基于爬虫的智能爬行算法研究

侯美静 崔艳鹏 胡建伟
(西安电子科技大学网络与信息安全学院 陕西 西安 710000)

摘 要 为了提高漏洞巡检的效率,过滤掉大部分结构相似的网页,提出一种智能爬行算法。对爬取过程中URL去重,丢弃重复的URL;计算两个URL对应页面的相似度值,具体是将页面解析成DOM树,根据节点的位置、DOM树的深度以及深度相同的节点数量,将权重分配给每个节点,再根据给定的公式计算网页的相似度;以相似度为基础,使用聚合式层次聚类思想将具有相似结构的网页聚为一组,每组只留下一个网页,达到去除大部分结构相似的网页的目的。实验结果表明,该智能爬行算法可以有效地减少结构相似的网页,提高漏洞巡检系统的巡检效率。

关键词 爬虫 智能爬行算法 URL去重 相似度 聚类

中图分类号 TP3 **文献标识码** A DOI:10.3969/j.issn.1000-386x.2018.11.037

INTELLIGENT CRAWLING ALGORITHM BASED ON CRAWLER

Hou Meijing Cui Yanpeng Hu Jianwei
(*School of Network and Information Security, Xidian University, Xi'an 710000, Shaanxi, China*)

Abstract An intelligent crawling algorithm was proposed to improve the efficiency of vulnerability inspection and filter out most of the structurally similar web pages. URL was deduplicated in the crawling process, and reduplicate URLs were discarded. We calculated the similarity values of the pages corresponding to the two URLs. The page was parsed into a DOM tree. According to the position of the node, the depth of the DOM tree and the number of nodes with the same depth, the weight was assigned to each node, and the similarity of the Web page was calculated on the basis of the given formula. By using aggregated hierarchical clustering thoughts based on similarity, we grouped the Web pages with similar structures, and only one Web page was left in each group, thus achieving the goal of removing most of the structurally similar Web pages. The experimental results show that the intelligent crawling algorithm can effectively reduce the similarity of Web pages and improve the inspection efficiency of the vulnerability inspection system.

Keywords Crawler Intelligent crawling algorithm URL deduplication Similarity Clustering

0 引 言

基于Web系统的多层体系结构以及与其他不同类型的子系统之间的复杂交互,增加了可以被攻击者利用的安全漏洞的数量。正如开放式Web应用程序安全项目OWASP(Open Web Application Security Project)[1]所强调的那样,攻击者可能会沿着基础结构跟踪各种路径,以发现可能会导致严重后果的安全漏洞。我们就需要对Web应用执行例行巡检,采取必要的行动降低安全风险。软件开发人员通常使用Web应用巡检系统来自动检查其基于Web应用程序的安全性,并对许多不同的Web应用程序进行大规模安全分析[2-3]。

在Web应用巡检系统里,爬虫是按照一定的爬行策略,做站内的URL爬取,从而获得目标网站下所有的可视或不可视的URL信息。巡检系统对于所有的URL进行安全审计,但是对于同源网站来说,存在大量结构相似的URL和网页,对于此类URL和网页,重复地审计不会发现新的漏洞。现在的爬虫技术多采用URL相似性去重,雷佳伟[4]通过研究Scrapy爬虫框架,分析页面爬取及解析的过程,并研究了URL相似

收稿日期:2018-03-31。**侯美静**,硕士生,主研领域:网络安全。**崔艳鹏**,副教授。**胡建伟**,副教授。

性去重,但是这种方法会将大量的非相似的网页去重,失误率过高。贺建英[5]针对搜索引擎提出了一种基于 Rabin 指纹算法进行 URL 去重、网页内容相似和相同去重,将网页文档进行定长分块,然后用 Rabin 指纹算法计算每个网页块的指纹,通过比对两个网页分块指纹的重复率来判定网页是否相似,这种方法只适用于网页结构绝对类似,网页内容相同的的情况。而其他人[6-7]关注更多的是爬虫的覆盖率,只是进行 URL 去重,并没有考虑网页相似的问题,而对参数进行模糊测试的方法查找漏洞。开发人员尽量避免在 URL 里显示明显的参数,所以已有的技术不能满足 Web 应用巡检系统的需求。

针对以上文献中的不足,本文针对智能巡检系统提出了一种基于 URL 去重和以网页相似度为基础的聚合式层次聚类的智能爬行算法,能有效地去除重复的 URL 和大量结构相似的网页,最大程度缩小巡检系统的测试目标,提高检测效率。

1 智能爬行策略

随着 Web 应用的发展,爬虫技术也在改进。从早期的通用爬虫到现在的主题爬虫[8]、深度爬虫[9]等等,满足了不同用户的不同需求。

通用爬虫的爬取过程比较简单,一般是从初始的 URL 开始爬取网页,将获得的 URL 放入待爬取的列表里,等初始页面爬完,再从待爬取列表里取出新的 URL 开始爬取,依次循环,直到待爬取列表为空或者满足设置的停止条件。这种方法简单,比较常用,但是这类爬虫只是针对 URL 进行单一爬取,不能满足用户额外的需求。

深度爬虫相对于通用爬虫来说比较复杂,对于获得的页面不光提取 URL,还会进行解析,对 URL 进行去重,提高了爬行效率避免陷入死循环。

在深度爬虫的基础上,本文提出了一种智能爬虫策略。智能爬虫的爬取过程是:以一个初始 URL 为起点爬取相关的所有网页,然后利用智能爬行算法爬取网站,最后得到无重复的 URL,且 URL 对应的网页结构都是不相似的。本文所提出的智能爬行算法,首先对 URL 去重丢弃重复的 URL。下一步利用页面相似度公式依次计算两个 URL 对应页面的相似度值,具体是将页面解析成 DOM 树,根据节点的位置、DOM 树的深度以及深度相同的节点数量,将权重分配给每个节点,再根据给定的公式计算网页的相似度。最后以相似度为基础,使用聚合式层次聚类思想将具有相似结构的网页聚为一组,只选取代表 URL 进行后续测试。

本文提出的智能爬行算法的计算过程分三个阶段,如图 1 所示。第一阶段需要对 URL 去重;第二阶段对网页解析并计算网页相似度;第三阶段将相似度满足设定阈值的网页进行聚类,并从每一类中选取一个 URL 作为代表进行后续检测。整个计算过程称为智能爬行算法。

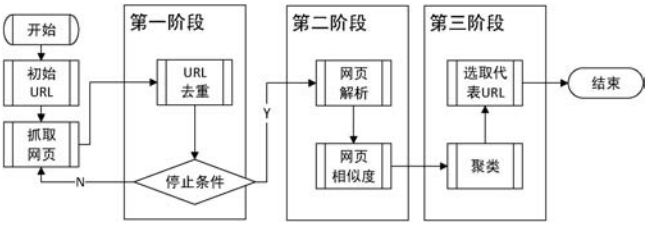

图 1 智能爬行算法图

2 智能爬行算法

2.1 URL 去重

URL 又称统一资源定位符,是对互联网上资源的位置和访问方法的一种简洁的表示,是互联网上标准资源的地址[10]。它可以理解为是互联网上资源的索引,通过它爬虫就可以找到新的资源,获取新的 URL。但是并非所有的 URL 都是可用的,对于重复的、相似的、存在包含关系的 URL 要进行过滤,可以减少重复爬取的时间,提高智能巡检系统的整体效率。

URL 的完整格式(其中带[ ]的为可选项)如图 2 所示。图 3 为 URL 分片对应图。

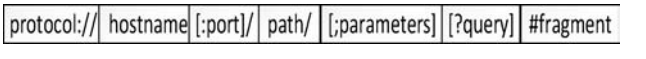

图 2 URL 格式图

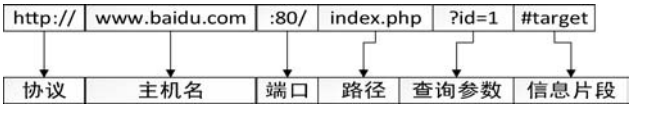

图 3 URL 分片对应图

URL 重复:两个 URL 的协议、主机名、端口、路径、参数名和参数值完全一样。

URL 相似:两个 URL 的协议、主机名、端口、路径、参数名和参数个数都相同。

URL 包含:两个 URL 记为 A 和 B,协议、主机名、端口、路径都相同。若 A 的参数个数大于或等于 B,且 B 的参数名列表与 A 的参数名列表存在包含关系。

在 HTML 中,URL 主要会在下列常见的标签中产生,可以通过其相关的属性值来获取,如表 1 所示。

表 1　存在 URL 的常见标签及属性

标签	属性	示例
a, link	href	< a href = "test. html" >
script, img	src	< script src = "test. js" > </script >
form	action	< form action = "test. php" > </form >
object	archive, codebase, data 和 usemap	< object data = "test. swf" > </object >
⋮	⋮	⋮

爬虫在爬取网页的过程中,总会出现很多重复的 URL。重复的爬取不会发现新的 URL 链接,只会浪费计算资源和延长爬行时间甚至使爬行陷入死循环。因此,需要过滤这类 URL,减少重复爬取。智能爬行算法的第一阶段就是基于 Rabin 指纹算法[4] 对 URL 去重。基本步骤如下:

(1) 输入一个 URL 值作为参数。

(2) 构造一个初始值为 0,长度为 n 的列表 L 用于存放指纹映射。再构造一个空列表 U,用于存放爬取过的 URL。

(3) 对该 URL 爬取出的所有 url 进行循环,循环包括:计算每个 url 的指纹值,将指纹值 r 映射到列表 L 中。判断列表中 L[r]的值是否为 0,若为 0,则将 L[r]置为 1,并将此 url 存放入列表 U 中。若值为 1,则舍弃。

2.2　页面相似度

对于同源网页,大量的网页结构是相似的,只是少量内容不同。在进行漏洞巡检时,会对每一个 URL 进行渗透测试,这样会做大量的无用工作,浪费时间。也有人通过比对 URL 的相似度,去除相似的 URL,但是这种方式误差太大,可用性不强。本文提出的智能爬行算法通过比对网页结构的相似度,当相似度达到设定阈值后,才判定为相似,这种方法对于网页去重更精确,更具可行性。

智能爬行算法的第二阶段就是计算网页的相似度。这里采用一种基于平均分配权重的网页结构相似度测量方法。首先对网页构造 DOM 树[11] 结构并预处理,将 DOM 树下的标签作为树的节点,对于文本内容则统一用 text 表示,只比较两个页面的结构相似度。再将权重平均分配给节点,具体步骤如下:

① 令整个 DOM 树的权重为 1,对于深度为 1 的根节点,权重为 1;对于深度为 2 的根的子节点,若数量为 n,则每个子节点的权重为 1/n。

② 将深度为 2 的节点的权重平均分给它的子节点。

③ 迭代分配,直到树的叶子节点。

④ 对于叶子节点 a 和 b,如果 a 等于 b,那么 a 和 b 的相似度就是它们所分配的权重,若不等于,那么相似度为 0。对于非叶子节点 a 和 b,如果 a 等于 b,那么 a 和 b 的相似度就是它们的子节点的权重总和,若不等于,那么相似度为 0。

⑤ 两棵 DOM 树的相似度就是它们根节点的相似度。

具体的计算过程,以两个简单的 HTML 页面为例,两个 HTML 页面的源代码如图 4 所示。

```
<html>                                  <HTML>
<head>                                  <HEAD>
<title>Your page title here</title>     <TITLE>Your Title Here</TITLE>
</head>                                 </HEAD>
<body>                                  <BODY>
<h1>Your major heading here</h1>        <CENTER><IMG SRC="clouds.jpg" ALIGN="BOTTOM" ></CENTER>
<a><IMG SRC="aaa.jpg" ALIGN="BOTTOM"></a>  <a href="http://baidu.com">Link Name</a>
<p>This is a regular text paragraph.</p>  www.baidu.com
<ul>                                    <H1>This is a Header</H1>
<li>First</li>                          <H2>This is a Medium Header</H2>
<li>Second</li>                         <P> This is a new paragraph!
<li>                                    <P> <B>This is a new paragraph!</B>
This is the <em>Third</em> bullet.      </BODY>
</li>                                   </HTML>
</ul>
</body>
</html>
```

图 4　两个 HTML 页面源码图

首先构建两个页面对应的 DOM 树,DOM 树的节点就是 HTML 页面的标签,文本内容统一用 text 表示,如图 5 所示。

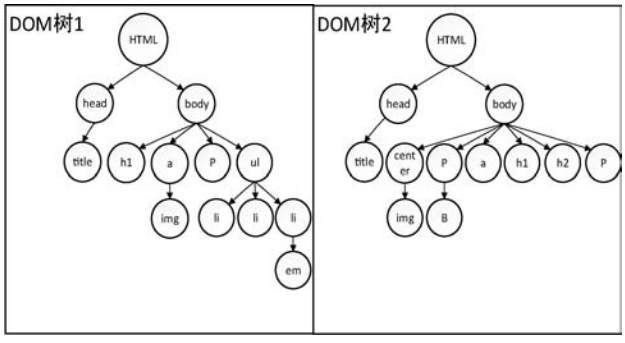

图 5　两个 HTML 页面的 DOM 树图

整个 DOM 树的权重设为 1,从根节点 HTML 开始依次平均下发到叶子节点,整棵树的权重分配如图 6 所示。

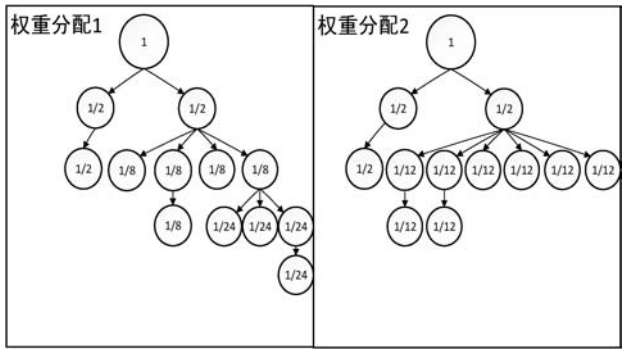

图 6　两棵 DOM 树的权重分配图

根节点的权重为 1,根节点下有两个子节点,平均分配,每个子节点的权重为 1/2。依次循环,直到整棵

树分配完毕,可以看出,所有叶子节点的权重相加等于1,所以,我们通过比较同一深度叶子节点的相似度,得到父节点的权重,即:所有相同的子节点的权重总和。但是如果父节点不相同的话,权重则为0。如 DOM 树1中,a 标签是 img 标签的父节点;DOM 树 2 中 center 标签是 img 标签的父节点,虽然两个叶子节点一样,但是父节点是不一样的,所以 a 标签和 center 标签的相似度为0。

计算过程可以描述为:

$$\text{similar}(D_1, D_2) = \text{similar}(D_1X_{11}, D_2X_{11})$$

$$\text{similar}(D_1X_{nm}, D_2X_{nm}) = \rho \times \left[\sum_{i=1}^{Num(n+1)} \text{similar}(D_1X_{n+1,i}, D_2X_{n+1,i})\right]$$

$$\rho = \begin{cases} 1 & D_1X_{nm} = D_2X_{nm} \\ 0 & \text{other} \end{cases}$$

式中:D 是 DOM 树,$X_{11}, X_{21}, X_{22}, \cdots, X_{nm}$是树的节点,$n$ 是树的深度,根节点的深度是 1,X_{11}代表根节点,m 是节点的序列号,X_{nm}代表深度为 n 的第 m 个节点,D_iX_{nm}代表第 i 个树的 X_{nm}节点,$Num(n)$代表深度为 n 的节点的数量。

2.3 聚 类

智能爬行算法的第三阶段就是以网页相似度为基础,利用聚合式层次聚类思想,将结构相似的网页聚为一组,取出一个作为代表 URL。

因为聚类的集群数量不确定,聚类的密度也高,只确定相似性的阈值大小,所以该算法对类似于K-means等需要在开始就确定集群数量的算法不太适用;对于基于密度的方法同样需要在开始时设置两个参数(半径和最小数量),但我们无法准确地对两个参数进行设置;基于网格的方法通常适用于多维数据;基于模型的方法则需要根据数据集的特征构建模型。综合以上问题,本文选择基于层次的聚类思想实现网页聚类。层次聚类有两种思想[12],一种是所有的文档为一个类,之后去做切分,每次可能就会多一个类,叫做分割式聚类;另一种是聚合式聚类,开始每个数据都单独成一类,之后根据相似度去比对阈值,然后聚类。

利用聚合式层次聚类的思想,以上面的网页相似度为基础,将相似度大于阈值的网页所对应的 URL 放在一个子集中;通过一系列的计算得到若干个子集;再从每个子集中取出一个作为代表 URL。

算法步骤如下:

第一步:聚类的对象是去重后的 URL 列表,设定一个初始相似度阈值 $T1$。

第二步:在列表中随机选择一个 URL 作为初始点,并计算初始点与列表中其他 URL 的相似度。

第三步:若网页之间的相似度大于阈值 $T1$,则将两个网页划分到一个子集中,并将此网页的 URL 在列表中置为0;若距离小于阈值 $T1$,则继续从列表中取值,直到此网页和剩余非0网页比较结束。

第四步:重复第二步和第三步,直到列表为空。

第五步:取代表 URL。

3 测 试

3.1 实验环境

智能爬行策略实验是在 kali 虚拟机、内存 2 GB 的硬件环境下进行的,使用 Python 语言实现。

3.2 实验内容

利用现有技术的深度爬虫和本文的智能爬虫分别对网站进行爬取。

3.3 爬取结果分析

分别对三个网站进行了测试,爬取结果如表 2 所示。

表 2 深度爬虫和智能爬虫对比结果

网站名称	深度爬虫/个	智能爬虫/个
https://www.shiyanlou.com	1 764	122
http://www.freebuf.com	139 460	2 367
https://www.douban.com	138 655	3 856

用深度爬虫对 http://www.freebuf.com 网站进行爬取,结果可达 14 万个,而智能爬虫的爬取结果只有 2 367 个,聚类效果明显,可大量节省 Web 应用巡检系统的审计时间。其他网站类似,对 https://www.douban.com 网站,深度爬虫爬取结果为 138 655 个,而智能爬虫的爬取结果为 3 856。对于 https://www.shiyanlou.com 网站,深度爬虫的爬取结果为 1 764,智能爬虫的爬取结果为 122。

对于 http://www.freebuf.com 网站的深度爬虫爬取结果进行分析,如表 3 所示。

表 3 http://www.freebuf.com 网站深度爬虫爬取结果分析

类似的 URL	数量/个
http://www.freebuf.com/jobs/ddd.html	8 026
http://www.freebuf.com/articles/xxx/ddd.html	30 156
http://www.freebuf.com/news/ddd.html	33 907
http://www.freebuf.com/? p=ddd	7 154

对于 https://www.douban.com 网站的深度爬虫爬取结果进行分析,如表 4 所示。

表 4　https://www.douban.com 网站深度爬虫爬取结果分析

类似的 URL	数量/个
https://www.douban.com/group/xxx	51 232
https://www.douban.com/event/ddd	4 318
https://www.douban.com/location/xxx	10 429
https://www.douban.com/people/xxx	38 345

由表 3、表 4 可知类似的 URL 有很多,它们对应的网页大多是结构相似的。而智能爬虫可以有效地将这些结构相似的网页进行聚类,减少后续巡检系统的测试工作。

为了更清晰地展示两种爬虫爬取的结果,我们展示部分爬取结果,图 7 是深度爬虫的部分爬取结果图。图 8 是智能爬虫的部分结果图。

```
1 http://www.freebuf.com/articles/terminal/100601.html
2 http://www.freebuf.com/articles/terminal/100581.html
3 http://www.freebuf.com/articles/terminal/100468.html
4 http://www.freebuf.com/author/Thatqier
5 http://www.freebuf.com/author/Thatboy
6 http://www.freebuf.com/author/Thanos
7 http://www.freebuf.com/sectool/5955.html
8 http://www.freebuf.com/sectool/5950.html
```

图 7　深度爬虫部分爬取结果图

```
1 http://www.freebuf.com/articles/terminal/100601.html
2 http://www.freebuf.com/author/Thatqier
```

图 8　智能爬虫部分爬取结果图

表 5 是智能爬行算法在聚类过程中比对的相似度值和聚类过程。

表 5　智能爬行算法聚类过程分析

序号	相似度值	对比的 URL	聚类结果
1	0.874 170 274 17	URL1 和 URL2	URL1、2、3、7、8 聚为一组
2	0.806 601 890 057	URL1 和 URL3	
3	0.701 784 197 111	URL1 和 URL4	
4	0.7	URL1 和 URL5	
5	0.691 088 260 497	URL1 和 URL6	
6	0.973 797 678 275	URL1 和 URL7	
7	0.928 235 662 802	URL1 和 URL8	
8	0.997 587 454 765	URL4 和 URL5	URL4、5、6 聚为一组
9	0.981 729 598 051	URL4 和 URL6	

从表中可以看出,智能爬行算法在聚类过程中,先随机选择一个 URL(表中为 URL1),与剩余的其他 URL 进行相似度比对,若相似度大于阈值(阈值 = 0.8)则聚为一组,表中 URL1、URL2、URL3、URL7、URL8 聚为一组。剩余的 URL4、URL5、URL6 再重复上面的步骤,直到所有 URL 都完成分组,表中 URL4、URL5、URL6 聚为一组。智能爬行算法最后会在每组中取一个代表 URL,图 8 就是爬行的结果。

从图 7 和图 8 的对比可以看出,深度爬虫会将所有的网页爬取出来,存在很多结构相似的网页。而智能爬虫可以有效地将结构相似的网页聚为一类,只取代表 URL 进行后续检测,极大地提高了巡检系统的效率。

4　结　语

本文提出了一种智能爬行算法,在爬行过程中采用基于 Rabin 指纹算法对 URL 去重,检索速度快,效率高,且避免爬虫在爬行过程中浪费不必要的时间,甚至陷入死循环;在计算网页相似度前对网页进行预处理,构建 DOM 树,并统一用 text 表示文本,只保留网页标签,简化了网页,并通过平均分配权重的方法计算网页结构相似度,使网页相似度计算更高效;采用聚合式层次聚类算法将相似性网页聚为一组,提高了巡检系统的效率。本文解决了网页结构大量相似的问题,提高了安全巡检的效率,而且本方法思路简单、易于实现、通用性强,对结构相似的网页识别率高,提高了爬虫的精确度。

参　考　文　献

[1] OWASP: the ten most critical web application security risks [R]. The Open Web Application Security Project,2013.

[2] Thakur K, Qiu M, Gai K, et al. An Investigation on Cyber Security Threats and Security Models[C]//2015 IEEE 2nd International Conference on Cyber Security and Cloud Computing(CSCloud). IEEE, 2015:307-311.

[3] Doupé A, Cavedon L, Kruegel C, et al. Enemy of the State: A State-Aware Black-Box Vulnerability Scanner[C]//Proceedings of the 21st USENIX conference on Security symposium. USENIX Association, 2012.

[4] 雷佳伟. 基于爬虫技术的 Web 应用程序漏洞检测方法[D]. 北京:北京工业大学,2016.

[5] 贺建英. Rabin 指纹去重算法在搜索引擎中的应用[J]. 计算机系统应用,2015,24(7):128-131.

[6] 张烨青. Web 应用安全漏洞扫描器爬虫技术的改进与实现[D]. 北京:北京邮电大学,2014.

[7] 王雪贞. 基于 Fuzzing 的 SQL 注入漏洞检测系统研究与实现[D]. 大连:大连海事大学,2017.

[8] 胡萍瑞,李石君. 基于 URL 模式集的主题爬虫[J]. 计算机应用研究,2018,35(3):694-699,726.

(下转第 277 页)

围绕预测结果上下浮动,基本上预测结果在训练集的中心部位,没有出现过拟合或梯度消失等现象。虽然只是用的简单的三层黄金分割比例的 BP 神经网络,可是对于 MNIST 数据集我们的正确率已经达到了 97.55%,比人眼识别率还要高,可见黄金比例运用到 BP 网络中效果特别好。

4　结　语

本文在基于 NAG 算法的基础上,引入了黄金分割比,有机地结合了自然美学和工程应用,在泛化能力和收敛速度上,都得到了一定程度的提高。应用实例表明,基于黄金分割比梯度动量更新策略的 BP 神经网络误差更小,收敛速度更快,因此泛化能力更强。对于别的数据集,有待进一步研究和实验。

参　考　文　献

[1] Williams D, Hinton GE. Learning representations by back-propagation by errors[J]. Nature, 1986, 323(9): 533-536.

[2] Qian N. On the momentum term in gradient descent learning algorithms[J]. Neural Netw, 1999, 12(1): 145-151.

[3] Nesterov Y. Gradient methods for minimizing composite objective function[J]. Core Discussion Papers, 2007, 140(1): 125-161.

[4] 王磊,王汝凉,曲洪峰,等. BP 神经网络算法改进及应用[J]. 软件导刊,2016,15(5):38-40.

[5] 周志华. 机器学习[M]. 北京:清华大学出版社,2016:97-106.

[6] 王磊,王汝凉. 基于改进的 BP 神经网络方法的数据挖掘[J]. 广西师范学院学报(自然科学版),2016,33(1):79-84.

[7] 蔡荣辉,崔雨轩,薛培静. 三层 BP 神经网络隐层节点数确定方法探究[J]. 电脑与信息技术,2017,25(5):29-33.

[8] Alli-Oke R O, Heath W P. A secant-based Nesterov method for convex functions[J]. Optimization Letters, 2017, 11(1): 81-105.

[9] Ninomiya H. A novel quasi-Newton-based optimization for neural network training incorporating Nesterov's accelerated gradient[J]. Nonlinear Theory and Its Applications, IEICE, 2017, 8(4): 289-301.

[10] Botev A, Lever G, Barber D. Nesterov's accelerated gradient and momentum as approximations to regularised update descent[C]//2017 International Joint Conference on Neural Networks (IJCNN). IEEE, 2017: 1899-1903.

[11] 于涛. BP 网络自适应学习率算法分析[D]. 大连:大连理工大学数学学院,2011.

(上接第 162 页)

作总结如下:

(1) 经检验,本文建立的分区质量评价函数能够反映分区的质量。网格质量与分区质量相关度较高。

(2) 遗传算法能够有效处理结构化网格分区问题。与传统手工划分网格方法相比,极大提高了网格生成的成功率与自动化程度。

(3) 在采用遗传算法分区前,需要预先设定对应面上的分割点数目,尚未解决全自动处理复杂几何结构分区的问题。

参　考　文　献

[1] 戴萍, 林枫. 燃气轮机叶片气膜冷却研究进展[J]. 热能动力工程, 2009, 24(1): 1-6.

[2] 龚勋. 涡轮冷却叶片结构网格参数化方法研究[D]. 南京:南京航空航天大学, 2016.

[3] 董平. 航空发动机气冷涡轮叶片的气热耦合数值模拟研究[D]. 哈尔滨:哈尔滨工业大学, 2009.

[4] Lai Y K, Kobbelt L, Hu S M. Feature aligned quad dominant remeshing using iterative local updates[J]. Computer-Aided Design, 2010, 42(2): 109-117.

[5] 周天孝, 白文. CFD 多块网格生成进展[J]. 力学进展, 1999, 29(3): 344-368.

[6] 岳孟赫, 刘勇, 赵璐,等. 涡轮叶片结构化网格自动分区策略研究[J]. 航空计算技术, 2017,47(5): 68-72.

[7] 周明,孙树栋. 遗传算法原理及应用[M]. 北京:国防工业出版社,2002.

[8] 张洪梅. 三维六面体网格自适应生成算法研究及其应用[D]. 济南:山东大学,2007.

[9] Thompson J F, Bharat K S, Weatherill N P. Handbook of grid generation[M]. CRC Press, 2009.

(上接第 219 页)

[9] 何国正. 分布式智能网络爬虫的设计与实现[D]. 北京:中国科学院大学,2016.

[10] Mun H J, Li Y. Secure Short URL Generation Method that Recognizes Risk of Target URL[J]. Wireless Personal Communications, 2017, 93(1): 269-283.

[11] Sumathia P, Manickachezianb R. Semantic Web Mining using Web Crawler and DOM Tree with EsvmModified SOM for Advanced Medical Information Retrieval System[J]. International Journal of Control Theory and Applications, 2017, 10(12): 247-264.

[12] 姜子林. 层次聚类的方法及应用[J]. 电子技术与软件工程,2018(1): 179-180.