一种求解集合覆盖问题的启发式算法*)

陈端兵　黄文奇
(华中科技大学计算机科学与技术学院　武汉 430074)

摘　要　集合覆盖问题是运筹学研究中的一个基本的组合优化问题，它通常描述成如下的一个覆盖问题：从一个 m 行、n 列的 0-1 矩阵 $(a_{ij})_{m\times n}$ 中选出若干列盖住所有的行，使得付出的代价最小。集合覆盖问题被广泛应用到航空人员行程安排、电路设计、运输的车辆路线安排等领域。对这一问题，国内外学者提出了诸如遗传算法、模拟退火算法、蚁群算法、人工神经网络算法等求解算法。本文以贪心算法为基础，利用人类的智慧和经验，提出了一种求解集合覆盖问题的启发式算法。算法的主要思想为：从某个解出发，随机移除一定比例的列，再用贪心策略加入若干列。用本文提出的算法，对 Beasley 提出的 45 个测试实例进行了实算测试，所得结果和最优解的平均相对差值为 0.44%，并且得到了其中 33 个实例的最优解，实算结果表明，本文提出的算法对求解集合覆盖问题是行之有效的。

关键词　集合覆盖，启发式算法，贪心策略，随机跳坑

A Heuristic Algorithm for Set Covering Problem

CHEN Duan-Bing　HUANG Wen-Qi
(School of Computer Science, Huazhong University of Science and Technology, Wuhan 430074)

Abstract　The set covering problem is a fundamental combinatorial problem in operations research. It is usually described as the problem of covering the rows of this m-row, n-column, 0-1 matrix $(a_{ij})_{m\times n}$ by a subset of the columns at minimal cost. This problem has many practice applications such as airline crew scheduling, circuit design, vehicle routing, etc. For solving this problem, many algorithms such as genetic algorithm, simulated annealing, ant colony algorithm and artificial neural network algorithm have been proposed. Based on the greedy algorithm, a heuristic algorithm for set covering problem is proposed according to wisdom and experience of human being in this paper. The main idea of the algorithm is that remove some columns from a solution randomly and add some new columns by greedy strategy. 45 instances provided by Beasley are tested by the produced algorithm, the average gap between the best solution and the optimal solution is 0.44% and 33 instances of them are achieved optimal solutions. Experimental results demonstrate that the algorithm proposed in this paper is efficient for solving the set covering problem.

Keywords　Set covering problem, Heuristic algorithm, Greedy strategy, Random off-trap

1　引言

集合覆盖问题是运筹学研究中的一个基本的组合优化问题，被广泛应用到航空的人员行程安排、电路设计、运输的车辆路线安排等领域[1]。它通常描述成如下的一个覆盖问题：从一个 m 行、n 列的 0-1 矩阵 $A=(a_{ij})_{m\times n}$ 中选出若干列盖住所有的行，使得盖住所有行所付出的代价最小。集合覆盖问题已被证明是一个 NP 完全问题[2]。

对集合覆盖问题，许多学者利用不同的思想提出了各种各样的算法。一些学者基于分支定界思想提出了求解此问题的完整算法[3]，Caprara 等人对几种完整算法进行了比较和分析，得出 CPLEX 是求解集合覆盖问题的最好的完整算法[4]。但对规模较大的问题，完整算法由于其计算时间太长而显得力不从心。为此，一些学者提出了求解此问题的启发式算法，使得在一个可以接受的时间内求得问题的近似最优解。贪心算法是一种很容易想到的启发式算法，它的优点是计算速度很快，但结果的优度不甚理想。一些学者为了克服这一问题，引入了随机跳坑策略来提高解的优度[5,6]。近年来，国内外一些学者利用自然界的物理现象和生物活动与进化规律，提出了许多好的启发式算法，如遗传算法[7-9]、模拟退火算法[10]、蚁群算法[11]、人工神经网络算法[12]等等。

受以上这些研究的启发，本文以贪心算法为基础，利用人类的智慧和经验，以最小化代价（即用于盖住所有行的列的代价总和）为优化目标，提出了一种集合覆盖问题的启发式算法。该算法首先利用贪心算法找到一个可行解。然后从此解出发，随机移除一定比例的列，再用贪心策略加入若干列，这样就可能得到比原来更优的解。按照此策略实施若干步之后，就可以得到问题的最优解或近似最优解。利用本文提出的算法，对 Beasley 提出的 7 组共 45 个测试实例[13]进行了实算测试并和贪心算法、PROGRES 算法[5]进行了对比。对 45 个实例，本文算法所得最好解和最优解的平均相对差值为0.44%，并且得到了其中 33 个实例的最优解；PROGRES 算法所得最好解和最优解的平均相对差值为 0.54%，且可得到其中 10 个实例的最优解；而用纯粹的贪心算法，所有 45 个实例都没有得到最优解，和最优解的平均相对差值也高达8.13%。测试结果表明，本文提出的算法对求解集合覆盖问题是行之有效的。

2　问题描述

集合覆盖问题可用如下的二值整数规划来确切地描述：

$$\min \quad \sum_{j=1}^{n} c_j x_j, \tag{1}$$

$$\text{s. t.} \quad \sum_{j=1}^{n} a_{ij} x_j \geqslant 1, i=1,2,\cdots,m \tag{2}$$

*)国家自然科学基金资助项目(10471051 和 973 项目 2004CB318000)。陈端兵　博士生，主要研究方向为 NP 问题高效求解；黄文奇　教授，博士生导师，主要研究方向为 NP 难问题高效求解、人工智能、博奕等。

$$x_j \in \{0,1\} j=1,2,,\cdots,n \quad (3)$$

式中：c_j——第 j 列的代价；x_j——取 0 或 1，取 1 表示第 j 列包含在解中，取 0 表示第 j 列没有包含在解中；a_{ij}——0-1 矩阵 $A=(a_{ij})_{m\times n}$ 第 i 行、j 列的值，$a_{ij}=1$ 表示第 j 列盖住了第 i 行，$a_{ij}=0$ 表示第 j 列没有盖住第 i 行。

3 算法描述

对集合覆盖问题，我们的算法思想就是从 n 列中，按照某种规则依次选择若干列盖住所有的行，使得这些被选中列的代价之和尽量地小。在某一时刻，已经按选择规则选择了若干列盖住若干行，还有若干行未被盖住，这种状态，称为一种格局。开始时，所有行都未被盖住，此时的格局称为初始格局；若所有行都被盖住了，此时的格局称为终止格局。算法的目标就是要找一个代价最小的终止格局。图 1 为一个集合覆盖的例子，A 为 5 行 6 列的 0-1 矩阵，$\vec{C}$ 为列的代价值，$\vec{X}$ 为解向量。图 1 表示的是一个终止格局，用第 1、3、4 列盖住了所有 5 行，代价为 6。对于这个例子，最优解为 $\vec{X}=(1,1,0,1,0,0)$，即用第 1、2、4 列盖住所有的行，其代价为 5。我们可用如下 3 个策略来描述本文的集合覆盖算法：冗余列处理策略、贪心选择策略、随机跳坑策略。

$\vec{X}$	1	0	1	1	0	0
	1	1	0	1	0	0
	0	0	0	1	1	0
A	1	0	1	0	1	1
	0	1	1	0	0	1
	1	0	0	0	0	1
$\vec{C}$	1	1	2	3	6	7

图 1 集合覆盖的例子

3.1 冗余列的处理

对集合覆盖问题，可将那些冗余的列事先剔除，以降低问题规模。对任一列 j，其代价为 c_j，设此列可盖住 p 行 $i_1,i_2,\cdots,i_p$。若能从剩下的列中，选出 q 列 $j_1,j_2,\cdots,j_q$ 盖住这 p 行 $i_1,i_2,\cdots,i_p$，并且 $\sum_{l=1}^{q} c_{j_l} \leqslant c_j$，则可将第 j 列剔除，剔除后不会影响解的优度。实际处理时，从可盖住第 $i_l(l=1,2,\cdots,p)$ 行的列中选具有最小代价的一列来盖住第 i_l 行。若这些列的代价之和小于 c_j，则可将第 j 列剔除掉。例如，如图 1 所示，第 6 列盖住了第 3、4、5 行，代价为 7，而盖住第 3、4、5 行且具有最小代价的列分别是第 1、2、1 列，因而可用第 1、2 列盖住第 3、4、5 行，代价之和为 2，小于第 6 列的代价，因此可将第 6 列剔除。同样道理，第 5 列盖住的行可用第 1、4 列盖住，第 3 列盖住的行可用第 1、2 列盖住，并且第 1、4 列代价之和小于第 5 列的代价，第 1、2 列的代价之和等于第 3 列的代价，因而可将第 3、5 列剔除，从而减小了原问题的规模。图 2 所示的集合覆盖是图 1 所示的集合覆盖经简化后的结果。简化后，实例的唯一解 $\vec{X}=(1,1,1)$ 也是其最优解，代价为 5。

$\vec{X}$	1	1	1
	1	1	1
	0	0	1
A	1	0	0
	0	1	0
	1	0	0
$\vec{C}$	1	1	3

图 2 经简化后的集合覆盖实例

3.2 贪心选择策略

为表述方便，我们用一个 m 维的布尔向量 $\vec{F}$ 表示行的盖住情况。在某一格局之下，若第 i 行已被盖住，则 $F_i=$ true，否则 $F_i=$ false $(i=1,2,\cdots,m)$。在此基础上，我们可用如下公式定义在当前格局下列 j 的平均代价：

$$v_j=\frac{c_j}{\sum_{i=1,F_i=false}^{m} a_{ij}} \quad (4)$$

贪心选择策略可描述如下：

(1)从某一格局出发，选择平均代价最小的列来覆盖那些还没有被盖住的行。若有多列，那么

(2)选择代价最小的列。若还有多列，那么

(3)选择序号最小的列。

按此策略选择一列后，便得到新的格局。从此新格局出发，再次使用贪心选择策略，直到所有行都被盖住或剩下的列中已没有列可盖住那些未被盖住的行。对用贪心选择策略得到的解，需将冗余的列从解中剔除，以提高解的优度。即：对解中某列 j，它盖住 p 行 $i_1,i_2,\cdots,i_p$，若可用解中除列 j 外的若干列盖住 p 行 $i_1,i_2,\cdots,i_p$，则可将列 j 从解中剔除。这时剩下的列仍能盖住所有的行，而代价却减小了。

3.3 随机跳坑策略

纯粹的贪心算法得到的解，其优度往往不甚理想。本文采用随机跳坑的策略，进一步提高解的优度。具体跳坑策略可描述如下：

(1)用一循环变量 k 来表示已经随机跳坑的次数，$k_{\max}$ 表示最大跳坑次数，$\vec{X}_{old}$ 为贪心算法所得的解，初始时，$k=0$；

(2)若 $k=k_{\max}$ 或 $\vec{X}_{old}$ 已是最优解，转到第(4)步。否则，从解 $\vec{X}_{old}$ 出发，随机选择一定比例(本文取 8%)的列从解中剔除。这时，便有若干行就没有被盖住了，$k\Leftarrow k+1$；

(3)从剩下的未被选择的列中按贪心选择策略选择若干列盖住这些行，得到新解 $\vec{X}_{new}$。若 $\vec{X}_{new}$ 为可行解且代价比 $\vec{X}_{old}$ 的代价小，则用新解替代原来的解 $\vec{X}_{old}\Leftarrow\vec{X}_{new}$，转到第(2)步。否则恢复到原来的解 $\vec{X}_{old}$，转到第(2)步；

(4)输出最终得到的解。

4 实算结果

对本文提出的算法，用 C#.net 语言编程实现，并对 Beasley 提出的 7 组（SCP 4，SCP 5，SCP 6，SCP A，SCP B，SCP C，SCP D）共 45 个测试实例[13] 在 pentium 4(256M RAM/ 2.0GHz CPU)微型机上进行了实算测试。其中 SCP 4 和 SCP 5 两组包含有 10 个实例，其它各组各有 5 个实例。这 45 个实例的最优解都是已知的，测试实例数据可以从 http://people.brunel.ac.uk/~mastjjb/jeb/orlib/files/上下载。

从实算结果来看，用本文算法求解集合覆盖问题，在可以接受的时间内，所得解的优度较高。我们将本文算法得到的结果和纯粹的贪心算法以及文[5]中给出的 PROGRES 算法的计算结果进行了对比，如表 1 和表 2 所示。PROGRES 算法所用的计算机为 pentium III 800MHz PC，贪心算法所用计算机为 pentium 4(256M RAM/ 2.0GHz CPU)微型机，表中的计算时间为相应算法所用计算机下的计算时间。从对比结果来看，本文算法计算所得结果的综合性能好于 PROGRES 算法和纯粹贪心算法所得的结果。用本文算法计算得到了 45 个测试实例中 33 个实例的最优解，用 PROGRES 算法可得到其中 10 个实例的最优解，而用纯粹贪心算法计算，所有实例都没有得到最优解。如表 1 的第 5、7、9 列所示。

我们还将各算法所得结果和最优解的相对差值进行了对比。相对差值由下式定义：

$$\frac{\text{算法所得最好解}-\text{最优解}}{\text{最优解}}\times 100\% \quad (5)$$

表2给出了7组实例的平均相对差值。从表2可看出，本文算法所得结果和最优解的相对差值比纯粹的贪心算法和PROGRES算法小。贪心算法所得结果和最优解的平均相对差值为8.13%，PROGRES算法为0.54%，本文算法为0.44%，如表2最后一行。

结论 在航空的人员行程安排、电路设计、运输的车辆路线安排等领域，提出了集合覆盖问题。本文在贪心算法的基础上，利用人类的智慧和经验，以最小化代价为优化目标，提出了一种求解集合覆盖问题的启发式算法。算法的主要思想为：从纯粹贪心算法得到的解出发，随机移除一定比例的列，再用贪心策略加入若干列。反复多次使用这一策略，即可得到问题的最优解或近似最优解。用本文提出的算法，对Beasley提出的45个测试实例进行了实算测试，所得结果和最优解的平均相对差值为0.44%，并且还得到了其中33个实例的最优解。实算结果表明，本文提出的算法对求解集合覆盖问题是行之有效的。

表1 本文算法和贪心算法及PROGRES算法的实算结果对比

实例	m	n	最优解	贪心算法		PROGRES		本文算法	
				代价	时间(秒)	代价	时间(秒)	代价	时间(秒)
SCP 4.1	200	1000	429	438	0.11	430	8.80	429	1.28
SCP 4.2	200	1000	512	556	0.12	514	8.60	512	0.94
SCP 4.3	200	1000	516	559	0.13	519	11.30	516	0.74
SCP 4.4	200	1000	494	514	0.06	498	18.30	494	4.49
SCP 4.5	200	1000	512	536	0.16	514	8.20	512	6.94
SCP 4.6	200	1000	560	588	0.12	564	8.60	560	0.93
SCP 4.7	200	1000	430	457	0.07	434	8.00	432	2.44
SCP 4.8	200	1000	492	530	0.10	493	8.60	492	1.15
SCP 4.9	200	1000	641	697	0.08	648	13.00	645	22.44
SCP 4.10	200	1000	514	533	0.08	516	13.10	514	0.56
SCP 5.1	200	2000	253	273	0.10	258	16.20	253	3.27
SCP 5.2	200	2000	302	329	0.14	307	14.90	302	4.92
SCP 5.3	200	2000	226	234	0.09	228	10.20	226	7.62
SCP 5.4	200	2000	242	254	0.06	244	10.40	242	1.16
SCP 5.5	200	2000	211	224	0.07	212	13.30	212	9.38
SCP 5.6	200	2000	213	230	0.10	214	15.40	213	1.96
SCP 5.7	200	2000	293	315	0.10	295	18.00	293	3.00
SCP 5.8	200	2000	288	311	0.17	290	16.00	288	1.41
SCP 5.9	200	2000	279	294	0.11	280	14.90	279	1.36
SCP 5.10	200	2000	265	278	0.15	267	13.10	265	2.51
SCP 6.1	200	1000	138	150	0.13	140	5.80	138	18.01
SCP 6.2	200	1000	146	169	0.13	147	4.80	146	88.67
SCP 6.3	200	1000	145	157	0.13	146	4.50	145	2.53
SCP 6.4	200	1000	131	136	0.11	131	5.00	131	1.36
SCP 6.5	200	1000	161	188	0.10	162	5.00	161	8.88
SCP A.1	300	3000	253	268	0.18	255	48.30	254	150.15
SCP A.2	300	3000	252	270	0.22	253	33.80	252	8.82
SCP A.3	300	3000	232	251	0.22	234	33.60	233	41.06
SCP A.4	300	3000	234	262	0.19	236	38.80	234	25.02
SCP A.5	300	3000	236	251	0.19	238	37.90	237	35.91
SCP B.1	300	3000	69	76	0.13	69	28.20	69	9.54
SCP B.2	300	3000	76	85	0.19	76	26.50	76	0.66
SCP B.3	300	3000	80	87	0.13	80	31.10	80	1.61
SCP B.4	300	3000	79	85	0.17	79	40.80	79	2.64
SCP B.5	300	3000	72	76	0.14	72	31.10	72	13.78
SCP C.1	400	4000	227	246	0.27	230	72.90	227	41.59
SCP C.2	400	4000	219	233	0.35	220	69.20	219	41.76
SCP C.3	400	4000	243	259	0.34	247	100.60	244	132.66
SCP C.4	400	4000	219	246	0.32	220	77.20	219	17.02
SCP C.5	400	4000	215	224	0.32	216	75.60	216	110.35
SCP D.1	400	4000	60	68	0.20	60	45.10	61	226.65
SCP D.2	400	4000	66	72	0.26	66	74.70	68	138.74
SCP D.3	400	4000	72	79	0.26	72	84.70	74	332.88
SCP D.4	400	4000	62	65	0.25	62	65.10	62	0.41
SCP D.5	400	4000	61	71	0.21	61	76.70	64	234.26

表2 各算法所得结果与最优解的平均相对差值(%)

实例	贪心算法	PROGRES	本文算法
SCP 4	5.38	0.57	0.10
SCP 5	5.93	0.88	0.04
SCP 6	10.66	0.69	0.00
SCP A	7.92	0.75	0.25
SCP B	8.78	0.00	0.00
SCP C	7.57	0.87	0.18
SCP D	10.68	0.00	2.48
平均	8.13	0.54	0.44

参考文献

1 Marchiori E, Steenbeek A. An evolutionary algorithm for large scale set covering problems with application to airline crew scheduling. Lecture Notes in Computer Science, 2000, 1803: 367～381

2 Lemke C E. Set covering by single-branch enumeration with linear-programming subproblems. Operations Research, 1971, 19(4): 998～1022

3 Mannino C, Sassano A. Solving hard set covering problems. Operations Research Letters, 1995, 18: 1～5

4 Caprara A, Toth P, Fischetti M. Algorithms for the set covering problem. Annals of Operations Research, 2000, 98: 353～371

5 Haouari M, Chaouachi J S. A probabilistic greedy search algorithm for combinatorial optimization with application to the set covering problem. Journal of the Operational Research Society, 2002, 53: 792～799

6 Yagiura M, Kishoda M, Ibaraki T. A 3-flip neighborhood local search for the se covering problem. European Journal of Operational Research, 2006, 172: 472～499

7 Beasley J E, Chu P C. A genetic algorithm for the set covering

problem. European Journal of Operational Research, 1996, 94: 392～404
8 Solar M, Parada V, Urrutia R. A parallel genetic algorithm to solve the set-covering problem. Computers & Operations Research, 2002, 29: 1221～1235
9 陈亮，任世军. 一种遗传算法在集合覆盖问题中的应用研究. 哈尔滨商业大学学报（自然科学版），2006, 22(2): 67～70
10 Jacobs L, Brusco M. Note: A local-search heuristic for large set-covering problems. Naval Research Logistics, 1995, 42: 1129～1140
11 Lessing L, Dumitrescu I, Stützle T. A Comparison between ACO algorithms for the set covering problem. Lecture Notes in Computer Science, 2004, 3172: 1～12
12 Ohlsson M, Peterson C, Söderberg B. An efficient mean field approach to the set covering problem. European Journal of Operational Research, 2001, 133: 583～595
13 Beasley J E. An algorithm for set covering problems. European Journal of Operational Research, 1987, 31: 85～93

（上接第 118 页）

SXBP 的性能，采用 Shakespeare 剧本[10]作为实验数据集，数据集为 7.65MB，共有 37 个文档，总的元素结点、属性结点和文本结点数分别为 179618、0、147525，CPU 为 P4 2.4G，操作系统为 Windows XP，内存为 512M，采用 JAVA 语言编程，数据库为 SQL Sever 2000 关系数据库。实验分别验证基于 Pri-order 编码的文档恢复效率和路径查询效率。

5.1 文档恢复率

验证 SXBP 的文档恢复率，对数据集中的 37 个文档分别取出不同的 N 个存储到关系表中，利用文档重组算法对存储的关系表进行操作，取 3 次重组时间的平均结果。实验结果如表 1 所示，由结果可以看到本文的文档重组算法有很高的可扩展性，计算重组时间与重组元组数的比值可以看到算法的时间复杂度与理论上分析的具有线性关系基本吻合。

表 1 文档重组时间（单位：s）

重组时间(s) / 元组数(n)	重组时间(s)	线性比(s/n)
6347	248	0.0391
56390	1872.2	0.0332
106379	3486.8	0.0328
179618	5863.7	0.0326
327143	10689.7	0.0327

5.2 路径查询效率

对于路径查询效率的测试，同样采用 Shakespeare 剧本的 37 个文档作为实验数据集。将 XML 文档解析、编码后存入到关系数据库中。针对具体的查询实例与传统的方法 FK[2]和 Xre[3]在查询所需单上比较，结果如表 7 所示；从表 2 可以看出在路径查询上，本文的存储方法对于简单路径的查询与 FK 和 Xrel 相比所需的时间要稍长些，对带有祖先后代关系（//）的查询时间影响不大，在查询实例中间带有“//”时所需的时间会更长些。而对带有谓词约束的查询与不带谓词约束的和简单路径相比所需时间略有增加，但不是很大。

表 2 实例查询时间和查询结果

查询实例	FK(秒)	Xrel(秒)	本文方法(秒)	查询结果数
/play	0.08	0.17	0.26	37
//scene/title	0.13	0.24	0.28	750
/play/act/title	13.0	0.36	0.34	185
//act//title	16.51	0.31	0.32	951
/play/act/scene/speech [speaker='curio']	19.31	2.75	2.73	4

结束语 本文提出一种通用的在关系数据库中存储和恢复 XML 文档的方法 SXBP，它扩展应用了 XRel 基于一种新的编码方式 Pri-order。Pre-order 用一对整数为文档结构树中结点进行编码，编码的第一部分是通过素数编码获得，另一部分是一个整数，用来表示结点在其兄弟结点间的顺序，这种编码方式在对原文档进行恢复时利用保持结点间的顺序，能达到文档恢复和更新间的平衡。SXBP 方法根据结点类型将树型结构分解，采用基于模型映射的方法将结点的信息分别存储到相应的关系模式中。这种方式能够处理任意 XML 文档，无论文档有无特定的模式，并能够按照结点编码和结点所在层数信息有效地对原 XML 文档进行恢复。文中通过对真实数据集的实验证明，这种存储方法有很高的文档恢复率，在路径查询方面与传统的方法相比具有较高查询效率，节省查询时间。

参 考 文 献

1 Shanmugasundaram J, Tufte D, He Gang. Relational Database for Querying XML Documents: Limitations and Opportunities. In: Proceedings of the 25th VLDB Conference, Edinburgh, Scotland, 1999. 302～314
2 Florescu D, Komman D. Storing and querying XML data using an RDBMS[J]. IEEE Data Engineering Bulletioa, 1999, 22(3): 27～34
3 Youshikawa M, Amagasa T. Xrel: A path-based approach to storage and retrieval of XML documents using relational database. ACM Trans. Internet Technology, 2001. 110～141
4 Wen Lun, Zhang Rui, Lu XianLiang. The Design of Efficient XML Document Model. In: Proceedings ofteh First Intemational Conference on Machine Leaming and Cybemetis, Beijing, 2002. 1102～1106
5 Jagadish H V, Al-Khalifa S, Chapman A, et al. TIMBER: Anative XML database. In: Proceedings of the 28th VLDB Conference, 2002. 274～291
6 Halverson A, Josifovshi V, Lohman G, et al. ROX: Relational Over XML. In: Proceeings of the 30th VLDB Conference, Toronto, Canada, 2004. 264～275
7 Pardede E, Rahayu J W, Taniar D. Preserving Composition in XML Object Relational Storage. In: 19th Intemational Conference on Advsanced Information Networking and Applications, 2005. 695～700
8 Wu Xiaodong, Lee M L, Hsu W. A Prime Number Labeling Schemes for Dynamic Ordered XML Trees. In: Proceeding of the 20th International Conference on Data Engineering ICDE, 2004. 66～78
9 Fujimoto K, Shimizu T, Kha D. A mapping Scheme of XML Documents into Relational Databases using Schema-based Path Identifiers. In: Proceedings of the 2005 International Workshop on Challenges in Web Information Retrieval and Integration, 2005. 82～90
10 Bosak J. The Plays of Shakespeare in XML [EB/OL]. http//metalab. unc. edu/xml/examples/Shakespeare/, 1999