D4C Recurrent Network: Deep Learning Based DDoS Attack Identification

1st Yichi Zhang

Zhejiang University
Hangzhou, China
3210103159@zju.edu.cn

***Abstract*—The growth of Distributed Denial of Service (DDoS) attacks has been rapid, making it one of the most lethal threats to the internet. To defend against such attacks, automatic detection of DDoS attack packets is required. The traditional defense strategy is to monitor network traffic and identify attack activity from legitimate traffic based on statistical differences. Machine learning is another method that improves identification performance based on statistical features. However, traditional machine learning techniques are limited by shallow representation models. In this paper, we propose a deep learning-based method for DDoS attack detection. Deep learning methods can automatically extract high-level features from low-level features and obtain powerful representation and inference capabilities. We designed a recurrent deep neural network that learns patterns from sequences of network traffic and tracks network attack activity. We named it D4C (Directional Deep DDoS Detection Convolutional Recurrent Network).**

***Index Terms*—DDoS Attack, Deep Learning, RNN, LSTM**

I. Introduction

Denial of Service (DoS) attacks deny legitimate users access to shared services and resources [1]. Distributed Denial of Service (DDoS) refers to coordinated DoS attacks launched by attackers using multiple distributed computing resources against one or more targets [2]. Such attacks primarily target system resources and network bandwidth, ranging from the network layer to the application layer. Since the first DDoS attack surge in 1999 [3], DDoS has become a critical, widespread, and rapidly evolving security threat worldwide. Currently, the main attack vectors of DDoS include UDP flood, HTTP flood, SYN flood, ICMP, DNS, etc. [4], posing severe threats to systems and networks.

DDoS detection is one of the primary DDoS defense mechanisms. However, DDoS attacks are difficult to automatically detect because in most cases, attack traffic is very similar to legitimate traffic. In the early stages, attacks with insufficient traffic were even considered legitimate traffic [5]. Many researchers have tried to use statistical and machine learning methods or a combination of both to identify DDoS attacks. However, they still have the following drawbacks: (1) Appropriate statistical features require extensive and in-depth knowledge of network security and a large number of DDoS attack experiments to determine. (2) The defense object is limited to one or a few DDoS vectors. (3) It requires frequent updates of its models and thresholds to keep up with the changes in systems and attack vectors. (4) It's difficult to detect slow-rate DDoS attacks.

In this paper, we propose a deep learning-based method to detect DDoS attacks in legitimate network traffic on the victim side. We named it **D4C** (**D**irectional **D**eep **D**DoS **D**etection **C**onvolutional Recurrent Network). We trained **D4C** using a large-scale dataset, the UNB ISCX Intrusion Detection Evaluation 2012 DataSet (referred to as ISCX2012 in this paper) [6], to address the complex identification problem.

The rest of this paper is organized as follows. Section II explains related work. In Section III, we propose **D4C**, data preprocessing method, and overall architecture. Section IV describes the experiments conducted on the ISCX2012 dataset, and compares our results with shallow machine learning methods. Finally, Section V summarizes our work.

II. Related Work

A. DDoS Attacks and Countermeasures

Host resources and network bandwidth are the two main targets of DDoS attacks. Most attacks target vulnerabilities in protocols and applications, such as SYN flood, UDP flood, ICMP flood, SIP flood, etc. Some attacks, such as UDP flood and ICMP flood, consume network bandwidth, while others, such as SYN flood and SIP flood, also consume victim's system resources (such as CPU and memory) [7]. DDoS attacks also exploit techniques such as IP spoofing, network amplifiers/reflectors, etc., to combine attack methods and avoid detection, thus causing more impact [5].

To defend against DDoS attacks, some solutions use both preventive and reactive mechanisms to mitigate the impact of DDoS attacks on victim networks, intermediate networks, and source networks [8]. The most commonly used mechanisms include attack prevention, attack detection, and attack response. Attack prevention attempts to filter inbound and outbound traffic before the attack causes damage. Attack response aims to minimize the losses caused by DDoS attacks.

D-WARD is a DDoS defense system deployed in the source-side network, which monitors bidirectional traffic. D-WARD proposes the ability to detect DDoS attacks based on statistical comparison between legitimate traffic and DDoS traffic for different protocols [9]. Bhuyan and Kalita compared four important information entropy metrics (i.e., Hartley entropy,

Shannon entropy, Renyin++s entropy, and Renyin++s generalized entropy) in DDoS detection [10]. Based on these four information entropy methods, they calculated the information distance in network traffic and detected low-rate and high-rate DDoS attacks. Chen identified DDoS attacks by conducting two statistical t-tests on the SYN arrival rate and the number of SYN and ACK packets [11]. Most statistical methods perform well in specific DDoS attack methods, but require a lot of effort in feature selection and preprocessing indices.

B. DDoS Defense Based on Machine Learning

Machine learning algorithms have been widely applied in DDoS defense, especially in the anomaly detection stage. The most commonly used algorithms include Naive Bayes, neural networks, support vector machines, decision trees, and K-nearest neighbors. Figure 1 depicts a general architecture of a DDoS attack detection system. In the first step, network traffic is filtered through filtering rules and stored in a database. Features are then extracted from the traffic, such as packet rate and protocol type, and normalized to speed up the training process. The training data is then used to train machine learning algorithms to classify each packet in real network traffic as either a DDoS attack or legitimate traffic. In the final step, the system discards DDoS packets detected by the machine learning algorithms and updates its filtering rules.

Xu, Sun, and Huang used Hidden Markov Models (HMM) and reinforcement learning to differentiate between legitimate traffic and DDoS attacks based on observed source IP advertisements [12]. Detection agents were placed at intermediate network nodes or near DDoS attack sources. HMM was proposed to compute the probability of a specific observation sequence for a new IP address. They claimed that in DDoS attacks, most source IP addresses are new to the victim. Similar to their work, Berral et al. collected traffic information from intermediate networks and let each node learn independently and share information [13]. Each node was equipped with Naive Bayes algorithm to detect DDoS attacks based on source and destination addresses and shared information. Their method reduces reaction time and mitigates the impact of DDoS attacks. Shon et al. proposed a Genetic Algorithm (GA) and Support Vector Machine (SVM) [14]. They included more network traffic domains by using GA-based feature selection. Then they classified packets using SVM to detect DDoS attacks.

In conclusion, DDoS detection methods using machine learning algorithms can effectively mitigate the impact of DDoS attacks on networks. When choosing methods and algorithms, it is essential to consider the type and characteristics of the attack to achieve better detection performance.

III. Deep Learning Based Approach

In general, it is difficult to detect low-rate attacks because they appear similar to legitimate network traffic from the victim's end. However, we note that DDoS attacks on victim systems must be sustained over a long period of time. Otherwise, they would not be malicious to network/system resources. This indicates the importance of historical information in DDoS detection. Since traditional learning models lack historical patterns, packet-based detection methods cannot improve the classifier's performance. Therefore, we designed a DDoS classifier based on Recurrent Neural Networks (RNN).

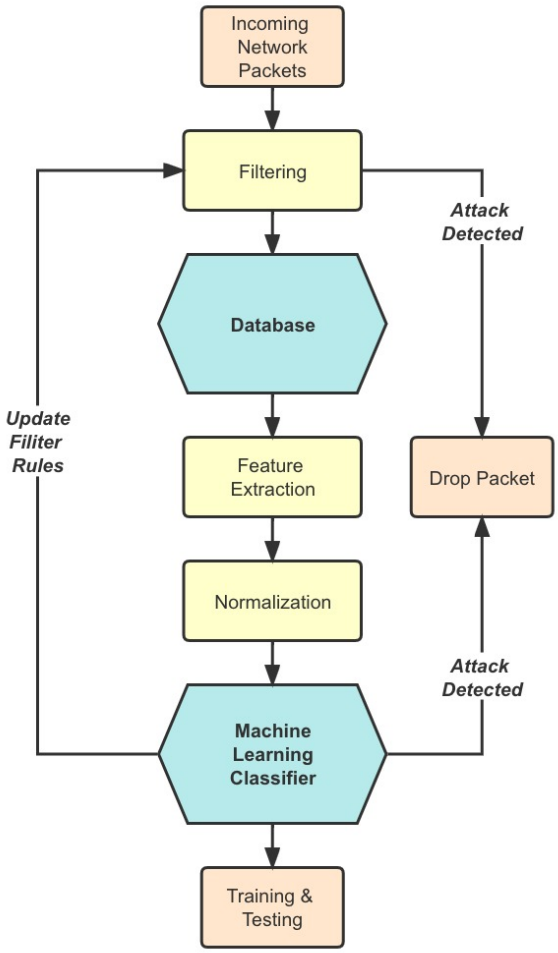

Fig. 1. Architecture of DDoS Attack Detection Based on Machine Learning

Our detection method utilizes continuous sequences of network packets and is able to discern subtle differences between attack traffic and legitimate traffic. Historical information is fed into the RNN model to identify DDoS attacks. It helps to find repeating patterns that represent DDoS attacks and locate them in long-term traffic sequences. Another advantage of RNN is its independence from the input window size. The window size used in previous machine learning methods is often task-dependent, which limits their ability to detect different types of attacks. Moreover, traditional machine learning methods are difficult to train for long-term sequences. However, RNN (especially gated RNN such as LSTM, GRU) has shown the ability to address these issues [15]. Considering these factors, we chose RNN as the main component of our DDoS classifier.

A. Feature Extraction and Transformation

We first extracted 28 network traffic fields from the ISCX2012 dataset. Table I shows examples and types of these fields. We used them as training features. Unlike other machine learning methods applied to DDoS detection, we do not need to select statistical features in our model.

TABLE I
NETWORK TRAFFIC FIELDS

Field	Field Example	Field Type
frame.len	206	Numerical
ip.hdr_len	20	Numerical
frame.protocols	eth:ethertype:ip:tcp:ssh	Text
ip.len	192	Numerical
ip.flags.rb	0	Boolean
ip.flags.df	1	Boolean
ip.flags.mf	0	Boolean
ip.frag_offset	0	Boolean
ip.ttl	128	Numerical
ip.proto	6	Numerical
ip.src	192.168.1.101	Text
ip.dst	192.168.5.122	Text
tcp.srcport	4175	Numerical
tcp.dstport	22	Numerical
tcp.len	152	Numerical
tcp.ack	1	Numerical
tcp.flags.res	0	Boolean
tcp.flags.ns	0	Boolean
tcp.flags.cwr	0	Boolean
tcp.flags.ecn	0	Boolean
tcp.flags.urg	0	Boolean
tcp.flags.ack	1	Boolean
tcp.flags.push	1	Boolean
tcp.flags.reset	0	Boolean
tcp.flags.syn	0	Boolean
tcp.flags.fin	0	Boolean
tcp.window_size	16697	Numerical
tcp.time_delta	0.000537000	Numerical

We classify fields into two types: Text fields, Boolean fields and numeric fields. For most Text fields , we use the LabelEncoder method to encode them. and Boolean fields, we perform transformations. We convert the majority of Boolean fields to 0 and 1.

However, due to computational resource constraints, we only have a limited number of training samples (50,000), and as a result, most of the ip.src and ip.dst for attack and normal traffic are similar. This type of data can lead to overfitting of the model. Therefore, in this experiment, we did not train on these three Text field features.

After feature transformation, we obtain an $m \times n'$ matrix, where m represents the number of network packets and n' represents the number of new features after transformation. To learn long-term and short-term patterns, we use sliding windows to separate continuous packets and reshape the data into a series of time windows of size T, where the label p in each window indicates the class of the last packet. After reshaping, we have a three-dimensional matrix of shape $(m-F) \times T \times n'$. Figure 2 illustrates the workflow of feature extraction, transformation, and reshaping.

Using this approach, we transform the traditional packet-based features into window-based features, which allows us to learn network patterns from the previous (T-1) packets and the current packet.

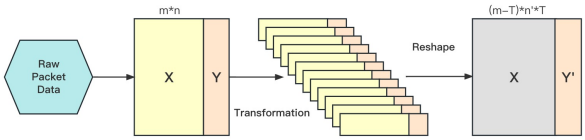

Fig. 2. Overall Network Architecture of **D4C**

B. Bidirectional Recurrent Neural Network Architecture

In this section, we designed a bidirectional recurrent neural network (RNN) in our **D4C** model. Each direction consists of two sequence-to-sequence recurrent layers. The recurrent layers help us to trace the history from previous network packets. To address the issue of the exploding gradients in deep RNN networks, we experimented with both LSTM and GRU in this paper. Specifically, LSTM aims to overcome the gradient vanishing problem of RNNs and uses a memory cell to represent previous timestamps [16]. GRU is a simpler variant of the standard LSTM and can be trained faster due to fewer parameters. The modified LSTM currently includes three gates in each unit: input, forget, and output. They are calculated as follows:

$$
\begin{aligned}
i_t &= \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{t-1} + b_{hi}) \\
f_t &= \sigma(W_{if}x_t + b_{if} + W_{hf}h_{t-1} + b_{hf}) \\
g_t &= \tanh(W_{ig}x_t + b_{ig} + W_{hg}h_{t-1} + b_{hg}) \\
o_t &= \sigma(W_{io}x_t + b_{io} + W_{ho}h_{t-1} + b_{ho}) \\
c_t &= f_t \odot c_{t-1} + i_t \odot g_t \\
h_t &= o_t \odot \tanh(c_t)
\end{aligned}
$$

i_t, f_t, and o_t denote the outputs of the input gate, forget gate, and output gate, respectively. g_t represents the candidate cell state at the current time step. $\sigma(\cdot)$ denotes the sigmoid activation function, and $\odot$ represents element-wise multiplication. W_{ii}, W_{if}, W_{ig}, W_{io}, W_{hi}, W_{hf}, W_{hg}, and W_{ho} correspond to the weight matrices for the input feature, the previous hidden state, and the gate unit at the current time step, respectively. b_{ii}, b_{if}, b_{ig}, b_{io}, b_{hi}, b_{hf}, b_{hg}, and b_{ho} correspond to the bias vectors for the respective gates.

In the model, we concatenate the outputs of the forward and backward LSTM layers along the time dimension and then connect them to subsequent layers. We designed fully connected layers before the recurrent neural network and used the Sigmoid function as the output function. To capture local information and simplify the deep neural network, we tried stacking 1D convolutional neural network layers before the recurrent neural network, using Rectified Linear Units (ReLU) as the activation function. The convolutional neural network layer has a kernel size of 3 and a stride of 1. After every two recurrent neural network layers and every fully connected layer, a batch normalization layer follows to accelerate the

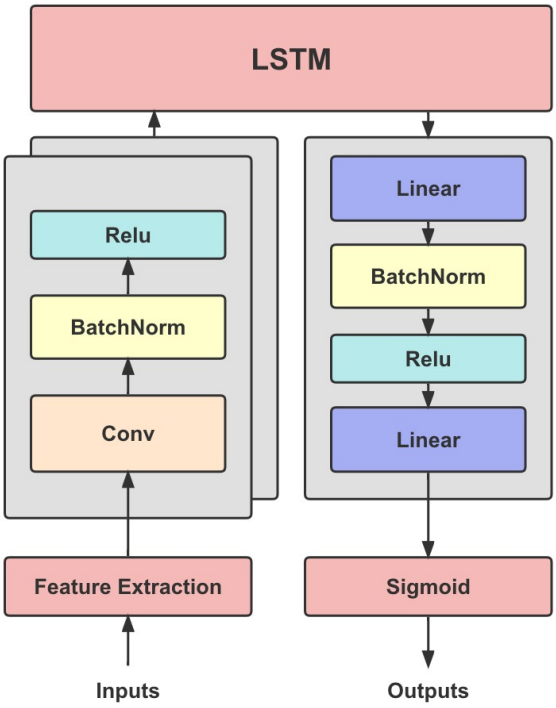

Fig. 3. Overall Network Architecture of **D4C**

training of the deep neural network. Figure 3 gives a more detailed model diagram.

IV. Experiment

A. Dataset Description and Preprocessing

We evaluated our **D4C** model on the ISCX2012 dataset, and due to computational resource limitations, we only selected 50,000 data records for training and testing.

B. Experimental Settings and Results

In this section, we mainly investigate the prediction accuracy of LSTM model, GRU-**D4C**(use GRU instead the LSTM) model, Bi-**D4C**(use bidirectional LSTM, the baseline model) model, and Uni-**D4C**(use unidirectional LSTM) model after undergoing the same training; the training performance and prediction accuracy of Bi-**D4C** model under different sliding window sizes.

To ensure the credibility of the experiment, we started our experiments with the initial parameters of $learning\_rate = 0.001$, $num\_epoch = 40$, $batch\_size = 128$, Adam optimizer, and using BCELoss. We also split the dataset into training and testing sets with a ratio of 4:1.

To ensure the reliability of the experiment, each data point was obtained from ten identical tests.

Comparison among Recurrent Neural Network models: In this experiment, we set the sliding window size to 25 for uniform training. After training, the specific test data is shown in Table II.

TABLE II
Comparison among **D4C** models

Model	Error Rate	Accuracy
LSTM	7.86%	92.14%
Bi-D4C	0.79%	99.21%
GRU-D4C	1.96%	95.04%
Uni-D4C	2.36%	97.64%

This table shows the results of testing different recurrent neural network models (LSTM model, GRU-**D4C** model, Bi-**D4C** model, and Uni-**D4C** model) on the same dataset. By comparing the error rates and accuracies of these models, the following conclusions can be drawn: On this dataset, the Bi-**D4C** model performed the best, with the lowest error rate and highest accuracy of 0.26% and 99.74%, respectively, which is better than all other models. The LSTM model performed worst, with an error rate of 7.86% and an accuracy of 92.14%.The performance of the GRU-**D4C** and Uni-**D4C** models was relatively poor, but they still achieved relatively high accuracies of 99.04% and 98.64%, respectively. Overall, if choosing a model to implement DDoS attack detection, the Bi-**D4C** model is the best choice as it has the lowest error rate and highest accuracy.

*Bi-**D4C** in Different Sliding Window Size:* In this experiment, we trained our model with different sizes of sliding window, and the specific test data after training is shown in Table III.

TABLE III
Bi-**D4C** in Different Sliding Window Size

Window Size	Error Rate	Accuracy	Training Time
5	6.32%	93.68%	5min37s
10	3.39%	96.61%	12min6s
20	1.24%	98.76%	22min58s
25	0.79%	99.21%	28min37s
50	0.01%	99.99%	1h2min12s

This table shows the performance of the Bi-**D4C** model using different sliding window sizes on a dataset. The dataset may be a network security dataset, as the model name includes DDoS (Distributed Denial of Service) detection.

Specifically, the table lists five sliding window sizes: 5, 10, 20, 25, and 50. For each window size, the table lists three metrics: error rate, accuracy, and training time. The error rate and accuracy are expressed as percentages, and the training time is given in minutes and seconds.

From the table, it can be seen that as the sliding window size increases, the performance of the model improves significantly. When the sliding window size is 50, the model has the lowest error rate of only 0.01% and the highest accuracy of 99.99%. However, the training time also increases with the window size, from 5 minutes and 37 seconds (window size of 5) to 1 hour, 2 minutes, and 12 seconds (window size of 50).

Therefore, the data in this table suggest that using larger sliding window sizes can improve the performance of the Bi-D4C model, but at the cost of longer training times. These

results may be useful for researchers and practitioners in the field of network security, who can choose an appropriate window size based on their needs and constraints to balance model performance and training time.

V. Conclusion

In this paper, we propose a deep learning-based DDoS attack detection technique, and design a basic DDoS attack classifier **D4C** based on this technique that can achieve high-precision identification of DDoS attack traffic and normal user access traffic. However, due to the lack of computational resources, we cannot test larger models on larger datasets and can only showcase the potential of this technique using a small model.

Our current ISCX2012 dataset was provided by UNB in 2012.

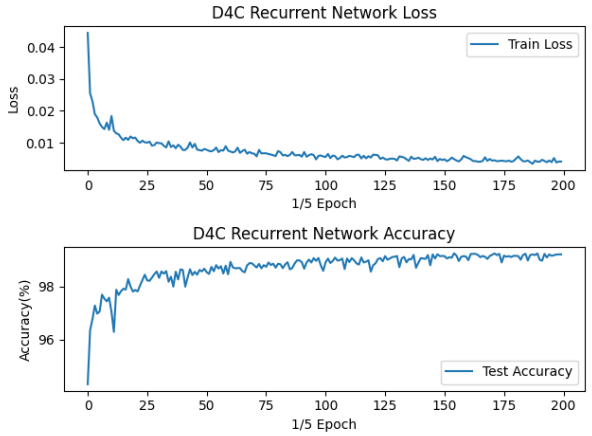

Fig. 4. Loss & Accuracy of **D4C** during training

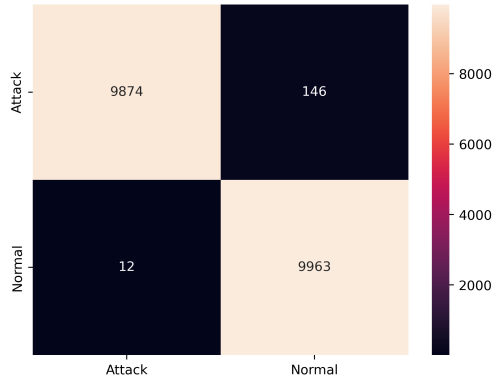

Fig. 5. Confusion Matrix of **D4C** during testing

References

[1] V. D. Gligor, "A note on denial-of-service in operating systems," *IEEE Transactions on Software Engineering*, no. 3, pp. 320–324, 1984. I

[2] L. D. Stein, *World Wide Web Security FAQ*. Lincoln D. Stein., 2002. I

[3] P. J. Criscuolo, "Distributed denial of service: Trin00, tribe flood network, tribe flood network 2000, and stacheldraht ciac-2319," California Univ Livermore Radiation Lab, Tech. Rep., 2000. I

[4] O. Kupreev, J. Strohschneider, and A. Khalimonenko, "Kaspersky ddos intelligence report for q3 2016," *Securelist-Kaspersky Labs cyberthreat research and reports*, 2016. I

[5] T. Peng, C. Leckie, and K. Ramamohanarao, "Survey of network-based defense mechanisms countering the dos and ddos problems," *ACM Computing Surveys (CSUR)*, vol. 39, no. 1, pp. 3–es, 2007. I, II-A

[6] A. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani, "Toward developing a systematic approach to generate benchmark datasets for intrusion detection," *computers & security*, vol. 31, no. 3, pp. 357–374, 2012. I

[7] S. Specht and R. Lee, "Taxonomies of distributed denial of service networks, attacks, tools and countermeasures," *CEL2003-03, Princeton University, Princeton, NJ, USA*, 2003. II-A

[8] J. Mirkovic and P. Reiher, "A taxonomy of ddos attack and ddos defense mechanisms," *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 2, pp. 39–53, 2004. II-A

[9] J. Mirkovic, G. Prier, and P. Reiher, "Attacking ddos at the source," in *10th IEEE International Conference on Network Protocols, 2002. Proceedings.* IEEE, 2002, pp. 312–321. II-A

[10] M. H. Bhuyan, D. Bhattacharyya, and J. K. Kalita, "An empirical evaluation of information metrics for low-rate and high-rate ddos attack detection," *Pattern Recognition Letters*, vol. 51, pp. 1–7, 2015. II-A

[11] C.-L. Chen, "A new detection method for distributed denial-of-service attack traffic based on statistical test." *J. Univers. Comput. Sci.*, vol. 15, no. 2, pp. 488–504, 2009. II-A

[12] X. Xu, Y. Sun, and Z. Huang, "Defending ddos attacks using hidden markov models and cooperative reinforcement learning," in *Intelligence and Security Informatics: Pacific Asia Workshop, PAISI 2007, Chengdu, China, April 11-12, 2007. Proceedings*. Springer, 2007, pp. 196–207. II-B

[13] J. L. Berral, N. Poggi, J. Alonso, R. Gavalda, J. Torres, and M. Parashar, "Adaptive distributed mechanism against flooding network attacks based on machine learning," in *Proceedings of the 1st ACM workshop on Workshop on AISec*, 2008, pp. 43–50. II-B

[14] T. Shon, Y. Kim, C. Lee, and J. Moon, "A machine learning framework for network anomaly detection using svm and ga," in *Proceedings from the sixth annual IEEE SMC information assurance workshop*. IEEE, 2005, pp. 176–183. II-B

[15] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional lstm and other neural network architectures," *Neural networks*, vol. 18, no. 5-6, pp. 602–610, 2005. III

[16] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997. III-B