불법/이상 행위 탐지를 위한 그래프 기반 가시화 툴 개발

문승현*, 전효림*, 서인*, 한욱신**
*포항공과대학교 창의 IT 융합공학과
**포항공과대학교 창의 IT 융합공학과/컴퓨터공학과
e-mail: shmoon@dblab.postech.ac.kr, hrjeon@dblab.postech.ac.kr,
iseo@dblab.postech.ac.kr, wshan@postech.ac.kr

Development of a Graph-based Visualization Tool for Fraud Detection

Seunghyun Moon*, Hyo-Rim Jeon*, In Seo*, Wook-Shin Han**
*Dept. of Creative IT Engineering, POSTECH
**Dept. of Creative IT Engineering/Dept. of Computer Science and Engineering, POSTECH

요 약

본 논문에서는 최근 금융, 보험 등에서 빈번하게 발생하는 불법/이상 행위를 탐지하기 위해 데이터 그래프에서 사용자가 찾고자 하는 이상 패턴을 찾아 결과를 보여주는 그래프 가시화 툴을 제안한다. 개발한 툴은 정점과 간선 추가 및 삭제 등의 유용한 기능을 제공하기 때문에, 동적 그래프에 대한 불법/이상 행위 탐지를 위한 응용 프로그램에서도 널리 사용될 수 있을 것이다.

1. 서론

최근 금융, 보험 등에서 불법/이상 행위가 빈번해지고 피해 규모가 꾸준히 증가하면서 이를 미연에 탐지하고 예방하는 기술에 대한 관심이 증가하고 있다. 미국, 유럽 은행 고객의 체납액 중 10~20%는 신용카드를 이용한 금융 불법/이상 행위로 인한 손실이며[1], 국내에서도 이러한 불법/이상 행위가 꾸준하게 증가하여 2012 년 이후 4 년간 3 개의 보험사에서 발각된 불법/이상 행위의 피해 규모가 1 조 2 천억원에 이른다[2]. 그래프 데이터 구조를 이용하면 고객들의 개인 정보, 사용 패턴 등을 포함하는 빅 데이터를 기반으로 연관관계를 분석하고, 이를 토대로 불법/이상 행위를 탐지할 수 있다. 예를 들어, A 의 전화번호와 B 의 신용카드 정보, 그리고 A 의 신용카드 정보와 B 의 전화번호를 이용하여 두 개의 금융 계정을 생성하면 이들이 연결된 링 패턴을 구성하게 된다. 따라서 불법/이상 행위 탐지는 특정 패턴을 찾는 서브그래프 매칭 문제와 연관된다.

정적 그래프에 대한 서브그래프 매칭 문제는 데이터 크기가 매우 큰 빅 그래프 데이터에 대해서도 효율적으로 풀어낼 만큼 많은 연구가 이루어져 있다. 그러나 실시간 업데이트가 요구되는 금융, 보험 데이터에 대해 이상 행위 탐지를 위해서는 동적 그래프에 대한 서브그래프 매칭 문제를 빠르고 정확하게 풀어내는 것이 중요하다.

또한, 불법/이상 행위 탐지 기법을 실제로 적용하기 위해서는 금융, 보험 등의 분야에 종사하는 사용자들에게 쉽고 직관적인 인터페이스를 제공해야 한다. 따라서 탐지하고자 하는 패턴의 입력과 매치되는 결과의 출력을 효과적으로 제공하는 가시화 툴의 개발이 필요하다.

본 논문에서는 이상 행위를 탐지하기 위해서 주어진 동적 그래프와 이에 대한 서브그래프 매칭의 결과를 가시화하여 보여주는 가시화 툴을 제안한다. 본 툴은 스트림 형태로 업데이트 내용을 전송하는 데이터 소스와 이를 효율적으로 적용하는 서버로 이루어진 프레임워크 상에서 동작한다. 툴에서는 서브그래프 형태로 질의를 입력하면 질의와 매칭되는 이상 행위 후보군을 가시화하여 확인할 수 있다.

본 연구의 공헌은 다음과 같다. 1) 동적 그래프의 처리를 지원하는 프레임워크와 연계하여 이상 행위 후보군을 효과적으로 가시화하며 2) 사용자에게 이상 행위 탐지 수행 및 결과와 관련된 직관적인 인터페이스를 제공하는 그래프 가시화 모듈의 개발이다.

본 논문의 구조는 다음과 같다. 2 절에서는 관련 연구인 동적 그래프 처리와 그래프 가시화에 대해 소개한다. 3 절에서는 가시화 툴을 포함한 프레임워크의 전체 구조와 수행 방식에 대해 소개하며, 4 절에서는 전체 프로세스에 필요한 세 가지 알고리즘인 질의, 재질의 및 데이터 수정에 대해 소개한다. 5 절에서는 개발된 툴을 활용하여 실제 이상 행위 탐지에 이용하는 몇 가지 패턴에 대한 실험과 그 결과를 보이고, 6 절에서 본 논문을 맺는다.

2. 관련 연구

동적 그래프 처리

INCISOMAT[3]은 업데이트 요청에 대해 업데이트 전의 서브그래프 매칭 결과 집합과 업데이트 후의 서

"본 연구는 미래창조과학부 및 정보통신기술진흥센터의 ICT 명품인재양성사업의 연구결과로 수행되었음" (IITP-R0346-16-1007)

브그래프 매칭 결과 집합을 구하고 이 두 집합의 차집한 연산을 통하여 어떤 후보군이 새로 추가되었는지 혹은 삭제 되었는지를 밝혀낸다. SJ-Tree 라 불리는 다른 방법[4]은 리프 노드가 간선 하나에, 내부 노드는 두 자식 노드의 조인 연산의 결과인 서브그래프에 대응되는 left-deep tree 을 이용한다. Tree 의 좌하단에서부터 하나씩 서브그래프 매칭을 수행하여 중간 결과를 저장하고 찾고자 하는 서브그래프가 저장된 루트 노드에 도달하면 대응하는 결과를 반환한다[4]. 본 연구에서는 동적 그래프를 가정하여 그래프 업데이트가 발생하는 환경에서 그래프 매칭 처리가 가능하도록 본 연구실에서 개발 중인 프레임워크를 활용한다.

그래프 가시화

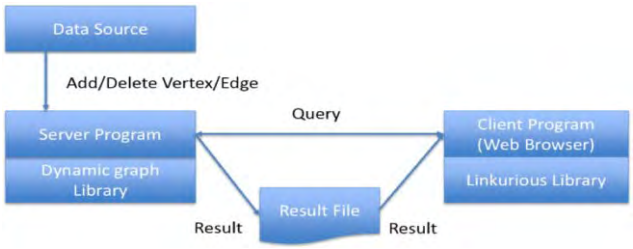
[5]에서 제안하는 기법은 주어진 사용자가 관심있는 focal point 를 중심으로 하여 그래프의 collapse, expansion 을 지원한다. 사용자에 의해 주어지는 매개변수 값에 대응하는 정점들의 집합이 focal point 에 해당하며, 이에 인접한 간선 중에서 레이블이 같은 것들을 1 개로 모아 표현한다. 그리고 각 간선에 연결된 정점을 루트로 하는 sub-tree 를 1 개로 모아 표현한다. OPAvion 은 큰 스케일의 그래프를 축소하여 가시화하는 시스템이다. APOLO 라 불리는 가시화 모듈은 기계학습 기법을 사용하여 사용자가 관심 있을 것으로 판단되는 정점을 추천해주고, 선정된 정점을 기준으로 그래프를 보여준다[6]. ForceAtlas2 은 물리 법칙을 이용하여 그래프를 배치하는 알고리즘이다. 정점 사이에는 척력이, 간선은 연결된 정점에 대해 인력이 작용하여 각 정점들이 힘의 균형을 이루어 배치된다[7]. 본 논문에서 사용한 linkurious 라이브러리 내의 알고리즘 역시 ForceAtlas2 을 기반으로 하는 레이아웃인 ForceLink 을 사용한다.

3. 가시화 프레임워크 구조

본 절에서는 프레임워크의 전체 구조와 각 파트의 기능에 대해 설명한다. 그림 1 은 가시화 프레임워크의 전체적인 구조를 보여준다. 프레임워크는 질의 그래프를 입력하는 클라이언트인 가시화 툴과 질의를 받아 데이터 그래프에 대해 서브그래프 매칭을 수행하는 서버 프로그램으로 이루어져있다. 또한, 동적 그래프에서의 처리를 지원하므로 그래프가 저장되어있는 서버에 정점과 간선의 추가, 삭제 혹은 변경 정보를 연속적으로 전달하는 데이터 소스가 존재한다.

사용자는 웹 브라우저 기반의 가시화 툴을 통해 본 프레임워크에 접근할 수 있다. 가시화 툴은 likurious 가시화 라이브러리를 이용하여 구현되어 있으며, 사용자가 그래프 형태로 질의를 입력하면 서버에서 수행한 서브그래프 매칭 결과를 그래프 형태로 확인할 수 있다.

서버는 서브그래프 검색 및 데이터 그래프 업데이트를 담당하는 동적 그래프 처리 라이브러리와 node.js 로 작성된 서버 프로그램으로 이루어진다.



(그림 1) 서브그래프 가시화 프레임워크의 구조

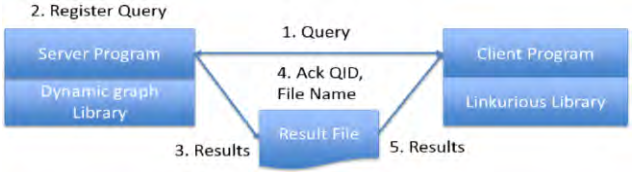
서버 프로그램은 여러 사용자의 그래프 수정 및 행위 패턴 검색을 지원한다. 여기에서 그래프 수정 연산은 정점과 간선에 대해 추가, 삭제, 수정으로 총 6 가지를 지원한다. 서버는 초기 그래프 데이터 및 이상 행위 패턴 검색 결과를 파일 형식으로 클라이언트에 전송한다. 여러 클라이언트는 전송된 파일의 이름을 식별자로 하여 자신이 입력한 질의 결과를 확인할 수 있다. 파일 형식으로 결과를 관리하는 것은 클라이언트가 전체 그래프가 아니라 업데이트된 그래프 정보만을 가져와서 효율적으로 가시화하기 위함이다.

4. 가시화 알고리즘

본 절에서는 가시화 알고리즘인 질의, 재질의 및 데이터 수정 알고리즘에 대해 설명한다.

질의

클라이언트는 질의 그래프의 클론인 queryGraphCopy 을 만들어 각각의 정점에 대하여 coordinate 와 size data 을 삭제, 각각의 간선에 대하여 size data 을 삭제하고 클라이언트 식별자와 queryGraphCopy 을 서버 측에 전송한다(Query). 클라이언트가 데이터 그래프에 업데이트를 반영하여 볼 수 있도록 서버는 수정된 정점과 간선에 대한 리스트를 전송한다. 이는 서버 측 그래프 데이터에서는 존재하는 수행 결과가 클라이언트 측 그래프 데이터에 존재하지 않을 경우 발생하는 에러를 처리하기 위함이다. 그 후 정점 및 간선 식별자 정보와 레이블 정보를 포함하는 queryInputString 이라는 문자열 정보를 가지는 텍스트 형식 파일을 만들어 클라이언트 식별자 이름으로 등록한다(Register Query). 동적 그래프 처리 라이브러리에 의해 질의에 대한 결과 파일이 생기면 이를 만들어둔 텍스트파일에 쓴다(Result). 결과 파일이 완성되면 서버는 클라이언트에 success 메시지를 전송하고(Ack QID, File Name), 클라이언트가 이를 받으면 결과 파일을 확인할 수 있다(Result). 아래 의사 코드 및 그림은 질의 단계의 전체적인 과정을 나타낸다.



(그림 2) 질의 과정

Algorithm 1 Query on client

```
Make a clone of the query graph, queryGraphCopy
for each vertex of the query graph do
    queryGraphCopy.Vertex.Delete(coordinate, size)
end for
for each edge of the query graph do
    queryGraphCopy.Edge.Delete(size)
end for
Send register query, Request(clientID, queryGraphCopy) to server end
queryGraphCopy and clientID data to server as a query request
if success then
    Download and print the result file
end if
```

(그림 3) 질의 과정의 클라이언트 알고리즘

Algorithm 2 Query on server

```
Send modified vertex/edge list to all clients
for each vertex/edge of the query graph do
    ChangeLabelType(stringLabel, intLabel)
end for
Add a string queryInputString involving (IDList, intLabelList)
Write a txt file clientID for queryInputString
Read a out file written by subgraph search library
for each subgraph case of out file do
    for each node of the subgraph case do
        ChangeLabelType(intLabel, stringLabel)
    end for
end for
Update the txt search file clientID
if IsFinished(clientID) then
    send success to client
end if
```

(그림 4) 질의 과정의 서버 알고리즘

재질의

이전에 요청했던 서브그래프 매칭 질의에 대한 결과를 이후 데이터 그래프 업데이트에 반영하도록 하기 위해 재질의 단계가 필요하다. 질의 단계 이후 클라이언트가 본인의 식별자를 포함하여 요청을 보내면 (ReQuery QID), 서버에서는 결과 파일이 존재하는지 확인하고 클라이언트에게 success 메시지를 전송한다 (Ack QID, File Name). 클라이언트는 이를 받고 결과 파일을 확인할 수 있다(Result). 아래 의사 코드 및 그림은 질의 단계의 전체적인 과정을 나타낸다.

(그림 5) 재질의 과정

Algorithm 3 Requery on client

```
Send refresh, Request(clientID) to server end clientID data to server as a
refresh request
if success then
    Download and print the result file
end if
```

(그림 6) 재질의 과정의 클라이언트 알고리즘

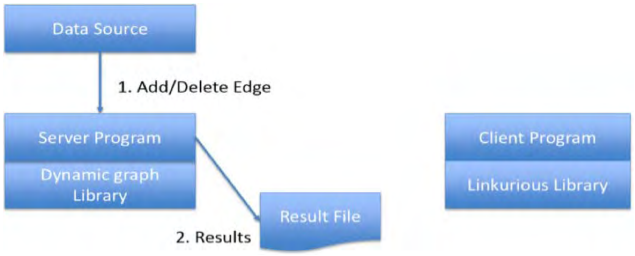
Algorithm 4 Requery on server

```
if IsFinished(clientID) then
    send success to client
end if
```

(그림 7) 재질의 과정의 서버 알고리즘

데이터 수정

클라이언트는 데이터 수정을 요청하기 위하여 다음과 같은 절차를 가진다. 의도한 데이터 수정이 정점 수정 요청이면 해당 데이터에 대해 정수형의 정점 식별자를 추가한다. 간선 수정 요청의 경우 해당 데이터에 대해 정수형의 정점 식별자를 콤마로 분리하여 2 개의 정수형 정점 식별자를 추가한다. 이는 각각 간선의 출발 및 도착 정점을 의미한다. 이후 서버로 데이터 수정 요청을 보낸다 (Add/Delete Edge). 서버는 이를 받아 요청 받은 레이블이 정의되어 있는 것인지 확인하고 새로운 레이블인 경우 추가한다. 이후 그래프 수정 라이브러리에 데이터 수정을 요청하는데, 이때 다른 클라이언트로부터 새로운 질의 그래프를 전달받아 질의가 새로이 등록되면 동적 그래프 처리 라이브러리는 수정한 정점 및 간선에 대하여 다시 서브그래프를 찾고 이에 대한 결과 파일을 업데이트한다. 마지막으로 클라이언트의 데이터 수정 요청이 정점 및 간선의 연산 타입에 따라 새로 얻어진 결과 리스트를 바탕으로 기존 결과를 업데이트한다(Results). 데이터 업데이트 연산은 삭제 후 추가 연산으로 생각할 수 있으므로 이를 반영하여 의사 코드를 작성하였다. 아래 의사 코드 및 그림은 데이터 수정 단계의 전체적인 과정을 나타낸다.



(그림 8) 데이터 수정 과정

Algorithm 5 Data modification on client

```
Send refresh, Request(clientID) to server
if IsModifyingVertex() then
    AddVertexID(data, intVID)
else
    SeparateAndAddVertexID(data, intVID)
end if
if IsModificationChanged() then
    AddStringLabel(data, firstLabel)
    AddStringLabel(data, secondLabel)
else
    AddStringLabel(datal, label)
end if
Send modification, Request(data) to server end data to server as a modifi-
cation request
```

(그림 9) 데이터 수정 과정의 클라이언트 알고리즘

Algorithm 6 Data modification on server

```
for each label of the Request(data) do
    if IsDefined(label) then
        AddLabel(label, mappingTable)
    end if
    ChangeLabelType(stringLabel, intLabel)
end for
Send modification, Request(data) to graph modification library
if IsSubgraphQueryRegisterd() then
    Re-evaluate subgraph only for those vertices/edges
    Read re-evaluated out file
    for each subgraph case of out file do
        for each node of the subgraph case do
            ChangeLabelType(intLabel, stringLabel)
        end for
    end for
    Write re-evaluated out file
    if IsAdding(data) then
        Add(clientID, updatedSearchList)
    else if IsRemoving(data) then
        Delete(clinetID, updatedSearchList)
    else
        Delete(clinetID, deleteUpdatedSearchList)
        Add(clientID, addUpdatedSearchList)
    end if
end if
```

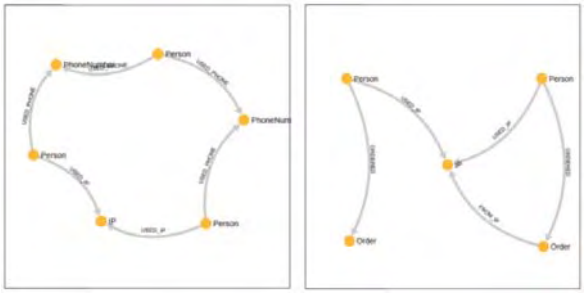
(그림 10) 데이터 수정 과정의 서버 알고리즘

5. 실험

본 절에서는 개발된 가시화 툴을 이용하여 두 가지 대표적인 이상 행위 패턴에 대해 실험하고 그 결과를 분석한다.

실험 환경

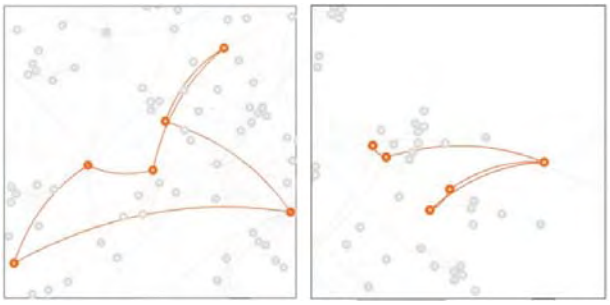
본 실험은 3.4GHz 클럭 사이클을 가지는 Intel i7-4770 CPU core, 16GB 메모리 스펙의 클라이언트 컴퓨터에서 수행하였다. 데이터 셋으로 사용한 그래프는 인터넷 상거래 트랜잭션 데이터인 retail fraud dataset을 이용하였다. 해당 데이터는 실제로 0.8M 개의 정점과 1.6M 개의 간선을 가지고 있으나, 가시화 결과를 쉽게 확인하기 위해서 데모에는 이 중 310 개의 정점과 665 개의 간선을 추출하여 사용하였다. 실험에서는 입력 질의 그래프로 대표적인 이상 행위 패턴인 링과 리본 패턴을 찾고, 정점/간선의 추가 및 삭제 시 그에 따른 패턴 수의 변화를 확인함으로서 본 툴이 데이터 수정 연산을 효과적으로 지원함을 보인다. 실험에 사용한 입력 질의 그래프는 아래와 같다.



(그림 11) 2 가지 패턴의 입력 질의 그래프

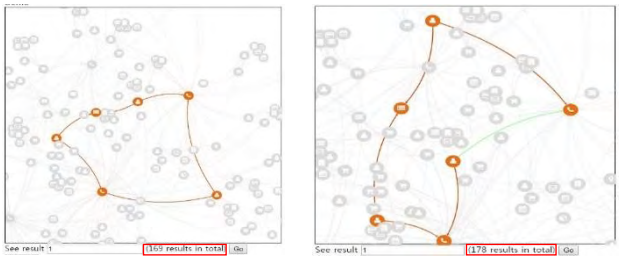
실험 결과

사용자가 찾고자 하는 질의 그래프를 전달하면 서버는 이를 받아 같은 형태의 그래프를 찾아 파일로 저장하고 클라이언트는 이를 가져와서 가시화하여 사용자에게 보여준다. 아래의 그림을 통해 질의 그래프와 같은 형태를 가지는 서브그래프를 찾아 가시화 해주는 것을 확인할 수 있다.

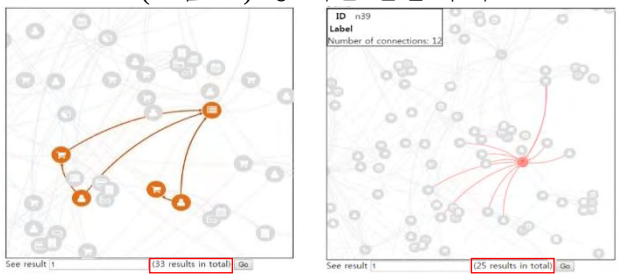


(그림 12) 입력 질의에 대한 결과

다음으로 첫 번째 링 패턴에 대한 매칭 결과는 초기 169 개였으나, 사람에서 휴대전화로의 간선을 하나 추가하였을 때 178 개로 증가하였다. 두 번째 리본 패턴에 대한 매칭 결과는 초기 33 개였으나, IP 정점을 하나 삭제하였을 때 25 개로 감소하였다. 이를 통해 본 툴이 데이터 수정에 대해 질의 결과를 효과적으로 업데이트 하여 제공함을 확인할 수 있다.



(그림 13) 링 패턴 간선 추가



(그림 14) 리본 패턴 정점 삭제

6. 결론

본 논문에서는 최근 금융, 보험 등에서 빈번하게 발생하고 있는 불법/이상 행위를 효과적으로 탐지하기 위해 동적 그래프에 대해 서브그래프 매칭을 수행한 이상 행위 후보군을 가시화하여 사용자에게 제공하는 그래프 기반 툴을 제안하였다. 본 툴은 상기 언급한 프레임워크와 연계하여 실시간으로 업데이트가 이루어지는 동적 그래프에 대해서도 효과적으로 이상 행위 후보군을 가시화 할 수 있으며, 동적 그래프를 고려한 다양한 기능을 제공하여 높은 활용도가 기대된다.

참고문헌

[1] http://www.experian.com/assets/decision-analytics/white-papers/first-partyfraud-wp.pdf.

[2] http://www.hani.co.kr/arti/economy/economy_general/660660/html

[3] W. Fan, J. Li, J. Luo, Z. Tan, X. Wang, and Y. Yu. "Incremental graph pattern matching," in *Proceedings of the ACM SIGMOD International Conference on Management of data*, 2011, pp. 925-936

[4] C. Wang and L. Chen. "Continuous subgraph pattern search over graph streams," in *Proceedings of the IEEE 25th International Conference on Data Engineering*, 2009, pp. 393-404.

[5] S. Sundara et al. "Visualizing Large-scale RDF data using subsets, summaries, and sampling in Oracle," in *Proceedings of the IEEE 26th International Conference on Data Engineering*, 2010, pp. 1048-1059.

[6] L. Akoglu et al. "OPAvion: Mining and visualization in large graphs," in *Proceedings of the ACM SIGMOD International Conference on Management of data*, 2012, pp. 717-720.

[7] Mathieu Jacomy, Tommaso Venturini, Sebastien Heymann, and Mathieu Bastian. "ForceAtlas2: A continuous graph layout algorithm for handy network visualization designed for the Gephi software," *PloS one*, June 2014, vol. 9, issue 6, e98679, pp. 1-12.