알고리즘 HW2 report

Graph Pattern Matching Challenge

2016-15579 신동섭 (기계항공공학부), 2016-11018 정호준 (기계항공공학부)

주어진 그래프에서 동일한 subgraph(vertex 와 edge 를 isomorphic 하게 대응시킬 수 있는 그래프)를 찾는 알고리즘(DAF 알고리즘)의 일부(BuildDAG, Backtrack)를 구현하고 추가적으로 성능(10 만개의 matching 을 찾는데 걸리는 시간)을 개선하기 위해 기본적인 Pruning 을 구현하였다. 사용한 언어는 c++ 이며, 참조에 적혀있는 논문을 참고하여 작성하였다.(박근수 교수님 연구실 논문)

Build DAG

***Algorithm* 1 : *BuildDAG*(*data*, *q*, *cs*)**

1. $r \leftarrow root\ with\ small\ candidate\ size\ \&\ large\ degree$
2. $build\ a\ DAG\ by\ the\ order\ of\ BFS(r)$

주어진 query graph 와 candidate set 을 matching 하기 위해, query graph 를 DAG 로 변환하여 DAG 와 candidate set 을 matching 하는 방식을 이용하였다. DAG 의 root 는 candidate size 가 작고, edge 의 개수(degree) 가 큰 vertex 로 설정하였다.

Backtracking

***Algorithm* 2 : *BackTrack*(*q*, *cs*, *M*, *extendable*)**

1. **if** $M.size = q.size$
2. **print** M
3. **else if** $M.size = \mathbf{0}$
4. **for each** v **in** $C(r)$
5. $\boldsymbol{M} \leftarrow \{(\boldsymbol{r}, \boldsymbol{v})\}$
6. **if** $childNode\ of\ r\ is\ extendable$ **then** $push\ in\ to\ extendable$
7. $mark\ v\ as\ visited$
8. $BackTrack(q, cs, M, extendable)$
9. $exclude\ v\ from\ M; mark\ v\ as\ unvisited$
10. **else**
11. $u \leftarrow extendable.pop$
12. **for each** $v\ in\ C(u)\ which\ is\ extendable$
13. **if** $v\ is\ unvisited$
14. $M' \leftarrow M \cup \{(u, v)\}; Mark\ v\ as\ visited;$
15. $BackTrack(q, cs, M', extendable); mark\ v\ as\ unvisited;$

extendable 이라는 priority queue 를 구현하여, extendable candidate size 순으로 extendable vertex 들을 계속 Update 해가며 저장해준다. ***line6*** 의 extendable 은 query graph 에서의 extendable vertex 를 말하는 것이고 ***line 12*** 의 extendable 은 candidate set 에서의 extendable candidate 를 말하는 것이다.

올바른 Matching 을 찾을 경우, 바로 print 하도록 구현하였다. ***line 2*** 의 print M 은 challenge 에서 지정한 형식에 맞게 M 을 콘솔에 출력하는 행위를 말한다.

Matching Order

query vertex 의 matching order 는 현재까지 부모 vertex 가 모두 matching 된 extendable 한 vertex 중, candidate 의 parent 가 이미 Matching 되어 있어서, extendable 한 candidate size 가 작은 vertex 들 부터 matching 하도록 하였다. extendable vertex 들을 extendable candidate size 의 크기로 정렬한 priority queue 를 유지하여 필요한 시점에 vertex 들을 pop 해서 사용하는 방식으로 구현하였다.

Basic Pruning

실제 구현에서 Matching 시, query vertex 의 candidate set 에 있는 vertex 들이 이미 모두 Matching 되어 더이상 매치할 수 없을 경우, loop 를 종료하도록 구현하였다.

Environment 및 실행 방법

사용한 언어 : c++ 11
컴파일러 : g++
컴파일 옵션 : -O3 -Wall -std=c++11
운영체제 : 윈도우 10

실행방법

mkdir build
cd build
cmake ..
make
./main/program <data graph file> <query graph file> <candidate set file>

Reference

[1] Myoungji Han, Hyunjoon Kim, Geonmo Gu, Kunsoo Park, and WookShin Han. Efficient Subgraph Matching: Harmonizing Dynamic Programming, Adaptive Matching Order, and Failing Set Together. In Proceedings of the ACM International Conference on Management of Data (SIGMOD '19), pages 1429–1446, 2019.