Graph-Pattern-Matching-Challenge 보고서

2017-17392 고수민 2016-17096 조영훈

Implementation (Matching Order & Backtracking)

1. 주어진 Data, Query, Candidate Set으로 DAG 구성

⑴ (Candidate개수 / Degree)값이 가장 작은 Vertex를 DAG의 Root로 정한다.

⑵ Root를 vector q에 저장 후 다음 시행 ①,②를 반복한다.

① vector q에 있는 vertex 중 Degree가 가장 크고, Candidate가 가장 적은 vertex u를 선택하고 u를 vector q에서 삭제한다.

② u와 인접한 vertex v가 u의 parent가 아니라면 v를 u의 child로 설정하고, v가 vector q에 저장된 적이 없으면 v를 vector q에 저장한다.

2. Candidate Set에서 각 candidate의 weight 계산

Path-Size Order에서 정의한 바와 같이 각 candidate의 weight를 계산한다.

3. Candidate가 하나뿐인 vertex에 대해서 우선 Matching을 완료한다

4. Backtrack

⑴ matching 된 수가 query data의 vertex 수와 같은 경우

① matching 결과를 출력하고, 찾은 matching 결과가 100000 이상인 경우, 작동을 중지한다.

⑵ matching 된 수가 0인 경우

① DAG Root의 candidate 중, candidate size가 1인 child에 대한 extendable candidate을 선정한다.

② extendable candidate에 대해 다음 시행 ⓐ, ⓑ를 반복한다.

ⓐ root child 중, 현재 candidate에 대한 extendable vertex의 weight과 extendable candidate set

을 계산한다.

ⓑ extendable vertex에 대한 정보를 가지고 있는 priority queue에 ⓐ의 결과를 삽입한 후, 다음 Backtracking을 실시한다. 이 때, matching 된 수를 하나 증가시킨다.

⑶ 그 외의 경우

① extendable vertex에 대한 정보를 가지고 있는 priority queue에서 weight이 가장 작은 vertex를 선정한다. weight이 같은 값이 존재할 경우, candidate size가 작은 순서로 선택한다.

② extendable candidate에 대해 다음 시행 ⓐ, ⓑ, ⓒ를 반복한다.

ⓐ 현재 candidate에 대해, candidate의 visited 여부를 확인한다.

ⓑ unvisited candidate인 경우, 현재 vertex의 child 중 extendable vertex의 weight과 extendable candidate set을 계산한다.

ⓒ priority queue에 ⓑ의 결과를 삽입한 후, 다음 Backtracking을 실시한다. 이 때, matching 된 수를 하나 증가시킨다.

Environment

- 실행 환경:

Ubuntu / g++ 7.5.0

- 실행 방법(Readme에 명시된 방식과 일치)

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