Vol.3, No.5 pp.179~186 pISSN: 2287-5905 http://dx.doi.org/10.3745/KTSDE.2014.3.5.179

RDBMS Based Efficient Method for Shortest Path Searching Over Large Graphs Using K-degree Index Table

Jihye Hong† · Yongkoo Han†† · Young-Koo Lee†††

ABSTRACT

Current networks such as social network, web page link, traffic network are big data which have the large numbers of nodes and edges. Many applications such as social network services and navigation systems use these networks. Since big networks are not fit into the memory, existing in-memory based analysis techniques cannot provide high performance. Frontier-Expansion-Merge (FEM) framework for graph search operations using three corresponding operators in the relational database (RDB) context. FEM exploits an index table that stores pre-computed partial paths for efficient shortest path discovery. However, the index table of FEM has low hit ratio because the indices are determined by distances of indices rather than the possibility of containing a shortest path. In this paper, we propose an method that construct index table using high degree nodes having high hit ratio for efficient shortest path discovery. We experimentally verify that our index technique can support shortest path discovery efficiently in real-world datasets.

Keywords : Shortest Path Search, RDBMS, Indexing

대용량 그래프에서 k-차수 인덱스 테이블을 이용한 RDBMS 기반의 효율적인 최단 경로 탐색 기법

홍 지 혜† · 한 용 구†† · 이 영 구†††

요 약

소셜 네트워크, 웹 페이지 링크, 교통 네트워크 등과 같은 최근의 네트워크들은 노드와 에지의 수가 방대한 빅 데이터이다. 소셜 네트워크 서비스나 네비게이션 서비스와 같이 이와 같은 네트워크를 이용하는 애플리케이션이 많아지고 있다. 대용량 네트워크는 전체를 메모리에 적재할 수 없어, 기존의 네트워크 분석 기술을 활용할 수 없다. 최근 대용량 그래프의 효율적 탐색을 제공하는 RDB 기반 연산자들이 프레임워크(Frontier-expand-merge framework, FEM)로 제안되었다. FEM은 효율적인 최단 경로 탐색을 위해 부분 최단 경로를 저장하는 RDB 기반의 인덱스 테이블을 구축하였다. 그러나 FEM의 인덱스 테이블은 최단 경로에 포함될 확률보다 인덱스의 거리에 의해 결정되기 때문에 인덱스 테이블 참조율이 떨어진다. 본 논문에서는 효율적인 최단 경로 탐색을 지원하는 인덱스 참조율이 높은 차수가 큰 노드들을 이용한 인덱스 테이블 구축 기법을 제안한다. 실험을 통하여 제안하는 인덱스 테이블 구축 기법이 실세계 데이터 셋에서 효율적인 최단 경로 탐색을 지원함을 보인다.

키워드 : 최단 경로 탐색, RDBMS, 인덱싱

1. 서 론

최근 소셜 네트워크의 등장과 센서 기술의 발달로 인해 소셜 네트워크, 웹 페이지 링크, 교통 네트워크와 같이 노드와 에지의 수가 방대한 빅 데이터가 등장하였다. 특히, 소셜 네트워크 서비스나 내비게이션 서비스와 같이 그래프 데이터를 이용하는 애플리케이션이 많아지고 있다. 이와 같은 그래프 데이터는 크기가 매우 빠르게 증가하고 있기 때문에, 데이터의 크기가 매우 방대하여 인-메모리(in-memory) 기법을 통해 연산하기 어렵다. 이에 대용량 그래프 데이터를 디스크 기반(disk-based)으로 처리하는 기법에 대한 관심이 증가하고 있다.

최근 대용량 그래프 상에서 부분 그래프나, 최단 경로 등

※ 이 논문은 2013년도 정부(교육과학기술부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(No. 2013R1A2A1A05056375).
† 준 회 원 : 경희대학교 컴퓨터공학과 석사
†† 비 회 원 : 경희대학교 컴퓨터공학과 연구교수
††† 종신회원 : 경희대학교 컴퓨터공학과 부교수
Manuscript Received : January 27, 2014
First Revision : February 28, 2014
Accepted : February 28, 2014
* Corresponding author : Young-Koo Lee(yklee@khu.ac.kr)

을 탐색하기 위한 기법[1-3]들이 연구되었다. 특히 FEM 프레임워크[3]는 그래프 탐색을 위한 RDB 기반의 관계형 연산자들을 제안하였다. RDB는 너비 우선 탐색, 도달 가능성 질의 등의 다양한 기능과 안정된 인프라를 제공하여 대용량 그래프를 처리하기에 적합하다. 이 연구는 또한 그래프 내에서 최단 경로를 효율적으로 탐색하기 위해, 미리 계산된 부분 경로를 저장하는 인덱스 테이블을 제안하였다. 인덱스 테이블은 임계 거리 미만의 최단 경로들로만 구축한다.

그러나 이와 같은 인덱스 테이블 구축 방법은 인덱스가 최단 경로에 포함될 확률을 고려하지 않고 단순히 거리만을 고려하여, 인덱스의 크기가 증가에 따라 인덱스 참조 비율이 낮아질 수 있다. 효과적인 인덱스 테이블을 구축하기 위해 최단 경로에 포함될 확률을 고려한 인덱스 테이블 구축 기법이 필요하다.

높은 차수의 노드들은 최단 경로에 포함되는 횟수인 매개 중심성이 크다. 따라서 높은 차수의 노드들을 중심으로 인덱스 테이블을 구축하면 인덱스 참조 비율이 높은 인덱스 테이블을 구축할 수 있다.

본 논문에서는 대용량 그래프에서 차수 인덱스 테이블을 이용한 RDB 기반의 효율적인 최단 경로 탐색 기법을 제안한다. 이를 위해 노드의 차수와 최단 경로와의 관계를 분석하고, 차수 기반의 인덱스 테이블로 효율적인 최단 경로를 탐색하는 알고리즘을 제안한다. 실험을 통해 제안하는 차수 기반 인덱스가 기존의 거리 기반 인덱스에 비해 최단 경로 탐색에 효율적임을 보인다.

본 논문의 구성은 다음과 같다. 2절에서는 관련 연구를 소개하고, 3절에서는 기존의 RDB 기반 그래프 탐색 프레임워크를 소개하며, 4절에서는 효율적인 최단 경로를 위한 차수 기반 인덱스 테이블 구축 기법을 제안한다. 5절에서는 실험을 통해 제안하는 기법을 통해 효율적으로 최단 경로를 탐색할 수 있음을 보이며, 6절에서는 결론을 맺는다.

2. 관련 연구

객체들 간의 관계를 표현하는 방법을 제공하는 그래프데이터는 소셜 네트워크나 웹페이지 링크, 텔레폰 네트워크 등 많은 애플리케이션에서 널리 사용된다. 이와 같은 그래프를 효과적, 효율적으로 분석하기 위해, 최단 경로[1-4], 최소 스패닝 트리[5], 순회 판매원 문제[6]와 같은 그래프 알고리즘들이 연구되었다.

최근 네트워크의 크기가 급격하게 증가하여, 네트워크 전체를 메모리에 적재할 수 없다. 따라서 기존의 메모리 기반의 알고리즘들은 성능이 저하되거나 제대로 동작하지 않는다. 대용량 그래프를 처리하기위해 다양한 디스크 기반의 기법들이 연구되었다. 첫 번째, 대용량 그래프를 메모리 크기의 부분그래프로 분할하는 기법[7]이 제안되었다. 두 번째, 맵리듀스 프레임워크를 이용한 분산 처리 기법들[8-9]이 제안되었다. 그러나 이와 같은 기법들은 복잡한 데이터의 스키마나 인덱스를 안정적으로 지원하지 않기 때문에 그래프 데이터를 처리하기 어렵다.

최근 RDBMS에서 대용량 그래프를 마이닝하기 위한 연구가 활발하게 진행되고 있다[10-12]. RDB는 너비 우선 탐색, 도달 가능성 질의 등의 그래프 탐색 기능과 안정된 인프라를 갖고 있기 때문에 그래프를 처리하기에 적합하다. HDB-SUBDUE[13]와 DB-FSG[14]는 기존의 메모리 기반의 빈발 부분그래프 마이닝을 RDB 기반으로 수행하는 효율적인 기법을 제안하였다. FEM 프레임워크[12]는 RDB 기반으로 그래프 매칭, 최단 경로 탐색 등 그래프 탐색에서 일반적으로 사용할 수 있는 관계형 연산자들을 제안하였다.

최단 경로를 효율적으로 탐색하기 위해 다양한 인덱싱 기법들이 제안되었다[12, 15] ALT 알고리즘[15]에서는 랜드마크로 여겨지는 몇 개의 주요 노드로부터 다른 모든 노드들까지의 최단 경로를 미리 계산하여 새로운 에지로 추가한다. 그러나 ALT 알고리즘은 메모리 기반으로 설계되어 대용량의 그래프에 대한 성능을 보장하기 어렵다.

최근 FEM 프레임워크[12]에서는 대용량 그래프에서 최단 경로 탐색을 효율적으로 수행하기 위해 특정 임계 거리 미만의 최단 경로를 미리 계산하는 인덱스 테이블 구축 기법을 제안하였다. 그러나 FEM 프레임워크에서 제안한 인덱스는 최단 경로에 포함될 확률과 무관하게 거리만을 고려하여 인덱스를 구축하므로, 전체 인덱스 크기에 대해 인덱스가 사용되는 비율을 예측하기 어렵다.

3. RDB 기반의 효율적인 그래프 탐색 프레임워크

최단 경로 탐색, 도달 가능성 탐색 등과 같은 대부분의 그래프 탐색 알고리즘들은 질의에 대한 결과를 포함할 가능성이 높은 노드들을 반복적으로 확장하며 결과 값을 찾는 공통된 패턴을 가지고 있다[12]. FEM 프레임워크[12]는 그래프 탐색에 공통적으로 사용되는 연산을 지원하기 위해 3가지 기본적인 연산자를 제공한다.

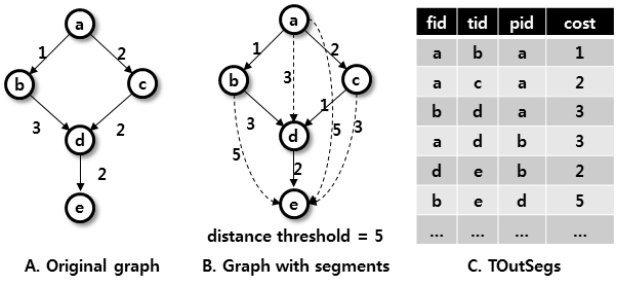
- F-operator는 현재까지 살펴본 노드들 가운데 다음 순서로 확장할 노드를 선택한다.
- E-operator는 F-operator를 통해 선택된 노드 집합 F^k에서 도달 가능한 모든 노드들로 확장한 결과 집합 E^k를 반환한다.

· M-operator는 현재까지 방문한 노드들의 집합인 A^k와 확장된 노드 집합 E^k를 기반으로 방문한 노드 집합을 A^{k+1}로 갱신한다.

Breadth First Search(BFS) 방식의 탐색은 한 번의 연산으로 많은 노드를 확장하여 탐색 공간과 시간 비용을 절약할 수 있으나, 대용량 그래프에서 긴 경로를 갖는 최단 거리 탐색이 발생할 경우 알고리즘의 반복 횟수가 성능에 영향을 미친다. FEM 프레임워크는 임계 거리 미만의 경로 세그먼트에 대해 미리 계산된 인덱스 테이블을 구성하여 효율적으로 최단 경로를 탐색하였다. 경로 세그먼트들을 기존의 노드 확장과 동일한 방법으로 확장하여, 불필요한 반복을 줄일 수 있다.

그림 1은 인덱스 테이블의 예시이다. 그림 1A 그래프의 모든 노드에서 l_{thd} 이하의 거리로 도달할 수 있는 부분 최단 경로인 경로 세그먼트가 TOutSegs에 저장된다. 그림 1C와 같이, TOutSegs 테이블은 fid, tid, pid, cost의 4개의 열로 구성된다. fid는 각 경로 세그먼트의 출발 노드, tid는 도착 노드, pid는 tid의 부모 노드이며, cost는 경로 세그먼트의 거리를 나타낸다. 그림 1B의 점선은 경로 세그먼트를 나타낸다.

예를 들어, $l_{thd}=5$이므로 $\delta(a,d)=4$인 경로 세그먼트 (a,d)가 생성되어야 한다. $TOutSegs$의 4번째 레코드는 이 세그먼트를 저장한다.



fid	tid	pid	cost
a	b	a	1
a	c	a	2
b	d	a	3
a	d	b	3
d	e	b	2
b	e	d	5
...	...	...	...

Fig. 1. Example of index table

4. 효율적인 최단 경로 탐색을 지원하는 RDB 기반의 k-차수 인덱스 테이블 구축 기법

본 절에서는 제안하는 효율적인 최단 경로 탐색을 위한 RDB 기반의 k-차수 인덱스 테이블 구축 기법을 소개한다. 4.1절에서는 k-차수 인덱스 테이블 구축을 위한 용어를 정의하며, 4.2절에서는 매개 중심성과 차수의 관계를 설명하고 k-차수 인덱스 테이블을 구축하는 기법을 설명한다.

4.1 용어정의

본 절에서는 k-차수 인덱스 테이블 구축과 관련된 용어들을 정의한다.

정의 1. 지역 최단 경로 $G=|V,E|$의 부분 그래프 $G=|V_{sub},E_{sub}|$에서 두 노드 $u,v\in V_{sub}$의 최단 경로이며, $lsp(u,v)=u\rightarrow m_1\rightarrow m_2\rightarrow\ldots\rightarrow v$로 표현한다. 이때, 지역 최단 경로의 모든 노드들은 V의 원소이다.

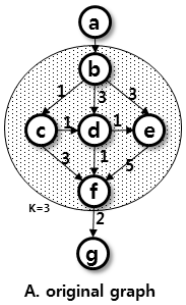
정의 2. 임계 차수 $G=|V,E|$의 어떤 $v\in V$의 인접한 모든 간선의 개수가 k이고 최솟값일 때, 그래프 G의 임계 차수는 k이다.

정의 3. k-차수 인덱스 그래프 임계 차수가 k일 때, 차수가 k 이상인 모든 노드들의 집합 V와 임의의 $u,v\in V$ 사이의 모든 에지 E로 구성된 부분 그래프 $G=|V,E|$에 대해, 모든 $u,v\in V$ 쌍에 대해, 최단 경로 $sp(u,v)$를 최단 길이를 가중치로 갖는 에지로 추가한 그래프이다.

정의 4. 인덱스 테이블 참조율 최단 경로 탐색을 위한 확장 횟수가 n회이고, 탐색 공간에 인덱스가 포함되는 횟수가 m회일 때, 인덱스 테이블 참조율은 m/n이다. 인덱스 테이블 참조율이 높을수록 인덱스 테이블을 통해 효율적으로 최단 경로를 탐색할 수 있다.

정의 2의 임계 차수를 결정하기 위해, 허용되는 인덱싱 공간의 크기 M과 차수별 노드의 분포 함수 $f(x)$를 고려하여 $\{\sum_{k}^{\lfloor H\rfloor}f(x)\}^2\leq M$을 만족하는 가장 작은 k 설정하여, 효율적으로 최단 경로를 탐색할 수 있다.

그림 2는 3-차수 인덱스 테이블의 예시이다. 그림 2A의 그래프에서 차수가 3 이상인 모든 노드와, 노드들 간의 에지로 부분 그래프 G가 형성되고, G 내의 모든 지역 최단



fid	tid	pid	cost
b	e	d	2
b	d	c	2
b	f	d	3
...	...	...	...

Fig. 2. Example of 3-degree index table

경로를 계산하여 그림 2B와 같이 테이블로 구성한다. 거리 기반 인덱스 테이블에서 임계 거리가 2라면, 거리가 3인 경로 세그먼트 (b,f)는 생성되지 않으나, 이 세그먼트는 (a,f), (a,g), (b,g) 등의 경로에 포함되는 매개중심성이 높은 세그먼트이다. 이처럼 차수 기반의 인덱스 테이블은 매개 중심성이 높은 경로 세그먼트를 생성한다.

4.2 k-차수 인덱스 테이블

최단 경로 탐색에서 매개 중심성(betweenness centrality)과 차수(degree)가 높은 노드는 탐색의 성능에 중요한 영향을 준다. 매개 중심성은 특정 노드를 거치는 최단 경로의 수를 나타내므로, 차수가 높은 노드가 큰 매개 중심성을 가질 확률이 높다[16-17]. 따라서 차수가 높은 노드가 인덱스 테이블에 포함되면 인덱스 테이블의 참조 비율이 높아서 최단 경로 탐색의 성능을 향상시킬 수 있다.

본 논문에서는 매개 중심성 계산이 모든 쌍의 최단 경로를 찾는 추가적인 연산이 필요한 점을 고려하여, 임계 차수 이상의 노드들 간의 최단 경로를 사전에 계산한 값들로 인덱스 테이블을 구축한다.

k-차수 인덱스 그래프에 대하여, 지역 최단 경로를 이용하여 인덱스 테이블을 구축하여도 올바른 최단 경로를 구할 수 있음을 정리 1에서 증명한다.

정리 1. 단방향 다익스트라 알고리즘에서, 그래프 G에서 노드 s와 d 사이의 최단 경로가 $sp(s,d) \in G$ 이고 k-차수 인덱스 그래프와 그래프 G를 병합한 그래프 $G+G_{sub}$에서 노드 s와 d 사이의 최단 경로가 $sp'(s,d)$라면, $sp(s,d)=sp'(s,d)$이다.

증명. 노드 u로부터 확장한 세그먼트 $seg=u\rightarrow...\rightarrow v$가 거리 k로 G_{sub}의 $lsp(u,v)$를 만족하고 $sp(u,v)$를 만족하지 못할 때, 만일 최단 경로가 $sp(s,d)=s\rightarrow...\rightarrow u\rightarrow...\rightarrow v\rightarrow...\rightarrow d$로 세그먼트를 포함한다면, 시작 노드로부터 현재까지 확장한 모든 노드들 중 최소 거리를 갖는 노드를 다음 확장할 노드로 선정하는 원리에 의해 이후의 확장에서 $sp(u,v)$에 도달하게 된다. 동일한 원리에 의해 $sp(s,u)+sp(u,v)+sp(v,d)$가 최단 경로로 결정되므로 $sp(s.d)=sp'(s,d)$이다. 만일 최단경로가 세그먼트를 지나지 않는 $sp(s,d)=s\rightarrow...\rightarrow u'\rightarrow...\rightarrow d$라면, $dist(p(s,u')) \leq k$를 만족하는 노드 $u' \in G_{sub}+G$가 확장되어, $sp(s,u')+sp(u',d)$가 최단 경로로 결정된다. 그러므로 인덱스 에지를 포함하여 탐색하여도 포함하지 않은 결과와 같다.

그림 3은 인덱스를 이용한 최단 경로 탐색 예제이다. 2번째 확장에서 $lsp(u,v)$가 확장되지만, $dist(p(u,v)) \leq k$이므로 v가 3번째 확장노드가 되지 않는다. $n-1$번째 확장에서

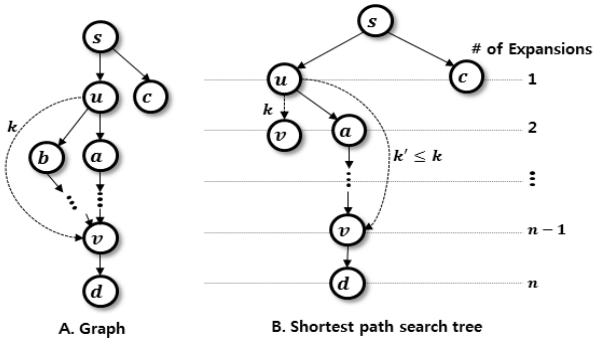


Fig. 3. Example of searching shortest path

$sp(u,v)$에 도달하여, 최단경로 $sp(s,d)=s\rightarrow u\rightarrow a\rightarrow...\rightarrow v\rightarrow d$를 올바르게 도출한다.

인덱스 테이블의 크기가 크면 클수록 사전에 계산한 경로가 많아지기 때문에, 단일 노드 간의 최단 경로 탐색의 성능이 개선된다. 그러나 인덱스 테이블은 디스크 공간에 유지되므로, 애플리케이션의 환경과 그래프의 크기를 고려하여 인덱스 테이블 크기를 결정해야 한다. 따라서 이들의 관계를 정리 2에서 추정한다.

정리 2. 임계 차수가 k이며, 차수에 따른 노드 수의 분포 함수가 $f(x)$일 때, k-차수 인덱스 테이블의 크기는 $O(\{\sum_{k}^{|V|} f(x)\}^2)$로 추정할 수 있다.

증명. k-차수 인덱스 그래프 $G_{sub}=|V_{sub},E_{sub}|$는 k 이상의 모든 노드와, 노드들 간의 에지들로 구성되며, 이때 모든 쌍의 최단 경로를 찾아 인덱스를 구성한다. 그러므로 그래프에서 k 이상의 차수를 갖는 노드의 수는 $|V_{sub}|=\sum_{k}^{|V|} f(x)$이며, 모든 쌍의 최단 경로에서 가능한 노드 쌍의 수는 $|V_{sub}|^2$와 같으므로, k-차수 인덱스 테이블의 크기는 $O(\{\sum_{k}^{|V|} f(x)\}^2)$이다.

RDB 기반으로 차수 인덱스 테이블을 구축하기 위한 알고리즘은 크게 2단계로 나뉜다. 첫 번째로 임계 차수 k 이상의 차수를 갖는 노드와 그 노드 간의 에지들로 부분 그래프를 구성한다. 그리고 두 번째 단계에서 이 부분 그래프를 이용하여 k 차수 인덱스 테이블을 구축한다.

알고리즘 1은 차수 인덱스 테이블을 구축하는 알고리즘이다. 인덱스 테이블을 구축하기 위한 임계 차수는 k이다. 일단 k-차수 인덱스 그래프를 생성하기 위해 k 이상의 차수를 갖는 노드 집합을 관계형 테이블로 구축하고(line 1), k-차수 인덱스 그래프의 에지를 저장하는 관계형 테이블을 구축한다(line 2). 임계 차수 이상의 노드들을 확장할 노드로 선택한다.(line 3) 선택된 노드들을 임계 차수 이상의 노드들

Algorithm 1. K-degree index table construction algorithm

Algorithm Construction Degree Index(Threshold k)
• **Input:** degree threshold k • **Output:** k-degree index table
1: Construct an node table of k-degree index graph G_{sub} using Table 1A 2: Construct an edge table of G_{sub} using Table 1B 2: **REPEAT** 3: Select a set of frontier nodes which have degree value more than k 4: Expand edges in k-degree index graph 5: Merge expanded nodes into k-degree index table S using Table 1C 6: **UNTIL** There is no frontier node 7: Insert remain edges into S using Table 1D 8. **RETURN** S

과 연결된 에지로 확장하고, (line 4) k-차수 인덱스 테이블에 병합한다.(line 5) 이 과정을 더 이상 확장할 노드가 없을 때까지 반복하여 k-차수 인덱스 테이블을 구축한다. 반복이 끝나면, k-차수 인덱스 테이블에 추가되지 않은 남은 에지들을 추가한다.(line 7)

표 1은 알고리즘 1을 수행하기 위한 SQL문을 나타낸다. 표 1A는 차수 임계값 thd 이상의 노드들에 대해 k-차수 인덱스 그래프를 구축하기 위한 노드 테이블 생성 SQL문이다. 표 1B는 k-차수 인덱스 그래프의 에지 테이블을 생성하기 위한 SQL문을 나타낸다. 표 1C는 생성한 k-차수 인덱스 그래프로부터 k-차수 인덱스 테이블을 구축하기 위해 확장할 노드를 선택하고 에지를 확장한 후, 최종적으로 인덱스 테이블에 병합하기 위한 SQL문을 나타낸다. 표 1D는 최종적으로 인덱스 테이블에 추가되지 않은 남은 에지들을 그래프에 인덱스 테이블에 추가하여 그래프를 완성한다.

알고리즘 1을 통해 구축한 k-차수 인덱스 테이블을 이용하여 최단 경로를 효율적으로 탐색하기 위해, 도착 노드에 도달 가능한 에지가 더 이상 없을 때까지 반복적으로 에지를 확장한다. k-차수 인덱스 테이블에 저장된 지역 최단 경

Table 1. SQL statements for construction k-degree index table

A:Construct a node table of k-degree index graph

```
CREATE TABLE
DEG_SUBGRAPH_NODE(nid,nlabel)
AS SELECT nid, label
FROM SMALL_NODE WHERE label>=thd;
```

B:Construct an edge table of k-degree index graph

```
CREATE TABLE
DEG_SUBGRAPH_EDGE(fid,tid,elabel)
AS SELECT fid, tid, elabel
FROM SMALL_EDGE e, DEG_SUBGRAPH_NODE n, DEG_
SUBGRAPH_NODE n2
WHERE (fid=n.nid and n.nlabel>=thd) AND
(tid=n2.nid and n2.nlabel>=thd)
```

C:Expand edges in k-degree index graph

```
MERGE INTO DegreeIndexTable
target USING (select nid, p2s, dist, rnum, '0', fid
FROM (select oe.tid as nid,oe.fid as p2s, oe.elabel+q.dist as dist,
row_number() over (partition by oe.tid order by oe.elabel+
q.dist asc) rnum, '0', q.srcid as fid FROM DegreeIndexTable
q, DEG_SUBGRAPH_EDGE oe
WHERE q.nid=oe.fid and q.flag=2)
WHERE rnum=1)
src ON (src.nid = target.nid and src.fid=target.srcid)
WHEN MATCHED THEN UPDATE SET target.dist=src.dist,
target.p2s=src.p2s
WHERE target.dist>src.dist
WHEN NOT MATCHED THEN
INSERT (nid, p2s, dist, rnum, flag, srcid)
VALUES (src.nid, src.p2s, src.dist, src.rnum, 0, src.fid)
```

D:Merge expanded nodes into k-degree index table

```
MERGE INTO DegreeIndexTable target
USING (select fid, tid, label from EdgeTable o) src ON
(src.tid=target.nid and src.fid=target.srcid) WHEN NOT MATCHED
THEN
INSERT (target.nid, target.dist, target.p2s, target.flag, target.rnum,
target.srcid)
VALUES (src.tid, src.label, src.fid, '0', '1', src.fid)
```

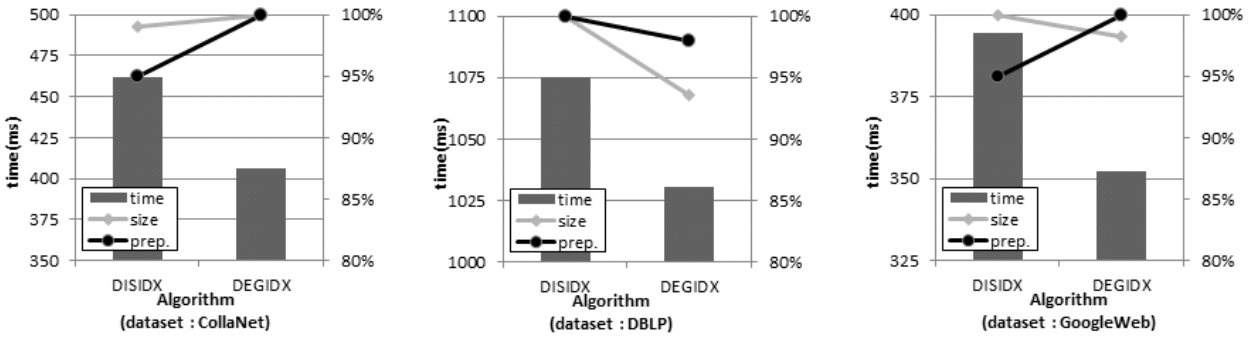


Fig. 4. Comparisons of average time costs, index construction time, and index sizes

로 및 에지를 함께 확장하여 최단 경로 탐색을 위한 반복 횟수를 줄일 수 있다. 기존의 거리 기반 인덱스 테이블과 비교하여, 차수 인덱스 테이블에 포함된 에지들의 평균 매개 중심성이 높다. 에지 e의 매개 중심성은 최단 경로에 포함되는 빈도수와 연관되므로, 평균 매개 중심성이 클수록 임의의 최단 경로 탐색에서 인덱스 테이블의 참조율이 높다. 따라서 제안하는 k-차수 인덱스 테이블을 이용하여 효율적으로 최단 경로를 탐색할 수 있다.

5. 실 험

5.1 실험 환경

실험을 위하여 3.3GHz Intel® CoreTM i7-3960X 프로세서를 사용하였으며, 실험에 사용한 운영체제는 Window 7이다. 표 2는 3개의 실세계 데이터 셋인 Collaborative network[18], DBLP[19], GoogleWeb[20] 데이터 셋들을 상용화 데이터베이스에 저장하여 실험을 진행하였다.

Table 2. Dataset description

Datasets	# of nodes	# of edges
Collaborative network	40,000	170,000
DBLP	312,967	1,149,663
GoogleWeb	875,713	5,105,039

제안하는 기법의 우수성을 입증하기 위해 기존의 거리 기반 인덱스(distance-based index, DISIDX)와 제안하는 k-차수 인덱스(degree-based index, DEGIDX)를 사용한 최단 경로 탐색 시간을 비교하였다. 또한, 제안하는 기법에서 임계 차수 k의 값의 변경에 따른 최단 경로 탐색의 수행 성능을 측정하였다. 최단 경로 탐색은 단방향 다익스트라 알고리즘[12]를 사용하였다.

5.2 실험 결과

그림 4는 데이터 셋에 따라 최단 경로 탐색 질의 성능, 인덱스 테이블 구축 시간과 인덱스 크기를 비교한 결과이다. 임의로 생성한 1,000개의 노드 쌍에 대해 최단 경로 탐색 질의를 수행하여 평균 시간을 측정하였다. 유사한 크기의 인덱스 테이블 구축을 위해 표 3과 같이 임계 거리와 임계 차수를 설정하였다.

실험 결과, 최단 경로 탐색 질의 성능은 기존의 인덱싱 기법인 DISIDX와 비교하여 제안하는 기법을 이용할 때, Collaborative network 데이터 셋에서는 약 13%, DBLP 데이터 셋에서는 약 5%, GoogleWeb 데이터 셋에서는 약

Table 3. Distance and degree threshold

Datasets	Distance threshold	Degree threshold
Collaborative network	3	5
DBLP	3	10
GoogleWeb	3	25

10%만큼 성능이 향상되었다. 제안하는 기법은 기존의 기법에 비해 인덱스 참조율이 높으므로 성능이 향상된다. 그림 4에서 인덱스의 크기는, 크기가 더 큰 기법을 100%로 하였을 때 상대적인 크기를 나타낸다. 모든 경우에 DISIDX에서 사용한 인덱스와 약 5% 미만의 차이를 갖는 유사한 크기로 임계 차수를 설정하였다. 또한, 인덱스 구축을 위한 전처리 시간은 크기에 비례하는 경향을 보이므로 95% 이상 유사하다.

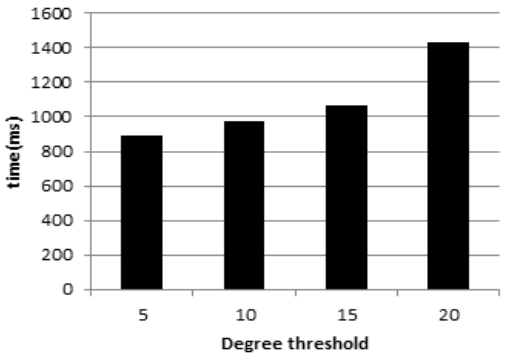


Fig. 5. Comparisons of time costs according to the degree threshold

그림 5는 차수의 증가에 따른 수행 시간을 나타낸다. 임의로 생성한 1,000개의 노드 쌍에 대해 최단 경로 탐색 질의를 수행하여 평균 시간을 측정하였다. 임계 차수가 커질수록 k-차수 인덱스 그래프의 규모가 작아지므로 수행 시간이 증가한다. 임계 차수가 작을수록 기 계산한 경로의 수가 많아지므로 성능이 개선된다.

6. 결 론

본 논문에서는 대용량 그래프에서의 최단 경로 탐색을 지원하는 k-차수 인덱스 테이블 구축 기법을 제안하였다. 제안하는 k-차수 인덱스 테이블은 기존의 거리 기반 인덱스 테이블에 비해 인덱스 테이블 참조율이 높아, 효율적인 최단 경로 탐색을 지원한다. 또한, 임계 차수 k를 사전에 결정할 수 있도록, 임계 차수에 따른 인덱스 테이블 크기를 추정하는 방법을 제안하였다. 실험을 통하여 차수를 고려한 제안하는 인덱스 테이블 구축 기법이 거리 기반 인덱스 테

이블을 이용한 기존의 기법에 비해 5~13%만큼 성능이 향상되었다.

Reference

[1] E. Dijkstra. A note on two problems in connexion with graphs. Numerische Mathematik, 1, 1, 1959.

[2] D. Wagner and T. Willhalm. Speed-up techniques for shortest-path computations. Proceedings of the 24th International Symposium on Theoretical Aspects of Computer Science, February 22-24, Aachen, Germany, 2007.

[3] A. Goldberg and C. Harrelson. Computing the shortest path:search meets graph theory. Proceedings of the 16th annual ACM-SIAM symposium on discrete algorithms SODA, January 23-25, Vancouver, British Columbia, 2005.

[4] F. Wei. Tedi: efficient shortest path query answering on graphs. Proceedings of the 29th ACM SIGMOD International Conference on Management of Data, June 6-11, Indianapolis, USA, 2010.

[5] D. Johnson and L. McGeoch. The traveling salesman problem: A case study in local optimization. Local search in combinatorial optimization, 1995.

[6] R. Prim. Shortest connection networks and some generalizations. Bell System Technical Journal, 36, 6, 1957.

[7] Y. Yuan, G. Wang, H. Wang and L. Chen. Efficient subgraph search over large uncertain graphs. PVLDB, 4, 11, 2011.

[8] J. Dean and S. Ghemawat. Mapreduce: Simplified data processing on large clusters. Proceedings of the 6th symposium on Operating Systems Design & Implementation, December 6-8, San Francisco, CA, 2004.

[9] B. Bahmani, K. Chakrabarti, and D. Xin. Fast personalized pagerank on mapreduce. Proceedings of the 30th ACM SIGMOD International Conference on Management of Data, June 12-16, Athens, Greece, 2011.

[10] C. Wang, W. Wang, J. Pei, Y. Zhu, and B. Shi. Scalable mining of large disk-based graph databases. In SIGKDD, pp.316-325, 2004.

[11] R. H. Möhring, H. Schilling, B. Sch¨utz, D. Wagner and T. Willhalm, "Partitioning Graphs to Speed Up Dijkstra's Algorithm," In: Nikoletseas, S.E. WEA. LNCS, Vol.3503, pp. 189-202, 2005.

[12] J. Gao, R. Jin, J. Zhou, J. Yu, X. Jiang and T. Wang, "Relational Approach for Shortest Path Discovery over Large Graphs," In: PVLDB, 5(4), pp.358-369, 2011.

[13] S. Padmanabhan and S. Chakravarthy. HDB-Subdue: A Scalable Approach to Graph Mining. Proceedings of the 11th International Conference on Data Warehousing and Knowledge Discovery, August 31-September 2, Linz, Austria, 2009.

[14] S. Chakravarthy and S. Pradhan. DB-FSG: An SQL-based approach for frequent subgraph mining. Proceedings of the 19th International Conference on Database and Expert Systems Applications, September 1-5, Turin, Italy, 2008.

[15] A. V. Goldberg and C. Harrelson. Computing the shortest path: A* search meets graph theory. In SODA, pp.156-165, 2005.

[16] A. B. David, M. Kamesh, "A graph-theoretic analysis of the human protein-interaction network using multicore parallel algorithms," Proc. 6th Workshop on HiCOMB, 2007.

[17] A. De. Montis, S. Caschili, "Nuraghes and landscape planning: Coupling viewshed with complex network analysis," In: Landscape and Urban Planning, Vol.105, Issue 3, pp.315-324, 2012.

[18] J. Yang and J. Leskovec. Defining and Evaluating Network Communities based on Ground-truth. Proceedings of the 18th ACM SIGKDD Workshop on Mining Data Semantics, August 12-16, Beijing, Chinam, 2012.

[19] J. Leskovec, K. Lang, A. Dasgupta and M. Mahoney. Community Structure in Large Networks: Natural Cluster Sizes and the Absence of Large Well-Defined Clusters. Internet Mathematics, 6, 1, 2009.

[20] M. E. J. Newman. The structure of scientific collaboration networks. National Academy of Sciences, 98, 2, 2001.

홍 지 혜

e-mail : hjhh@khu.ac.kr

2011년 경희대학교 컴퓨터공학과(학사)

2014년 경희대학교 컴퓨터공학과 석사

관심분야 : 대용량 데이터 관리, 데이터 마이닝

한 용 구

e-mail : ykhan@khu.ac.kr

2005년 경희대학교 컴퓨터공학과(학사)

2007년 경희대학교 컴퓨터공학과(석사)

2012년 경희대학교 컴퓨터공학과(박사)

2013년 경희대학교 컴퓨터공학과 연구교수

관심분야 : 대용량 데이터 관리, 데이터 마이닝

이 영 구

e-mail : yklee@khu.ac.kr

1992년 한국과학기술원 전산학과(학사)

1994년 한국과학기술원 전산학과(석사)

2002년 한국과학기술원 전산학과(박사)

2004년 미국 UIUC 전산학과 Post Doctoral Research Fellow

2006년~현 재 경희대학교 컴퓨터공학과 부교수

관심분야 : 대용량 데이터 관리, 클라우드 컴퓨팅, 데이터 마이닝