**DCT 연산을 통한 PPI 노드간 상호 거리기반 클러스터링**

안희영,이승열,추재우,오지연

**Ⅰ.서론**

본 알고리즘을 통해 확인하고자 하는 가설은 다음과 같다.

특정 노드집합에 대한 노드의 거리가 보이는 증감이 같은 패턴을 보이며 증감하는 노드들을 가정한다. 해당 노드 집합의 원소인 노드들은 거리를 측정한 노드들을 루트로 하는 트리를 만들었을때, 증감패턴이 동일한 노드들을 같은 모양의 subgraph로 지니고 있을 확률을 가진다고 가정한다.

해당 가정을 바탕으로 같은 증감 패턴을 가진 노드들은 실제 네트워크를 이루는 그래프에서 해당 노드 집합에 접근하기 위해 동일한 브릿지 노드를 통과하여 접근한다고 유추해볼 수 있다. 그리고 동일한 브릿지 노드를 다수 공유하는 노드들은 클러스터일 가능성이 높다고 유추된다.

이 가설을 적용하여 ppi 네트워크의 클러스터의 방법에 대한 아이디어의 한 가지로 특정 노드에 대해 같은 그래프에 속한 노드들 간의 상호 거리에 대한 변화율을 활용한 클러스터링의 가능성에 대해 확인하려 한다.

본 아이디어를 검증하는데 있어서 각 인덱스마다 저장된 거리의 값을 저장하는 배열과 각 인덱스에 저장된 픽셀간의 값을 처리하여 픽셀값의 변화를 바탕으로 이미지의 경계를 찾는 영상처리기법에 대한 유사성을 바탕으로 영상처리에서 주로 사용하는 기법인 DCT 변환을 응용하여 노드간의 거리의 변화와 이에 따른 클러스터링 가능성에 대해 확인하는 아이디어를 검증하고자 한다.

이를 검증하기 위해 고안한 본 알고리즘의 작동 방식은 다음과 같다. 네트워크 클러스터링에 대해 검증하기 위한 방법으로 그래프 내의 모든 노드들과의 거리 행렬을 만들고, DCT 연산으로 얻은 AC계수의 일부값을 이용해 해당 거리 행렬에서 노드들간의 거리 변화를 측정한다.

해당 데이터를 기반으로 기존 클러스터링 알고리즘을 적용하여 클러스터를 만든다. 생성된 클러스터링 결과의 f-score를 기존 알고리즘과 비교하여 유의미한 정확도를 가진 클러스터가 생성되는지 확인한다.

**Ⅱ.실험 방법**

**2.1 데이터 전처리**

클러스터링의 대상이 되는 Biogrid Human ppi 데이터는 307679개의 에지를 이루는 19207개의 human protein 쌍을 저장한 데이터이며 해당 데이터를 클러스터링하여 비교하는 ground truth 로는 Complex human cln flt 데이터를 사용하였다.

ground truth의 데이터를 이루는 protein의 종류와 Biogrid Human ppi 데이터를 구성하는 protein간 겹치는 노드는 4825개의 protein이 속해있었으며 이들이 서로 연결된 edge는 110343개만이 해당 노드들 간의 연결이었다.

Biogrid Human ppi 데이터를 그대로 사용하는 것은 클러스터링 알고리즘의 성능 이상으로 정확도에 영향을 끼친다고 판단된다. Ground truth를 통한 검증의 정확도를 높이기 위해 Biogrid의 노드들 중에서 ground truth에 존재하지 않는 protein data들은 전부 삭제하고 진행한다.

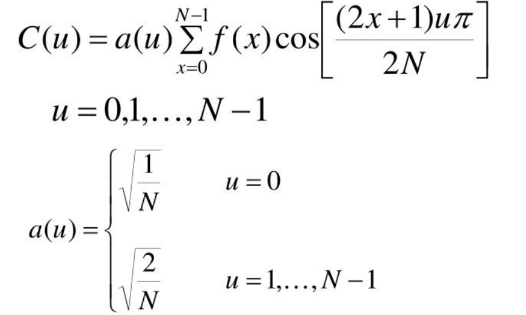
또한 노드가 3개 이하인 분리된 그래프의 경우 unweighted graph 하에서 가능한 클러스터링 결과는 본 알고리즘의 정확도를 확인하는 부분에 있어 이상치에 가까운 데이터기 때문에 독립된 클러스터로 분리하여 알고리즘의 적용 대상에서 제외하였다.

이후 노드가 4개 이상인 그래프들을 서로 연결을 가지지 않는 그래프로 분리해 각 그래프에 대해 개별적으로 클러스터링을 실시한다.

**2.2 거리행렬 생성**

그래프를 구성하는 노드의 수가 M개라고 할때, 해당 그래프의 M개의 노드를 출발점으로 하여 전체 그래프 노드들과의 거리를 BFS 방식으로 계산하여 저장한 M\*M 크기의 거리 행렬을 만든다.

**2.3 DCT**

 DCT는 이산 푸리에 변환의 변형의 일종으로, 결과가 실수 결과값만을 가지는 이점이 있어 신호처리나 영상처리등의 분야에서 공간적, 시간적 변화를 가지는 데이터를 주파수 영역으로 바꿀때 주로 사용된다.

(1)

크기 N의 배열에 대한 DCT 식은 위의 식 (1)과 같다. u는 변환한 값이 저장될 인덱스이며 x는 변환 이전의 값의 인덱스이고 f(x)는 배열에서의 위치 x의 값이다.

해당 계산을 통해 얻을 수 있는 DCT 계수들은 몇가지 특징을 가지고 있는데 인덱스 0에 가까운 값일수록 큰 에너지를 가진다는 점이며 인덱스 0의 값은 DC계수라는 명칭을 가지며 DC계수를 제외한 값을 AC계수라고 부른다. DC계수에는 전체 배열의 평균값에 대한 정보가 저장된다.

또한 인덱스값이 커질수록 작은 에너지의 정보에 대한 값을 저장하기 때문에 DC값에서 거리가 먼 AC값을 적용하지 않고 계산하더라도 해당 값은 DCT를 거치기 전의 값과 크게 다르지 않은 값을 보이게 된다.

이 특징을 사용하여 DCT 계산을 임의로 설정한 M개의 노드에 대해 M/4 크기의 인덱스까지만 계산하고 이후 계산은 무시하는 것으로 계산량을 줄이는 것을 시도한다.

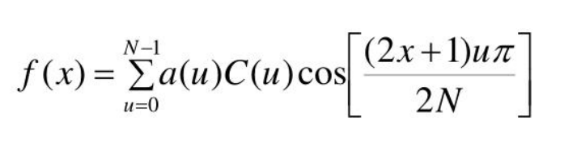
또한 DC값을 0으로 치환하는 것으로 전체 거리의 평균값을 빼는 효과를 낼 수 있으며 본 알고리즘의 가장 heuristic한 결과값은 상호 노드간의 거리의 평균값을 뺴는 것으로 확인 가능하다는 결론을 낼 수 있다.

평균값을 뺴는것 이상으로 정확도를 높이기 위해 DC값에 가까운 인덱스의 AC계수들을 임의의 값인 M/8 까지의 인덱스를 0으로 치환하는 것으로 인접한 인덱스와의 변화값에 가까운 값을 남기도록 조정한다.

더 빠른 DCT 연산을 위해 고속 푸리에 변환을 응용한 FDCT등의 연산법을 적용하거나 M을 임의의 크기의 배열로 나누어 DCT 연산을 적용하는 방식으로 필요한 계산을 줄이고 DC값만으로 인접한 값들의 변화만을 남기는 방법도 적용 가능하나 본 알고리즘에서는 기본적인 DCT 연산만을 사용하여 알고리즘의 컨셉이 가진 기본적인 성능을 확인한다.

**2.3 IDCT**

IDCT는 DCT의 결과값을 다시 원래 값으로 복구시킨다.

(2)

IDCT의 식은 식 (2)와 같다. a(u)의 값은 DCT와 같은 값을 공유한다.

DCT연산으로 변환된 값을 IDCT로 변환하여 나타낸 변화값에 대해 전체 M\*M의 IDCT 거리 행렬 최솟값을 최종적으로 정규화한 값을 최종 IDCT 거리행렬의 값으로 설정한다.

**2.3 DBSCAN**

DBSCAN은 각 데이터간의 거리를 기반으로 클러스터링하는 기본적인 클러스터링 방식이며 파라미터로 주어진 거리 epsilon값 이내에 이웃한 데이터를 임의의 개수인 n개 이상 있는 점을 기준으로 해당 클러스터의 모든 구성원들에 대해 주어진 거리 이내의 모든 데이터들을 클러스터로 추가하는 클러스터링 방식이다.

DBSCAN은 같은 거리를 기반으로 하는 클러스터링 기법인 k-means등의 알고리즘 중에서도 비교적 노이즈에 강하고 클러스터의 숫자를 지정하지 않아도 클러스터링이 가능하다는 장점이 있으며 알고리즘의 목표인 노드간 집합에 대해 특정 노드간 유사한 거리 변화 패턴이 클러스터링에 사용될 정도로 적합한 값인지 확인하는데 있어 클러스터의 모양과 전체 데이터의 분포에 상관없이 클러스터링이 가능한 알고리즘이기에 검증에 적합하다고 판단되어 본 프로젝트에서 적용하는 클러스터링 알고리즘으로 선정하였다.

IDCT를 통해 만들어진 최종적인 M\*M 크기의 IDCT 거리행렬을 DBSCAN에 M개의 축에 대한 좌표값으로 설정하여 클러스터링을 진행한다. 거리값인 epsilon 값은 0.5로 설정하여 진행했다.

**Ⅲ.알고리즘 적용**

**3.1 프로젝트 기여**

해당 알고리즘을 구현하고 비교함에 있어 담당한 역할은 다음과 같다.

안희영 : 신규알고리즘 구현, 레포트 작성

이승열 : fscore 메소드, k-clique 알고리즘 구현,

데이터 분석

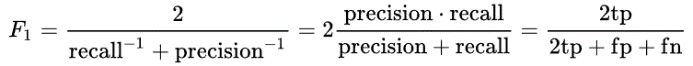
추재우 : 데이터 분석, markov, k-core 알고리즘

오지연 : Louvian 알고리즘 구현, 결과 데이터 정리,

project 발표자료 작성

**3.2 기존 알고리즘과의 비교**

기존 알고리즘과의 비교를 함에 있어 가장 기본적인 scoring 방식인 f-measure 를 채택하여 계산된 f-score를 전체 ground truth의 클러스터와 계산하여 나온 최댓값을 클러스터의 f-score로 삼고 전체 클러스터의 f-score의 평균을 알고리즘의 f-score로 설정하여 각 알고리즘의 f-score와 클러스터 개수, 클러스터의 평균 크기를 확인했다.

(3)

f-score는 식 (3)과 같이 계산된다.

신규 알고리즘과의 비교를 위해 Markov, K-core, Louvian, K-clique 알고리즘을 비교 대상으로 삼아 실제 결과를 비교하였다.

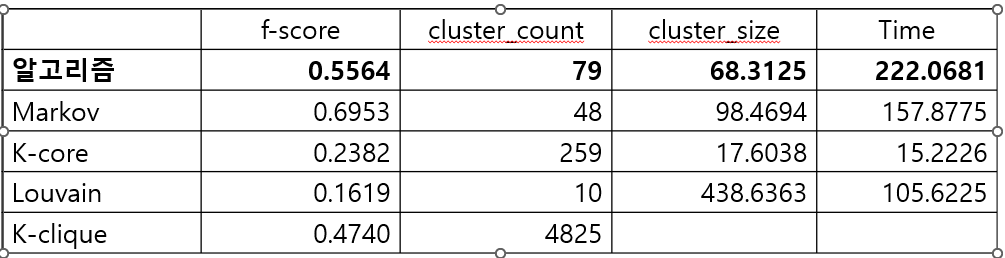
 신규 알고리즘과 기존 알고리즘의 결과는 표 1과 같다.

표 1. 기존 알고리즘과 신규 알고리즘의 비교

신규 알고리즘의 f-score는 0.5564로 markov의 계산시간과 f-score 결과와 비교하면 계산시간과 f-score에서 더 낮은 결과가 나왔지만 k-clique등 markov 이외의 알고리즘에 대해 유의미한 결과를 보인다.

**Ⅳ.개선가능성**

본 알고리즘 컨셉은 markov보다 느린 계산결과를 보여준다. 그러나 알고리즘의 수준이 기본적인 알고리즘의 아이디어에 대한 검증에 머물러 있기 때문에 적용되지 않은 개선방안을 적용하여 계산에 대해 개선될 여지를 남겨두고 있다.

DCT와 IDCT의 경우 푸리에 변환을 기반으로 변형한 변환이기 때문에 고속 푸리에 변환을 응용하여 계산 복잡도를 줄인 FDCT 등의 방식이 존재하며 알고리즘을 구현함에 있어 계산시간의 최소화를 위해 임의의 고주파 값을 0으로 치환하는 단계를 진행했으나 DCT를 지정된 인덱스 이후로 하지 않는 방식을 통해 DCT와 IDCT의 계산 시간을 더 짧게 줄일 수 있다.

DCT의 경우 임의의 크기로 크기 M의 한 열을 나눠 DCT를 진행하는 방법을 적용하는 것도 가능하다. 거리값의 변화를 계산하는데 필요한 값은 local한 변화값만을 필요로 하기 때문에 적절한 크기로 배열을 분할하여 DCT를 진행한다면 더 적은 양의 계산으로 정확한 변화도값만을 남긴 거리배열이 생성 가능하다. 이것은 jpg 등의 DCT 기반 손실압축 이미지 포맷에서 이미 적용되는 방식이다. 배열을 일정 크기로 나눠 DCT를 진행한다면 특정 구간에 위치한 DCT 결과의 AC계수 값만을 비교하여 IDCT 등을 통하지 않고도 클러스터링을 시도해보는 방법을 시도하는 하는 것도 가능하다.

DCT 연산을 제외하고 인접한 배열의 값과 서로 크기의 차이를 그림 1의 중앙차분 마스크 등을 통해 연산하여 적용하는 방안도 가능하며 이 경우 계산복잡도를 크게 줄이는 것이 가능하다.

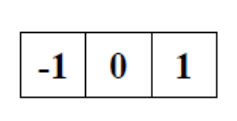


그림 1.

그림 1의 마스크는 이미지 영상 처리에서 외곽선을 찾아내는 방식으로 사용되곤 하며 본 알고리즘의 주요 아이디어는 노드간의 거리에 대한 변화를 기반으로 하고 있기 때문에 영상처리 분야에 적용되는 edge detecting 마스크들과 관련 메소드들을 적용하여 응용이 가능하다고 여겨진다.

계산을 진행함에 있어 가장 길게 걸리는 작업은 M\*M 크기의 거리 행렬을 만드는 계산으로 측정되었다.이는 M\*M의 거리 행렬을 계산하는 부분이 M개의 열에 대하여 각각 독립적으로 이뤄졌기 때문이다.

거리 행렬을 만드는 부분에 대해 개선의 여지가 있으며 이 부분을 개선하는 것이 본 알고리즘의 계산시간을 개선함에 있어 가장 영향이 클 것이다.

혹은 일정 거리 이상의 노드들에 대해선 탐색을 하지 않고 탐색되지 않은 노드들에 충분한 거리를 주는 것으로 거리행렬을 만드는 복잡도를 줄이는 방안도 적용해볼 수 있을 것이다.

정확도에 있어서 개선할 여지로는 DBSCAN의 변경이나 임의로 설정한 파라미터를 계산하는 부분에서 개선의 여지가 남아있다.

DBSCAN은 많은 축을 가지는 데이터에 대해 약점을 가지는데, 본 알고리즘에 적용된 데이터는 4825개의 축을 가지고 있기 때문에 차원을 감소시키는 방식을 적용하거나 DBSCAN 이외의 알고리즘을 적용하는 것으로 더 높은 정확도를 얻을 여지가 존재한다.

DBSCAN은 거리값으로 주어지는 epsilon 파라미터에 매우 민감하기 때문에 IDCT를 통해 얻어진 최종 거리 행렬을 정규화 하여 클러스터링에 사용하기 알맞은 거리를 계산하는 것으로 더욱 정확한 클러스터링이 가능하며 적절히 거리를 조정하는 것으로 클러스터들을 merge 하거나 분리하는 방식을 적용 가능할 것이다.

DCT 연산에 대해서도 임의로 지정한 M/4와 M/8의 인덱스로 연산할 범위를 지정했지만 어떤 값이 원하는 변화값을 가지고 있는지 계산해볼 필요성이 존재하며 거리 행렬을 일정 크기로 나누어 계산할 경우 적절한 크기로 나누어 DC계수와 일정 크기의 AC계수만을 0으로 치환하고 연산할 수 있다.

데이터의 M\*M 거리행렬을 M번 1-D DCT하는 과정에서 M의 크기가 DCT에 적용했을 경우 그 값에 대해 인접한 값과 큰 차이를 보이지 않았을 가능성이 존재하기 때문에 거리 배열을 적절한 크기로 나누어 DCT를 적용한다면 현재 결과와 차이를 보이는 결과가 나올 것으로 예상된다.

DC계수를 이용하여 낮은 순으로 정렬하면 가장 노드간의 거리의 총합이 낮은 노드순으로 정렬된다. 이를 이용하여 K-medoids 등의 알고리즘을 적용하거나 클러스터링을 시작하는 시작점으로 사용하는 등의 응용이 가능하다.

**Ⅴ.결론**

본 알고리즘을 구현하여 실험하는 것으로 노드들간의 상호 거리를 이용하여 클러스터링을 진행하는 것이 가능하다는 결론을 내릴 수 있었다.

충분한 실험과 연구가 이루어지지 않아 본문에 추가되진 않았지만 거리행렬만을 이용하여 DBSCAN을 적용하는 것에 비하여 DCT를 이용한 값은 0.2 이상의 정확도 차이를 보여주었고 ground truth에 속하는 protein 을 필터링 하지 않은 데이터에 대해서는 markov 알고리즘보다 0.02 더 높은 값을 보이는 경향이 있었기 때문에 이에 대한 결과의 분석과 추가적인 실험과 연구를 진행할 필요가 있다.

알고리즘에 대한 개선의 여지는 충분하며 개선방안에 대한 적용이 없는 상태의 기본적인 적용 또한 기존 알고리즘과 비교하여 유의미한 결과를 보여준다. 이를 통해 충분히 개선된 본 알고리즘의 아이디어는 유의미한 결과를 보여줄 것이라는 결론을 내릴 수 있다.

본 알고리즘 아이디어는 알고리즘의 추후 개선과 실험을 통해 통용될만한 알고리즘으로 발전할 가능성이 있다고 기대할 수 있다.