowledge? The aim of this paper is to answer both these questions systematically.

1.1. *Interaction Networks are Graphs*

We begin our study by observing that interaction networks are graphs, where each node represents a protein and each edge represents the presence of an interaction. Conventionally there are two ways of measuring similarity between graphs.

One approach is to perform a pairwise comparison of the nodes and/or edges in two networks, and calculate an overall similarity score for the two networks from the similarity of their components. This approach takes time quadratic in the number of nodes and edges, and is thus computationally feasible even for large graphs. However, this strategy is flawed in that it completely neglects the structure of the networks, treating them as sets of nodes and edges instead of graphs.

A more principled alternative would be to deem two networks similar if they share many common substructures, or more technically, if they share many common subgraphs. To compute this, however, we would have to solve the so-called subgraph isomorphism problem which is known to be NP-complete, *i.e.,* the computational cost of this problem increases exponentially with problem size, seriously limiting this approach to very small networks [1]. Many heuristics have been developed to speed up subgraph isomorphism by using special canonical labelings of the graphs; none of them, however, can avoid an exponential worst-case computation time.

Graph kernels as a measure of similarity on graphs offer an attractive middle ground: they can be computed in polynomial time, yet, they compare non-trivial substructures of graphs. In spite of these attractive properties, as they exist, graph kernels neither scale to large interaction networks nor do they address the issue of missing interactions. In this paper, we present fast algorithms for computing graph kernels which scale to large networks. Simultaneously, by using a complement graph – a graph

made up of all the nodes and the missing edges in the original graph – we address the issue of missing interactions in a principled manner.

Outline The remainder of this article is structured as follows. In Section 2, we will review existing graph kernels, and illustrate the problems encountered when applying graph kernels to large networks. In Section 3, we will present algorithms for speeding up graph kernel computation, and in Section 4, we will define graph kernels that take into account missing interactions as well. In our experiments (see Section 5), we employ our fast and enhanced graph kernels for disease outcome prediction, before concluding with an outlook and discussion.

2. Review of Existing Graph Kernels

Existing graph kernels can be viewed as a special case of R-Convolution kernels proposed by Haussler [2]. The basic idea here is to decompose the graph into smaller substructures, and build the kernel based on similarities between the decomposed substructures. Different kernels mainly differ in the way they decompose the graph for comparison and the similarity measure they use to compare the decomposed substructures.

Random walk kernels are based on a simple idea: Given a pair of graphs decompose them into paths obtained by performing a random walk, and *count* the number of *matching* walks [3–5]. Various incarnations of these kernels use different methods to compute similarities between walks. For instance, Gärtner et al. [3] count the number of nodes in the random walk which have the same label. They also include a decay factor to ensure convergence. Borgwardt et al. [5] on the other hand, use a kernel defined on nodes and edges in order to compute similarity between random walks. Although derived using a completely different motivation, it was recently shown by Vishwanathan et al. [6] that the marginalized graph kernels of Kashima et al. [4] are also essentially a random walk kernel. Mahé et al. [7] extend the marginalized graph kernels in two ways. They enrich the labels by using the so-called Morgan index, and modify the kernel definition to prevent tottering, *i.e.,* similar smaller substructures from generating high similarity scores. Both these extensions are particularly relevant for chemoinformatics applications. Other decompositions of graphs, which are well suited for particular application domains, include subtrees [8], molecular fingerprints based on various types of depth first searches [9], and structural elements like rings, functional groups and so on [10].

While many domain specific variants of graph kernels yield state-of-the-

art performance, they are plagued by computational issues when used to compare large graphs like those frequently found in PPI networks. This is mainly due to the fact that the kernel computation algorithms typically scale as $O(n^6)$ or worse. Practical applications therefore either compute the kernel approximately or make unrealistic sparsity assumptions on the input graphs. In contrast, in the next section, we discuss three efficient methods for computing random walk graph kernels which are both theoretically sound and practically efficient.

3. Fast Random Walk Kernels

In this section we briefly describe an unifying framework for random walk kernels, and present fast algorithms for their computation. We warn the biologically motivated reader that this section is rather technical. But, the algorithms presented below allow us to efficiently compute kernels on large graphs, and hence are crucial building blocks of our classifier for disease outcome prediction.

3.1. *Notation*

A graph $G(V, E)$ consists of an ordered and finite set of n vertices V denoted by $\{v_1, v_2, \ldots, v_n\}$, and a finite set of edges $E \subset V \times V$. G is said to be undirected if $(v_i, v_j) \in E \iff (v_j, v_i) \in E$ for all edges. The unnormalized adjacency matrix of G is an $n \times n$ real matrix P with $P_{ij} = 1$ if $(v_i, v_j) \in E$, and 0 otherwise. If G is weighted then P can contain non-negative entries other than zeros and ones, *i.e.,* $P_{ij} \in (0, \infty)$ if $(v_i, v_j) \in E$ and zero otherwise.

Let D be an $n \times n$ diagonal matrix with entries $D_{ii} = \sum_j P_{ij}$. The matrix $A := PD^{-1}$ is called the normalized adjacency matrix, or simply adjacency matrix. A walk w on G is a sequence of indices $w_1, w_2, \ldots w_{t+1}$ where $(v_{w_i}, v_{w_{i+1}}) \in E$ for all $1 \leq i \leq t$. The length of a walk is equal to the number of edges encountered during the walk (here: t). A random walk is a walk where $\mathbb{P}(w_{i+1}|w_1, \ldots w_i) = A_{w_i, w_{i+1}}$, *i.e.,* the probability at w_i of picking w_{i+1} next is directly proportional to the weight of the edge $(v_{w_i}, v_{w_{i+1}})$.

Let $\mathcal{X}$ be a set of labels which includes the special label ϵ. An edge labeled graph G is associated with a label matrix $L \in \mathcal{X}^{n \times n}$, such that $L_{ij} = \epsilon$ iff $(v_i, v_j) \notin E$. In other words, only those edges which are present in the graph get a non-ϵ label. Let $\mathcal{H}$ be the RKHS endowed with the kernel $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, and let $\phi : \mathcal{X} \to \mathcal{H}$ denote the corresponding feature map

which maps ϵ to the zero element of $\mathcal{H}$. We use $\Phi(L)$ to denote the feature matrix of G. For ease of exposition we do not consider labels on vertices here, though our results hold for that case as well.

3.2. *Product Graphs*

Given two graphs $G(V, E)$ and $G'(V', E')$, the product graph $G_\times(V_\times, E_\times)$ is a graph with nn' vertices, each representing a pair of vertices from G and G', respectively. An edge exists in $E_\times$ iff the corresponding vertices are adjacent in both G and G'. Thus

$$V_\times = \{(v_i, v'_{i'}) : v_i \in V \wedge v'_{i'} \in V'\}, \tag{1}$$

$$E_\times = \{((v_i,v'_{i'}),(v_j,v'_{j'})) : (v_i, v_j) \in E \wedge (v'_{i'}, v'_{j'}) \in E'\}. \tag{2}$$

If A and A' are the adjacency matrices of G and G', respectively, the adjacency matrix of the product graph $G_\times$ is given by $A_\times := A \otimes A'$, where $\otimes$ represents the Kronecker product of matrices.

If G and G' are edge-labeled, we can associate a weight matrix $W_\times \in \mathbb{R}^{nn' \times nn'}$ with $G_\times$, defined as $W_\times = \Phi(L) \otimes \Phi(L')$. Recall that $\Phi(L)$ and $\Phi(L')$ are matrices defined in an RKHS. Hence we use a slightly extended version of the Kronecker product and define the $(in+j, i'n'+j')$-th entry of $W_\times$ as $\kappa(L_{ij}, L_{i'j'})$. As a consequence of the definition of $\Phi(L)$ and $\Phi(L')$, the entries of $W_\times$ are non-zero only if the corresponding edges exist in the product graph.

We assume that $\mathcal{H} = \mathbb{R}^d$ endowed with the usual dot product, and that there are d distinct edge labels $\{1, 2, \ldots, d\}$. Moreover we let κ be a delta kernel, *i.e.,* its value between any two edges is one iff the labels on the edges match, and zero otherwise. Let ${}^l\!A$ denote the adjacency matrix of the graph filtered by the label l, *i.e.,* ${}^l\!A_{ij} = A_{ij}$ if $L_{ij} = l$ and zero otherwise. Some simple algebra (omitted for the sake of brevity) shows that the weight matrix of the product graph can be written as

$$W_\times = \sum_{l=1}^{d} {}^l\!A \otimes {}^l\!A'. \tag{3}$$

Let p and p' denote initial probability distributions over vertices of G and G'. Then the initial probability distribution $p_\times$ of the product graph is $p_\times := p \otimes p'$. Likewise, if q and q' denote stopping probabilities (*i.e.,* the probability that a random walk ends at a given vertex), the stopping probability $q_\times$ of the product graph is $q_\times := q \otimes q'$.

3.3. *Kernel Definition*

An edge exists in the product graph if, and only if, an edge exits in both G and G'. Therefore, performing a simultaneous random walk on G and G' is equivalent to performing a random walk on the product graph [11]. Given the weight matrix $W_\times$, initial and stopping probability distributions $p_\times$ and $q_\times$, and an appropriately chosen discrete measure μ, we can define a random walk kernel on G and G' as

$$k(G, G') := \sum_{k=0}^{\infty} \mu(k)\, q_\times^\top W_\times^k p_\times. \tag{4}$$

A popular choice to ensure convergence of (4) is to assume $\mu(k) = \lambda^k$ for some $\lambda > 0$. If λ is sufficiently small[a] then (4) is well defined, and we can write

$$k(G, G') = \sum_k \lambda^k q_\times^\top W_\times^k p_\times = q_\times^\top (\mathbf{I} - \lambda W_\times)^{-1} p_\times, \tag{5}$$

where $\mathbf{I}$ denotes the identity matrix. It can be shown (see Vishwanathan et al. [6]) that the marginal graph kernels of Kashima et al. [4] as well as the geometric graph kernels of Gärtner et al. [3] are special cases of (5).

3.4. *Fast Computation*

Direct computation of (5) is prohibitively expensive since it involves the inversion of a $nn' \times nn'$ matrix, which scales as $O(n^6)$. We now outline three efficient schemes whose worst case computational complexity is lower, and whose practical performance as measured by our experiments is up to three orders of magnitude faster. Vishwanathan et al. [6] contains more technical and algorithmic details.

3.4.1. *Sylvester Equation Methods*

Consider the following equation, commonly known as the Sylvester or Lyapunov equation:

$$X = SXT + X_0. \tag{6}$$

Here, $S, T, X_0 \in \mathbb{R}^{n \times n}$ are given and we need for solve for $X \in \mathbb{R}^{n \times n}$. These equations can be readily solved in $O(n^3)$ time with freely available code [12], *e.g.* Matlab's `dlyap` method.

[a]The values of λ which ensure convergence depend on the spectrum of $W_\times$.

It can be shown that if the weight matrix $W_\times$ can be written as (3) then the problem of computing the graph kernel (5) can be reduced to the problem of solving the following generalized Sylvester equation:

$$X = \sum_i {}^iA' \lambda \, X \; {}^iA^\top + X_0, \tag{7}$$

where $\mathrm{vec}(X_0) = p_\times$, with $\mathrm{vec}(\cdot)$ being the function that flattens a matrix by vertically concatenating its columns.

3.4.2. *Conjugate Gradient Methods*

Given a matrix M and a vector b, conjugate gradient (CG) methods solve the system of equations $Mx = b$ efficiently [13]. They are particularly efficient if the matrix is rank deficient, or has a small *effective rank*, *i.e.,* number of distinct eigenvalues. Furthermore, if computing matrix-vector products is cheap the CG solver can be sped up significantly [13].

The graph kernel (5) can be computed by a two-step procedure: First we solve the linear system

$$(\mathbf{I} - \lambda W_\times) \, x = p_\times, \tag{8}$$

for x, then we compute $q_\times^\top x$. By using extensions of tensor calculus rules to RKHS, one can compute $W_\times r$ for an arbitrary vector r rather efficiently, which in turn can be used to speed up the CG solver.

3.4.3. *Fixed-Point Iterations*

Fixed-point methods begin by rewriting (8) as

$$x = p_\times + \lambda W_\times x. \tag{9}$$

Now, solving for x is equivalent to finding a fixed point of the above iteration [13]. Letting x_t denote the value of x at iteration t, we set $x_0 := p_\times$, then compute

$$x_{t+1} = p_\times + \lambda W_\times x_t \tag{10}$$

repeatedly until $||x_{t+1} - x_t|| < \varepsilon$, where $||\cdot||$ denotes the Euclidean norm and ε some pre-defined tolerance. Observe that each iteration of (10) involves computation of the matrix-vector product $W_\times x_t$, and hence the extensions of tensor calculus to RKHS mentioned previously can again be used to speed up the computation.

4. Composite Graph Kernel

The presence of an edge in a graph signifies interactions between the end nodes. In many applications these interactions are significant. For instance, in chemoinformatics the presence of an edge indicates the presence of a chemical bond between two atoms. In the case of the PPI networks, the presence of an edge indicates that the corresponding proteins interact. But, when studying protein interactions in disease, not just the presence but also the absence of interactions is significant. Existing graph kernels (*e.g.* (5)) cannot take this into account. We propose to modify the existing kernels to take this information into account. Key to our exposition is the notion of a complement graph which we define below.

Suppose $G(V, E)$ is a graph with vertex set V and edge set E. Then, its complement $\bar{G}(V, \bar{E})$ is a graph with the same vertex set V, but with a different edge set $\bar{E} := V \times V \setminus E$. In other words, the complement graph is made up of all the edges missing from the original graph.

Using the concept of a complement graph we can now define a composite graph kernel as follows:

$$k_{comp}(G, G') = k(G, G') + k(\bar{G}, \bar{G}'). \tag{11}$$

Although this kernel seems simple minded at first, it is in fact rather useful. To see this consider the product graph $\bar{G}_\times(V_\times, \bar{E}_\times)$ of the complement graphs $\bar{G}$ and $\bar{G}'$. An edge exists in this graph if, and only if, the corresponding edge is absent in both G and G'. In other words, this graph characterizes all the missing interactions which are absent in both the PPI networks. As demonstrated by our experiments, this insight leads to gains in performance when comparing co-integrated gene expression/protein interaction networks.

5. Experiments

The aim of our experiments is to predict disease outcome (dead or alive) of cancer patients using a combination of human PPI data and clinical gene expression data.

Leukemia data For our first experiment, we employed a dataset of 119 microarrays of leukemia patients from Bullinger et al. [14], and co-integrated the expression profiles of these patients and known human PPI from Rual et al. [15]. This approach of co-integrating PPI and gene expression data is built on the assumption that genes with similar gene expression levels

are translated into proteins that are more likely to interact. Recent studies confirm that this assumption holds significantly more often for co-expressed than for random pairs of proteins [16, 17].

Specifically, we transformed a patient's gene expression profile into a graph as follows: A node is created for every protein which participates in a protein interaction, and whose corresponding gene expression level was measured on this patient's microarray. We connect two proteins in this graph by an edge if Rual et al. [15] list these proteins as interacting, and both genes are both up or downregulated with respect to a certain reference measurement. We found 2167 proteins from Rual et al. [15] for which Bullinger et al. [14] report gene expression levels.

The CPU runtimes of our CG, fixed-point, and Sylvester equation approaches to graph kernel computation (as described in Section 3) on the 119 patients modeled as graphs is contrasted with that of the direct approach in Table 1. 50 of the 119 patients survived leukemia. Using the computed kernel and a support vector machine (SVM) we tried to predict the survivors, in the first variant by using a vanilla graph kernel (5), and in the second variant by using the composite graph kernel (11). The average prediction accuracy obtained by performing 10-fold cross validation (with 10 repetitions) is reported in Table 2 for both approaches.

Table 1. Average time (in seconds) taken by different methods to compute the graph kernel on protein interaction networks.

	Leukemia dataset		*Breast cancer dataset*	
Computation approach	Vanilla	Composite	Vanilla	Composite
Direct	8,123	18,749	14,476	30,285
Sylvester	723	1,541	1,221	2,751
CG	6	13	13	28
Fixed	4	7	8	17

Breast cancer data This dataset consists of the microarrays of 78 breast cancer patients, of which 44 survived the disease [18]. When generating co-integrated graphs, we found 2429 proteins from Rual et al. [15] for which van 't Veer et al. [18] measure gene expression. As before, we report runtimes for kernel computations and accuracy levels for different variants of graph kernels in Table 1 and Table 2, respectively.

Table 2. Prediction accuracy (and standard deviation) of vanilla and composite graph kernels, averaged over 10 repetitions of 10-fold cross validation.

Graph kernel variant	Leukemia dataset	Breast cancer dataset
Vanilla	59.17 (2.49)	56.41 (2.12)
Composite	63.33 (1.76)	61.54 (1.54)

Results On both datasets, our approaches to fast graph kernel computation lead to up to three orders of magnitude gain in speed. The composite graph kernel outperforms the vanilla graph kernel in accuracy in both experiments, with an increase in prediction accuracy of around 4-5%. The vanilla random walk kernel suffers from its inability to measure network discrepancies, the simplicity of the graph model employed, and the fact that only a small minority of genes could be mapped to interacting proteins; due to these problems, its accuracy is close to that of a random classifier. But, since the composite kernel also models the missing interactions, even a simple model is able to capture relevant biological information, which in turn leads to better classification accuracy on these challenging datasets [19].

6. Outlook and Discussion

Two major stumbling blocks prevented the analysis of large datasets of PPI networks by modeling them as graphs. First, the scalability of the similarity metric to large graphs. Second, the inability to effectively use biological domain knowledge, viz. the presence or absence of certain protein interactions. In this article, we addressed both these issues by extending graph kernels to make them applicable to PPI networks. We sped up the computation of graph kernels by up to three orders of magnitude, without resorting to heuristics or approximations, thus making them practical for large problems. By using a composite kernel, we are able to model both the presence or absence of interactions between proteins. This leads to noteworthy improvements in classification accuracies in our experiments on disease outcome prediction for cancer patients.

With both these features — graph complement comparison and scalability — we have laid the foundation for future application of graph kernels in research on PPI networks in particular, and proteomics in general. While we have proposed approaches to overcome general problems, future studies will look at even more refined graph kernels that will further increase

prediction accuracy for specific tasks.

Acknowledgments

The authors thank Matthias Siebert for preprocessing the network data. National ICT Australia is funded by the Australian Government's Department of Communications, Information Technology and the Arts and the Australian Research Council through Backing Australia's Ability and the ICT Center of Excellence program. This work is supported by the IST Program of the European Community, under the Pascal Network of Excellence, IST-2002-506778, and by the German Ministry for Education, Science, Research and Technology (BMBF) under grant no. 031U112F within the BFAM (Bioinformatics for the Functional Analysis of Mammalian Genomes) project, part of the German Genome Analysis Network (NGFN).

References

1. M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Series of Books in Mathematical Sciences. W. H. Freeman, 1979.
2. D. Haussler. Convolutional kernels on discrete structures. Technical Report UCSC-CRL-99 - 10, Computer Science Department, UC Santa Cruz, 1999.
3. T. Gärtner, P. Flach, and S. Wrobel. On graph kernels: Hardness results and efficient alternatives. In B. Schölkopf and M. K. Warmuth, editors, *Proc. Annual Conf. Computational Learning Theory*. Springer, 2003.
4. H. Kashima, K. Tsuda, and A. Inokuchi. Kernels on graphs. In K. Tsuda, B. Schölkopf, and J. Vert, editors, *Kernels and Bioinformatics*, Cambridge, MA, 2004. MIT Press.
5. K. M. Borgwardt, C. S. Ong, S. Schonauer, S. V. N. Vishwanathan, A. J. Smola, and H. P. Kriegel. Protein function prediction via graph kernels. *Bioinformatics*, 21(Suppl 1):i47–i56, 2005.
6. S. V. N. Vishwanathan, K. Borgwardt, and N. N. Schraudolph. Fast computation of graph kernels. Technical report, NICTA, 2006.
7. P. Mahé, N. Ueda, T. Akutsu, J.-L. Perret, and J.-P. Vert. Extensions of marginalized graph kernels. In *Proceedings of the Twenty-First International Conference on Machine Learning*, 2004.
8. J. Ramon and T. Gärtner. Expressivity versus efficiency of graph ker-

nels. Technical report, First International Workshop on Mining Graphs, Trees and Sequences (held with ECML/PKDD'03), 2003.
9. L. Ralaivola, S. J. Swamidass, H. Saigo, and P. Baldi. Graph kernels for chemical informatics. *Neural Networks*, 18(8):1093 – 1110, 2005.
10. H. Fröhlich, J. K. Wegner, F. Siker, and andreas Zell. Kernel functions for attributed molecular graphs – a new similarity based approach to adme prediction in classification and regression. *QSAR and Combinatorial Science*, 25(4):317 – 326, 2006.
11. F. Harary. *Graph Theory*. Addison-Wesley, Reading, MA, 1969.
12. J. D. Gardiner, A. L. Laub, J. J. Amato, and C. B. Moler. Solution of the Sylvester matrix equation $AXB^\top + CXD^\top = E$. *ACM Transactions on Mathematical Software*, 18(2):223–231, 1992.
13. J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer Series in Operations Research. Springer, 1999.
14. L. Bullinger, K. Dohner, E. Bair, S. Frohling, R. F. Schlenk, R. Tibshirani, H. Dohner, and J. R. Pollack. Use of gene-expression profiling to identify prognostic subclasses in adult acute myeloid leukemia. *N Engl J Med*, 350(16):1605–1616, 2004.
15. J. F. Rual, K. Venkatesan, T. Hao, T. Hirozane-Kishikawa, A. Dricot, N. Li, et al. Towards a proteome-scale map of the human protein-protein interaction network. *Nature*, 437(7062):1173–1178, 2005.
16. H. B. Fraser, A. E. Hirsh, D. P. Wall, and M. B. Eisen. Coevolution of gene expression among interacting proteins. *Proc Natl Acad Sci U S A*, 101(24):9033–9038, 2004.
17. N. Bhardwaj and H. Lu. Correlation between gene expression profiles and protein-protein interactions within and across genomes. *Bioinformatics*, 21(11):2730–2738, 2005.
18. L. J. van 't Veer, H. Dai, M. J. van de Vijver, Y. D. He, A. A. M. Hart, et al. Gene expression profiling predicts clinical outcome of breast cancer. *Nature*, 415:530–536, 2002.
19. P. Warnat, R. Eils, and B. Brors. Cross-platform analysis of cancer microarray data improves gene expression based classification of phenotypes. *BMC Bioinformatics*, 6:265, 2005.