Graph Matching Challenge Report

2019-14144 통계학과 성우석

2019-11167 통계학과 이동현

1. How to choose a matching order

Step 1) 우선, share의 개념을 정의하자.

쿼리 그래프에서 점 i에 이웃인 점을 a_{i1}, …, a_{in_i}라 할 때, 점 i의 candidate set의 크기를 c_i라 할 때, 점 i의 candidate set과 점 a_{i1}, …, a_{in_i}의 candidate set 중 실제 이웃인 edge의 개수를 N이라 하자. 이때, share을 다음과 같이 정의한다.

$$\text{share} := \frac{N}{c_i \cdot \sum_{k=1}^{n_i} c_k}$$

즉, share의 개념은 그래프의 각 점이 candidate set에서 쿼리 상에서 neighbor인 candidate set과 연결 가능 정도에 비해 실제 연결된 비율을 나타낸 변수다. 그 후 share의 개념을 이용하여 matching order를 정해주며 그 기준은 그래프의 특성에 따라 결정된다.

Step 2) 그래프의 특성 결정 and 순서 결정

그래프의 특성을 결정하는 변수인 mean_share은 share 값의 평균 값으로 계산했으며 그 특성인 dense와 sparse의 차이는 테스트 케이스를 분석해보면서 기준을 정했고 그 값은 0.85다.

Dense한 그래프에 대해서는 share의 값이 작은 순서대로 탐색했다. 즉, sparse한 점부터 dense한 점 순서로 탐색을 진행한다. 반면, sparse한 그래프에 대해서는 share의 값이 큰 순서에서 작은 순서로 탐색을 진행한다. 이 이유는 특정 데이터에서 예외적인 사항들에 대한 탐색을 우선적으로 마치고 가능한 경우들을 탐색해보면 그 이후에 큰 부분에 대하여 경우의 수를 크게 줄일 수 있기 때문이다.

Step 3) Share가 1인 케이스

Share가 1인 점들에 대해서는 항상 이웃한 쿼리 그래프의 vertex의 cs내의 vertex와 share가 1인 vertex의 cs내의 vertex가 모두 연결되어 있으므로 다른 경우에 대해서만 탐색하고 탐색이 종료되었을 때, share이 1인 점들은 for loop를 통해 모든 경우에 대해 조합만 만들어주면 되므로 먼저 조합을 만들어주고 다른 경우에 대한 탐색을 진행했다.

2. How to performs backtracking

Step 1) 앞에 matching order를 정해주는 과정 전에 candidate_Size = 1인 점들은 answer에 먼저 넣고 전체적인 과정을 진행한다

Step 2) 점의 share가 1이 아니면 supply_queue를 만들어서 대입한다. 이때 share 값이 작은 순서대로 matching order로 잡아준 것은 share가 작은 순서대로 queue에 넣어주고 큰 순서대로 order를 잡아준 것은 share가 큰 순서대로 queue에 넣어준다.

Step 3) supply_queue가 empty state가 될 때까지 BFS를 진행하면서 탐색할 edge들을 선택하고 그 edge들에 대해서 DFS로 탐색을 진행한다.

Step 4) Matching되는 결과들에 대하여 출력을 진행한다. 이때 100000개가 넘는 경우 100000개째를 출력하고 코드를 정지한다.

3. The environment

Code::Blocks 16.01에서 해당 프로젝트를 진행하였습니다.

4. How to run program

1) 프로그램이 있는 디렉토리 파일로 이동

2) mkdir build -> cd build -> cmake .. -> make

3) ./main/program ../data/lcc_human.igraph ../query/lcc_human_n3.igraph ../candidate_set/lcc_human_n3.cs (human_n3에 대한 example)

4) 이후에는 원하는 파일에 따라 make & 3) 반복