(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2016년09월09일
(11) 등록번호 10-1656619
(24) 등록일자 2016년09월05일

(51) 국제특허분류(Int. Cl.)
G06F 17/30 (2006.01)
(52) CPC특허분류
G06F 17/30651 (2013.01)
G06F 17/30958 (2013.01)
(21) 출원번호 10-2015-0079652
(22) 출원일자 2015년06월05일
심사청구일자 2015년06월05일
(56) 선행기술조사문헌
KR1020060045780 A

(73) 특허권자
포항공과대학교 산학협력단
경상북도 포항시 남구 청암로 77 (지곡동)
서울대학교산학협력단
서울특별시 관악구 관악로 1 (신림동)
(72) 발명자
한욱신
경상북도 포항시 남구 청암로 77 창의IT융합공학과 (지곡동, 포항공과대학교)
이준영
대구광역시 수성구 청호로69길 30, 201동 1002호 (황금동, 가든하이츠2차아파트)
(뒷면에 계속)
(74) 대리인
특허법인이룸리온, 특허법인리온

전체 청구항 수 : 총 4 항 심사관 : 이복현

(54) 발명의 명칭 **RBI 그래프 기반의 서브 그래프 리스팅 방법**

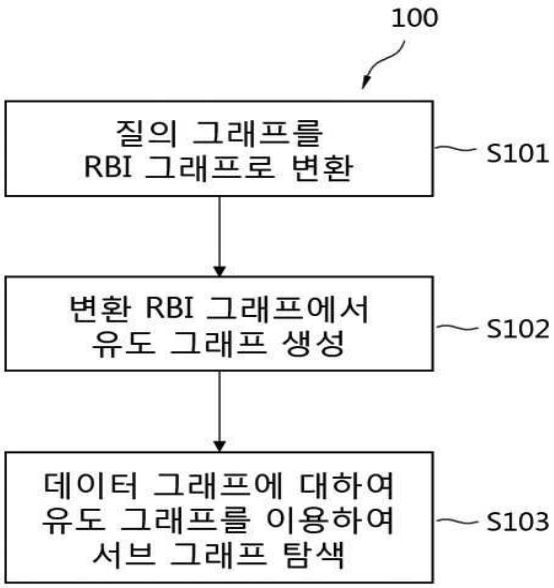
(57) 요 약

데이터 그래프에서 질의 그래프에 대한 서브 그래프 리스팅 방법에 있어서,

상기 질의 그래프에서 데이터 그래프 내 정점을 디스크로부터 직접 액세스 해야 하는 제 1 정점, 두 개 이상의 상기 제 1 정점의 인접 정보의 교집합을 통해 매핑 가능한 제 2 정점, 및 하나의 상기 제 1 정점의 인접 정보를

(뒷면에 계속)

대 표 도 - 도1



이용하여 매핑 가능한 제 3 정점으로 질의 그래프 정점을 분류하는 단계; 상기 제 1 정점 및 제 1 정점의 간선으로 구성된 유도 그래프를 생성하는 단계; 및 상기 유도 그래프를 이용하여 서브 그래프를 탐색하는 단계;를 포함하는 것을 특징으로 하는 서브 그래프 리스팅 방법이 제공된다.

본 발명의 서브 그래프 리스팅 방법에 의하면 데이터 그래프가 큰 경우라도, 디스크 액세스 횟수를 최소화하여 데이터 그래프의 질의 그래프에 대한 서브 그래프 리스팅을 효율적으로 수행할 수 있다.

본 발명은 대규모의 데이터 그래프에서 서브 그래프 리스팅을 효율적으로 처리할 수 있는 RBI 그래프를 도입하여 디스크의 I/O 횟수를 대폭 감소 시킬 수 있다.

본 발명의 서브 그래프 리스팅 방법은 메모리 버퍼를 내부, 피벗 및 외부 서브 그래프로 나눌 수 있으며, 그에 따라 내부, 피벗 및 외부 리스팅을 독립적으로 수행할 수 있어 동시 작업이 가능하다.

본 발명의 서브 그래프 리스팅 방법은 RBI 그래프 및 이를 이용한 매칭을 통해 특히 대규모 데이터 그래프에서 서브 그래프 리스팅을 효과적으로 처리할 수 있다.

또한, 본 발명의 서브 그래프 리스팅 방법은 깊이 우선 탐색을 기반으로 한 매핑 방식을 이용하여 매핑 중간 결과를 저장하지 않아 저장 공간의 오버헤드 없이 효과적으로 매핑을 수행할 수 있다.

본 발명의 서브 그래프 리스팅 방법은 매핑 중간 결과를 저장하는 일이 줄게 되어 서브 그래프 리스팅의 수행 처리 속도가 크게 감소할 수 있다.

(72) 발명자

김현지

울산광역시 동구 월봉12길 50, 1C동 308호 (화정동, 송정타워맨션3차)

이진수

경상북도 경산시 신천길 28 (점촌동)

이 발명을 지원한 국가연구개발사업
과제고유번호 2012M3C4A7033342
부처명 미래창조과학부
연구관리전문기관 한국연구재단
연구사업명 차세대정보컴퓨팅기술개발사업
연구과제명 소셜 및 정보 네트워크 빅데이터 마이닝 소프트웨어 원천 기술 개발
기 여 율 5/100
주관기관 서울대학교
연구기간 2014.07.01 ~ 2015.06.30
이 발명을 지원한 국가연구개발사업
과제고유번호 2015021977
부처명 미래창조과학부
연구관리전문기관 한국연구재단
연구사업명 중견연구자지원사업
연구과제명 신약 발견 및 광고 대행 추천을 위한 고성능 top-K 검색 엔진의 개발
기 여 율 25/100
주관기관 포항공과대학교 산학협력단
연구기간 2015.05.01 ~ 2016.04.30
이 발명을 지원한 국가연구개발사업
과제고유번호 IITP-2015-R0346-15-1007
부처명 미래창조과학부
연구관리전문기관 정보통신기술진흥센터
연구사업명 ICT명품인재양성사업
연구과제명 미래IT융합연구원
기 여 율 25/100
주관기관 포항공과대학교 산학협력단
연구기간 2015.01.01 ~ 2015.12.31
이 발명을 지원한 국가연구개발사업
과제고유번호 40011141
부처명 한국마이크로소프트(유)
연구관리전문기관 한국마이크로소프트(유)
연구사업명 일반산업체 과제
연구과제명 매시브 네트워크에서 효율적인 서브그래프 리스팅
기 여 율 45/100
주관기관 포항공과대학교 산학협력단
연구기간 2014.06.13 ~ 2015.06.12

명 세 서

청구범위

청구항 1

데이터 그래프에서 질의 그래프에 대한 서브 그래프 리스팅 방법에 있어서,

질의 그래프의 정점들을 메모리에 로드된 상기 데이터 그래프의 정점에 직접 매핑(Retrieval) 해야 하는 제 1 정점, 두 개 이상의 상기 제 1 정점의 인접 정보의 교집합을 통해 매핑 가능한 제 2 정점, 및 하나의 상기 제 1 정점의 인접 정보를 이용하여 매핑 가능한 제 3 정점으로 질의 그래프 정점을 분류하는 단계;

상기 제 1 정점 및 제 1 정점의 간선으로 구성된 유도 그래프를 생성하는 단계; 및

상기 유도 그래프를 이용하여 서브 그래프를 탐색하는 단계;

를 포함하는 것을 특징으로 하는 서브 그래프 리스팅 방법.

청구항 2

삭제

청구항 3

삭제

청구항 4

삭제

청구항 5

삭제

청구항 6

제 1 항에 있어서,

상기 제 1 정점 외의 정점은 항상 제 1 정점과 연결되도록 상기 제2 정점 및 제3 정점을 분류하는 것을 특징으로 하는 서브 그래프 리스팅 방법.

청구항 7

삭제

청구항 8

제 1 항에 있어서,

상기 제 1 정점의 인접 리스트를 이용하여 상기 제 2 정점 및 상기 제 3 정점을 매핑하는 것을 특징으로 하는 서브 그래프 리스팅 방법.

청구항 9

제 1 항에 있어서,

상기 제 1 정점의 매칭 순서가 상기 제 2 정점 및 상기 제 3 정점보다 반드시 앞서야 하는 것을 특징으로 하는 서브 그래프 리스팅 방법.

발명의 설명

기 술 분 야

[0001] 본 발명은 서브 그래프 리스팅 방법에 관한 것이다.

배 경 기 술

[0002] 서브 그래프 리스팅은 그래프 및 네트워크 분석을 위한 기초적인 연산으로서 네트워크 분석과 데이터 마이닝 분야에서 폭넓게 활용되는 것으로서, 서브 그래프 리스팅은 질의 그래프가 주어졌을 때 대규모 데이터 그래프로부터 해당 질의 그래프와 동형인 모든 서브 그래프를 나열하는 것이다.

[0003] 대규모 데이터 그래프를 처리 하는 방법으로는 크게 데이터 그래프를 여러 대의 머신에 분해하여 처리하는 분산 처리 방법과 데이터 그래프를 단일 머신의 디스크에 저장하는 디스크 기반 방법이 있다.

[0004] 분산 처리 방법은 그래프 파티셔닝 알고리즘을 사용하여 데이터 그래프를 머신 수만큼으로 분해하여 각 머신에 분산하여 저장한 후, 각 머신에서 분해된 그래프를 자신의 메인 메모리에 상주시킨 후에, 그래프 처리를 수행한다.

[0005] 그리고, 디스크 기반 방법은 단일 머신의 제한된 메모리를 최대한 활용하여 디스크 액세스 횟수를 최소화하는 방법으로 그래프 처리를 수행한다.

[0006] 분산 처리에 기반하는 서브 그래프 리스팅 방법으로 PSgL 방법이 제시되었다.

[0007] PSgL 방법은 그래프 탐색을 통해 질의 그래프의 각 정점을 데이터 서브 그래프의 각 정점과 차례로 매핑하는 방식으로 서브 그래프를 찾는다. 이 방법은 질의의 서브 그래프와는 매치되지만 최종 질의 결과에는 포함되지는 않는 다수의 중간 결과를 생성하는 조인 연산 기반의 방법보다 상대적으로 우수한 성능을 보인다.

[0008] 그러나, PSgL 방법은 질의 처리를 위해 대용량의 메모리를 필요로 하는 문제를 가진다. PSgL 방법은 데이터 그래프에서 너비 우선 탐색을 수행하며 질의 그래프와 데이터 그래프의 서브 그래프 사이에서 매핑을 진행하는데, 매핑의 중간 결과를 메모리에 모두 저장해야만 하고, 이에 따라 중간 결과의 크기가 데이터 그래프보다 훨씬 커질 수 있어, 질의 그래프와 부분적으로 매치되는 데이터 서브 그래프가 많으면 메모리 초과 현상이 발생한다.

[0009] 이를 위해, 단일 머신에서 소규모의 그래프 데이터를 처리하는 메모리 기반 서브 그래프 매칭 알고리즘이 고안되었다.

[0010] 그러나 이 방법을 디스크 기반 환경에 그대로 적용하는 경우 많은 수의 임의 디스크 액세스를 수반하게 되며, 디스크 액세스는 메모리 액세스에 비해 단위 액세스 당 시간이 많이 걸리므로, 많은 수의 디스크 액세스가 발생하면 성능이 크게 저하된다.

[0011] 이에 따라, 디스크 기반 환경에서 삼각형 형태의 제한적인 질의 그래프에 대해서 효율적인 알고리즘이 제안되었다.

[0012] 이 알고리즘에서는 삼각형의 세 정점 중 하나에 대응되는 데이터 정점들은 디스크로부터 액세스 하지 않고, 다른 두 정점에 대응되는 데이터 정점들의 인접 리스트의 교집합으로부터 구하는 방법을 이용함으로써, 디스크 액세스를 줄인다.

[0013] 그러나, 위 알고리즘을 질의 그래프가 삼각형이 아닌 임의의 그래프로 주어졌을 때 서브그래프를 찾을 수 있도록 확장하는 것은 쉽지 않다.

[0014] 그에 따라, 임의의 질의 그래프가 주어졌을 때, 대규모의 데이터 그래프로부터 디스크 액세스를 줄이면서 서브 그래프 리스팅을 효율적으로 처리하기 위한 새로운 방안이 요구된다.

발명의 내용

해결하려는 과제

[0015] 상기와 같은 종래 기술의 문제점을 해결하기 위해, 본 발명은 임의의 질의 그래프에 대해 디스크 액세스를 최소화하는 서브 그래프 리스팅 방법을 제공하고자 한다.

과제의 해결 수단

[0016] 위와 같은 과제를 해결하기 위한 본 발명의 일 측면에 따르면, 데이터 그래프에서 질의 그래프에 대한 서브 그래프 리스팅 방법에 있어서, 질의 그래프의 정점들을 메모리에 로드된 데이터 그래프의 정점에 직접 매핑(Retrieval) 해야 하는 제 1 정점, 두 개 이상의 상기 제 1 정점의 인접 정보의 교집합을 통해 매핑 가능한 제 2 정점, 및 하나의 상기 제 1 정점의 인접 정보를 이용하여 매핑 가능한 제 3 정점으로 질의 그래프 정점을 분류하는 단계; 상기 제 1 정점 및 제 1 정점의 간선으로 구성된 유도 그래프를 생성하는 단계; 및 상기 유도 그래프를 이용하여 서브 그래프를 탐색하는 단계;를 포함하는 것을 특징으로 하는 서브 그래프 리스팅 방법이 제공된다.

[0017] 상기 서브 그래프 리스팅을 수행하는 시스템의 메모리 버퍼가 내부 구역, 피벗 구역, 및 외부 구역으로 분할되어 있는 것을 특징으로 할 수 있다.

[0018] 상기 외부 구역의 개수는 상기 제 1 정점의 개수 - 2 인 것을 특징으로 할 수 있다.

[0019] 상기 피벗 구역과 상기 외부 구역은 상기 내부 구역에 위치하지 않은 정점을 로드하는 구역인 것을 특징으로 할 수 있다.

[0020] 상기 내부 구역의 정점에 인접한 정점은 상기 피벗 구역에, 상기 피벗 구역 혹은 상기 내부 구역과 연결된 정점은 외부 구역에 차례대로 로드되는 것을 특징으로 할 수 있다.

[0021] 상기 제 1 정점 외의 정점은 항상 제 1 정점과 연결되도록 상기 제2 정점 및 제3 정점을 분류하는 것을 특징으로 할 수 있다.

[0022] 상기 서브 그래프는 상기 내부 구역에 질의 그래프의 제 1 정점과 대응되는 모든 정점이 로드되어 있는지 여부에 따라 내부 서브 그래프와 외부 서브 그래프로 구별되는 것을 특징으로 할 수 있다.

[0023] 상기 제 1 정점의 인접 리스트를 이용하여 상기 제 2 정점 및 상기 제 3 정점을 매핑하는 것을 특징으로 할 수 있다.

[0024] 상기 제 1 정점의 매칭 순서가 상기 제 2 정점 및 상기 제 3 정점보다 반드시 앞서야 하는 것을 특징으로 할 수 있다.

발명의 효과

[0025] 본 발명의 서브 그래프 리스팅 방법에 의하면 데이터 그래프가 큰 경우라도, 디스크 액세스 횟수를 최소화하여 데이터 그래프의 질의 그래프에 대한 서브 그래프 리스팅을 효율적으로 수행할 수 있다.

[0026] 본 발명의 서브 그래프 리스팅 방법에 의하면, 질의 그래프를 RBI 그래프라 불리는 유도 그래프로 변환하고, 질의 그래프 대신 유도 그래프를 사용하여, 서브 그래프를 리스팅 한다. 이 방법은 대규모의 데이터 그래프로부터 서브 그래프 리스팅을 구할 때 유도 그래프의 정점에 대응하는 데이터 정점과 인접 정보만을 디스크로부터 읽어들인다. 따라서 디스크의 I/O 횟수를 대폭 감소 시킬 수 있다.

[0027] 본 발명의 서브 그래프 리스팅 방법은 메모리 버퍼를 내부, 피벗 및 외부 구역으로 구별하고, 내부 서브 그래프 리스팅 및 피벗 구역과 외부 구역의 서브 그래프 리스팅을 멀티 스레드를 기반으로 처리하여 수행 속도를 크게 개선할 수 있다.

[0028] 또한, 본 발명의 서브 그래프 리스팅 방법은 깊이 우선 탐색을 기반으로 RBI 그래프를 매핑하기 때문에, 중간 결과를 저장하지 않아 저장 공간의 오버헤드 없이 효과적으로 서브 그래프를 리스팅 할 수 있다.

도면의 간단한 설명

[0029] 도 1은 본 발명의 한 실시예에 따른 서브 그래프 리스팅 방법을 개략적으로 보여준다.

도 2는 본 발명의 한 실시예에 따른 질의 그래프 및 RBI 그래프를 나타낸다.

도 3은 본 발명의 한 실시예에 따른 메모리 버퍼 분할 방식을 나타낸다.

도 4는 본 발명의 한 실시예에 따른 디스크 기반 서브 그래프 리스팅에서의 서브 그래프를 나타낸다.

도 5는 본 발명의 한 실시예에 따른 오각형 모양의 데이터 그래프 및 사각형 모양의 질의 그래프의 서브그래프 리스팅을 수행했을 때 나타나는 깊이 우선 탐색 호출 그래프를 나타낸다.

도 6은 본 발명의 한 실시예에 따른 데이터 그래프 상의 정점 방문 순서를 나타낸다.

도 7은 본 발명의 한 실시예에 따른 질의 그래프 및 질의 그래프의 부분 순서 집합을 나타낸다.

도 8는 본 발명의 한 실시예에 따른 서브그래프 리스팅 방법을 리얼 데이터셋에 대하여 수행한 시간을 나타낸다.

도 9는 본 발명의 한 실시예에 따른 서브그래프 리스팅 방법을 리얼 데이터셋에 대하여 수행시 디스크 I/O 횟수를 나타낸다.

발명을 실시하기 위한 구체적인 내용

[0030] 본 발명의 서브 그래프 리스팅 방법은 질의 그래프의 정점들을 메모리에 로드된 데이터 그래프의 정점에 직접 매핑(Retrieval) 해야 하는 제 1 정점, 제 1 정점들의 인접 정보를 교집합(Intersection)하여 매핑 가능한 제 2 정점, 및 제 1 정점의 인접 정보를 바로 읽어(Browsing) 매핑 가능한 제 3 정점으로 분류하여 제 2 정점 및 제 3 정점에 대하여 디스크를 액세스 하지 않아도 서브 그래프 리스트를 수행할 수 있어, 디스크 액세스를 최소화 할 수 있다.

[0031] 본 발명의 서브 그래프 리스팅 방법은 질의 그래프를 상기 제 1 정점, 제 2 정점 및 제 3 정점으로 구별하고, 제 1 정점과 제 1 정점의 간선으로 이루어진 그래프를 유도 그래프를 이용하여 서브 그래프를 탐색한다.

[0032] 서브 그래프 탐색 방법으로는 디스크 페이지 블럭 단위의 깊이 우선 탐색 기법을 도입하였다. 깊이 우선 탐색 방법은 정점 단위의 깊이 우선 탐색 기법에 비해 디스크에 대한 랜덤 액세스 회수가 줄어 효율적인 디스크 I/O가 가능하다.

[0033] 이하, 첨부한 도면을 참고로 하여 본 발명의 실시예에 대하여 본 발명이 속하는 기술분야에서 통상의 지식을 가진 자가 용이하게 실시할 수 있도록 상세히 설명한다. 본 발명은 여러 가지 상이한 형태로 구현될 수 있으며 여기에서 설명하는 실시예에 한정되지 않는다. 도면에서 본 발명을 명확하게 설명하기 위해서 설명과 관계없는 부분은 생략하였으며, 명세서 전체를 통하여 동일 또는 유사한 구성요소에 대해서는 동일한 참조부호를 붙였다.

[0034] 먼저, 본 발명에서 서브 그래프 리스팅은 데이터 그래프 $G_D = (V_D, E_D)$와 질의 그래프 $G_Q = (V_Q, E_Q)$의 두 그래프가 주어졌을 때, 데이터 그래프 G_D의 서브 그래프 $G_0 = (V_0, E_0) : V_0 \subseteq V_D,\ E_0 = E_D \cap (V_0 \times V_0)$ 가운데 $G_0 \cong G_Q$(본 발명에서 '동형'이라고 지칭하기도 한다)인 모든 G_0를 찾는 것이다.

[0035] 이때, $G \cong G'$ 는 다음과 같이 정의된다.

[0036] 두 그래프 $G = (V,E)$와 $G' = (V', E')$에서 V와 V'의 정점이 서로 일대일 대응이 가능하고, $v, u \in V$와 대응되는 정점 $v', u' \in V'$에 대해 $(v', u') \in E'$ 이면 $(v,u) \in E$ 이고 그 역도 성립하는 경우의 $G \cong G'$이다.

[0037] 이하, 도 1 내지 도 6을 참조하여, 본 발명의 서브 그래프 리스팅 방법(100)을 보다 상세히 설명하도록 한다.

[0038] 본 발명의 서브 그래프 리스팅 방법(100)은 도 1에 도시된 바와 같이, 질의 그래프를 RBI 그래프로 변환하는 단계(S101), 변환된 RBI 그래프에서 유도 그래프를 생성하는 단계(S102) 및 데이터 그래프에 대하여 유도 그래프를 이용하여 서브 그래프를 탐색을 수행하는 단계(S103)를 포함한다.

[0039] 먼저, 질의 그래프를 RBI 그래프로 변환한다(단계 S101).

[0040] 본 발명의 질의 그래프 변환 방법에 의하면, 서브 그래프 리스팅에서 디스크에 대한 직접 액세스 여부에 따라 질의 그래프 G_Q의 정점들을 다음과 같이 분류한다.

[0041] 즉, 질의 그래프 G_Q의 정점들을 데이터 정점으로 매핑하기 위해 디스크에서 직접 검색(Retrieval) 해야 하는 제 1 정점, 자신과 연결된 제 1 정점들에 매핑된 데이터 정점들의 인접 정보들을 교집합(Intersection)하여 매핑 가능한 제 2 정점, 및 자신과 연결된 제 1 정점의 인접 정보를 바로 읽어(Browsing) 매핑 가능한 제 3 정점으로 분류한다.

[0042] 본 실시예에서는 제 1 정점을 R (Red; 레드) 정점으로, 제 2 정점을 B (Black; 블랙) 정점, 그리고 제 3 정점을 I (Ivory; 아이보리) 정점으로 정의한다.

[0043] 그에 따라, 질의 그래프 G_Q 내 모든 정점은 R 정점, B 정점, 및 I 정점으로 분류되며, 이와 같이 질의 그래프 G_Q의 정점들을 R 정점, B 정점, 및 I 정점으로 분류하여 매칭하는 것을 RBI-Match라 정의한다.

[0044] 이러한 RBI-Match에 따라, 질의 그래프의 모든 정점은 R 정점, B 정점, 및 I 정점으로 표현되고, 이와 같이 R 정점, B 정점, 및 I 정점으로 표현된 질의 그래프를 RBI 그래프로 지칭한다.

[0045] 질의 그래프 G_Q의 크기가 데이터 그래프 G_D의 크기보다 현저히 작고, 질의 그래프 G_Q를 RBI 그래프로 변환하는 과정은 간단한 알고리즘으로 구성되므로 질의 그래프 G_Q의 RBI 그래프로의 변환 시간이 전체 서브 그래프 리스팅 수행 시간에 미치는 악영향은 미미하다.

[0046] 이하, 도 2를 참조하여, 질의 그래프 G_Q를 RBI 그래프로 변환하는 방법을 실시예로 들어 설명한다.

[0047] 먼저, 도 2(a)의 질의 그래프 G_Q의 정점들 중에서 R 정점, 즉 V_{red}를 결정한다.

[0048] 이때, R 정점 이외의 정점은 항상 R 정점과 인접하고, R 정점이 아닌 두 정점을 잇는 간선은 없도록 질의 그래프 G_Q의 정점들 중에서 V_{red}를 결정한다.

[0049] 그리고, V_{red}가 결정되면 V_{red}를 제외한 정점에 대해 인접한 R 정점의 개수가 2개 이상인 경우 I 정점, 1개인 경우로 B 정점으로 분류한다.

[0050] 여기서, V_{red}는 MCVC(최소 연결 정점 덮개; Minimum Connected Vertex Cover)로 선택된다. MCVC에 인접한 질의 정점은 B 정점 혹은 I 정점이며, 이들을 데이터 정점에 매핑할 때는 디스크 액세스가 불필요하므로, 불필요한 디스크 액세스를 줄일 수 있다.

[0051] 질의 정점 집합의 MCVC가 V_{red}가 되는 이유는 아래와 같다. Red 정점 이외의 정점은 항상 Red 정점과 연결되고, Red 정점이 아닌 두 정점을 잇는 간선은 없어야 한다. 이는 정점 덮개(vertex cover)의 정의와 같게 되어 V_{red}는 정점 덮개가 된다. 즉, G_{red}가 연결 그래프임을 보장하면서 $|V_{red}|$ 를 최소화할 수 있는 정점 집합은 MCVC이다. 따라서, V_{red}를 MCVC로 선택한다.

[0052] 그에 따라, 도 2 (b)에 도시된 바와 같이, 질의 그래프 G_Q의 정점들 중 u_1, u_2, u_3는 R 정점, u_4, u_5는 I 정점, u_6는 B 정점이 된다.

[0053] 즉, R 정점인 u_1, u_2, u_3의 인접 리스트 사이의 교집합 연산으로부터 구해질 수 있는 u_4, u_5 정점은 I 정점이 되며, R 정점인 u_3와 매핑된 데이터 정점의 인접 리스트를 즉시 이용할 수 있는 u_6는 B 정점이 되는 것이다.

[0054] 이와 같이 본 발명에 따른 서브 그래프 리스팅 방법에 의하면, 질의 그래프와 동형인 서브 그래프는 먼저 R 정점을 결정하고, R 정점이 결정되면 I 정점 또는 B 정점들과 매핑되는 데이터 정점 정보들은 R 정점과 매핑된 데이터 정점의 인접 정보를 통해서 얻을 수 있게 된다. 그에 따라, R 정점과 매핑되는 데이터 그래프 정점만을 방문하고, I 정점 또는 B 정점과 매핑되는 데이터 그래프 정점은 방문할 필요가 없게 되어 디스크 액세스 횟수를 감소 시킬 수 있다.

[0055] 다음으로, 변환된 RBI 그래프에서 유도 그래프를 생성한다(단계S102).

[0056] 즉, RBI 그래프에서 V_{red} 정점으로 이루어진 유도 그래프(induced graph) $G_{red}=(V_{red}; E_{red})$를 생성한다. 이때, G_{red}는 V_{red} 정점과 V_{red} 정점 간의 간선(Edge) E_{red}로 이루어진다.

[0057] 질의 그래프 G_Q를 RBI 그래프로 변환하는 알고리즘을 표로 정리하면 아래 알고리즘 1과 같게 된다.

[0058] [알고리즘 1]

Algorithm 1. REWRITERBIQUERYGRAPH

Input: query graph $G_Q(V_Q, E_Q)$
Result: $G_{RBI}(V_{RBI}, V_{red}, V_{black}, V_{ivory}, E_{RBI})$, $G_{red}(V_R, E_R)$
1: $G_{RBI}(V_{RBI}, E_{RBI}) \leftarrow G_Q(V_Q, E_Q)$
2: **for** $(i \leftarrow 1$ to $|V_Q|)$ **do**
3: | **if** (size i Connected Vertex Cover(CVC) exists) **then** $MCVC \leftarrow CVC$ and then
| break;
4: **end**
5: $G_{RBI}(V_{red}) \leftarrow MCVC$
6: $G_{red} \leftarrow$ Induced Graph of G_Q, which $V_R = MCVC$
7: **for** $(i \leftarrow 1$ to $|V_Q|)$ **do**
8: | **if** $(u_i \notin G_{RBI}(V_{red})$ && $|n(u_i) > 1|)$ **then** $G_{RBI}(V_{ivory}) \leftarrow G_{RBI}(V_{ivory}) \cup u_i$;
9: | **if** $(u_i \notin G_{RBI}(V_{red})$ && $|n(u_i) = 1|)$ **then** $G_{RBI}(V_{black}) \leftarrow G_{RBI}(V_{black}) \cup u_i$;
10: **end**

[0059]

[0060] 알고리즘 1에 도시된 바와 같이, 질의 그래프 G_Q는 RBI 그래프 G_{RBI}와 유도 그래프 G_{red}로 변환된다.

[0061] 알고리즘 1에 도시된 1 내지 10 단계는 질의 그래프 G_Q가 RBI 그래프와 유도 그래프 G_{red}로 변환되는 순서에 대한 알고리즘을 보여준다.

[0062] 다음으로, RBI 그래프를 이용해 데이터 그래프에서 서브 그래프를 탐색한다(단계S103).

[0063] 본 발명에서, 서브 그래프 리스팅은 내부 서브 그래프 리스팅과 외부 서브 그래프 리스팅으로 구별되어 이루어진다.

[0064] 본 발명의 서브 그래프 리스팅을 설명하기 앞서, 먼저 내부 서브 그래프 리스팅과 외부 서브 그래프 리스팅을 설명한다.

[0065] 대부분의 경우, 데이터 그래프 크기가 메인 메모리 용량을 초과하기 때문에, 메모리에 전체 데이터 그래프를 로드할 수 없고, 따라서, 디스크의 데이터 그래프를 여러 개의 청크로 나눌 필요가 있다. 하나의 청크의 크기는 메모리 버퍼의 크기보다 작다. 이 때 찾고자 하는 서브 그래프의 모든 간선 정보가 한 청크 안에 포함되어 있는 경우 이를 내부 서브 그래프라 하고, 특정 서브 그래프의 간선 정보가 여러 청크에 나누어져 있을 경우 이를 외부 서브 그래프(external subgraph)라고 한다.

[0066] 도 2에 도시된 실시예를 참조하여 서브 그래프 탐색을 설명하면, 가장 먼저 질의 그래프의 u_1과 데이터 그래프의 데이터 정점, 예를 들어 v_1을 매핑하고, 다음에는 질의 그래프의 u_1과 인접한 u_2에 매칭을 시도한다.

[0067] 이때, v_1의 이웃한 정점 집합인 $n(v_1)$이 u_2와 매핑 가능한 후보 집합이 된다. 그리고, 질의 그래프의 u_2에 대응하는 데이터 그래프의 데이터 정점, 예를 들어 v_2가 매핑되면, 질의 그래프의 u_2와 인접한 u_3의 매칭을 시도한다. 그리고, 이러한 방법으로 모든 유도 그래프의 정점이 매칭되면, 매핑된 데이터 정점 v를 이용하여 나머지 질의 그래프의 정점 u_4, u_5, u_6에 대한 데이터 정점에 대한 정보를 수득하는 방법으로 서브 그래프를 탐색하는 것이다.

[0068] 이와 같이, 질의 그래프 G_Q 내에서 R 정점 사이의 간선 관계를 따라 정점 매칭이 순차적으로 진행되므로 유도 그래프 G_{red}는 반드시 연결 그래프여야 한다.

[0069] 만약 G_{red}가 연결 그래프가 아니고 R 정점 사이의 간선 관계를 파악할 수 없다면 V_{red}에 속하는 정점들의 매핑이 독립적으로 이루어지게 된다. 이 경우 질의 정점 사이의 간선 정보가 없기 때문에 질의 정점에 대응하는 후보 데이터 정점들이 저장된 디스크 페이지의 모든 조합을 고려해야 하므로 많은 비용이 소모된다.

[0070] 본 발명에서 서브 그래프 리스팅 방법은 내부 서브 그래프 리스팅(internal subgraph listing)과 외부 서브 그래프 리스팅(external subgraph listing)으로 나누어 처리한다.

[0071] 이하, 도 3을 참조하여 본 실시예에서, 유도 그래프 G_{red}를 이용하여 데이터 그래프에서 서브 그래프를 탐색하는 방법을 설명한다.

[0072] 본 발명의 서브 그래프 리스팅 방법에 의하면, 바람직하게는 메모리 버퍼 MEM을 내부 구역 MEM_{int}, 피벗 구역 MEM_{pivot}, 외부 구역으로 분할한다. 이때, 각 구역은 정점의 집합으로 정의되는데 실제로는 데이터를 디스크에서 메모리 버퍼로 페이지 단위로 로드하므로 구역은 인접 리스트가 저장되어 있는 디스크 페이지의 집합으로 정의될 수 있다. 각 구역의 크기는 적어도 1개 이상의 인접 리스트를 포함할 수 있어야 한다.

[0073] 도 3에 도시된 바와 같이, 내부 구역 MEM_{int} 은 내부 서브 그래프가 정의되는 구역으로 내부 서브 그래프 리스팅은 내부 구역에 로드된 정점의 인접 리스트만으로 수행된다.

[0074] 피벗 구역 MEM_{pivot} 은 내부 구역의 정점에 인접한 정점을 적어도 하나는 포함하는 페이지들이 위치하는 공간이다. 외부 서브 그래프 리스팅을 수행할 때 메모리 내부 구역의 정점과 인접한 정점 중 내부 구역에 위치하지 않은 정점을 추가적으로 로드하게 되는데, 이렇게 로드 된 정점의 집합은 피벗 구역에 위치하게 된다.

[0075] 외부 구역에는 메모리 버퍼 내의 다른 구역의 정점과 인접한 정점 중 해당 구역에 포함되지 않는 정점들이 위치한다.

[0076] 데이터 그래프 전체에 대하여, 본 발명에서 외부 서브 그래프를 빠짐없이 찾기 위해서는 $|V_{red}|$ 개의 메모리 구역이 필요하다. 왜냐하면, V_{red} 와 대응되는 데이터 그래프의 정점 집합 V_0 의 정점이 모두 서로 다른 디스크 페이지에 저장되어 있는 경우 $|V_0|$ 개의 페이지가 메모리에 로드되어야 하고, 서로 다른 구역에 위치할 수 있기 때문에 $|V_{red}|$ 개의 메모리 구역이 필요한 경우가 존재하기 때문이다.

[0077] 본 실시예의 도 3에서는 내부 구역, 피벗 구역과 $|V_{red}|-2$ 개의 외부 구역으로 구별하여 도시되었다.

[0078] 본 실시예에서, 유도 그래프 $G_{red}=(V_{red},E_{red})$ 상의 그래프 탐색을 통하여 외부 서브 그래프 리스팅을 수행하는 과정을 설명한다.

[0079] 외부 서브 그래프 리스팅은 V_{red} 에 대응하는 데이터 그래프의 정점들 중 내부 구역에 로드된 정점 v를 확인하는 것으로 시작한다.

[0080] 그 후, 현재 내부 구역에 로드된 정점 v에 대하여 인접 리스트를 스캔한다. 이때, 내부 구역 MEM_{int}에 로드된 정점 v에 인접한 정점 $n(v) \notin MEM_{int}$ 를 모아 피벗 후보 정점 집합이라 한다.

[0081] 다음으로, 피벗 구역 MEM_{pivot}에 피벗 후보 정점 집합에 속하는 정점을 하나 이상 포함하는 페이지들 중에 페이지 식별자가 작은 것부터 순서대로 페이지를 로드한다.

[0082] 본 발명에서는 페이지 식별자가 작은 것부터 순서대로 페이지를 로드함으로써, 임의 순서로 페이지를 로드할 때에 비해 디스크 접근 비용을 줄인다.

[0083] 도 3에서 $\{v_3,v_4,v_6\}$는 페이지 1과 2가 내부 구역에 로드되었을 때의 피벗 후보 정점 집합이며, 따라서 피벗 구역 MEM_{pivot}에는 $\{v_3,v_4\}$를 포함하는 페이지 3이 로드된다.

[0084] 피벗 구역 MEM_{pivot}에 페이지가 로드 된 다음에는, 질의 그래프의 모양에 따라 내부 영역에 있는 정점과 인접한 정점의 집합을 구하거나 혹은 피벗 후보 정점 집합에 속한 정점 중 MEM_{pivot}에 로드된 정점과 인접한 정점의 집합을 구한다. 이 정점 집합은 외부 후보 정점 집합이라 한다.

[0085] 외부 구역에는 적어도 하나 이상의 외부 후보 정점이 속한 페이지를 로드한다. 도 3은, 내부 구역과 연결된 정

점은 피벗 구역에, 피벗 구역과 연결된 정점은 외부 구역에 차례대로 로드되는 것을 보여준다.

[0086] 이와 같이, 본 실시예에서 이러한 정점 후보 집합은 메모리에 로드된 정점들의 인접 리스트를 스캔하는 과정을 통해 만들어진다.

[0087] 한편 본 발명의 서브 그래프 리스팅 방법에서 바람직하게는 질의 결과에 자가 동형 서브 그래프가 포함되지 않도록 질의 그래프 G_Q 의 정점의 매핑 순서를 결정하는 부분 순서(partial order)를 활용한다.

[0088] 부분 순서는 사용자로부터 입력되며, 이 순서에 따라 질의 정점을 매핑함으로써 질의 결과에 포함되지 않는 자가 동형(automorphism)의 서브 그래프를 찾는 중복을 방지할 수 있고, 전체 서브 그래프 리스트의 결과를 가능한 σ 의 개수만큼 감소시킬 수 있다.

[0089] 여기서 σ 는 정점 집합의 순열인데, 질의 그래프 $G_Q = (V_Q, E_Q)$ 의 정점 집합 V_Q의 임의의 두 정점 u, v 가 간선 (u, v)를 형성한다면 반드시 $(\sigma(u), \sigma(v))$가 간선을 형성하고, 그 역도 반드시 성립한다.

[0090] 그에 따라, 질의 그래프의 자가동형 G_a은 이러한 순열 σ 를 통해 변환된 정점과 간선으로 정의된다.

[0091] 예를 들면, G_Q가 정점이 3개인 삼각형이라고 할 때 3 × 2 = 6 즉, 총 6개의 자가동형이 존재하게 되며, 따라서 가능한 σ 는 6개가 존재한다.

[0092] 부분 순서는 특정 두 질의 정점 u_1, u_2 에 대해 데이터 정점으로의 매핑 M이 주어졌을 때 $vid(M[u_1]) < vid(M[u_2])$ 을 만족해야 한다는 형태의 조건으로 주어지며, 가능한 σ 중 하나만 유효하도록 몇몇 질의 정점들간에 부분 순서를 줌으로써 자가 동형의 서브 그래프를 찾는 중복을 방지한다.

[0093] 본 발명에서는 부분 순서를 만족하면서 질의 그래프에 대응하는 데이터 서브 그래프를 인접한 정점을 따라 순회함으로써, 자가 동형의 서브 그래프를 찾는 중복을 방지한다.

[0094] 본 발명의 서브 그래프 리스팅 방법에서 내부 서브 그래프 리스팅은 메모리의 내부 구역 크기만큼의 연속된 디스크 페이지를 로드한 후 해당 페이지 청크 내에서 내부 서브 그래프를 찾는다.

[0095] 이하, 본 발명에 따른 내부 서브 그래프 리스팅 알고리즘을 설명한다.

[0096] 먼저 질의 그래프의 구조로부터 입력으로 주어지는 부분 순서를 고려하여 바람직한 질의 그래프 정점의 매칭 순서를 정한다.

[0097] 이때, V_{red} 의 인접 리스트를 이용해 나머지 정점을 매핑하기 때문에 매칭 순서에서 V_{red}가 V_{black}, V_{ivory}보다 반드시 앞서야 한다.

[0098] 정해진 매칭 순서에 따라 메모리 내부 구역에서 V_{red} 에 속하는 질의 정점의 매핑을 메모리 내부 구역에서 형성한 후, V_{red} 와 매핑된 정점의 인접 리스트들을 이용하여 V_{black}, V_{ivory} 를 매핑하여 서브 그래프를 리스팅한다.

[0099] V_{red} 에 속하는 질의 정점의 매핑은 질의 정점의 매핑 순서를 고정한 다음 매핑된 데이터 정점과 연결된 다른 데이터 정점을 따라가면서 재귀적으로 형성한다.

[0100] 그리고, 본 발명의 서브 그래프 리스팅 방법에서, 외부 서브 그래프 리스팅을 수행한다. 이때 외부 서브 그래프 탐색은 내부 구역에 로드된 정점 하나 이상을 반드시 포함하고, 내부 구역 외부의 정점 또한 하나 이상 반드시 포함하는 경우를 탐색한다.

[0101] 그리고, 현재 메모리 내의 내부 구역에서 내부 및 외부 서브 그래프 리스팅을 완료하면, 다음 디스크 페이지 청크를 로드한다. 이러한 내부 서브 그래프 리스팅 및 외부 서브 그래프 리스팅 과정은 데이터 그래프 전체를 구성하는 디스크 페이지 개수 P(G)를 내부 구역을 구성하는 페이지의 개수인 $P(MEM_{int})$으로 나눈 $\left\lceil \frac{P(G)}{P(MEM_{int})} \right\rceil$ 횟

수만큼 반복하게 된다.

[0102] 본 발명의 서브 그래프 탐색에 대한 전체 프로세스는 알고리즘 2와 같다.

[0103] 알고리즘 2

Algorithm 2. RBI-MATCH

```
Input: query graph G_Q, partial orders PO
⟨G_RBI, G_red⟩ ← REWRITERBIQUERYGRAPH(G_Q);
initialize the bitmap sets in CandidateTree
while ( there are more chunks to process) do
    C_cur ← CACULATENEXTCHUNKRANGE(m_int);
    PINPAGESANDFILLCANDINFO(C_cur, −1);        /* −1 : Current Index on G_red */
    DELEGATEEXTSUBGRAPHLISTING(G_RBI, G_red, C_cur, CT, PC);  /* ext. listing */
    INTSUBGRAPHLISTING(G_RBI, G_red, C_cur);                  /* int. listing */
    pool.Sync();
    UNPINPAGES(C_cur);
end
Function PINPAGESANDFILLCANDINFO
    Input: C_cur, t_last : Current Index on G_red
    foreach ( pid ∈ P(C_cur)) do
        AsyncRead(pid, UPDATECANDIDATES(pid, CT, t_last));
    end
    wait until all invocations of UPDATECANDIDATES executions are finished
end
```

[0104]

[0105] 알고리즘 2에 도시된 바와 같이, 먼저, 초기화 과정에서는 입력 받은 질의 그래프 G_Q를 RBI 그래프로 변환하고 외부 서브 그래프 리스팅에서 이용하는 자료구조를 초기화한다.

[0106] 그 다음, 내부구역의 크기만큼 데이터 그래프에서 청크를 로드한 뒤, 외부 서브 그래프 리스팅과 내부 서브 그래프 리스팅을 병렬로 수행한 뒤, 두 작업이 모두 끝나면 청크를 메모리에서 해제한다.

[0107] 본 알고리즘은 데이터 그래프에서 더 이상 로드할 청크가 없을 때까지 반복적으로 위 과정을 수행한다.

[0108] 본 발명에서 외부 서브 그래프 리스팅은 RBI 그래프의 V_{red}의 정점과 매핑할 데이터 그래프의 정점 집합 V_0의 인접 리스트를 메모리로 로드하는 단계와 메모리에 위치한 인접 리스트를 이용하여 V_0를 포함한 서브 그래프 중 RBI 그래프와 동형인 서브 그래프를 찾는 두 단계로 이루어진다.

[0109] 이하, 데이터 그래프는 도 5(a)와 같고, 데이터 그래프의 일부인 v_1, v_2, v_3만 메모리 버퍼에 위치하고 있는 경우를 본 발명의 실시예로 하여, 서브 그래프 리스팅 방법을 도 4를 참조하여 설명한다.

[0110] 이때, 도 5(b)와 동형인 서브 그래프는 $\square^{v_1v_2v_3v_4}$, $\square^{v_1v_2v_3v_5}$, $\square^{v_1v_2v_4v_5}$, $\square^{v_1v_3v_4v_5}$, $\square^{v_2v_3v_4v_5}$의 5개가 된다.

[0111] 그런데, 서브 그래프 중 $\square^{v_1v_2v_3v_4}$, $\square^{v_1v_2v_3v_5}$ 는 도 4에 도시된 바와 같이, 메모리 버퍼에 정점 v_1, v_2, v_3 이 있으므로, 메모리 버퍼에 있는 간선 정보만을 사용하여 탐색될 수 있다. 즉, 메모리 버퍼 내의 v_1, v_2, v_3정점의 인접 리스트 만으로 질의 그래프와 동형인지 판단할 수 있다.

[0112] 본 발명의 외부 서브 그래프 리스팅은 가능한 모든 페이지의 조합을 조사하지 않고, 이미 메모리 로드된 페이지에 저장된 정점 v_1, v_2, v_3 들의 인접 리스트를 스캔하여 후보 정점 집합을 구한 뒤에, 후보 정점 집합에 속하는 정점을 하나 이상 포함하고 있는 페이지만 로드하는 방식을 이용한다.

[0113] R 정점 집합이 V_{red}={ u_1, u_2, u_3} 일 경우, 호출 그래프(call graph)는 도 5의 u_4에 대응되는 마지막 단계를 없앤 깊이 3의 그래프로 표현된다.

[0114] 따라서, 앞서 가정한 서브 그래프는 메모리 버퍼가 3개 크기이고, LRU(Least Recently Used) 방식으로 페이지 교체가 이루어 짐에 따라 해당 서브 그래프는 정점 방문 9번 중에 3번의 페이지 폴트가 발생한다.

[0115] 본 발명에서는 페이지의 후보 집합을 만든 뒤 디스크에 디스크 페이지 여러 개에 대한 비동기적 I/O를 한 번에 요청한다. 따라서 콜백 함수의 연산이 수행되는 동시에 디스크에서는 I/O 작업을 진행할 수 있다.

[0116] 다음으로, 도 6을 참조하여 본 발명의 외부 서브 그래프 리스팅 방법으로 하나의 외부 서브 그래프를 탐색하는 과정을 설명한다.

[0117] 도 6 (a)에서 $V_{red} = \{u_1, u_2, u_3\}$이다. 첫 단계에서 RBI 그래프의 u_1과 메모리 내부 구역의 v_1이 매핑 후보로 지정된다.

[0118] 도 6 (b)는 질의 정점 u_2가 피벗 정점에 매핑 될 때, 본 발명의 한 실시예에 따른 데이터 그래프 상의 정점 방문 순서를 나타낸다.

[0119] 다음 단계에서는 V_{red}에 속하는 정점 중 u_1과 인접한 정점인 u_2의 매핑 후보를 찾는다. RBI 그래프가 간선 (u_1, u_2)을 포함하기 때문에 간선 (v_1, v_x)이 데이터 그래프에 존재하는 경우에만 정점 v_x가 u_2와 매핑될 수 있다. v_1의 인접 리스트가 이미 메모리에 로드 되어 있으므로 $n(v_1)$을 통해 v_x의 후보 집합을 얻을 수 있다.

[0120] 또한, 도 6에서 v_3, v_4가 u_2와 매핑될 수 있는 후보 정점이다. 따라서, 후보 정점 집합 $\{v_3, v_4\}$의 인접 리스트를 메모리에 로드한다. 즉, 도 6에서는 두 인접 리스트가 같은 페이지에 존재하므로 하나의 페이지만 읽어온다.

[0121] 마지막으로 u_3의 매핑 후보를 찾는다. 마찬가지로, 도 6에서 간선 (u_2, u_3)가 RBI 그래프에 존재하므로 $(v_3, v_x) \in E_d$ 혹은 $(v_4, v_y) \in E_d$를 만족하는 정점만이 u_3와 매핑될 수 있다.

[0122] 즉, $n(v_3)$와 $n(v_4)$를 이용하여 u_3에 해당하는 외부 후보 정점 집합을 찾는다. 예를 들면, v_1, v_2, v_5, v_7이 u_3와 매핑될 수 있는 후보 정점이고, v_1과 v_2가 속한 페이지 1은 이미 메모리에 존재하기 때문에 최종적으로 v_5가 속한 페이지 3, 그리고 v_7가 속한 페이지 4를 메모리로 로드 한다.

[0123] 이때, 도 6에 도시된 바와 같이, v_5와 v_7은 각기 다른 페이지에 존재하므로 각각의 페이지를 차례대로 읽는다.

[0124] 하나의 외부 구역의 크기가 페이지 2개 이상인 경우 두 정점을 동시에 로드하고 다음 과정인 매핑을 수행하나, 외부 구역의 크기가 페이지 2개보다 작을 경우 각 페이지를 하나씩 로드하여 매핑을 처리한 후에 언로드 하고 다음 페이지를 로드하여 매핑을 처리하는 과정을 반복 수행한다. 도 6에서는 외부 구역이 1개 페이지로만 구성되므로, 페이지 3을 먼저 로드하여 매핑을 완료한 다음 페이지 3을 메모리 버퍼에서 언로드 하고 페이지 3대신 페이지 4를 로드하여 매핑하면 된다.

[0125] 본 발명의 서브 그래프 리스팅은 페이지 블럭 단위로 깊이 우선 탐색을 진행하고, 하나의 페이지 블럭에 속한 정점은 한꺼번에 처리하므로, 데이터 정점 단위 깊이 우선 탐색이 유발할 수 있는 비효율적인 디스크 액세스를 방지한다. 또한 페이지 블록 단위 깊이 우선 탐색에서는 V_{red}의 현재 정점 후보를 메모리에 저장하는 방식을 사용하기 때문에 모든 부분 매핑을 메모리에 저장하는 방식을 사용할 때 발생할 수 있는 메모리 부족 현상이 본 발명에서는 일어나지 않는다.

[0126] 본 발명에서는 설명의 편의를 위해 첫 단계의 매핑 후보로 정점 하나를 지정하였지만, 이에 제한되지 않으며, 실제로는 페이지 블럭에 속한 정점 집합이 매핑 후보로 지정될 수 있다.

[0127] 즉, 내부 구역의 모든 정점이 G_{RBI}의 정점 $u \in V_{red}$의 매핑 후보가 되고, 내부 구역의 정점들과 연결된 모든 정점이 다음 매핑 후보가 된다.

[0128] 그리고 데이터 그래프를 이루는 페이지 중 매핑 후보에 속하는 데이터 정점을 적어도 하나 이상 포함하는 페이지를 MEM_{pivot}의 크기만큼 선택하여 메모리에 로드한 다음 단계를 진행한다.

[0129] 도 6에 이를 적용한 실시예를 보면 u_1의 매핑될 데이터 정점의 후보 집합은 $\{v_1, v_2\}$, u_2의 후보 집합은 $\{v_3, v_4, v_6\}$, u_3의 후보 집합은 $\{v_1, v_2, v_5, v_7\}$이 된다.

[0130] 한편, 필요한 모든 페이지 블럭들이 메모리에 로드되었을 때 정점 단위의 외부 서브그래프 리스팅을 수행함으로써 모든 외부 서브그래프들을 찾는다.

[0131] 정점 단위의 외부 서브그래프 리스팅은 내부 서브 그래프 리스팅과 유사한 과정으로 서브그래프를 찾으나 올바른 외부 서브그래프를 찾기 위해 다음과 같은 두 가지 추가 조건을 필요로 한다.

[0132] 첫 번째로 정점 단위의 외부 서브 그래프 리스팅은 각 질의 정점이 매핑된 데이터 정점이 해당 질의 정점의 후보 정점 집합에 속하는 서브그래프만 찾는다.

[0133] 두 번째로 정점 단위의 외부 서브 그래프 리스팅은 피벗 정점의 성질을 만족시키는 서브 그래프만 찾는다.

[0134] 이하, 알고리즘을 참조하여, 본 발명의 내부 서브 그래프 리스팅의 세부 알고리즘을 설명한다.

[0135] 먼저, 알고리즘 3을 참조하여, 본 실시예의 내부 서브 그래프 리스팅에 대하여 R 정점을 매핑할 때 이용하는 RedIntVertexMapping 함수와 내부 서브 그래프 리스팅과 외부 서브 그래프 리스팅에서 B 정점, I 정점을 매핑할 때 이용하는 함수인 NonRedVertexMatching 함수를 설명한다.

[0136] 알고리즘 3

```
   Input: current chunk range C_cur
   /* Step 1. Generate matching order of G_red                    */
1: qo ← GenISLMatchOrder(G_red);
   /* Step 2. Start matching                                      */
2: foreach (v in all data vertices in C_cur) do
3:     M[qo[1]] ← v;
4:     if (deg(v) < deg(qo[1])) then continue;
5:     RecIntSubgraphListing(C_cur, qo, 2);
6:     M[qo[1]] ← INVALID;
7: end
8: Function RecIntSubgraphListing
       Input: current chunk range C_cur, query matching order qo, depth d_INT
9:     if (d = |G_red|) then
10:        NonRedVertexMatching(qo);
11:        return;
12:    end
13:    u_cur ← qo[d_INT];
14:    U_CON ← n(u_cur) ∩ qo[1 : d_INT − 1];
15:    foreach (v in ⋂_{u∈U_CON}(N(M[u]))) do
16:        if (v is visited) then continue;
17:        if (deg(v) < deg(u_cur)) then continue;
18:        if (v.pid ∉ P(C_cur)) then continue;
19:        if (M[u_cur] violates partial order PO) then continue;
20:        M[u_cur] ← v;
21:        RecIntSubgraphListing(C_cur, qo, d_INT + 1);
22:        M[u_cur] ← INVALID;
23:    end
24: end
```

[0137]

[0138] 알고리즘 3 에서 볼 수 있는 바와 같이, 우선, GenISLMatchOrder 함수를 통해 질의 그래프 정점의 매핑 순서를 정한다. R 정점의 인접 리스트를 이용해 B 정점, I 정점과 데이터 그래프의 정점을 매핑하므로 R 정점의 매핑 순서가 B 정점, I 정점보다 반드시 앞서야 하며, R 정점에서는 그래프 매칭 순서를 따른다.

[0139] 다음으로, RecIntSubgraphListing 함수에서는 정해진 매칭 순서에 따라 V_{red}의 매핑을 재귀적으로 형성한다. 모든 R 정점의 매핑이 완료되면 NonRedVertexMatching 함수를 통해 V_{red}와 매핑된 정점의 인접 리스트들을 이용하여 V_{black}, V_{ivory}를 매핑한다.

[0140] 다음으로, 알고리즘 4를 참조하여, I 정점 및 B 정점을 매핑하는 함수를 설명한다.

[0141] 알고리즘 4

```
   Input: a red graph matching sequence qseq
1: C ← ∅;
2: for (i ← 1 to |V(Q_R)|) do
   |  /* from topology-data node mapping to query-data node mapping */
3: |  M'[qseq[i]] ← M[t_i];
4: end
5: RecNonRedVertexMatching(1, M');
6: Function RecNonRedVertexMatching
   |  Input: depth i, mapping M'
7: |  if (i >the number of non-red vertices) then
   |  |  /* full mapping */
8: |  |  report M';
9: |  |  return;
10: |  u_t ← i-th non-red query vertex;
11: |  if (u_t is a ivory query vertex) then
   |  |  /* an ivory query vertex has more than two neighborhoods */
12: |  |  N ← N(u_t);
13: |  |  foreach (data vertex v in ∩_{u∈N}(N(M'[u]))) do
14: |  |  |  if (v is already matched) then continue;
15: |  |  |  if (M'[u_t] = v violates partial order PO) then continue;
16: |  |  |  NonRedVertexMatching(i + 1, M');
17: |  |  end
18: |  else if (u_t is a black query vertex) then
   |  |  /* a black query vertex has only one red query vertex as a
   |  |     neighborhood */
19: |  |  n ← the first element in N(u_t);
20: |  |  foreach (data vertex v in N(M'[n])) do
21: |  |  |  if (v is already matched) then continue;
22: |  |  |  if (M'[u_t] = v violates partial order PO) then continue;
23: |  |  |  NonRedVertexMatching(i + 1, M');
24: |  |  end
25: |  end
26: end
```

[0142]

[0143] 서브 그래프 리스팅은 정해진 매칭 순서에 따라 정점을 매칭하는데 I 정점의 경우 2개 이상의 R 정점과 인접해 있으므로 R 정점들이 매칭된 데이터 그래프 정점의 인접 리스트들을 교집합 연산 함으로써 매핑을 구한다.

[0144] B 정점의 경우는 1개의 R 정점과 인접해 있으므로 인접 리스트를 한번 스캔함으로써 구할 수 있다.

[0145] 이때, 함수 RecNonRedVertexMatching은 불가능한 해를 피하기 위한 세부적인 조건을 포함할 수 있다.

[0146] 이하, 도 7 내지 도 9를 참조하여, 본 발명의 서브 그래프 리스팅 방법의 효과를 보여주는 실험(이하, 간단히 '본 실험'이라 지칭하기도 한다)에 사용한 실험 환경 및 실험 데이터셋에 대해 설명한다.

[0147] 도 7은 본 발명의 실시예에 따른 서브 그래프 리스팅 방법을 실험한 질의 그래프 및 질의 그래프의 부분 순서 집합을 설명한다.

[0148] 본 실험에서는, 도 7 (a) 에 도시된 바와 같은, QG1에서 QG9까지의 9가지 질의 그래프를 사용하였다.

[0149] 도 7 (b)는 각각의 질의 그래프에 매칭되는 데이터 서브 그래프의 자가 동형을 방지하기 위해 사용된 해당 질의 그래프의 부분 순서를 보여준다.

[0150] 본 실험에서는 다양한 데이터셋을 사용하였으며, 데이터셋들은 크게 리얼 데이터셋과 합성 데이터셋의 두 가지 종류의 그래프 데이터셋으로 구분될 수 있다.

[0151] 먼저, 리얼 데이터셋은 각종 웹 그래프, 소셜 네트워크 그래프를 포함하고 있는 총 7개의 데이터셋을 포함하고 있다.

[0152] 합성 데이터셋은 R-MAT 모델을 사용하여 정점의 수를 변화시키며 생성한 5개의 데이터셋을 포함하고 있다.

[0153] 표 1은 본 실험에서 이용하는 리얼 데이터셋들의 정점과 간선 개수를 표로 나타낸다.

[0154] R-MAT 모델을 사용하면 소셜 네트워크 그래프와 특성이 매우 유사한 데이터 그래프를 생성할 수 있다.

[0155] 표 2는 본 실험에서 사용한 합성 데이터셋의 통계 정보를 표로 나타낸다. 본 실험에서는 최소의 버퍼 크기가 주어진다 하더라도 주어진 질의를 수행할 수 있도록 하기 위하여, 필요한 메모리 버퍼 크기는 사용자로부터 주어진 크기에 스레드 수만큼의 프레임을 추가로 할당하여 실험을 수행하였다.

[0156] 표 1

	Wikitalk	cit-Patents	web-Google	LiveJournal	Twitter	Orkut	UK
$\|V\|$	2.4M	3.8M	0.9M	4.8M	42M	3.1M	106M
$\|E\|$	5.0M	18M	5.1M	69M	1,468M	117M	3,739M

[0157]

[0158] 표 2

	RMAT_v4	RMAT_v8	RMAT_v16	RMAT_v32	RMAT_v64
$\|V\|$	4M	8M	16M	32M	64M

[0159]

[0160] 본 실험에서는 여러 환경 상에서의 성능을 측정하여 본 발명의 서브 그래프 리스팅 방법의 신뢰성을 보이기 위하여, 서로 다른 두 가지 환경에서 실험을 진행하였다.

[0161] 실험 환경은 24G 바이트 메모리, 1테라 바이트 SSD(Samsung 850Pro), Intel Core i7 3930K CPU 3.20GHz CPU(총 6 코어)를 가지고 있다.

[0162] 그리고, Windows 7, D-ENV는 각각 MS windows 7과 MS windows 8 운영 체제를 탑재하고 있다.

[0163] 실험 성능을 측정하기 위한 측정 단위로는 디스크 I/O 횟수와 수행 시간을 사용하였다.

[0164] 이하, 도 8 및 도 9을 참조하여, 멀티 스레드 환경에서 본 발명의 한 실시예에 따른 종래의 분산 처리 방식인 PSgL을 디스크 기반 알고리즘으로 구현한 것과 비교하여 성능이 향상된 것을 나타낸다.

[0165] 도 8은 본 발명의 한 실시예에 따른 서브그래프 리스팅 방법을 리얼 데이터셋에 대하여 수행한 시간을 나타낸다.

[0166] 도 9는 본 발명의 한 실시예에 따른 서브그래프 리스팅 방법을 리얼 데이터셋에 대하여 수행시 디스크 I/O 횟수를 나타낸다.

[0167] 도 8의 (d) 내지 (e)에 도시된 바와 같이, QG3, QG5, QG8 및 QG9의 결과가 나타나지 않는 것을 확인할 수 있는데, 이는 PSgL의 부분 결과 용량이 디스크 용량을 초과하여 결과를 산출할 수 없기 때문이다.

[0168] 즉, 도 8 및 도 9를 참조하면, 본 발명의 서브 그래프 리스팅 방법은 종래의 PSgL보다 수행 시간 측면에서 우수한 결과를 나타낸다.

[0169] 구체적으로, 본 발명의 서브 그래프 리스팅 방법의 수행 시간은 최소 3.6배에서 662배로 성능이 향상되었고, 디스크 I/O 횟수를 1.2배에서 7436배까지 감소 시켰다.

[0170] 이하, 표 3을 참조하여, 본 발명의 삼각형 리스팅 수행 시간에 대해 설명한다.

[0171] 표 3은 스레드 수 6개에서 본 발명의 서브 그래프 리스팅 방법과 종래의 OPT의 삼각형 리스팅 수행 시간을 나타낸다.

[0172] 표 3

	RBI-Match	OPT
Twitter	451	534
UK	488	520

[0173]

[0174] 본 발명의 서브 그래프 리스팅 방법은 OPT와 비교하여 UK와 Twitter 데이터셋에서 각각 1.06배와 1.18배 빠른 검색 속도를 보인다.

[0175] 본 발명의 효과를 더욱 구체적으로 설명하자면, 대부분의 질의 그래프와 데이터 그래프의 경우에 있어서 종래의 방법들보다 성능이 크게 향상되었고, 종래의 PSgL에서 51대의 머신을 사용하여 측정한 방법과 비교하여, 최대 44배 성능이 향상되었다.

[0176] PSgL을 디스크 기반으로 수정한 방법과 본 발명의 서브 그래프 리스팅 방법을 비교하면 페이지 액세스 횟수를

최대 7236배로 줄였으며 성능을 최대 662배까지 향상시켰다.

[0177] 또한, 본 발명은 OPT보다도 우수한 성능을 보였다.

[0178] 이와 같은 결과를 기반으로, 본 발명의 서브 그래프 리스팅 방법은 대규모 데이터 그래프에 대한 서브 그래프 리스팅의 성능을 크게 향상시킬 수 있다.

[0179] 이와 같은 방법에 의해, 본 발명의 서브 그래프 리스팅 방법에 의하면 데이터 그래프가 큰 경우라도, 디스크 액세스 횟수를 최소화하여 데이터 그래프의 질의 그래프에 대한 서브 그래프 리스팅을 효율적으로 수행할 수 있다.

[0180] 본 발명은 대규모의 데이터 그래프에서 서브 그래프 리스팅을 효율적으로 처리할 수 있는 RBI 그래프를 도입하여 디스크의 I/O 횟수를 대폭 감소 시킬 수 있다.

[0181] 본 발명의 서브 그래프 리스팅 방법은 메모리 버퍼를 내부, 피벗 및 외부 서브 그래프로 나눌 수 있으며, 그에 따라 내부 및 외부 리스팅을 독립적으로 수행할 수 있어 동시 작업이 가능하다.

[0182] 본 발명의 서브 그래프 리스팅 방법은 RBI 그래프 및 이를 이용한 매칭을 통해 특히 대규모 데이터 그래프에서 서브 그래프 리스팅을 효과적으로 처리할 수 있다.

[0183] 또한, 본 발명의 서브 그래프 리스팅 방법은 깊이 우선 탐색을 기반으로 한 매핑 방식을 이용하여 매핑 중간 결과를 저장하지 않아 저장 공간의 오버헤드 없이 효과적으로 매핑을 수행할 수 있다.

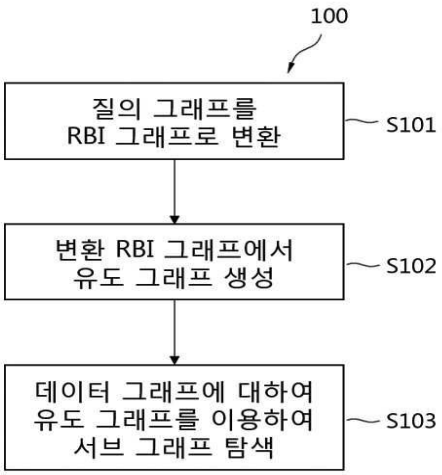
[0184] 이상에서는 본 발명을 실시예에 기초하여 설명하였으나, 본 발명의 사상은 상기 실시예에 제한되지 아니하며, 본 발명의 사상을 이해하는 당업자는 동일한 사상의 범위 내에서, 구성요소의 부가, 변경, 삭제, 추가 등에 의해서 다른 실시 예를 용이하게 제안할 수 있을 것이나, 이 또한 본 발명의 사상범위 내에 든다고 할 것이다.

부호의 설명

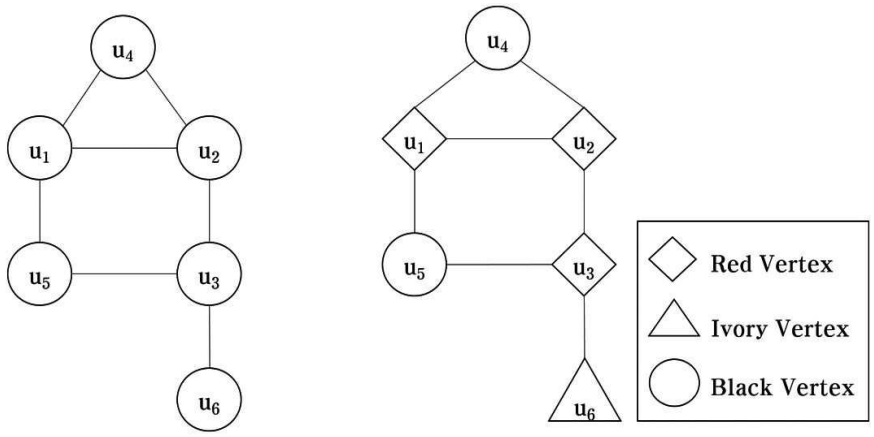
[0185] 100 : 서브 그래프 리스팅 방법

도면

도면1

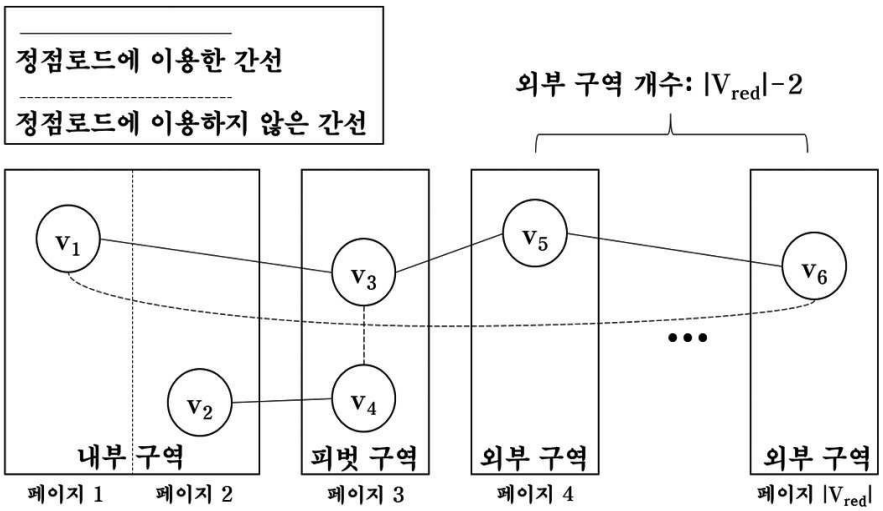


도면2

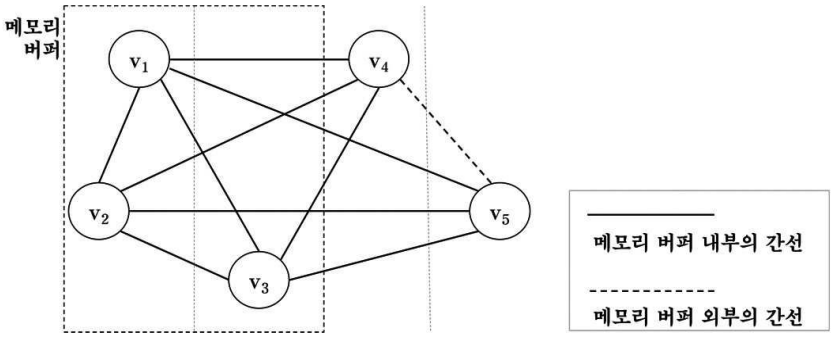


(a) 질의 그래프 G_Q (b) G_Q를 변형한 RBI 그래프 G_{RBI}

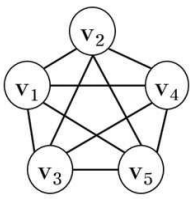
도면3



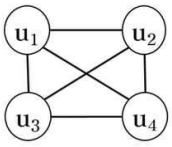
도면4



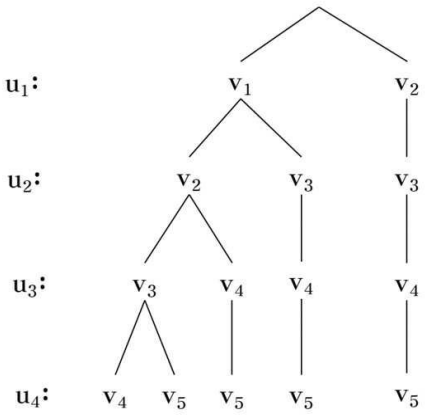
도면5



(a) 데이터 그래프 G_D

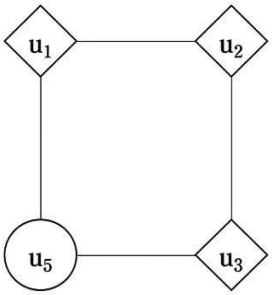


(b) 질의 그래프 G_Q

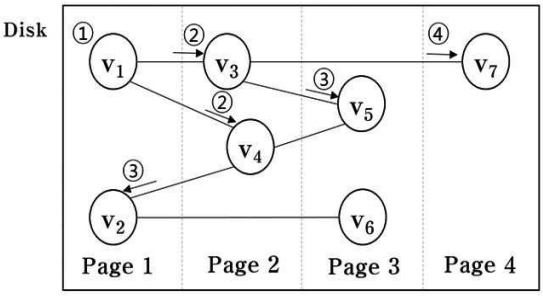


(c) G_D, G_Q에서의 호출 그래프

도면6

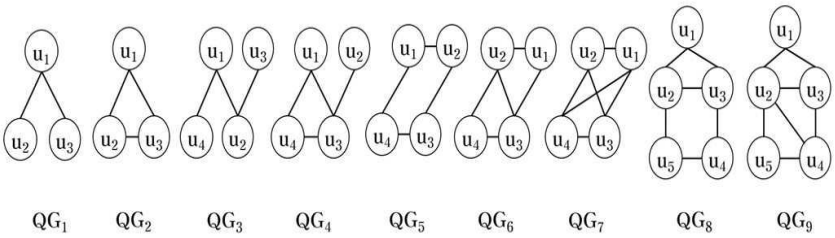


(a)RBI 그래프, V_{red} = $\{u_1, u_2, u_3\}$



(b) 데이터 그래프 상의 정점 방문 순서의 예

도면7

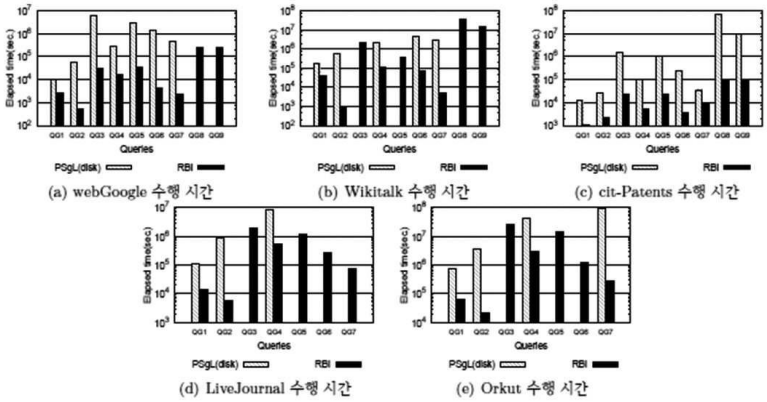


(a) 질의 그래프

QG_1	QG_2	QG_3	QG_4	QG_5	QG_6	QG_7	QG_8	QG_9
$u_2 < u_3$	$u_1 < u_2$ $u_1 < u_3$ $u_2 < u_3$	$u_1 < u_2$	$u_1 < u_2$	$u_1 < u_2$ $u_1 < u_3$ $u_2 < u_3$ $u_2 < u_4$	$u_1 < u_4$ $u_2 < u_3$	$u_1 < u_2$ $u_1 < u_3$ $u_1 < u_4$ $u_2 < u_3$ $u_2 < u_4$ $u_2 < u_4$	$u_2 < u_3$	$u_1 < u_5$

(b) 질의 그래프의 부분 순서 집합

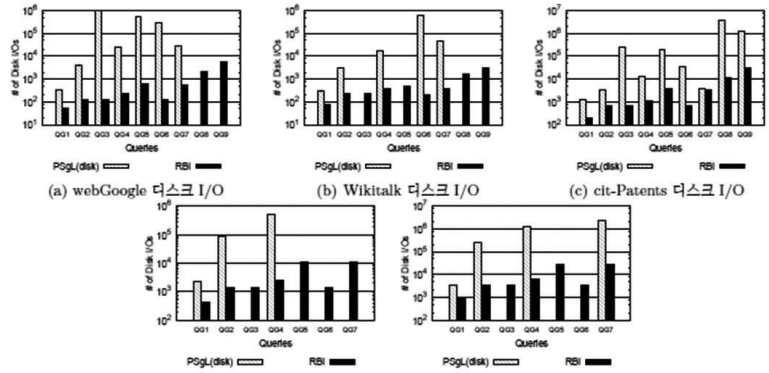
도면8



(a) webGoogle 수행 시간 (b) Wikitalk 수행 시간 (c) cit-Patents 수행 시간

(d) LiveJournal 수행 시간 (e) Orkut 수행 시간

도면9



(a) webGoogle 디스크 I/O (b) Wikitalk 디스크 I/O (c) cit-Patents 디스크 I/O

(d) LiveJournal 디스크 I/O (e) Orkut 디스크 I/O