doi:10.3969/j.issn.1000-436x.2014.z1.012

基于统计的高效决策树分组分类算法

陈立南[1,2]，刘阳[3]，马严[1,4]，黄小红[1]，赵庆聪[2]，魏伟[1]

（1. 北京邮电大学 信息网络中心，北京 100876；2. 北京信息科技大学 信息管理学院，北京 100192；
3. 国家计算机网络应急技术处理协调中心，北京 100029；4. 移动互联网安全技术国家工程实验室，北京 100876）

摘 要：基于决策树的分组分类算法因易于实现和高效性，在快速分组分类中广泛使用。决策树算法的基本目标是构造一棵存储高效且查找时间复杂度低的决策树。设计了一种基于规则集统计特性和评价指标的决策树算法——HyperEC 算法。HyperEC 算法避免了在构建决策树过程中决策树高度过高和存储空间膨胀的问题。HyperEC 算法对 IP 地址长度不敏感，同样适用于 IPv6 的多维分组分类。实验证明，HyperEC 算法当规则数量较少时，与 HyperCuts 基本相同，但随着规则数量的增加，该算法在决策树高度、存储空间占用和查找性能方面都明显优于经典的决策树算法。

关键词：决策树；分组分类；统计；规则

中图分类号：TP393.03 **文献标识码**：A **文章编号**：1000-436X(2014)Z1-0058-07

Efficent-cutting packet classification algorithm based on the statistical decision tree

CHEN Li-nan[1,2], LIU Yang[3], MA Yan[1,4], HUANG Xiao-hong[1], ZHAO Qing-cong[2], WEI Wei[1]

(1. Network Information Center, Institute of Network Technology, Beijing University of Posts and Communications, Beijing 100876, China;
2. School of Information Management, Beijing Information Science & Technology University, Beijing 100192, China;
3. CNCERT, Beijing 100029, China; 4. National Engineering Laboratory for Mobile Network Security, Beijing 100876, China)

Abstract: Packet classification algorithms based on decision tree are easy to implement and widely employed in high-speed packet classification. The primary objective of constructing a decision tree is minimal storage and searching time complexity. An improved decision-tree algorithm is proposed based on statistics and evaluation on filter sets. HyperEC algorithm is a multiple dimensional packet classification algorithm. The proposed algorithm allows the tradeoff between storage and throughput during constructing decision tree. For it is not sensitive to IP address length, it is suitable for IPv6 packet classification as well as IPv4. The algorithm applies a natural and performance-guided decision-making process. The storage budget is preseted and then the best throughput is achieved. The results show that the HyperEC algorithm outperforms the HiCuts and HyperCuts algorithm, improving the storage and throughput performance and scalable to large filter sets.

Key words: decision tree; packet classification; statistic; filters

1 引言

当前，互联网的飞速发展导致网络带宽和流量日益增加，使互联网的安全问题已经成为事关国家稳定、社会安定和经济繁荣的全局性问题。分组分类是实现网络安全、网络虚拟化和服务质量的基础

收稿日期：2014-10-15

基金项目：国家高技术研究发展计划（“863”计划）基金资助项目（2013AA014702）；北京市教委科研计划：基于数据流回放和标注的数据集生成技术研究基金资助项目（KM201311232017）；中央高校基本科研业务费基金资助项目（2014PTB-00-04, 2014ZD03-03）；中国下一代互联网基金资助项目（CNGI-12-02-027）；DNSLAB 基金资助项目

Foundation Items: The National High Technology Research and Development Program of China(863 Program)(2013AA014702); Beijing Municipal Commission of Education Research Project: Research on Dataset Generation Technique Based on Data Flow Replaying and Marking (KM201311232017); Fundamental Research Funds for the Central Universities (2014PTB-00-04, 2014ZD03-03); China Next Generation Internet Project (CNGI-12-02-027); DNSLAB

技术。虽然国际上已经对多维分组分类算法进行了大量研究，出现了一些优秀的算法，但是分组分类技术随着互联网流量的快速发展出现了新的问题与挑战，不断增长的过滤规则数对网络设备的数据分组快速分类的提出了更高的要求。

尽管目前高速分组分类方案中大量使用 TCAM 和 FPGA，但是由于硬件成本高、功耗大等缺点，业界更倾向于使用软件进行高效的数据分组分类。

软件实现的分组分类算法具有的灵活性对算法的实现起着重要的作用。决策树算法在空间复杂度上具有很大的灵活性。算法通过对规则空间递归地划分建立一棵决策树。当某子集包含的规则数目少于预定义的用户参数时，停止对该规则子集进行划分。很多研究表明构建具有最小查找速度、最少内存占用的理想决策树分组分类是 NP 难问题[1]。不管使用什么样的评价方法，建立一棵理想的决策树是不可能的[2]。因此，目前所有的决策树算法都是集中于基于本地优化的启发式算法。基于决策树的分组分类算法在软件或硬件上都易于实现，效率高，适用于对速度要求较高的实时分组分类的系统。

2　基于决策树的分组分类算法存在的问题

基于规则空间划分的决策树算法在递归地对规则集空间进行划分的过程中构建决策树。每个节点对应的规则子集空间 S 都将划分成多维等分的空间子集 $s_1, s_2, \cdots, s_n$。每个子集将生成一个当前节点的孩子节点。可知，$S = s_1 \cup s_2 \cup \cdots \cup s_n$。定义 $binth$ 为预定义的决策树叶子节点中存储的最大规则数。如果对于 $\forall i$，$|s_i| > binth$，则将对该子集继续进行规则子集空间的划分。对任意的 i 和 j，$s_i \cap s_j \neq \varnothing$，$|s_i| \neq |s_j|$，$\sum_{j=1}^{n} |s_j| \geqslant |S|$。

决策树的分支节点存储覆盖其孩子及子孙的规则子集范围，而不存储规则本身。决策树中只有叶子节点存储对应的规则子集。

Woo[3]、HiCuts[4]和 HyperCuts[5]是经典的决策树算法。Woo 算法每次只选择一位对规则集进行划分的特点，将导致决策树的高度过大。HiCuts 算法在建立决策树时，每次只能针对一个维度进行规则集划分，对于高维规则集，决策树的高度将会有很大的增长。

HyperCuts 算法针对 HiCuts 算法每次只能针对一个维度划分规则集进行了改进，增加了预定义参数 $binth$，对决策树叶子节点存储的最大规则数进行了限制。HyperCuts 构建决策树时可以同时在多个维度上对规则集进行划分。但是对 HyperCuts 研究中发现，该算法有时会出现孩子节点中规则数量相差悬殊而导致决策树左右孩子高度不平衡和决策树高度增加的现象，这使得对决策树进行查找时出现访问内存次数过多、算法性能下降的现象。

3　HyperEC 算法思想

HyperEC 算法在结合经典决策树类算法优点的基础上，通过启发方法和性能评价函数获取高效时空复杂度的算法。

3.1　规则集

HyperEC 是基于空间划分决策树的多维分组分类算法。与其他决策树一样，树中的每个节点都存储其对应的规则子集的范围区间，决策树的分支节点不存储规则集本身，仅在决策树的叶子节点存储规则子集。

规则集中规则的每个域都以范围形式表示，IP 地址、端口、协议等域都转换成区间范围。

3.2　规则集划分方式

决策树的根节点存储规则集对应的全部范围。通过如下 2 个步骤确定对节点规则集区间进行划分的次数。

步骤 1　选择待划分维度

维度的选择与 HyperCuts 相同，即分别在每个维度上统计当前节点对应的规则子集中唯一规则数 nu_i，并计算唯一规则数的平均值。选择所有唯一规则数大于均值的维度对规则子集进行划分。规则子集在选定的每个维度上至少划分 2 次。

步骤 2　确定对规则子集空间划分的方式和次数

为表达清楚，定义以下参数：

nu_i：第 i 维唯一规则的数量；

nc_i：在第 i 维上划分的次数；

NC：对当前规则子集划分的总次数，$NC = \prod_{i=1}^{n} nc_i$。

通过启发和迭代确定在每一被选维度上划分的具体次数。对每一选择的维度的具体操作如下：

1) 对该维度划分 $nc_i \times 2$ 次；

2) 计算评价函数 $EF(i)$；

3) 确定在 EF 值最小的第 i 维上进一步划分，即该维划分次数 $nc_i \times 2$。

当前节点 v 的孩子数和决策树的总存储空间满

足以下3个条件时将继续对该规则集进行划分。

条件1 基本划分条件

在构建决策树过程中，对当前节点进一步划分的基本条件是该节点对应的规则数目大于预定义的限制值*binth*。$|FS(u_i)|<binth$时，停止迭代。

条件2 划分次数条件

若当前的划分次数*NC*满足以下条件时，继续迭代。

$$NC \leqslant \min\left(NC_{lmt}, spfac\times\sqrt{N}\right) \tag{1}$$

其中，

NC_{lmt}：预定义的节点最大划分数量；

spfac：可调节的时空参数，该参数对规则集划分的次数进行了约束；

$N=|FS(v)|$：节点*v*对应的规则子集中规则数量。

条件3 决策树占用总空间的限制条件

决策树算法中普遍存在的另一个问题是构建决策树时只进行局部的评估和优化，而对整棵树所占用的存储空间无法控制。HyperEC算法对此进行了优化，在进行迭代划分的同时，统计本次划分将占用的存储空间，进而计算出整棵决策树占用的存储空间。

当前决策树占用的存储空间包括决策树节点占用的存储空间与叶子节点存储的规则子集所占用的存储空间之和，总存储空间小于预先定义的总存储空间参数时，继续迭代。反之，停止迭代。

3.3 评价函数

评价函数对规则集此次划分*c*从存储空间和决策子树的平衡性进行定量计算。

若本次划分将节点*v*对应的规则子集划分了*n*次，则节点*v*将生成*n*个孩子节点$u_1,u_2,\cdots,u_n$。若定义$FS(u_i)$表示节点u_i对应的规则子集，则$afs(v,c)=\sum_{i=1}^{n}|FS(u_i)|/n$表示在划分策略*c*中节点*v*所有孩子节点存储规则数量的平均值，$opfs(v)=|FS(v)|/NC_{lmt}$表示理想情况下节点*v*的孩子节点平均存储的规则数。

定义1 存储规则重复存储指标，该指标表示节点*v*的孩子在该次划分策略*c*中存储规则的重复度评价值。该值越小，规则的重复存储率越低。

$$E_{dup}(v,c)=\frac{\left(\sum_{i=1}^{n}|FS(u_i)|\right)-|FS(v)|}{|FS(v)|} \tag{2}$$

定义2 决策树平衡性评价，该值越低表示节点*v*的孩子节点对应的规则数越相近，即孩子节点存储的规则数越接近，建立的子树也越平衡。

$$E_{bal}(v,c)=\frac{||FS(u_i)|-afs(v,c)|}{|FS(v)|} \tag{3}$$

定义3 孩子节点规则数评价，是衡量规则集决策树占用存储空间的重要指标。该值越小，节点*v*的孩子节点占用的存储空间越小。

$$E_{afs}(v,c)=\frac{afs(v,c)}{opfs(v)} \tag{4}$$

定义4 评价函数*EF*

$$EF(v,c)=\alpha E_{dup}(v,c)+\beta E_{bal}(v,c)+\gamma E_{afs}(v,c) \tag{5}$$

使用参数α、β和γ($\alpha+\beta+\gamma=1$)确定对节点*v*对应的规则空间划分时规则重复存储率和平衡率的权重。α、β和γ设置不同的值将会影响存储规则占用的存储空间和决策树的节点总数及内存查找效率。

3.4 决策树建立算法

构建决策树的目标是在总内存不超过预定值的情况下，建立一棵高度最矮、孩子节点存储规则数目尽量平衡的决策树。

采用与HiCuts相似的建立决策树方法。本算法在对决策树节点划分时采用了高效的评价体系，该评价体系结合了HyperCuts算法的优点并对其进行了改进，使得通过HyperEC的启发式算法建立的决策树可以占用更小的存储空间、更高效的查找性能。

初始建树的过程与HyperCuts类似。初始状态，决策树只有一个根节点，根节点对应规则集的全部空间。

算法1 choose_dimension

输入：节点*v*

输出：待划分维集合Dims

if (|*v*| < bucket Size) return null;

for *i* from 1 to *k*

  N_i = number Of Unique Values On Dim(*v*, *i*);

Mean = mean($N_1,\cdots, N_k$);

for *i* from 1 to *k*

  if (N_i > Mean) Dims = Dims ∪ N_i;

return Dim;

算法2 CreateNode

输入：规则集

输出：决策树

CreateNode($l_1,r_1,l_2,r_2,\cdots,l_k,r_k,v$);

while (DT current storage ≤ the predefined storage)

Dims = choose_dimension(v);

for i ∈ Dims

nc(i) = 2;

$NC = 2^{|\text{Dims}|}$;

while (NC×2 < predefined value && total_rule_num < predefined storage)

{ $NC = NC$×2;

for each i∈ Dims

计算评价函数 EF 的值;

选择 pef 最小的维上做进一步划分;

计算存储的规则总数;

}

for i=1 to NC

CreateNode (l_{i1},r_{i1},…,l_{ik},r_{ik},v_i);

3.5　查找和更新

HyperEC 算法与 HyperCuts 算法的查找方法相同。

HyperEC 支持增量更新。当插入或删除一条规则时，仅该规则对应的决策树分支需要更新。由于决策树是在对规则集进行统计的基础上构建的，因此，更新可能会破坏原有决策树的平衡性。如果更新的规则与原有规则集有相同的统计特征，则少量规则的更新对原有决策树的影响不大。如果需要进行大量规则的更新，就需要重新构建整棵决策树。

与其他决策树算法相同，增量更新的时间复杂度取决于更新规则覆盖的空间范围。较大的更新规则空间将导致较差的更新性能。最坏情况下，更新一条规则可能会遍历整棵决策树。

4　性能测试与比较

为验证算法的性能，在 Linux 操作系统中开发了分组分类原型系统，在原型系统中实现了 HiCuts 算法、HyperCuts 算法和 HyperEC 算法。在普通 PC 机上运行该原型系统，该 PC 机配置为 Intel 2.53 GHz CPU，2 GB 内存。测试中使用的规则集都由 ClassBench[6]生成，规模从 1 000 条到 10 000 条。

首先对HyperEC算法与其他2种经典算法在决策树高度、内存空间占用和查找效率方面进行比较。

从图 1 可以看出，HyperEC 算法针对各种规则数目所建立的决策树高度都小于 HiCuts 算法和 HyperCuts 算法。随着规则数的增加，HyperEC 算法所构建的决策树高度远远小于其他 2 种算法。当规则数达到 10K 时，决策树高度相对 HyperCuts 算法降低了 63%，相对 HiCuts 算法降低了 90%。

从图 2 可以看出，当 *binth* 设置为 16 时，HyperEC 算法占用的存储空间相对其他 2 种算法有很大的减少。当 *binth* 设置为 32 时，3 种算法中 HyperEC 算法所占用的存储空间最小。

对每个数据分组分类时访问内存的平均字数是衡量算法时间复杂度的重要度量标准。从图 3 可以看出，HyperEC 算法与 2 种经典的决策树算法相比较，对数据分组进行分类时需要访问内存的字数有了很大的改善。当规则数是 1K 时对数据分组分类时访问内存的字数减少了 20%，当规则数是 10K 时访问内存的字数最多减少了 70%。

通过对图 1~图 3 的分析比较可知，HyperEC 算法对于较大的规则集，HyperEC 算法在决策树高度、内存占用和查找时内存比较的字节数都比其他 2 种

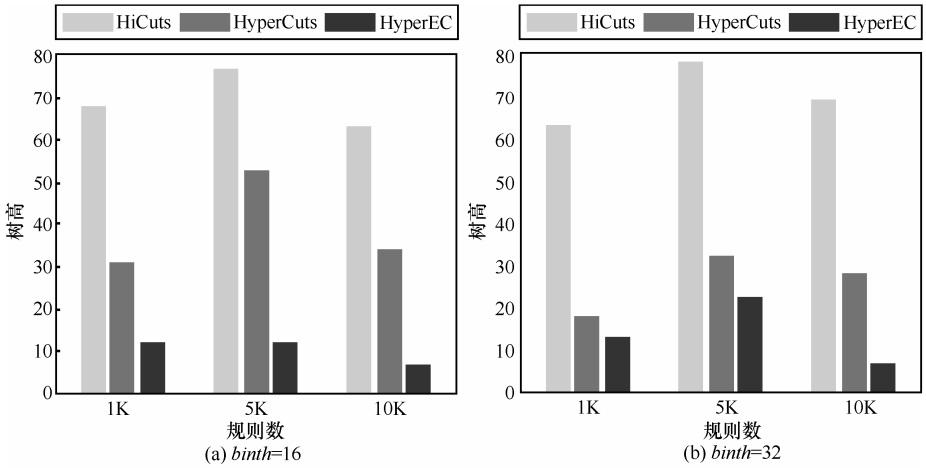


图 1　与其他算法在决策树高度上的比较

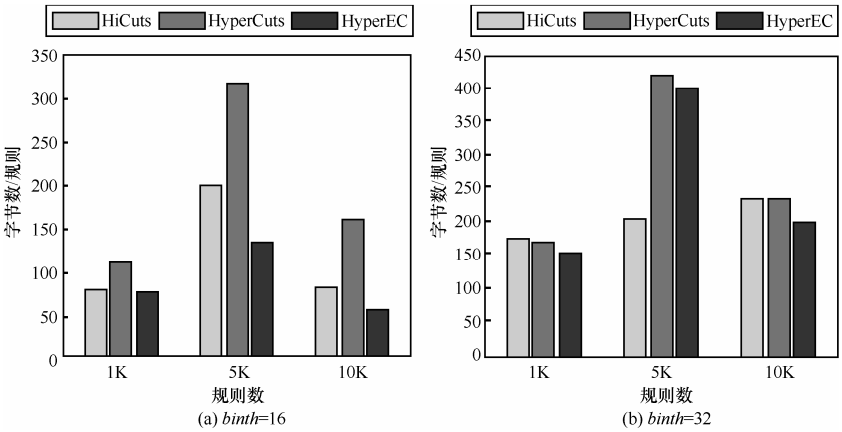


图 2　与其他算法在存储空间占用上的比较

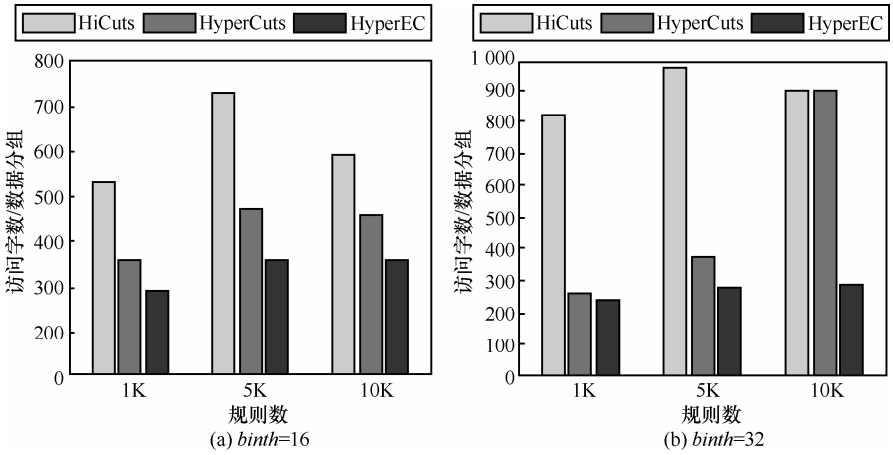


图 3　与其他算法在访问内存效率上的比较

算法有了很大的改进。

下面分析 HyperEC 算法中 *spfac* 参数及 α、β 和 γ 参数对算法各个度量指标的影响。

从图 4 可以看出，决策树的高度会随着 *spfac* 值的增大而减小。针对 10K 规则数，当 *spfac* =1 时树高为 24，当 *spfac* =5 时树高降低到 17。内存占用也会随着 *spfac* 值的增大而逐渐稳定，数据分组查找时访问内存的字数、最坏情况下的查找时间也明显下降。

对 HyperEC 算法的 α、β 和 γ 参数的设置对算法性能的进一步优化起到重要的作用。图 5 和图 6 是 HyperEC 算法针对 10K 的规则集、*binth*=64 与 *spfac*=1 时的测试结果。

图 5 中参数 γ 保持不变，通过改变 α 与 β 2 个参数的值对性能的影响进行比较。从图 5 可以看出，随着 α 值的增大，决策树中存储的规则总数和决策树规模增大，因此导致平均访问内存字数增加。当该值增大时，决策树中存储的冗余的规则数目增大，从而验证了算法中 α 参数的有效性。

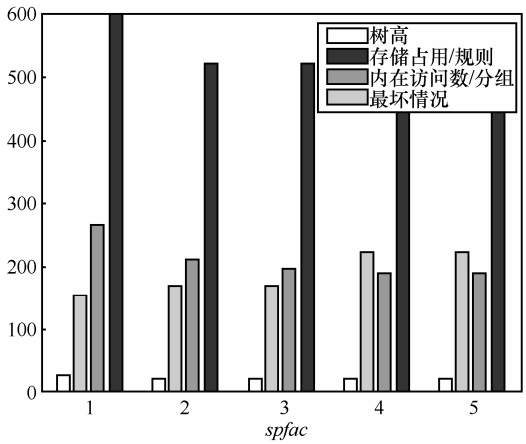
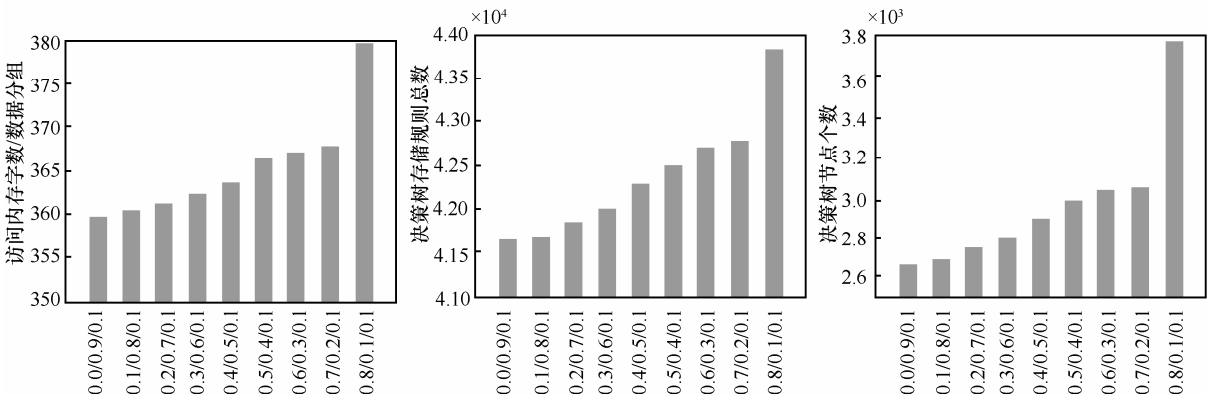


图 4　*spfac* 参数对 HyperEC 算法的影响

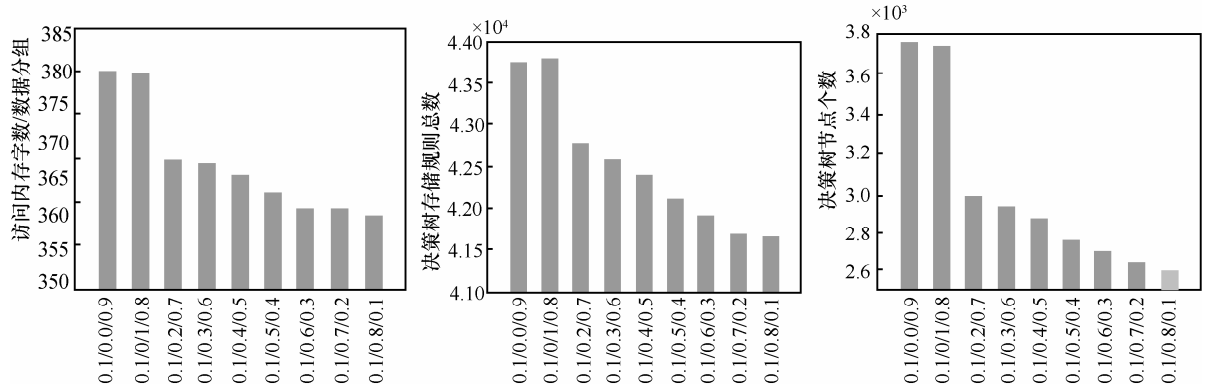
同时可以看出，随着 β 值的增大，决策树中存储的规则总数和决策树规模降低，因此导致平均访问内存字数下降。同时验证了算法中 β 参数对决策树平衡性进行控制的有效性。

图 6 中参数 α 设置为 0.1，通过改变 β 与 γ 2 个



(a) 对内存访问效率的影响　(b)对决策树存储规则总数的影响　(c)对决策树节点个数的影响

图 5　当 γ=0.1，改变 α 与 β 对算法的影响



(a) 对内存访问效率的影响　(b)对决策树存储规则总数的影响　(c)对决策树节点个数的影响

图 6　当 α=0.1，改变 β 与 γ 对算法的影响

参数的值对性能的影响比较。从图 6 可以看出，γ 值越小，决策树的节点数越小、决策树中存储的规则总数越小，因此占用的存储空间越小。随着 β 值的增大，决策树中存储的规则总数和决策树规模降低，因此导致平均访问内存字数下降。由于 β 参数增加时，建立决策树的平衡性增大，因此，平均访问内存的字数减小，从而验证了算法中 β 参数与 γ 参数的有效性。

5　结束语

本文提出了基于决策树的 HyperEC 多维分组分类算法。HyperEC 算法根据规则子集的统计特征选择待划分的维，并为划分得到的每一维通过评价指标和迭代方式计算每一维上划分的次数，建立一棵决策树高度与存储空间可控的决策树。HyperEC 算法对 IP 地址的长度不敏感，因此也可用于 IPv6 的分组分类。下一步将对 HyperEC 算法进行 IPv6 规则集的测试与改进工作。与得到广泛应用的 HiCuts 算法和 HyperCuts 算法相比，HyperEC 算法在内存占用、决策树高度和查找时间上都有很大提高。

参考文献：

[1] HYAFIL L, RIVEST R L. Constructing optimal binary decision trees is NP-complete[J]. Inf Process Lett, 1976,(5): 15-17.

[2] MURPHY O J, MCCRAW R L. Designing storage efficient decision trees[J]. IEEE Transaction on Computers, 1991, 40(3):315-320.

[3] WOO T Y C. A modular approach to packet classification: algorithms and results[A]. IEEE International Conference on Computer Communications[C]. 2000.1213-1222.

[4] GUPTA P, MCKEOWN N. Packet classification using hierarchical intelligent cuttings[A]. IEEE High-Performance Interconnects[C]. 1999. 34-41.

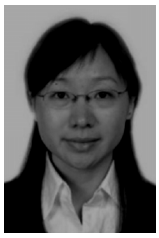
[5] SINGH S, BABOESCU F, VARGHESE G, *et al*. Packet classification using multidimensional cutting[A]. ACM Special Interest Group on Data Communication[C]. 2003.213-224.

[6] TAYLOR D E, TURNER J S. ClassBench: a packet classification benchmark[A]. IEEE International Conference on Computer Communications[C]. 2005. 2068-2079.

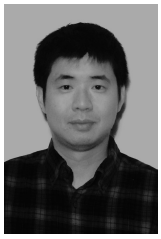
[7] SONG H Y, TURNER J S. ABC: adaptive binary cuttings for multidimensional packet classification[J]. IEEE Transactions on Networking, 2013, 21(1): 98-109.

作者简介：

陈立南（1972-），女，黑龙江勃利人，北京邮电大学博士生，北京信息科技大学副教授，主要研究方向为新一代互联网关键技术等。



黄小红（1978-），女，广东河源人，博士，北京邮电大学信息网络中心主任，主要研究方向为下一代互联网关键技术。



刘阳[通信作者]（1978-），男，辽宁营口人，国家计算机网络与应急技术处理协调中心工程师，主要研究方向为信息安全。E-mail: liuyang5599@163.com。

赵庆聪（1978-），女，山西祁县人，博士，北京信息科技大学副教授，主要研究方向为信息安全、信息管理。

马严（1955-），男，北京人，北京邮电大学教授、博士生导师，主要研究方向为基于 TCP/IP 网络的网络管理技术、网络安全技术、移动 IP 技术、IPv6 技术及其应用。

魏伟（1989-），男，湖北宜昌人，北京邮电大学硕士生，主要研究方向为 IPv6 数据分组分类算法。