Node, Motif and Subgraph: Leveraging Network Functional Blocks Through Structural Convolution

Carl Yang[#], Mengxiong Liu[#], Vincent W. Zheng[*], Jiawei Han[#]

[#]*University of Illinois, Urbana Champaign, USA,* [*]*Advanced Digital Sciences Center, Singapore*

[#]{jiyang3, mliu60, hanj}@illinois.edu, [*]vincent.zheng@adsc.com.sg

***Abstract*—Networks or graphs provide a natural and generic way for modeling rich structured data. Recent research on graph analysis has been focused on representation learning, of which the goal is to encode the network structures into distributed embedding vectors, so as to enable various downstream applications through off-the-shelf machine learning. However, existing methods mostly focus on node-level embedding, which is insufficient for subgraph analysis. Moreover, their leverage of network structures through path sampling or neighborhood preserving is implicit and coarse. Network motifs allow graph analysis in a finer granularity, but existing methods based on motif matching are limited to enumerated simple motifs and do not leverage node labels and supervision. In this paper, we develop NEST, a novel hierarchical network embedding method combining motif filtering and convolutional neural networks. Motif-based filtering enables NEST to capture exact small structures within networks, and convolution over the filtered embedding allows it to fully explore complex substructures and their combinations. NEST can be trivially applied to any domain and provide insight into particular network functional blocks. Extensive experiments on protein function prediction, drug toxicity prediction and social network community identification have demonstrated its effectiveness and efficiency.**

I. Introduction

Network data are being generated in extraordinary volume and speed nowadays. Figure 1 (a) illustrates a small real-world Facebook social network collected by [1], which encodes rich information including various attributes and complex interactions. In domains from computational biology and chemoinformatics to robotics, program analysis and social networks, we are often interested in leveraging the rich attributes and links in networks, to cater various domain-specific applications.

Among many network analysis techniques, *node-level* representation learning has been attracting intense research attention for a long time [2]. While traditional algorithms perform with mathematical guarantees on small graphs (with thousands of nodes), they usually require global graph operations such as eigenvalue decomposition, for which the performance degrades largely due to sampling and approximation when scaled to current-scale real-world networks (with millions of nodes and more) on distributed systems.

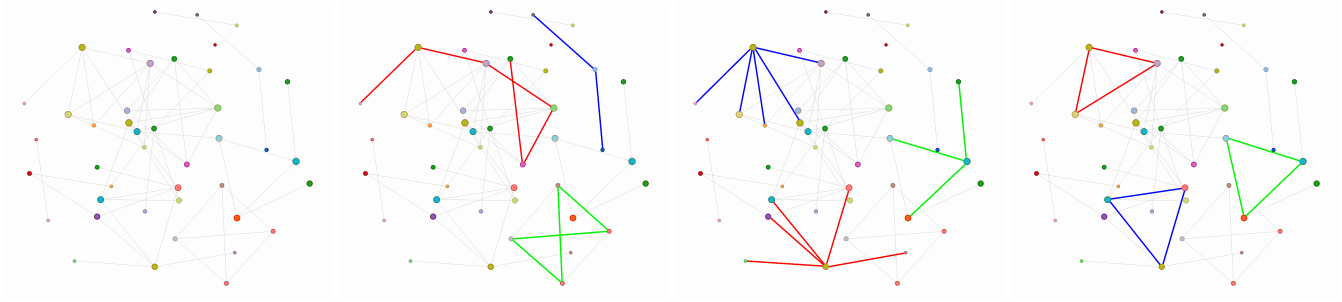
Recently, neural networks have been explored for graph representation learning, bringing a new trend to node-level network modeling. There are largely two groups of them: Path sampling based methods [3], [4], [5], [6] that efficiently leverage NLP models such as Skipgram [7], and neighborhood preserving based methods [8], [9], [10], [11] that optimize over local neighborhoods. Both groups can capture network structures to some extent, but as we illustrate in Figure 1 (b-c), their capture over network structures is implicit and coarse– path sampling based algorithms are inefficient in differentiating small complex structures and neighborhood preserving based ones are unaware of exact network structures.

To capture network structures in a finer granularity, researchers in computational biology and chemoinformatics have been working on finding *motif-level* patterns within networks for decades [12], which can be viewed as network building blocks with particular functions. Built on exact motif matching, graph kernel techniques compute the similarity among subgraphs [13], [14]. Such techniques imply a *subgraph-level* embedding, which can enable many new applications that cannot be sufficiently modeled by node-level embedding. However, as we illustrate in Figure 1 (d), current motif matching based methods are limited to a few small simple substructures and unable to capture those not explicitly enumerated.

In this paper, we develop NEST (*NEtwork Structural convoluTion*), a novel neural architecture for network embedding from node-level to motif- and subgraph-level. The main advantages of NEST over many existing embedding methods are as follows:

1) **Multi-resolution**: NEST starts with node-level attribute embedding, continues with motif-level order-invariant filtering, and all the way up to subgraph-level pattern combination. Expressive nonlinear neural networks are built into each layer to comprehensively explore network attributes and links in different granularities.
2) **Task-driven**: NEST is an end-to-end deep learning framework trained under the supervision of ground-truth subgraph labels. Such integration allows it to perform supremely across various domains.
3) **Interpretable**: The learned motif-based convolution networks explicitly indicate what motifs and which nodes are dominating the learning results, which provides valuable insight into the functional blocks and higher-order organizations of various networks.

Extensive experimental evaluations on subnetwork identification enabled by our subgraph-level representation demonstrate the effectiveness and efficiency of NEST, while we expect general utility of all three level representations in various other network downstream applications.



(a) A small FB network (b) Path sampling (c) Neighborhood convolution (d) Motif matching

Fig. 1. (a) A small real-world FB network: nodes with different colors are students in different majors and edges are their undirected friendships. (b) Path sampling: different small structures can lead to similar sets of sampled paths. (c) Neighborhood preserving: neighboring nodes are treated similarly no matter how they connect with each other. (d) Motif matching: unable to capture substructures that are not explicitly enumerated.

II. Related Work

Due to the advanced data warehousing techniques and much effort put into knowledge acquisition, lots of data are stored with structures, ready to be modeled by networks in a generic way. Network embedding, by computing distributed representations that preserve network information, provides a general solution to various downstream applications. In this section, we review related work in network embedding through a three-level hierarchy: node, motif and subgraph.

A. Node-Level: Network Embedding

Earlier studies on network embedding are focused on node-level. They tend to first construct the complete node affinity graph based on node feature similarity and then compute the eigenvector of the affinity matrix as node embedding [2], [15]. Such computations are usually global and hard to be parallelized, which limits their usage for nowadays large-scale real-world networks.

Recently, the successful Skipgram model for word embedding [7] has motivated many node-level embedding techniques. By sampling truncated random walks (*e.g.*, [3], [4], [6], [5]) or designing proximity preserving objectives (*e.g.*, [9], [8], [16], [17]), the embedding they compute can preserve network structures in an implicit and coarse way. However, as we discussed in Section I, these techniques are unaware of exact network structures and thus unable to differentiate small complex structures. Moreover, most of them are unsupervised and unable to handle node attributes, and thus do not efficiently explore important network patterns *w.r.t.* different domains. Finally, they only provide node-level embedding, which is insufficient for higher-level tasks like subgraph identification.

B. Motif-Level: Network Kernel

Network motifs have been studied for decades by researchers in computational biology and chemoinformatics [12]. Most analyses have been around simple statistics such as frequency and correlation. Recently, motif is studied on general networks [18], [19], [20], due to its implication into higher-order network organization. They do not compute network embedding, but instead only leverage the counting of basic motifs matched on the network.

The group of algorithms that compute network embedding based on motifs are called graph kernels [21], [22]. They measure the similarity between a pair of networks by decomposing them into atomic substructures and defining a kernel function that corresponds to an inner product over the substructures in reproducing kernel Hilbert space. The state-of-the-art network kernel methods are able to handle correlated substructures and alleviate diagonal dominance by capturing the similarity among substructures [13], [14]. However, as illustrated in Figure 1 (d), while kernel methods capture exact motif-level substructures, they are limited to the few enumerated simple motifs and do not comprehensively explore more complex structures and their combinations. Moreover, the computed kernel functions are learned without supervision, separately from the final subgraph classification objectives, and it is non-trivial to consider node attributes, which is hard to perform well across different domains.

C. Subgraph-Level: Network Convolution

Convolutional neural networks have been extremely successful in machine learning problems where the coordinates of the underlying data have a grid structure [23]. Network data are naturally not on such fine grids, but valid translations can be formed through either a spatial approach [24], [25], [26] or a spectral one [10], [11], [27]. Although their aim is on node-level embedding, the convolutions along domain hierarchies or graph Laplacian spectrum effectively collect neighboring nodes in hierarchies and thus can be interpreted as multi-resolution embedding in the spatial neighborhoods or Fourier domain. However, their subgraph-level embedding is still coarse, because the capture of exact structures is implicit.

The two approaches of translation both suffer from some additional limitations. For spatial translation, [25], [26] designed multi-scale hierarchical local receptive fields, but there is no good way to choose the optimal hierarchies; [24] proposed to determine a unique node ordering for computing neighborhood graphs, but a fixed ordering limits the exploration of arbitrary complex structures, and what is a good ordering is still an open question. On the other hand, [10], [11], [27] developed recursive spectral filtering based on graph Laplacian with solid mathematical guarantees, but their results without the consideration of the specific spatial structures are less conceivable.

Symbol	Definition
$\mathcal{N}$	Network
$\mathcal{V} = \{v_1, \ldots, v_n\}$	Objects (chemical compounds, users, *etc.*)
$\mathcal{E}$	Links (interactions, friendships, *etc.*)
$\mathcal{A}$	Attributes of objects
$\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathcal{A}\}$	Graph for modeling a network
$\mathcal{M} = \{\mathcal{V}_m, \mathcal{E}_m, \mathcal{A}_m\}$	Motif-level subset of the graph
$\mathcal{S} = \{\mathcal{V}_s, \mathcal{E}_s, \mathcal{A}_s\}$	Subgraph-level subset of the graph
$\mathbf{x}(v)$	Node-level embedding
$\mathbf{x}(\mathcal{M})$	Motif-level embedding
$\mathbf{x}(\mathcal{S})$	Subgraph-level embedding
$y(\mathcal{S})$	True label of a subnetwork
$\hat{y}(\mathcal{S})$	Predicted label of a subnetwork

TABLE I
SYMBOLS AND DEFINITIONS USED THROUGHOUT THE PAPER.

III. Problem Formulation

A. Problem Definition

In this paper, we consider hierarchical embedding of networks, from nodes to motifs and subgraphs. While our node-level embedding can be used for fundamental network learning tasks like node classification, network clustering, missing link prediction and so on, in this work, we focus on leveraging the subgraph-level embedding for the challenging task of subnetwork identification.

Input. Given a network $\mathcal{N}$ that can be from any domain, we represent its set of total n objects as $\mathcal{V} = \{v_1, \ldots, v_n\}$, their m links as $\mathcal{E}$, and their attributes as $\mathcal{A}$. Therefore, $\mathcal{N}$ can be represented as a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathcal{A}\}$. For most networks like chemical compound networks and social networks, $\forall e_{ij} \in \mathcal{E}$ is usually binary, where $e_{ij} = 1$ indicates the existence of an interaction or a link between v_i and v_j and 0 otherwise. $\forall \boldsymbol{a}_i \in \mathcal{A}$ records the l-dimensional attributes of v_i. Each component of $\boldsymbol{a}_i$ can be numerical (*e.g.*, age, income) or categorical (*e.g.*, school, company), describing the status of v_i. Some network datasets may not include the attribute set at all [28].

Output. Our objective is to learn a multi-resolution embedding of the network in three levels: node-level $x(v)$, motif-level $x(\mathcal{M})$ and subgraph-level $x(\mathcal{S})$, which explores and preserves network functional blocks in different resolutions. For subnetwork identification, we also predict a class label $\hat{y}(\mathcal{S})$ for each candidate subgraph, which can be supervised by the true labels $y(\mathcal{S})$ to effectively explore $\mathcal{N}$ *w.r.t.* different domains.

B. Toy Example

Figure 2 provides a toy example of a hierarchical network embedding pipeline. It starts from node-level embedding, which explores available node attributes captured by the particular network. Motif-level embedding is then computed by assembling the corresponding node-level embedding. The motif instances are later utilized as building blocks of more complex network structures, *i.e.*, subgraphs, whose embedding is computed as an organic combination of motif-level embedding. The subgraph-level embedding can be further convolved and filtered through proper neural networks and used to produce subgraph predictions for supervision. If no supervision is available, heuristic metrics of network similarity such as edit distance can be computed to form an unsupervised objective, requiring neighboring subnetworks to be close.

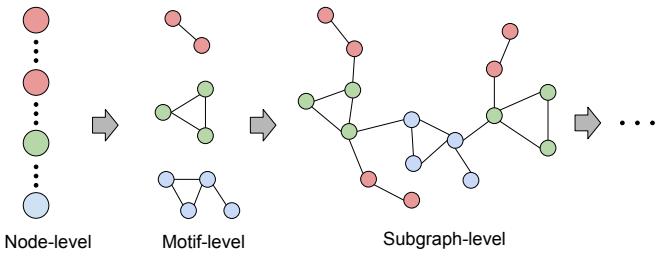


Fig. 2. A toy example illustrating a hierarchical network embedding pipeline.

C. Desired Properties

Our goal is to identify subnetworks with prominent patterns in $\mathcal{N}$, typically indicated by particular node-level features, motif-level structures and subgraph-level patterns.

Definition III.1. *Subnetwork identification. Given a subnetwork $\mathcal{S} = \{\mathcal{V}_s, \mathcal{E}_s, \mathcal{A}_s\}$, the goal of subnetwork identification is to predict a class label $\hat{y}(\mathcal{S})$, indicating the function of $\mathcal{S}$.*

The functions of subnetworks are different across domains. For example, in a biochemistry network, specific subnetworks might indicate a pathway or disease, while in a social network, they may correspond to user groups of different interests.

Since we want our algorithm to be efficient in identifying subnetworks, we study the challenges in exploring networks.

Definition III.2. *Complexity. Node attributes and interactions are usually noisy, complex and multi-scale. Therefore, an ideal algorithm should explore the networks in different resolutions and comprehensively capture fuzzy complex network patterns.*

Definition III.3. *Variety. Network patterns and functions vary across different domains. Therefore, an ideal algorithm should leverage domain-specific node attributes and class supervision, preferably in an end-to-end coherent learning framework.*

Definition III.4. *Intractability. Network embedding methods are often intractable in terms of the meaning and process of representation learning. Therefore, an ideal algorithm should respect and leverage specific network structures, providing insightful interpretation into network functional blocks.*

IV. Network Structural Convolution

In this section, we develop NEST (NEtwork Structural convoluTion), to provide a generic network embedding framework that simultaneously address the aforementioned three challenges.

Multi-Resolution Embedding Architecture. In many real-world networks, we can acquire labels for subnetworks. For example, in biochemistry networks, we know some disease pathways and are interested in identifying similar ones; alike in social networks, it is natural to look for similar user groups based on the knowledge of some identified ones. Such domain-specific supervision, although often limited, is valuable in guiding the exploration of network functional blocks.

Figure 3 showcases the neural architecture of our NEST embedding framework. Consider a subnetwork represented by

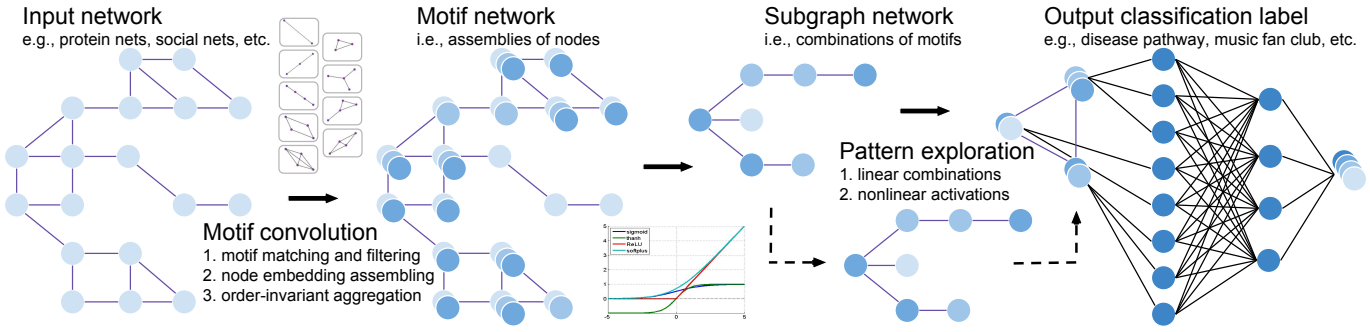


Fig. 3. The neural architecture of our proposed NEST multi-resolution network embedding framework.

$\mathcal{S}$, which is a subset of the whole network $\mathcal{N}$, such as a group of proteins and their interactions in a biochemistry network, or some users and their relationships in a social network. NEST computes a multi-resolution embedding over $\mathcal{S}$, which includes a pipeline of node-level attribute embedding, motif-level structural convolution and subgraph-level pattern exploration, and finally produces a classification label $\hat{y}(\mathcal{S})$.

Since we focus on the task of supervised subnetwork identification, which is essentially a subgraph-level multi-class classification problem, we apply the standard softmax prediction over the final subnetwork embedding $\mathbf{x}(\mathcal{S})$ and compute a cross-entropy loss for each subnetwork as in the training set as

$$\mathcal{L}_s = -\log \mathcal{L}(\Theta|y,\hat{y}) = -\sum_{c=1}^{C} \mathbb{I}(y(\mathcal{S})=c) log(\hat{y}_c(\mathcal{S})), \quad (1)$$

where $\mathbb{I}(\cdot)$ is an indicator function that outputs 1 when the argument is true and 0 otherwise, and $\hat{y}_c$ is computed as

$$\hat{y}_c(\mathcal{S}) = p(\hat{y}(\mathcal{S})=c) = \frac{exp(\theta_c^T \mathbf{x}(\mathcal{S}))}{\sum_{i=1}^{C} exp(\theta_i^T \mathbf{x}(\mathcal{S}))}. \quad (2)$$

The total loss for all m subgraphs in the training set is thus

$$\mathcal{L} = \sum_{i=1}^{m} \mathcal{L}_{s_i} = -\sum_{i=1}^{m}\sum_{c=1}^{C} \mathbb{I}(y(\mathcal{S}_i)=c) log(\hat{y}_c(\mathcal{S}_i)). \quad (3)$$

Node-Level Attribute Embedding. An ideal network embedding algorithm should be able to fully explore domain-specific node attributes. However, existing embedding frameworks usually start with preserving network structures [3], [4], [8], and then incorporate node attributes through graph augmentation [29], [30]. Such methods are inefficient in capturing high-dimensional noisy attributes. Some recent methods like [6] start with node embedding computed based on node attributes, which is shown to better leverage the similarity among nodes.

In this work, we start from node-level attribute embedding, by connecting a fully connected linear embedding layer to the input of original node attributes as

$$\mathbf{x}(v_i) = f_n(\mathbf{a_i}) = \mathbf{W}_n \mathbf{a_i} + \mathbf{b}_n, \quad (4)$$

where f_n is the attribute embedding function and $\Theta_n = \{\mathbf{W}_n, \mathbf{b}_n\}$ is the set of parameters.

This layer is easy to learn, while effectively explores all linear combinations of various node features. With proper feature preprocessing such as sparse one-hot embedding, these combinations account for most straightforward feature configurations such as all one-feature filtering rules connected by AND and OR operations, representing the status of individual nodes in a network. For example, by considering different combinations of age ranges, education backgrounds and publication numbers, a simple linear layer is able to well differentiate the roles of professors, lecturers and students in a publication network.

Our one-layer attribute embedding is similar to that of [6], and can be easily extended to multiple layers of nonlinear feedforward neural networks, given complex node features and sufficiently large training data. We use O to denote the number of hidden layers here and experiment on different settings of O in Sec V.3. However, [6] considers network structures by incorporating network path sampling to incur an unsupervised joint training objective, which is unable to capture small exact network patterns, produce multi-resolution representations and provide insightful interpretations of functional blocks. To address these issues, we continue with hierarchical assemblies of nodes into motifs and combinations of motifs into subgraphs.

Motif-Level Structural Convolution. Network structures are usually complex and hard to understand. Therefore, most embedding algorithms chose to ignore the exact structures, but implicitly maintain a network-based proximity through path sampling [3], [4] or neighborhood preserving [8], [10]. Such leverage of network structures is coarse and does not provide interpretation into network functional building blocks.

Convolutional neural networks are known to be powerful in data with spatial orders, such as images, due to their automatic exploration of important structural features like edges of some orientations and blotches of some colors indicating objects in images. However, networks do not have well defined spatial orders, and thus convolution can not be directly applied. As we discussed in Section II, existing convolutional methods for network embedding do not provide satisfactory results, due to the lack of a unique and interpretable ordering or clustering technique [24], [25].

In this work, we enumerate a set of K basic network motifs $\Omega = \{\mathcal{M}^1, \ldots, \mathcal{M}^K\}$ as functional building blocks of larger complex networks, and construct a set of filter windows corresponding to each of the motifs (*e.g.*, the nine motifs in Figure 3 are all motifs composed of 2-4 nodes without isolated

node). Then we convolve these windows along the network by matching the corresponding motifs (*i.e.*, finding instances of each motif), assemble the nodes matched within each instance through vector concatenation of their node-level embedding, and connect a fully connected linear layer to yield an instance-level embedding $\mathbf{x}(\mathcal{M}_t)$. To put it formally, for each motif $\mathcal{M}^k \in \Omega$, we have

$$\mathbf{x}(\mathcal{M}_t^k) = \mathbf{f}_m^k([\mathbf{x}(v_{t1}^k), \ldots, \mathbf{x}(v_{t|\mathcal{M}^k|}^k)]), \tag{5}$$

where $\mathbf{f}_m^k(\mathbf{x}) = \mathbf{W}_m^k\mathbf{x} + \mathbf{b}_m^k$, and $\{v_{t1}^k, \ldots, v_{t|\mathcal{M}_k|}^k\}$ is the set of nodes matched on the t-th instance of $\mathcal{M}^k$. Since each motif can find multiple instances in a network, we aggregate the instance-level embedding of each motif into a motif-level embedding through an order-invariant operation such as summation or averaging. Assuming $\mathcal{M}^k$ was matched with T instances on $\mathcal{S}$, we have

$$\mathbf{x}(\mathcal{M}^k) = \mathbf{f}_{ma}(\mathbf{x}(\mathcal{M}_1^k), \ldots, \mathbf{x}(\mathcal{M}_T^k)), \tag{6}$$

where $\mathbf{f}_{ma}(\cdot)$ is an aggregation function within the motif-level embedding layer (*e.g.*, point-wise summation, averaging, *etc*.).

Finally, we stack the K motif-level embedding and use them as input to the next layer.

$$\mathbf{x}(\mathcal{M}) = [\mathbf{x}(\mathcal{M}^1), \ldots, \mathbf{x}(\mathcal{M}^K)]. \tag{7}$$

In motif-level embedding, we generalize the idea of convolution by using learnable functions $\{\mathbf{f}_m^k\}_{k=1}^K$ to apply the same operations locally everywhere *w.r.t.* network motifs, and combine information through a global aggregation step. In this way, instead of explicitly aligning nodes in the networks, we can automatically explore the efficient ways of assembling node-level embedding into instance and motif-level. While we start with simple fully connected linear layers, it is also possible to stack more nonlinear layers and we experiment on the influence of number of hidden layers (P) in the motif-level embedding in Sec V.3.

Parameters Θ_m inside functions $\{\mathbf{f}_m^1, \ldots, \mathbf{f}_m^K\}$ *w.r.t.* motifs in Ω are shared across the whole network, which effectively explores the global consistency of network functional blocks. For example, in a specific domain like a gene-protein network, a three-node motif of a triangle composed of two specific genes and one specific protein might be indicative for a certain disease pathway, which can be well captured by our motif-based filtering. The probability of this disease might be even larger if this motif is instantiated with some other indicative motif instances, which can be further captured by our next layer of combining motifs into more complex subgraphs.

Subgraph-Level Pattern Exploration. The idea of exactly matching small fixed-sized motifs for capturing network structures has been leveraged by many graph kernel methods [13], [31], [22], [21]. They usually focus on resolving substructure similarity and diagonal dominance, and are shown to be effective in computing network structural similarity. However, due to combinatorial complexity of motif enumeration and matching, their consideration of motifs are restricted to the enumerated small subgraphs with a few nodes.

In this work, based on motif-level embedding $\mathbf{x}(\mathcal{M})$, we add multiple (Q) layers of nonlinear feedforward neural networks to comprehensively explore the patterns of various combinations of the small simple motifs into more complex larger subgraphs. Thus we have

$$\mathbf{x}(\mathcal{S}) = \mathbf{f}_s^Q(\ldots \mathbf{f}_s^1(\mathbf{f}_s^0(\mathcal{S}))\ldots), \tag{8}$$

where

$$\mathbf{f}_s^q(\mathbf{x}) = \mathbf{f}_{sn}(\mathbf{W}_s^q\mathbf{f}_s^{q-1}(\mathbf{x}) + \mathbf{b}_s^q), \tag{9}$$

for $q \geq 1$, and $\mathbf{f}_s^0(\mathcal{S}) = \mathbf{x}(\mathcal{M})$. $\mathbf{f}_{sn}$ is a point-wise nonlinear activation function such as sigmoid or ReLU. Θ_s is the set of parameters in the Q subgraph-level embedding layers.

In the motif-level embedding, each embedded dimension can be understood as a possible function of the specific motif and its status, which can be further activated or de-activated based on the existence and status of other motifs in the subgraph-level embedding. The subgraph-level embedding through the nonlinear pattern exploration layers are finally converted into classification label prediction through a softmax function, which can be used to compute a supervised loss *w.r.t.* ground-truth subgraph labels. Gradients of the loss are back-propagated to all three of the subgraph-level, motif-level and node-level embedding layers, to guide the exploration of network functional blocks in different granularities simultaneously in a coherent way.

Implementation, Training and Efficiency. We implement the NEST neural framework with TensorFlow[1]. We also use an existing motif matching algorithm called SubMatch[2] [32]. The codes will be made available upon acceptance of the work.

To train the NEST model, we optimize the loss function in Eq. 3 by performing stochastic gradient descent with mini-batch Adam [33]. Before training, we preprocess node attributes in each dataset by applying sparse one-hot encoding and recording them into an attribute tensor. Then we preprocess the subgraphs in training and testing sets by matching a set of K enumerated motifs and collecting the ids of matched nodes into a structure tensor. During training, in each batch, we sample one subgraph from the training set and input its corresponding part of the attribute and structure tensors to compute a partial loss function in Eq. 1 and execute one step of gradient descent for all parameters $\Theta = \{\Theta_n, \Theta_m, \Theta_s\}$ in the three embedding layers respectively, until the overall loss $\mathcal{L}$ converges or is sufficiently small.

Although we sample one subgraph to train the model in each batch sequentially, it is possible to compute multiple batches and optimize the model in parallel. Matching graph motifs can be efficiently done through well-developed techniques offline before training. Many industrial graphs are developed with such functionalities to enable various data analysis. Training the multi-resolution model is just as efficient as training a feedforward neural network with $O + P + Q$ layers. In our experiments, NEST runs efficiently on CPU, while it is expected to run faster on GPU.

[1]https://www.tensorflow.org/

[2]https://sites.google.com/site/fangyuan1st/data-and-tools/submatch

V. EXPERIMENTS

In this section, we evaluate NEST for subnetwork identification with extensive experiments on 12 real-world networks.

A. Experimental Setups

Datasets. We use 12 datasets, among which 6 are standard benchmark datasets from bioinformatics, namely, MUTAG (mutagenic aromatic and heteroaromatic nitro compounds) [34], PTC (chemical compounds reporting rats' carcinogenicity) [35] , PROTEINS (secondary structure elements linked with neighbors in the amino-acid sequence or in 3D space) [36], D&D (enzymes and non-enzymes protein structures) [37], NCI1 and NCI109 (chemical compounds screened for activity against the growth of a panel of human tumor cell lines) [38], and the other 6 are public social network datasets processed by [13], *i.e.*, COLLAB (a scientific-collaboration dataset derived from three public collaboration datasets of researchers in different research fields), IMDB-Binary, IMDB-Multi (two movie-collaboration dataset with actor/actress and genre information of different movies on IMDB[3]), Reddit-Binary, Reddit-Multi-5K, and Reddit-Multi-12K (three social networks crawled from Reddit[4] composed of submission graphs from popular subreddits). Their properties are shown in the first few columns in Table II. To highlight the ability of NEST to leverage various node attributes, besides the original node features, we also compute and incorporate normalized node degrees and page rank scores as initial node attributes.

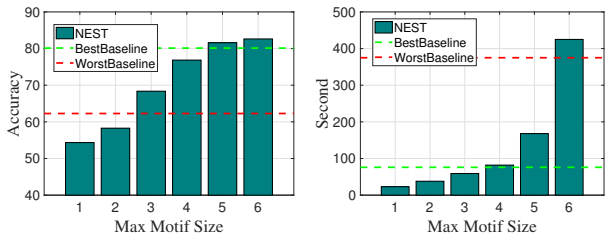
We perform 10-fold cross-validation, use 9 folds for training and 1 for testing, and repeat the experiments for 10 times to report average classification accuracies and standard deviations.

Compared algorithms. Since most network embedding algorithms only compute node-level embedding and are not suitable for subnetwork identification, we compare NEST with the following subgraph-level embedding algorithms:

- **SP** [39]: shortest-path kernel.
- **RW** [40]: random walk kernel.
- **WL** [22]: Weisfeiler-Lehman subtree kernel.
- **GK** [21]: graphlet count kernel.
- **DGK** [13]: deep graphlet count kernel.
- **PSCN** [24]: Patchy-San graph convolutional networks.

Parameter settings. When comparing NEST with the baseline methods, we set its parameters to the following default values: for motif set Ω, we enumerate all connected small structures composed of 1-5 nodes, which results in $K = 31$ motifs in total; for node-level embedding, we set the number of hidden layers O to 1 and the embedding size $|\mathbf{x}(v)|$ to 32; for motif-level embedding, we set the number of hidden layers P to 1 and the embedding size $|\mathbf{x}(\mathcal{M})|$ to 32; for subgraph-level embedding, we set the number of hidden layers Q to 3 and the sizes of the layers to $128 \rightarrow 64 \rightarrow 32$, so the embedding size $|\mathbf{x}(\mathcal{S})|$ is 32; for the transformation functions, we use the popular ReLU for nonlinear activation in $\mathbf{f}_{nn}$, $\mathbf{f}_{mn}$ and $\mathbf{f}_{sn}$ (if any), and point-wise summation for $\mathbf{f}_{ma}$. In addition to these default values, we also evaluate the effects of different parameter settings on the performance of NEST in Sec V.3.

To make a fair comparison, we follow previous works using the same datasets [13], [24] to set parameters for baseline algorithms. For RW, we choose the decay factor from $\{10^{-6}, 10^{-5}, \ldots, 10^{-1}\}$; for WL, we set the height parameter to 2; for GK and DGK, we set the size of the graphlets to 6; for PSCN, we set the size of receptive fields to 10 and the convolution architecture as suggested in their original experiments.



(a) Performance (b) Runtime

Fig. 4. **NEST with different motif sets on NCI1.**

Research problems. Our experiments are designed to answer the following research questions:

- **Q1. Performance:** Does NEST provide better solutions for network embedding and subnetwork identification? How much does it improve on the state-of-the-art?
- **Q2. Robustness:** How does the exact design of neural architecture influence the performance of NEST? Do more complex neural networks help?
- **Q3. Interpretability:** Does NEST capture functional building blocks and provide insightful interpretations into the formation of networks in different domains?

B. Q1. Performance

We quantitatively evaluate NEST against all baselines on standard subnetwork classification. Table II shows the performance in terms of classification accuracies on the 12 datasets. NEST is able to outperform all compared algorithms on 10 out of the 12 datasets, and is highly competitive in the remaining 2. The improvements of NEST over the outperformed algorithms all passed the paired t-tests with significance value $p < 0.01$.

The consistent promising performances on all compared datasets demonstrate the general advantages of NEST to explore complex network structures and leverage domain-specific supervision. Moreover, we observe large performance gain on MUTAG, PTC, and IMDB datasets, where the average numbers of nodes in the subnetworks are comparatively small, indicating the significance and the power of NEST to capture exact small substructures. NEST is also advantageous in leveraging node attributes (labels) in datasets like PTC, NCI1 and NCI109, which have 19, 37 and 38 distinct node labels respectively. As the number of graphs increases, we experience significant improvements on relative performance on datasets such as the NCI, COLLAB and RED (Reddit) ones. Finally,

[3]http://www.imdb.com/
[4]https://www.reddit.com/

	Dataset properties		Algorithm performances						
Dataset	**#graphs**	**Avg.nodes**	**SP**	**RW**	**WL**	**GK**	**DGK**	**PSCN**	**NEST**
MUTAG	188	17.9	85.22 ± 2.43	83.68 ± 1.66	82.94 ± 2.68	81.66 ± 2.11	82.66 ± 1.45	*88.95 ± 4.37*	$\mathbf{91.85 \pm 1.57}$
PTC	344	25.6	58.24 ± 2.44	57.26 ± 1.30	56.97 ± 2.01	57.26 ± 1.41	57.32 ± 1.13	*62.29 ± 5.68*	$\mathbf{67.42 \pm 1.83}$
PROTEINS	1,113	39.1	*75.07 ± 0.54*	74.22 ± 0.42	72.92 ± 0.56	71.67 ± 0.55	71.68 ± 0.50	75.00 ± 2.51	$\mathbf{76.54 \pm 0.26}$
D&D	1,178	284.3	$>$ 3 days	>3 days	77.95 ± 0.70	*78.45 ± 0.26*	$\mathbf{78.50 \pm 0.22}$	76.27 ± 2.64	78.11 ± 0.36
NCI1	4,110	29.9	73.00 ± 0.24	$>$ 3 days	*80.13 ± 0.50*	62.28 ± 0.29	62.40 ± 0.25	76.34 ± 1.68	$\mathbf{81.59 \pm 0.46}$
NCI109	4,127	29.6	73.00 ± 0.21	$>$ 3 days	*80.22 ± 0.34*	62.60 ± 0.19	62.69 ± 0.23	76.69 ± 1.71	$\mathbf{81.72 \pm 0.41}$
COLLAB	5,000	74.5	$>$ 3 days	$>$ 3 days	72.01 ± 0.35	72.84 ± 0.28	*73.09 ± 0.25*	72.60 ± 2.15	$\mathbf{74.55 \pm 0.60}$
IMDB-B	1,000	19.8	*71.26 ± 1.04*	67.94 ± 0.77	67.40 ± 1.26	65.87 ± 0.98	66.96 ± 0.56	71.00 ± 2.29	$\mathbf{73.26 \pm 0.72}$
IMDB-M	1,500	13	*51.33 ± 0.57*	46.72 ± 0.30	45.42 ± 0.78	43.89 ± 0.38	44.55 ± 0.52	45.23 ± 2.84	$\mathbf{53.08 \pm 0.31}$
RED-B	2,000	429.6	$>$ 3 days	$>$ 3 days	76.56 ± 0.20	77.34 ± 0.18	78.04 ± 0.39	*86.30 ± 1.58*	$\mathbf{88.52 \pm 0.64}$
RED-M5K	5,000	508.8	$>$ 3 days	$>$ 3 days	39.29 ± 0.15	41.01 ± 0.17	41.27 ± 0.18	$\mathbf{49.10 \pm 0.70}$	*48.61 ± 0.46*
RED-M12K	11,929	391.4	$>$ 3 days	$>$ 3 days	31.08 ± 0.08	31.82 ± 0.08	32.22 ± 0.10	*41.32 ± 0.42*	$\mathbf{42.80 \pm 0.28}$

TABLE II

PROPERTIES OF DATASETS AND PERFORMANCES OF NEST COMPARED WITH OTHER STATE-OF-THE-ART ALGORITHMS.

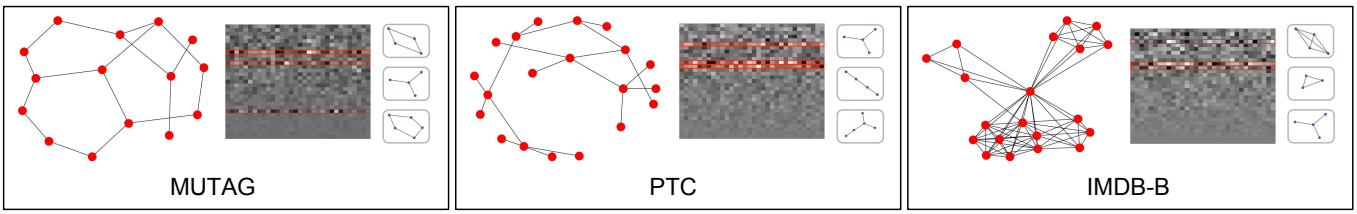


Fig. 5. Important motifs as network functional building blocks found by NEST in datasets from different domains.

Architecture	$\|\mathbf{x}\| = 8$	$\|\mathbf{x}\| = 16$	$\|\mathbf{x}\| = 32$	$\|\mathbf{x}\| = 64$
$\mathbf{O = 1, P = 1, Q = 1}$	56.48	58.53	59.16	59.14
$\mathbf{O = 2, P = 1, Q = 1}$	56.42	57.86	58.35	59.07
$\mathbf{O = 1, P = 2, Q = 1}$	56.46	56.52	57.21	57.81
$\mathbf{O = 1, P = 1, Q = 2}$	79.51	79.93	80.12	80.88
$\mathbf{O = 1, P = 1, Q = 3}$	80.46	81.23	81.59	81.67
$\mathbf{O = 1, P = 1, Q = 4}$	80.37	81.41	81.72	81.74
$\mathbf{O = 2, P = 2, Q = 3}$	80.62	81.31	81.55	81.65

TABLE III

NEST WITH DIFFERENT NEURAL ARCHITECTURES ON NCI1.

while some algorithms like SP and RW are too costly in computation and unable to be finished in several days on larger datasets, NEST is able to be easily scaled to large networks and its runtimes are of the same magnitude with the state-of-the-art neural algorithms like DGK and PSCN.

C. Q2. Robustness

As we can observe in Table II, the results of NEST with the specific default neural configuration are quite robust with relatively small variance, probably due to our convolution based on exactly matching the network structures. In this subsection, we study the robustness of NEST through an in-depth analysis on how different designs of the neural embedding architectures influence its performance. We conduct all experiments in this subsection on the two NCI benchmark datasets.

Table III lists the performance of NEST with varying neural architectures on NCI1. The results for NCI109 are almost identical to NCI1 and are thus omitted. O, P, Q are the numbers of layers in the node-level, motif-level, subgraph-level embedding, respectively. As we can observe, when all three layers are composed of only one linear fully connected neural network without nonlinear activation, the performance is rather poor– only slightly better than random guess given the fact that the two NCI datasets are both balanced with binary class labels. As we stack linear and nonlinear layers to the node-level or motif-level embedding alone, the performance does not increase, probably because linear combinations are sufficient for most node-level and motif-level patterns. However, as we stack linear and nonlinear layers to the subgraph-level embedding, the performance improves significantly, especially when the first linear and nonlinear layers are added. This indicates the significance and the power of NEST to explore complex subgraph patterns as arbitrary combinations of motif-level patterns. Finally, the improvements brought by increasing the embedding sizes are marginal but still observable in most cases, especially when the sizes are small.

The performance of NEST is generally robust when the neural networks are sufficiently complex. Since network patterns as functional blocks are repeated many times even in a few small subnetworks, the model does not easily overfit the data.

Figure 4 shows the performance and runtime of NEST on NCI1 when we use different sets of enumerated motifs as convolution windows. We experiment on motif sets with maximum number of nodes from 1 to 6. Other parameters are set as default. The runtimes are recorded on a local PC with two 2.5 GHz Intel i7 CPUs and 32GB memory (the offline motif enumeration and matching time before training are excluded). It can be observed that the size of motifs being considered has a significant influence on the performance of NEST, especially when the considered motifs are small. Motifs of sizes 1 and 2 alone obviously do not deliver satisfactory results, but including those of sizes 3, 4 and 5 clearly helps, probably by covering most basic functional building blocks in the network. The runtimes increase drastically as the sizes of motifs grow, basically because of the combinatorial complexity of larger motifs, leading to large numbers of neural cells in the motif-level embedding layers. Fortunately, motifs larger than 6 do not seem to contribute significantly in boosting the performance of NEST, because exact instances of large motifs are usually scarce and thus hard to leverage.

D. Q3. Interpretation

In this section, we study the power of NEST to discover network functional building blocks and provide insightful interpretations into the formation of different networks. Figure 5 illustrates some results we got on three datasets, *i.e.*, MUTAG, PTC and IMDB-B. For each dataset, we present an example positive subnetwork and visualize the 31×32 neuron outputs in the motif-embedding layer (31 motifs and 32 neurons) when this example subnetwork is input into a NEST model trained with all data in the corresponding dataset. Then we pick out the first 3 rows in the neuron output matrix which have the largest sum of absolute values, roughly indicating the high activeness of the corresponding motifs, which are also listed in the figure. These motifs are frequently seen in many subnetworks and may well be seen as discriminative patterns with specific functions.

VI. Conclusion

In this paper, we develop a novel neural architecture to compute multi-resolution network embeddings through a three-level hierarchy: node, motif and subgraph. NEST is developed with special consideration of network complexity, variety and intractability, to coherently explore complex network structures, node attributes and domain-specific supervision, while it is also able to capture exact small substructures and provide insightful interpretations into their various combinations as network functional building blocks. Extensive experiments on subnetwork identification in different domains show the effectiveness of our approach. For future work, it is promising to leverage the multi-resolution embeddings for node-level tasks like classification and clustering, to derive unsupervised heuristics and loss functions for unsupervised representation learning, to extend to heterogeneous networks by deriving more efficient motif enumeration methods.

References

[1] J. J. McAuley and J. Leskovec, "Learning to discover social circles in ego networks," in *NIPS*, vol. 2012, 2012, pp. 548–56.

[2] M. Belkin and P. Niyogi, "Laplacian eigenmaps and spectral techniques for embedding and clustering," in *NIPS*, 2002, pp. 585–591.

[3] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *KDD*. ACM, 2014, pp. 701–710.

[4] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *KDD*. ACM, 2016, pp. 855–864.

[5] C. Yang, L. Bai, C. Zhang, Q. Yuan, and J. Han, "Bridging collaborative filtering and semi-supervised learning: A neural approach for poi recommendation," in *KDD*, 2017, pp. 1245–1254.

[6] Z. Yang, W. Cohen, and R. Salakhutdinov, "Revisiting semi-supervised learning with graph embeddings," in *ICML*, 2016.

[7] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *NIPS*, 2013, pp. 3111–3119.

[8] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "Line: Large-scale information network embedding," in *WWW*, 2015, pp. 1067–1077.

[9] S. Cao, W. Lu, and Q. Xu, "Grarep:learning graph representations with global structural information," in *CIKM*, 2015, pp. 891–900.

[10] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *ICLR*, 2017.

[11] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *NIPS*, 2016.

[12] R. Milo, S. Shenorr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon, "Network motifs: simple building blocks of complex networks." *Science*, vol. 298, 2002.

[13] P. Yanardag and S. Vishwanathan, "Deep graph kernels," in *KDD*. ACM, 2015, pp. 1365–1374.

[14] A. Narayanan, M. Chandramohan, L. Chen, Y. Liu, and S. Saminathan, "subgraph2vec: Learning distributed representations of rooted sub-graphs from large graphs," in *KDD Workshop*, 2016.

[15] J. B. Tenenbaum, V. De Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, no. 5500, pp. 2319–2323, 2000.

[16] D. Wang, P. Cui, and W. Zhu, "Structural deep network embedding," in *KDD*, 2016, pp. 1225–1234.

[17] C. Yang, H. Lu, and K. C.-C. Chang, "Cone: Community oriented network embedding," *arXiv preprint arXiv:1709.01554*, 2017.

[18] A. R. Benson, D. F. Gleich, and J. Leskovec, "Higher-order organization of complex networks." *Science*, vol. 353, no. 6295, p. 163, 2016.

[19] P. Li and O. Milenkovic, "Inhomogeneous hypergraph clustering with applications," in *NIPS*, 2017, pp. 2308–2318.

[20] P. Li, H. Dau, G. Puleo, and O. Milenkovic, "Motif clustering and overlapping clustering for social network analysis," in *INFOCOM*, 2017, pp. 1–9.

[21] N. Shervashidze, S. V. N. Vishwanathan, T. H. Petri, K. Mehlhorn, and K. M. Borgwardt, "Efficient graphlet kernels for large graph comparison," *Aistats*, pp. 488–495, 2009.

[22] N. Shervashidze, P. Schweitzer, E. J. V. Leeuwen, K. Mehlhorn, and K. M. Borgwardt, "Weisfeiler-lehman graph kernels," *JMLR*, vol. 12, no. 3, pp. 2539–2561, 2011.

[23] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *NIPS*, 2012, pp. 1097–1105.

[24] M. Niepert, M. Ahmed, and K. Kutzkov, "Learning convolutional neural networks for graphs," in *ICML*, 2016, pp. 2014–2023.

[25] A. Coates and A. Y. Ng, "Selecting receptive fields in deep networks," in *NIPS*, 2011, pp. 2528–2536.

[26] S. F. Mousavi, M. Safayani, A. Mirzaei, and H. Bahonar, "Hierarchical graph embedding in vector space by graph pyramid," *Pattern Recognition*, vol. 61, pp. 245–254, 2017.

[27] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *NIPS*, 2017, pp. 1025–1035.

[28] C. Yang, L. Zhong, L.-J. Li, and L. Jie, "Bi-directional joint inference for user links and attributes on large social graphs," in *WWW*, 2017, pp. 564–573.

[29] J. Tang, M. Qu, and Q. Mei, "Pte: Predictive text embedding through large-scale heterogeneous text networks," in *KDD*, 2015, pp. 1165–1174.

[30] C. Yang, Z. Liu, D. Zhao, M. Sun, and E. Y. Chang, "Network representation learning with rich text information." in *IJCAI*, 2015, pp. 2111–2117.

[31] S. V. N. Vishwanathan, N. N. Schraudolph, R. Kondor, and K. M. Borgwardt, "Graph kernels," *JMLR*, vol. 11, no. Apr, pp. 1201–1242, 2010.

[32] Y. Fang, W. Lin, V. W. Zheng, M. Wu, K. C. Chang, and X. Li, "Semantic proximity search on graphs with metagraph-based learning," in *ICDE*, 2016, pp. 277–288.

[33] D. Kinga and J. B. Adam, "A method for stochastic optimization," in *ICLR*, 2015.

[34] A. K. Debnath, L. D. C. Rl, G. Debnath, A. J. Shusterman, and C. Hansch, "Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. correlation with molecular orbital energies and hydrophobicity." *JMC*, vol. 34, no. 2, p. 786, 1991.

[35] H. Toivonen, A. Srinivasan, R. D. King, S. Kramer, and C. Helma, "Statistical evaluation of the predictive toxicology challenge," *Bioinformatics*, vol. 19, no. 10, pp. 1183–93, 2003.

[36] K. M. Borgwardt, C. S. Ong, S. Schnauer, S. V. N. Vishwanathan, A. J. Smola, and H. P. Kriegel, "Protein function prediction via graph kernels." *Bioinformatics*, vol. 21 Suppl 1, 2005.

[37] P. D. Dobson and A. J. Doig, "Distinguishing enzyme structures from non-enzymes without alignments," *Journal of Molecular Biology*, vol. 330, no. 4, pp. 771–783, 2003.

[38] N. Wale and G. Karypis, "Comparison of descriptor spaces for chemical compound retrieval and classification," in *ICDM*, 2006, pp. 678–689.

[39] K. M. Borgwardt and H. P. Kriegel, "Shortest-path kernels on graphs," in *ICDM*, 2005.

[40] T. Gärtner, P. Flach, and S. Wrobel, "On graph kernels: Hardness results and efficient alternatives," *Lecture Notes in Computer Science*, vol. 2777, pp. 129–143, 2003.