Graph Pattern Matching Challenge

2019-15861 염준영, 황희담

June 8, 2021

1 알고리즘

1.1 Matching Order

[1]에 있는 Weight array 방법(Path size order)을 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. `DAG::InitWeight` 함수에서 확인할 수 있다.
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. (`Backtrack::AdaptiveMatching` 함수에서 extendable을 sort하는 부분)

1.2 Backtracking

기본적으로 [1], "Algorithm 2"의 방법을 따른다, 다만 앞서 Matching order절에서 언급한 바와 같이, $v \in C_M(u)$를 순회하며 매칭할 때 $C_M(u)$의 순서를 정렬한 후 탐색한다.

2 실행 환경

Linux 및 macOS환경에서, readme에 언급된 바와 같이 cmake와 make를 이용하여 컴파일하였다.

References

[1] Myoungji Han et al. “Efficient Subgraph Matching: Harmonizing Dynamic Programming, Adaptive Matching Order, and Failing Set Together”. In: *Proceedings of the 2019 International Conference on Management of Data*. SIGMOD ’19. Amsterdam, Netherlands: Association for Computing Machinery, 2019, pp. 1429–1446. ISBN: 9781450356435. DOI: 10.1145/3299869.3319880. URL: https://doi.org/10.1145/3299869.3319880.