A Packet Classification Algorithm based on Improved Decision Tree

Shanliang Zhang and Lihua Zhu
Anyang Institute of Technology, Anyang, Henan, 455000, China
Email: daxuejsj@163.com, zhulihuagg@sohu.com

Abstract—Since the packet classification algorithm has its great importance and extensive applications, lots of scholars have done tons of researches on the algorithm. This paper presents a new method of a packet classification algorithm based on improved decision tree. In the proposed algorithm, multi-point segmentation and redundancy deletion are adopted to compress the depth of decision tree. A new data structure and the interval binary search are introduced to optimize the time performance. Simulation results show that, the average decision tree depth of MP^2S is approximately 60% of HyperSplit, and the memory access of MP^2S is 10% less than HyperSplit.

Index Terms—Packet Classification; The Decision Tree; Multi-Point Segmentation; MP^2S

I. INTRODUCTION

The packet classification plays an indispensable role in many Internet applications, such as the network safety, the quality of service, the load balance, the network measurement, the traffic shaping and so on. The packet classification assign the incoming data package to the different data streams based on the designed classification rules. The rule-set is a series of regulation aggregation, which the each rule contains matching condition, data streams identification and relevant processing method corresponding to the certain domains in the packet header. If the related domains satisfy all of the matching conditions in certain rule, that is to say the rule is matched by the data packet. The main function of router is to forward the IP datagram from one network to the other network. The traditional router only forward the data packets based on the destination address of the data packet, and provide the Best Effort Service that fail to distinguish [1]. A typical form of one-dimensional packet classification is to provide the same treatment for all users’ data packet [2]. However, with the fast development and the scale getting enlarged continuously of Internet, the rapidly efficiency classification is required in more and more applications, in order to distinguish and process the serve of the different levels. So the router is required for further processing the data packet. The commonest form of treatment is the packet filtering for the security needs of the system, which block the data packet with a huge security issues. Therefore, it is very important to research the high speed packet classification.

Since the packet classification algorithm has its great importance and extensive applications, lots of scholars have done tons of researches, which many excellent algorithms are proposed. The packet classification is essentially addressing problems in the multi-domain space [3] [4]. Some existing typical algorithms published are divided into three categories. The first type is based on decomposition along different dimensions. The attractive advantage goes the fast speed, and yet the updating algorithm will waste relating huge volume of storage. The second type is based on the segmentation of the geometry region, which the algorithm has a large memory overhead when frequent copy happen. The final category is based on tuple spaces, which have a small space efficiency and good average performance, but the uses of Hash function lead to the unstable performance. At the moment, the continuous improvement of the packet classification algorithm is done by many researchers. The typical algorithm is HyperSplit, which segmentation scheme is presented based on the regular projective endpoint, in order to reduce the amount of the copying rule [5] [6]. However, the duplicating rule of the Hpdewer still exists and the depth of decision tree is clearly not ideal. Based on the However, the binary search tree has been improved. So the paper proposes the MP^2S that replace the binary search algorithm with multipoint search. This will compress the depth of the Hpdewer decision tree, reduce the number of the traversing tree node in search, and improve the speed of searching. At the same time, redundancy deletion is adopted to decrease the number of nodes for reducing the memory occupancy.

II. THE RELATED WORKS

The packet classification algorithm is the basis of QoS. Only after the different packets are distinguished, the QoS of the processing and relevant business security may be conducted. The classification is the definitions of the transmission category, and indicates the category to which each packet belongs. There are two basic classification patterns in TCP/IP packet: behavior aggregate (BA) and multi-field (MF). BA is the foundations of the discriminating serve code points (DSCP), which it has good expansibility and apply to the core network [7]. MF is based on one or more domains (field) TCP/IP header. In principle, the entire field can be used to categorize data, for example, classical
five-element model (the source port, the destination port, source address, destination address, the protocol type). MF can be achieved on the network border, and is a widely used and flexible packet classification method. High performance routers should can provide flexible categorization strategy and implement an efficient classification algorithm [10, 11, 12]. According to the information of the data packet’s head in the networks, packet classification usually is categorized based on a set of rules. The transmission data is encapsulated by the packet header of all layer’s protocol in sequence, based on the OSI network model. The packet header of each layer contains several domains (field), which stands for the feature data in protocol.

The segmentation [13, 14, 15, 16] of the geometry region is a essentially point addressing problems in space. Each dimension in classification rules can be noted as a interval on the number axis. So the classification rules of the d-dimension can be considered as hypercubes. The packet headers can be rendered as a point in d-dimension Euclidean space, so the packet classification is to find highest priority hypercubes containing the point. The rule-set can be equally distributed into hypercubes by the region segmentation algorithm, based on the division of each dimension, and then the decision tree of the rule-set is constructed. The typical algorithms based on the region segmentation, are HiCuts and HyperSplit.

The basic idea of HiCuts is isometric grouping for packet step by step in order to create the packet classification decision tree, which the packet domain can be arranged into the hierarchy. A traversal can be done for matching a leaf node storing a few rules in the decision tree when the packets get to their destination. Then use the linear search algorithm to find the best matches [17, 18, 19]. However, the isometric grouping cannot flexibly select the optimal cut-off point based on the distribution of rule in the processing of creating decision tree, which cause a lot of duplication of the rules. Besides, the information redundancy in the decomposition process has yet to be fully addressed by the decision tree structure of the HiCuts, resulting in an unnecessary waste of storage space [20, 21].

The design goal of HyperSplit is to significantly reduce the memory usage of previous algorithms with efficient classifying ability steady in interval search algorithm [22-26]. Then, take rule-set with eight rules as an example to explain the build process in HyperSplit decision tree. The packet header domain d1, d2 ranges from [0, 7], the rule-set R is closely related to d1, d2, as shown in Table 1.

<table>
<thead>
<tr>
<th>Rules</th>
<th>d1</th>
<th>d2</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>[0, 2]</td>
<td>[0, 3]</td>
</tr>
<tr>
<td>R2</td>
<td>[0, 2]</td>
<td>[4, 7]</td>
</tr>
<tr>
<td>R3</td>
<td>[3, 3]</td>
<td>[0, 2]</td>
</tr>
<tr>
<td>R4</td>
<td>[0, 3]</td>
<td>[4, 6]</td>
</tr>
<tr>
<td>R5</td>
<td>[3, 3]</td>
<td>[7, 7]</td>
</tr>
<tr>
<td>R6</td>
<td>[4, 7]</td>
<td>[0, 2]</td>
</tr>
<tr>
<td>R7</td>
<td>[6, 7]</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 1. The rule-set R

Figure 1 is the rule spatial distribution map, where the packet-header domain d1, d2 correspond to X, Y axis, respectively, and there is a one-to-one correspondence between the rectangular box and the rules space.

Figure 2 is a packet classification decision tree generated by HyperSplit. Because the HyperSplit do not give the corresponding optimization problems in the copying rule, we can see from decision tree that the rule has been replicated.

When comparing these two algorithms, we can easily find the HyperSplit decision tree is a binary tree, while the bifurcate node can be selected by heuristic techniques in HiCuts. That is to say we can create a ‘fat’ decision tree to reduce the depth for diminishing the frequencies of reading data in memory. In improved HyperSplit algorithmic implementations, Qi et al. proposed a Node-Merging for reducing the amount of the intermediate node, which merge non-leaf node and its child nodes into a single node, store the two compartments of three nodes and dual points information into a node. Comparing the coming packet information and the corresponding node information for searching, get the matching results. As far as form is concerned, the method reduces the depth of a decision tree. However, at bottom, it has been not offered for the optimization for the depth of a decision tree, so it had done nothing to reduce traversal information. Further, the selected dimension is heuristically got in fragmentation each time.

So the algorithm cannot trim the search tree like EGT-PC, based on the Jumping Pointer for reducing the memory consumption.

For all this, this paper presents a improvement packet classification algorithm MP’S for globally solving the
problem of the poor depth and rule copying in a HyperSplit decision tree. Multiple segmentation is brought together to finish in MP$^3$S, the rule-set is divided into more subsets in the basis of hyperplane, so a very shallow depth decision tree is created to reduce total number of memory access. Consequently, MP$^3$S decrease the rule-copying to some extent and improves the efficiency of the usage of Memory by redundancy deletion.

III. PROPOSED SCHEME

A. Selection of the Cut-off Point

The algorithm is different from HiCuts and HyperCuts, the cut-off point is not selected in several point with a uniform distribution, but precisely choose the projection point lie in the regular edge as the cut-off point. The five-tuple model space is divided repeatedly by hyperplane. Based on results of a previous run, the algorithm chooses the optimal result as the cut-off point over and over again.

Firstly, the definition is given out, as follow:

Definition 1 rule projection point: let assume the rule-set contain n rules, projection point of the n rules consist of the projection point set in the \( l(1 \leq l \leq d) \) dimension, which there is m \( (1 \leq m \leq 2n) \).

Definition 2 projection point array: The m projection points are put into a queue according to the increasing value one by one, then are stored in an array \( Pt[j] \), where \( 0 < j < m + 1 \).

Definition 3 projection interval array: the adjacent endpoints form a interval. Namely, \( m \) points constitute \( m-1 \) intervals. We can use \( Sg[k] \) to denote, where \( 0 < k < m \), so the rule number located in some interval is denoted \( nSR[k] \).

In the method of selection cut-off dimension, the MP$^3$S is to take heuristic method in the HyperSplit. The rule number in the disjointed interval is called as interval, and then chooses the dimension of the minimum average weight for division. Let assume the cut-off dimension is \( L \), so the selection method of the cut-off dimension is as follow:

\[
L = \min_{l=1}^{l=d} \frac{1}{m-1} \sum_{k=0}^{m} nSR(k) \tag{1}
\]

In the selection method of the MP$^3$S cut-off point, the basic idea is that the rule number in the disjointed interval is used as a reference value, this will make the total rules of each sub-tree be equal. Let \( s_x(1 < s_x < m) \) and \( n(1 = 1,2,3 \ldots) \) be the location of the projection point and the location of the cut-off point, respectively. The basic ideas is that the Equation (2), (3) indicate.

\[
P_x = Pt[s_1], P_{s_1} = Pt[s_2], \ldots, P_{s_n} = Pt[s_n] \tag{2}
\]

\[
\sum_{s_x} nSR[s_x] = \sum_{s_x} nSR[s_1] = \cdots = \sum_{s_x} nSR[s_n] \tag{3}
\]

Based on the analysis, the MP$^3$S algorithm adopts a greedy heuristic to select a cut-off point. Namely, firstly select the cut-off point as much as possible and compute the rule number in the interval based on the number of the cut-off point. Then the cut-off points are dynamically selected to create cut-off interval, according to the increasing value one by one. After cut-off. Finally, remove the rest of interval after cut-off. The algorithm is introduced as follows:

Let assume the projection point array of the rule-set \( R \) is \( Pt[j] \), the array is \( nSR[k] \) created by rule number of the projection interval, the number of the rule projection points and default maximum cut-off point are \( m \) and \( mSplit \), respectively. So the selected the number of the cut-off point is

\[
N' = \min(m-2, mSplit) \tag{4}
\]

According to Equation (2), (3), (4), after cut-off the rule number \( splitSeg \) (it is rounded down) contained in each interval is as follow,

\[
splitSeg = \frac{1}{N' + 1} \sum_{k=1}^{m} nSR[k] \tag{5}
\]

Let assume the interval consisted of the \( n-th \) cut-off point \( P_{s_n} \) and the \( (n-1)-th \) cut-off point \( P_{s_{n-1}} \), has \( RS[s_n] \) rules.

\[
RS[s_n] = \begin{cases} 
\sum_{k=s_n+1}^{s-1} nSR[k], n > 1 \\
\sum_{k=1}^{s_n} nSR[k], n = 1 
\end{cases} \tag{6}
\]

According to the greedy cut-off point selection, when select \( n-th \) cut-off point, the rule number in \( RS[x] \) increase with increasing the value of the dynamic cut-off point \( P_s = Pt[x] \), where \( (x = s_{n-1} < s_n < s_{n-1} + 1 < s_{n-1} + 2 \ldots) \). If \( P_s \) is equal to a certain number, the \( RS[x] \) exactly meet the following Equation (7).

\[
RS[x] = splitSeg \tag{7}
\]

So the \( n-th \) cut-off point is \( P_{s_n} = Pt[s_n](s_n = x) \)

Because the rule number isn’t successive in projection interval, \( S[x] \) is not exactly \( splitSeg \). Only in the case of gradually increasing \( P_s \), the first point \( Pt[x] \) make the Equation (8) be hold.

\[
RS[x] \geq splitSeg \tag{8}
\]

So the \( n-th \) cut-off point lies in \( Pt[x] \). According to the situation, \( RS[s_n] \geq splitSeg \), the latter cut-off interval cannot meet the need of Equation (8). Thus, after the first \( n \) cut-off point is determined, the creation processing of the cut-off point is end if the sum of the rest projection interval array meets the following Equation (9).

\[
\sum_{s_x} nSR[s_x] = \sum_{s_x} nSR[s_1] = \cdots = \sum_{s_x} nSR[s_n] \]
\[ \sum_{k=1}^{nSR} nSR[k] \leq \text{splitSeg} \quad (9) \]

Then remove the rest of \( N' - n \) cut-off point to get finally the number of cut-off point is \( N = n \).

Taking the rule-set in Table 1 as an example, we will brief the selection processing of the cut-off point. According the Equation (1), (4) in the first cut-off, 3 cut-off point are selected in \( d_1 \) domain. When the total rule number is 10 in the disjointed interval, we can get \( \text{splitSeg} = 3 \) based on Equation (5). According to the greedy cut-off point selection, the first cut-off point lies in \( P_1 \). If this point meet Equation (8), so \( P_2 = P[1] \).

The second cut-off point lies in \( P[2] \). If also meet Equation (8), so \( \text{splitSeg} = P[2] \). At the same time, if the sum of the rest projection interval array meets the following Equation (9), remove the last cut-off point. So far, the candidate values of all the cut-off point have been selected. The current round of cut-off has been finished, as shown in Figure 3.

![Figure 3. The constructed example of the MP'S decision tree](image)

In this process, we choose 2 cut-off point, the rule-set is divided into three parts: \( \{ R_1, R_2, R_4 \}, \{ R_3, R_4, R_5 \} \) and \( \{ R_6, R_7, R_8 \} \), which form three subspaces separately. The decision tree was achieved by the iteration procedure mentioned above in the three subspaces.

B. The Covering Deletion

With the increasing of the rule-set, the probability of the alternative covering greatly increased between rules. HyperSplit only can delete the fully overlapping rules based on the priority, and do not take into account the inclusion relation between rules. So the number of the leaf nodes is extremely inflating, and occupies so much space with the increasing of the rule-set.

As shown in Figure 4, there are two rules \( R_1, R_2 \) in subinterval, \( R_1 \) takes precedence over \( R_2 \), and \( R_2 \) is fully included in \( R_1 \). Using the selection method of cut-off point in HyperSplit. The projection starting point of \( R_2 \) will be selected onto the \( X \) axes as the cut-off point, so the interval is divided into the two parts. Due to \( R_2 \) has lower precedence than \( R_1 \), the rule \( R_2 \) ever is completely covered by the rule \( R_1 \) in subinterval. Actually, the divided child nodes will appear to be two \( R_1 \) rules, which is a great waste of memory.

![Figure 4. The duplication of the rules in HyperSplit](image)

In order to solve the waste problems of memory storage caused by the redundant rules in HyperSplit, the covering deletion is introducing into the divided subinterval in MP'S. According to the priority and the inclusion relations each other, the low-priority covered by the high-priority will be deleted. The cut-off point selection will keep away from the extra projection point brought by redundant rule. So the structure of decision tree can be optimized, and the algorithm avoid the duplication of the rules massively appears.

Let assume the rule-set is \( R' \) in interval, the ordinal number of the priority level in the rules \( r_i, r_j \in R \) are denoted by \( r_i, pri, r_j, pri \). If \( r_i, r_j \) can be met by the following equation:

\[
\begin{align*}
& r_i \subseteq r_j \\
& r_i, pri \geq r_j, pri
\end{align*}
\]

So the low-priority rule \( r_j \) will be deleted from subinterval, only the high-priority rule \( r_i \) are retained.

The realization process of the covering deletion is shown in Figure 5. If \( R_2 \) has lower precedence than \( R_1 \), and the rule \( R_2 \) ever is completely covered by the rule \( R_1 \) in subinterval, MP'S will delete the rule \( R_2 \). So the \( R_1 \) directly compose a child node when create a decision tree, avoid appearing the redundant node.

![Figure 5. The covering deletion](image)

C. The Lookup Algorithm and Data Structure

The number of the cut-off point and the distance between the two cut-off points are regular in HiCuts. It can realize the location quickly for corresponding cut-off point in packet based on the pre-defined index created by cut-off point. The cut-off method in MP'S is different from the equally proportional cut-off in HiCuts. MP'S adopt an equidensity cut-off idea, so these cut-off point do not form arithmetic progression. That is to say the lookup method based on cut-off index does not apply to MP'S. So MP'S adopt the interval binary search for reducing total number of memory access. The lookup algorithm may be introduced as follows [27-29].
Let assume the current lookup interval is \( ns[i] (m \leq i \leq n) \), where \( m, n \) are the starting point and terminal point, respectively. The value stored between the packet-header and intermediate node is \( p[l] (1 \leq l \leq d) \), and the cut-off interval index corresponding the packet is \( \text{result} \). The algorithm flow is given as follows:

1. If \( m - n = 1 \), do steps 3; if \( m = n \), do steps 4.
2. \( m = (m + n) / 2 \), if \( p[l] \geq ns[\frac{m + n}{2}] \) otherwise \( n = (m + n) / 2 \), then do steps 1.
3. if \( p[l] \geq ns[n] \), result = n + 1; if \( p[l] < ns[m] \), result = m, else result = n, then do steps 5.
4. If \( p[l] \geq ns[n] \), result = n + 1, else result = n. Then do steps 5.
5. Feedback result when it completes duty.

Taking the intermediate node having five cut-off points as an example, the domains is divided into six continuous intervals by the five cut-off points, which corresponding respectively to the nodes in the next layer of the decision tree. Assume the value of the packet in query domain is 3.

In the light of specific examples, by analyzing the vested data structure, perform the following steps.

Set the current node as V

If the address information of the node V is equal to zero, do steps 5.

Get the value corresponding multi-domain in the current node and use the binary search algorithm to compare with N cut-off point in order to determine the cut-off point interval n.

Assume the child node of V is \( V(n) \), its memory address is denoted as” the current node address" + "the offset address corresponding the first n cut-off interval”, then do steps 2.

Get the matching decision of the child node, and then carry out corresponding orders.

D. An Example on the Generating MP\( ^3 \)S

Take the rule-set having 8 rules as an example to introduce the creation process of MP\( ^3 \)S decision tree, as shown in following figure.

The rule-set is divided into three rule-subset \{R1, R2, R4\}, \{R3, R4, R5\}, \{R6, R7, R8\} in the first cut-off, at the same time remove the redundant rules R4 in the subset. Then the longitudinal axis point is selected as cut-off point in \{R1, R2\} and hyperspace corresponding R1, R2 is cut. Perform the same steps for the other two subsets till the rule number in subset is less than or equal to a prescribed value (here is initially set to 1). So far, the decision tree is accomplished and created, as shown in Figure 8.9.

In the light of specific examples, by analyzing the characteristics of the decision tree, we can see the depth of the MP\( ^3 \)S decision tree be optimized compared with HyperSplit, which vary from 3.125 to 1.5. The same rule R2 copied is not also observed on MP\( ^3 \)S.

The data structure of MP\( ^3 \)S is shown in following figure. Each node contains N+1 address space. The first 32bit of the first address space have the information of 4bit cut-off point, 4bit cut-off domain information, while the last 32bit is offset address information of the child node corresponding cut-off interval. In the rest of N address space, the first 32bit save the offset address of seg(i+1) corresponding child node, while the last 32bit store the first i (0 < i < N+1) cut-off point information. The leaf node occupy 1 address space, where the first 32 bit is zero and the last 32bit is matching decision.

According to the vested data structure, perform the following steps.

TABLE II. STORAGE STRUCTURE IN INTERMEDIATE NODE

<table>
<thead>
<tr>
<th>Field</th>
<th>size(bit)</th>
</tr>
</thead>
<tbody>
<tr>
<td>content</td>
<td>space(bit)</td>
</tr>
<tr>
<td>Cut-off point number</td>
<td>4</td>
</tr>
<tr>
<td>Multi-domain information</td>
<td>4</td>
</tr>
<tr>
<td>Offset address information</td>
<td>(N+1)*32</td>
</tr>
<tr>
<td>Cut-off point information</td>
<td>N*32</td>
</tr>
</tbody>
</table>

To the leaf node same with HyperSplit, only require to store the corresponding matching decision, which need 32bit, as shown in Table 3.

TABLE III. STORAGE STRUCTURE IN THE LEAF NODE

<table>
<thead>
<tr>
<th>Field</th>
<th>size(bit)</th>
</tr>
</thead>
<tbody>
<tr>
<td>content</td>
<td>space(bit)</td>
</tr>
<tr>
<td>matching decision</td>
<td>32</td>
</tr>
</tbody>
</table>
The selection of the MP$^2$S cut-off point must follow projection endpoint selection principle. Comparing with HyperSplit, MP$^2$S don't introduce the extra rule replication. Moreover, MP$^2$S adopt an equidensity cut-off idea, the decision tree do not deflect when reducing the depth of the MP$^2$S decision tree. This newly introduced covering deletion method not only optimize the part of redundancy rules, but also decrease the number of leaf node in MP$^2$S, which could greatly reduce the algorithm complexity.

![Figure 8. The regular plan in MP$^2$S](image)

![Figure 9. The decision tree in MP$^2$S](image)

IV. SIMULATION RESULTS

In many real networks, the rule-set usually exists in a router or a firewall. Due to the safety, etc factors, it is difficult to acquire the rule-set. Moreover, only a few rules are contained in real rule-set. Generally, there are only several hundred rules. The entire aspect simulation will be very hard for the packet classification performance. Through analysis on the rule-set in real networks, Taylor in Washington University in St. Louis develop an open-source ClassBench, in order to generate effects closer to real rule-set for industrial and academic research and the performance simulation. The data-set produced by ClassBench is widely used by the packet classification research, which mainly contain ACL (Access Control List), FW(Fire Wall)and IPC (Linux IP Chains). For example, FW1-1K has1000 firewall security policy rules. All the rules are constituted by IPv4 5-tuple. They include 32-bit source address, 32-bit destination address, 16-bit source port number, 16-bit destination port number and 8-bit transport layer protocol. In all algorithms of this paper, all data are from ClassBench.

The simulation is programmed by c language, compiled by gcc-4.4.3 in Ubuntu 10.04 LTS.

The average depth is compared in HyperSplit and MP$^2$S decision trees, as shown in Figure 10, which the average depth of the child node is marked as the appraisal index. It can be seen that the average depth of MP$^2$S is about 28% of HyperSplit.

![Figure 10. The comparison of average depth in decision trees](image)

The worst depth of decision trees is compared in HyperSplit and MP$^2$S, as shown in Figure 10. It can be seen that the worst depth of MP$^2$S is about 90%-95% of HyperSplit. The worst decision trees depth is significantly decreased in ACL rule-set, while IPC is roughly same.

![Figure 11. The worst depth of decision trees](image)

The memory occupancy is compared in HiCuts, HyperSplit and MP$^2$S, where the logarithmic coordinates is applied to describe in Y axes. As shown in Figure 10, it can be seen that memory occupancy of the latter two algorithms is reduction of 1~2 order of magnitude. Due to the muti-cutoff points adopted to segment the multi-domain space, the number of the node declined rapidly. However, the intermediate node will store the address information of many cut-off intervals, which this cause the memory occupancy of the intermediate node is increased to some extent. So the memory occupancy is roughly same in MP$^2$S and HyperSplit. The optimization performance of MP$^2$S is better in above IPC1-5K rule-set. The memory occupancy is reduced by about 20% and 30%, respectively (IPC1-5K, IPC1-10K).
The average memory access is compared in HiCuts, HyperSplit, and MP2S. As shown in Figure 10, it can be seen that memory access of the latter two algorithms have a substantial decrease. The average memory access of MP2S is reduced by about 20% compared with HyperSplit. The memory access is reduced by about 30% in IPC rule-set.

V. CONCLUSION

Based on the HyperSplit algorithm, this paper proposes an improved algorithm MP2S. The multi-cutoff points adopted to segment the multi-domain space, and the covering deletion adopted to reduce the number of the redundancy rules. Moreover, the interval binary search and new data structure are introduced to decrease the average memory access. The simulation result indicated that the algorithm improve HyperSplit decision tree structure, reduce the average depth of the decision tree, optimize average algorithm of time performance.

REFERENCES


**Shanliang Zhang** (1979. 7) Female, Anyang, Master of Engineering, lecturer, research direction: Computer Applications, Intelligent Information Processing, Distributed computing, Computer System Architecture

**Lihua Zhu** (1981. 2) Female, Anyang, Master of Engineering, lecturer, research direction: Computer Applications, Computer Networks and Information Processing, Computer Communications