웹 환경에서 전회로나열 다항식 알고리즘

김능회, 마지형, 이우기
성결대학교 컴퓨터학부, 경기도 안양시 안양8동 산 147-2, 430-740
nunghoy@hanmail.net, mjh831@nate.com, wookeylee@gmail.com

Circuit Enumeration Polynomial Algorithm in Web Environment

Nunghoi Kim, Jihyong Ma, Wookey Lee
Div. Computer Science, Sungkyul University, Anyang, Korea

요 약

웹의 규모가 시간이 지날수록 팽창하고 종류가 방대해져 가면서 그 구조화가 중요해지고 있다. 웹은 홈페이지와 여러 가지 일반 웹 페이지들이 하이퍼텍스트 링크로서 서로 복잡하게 연결된 유방향 그래프(directed graph)의 특성을 가진다. 그러나 웹을 그래프로서 탐색하면 중복된 링크나 다시 같은 페이지로 돌아오는 링크 즉, 회로(circuit)들이 무수히 존재하고, 또한 그 경로의 길이가 방대하기 때문에 구조화에 있어 큰 장애가 된다. 본 연구에서는 웹에서 회로를 효과적으로 탐색 및 나열하는 알고리즘을 제시하였다. 기존의 회로 탐색 중심의 알고리즘에 비해 복잡도가 개선된 전회로나열 다항식 알고리즘을 개발하고, 입증하였다.

1. 서론

웹(WWW)은 다양한 정보들의 집합체가 되면서 정보의 양이 폭발(explosive)하고 있다. 이러한 웹 정보를 효과적으로 검색 저장하기 위하여 구조화해야하는 중요성이 점점 커지고 있다. 웹에는 수많은 링크가 있고, 이로 인해 계속 발산하거나, 특정 웹페이지에서 출발하여 다시 돌아오는 중복된 링크가 많이 존재한다. 특히 중복된 경우를 '회로(Circuit)'라고 하는데, 이러한 회로는 웹구조화의 독특한 특성이며 주요한 장애요소의 하나가 된다. 본 연구는 이러한 웹 회로를 효과적으로 탐색 나열하는 알고리즘을 제안하고 분석하고자 한다.

웹은 아래 그림 1과 같이 웹 페이지는 웹의 vertex로써 또한 하이퍼텍스트 링크는 edge로 인식하는 그래프접근법(Web as a graph)이 다수 이론이다[1, 2, 3]. 웹은 실제 그래프의 형태와 매우 흡사하고, 그래프로 표현 시 저장과 탐색에서 그래프에서의 해법들을 적용하기가 용이하다. 따라서 본 논문은 웹페이지를 구조화하는데 문제가 되는 회로를 나열하기 위하여, 웹페이지를 그래프의 형태로 표현하고, 그 그래프를 탐색하여 모든 회로를 찾아내고 회로의 중복을 제거하는데 목적이 있다.

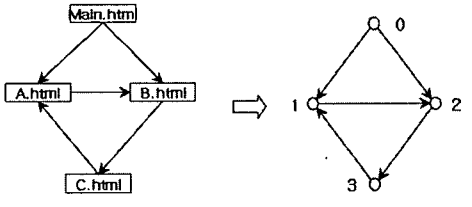


그림 1. 웹의 그래프 표현

2. 회로(Circuit) 및 회로경로(Circuit Path)

유방향 그래프(Directed graph)로서 G=(V,E)는 vertices V와 G의 edge로 불리어지는 vertices E의 정리된 쌍으로 구성되어 있다. 만약 (v, w)가 G의 edge라면, vertices v와 w는 인접(adjacent)하다. graph에서 path는 edges (v_1, v_2), (v_2, v_3), ⋯ ,(v_{n-1}, v_n)의 sequence 로서, 나중의 vertex는 다음 edge의 처음 vertex 이다. path는 vertices의 연속에 의해서 나타내어진다. elementary path는 같은 vertex를 내포하지 않는다. elementary circuit는 elementary path에서 그 처음과 마지막 vertices가 일치하는 path 이다. 두 개의 elementary circuit이 각각의 회로 순열이 동일할 경우 중복 circuit 라고 본다. 이 경우 (v, v) edge의 경우처럼 self-circuit이 없는 경우를 가정한다. 또한 C = $(v_1, v_2, \cdots, v_n, v_1)$이 graph G에서 elementary circuit 이다. satisfying $v_i > v_1$, for $2 \leq i \leq n$. elementary circuit path는 (v_1, v_2), (v_2, v_3), ⋯ , (v_{n-1}, v_n), (v_n, v_1)의 sequence 이다.

elementary circuit이 각각 graph G에서 $C_1 = (v_1, v_2, \cdots, v_n, v_1)$이고, $C_2 = (v_2, v_3, \cdots, v_n, v_1, v_2)$일 때, C_1과 C_2는 같은 circuit을 가지는 즉, 중복(Repetition) 회로(circuit)라 부른다.

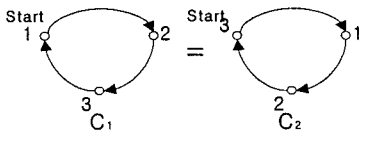


그림 2. 중복회로의 예

3. 알고리즘 분석 및 복잡도

3.1 회로탐색 알고리즘

회로탐색 알고리즘은 기본적으로 깊이 우선 탐색 방법으로 순회한다 [3, 4, 5]. 정점을 방문하는 순서대로 스택에 저장하고 방문기록을 남김으로써 방문한 정점을 방문할 경우 회로가 검출되도록 하는 방식이다. 정점이 V개, 간선이 E개 일 때, 탐색하는데 걸리는 시간은 (7)에 의해 O(E) 이고, 회로가 발견될 때마다 걸리는 시간은 (6)과 (7), 그리고 (14)에 따라 O(VEC)를 가진다. 따라서 복잡도는 O(E·V·(C + 1))를 가진다. (14)에서는 스택에 있는 모든 정점과 새로 들어가는 정점이 모두 비교해야 함으로 O(V·E·C(C-1)/2)의 복잡도를 가진다. 하지만 중복 회로를 다수 찾는다는 단점이 있다.

```
(1) procedure circuit enumeration
(2)    begin
(3) procedure VisitRec(integer value v)
(4)     begin
(5)       mark (v) := true;
(6)         s := 1 do
(7)         for w ∈ A(v) do
(8)         if(mark (w) = false) then
(9)           begin
(10)              place w on stack;
(11)            VisitRec(w);
(12)          end
(13)        else if
(14)          output circuit from s to v to s given by stack;
(15)      u := top of stack;
(16)      delete u from stack;
(17)      mark (u) := false;
(18)    end
(19)  end
```

그림 3.1 VisitRec 알고리즘

3.2 ReNode 알고리즘

VisitRec 회로탐색 알고리즘에서 발견하지 못하는 회로를 찾기 위해서 회로탐색 후 모든 정점을 방문하였는지 확인하여 방문하지 않은 정점에서 다시 VisitRec 회로탐색을 하는 방법이 있다[1]. 탐색하는데 걸리는 시간은 (6)과 (18)에 의해 O(VE)이며, 회로가 발견될 때마다 걸리는 시간은 (6)과 (18), 그리고 (13)에 의해 O(V·E·C)가 된다. 그래서 복잡도는 O(V·E·(C+1))를 가진다. (13)에서는 VisitRec 회로탐색과 마찬가지고 스택에 있는 모든 정점을 비교해 보아야 함으로 V ·(C(C-1)/2))의 복잡도를 가지고, 전체적으로 O(V·E·C^2(C-1)/2)의 복잡도가 된다. 하지만 이는 VisitRec 회로탐색에서 발견한 중복회로보다 더 많은 중복회로가 발생하는 단점이 있다.

```
(1) procedure circuit enumeration
(2)    begin
(3) procedure ReNode(integer value v)
(4)     begin
(5)       mark (v) := true;
(6)       for w ∈ A(v) do
(7)         if(mark (w) = false) then
(8)           begin
(9)             place w on stack;
(10)            ReNode(w);
(11)          end
(12)        else if
(13)          output circuit from s to v to s given by stack;
(14)      u := top of stack;
(15)      delete u from stack;
(16)      mark (u) := false;
(17)    end
(18)    for s := 1 until v do
(19)   Remove repetitive circuit
(20)  end
```

그림 3.2 ReNode 알고리즘

3.3 RemoveN 알고리즘

본 연구에서 제시하는 방식은 깊이 우선 탐색으로 순회하면서, 모든 정점에서 회로를 탐색하며, 회로는 출발한 정점으로 시작하는 회로만을 탐색한다. 그리고 한번 방문한 정점을 제거하여 중복된 회로를 발견하지 않는다. 모든 정점에서 회로탐색을 하므로 모든 그래프를 탐색하는데 걸리는 시간은 (6)과 (21)에 의해 $O(VE)$을 가지고, 정점에서 회로가 발견될 때마다 걸리는 시간은 (6)과 (21), 그리고 (16)에 의해 $O(VEC)$을 가진다. 그래서 $O(VE(C + 1))$의 복잡도를 가진다. (16)에서는 시작점으로 시작하는 circuit만을 찾아내기 때문에 스택에 있는 모든 정점과 비교하지 않는다. 시작점이 곧 circuit의 처음 정점이다. 그래서 O(1)의 복잡도를 가진다.

```
(1) procedure circuit enumeration
(2)    begin
(3) procedure RemoveN(integer value v, integer value s)
(4)     begin
(5)       mark (v) := true;
(6)       for w ∈ A(v) do
(7)         if(w ≠ s) then
(8)          begin
(9)            if(mark (w) = false) then
(10)             begin
(11)               place w on stack;
(12)               RemoveN(w, s);
(13)             end
(14)         end
(15)       else if
(16)         output circuit from s to v to s given by stack;
(17)     u := top of stack;
(18)     delete u from stack;
(19)     mark (u) := false;
(20)    end
(21)     for s := 1 until v do
(22)          place s on stack;
(23)          RemoveN(s, s);
(24)          u := top of stack;
(25)          delete u from stack;
```

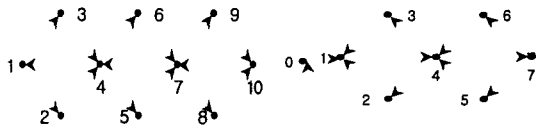
```
(26)        delete s from A(v)
(27)    end
```

그림 3.3 RemoveN 알고리즘

4. 성능평가

Tiernan과 Weinblatt 모형[7]을 가지고 실험하였다. 그 결과 RemoveN 방식은 완전그래프에서 이론적으로 발생가능한 모든 회로를 검출하며 중복된 검출을 하지 않는다. 그러나 VisitRec방식이나 ReNode방식의 경우는 표와 같이 노드수가 증가할수록 중복 회로를 찾아내고 또한 이의 중복여부를 확인하기위한 수행도에서의 약점이 노정됨을 확인할 수 있었다.

다음 표에서 보이는 바와 같이 임의로 정점 수를 변화시키면서 비교하였다. 정점에 따른 회로개수 비율로서 간선이 많아질수록, 단일출발점 회로탐색이 급격하게 많은 회로개수를 발견하는 것을 알 수 있다.



(a) Tiernan 모델 (b) Weinblatt 모델

그림 4.1 실험에 사용된 모델

표 4.1 Tiernan 모델의 성능평가

정 점 의 수	4	7	10	13	16
VisitRec	2	6	14	30	62
ReNode	4	12	28	60	124
RemoveN	2	4	6	8	10

표 4.2 Weinblatt 모델의 성능평가

정점의수	5	8	11	14	17
VisitRec	6	12	22	40	74
ReNode	12	24	44	80	148
RemoveN	4	6	8	10	12

정점에 따른 회로탐색시간에 있어 VisitRec 회로탐색이 RemoveN 회로탐색보다 급격하게 증가하는 것을 알 수 있다. 정해진 정점에서 간선이 증가함에 따라 기하급수적으로 회로가 많이 발생하게 되는, VisitRec 회로탐색 알고리즘은 중복된 회로까지 발견하게 되므로 Edge가 증가함에 따라 회로수가 크게 증가하는 것을 볼 수가 있다.

위 실험과 같이 VisitRec 및 ReNode 알고리즘은 중복된 회로가 많이 발견됨으로 중복을 제거하려면, 발견된 회로경로를 일일이 비교해서 같은 경로를 제거해야만 하기 때문에 많은 연산량이 요구된다는 것을 알 수 있다. 그러므로 RemoveN 알고리즘은 WWW처럼 규모가 커질수록 효과적인 방식이 된다는 것을 알 수 있다.

5. 결론

본 연구에서는 웹을 구조화하기 위해 방향그래프를 이용해 모든 회로를 탐색하고 그 경로를 발견하는 알고리즘을 제시하였다.

본 연구의 공헌요소는 다음과 같다. 첫째, 웹 환경을 그래프로 전환하여 문제를 해결하는 첫 접근법이다. 둘째, 기존의 알고리즘이 회로를 찾는 데에서는 유사한 성능을 보이지만, 대규모 그래프에서는 회로를 중복적으로 검출하므로 복잡도와 성능이 나빠지는 원인을 밝혔다. 셋째, 웹의 경우 오프라인 분석보다 검색로봇 등의 즉각적(on the fly) 수행도가 중요하므로 본 연구와 같은 다항식 복잡도 알고리즘을 제시하고 입증하였다. 향후 웹 구조화 관련 영역에 적용할 계획이다.

Acknowledgement

본 연구는 청단정보기술연구센터를 통하여 과학기술부/한국과학재단의지원을 받았음.

참고문헌

[1] S.Chakrabarti, B.Dom, R.Kumar, P.Raghavan, S.Rajagopalan, A.Tomkins, D.Gibson, J.Kleinberg, "Mining the Web's Link Structure," IEEE Computer 32(8): 60-67, 1999.

[2] W. Lee, "Discriminating Intentionally Spammed Web Pages Using Lexical Signature Filtering Algorithm," LNCS 3733: 585-594, 2005.

[3] W. Lee, G. James, "Semantic Hierarchical Abstraction of Web Site Structures for Web Searchers," Journal of Research and Practice of Information Technology, 36(1): 71-82, 2004.

[4] G. Nivasch, "Circuit Detection Using a Stack", Information Processing Letters, Volume 90, 2004

[5] S. Sahni, E. Horowitz, Fundamental of Computer Algorithm, Addison-Wesley, 1985

[6] J. Szwarcfiter, G. Navarro, R. Baeza-Yates, J. Oliveira, W. Cunto, N. Ziviani, "Optimal binary search trees with costs depending on the access paths", Theoretical Computer Science, 290(3): 1799-1814, 2003

[7] J. Szwarcfiter, P. Lauer, A Search Strategy for the elementary Circuits of a directed graph, 1976