软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2012,23(9):2261−2272 [doi: 10.3724/SP.J.1001.2012.04098]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

正则表达式分组的 1/(1−1/*k*)-近似算法*

柳厅文[1,2,3+], 孙 永[1,3], 卜东波[1], 郭 莉[1,3], 方滨兴[1,3]

[1](中国科学院 计算技术研究所,北京 100190)

[2](中国科学院 研究生院,北京 100049)

[3](信息内容安全技术国家工程实验室,北京 100190)

1/(1−1/*k*)-Optimal Algorithm for Regular Expression Grouping

LIU Ting-Wen[1,2,3+], SUN Yong[1,3], BU Dong-Bo[1], GUO Li[1,3], FANG Bin-Xing[1,3]

[1](Institute of Computing Technology, The Chinese Academy of Sciences, Beijing 100190, China)

[2](Graduate University, The Chinese Academy of Sciences, Beijing 100049, China)

[3](National Engineering Laboratory for Information Security Technologies, Beijing 100190, China)

+ Corresponding author: E-mail: liutingwen@nelmail.iie.ac.cn

Liu TW, Sun Y, Bu DB, Guo L, Fang BX. 1/(1-1/*k*)-Optimal algorithm for regular expression grouping. *Journal of Software*, 2012,23(9):2261−2272 (in Chinese). http://www.jos.org.cn/1000- 9825/4098.htm

Abstract: Dividing regular expression sets into multiple groups is an important process to solve the problem of DFA state explosion. Previous grouping algorithms are heuristic or are done by brute-force, which have poor grouping results. This paper analyzes the reasons of states explosion and summarizes conflicting relationship among regular expressions of some types. When conflicts are non-negative and independent, the optimum *k*-grouping problem of regular expression sets can be reduced to the maximum *k*-cut problem, which is NP-hard. Based on the idea of local searching, a new grouping algorithm named GRELS is introduced to solve the problem efficiently, which is 1/(1−1/*k*) -approximation for maximum *k*-cut problem. Comparing with previous grouping algorithms, GRELS has the minimum number of states for the same number of groups, and requires the least time to update grouping results when pattern sets change.

Key words: regular expression; deep packet inspection; grouping algorithm; local searching; 1/(1−1/*k*)-approximation

摘 要: 对正则表达式集合进行分组是解决 DFA 状态膨胀问题的一种重要方法.已有的分组算法大都是启发式的或蛮力的,分组效果很差.分析了 DFA 状态膨胀的原因,总结了某些正则表达式间的冲突状况.证明了当冲突非负和冲突独立时,正则表达式集合的最优 *k* 分组问题可归结为最大 *k* 割问题,从而说明该问题是 NP-Hard 的.基于局部搜索的思想,提出了一种分组算法 GRELS 来解决分组问题,并证明对最大 *k* 割问题,该算法的近似比是 1/(1−1/*k*).与已有的分组算法相比,当分组数目相同时,GRELS 算法分组结果的状态总数最少,并且集合发生变化时所需的更新时间最短.

* 基金项目: 国家自然科学基金(61070026); 国家重点基础研究发展计划(973)(2007CB311100); 国家高技术研究发展计划(863)(2011AA010703); 中国科学院战略性先导科技专项(XDA06030200)

收稿时间: 2010-09-17; 修改时间: 2011-04-28; 定稿时间: 2011-07-21

关键词: 正则表达式;深度包检测;分组算法;局部搜索;1/(1−1/k)近似
中图法分类号: TP301 文献标识码: A

正则表达式匹配是计算机科学与技术领域的一个经典问题,目前已有很多的文献从算法和理论方面对正则表达式匹配做了大量的研究工作[1−3].正则表达式匹配最初应用于语言的词法分析、文本检索和计算生物学等领域,近年来也被应用于网络安全检测等领域.面向网络安全的正则表达式匹配具有不同的应用特点:大规模的正则表达式集合、高速到达的待检测数据流、只关心是否有正则表达式匹配而不必找到其所有匹配子串等.

1 绪 论

越来越多的网络安全应用和服务在对进出链路的数据包的包头进行检测的同时,还需要深入到数据包的负载进行检测.深度包检测(deep packet inspection,简称 DPI)是一种强大且重要的技术,它将数据包携带的负载内容与预先设定的模式集(表示入侵特征、病毒特征等)进行匹配,从而确定数据包中是否有非法内容出现.目前,深度包检测技术被广泛地应用在各种网络应用和网络服务中,如网络入侵检测、防火墙、垃圾邮件过滤、协议识别、内容计费等.

传统的深度包检测技术采用精确字符串来表示模式,但随着网络的发展,入侵特征等变得越来越复杂,因此,表达能力更强的正则表达式(regular expression,简称RE)逐渐取代精确字符串作为表示模式的工具.例如:Linux下的开源协议识别引擎L7-Filter** 全部采用正则表达式描述应用层协议特征;著名的开源网络入侵检测系统Snort*** 的模式库从 2003 年 4 月的 0 条正则表达式增长到 2006 年 2 月的 1 131 条,截止 2009 年 6 月已增长到 2 971 条.

经典的正则表达式匹配技术是将其转换为表达能力等价的非确定型有限自动机(non-deterministic finite automaton,简称 NFA)或确定型有限自动机(deterministic finite automaton,简称 DFA)来进行模式匹配.DFA 具有两个显著优点:

(1) 在匹配过程的任意时刻,DFA 只有一个活跃状态,处理每个待匹配字符只需访问一次状态转移表;

(2) 可以为多条正则表达式生成一个合并的 DFA,从而在一遍扫描过程中完成对所有正则表达式的匹配.

与 DFA 不同,NFA 在最坏情况下处理每个字符需要 $O(n)$次访问状态表(n 是 NFA 的状态数目).因此在高速的网络环境下,大部分应用都采用基于 DFA 的匹配技术完成实时深度包检测.然而,DFA 存在着状态膨胀的问题(某些情况下,DFA 状态数目甚至呈指数增长),导致 DFA 可能消耗巨大的存储空间甚至无法生成.例如,L7-Filter 的正则表达式集合(107 条)生成的合并 DFA 需要占用超过 16GB 的内存,导致在普通机器上无法使用.因此,如何减少 DFA 的巨大空间消耗,是实现高速正则表达式匹配的重要问题.

本文从将正则表达式集合分组的角度出发考,虑如何减少 DFA 的状态数目.主要贡献有:

- 问题难度:对正则表达式集合的分组问题进行了形式化描述,当表达式间的状态冲突是非负和独立时,正则表达式集合的最优 k 分组问题可归结为最大 k 割问题,从而证明了该问题是 NP-困难的;
- 实用算法:提出了一种基于局部搜索策略的实用分组算法 GRELS(grouping regular expression with local searching),并从理论上证明了对最大 k 割问题该算法的近似比是 $1/(1-1/k)$;
- 分组评价:实现了 GRELS 算法和其他 4 种分组算法,并从 DFA 状态总数、分组数目和分组更新时间 3 个指标对它们进行比较,最后,对不同算法的优缺点进行了总结.实验结果表明,尽管冲突非负和冲突独立的条件并不总能严格满足,GRELS 算法的分组结果仍明显优于已有分组算法,具有很好的实用性.

** Application Layer Packet Classifier for Linux. http://l7-filter.sourceforge.net/

*** SNORT Network Intrusion Detection System. http://www.snort.org

2 相关工作

随着正则表达式在网络安全系统和服务中的广泛应用,近年来的研究多集中在大规模正则表达式集合下如何减少DFA的存储空间开销.DFA的存储空间开销主要来自其状态转移表,表的行宽和列宽分别对应DFA的状态数目和每个状态的转移边数目|Σ|(Σ是输入字符集).目前,解决DFA存储开销过大问题的思路可以分为减少状态数目[4−6]和压缩转移边两种,通过对正则表达式集合进行分组来压缩空间属于第一种解决思路.

Yu等人分析了Snort等系统中真实的正则表达式,并按照其DFA状态数目的增长复杂度不同将其分作 5 类[7].作者进一步对多项式增长和指数增长的两类正则表达式提出了改写规则,使得其DFA状态数目大大减少.然而,改写规则只能这两类中某些特殊的正则表达式有效,无法解决其他的情况,因而不具有通用性.文献[7]还指出,为多个增长复杂度为线性的正则表达式(如".*string1.*string2")生成合并DFA时,其状态数目可能呈指数增长.基于此,作者首先提出了正则表达式集合的分组问题,并给出了一种启发式的分组算法Fang Yu(以作者姓名命名):获得任意两表达式间是否冲突后,不断将与当前分组中表达式间冲突最少的未分组的表达式加入到当前分组,直到该分组DFA的状态数目超过某个阈值.该分组算法没有充分地利用表达式间的状态冲突信息,而且分组策略过于简单.GRELS分组算法在解决思路上与其类似,但充分利用了冲突信息并采取了更有效的策略.

徐乾等人提出一种基于DFA的正则表达式压缩算法[8].该算法首先定义了膨胀率来描述正则表达式的膨胀特性,并提出了一种分片的算法RECCADR,通过有效地选择并隔离导致DFA状态膨胀的片段,降低了单个正则表达式存储需求.同时,基于正则表达式间的组合关系提出一种选择性分组算法REGADR,在可接受的存储需求总量下,通过选择性分群大幅度减少了状态机的个数,有效地降低了匹配算法的复杂性.但对正则表达式分片后再进行匹配并得到的匹配结果存在假阳性(false positive),因此该算法的匹配结果需要进一步地进行验证.

Becchi等人提出了一种基于状态合并的压缩算法[9],并给出了该算法的最差性能保证.该算法的主要思想是,通过巧妙地标记转移边,使得大量非等价的状态能够被合并到一起,极大地减少DFA的状态数目.在文献[10]中,Becchi等人引入了一种当合并的DFA状态数目过大时,如何对模式集进行自动分组的策略.该策略设置了一个阈值σ,然后循环地二分模式集,直到每个分组DFA的状态数目都小于阈值σ.

Majumder等人从减少Cache Miss的角度出发考虑对正则表达式集合分组[11]:

Step 1. 定义一个计算分组的处理时间开销的函数;

Step 2. 初始状态时,把每条正则表达式作为一个分组;

Step 3. 如果两个分组合并后能得到处理时间开销最小的分组集合,那么就合并它们.

Rohrer等人将分组问题描述为能量最小化问题,并应用已有的优化算法和启发式方法(如整数线性规划、模拟退火等)来获得好的分组结果[12].

上述几种方法都是通过减少DFA的状态数目来降低DFA的存储空间开销,另外一条研究路线是通过压缩转移边来减少存储空间开销[13−17].例如,文献[13]通过将DFA的某些状态的一组边用单条缺省边来代替,提出一种称为延迟输入DFA(delay input DFA,简称D^2FA)的自动机,实验结果表明,D^2FA能减少 95%以上的存储空间.但是,D^2FA处理每个输入字符可能需要沿着缺省边多次迁移,从而需要多次访存操作.而且D^2FA等压缩转移边的方法都是对已生成的DFA进行压缩,因此需要事先生成模式集的合并DFA.但通常情况下,对实际系统中的正则表达式集合构造合并DFA是不可行的[11],如前面提到的L7-Filter.

本文主要解决在无法生成合并 DFA 的情况下,如何快速高效地对正则表达式模式集进行分组,从而可以构造出状态总数较少的分组 DFA.分组算法的一个优点是,它和压缩转移边的方法是直交的,因此可以通过压缩各个分组的 DFA 来进一步减少进行正则表达式匹配所需要的存储空间.虽然分组后处理每个字符需要在所有分组的 DFA 上迁移,但现在多核架构、图形处理单元等并行处理设备已得到广泛应用,该时间复杂度可以通过将这些 DFA 在并行处理设备上同时运行来降低.

3 分组前提与分组评价

对正则表达式集合进行分组的目标是减少 DFA 的状态数目,也就是各分组 DFA 的状态数目总和应小于不分组时的合并 DFA 的状态数目.该目标能够实现的前提是正则表达式之间存在冲突,并且不同表达式之间的冲突程度不同.

3.1 正则表达式间的冲突

在Fang Yu所分类的正则表达式中[7],有些表达式生成的DFA的状态数目与正则表达式的长度呈线性关系,它们自身并不会引起状态数目的膨胀.但为它们生成合并的DFA时,对应的合并DFA的状态数目会急剧膨胀,甚至可能呈指数增长.假设有 3 条正则表达式r_1=".*ab.*cd",r_2=".*ef.*gh"和r_3=".*ab.*gh",它们各自的DFA以及合并后的DFA如图 1 所示.

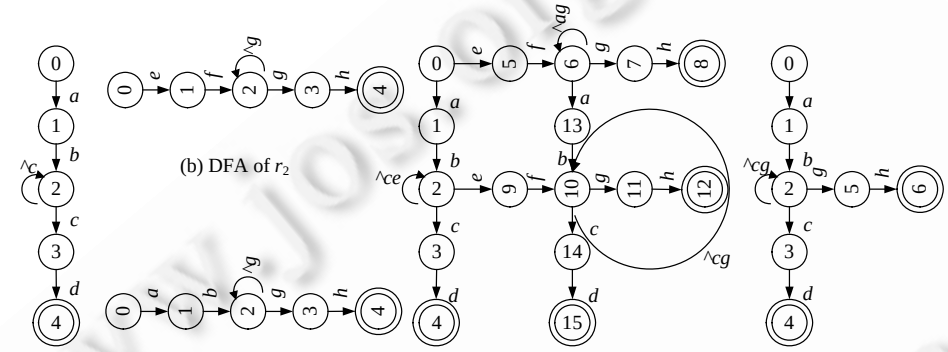


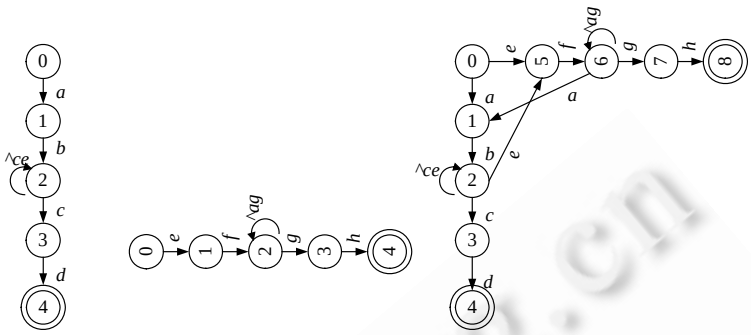
Fig.1 DFA for regular expression r_1, r_2, r_3, $r_1|r_2$, $r_1|r_3$

图 1 正则表达式r_1,r_2,r_3,$r_1|r_2$,$r_1|r_3$对应的DFA

图 1 中,双圆圈括起来的状态表示接受状态.图 1(d)是把图 1(a)和图 1(b)合并到一起生成的DFA.可以很明显地看出:状态 9~状态 15 是合并两个表达式后由于冲突而引入的新状态(以下称冲突状态).这是因为表达式".*ab.*cd"中的".*"可以匹配 0 个或任意多个字符,因此需要引入新状态来判断".*"是否已匹配表达式".*ef.*gh"的前缀".*ef".正是这种模糊性,导致了状态的膨胀.当m个该格式的正则表达式合并到一起时,生成的合并DFA状态数目呈$O(2^m)$增长.

对某些表达式来说,其合并DFA的状态数目膨胀程度较低,可能呈线性增长,如表达式".*ab.*cd"和".*ef".某些表达式的合并甚至会导致状态数目减少,如图 1(e)的状态数目小于图 1(a)和图 1(c)的状态数目之和.有趣的是,如果我们把r_1和r_2稍作修改,得到r_4=".*ab[^e]*cd"和r_5=".*ef[^a]*gh",它们的合并DFA没有发生状态膨胀,如图 2 所示.

从上述分析可以看出,正则表达式间的冲突关系是相对的,即表达式r_i与表达式r_j冲突的同时,表达式r_i和表达式r_j可能不冲突,并且不同正则表达式间的冲突程度是不同的.我们把正则表达式间冲突的原因归结为开始位置模糊性、字符集模糊性、重复次数模糊性和表达式前缀的可现性.不具有模糊性和前缀不可现的正则表达式是不会与任何正则表达式冲突的,如"^string"(带首锚^的精确字符串).因此,对正则表达式集合分组时需要将冲突严重的表达式分配到不同组中,以尽量减少DFA的状态数目,这就需要一个高效的正则表达式分组算法.



(a) DFA of r_4 (b) DFA of r_5 (c) DFA of $r_4|r_5$

Fig.2 DFA for regular expression r_4, r_5, $r_4|r_5$

图 2 正则表达式r_4,r_5,$r_4|r_5$对应的DFA

3.2 分组算法的评价指标

如上所述,基于 DFA 进行正则表达式匹配时,一种策略是为 n 条表达式生成一个合并的 DFA,另一种策略是为每条表达式生成一个 DFA,自动机的个数为 n.前一种策略处理一个字符只需访问一次状态转移表,但通常合并 DFA 的状态数目会剧烈膨胀;后一种策略的 DFA 状态总数比较少,但处理字符时需要在每个 DFA 中都进行一次状态转移,因而需要 n 次访问状态转移表.

对正则表达式集合的分组,本质上是这两种策略的折衷,即将 n 条表达式分到 k 个分组里($1\leqslant k\leqslant n$),此时,处理一个字符仅需 k 次访问状态转移表,并且 k 个分组 DFA 的状态总数远远小于不分组时的一个合并 DFA.

因此,评价正则表达式集合的分组结果常用的指标有 DFA 的状态总数和分组数目两个:

• DFA 状态总数:该指标主要从减少 DFA 存储空间的角度考虑.DFA 的状态数目是决定 DFA 占用存储空间大小的关键因素,DFA 的状态数目越少,它所占用的存储空间就越小,这样才有可能在较小的存储空间下生成和使用 DFA;

• 分组数目:该指标主要从匹配速度的角度考虑,分组数目决定了处理每个字符时状态转移表的访问次数,最终影响正则表达式的匹配速度.

显然,这两个指标是相互矛盾的.一般情况下,分组数目越大,DFA 的状态总数越小;分组数目越小,DFA 的状态总数越大.

另外,本文引入了一个对分组问题的新的合理评价指标,即分组更新时间.随着互联网的发展,网络应用、攻击手段、病毒变种等也在不断发展,因此,深度包检测系统中的正则表达式集合也随着在不断变化(需要增加、删除或更新正则表达式).因此,理想的分组算法应当能够尽量减少正则表达式集合变化时,更新分组结果所带来的时间开销.

4 正则表达式集合的分组

4.1 分组问题的形式化描述

对规模为n的正则表达式集合R中的任意两条表达式r_i和r_j,它们各自的DFA状态数目记为S_i和S_j,合并生成的DFA状态数目记为S_{ij}.M_{ij}表示r_i和r_j间冲突状态的规模,且定义$M_{ij}=S_{ij}-S_i-S_j$.显然,二维矩阵M存储的是相应行列的正则表达式由于合并导致DFA状态增长的数目.

为了简化对正则表达式集合分组问题的分析和求解,我们做出如下两个假设:

(1) 冲突非负:任意两条正则表达式间至少有 0 个冲突状态,即$M_{ij}\geqslant 0$;

(2) 冲突独立:正则表达式间的冲突状态不会引入新的冲突状态,即如果分组R_m($1\leqslant m\leqslant k$)中增加了一条

新表达式$r_j(r_j \notin R_m)$,那么导致分组R_m增加的真实冲突状态数目$T(R_m, j) = \sum_{r_i \in R_m} M_{ij}$.

由前文所述可知,具有公共前缀的两条表达式合并生成的DFA可能引起状态数目减少,多条含".*"的表达式合并会导致状态数目呈指数增长,因此这两个假设并不总是成立.若冲突非负假设不成立,即$M_{ij}<0$,我们置$M_{ij}=0$;若冲突独立假设不成立,我们将增加冲突状态的估计数目$\sum_{r_i \in R_m} M_{ij}$赋值给增加冲突状态的真实值$T(R_m,j)$,作为对新增冲突状态数目的估计.这样,冲突非负和冲突独立的假设都成立了.通过实验部分可以看到,我们的修正并不对最终分组结果有太大影响.

记$T_1(R_m)$为分组R_m中各表达式的DFA的状态数目之和,$T_2(R_m)$为分组R_m因合并生成分组DFA所增加的状态数目.当冲突非负和冲突独立时,分组R_m的状态总数:

$$T(R_m) = T_1(R_m) + T_2(R_m) = \sum_{r_i \in R_m} S_i + \sum_{r_i, r_j \in R_m : i<j} M_{ij} \tag{1}$$

正则表达式集合的分组,使得 R 中的每条表达式都出现且仅出现在一个分组中.因此,将 R 分为 k 组并为每个分组生成一个合并 DFA,其状态总数:

$$\begin{aligned} C_k(R) &= \sum_{m=1}^{k} T(R_m) \\ &= \sum_{m=1}^{k} T_1(R_m) + \sum_{m=1}^{k} T_2(R_m) \\ &= \sum_{i=1}^{n} S_i + \sum_{m=1}^{k} \sum_{r_i, r_j \in R_m : i<j} M_{ij} \\ &= C_n(R) + \sum_{m=1}^{k} \sum_{r_i, r_j \in R_m : i<j} M_{ij} \end{aligned} \tag{2}$$

对给定的规模为n的正则表达式集合R来说,$C_n(R)$为n条表达式各自生成的DFA的状态数目总和,即对模式集R进行n分组时DFA的状态总数,该值是固定的.因此,最小化k分组时,DFA的状态总数等价于最小化k分组比n分组增加的状态数目,即

$$\text{Min } E_k(ingroup) = \sum_{m=1}^{k} \sum_{r_i, r_j \in R_m : i<j} M_{ij} \tag{3}$$

4.2 最优k分组问题的难度分析

定理 1. 当冲突非负和冲突独立时,正则表达式集合的最优k分组问题是一个最小边缺失的k分割问题[****].

证明:最小边缺失的 k 分割问题描述为:给定一个无向图 $G=(V,E)$和一个权值函数 $w:E\to N$,求得一个 k 分割,如对无向图 G 中的每个顶点染色 $c:V\to\{1,2,...,k\}$,使得具有两个相同颜色顶点的边的权值总和最小,即

$$\text{Min} \sum_{(v_1,v_2)\in E: c(v_1)=c(v_2)} w(v_1, v_2).$$

对给定的规模为n的正则表达式集合R,我们可以构造一个无向图$G=(V,E)$,$|V|=n$:顶点i对应正则表达式r_i;若r_i和r_j冲突,则在顶点i和j间引入一条权值为M_{ij}的无向边.对正则表达式集合R进行k分组后,相应的顶点集V也被分为k个子集.如果同一子集内的顶点涂上相同颜色,不同子集间的顶点颜色不同,那么边集E两个顶点在同一子集内的边则具有相同的顶点颜色.与n分组R相比,该分组结果增加的状态数目等于具有相同顶点颜的边的权值总和,即 $E_k(ingroup) = \sum_{m=1}^{k} \sum_{r_i, r_j \in R_m : i<j} M_{ij} = \sum_{(v_1,v_2)\in E: c(v_1)=c(v_2)} w(v_1, v_2).$ □

这样,我们就把正则表达式集合的最优 k 分组问题转换为图的最小边缺失的 k 分割问题.解决了任何一个问题,相应地就能得到另外一个问题的答案.

[****] Minimum Edge Deletion k-Partition Problem. http://www.ensta.fr/~diam/ro/online/viggo_wwwcompendium/node43.html#GT33

定理 2. 当冲突非负和冲突独立时,正则表达式集合的最优 k 分组问题等价于最大 k 割问题(maximum k-cut problem).

证明:最大多割问题是图论中关于图分割的一个经典问题,它可以描述为:给定一个无向图$G=(V,E)$、一个权值函数$w:E\to N$和一个正整数$k\in[2,|V|]$,将V分为k个不相交的子集$F=\{C_1,C_2,\ldots,C_k\}$,使得两个顶点属于不同子集的边的权值总和最大,即 $\mathrm{Max}\sum_{m=1}^{k-1}\sum_{t=m+1}^{k}\sum_{v_1\in C_m,v_2\in C_t}w(v_1,v_2)$.

相应地,按照定理 1 中所示方法对给定的规模为 n 的正则表达式集合 R 构造无向图 $G=(V,E)$,该无向图的所有边的权值之和 $E(total)$等于将 R 分为 1 组(即不分组)时比分为 n 时组增加的 DFA 状态数目.该值为一个常数,并且 $E(total)=E_k(ingourp)+E_k(crossgroup)=\sum_{m=1}^{k}\sum_{r_i,r_j\in R_m:i<j}M_{ij}+\sum_{m=1}^{k-1}\sum_{t=m+1}^{k}\sum_{r_i\in R_m,r_j\in G_t}M_{ij}$.

$E_k(crossgroup)$是k分割无向图G后的割边的权值总和,因此,求$E_k(ingroup)$的最小值就等价于求$E_k(crossgroup)$的最大值.

所以,正则表达式集合的最优 k 分组问题等价于最大 k 割问题. □

无向图的最大 k 割问题是最大割问题的扩展.当 k=2 时,最大 k 割问题就是最大割问题.最大 k 割问题是 NP-Hard 的,因此正则表达式集合的最优 k 分组问题也是 NP-hard 的.

4.3 GRELS算法

既然正则表达式集合的最优 k 分组问题是 NP-Hard 的,所以在线性时间内无法获得其最优分组结果.本文基于局部搜索的思想,提出了一种新的分组算法 GRELS,通过近似地解决最大 k 割问题来获得较好的分组结果.

4.3.1 算法伪代码

算法输入:规模为 n 的正则表达式集合 R、待分组个数 k、表达式间冲突矩阵 $M[n][n]$;

算法输出:集合R的k个分组$\{R_1,R_2,\ldots,R_k\}(1\leqslant m\leqslant k)$.

```
1   声明布尔型变量 nochange,初始值为 true;
2   构造k个分组{R1,R2,...,Rk}(1≤m≤k),并指其初始值为NULL;
3   随机地分配每条表达式到一个分组;
4   while (true) {
5      赋值 nochange 为 true;
6      for (i=1; i≤n; i++) {
7         for (m=1; m≤k; m++)   F[m] = Σ_{j∈Rm:j≠i} Mij
8         t 是数组 F 中最小值所在位置的索引;   //F[t]在数组 F 的所有元素中最小
9         如果正则表达式ri不在分组Rm中,将ri从原分组移到Rm中,并赋值nochange为false;
10        }   //end of for (i=1; i≤n; i++)
11     如果 i 大于 n 并且 nochange 为 true,跳出 while 循环;
12  }   //end of while (true)
```

该算法首先随机地将每条正则表达式不重复地分配到一个分组,对当前的分组结果,判断是否存在一条表达式,当把它从目前所在分组移动到另一个分组时会减少冲突状态数目.如果存在,则将其移动至减少冲突状态数目最多的分组中,然后开始下一次判断;如果不存在,那么移动任何表达式都不会带来状态数目减少,表明此时的分组结果经是局部最优的.

4.3.2 算法分析

- 正确性分析

从两个方面来说明 GRELS 算法的正确性:1) GRELS 算法能得到正确的分组结果,即能保证每条正则表达式都会出现在唯一的分组里;2) GRELS 算法能够终止.每次 while 循环的执行都会减少冲突状态的数目,而冲突

状态的最大值为 $E(total)$,因此,该算法最多执行 $E(total)$次 while 循环便会结束.该算法的时间复杂度为 $O(E(total)nk)$.通常情况下,实际的执行步数要远远小于该值.

- 近似比分析

定理 3. 对最大 k 割问题,GRELS 算法的近似比是 $1/(1-1/k)$.

证明:假设$\{R_1,R_2,...,R_k\}$是GRELS算法得到集合R的k分组结果,$\{R_1^*,R_2^*,...,R_k^*\}$是对集合$R$进行$k$分组的最优结果(同时也是对应最大$k$割问题实例的最优结果).从GRELS算法分组结果的局部最优性可知,对$\forall i\in R_m$, $\sum_{j\in R_m} M_{ij} \leqslant \sum_{j\in R_t,t\neq m} M_{ij}$,可得:

$$\sum_{i\in R_m,j\in R_m} M_{ij} \leqslant \sum_{i\in R_m,j\in R_t,t\neq m} M_{ij},(k-1)\sum_{i\in R_m,j\in R_m} M_{ij} \leqslant \sum_{t=1}^{k}\sum_{i\in R_m,j\in R_t,t\neq m} M_{ij},$$

从而有

$$\begin{aligned}\sum_{m=1}^{k-1}\sum_{t=m+1}^{k}\sum_{i\in R_m^*,j\in R_t^*} M_{ij} &\leqslant \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} M_{ij}\\ &=\frac{1}{2}\sum_{m=1}^{k}\sum_{i\in R_m,j\in R_m} M_{ij}+\sum_{m=1}^{k-1}\sum_{t=m+1}^{k}\sum_{i\in R_m,j\in R_t} M_{ij}\\ &\leqslant \frac{1}{2}\sum_{m=1}^{k}\frac{1}{k-1}\sum_{t=1}^{k}\sum_{i\in R_m,j\in R_t,t\neq m} M_{ij}+\sum_{m=1}^{k-1}\sum_{t=m+1}^{k}\sum_{i\in R_m,j\in R_t} M_{ij}\\ &=\sum_{m=1}^{k-1}\sum_{t=m+1}^{k}\sum_{i\in R_m,j\in R_t}\frac{1}{k-1}M_{ij}+\sum_{m=1}^{k-1}\sum_{t=m+1}^{k}\sum_{i\in R_m,j\in R_t} M_{ij}\\ &=\frac{k}{k-1}\sum_{m=1}^{k-1}\sum_{t=m+1}^{k}\sum_{i\in R_m,j\in R_t} M_{ij}.\end{aligned}$$

□

由上述证明可知,最大 k 割问题的最优结果不过超过 GRELS 算法结果的 $\frac{k}{k-1}$ 倍,因此对最大 k 割问题,该算法的近似比是 $1/(1-1/k)$.当冲突非负和冲突独立时,正则表达式集合的最优k分组问题对应的最大k割问题实例也能通过GRELS算法得到不大于该近似比的结果.目前,基于半正定性规划松弛模型[18]能得到对最大k割问题的最好近似结果,其近似比为 $1/(1-1/k+2\ln k/k^2)$,但该方法复杂度较高,无法用于正则表达式集合的分组问题,因而只具有理论价值.

5 实验结果

本文的实验以 Becchi 开源的正则表达式匹配引擎(Becchi Michela. Regular expression processor. http://regex.wustl.edu/)为工具生成 DFA,实验环境为 Intel Core 2 Duo 2.4 GHZ,2 GB 内存.为了使实验结果更具有可比性,我们从 L7-Filter,Snort 和 Bro(Bro intrusion detection system. http://bro-ids.org/Overview.html)这 3 个开源系统中提取 4 个有代表性的真实正则表达式集合作为实验模式集.

5.1 假设验证

5.1.1 冲突非负假设验证

各实验模式集中表达式间的冲突统计信息见表 1.

Table 1 Statistics of conflicts for regular expression pairs in pattern sets

表 1 各模式集表达式间的冲突统计信息

Pattern set	# of regular expressions	Max negative conflicts	Avg. negative conflicts	Max positive conflicts	Avg. positive conflicts
L7-Filter	107	19	1.88	99 966	281
Snort-Web	98	52	1.59	99 874	1 693
Snort-Backdoor	158	17	1.79	1 038	48.88
Bro831	831	85	1.05	213	11.08

表 1 中第 2 列为模式集的规模,第 3 列是各模式集中任意两条表达式合并生成 DFA 导致状态减少的最大值,第 4 列是状态减少的平均值,第 5 列是状态增加的最大值,第 6 列是状态增加的平均值.可以发现:1) 不论是最大值还是平均值,两条正则表达式合并引起状态增加的程度远远大于状态减少的程度.这说明冲突非负的假设是合理的,对负冲突的修正并不会对分组结果有大的影响;2) L7-Filter 模式集规模适中,并且模式比较复杂,相互之间冲突比较大;Snort-Web 模式集规模是最小的,但模式却是最复杂的,各模式之间平均引入 1 693 个状态;Snort-Backdoor 模式集规模适中,但表达式相对简单;Bro831 模式集规模是最大的,有 831 条正则表达式,但其中精确字符串居多,是 4 个模式集中最简单的.这 4 个模式集在规模和复杂度上是很有代表性的.

5.1.2 冲突独立假设验证

为了验证冲突独立假设,我们比较 DFA 的真实冲突数目和在该假设下的估计冲突数目之间的关系.

对规模为n的模式集,真实冲突数目为不分组时的合并DFA状态数目与分n组时各DFA状态数目总和之差,估计冲突数目为不分组时正则表达式间的冲突数目之和,即$E_1(ingroup)$.由于我们的实验模式集在不分组时其合并DFA的状态数目超过 100 万,因此我们从中随机抽取若干条表达式构成验证该假设的模式集.初始状态置验证模式集为空,不断往其中加入正则表达式,从而观察真实冲突数目和估计冲突数目随验证模式集规模的变化趋势.实验结果如图 3 所示,从该图中我们可以发现:冲突独立的假设并不严格成立,通常情况下,估计冲突数目小于真实冲突数目,但两者保持着同样的增长趋势,从而使得GRELS算法能够在两种情况下做出基本相同的分组选择.因此,GRELS算法使用估计冲突数目进行分组,这样能够避免通过尝试生成DFA获得分组的真实冲突数目,从而减少生成分组所带来的时间开销.

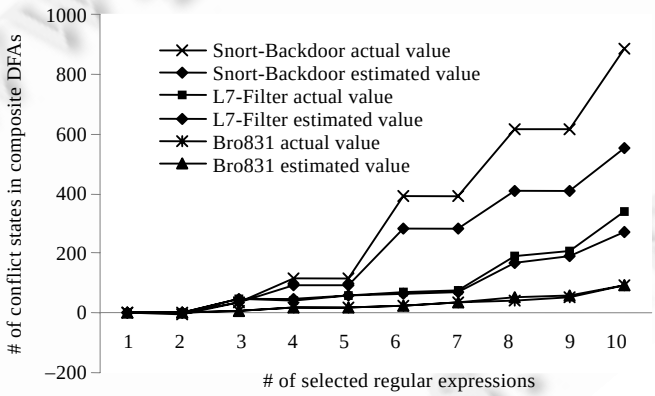


Fig.3 Comparison of actual number and estimated number of conflict states in composite DFAs

图 3 合并 DFA 冲突状态的真实数值和估计数值的比较

5.2 算法分组结果评价

在前人工作的基础上,本文实现了REGADR[8],Fang Yu[7],Random(将模式集中的表达式随机分组),Becchi[10]和GRELS这 5 种分组算法,并分别在 4 个实验模式集上比较了状态总数(state)、分组个数(k)和分组更新时间(time)这3个指标.本实验中,分组更新时间以模式集R中新增1条表达式r_i为例,其值为分组算法从已有的模式集R的分组结果获得$R'=R\cup\{r_i\}$的分组结果的时间.这里,我们提出两种策略来更新分组:1) 直接将r_i加入到状态数目最少的分组中,若此时该分组DFA状态数过大,则将r_i放在一个新分组中;2) 利用分组算法更新分组结果.第 1 种策略更新时间较短,且与分组算法无关,但更新后的分组可能效果很差;第 2 种策略能得到较好的分组,但需要的时间较长.因此,可以利用第 1 种策略快速更新分组,待第 2 种策略离线处理完后替换第 1 种策略的分组结果.离线处理时,我们可以认为各分组算法关于模式集R'的一些需要信息已经保存,如REGADR算法的膨胀率和GRELS算法的冲突矩阵.因此,GRLES算法的更新过程为:计算r_i与R中各表达式间的冲突关系,然后调用GRELS算法获得R'的分组.实验结果见表 2~表 5,各表中的时间为采用第 2 种策略时的分组更新时间.

从表 2~表 5 所示结果可以看出:

1) REGADR 分组算法所得分组结果的 DFA 状态总数很大,并且分组更新时间较长.该算法有 3 个参数调节分组结果,且参数都依赖于具体模式集,因此对不同的模式集来说,如果想获得较好的分组结果需要不断调整参数.该算法的优点是不需要事先获得模式集中表达式两两之间的冲突信息;

2) Fang Yu 分组算法优于 REGADR 算法,该算法的输入参数是某个阈值σ,用来约束分组 DFA 状态数的上限;

3) Becchi 分组算法通过不断地二分模式集来进行分组,分组的结果依赖于模式集中的表达式输入顺序.该算法的分组效果很差,并且分组更新时间是最长的.该算法也不需要事先获得表达式间的冲突关系,它的参数输入与 Fang Yu 算法一样;

4) Random 分组算法随机地分组正则表达式,没有任何措施避免将冲突较严重的表达式分到不同组里,因而分组效果一般是最差的.该算法的分组更新时间较短,且不需要事先获得冲突信息,其参数输入是待分组的个数 k;

5) 当分组数目相同时,GRELS 算法是这 5 种算法中 DFA 状态总数最少的一种算法,并且分组更新时间最短.这主要是因为其他算法需要通过生成 DFA 来不断地尝试分组,而我们的算法是通过计算冲突状态数目的估计值,从而避免了相应时间开销.GRELS 算法在 4 个具有代表性的实验模式集上得到的结论是一致的,这说明我们的算法是通用的,即使冲突独立和冲突非负不成立,GRELS 算法仍能快速高效地对正则表达式集合分组.

Table 2 Grouping results for L7-Filter pattern set

表 2 L7-Filter 模式集的分组结果

Grouping algorithm	k=6		k=7		k=8		k=9	
	State	Time (s)	State	Time (s)	State	Time (s)	State	Time (s)
REGADR	160 871	11 430	94 586	7 934	81 191	15 300	48 032	3 777
Fang Yu	33 628	10 083	29 047	8 824	22 591	5 960	16 877	4 902
Becchi	529 327	11 462	314 425	10 337	192 530	8 129	115 217	5 185
Random	—	—	122 051	3 213	192 530	2 475	72 848	678
GRELS	23 808	638	13 537	401	11 994	410	10 993	175

Table 3 Grouping results for Snort-Web pattern set

表 3 Snort-Web 模式集的分组结果

Grouping algorithm	k=14		k=15		k=16		k=17	
	State	Time (s)	State	Time (s)	State	Time (s)	State	Time (s)
REGADR	44 411	7 245	35 715	5 383	29 844	3 924	21 698	4 657
Fang Yu	32 483	6 689	20 849	6 369	17 172	5 811	15 609	6 056
Becchi	304 006	31 382	181 769	21 210	109 909	10 422	83 486	5 227
Random	—	—	552 039	5 340	877 739	13 146	781 799	8 986
GRELS	33 873	3 995	19 771	4 897	20 198	4 442	10 292	3 954

Table 4 Grouping results for Snort-Backdoor pattern set

表 4 Snort-Backdoor 模式集的分组结果

Grouping algorithm	k=4		k=5		k=6		k=7	
	State	Time (s)	State	Time (s)	State	Time (s)	State	Time (s)
REGADR	48 401	18 902	31 453	12 837	25 080	9 620	19 832	7 957
Fang Yu	14 371	2 095	13 637	1 833	8 420	711	6 164	1 758
Becchi	65 497	1 953	43 695	955	32 316	823	17 040	734
Random	131 748	3 125	114 711	1 239	69 386	621	56 639	638
GRELS	11 040	398	7 677	223	5 736	177	4 672	139

Table 5 Grouping results for Bro831 pattern set

表 5 Bro831 模式集的分组结果

Grouping algorithm	k=8		k=9		k=10		k=11	
	State	Time (s)	State	Time (s)	State	Time (s)	State	Time (s)
REGADR	71 050	20 429	56 503	11 115	45 205	7 432	43 593	6 558
Fang Yu	15 009	14 426	15 106	11 979	13 027	7 621	12 116	6 291
Becchi	244 839	14 061	212 575	10 903	124 440	7 573	93 487	3 765
Random	199 866	1 874	222 171	1 647	172 169	1 281	121 634	858
GRELS	11 879	121	12 085	123	10 668	110	9 417	94

6 结束语

网络带宽的爆炸增长和网络应用的不断丰富,给实时深度包检测带来巨大压力.传统的基于 NFA 的正则表达式匹配技术已经不能够满足网络安全应用和服务的实时检测要求,如入侵检测、协议时别、垃圾邮件过滤等.因此,相关的研究热点已经转向匹配速度更快的基于 DFA 的正则表达式匹配技术.然而 DFA 存在着状态数目膨胀的问题,在某些情况下甚至呈指数增长.本文通过分析正则表达式间的相互冲突情况,发现不同的表达式相互之间的冲突程度是不相同的,因此可以通过合理地将正则表达式集合分组来降低 DFA 的状态总数,并给出了 3 个合理的指标来对正则表达式集合的分组结果进行评价.本文对正则表达式集合的最优 k 分组问题进行了公式化描述,并证明当冲突非负和冲突独立时可转化为 NP-Hard 的最大 k 割问题.基于此,本文提出了一种近优的 GRELS 分组算法,并证明了对最大 k 割问题该算法的近似比是 $1/(1-1/k)$.实验结果表明,即使冲突非负和冲突独立不严格成立,GRELS 分组算法可能得不到满足近似比的结果,但它仍然比已有的分组算法更新速度快,而且分组的结果更好.

References:

[1] Thompson K. Programming techniques: Regular expression search algorithm. Communications of the ACM, 1968,11(6):419−422. [doi: 10.1145/363347.363387]

[2] Myers G. A four Russians algorithm for regular expression pattern matching. Journal of the ACM, 1992,39(2):432−448. [doi: 10.1145/128749.128755]

[3] Hopcroft JE, Motwani R, Ullman JD. Introduction to Automata Theory, Languages, and Computation. 2nd ed., Boston: Addison Wesley, 2000. 1−50.

[4] Kumar S, Chandrasekaran B, Turner JS, Varghese G. Curing regular expressions matching from insomnia, amnesia, and acalculia. In: Proc. of the ANCS 2007. New York: ACM Press, 2007. 155−164. [doi: 10.1145/1323548.1323574]

[5] Becchi M, Crowley P. A hybrid finite automaton for practical deep packet inspection. In: Proc. of the CoNEXT 2007. New York: ACM Press, 2007. [doi: 10.1145/1364654.1364656]

[6] Smith R, Estan C, Jha S. XFA: Faster signature matching with extended automata. In: Proc. of the S&P 2008. New York: IEEE Inc., 2008. 187−201. [doi: 10.1109/SP.2008.14]

[7] Yu F, Chen ZF, Diao YL, Lakshman TV, Katz RH. Fast and memory-efficient regular expression matching for deep packet inspection. In: Proc. of the ANCS 2006. New York: ACM Press, 2006. 93−102. [doi: 10.1145/1185347.1185360]

[8] Xu Q, E YP, Ge JG, Qian HL. Efficient regular expression compression algorithm for deep packet inspection. Journal of Software, 2009,20(8):2214−2226 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/3311.htm [doi: 10.3724/SP.J.1001.2009.03311]

[9] Becchi M, Cadambi S. Memory-Efficient regular expression search using state merging. In: Proc. of the INFOCOM 2007. Piscataway: IEEE Inc., 2007. 1064−1072. [doi: 10.1109/INFCOM.2007.128]

[10] Becchi M, Franklin M, Crowley P. A workload for evaluating deep packet inspection architectures. In: Proc. of the IISWC 2008. Piscataway: IEEE Computer Society, 2008. 79−89. [doi: 10.1109/IISWC.2008.4636093]

[11] Majumder A, Rastogi R, Vanama S. Scalable regular expression matching on data streams. In: Proc. of the SIGMOD 2008. New York: ACM Press, 2008. 161−172. [doi: 10.1145/1376616.1376635]

[12] Rohrer J, Atasu K, Lunteren JV, Hagleitner C. Memory-Efficient distribution of regular expressions for fast deep packet inspection. In: Proc. of the CODES+ISSS 2009. New York: ACM Press, 2009. 147−154. [doi: 10.1145/1629435.1629456]

[13] Kumar S, Dharmapurikar S, Yu F, Crowley P, Turner J. Algorithms to accelerate multiple regular expressions matching for deep packet inspection. ACM SIGCOMM Computer Communication Review, 2006,36(4):339−350. [doi: 10.1145/1159913.1159952]

[14] Kumar S, Turner J, Williams J. Advanced algorithms for fast and scalable deep packet inspection. In: Proc. of the ANCS 2006. New York: ACM Press, 2006. 81−92. [doi: 10.1145/1185347.1185359]

[15] Becchi M, Crowley P. An improved algorithm to accelerate regular expression evaluation. In: Proc. of the ANCS 2007. New York: ACM Press, 2007. 145−154. [doi: 10.1145/1323548.1323573]

[16] Ficara D, Giordano S, Procissi G, Vitucci F, Antichi G, Pietro AD. An improved DFA for fast regular expression matching. ACM SIGCOMM Computer Communication Review, 2008,38(5):29−40. [doi: 10.1145/1452335.1452339]

[17] Liu TW, Yang YF, Liu YB, Sun Y, Guo L. An efficient regular expressions compression algorithm from a new perspective. In: Proc. of the INFOCOM 2011. Piscataway: IEEE Inc., 2011. 2129−2137. [doi: 10.1109/INFCOM.2011.5935024]

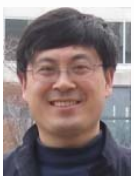
[18] Frieze A, Jerrum M. Improved approximation algorithms for MAX *k*-CUT and MAX BISECTION. Algorithmica, 1997,18(1): 67−81. [doi: 10.1007/BF02523688]

附中文参考文献:

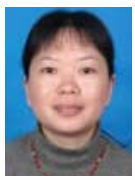
[8] 徐乾,鄂跃鹏,葛敬国,钱华林.深度包检测中一种高效的正则表达式压缩算法.软件学报,2009,20(8):2214−2226. http://www.jos.org.cn/1000-9825/3311.htm [doi: 10.3724/SP.J.1001.2009.03311]

柳厅文(1986－),男,安徽临泉人,博士生,主要研究领域为信息安全,模式匹配.

孙永(1976－),男,博士,工程师,主要研究领域为信息安全,数据流.



卜东波(1973－),男,博士,研究员,博士生导师,主要研究领域为算法分析与设计,生物信息学.



郭莉(1969－),女,正研级高工,CCF 会员,主要研究领域为内容安全管理,网络安全,数据流处理.

方滨兴(1960－),男,博士,教授,博士生导师,中国工程院院士,主要研究领域为计算机结构,计算机网络与信息安全.