Graph Pattern Matching Challenge

2019-15861 염준영, 황희담

June 6, 2021

1 알고리즘

1.1 Matching Order

기본적으로 [1], p25-27에 있는(report 폴더의 논문3.pdf) Weight array 방법을 사용하여 Vertex의 순서를 정하였다. 다만 매번 $\sum_{v \in C_M(u)} W_u(v)$를 계산하는 것보다, 그냥 $\sum_{v \in C(u)} W_u(v)$ 를 계산해 놓고 그 weight대로 매칭하는 것이 실제 성능이 더 잘 나와서 $\sum_{v \in C(u)} W_u(v)$를 사용하였다.

```
    if (allmatched)
    {
      int tempweight = 0;
//      std::vector<Vertex> C_M_u = extendable(acc, i, cs, data);

//      for (Vertex v: C_M_u)
//      {
//        tempweight += w[i][v];
//      }
      // C_M_u.size() 를 쓰면 Candidate Size order
      // tempweight = C_M_u.size();
```

```
    tempweight = weight[i];
  }
```

(해당 코드에서, for 루프 부분은 $\sum_{v \in C_M(u)} W_u(v)$, `weight[i]`는 $\sum_{v \in C(u)} W_u(v)$, `C_M_u.size()` 는 $|C_M(u)|$)
또한, u를 정한 후, $v \in C_M(u)$에 대하여 $M' \leftarrow M \cup \{(u, v)\}$를 추가한 후 매칭할 때, v를 선택하는 순서를 u의 이웃이 가진 라벨 중 가장 빈도가 높은 라벨을 l이라 할 때, v의 이웃이 가진 l의 수를 기준으로 하여 내림차순으로 매칭하게 하여, 더 가능성이 높은 구조를 찾아 더 빠르게 매칭되도록 하였다.(이부분 다양한 실험 중) ($|C_M(u)|$를 사용하면 Candidate Size order, $\sum_{v \in C_M(u)} W_u(v)$를 사용하면 Path size order)

1.2 Backtracking

2 실행 환경

References

[1] 한명지. “An Efficient Algorithm for Subgraph Isomorphism using Dynamic Programming on Directed Acyclic Graphs”. eng. In: (2018).