doi:10.11887/j.cn.202203022 http://journal.nudt.edu.cn

决策树报文分类算法*

吕高锋,谭 靖,乔冠杰,严锦立
(国防科技大学 计算机学院,湖南 长沙 410073)

摘 要:报文分类是网络的基本功能,研究人员在过去二十年提出了众多解决方案,其中决策树报文分类算法由于吞吐量高、适用于多字段、可流水线化等特点受到了广泛关注和深入研究。介绍了决策树算法最新研究成果,阐述了决策树报文分类算法的几何意义、常用技术和测试基准,从节点切割技术和规则集分组技术两个维度对决策树算法进行了系统分析和归纳。针对两类常用的决策树构建技术介绍了其中的典型算法,对比了各种典型算法的设计思路和特点,分析了它们的适用场景。总结并展望了决策树算法的下一步研究方向。

关键词:报文分类;决策树算法;节点切割;机器学习
中图分类号:TP393 **文献标志码**:A **文章编号**:1001-2486(2022)03-184-10

Decision tree algorithm for packet classification

LYU Gaofeng, TAN Jing, QIAO Guanjie, YAN Jinli
(College of Computer Science and Technology, National University of Defense Technology, Changsha 410073, China)

Abstract: Packet classification is the fundamental function of network, and researchers have proposed many packet classification solutions in the past two decades. Among them, the decision tree algorithm for packet classification has received extensive attention and in-depth research due to its high throughput, suitable for multiple fields and pipelining. The recent research on the decision tree algorithm for packet classification was introduced, the geometric meaning, common techniques and test benchmarks of the decision tree algorithm were described, and the decision tree algorithm from the two dimensions of node cutting technology and rule set grouping technology were systematically analyzed. The typical algorithms of the two types of common technologies for building decision tree were introduced respectively, the design ideas and characteristics of various algorithms were compared, and their applicable scenarios were given. The conclusion and discuss the future work of decision tree algorithms were stated out.

Keywords: packet classification; decision tree algorithm; node cutting; machine learning

报文分类是各种网络服务所需的基本技术之一,如安全防护、策略路由和服务质量(quality of service,QOS)等。因此,报文分类在云服务提供商、互联网服务提供商(internet service provider,ISP)和互联网交换中心(internet exchange point,IXP)的核心网络设备中广泛部署[1]。报文分类的目的是根据网络报文头部的多个字段值来区分报文类型,从而对报文执行相应的差异化操作。

报文分类解决方案可分为两大类:硬件方案和软件方案[2]。其中基于三态内容可寻址存储器(ternary content addressable memory,TCAM)的硬件方案[3-6]在行业中占据主导地位,其利用TCAM将所有规则存储在相联存储器中,然后将报文与这些规则并行匹配,优点是提供了常量的分类时间、实现了线速查找,但存在高成本、高功耗、可扩展性较差等缺点[7]。软件方案主要有决策树算法、元组空间搜索算法[8-10]等,其中决策树算法由于具有吞吐量高、适用于多字段和可流水线化[11]等特点,被认为是最有希望替代TCAM方案的方案之一。

由于报文分类具有重要的研究和实用价值,近十年一直是人们的研究热点。并且网络带宽的持续增长和网络应用复杂性的不断增加[12],给报文分类带来了高维度和动态性等新挑战,研究人员仍在寻求新的、更高效的报文分类解决方案。近年来决策树报文分类算法研究取得了重要进展,新的算法层出不穷,并且出现了跨领域的趋势,例如与机器学习结合取得了初步成果,但是相关研究的综述性文章[13-15]并未涉及决策树算法新的研究进展,对近年来优秀的研究成果也缺乏

* 收稿日期:2020-11-02
基金项目:国家重点研发计划资助项目(2018YFB1800505)
作者简介:吕高锋(1980—),男,陕西扶风人,副研究员,博士,硕士生导师,E-mail:lvever@nudt.edu.cn

系统的分析与总结。

为进一步推动基于决策树的报文分类算法的研究与发展,本文从节点切割技术、规则集分组技术两个维度对决策树算法进行了系统的分析和总结,并对比了各类算法的设计思路和特点。

1　决策树算法分析

1.1　报文分类问题

报文分类器含有一个规则列表,其中每条规则由优先级、匹配域(match field)和匹配报文时采取的动作(action)组成。报文分类问题是从规则列表中找到报文所匹配的规则,并返回其中优先级最高的匹配规则。

对报文分类问题的正式定义[16]如下:

规则列表中含有 n 条规则,每条规则由 d 个匹配域(维度)和动作域组成,其中每个匹配域 $f[i](1 \leqslant i \leqslant d)$ 都是关于报文头部的第 i^{th} 字段的表达式。表达式的形式可以为精确匹配、前缀匹配或范围匹配形式。如标准 IPv4 五元组中的源和目的 IP 地址属于前缀匹配,源和目的端口号属于范围匹配,协议类型则属于精确匹配。

如果 $\forall i(1 \leqslant i \leqslant d)$,报文 P 的第 i^{th} 报头字段值都能满足规则 $R_t(1 \leqslant t \leqslant n)$ 的第 i 个匹配域 $f[i]$,则认为报文 P 与特定规则 R_t 匹配。如果报文 P 同时匹配了多条规则,则返回优先级最高的匹配规则。

表 1 给出了含有 6 条规则的二维分类器示例。其中每个维度的位宽为 4,“ * ”表示通配符,优先级数值越小的规则对应的优先级越高。

表 1　二维分类器示例

Tab. 1　Example 2-dimensional classifier

规则	优先级	X 域	Y 域	动作
R_1	1	111 *	*	转发
R_2	2	110 *	*	转发
R_3	3	*	010 *	转发
R_4	4	*	011 *	转发
R_5	5	01 **	10 **	修改
R_6	6	*	*	丢弃

1.2　算法基本思想

报文分类问题可等价为计算几何中的多维空间点定位问题,即报文头部中的字段(如源和目的 IP 地址、源和目的端口号以及协议类型)表示几何空间中的维度,报文被表示为该空间中的点,而规则被表示为空间中的超立方体。然后,报文分类问题等价于找到包含点(报文)的优先级最高的超立方体(规则):如果报文 P 与特定规则 R 匹配,则表示报文 P 的点将落入规则 R 指定的超立方体中。

经过数学推导[17],在具有 n 个互不重叠的超立方体的 d 维几何空间中,当 $d>3$ 时,多维空间点定位问题理论上具有 $O(\log_2 n)$ 时间复杂度和 $O(n^d)$ 空间复杂度,或者具有 $O(\log_2 n)^{d-1}$ 时间和 $O(n)$ 空间复杂度。而对于五元组或更高维度的报文分类,这两个选择都不具吸引力。因此,需要采取有效的数据结构来组织规则集,决策树算法便是理想的选择之一。

在基于决策树算法的方案中,对报文分类问题采取几何视图,并构建决策树。树的根节点覆盖了包含所有规则的整个搜索空间,然后递归地将搜索空间切割为更小的子空间,直到每个子空间包含的规则数不大于预定义的阈值时停止切割,并将当前子空间对应的节点设为叶节点。如果规则了跨越多个子空间,则在切割时会发生不必要的规则复制。当有报文到达时,基于报文头部字段中的关键字值遍历决策树,以在叶节点处找到匹配规则。表 1 中的二维规则集对应的几何视图如图 1 所示。

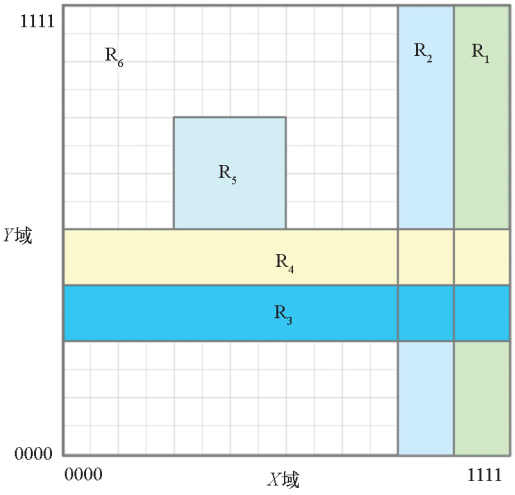


图 1　表 1 中规则集的几何视图

Fig. 1　Geometric view of rules in Tab. 1

1.3　算法类型

现有报文分类问题解决方案主要分为硬件方案和软件方案,基本框架如图 2 所示。

其中硬件方案主要有 TCAM 和位向量[18-19],决策树算法属于软件解决方案。构建决策树有两类常用技术:节点切割和规则集分组。

1.3.1　节点切割

节点切割基本思想是将整个多维规则集视为

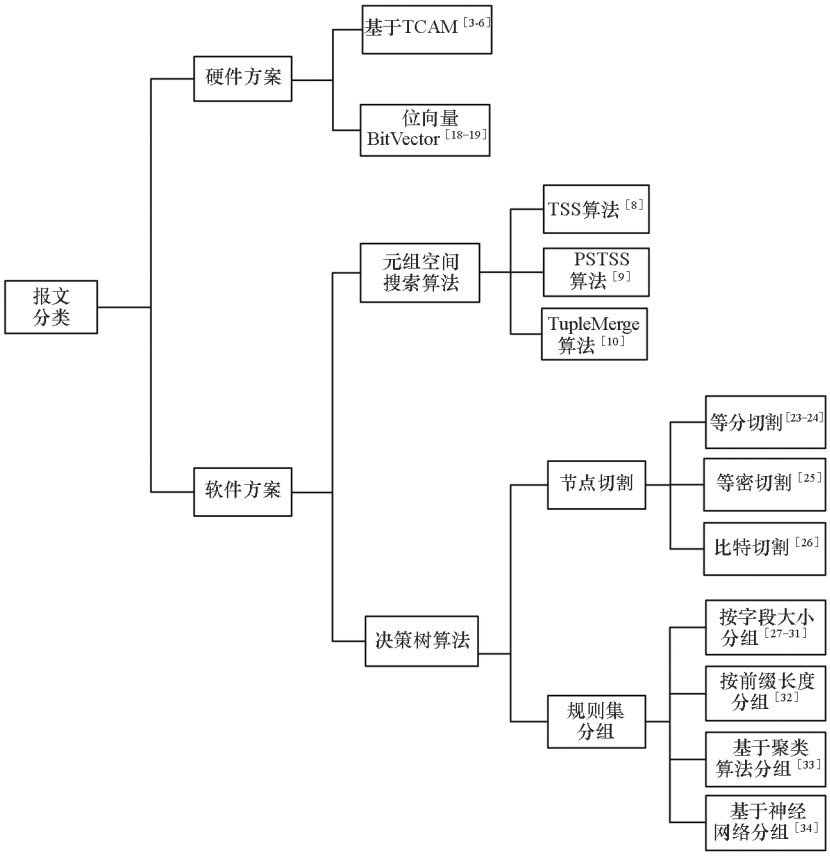


图 2 报文分类解决方案的基本框架

Fig. 2 Basic framework for the solution of packet classification problems

树的根节点,然后沿一个或多个特定的维度切割节点构建决策树。算法从包含所有规则的根节点开始,迭代地切割节点,直到每个叶节点包含的规则数不大于预定义的阈值时停止切割,从而构建单棵决策树。各类决策树算法在节点切割方面的主要区别为:

1)切割维度的选择。选择哪个维度切割最优;一次选择单个维度还是多个维度进行切割。

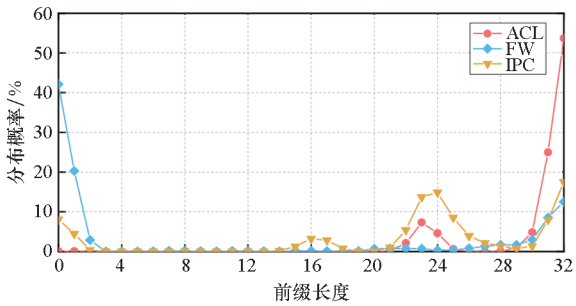
2)切割端点的选择。包括对节点进行等分切割或者等密切割。等分切割能快速将节点等分为 2^n 个子节点,但会带来严重的规则复制问题,即同一条规则分布在多个子节点中;而等密切割能够缓解规则复制问题,但也存在树深度增加、节点索引复杂等问题。

此外,新提出的比特切割技术使用比特级视图(bit-level view),利用规则每一位都可表示为 0、1 或者 *(通配符)的特点,选择其中有效比特将规则映射到各个子节点中,从而避免了盲目地切割整个搜索空间。从另一个角度看,等分切割也是一种特殊的比特切割,但只允许使用连续的最高有效位。

1.3.2 规则集分组

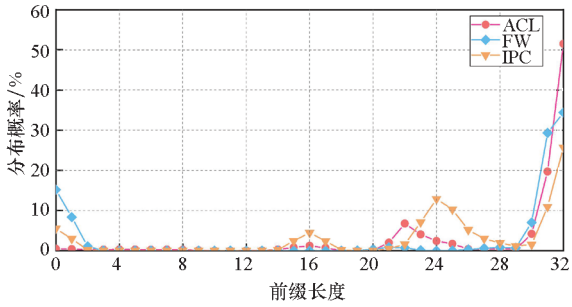
事实上,规则集中部分规则之间存在明显的差异。对访问控制列表(access control list, ACL)、防火墙(firewall, FW)和 IP 链(IP chain, IPC)类型规则集进行统计分析,结果如图 3 所示。从图中可得,IP 地址字段前缀长度为边缘分布,即大部分位于 0 附近或 32 附近。

因此不加任何区分直接切割整个搜索空间将导致严重的规则复制,其中一个解决方案便是分治思想,即将具有相似特征的规则放到一个规则子集中,然后应用节点切割技术为每个子集单独构建决策树,最后形成多棵决策树。规则集分组



(a) 源 IP 地址前缀分布

(a) Distribution of source IP address



(b) 目的 IP 地址前缀分布

(b) Distribution of destination IP address

图 3 地址前缀分布

Fig. 3　Distribution of address

的主要方法有:

1)按字段大小分组。根据规则在每个字段覆盖的范围来划分规则子集,该类方法应用最广泛。

2)按前缀长度分组。根据规则在特定字段的前缀长度来划分规则集。

3)基于聚类算法分组。使用聚类算法来划分规则子集。

4)基于深度神经网络模型分组。将机器学习技术应用到报文分类问题中,如使用深度神经网络模型来学习优化节点切割和规则集分组,以获得最大的奖励函数(分类速度、内存消耗等),从而构建性能优异的决策树。

按字段大小、前缀长度分组等启发式策略建立在对规则集分布特征观察的基础之上,原理相对简单、容易实现,但对于不同的规则集,往往需要手动调整部分参数以获得理想结果;聚类算法、神经网络模型则可以使用机器学习来替代人工调整,实现对规则子集的自适应划分,但需要经过大量的训练和迭代才能收敛。

1.4　面临的挑战

设计高效的决策树算法面临的最大挑战是规则复制问题和数据结构更新问题。

决策树算法牺牲了线性空间来获得较高的分类速度,是典型的以空间换时间。规则复制问题是困扰决策树类型解决方案的主要问题之一,早期的决策树算法在规则集规模较大的时候产生大量规则复制,甚至耗尽系统内存。规则集分组技术的主要目的就是解决规则复制问题。

决策树算法存在的另一大挑战是规则更新速度较慢。在软件定义网络中,虚拟交换机得到大规模部署,目前主流的虚拟交换机如 Open vSwitch,要求算法的数据结构规则更新时间复杂度为常量级别,即时间复杂度为 $O(1)$,因此采用了优先级元组空间搜索(priority sorted tuple space search, PSTSS)算法。而决策树算法由于存在规则复制问题,导致数据结构更新困难,甚至在更新时需要重新构造决策树。更新速度较慢已经成为限制决策树算法应用的关键因素之一。

决策树算法目前的研究热点是在保证算法性能的基础上,解决规则复制问题和更新速度问题,在算法吞吐量、内存消耗和更新速度等重要指标之间取得更好的折中。

1.5　测试基准

报文分类算法的测试基准为 ClassBench[20] 和 ClassBench-ng[21]。

ClassBench 是 2007 年由 Taylor 等提出,用于对报文分类算法进行基准测试的开源工具。ClassBench 能够接收规则集参数配置文件,生成精确模拟实际规则集分布特征的规则集。ClassBench 中共提供了 12 个参数文件,支持生成 ACL、FW 和 IPC 三种类型、不同规模的 IPv4 五元组规则集。此外,ClassBench 还能够针对规则集生成特定长度的报文头部序列,以对算法性能进行测试。

IPv6 协议和软件定义网络(software defined network, SDN)技术的发展给报文分类问题带来了新的挑战,而 ClassBench 仅支持生成 IPv4 环境下的规则集。针对这一问题,Matoušek 等于 2017 年提出了新的开源基准测试工具 ClassBench-ng,能够生成 IPv4、IPv6 规则集和 OpenFlow[22] 1.0 流表。此外,ClassBench-ng 还提供了一种从实际规则集提取参数文件的机制,并可上传至 ClassBench-ng 工具库,以生成不同应用环境下的规则集,适应不同研究人员的需求。

2　基于节点切割的单棵决策树

节点切割技术采用几何视图,将包含所有规则的整个搜索空间视作根节点,利用启发式等策略选择合适的切割维度和端点切割当前节点,获得子空间,然后递归地将搜索子空间分解为更小的空间,直到满足特定条件停止递归,最终构建基本的单棵决策树。节点切割方案主要有等分切割、等密切割和比特切割。

2.1　等分切割

HiCuts 算法[23] 是由 Gupta 等提出的最早用于报文分类的决策树算法,一次选择单个切割维度来等分规则空间,目的是将规则尽可能均匀地分离到树的叶节点中。HiCuts 从覆盖整个规则空间的根

节点开始,在单个维度上切割搜索空间,以创建一组大小相等的子空间。HiCuts 算法中应用了多种启发式策略来选择最优的切割维度和切割次数。

HiCuts 算法存在两个主要缺陷:首先, HiCuts 算法仅考虑在一个节点处切割一个维度,这将增加树的深度,从而增加了树遍历所需的时间。其次,算法在构建决策树后存在大量的规则冗余,消耗了大量内存。针对上述缺陷, Singh 等提出了 HyperCuts 算法[24],改进了树的深度和内存消耗问题。HyperCuts 算法提出同时切割多个维度,而不是在一个节点处只切割单个维度,从而减少了树的深度。其次,HyperCuts 提出了公共规则向上推送的优化方法,允许将所有同级子节点通用的规则向上移动到父节点处,以缓解规则复制问题。

HyperCuts 算法虽然在一定程度上缓解了树的深度较大和规则复制问题,但仍存在相当大的规则冗余,尤其是对于大规模的规则集。其问题在于等分切割本身的局限性,即等分切割适用于规则分布均匀的场合,而实际上规则集的分布并不是均匀的。

2.2 等密切割

针对 HiCuts 和 HyperCuts 算法等分切割搜索空间导致大量规则冗余的问题, Qi 等提出了 HyperSplit 算法[25]。HyperSplit 的思想是构建一个平衡的二叉树,以便规则均匀地分布在其子节点中,并沿着规则边界进行切割,从而防止过多的规则复制。HyperSplit 算法在每个内部节点处采用启发式策略选择一个最佳的切割维度和切割端点,将搜索空间拆分成两个大小不等的子空间,其中包含的规则数近似相等。选择切割端点的策略包括规则平衡、范围平衡和加权分段平衡。

HyperSplit 算法提出了等密切割的思路,进一步缓解了规则复制问题,但由于算法构建的决策树为二叉树,且每次只能判断一次维度,因此随着规则集规模的扩大和维度的增加,树的深度也会增加,相应的遍历决策树所需的访存次数也将增加。

2.3 比特切割

Liu 等提出的 BitCuts 算法[26]是对节点切割技术一次新的尝试。不同于等分切割和等密切割算法, BitCuts 算法充分利用了规则的二进制特点,即对于精确值和前缀表达式的字段,其每一位取值均为 0、1 或 *,而范围表达式可转换为前缀表达式,然后通过贪心策略迭代地选择其中的有效比特,将规则分离到叶节点中。

BitCuts 算法利用了规则的二进制特点,迭代地选择有效比特来构建决策树,在内存访问次数上相比 HiCuts、HyperCuts 和 HyperSplit 算法有了大幅度减少,因此吞吐量更高。但比特选择策略的效率很大程度上决定着算法性能。典型的比特选择策略有暴力策略、启发式策略等。

对表 1 中的规则集分别使用 HiCuts 算法、HyperCuts 算法、HyperSplit 算法和 BitCuts 算法构建决策树,结果如图 4 ~ 7 所示,其中叶节点中最多可包含的规则数量为 4。采用不同节点切割方案的决策算法设计思路及特点如表 2 所示。

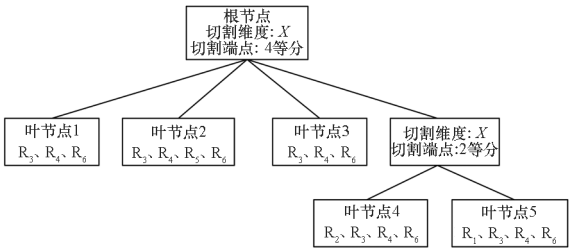


图 4 HiCuts 算法构建的决策树

Fig. 4 Decision tree built by HiCuts

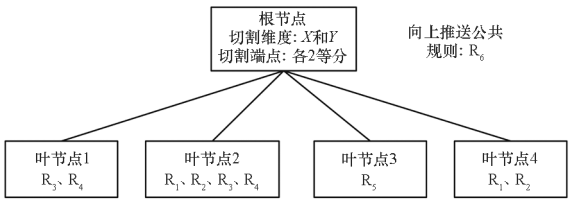


图 5 HyperCuts 算法构建的决策树

Fig. 5 Decision tree built by HyperCuts

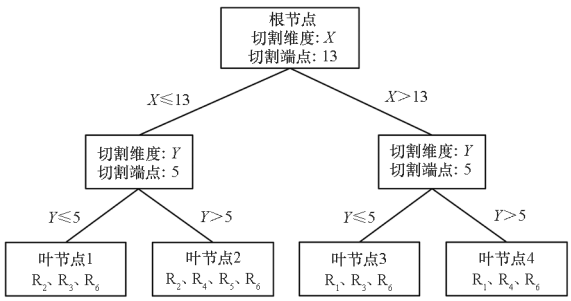


图 6 HyperSplit 算法构建的决策树

Fig. 6 Decision tree built by HyperSplit

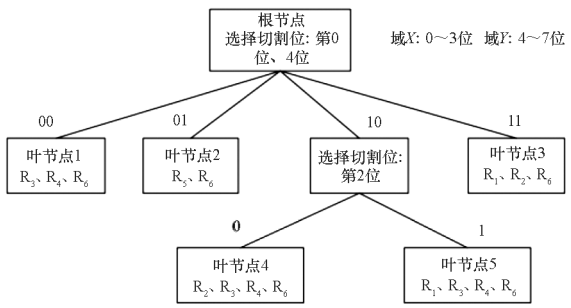


图 7 BitCuts 算法构建的决策树

Fig. 7 Decision tree built by BitCuts

表 2　节点切割的决策树算法

Tab. 2　Decision tree algorithms using node cutting

类型	典型算法	设计思路	优点	缺点
等分切割	HiCuts	一次选择单个维度,将搜索空间划分为等大小的子空间	首个决策树分类算法,快速切割规则空间	树深度较大,规则复制问题严重,内存消耗大
等分切割	HyperCuts	一次选择多个维度,将搜索空间划分为等大小的子空间	HiCuts 的改进,树深度较小,分类速度较快,规则复制相比 HiCuts 有所优化	内存消耗仍然较大,可扩展性较差
等密切割	HyperSplit	一次选择单个维度和特定的端点,将搜索空间划分为两个规则数几乎相等的子空间	进一步优化规则复制问题,内存消耗小	树深度较大,遍历树所需的访存次数较多
比特切割	BitCuts	选择规则中的有效比特来划分搜索空间	分类速度在 4 种算法中最快,吞吐量高	内存消耗比 HyperSplit 算法大

总的来说,三类节点切割算法中,等分切割优点在于简单快速,分类速度较快,但是内存消耗较大,适用于对分类速度要求高而对内存消耗不敏感的场景。而等密切割则在内存消耗方面占有优势,但算法吞吐量有所下降,适用于对内存要求严格的场景。比特切割是一种更新颖和灵活的切割算法,在时间性能和空间性能之间实现了更好的折中,能够以合理的内存消耗提供较高的分类速度。

3　基于规则集分组的多棵决策树

研究人员在节点切割技术方面开展了深入的研究,充分挖掘了单棵决策树的性能,但仍无法有效解决规则复制问题。针对单棵树的固有缺陷,提出了规则集分组的想法。根据规则集的某些特征来划分子集,将具有相似特征的规则放在一个子集里单独构建决策树,最终形成多棵决策树。将规则集分成子集的规则集分组技术可减少每个子集中的规则重叠,从而改善单棵树中严重的规则复制问题。

规则集分组的主要方法有:按字段大小分组、按前缀长度分组、基于聚类算法分组和基于深度神经网络分组。

3.1　按字段大小分组

EffiCuts 算法[27]是使用规则集分组最具代表性的算法。EffiCuts 算法基于两个关键结论:①规则大小的变化:规则集中有许多规则重叠,重叠的规则在大小上差异很大。因此,具有重叠的小规则和大规则的单棵树需要精细切割来分隔小规则,但会复制大规则。②规则空间密度的变化。针对重叠规则大小变化的情况,EffiCuts 算法提出了“可分离树”的思想,通过将小规则和大规则放在不同的树中减少复制,再应用 HyperCuts 算法为每个子集构建决策树。例如对于图 2 中的 6 条规则,可根据规则在 X、Y 两个域上的大小,分为 4 个子集$\{R_1,R_2\}$、$\{R_3,R_4\}$、$\{R_5\}$和$\{R_6\}$。

对于字段大小的区分,算法通过规则投影到每个字段上的覆盖区间的大小来界定,定义了一个称为大分数(largeness fraction)的分界点,一般情况下取值为 0.5。对于 d 维规则集,EffiCuts 算法最多能产生 2^d 个子集,呈指数级别增长。EffiCuts 算法虽然考虑子树过多的情况,采用子树合并方法来消除部分子树,但仍然会生成大量的子树,这将导致较多的内存访问次数,不利于算法的扩展。使用 EffiCuts 算法为表 1 中的二维规则集构建的决策树如图 8 所示。

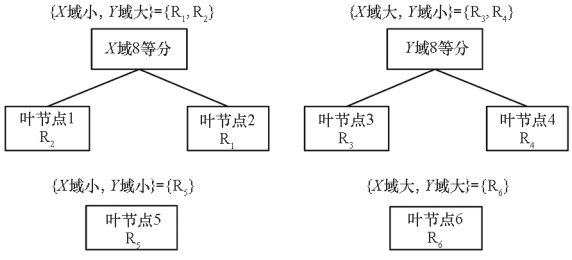


图 8　EffiCuts 算法构建的决策树

Fig. 8　Decision tree built by EffiCuts

HybridCuts 算法[28]和 SmartSplit 算法[29]针对 EffiCuts 算法中子类别过多的问题进行改进。在这两个算法中仅使用了源 IP 地址和目的 IP 地址两个字段而不是全部的字段来划分规则集,限制了规则子集的数量最多为 4。

Li 等将按字段大小分组扩展到 OpenFlow 等多维规则集,提出了 CutSplit 算法[30]。CutSplit 的基本思想是基于很少的几个字段对规则集进行分组,然后预切割和后拆分结合构建决策树。CutSplit 算法的框架如图 9 所示。

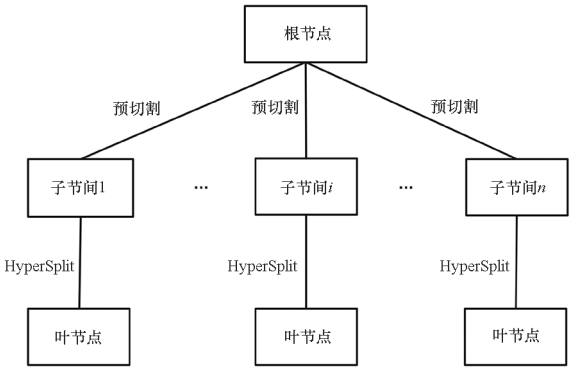


图 9 CutSplit 算法框架

Fig. 9 Framework of CutSplit

算法具体实现为:根据小的维度将整个规则集划分为若干子集,划分依据是从 F 维中挑选前 m 个包含最多的小规则数量的维度,保证这 m 个小维度覆盖绝大部分的规则(≥95%),并且在算法中定义了阈值向量,使得规则"大""小"定义更多灵活;在划分子集后,对每个子规则集运用简单、快速的固定小字段等分切割,以得到更小的子空间;对每个子空间运用 HyperSplit 算法,以尽量减小冗余并且达到最坏情况下的有界性能。

CutSplit 算法相比 EffiCuts 算法,通过精心挑选少数维度划分规则子集,性能有了改善,但存在切割和拆分的边界难以确定、算法性能不稳定等缺点。

TabTree 算法[31]是继 CutSplit 算法之后的一项新的研究成果,其规则集分组方法类似 CutSplit,但对于每个子集使用了平衡位选择的方法来构建决策树,而不是预切割和后拆分的组合,从而实现了高速分类和规则快速更新。

3.2 按前缀长度分组

Daly 等提出的 ByteCuts 算法[32],根据规则在 IP 地址字段的前缀长度来划分规则子集。算法将具有相同的核心位集合的所有规则放在同一树中,然后对单棵树使用字节切割(以半字节为单位的比特切割)构建决策树。ByteCuts 算法对子树数量与子树大小之间的平衡进行了深入研究:子树数量过多会导致访存次数增加;过少会使得单棵树的体积大,从而在单棵树内部产生更多的规则复制。ByteCuts 算法定义了两个阈值:切割效率(决定子树内部冲突率)和树大小,目标是最大化树中规则的数量(最小化树的数量),同时满足一定的平衡要求。

算法具体实现为:通过两个阈值(切割效率和决策树大小)来选择字段长度对(f, W_f);然后将为字段 f 上前缀长度至少为 W_f 的所有规则放入同一个子集中,并创建单个决策树;对剩下规则重复该过程,直到剩余规则的数量低于某个特定阈值 τ(如 5%)停止。

ByteCuts 算法因以半个字节为单位切割搜索空间,构建决策树数据结构所需时间较短。此外,算法还具有较强的灵活性,可通过调整阈值参数在时间性能和空间性能之间取得较好的折中。

3.3 基于聚类算法分组

Fong 等提出的 ParaSplit 算法[33]的基本思想是使用聚类算法划分子集,然后对每个子集使用 HyperSplit 算法构建决策树。ParaSplit 算法首先使用范围 - 点转换算法,通过将每个字段的起点和终点视为单独的维度,从而将 F 维空间中的超矩形转化为 $2F$ 维空间中的点(F 维规则被表示为 $2F$ 维空间中的点)。然后利用 K-means 聚类算法有效划分规则子集,应用模拟退火算法来寻找最优划分,并对每个子集使用 HyperSplit 算法构建决策树。在使用 K-means 聚类算法划分子集时,提出了最小距离、最大距离和距离原点距离三种启发式策略,其中最小距离策略的性能更优。

ParaSplit 算法通过聚类算法实现了规则集的自适应划分,但 K-means 算法的性能受初始 k 个中心点的选择影响较大,且算法需要上万次的迭代才能获得理想结果,预处理时间较长。

3.4 基于深度神经网络分组

NeuroCuts 算法[34]使用了深度神经网络模型来学习构建高效的决策树,算法框架如图 10 所示。神经网络模型的环境由规则集和当前决策树组成;代理使用一个神经网络模型,选择最佳的切割或规则集分组操作来增量地构建树;当前节点的可用操作(节点切割或规则集分组)在每个步骤由环境通告,代理选择其中一个操作生成决策树。随着时间推移,代理学会优化其决策,以最大化从环境中获得的奖励函数,最终构建性能优异的决策树。奖励函数可以是分类速度、消耗内存或两者的结合。NeuroCuts 算法在训练过程中使用了马尔可夫决策过程,并且结合了已有的研究成果,如按字段分组等策略。

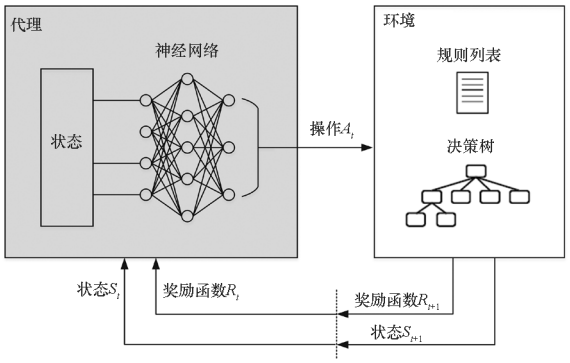


图 10　NeuroCuts 算法框架

Fig. 10　Framework of NeuroCuts

NeuroCuts 算法在报文分类和机器学习结合方面开创先河,是一次有益的大胆尝试,有助于启发新一代基于机器学习的分类算法。

表 3 给出了采用不同规则集分组方案的决策树算法的设计思路和特点。总的来说,按字段大小分组和按前缀长度分组原理简单,容易实现,较好地利用了规则集的几何分布特征,达到了较好的性能,且更新速度也有优势。但缺点在于需要自定义阈值参数,且在面对不同的规则集时可能需要重新调整参数。按字段大小分组适用于规则集更新频繁,或者面向特定应用领域和特定规则集的场景。

表 3　规则集分组的决策树算法

Tab. 3　Decision tree algorithms using rule set partition

类型	典型算法	设计思路	优点	缺点
按字段大小分组	EffiCuts CutSplit	根据规则在每个字段上覆盖的范围大小划分规则子集	易于实现,预处理速度较快,应用广泛	需定义阈值参数,在面对不同规则集时面临调整参数的问题
按前缀长度分组	ByteCuts	根据规则在 IP 地址字段上的前缀长度划分规则子集	易于实现,预处理速度较快,分类速度较快	仅适用于 IP 地址字段,需要定义阈值参数
基于聚类算法分组	ParaSplit	基于 *K*-means 聚类算法和模拟退火算法划分规则子集	实现了规则集的自适应分组,通用性较强	运算量大,处理速度较慢,算法需多次迭代才能收敛
基于深度神经网络分组	NeuroCuts	使用深度神经网络模型学习如何划分规则子集	与机器学习较好地结合,学习构建高效的决策树,通用性较强	训练时间较长,需要大量的计算

而基于聚类算法分组和基于深度神经网络分组则将报文分类问题与机器学习较好结合在一起,利用机器学习技术来解决经典的网络问题,实现了规则集的自适应分组,通用性较强。但预处理速度较慢,往往需要大量的计算和迭代才能收敛,从而获得理想的性能。适用于规则更新频率较低,但对算法性能要求较高的场景。

4　决策树算法研究方向

网络技术的进步以及网络应用的复杂性和多样性,给报文分类带来了新挑战,研究人员仍致力于提出更高效的决策树报文分类算法。决策树下一步的研究方向主要有以下四个方面:

一是节点切割技术创新。利用规则集分布特征,提出新颖的、更高效的节点切割技术或者规则集分组技术,例如新的比特切割技术等。目前常用的有效比特选择标准有规则可分离性[26]、子节点内规则数量最少化[35]、信息熵[36]和子节点内规则数量平衡[37]等,但以上方法存在预处理时间较长、容易陷入局部最优等缺点,因此研究更好的比特切割技术,有助于提升决策树算法的时间性能和空间性能。

二是规则集分组与节点切割技术结合使用。根据各类规则集分组技术与节点切割技术的特点,积极尝试将它们结合使用以得到性能更优的决策树,这是当前研究的热点,新的研究成果也多集中在该方向上。如按字段大小分组与等分切割结合使用能够取得较好效果[30],而按前缀长度分组后则更适合使用比特切割来构建决策树。

三是异构的算法架构。即各类软件方案、软件方案和硬件方案的结合使用。如 Partition Sort 算法[38]就充分利用了元组空间搜索(tuple space search, TSS)算法和决策树算法的优点,从而兼顾了决策树算法的高吞吐量和 TSS 算法的快速更

新。CutTSS 算法[39]则创造性地将等分切割与元组空间搜索算法结合使用,因此能够同时支持高速分类和快速更新。此外,软硬件协同处理也是一个研究热点,首先设计高性能的决策树算法,再将决策树映射到 FPGA 等硬件平台上[40-41],利用硬件大规模并行、流水线架构的特点来提升算法性能,以兼顾软件算法灵活性和硬件高性能的优点。

四是跨领域研究。将决策树算法与机器学习等技术结合,借助于其他领域的专业知识来解决报文分类问题。如使用深度神经网络模型学习构建高效的决策树是一个新的研究方向,对神经网络模型的研究和改进也有助于提升决策树算法的性能。

5 结论

本文主要介绍了最新的决策树研究成果,从节点切割和规则集分组技术两个方面对决策树算法进行了系统分析和总结。总的来说,构建单棵决策树的技术主要包括等分切割、等密切割和比特切割等,已经得到了较为深入的研究。规则集分组技术主要包括按字段大小分组、按前缀长度分组、基于聚类算法分组和基于深度神经网络分组等,目前仍是研究热点。一个好的规则集分组方案能够有效解决困扰决策树算法已久的规则复制问题,同时提高算法的数据结构更新速度,从而提升算法的应用价值。

随着网络带宽的持续增长和网络应用复杂性的不断增加,决策树报文分类算法的发展趋势是更高的吞吐量、更小的内存消耗和更快的更新速度,并支持高维度扩展,尤其是在数据中心等规则集频繁更新的应用环境中,算法的更新性能显得越来越重要。

参考文献(References)

[1] BABOESCU F, SINGH S, VARGHESE G. Packet classification for core routers: is there an alternative to CAMs? [C]//Proceedings of 22nd Annual Joint Conference of the IEEE Computer and Communications Societies, 2003.

[2] TAYLOR D E. Survey and taxonomy of packet classification techniques[J]. ACM Computing Surveys, 2005, 37(3): 238-275.

[3] LAKSHMINARAYANAN K, RANGARAJAN A, VENKATACHARY S. Algorithms for advanced packet classification with ternary CAMs [J]. ACM SIGCOMM Computer Communication Review, 2005, 35(4): 193-204.

[4] KOGAN K, NIKOLENKO S, ROTTENSTREICH O, et al. SAX-PAC (scalable and expressive packet classification)[J]. ACM SIGCOMM Computer Communication Review, 2014, 44(4): 15-26.

[5] HE P, ZHANG W Y, GUAN H T, et al. Partial order theory for fast TCAM updates [J]. IEEE/ACM Transactions on Networking, 2018, 26(1): 217-230.

[6] ULLAH I, ULLAH Z, AFZAAL U, et al. DURE: an energy- and resource-efficient TCAM architecture for FPGAs with dynamic updates[J]. IEEE Transactions on Very Large Scale Integration Systems, 2019, 27(6): 1298-1307.

[7] MISHRA T, SAHNI S. PETCAM—a power efficient TCAM architecture for forwarding tables[J]. IEEE Transactions on Computers, 2012, 61(1): 3-17.

[8] SRINIVASAN V, SURI S, VARGHESE G. Packet classification using tuple space search[J]. ACM SIGCOMM Computer Communication Review, 1999, 29(4): 135-146.

[9] PFAFF B, PETTIT J, KOPONEN T, et al. The design and implementation of Open vSwitch[C]// Proceedings of 12th USENIX Symposium on Networked Systems Design and Implementation, 2015: 117-130.

[10] DALY J, BRUSCHI V, LINGUAGLOSSA L, et al. TupleMerge: fast software packet processing for online packet classification[J]. IEEE/ACM Transactions on Networking, 2019, 27(4): 1417-1431.

[11] JIANG W R, PRASANNA V K. Large-scale wire-speed packet classification on FPGAs [C]//Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays, 2009: 219-228.

[12] FIRESTONE D, PUTNAM A, MUNDKUR S, et al. Azure accelerated networking: SmartNICs in the public cloud [C]// Proceedings of 15th USENIX Symposium on Networked Systems Design and Implementation, 2018: 51-64.

[13] 孙毅, 刘彤, 蔡一兵, 等. 报文分类算法研究[J]. 计算机应用研究, 2007, 24(4): 5-11.
SUN Y, LIU T, CAI Y B, et al. Research on packet classification algorithm [J]. Application Research of Computers, 2007, 24(4): 5-11. (in Chinese)

[14] 亓亚烜, 李军. 高性能网包分类理论与算法综述[J]. 计算机学报, 2013, 36(2): 408-421.
QI Y X, LI J. Theoretical analysis and algorithm design of high-performance packet classification algorithms [J]. Chinese Journal of Computers, 2013, 36(2): 408-421. (in Chinese)

[15] 张杰鑫, 张铮. 包分类算法研究综述[J]. 计算机工程, 2015, 41(12): 111-118.
ZHANG J X, ZHANG Z. Overview of packet classification algorithm research [J]. Computer Engineering, 2015, 41(12): 111-118. (in Chinese)

[16] GUPTA P, MCKEOWN N. Algorithms for packet classification[J]. IEEE Network, 2001, 15(2): 24-32.

[17] OVERMARS M H, VAN DER STAPPEN F A. Range searching and point location among fat objects[J]. Journal of Algorithms, 1996, 21(3): 629-656.

[18] LAKSHMAN T V, STILIADIS D. High-speed policy-based packet forwarding using efficient multi-dimensional range matching[J]. ACM SIGCOMM Computer Communication Review, 1998, 28(4): 203-214.

[19] JIANG W R, PRASANNA V K. Field-split parallel architecture for high performance multi-match packet classification using FPGAs [C]//Proceedings of the 21st Annual Symposium on Parallelism in Algorithms and Architectures, 2009: 188-196.

[20] TAYLOR D E, TURNER J S. ClassBench: a packet classification benchmark [J]. IEEE/ACM Transactions on Networking, 2007, 15(3): 499 - 511.

[21] MATOUŠEK J, ANTICHI G, LUČANSKÝ A, et al. ClassBench-ng: recasting ClassBench after a decade of network evolution [C]//Proceedings of ACM/IEEE Symposium on Architectures for Networking and Communications Systems, 2017: 204 - 216.

[22] MCKEOWN N, ANDERSON T, BALAKRISHNAN H, et al. OpenFlow: enabling innovation in campus networks [J]. Computer Communication Review, 2008, 38(2): 69 - 74.

[23] GUPTA P, MCKEOWN N. Packet classification using hierarchical intelligent cuttings [C]//Proceedings of IEEE Hot Interconnects, 1999.

[24] SINGH S, BABOESCU F, VARGHESE G, et al. Packet classification using multidimensional cutting [C]// Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, 2003: 213 - 224.

[25] QI Y, XU L, YANG B, et al. Packet classification algorithms: from theory to practice [C]//Proceedings of IEEE INFOCOM, 2009: 648 - 656.

[26] LIU Z, WANG X, YANG B H, et al. BitCuts: towards fast packet classification for order-independent rules [J]. Computer Communication Review, 2015, 45(5): 339 - 340.

[27] VAMANAN B, VOSKUILEN G, VIJAYKUMAR T N. EffiCuts: optimizing packet classification for memory and throughput [J]. Computer Communication Review: A Quarterly Publication of the Special Interest Group on Data Communication, 2010, 40(4): 207 - 218.

[28] LI W J, LI X F. HybridCuts: a scheme combining decomposition and cutting for packet classification [C]// Proceedings of IEEE 21st Annual Symposium on High-Performance Interconnects, 2013: 41 - 48.

[29] HE P, XIE G G, SALAMATIAN K, et al. Meta-algorithms for software-based packet classification [C]//Proceedings of IEEE 22nd International Conference on Network Protocols, 2014: 308 - 319.

[30] LI W J, LI X F, LI H, et al. CutSplit: a decision-tree combining cutting and splitting for scalable packet classification [C]//Proceedings of IEEE INFOCOM, 2018: 2645 - 2653.

[31] LI W J, YANG T, CHANG Y K, et al. TabTree: a TSS-assisted bit-selecting tree scheme for packet classification with balanced rule mapping [C]//Proceedings of ACM/IEEE Symposium on Architectures for Networking and Communications Systems, 2019: 1 - 8.

[32] DALY J, TORNG E. ByteCuts: fast packet classification by interior bit extraction [C]//Proceedings of IEEE INFOCOM, 2018: 2654 - 2662.

[33] FONG J, WANG X, QI Y X, et al. ParaSplit: a scalable architecture on FPGA for terabit packet classification [C]// Proceedings of IEEE 20th Annual Symposium on High-Performance Interconnects, 2012: 1 - 8.

[34] LIANG E, ZHU H, JIN X, et al. Neural packet classification [C]//Proceedings of the ACM Special Interest Group on Data Communication, 2019: 256 - 269.

[35] YANG B H, FONG J, JIANG W R, et al. Practical multituple packet classification using dynamic discrete bit selection [J]. IEEE Transactions on Computers, 2014, 63(2): 424 - 434.

[36] HSIEH C L, WENG N. Many-field packet classification for software-defined networking switches [C]//Proceedings of ACM/IEEE Symposium on Architectures for Networking and Communications Systems, 2016: 13 - 24.

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

[38] YINGCHAREONTHAWORNCHAI S, DALY J, LIU A X, et al. A sorted-partitioning approach to fast and scalable dynamic packet classification [J]. IEEE/ACM Transactions on Networking, 2018, 26(4): 1907 - 1920.

[39] LI W J, YANG T, ROTTENSTREICH O, et al. Tuple space assisted packet classification with high performance on both search and update [J]. IEEE Journal on Selected Areas in Communications, 2020, 38(7): 1555 - 1569.

[40] JIANG W R, PRASANNA V K. Scalable packet classification on FPGA [J]. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2012, 20(9): 1668 - 1680.

[41] XIN Y, LI W J, WANG Y, et al. An FPGA-based high-throughput packet classification architecture supporting dynamic updates for large-scale rule sets [C]//Proceedings of IEEE Conference on Computer Communications Workshops, 2021: 1 - 2.