面向入侵检测系统的模式匹配算法研究

徐周波 张永超 古天龙 宁黎华
(桂林电子科技大学广西可信软件重点实验室 桂林 541004)

摘 要 入侵检测系统Snort检测的基本原理是模式匹配。为了提高模式匹配算法的效率,从两方面对Snort中的BM算法进行改进。首先,为了增大模式串移动的距离,改进算法利用了与模式串最右端对齐的下一个及第二个文本字符,以及这两个字符再向右偏移模式串长度所对应字符在模式串中的出现情况,最大移动距离达到了$2m+2$。其次,为了增大失配时大的移动距离出现的概率,利用了最右端字符与其下一个字符的组合概率特性。最后,对算法进行了性能测试。测试结果表明改进算法减少了窗口移动次数和字符比较次数,提高了匹配效率。

关键词 入侵检测,Snort,模式匹配,BM改进算法
中图法分类号 TP312 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2017.09.025

Research on Pattern Matching Algorithm in Intrusion Detection System

XU Zhou-bo ZHANG Yong-chao GU Tian-long NING Li-hua
(Guangxi Key Laboratory of Trusted Software, Guilin University of Electronic Technology, Guilin 541004, China)

Abstract As an network intrusion detection system, Snort's detection principle is based on pattern matching. In order to improve the efficiency of the matching algorithm, the BM algorithm in Snort was improved from two aspects. Firstly, in order to increase the moving distance when missing match, the two characters following the character which is aligned with the rightmost location of the pattern in the text and the two corresponding characters moved by length of the pattern are taken into consideration. And the most moving distance is $2m+2$. Furthermore, the appearance frequency of the bigger moving distance when missing match is increased by using the probability characteristic of the combination of the rightmost and its next characters. Finally, experiments on these algorithms were conducted. The experimental results show that the proposed algorithm can effectively reduce the times of moving windows and comparing character. As a result, the matching efficiency is improved.

Keywords Intrusion detection, Snort, Pattern matching, Improved BM algorithm

1 引言

网络入侵检测系统(Network Intrusion Detection System,NIDS)是一个能够弥补防火墙不足的主动防御系统,在NIDS中入侵行为经常采用特定的字符串进行描述。Snort是一个开源的、强大的轻量级NIDS[1],其中的单模式匹配算法采用了经典的BM(Boyer-Moore)算法,它通过实时获取网络数据包,并将其与已知的入侵规则库进行匹配来探测入侵行为。根据Snort的作者Marty Roesch的测试统计,模式匹配过程所用时间占NIDS总处理时间的30%左右,当网络通信量密集时甚至高达80%。此外,网络传输速度的提升对模式匹配速度的要求更高,模式匹配速度的快慢直接影响检测系统的准确性和实时性[2]。BM算法的"好后缀"规则的预处理和计算过程比较复杂,并且研究表明"好后缀"规则在匹配过程中的使用概率仅为5.97%,严重限制了匹配的速度。

为了简化BM算法,Horspool[3]和Sunday[4]相继提出了BMH算法和BMHS算法。这两种算法都去掉了BM算法的"好后缀"规则,只保留了"坏字符"规则的思想。BMH算法利用文本串中与模式串最右端对齐的字符来决定模式串向右移动的距离,最大移动距离为m(模式串长度);BMHS算法则利用文本串中与模式串最右端对齐的字符的下一字符来决定右移距离,最大移动距离为$m+1$。牟永敏等[5]将BM和BMHS算法进行了结合,移动距离取两者的较大值。Qiao J X等[6]对BM算法的"坏字符"规则进行了修改,统计模式串中出现频率最低的字符,在每一轮匹配前都先对该字符进行匹配。范洪博等[7]提出了一种基于位并行机制的改进算法,

到稿日期:2016-08-24 返修日期:2016-11-05 本文受国家自然科学基金(61572146,61363030,U1501252),广西自然科学基金(2016GXNSFDA380006,2014GXNSFAA118354),广西高等学校高水平创新团队及卓越学者计划资助。
徐周波(1976-),女,博士,副教授,CCF高级会员,主要研究领域为符号计算、智能规划与约束求解,E-mail:xzbli_11@guet.edu.cn;张永超(1988-),男,硕士生,主要研究领域为计算机网络、信息安全,E-mail:1534450577@qq.com;古天龙(1964-),男,博士,教授,博士生导师,主要研究领域为形式化方法、符号计算、知识工程,E-mail:cctlgu@guet.edu.cn;宁黎华(1981-),女,博士生,主要研究领域为智能计算、约束求解。

该机制维护一个位掩码，位掩码的每一位对应一个模式字符，一旦模式串长度大于机器字长，匹配性能会随着 m/w（模式串长度/机器字长）比值的上升而下降[8]。以上几种改进算法在时间性能上均未有较大提高，难以满足检测系统实时性的要求。Yun S[9]，Lin C H 等[10]以及 Erdem O[11]用硬件实现了匹配算法，匹配速度快，但是其受片上 RAM 存储空间的限制，并且 TCAM(Ternary Content Addressable Memory)价格昂贵，功耗大。孙文静等[12]提出了一种 BM 改进算法，根据文本串所对应的模式串最后位置的下一个字符位置的信息确定右移量，最大移动距离为 $m+1$。马占飞等[13]提出了 BM 改进算法，利用模式串末字符和末字符对应文本串的后两个字符来提高右移量，最大移动距离为 $m+2$。Cho S 等[14]将 BMH 算法的“好后缀”规则运用到保序模式匹配问题中，比较适合于顺序同构范畴的模式匹配。Nsira N B 等[15]针对 DNA 序列不同部分相似性极高的特点，对 BM 算法进行了改进，当失配发生在模式串最后一位时使用“坏字符”规则，其他情况使用“好后缀”规则，该方法比较适用于对 DNA 序列进行匹配。张宏莉等[16]提出了一种基于 WM(Wu and Manber)算法的多模式改进算法，其与单模式匹配算法相比内存占用较大。

鉴于 BM 算法是目前公认效率较高、使用最多的单模式匹配算法，算法实现简单，而多模式匹配算法普遍存在实现复杂、空间效率低下的问题，且网络带宽的增加使得实际应用中的 BM 算法的匹配性能并不理想，匹配速度有待进一步提高，本文提出了一种 BM 匹配算法的改进算法。由于在实际应用中 BM 算法的“坏字符”规则的应用次数远大于“好后缀”规则，并且“好后缀”不容易实现，因此改进算法抛弃了 BM 算法的“好后缀”规则，利用单字符和双字符组合的特性，增大不匹配时模式串移动的距离和大的移动距离出现的概率，最大移动距离达到了 $2m+2$。通过实验对比得出，改进算法减少了窗口移动次数和字符比较次数，最快速度是原算法的 4.4 倍以上，平均值达到了 2.6 倍。实验结果表明，改进后的算法具有更好的性能。

2 Snort 中的 BM 算法

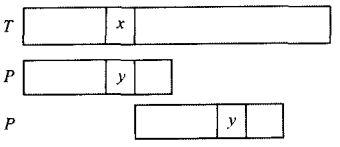
Snort 中的单模式匹配算法采用经典的 BM 算法，该算法由 Boyer 和 Moore[17]共同提出。BM 算法的基本思想为：首先令模式串和文本串左对齐，从模式串的右端开始向左匹配。当发生失配时，使用“坏字符”规则和“好后缀”规则决定模式串右移的距离，并且选择两个规则移动距离的较大值作为最终的移动距离。“坏字符”规则和“好后缀”规则分别如下。

“坏字符”规则：如果文本串 T 中的位 x 与模式串 P 中的位 y 不相同，则称发生了失配。发生失配时，分为两种情况：

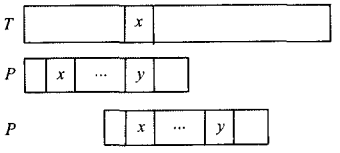
(1)如果模式串 P 中字符 y 的左边不存在字符 x，则此匹配窗口中的文本串不会与模式串 P 匹配。因此跳过字符 x，使得模式串 P 的首字符与字符 x 的下一位字符对齐；

(2)如果模式中 P 中字符 y 的左边存在字符 x，则将字符 y 左边与字符 x 相同的最右字符与文本串 T 中的 x 字符对齐。

上述两种情况分别如图 1(a)和图 1(b)所示。



(a)“坏字符”规则 1



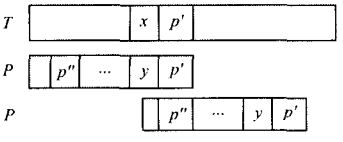
(b)“坏字符”规则 2

图 1 BM 算法“坏字符”规则

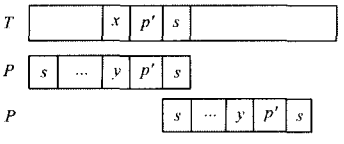
“好后缀”规则也分两种情况：

(1)如果模式串 P 中存在与已经匹配成功的字符串 p' 相匹配的子串，并且该子串是符合条件的最右子串，则移动模式串 P，使得该子串与已经匹配的字符串 p' 对齐；

(2)设已经匹配的子串 p' 的长度为 len，如果模式串 P 中不存在与 p' 匹配的子串，则查看 p' 的后 $len-1$ 位是否与模式串的前 $len-1$ 位相同，若相同，则将模式串 P 的前缀与之对齐，否则接着查看 p' 的后 $len-2$ 位是否与模式串的前 $len-2$ 位相同，以此类推。两种情况分别如图 2(a)和图 2(b)所示。



(a)“好后缀”规则 1



(b)“好后缀”规则 2

图 2 BM 算法的“好后缀”规则

“好后缀”移动函数的描述如下：

$$shift(j)=\min\{s|(P[j+1,j+2,\cdots,m-1])=P[j-s+1,\cdots,m-1-s]且(P[j]\neq P[j-s])(j>s),P[s+1,s+2,\cdots,m-1]=P[0,\cdots,m-1-s](j\leqslant s)\}$$

其中，$shift(j)$为模式串 P 的移动距离，j 为当前所匹配字符的位置，m 为模式串 P 的长度，s 为两部分匹配子串的距离。

BM 算法的最大移动距离为模式串长度 m，其匹配过程如表 1 所列。

表 1 BM 算法的匹配过程

文本串	s u b d a h w h u s u c r h c h a e h h k d e r s e a r c h
1	s e a r c h
2	s e a r c h
3	s e a r c h
4	s e a r c h
5	s e a r c h
6	s e a r c h
7	s e a r c h
8	s e a r c h

3 相关定理

在模式匹配中存在两个基本定理[18]，分别如下：

定理 1 任何字符串匹配算法在最坏情况下必须至少匹配 $n-m+1$ 个文本串字符。

定理 2 任何字符串匹配算法至少检查 n/m 个字符。

BM 算法在预处理阶段的时间复杂度为 $O(m+n)$，在搜索阶段最坏情况下的时间复杂度为 $O(m*n)$，最好情况的时间复杂度为 $O(n/m)$。由定理 1、定理 2 可知，没有算法比 BM 算法拥有更好的计算复杂度。但是，可以通过改进来提高最大移动距离和大的移动距离出现的概率。因此，本文的出发点是让文本串指针以更大的概率向前移动更大的距离，以此减少窗口匹配次数和字符比较次数，提高算法的平均匹配性能。

4 改进的 BM 算法

4.1 算法的基本思想

在 BM 算法的匹配过程中发生失配时需要向右移动模式串，进行下一轮的匹配。发生失配时，如何让模式串跳过不必要的字符匹配，增大模式串向右移动的距离是加快匹配速度、提高算法匹配性能的关键。此外，增加大的移动距离出现的概率对于算法匹配性能的提升也至关重要。文献[19]提到，对于 ASCII 匹配字符集而言，其总共包含 2^8 个字符，其中每一个字符出现的概率为 $p=1/2^8$，某字符出现在长度为 m 的模式串中的概率为 $1-(1-p)^m$，当 m 不太大时 $(1-(1-p)^m)\ll 1$。根据乘法原理，两个字符的组合在模式串中再次出现的概率更低。因此，利用了文本串中与模式串最右端对齐的字符的下一个及第二个字符，以及这两个字符再向右偏移模式串长度所对应的字符在模式串中的出现情况，根据出现情况决定移动距离，增大失配时模式串向右移动的最大距离；利用文本串中与模式串最右端对齐字符和最右端的下一个字符组合再次出现的小概率特性来增大失配出现的概率，从而减少字符匹配次数，缩短匹配时间，提高匹配效率。

综上所述，本文提出的 BM 模式匹配改进算法的基本思想如下：设文本串为 $T[0,1,\cdots,n-1]$，长度为 n，模式串为 $P[0,1,\cdots,m-1]$，长度为 m，$T[i]$ 为文本串中与模式串最右端对齐的字符。在匹配之前先对模式串进行预处理，统计模式串中各字符出现的次数，用于在匹配过程中判断某一个字符是否出现在模式串中。预处理函数的取值如下：

$$f(x)=\begin{cases}0, & \text{字符 } x \text{ 不出现在模式串中}\\ k, & \text{字符 } x \text{ 在模式串中出现 } k \text{ 次}(k\geqslant 1)\end{cases}$$

匹配时首先将文本串与模式串左端对齐，从模式串的右端开始进行匹配，当匹配到模式串的某位发生失配时，继续进行下一轮的匹配时 $T[i+1]$ 肯定在模式串与文本串的匹配窗口之内，因此可以分为以下两种情况进行处理。

情况 1　如果 $T[i+1]$ 不在模式串中，则证明 $T[i]T[i+1]$ 的组合不出现在模式串中，此时判断 $T[i+2]$ 是否等于模式串首字符 $P[0]$。

(1)若 $T[i+2]=P[0]$，则判断文本串中与模式串最右端对齐的字符的下一个字符再向右移动模式串长度所对应的字符是否在模式串中出现，即判断 $T[i+1+m]$ 是否出现在模式串中。若 $T[i+1+m]$ 出现在模式串中，则模式串向右移动 $m+1$ 位，否则模式串向右移动 $2m+1$ 位（模式串长度的 2 倍再加 1），移动后进行下一轮的匹配。因此，定义函数 $next(f(y))$，此函数用于计算字符 y 使模式串移动的距离，定义如下：

$$next(f(y))=\begin{cases}0, & \text{字符 } y \text{ 出现在模式串中}\\ m, & \text{字符 } y \text{ 不出现在模式串中}\end{cases}$$

(2)若 $T[i+2]\neq P[0]$，则判断文本串中与模式串最右端所对齐的字符向后数第二个字符，然后判断再向右移动模式串长度所对应的字符是否在模式串中出现，即判断 $T[i+2+m]$ 是否出现在模式串中。若 $T[i+2+m]$ 出现在模式串中，则模式串向右移动 $m+2$ 位，否则模式串向右移动 $2(m+1)$ 位（模式串长度的 2 倍加 1），移动后进行下一轮的匹配。

令 $T[i+2]=\alpha$，则情况 1 的模式串移动距离可以用函数 $skip1$ 表示，具体定义如下：

$$skip1(\alpha)=\begin{cases}m+1+next(f(T[i+1+m])), & T[i+1]\notin P \text{ 且 } \alpha=P[0]\\ m+2+next(f(T[i+2+m])), & T[i+1]\notin P \text{ 且 } \alpha\neq P[0]\end{cases}$$

情况 2　如果 $T[i+1]$ 出现在模式串中，则又分为以下两种情况。

(1)$T[i]T[i+1]$ 的组合不在模式串中

若 $T[i]T[i+1]$ 的组合不在模式串中，将模式串移动到任何与该组合整体重合的位置都不会匹配成功，此时判断 $T[i+1]$ 是否等于 $P[0]$。如果 $P[0]=T[i+1]$，则与第一种情况相似，继续判断与模式串最右端对齐的字符再向右偏移模式串长度所对应的文本字符是否在模式串中出现，即判断 $T[i+m]$ 是否在模式串中出现。如果出现，则将模式串移动到 $P[0]$ 与 $T[i+1]$ 对齐的位置，即模式串右移 m 位；否则模式串右移 $2m$ 位。如果 $P[0]\neq T[i+1]$，同理，判断 $T[i+1+m]$ 是否在模式串中出现，如果出现，则模式串右移 $m+1$ 位，否则模式串右移 $2m+1$ 位。

(2)$T[i]T[i+1]$ 的组合在模式串中

若 $T[i]T[i+1]$ 的组合在模式串中，则可能会出现多个这样的组合字符，计算出 $T[i]T[i+1]$ 与模式串中等于该组合的最右字符组合对齐时所移动的距离 d。

1)如果移动距离 $d>1$，则证明 $T[i+2]$ 肯定在匹配的窗口之内，因此先不移动模式串，而是判断 $T[i+2]$ 是否在模式串中。

①如果 $T[i+2]$ 不在模式串中，此时将模式串移动到任何模式串字符与 $T[i+2]$ 对齐的位置都不会匹配成功，因此可以直接右移 $m+2$ 位，跳过 $T[i+2]$ 以左的区域进行下一轮匹配。

②如果 $T[i+2]$ 在模式串中，则计算出 $T[i+2]$ 与模式串中等于 $T[i+2]$ 的最右字符对齐时所移动的距离 $d1$，模式串右移 $\max(d,d1)$ 位。

2)否则,模式串右移 d 位。

令 $T[i]T[i+1]=\beta$,则 $T[i]T[i+1]$的组合在模式串中的移动函数为

$$comb(\beta)=\begin{cases}m+2, & d=m-1-j>1,\\ & j=\max\{j\mid P_jP_{j+1}=\beta,0\leqslant j\leqslant m-1\}\\ & \text{且 } f(T[i+2])=0\\ \max(d,d1), & d>1,d1=m+1-l,\\ & l=\max\{l\mid P_l=T[i+2],0\leqslant j\leqslant m-1\}\\ & \text{且 } f(T[i+2])=k\geqslant 1\\ 1, & d=1\end{cases}$$

情况 2 的模式串移动距离可以用函数 $skip2$ 表示,具体定义如下:

$$skip2(\beta)=\begin{cases}m+next(f(T[i+m])), & \beta\notin P \text{ 且}\\ & T[i+1]=P[0]\\ m+1+next(f(T[i+1+m])), & \beta\notin P \text{ 且}\\ & T[i+1]\neq P[0]\\ comb(\beta), & \beta\in P\end{cases}$$

综合以上两种情况,改进算法的移动函数为:

$$dist(T[i+1])=\begin{cases}skip1(T[i+2]),\\ \quad f(T[i+1])=0,\text{即 } T[i+1]\text{不在模式串中}\\ skip2(\beta),\\ \quad f(T[i+1])=k\geqslant 1,\text{即 } T[i+1]\text{在模式串中}\end{cases}$$

利用改进后的算法进行匹配时,从文本串的 i 位置自右向左分别与模式串的 $P[n-1],P[n-2],\cdots,P[0]$相比较,当比较到某位发生失配时,令 $i=i+dist(T[i+1])$,重新进行新一轮匹配。

4.2 算法实例

改进后的算法仍以第 3 节中的文本串 subdahwhusucrhchaehhkdersearch 和模式串 search 为例,匹配过程如表 2 所列。

表 2 改进算法的匹配过程

文本串	s	u	b	d	a	h	w	h	u	s	u	c	r	h	c	h	a	e	h	h	k	d	e	r	s	e	a	r	c	h
1	s	e	a	r	c	h																								
2							s	e	a	r	c	h																		
3																									s	e	a	r	c	h
4																													s	e

(1)首先将文本串 T 与模式串 P 左对齐,从模式串的右端开始匹配,$m=6,i=5,T[5]=h=P[5]$,继续向左比较 $T[4]$与 $P[4],T[4]=a,P[4]=c,T[4]\neq P[4]$,匹配失败。此时 $T[i+1]=w$不在模式串中出现,并且$T[i+2]=h\neq P[0]=s$,需要判断 $T[i+2+m]$是否在模式串中,而 $T[i+2+m]=T[13]=h$ 在模式串中,因此模式串右移 $m+2=8$ 位,进行下一轮匹配。

(2)第二轮匹配时 $i=13$,同样是自右向左匹配,$T[13]=h=P[5],T[12]=r\neq P[4](P[4]=c)$,匹配失败。此时$T[i+1]=c$出现在模式串中,但是 $T[i]T[i+1]$的组合 hc 不在模式串中,并且 $T[i+1]=T[14]=c\neq P[0](P[0]=s)$,需要判断 $T[i+1+m]$是否在模式串中。$T[i+1+m]=T[20]=k$ 不在模式串中,因此直接跳过该区域,向右移动 $2m+1=13$ 位进行下一轮的匹配。

(3)第三轮匹配时 $i=26,T[26]=a\neq P[5](P[5]=h)$,匹配失败。$T[i+1]=T[27]=r$ 出现在模式串中,并且 $T[i]T[i+1]$的组合 ar 也在模式串中,模式串与该组合对齐时需要右移$d=3$ 位。$T[i+2]=T[28]=c$ 也在模式串中,模式串与该字符对齐时也需要右移 $d1=3$ 位,因此模式串右移 $\max(d,d1)=3$ 位,进行下一轮的匹配。

(4)第四轮匹配成功。

由表 2 的匹配过程可以看出,改进的算法只需要移动模式串 3 次,而表 1 中的 BM 算法的匹配过程需要移动模式串 7 次。改进算法的移动次数远小于 BM 算法的移动次数,匹配效率明显提高。

5 测试结果与分析

本文随机选取不同长度的字符串作为模式串,采用大小为 2.51MB 的纯英文小说文本作为文本串,共有 260 多万个字符,可以减小实验结果的测试误差。使用 C 语言实现算法程序,编译环境为 Visual C++ 6.0,在 Windows7 64 位操作系统、Intel 处理器、Core(TM) i5-4590 CPU、3.30GHz、4.00GB内存环境下分别对 Snort 中的 BM 算法和本文提出的改进算法进行测试。每个算法执行 1000 次,匹配所用的时间取平均值。模式串长度不同时窗口移动次数和匹配所用时间的结果对比分别如图 3、图 4 所示,改进算法的窗口移动次数和匹配消耗时间都远远小于原算法。通过大大减少窗口移动次数,减少了字符比较次数,加快了匹配速度。

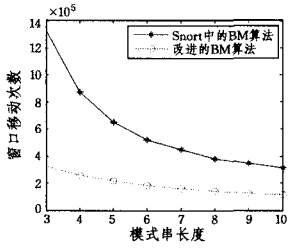


图 3 窗口移动次数对比

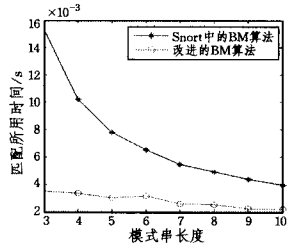


图 4 匹配所用时间对比

本文改进算法与 BM 算法的速度对比如表 3 所列,本文改进算法与其他 BM 改进算法的加速度对比如表 4 所列。表中,加速比=Snort 中 BM 算法所用时间/改进的 BM 算法所用时间。由对比结果可知,本文提出的改进算法的最高匹配速度是原算法的 4.4 倍以上,平均速度是原算法的 2.6 倍,匹配效率明显提高。

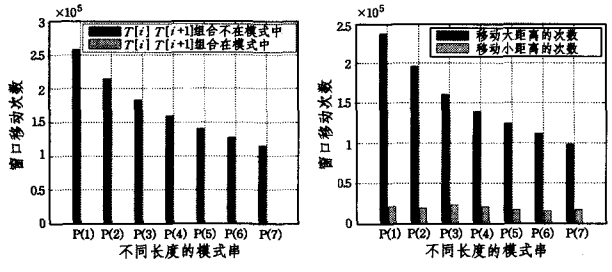
表 3 原算法与本文改进算法的速度对比

模式串长度	3	4	5	6	7	8	9	10	平均值
Snort 中的 BM 算法/ms	15.378	10.201	7.802	6.525	5.457	4.891	4.359	3.939	7.319
改进 BM 算法/ms	3.477	3.362	3.015	3.168	2.554	2.503	2.239	2.192	2.814
差值/ms	11.901	6.839	4.787	3.357	2.903	2.388	2.120	1.747	4.505
加速比	4.423	3.034	2.588	2.060	2.137	1.954	1.947	1.797	2.601

表 4　几种 BM 改进算法的加速比对比

模式串长度	BMH 算法的加速比	BMHS 算法的加速比	文献[6]算法的加速比	文献[12]算法的加速比	本文算法的加速比
3	0.970	1.300	—	—	4.423
4	1.070	1.230	—	—	3.034
5	1.040	1.190	—	1.447	2.588
6	1.029	1.160	—	—	2.060
7	1.057	1.140	—	—	2.137
8	1.080	1.120	—	—	1.954
9	1.024	1.110	—	—	1.947
10	0.929	1.100	—	—	1.797
15	—	1.080	—	1.493	1.792
最大加速比	1.080	1.300	1.500	—	4.423

另外，改进算法的最大移动距离为 $2m+2$，较大的移动距离分别为 $2m+1$ 和 $2m$。将这 3 个移动距离称为大距离，其他情况下的移动距离称为小距离。在改进算法中，大移动距离出现在 $T[i]T[i+1]$的组合不在模式串中出现的情况中，而小移动距离出现在 $T[i]T[i+1]$的组合出现在模式串中的情况中。为了验证本文算法利用 $T[i]T[i+1]$的组合概率特性来增大移动大距离出现的概率的正确性，随机选取字符串 subqkdjsum，分别以其不同长度的前缀 subq，subqk，subqkd，subqkdj，subqkdjs，subqkdjsu，subqkdjsum 作为模式串，对应编号为 P(1)－P(7)。在上述 2.51MB 纯文本中进行匹配，统计 $T[i]T[i+1]$组合出现和未出现在模式串中的窗口移动次数及窗口移动大距离和小距离的次数，结果如图 5 所示。由结果可以看出，$T[i]T[i+1]$的组合出现在模式串中的概率非常小，而相应的窗口移动小距离的次数也远远小于移动大距离的次数，实验结果与第 4 节的分析相吻合。因此，利用 $T[i]T[i+1]$的组合出现的小概率性增大了移动大距离出现的次数。



(a) $T[i]T[i+1]$的组合在模式串中的出现情况　　(b)(a)中对应情况下移动大、小距离出现次数对比

图 5　改进算法中 $T[i]T[i+1]$组合串对移动大距离的影响

本文提出的改进算法在最好情况下的时间复杂度接近亚线性，最坏情况下的时间复杂度为 $O(m(n-m))$。由于改进算法增大了最大移动距离，并且以 4.1 节的理论分析结论为指导，利用 $T[i]T[i+1]$的组合概率特性增大了移动大距离出现的概率，使得绝大多数情况下都是以 $2m+2$，$2m+1$ 或 $2m$ 的距离移动，因此本文算法的平均时间复杂度优于 BM 算法及其他改进算法。

由以上实验结果可知，改进算法减少了窗口移动次数和字符比较次数，加快了匹配速度，提升了原算法的平均匹配性能，将其应用到网络入侵检测系统中可以提高检测效率。

结束语　本文对 BM 算法进行了深入的研究。针对 BM 算法在失配时模式串移动距离较小、匹配速度较慢的问题，本文提出了 BM 算法的改进算法。改进算法增大了移动的最大距离，最大移动距离是 BM 算法的 2 倍以上。利用最右端字符与其下一字符的组合概率特性，增大了失配时大的移动距离出现的概率。实验结果表明，该算法有效地减少了窗口移动次数和字符比较次数，加快了匹配速度，提高了平均匹配效率。将改进算法运用到网络入侵检测系统中将会大大提高检测效率，对解决当前 Snort 入侵检测系统面临的问题具有重要的意义。此外，基于模式匹配的入侵检测是一种精确字符匹配，不能满足检测隐蔽性入侵行为的需要，且该方法无法发现新的入侵攻击行为[20]，因此将数据挖掘理论运用到其中来提高检测未知入侵攻击的能力将是下一步的研究内容。

参 考 文 献

[1] KURUNDKAR G D, NAIK N A, KHAMITKAR S D. Network intrusion detection using Snort[J]. International Journal of Engineering Research and Applications, 2012, 2(2): 1288-1296.

[2] OUYANG W L, TOMBARI F, MATTOCCIA S, et al. Performance evaluation of full search equivalent pattern matching algorithms[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2012, 34(1): 127-143.

[3] HORSPOOL N R. Practical fast searching in strings [J]. Software Practice and Experience, 1980, 10(6): 501-506.

[4] SUNDAY D M. A very fast substring search algorithm[J]. Communications of the ACM, 1990, 33(8): 132-142.

[5] MU Y M, LI M G, LIANG Q. The survey of the pattern matching algorithm in intrusion detection system[J]. ACTA Electronica Sinica, 2006, 34(S1): 2488-2490. (in Chinese)
牟永敏，李美贵，梁琦. 入侵检测系统中模式匹配算法的研究[J]. 电子学报，2006，34(S1)：2488-2490.

[6] QIAO J X, ZHANG H. Improvement of BM algorithm in intrusion detection system[C]// IEEE International Conference on Software Engineering and Service Science. Beijing, IEEE, 2015: 652-655.

[7] FAN H B, YAO N M. A fast and exact single pattern matching algorithm[J]. Journal of Computer Research and Development, 2009, 46(8): 1341-1348. (in Chinese)
范洪博，姚念民. 一种高速精确单模式串匹配算法[J]. 计算机研究与发展，2009，46(8)：1341-1348.

[8] SIMONE F, OĞUZHAN KÜLEKCI M. Fast and flexible packed string matching[J]. Journal of Discrete Algorithms, 2014, 28: 61-72.

[9] YUN S. An efficient TCAM-based implementation of multi-pattern matching using covered state encoding[J]. IEEE Transactions on Computers, 2012, 61(2): 213-221.

[10] LIN C H, CHANG S C. Efficient pattern matching algorithm for memory architecture [J]. IEEE Transactions on Very Large Scale Integration Systems, 2011, 19(1): 33-41.

[11] ERDEM O. Tree-based string pattern matching on FPGAs[J]. Computers and Electrical Engineering, 2016, 49: 117-133.

[12] SUN W J, QIAN H. Improved BM algorithm and its application in network intrusion detection[J]. Computer Science, 2013, 40(12): 174-176. (in Chinese)
孙文静，钱华. 改进 BM 算法及其在网络入侵检测中的应用[J]. 计算机科学，2013，40(12)：174-176.

[13] MA Z F, YANG S Y, GUO G F. A fast improved pattern matching algorithm based on BM[J]. Control and Decision, 2013, 28

(12):1855-1859. (in Chinese)
马占飞,杨树英,郭广丰. 一种快速的基于 BM 模式匹配的改进算法[J]. 控制与决策,2013,28(12):1855-1859.

[14] CHO S,NA J C,PARK K,et al. A fast algorithm for order-preserving pattern matching[J]. Information Processing Letters,2015,115(2):397-402.

[15] NSIRA N B,LECROQ T,ELLOUMI M. A fast Boyer-Moore type pattern matching algorithm for highly similar sequences [J]. International Journal of Data Mining and Bioinformatics,2015,13(3):266-288.

[16] ZHANG H L,XU D L,LIANG M,et al. Massive string efficient matching method research[J]. Acta Electronica Sinica,2014,42(6):1220-1224. (in Chinese)
张宏莉,徐东亮,梁敏,等. 海量模式高效匹配方法研究[J]. 电子学报,2014,42(6):1220-1224.

[17] BOYER R S,MOORE J S. A fast string searching algorithm [J]. Communications of the ACM,1977,20(10):762-772.

[18] LI X B,LIU B X,XU R S. Research of string matching techniques[J]. Computer Engineering,2004,30(22):24-26. (in Chinese)
李雪宝,刘宝旭,许榕生. 字符串匹配技术研究[J]. 计算机工程,2004,30(22):24-26.

[19] LIU G S,WANG Y C,XU H Q. A single pattern matching algorithm based on character frequency[J]. Acta Electronica Sinica,2002,30(12A):2079-2082. (in Chinese)
刘功申,王永成,许欢庆. 基于字频的单模式匹配算法[J]. 电子学报,2002,30(12A):2079-2082.

[20] AHMED M,MAHMOOD A N,HU J K. A survey of network anomaly detection techniques[J]. Journal of Network and Computer Applications,2016,60:19-31.

(上接第 109 页)

平台对 3D-LE-LPCCA 算法的节点能耗进行了仿真实验,并与基于 RSSI 测距的多边测量法进行了对比。

在相同的网络参数下,节点数目为 20～240 的不同网络规模中,上述两种算法的数据包平均能耗如图 8 所示。可以看出,3D-LE-LPCCA 算法的数据包平均能耗明显低于一般的基于 RSSI 测距的多边测量算法的能耗。这是由于 3D-LE-LPCCA 算法在临近节点集的求解中采用跳距约束,若一个数据包经过 4 次转发仍未到达未知节点,则该消息不再转发,即信标节点只转发跳距约束以内的数据包。因此,网络中转发和接收的数据包减少,节点平均能耗也相应降低。

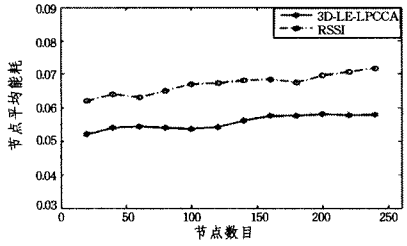


图 8 不同节点数目下两种定位算法的平均能耗数据对比

综上所述,本文提出的 3D-LE-LPCCA 定位算法在定位误差率、定位覆盖率上均优于 3D-IDCP、ILAH-3D 和最小二乘定位法。相比于一般的基于 RSSI 测距的定位算法,3D-LE-LPCCA算法的平均节点能耗较小。实验结果表明,3D-LE-LPCCA定位算法能提高三维空间定位算法的定位精度和稳定性,降低节点能耗。

结束语 本文在 LPCCA 模型的基础上构造三维定位算法 3D-LE-LPCCA,该算法具有定位精度高、稳定性好以及能耗小的优点。由于现实定位环境错综复杂,难免有障碍物对节点定位产生影响,因此对复杂环境下的三维定位算法的研究将是下一步研究的重点。

参 考 文 献

[1] HAN G,XU H,DUONG T,et al. Localization Algorithms of Wireless Sensor Networks: a survey[J]. Telecommunication Systems,2013,52(4):2419-2436.

[2] KHELIFI M,MOUSSAOUI S,SILMI S,et al. Localisation Algorithms for wireless sensor networks: a review[J]. International Journal of Sensor Networks,2015,19(2):114-129.

[3] ZHAO F,LUO H,GENG H,et al. An RSSI Gradient-based AP Localization Algorithm[J]. China Communications,2014,11(2):100-108.

[4] SAHU P K,WU E H-K,SAHOO J. DuRT: Dual RSSI Trend Based Localization for Wireless Sensor Networks[J]. IEEE Sensors Journal,2013,13(8):3115-3123.

[5] WANG S,LI Y. Node Localization Algorithm Based on RSSI in Wireless Sensor Network[C]// 2012 6th International Conference on Proceedings of the Signal Processing and Communication Systems (ICSPCS). IEEE,2012.

[6] LUO Q,PENG Y,LI J,et al. RSSI-Based Localization Through Uncertain Data Mapping for Wireless Sensor Networks[J]. IEEE Sensors Journal,2016,16(9):3155-3162.

[7] BISHOP E. Differentiable Manifolds in Complex Euclidean space [M]// Selected Papers of Errett Bishop,1986:363-383.

[8] GU J J,CHEN S C,ZHUANG Y. Localization in Wireless Sensor Network Using Locality Preserving Canonical Correlation Analysis[J]. Journal of Software,2010,21(11):2883-2891. (in Chinese)
顾晶晶,陈松灿,庄毅. 用局部保持典型相关分析定位无线传感器网络节点[J]. 软件学报,2010,21(11):2883-2891.

[9] MAO K J,ZHAO X M,SHAO B,et al. Three Dimensional Localization Algorithm Based on Degree of Coplanarity for Wireless Sensor Networks[J]. Chinese Journal of Sensors and Actuators,2011,24(10):1484-1488. (in Chinese)
毛科技,赵小敏,邵奔,等. 无线传感网络中基于共面度的三维定位算法研究与设计[J]. 传感技术学报,2011,24(10):1484-1488.

[10] SONG G,TAM D,LIAO D,et al. 3D Localization Algorithm for Wireless Sensor Networks Based on DCP and VRT [M]// Embedded System Technology. Springer:58-67.

[11] XU Y,ZHUANG Y,GU J. An improved 3d localization algorithm for the wireless sensor network [J]. International Journal of Distributed Sensor Networks,2015,2015(98).

[12] QI R B,LI S J,MA T Y,et al. Iteration-Based Localization Algorithm for Wireless Sensor Network in Three-Dimensional Space[J]. Chinnese Journal of Sensors and Actuators,2012,25(5):644-650. (in Chinese)
祁荣宾,李思瑾,马天义,等. 基于迭代的无线传感器网络三维定位算法[J]. 传感技术学报,2012,25(5):644-650.