doi:10.3969/j.issn.1000-436x.2013.08.014

基于状态分组的高效 *i*-DFA 构造技术

乔登科[1,2]，王卿[3]，柳厅文[1,2]，孙永[1,2]，郭莉[1,2]
（1. 中国科学院 信息工程研究所，北京 100093；2. 信息内容安全技术国家工程实验室，北京 100093；
3. 国家计算机网络应急技术处理协调中心，北京 100029）

摘 要：正则表达式匹配在很多网络安全领域起着非常重要的作用。确定性有限自动机 (DFA, deterministic finite automaton) 具有线速稳定的匹配性能，因而更适合在高速网络环境下执行正则表达式匹配。但 DFA 可能由于状态膨胀而占用巨大的内存空间。作为状态膨胀问题的一种经典解决方案，*i*-DFA 在大幅降低内存开销的同时，还能保证最差匹配性能。然而，已有方法构造 *i*-DFA 时在时间和空间上都是非常低效的。基于状态分组的思想，提出了一种高效的 *i*-DFA 构造方法。进一步地，对状态分组进行了形式化描述，并证明了获得最优状态分组是 NP 困难的，并基于局部搜索的思想提出了一种近优的状态分组算法。实验结果表明，相比经典的 *i*-DFA 构造方法，所做的工作在时间和空间上都有极大的改进：*i*-DFA 的状态规模可能只是已有方法的 2/3，而构造 *i*-DFA 所用时间仅是已有方法的 1/16。

关键词：正则表达式；状态膨胀；状态分组；局部搜索
中图分类号：TP393.08 **文献标识码**：A **文章编号**：1000-436X(2013)08-0102-08

Efficient *i*-DFA construction algorithm based on state grouping

QIAO Deng-ke[1,2], WANG Qing[3], LIU Ting-wen[1,2], SUN Yong[1,2], GUO Li[1,2]
(1. Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China;
2. National Engineering Laboratory for Information Security Technologies, Beijing 100093, China;
3. National Computer Network Emergency Response Technical Team/Coordination Center of China, Beijing 100029, China)

Abstract: Regular expression matching plays an important role in many network and security applications. DFA is the preferred representation to perform regular expression matching in high-speed network, because of its high and stable matching efficiency. However, DFA may experience state explosion, and thus consume huge memory space. As a classical solution for the problem of state explosion, *i*-DFA can reduce the memory consumption significantly and guarantee the worst matching performance at the same time. However, prior methods are inefficient in both time and space during the construction of *i*-DFA. An efficient *i*-DFA construction algorithm based on the idea of state grouping was proposed. Furthermore, a formal description for the problem of state grouping was given, and it was proved that it was NP-hard to get the best state grouping result. Thus, based on local search strategy, a near-optimal algorithm was introduced to divide states into different groups. Compared with the classical construction method, the significant improvement in both time and space is achieved; the *i*-DFA of the proposed method may have 2/3 states as that of prior method and the proposed *i*-DFA is constructed with only 1/16 time of it.

Key words: regular expression; state explosion; state grouping; local search

收稿日期：2013-05-02；**修回日期**：2013-06-01
基金项目：国家高技术研究发展计划(“863”计划)基金资助项目(2011AA010703, 2011AA010705)；国家自然科学基金资助项目(61070026, 61003295)；国家 242 信息安全计划基金资助项目(2011F47)
Foundation Items: The National High Technology Research and Development Program of China (863 Program)(2011AA010703, 2011AA010705);The National Natural Science Foundation of China (61070026, 61003295); 242 National Information Security Program (2011F47)

1　引言

网络在给人们生活带来方便的同时也带来了诸多困扰，如恶意代码、垃圾邮件、病毒等。为了躲避检测，一些恶意特征被隐藏在数据分组的负载中。深度分组检测（DPI, deep packet inspection）技术通过将网络数据分组的内容负载与一些预先设定的模式集（入侵特征、病毒特征等）进行匹配以检测该数据分组是否携带恶意数据。深度分组检测技术被广泛应用在各种网络和安全系统[1,2]中以进行基于感知流量内容的处理。随着网络的发展，入侵特征等越来越复杂，正则表达式(RE, regular expression)以其简洁和表达能力强等优点取代字符串成为描述特征的主要手段。网络带宽和模式集规模的快速增长以及模式规则的复杂化使 DPI 应用中的正则表达式匹配（REM, RE matching）面临严峻挑战。

REM 需要在有限的内存资源下进行高速网络环境下的线速匹配。NFA 和 DFA 是进行正则表达式匹配的 2 种经典的自动机，相应地，REM 的匹配方法可以分为 2 种：基于 NFA 的匹配方法和基于 DFA 的匹配方法。前者具有占用内存小的优点，但在匹配过程中，NFA 可能有多个状态同时活跃，每个活跃状态经过一个字符可能会到达多个状态，导致该类方法状态转换的时间复杂度为 $O(n^2)$（n 是 NFA 的状态数目），无法满足线速匹配要求。与 NFA 相反，DFA 在匹配的任意时刻只有一个活跃状态，处理每个字符只需一次状态迁移，因此具有线速稳健的匹配性能。因而在高速网络环境下，大部分应用都采用基于 DFA 的匹配方法完成实时深度分组检测。

NFA 经过确定化可以生成 DFA。确定化的基本思想是对每个可能同时活跃的 NFA 状态集，都用一个 DFA 状态表示，本文称这些可能同时活跃的 NFA 状态集称为 NFA 的活跃状态集。真实的正则表达式集合的 NFA 在确定化时极易发生状态膨胀，某些情况下 DFA 状态规模甚至呈指数增长。这导致基于 DFA 的匹配方法消耗巨大的内存空间，甚至超过现有 DPI 设备的内存上限。

文献[3]提出了一种称为 *i*-DFA 的复合自动机来解决该问题。通过对将每个 NFA 的活跃状态集分割为若干个子集（子集总数不超过预设阈值 i），*i*-DFA 在大幅降低存储开销的同时，还能保证匹配过程中同时活跃状态数目的最坏情况。如何对 NFA 的活跃状态集进行分割是影响 *i*-DFA 构造结果的关键，但文献[3]并未给出一种高效的分割算法。

本文提出了一个高效的活跃状态集分割算法——IDFA-GROUP (*i*-DFA bases on nfa state grouping)，其基本思想是利用对 NFA 状态集的分组结果对活跃状态集进行分组，并构造对应的 *i*-DFA。本文还提出了一个基于局部搜索思想对 NFA 状态集进行分组的算法 GNSLS (grouping NFA states with local searching)，该算法基于 NFA 状态对之间的相交关系对 NFA 状态进行最优分组。实验结果表明，相比经典的 *i*-DFA 构造方法，本文的工作在时间和空间上都有极大的改进：*i*-DFA 的状态规模有可能只是已有方法的 2/3，而构造 *i*-DFA 所用时间仅是已有方法的 1/16。

2　相关工作

DFA 的状态膨胀使其难以在 DPI 应用中执行高效匹配，因此需要对 DFA 的存储空间进行压缩。DFA 的存储开销主要来自于其状态转移表，表的行宽和带宽分别对应于 DFA 的状态数目和字符集的大小（ASCII 编码该值为 256，每个字符对应一条转移边）。相应地，压缩 DFA 存储空间的工作就分为 2 类：压缩 DFA 状态转移边和减少 DFA 状态数目。

压缩 DFA 的状态转移边的工作基于 DFA 的状态转移边中存在的大量的冗余项，通过消除 DFA 状态之间的冗余边降低 DFA 状态转移表的存储开销。例如，Kumar 等人在文献[4]中提出一种称为延迟输入 DFA (delay input DFA) 的自动机来减少内存开销，它通过引入默认转移边来消除相邻 DFA 状态间的相同转移边；FICARA 等人在文献[5]中通过消除一条转移边的出发状态和目的状态的相同跳转项来减少需要保存的转移边，提出了 δFA 的概念。但是这 2 种自动机处理每个匹配字符都需要多次状态迁移。且该方法需要事先生成模式集的 DFA，而这一点在实际情况中是很难满足的。

很多正则表达式单独生成 DFA 不会引起状态膨胀，但是生成其对应的合并 DFA 时却会引起剧烈的状态爆炸，因此可以通过生成多个 DFA 来减少 DFA 的状态数目。YU F 等人[6]首次提出正则表达式分组的想法，但是所提出的基于启发式策略的分组算法分组效率十分低下，无法在大规模正则表达式下面对正则表达式进行高效分组。柳厅文等人[7]基于正则表达式之间的冲突关系将正则表达式的最优分组问题转换为无向图的最大 k 割问题，并提

出了基于局部搜索的分组算法——GRELS (grouping regular expressions with local searching)算法，通过近似地解决无向图的最大 k 割问题来对正则表达式进行分组。实验证明，GRELS 算法不仅可以更好对正则表达式集进行分组，并且算法的处理效率也大大优于 Fang Yu 的分组算法。

为在大幅降低自动机存储开销的同时尽量保持 DFA 的匹配效率，复合自动机技术得到广泛研究[3,8,9]。例如，Hybrid-FA[8]依据经验启发式地寻找导致 DFA 状态膨胀的位置，然后在这些位置不进行确定化来避免状态膨胀。由于不能准确找到导致膨胀的位置，Hybrid-FA 适用的范围很有限，其构造过程花费时间过长并且在某些正则表达式上效果很差。XU Y 等人[3]提出的 i-DFA 可以保证在匹配过程中最多只有 i 个活跃状态，但是作者提出的 i-DFA 启发式构造方法效率异常低下，无法保证最后生成的 i-DFA 是最优的。本文提出了一种新的构造 i-DFA 的高效方法，是对 XU Y 等人工作的改进。

3 基本概念

有限自动机（FA, finite automaton）通常用五元组 $M=(Q,\Sigma,\delta,q_0,F)$ 表示，其中，Q 是状态集合，Σ 是字符集，δ 是状态转移函数，q_0 是自动机的开始状态($q_0\in Q$)，F 是自动机的接受状态的集合($F\subseteq Q$)。NFA 和 DFA 是 2 种经典的 FA，两者的区别在于 NFA 的状态转移函数（$\delta:Q\times\Sigma\to\mathcal{P}(Q)$，$\mathcal{P}(Q)$ 是 Q 的幂集）是非确定性的，即 NFA 每个状态处理一个字符后可以到达多个状态；DFA 的状态转移函数（$\delta:Q\times\Sigma\to Q$）是确定性的，即 DFA 的每个状态处理一个字符后只能到达一个确定的状态。运用子集构造法[10]可以将一个 NFA 确定化为 DFA。

在匹配时，FA 从初始状态开始依次处理待匹配输入串，处理一个字符时自动机沿着当前状态的所有标记为该字符的所有转移边进行迁移，当某个接受状态被访问时，即可认为匹配成功。由于 NFA 状态对应一个字符可能具有多条转移边，因此在 NFA 的匹配过程中可能有多个活跃状态，这些同时活跃的状态集合就是 NFA 的活跃状态集。用 $N=\{N_s\mid\forall s\in\Sigma^+\}$ 表示 NFA M 的活跃状态集的集合，其中 Σ^+ 是 NFA 状态表达的语言集合，N_s 是语言 S 对应的活跃状态集，定义为 $\forall s\in\Sigma^+, N_s=\{q\in Q\mid q_0\xrightarrow{s}q\}$。

给定 FA $M=(Q,\Sigma,\delta,q_0,F)$，对任意两状态 $x,y\in Q$，认为状态 y 能被字符串 $w\in\Sigma^+$ 从状态 x 激活如果存在着一条从 x 到 y 并且标记为 w 的路径，简记为 $x\xrightarrow{w}y$。那么，针对任意一个状态 x 可以定义一个字符串集合 $L(x)=\{w\in\Sigma^+\mid q_0\xrightarrow{w}x\}$。YANG Y H 等人[11]依据自动机状态的字符串集合间的关系定义了状态间的 3 种状态关系，即互斥、包含、相交。作者还证明了 NFA 状态间的相交关系导致了确定化时的状态膨胀。

乔登科等人[12]用 $x\wedge y$、$\bar{x}\wedge y$ 和 $x\wedge\bar{y}$ 这 3 个变量来描述 FA 状态 x 和 y 的关系，并将状态关系扩展成 5 种，分别是互斥、等价、包含、包含于和相交。FA 的活跃状态集暗含了 FA 状态的语言集，如果状态 $x\in N_s$，那么 $s\in L(x)$，因此获得了 FA 的活跃状态集就等于获得了 FA 状态的语言集合。基于 FA 的活跃状态集，文献[12]提出了一个高效获得 FA 状态关系的算法——ROBAS 算法。该算法只需扫描一遍 FA 的活跃状态集，便可高效准确定量地求出 FA 的所有状态关系。NFA 的活跃状态集就是 NFA 确定化后生成的 DFA 状态所对应的 NFA 状态集合，因此可以用子集构造法求出 NFA 所有的活跃状态集，但是该方法效率很低下。文献[12]还提出了一个高效获得 FA 活跃状态集的算法——ASSSUBSET 算法。该算法基于子集构造法提出了 2 种改进策略以高效获得 FA 的活跃状态集：首先，该算法不生成最后的 DFA，大大降低了算法的空间开销；其次，该算法首先对 NFA 状态的转移边进行预处理，去除一些无用的转移边，大大降低了算法的时间开销。

4 *i*-DFA 介绍

在 NFA 确定化生成 DFA 的过程中，子集构造法从 NFA 的开始状态 q_0 开始，遍历 NFA 的所有可能活跃状态集，对于每一个活跃状态集都用 DFA 的一个状态表示。

与 DFA 不同，i-DFA 采用最多 i 个状态表示 NFA 的一个活跃状态集，在这里将这些 i-DFA 状态称为 NFA 的活跃状态集的状态表示集。在匹配过程的任意时刻，i-DFA 最多只有 i 个活跃状态。由于每个 i-DFA 状态均可能出现在 NFA 的多个活跃状态集的状态表示集中，所以 i-DFA 的状态规模要远远小于 DFA 的状态规模。

以正则表达式“*ab.* c*”、“*ab.* e*”和“*f*”为例，

它们所对应的 NFA 和 *i*-DFA 如图 1 所示（$i = 2$）。假定待匹配的字符串为“*fabc*”，*i*-DFA 的匹配过程如下：从初始活跃状态集 {0} 出发，查询 SSL 表(如图 1 (b) 所示)，找出初始的 *i*-DFA 状态 0，再通过 *i*-DFA structure（如图 1 (c) 所示）和转移字符“*f* ”，找出下一个活跃状态集 {0,1}，紧接着通过 SSL，找到所对应的 *i*-DFA 活跃状态集 {0,1}，1 是接收状态，接收规则是“*f* ”；以此类推，反复查找 structure 和 SSL，便可依次找出 *i*-DFA 的活跃状态集 {0,2}，{03} 和 {035}，其中状态 035 是接收状态，接受规则是“*ab.*c*”。由上述匹配过程可以看出，$i = 2$ 时，*i*-DFA 在匹配过程中同时活跃的状态数目最多只有 2，*i*-DFA 可以保证匹配过程中的最坏匹配复杂度。

5　IDFA-GROUP 算法

5.1　算法动机

由 *i*-DFA 的定义可知，如何对 NFA 的活跃状态集进行分割是影响最后生成的 *i*-DFA 优劣的关键因素。

XU Y 等给出了一个简易的 *i*-DFA 构造算法。算法思路如下：1) 首先对 NFA 的每一个活跃状态集进行分割，每个活跃状态集最多可以被分成 *i* 份，其中至少有一份的状态数目小于或等于 *i*，每个活跃状态集对应着多个分割形式，而最后只能选择其中一种；2) 扫描所有可能的分割形式，选择其中出现次数最多的状态集，然后生成对应的 *i*-DFA 的状态，同时更新 SSL 表和 NFA 活跃状态集的分割形式；3) 重复第 2)步，直到 NFA 的所有活跃状态集均被表示完为止。

假设最后生成的 *i*-DFA 状态数目是 N_{idfa}，NFA 状态数目是 N，NFA 的平均活跃状态数目是 N'，那么 XU Y 等人所提出的算法的空间复杂度是 $O((C_{N'}^{0} + C_{N'}^{1} + \cdots + C_{N'}^{i}) \times 2^{N+1})$，其中 i 的取值范围是 $[0, N']$。当 $i = N'$ 时，算法的空间复杂度为 $O(2^{N+N'+1})$。由于每次扫描所有的分割形式后只能生成一个 *i*-DFA 状态，所以算法的时间复杂度是 $O(N_{\text{idfa}} \times 2^{N+N'+1})$。可见，该方法不管时间复杂度还是空间复杂度都太高，很难满足实际需求。

本文提出了一个新的 *i*-DFA 构造算法——IDFA-GROUP 算法。该算法基于 NFA 状态的分组结果，依次扫描 NFA 的所有活跃状态集，然后动态生成所有的 *i*-DFA 状态和 SSL 表，算法只需要扫描 NFA 的所有活跃状态集一遍，便可生成对应的 *i*-DFA。

5.2　算法思路

假设已经获得了 NFA M 的状态分组结果 $R = (R_1, R_2, \cdots, R_k)$，分组个数是 k。

算法 1 是 IDFA-GROUP 算法的伪代码。NFA 的活跃状态集可由 ASS-SUBSET 算法求出。

算法 1　IDFA-GROUP 算法——根据 NFA 的分组结果和活跃状态集求出对应的 *i*-DFA。

算法输入：NFA 的状态分组 $R = (R_1, R_2, \cdots, R_k)$、NFA 的所有活跃状态集的集合 S

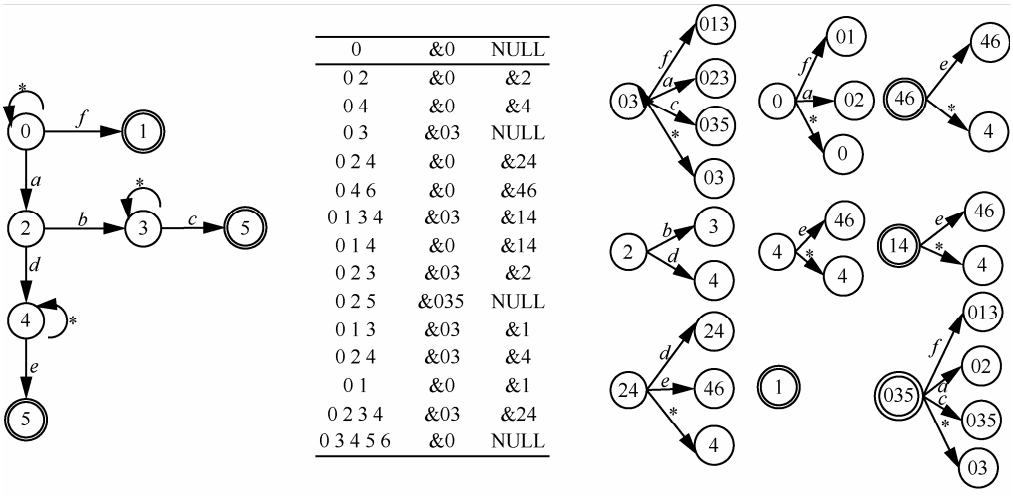
算法输出：该 NFA 所对应的 EFA

1) 初始化 *i*-DFA **i*DFA = NULL;

2) While (S 不为空) {

3)　　从 S 中取出一个元素 S_i，并从 S 中删除 S_i;

4)　　For (int j = 0;　j <= k;　j++) {



(a) r_1, r_2 和 r_3 对应的NFA　　(b) r_1, r_2 和 r_3 对应的*i*-DFA　　(c) r_1, r_2 和 r_3 对应的*i*-DFA Structure

图 1　r_1=“*ab.*c*”, r_2=“*ab.*e*”和 r_3=“*f* ”对应的 NFA 和 *i*-DFA

5) 求出 S_i 和 R_j 交集的集合 S_{ij};

6) If (S_{ij} 为空) continue ;

7) If (iDFA 没有 S_{ij} 对应的状态) {

8) 生成 SC_i 对应的 *i*-DFA 状态，并更新状态接受规则和状态转移边;

9) 更新 iDFA 的状态集合;

10) }

11) 更新 iDFA SSL 表 S_i 的状态表示集;

12) }

13) }

14) Return iDFA;

5.3 算法分析

$R=(R_1,R_2,\cdots,R_k)$ 可以覆盖所有的 NFA 状态，所以 $S_{i1}\bigcup S_{i2}\bigcup\cdots S_{ik}=S_i$，IDFA-GROUP 算法可以生成正确的 *i*-DFA；IDFA-GROUP 算法只需扫描 NFA 的活跃状态集一遍便可生成对应的 *i*-DFA，其时间复杂度和空间复杂度都是 $O(2^N)$。相比现有算法的时间复杂度 $O(N_{\text{idfa}}\times 2^{N+N'+1})$ 和空间复杂度 $O(2^{N+N'+1})$，IDFA-GROUP 算法在时间复杂度和空间复杂度上都有大幅提升。

6 NFA 状态分组

6.1 分组动机

由 IDFA-GROUP 算法的思路可以看出，最终生成的 *i*-DFA 的优劣取决于对 NFA 状态集的分组结果。通过对 NFA 状态集的分组，希望分组的数目最少同时构造的 *i*-DFA 的状态数目最少。但这 2 个目标通常是矛盾的。如果所有状态分为一组，那么构造的 *i*-DFA 实际上是 NFA 完全确定化生成的 DFA；如果每个 NFA 状态作为一个分组，那么构造的 *i*-DFA 实质上就是完全未确定化的 NFA。因而，本文的目标在给定状态分组个数情况下，最小化构造的 *i*-DFA 的状态数目。

6.2 分组的形式化描述

每个 *i*-DFA 状态都对应一个 NFA 状态集合，这个集合中的所有状态都属于同一个 NFA 状态组。用 C_i 表示 NFA 状态组 R_i 里的状态互相组合形成的 *i*-DFA 状态数目，则最后生成的 *i*-DFA 状态数 $C=C_1+C_2+...+C_k=\sum_{i=1}^{k}C_i$。

YANG Y H 等人证明了只有当 NFA 的状态间存在相交关系时，NFA 的确定化才会发生状态膨胀，而等价、互斥和包含关系不会引起 DFA 状态膨胀。作者同时还证明如何 NFA 的状态两两都不相交，那么该 NFA 确定化生成的 DFA 状态数目不会大于 NFA 的状态数目。因此，得到了 C_i 的近似表示形式 $C_i\approx C(R_i)+\sum_{N_p,N_q\in R_i;p<q,}w_{p,q}$。其中，$C(R_i)$ 是 R_i 的状态数目，$w_{p,q}$ 是状态 *p* 和状态 *q* 的相交权值，NFA 的状态相交权值可由 ROBAS 算法求出。令 $C(M)$ 表示 NFA 的状态数目。于是 *i*-DFA 的状态总数为

$$C=C_1+C_2+\cdots+C_k=\sum_{i=1}^{k}C_i$$

$$\approx C(R_1)+C(R_2)+\cdots+C(R_k)+\sum_{i=1}^{k}\sum_{N_p,N_q\in R_i;p<q}w_{p,q}$$

$$=C(M)+\sum_{i=1}^{k}\sum_{N_p,N_q\in R_i;p<q}w_{p,q}$$

可以看出，最小化 *i*-DFA 的状态规模等价于最小化 $\sum_{i=1}^{k}\sum_{N_p,N_q\in R_i;p<q}w_{p,q}$，因而进行状态分组的目标变为了最小化 $\sum_{i=1}^{k}\sum_{N_p,N_q\in R_i;p<q}w_{p,q}$。

基于 NFA 状态间的相交关系，可以根据 NFA 构造一个无向图，其中无向图的顶点为 NFA 的状态，2 个 NFA 状态间如果是相交的那么在相应顶点间存在一条边。称这个无向图为 NFA 的状态关系图。

给定无向图 $G=(V,E)$，其中，$V=(1,2,\cdots,n)$ 是无向图顶点的集合；$E\subseteq V\times V$ 是它的边的集合；$w_{i,j}$ 为连接顶点 *i* 和顶点 *j* 的边上的权值。在本文中，*V* 则代表着 NFA 的状态集合；若 NFA 状态 *i* 和 NFA 状态 *j* 有相交关系，则将边 $e_{i,j}$ 加入到关系图的边的集合 *E* 中，同时赋予 $w_{i,j}$ 这 2 个状态的相交权值。由于 DFA 任意 2 个状态都不相交，所以 DFA 的状态关系图就是一堆孤立的点的集合。于是，NFA 的状态分组便变成了无向图的最大 *k* 割问题。该问题是一个 NP-Hard 问题，因此 NFA 的状态分组问题也是一个 NP-Hard 问题。

令：$\mu_{i,j}$ 表示 *i* 和 *j* 是否属于同一个集合，$\mu_{i,j}=0,\forall i,j\in S_p$；$\mu_{i,j}=1,\forall i\in S_p\wedge j\in S_q\wedge p\neq q$。

令：$\chi=\{e_{i,j}\in E;\mu_{i,j}=1\}$。即 χ 为边的集合，这些边的 2 个顶点分属于不同的集合。则由 χ 确定

的 $cut(\chi)$ 即为

$$cut(\chi)=\sum_{e_{i,j}\in\chi(s)} w_{i,j}=\sum_{i<j} w_{i,j}\times\mu_{i,j}$$

最大 k 割问题就是找一个集合 χ，使 $cut(\chi)$，即 χ 中的边的权值总和最大。

求 $cut(\chi)$ 最大，其实也相当于求 $optimal(\chi)$ 最小，$optimal(\chi)$ 的定义为

$$optimal(\chi)=\sum_{i=1}^{k}\sum_{N_p,N_q\in R_i;p<q} w_{p,q}$$
$$=\sum_{i<j} w_{i,j}\times(1-\mu_{i,j})$$

称 $optimal(\chi)$ 为该无向图即 NFA 的状态关系图的适应值函数。

于是 NFA 的状态分组的的数学模型等价于优化下面的问题：

$$\mathrm{Min}\sum_{i<j} w_{i,j}\times(1-\mu_{i,j}),\mu_{i,j}\in\{0,1\}$$

6.3　算法思路

NFA 状态的分组问题是一个 NP-hard 问题，因此在线性时间内无法获得最优的状态分组结果。文献[7]提出了一个近似求解无向图的最大 k 割问题的算法——GRELS 算法。借鉴 GRELS 算法的思想，本文提出了一个 NFA 状态分组算法——GNSLS 算法，可以近似获得较好的状态分组结果。

算法 2 是给出的 GNSLS 算法的伪代码。NFA 的状态关系矩阵 $\boldsymbol{w}[n][n]$ 可由 ROBAS 算法求出。$w[i][j]=k(k\geqslant 0)$，表明状态 i 和状态 j 相交，并且权值是 k；$w[i][j]=0$，表明状态 i 和状态 j 非相交，不会导致状态膨胀。

算法 2　GNSLS 算法——根据自动机的状态关系求出自动机的状态分组结果。

算法输入：NFA 状态间的关系矩阵 $\boldsymbol{w}[n][n]$、NFA 分组数目 k

算法输出：NFA 的状态分组结果 $R=(R_1,R_2,\cdots,R_k)$

1) 初始化 NFA 的状态关系图 ：申请二维矩阵 $\boldsymbol{\mu}[n][n]$，初始化为 0，$\mu[i][j]$ 即代表状态 i 和状态 j 是否归属于同一个状态组；

2) 初始化 $R=(R_1,R_2,\cdots,R_k)$，均值为空；初始化布尔型变量 *nochange*，赋为 true；

3) 随机地将每个 NFA 状态不重复地分到一个状态组中，同时更新 $\boldsymbol{\mu}[n][n]$；

4) 根据状态关系图的适应值函数计算此时 NFA 状态关系图的适应值 min；

5) While (*nochange*) {
6)　　*nochange* = *false*;
7)　　Foreach (状态 id *x* 大于等于 0 小于等于 *n*) {
8)　　For (*i* = 1; *i* <= *k*; *i*++) {
9)　　将状态 *x* 从原先的状态组中移到新状态组 R_i 中，并更新 $\mu[n][n]$；
10)　　计算此时的适应值 *temp*;
11)　　If (*temp* < min) {
12)　　*nochange* = true;
13)　　min = *temp*;
14)　　*index* = *i*;
15)　　}
16)　　}
17)　　将状态 *x* 加到状态组 R_{index}；
18)　　}
19) }
20) 返回状态分组结果 $R=(R_1,R_2,\cdots,R_k)$

该算法首先随机地将 NFA 的每个状态分配到一个分组中，然后扫描所有的状态，判断是否存在一个状态，当把它从当前的分组移动到另一个分组时 NFA 状态关系图的适应值会减少。如果存在，则将其移动至使得 NFA 状态关系图的适应值最小的那个分组中；如果不存在，则移动任意一个状态都不会导致 NFA 状态关系图的适应值减少。该算法没有考虑同时移动 2 个或者多个状态的情况，所以只是局部最优的，但是该算法依然可以近似地求出最好的分组结果。

6.4　算法分析

正确性分析：GNSLS 算法可以保证最终 NFA 的每一个状态都只会出现在一个项目组中，所以 GNSLS 算法最后可以得到正确的分组结果。

复杂度分析：当每个 NFA 状态都是一个单独的状态组时，$\boldsymbol{\mu}[n][n]$ 全部元素均为 0，此时 NFA 的状态关系图的适应值最大，即 $\mathrm{Max}=\sum_{i=1}^{i=n-1}\sum_{j=i+1}^{j=n} w_{i,j}$。由于算法每次循环都使得适应值至少减少 1，所以算法至多循环 Max 次 while 循环结束，GNSLS 算法收敛，时间复杂度是 $O(\mathrm{Max}\times n\times k)$。但是由于最终状态关系图的适应值肯定大于 0 并且很多次循环减少的适应值都远大于 1，所以实际情况下算法的执行步数要远远小于该值。

近似比分析：文献[7]证明，对于最大 k 割问题，

GRELS 算法的近似比是$1/(1-1/k)$。因此，GNSLS 算法求出的 NFA 状态分组结果对于 NFA 状态的最优分组的相似比也是$1/(1-1/k)$。

7 实验结果

为了使实验结果更具有可比性，从 Snort、Bro、L7filter、ClamAV 等 4 个开源的正则表达式规则集中随机提取 8 个有代表性的真实正则表达式集合，即 Snort24、Snort31、Snort34、Snort45、ClamAV15、L7filter8、L7filter21 和 Bro217 作为实验模式集。本文实验环境为 Intel Core2 Duo 2.1 GHz, 512 MB 内存。

本文将 XU Y 等人的算法称为 XUY 算法。在本文实验过程中，先用 GNSLS 算法对这 8 个模式集对应的 NFA 状态集进行分组，然后分别用 IDFA-GROUP 算法和 XUY 算法分别生成这 8 个模式集的 *i*-DFA。这 2 种算法所生成的 *i*-DFA 和模式集的 NFA、DFA 的状态规模和处理时间如表 1 所示。这 8 个模式集对应的 NFA、*i*-DFA (XUY)和 *i*-DFA (IDFA-GROUP) 在实际网络流量下的匹配效率比较如表 2 所示。

从表 1 可以看出，GNSLS 算法结合 IDFA-GROUP 算法可以生成状态数目更少的 *i*-DFA。具体地，对于模式集 Snort45、ClamAV15、L7filter8 和 L7filter21，IDFA-GROUP 和 XU Y 构造的 *i*-DFA 状态数目大致相等。但对于规则集 Snort24、Snort31、Snort34 和 Bro217，IDFA-GROUP 构造的 *i*-DFA 的状态规模大概比 XU Y 小 1/3 左右。

从表 2 可以看出，IDFA-GROUP 算法构造得到的 *i*-DFA 和 XU Y 构造的 *i*-DFA 在活跃状态集规模上非常接近，均大大少于 NFA 的活跃状态集规模。但 IDFA-GROUP 算法的时间消耗明显小于 XUY 算法。对于模式集 Snort24、Snort31、Snort34 和 Snort45，IDFA-GROUP 算法相对 XUY 算法分别节省了 93.4%、94.9%、93.8%和 93%的时间。对于模式集 L7filter8 和 L7filter21，IDFA-GROUP 算法分别节省了 88.7%和 89%的时间。即使对于模式集相对简单的 ClamAV15 和 Bro217，IDFA-GROUP 算法依然节省了 70%左右的时间。

结合表 1 和表 2 可以看出，GNSLS 算法和 IDFA-GROUP 算法构造的 *i*-DFA 的状态规模大概是 NFA 状态规模的 2~7 倍，但远小于 DFA 的状态规模，甚至于只有 DFA 状态规模的 1/20 左右。本实验中的规则集是随机从开源正则表达式集合中提

表 1　IDFA-GROUP 算法和 XUY 算法的效果比较

规则集	NFA	DFA	*i*-DFA (XU Y)		*i*-DFA (IDFA-GROUP)	
	状态数	状态数	状态数	构造时间/s	状态数	构造时间/s
Snort24	575	14 356	2 528	252.56	1 662	16.69
Snort31	917	21 419	3 683	860.45	2 368	43.91
Snort34	891	14 358	3 101	409.24	2 319	25.61
Snort45	975	72 025	3 643	2 878.52	4 209	200.99
ClamAV15	449	68 176	3 055	413.85	3 263	125.59
L7filter8	251	21 528	1 700	180.55	1 540	20.39
L7filter21	397	19 710	1 999	241.67	1 846	26.40
Bro217	1 999	6 591	2 862	91.51	1 999	27.44

表 2　NFA、*i*-DFA (XUY) 和 *i*-DFA (IDFA-GROUP) 的匹配性能比较

规则集	NFA			*i*-DFA (XU Y)			*i*-DFA (IDFA-GROUP)		
	活跃集最大规模	活跃集平均规模	匹配时间/s	活跃集最大规模	活跃集平均规模	匹配时间/s	活跃集最大规模	活跃集平均规模	匹配时间/s
Snort24	5	1.077 990	233.13	2	1.000 594	25.48	2	1.008 965	26.05
Snort31	13	1.183 184	233.65	2	1.008 250	38.96	2	1.050 783	40.22
Snort34	6	1.062 502	219.77	2	1.173 985	23.94	2	1.050 182	24.21
Snort45	6	1.136 063	238.17	2	1.002 471	24.08	2	1.124 508	24.82
ClamAV15	4	2.041 640	413.85	2	1.085 436	63.60	2	1.956 126	64.35
L7filter8	6	1.956 207	398.64	2	1.047 381	51.76	2	1.897 289	52.19
L7filter21	9	2.886 746	575.13	2	1.042 571	113.82	2	1.891 352	115.32
Bro217	14	6.267 370	1 256.50	2	1.228 485	200.45	2	1.231 970	194.09

取的，因而具有很强的代表性，所以这 2 个算法可以应用到大规模正则表达式集上以构造对应的 *i*-DFA 完成高性能正则表达式匹配。

8　结束语

网络带宽的爆炸增长和网络入侵特征的不断复杂化给深度分组检测带来了巨大的压力。传统的基于 NFA 的匹配技术和基于 DFA 的匹配技术均无法满足当前高速网络环境下深度分组检测有限内存环境下线速匹配的需求。越来越多的人开始研究复合自动机的构造方法，但是市场上目前出现的大部分复合自动机无法保证匹配过程中的最坏情况。

i-DFA 不仅可以降低自动机的存储开销，还能保证最差匹配性能，对待检测的流量特征不敏感，因此受到了广泛关注。但是目前尚没有一个时空高效的 *i*-DFA 构造方法。本文提出了一个基于状态分组思想的高效构造 *i*-DFA 的算法——IDFA-GROUP 算法。该算法利用对 NFA 状态集的分组结果，分别计算 NFA 活跃状态集和各个分组的交集，然后用一个状态表示这些交集。基于 NFA 状态对之间的相交关系，本文对 NFA 状态集的分组问题进行了形式化描述，并利用 NFA 状态关系图的适应值作为指标衡量 NFA 状态分组结果的优劣。证明了 NFA 状态集的最优 k 分组问题等价于经典的最大 k 割问题，因而是一个 NP 困难的。本文最后基于局部搜索的思想给出了一个近似最优的 NFA 状态分组算法。实验结果表明，相比经典的 *i*-DFA 构造方法，本文的工作在时间和空间上都有极大的改进：*i*-DFA 的状态规模可能只是已有方法的 2/3 左右，而构造 *i*-DFA 所用时间则仅是已有方法的 1/16。

参考文献：

[1] VERN P. Bro: a system for detecting network intruders in real-time[J]. Computer Networks, 1999, 31(23):2435-2463.

[2] MARTIN R. Snort - lightweight intrusion detection for networks[A]. Proc USENIX LISA[C]. Berkeley, USA. 1999. 229-238.

[3] XU Y, JIANG J C, SONG Y, *et al*. *i*-DFA: a Novel Deterministic Finite Automaton without State Explosionp[R]. Polytechnic Institute of NYU 2010: Technical Report, 2010.

[4] KUMAR S, DHARMAPURIKAR S, FANG Y, *et al*. Algorithms to accelerate multiple regular expressions matching for deep packet inspection[J]. ACM SIGCOMM Computer Communication Review, 2006, 36(4): 339-350.

[5] FICARA D, PIETRO A D, GIORDANO S, *et al*. Differential encoding of DFAs for fast regular expression matching[J]. IEEE/ACM Transactions on Networking (TON), 2011, 19(3):683-694.

[6] YU F, CHEN Z F, DIAO Y L, *et al*. Fast and memory-efficient regular expression matching for deep packet inspection[A]. Proc of the ANCS'06[C]. New York: ACM, 2006. 93-102.

[7] 柳厅文, 孙永, 卜东波等. GRELS：正则表达式分组的 1/(1-1/*k*)-近似算法[J]. 软件学报, 2012, 23(9):2261-2272.
LIU T W, SUN Y, PU D P, *et al*. GRELS: regular expressions grouping algorithm based on local searching[J]. Journal of Software, 2012, 23(9):2261-2272.

[8] BECCHI M, CROWLEY P. A hybrid finite automaton for practical deep packet inspection[A]. Proc of the 2007 ACM CoNEXT Conference[C]. New York: ACM, 2007.

[9] TANG Y, JIANG J, HU C, *et al*. Managing DFA history with queue for deflation DFA[J]. Journal of Network and Systems Management, 2012, 20(2):155-180.

[10] AHO A V, LAM M S, SETHI R, *et al*. Compilers: Principles, Techniques, and Tools[M]. Boston: Addison Weslay, 1986.

[11] YANG Y H, PRASANNA V. Space-time tradeoff in regular expression matching with semi-deterministic finite automata[A]. Proc of IEEE INFOCOM, 2011[C]. Piscataway, 2011. 1853-1861.

[12] 乔登科,柳厅文,孙永等. 一种获得有限自动机状态间关系的高效算法[EB/OL]. http://www. cnki. com. cn /Article/CJFDT otal-JFYZ 2012S2023. htm.
QIAO D K, LIU T W, SUN Y, *et al*. An efficient algorithm to obtain relations between states of finite automata[EB/OL]. http://www. cnki. com. cn /Article/CJFDT otal-JFYZ 2012S2023. htm..

[13] QIAO D K, LIU T W, SUN Y, *et al*. Poster abstract: EFA for efficient regular expression matching in NIDS[A]. Proc of The 15th International Symposium on RAID[C]. Amsterdam, Netherlands, 2012. 382-383.

[14] ZU Y, YANG M, XU Z, *et al*. GPU-based NFA implementation for memory efficient high speed regular expression matching[A]. Proc ACM PPoPP[C]. New York, USA, 2012. 129-140.

作者简介：

乔登科（1987-），男，河南三门峡人，中国科学院硕士生，主要研究方向为信息安全、模式匹配。

王卿（1983-），男，山东济南人，博士，国家计算机网络应急技术处理协调中心工程师，主要研究方向为网络信息论、无线网络资源优化算法、网络编码技术等。

柳厅文[通信作者]（1986-），男，安徽临泉人，博士，中国科学院助理研究员，主要研究方向为模式匹配、数据泄露防护、算法分析与设计。E-mail: liutingwen@iie.ac.cn。

孙永（1976-），男，辽宁阜新人，博士，中国科学院副研究员，主要研究方向为信息安全、数据流处理等。

郭莉（1969-），女，湖南株洲人，中国科学院高级工程师、博士生导师，主要研究方向为信息内容安全管理，网络安全，数据流处理，网络流计算等。