

## 利用蚁群算法对 PageRank 算法的改进

丁岳伟, 郭 辉

(上海理工大学 光电信息与计算机工程学院, 上海 200093)

(guohui8411@live.cn)

**摘 要:**在 PageRank 算法的基础上应用蚁群算法的主要思想,对网页按关键字检索后被点击的次数进行统计,根据其在初始排序结果中的位置对网页进行分类,通过给定的函数变换对按照以上两个因素统计分析所得的结果进行运算,将其作为网页与关键字关联度的一个评判依据。从而对网页的权值(PR 值)进行迭代修正,并返回一个新的排序结果。通过模拟实验表明,此方法在使得返回结果中相关度较高的网页通过人们的自主选择获得了不同程度的加权,使得其在返回结果中的排名得到提升,更容易被检索到,提高了查准率。

**关键词:**PageRank 算法; 蚁群优化; PR 值; 排序

**中图分类号:** TP18; TP311 **文献标志码:** A

## Improvement of PageRank algorithm by ant colony algorithm

DING Yue-wei, GUO Hui

(School of Optical-Electrical and Computer Engineering, University of Shanghai for Science and Technology, Shanghai 200093, China)

**Abstract:** This paper adopted the main idea of ant colony algorithm to improve the PageRank algorithm. Categorize the Web page according to the position of it in original sequence, compute the clicked number of the Web page acquired through the keyword search, and use the existing transfer function to operate the acquired result. The result was used as a factor to evaluate the degree of correlation between keywords and Web pages, so as to update the weight of the Web page, then obtain a new sort result. The simulation experiments show that this method is through people's subjective choice, so closely associated with the returned results page, their weight has been added with various degrees of growth. Its ranking in the returned results would be enhanced and more easily retrieved and the precision ratio was improved.

**Key words:** PageRank algorithm; ant colony optimization; PR value; sort

### 0 引言

搜索引擎在网络中的作用越来越重要。人们通过搜索引擎在海量的互联网信息中查找自己所需的信息。互联网上的信息包罗万象,几乎包含了整个人类发展历史中所积累的全部知识,并且还在以每天超过 100 万张网页的速度增长。如何在此巨大的信息海洋中快速检索到自己想要的信息成为人们最关注的问题。1998 年,斯坦福大学的 Sergey Brin 和 Lawrence Page 提出了 PageRank 算法,并以此为核心开发出的搜索引擎 Google 在商业应用中获得极大成功。由于人们都希望通过搜索引擎尽快找到自己真正所需的信息,作为搜索引擎的核心部分,对所搜索网页的排名算法的优劣自然成为评价一个搜索引擎好坏的主要指标。PageRank 算法作为著名搜索引擎 Google 的核心算法而备受瞩目,但仍有自己的优缺点,因此许多学者提出了一些改进的方法以提高其检索的查准率。

### 1 PageRank 算法

#### 1.1 PageRank 算法简介

PageRank 算法的主要设计理念是每一个到该网页的链接就是对此网页的一次投票,被链接得越多,就说明有越多的网页愿意将它们自己与此网页挂钩,即链接流行度<sup>[1]</sup>越高。链接流行度越高,此网页的权值就越大,排名也会更靠前。

PageRank 算法通过分析此网页被链接的数量和接入网页的质量来确定网页本身最终的权值。

PageRank 算法<sup>[2]</sup>模拟用户随机浏览的过程,即当用户浏览网页时,其跳转到一个随机页面上的概率是  $d$ ,即其沿着一个(当前页的)随机链接迁移的概率为  $1 - d$ 。假定这个用户不会回退浏览以前访问过的网页,则此过程可以用 Markov 链来建模,从而求出每个页面的平均概率。我们将整个网络看成一个有向图,则网页为此有向图中的节点,网页间的链接看作有向边。若网页  $A$  中有连接到网页  $B$  的链接,即  $A$  指向  $B$ ,为  $A$  对  $B$  的一次投票。假设网页  $A$  有  $n$  个网页对其进行了投票,记为  $(T_1, T_2, \dots, T_n)$ ,设  $C(T_i)$  为网页  $T_i$  的外向链接数, $PR(T_i)$  为页面  $T_i$  的权值,则可以得到此网页的权值计算公式:

$$PR(A) = d + (1 - d) \sum_{i=1}^n \frac{PR(T_i)}{C(T_i)} \quad (1)$$

其中: $d$  为系统设定的经验值,一般取 0.15; $PR(T_i)/C(T_i)$  为网页  $T_i$  的权值  $PR(T_i)$  被平均分成  $C(T_i)$  份后对网页  $A$  的投票; $(1 - d)$  是为了防止接入页面产生过大的影响而对其传给网页的权值进行阻尼; $d$  是为了弥补阻尼掉的权值为防止网页无外部链接时初始  $PR$  值为 0。

PageRank 算法通过重复执行算法对网页的权值进行迭代,从而得出网页的 PageRank 值。

#### 1.2 PageRank 算法相关研究

人们的查询具有主题特征,PageRank 忽略了主题相关

收稿日期:2009-04-09;修回日期:2009-05-29。

作者简介:丁岳伟(1961-),男,上海人,教授,主要研究方向:信息安全、软件工程、知识挖掘;郭辉(1984-),男,河南卫辉人,硕士研究生,主要研究方向:软件工程、知识挖掘。

性,导致结果的相关性和主题性较低;PageRank 对新网页有很严重的歧视,即越旧的网页越靠前。针对这些问题,很多学者提出了自己的改进办法,如文献[4-5]作者分别提出了具有时间反馈的改进算法用于 PageRank 算法偏重于旧网页的问题。文献[6]作者提出了一种主题敏感的 PageRank 改进算法(TH-PageRank 算法),文献[7]作者提出了一个结合链接分析和文本内容的 PageRank 改进算法(MP-PageRank 算法)。这两种算法通过利用查询主题以外的信息来提高对网页的辨识能力,以减少主题漂移现象的发生。本文从网页相对于关键字的点击率入手,通过将蚁群算法的思想引入到 PageRank 值的计算中去,以用户对相关网页的点击次数作为反馈信息,对网页的 PR 值进行修正,从而优化排序结果,使相关的网页排在前面。

## 2 PageRank 算法的改进

### 2.1 蚁群算法简介

蚁群算法是一种用来在图中寻找优化路径的几率型技术。其原理如下:

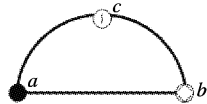


图1 蚁群算法示意图

设蚁穴为点  $a$ , 食物为点  $b$ 。蚂蚁从  $a$  出发,在  $a$  与  $b$  之间爬行并将  $b$  运回  $a$ ;  $a$  与  $b$  之间有两路  $ab$  与  $acb$ , 其中  $acb = 2ab$ , 但对蚂蚁来说是未知的。现在有  $n$  只蚂蚁来运送食物, 首先, 蚂蚁会从未知的两条路中随机地选择一条作为自己要走的路径, 这时两条路的概率是相等的, 都是  $1/2$ 。但蚂蚁爬行的时候会在其所选的路上留下信息素, 作为其判断路径的依据, 记  $I(ab)$  为线段  $ab$  上的信息素。一开始由于两条路选择的几率一样大, 所以两条路上所留的信息素是相同的, 即  $I(ab) = I(acb) = n/2$ ; 开始运送时, 蚂蚁的爬行速度相差无几, 但由于两条路的长短不同, 选择路径  $ab$  的蚂蚁会先到达食物  $b$ , 而当  $ab$  上的第一只蚂蚁到达  $b$  处时,  $acb$  上的蚂蚁刚到达点  $c$ , 由随机性可知,  $ab$  路径上的蚂蚁到达食物  $b$  处开始往回走的时候仍然面临选择, 这时总共有  $n/2$  只蚂蚁, 由于路径  $ab$  上有信息素为  $n/2$ , 即  $I(ab) = n/2$ ; 但路径  $acb$  上由于蚂蚁刚到点  $c$  处,  $I(cb) = 0$ ; 所以蚂蚁会选择路径  $ba$  返回, 此时两条路上的信息素差别显现出来, 由于选择路径  $ba$  的蚂蚁增多, 其信息素也随之增大, 则  $I(ba) = n/2 + n/2 = n$ ; 当路径  $acb$  上的蚂蚁到达点  $b$  时, 此时  $I(cb) = n/2$ ;  $I(ba) = n$ , 由于  $I(ab) > I(bc)$ ; 蚂蚁都会选择信息素高的路径即从路径  $ab$  上回点  $a$ , 最终所有的蚂蚁都会聚集到信息素高的路径即较短的路径上来, 从而得到最优路径。

### 2.2 蚁群算法在排序中的应用

考虑搜索完成以后所显示网页被点击的次数<sup>[8]</sup>。对应于蚁群算法, 建立以下映射将其过程用到网络中网页排序的过程。

- 1) 用户→蚂蚁;
- 2) 关键字→食物;
- 3) 网络链接→不同路径;
- 4) 用户对链接的点击→蚂蚁留下的信息素。

当用户通过关键字检索后, 根据自己的主观判断对搜索显示的网页做出自己的选择。因为每个人判断的主要依据是网页链接与其所需信息的关联程度, 所以那些真正被用户需

求的网页会被大部分用户点击, 其相应的信息度就会越来越高。我们可以通过对当前网页被点击的次数进行一定的函数变换, 将其值作为网页本身权值的修正项, 从而影响网页的排名, 使用户能更快地找到其所需的信息。

根据北京大学的网络与分布式系统研究室对北大天网系统的研究发现<sup>[9]</sup>: 用户一般对搜索引擎推荐结果的前面 5 页表现出浓厚的兴趣, 其中用户在第 1 页的点击数占总点击数的 47%, 在前面 5 页的点击数占到了总点击数的 75% 以上 (图 2)。

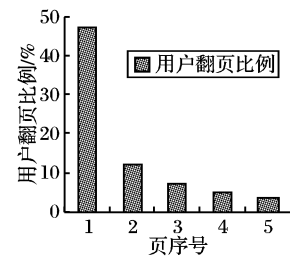


图2 点击前 5 页的用户比例

根据上面两种思想做以下设定。

推论 1 在搜索完成后, 对其所搜到的网页进行排序并返回显示结果, 其网页排名越靠前, 则其被用户点击到的几率就越大。

推论 2 只有和关键字有一定联系的网页才有可能排在前面, 只有和关键字关联度紧密的网页才有可能被用户点击到。

推论 3 网页在搜索结果中的排名越靠后, 其被点击到后包含的信息熵就越高。

在一定时期内,  $C_i(T, key, s)$  表示网页  $T$  在搜索关键字  $key$  的情况下在搜索中第  $i$  次出现, 并且排在  $s$  位时对应的网页点击加权值; 即:

$$C_i(T, key, s) = \begin{cases} 1, & 0 < s \leq 10 \\ 2, & 10 < s \leq 30 \\ 4, & 30 < s \leq 50 \\ 8, & 50 < s \leq \infty \end{cases} \quad (2)$$

其中, 网页排名的次序  $s$  所对应的取值区间的划分是根据图 2 的研究结果设定的, 其具体含义在下面会详细讨论。

在一定时期内,  $P_i(T, key)$  表示网页  $T$  在关键字为  $key$  的情况下被搜索引擎的第  $i$  次搜索中是否被点击。即:

$$P_i(T, key) = \begin{cases} 0, & \text{网页未被点击} \\ 1, & \text{网页被点击} \end{cases} \quad (3)$$

则  $\sum_{i=1}^n C_i(T, key, s) \times P_i(T, key)$  就是在一定时期内网页  $T$  在搜索关键字为  $key$  的情况下经过函数  $C_i(T, key, s)$  加权后被点击的次数。我们可以通过这个数值对网页的权值进行修正。

构造权值修正函数  $M(T, key)$  为网页  $T$  相对于关键字  $key$  的修正权值, 具体函数如下:

$$M(T, key) = \alpha \left[ \lg \sum_{i=1}^n C_i(T, key, s) \times P_i(T, key) \right]^2 \quad (4)$$

其中  $\alpha$  为阻尼系数, 用来限制点击次数对网页本身权值产生过大影响。由于 PageRank 值相差 1 就会产生很大的影响, 所以我们将阻尼系数  $\alpha$  取为 0.02。

由此可以得到一个网页  $T$  相对于关键字  $key$  的最终排序值公式:

$$PR(T) = d + (1 - d) \sum_{i=1}^n \frac{PR(T_i)}{C(T_i)} + M(T, key) \quad (5)$$

可以从以下方面考虑:用户搜索自己所需的信息,在第一页就得到真正所需信息的情况下可能会放弃对其他搜索结果的查询,由于前 10 页是最先默认显示的网页,用户一定可以看到并且点击网页的几率是很高的。在这种情况下,没有必要使其被点击的值对其排列产生较大的影响;但由于其排名靠前,其关联度也应该是很高的,为了不使其因为其他网页的加权而造成网页本身名次的隐性下降,可以将它的点击率加权值设为 1。由于第一页的点击率很高,所以虽然其加权的值没有其后面页面的加权值高,但其较高的点击率仍然可以确保其权值不会快速地隐性下降。随着网页排名的下降,其被用户点击的概率也随之下降,此时,我们通过加大名次靠后的网页的点击次数加权值来平衡其因名次下降而导致的点击率的下降,使得网页的点击次数对网页排名的次序以一种我们可以感觉到的速率增长。

在这里,通过函数  $C_i(T, key, s)$  对搜索结果中排名次序不同的网页的点击次数进行不同权值的加权,使排名靠后的和搜索相关的网页可以在多次点击的情况下快速地提到前面来,进而被用户检索到。

我们可以通过伪代码对其流程进行说明,首先对一些要用到的函数进行定义。

定义存储在索引文件中的相对于关键字的 url 的结构体:

```
struct record
{
    string url; //url 地址
    int score; //关键字出现的次数
    short click Num; //该 url 被点击的次数
}
```

定义函数统计网页根据排名加权后的点击次数:

//检查网页是否被点击,是返回 1, 否则为 0

```
int check click(string url, string key)
```

```
{
    if (网页被点击)
```

```
        return 1;
```

```
    else
```

```
        return 0;
```

```
}
```

//根据 url 的排名确定其点击次数的加权值

```
void check URL level(string url, string key, int order, int *C)
```

```
{
```

```
    if (order > 0 && order <= 10)
```

```
        C = 1;
```

```
    else if (order > 10 && order <= 30)
```

```
        C = 2;
```

```
    else if (order > 30 && order <= 50)
```

```
        C = 4;
```

```
    else if (order > 50)
```

```
        C = 8;
```

```
}
```

```
void get click num(string url, string key, short record.click Num)
```

```
{
```

```
    int c;
```

```
    int p = check click(url, key);
```

```
    check URL level(url, key, order, &c);
```

```
    record.click Num = record.click Num + c * p;
```

```
}
```

```
short modify PR value(string url, string key, short record.click num)
```

```
{ short n = * record.click num;
```

```
    double temp = * mp;
```

```
    Temp =  $\alpha * \lg n * \lg n$ ;
```

```
    return temp;
```

```
}
```

通过以上函数进行下面的过程说明:

```
Input(string key);
```

```
Processed:
```

```
String url = get Url List(string key);
```

```
//根据关键字获得相关的 url 列表;
```

```
get PR value(string url);
```

```
//获得 url 现有的 PR 值;
```

```
final PR value(url, key) = get PR value(url) + modify PR value(url, key, record.click Num);
```

```
//获得修正后的 PR 值
```

```
check click(string url, string key);
```

```
check URL level(string url, string key, int order, int *C);
```

```
get click num(string url, string key, short record.click Num);
```

```
//显示最后的排序结果
```

```
display url by final value();
```

考察网页被点击的次数,取样本点为起点中文网的最热门小说的点击率为 57 533 486 次,这是经过将近一年时间且不断更新内容的结果。天涯论坛上点击率最高的帖子被点击的次数为 11 507 868 次,而这是用了 3 年时间不断更新的结果(截止 09/3/30)。由此可以推得:一个普通的网页被搜索到并被点击到的次数最多也不会超过千万级。

由此可以来检验构造的函数对权值修正的上限是否越界,从而验证修正函数对网页权值的影响。

当网页在搜索中被成功点击的时候,它相应的修正权值也会随之增长。具体修正值如图 3 所示。

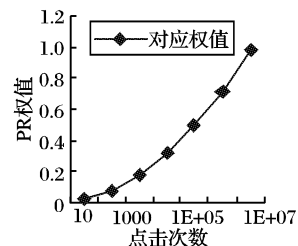


图3 修正权值随点击次数的增长率

由图3可以看出,当网页被点击 1000 次的时候,其修正权值为 0.18;被点击 10 000 次时其权值增加至 0.32,由此可见,当用户按照自己的选择点击网页时,那些真正被用户所需要的网页会被群体中的大部分用户点击,从而其信息熵会不断地增加,信息熵通过函数  $M(T, key)$  叠加到网页的 PR 值上,对其原始的 PR 值进行修正后重新排列显示,这样,点击率高的网页排名也会逐渐靠前。

### 3 实验结果分析

为了测试本算法的效果,我们利用模拟实验对其结果进行比较分析。我们通过开源的搜索引擎 Nutch 对任一主题进行搜索,在此选小说“风姿物语”作为搜索关键字,由于绝大部分人搜索小说都是想找到小说本身下载或者在线阅读,而出现的结果因为网页本身权值的问题,很有可能是权值较高但相关度较低的网页的排在前面。实验时通过搜索引擎 Nutch 在网上抓取到 129 000 张相关网页存储到本地,用原始的 PageRank 算法计算出网页的权值,然后用关键字查询并返回推荐结果,实验期间用  $F_i$  对首次返回的排序结果按次序命名。通过用户的主观判断对相关网页进行模拟点击,并将此关键字在搜索到的集合中建立点击次数属性用于改进结果。运行一段时间后,用改进后的排序算法来计算网页的权值并重新排序。其实验结果如表 1。(下转第 2740 页)

本缩减时间),  $N_{sv}$  表示支持向量数,  $P_r$  表示预测正确率。

通过表 2 中的实验数据可以看出, LSTSRS 较 Shrinking 技术和 PSCC 算法有了更高的效率, 且通过有效去除类内混

杂的非边界孤立点数据, 提高了 SVM 的泛化性能, 使分类正确率有了一定程度的提高, 故该缩减策略是有意义的, 且符合实际应用的要求。

表 2 LSTSRS 方法与 PSCC 和 Shrinking 方法的比较

Data	dim	$N_{tr}$	$N_t$	SVM			Shrinking + SVM			PSCC + SVM			LSTSRS + SVM		
				$T_{tr}/s$	$N_{sv}$	$P_r/\%$	$T_{tr}/s$	$N_{sv}$	$P_r/\%$	$T_{tr}/s$	$N_{sv}$	$P_r/\%$	$T_{tr}/s$	$N_{sv}$	$P_r/\%$
a5a	123	5318	7000	76.875	2330	74.0857	39.859	2338	74.0857	35.7521	2311	73.6849	32.3516	2228	75.8320
a6a	123	11220	9945	281.249	4004	79.2459	139.141	4004	79.2459	106.1847	3911	77.7329	95.3830	3814	79.2547

## 4 结语

本文提出的 SVM-LSTSRS 从分析单个样本点与相对样本集合的关系出发, 给出了潜在支持向量的概念, 并以得到潜在支持向量为目的进行样本集的缩减。在 LSTSRS 中为了避免大量的计算负担, 引入了模糊 C-均值聚类, 并在得到隶属度矩阵的基础上对样本点与另一类样本集的相对关系给出了合理的判定规则。通过实验证明该缩减策略在保证分类精度的同时能够大大缩小训练样本集的规模, 并降低了孤立点对 SVM 泛化性能的影响, 具有较高的执行效率, 是有效且可行的。

### 参考文献:

- [1] AGARWAL D K. Shrinkage estimator generalizations of proximal support vector machines[C]// Proceedings of the 8th ACM SIGKDD International Conference of Knowledge Discovery and Data Mining. New York: ACM Press, 2002: 173-182.
- [2] DANIAEL B, CAO D. Training support vector machines using adaptive clustering[C/OL]. [2009-02-01]. [http://www.siam.org/](http://www.siam.org/proceedings/datamining/2004/dm04_012boleayd.pdf)

[proceedings/datamining/2004/dm04\\_012boleayd.pdf](http://www.siam.org/proceedings/datamining/2004/dm04_012boleayd.pdf).

- [3] 罗瑜. 支持向量机在机器学习中的应用研究[D]. 成都: 西南交通大学, 2007.
- [4] 肖小玲, 李腊元, 张翔. 提高支持向量机训练速度的 CM-SVM 方法[J]. 计算机工程与设计, 2006, 27(22): 4183-4184.
- [5] 李红莲, 王春花, 袁保宗. 一种改进的支持向量机 NN-SVM[J]. 计算机学报, 2003, 26(8): 1015-1020.
- [6] 曾志强. 支持向量机分类机的训练与简化算法研究[D]. 杭州: 浙江大学, 2007.
- [7] 谭冠群, 丁华福. 支持向量机方法在文本分类中的改进[J]. 信息技术, 2008, 2(1): 83-88.
- [8] 曹淑娟, 刘小茂, 张钧, 等. 基于类中心思想的去边缘模糊支持向量机[J]. 计算机工程与应用, 2006, 42(22): 146-149.
- [9] CRISTIANINI N, SHAWE-TAYLOR J. An introduction to support vector machines and other kernel-based learning methods[M]. 李国正, 王猛, 曾华军, 译. 北京: 电子工业出版社, 2006.
- [10] 程其襄, 张奠宙, 魏国强, 等. 实变函数与泛函分析基础[M]. 2 版. 北京: 高等教育出版社, 2003: 31-37.

(上接第 2728 页)

表 1 算法改进前后 PR 值及排序

网页	改进前 PR 值	改进前排名	改进后 PR 值	改进后排名
$F_1$	0.4535	1	0.6025	1
$F_2$	0.4309	2	0.4309	8
$F_3$	0.3771	3	0.5474	2
...	...	...	...	...
$F_7$	0.3426	7	0.4837	4
$F_8$	0.3314	8	0.3746	11
...	...	...	...	...
$F_{12}$	0.2987	12	0.2987	12
$F_{13}$	0.2814	13	0.4634	7
$F_{14}$	0.2792	14	0.2792	14
...	...	...	...	...

由于实验原因, 部分相关度不高的网页未被模拟点击。而  $F_2$  是关于风姿物语的百度百科, 虽然与主题相关但与所求无关, 所以被排除掉, 因此权值不变。 $F_1, F_7$  和  $F_{13}$  由于是专门的小说网站上的网页, 被用户识别并点击的次数很大,  $F_{13}$  由于本身位于第二页, 其加权率本身就高, 所以在较少的点击率下排名上升到第 7 位, 在排序结果的第一页显示。由表 1 可以看出, 改进后的算法提升了用户真正感兴趣的网页的名次, 从而使用户可以更快地找到自己所需的结果。

## 4 结语

本文从网页相对于关键字的点击率出发, 通过蚁群算法的信息熵的概念将用户的群体选择加入到网页权值计算中去, 提高了相关网页的查准率。但其本身仍一定的问题。如网

页加权的权值是否可以继续传递下去, 这样会造成一部分相关网页权值的整体上升等, 这些问题还需要进一步的研究。

### 参考文献:

- [1] BRINKMEIER M. PageRank revisited[J]. ACM Transactions on Internet Technology, 2006, 6(3): 282-301.
- [2] RICARDO B-Y, BERTHIER R-N. Modern information retrieval[M]. 王知津, 贾福新, 郑红军, 等译. 北京: 机械工业出版社, 2004.
- [3] 黄德才, 戚华春, 钱能. 基于主题相似度模型的 TS-PageRank 算法[J]. 小型微型计算机系统, 2007, 28(3): 510-513.
- [4] 宋聚平, 王永成, 尹中航, 等. 对网页 PageRank 算法的改进[J]. 上海交通大学学报, 2003, 37(3): 397-400.
- [5] 戚华春, 黄德才, 郑月锋, 等. 具有时间反馈的 PageRank 改进算法[J]. 浙江工业大学学报, 2005, 33(3): 272-275.
- [6] HAVELIWALA T H. Topic-sensitive PageRank[C]// Proceedings of the Eleventh International World Wide Web Conference. Honolulu: ACM Press, 2002: 517-526.
- [7] RICHARDSON M, DOMINGOS P. The intelligent surfer: Probabilistic combination of link and content information in PageRank[C]// Advances in Neural Information Processing Systems. Cambridge: MIT Press, 2002: 1441-1448.
- [8] ZHE CAO, TAO QIN, LIU TIE-YAN, et al. Learning to rank: From pairwise approach to listwise approach[C]// Proceedings of the 24th International Conference on Machine Learning. New York: ACM Press, 2007: 129-136.
- [9] 王建勇, 单松巍, 雷鸣, 等. 海量 Web 搜索引擎系统中用户行为的分布特征及其启示[J]. 中国科学: E 辑, 2001, 31(4): 372-384.