韓國컴퓨터情報學會 論文誌
第16卷 第12號, 2011. 12.

2011-16-12-1-4

순환 그래프 마이닝에서 중복된 그래프 패턴의 확장을 피하는 효율적인 기법

노 영 상*, 윤 은 일*, 편 광 범*, 양 흥 모*, 이 강 인*, 류 근 호*, 이 경 민**

An efficient approach of avoiding extensions of duplicated graph patterns in cyclic graph mining

YoungSang No*, Unil Yun*, Gwangbum Pyun*, Heungmo Ryang*, Gangin Lee*, Keun Ho Ryu*, Kyung-Min Lee**

요 약

그래프 마이닝에서 복잡한 그래프 구조로 인해, 중복된 확장 연산이 수행되며 이로인해 낮은 효율성을 가지게 된다. 본 논문에서는 순환그래프에서 중복된 그래프 패턴으로의 확장을 최소화하기위해 중복 판단을 효율적으로 하는 그래프 마이닝 알고리즘을 제안한다. 제안하는 기법에서는 순환간선의 우선순위를 고려하여 우선순위가 낮은 간선을 먼저 확장하게 함으로써 중복확장을 줄이도록 하였다. 이 기법의 성능을 평가하기 위해, 알고리즘을 구현하고 그래프 마이닝의 대표 알고리즘인 가스톤 알고리즘과 성능 평가를 하였으며, 제안하는 알고리즘이 복잡한 그래프 구조에서 반복되어 발생하는 연산중 하나인 순환 그래프에서 패턴 확장 시에 필요한 연산을 효율적으로 줄이도록 하여 전체 마이닝의 성능이 향상됨을 보인다.

▸ Keyword : 순환 그래프, 중복 판단, 서브 그래프 마이닝, 패턴 확장

Abstract

From Complicated graph structures, duplicated operations can be executed and the operations give low efficiency. In this paper, we propose an efficient graph mining algorithm of minimizing the extension of duplicated graph patterns in which the priorities of cyclic edges are considered. In our approach, the cyclic edges with lower priorities are first extended and so duplicated extensions can

• 제1저자 : 노영상 • 교신저자 : 윤은일
• 투고일 : 2011. 07. 16, 심사일 : 2011. 08. 04, 게재확정일 : 2011. 10. 30.
* 충북대학교 전자정보대학 컴퓨터과학과 및 컴퓨터정보통신연구소(Dept. of Computer Science, Chungbuk National University)
** 텍사스 A&M 대학교, 부설 텍사스 농업생명 연구소, 텍사스 화학 물질 규제 연구실 (Office of the Texas State Chemist, Texas AgriLife Research, Texas A&M University)
※ "이 논문은 2011년도 충북대학교 학술연구지원사업의 연구비 지원에 의하여 연구되었음 (This work was supported by the research grant of the Chungbuk National University in 2011)".

be reduced. For performance test, we implement our algorithm and compare our algorithm with a state of the art, Gaston algorithm. Finally, We show that ours outperforms Gaston algorithm.

▸ Keyword : Cyclic graph, duplication estimation, Subgraph mining, pattern extension

I. 서 론

데이터 마이닝 [1, 4]의 개념이 제안되어 온 이래로 다양한 마이닝 기법이 제시되어 왔으며 최근에는 보다 복잡한 구조를 가진 데이터에 대한 마이닝 알고리즘이 제안되고 있다. 이들 마이닝 알고리즘들은 다양한 응용 영역[4]을 가지고 있기 때문에 폭넓게 사용되고 있다.

그래프는 G 또는 g로 표현하기로 정하며, 그래프를 구성하고 있는 노드의 집합을 V로 정하고, 간선의 집합을 E = {(vl,v2) | v1,v2∈V and v1≠v2}로 정한다. 간선에는 방향성이 없으므로, 두 간선 (v1,v2)선 (v2,v1)는 같은 간선으로 본다. 구성요소들이 가질 수 있는 레이블 집합은 L로 표현하며, 함수 l(V)나 l(E)는 각 간선이나 노드의 레이블을 반환한다. 두 개의 므로, G1, G2가 있을 때, 그래프 G1의 내부 구조 안에 그래프 G2의 구조와 동일한 구조의 므로,가 있을 때 그래프 G2를 G1의 없그래프(Subgraph)라 하고, G1⊃G2로 표현한다. 본 논문의 범위에서 동일한 구조란 노드와 간선의 구조가 정확하게 일대일로 상대되며, 또한 그 각각의 레이블이 동일한 경우를 말한다. 수식으로 표현했을 때, 그래프 G1=(V1, E1, L1), G2=(V2, E2, L2)에 대하여 G1⊃G2 라면, 다음의

두 식을 만족하는 일대일 함수 f(V1) → V2 가 존재한다.

⑴. ∀v∈V1 ⇒ l(v)=l(f(v))

⑵. ∀(v1,v2)∈E1 ⇒ (f(v1),f(v2))∈E2

and l1(v1,v2)=l2(f(v1),f(v2))

G1⊃G2이고 G2⊃G1인 그래프를 동형그래프라고 한다. 경로(path)란 두 노드 사이에서 간선들을 통한 연결을 말하며, 하나의 그래프 내에서 모든 노드들 사이에 경로가 존재할 경우 이를 연결그래프(connected graph)라고 한다. 경로그래프란 순환이 포함되지 않는 단순경로그래프이며 순환그래프(cyclic graph)란 그래프 내에 순환구조(cycle)가 포함되어 있는 일반적인 그래프이다. 그래프 패턴의 확장탐색에서 패턴들은 먼저 경로그래프(path)에서 시작하여 점차 확장되다가 자유트리그래프 또는 순환그래프의 패턴으로 확장되게 된다. 순환그래프는 같은 노드의 개수를 가지는 여러 다른 형태의 트리구조들을 포함하고 있다. 이러한 트리들을 그래프의 스패닝 트리라고 한다. 스패닝 트리들은 자유트리의 정규형 표현으로서 서로 우선순위를 비교할 수 있다. 이중 트리의 크기가 가장 작고(즉, 최대길이경로가 가장 작은 트리), 중심노드를 루트로 한 정규형 트리의 우선순위가 가장 높은 스패닝 트리를 하나 선정할 수 있다. 이는 트리영역에서 백본이 되는 경로그래프가 하는 일과 마찬가지로 순환그래프의 영역을 각각의 트리그래프들로 분할한다고 할 수 있다. 그러므로 모든 순환그래프들은 이 가장 높은 우선순위를 가지는 트리로부터만 확장될 수 있도록 허용함으로써 중복된 확장을 하지 않도록 만들 수 있다. 그러나 백본제약처럼 최고우선순위트리를 변화시키지 않는 순환간선의 확장방법은 아직 없다 [10]. 그러므로 순환간선을 확장하여 패턴을 생성한 후 별도의 확인절차를 걸쳐서 중복여부를 판단한다. 하지만 그래프내의 모든 스패닝 트리를 나열하는 방법이 지수 함수적 계산시간이 필요하다[5]. 게다가 자유트리로부터 모든 순환간선의 확장을 허락하므로, 생성된 그래프의 순환간선의 개수만큼의 중복생성이 일어나게 된다. 뿐만 아니라 패턴이 확장될 때 마다 스패닝 트리를 완전히 새로 찾아야 하기 때문에 스패닝 트리를 이용한 순환그래프의 동형판단은 매우 비효율적이다. 그래프 트랜잭션 데이터베이스는 하나의 트랜잭션의 하나의 연결그래프로 구성되어있다. 어떤 그래프 구조가 여러 개의 트랜잭션에서 나타난다면, 이 구조는 관심의 대상이 된다.

마이닝 작업은 일정 빈도수 이상의 이러한 패턴들을 모두 찾아내는 작업이다. 사용자가 관심을 가지는 최소 지지도 임계값 (minimum support threshold)를 제시하면, 그것과 같거나 그 이상의 빈도수를 가지는 서브그래프패턴들을 모두 찾아내게 된다. 한 가지 주의 할 점은 어떤 서브그래프는 하나의 트랜잭션에서 여러 번에 걸쳐서 나타날 수 있으나 빈도수에는 단 한 번만 계산된다는 것이다. 그래프마이닝은 기존의 패턴 마이닝과 다르게 그래프마이닝 작업의 전체 소요시간에서 그래프 동형판단이 차지하는 비율은 매우 높다. 그렇기 때문에 그래프마이닝 알고리즘은 그래프 동형판단의 문제를 최대한 효과적으로 해결해야 한다.

본 논문은 성능에 영향을 많이 미치는 그래프 동형판단을 효율적으로 하기 위해 확장된 패턴을 활용하여 불필요한 확장을 효과적으로 회피하는 마이닝 기법을 제안하여 사이클을 가진 복잡한 그래프 데이터를 효과적으로 마이닝 하는 알고리즘을 구현하며 성능을 평가한다.

II. 관련 연구

그래프 마이닝의 초기 알고리즘[4, 5]은 Apriori 알고리즘[2, 5, 8]을 기반으로 한 방법들을 제시했었다. 하지만 이는 그래프동형판단의 문제로 매우 많은 계산시간을 요구한다[2, 5, 8]. 패턴확장기법은 트리구조의 마이닝 알고리즘에서 먼저 적용되었다. 깊이우선탐색을 이용하여 정규형 트리를 정의함으로써 패턴확장기법을 효과적으로 구현하게 된다. 대부분의 그래프마이닝 알고리즘들은 트리구조를 기반으로 그래프의 동형판단[10, 14]을 하게 되며 이를 통해 확장탐색도중 생성된 그래프패턴의 중복성 판단을 효과적으로 할 수 있다.

그래프 마이닝에 패턴확장기법을 처음 적용한 알고리즘은 지스팬(gspan) 알고리즘 [14]이다. 지스팬은 깊이우선탐색에서의 탐색우선순위와 레이블의 우선순위를 결합하여 탐색우선순위를 정함으로써, 특정한 한 노드를 루트로 하는 확장에 대해서는 중복생성유무의 판단을 완전히 배제시켰다. 이로서 매우 좋은 효과를 볼 수 있으나, 그래프는 어떤 한 노드를 루트로 하지 않기 때문에 중복생성을 피할 수는 없다. 이런 중복생성에 대해서는 이미 찾아진 그래프인지 판단하기 위하여 이미 찾아진 것들과 동형판단을 해야만 한다. 하지만 이 알고리즘은 서브그래프와 트랜잭션과의 동형판단 작업에 대한 효율성은 제시하지 못한다.

그래프 마이닝에 대한 성능을 향상시키는 연구가 지속적으로 되어왔고 좋은 성능을 가진 알고리즘들 [6, 7]이 제안되어 왔다. 또한, 가중화 그래프 마이닝 [3, 9], 제한조건을 이용한 그래프 마이닝기법 [10], 상관관계를 가지는 그래프 마이닝 [11], 중요 그래프 마이닝 [12], 대표그래프 패턴 마이닝 [15], 근접 그래프 패턴 마이닝 [17]등 심화된 그래프 패턴 마이닝에 대한 연구가 지속적으로 이루어지고 있다. 이중 패턴 확장 기법을 이용하여 효율적 그래프 패턴을 마이닝 하는 대표적인 알고리즘이 가스톤(gaston)[10]이다. 가스톤은 패턴확장도중 생겨나는 모든 패턴들을 경로그래프(path), 자유트리그래프(tree), 순환그래프(cyclic graph)로 나누어 중복된 패턴의 확장을 최소화하는 좋은 성능을 보인다. 특히, 가스톤 알고리즘은 자유트리영역에서의 중복생성에 대하여 매우 효과적인 제어를 한다. 유일한 경로그래프로부터 확장된 유일한 자유트리는 중복판단을 하지 않고도 중복생성을 제어할 수 있어 좋은 성능향상을 가진다 [10].

III. 패턴 확장 알고리즘

패턴확장기법(pattern-growth method) [14]은 후보자를 생성하지 않고 빈발한 그래프에서 빈발하는 간선을 확장해 가며 패턴을 찾는 방법이다. 이는 후보자패턴의 생성 없이 빈발하는 간선만을 확장하며 탐색한다.

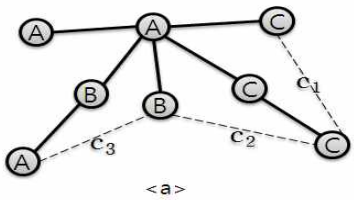
표 1. 그래프 패턴 확장 알고리즘

```
1:  Input : database D, minmum frequency threshold f
2:  Output : S // frequenct
3:  Define : legs set L
4:  Sub_Tree( treepattern T, leg l, legsset L, frequence f ){
5:    generat pattern T' from T with l;
6:    if P' is duplicate pattern then return ∅;;
7:    set S = ∅; L' = ∅;
8:    L' = L' ∪ Extension( l, f );
9:    for( i=0; i<size of L; i++)
10:     L' = L' ∪ Join( l, L[i], f );
11:   for( i=0; i<size of L'; i++){
12:     if ( legs[i] is node refinement ){
13:       S = S ∪ Sub_Tree( T', legs[i], L', f );
14:     }
15:     if ( legs[i] is cyclic refinement ){
16:       S = S ∪ Sub_Graph( T', legs[i], L', f );
17:     }
18:   }
19:   return S;
20: }
21: Sub_Graph( cyclicpattern G, leg l, L, f ){
22:   G' = generate_cyclicgraph( G, l );
23:   if  G' is duplicate pattern then return ∅;
24:   set S = ∅, L' = ∅;
25:   for( i=0; i<size of L; i++)
26:     if( legs[i]==cyclic & legs[i]≤l of priority )
27:       L'=L'∪Join(l,legs[i],f);
28:   for( i=0; i<size of L'; i++){
29:     S = S ∪ CyclicGraph( G', legs[i], L', f );
30:   }
31:   return S;
32: }
33: Join( leg l1, leg l2, frequency f ){
34:   set l'.refinement = l2.refinement;
36:   set l'.embeddinglist =
37:     {  (k, tj.graph, tj.node) |
38:       tk∈l1.embeddinglist, tj∈l2.embeddinglist,
39:       tk.parent == tj.parent   };
40:   if ( l' ≥ f ) return l' else return ∅;
41: }
42: Extension( leg l, frequency f ){
43:   set candidate legs C;
44:   for(i=0; i<size of embedding list of l; i++){
45:     for(j=0; j<size of all adjacent node of tk[i]' node;
j++){
46:       if ( adjacent[j] is an ancestor of tk ){
47:         append index k to c.embeddinglist of
48:         the corresponding cycle closing leg c∈C;
49:       }else {
50:         append index ( k, adjacent[j], tk.graph )
51:         to c.embeddinglist of
52:         the corresponding node refinement leg c∈C;
```

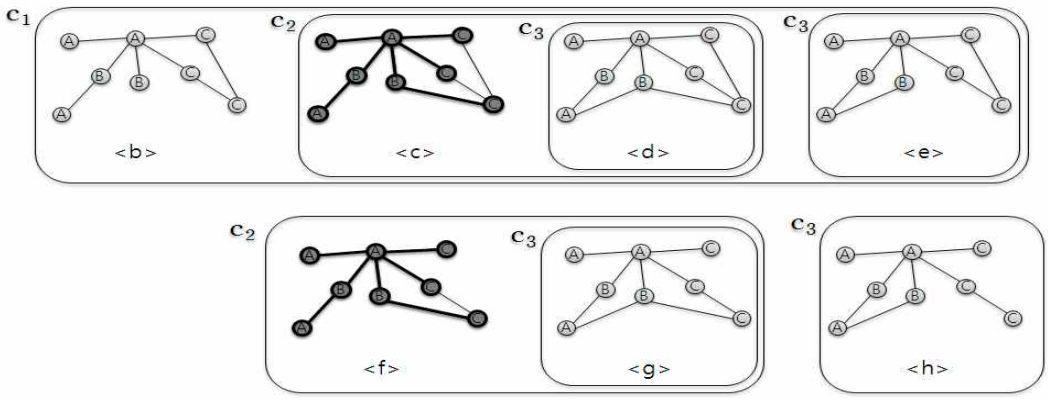
```
53:          }
54:       }
55:    }
56:    set frequent legs F = { c | c( ≥ f ) ∈C };
57:    return F;
58: }
```

후보자가 없는 것은 좋으나, 단순히 빈도수에만 의지해 확장탐색을 한다면 어떤 K크기의 그래프에 대해서 K번의 중복 생성이 있게 될 것이다. 표 1은 패턴확장기법의 기본적인 알고리즘이 다. 패턴확장기법은 먼저 데이터베이스에서 모든 빈발하는 간선을 찾은 뒤, 각각의 빈발간선들로부터 확장 될 수 있는 빈발간선들을 모두 찾아 확장해 나간다. 각각의 확장된 패턴들은 데이터베이스의 탐색을 통하여 확장될 수 간선들을 확인하고 하나씩 차례대로 확장해 나간다. 이때 재귀적인 방법을 통하여 확장된 패턴에서 확장 가능한 패턴들을 다시 모두 찾는다. 알고리즘은 어떤 간선의 확장으로 찾을 수 있는 모든 패턴을 찾은 후에야 다시 그 패턴으로 돌아와 다른 간선을 확장시키고 다시 재귀적 방법을 통하여 패턴을 찾는다. 이러한 반복적인 작업은 더 이상 반복되는 패턴이 발견되지 않을 때 까지 계속된다 (line 4-32).

패턴확장기법은 모든 빈발하는 간선을 확장시키므로 중복된 데이터를 생성할 수도 있다. 확장된 패턴이 중복된 패턴이라면 알고리즘은 더 이상 빈발하는 패턴을 찾지 않고 백트랙킹을 한다. 중복된 패턴에서 생성되는 패턴역시도 중복된 패턴이기 때문이다(line 6, 23). 데이터마이닝에서 빈도수계산의 효율성은 매우 중요한 부분이다. 데이터베이스를 반복적으로 스캔하지 않도록 빈발하는 간선들이 데이터베이스에 나타난 위치들을 모두 기억할 수 있다. 깊이우선탐색의 트리구조를 이용하게 되면 이 위치데이터들 간의 연관성을 알 수 있다. 확장된 패턴에서의 확장될 수 있는 빈발간선들의 위치데이터는 확장되기 이전패턴의 빈발간선들의 위치데이터에 모두 포함된다. 이로 인해 확장된 패턴에서의 빈발간선들의 위치를 찾을 때 확장되기 이전의 빈발간선들이 위치데이터에서만 탐색함으로써 계산을 줄일 수 있게 된다 (line 33, 58).



< 기본 자유트리와 확장 가능한 순환간선 (c_1, c_2, c_3) >



< 패턴확장기법에 의해 생겨나는 패턴의 종속성 >
(순환간선의 우선 순위 : c_1 > c_2 > c_3)

그림 1. 확장 가능한 순환간선 및 패턴의 종속성

IV. 패턴 확장 알고리즘

1. 순환그래프의 중복 판단의 필요성

가스톤 알고리즘 [10]은 확장탐색영역을 위와 같이 3부분으로 나누어 다룸으로써 수행시간을 효과적으로 단축시켰다. 확장된 패턴이 간단한 구조일수록 중복성 판단을 효과적으로 할 수 있기 때문이다. 또한 트리그래프 패턴들은 그 패턴에 포함된 경로그래프들에 의해 정규화 될 수 있기 때문에, 이를 토대로 중복판단 없이도 중복생성회피를 할 수 있으므로 더욱 효율적이다. 순환그래프 역시도 그 패턴에 포함된 자유트리그래프로서 정규화 시킬 수 있다. 순환그래프들은 최소신장 스패닝 트리를 기준으로 정규형을 이룬다. 또한 최소신장 스패닝 트리는 레이블의 우선순위에 따른 깊이우선탐색의 정규형을 따른다. 어떤 순환그래프의 깊이우선순위에 따르는 정규형 최소신장 스패닝 트리는 하나만이 존재한다. 그러므로 순환그래프내에 정규형 최소신장 스패닝 트리가 변하지 않도록 확장을 한다면 중복생성을 억제할 수 있다. 하지만 순환그래프를 중복판단 없이 중복생성 회피할 수 있는 방법은 아직 없으며 [10] 패턴을 확장할 때 마다 반복적으로 스패닝 트리 나열 연산을 하게 되는데 이는 매우 많은 시간을 소요한다.

2. 확장된 패턴을 활용한 불필요한 확장회피 기법

본 절에서는 순환그래프영역에서의 중복패턴으로의 확장을 좀 더 회피할 수 있는 방법에 대하여 설명하며, 스패닝 트리나열 연산을 좀 더 효과적으로 운영할 수 있는 기법을 설명한다. 자유트리는 하나의 최소신장 스패닝 트리로서 여겨질 수 있다. gaston 알고리즘은 자유트리에서 순환간선이 확장되면 더 이상 노드확장을 하지 않고 순환간선의 확장만을 한다. 왜냐하면 노드의 확장이 먼저 일어난 자유트리에서 동일한 순환간선의 확장으로 생겨지는 그래프와 중복되기 때문이다. 순환간선의 확장으로 순환그래프가 생겨나면 그 순환그래프의 모든 스패닝 트리를 나열하여 처음 순환간선이 확장되었던 자유트리보다 더 작은 우선순위의 스패닝 트리가 발견되면 이 패턴은 중복된 패턴으로 여겨지므로 이를 폐기한다.

순환간선의 확장은 기존 그래프의 최소신장 스패닝 트리보다 높은 우선순위의 최소신장 스패닝 트리를 만들어 낼 수는 있지만, 기존의 최소신장 스패닝 트리를 제거할 수는 없다. 이는 중복된 작업을 회피할 수 있는 원리가 된다. 순환그래프 G의 확장가능한 모든 순환간선들을 c0... ci,.. cj,.. cn (0<i<j<n 인 임의의 i, j)라고 가정하자. G에서 cj가 확장되어진 순환그래프에서 더 높은 우선순위의 최소신장 스패닝 트리가 만들어 졌다면, 그 순환그래프는 중복생성된 것이므로 알고리즘은 그 순환그래프를 폐기하고 G로 되돌아가 다른 순환간선을 확장한다. 한편에 다른 순환간선 ci가 확장되어 생겨난 순환그래프 G′에서 더 높은 우선순위의 최소신장 스패닝 트리가 발견되지 않았다면, 이 패턴은 처음으로 발견된 것으로 간주하고, 이를 찾아진 패턴에 포함시킨 뒤 추가적인 확장을 하게 된다. 이때 모든 패턴을 확장해야 하므로 다른 모든 순환간선과의 조합을 고려해 보게 되는데, 중복된 패턴의 생성을 최소화하기 위하여 우선순위 하향식으로 모든 조합을 만들어 본다. 즉 ci에서 cn 까지의 모든 순환간선을 추가적으로 확장해 본다. 그런데 순환간선 cj가 G에 추가되면서 더 높은 우선순위의 최소신장 스패닝 트리를 만들어 냈다면 G′에서도 추가되어도 동일한 최소신장 스패닝 트리를 만들어 내게 된다. 그렇기 때문에 cj가 최소신장 스패닝 트리를 만들어 낸 다는 것을 알았다면, 이후 어떠한 순환간선의 조합도 할 필요 없이 제거되면 된다.

그림1은 자유트리와 확장 가능한 순환간선 c1, c2, c3을 보여주고, 이 순환간선들로 조합할 수 있는 모든 순환그래프를 보여준다. 중복 없는 순환간선의 조합을 위한 우선순위 하향식 패턴확장에 의해 생겨나는 패턴의 조합들은 {{c1},{c1, c2}, {c1, c2, c3}, {c1, c3}, {c2}, {c2, c3}, {c3}}가 되며, 확장영역의 종속성은 그림2에서 순환그래프들을 둘러싸고 있는 선과 같이 나타난다. 이때 순환간서 c2가 확장되어 생성된 순환그래프는 기본이 되었던 자유트리보다 더 작은 최소신장 스패닝 트리를 만들어 낸다. 순환간선 c2는 순환간선 c1과 조합된 순환그래프에서도 역시 동일한 최소신장 스패닝 트리를 만들어 내는 것을 볼 수 있다. 하지만 일반적인 방법으로 확장탐색을 하게 되면 탐색은 b, c, d, e, f, g, h 순서대로 이루어진다. c,에서 c2가 최소신장 스패닝 트리를 생성해 낸다고 해도 f에서도 순환간선의 생성을 만들어 낸다고 보장할 수 없으므로, f에서는 다시 스패닝 트리 나열연산을 해야만 한다. f를 c보다 먼저 찾아지게 한다면, c를 확장하지 않을 수 있다. 순환간선의 우선순위는 유지한 상태에서 우선순위가 낮은 간선들 먼저 확장하게 된다면 이를 해결할 수 있다. 낮은 우선순위의 패턴을 먼저 확장한다면 패턴의 종속성은 그대로 유지되지만 탐색의 순서는 h, f, g, b, e, c, d 로 변하게 된다.

3. egaston-C 알고리즘

표2에서는 egaston-C의 알고리즘이다. 여기에서 경로그래프영역은 생략되어 있는데, 경로그래프는 경로그래프의 양 끝단에서 순환간선이 확장되는 것만 허용하고 나머지 부분에

서 확장되는 순환간선은 확장하지 않는다. 이는 다른 부분에서 확장되는 순환간선으로부터 생겨난 순환그래프는 더 작은 크기의 스패닝 트리를 반드시 만들어 내기 때문이다.

경로그래프의 양 끝단의 순환간선이 확장되어진 후 더 이상 순환간선을 확장하지도 않는데 그 이유는 동일하다. 어떤 부모패턴으로부터 우선순위가 낮은 순환간선들부터 확장탐색을 하면서(line 10, 31) 확장된 순환간선이 더 높은 우선순위를 가지는 스패닝트리를 만들어 낸다면 이를 부모패턴의 순환간선에 표시한다(line 22-25). 그리고 그 순환간선보다 높은 우선순위의 순환간선을 확장할 때에 표시가 있는 순환간선과의 조인연산은 완전히 제외한다.(line 29) 순환그래프 영역뿐이라면 완전히 그 순환간선을 완전히 제거하여도 되지만, 자유트리 영역의 순환그래프는 비록 더 높은 우선순위의 스패닝을 만들어 낸다고 해도 그 순환간선을 제거해서는 안 된다. 이유는 그 자유트리를 확장한 다른 자유트리에서는 더 높은 우선순위의 스패닝트리를 만들어내지 않을 수 도 있기 때문이다.

표 2. egaston-C의 알고리즘

```
FreeTree( path or tree T, leg l, L, f ){
 if ( l is not allow by Backbone Constraint ) return ∅;
  T' = generate_tree( T, l );
  set S = ∅, L' = ∅;
  L' = L' ∪ Restrict_Extension(l,f); // by Backbone
Constraint
  for( i=0; i<size of L; i++){
    if ( legs[i] is leg refinement )
     if( legs[i] ≦ l of priority ) L' = L' ∪ Join(l,legs[i],f);
  }
  for( i = from lower priority to higher priority ){
    if ( legs[i] is node refinement ){
              S = S ∪ FreeTree( T', legs[i], L', f );
    }
    if ( legs[i] is cyclic refinement )
              S = S ∪ CyclicGraph( T', legs[i], L', f );
    }
  return S;
}

CyclicGraph( path or tree or cyclicgraph G, leg l, L, f ){
 G' = generate_cyclicgraph( G, l );
  if ( Find_HigherPriority_SpanningTree(G') == true ){
    l.makelower = true;
    return ∅;
  }
  set S = ∅, L' = ∅;
  for( i=0; i<size of L; i++)
  if ( legs[i]==cyclic & legs[i]≦l of priority
     & legs[i].l.makelower != true;  )
   L' = L' ∪ Join(l,legs[i],f);
  for( i = from lower priority to higher priority ){
   S = S ∪ CyclicGraph( G', legs[i], L', f );
  }
  return S;
}
```

IV. 성능평가

우선순위 하향식 방법을 적용하여 egaston-C 알고리즘을 구현하였으며 그래프 패턴 확장 알고리즘 중 가장 대표적인 가스톤 알고리즘[10]과 성능을 평가하였다. 실험환경은 core2X64, 2GB memory, Opensolarise10에서 실험하였다. 실험에 쓰인 그래프데이터 들은 PTE, DTP로서, PTE는 340개의 그래프 트랜잭션으로 이루어져 있으며, DTP는 422개의 트랜잭션으로 구성되어 있다. 성능평가를 위한 데이터셋에 대한 자세한 정보는 [10]에서 볼 수 있다. 두 데이터에서 최소빈도수임계값에 따라 찾아지는 패턴의 수는 표3과 표4와 같다.

빈도수의 값이 작아질수록 찾아지는 패턴의 개수는 기하급수적으로 늘어난다. 빈도수가 낮아질수록 찾아지는 패턴에서 순환그래프의 비율이 증가는 것을 볼 수 있다.

표 3. 패턴의 개수 (PTE)

suppport / patterns	1.5%	1.8%	2.1%	2.4%	2.6%	2.9%
Path	2817	2234	1724	1381	1065	903
FreeTree	586148	284294	117654	47496	27809	19578
Cyclic Graph	132248	57951	17571	5743	3164	2277
Total	721213	344479	136949	54620	32038	22758

표 4. 패턴의 개수 (DTP)

suppport / patterns	3.6%	4.3%	5.0%	5.7%	6.4%
Path	4383	3474	2894	2289	1904
FreeTree	22821527	1470771	672979	260874	75046
Cyclic Graph	9037845	434760	229419	66090	14436
Total	31863755	1909005	905292	329253	91386

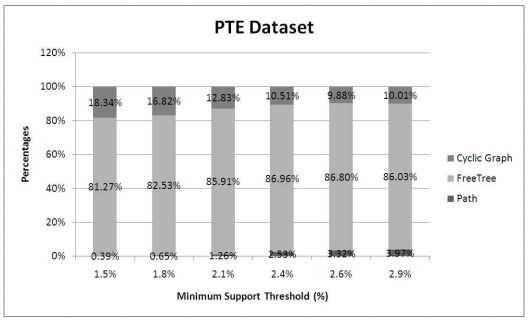


그림 2. 전체 패턴의 비율 (PTE)

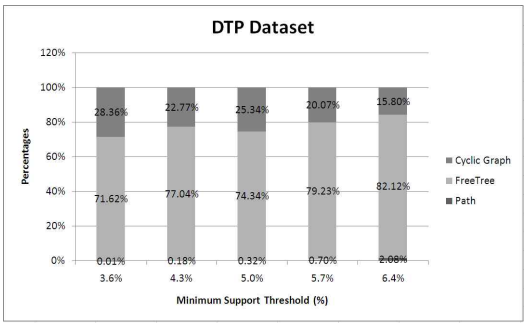


그림 3. 전체 패턴의 비율 (DTP)

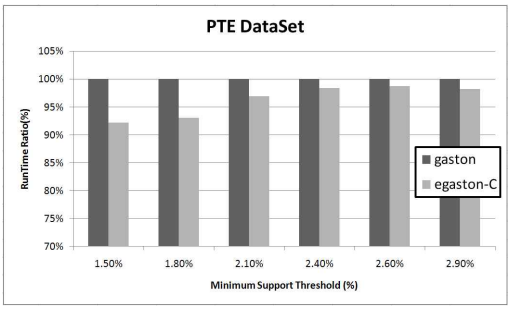


그림 4. egaston-C 수행시간 - PTE

그림2 와, 그림3 에서는 이를 잘 보여주며, DTP데이터 셋에서는 PTE 데이터 셋 보다 순환그래프의 비율이 훨씬 높은 것을 볼 수 있다. 그림 4에서 그림 7까지는 가스톤 알고리즘과 비교한 egaston-C의 수행시간의 향상을 보여준다. 데이터의 가로축은 최소 임계값 지지도이며, 세로축은 가스톤 알고리즘의 수행시간의 기준으로 한 egaston-C 알고리즘의 수행시간의 백분율 값이다. egaston-C 알고리즘은 대부분의 최소임계값에 대하여 더 빠른 수행시간을 보여주며, 특히 임계값이 작아질수록 향상된 수행시간을 보여준다. 또한 예상했던 대로 PTE데이터 보다 DTP데이터에서 더 많은 수행시간의 감소가 보인다. DTP데이터의 최소 지지도 임계값 3.6%에서는 수행시간이 20%까지 단축되는 것을 볼 수 있다. 이때

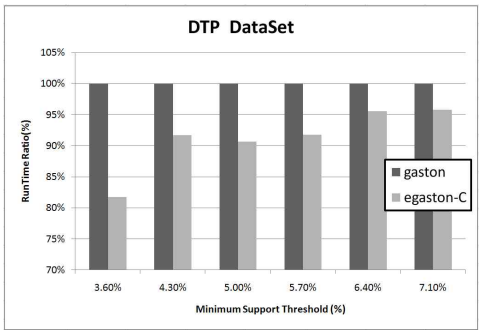


그림 5. egaston-C 수행시간 - DTP

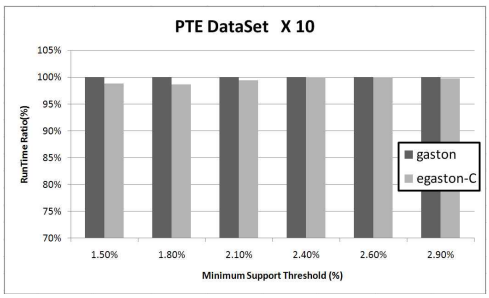


그림 6. egaston-C 수행시간 - PTE x 10

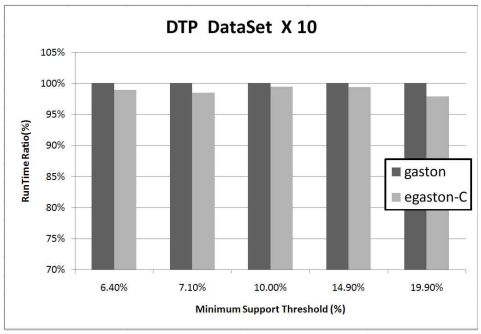


그림 7. egaston-C 수행시간 - DTP x 10

가스톤 알고리즘의 수행시간은 2000초가량 이었고, 제시된 방법에 의해 거의 400초가량의 수행시간이 단축되었다.

IV. 결론 및 추후연구

본 논문에서는 패턴확장기법의 운용을 세심히 분석하여 반복적으로 시행되는 불필요한 연산을 제거함으로서 수행시간을 단축을 이루어 냈다. 이는 그래프 마이닝에서 그래프 동형판단이 차지하는 비율이 매우 크기 때문이며, 또한 유사

한 작업이 반복적으로 행해지기 때문이다. 본 논문에서 시도한 부분은 이러한 유사한 작업들의 부분을 제거하여서 성능을 향상시켰다. 패턴확장 기법에서 불필요한 순환간선의 상속이 가장 많이 일어나는 곳은 자유트리그래프영역과 순환그래프영역의 경계부분이다. 순환간선의 확장은 최소신장 스패닝 트리를 변화시키지 않지만, 노드확장간선은 최소신장트리를 변화시킬 수 있기 때문이다. 그러나 무시할 수 없을 만큼 많은 유사한 작업들이 존재하기 때문에 이에 대한 추후연구가 필요하다.

"이 논문은 2011년도 충북대학교 학술연구지원사업의 연구비 지원에 의하여 연구되었음

참고문헌

[1] R. Agrawal, T. Imilienski, and A. Swami. "Mining association rules between sets of items in large datasets." In Proceedings of SIGMOD, May, 1993.

[2] P. Dmitriev, and C. Lagoze. Mining Generalized Graph Patterns Based on User Examples. ICDM, Dec. 2006. pp. 857-862.

[3] S. Gunnemann and T. Seidl, Subgraph Mining on Directed and Weighted Graphs. PAKDD, June 2010, pp. 133-146.

[4] J. Han, H. Cheng, D. Xin, and X. Yan, Frequent pattern mining: current status and future directions, Data Mining and Knowledge Discovery, 2007, 15(1), pp. 55-86.

[5] A. Inokuchi, T. Washio, and H. Motoda, An apriori-based algorithm for mining frequent substructures from graph data, PAKDD, Sep. 2000, pp. 13-23.

[6] U. Kang, D. H. Chau, and C. Faloutsos, Mining large graphs: Algorithms, inference, and discoveries. ICDE, April 2011, pp. 243-254

[7] Y. Ke, J. Cheng, J. X. Yu, Top-k Correlative Graph Mining. SDM, May 2009, pp. 1038-1049.

[8] M. Kuramochi and G. Karypis, An Efficient Algorithm for Discovering Frequent Subgraphs, TKDE 16(9), pp. 1038-1051, 2004.

[9] M. McGlohon, and L. Akoglu, C. Faloutsos. Weighted graphs and disconnected components: Patterns and a generator, ACM SIGKDD, August 2008, pp. 524-532.

[10] S. Nijssen, and J. N. Kok, A quickstart in frequent structure mining can make a difference, ACM SIGKDD, August 2004, pp. 647-652.

[11] T. Ozaki, and T. Ohkawa, Mining Correlated Subgraphs in Graph Databases. PAKDD, May 2008, pp.272-283.

[12] S. Ranu, and A. K. Singh, GraphSig: A Scalable Approach to Mining Significant Subgraphs in Large Graph Databases. ICDE, March 2009, pp.844-855.

[13] Y. Xie, and P. S. Yu, Max-Clique: A Top-Down Graph-Based Approach to Frequent Pattern Mining. ICDM, Dec. 2010, pp.1139-1144.

[14] X. Yan, and J. Han, gSpan: Graph-based substructure pattern mining, ICDM, Dec. 2002, pp. 721-724.

[15] X. Yan, and J. Han, CloseGraph: Mining Closed Frequent Graph Patterns, ACM SIGKDD, August 2003, pp. 286-295.

[16] F. Zhu, X. Yan, J. Han, and P. S. Yu, gPrune: A Constraint Pushing Framework for Graph Pattern Mining, May PAKDD 2007, pp. 388-400.

[17] Z. Zou, J. Li, H. Gao, and S. Zhang, Mining Frequent Subgraph Patterns from Uncertain Graph Data. TKDE, 2010, 22(9), pp. 1203-1218.

저 자 소 개

노 영 상

2007 : 충북대학교 컴퓨터공학 학사.
2009 : 충북대학교, 컴퓨터공학 석사.
관심분야 : 데이터마이닝, 데이터베이스
Email : ysnoh@chungbuk.ac.kr

윤 은 일

1997 : 고려대학교 이학석사.
1997~2006 : 한국통신 멀티미디어 연구소 전임/선임연구원.
2005 : Texas A&M Univ. 공학박사
2005~2006 : Texas A&M Univ. 포스닥연구원.
2006~2007 : 한국전자통신연구원, 선임연구원.
2007~현재 : 충북대학교 전자정보대학 컴퓨터전공 조교수
관심분야 : 데이터마이닝, 정보검색, 데이터베이스
Email: yunei@chungbuk.ac.kr

편 광 범

2010 : 충북대학교 컴퓨터공학전공 학사.
2010 - 현재 : 충북대학교, 컴퓨터공학 석사과정.
관심분야 : 데이터마이닝, 정보검색, 데이터베이스
Email: pyungb@chungbuk.ac.kr

양 흥 모

2011 : 충북대학교 컴퓨터공학전공 학사.
2011~현재 : 충북대학교, 컴퓨터공학 석사과정.
관심분야 : 데이터마이닝, 정보검색, 데이터베이스
Email: riang@chungbuk.ac.kr

이 강 인

2005~현재 : 충북대학교, 컴퓨터공학 학사과정.
관심분야 : 데이터마이닝, 정보검색, 데이터베이스
Email: abcnarak@chungbuk.ac.kr

류 근 호

1976 : 숭실대학교 공학사.
1980 : 연세대학교 공학석사.
1980~1983 : 한국전자통신연구원 연구원.
1983~1986 : 한국방송통신대학교 조교수.
1988 : 연세대학교 공학박사
1986~현재 : 충북대학교 전자정보대학 컴퓨터전공 교수
관심분야 : 데이터베이스, 데이터마이닝, 바이오인포매틱스
Email: khryu@dblab.chungbuk.ac.kr

이 경 민

1998 : Lund University (Sweden) 식품생명공학 석사
1995~2000 : (주)농심 기술개발 연구소 연구원
2004 : Kansas State Uni. 농학박사
2005~2006 : Texas A&M Uni. 포스닥 연구원
2006~현재 : Texas A&M Uni. 전임/선임 연구원
관심분야 : 식품안전성, 통계응용, 식품화학, 식품가공개발, 식품공학, 기기분석
Email:kml@otsc.tamu.edu