

算法设计： k -VCC的查找

石依凡 2020202264

目录

算法设计： k -VCC的查找

石依凡 2020202264

目录

1 问题概览

2 实验完成度

3 实验说明

4 测试结果

附录A 测试数据集

1 问题概览

k -VCC (k -Vertex Connected Component) 指无向图 G 中的一个极大子图 G' ，满足 G' 是一个 k -点连通图，即至少删去 k 个顶点才能破坏其连通性，或者说 G' 的最小点割集的基数为 k 。

k -VCC 在很多领域均有着广泛的应用：在社交网络中，计算出社交关系图的所有 k -VCCs 可以帮助确定社交网络中高度相关的社群用户，为广告和推荐系统提供宝贵的数据支撑；在共同作者网络中，确定 k -VCCs 可以帮助人们确定网络中的研究小组。

本次实验的目的，即为查找无向图网络中的所有 k -VCC。

2 实验完成度

本实验完成了对中、小型数据集的全部 k -VCC 查找，并根据论文的内容实现了简单优化。

程序对于超大数据集无法在可接受的时间完成搜索。

3 实验说明

本实验采用C++程序设计语言编写，使用面向对象的思想，设计类**Graph**和相关方法，实现**k-VCC**的搜索。程序主要部分有：

- 使用最大流算法（*Edmond-Karp Algorithm*）搜索有向图从给定源点到汇点的最大流。
- 使用深度优先搜索，根据最大流算法搜索有向图的最小边割集。
- 使用无向图拆点边有向图的方法，把有向图的最小边割集转化为原图的最小点割集。
- 使用给定的算法，求解全图的最小割。
- 使用**k-VCC**的性质，求出全图的所有**k-VCCs**。
- 使用优化算法，对求解过程进行优化。

本程序使用到的STL库有：

```
1  #include <vector>
2  #include <map>
3  #include <string>
4  #include <iostream>
5  #include <fstream>
6  #include <sstream>
7  #include <set>
8  #include <queue>
9  #include <stack>
10 #include <cstring>
11 #include <algorithm>
```

程序关于类**Graph**的设计如下：

```
1  class Graph
2  {
3  public:
4      // Construction Methods
5      Graph() = default;
6      Graph(const Graph & other) = default;
7      Graph(const string & path);
8      void setK(int s);
```

```

9
10 // Normal Methods
11 void addEdge(int from, int to);
12 void delEdge(int from, int to);
13 void remove(int v);
14 void remove(const set <int>& v);
15 Graph subGraph(const set <int> & v);
16 static vector <Graph> connectComponents(Graph g);
17 Graph genDir();
18 set <int> vertex();
19 set <int> canAccess(int source);
20
21 // Max Flow Methods
22 pair <int, set <Edge> > minEdgeCut(int u, int v);
23 vector <int> BFSFindPath(int u, int v);
24 void reverse(int u, int v);
25
26 // K-VCC Methods
27 set <int> locCut(int u, int v);
28 set <int> globalCut();
29 set <set<int> > kvcc();
30 static set<set<int> > kvccEnum(Graph g, int k);
31 static vector <Graph> overlapPartition(Graph G, const set
<int>& S);
32
33 // Reduction Methods
34 bool isStrongVertex(int u);
35 void sweep(int v, bool * pru, int * deposit);
36 priority_queue <Node> getDistance(int u);
37
38 private:
39     map<int, set <int> > edges;
40     int k;
41     int degree = -1;
42 };

```

程序使用了两个全局函数：

```

1 bool isin(int u, const set<int>& v);
2 void report(string path, const set<set<int> > & result);

```

程序自定义了类型Edge和结构体Node如下：

```

1 typedef pair<int, int> Edge;
2 struct Node
3 {
4     int v;
5     int dis;
6     bool operator<(const Node & other) const
7     {
8         if(dis != other.dis)
9             return dis < other.dis;
10        else
11            return v < other.v;
12    }
13 };

```

4 测试结果

对于给定的数据集（MiniDataSet），运行程序（注意，应事先删除第一行图的总体信息），得出 k 取2、3、5、7时的结果如下：

当 $k = 2$ 时：

```

1 VCC Num: 4
2
3 K-VCC(No. 1): node_nums = 47
4 Node_ID: 0
5 Node_ID: 2
6 Node_ID: 3
7 Node_ID: 5
8 Node_ID: 6
9 Node_ID: 8
10 Node_ID: 9
11 Node_ID: 11

```

12	Node_ID: 14
13	Node_ID: 15
14	Node_ID: 16
15	Node_ID: 17
16	Node_ID: 19
17	Node_ID: 21
18	Node_ID: 22
19	Node_ID: 23
20	Node_ID: 26
21	Node_ID: 27
22	Node_ID: 29
23	Node_ID: 32
24	Node_ID: 36
25	Node_ID: 37
26	Node_ID: 40
27	Node_ID: 41
28	Node_ID: 42
29	Node_ID: 43
30	Node_ID: 44
31	Node_ID: 46
32	Node_ID: 47
33	Node_ID: 48
34	Node_ID: 50
35	Node_ID: 51
36	Node_ID: 52
37	Node_ID: 53
38	Node_ID: 55
39	Node_ID: 57
40	Node_ID: 58
41	Node_ID: 60
42	Node_ID: 64
43	Node_ID: 65
44	Node_ID: 67
45	Node_ID: 69
46	Node_ID: 71
47	Node_ID: 72
48	Node_ID: 73
49	Node_ID: 76
50	Node_ID: 78

```

51
52 K-VCC(No. 2): node_nums = 16
53 Node_ID: 1
54 Node_ID: 4
55 Node_ID: 10
56 Node_ID: 12
57 Node_ID: 28
58 Node_ID: 30
59 Node_ID: 31
60 Node_ID: 35
61 Node_ID: 50
62 Node_ID: 54
63 Node_ID: 62
64 Node_ID: 63
65 Node_ID: 66
66 Node_ID: 68
67 Node_ID: 70
68 Node_ID: 77
69
70 K-VCC(No. 3): node_nums = 5
71 Node_ID: 18
72 Node_ID: 19
73 Node_ID: 20
74 Node_ID: 39
75 Node_ID: 61
76
77 K-VCC(No. 4): node_nums = 6
78 Node_ID: 25
79 Node_ID: 33
80 Node_ID: 38
81 Node_ID: 67
82 Node_ID: 74
83 Node_ID: 75

```

当 $k = 3$ 时:

```

1 # k = 3
2 VCC Num: 5

```

```
3
4 K-VCC(No. 1): node_nums = 15
5 Node_ID: 0
6 Node_ID: 2
7 Node_ID: 3
8 Node_ID: 5
9 Node_ID: 6
10 Node_ID: 11
11 Node_ID: 15
12 Node_ID: 17
13 Node_ID: 23
14 Node_ID: 29
15 Node_ID: 42
16 Node_ID: 44
17 Node_ID: 50
18 Node_ID: 55
19 Node_ID: 60
20
21 K-VCC(No. 2): node_nums = 15
22 Node_ID: 1
23 Node_ID: 4
24 Node_ID: 10
25 Node_ID: 12
26 Node_ID: 28
27 Node_ID: 30
28 Node_ID: 31
29 Node_ID: 35
30 Node_ID: 54
31 Node_ID: 62
32 Node_ID: 63
33 Node_ID: 66
34 Node_ID: 68
35 Node_ID: 70
36 Node_ID: 77
37
38 K-VCC(No. 3): node_nums = 30
39 Node_ID: 8
40 Node_ID: 9
41 Node_ID: 14
```

```
42 Node_ID: 16
43 Node_ID: 21
44 Node_ID: 22
45 Node_ID: 23
46 Node_ID: 26
47 Node_ID: 27
48 Node_ID: 32
49 Node_ID: 36
50 Node_ID: 37
51 Node_ID: 40
52 Node_ID: 41
53 Node_ID: 43
54 Node_ID: 46
55 Node_ID: 47
56 Node_ID: 48
57 Node_ID: 51
58 Node_ID: 52
59 Node_ID: 53
60 Node_ID: 57
61 Node_ID: 58
62 Node_ID: 64
63 Node_ID: 65
64 Node_ID: 69
65 Node_ID: 71
66 Node_ID: 72
67 Node_ID: 76
68 Node_ID: 78
69
70 K-VCC(No. 4): node_nums = 4
71 Node_ID: 18
72 Node_ID: 20
73 Node_ID: 39
74 Node_ID: 61
75
76 K-VCC(No. 5): node_nums = 6
77 Node_ID: 25
78 Node_ID: 33
79 Node_ID: 38
80 Node_ID: 67
```



```
81 Node_ID: 74
82 Node_ID: 75
```

当 $k = 5$ 时:

```
1  # k = 5
2  VCC Num: 4
3
4  K-VCC(No. 1): node_nums = 6
5  Node_ID: 2
6  Node_ID: 3
7  Node_ID: 6
8  Node_ID: 15
9  Node_ID: 42
10 Node_ID: 60
11
12 K-VCC(No. 2): node_nums = 6
13 Node_ID: 4
14 Node_ID: 30
15 Node_ID: 54
16 Node_ID: 62
17 Node_ID: 66
18 Node_ID: 68
19
20 K-VCC(No. 3): node_nums = 19
21 Node_ID: 9
22 Node_ID: 14
23 Node_ID: 21
24 Node_ID: 22
25 Node_ID: 32
26 Node_ID: 36
27 Node_ID: 37
28 Node_ID: 40
29 Node_ID: 41
30 Node_ID: 43
31 Node_ID: 47
32 Node_ID: 52
33 Node_ID: 58
```

```
34 Node_ID: 64
35 Node_ID: 69
36 Node_ID: 71
37 Node_ID: 72
38 Node_ID: 76
39 Node_ID: 78
40
41 K-VCC(No. 4): node_nums = 6
42 Node_ID: 27
43 Node_ID: 46
44 Node_ID: 48
45 Node_ID: 51
46 Node_ID: 53
47 Node_ID: 57
```

当 $k = 7$ 时:

```
1 # k = 7
2 VCC Num: 1
3
4 K-VCC(No. 1): node_nums = 15
5 Node_ID: 9
6 Node_ID: 32
7 Node_ID: 36
8 Node_ID: 37
9 Node_ID: 40
10 Node_ID: 41
11 Node_ID: 43
12 Node_ID: 47
13 Node_ID: 52
14 Node_ID: 58
15 Node_ID: 64
16 Node_ID: 69
17 Node_ID: 72
18 Node_ID: 76
19 Node_ID: 78
```

附录A 测试数据集

MiniDataSet的点对表示如下:

1	0	5
2	0	11
3	0	17
4	0	23
5	0	29
6	0	44
7	0	55
8	1	30
9	1	31
10	1	35
11	1	54
12	1	62
13	1	63
14	1	68
15	1	77
16	2	3
17	2	6
18	2	15
19	2	42
20	2	60
21	3	6
22	3	15
23	3	42
24	3	50
25	3	60
26	4	10
27	4	30
28	4	54
29	4	62
30	4	66
31	4	68
32	4	70
33	5	11
34	5	17
35	5	23

36	5	29
37	5	44
38	5	55
39	6	11
40	6	15
41	6	42
42	6	60
43	7	45
44	8	23
45	8	27
46	8	44
47	8	51
48	8	53
49	9	32
50	9	36
51	9	37
52	9	40
53	9	41
54	9	43
55	9	47
56	9	58
57	9	64
58	9	72
59	9	76
60	9	78
61	10	66
62	10	68
63	10	70
64	11	15
65	11	55
66	11	60
67	12	28
68	12	31
69	12	35
70	12	50
71	12	63
72	12	77
73	13	33
74	13	34

75	14	21
76	14	22
77	14	32
78	14	41
79	14	47
80	14	71
81	14	73
82	15	42
83	15	50
84	15	60
85	16	26
86	16	36
87	16	40
88	16	65
89	16	67
90	17	23
91	17	29
92	17	55
93	18	19
94	18	20
95	18	39
96	18	61
97	19	22
98	19	39
99	19	73
100	20	39
101	20	61
102	21	22
103	21	32
104	21	41
105	21	47
106	21	49
107	21	71
108	21	72
109	22	32
110	22	47
111	22	71
112	22	73
113	22	78

114	23	27
115	23	29
116	23	44
117	23	53
118	24	49
119	25	33
120	25	74
121	25	75
122	26	36
123	26	40
124	26	48
125	26	57
126	26	65
127	26	67
128	27	46
129	27	48
130	27	51
131	27	53
132	27	57
133	28	31
134	28	50
135	28	63
136	29	55
137	30	54
138	30	62
139	30	63
140	30	66
141	30	68
142	30	77
143	31	35
144	31	63
145	31	77
146	32	37
147	32	41
148	32	43
149	32	47
150	32	64
151	32	71
152	32	72

153	32	76
154	32	78
155	33	38
156	33	67
157	33	74
158	35	63
159	35	77
160	36	37
161	36	40
162	36	43
163	36	46
164	36	48
165	36	52
166	36	57
167	36	64
168	36	65
169	36	69
170	36	72
171	37	40
172	37	41
173	37	43
174	37	47
175	37	52
176	37	58
177	37	64
178	37	72
179	37	76
180	37	78
181	38	67
182	38	74
183	38	75
184	39	61
185	40	43
186	40	46
187	40	48
188	40	52
189	40	64
190	40	65
191	40	69

192	40	72
193	40	76
194	41	43
195	41	47
196	41	52
197	41	64
198	41	71
199	41	72
200	41	76
201	41	78
202	42	44
203	42	50
204	42	60
205	43	47
206	43	52
207	43	58
208	43	64
209	43	69
210	43	71
211	43	72
212	43	76
213	43	78
214	46	48
215	46	51
216	46	53
217	46	57
218	47	58
219	47	64
220	47	71
221	47	72
222	47	76
223	47	78
224	48	51
225	48	53
226	48	57
227	51	53
228	51	57
229	52	64
230	52	65

231	52	69
232	52	72
233	52	76
234	53	57
235	54	62
236	54	66
237	54	68
238	54	70
239	55	60
240	56	59
241	58	64
242	58	72
243	58	78
244	62	66
245	62	68
246	62	77
247	63	77
248	64	69
249	64	72
250	64	76
251	64	78
252	65	69
253	66	68
254	66	70
255	67	75
256	68	70
257	69	72
258	69	76
259	71	72
260	71	76
261	72	76
262	72	78
263	74	75