# On the Possibility of Structure Learning-Based Scene Character Detector

Yugo Terada, Rong Huang, Yaokai Feng and Seiichi Uchida
*Kyushu University, Fukuoka, Japan*

***Abstract*—In this paper, we propose a structure learning-based scene character detector which is inspired by the observation that characters have their own inherent structures compared with the background. Graphs are extracted from the thinned binary image to represent the topological line structures of scene contents. Then, a graph classifier, namely gBoost classifier, is trained with the intent to seek out the inherent structures of character and the counterparts of non-character. The experimental results show that the proposed detector achieves the remarkable classification performance with the accuracy of about 70%, which demonstrates the existence and separability of the inherent structures.**

## I. Introduction

Detecting characters in natural scenes has been widely considered as an indispensable technical prerequisite for a variety of applications such as semantic image archive, scene matching and automatic navigation. However, scene character detection is still a difficult task full of challenges. As demonstrated by Epshtein *et al.* in [1], off-the-shelf Optical Character Recognition (OCR) engines would lost their power when equipped for reading scene characters. The limited capability of OCR has spurred a great deal of intensive research activities.

According to the comprehensive survey [2], existing methods could be roughly classified into two categories: texture-based and region-based. Texture-based methods assume that characters have especial textural features compared with the background. Typically, local regions are cut out of the original image by sliding a window. For each local region, various features like spatial variance, Fourier coefficient, wavelet coefficient or Local Binary Pattern (LBP) are extracted, and then fed into a classifier to determine the existence of characters. Kim *et al.* [3] fed raw pixel-based patterns into a Support Vector Machine (SVM), then identified text regions through an adaptive mean shift algorithm. Chen *et al.* [4] selected reliable text features, and utilized a cascade AdaBoost classifier to locate candidate text regions. After adaptive binarization, a commercial OCR was applied to filter out non-text regions. Ye *et al.* [5] extracted multiscale wavelet feature, and located text line based on a coarse-to-fine classifier. However, texture-based methods usually require an exhaustive scan of the original image due to the lack of prior knowledge of character size. This process dramatically increases the computational complexity, and thus restricts the real-time applications.

Region-based methods focus on the properties that the scene character generally takes the appearance of approximately uniform color, stroke width and high contrast edge. Versatile image operator, for example binarization, is first applied to the input image to get Connected Components (CCs). Then, heuristic rules or classifiers help to filter out non-character CCs. Shivakumara *et al.* [6] performed K-means to classify all pixels into text and non-text clusters in a maximum difference map, and utilized skeleton to form text strings with multiple orientations. Shi *et al.* [7] employed Maximally Stable Extremal Regions (MSERs) as the CC detector, and removed non-character CCs via a graph cut algorithm. Chen *et al.* [8] proposed an edge-enhanced MSERs and utilized distance transform to efficiently compute the stroke width. Epshtein *et al.* [1] invented a Stroke Width Transform (SWT) operator which sought to calculate pixel-level stroke width value by shooting rays from edge pixels along the gradient direction.

According to the report of ICDAR 2011 robust reading competition [9], the existing methods performed far from ideal. This is because the scene characters appear with a great variety of sizes, fonts, colors and special decoration. Moreover, their appearances may be further distorted by perspective, blur, non-uniform lighting, etc. Nevertheless, based on our observation, no matter how protean the scene characters are, their appearances would hold constant structures with the intent to preserve the semantic information. A pioneer work [10] bridged the gap between the highly decorated characters and the limited capability of OCR through the essential character structure. Unfortunately, directly extending this method to accommodate the scene character is not a trivial issue.

In this paper, we propose a structure learning-based method for scene character detection, which effectively leverages a graph classifier, namely gBoost classifier [11], to probe the inherent structures of character and the counterparts of non-character. In order to confirm the existence and separability of the inherent structures, in the current trial, the scene character detector only utilizes the topological line structure, and works independent of some frequently-used properties like texture, edge and stroke width. The quantitative experimental results demonstrate that the proposed method has a remarkable classification capability between the scene characters and the background with the accuracy of around 70%, indicating that the structure learning-based approach is a promising research direction.

The remainder of the paper is organized as follows. Section II elaborates the proposed structure learning-based scene character detector. In section III, we conduct the experiment, and exhibit the corresponding results. Section IV concludes the whole paper and outlines our future works.

## II. Methodology

We first give an overall description of the proposal. The proposed method starts with binarization and CC analysis. After necessary preprocessing, a thinning processing is applied to each CC for representing its topological structure. Graphs are extracted from the thinned image, and normalized to a roughly similar size. Before the gBoost training stage, a gSpan method [12] is adopted to mine and enumerate frequent subgraphs via DFS (Depth First Search) without duplication. Then, gBoost iteratively selects subgraphs from a DFS code tree and updates

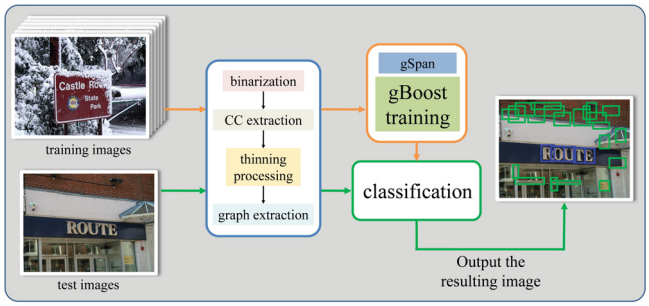


Fig. 1: Diagram of the structure learning-based detector.

the optimal solutions until a stopping criterion is achieved. For each input graph, the trained gBoost classifier makes a decision, namely character or non-character graph, through checking the presence or absence of subgraphs, and combining the corresponding weights. Compared with AdaBoost, gBoost behaves more compatibly with respect to the fewer iterations [11]. A diagram of the proposed detector is illustrated in Fig.1. We describe the details in the following.

### A. Binarization and Thinning Processing

As a widely used operator in the scene character detection community, binarization operator potentially considers the essential connectivity, and thus can preserve the shape of objects under an appropriate threshold setting. Global binarization method is simple and lightweight since only one threshold is required to determine. However, global method may erase the wanted characters no matter how carefully one tunes the threshold, especially when the characters are embedded into the complex background or suffer from specularities. See an example in Fig.2 (b).

Local binarization method is generally regarded as a straightforward way to alleviate this problem. In this paper, we employ the Niblack algorithm which is a representative local method to binarize the grayscale images. To generate local regions, a square is slid by one pixel along the original grayscale image from the upper left corner right and down to the lower right corner. For each local region, we calculate the average $m(x, y)$ and standard deviation $\sigma(x, y)$, where $(x, y)$ denotes the central coordinate of the local region. The local threshold $T(x, y)$ takes the form $T(x, y) = m(x, y) + k \cdot \sigma(x, y)$. The parameter $k$ is confined to the interval [0,1]. Then, a grayscale pixel $v(x, y)$ transforms to binary through

$$B(x,y) = \begin{cases} 1 & \text{if } v(x,y) > T(x,y) \\ 0 & \text{otherwise} \end{cases}$$

The contrast between Fig.2 (b) and (c) indicates that the Niblack algorithm is feasible for scene images. To detect both dark-on-bright and bright-on-dark characters, the Niblack algorithm is performed twice, once on the original grayscale image and again on its inverse version.

Then, CCs are extracted from the binary image. Since thinning processing is sensitive to noise, we filter out isolated points as well as tiny CCs in advance to prevent the annoying short lines. Moreover, the small and individual holes in CCs are filled to ease the complexity of thinning processing. Figure 2 (d) displays an example resulting image.

### B. Graph Construction and Decomposition

After the thinning processing, each CC is spanned by lines, and these support lines represent a specific topological structure, called line structure. Based on this observation, the

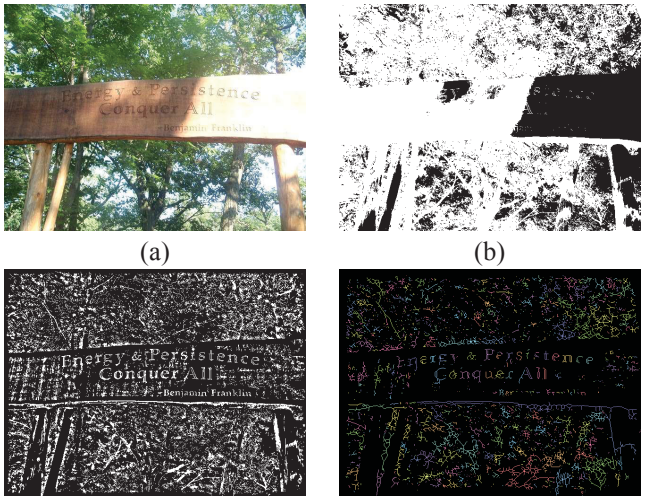


Fig. 2: Resulting images of binarization and thinning processing. (a) Original image. (b) Global binary image. (c) Niblack binary image. (d) Resulting image of thinning processing.

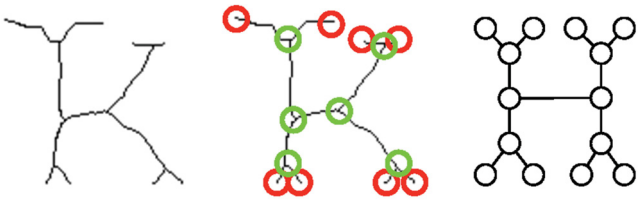


Fig. 3: Graph construction. (a) The line structure. (b) Vertices detection. (c) Constructed graph.

structure of each CC can be mapped to a graph. Specifically, the cross point and endpoint of the line structure serve as vertices, and the support lines are transmuted into the edges of a graph. For constructing a pure graph, multiple support lines between a pair of vertices are normalized into the same edge. Likewise, multiple vertices within a small range are clustered into a single vertex. Figure 3 illustrates an example of graph construction. As shown in Fig.3 (b), the green circles stand for the cross points, while the endpoints are represented by red ones. Figure 3 (c) exhibits the constructed graph corresponding to the line structure in Fig.3 (a).

Some scenes like lawn, foliage or brick wall may form a complex graph with large size, which renders the computational complexity prohibitive. To alleviate this problem, a large graph $G$ is decomposed into multiple small ones by setting a diameter threshold $D$. To this aim, we first define the distance $d_{ij}$ between vertex $v_i$ and $v_j$ as the minimum number of edges along the path from $v_i$ to $v_j$. For each vertex $v_i \in G$, the original graph $G$ is decomposed into a small size graph $G_i$ such that $d_{ij} \leq D$ holds for $\forall v_j \in G_i \subseteq G$.

Figure 4 gives an example of the graph decomposition stage. For the sake of simplicity, each vertex is labeled an artificial index number. The maximum diameter of the graph $G$ is 5 by counting the minimum number of edges along the longest paths from $v_1$ or $v_3$ to $v_9$ or $v_{11}$. If the maximum diameter exceeds the threshold $D$, the graph decomposition stage will be performed. In this example, we fix $D = 2$. Then, we can obtain ten decomposed graphs without duplication. Figure 4 (b) shows a decomposed graph $G_2$.

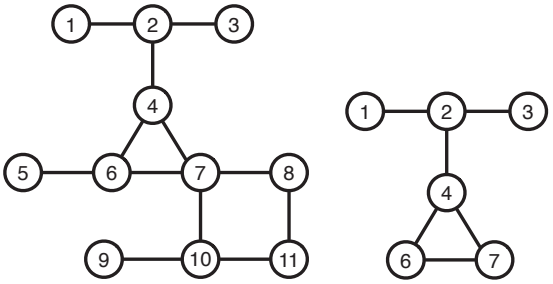


Fig. 4: An example of graph decomposition. (a) The original graph $G$. (b) A decomposed graph $G_2$.

### C. Graph Labeling

For the subsequent supervised learning stage, the extracted graphs should be labeled "character" or "non-character" according to the ground-truth image. To be specific, a graph is labeled "character" if all its vertices map to the character pixels of the ground-truth image, otherwise it will be tagged with "non-character" label.

### D. Classification by gBoost

Saigo *et al.* extended LPBoost [13] by combining an optimal pattern search algorithm, which renders gBoost more efficient in the course of learning. First, gSpan method [12] mines frequent subgraphs and constructs a canonical search space in the form of DFS code tree. The tree structure and DFS code are the prerequisites for achieving the optimal search.

Let $\{g_t\}_{t=1}^{T}$ denote a set of frequent subgraphs obtained from the gSpan. Given the learning graphs $\{(\mathbf{x}_n, y_n)\}_{n=1}^{N}$, the hypotheses are defined as:

$$h(\mathbf{x}_n, g_t) = \begin{cases} 1 & g_t \in \mathbf{x}_n \\ -1 & g_t \notin \mathbf{x}_n \end{cases}$$

where $y_n \in \{1, -1\}$ ($\{$"character", "non-character"$\}$) is the supervised label of the learning graph $\mathbf{x}_n$. Then, the training problem can be expressed below.

$$\begin{aligned} \min_{\lambda,\gamma} \ & \gamma \\ \text{s.t.} \ & \sum_{n=1}^{N} \lambda_n y_n h(\mathbf{x}_n, g_t) \le \gamma \ , \quad t = 1, 2, \cdots, T \\ & \sum_{n=1}^{N} \lambda_n = 1 \ , \ 0 \le \lambda_n \le D, \end{aligned}$$

where $D = \frac{1}{cN}$, $c \in (0, 1)$ is the parameter controlling the cost of misclassification [11], [13]. The above linear program contains intractably large number of constraints. To overcome this problem, the column generation technique is adopted by the gBoost. In each iteration, the subgraph $g_t$ whose corresponding constraint is violated the most is selected instead of considering all constraints. At the $k^{\text{th}}$ iteration, the constraints are formulated as

$$\sum_{n=1}^{N} \lambda_n^{(k)} y_n h(\mathbf{x}_n, g_t) \le \gamma^{(k)} \ , \quad t \in \mathcal{T}^{(k)},$$

where $\mathcal{T}^{(k)}$ is a set collecting the index number of the selected subgraphs, while the superscript $(k)$ stands for the current number of iterations. The set $\mathcal{T}^{(0)}$ is initialized as empty, and $\lambda_n^{(0)} = \frac{1}{N}$. The optimal solutions $\lambda_n^{(k)}$ and $\gamma^{(k)}$ are updated iteratively.

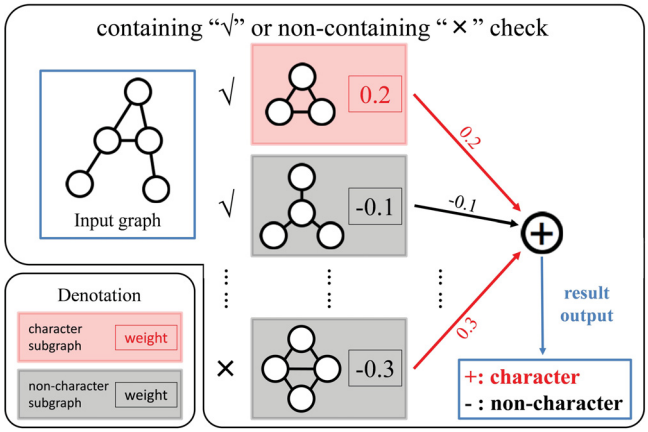


Fig. 5: An example of classification.

Then, the new subgraph which violates its constraint the most is selected. That is

$$t^{\star} = \underset{t=1,2,\cdots,T}{\arg\max} \sum_{n=1}^{N} \lambda_n^{(k)} y_n h(\mathbf{x}_n, g_t).$$

The set $\mathcal{T}^{(k)}$ is updated by adding the new index number $t^{\star}$: $\mathcal{T}^{(k+1)} = \mathcal{T}^{(k)} \cup \{t^{\star}\}$. The gBoost algorithm utilizes a tree pruning, and guarantees an optimal way to find the target subgraph $g_{t^{\star}}$. Details can be found in [11]. The learning will be stopped at iteration $K$ if all the constraints are satisfied. In other words, after the $K^{\text{th}}$ iteration, if there is no need to select more subgraphs to correct the solutions $\lambda_n^{(K)}$ and $\gamma^{(K)}$, the optimization problem of learning is solved.

After that, a sparse weight vector $\alpha$ for each subgraph can be obtained from the Lagrange multipliers [11], [13]. Note that only the selected subgraphs have nonzero weights. Under our setting $1$ : "character" label and $-1$ : "non-character" label, a positive weight means that its corresponding subgraph is the inherent structure of character. On the contrary, the negative one suggests the non-character structure. For a specific test graph $\mathbf{x}$, the decision criterion takes the following form

$$y = \text{sgn}\left( \sum_{t \in \mathcal{T}^{(K)}} \alpha_t h(\mathbf{x}, g_t) \right).$$

The operator $\text{sgn}(\cdot)$ extracts the sign of a real number. Likewise, a test graph is labeled "character" if $y = 1$, and "non-character" if $y = -1$. See an example in Fig.5.

Since one of our objectives is to confirm the existence and separability of the inherent structures, we fulfill the structure learning-based detector without the aid of some general properties like texture, color or stroke width. This is the reason why the extracted graphs are pure in the sense that neither vertices nor edges are tagged with labels. Nevertheless, it is worthwhile to point out that gBoost is compatible with labeled graphs, which implies that it is feasible and tractable to further extend the structure learning-based detector by combining the general properties or existing methods.

## III. Experiment

In this section, we evaluated the performance of the proposed detector. Both training and test images were harvested from the Internet by searching the keywords "sign" and "park". We manually prepared the ground-truth images.

As described before, the training and test graphs were extracted from the thinned binary images, and then labeled


Fig. 6: Original images, ground-truth images and the resulting images of graph classification.

"character" or "non-character" according to the ground-truth images. In this experiment, we used ten natural scene images so that there existed ten corresponding graph sets. The number of character and non-character graphs for each set was listed in the first two columns of Table I. We adopted ten-fold cross-validation, and evaluated the classification performance through following four measurements.

- True Positive (TP): an actual character graph was correctly classified as a character graph.
- False Negative (FN): an actual character graph was improperly classified as a non-character graph.
- False Positive (FP): a non-character graph was improperly classified as a character graph.
- True Negative (TN): a non-character graph was correctly classified as a non-character graph.

The last four columns of Table I exhibited the detailed results of cross-validation. Two final accuracies of classification were calculated by TP/(FN+TP) and TN/(TN+FP). The quantitative results demonstrated that the proposed detector achieved classification accuracy of around 70%, indicating the existence and separability of the inherent structures.

Intuitively, Fig.6 exhibited four classification results. The large number of green bounding boxes which covered the correctly rejected non-character graphs demonstrated that the structure learning-based detector had remarkable ability to suppress the complex backgrounds. Particularly, as shown in

TABLE I: The number of character and non-character graphs for each image and the detailed results of cross-validation.

| Image ID | #CG | #NCG | TP | FN | TN | FP |
|---|---|---|---|---|---|---|
| 1 | 21 | 878 | 16 | 5 | 529 | 349 |
| 2 | 15 | 2271 | 15 | 0 | 1562 | 709 |
| 3 | 34 | 1928 | 33 | 1 | 1245 | 683 |
| 4 | 31 | 932 | 22 | 9 | 628 | 304 |
| 5 | 23 | 1724 | 20 | 3 | 1183 | 541 |
| 6 | 73 | 1384 | 35 | 38 | 942 | 406 |
| 7 | 150 | 1360 | 99 | 51 | 1065 | 295 |
| 8 | 79 | 1685 | 75 | 4 | 1196 | 489 |
| 9 | 79 | 964 | 77 | 2 | 570 | 394 |
| 10 | 30 | 1216 | 7 | 23 | 723 | 493 |
| total | 535 | 14306 | 399 | 136 | 9643 | 4663 |
| final accuracy (%) | | | TP/(TP+FN) 74.6 | | TN/(TN+FP) 67.4 | |

#CG = the number of character graph
#NCG = the number of non-character graph

Fig.6 (d), the proposed method correctly detected most Korean characters even there were no training images containing such characters. This surprised result indicated that the structure learning stage allowed us to catch the inherent structure of multilingual characters.

We gave two successful classification examples in Fig.7. Both cropped regions were obtained from Fig.6 (a). More specifically, the "A"-like line structure located at the second row and the first alphabet position of the word "All", while the "E"-like line structure located at the leading position of the first row. Figure 7 (a.ii) and (b.ii) displayed their corresponding graphs, respectively. For the "A"-like line structure, its graph contained a character subgraph with a large weight $\alpha = 2.033 \times 10^{-11}$ as shown in Fig.7 (a.iii). This large weight would definitively affect the final decision so that the "A"-like line structure was correctly judged as character. Slightly differently, the "E"-like line structure contained a lightweight character subgraph ($\alpha = 2.399 \times 10^{-15}$). Fortunately, there did not exist other heavyweight non-character subgraphs, either. As a result, the character "E" was also successfully detected.

## IV. Conclusion

In this paper, we employ gBoost to probe the inherent structures of character and the counterparts of non-character. Graphs are extracted from the thinned binary images, and normalized to a similar size in order to ease the complexity of learning. The gSpan method mines the frequent subgraphs, and

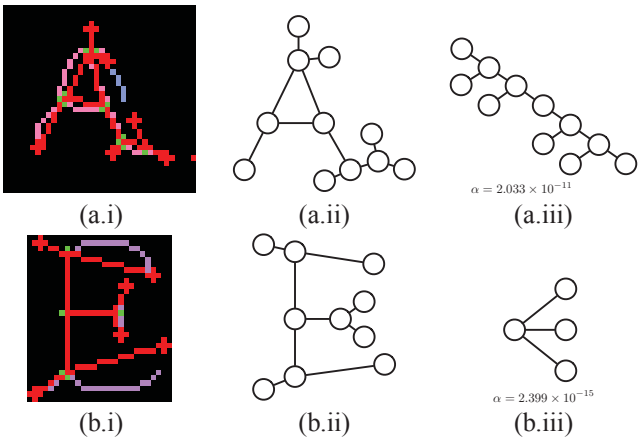


Fig. 7: Two successful examples.

organizes them into a DFS code tree. Then, gBoost iteratively selected subgraphs and updated solutions in an efficient way. The proposed method is fulfilled without the aid of some popular properties like texture, color and stroke width with the intent to confirm the feasibility of structure learning-based research direction. The experimental results demonstrate that the trained classifier has remarkable ability to distinguish characters and non-characters with the accuracy of about 70%. Moreover, benefitting from searching the essential structure information, the proposed detector behaves flexible in the sense of multilingual compatibility. This paper takes the first step towards a structure learning-based method. Our future works include attribute preserving graph construction, gBoost with labels and multilingual analysis.

## References


[1] B. Epshtein, E. Ofek, and Y. Wexler, Detecting Text in Natural Scenes with Stroke Width Transform, *CVPR*, 2010.

[2] K. Jung, K. Kim and A. Jain, Text Information Extraction in Images and Video: A Survey, *Pattern Recognition*, vol.37, no.5, pp.977-997, 2004.

[3] K. Kim, K. Jung and J. Kim, Texture-Based Approach for Text Detection in Images Using Support Vector Machines and Continuously Adaptive Mean Shift Algorithm, *PAMI*, vol.25, no.12, pp.1631-1639, 2003.

[4] X. Chen and A. Yuille, Detecting and Reading Text in Natural Scenes, *CVPR*, 2004.

[5] Q. Ye, Q. Huang, W. Gao and D. Zhao, Fast and Robust Text Detection in Images and Video Frames, *Image and Vision Computing*, vol.23, no.6, pp.565-576, 2005.

[6] P. Shivakumara, T. Phan and C. Tan, A Laplacian Approach to Multi-Oriented Text Detection in Video, *PAMI*, vol.33, no.2, pp.412-419, 2011.

[7] C. Shi, C. Wang, B. Xiao, etc., Scene Text Detection Using Graph Model Built Upon Maximally Stable Extremal Regions, *Pattern Recognition Letters*, vol.34, no.2, pp. 107-116, 2013.

[8] H. Chen, S. Tsai, G. Schroth, etc., Robust Text Detection in Natural Images with Edge-Enhanced Maximally Stable Extremal Regions, *ICIP*, 2011.

[9] A. Shahab, F. Shafait and A. Dengel, ICDAR 2011 Robust Reading Competition Challenge 2: Reading Text in Scene Image, *ICDAR*, 2011.

[10] S. Omachi, M. Inoue and H. Aso, Structure Extraction from Decorated Characters Using Multiscale Images, *PAMI*, vol.23, no.2, pp.315-322, 2001.

[11] H. Saigo, S. Nowozin, T. Kadowaki, etc., gBoost: A Mathematical Programming Approach to Graph Classification and Regression, *Machine Learning*, vol.75, no.1, pp.69-89, 2009.

[12] X. Yan and J. Han, gSpan: Graph-Based Substructure Pattern Mining, *ICDM*, 2002.

[13] A. Demiriz, K. Bennett and S. John, Linear Programming Boosting via Column Generation, *Machine Learning*, vol.46, no.1-3, pp.225-254, 2002.