(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2017년11월24일
(11) 등록번호 10-1801468
(24) 등록일자 2017년11월20일

(51) 국제특허분류(Int. Cl.)
G06F 17/30 (2006.01)
(52) CPC특허분류
G06F 17/30958 (2013.01)
G06F 17/30445 (2013.01)
(21) 출원번호 10-2016-0059818
(22) 출원일자 2016년05월16일
심사청구일자 2016년05월16일
(56) 선행기술조사문헌
KR1020060045780 A
KR1020160015080 A
KR101556060 B1

(73) 특허권자
포항공과대학교 산학협력단
경상북도 포항시 남구 청암로 77 (지곡동)
(72) 발명자
한욱신
경상북도 포항시 남구 청암로 77 창의IT융합공학과 (지곡동, 포항공과대학교)
김현지
울산광역시 동구 월봉12길 50, C동 308호 (화정동, 송정타워맨션3차)
이준영
대구광역시 수성구 청호로69길 30, 201동 1002호 (황금동, 가든하이츠2차아파트)
(74) 대리인
특허법인이룸리온

전체 청구항 수 : 총 4 항 심사관 : 홍경아

(54) 발명의 명칭 **단일머신 상의 대규모 그래프에서 서브그래프를 열거하는 병렬 기법**

(57) 요 약

본 발명은 단일머신 상의 대규모 그래프에서 서브그래프를 병렬적으로 열거하는 방법에 관한 것으로, 본 발명은 a) 대칭-파괴 알고리즘을 이용하여 질의 그래프로부터 부분 순서들(partial orders)의 세트(PO)를 검출하는 단계와, b) 상기 부분순서들의 세트(PO)를 이용하여 RBI(Red, Black, Ivory) 질의 그래프(q_{RBI})및 적색 질의 그래프

(뒷면에 계속)

대 표 도 - 도1

Algorithm 1. DUALSIM

Input: A data graph g, A query graph q
/* 1. Preparation step */
$PO \leftarrow$ FINDPARTIALORDERS(q);
$\langle q_{RBI}, q_R \rangle \leftarrow$ GENERATERBIQUERYGRAPH(q, PO);
$\{vgs_i\} \leftarrow$ FINDVGROUPSEQUENCES(q_R, PO);
$mo_g \leftarrow$ FINDGLOBALMATCHINGORDER($\{vgs_i\}$));
$\{vgf_i\} \leftarrow$ BUILDVGROUPFORESTS($\{vgs_i\}, mo_g$);

/* 2. Execution step */
INITIALIZECANDIDATESEQUENCES(root nodes in $\{vgf_i\}$);
foreach (merged vertex/page window (mvw_1, mpw_1) from $\{vgf_i[1]\}$) **do**
  foreach (page id $pid \in mpw_1$) **do**
    AsyncRead(pid, COMPUTECANDIDATESEQUENCES(pid, $\{cvw_{i,1}\}$, all child nodes of $\{vgf_i[1]\}$);
  end
  wait until COMPUTECANDIDATESEQUENCES executions are finished
  $level \leftarrow 2$;
  DELEGATEEXTERNALSUBGRAPHENUMERATION($level$, q_{RBI}, $\{vgs_i\}$, $\{vgf_i\}$, $\{mvw_j\}$, $\{mpw_j\}$);
  INTERNALSUBGRAPHENUMERATION(mvw_1, mpw_1);
  UNPINPAGES(mpw_1);
  CLEARCANDIDATESEQUENCES(the children of $\{vgf_i[1]\}$);
end

(q_R)를 생성하는 단계와, c) 상기 RBI 질의 그래프를 이용하여 모든 v-그룹 시퀀스들을 검출하고, 모든 v-그룹 시퀀스들을 고려하여 글로벌 매칭 순서를 검출하는 단계와, d) 상기 글로벌 매칭 순서를 이용하여 각 v-그룹 시퀀스에 대한 v-그룹 포리스트(forest)를 구축하는 단계와, e) v-그룹 포리스트의 모든 루트 노드에 대한 후보 정점/페이지 시퀀스들을 초기화하는 단계와, f) 레벨 1에서 병합된 정점 윈도우(mvw_1)와 페이지 윈도우(mpw_1)를 획득하는 단계와, g) 상기 페이지 윈도우(mpw_1)의 각 페이지마다, 페이지를 비동기 판독하는 단계와, h) 상기 병합된 정점 윈도우로부터 연결된 모든 외부 서브그래프를 찾도록 재귀 함수 DelegateExternalSubgraphEnumeration(·)를 인보크(invoke)하고, 메인 스레드는 외부 서브그래프를 나머지 스레드들에 위임한 후, 메인 스레드는 내부 영역에 로딩된 페이지들을 이용하여 내부 서브그래프 열거를 실행하는 단계를 포함한다.

(52) CPC특허분류

G06F 17/30545 (2013.01)

이 발명을 지원한 국가연구개발사업

과제고유번호 NRF-2014R1A2A2A01004454
부처명 미래창조과학부
연구관리전문기관 한국연구재단
연구사업명 중견연구자 지원 사업
연구과제명 신약 발견 및 광고 대행 추천을 위한 고성능 Top-K 검색 엔진의 개발
기 여 율 1/3
주관기관 포항공과대학교 산학협력단
연구기간 2016.05.01 ~ 2017.04.30

이 발명을 지원한 국가연구개발사업

과제고유번호 IITP-R0346-16-1007
부처명 미래창조과학부
연구관리전문기관 정보통신기술진흥센터
연구사업명 ICT명품인재양성사업
연구과제명 미래IT융합연구원
기 여 율 1/3
주관기관 포항공과대학교 산학협력단
연구기간 2016.01.01 ~ 2016.12.31

이 발명을 지원한 국가연구개발사업

과제고유번호 IITP-R0346-16-1008
부처명 미래창조과학부
연구관리전문기관 한국마이크로소프트
연구사업명 산업체과제
연구과제명 매시브 네트워크에서 효율적인 서브그래프 리스팅
기 여 율 1/3
주관기관 포항공과대학교 산학협력단
연구기간 2014.06.30 ~ 2015.06.12

명 세 서

청구범위

청구항 1

a) 대칭-파괴 알고리즘을 이용하여 부분 순서들(partial orders)의 세트(PO)를 검출하는 단계;

b) 상기 부분순서들의 세트(PO)를 이용하여 RBI(Red, Black, Ivory) 질의 그래프(q_{RBI})및 적색 질의 그래프(q_R)를 생성하는 단계;

c) 상기 RBI 질의 그래프를 이용하여 모든 v-그룹 시퀀스들을 검출하고, 모든 v-그룹 시퀀스들을 고려하여 글로벌 매칭 순서를 검출하는 단계;

d) 상기 글로벌 매칭 순서를 이용하여 각 v-그룹 시퀀스에 대한 v-그룹 포리스트(forest)를 구축하는 단계;

e) v-그룹 포리스트의 모든 루트 노드에 대한 후보 정점/페이지 시퀀스들을 초기화하는 단계;

f) 레벨 1에서 병합된 정점 윈도우(mvw_1)와 페이지 윈도우(mpw_1)를 획득하는 단계;

g) 상기 페이지 윈도우(mpw_1)의 각 페이지마다, 페이지를 비동기 판독하는 단계;

h) 상기 병합된 정점 윈도우로부터 연결된 모든 외부 서브그래프를 찾도록 재귀 함수 DelegateExternalSubgraphEnumeration(·)를 인보크(invoke)하고, 메인 스레드는 외부 서브그래프를 나머지 스레드들에 위임한 후, 메인 스레드는 내부 영역에 로딩된 페이지들을 이용하여 내부 서브그래프 열거를 실행하는 단계를 포함하는 단일머신 상의 대규모 그래프에서 서브그래프를 병렬적으로 열거하는 방법.

청구항 2

제1항에 있어서,

상기 b) 단계에서 RBI 질의 그래프는, 질의 그래프의 정점들을 적색, 검정색, 아이보리색으로 표시하되, 적색 정점은 직접 억세스하여 매칭하는 정점이고, 아이보리색 정점은 적어도 두 적색 정점에 연결된 정점이고, 검정색 정점은 하나의 적색 정점에 연결된 정점으로 하는 것을 특징으로 하는 단일머신 상의 대규모 그래프에서 서브그래프를 병렬적으로 열거하는 방법.

청구항 3

제1항에 있어서,

상기 c) 단계의 v-그룹 시퀀스는,

모든 가능한 질의 시퀀스들을 열거하여 전체 순서 질의 시퀀스를 획득하는 단계;

상기 전체 순서 질의 시퀀스를 아래의 정의 1과 특성 1에 따라 페이지 시퀀스를 비감소 순서로 고정하여 모든 전체 순서 질의 시퀀스를 가변하는 단계; 및

상기 전체 순서 질의 시퀀스들을 아래의 정의 2에 따라 그룹화하는 단계를 통해 획득되는 것을 특징으로 하는 단일머신 상의 대규모 그래프에서 서브그래프를 병렬적으로 열거하는 방법.

정의 1은 RQG $q_R(V_R, E_R)$ 및 부분차수 PO의 세트가 있는 경우, 전체 순서 질의 시퀀스 qs는, PO가 qs의 질의 정점들 중에서 전체 순서 (qs[1] < qs[2] < ...< qs[|V_R|])의 서브세트이도록 V_R에서의 정점들의 순열 시퀀스이고,

특성 1은 RQG $q_R(V_R, E_R)$이 주어진 경우, 매핑(주입) m을 이용하여 전체 순서 질의 시퀀스 qs가 크기 |V_R|의 데

이터 정점 시퀀스와 매칭되면, m[qs[1]] < m[qs[2]] < ...<m[qs[|V_R|]] 이며,

정의 2는 QS를 RQG $q_R(V_R, E_R)$에 대한 모든 전체 순서 질의 시퀀스들의 세트라 한다. $\cong$ 를, $qs_i \cong qs_j, if\ and\ only\ if\ \forall 1 \le k,k' \le |V_R|, if\ (qs_i[k], qs_i[k']) \in E_R, (qs_j[k], qs_j[k']) \in E_R$ 을 충족하는 QS에 대한 등가 관계라 함

청구항 4

제1항에 있어서,

상기 d) 단계의 v-그룹 포리스트는, 데이터 그래프를 횡단할 때, 모든 가능한 부분 해들을 세이브하기 보다는 적색 질의 정점마다 후보 정점들을 저장하기 위한 비순환적 횡단 구조이며,

상기 v-그룹 시퀀스마다 부분 해들의 총 크기가 $O(|V_R| * |V_G|)$ 에 의해 한정되는 것을 특징으로 하는 단일머신 상의 대규모 그래프에서 서브그래프를 병렬적으로 열거하는 방법.

발명의 설명

기 술 분 야

[0001] 본 발명은 단일머신 상의 대규모 그래프에서 서브그래프를 병렬적으로 열거하는 방법에 관한 것으로, 더 상세하게는 디스크 기반이며, 단일 머신에서 병렬적으로 중간 결과 수의 급증 없이 대규모 그래프를 처리할 수 있는 서브그래프를 열거하는 방법에 관한 것이다.

배 경 기 술

[0002] 일반적으로 서브그래프의 열거는 소셜네트워크의 진화 연구, 소형 커널 연산, 네트워크 모티브 탐사, 서브그래프 프리퀀시스 등의 많은 분야에서 중요하게 다루어진다.

[0003] 대부분의 서브그래프 열거에 대한 선행연구들은 그래프들이 메모리에 상주하는 것으로 가정하여 매우 심각한 문제에 직면하였다. 최근 대규모 그래프에서 모든 서브그래프를 열거하려는 노력들은 맵리듀스(MapReduce)와 분산 그래프 엔진들과 같은 분산 프레임 워크의 활용 및 데이터 그래프의 분할에 의해 성공하는 듯이 보였다. 하지만, 현존하는 분산 접근법들은 부분적인 결과 수의 급증에 기인하여 서브그래프 열거에 대한 심각한 성능문제가 있다.

[0004] 이 발명이 속하는 기술분야의 선행 기술인 "F. N. Afrati, D. Fotakis, and J. D. Ullman.Enumerating subgraph instances using map-reduce. In ICDE, pages 62-73, 2013."에서는 서브그래프 열거를 위한 단일 맵리듀스 라운드 방법을 제안하였다.

[0005] 위의 방법은 맵단계에서 각각 다른 머신들과 다양한 방법으로 연결할 수 있는 다중의 머신들에서 간선들을 수회 이중화 하는 방법을 사용한다. 비록 이 다중(multy-way) 연결이 단일 맵리듀스 라운드(single map reduce round)에 기초하고 있지만, 이 방법은 질의 그래프가 복잡하다면 각 머신의 메모리에 거의 모든 그래프를 저장하고 있을 가능성이 높다.

[0006] 그리고 "Y. Shao, B. Cui, L. Chen, L. Ma, J. Yao, and N. Xu. Parallel subgraph listing in a large-scale graph. In SIGMOD, pages 625-636, 2014."에서는 값비싼 조인 동작을 피하기 위하여, 조인 없이 병렬 BFS(breadth-first search)를 이용하여 부분적인 서브그래프 결과를 레벨에 따라 생성하는 PSgL이라는 방법을 제안하였다. 상기 부분적인 서브그래프 결과들은 각 머신의 메인 메모리에 저장된다. 그러나 이러한 방법은 질의 정점의 수를 변화시킴으로써 부분적인 솔루션의 크기가 급증하는 것으로 관찰되었다. 따라서 이 방법은 대규

모의 그래프의 처리뿐만 아니라 적당한 크기의 그래프 에 대해서 조차도 대부분의 질의를 처리할 수 없는 문제점이 있었다.

[0007] 또한 "L. Lai, L. Qin, X. Lin, and L. Chang. Scalable subgraph enumeration in mapreduce. PVLDB, 8(10):974-985, 2015."에서는 질의 그래프를 단일 간선 또는 한 정점을 중심으로 한 두 개의 간선으로 구성된 조인 유닛으로 분해한 뒤 레프트 딥 조인을 실행하는 인핸스드 맵 리듀스 방법인 'twintwigjoin'을 제안했다. 그러나 이 조인 기반 접근법 또한 병렬 BFS 검색의 변형으로 볼 수 있다. 따라서 이 방법에서도 부분적인 솔루션들의 수가 급증 할 수밖에 없었다.

[0008] 이처럼 현존하는 분산 접근법들은 서브그래프 열거에 대한 심각한 성능 문제를 가지고 있다.

[0009] 또한, 비록 분산 컴퓨팅 자원들은 최근의 클라우드를 통해 쉽게 이용할 수 있지만, 대규모 그래프 연산은 종종 머신들간의 통신을 최소화하는 클러스터 머신들 간에 그래프의 바람직한 분할을 요구하는 알려진 어려운 문제점이 있다.

[0010] 아울러 최종 사용자의 관점에서 디버깅 및 분산 알고리즘을 최적화 하는 것은 어렵다. 이에 더하여 이러한 방법은 일반사용자 또는 소규모 연구그룹이 이러한 클러스트 노드를 구입하고 유지하기에는 고가라는 문제점이 있었다.

발명의 내용

해결하려는 과제

[0011] 상기와 같은 문제점을 감안한 본 발명이 해결하고자 하는 과제는, 디스크 기반이며, 부분적인 결과 수의 급증 없이 대규모 그래프를 처리할 수 있는 단일머신 상의 대규모 그래프에서 서브그래프를 병렬적으로 열거하는 방법을 제공함에 있다.

과제의 해결 수단

[0012] 상기와 같은 과제를 해결하기 위한 단일머신 상의 대규모 그래프에서 서브그래프를 병렬적으로 열거하는 방법은, a) 대칭-파괴 알고리즘을 이용하여 부분 순서들(partial orders)의 세트(PO)를 검출하는 단계와, b) 상기 부분 순서들의 세트(PO)를 이용하여 RBI(Red, Black, Ivory) 질의 그래프(q_{RBI})및 적색 질의 그래프(q_R)를 생성하는 단계와, c) 상기 RBI 질의 그래프를 이용하여 모든 v-그룹 시퀀스들을 검출하고, 모든 v-그룹 시퀀스들을 고려하여 글로벌 매칭 순서를 검출하는 단계와, d) 상기 글로벌 매칭 순서를 이용하여 각 v-그룹 시퀀스에 대한 v-그룹 포리스트(forest)를 구축하는 단계와, e) v-그룹 포리스트의 모든 루트 노드에 대한 후보 정점/페이지 시퀀스들을 초기화하는 단계와, f) 레벨 1에서 병합된 정점 윈도우(mvw_1)와 페이지 윈도우(mpw_1)를 획득하는 단계와, g) 상기 페이지 윈도우(mpw_1)의 각 페이지마다, 페이지를 비동기 판독하는 단계와, h) 상기 병합된 정점 윈도우로부터 연결된 모든 외부 서브그래프를 찾도록 재귀 함수 DelegateExternalSubgraphEnumeration(·)를 인보크(invoke)하고, 메인 스레드는 외부 서브그래프를 나머지 스레드들에 위임한 후, 메인 스레드는 내부 영역에 로딩된 페이지들을 이용하여 내부 서브그래프 열거를 실행하는 단계를 포함한다.

발명의 효과

[0013] 본 발명 단일머신 상의 대규모 그래프에서 서브그래프를 병렬적으로 열거하는 방법은, 디스크 액세스 횟수를 감소시킬 수 있으며, 메모리의 사용을 줄이고, 모든 데이터셋 및 그래프 크기에 대하여 작동할 수 있는 효과가 있다.

[0014]

도면의 간단한 설명

[0015] 도 1은 본 발명 단일머신 상의 대규모 그래프에서 서브그래프를 병렬적으로 열거하는 방법을 구현한 알고리즘이다.

도 2 내지 도 14는 각각 도 1의 각 과정을 설명하기 위한 설명도이다.

도 15는 q1과 q4를 처리하는 과정에서 본 발명의 시간에 따른 메모리 버퍼 사이즈를 측정한 결과표이다.

도 16은 실행 스레드의 수를 다양화하여 스피드업을 측정한 결과이다.

도 17은 본 발명과 종래의 TWINTWIGJOIN방식에 대하여 데이터셋에서 경과시간을 보인 그래프이다.

도 18은 q1 내지 q5 각각을 본 발명과 종래 TWINTWIGJOIN 방식으로 특정 데이터셋에서 처리하였을 때 경과시간을 비교한 표이다.

도 19는 본 발명과 TWINTWIGJOIN 방식을 사용하여 그래프 크기의 변화에 따른 경과시간을 측정한 그래프이다.

도 20은 본 발명과 TWINTWIGJOIN 방식에 PSgL을 추가하여 다수의 데이터셋에서 경과시간을 측정한 그래프이다.

도 21은 도 12의 조건으로 q1 내지 q5를 처리할 때 경과시간 측정 그래프이다.

도 22는 도 12의 조건으로 그래프 크기의 변화에 따른 경과시간을 측정한 그래프이다.

발명을 실시하기 위한 구체적인 내용

[0016] 이하, 본 발명 단일머신 상의 대규모 그래프에서 서브그래프를 병렬적으로 열거하는 방법에 대하여 첨부한 도면을 참조하여 상세히 설명한다.

[0017] 도 1은 본 발명 단일머신 상의 대규모 그래프에서 서브그래프를 병렬적으로 열거하는 방법을 구현한 알고리즘이고, 도 2 내지 도 14는 도 1의 각 과정을 설명하기 위한 설명도이다.

[0018] 도 1과 도 2 내지 도 14를 각각 참조하면, 본 발명은 두 개의 단계, 즉, 준비 단계와 실행 단계로 이루어진다.

[0019] 상기 준비 단계에서, 본 발명은, 우선, 대칭-파괴 알고리즘을 이용하여 부분 순서들(partial orders)의 세트(PO)를 찾는다(라인 1). 상기 대칭 파괴 알고리즘은 "J. A. Grochow and M. Kellis. Network motif discovery using subgraph enumeration and symmetry-breaking. In RECOMB, pages 92-106, 2007."을 참고한다.

[0020] 이어서, 상기 부분순서들의 세트(PO)를 이용하여 RBI(Red, Black, Ivory) 질의 그래프(q_{RBI})및 적색 질의 그래프(q_R)를 생성한다(라인 2).

[0021] 다음으로, 모든 v-그룹 시퀀스들을 찾는다(라인 3).

[0022] 그 다음, 모든 v-그룹 시퀀스들을 고려하여 글로벌 매칭 순서를 찾는다(라인 4).

[0023] 다음으로, 글로벌 매칭 순서를 이용하여 각 v-그룹 시퀀스에 대한 v-그룹 포리스트(forest)를 구축한다(라인 5).

[0024] 그 다음, 실행 단계에서, 본 발명은 v-그룹 포리스트의 모든 루트 노드에 대한 후보 정점/페이지 시퀀스들을 초기화한다(라인 6). 이 초기화는 모든 루트 노드를 모든 정점/페이지에 대응하게 한다. 이제, v-그룹 포리스트를 이용함으로써 데이터 그래프를 레벨 단위로 횡단한다.

[0025] 우선, 레벨 1에서 현재 병합된 정점/페이지 윈도우인 mvw_1 및 mpw_1을 얻는다(라인 7).

[0026] 그 다음, mpw_1의 각 페이지마다, 페이지를 비동기 판독한다(라인 8 내지 라인 10). 비동기 I/O를 요청할 때 콜백 함수를 패스할 수 있다는 점에 주목한다.

[0027] 여기서, 요청된 페이지가 버퍼에 로딩되자마자 콜백 함수 ComputeCandidateSequences(·)를

인보크(invoke)한다.

[0028] 상기 콜백 함수는 루트 노드들의 모든 자식 노드들에 대하여 후보 정점/페이지 시퀀스들을 연산한다. 모든 요청된 페이지들이 버퍼에 로딩될 때까지 대기할 필요가 있다.

[0029] 그 다음, 현재 병합된 정점 윈도우로부터 연결된 모든 외부 서브그래프를 찾도록 재귀 함수 DelegateExternalSubgraphEnumeration(·)를 인보크한다(라인 13).

[0030] 이와 같은 방법으로, 메인 스레드는 외부 서브그래프를 나머지 스레드들에 위임한다. 메인 스레드는 내부 영역에 로딩된 페이지들을 이용하여 내부 서브그래프 열거를 실행한다(라인 13).

[0031] 내부 서브그래프 열거 또는 외부 서브그래프 열거가 상대측 열거보다 먼저 종료되면, 이용가능한 스레드들을 미종료된 서브그래프 열거에 할당하는 스레드 모핑(Thread Morphing)을 수행한다. 상기 스레드 모핑은 "J. Kim, W. Han, S. Lee, K. Park, and H. Yu. OPT: a new framework for overlapped and parallel triangulation in large-scale graphs. In SIGMOD, pages 637-648, 2014."를 참고한다.

[0032] 이하, 상기와 같이 구성되는 본 발명의 바람직한 실시예에 따른 단일머신 상의 대규모 그래프에서 서브그래프를 병렬적으로 열거하는 방법의 구성과 작용에 대하여 보다 상세히 설명한다.

[0033] 먼저, 도 1에서 라인 1에 기재된 바와 같이 우선, 대칭-파괴 알고리즘을 이용하여 부분 순서들의 세트(PO)를 찾는다. 상기 대칭 파괴 알고리즘은 "J. A. Grochow and M. Kellis. Network motif discovery using subgraph enumeration and symmetry-breaking. In RECOMB, pages 92-106, 2007."을 참고한다.

[0034] 도 2에서 질의 그래프의 부분 순서들의 세트(PO)는 하나의 부분 순서로 이루어져 있다.

[0035] 그 다음, 라인 2에 기재한 바와 같이 상기 부분 순서의 세트(PO)를 이용하여 RBI(Red, Black, Ivory) 질의 그래프(q_{RBI})및 적색 질의 그래프(q_R)를 구한다.

[0036] 도 3은 질의 그래프를 RBI 질의 그래프(q_{RBI})로 변환한 예시도이다.

[0037] 디스크 억세스 회수를 최소화하기 위하여, 그래프 운행 동안 최소화된 질의 정점들의 수를 사용할 필요가 있다. 질의 정점의 서브셋을 위한 인접 리스트들은 디스크로부터 검색되고, 인접리스트 검색으로 남은 질의 정점들을 매칭시킬 수 있다.

[0038] 앞서 설명한 바와 같이 본 발명에서는 상기 정점들에 색상을 부여하여 질의 정점 그래프(q)를 RBI 정점 그래프(q_{RBI})로 변환하고, 특정한 질의 정점이 적색이면, 매칭을 위한 데이터 정점의 인접 리스트를 검색할 필요가 있다.

[0039] 만약 특정한 질의 정점이 다수의 적색 질의 정점에 인접하는 경우, 그 정점은 아이보리색을 띈다. 결과적으로 특정한 정점에 상응하는 데이터 정점의 식별을 위하여, 적색 질의 정점들을 위한 데이터 정점들의 인접 리스트의 교차를 수행할 필요가 있다.

[0040] 특정한 정점이 단지 하나의 적색 질의 정점에 인접하다면, 그 특정한 정점은 검정색으로 표시된다.

[0041] 이때 질의 정점과 관련된 데이터 정점을 식별하기 위하여, 우리는 적색 질의 정점에 연관된 데이터 정점의 인접 리스트를 검색할 필요가 있다. 예를 들어 도 3(b)에 도시된 질의 정점 그래프는 5개의 질의 정점(u_1,u_2,u_3,u_4,u_5)들과 6개의 간선들을 가지고 있으며, 단지 u_1,u_2,u_3가 적색으로 표시되고 그래프 이동에 사용된다. 다른 두 질의 정점 u_4, u_5는 아이보리 색으로 표시된다.

[0042] 따라서 u_4와 u_5와 연관된 데이터 정점의 식별을 위하여 양방향 교차를 수행할 수 있다. 예를 들어 u_1과 u_3를 위한 두 인접 리스트가 얻어지면, 우리는 두 인접 리스트들의 교차에 의해 u_4와 연관된 데이터 정점을 얻을 수 있다.

[0043] 적색 질의 정점은 직접 억세스하여 매칭을 해야 하는 정점이다.

[0044] 도 3의 (a)에서 u_1, u_2는 적색이고 u_4와 u_5는 u_1, u_2에 각각 연결되어 있기 때문에 아이보리색이다. 반면에 u_3는 u_2에만 연결되어 있기 때문에 검정색이다.

[0045] 도 3의 (b)에서 u_1, u_2, u_3는 적색이다. u_4, u_5가 두 적색 질의 정점 양측에 연결되어 있기 때문에 아이보리색이며, 여기에는 검정색 정점이 없다.

[0046] 적색 질의 그래프(RQG) $qr(V_R,E_R)$는 $V_{R\cdot)}V_Q$, $E_{R\cdot)}E_Q$의 범위에서 RBI 질의 그래프($q_{RBI}(vq,eq)$)에서 적색 질의 정점의 유도 그래프이다. 예를 들어 도 3의 (a)에서 RQG는 질의 정점 (u_1,u_2)의 유도 질의 그래프이고, 도 3의 (b)에서는 질의 정점(u_1,u_2,u_3)의 유도 서브그래프이다.

[0047] 적색 질의 정점의 수를 최소화하기 위하여 MVC(Minimum Vertex Cover)를 적용할 수 있다. 선택된 MVC 내의 모든 질의 정점들은 적색 질의 정점이다.

[0048] 상기 적색 질의 정점의 수가 약간 증가하지만, 알려진 MCVC(Minimum Connected Vertex Cover)를 사용하는 것이 바람직하다. 이러한 접근 방식은 첫번째 적색 질의 정점과 연관된 데이터 정점으로부터 데이터 정점들에 도달하는 것을 가능하게 한다. 그렇지 않으면 모든 정점을 스캔해야 한다.

[0049] 이러한 MCVC에 대해서는 "J. Cardinal and E. Levy. Connected vertex covers in dense graphs. Theor. Comput. Sci., 411(26-28):2581-2590, 2010."에 기재되어 있으며, 본 발명에서는 MCVC를 MVC로 변환 적용할 수 있다.

[0050] MVC를 찾는 과정은 비결정 난해(NP-hard)에 속하지만, 대부분의 경우 질의 그래프 정점의 수, |Vq|가 매우 작아서 지수의 복잡성은 현실에서 문제가 되지 않는다.

[0051] 질의 그래프는 하나 이상의 MCVC를 포함하며, 아래의 두 제안 규칙을 통해 하나 이상의 MCVC를 포함하는 질의 그래프에서 하나의 적색 질의 그래프를 선택할 수 있다.

[0052] 규칙1은 주어진 두 적색 질의 그래프(RQG)에서 적색 질의 그래프(RQG)의 부분 순서가 많은 것을 선택하는 것이고, 규칙2는 위의 규칙1에서 두 적색 질의 그래프(RQG)가 동일한 경우 더 많은 간선을 가지는 것을 선택한다.

[0053] 이러한 규칙에 의해 디스크의 억세스 및 CPU의 동작시간을 줄일 수 있게 된다.

[0054] 다음으로, 라인 3의 기재와 같이 모든 v-그룹 시퀀스들을 찾는다.

[0055] 도 4에는 전체 순서 질의 시퀀스를 v-그룹 시퀀스로 변환하는 과정을 도시하고 있다.

[0056] v-그룹 시퀀스를 찾기 위해 본 발명은 듀얼 방식을 사용한다.

[0057] 듀얼 방식에서는, 데이터 그래프의 역할과 질의 그래프의 역할을 교환한다. 특히, 질의의 매칭 순서를 고정한 후 데이터 정점들을 매칭하는 것 대신에, 디스크 페이지들의 세트를 고정함으로써 데이터 정점들을 고정한 후 이러한 페이지들에서 질의 그래프 정점의 매칭 순서를 변경하며 모든 서브그래프 매칭(즉, 매칭되는 질의 정점들)을 찾는다. 이는 디스크 I/O 횟수를 크게 감소시킬 수 있다.

[0058] 서브그래프들을 열거하기 전에, 듀얼 방식은, 우선, 모든 가능한 질의 시퀀스들을 열거할 필요가 있으며, 이러한 각 시퀀스는 순서화된 데이터 시퀀스와 매칭되며, 이에 따라 데이터 시퀀스를 고정하게 된다. 이러한 질의 시퀀스들은 전체 순서(full-order) 질의 시퀀스들이라 칭한다. 정의 1에서 질의 정점들의 선형화인 전체 순서 질의 시퀀스의 공식적인 정의를 제공한다.

[0059] 정의 1. RQG $q_R(V_R, E_R)$ 및 부분차수 PO의 세트가 있는 경우, 전체 순서 질의 시퀀스 qs는, PO가 qs의 질의 정점들 중에서 전체 순서 (qs[1] < qs[2] <....< qs[$|V_R|$])의 서브세트임을 만족하는 V_R에서의 정점들의 순열 시퀀

스이다.

[0060] 아래의 특성 1은, 전체 순서 질의 시퀀스가 크기 $|V_R|$ 요소의 요소들로 순서화된 데이터 시퀀스와 매칭하는 것을 나타낸다.

[0061] 특성 1. RQG $q_R(V_R, E_R)$이 주어진 경우, 매핑(주입) m을 이용하여 전체 순서 질의 시퀀스 qs가 크기 $|V_R|$의 데이터 정점 시퀀스와 매칭되면, $m[qs[1]] < m[qs[2]] < ....<m[qs[|V_R|]]$ 를 충족한다.

[0062] 보제 1. 정점들이 id(·)에 따라 페이지들에 저장된다고 가정한다. RQG $q_R(V_R, E_R)$이 주어진 경우, 주입 m을 이용하여 전체 순서 질의 시퀀스 qs가 크기 $|V_R|$의 데이터 정점 시퀀스와 매칭되면, $P(m[qs[1]]) \leq P(m[qs[2]]) \leq ... \leq P(m[qs[|V_R|]])$ 를 충족한다.

[0063] 증명. 특성 1에 따르면, $m[qs[1]] < m[qs[2]] < ... \ m[qs|V_R|]$. 정점들은 id(·)에 따라 저장된다. 따라서, $id(v_i) < id(v_j)$이면, $P(v_i) = P(v_j)$이다. 따라서, 조건 $P(m[qs[1]]) \leq P(m[qs[2]]) \leq ... \leq P(m[qs[|V_R|]])$ 이 충족된다.

[0064] 이러한 식으로, 크기 $|V_R|$의 페이지 시퀀스를 비감소 순서로 고정하여, 모든 전체 순서 질의 시퀀스들을 가변한다. q_R의 많은 전체 순서 질의 시퀀스들이 있을 수 있으며, 이에 따라, 이러한 시퀀스마다 서브그래프 매핑을 찾게 되면 과잉 매칭이 발생할 수 있다. 이러한 문제에 대처하도록, 전체 순서 질의 시퀀스들을 이들의 토폴로지에 의해 그룹화한다. 정의 2는 토폴로지 등가 관계의 공식적인 정의를 제공한다.

[0065] 정의 2. QS를 RQG $q_R(V_R, E_R)$에 대한 모든 전체 순서 질의 시퀀스들의 세트라 한다. $\cong$ 를,

$$qs_i \cong qs_j, if\ and\ only\ if\ \forall\ 1 \leq k, k' \leq$$

[0066] $$|V_R|, if\ (qs_i[k], qs_i[k']) \in E_R, (qs_j[k], qs_j[k']) \in E_R$$ 을 충족하는 QS에 대한 등가 관계라 한다.

[0067] qs_i의 등가 클래스는, qs_i에 등가인(즉, $\cong$ 인) 전체 순서 질의 시퀀스들의 세트이다. 이 클래스는, v-그룹 시퀀스의 모든 전체 순서 질의 시퀀스들이 동일한 데이터 정점들과 매칭되므로, v-그룹 시퀀스라 칭한다. 예를 들어, 도 2의 (b)에서, $qs_3(u_2, u_1, u_3)$과 $qs_4(u_2, u_3, u_1)$는 v-그룹 시퀀스를 구성한다. 따라서, (v_1, v_3, v_5)가 qs_3에 매칭되는 경우, qs_4에도 매칭된다.

[0068] 그 다음, 라인 4와 같이 모든 v-그룹 시퀀스들을 고려하여 글로벌 매칭 순서를 찾는다.

[0069] 도 5에는 페이지 시퀀스와 v-그룹 시퀀스를 매칭시키는 예가 도시되어 있다.

[0070] 상기와 같이 일단 모든 v-그룹 시퀀스들이 얻어졌으므로, 데이터 그래프, 즉 데이터베이스의 페이지들을 횡단하도록 이들의 각각에 대한 매칭 순서를 결정할 필요가 있다. 다수의 v-그룹 시퀀스들이 존재할 수 있으므로, v-그룹 시퀀스의 개수만큼 데이터 그래프를 반복적으로 횡단할 수 있다. 본 발명에서는 모든 v-그룹 시퀀스들에 기초하여 데이터 그래프를 (즉, 디스크의 페이지들) 한 번만 횡단하는 데 사용되는 글로벌 매칭 순서를 결정하는 방법을 제공한다.

[0071] 다음으로, 라인 5와 같이 글로벌 매칭 순서를 이용하여 각 v-그룹 시퀀스에 대한 v-그룹 포리스트(forest)를 구축한다.

[0072] 과도한 부분 해 크기의 문제를 해결하도록, 데이터 그래프를 횡단할 때, 모든 가능한 부분 해들을 세이브하기보다는 적색 질의 정점마다 후보 정점들을 유지한다. 이를 위해, v-그룹 시퀀스마다 v-그룹 포리스트라 칭하는

비순환적 횡단 구조를 구축한다. 이러한 식으로, v-그룹 시퀀스마다 부분 해들의 총 크기가 $O(|V_R| * |V_G|)$ 에 의해 한정된다. 비트맵을 사용하는 경우, 그 크기는 그래프 크기보다 상당히 작다.

[0073] v-그룹 포리스트의 각 노드는 v-그룹 시퀀스들에 대한 어레이 인덱스를 저장한다. 각 노드는 어레이 인덱스의 레벨과 연관된다. 각 레벨에서는, 하나의 노드만을 갖는다. 따라서, 레벨들의 개수는 $|V_R|$과 같다. 어레이 인덱스는 v-그룹 시퀀스들의 총 순서를 실시하는 데에도 사용된다. 도 6에는 어레이 인덱스들의 글로벌 매칭 순서가 (3, 2, 1)인 경우 vgs_1에 대한 v-그룹 포리스트를 도시한다.

[0074] 두 개의 간선을 갖는 이 포리스트에서는 하나의 트리만이 존재하는 한편, vgs_2에 대한 v-그룹 포리스트는 두 개의 트리로 구성된다. 제2 v-그룹 포리스트에서, 2를 어레이 인덱스로서 저장하는 노드는 3을 저장하는 노드에 연결되어 있지 않다. 1을 어레이 인덱스로서 저장하는 노드는 나머지 두 개의 노드의 자식 노드일 수 있는데, 이 때 루트 노드로부터의 경로의 길이에 대하여 루트 노드로부터 가장 멀리 있는 노드를 선택한다.

[0075] 이상으로 준비 단계가 완료되며, 그 다음 실행 단계로서 라인 6과 같이 상기 v-그룹 포리스트의 모든 루트 노드에 대한 후보 정점/페이지 시퀀스들을 초기화한다. 이 초기화는 모든 루트 노드를 모든 정점/페이지에 대응하게 한다.

[0076] 상기 v-그룹 포리스트를 이용함으로써 데이터 그래프를 레벨 단위로 횡단한다.

[0077] 글로벌 매칭 순서가 주어진 경우, 모든 v-그룹 포리스트의 토폴로지 정보 및 매칭 순서를 고려하여 데이터 그래프를 어떻게 횡단하는지를 설명한다. 이를 위해, 각 v-그룹 포리스트마다, 우선, 그 토폴로지를 고려함으로써 레벨 l에서 분석할 후보 페이지들의 후보를 결정한다. 이어서, 데이터 그래프 횡단을 v-그룹 포리스트마다 반복하지 않고 한 번에 수행하기 위해서 각 레벨 에서 분석할 페이지들의 세트는 모든 v-그룹 포리스트의 각 레벨 노드들에 대한 후보 페이지들의 세트들의 합집합(union)이어야 한다.

[0078] 도 6에는 두 개의 v-그룹 포리스트인 vgf_1과 vgf_2를 이용함으로써 방문된 페이지들을 도시한다. 제1 레벨에서, 데이터베이스의 임의의 페이지(p_0 ~ p_3)를 매칭할 수 있다. 제1 레벨에서의 각 페이지마다, 해당 레벨과 다음 레벨 모두의 v-그룹 포리스트들의 토폴로지를 고려하여 상기 다음 레벨의 페이지들에 액세스할 필요가 있다. 제1 v-그룹 포리스트만을 고려한다면, 제1 레벨의 페이지로부터 연결된 페이지들에만 액세스하면 된다. 그러나, 제2 v-그룹 포리스트를 고려한다면, 제1 레벨에서 검색된 페이지마다 모든 페이지들을 고려할 필요가 있어서, 데카르트 곱이 발생한다. 페이지들 사이의 점선이 데카르트 곱을 나타낸다는 점에 주목한다. 따라서, 제2 레벨에서 검색할 페이지들의 세트는 양측 v-그룹 포리스트로부터의 후보 페이지들의 합집합이어야 한다. 이러한 경우에도, 각 v-그룹 시퀀스의 전체 순서를 여전히 이용할 수 있다는 점에 주목한다. 따라서, 이 예에서, 레벨 1의 p_0는 레벨 2의 p_1, p_2, p_3와 매칭되지 않는 한편, 레벨 1의 p_3는 p_0 ~ p_3와 매칭된다.

[0079] 도 7은 글로벌 매칭 순서(1, 2, 3)으로부터 도출된 vgf_3과 vgf_4를 이용함으로써 방문된 페이지들을 도시한다. 도 6과는 대조적으로, 데이터 그래프를 횡단하는 경우 데카르트 곱이 필요 없다.

[0080] 서로 다른 매칭 순서들은 데카르트 곱의 서로 다른 개수를 생성할 수 있다. 따라서, 모든 가능한 매칭 순서들을 열거하고, 데카르트 곱들의 최소 개수를 생성하는 매칭 순서를 선택한다. 적색 질의 정점들의 개수가 데이터 그래프 정점들의 개수에 비해 매우 적으므로, 이러한 열거 비용은 서브그래프 열거 비용에 비하면 무시할만하다.

[0081] 도 8 내지 도 10은 각각 v-그룹 포리스트, 각 레벨에서의 현재 정점 윈도우 및 현재 페이지 윈도우의 일실시 구성도이다.

[0082] 도 8을 참조하면 버퍼 프레임의 개수는 통상적으로 디스크 페이지의 총 개수보다 작으므로, 한번에 페이지들의 서브세트만을 로딩할 수 있다. 정점들/페이지들을 레벨 단위로 검색함으로써 v-그룹 포리스트에 기초하여 데이터 그래프를 횡단한다. 두 개의 중요한 개념인 현재 정점/페이지 윈도우 및 후보 정점/페이지 시퀀스를 정의한다.

[0083] vgs_i라는 용어는 i번째 v-그룹 시퀀스를 나타내고, $\{vgs_i\}$는 모든 v-그룹 시퀀스들의 세트이다. vgf_i라는 용어는 vgs_i에 대한 v-그룹 포리스트를 나타내고, $\{vgf_i\}$는 모든 v-그룹 포리스트의 세트이다. $vgf_i[j]$는 vgf_i의 레벨 j에서의 노드를 나타낸다.

[0084] v-그룹 포리스트의 각 레벨에서, 우선, 후보 정점 시퀀스 및 후보 페이지 시퀀스를 식별할 필요가 있다. 이어서, (마지막 윈도우를 제외한) 각 윈도우가 동일한 개수의 디스크 I/O를 발생시키도록 후보 시퀀스를 가변-크기의 서로소 윈도우들로 분할한다. 후보 시퀀스의 정점들/페이지들 중 일부는 버퍼에 이미 존재할 수 있으므로, 서로소 윈도우들의 크기는 가변될 수 있다. 일단 서로소 윈도우들이 식별되면, 이들을 하나씩 반복할 수 있다. 처리를 위해 버퍼에 현재 있는 정점/페이지 윈도우를 현재 정점 윈도우(CVW) 및 현재 페이지 윈도우(CPW)라 칭한다. 이제, $vgf_i[j]$에 대한 후보 정점/페이지 시퀀스들(CVS 및 CPS)을, $vgf_i[j]$가 속하는 트리의 루트 노드에 대응하는 현재 정점/페이지 윈도우로부터 도달가능한 정점들/페이지들의 시퀀스라고 정의한다.

[0085] 데이터 그래프는 5개의 페이지 p_0 내지 p_4에 저장되는 한편, 4개의 메모리 버퍼 프레임을 갖고 있다. 부분 차수 ($u_1 < u_2 < u_3$)로 인해, 이에 따라 하나의 v-그룹 시퀀스 및 하나의 v-그룹 포리스트 vgf_1만을 갖는다. 도 9를 참고하면, 레벨 1에서, 모든 정점들을 검색할 필요가 있다. 따라서, cvs_1로 표시된 후보 정점 시퀀스 (v_1, v_2,...,v_9)인 한편, (cps_1로 표시된) 후보 페이지 시퀀스는 (p_0, p_1,...,p_4)이다. 제1 레벨에 대하여 두 개의 메모리 프레임이 할당된다고 가정한다. 이에 따라, 첫 번째 두 개의 페이지 p_0와 p_1을 버퍼에 로딩한다. 따라서, 현재 페이지 윈도우(cpw_1)는 (p_0, p_1)인 한편, 현재 정점 윈도우는 (v_1, v_2, v_3, v_4)이다. 일단 p_0와 p_1이 로딩되면, cps_2와 cvs_2를 식별할 수 있다. 여기서, cps_2는 (p_0, p_1, p_2, p_3)이고, cvs_2는 (v_2, v_3, v_4, v_5, v_7)이다. 제2 레벨에 대하여 하나의 메모리 프레임을 할당한다고 가정한다. 이에 따라, 제2 레벨에서의 현재 페이지 윈도우(cpw_2)는 (p_0, p_1, p_2)인 한편, 제2 페이지 윈도우는 p_3이다. cpw_2의 p_0와 p_1은 버퍼에 이미 존재하고 있다. cpw_2를 로딩한 후, 최종 레벨의 후보 정점/페이지 시퀀스를 식별할 수 있다. 도 10은, $w_{1,1}$, $w_{2,2}$, $w_{3,1}$이 현재 페이지 윈도우들로서 선택된 경우 각 레벨의 CPS, CVS, CPW, 및 CVW를 도시한다.

[0086] 일반적으로, 다수의 v-그룹 포리스트들이 존재할 수 있으므로, 모든 후보 시퀀스들을 병합하고 병합된 시퀀스를 서로소 윈도우들로 분할한다. 즉, 라인 7과 같이 레벨 1에서 현재 병합된 정점/페이지 윈도우인 mvw_1 및 mpw_1을 얻는다. 병합된 시퀀스를 이용하여 다음 레벨에서의 후보 시퀀스를 얻는 경우, 이에 따라 그 후보 시퀀스를 병합하기 전 시퀀스들로 분할하고 그러한 시퀀스들을 다수의 v-그룹 포리스트들에 각각 할당한다.

[0087] 도 11 내지 도 14는 각각 각 레벨에서의 병합된 정점/페이지 및 이들의 윈도우들의 일실시예도이다. 데이터 그래프는 도 8의 g와 동일하다. 도 11에 도시한 바와 같이 두 개의 v-그룹 포리스트 vgf_1과 vgf_2를 갖는다. 도 12와 도 13은 각 v-그룹 포리스트에 대한 CPS와 CVS, 및 이들의 현재 페이지 윈도우를 나타낸다. 도 14를 참고하면 mvs_i로 표시된 i번째 레벨에 대한 병합된 정점 시퀀스는 $cvs_{1,i}$와 $cvs_{2,i}$를 이용하는 병합된 시퀀스이다. mvs_i로 표시된 i번째 레벨에 대한 병합된 페이지 시퀀스는 $cps_{1,i}$와 $cps_{2,i}$를 이용하는 병합된 시퀀스이다. 현재 병합된 페이지 윈도우(mpw_1)은 (p_0, p_1)인 한편, 현재 병합된 정점 윈도우(mvw_i)는 (v_1, v_2, v_3, v_4)이다. 일단 p_0와 p_1이 로딩되면, $cps_{1,2}$, $cps_{2,2}$, $cps_{2,3}$를 식별할 수 있다. 이러한 세 개의 후보 페이지 시퀀스는 이 경우 (p_0, p_1, p_2, p_3)와 동일하다. 또한, $cps_{1,2}$, $cps_{2,2}$, $cps_{2,3}$을 (v_2, v_3, v_4, v_5, v_7)으로서 식별할 수 있다. 제2 레벨에 대하여 하나의 메모리 프레임을 할당한다고 가정한다. 이에 따라, 현재 병합된 페이지 윈도우(mpw_2)는 (p_0, p_1, p_2)이다. mpw_2를 로딩한 후, vgf_1의 최종 레벨에서 후보 정점/페이지 시퀀스들($cvs_{1,3}$, $cps_{1,3}$)를 식별할 수 있다. 최종 레벨에 대하여 하나의 메모리 프레임을 할당한다고 가정한다. 이에 따라, 현재 병합된 페이지 윈

도우(mpw_3)는 (p_0, p_1, p_2, p_3)이다.

[0088] 라인 8 에서 라인 10의 기재와 같이 상기 mpw_1의 각 페이지마다 비동기 I/O를 요청한다. 비동기 I/O를 요청할 때 콜백 함수를 패스할 수 있다는 점에 주목한다.

[0089] 여기서, 요청된 페이지가 버퍼에 로딩되자마자 콜백 함수 ComputeCandidateSequences(·)를 인보크(invoke)한다.

[0090] 상기 콜백 함수는 루트 노드들의 모든 자식 노드들에 대하여 후보 정점/페이지 시퀀스들을 연산한다. 모든 요청된 페이지들이 버퍼에 로딩될 때까지 대기할 필요가 있다.

[0091] 그 다음, 라인 13과 같이 현재 병합된 정점 윈도우로부터 연결된 모든 외부 서브그래프를 찾도록 재귀 함수 DelegateExternalSubgraphEnumeration(·)를 인보크한다.

[0092] 이와 같은 방법으로, 메인 스레드는 외부 서브그래프를 나머지 스레드들에 위임한다. 메인 스레드는 내부 영역에 로딩된 페이지들을 이용하여 내부 서브그래프 열거를 실행한다.

[0093] 내부 서브그래프 열거 또는 외부서브그래프 열거가 상대측 열거보다 먼저 종료되면, 이용가능한 스레드들을 미종료된 서브그래프 열거에 할당하는 스레드 모핑(Thread Morphing)을 수행한다. 상기 스레드 모핑은 "J. Kim, W. Han, S. Lee, K. Park, and H. Yu. OPT: a new framework for overlapped and parallel triangulation in large-scale graphs. In SIGMOD, pages 637-648, 2014."를 참고한다.

[0094] 이것이 바로 내부 서브그래프 열거를 외부 서브그래프 열거와 중첩하는 방식이다.

[0095] 예를 들어, 도 14에 도시한 바와 같이, 제1 열거(라인 8)의 mpw_1은 (p_0, p_1)이고, 제1 열거의 mvw_1은 (v_1, v_2, v_3, v_4)이다. 제1 열거에서, mpw_1의 두 개의 페이지 p_0와 p_1은 AsyncRead를 호출함으로써 내부 영역에 로딩된다. 요청된 페이지들의 각각이 버퍼에 로딩될 때 ComputeCandidateSequences(·)를 인보크한다. 이어서, ComputeCandidateSequences(·)는, $cvw_{1,1}$의 각 v에 대하여 $cvs_{1,2}$를 adj(v)로 연산하고 $Cps_{1,2}$를 로 연산한다. 이 함수는, 또한, $cvs_{2,2}$, $cps_{2,2}$, $cvs_{2,3}$, 및 $cps_{2,3}$를 연산한다. 이어서, DelegateExternalSubgraphEnumeration(·)를 호출함으로써 외부 서브그래프 열거를 나머지 스레드들에 위임한다. 외부 서브그래프 열거의 위임 후, 메인 스레드는 InternalSubgraphEnumeration(·)를 인보크한다. 양측 함수의 실행이 종료된 후, 페이지 p_0와 p_1은 페이지 교체를 위해 언핀(unpin)된다. 이러한 단계들은 모든 병합된 정점/페이지 윈도우들이 처리될 때까지 반복된다.

[0096] 이제, 외부 서브그래프 열거를 실행하는 DelegateExternalSubgraphEnumeration(·)를 설명한다. 이 함수에 대한 입력은, l, q_{RBI} , {vgs_i}, {vgf_i}, mvw_l, mpw_l이고, 여기서, l은 처리할 현재 레벨이다. 병합된 정점/페이지 윈도우들에 대한 루프를 갖는다. 각 반복시, 현재 병합된 정점/페이지 윈도우인 mvw_l과 mpw_l을 식별한다. l이 최종 레벨이면, 버퍼에 로딩된 데이터 정점들을 이용하여 모든 정점-레벨 매핑을 찾는 콜백 ExtVertexMapping에 의해 판독 요청을 인보크한다. 일단 모든 레벨에서의 현재 정점 윈도우가 버퍼에 로딩되면, 외부 서브그래프를 찾을 수 있다. 여기서, 모든 내부 서브그래프가 내부 영역에서 열거되므로, 중복 결과를 생성하지 않도록 모든 적색 질의 정점들을 내부 영역의 데이터 서브그래프와 매칭하는 것을 피한다. 그 외의 경우에는, mpw_l의 각 페이지마다, 콜백 함수 ComputeCandidateSequences(·)에 의해 비동기 I/O를 요청한다. 이어서, 다음 레벨에 대하여 DelegateExternalSubgraphEnumeration(·)를 재귀적으로 인보크한다.

[0097] 예를 들어, 도 14에서 레벨 l이 2인 경우, 현재 병합된 정점/페이지 시퀀스인 mpw_2(=(p_0, p_1, p_2)) 및 mvw_2(=(v_2, v_3, v_4, v_5))를 연산한다. 이어서, p_0, p_1, p_2에 대하여 콜백 함수 ComputeCandidateSequences(·)에 의해 세 개의 비동기 판독을 요청한다. 외부 서브그래프 열거를 이용가능한 스레드들에 재귀적으로 위임한다. DelegateExternalSubgraphEnumeration(·)이 레벨 3(이 예에서의 최종 레벨)에서 호출되면, 레벨 3에서 로딩된 각 페이지부터 시작하여 외부 서브그래프들을 찾을 수 있도록 다른 호출 함수 ExtVertexMapping(·)에 의해 비동기 판독을 요청한다.

[0098] 이제, 본 발명에 따른 디스크의 I/O 비용을 분석한다. |E|를 데이터 그래프의 간선의 개수, M을 메모리 버퍼 크기, B를 페이지 크기로 정한다. 각 간선이 하나의 메모리 워드에 저장되고, M은 ($|V_R|$-1)개의 영역들로 균등하게 분할된다고 가정한다. 최종 레벨에서 페이지를 스캔하는 데 상수 개(스레드의 개수)의 메모리 프레임만이 필요하다는 점에 주목한다. 이러한 추가 메모리 프레임은 점근적 분석에서 고려되지 않는다. 각 레벨에서 로딩된 페이지들로부터 연결된 페이지들에만 액세스하는 경우가 있기 때문에 감소 인자가 추가될 수 있다. s_j는 레벨 j에 대한 평균 감소 인자 를 나타낸다. 이어서, 레벨 1에서, 본 발명은 아래의 수학식1을 필요로 한다.

수학식 1

$$s_1(=l) \times \frac{|E|}{\frac{M}{|V_R|-1}} \times s_2 \times \frac{|E|}{\frac{M}{|V_R|-1}} \times \ldots \times s_l \times \frac{|E|}{B}\ disk\ I/Os.$$

[0099]

[0100] 따라서, 총 I/O 비용은 다음의 수학식 2와 같이 각 레벨에 대한 디스크 I/O들의 합이다.

수학식 2

$$\sum_{1 \le l \le |V_R|-1} \prod_{i=1}^{l} s_i \times \left(\frac{E}{\frac{M}{|V_R|-1}}\right)^{l-1} \times \frac{|E|}{B}$$

[0101]

[0102] 도 15는 q_1과 q_4를 처리하는 과정에서 본 발명의 메모리 버퍼 사이즈에 따른 성능을 측정한 결과표이다.

[0103] 본 발명은 버퍼 사이즈는 그래프 사이즈의 5% 이내로 매우 작은 경우를 제외하고 일정한 성능을 보인다. 특히 q_1에서는 다양한 버퍼사이즈에서 1배 정도로 유지되며, 질의 q_4는 버퍼 크기가 25에서 5%로 5배 감소할 때 1.45 내지 1.59배의 성능 저하를 보인다.

[0104] 전반적으로 본 발명은 제한된 메모리에서도 강력한 성능을 발휘한다.

[0105] 도 16은 실행 스레드의 수를 다양화하여 스피드업을 측정한 결과이다.

[0106] 이와 같은 실험을 통해 CPU의 병렬 처리 효과를 평가할 수 있다. 스레드의 수를 1에서 6까지 증가시켰을 때 q_1은 5.4, q4는 5.35로 스피드 업이 되었다.

[0107] 이러한 결과는 본 발명이 멀티 코어 병렬 처리를 거의 완벽하게 사용할 수 있음을 뜻한다.

[0108] 도 17은 본 발명과 종래의 TWINTWIGJOIN 방식에 대하여 데이터셋에서 경과시간을 비교한 그래프이다. 데이터셋은 wepGoogle(WG), Wikitalk(WT), LiveJournal(LJ) 등을 사용하였다.

[0109] 도 17의 결과와 같이 본 발명은 모든 데이터셋에서 종래의 TWINTWIGJOIN에 비하여 향상된 성능을 나타낸다. 최

대 205.85배 더 적은 메모리를 사용한다.

[0110] 도 18은 q_1 내지 q_5 각각을 본 발명과 종래 TWINTWIGJOIN 방식으로 특정 데이터셋에서 처리하였을 때 경과시간을 비교한 표이다.

[0111] 도 18을 참조하면 본 발명은 TWINTWIGJOIN방식에 비하여 q_1,q_2,q_3,q_4에서 각각 58.47배, 799.75배, 253.68배, 205.85배 더 짧은 경과시간을 나타낸다. 특히 본 발명은 q_5를 모든 데이터셋에서 처리할 수 있으나, 비용이 많이 소모되는 TWINTWIGJOIN 방식에서는 처리하지 못했다.

[0112] 도 19는 본 발명과 TWINTWIGJOIN 방식을 사용하여 그래프 크기의 변화에 따른 경과시간을 측정한 그래프이다.

[0113] 도 19를 참조하면 그래프 크기를 20%, 40%, 60% 및 80%로 변경하면서, q1 내지 q4에 대한 처리를 각각 수행하였으며, 모든 그래프 크기에서 본 발명은 TWINTWIGJOIN 방식에 비하여 월등한 성능 차이를 보였다.

[0114] 본 발명과 TWINTWIGJOIN 방식의 성능 차이는 그래프의 크기가 커질수록 더 커졌다. 예를 들어 q_1에서 성능 차이는 그래프의 크기가 커질수록 19.5, 17.9, 20.5, 23.2, 23.7이 되며, TWINTWIGJOIN 방식은 일부에서 처리하지 못하는 경우가 발생하였다.

[0115] 도 20은 본 발명과 TWINTWIGJOIN 방식에 PSgL을 추가하여 다수의 데이터셋에서 경과시간을 측정한 그래프이다.

[0116] 이에 도시한 바와 같이 본 발명은 TWINTWIGJOIN 방식뿐만 아니라 PSgL에 비해서도 월등한 성능을 보인다.

[0117] 도 21은 도 20의 조건으로 q_1 내지 q_5를 처리할 때 경과시간 측정 그래프이다.

[0118] 도 21을 참조하면 본 발명은 q_1 내지 q_4 각각에 대하여 TWINTWIGJOIN 방식에 비하여 최대 179.80, 766.31, 81.82 및 60.94배 개선되었다. TWINTWIGJOIN 방식은 q_5를 처리할 수 없었다.

[0119] 또한 본 발명은 q_1 내지 q_4 각각에 대하여 PSgL 방식에 비하여 최대 7.62, 7.88, 26.24 및 14.25배 개선되었다. PSgL 방식은 LJ 데이터셋에서 q_2와 q_3를 처리할 수 없었으며, 모든 데이터셋에서 q_5를 처리할 수 없었다.

[0120] 도 22는 도 20의 조건으로 그래프 크기의 변화에 따른 경과시간을 측정한 그래프이다.

[0121] 도 22를 참조하면 본 발명은 PSgL과 TWINTWIGJOIN 방식에 비해 최대 7.71과 4.73배 향상된 성능을 보인다. PSgL은 q_1에서 그래프 크기가 80% 및 100%일 때 처리하지 못했으며, q_4에서는 60% 이상일 때 처리하지 못했다.

[0122] 전술한 바와 같이 본 발명에 대하여 바람직한 실시예를 들어 상세히 설명하였지만, 본 발명은 전술한 실시예들에 한정되는 것이 아니고, 특허청구범위와 발명의 상세한 설명 및 첨부한 도면의 범위 안에서 여러 가지로 변형하여 실시하는 것이 가능하고 이 또한 본 발명에 속한다.

도면

도면1

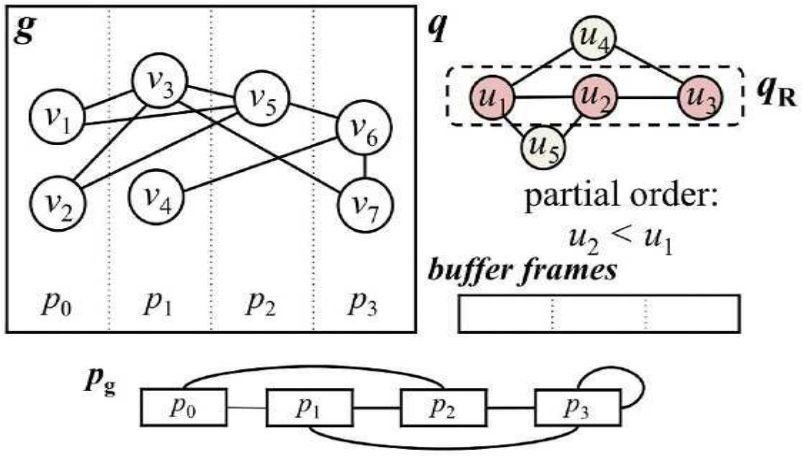
Algorithm 1. DUALSIM

Input: A data graph g, A query graph q

/* 1. Preparation step */
1: $PO \leftarrow$ FINDPARTIALORDERS(q);
2: $\langle q_{RBI}, q_R \rangle \leftarrow$ GENERATERBIQUERYGRAPH(q, PO);
3: $\{vgs_i\} \leftarrow$ FINDVGROUPSEQUENCES(q_R, PO);
4: $mo_g \leftarrow$ FINDGLOBALMATCHINGORDER($\{vgs_i\}$));
5: $\{vgf_i\} \leftarrow$ BUILDVGROUPFORESTS($\{vgs_i\}, mo_g$);

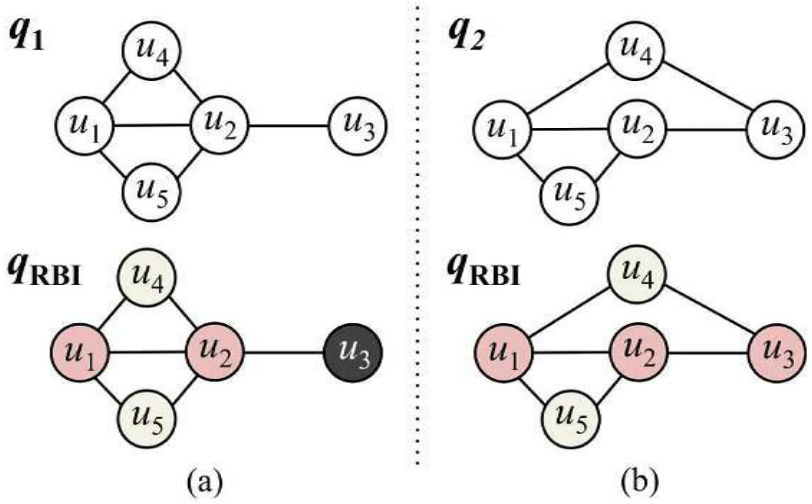
/* 2. Execution step */
6: INITIALIZECANDIDATESEQUENCES(root nodes in $\{vgf_i\}$);
7: **foreach** (merged vertex/page window (mvw_1, mpw_1) from $\{vgf_i[1]\}$) **do**
8: **foreach** (page id $pid \in mpw_1$) **do**
9: AsyncRead(pid, COMPUTECANDIDATESEQUENCES(pid, $\{cvw_{i,1}\}$, all child nodes of $\{vgf_i[1]\}$);
10: **end**
11: wait until COMPUTECANDIDATESEQUENCES executions are finished
12: $level \leftarrow 2$;
13: DELEGATEEXTERNALSUBGRAPHENUMERATION($level$, q_{RBI}, $\{vgs_i\}$, $\{vgf_i\}$, $\{mvw_j\}$, $\{mpw_j\}$);
14: INTERNALSUBGRAPHENUMERATION(mvw_1, mpw_1);
15: UNPINPAGES(mpw_1);
16: CLEARCANDIDATESEQUENCES(the children of $\{vgf_i[1]\}$);
17: **end**

도면2

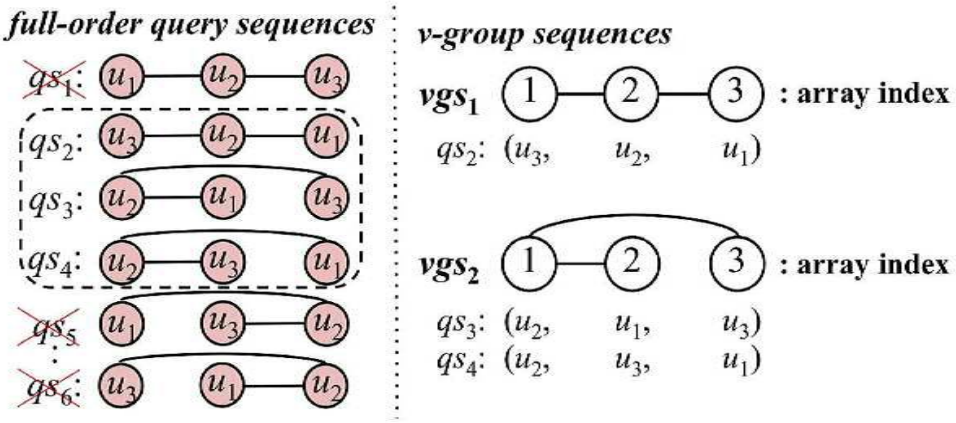


Data graph g, query graph q, and page graph p_g.

도면3



도면4



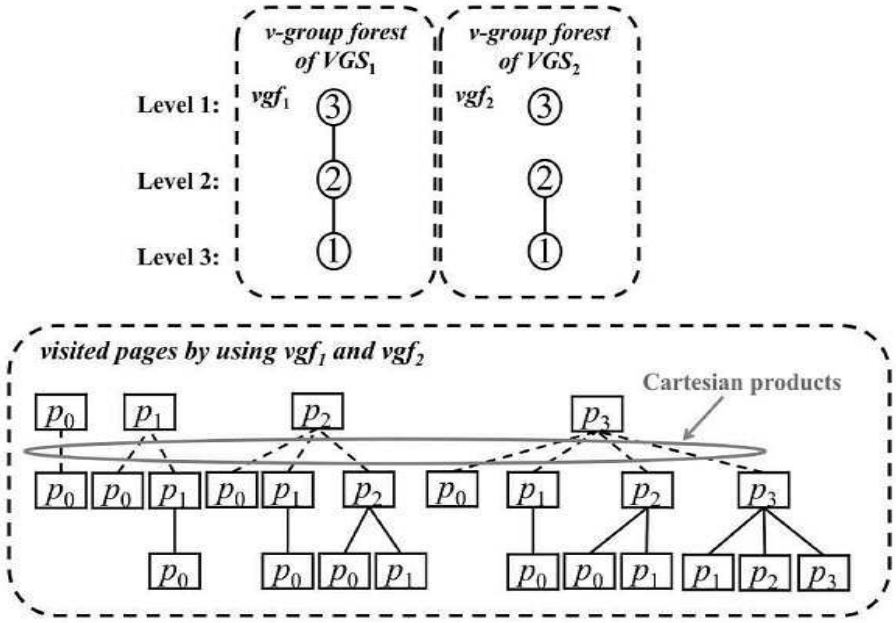
Full-order query sequences and *v*-group sequences.

도면5

page sequences	matching v-group sequences
①. (p_0, p_1, p_1)	vgs_2
②. (p_0, p_1, p_2)	vgs_1, vgs_2
③. (p_0, p_1, p_3)	vgs_1
④. (p_0, p_2, p_2)	vgs_2
⑤. (p_0, p_2, p_3)	vgs_1
⑥. (p_1, p_2, p_2)	vgs_2
⑦. (p_1, p_2, p_3)	vgs_1, vgs_2
⑧. (p_1, p_3, p_3)	vgs_2
⑨. (p_2, p_3, p_3)	vgs_1, vgs_2
⑩. (p_3, p_3, p_3)	vgs_2

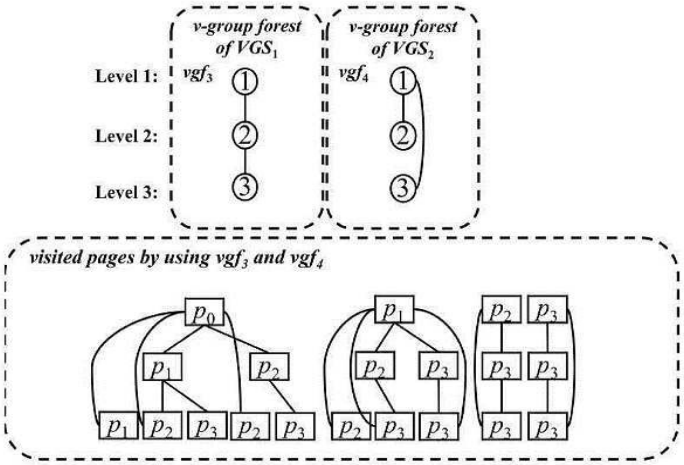
(c) A page mapping example.

도면6



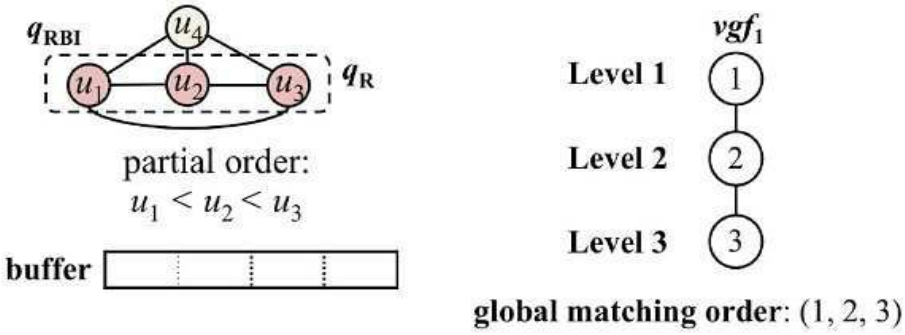
Global matching order: (3, 2, 1).

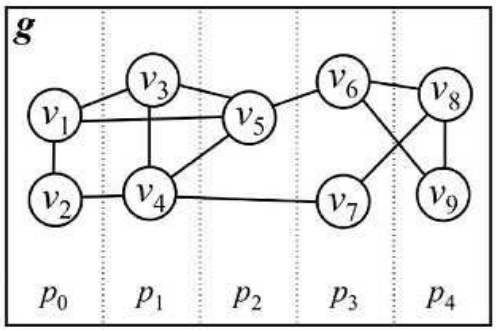
도면7



Global matching order: (1, 2, 3).

도면8





Data structures.

도면9

Level 1 *disjoint windows*

	$w_{1,1}$	$w_{1,2}$	$w_{1,3}$	
cps_1	$(p_0, p_1,$	$p_2, p_3,$	$p_4)$	cpw_1 $[p_0, p_1]$
cvs_1	$(v_1, v_2, v_3, v_4,$	$v_5, v_6, v_7,$	$v_8, v_9)$	cvw_1 $[v_1, v_2, v_3, v_4]$

Level 2

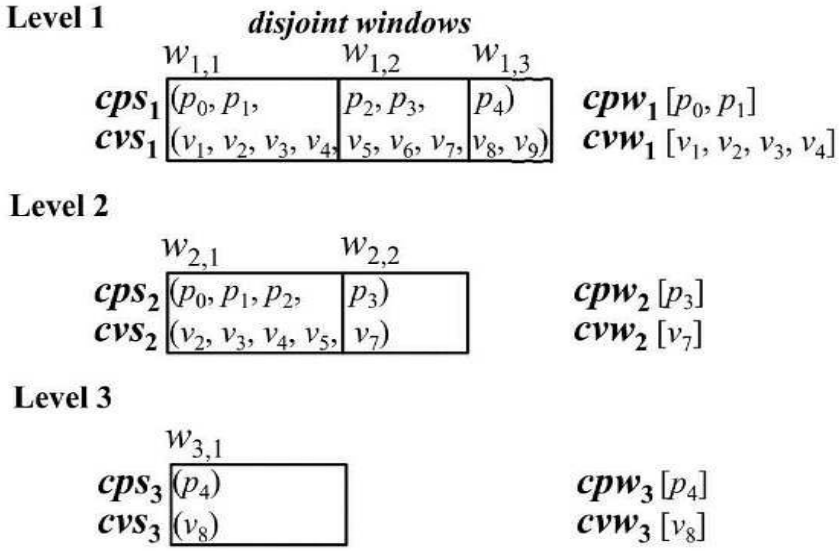
	$w_{2,1}$	$w_{2,2}$	
cps_2	$(p_0, p_1, p_2,$	$p_3)$	cpw_2 $[p_0, p_1, p_2]$
cvs_2	$(v_2, v_3, v_4, v_5,$	$v_7)$	cvw_2 $[v_2, v_3, v_4, v_5]$

Level 3

	$w_{3,1}$	
cps_3	(p_1, p_2, p_3)	cpw_3 $[p_1, p_2, p_3]$
cvs_3	(v_4, v_5, v_6, v_7)	cvw_3 $[v_4, v_5, v_6, v_7]$

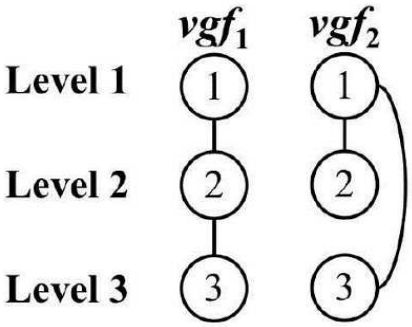
When current page windows for $w_{1,1}$, $w_{2,1}$, $w_{3,1}$ are chosen.

도면10



When current page windows for $w_{1,1}$, $w_{2,2}$, $w_{3,1}$ are chosen.

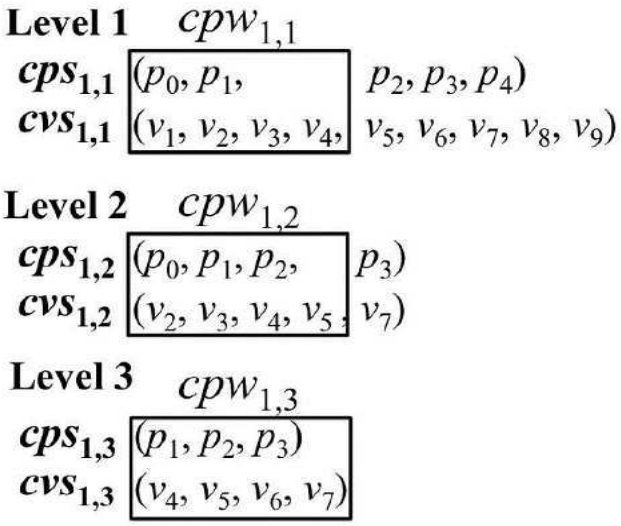
도면11



global matching order: (1, 2, 3)

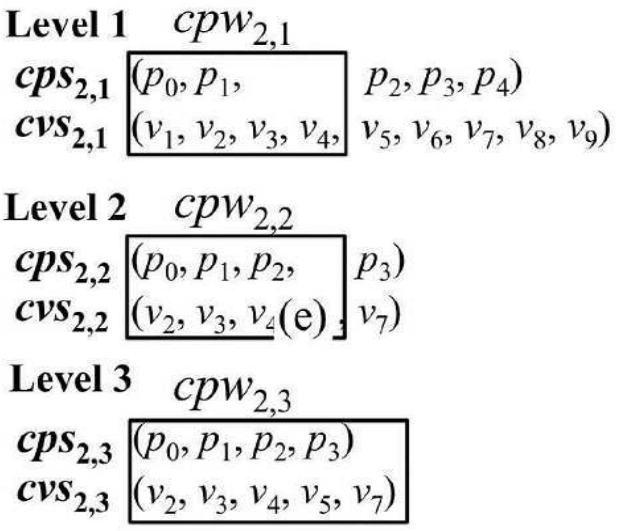
V-group forests vgf_1, vgf_2.

도면12



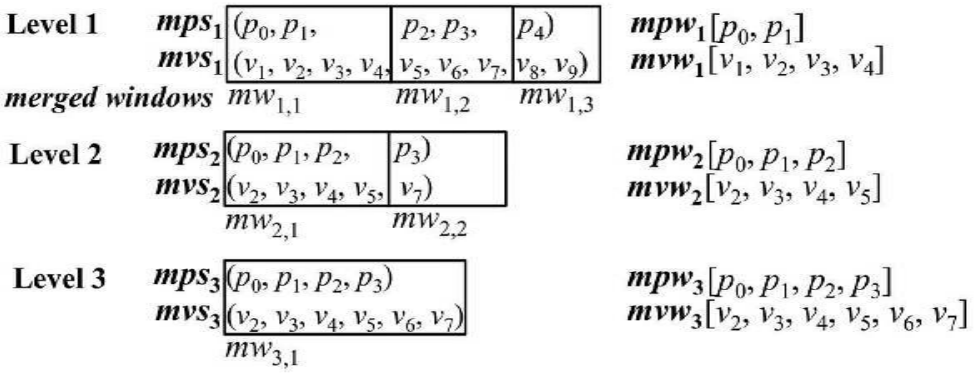
Candidate page/vertex sequences for vgf_1.

도면13



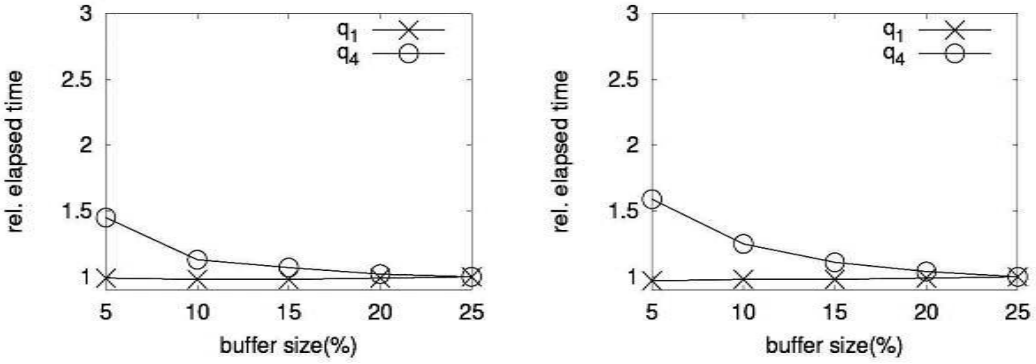
Candidate page/vertex sequences for vgf_2.

도면14



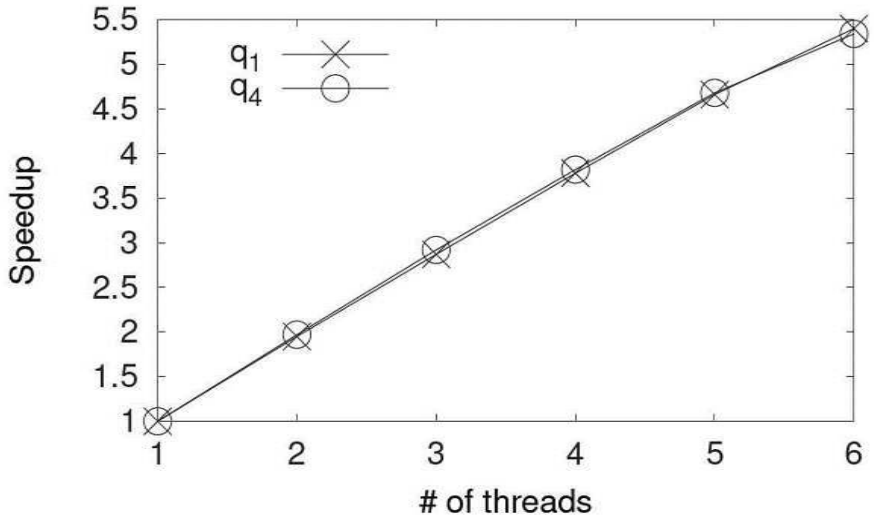
Merged vertex/page sequences and their current windows.

도면15

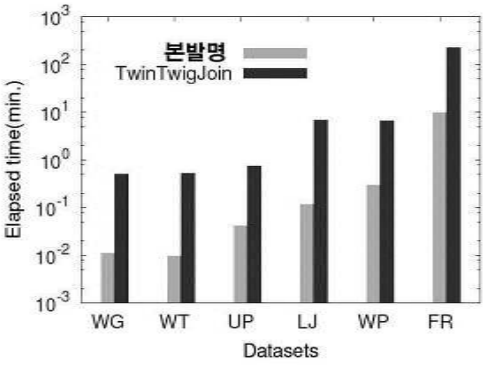


LJ. OK.

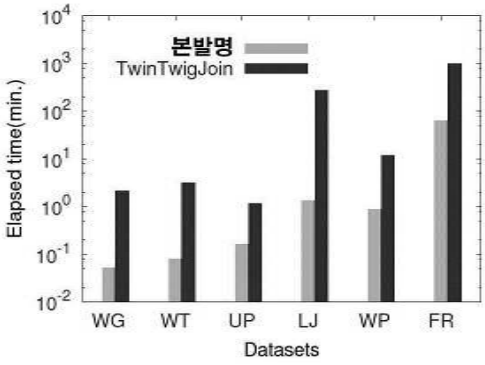
도면16



도면17

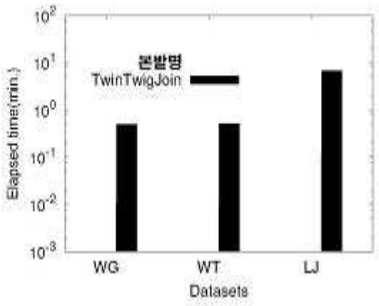


Query q_1.

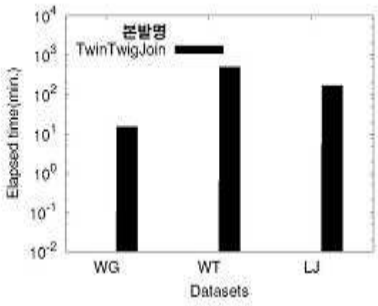


Query q_4.

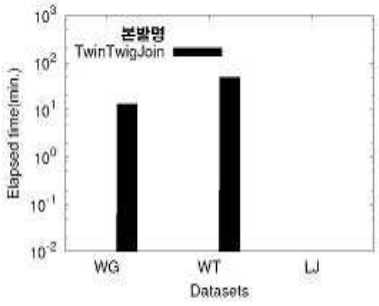
도면18



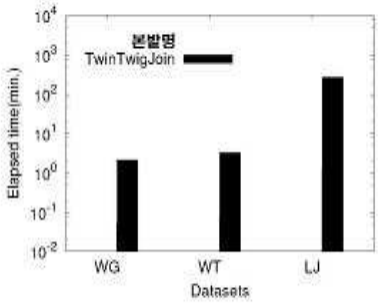
Query q_1.



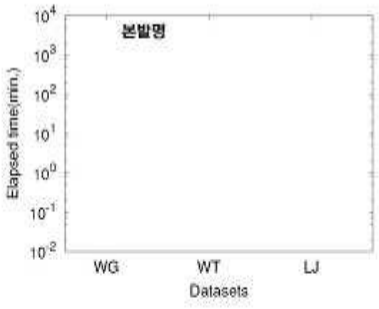
Query q_2.



Query q_3.

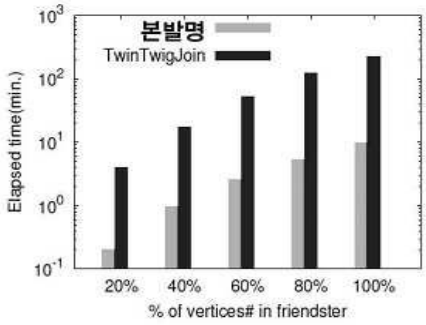


Query q_4.

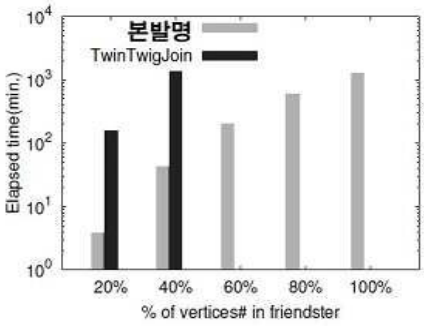


Query q_5.

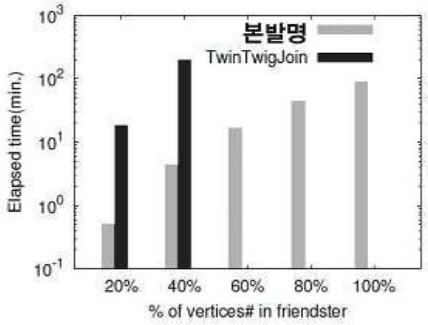
도면19



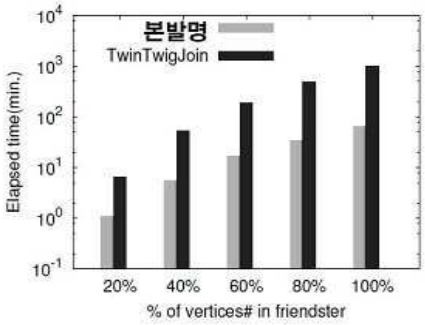
Query q_1.



Query q_2.

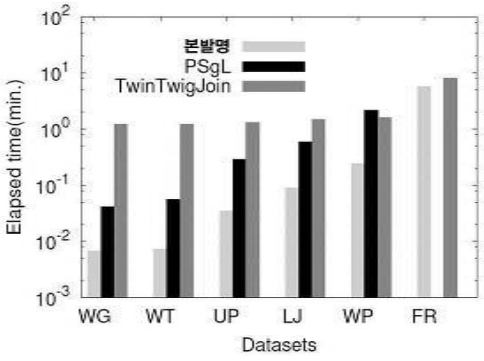


Query q_3.

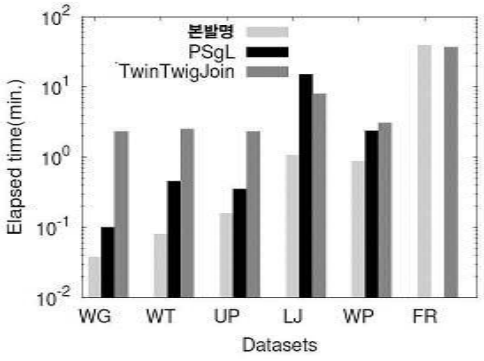


Query q_4.

도면20

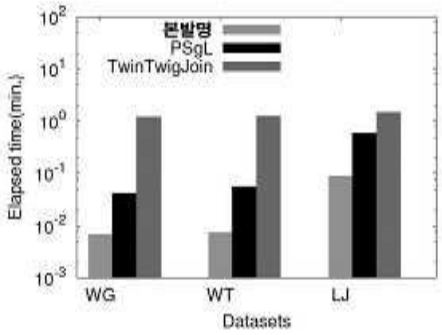


Query q_1.

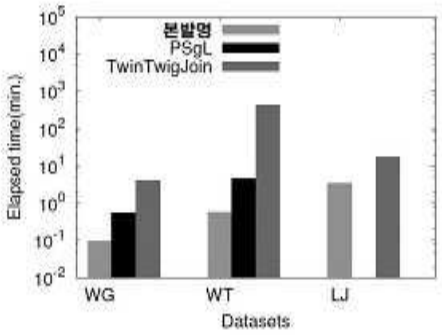


Query q_4.

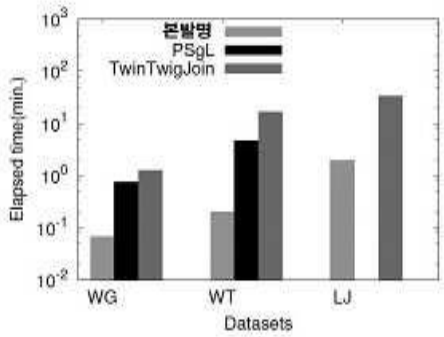
도면21



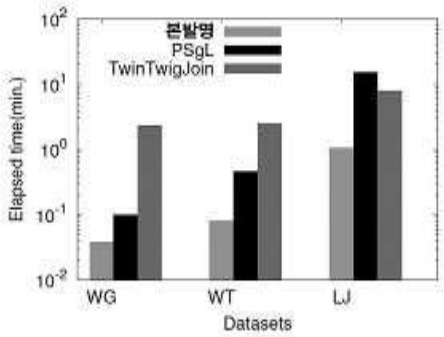
Query q_1.



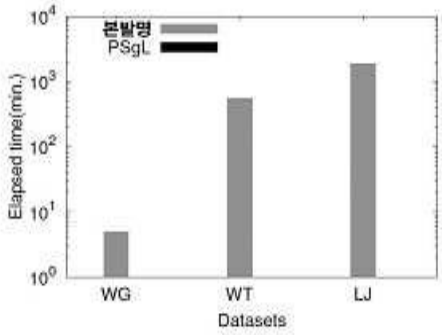
Query q_2.



Query q_3.

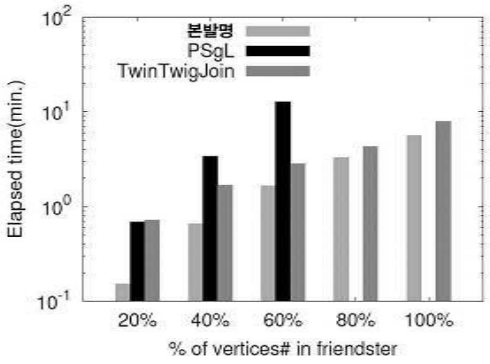


Query q_4.

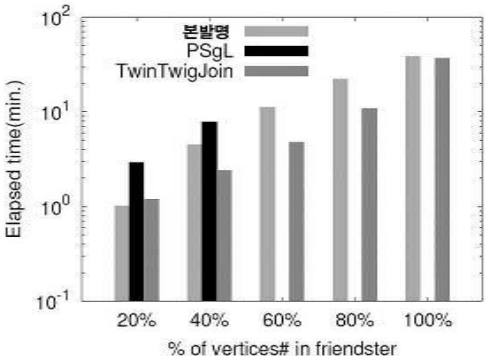


Query q_5.

도면22



Query q_1.



Query q_4.