Algorithms: Optimizing Subgraph Isomorphism

2017-12690 **Muhwan Kim** 2017-17319 **Dongwoo Kim** 2017-18570 **Sungchan Yi**

Department of Computer Science and Engineering

June 19th, 2019

Contents

1 Introduction

In this homework, we optimize a subgraph matching algorithm introduced in the paper we were given. Given a data graph G and a query graph q, the DAF algorithm finds all matches of q in G from a directed acyclic graph (DAG) of the query graph q. There are many ways to choose a DAG from the query graph, and our job for this assignment is to find a rooted DAG of the query graph that will show better performance with the DAF algorithm within 1 minute.

The performance of the algorithm will be measured as follows.

- Average Total Time per Query
- Number of Recursive Calls
- Number of Solved Queries

2 Observation

Our first observation was the selection of the root for the DAG. Originally, the root r was chosen by

$$r = \underset{u \in V(q)}{\operatorname{argmin}} \frac{|C_{\text{ini}}(u)|}{\deg_q(u)} \tag{1}$$

This was because:

- The size of the initial candidate set $C_{\text{ini}}(u)$ should be minimized to reduce the search space for backtracking.
- $\deg_q(u)$ should be maximized since we want to choose a vertex with a higher degree to rule out more possible candidates for the root.

Our observation was that the ratio of these two factors would not guarantee the optimal choice for the root. Thus we tried choosing a root only from $|C_{\text{ini}}(u)|$ and saw a slight increase in performance. After many hours of trial and error, we couldn't get any better. Then we realized that $\deg_q(u)$ was a factor that could affect the performance. Even though $|C_{\text{ini}}(u)|$ is the same, the degree of a vertex could affect the performance when backtracking. So we added the $\deg_q(u)$ term again and tried other equations such as $\operatorname{argmin} \frac{|C_{\text{ini}}(u)|}{\log(1+\deg_q(u))}$, but didn't get any significant result.
Then we thought that reducing $\deg_q(u)$ would be better for backtracking, so we added a parameter k and used the equation

$$r = \underset{u \in V(q)}{\operatorname{argmin}} \left\{ |C_{\text{ini}}(u)| + k \cdot \deg_q(u) \right\} \tag{2}$$

The parameter k represents how much weight we want to put on the degree of the vertex.

3 Implementation

We only changed a single line.

```
// dag.cpp, Line 351
    int k = 5; // Also : 10, 15, 20, 25, 30
    rank = numInitCand + k * degree;
```

4 Experiments

4.1 Environment

The following is the specification for the test environment.

- OS: Ubuntu 18.04.2

- CPU: Intel(R) Xeon(R) W-2133 CPU @ 3.60GHz × 6
- g++: `7.4.0 -std=c++11`
- RAM: 32GB

Compile Options: `g++ -std=c++11 main.cpp dag.cpp -o program`

4.2 Results

The following are the results for the DAF algorithm.

4.2.1 Original Implementation

This is the result generated by choosing the root of the rooted DAG from equation (1).

Average Total Time	Recursive Calls	Solved Queries
75.3564 ms	30885.7683	82

Table 1. Results of original implementation

4.2.2 Our Implementation

This is the result generated by choosing the root of the rooted DAG from equation (2). We ran some experiments to find an optimal value of parameter k, to check if it existed. The last column shows the reduced average total time compared to the original implementation.

k	Average Total Time	Recursive Calls	Solved Queries	Reduced (%)
$k = 5$	71.4657 ms	26117.5882	85	5.1
$k = 10$	67.6965 ms	25183.1461	89	10.1
$k = 15$	67.7463 ms	24944.4111	90	10.1
$k = 20$	67.2010 ms	24836.4396	91	10.8
$k = 25$	**66.6055** ms	24847.1758	**91**	**11.6**
$k = 30$	67.3575 ms	24834.2967	91	10.6

Table 2. Results of our implementation for different values of k

4.3 Analysis

As k increased, we observed that the performance got better, and performance started to decrease after $k = 30$. Thus we found that setting $k = 25$ gave the best performance in terms of average total time and solved queries. We can infer that the degree of a vertex should have about 25 times more weight when choosing a root for the root DAG, to improve the performance. But the major flaw here is that this setting may only work

for this `human` dataset we were given to experiment with. Whether this could be a general method for choosing the root is unknown.

5 Conclusion

Through this assignment, we gained knowledge about subgraph isomorphism problem, and we tried to optimize the preprocessing of query graph q into a DAG. We mainly focused on choosing the best root for the rooted DAG, and improved the performance of the DAF algorithm by choosing the root by equation (2). We arrived at a conclusion that the degree of a vertex should be taken into account when choosing the root for the DAG, and setting 25 times more weight onto the degree gave the best performance among our experiments.