



(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2022-0074725
(43) 공개일자 2022년06월03일

- (51) 국제특허분류(Int. Cl.)
G16B 45/00 (2019.01) G06F 7/58 (2006.01)
G06F 9/30 (2018.01) G16B 40/20 (2019.01)
- (52) CPC특허분류
G16B 45/00 (2019.02)
G06F 7/58 (2018.05)
- (21) 출원번호 10-2021-0139851
(22) 출원일자 2021년10월20일
심사청구일자 2021년10월20일
- (30) 우선권주장
1020200163219 2020년11월27일 대한민국(KR)

- (71) 출원인
한양대학교 산학협력단
서울특별시 성동구 왕십리로 222(행당동, 한양대 학교내)
울산과학기술원
울산광역시 울주군 언양읍 유니스트길 50
- (72) 발명자
서지원
서울특별시 성동구 왕십리로 222
최강
서울특별시 성동구 왕십리로 222
(뒷면에 계속)
- (74) 대리인
심경식, 홍성욱

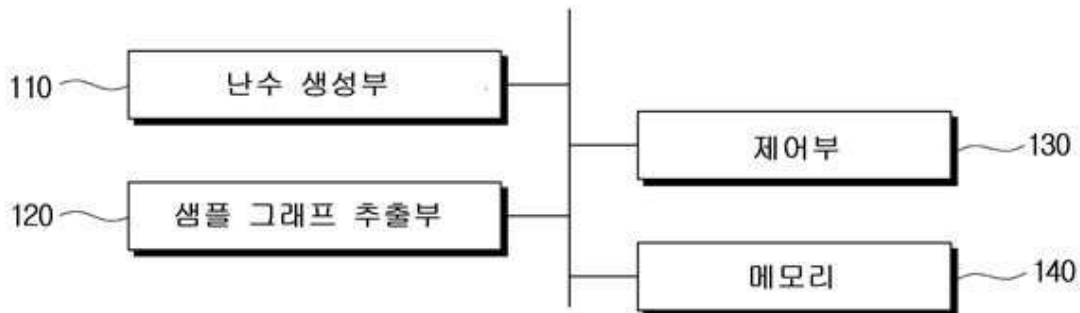
전체 청구항 수 : 총 13 항

(54) 발명의 명칭 불확실성 그래프 분석을 위한 샘플링 방법, 캐싱 방법 및 샘플링 장치

(57) 요약

본 발명의 일 실시예에 따른 불확실성 그래프 분석을 위한 샘플링 장치의 샘플링 방법으로서, 네트워크 인스턴스에 기초하여 난수 생성 기법을 통해 복수개의 샘플 그래프를 추출하는 단계; 및 샘플링된 그래프의 버텍스(vertex) 데이터와 엣지(Edge)의 확률을 메모리에 저장하는 단계를 포함하고, 상기 난수 생성 기법은 고정된 특정값을 난수의 씨드(seed)로 설정하여 난수를 생성하는 것을 특징으로 한다.

대표도 - 도1



(52) CPC특허분류

G06F 9/30047 (2013.01)

G06F 9/30138 (2013.01)

G16B 40/20 (2019.02)

(72) 발명자

노삼혁

울산광역시 울주군 언양읍 유니스트길 50

이은재

울산광역시 울주군 언양읍 유니스트길 50

이 발명을 지원한 국가연구개발사업

과제고유번호 1711103335

과제번호 2013-2-00109-008

부처명 과학기술정보통신부

과제관리(전문)기관명 정보통신기획평가원

연구사업명 한국연구재단 부설 정보통신기획평가원 / 정보통신방송 연구개발사업 / 혁신성장동
력프로젝트(과기정통부)사업-인공지능사업(통합이지 바로)

연구과제명 (엑소브레인-2세부) WiseKB: 빅데이터 이해 기반 자가학습형 지식베이스 및 추론 기
술 개발

기여율 50/100

과제수행기관명 주식회사 솔트룩스

연구기간 2020.01.01 ~ 2020.12.31

이 발명을 지원한 국가연구개발사업

과제고유번호 1711116829

과제번호 2020-0-01373-001

부처명 과학기술정보통신부

과제관리(전문)기관명 정보통신기획평가원

연구사업명 한국연구재단 부설 정보통신기획평가원 / 정보통신방송 연구개발사업 / 정보통신방
송혁신 인재양성-교육훈련지원사업

연구과제명 인공지능대학원지원(한양대학교)

기여율 30/100

과제수행기관명 한양대학교산학협력단

연구기간 2020.04.01 ~ 2020.12.31

이 발명을 지원한 국가연구개발사업

과제고유번호 1711104146

과제번호 2016M3C4A7952635

부처명 과학기술정보통신부

과제관리(전문)기관명 한국연구재단

연구사업명 원천기술개발사업(차세대정보컴퓨팅기술개발사업)

연구과제명 기계학습 프레임워크를 활용한 Genome 분석 소프트웨어 개발

기여율 20/100

과제수행기관명 울산과학기술원

연구기간 2020.02.01 ~ 2021.06.30

명세서

청구범위

청구항 1

불확실성 그래프 분석을 위한 샘플링 장치의 샘플링 방법으로서,

네트워크 인스턴스에 기초하여 난수 생성 기법을 통해 복수개의 샘플 그래프를 추출하는 단계; 및

샘플링된 그래프의 버텍스(vertex) 데이터와 엣지(Edge)의 확률을 메모리에 저장하는 단계

를 포함하고,

상기 난수 생성 기법은 고정된 특정값을 난수의 씨드(seed)로 설정하여 난수를 생성하는 것을 특징으로 하는 샘플링 방법.

청구항 2

제1항에 있어서,

상기 고정된 특정값은 각 엣지를 구성하는 소스 버텍스(source vertex)와 타겟 버텍스(target vertex)의 ID의 해시값인 것을 특징으로 하는 샘플링 방법.

청구항 3

제1항에 있어서,

미리 정해진 그래프 분석 알고리즘에 따른 그래프 분석을 위해 상기 난수 생성 기법을 통해 상기 고정된 특정값을 사용하여 동일한 난수를 생성하고, 상기 난수와 엣지의 확률에 기초하여 버텍스 사이의 연결여부를 결정하는 것을 특징으로 하는 샘플링 방법.

청구항 4

불확실성 그래프 분석을 위한 샘플링 장치의 캐싱 방법으로서,

네트워크 인스턴스에 기초하여 난수 생성을 통해 복수개의 샘플 그래프를 추출하는 단계;

샘플링된 그래프의 버텍스(vertex) 데이터와 엣지(Edge)의 확률을 메모리에 저장하는 단계;

버텍스의 활성여부에 기초하여 버텍스 사용 우선순위를 설정하는 단계; 및

상기 버텍스 사용 우선순위에 기초하여 미리 설정된 개수의 버텍스 데이터를 캐시하는 단계

를 포함하는 것을 특징으로 하는 캐싱 방법.

청구항 5

제4항에 있어서,

버텍스의 패스카운트(pathcount)의 수의 크기가 클 수록 높은 우선순위를 부여하는 것을 특징으로 하는 캐싱 방법.

청구항 6

제4항에 있어서,

상기 캐시에 저장된 버텍스 데이터에 기초하여 미리 저장된 버텍스 프로그램을 실행하는 단계를 더 포함하는 캐싱 방법.

청구항 7

제6항에 있어서,

각 버텍스에 대하여 버텍스 프로그램의 실행이 완료되면 실행이 완료된 버텍스를 제외하고 상기 버텍스 사용 우선순위를 재설정하는 것을 특징으로 하는 캐싱 방법.

청구항 8

불확실성 그래프 분석을 위한 샘플링 장치로서,

고정된 특정값을 난수의 씨드(seed)로 설정하여 난수 생성 기법을 통해 난수를 생성하는 난수 생성부;

네트워크 인스턴스 및 생성된 난수에 기초하여 복수개의 샘플 그래프를 추출하는 샘플 그래프 추출부; 및

샘플링된 그래프의 버텍스(vertex) 데이터와 엣지(Edge)의 확률을 저장하는 메모리

를 더 포함하는 샘플링 장치.

청구항 9

제8항에 있어서,

상기 고정된 특정값은 각 엣지를 구성하는 소스 버텍스(source vertex)와 타겟 버텍스(target vertex)의 ID의 해시값인 것을 특징으로 하는 샘플링 장치.

청구항 10

제8항에 있어서,

미리 정해진 그래프 분석 알고리즘에 따른 그래프 분석을 위해 상기 난수 생성부를 통해 상기 고정된 특정값을 씨드로 하여 동일한 난수를 생성하도록 제어하고, 상기 난수와 엣지의 확률에 기초하여 버텍스 사이의 연결여부를 결정하는 제어부;

를 더 포함하는 샘플링 장치.

청구항 11

제8항에 있어서,

버텍스의 활성여부에 기초하여 버텍스 사용 우선순위를 설정하고, 상기 버텍스 사용 우선순위에 기초하여 미리 설정된 개수의 버텍스 데이터를 캐시 메모리에 저장하는 캐시 관리부;

를 더 포함하는 샘플링 장치.

청구항 12

제11항에 있어서,

상기 캐시 관리부는 버텍스의 패스카운트(pathcount)의 수의 크기가 클 수록 높은 우선순위를 부여하는 것을 특징으로 하는 샘플링 장치.

청구항 13

제11항에 있어서,

상기 캐시 메모리에 저장된 버텍스 데이터에 기초하여 미리 저장된 버텍스 프로그램을 실행하는 제어부를 더 포함하고,

상기 캐시 관리부는 각 버텍스에 대하여 버텍스 프로그램의 실행이 완료되면 실행이 완료된 버텍스를 제외하고 상기 버텍스 사용 우선순위를 재설정하는 것을 특징으로 하는 샘플링 장치.

발명의 설명

기술 분야

[0001] 본 발명은 그래프 분석을 위한 샘플링 및 그래프 캐싱 장치에 관한 것이다.

배경 기술

[0003] 최근에는 소셜 네트워크, 생물학 네트워크, 단백질-단백질 상호작용 네트워크 등에서 개체와 개체 간의 복잡한 관계를 표현하기 위해, 그래프 데이터 구조가 주목 받고 있다. 그래프는 정점과 간선으로 표현되고, 각 정점은 개체를 나타내고, 간선은 개체 간의 관계를 나타낸다. 그래프를 이용한 표현 방식은 데이터 분석 및 데이터 마이닝 등 여러 분야에서 널리 활용되고 있다.

[0004] 많은 불확실성 그래프 알고리즘은 네트워크 인스턴스를 샘플링하고, 샘플링된 네트워크에 대해 분석을 수행한 후, 분석 결과를 집계하는 공통 구조를 가지고 있다. 이러한 알고리즘을 수행하기 위해, 샘플 그래프 데이터를 저장해야 하며, 샘플 개수에 비례하는 메모리 자원이 요구된다. 즉, 대규모 샘플링을 진행할 경우, 대용량의 메모리가 요구된다. 기존의 시스템에서는 개발자가 샘플링 기능을 추가 구현해야 하며, 이로 인한 메모리 관리 역시 구현해야 한다.

[0005] 기존 디스크 기반 그래프 분석 시스템에서는 샘플링 기능을 제공하지 않으며, 샘플 그래프의 모든 데이터를 디스크에 저장할 경우, 많은 양의 입출력(IO)가 발생하여 성능이 느려진다는 문제가 있다.

[0006] 따라서, 상술한 문제점을 해결하면서, 최적화된 메모리를 관리하는 프레임워크에 대한 필요성이 대두되고 있다.

발명의 내용

해결하려는 과제

[0008] 본 발명은 샘플 그래프 저장을 위한 메모리 관리 및 불확실성 그래프 알고리즘의 계산 오버헤드 등을 줄이기 위한 장치 및 방법을 제공하고자 한다.

[0009] 본 발명이 해결하고자 하는 과제는 이상에서 언급한 과제(들)로 제한되지 않으며, 언급되지 않은 또 다른 과제(들)는 아래의 기재로부터 당업자에게 명확하게 이해될 수 있을 것이다.

과제의 해결 수단

[0011] 상기 목적을 달성하기 위한 본 발명의 일 실시예에 따른 불확실성 그래프 분석을 위한 샘플링 장치의 샘플링 방

법은, 네트워크 인스턴스에 기초하여 난수 생성 기법을 통해 복수개의 샘플 그래프를 추출하는 단계; 및 샘플링된 그래프의 버텍스(vertex) 데이터와 엣지(Edge)의 확률을 메모리에 저장하는 단계를 포함하고, 상기 난수 생성 기법은 고정된 특정값을 난수의 씨드(seed)로 설정하여 난수를 생성하는 것을 포함한다.

- [0012] 일 실시예에서, 상기 고정된 특정값은 각 엣지를 구성하는 소스 버텍스(source vertex)와 타겟 버텍스(target vertex)의 ID의 해시값인 것을 특징으로 한다.
- [0013] 일 실시예에서, 미리 정해진 그래프 분석 알고리즘에 따른 그래프 분석을 위해 상기 난수 생성 기법을 통해 상기 고정된 특정값을 사용하여 동일한 난수를 생성하고, 상기 난수와 엣지의 확률에 기초하여 버텍스 사이의 연결여부를 결정하는 것을 특징으로 한다.
- [0014] 본 발명의 다른 실시예에 따른 불확실성 그래프 분석을 위한 샘플링 장치의 캐싱 방법은, 네트워크 인스턴스에 기초하여 난수 생성을 통해 복수개의 샘플 그래프를 추출하는 단계; 샘플링된 그래프의 버텍스(vertex) 데이터와 엣지(Edge)의 확률을 메모리에 저장하는 단계; 버텍스의 활성여부에 기초하여 버텍스 사용 우선순위를 설정하는 단계; 및 상기 버텍스 사용 우선순위에 기초하여 미리 설정된 개수의 버텍스 데이터를 캐시하는 단계를 포함하는 것을 특징으로 한다.
- [0015] 일 실시예에서, 버텍스의 패스카운트(pathcount)의 수의 크기가 클 수록 높은 우선순위를 부여하는 것을 특징으로 한다.
- [0016] 일 실시예에서, 상기 캐시에 저장된 버텍스 데이터에 기초하여 미리 저장된 버텍스 프로그램을 실행하는 것을 특징으로 한다.
- [0017] 일 실시예에서, 각 버텍스에 대하여 버텍스 프로그램의 실행이 완료되면 실행이 완료된 버텍스를 제외하고 상기 버텍스 사용 우선순위를 재설정하는 것을 특징으로 한다.
- [0018] 본 발명의 다른 실시예에 따른 불확실성 그래프 분석을 위한 샘플링 장치는, 고정된 특정값을 난수의 씨드(seed)로 설정하여 난수 생성 기법을 통해 난수를 생성하는 난수 생성부; 네트워크 인스턴스 및 생성된 난수에 기초하여 복수개의 샘플 그래프를 추출하는 샘플 그래프 추출부; 및 샘플링된 그래프의 버텍스(vertex) 데이터와 엣지(Edge)의 확률을 저장하는 메모리를 더 포함하는 것을 특징으로 한다.
- [0019] 일 실시예에서, 불확실성 그래프 분석을 위한 샘플링 장치는, 버텍스의 활성여부에 기초하여 버텍스 사용 우선순위를 설정하고, 상기 버텍스 사용 우선순위에 기초하여 미리 설정된 개수의 버텍스 데이터를 캐시 메모리에 저장하는 캐시 관리부를 더 포함한다.

발명의 효과

- [0021] 본 발명의 일 실시예에 따른 불확실성 그래프 분석을 위한 샘플링 장치는 샘플링을 별도의 개발없이, API(application programming interface)를 제공하여 샘플링을 가능하게 하며, 샘플링으로 인해 발생하는 메모리 자원 소모를 경감하며, 입출력 성능을 최적화하는 효과가 있다.
- [0022] 본 발명의 효과들은 이상에서 언급한 효과들로 제한되지 않으며, 언급되지 않은 또 다른 효과들은 청구범위의 기재로부터 당업자에게 명확하게 이해될 수 있을 것이다.

도면의 간단한 설명

- [0024] 도 1은 본 발명의 일 실시예에 따른 불확실성 그래프 분석을 위한 샘플링 장치의 구성을 나타내는 블록도이다.
- 도 2는 본 발명의 다른 실시예에 따른 불확실성 그래프 분석을 위한 샘플링 장치의 구성을 나타내는 블록도이다.
- 도 3은 본 발명의 일 실시예에 따른 불확실성 그래프 분석을 위한 샘플링 장치의 샘플링 방법을 설명하기 위한 흐름도이다.
- 도 4는 본 발명의 일 실시예에 따른 샘플 그래프이다.
- 도 5는 본 발명의 일 실시예에 따른 불확실성 그래프 분석을 위한 샘플링 장치의 캐싱 방법을 설명하기 위한 흐름도이다.

름도이다.

도 6은 본 발명의 일 실시예에 따라 사용자 정의 함수에 관한 인터페이스이다.

도 7은 본 발명의 일 실시예에 따른 벡터 프로그램이 수행되는 알고리즘의 일 예이다.

도 8은 본 발명의 일 실시예에 따른 벡터 사용 우선순위를 계산하기 위한 알고리즘의 일 예이다.

도 9는 본 발명의 일 실시예에 따른 Top-K Reliability Search 알고리즘의 일 예이다.

발명을 실시하기 위한 구체적인 내용

- [0025] 본 발명은 다양한 변경을 가할 수 있고 여러 가지 실시예를 가질 수 있는 바, 특정 실시예들을 도면에 예시하고 상세한 설명에 상세하게 설명하고자 한다. 그러나, 이는 본 발명을 특정한 실시 형태에 대해 한정하려는 것이 아니며, 본 발명의 사상 및 기술 범위에 포함되는 모든 변경, 균등물 내지 대체물을 포함하는 것으로 이해되어야 한다. 각 도면을 설명하면서 유사한 참조부호를 유사한 구성요소에 대해 사용하였다.
- [0026] 제1, 제2, A, B 등의 용어는 다양한 구성요소들을 설명하는데 사용될 수 있지만, 상기 구성요소들은 상기 용어들에 의해 한정되어서는 안 된다. 상기 용어들은 하나의 구성요소를 다른 구성요소로부터 구별하는 목적으로만 사용된다. 예를 들어, 본 발명의 권리 범위를 벗어나지 않으면서 제1 구성요소는 제2 구성요소로 명명될 수 있고, 유사하게 제2 구성요소도 제1 구성요소로 명명될 수 있다. 및/또는 이라는 용어는 복수의 관련된 기재된 항목들의 조합 또는 복수의 관련된 기재된 항목들 중의 어느 항목을 포함한다.
- [0027] 어떤 구성요소가 다른 구성요소에 "연결되어" 있다거나 "접속되어" 있다고 언급된 때에는, 그 다른 구성요소에 직접적으로 연결되어 있거나 또는 접속되어 있을 수도 있지만, 중간에 다른 구성요소가 존재할 수도 있다고 이해되어야 할 것이다. 반면에, 어떤 구성요소가 다른 구성요소에 "직접 연결되어" 있다거나 "직접 접속되어" 있다고 언급된 때에는, 중간에 다른 구성요소가 존재하지 않는 것으로 이해되어야 할 것이다.
- [0028] 본 출원에서 사용한 용어는 단지 특정한 실시예를 설명하기 위해 사용된 것으로, 본 발명을 한정하려는 의도가 아니다. 단수의 표현은 문맥상 명백하게 다르게 뜻하지 않는 한, 복수의 표현을 포함한다. 본 출원에서, "포함하다" 또는 "가지다" 등의 용어는 명세서상에 기재된 특징, 숫자, 단계, 동작, 구성요소, 부품 또는 이들을 조합한 것이 존재함을 지정하려는 것이지, 하나 또는 그 이상의 다른 특징들이나 숫자, 단계, 동작, 구성요소, 부품 또는 이들을 조합한 것들의 존재 또는 부가 가능성을 미리 배제하지 않는 것으로 이해되어야 한다.
- [0029] 다르게 정의되지 않는 한, 기술적이거나 과학적인 용어를 포함해서 여기서 사용되는 모든 용어들은 본 발명이 속하는 기술 분야에서 통상의 지식을 가진 자에 의해 일반적으로 이해되는 것과 동일한 의미를 가지고 있다. 일반적으로 사용되는 사전에 정의되어 있는 것과 같은 용어들은 관련 기술의 문맥 상 가지는 의미와 일치하는 의미를 가지는 것으로 해석되어야 하며, 본 출원에서 명백하게 정의하지 않는 한, 이상적이거나 과도하게 형식적인 의미로 해석되지 않는다.
- [0030] 이하, 본 발명에 따른 바람직한 실시예를 첨부된 도면을 참조하여 상세하게 설명한다.
- [0031] 본 명세서에서 네트워크는 그래프라고도 하고, 벡터스와 엣지로 이루어진 자료 구조를 의미한다. 네트워크 인스턴스 각각을 의미하는 벡터스와 인스턴스 간의 연결 관계를 엣지라고 한다. 네트워크 구조를 갖고 있는 데이터를 분석하는 것을 네트워크 분석이라고 한다.
- [0033] 도 1은 본 발명의 일 실시예에 따른 불확실성 그래프 분석을 위한 샘플링 장치의 구성을 나타내는 블록도이다.
- [0034] 도 1을 참조하면, 샘플링 장치(100)는 난수 생성부(110), 샘플 그래프 추출부(120), 제어부(130) 및 메모리(140)를 포함할 수 있다.
- [0035] 난수 생성부(110)는 미리 저장된 난수 생성 알고리즘에 기초하여 무작위하게 난수를 생성한다. 여기서, 난수는 특정한 배열순서나 규칙을 가지지 않는 값이지만, 씨드(seed)가 동일하면, 동일한 값의 난수가 생성된다.
- [0036] 일 실시예에서 난수 생성부(110)는 각 벡터(vertex)의 엣지(Edge)에 대하여 고정된 특정값을 난수의 씨드로 설정하여 난수를 생성할 수 있다. 여기서 고정된 특정값은 각 엣지를 구성하는 소스 벡터(source vertex)와 타겟 벡터(target vertex)의 ID의 해시값일 수 있다. 이에 따라, 동일한 벡터의 엣지에서는 동일한 씨드가 사용되므로 시간의 경과에 상관없이 동일한 난수가 생성될 수 있다.

- [0037] 샘플 그래프 추출부(120)는 샘플링 장치(100)에서 제공하는 사용자 정의 함수에 관한 인터페이스에 기초하여 네트워크 인스턴스를 입력받을 수 있다.
- [0038] 샘플 그래프 추출부(120)는 입력받은 네트워크 인스턴스 및 상기 난수 생성부(110)에 의해 생성되는 난수에 기초하여 복수개의 샘플 그래프를 추출한다.
- [0039] 제어부(130)는 샘플링 장치(100)의 각 구성요소를 제어하여 샘플링 장치에 의해 샘플링하는 방법 및/또는 캐싱 방법을 제어한다.
- [0040] 일 실시예에서, 제어부(130)는 미리 정해진 그래프 분석 알고리즘에 따른 그래프 분석을 위해 난수 생성부(110)를 통해 상기 고정된 특정값을 씨드로 하여 동일한 난수를 생성하도록 제어하고, 상기 난수와 엣지의 확률에 기초하여 버텍스 사이의 연결여부를 결정한다.
- [0041] 메모리(140)는 제어부(130)의 샘플링 및/또는 캐싱을 위해 필요한 데이터 및 프로그램을 저장한다. 예를 들어 메모리(140)는 난수 발생 알고리즘을 저장하거나, 네트워크 인스턴스에 기초한 버텍스 데이터와 엣지의 확률을 저장할 수 있다.
- [0042] 이와 같은 본 발명의 일 실시예에 따르면, 종래의 불확실성 그래프 분석을 위한 샘플링에서 샘플 그래프의 연결 여부를 직접 메모리에 저장하지 않고, 동일한 난수를 발생시켜 복수개의 샘플 그래프를 추출하되, 추출되는 복수개의 샘플 그래프는 동일한 순서로 추출가능하다. 이로써, 적은 메모리로도 불확실성 그래프 분석을 가능하게 한다는 장점이 있다. 특히 샘플링의 수가 많을 경우, 그 효과는 더욱 두드러지게 나타날 수 있다.
- [0044] 도 2는 본 발명의 다른 실시예에 따른 불확실성 그래프 분석을 위한 샘플링 장치의 구성을 나타내는 블록도이다.
- [0045] 도 2를 참조하면, 샘플링 장치(200)는 난수 생성부(210), 샘플 그래프 추출부(220), 캐시 관리부(225), 제어부(230), 메인 메모리(240) 및 캐시 메모리(250)를 포함할 수 있다.
- [0046] 난수 생성부(210)는 미리 저장된 난수 생성 알고리즘에 기초하여 무작위하게 난수를 생성한다. 여기서, 난수는 특정한 배열순서나 규칙을 가지지 않는 값이지만, 동일한 씨드를 입력받아 난수를 생성하는 경우, 동일한 난수를 발생시킨다.
- [0047] 샘플 그래프 추출부(220)는 샘플링 장치(200)에서 제공하는 사용자 정의 함수에 관한 인터페이스에 기초하여 네트워크 인스턴스를 입력받을 수 있다.
- [0048] 샘플 그래프 추출부(220)는 입력받은 네트워크 인스턴스 및 상기 난수 생성부(210)에 의해 생성되는 난수에 기초하여 복수개의 샘플 그래프를 추출한다.
- [0049] 캐시 관리부(225)는 버텍스의 활성여부에 기초하여 버텍스 사용 우선순위를 설정하고, 상기 버텍스 사용 우선순위에 기초하여 미리 설정된 개수의 버텍스 데이터를 캐시 메모리(250)에 저장한다.
- [0050] 일 실시예에서, 캐시 관리부(225)는 Schedule-aware Cache 기법을 사용할 수 있으나, 이에 한정하는 것은 아니다.
- [0051] 일 실시예에서, 캐시 관리부(225)는 버텍스의 패스카운트(pathcount)의 수의 크기가 클 수록 높은 우선순위를 부여한다.
- [0052] 제어부(230)는 샘플링 장치(100)의 각 구성요소를 제어하여 샘플링 장치에 의해 샘플링하는 방법 및/또는 캐싱 방법을 제어한다.
- [0053] 일 실시예에서, 제어부(230)는 캐시 메모리에 저장된 버텍스 데이터에 기초하여 미리 저장된 버텍스 프로그램을 실행한다.
- [0054] 캐시 관리부(225)는 각 버텍스에 대하여 버텍스 프로그램의 실행이 완료되면 실행이 완료된 버텍스를 제외한 나머지 버텍스들 중에서 사용 우선순위를 재설정한다.
- [0055] 메인 메모리(240)는 제어부(130)의 샘플링 및/또는 캐싱을 위해 필요한 데이터 및 프로그램을 저장한다. 예를 들어 메모리(140)는 난수 발생 알고리즘을 저장하거나, 버텍스 프로그램을 저장할 수 있다.
- [0056] 캐시 메모리(250)는 메인 메모리(240)에 저장된 정보를 입출력하는 시간보다 더 짧은 입출력 시간을 갖는 고속

기억 메모리이다.

- [0057] 일 실시예에 따른 캐시 메모리(250)는 우선순위가 높은 미리 설정된 개수의 버텍스 데이터를 저장하고 캐시 관리부(225)의 지시에 따라 재설정된 우선순위에 따라 저장되는 버텍스 데이터를 업데이트할 수 있다.
- [0058] 이와 같이 버텍스의 패스카운트에 기초하여 설정된 우선순위에 따라 캐시 메모리(250)에 버텍스를 저장함으로써, 불확실성 그래프 분석시 디스크 입출력(IO) 성능을 향상시킬 수 있다.
- [0060] 도 1 및 도 2의 샘플링 장치들의 동작 방법에 대하여는 도 3 내지 도 9에 기초하여 상세히 설명한다.
- [0062] 도 3은 본 발명의 일 실시예에 따른 불확실성 그래프 분석을 위한 샘플링 장치의 샘플링 방법을 설명하기 위한 흐름도이고, 도 4는 본 발명의 일 실시예에 따른 샘플 그래프이다.
- [0063] 도 3을 참조하면, 단계 S110에서, 샘플링 장치는 API를 제공하여 네트워크 인스턴스에 기초하여 고정된 난수값을 이용하여 난수 생성 기법을 통해 복수개의 샘플 그래프를 추출한다.
- [0064] 도 4를 참조하면, 예를 들어, 불확실한 네트워크(G)에 대하여 랜덤하게 생성된 5개의 샘플 네트워크(G_1, G_2, G_3, G_4, G_5)를 추출하는 것을 가정한다.
- [0065] 불확실한 네트워크(G)는 에지 존재 확률이 있는 6개의 네트워크 인스턴스를 갖는다.
- [0066] 불확실한 네트워크(G)에서는 6개의 버텍스($V_1, V_2, V_3, V_4, V_5, V_6$)가 예시되어 있으나, 버텍스의 개수는 설명의 편의를 위하여 간소화한 것으로 이에 한정하는 것이 아니다. 각 버텍스 간의 연결 확률은 연결 가능한 버텍스들 사이에 숫자로 표시할 수 있다.
- [0067] 랜덤하게 샘플 그래프를 생성하기 위하여, 각 버텍스를 분리하고 그 사이에 무작위로 연속 가장자리를 추가한다. 이에 따라 5개의 샘플 그래프 G_1, G_2, G_3, G_4, G_5 가 생성될 수 있다. 각 샘플 그래프들은 네트워크(G)에 포함된 모든 버텍스를 포함하지만, 샘플 그래프들은 서로 다른 엣지를 갖는다.
- [0068] 예를 들어, 샘플 그래프 G_1 는 버텍스($V_1, V_2, V_3, V_4, V_5, V_6$)를 모두 포함하지만, 버텍스 V_1 은 버텍스 V_2 에 연결되고, 버텍스 V_2 는 버텍스 V_5 에 연결되며, 버텍스 V_4 는 버텍스 V_5 에 연결된다. 버텍스 V_3 및 버텍스 V_6 는 어떠한 버텍스와도 연결되지 않는다. 샘플 그래프 G_2 는 버텍스($V_1, V_2, V_3, V_4, V_5, V_6$)를 모두 포함하지만, 버텍스 V_1 은 버텍스 V_2 에 연결되고, 버텍스 V_2 는 버텍스 V_5, V_4 에 연결되며, 버텍스 V_4 는 버텍스 V_6 에 연결된다. 버텍스 V_3 는 어떠한 버텍스와도 연결되지 않는다. 샘플 그래프 G_3 는 버텍스($V_1, V_2, V_3, V_4, V_5, V_6$)를 모두 포함하지만, 버텍스 V_1 은 버텍스 V_2 에 연결되고, 버텍스 V_2 는 버텍스 V_5 , 에 연결되며, 버텍스 V_5 는 버텍스 V_6 에 연결된다. 버텍스 V_3 는 버텍스 V_4 에 연결된다. 이때 샘플 그래프 생성순서는 난수 발생기에 의해 생성되는 난수에 의해 결정되며, 엣지 마다 엣지를 구성하는 소스 버텍스(Source Vertex)와 타겟 버텍스(Target Vertex)의 ID의 해시 값을 난수의 씨드로 사용하여 고정한 후, 샘플 개수만큼 엣지의 연결여부를 생성한다. 따라서, 난수를 다시 생성하더라도 동일한 엣지의 연결 결과가 나오기 때문에, 샘플 그래프를 메모리에 별도로 저장할 필요가 없게 된다. 다시 말해서, 메모리에는 엣지의 연결 확률만 저장하고, 각 샘플 그래프의 연결여부는 저장하지 않는다.
- [0069] 난수 발생기는 고정된 특정값을 난수의 씨드(seed)로 설정한다. 이로써, 샘플 그래프를 생성할 때마다 고정된 특정값을 씨드로 설정하므로 다음번에도 동일한 순서의 샘플 그래프를 출력할 수 있게 된다.
- [0070] 단계 S120에서, 샘플링된 그래프의 버텍스(vertex) 데이터와 엣지(Edge)의 확률을 메모리에 저장한다.
- [0071] 종래의 불확실성 그래프 분석을 위해서는 단계 S110에서 생성된 샘플 그래프를 모두 메모리에 저장하여, 그래프 분석을 진행하였으나, 본 발명의 일 실시예에서는 샘플 그래프를 저장하지 않고, 필요시, 고정된 난수값을 이용하여 동일한 샘플 그래프를 추출할 수 있게 되므로, 메모리 오버헤드를 경감할 수 있다.
- [0072] 도 5는 본 발명의 일 실시예에 따른 불확실성 그래프 분석을 위한 샘플링 장치의 캐싱 방법을 설명하기 위한 흐름도이다.
- [0073] 도 5를 참조하면, 단계 S210에서, 네트워크 인스턴스에 기초하여 난수 생성 기법을 통해 복수개의 샘플 그래프

를 추출한다.

[0074] 도 4에서 예시한 바와 같이, 불확실한 네트워크($\mathcal{G}$)에 대하여 랜덤하게 생성된 5개의 샘플 네트워크(G_1, G_2, G_3, G_4, G_5)를 가정할 수 있다.

[0075] 이 경우, 불확실한 네트워크($\mathcal{G}$)는 에지 존재 확률이 있는 6개의 인스턴스를 갖고, 각 버텍스 간의 연결 확률은 숫자로 표시될 수 있다.

[0076] 랜덤하게 샘플 그래프를 생성하기 위하여, 각 버텍스를 분리하고 그 사이에 무작위로 연속 가장자리를 추가한다. 이에 따라 5개의 샘플 그래프 G_1, G_2, G_3, G_4, G_5 가 생성될 수 있다. 각 샘플 그래프들은 네트워크($\mathcal{G}$)에 포함된 모든 버텍스를 포함하지만, 샘플 그래프들은 서로 다른 엣지를 갖는다.

[0077] 단계 S220에서, 샘플링된 그래프의 버텍스(vertex) 데이터와 엣지의 확률을 메모리에 저장한다.

[0078] 단계 S230에서, 각 버텍스의 활성여부에 기초하여 버텍스 사용 우선순위를 설정한다. 일 실시예에서, 버텍스의 패스카운트(pathcount)의 수의 크기가 클 수록 높은 우선순위를 부여할 수 있다. 여기서 패스카운트는 모든 샘플에 대하여 소스 버텍스와 임의의 버텍스간의 연결된 수를 말한다. 예를 들어, 도 4에서, 모든 샘플에 대하여 소스 버텍스 V1에 연결된 버텍스는 흰색으로 표시되고, 소스 버텍스 V1과 연결되지 않은 버텍스는 검은색으로 표시할 경우, 버텍스 V2는 4개의 샘플에서 흰색으로 표시됨을 확인할 수 있다. 따라서, 버텍스 V2의 패스카운트는 4가 된다. 이와 같은 방법으로 버텍스 V5와 버텍스 V6의 패스카운트도 4임을 확인할 수 있고, 버텍스 V3의 패스카운트는 1이 되고, 버텍스 V4의 패스카운트는 2가 되는 것을 확인할 수 있다. 따라서, 도 4의 예에서, 패스카운트의 수가 큰 미리 설정된 개수(예컨대 3개)의 버텍스를 선정하는 경우, 버텍스 V2, 버텍스 V5 및 버텍스 V6가 선정될 수 있다. 미리 설정된 개수가 4개인 경우, 버텍스 V2, 버텍스 V5, 버텍스 V6 및 버텍스 V4가 선정될 수 있다. 다만 동일한 우선순위를 갖는 버텍스가 미리 설정된 개수보다 더 많은 경우, 먼저 샘플 추출되는 순서 또는 더 많은 엣지를 갖는 순서 등으로 미리 설정된 개수만큼을 선정하도록 할 수 있으나, 이에 한정하는 것은 아니다.

[0079] 단계 S240에서, 상기 버텍스 사용 우선순위에 기초하여 미리 설정된 개수의 버텍스 데이터를 캐시 메모리에 저장한다.

[0080] 단계 S250에서, 상기 캐시 메모리에 저장된 버텍스 데이터에 기초하여 미리 저장된 버텍스 프로그램을 실행한다. 버텍스 프로그램은 도 7을 참조하여 후술된다. 버텍스 프로그램은 목적지 버텍스를 변경하면서 계속해서 실행되며, 모든 목적지 버텍스에 대하여 실행되면 버텍스 프로그램은 완료된다.

[0081] 단계 S250이 완료되면, 단계 S260에서 실행이 완료된 버텍스를 제외하고 상기 버텍스 사용 우선순위를 재설정하고, 단계 S250으로 되돌아가서 버텍스 프로그램을 실행한다.

[0082] 모든 버텍스 프로그램이 완료될 때까지 단계 S250과, 단계 S260을 반복적으로 수행한다.

[0083] 도 6은 본 발명의 일 실시예에 따라 제공되는 사용자 정의 함수에 관한 인터페이스이고, 도 7은 본 발명의 일 실시예에 따른 버텍스 프로그램이 수행되는 알고리즘의 일 예이다.

[0084] 도 6을 참조하면 알 수 있는 바와 같이, 샘플링을 위해 제공되는 사용자 정의 함수는 Scatter()와 Gather(), Reduce() 등을 포함할 수 있고, 도 7을 참조하면 알 수 있는 바와 같이, uperstep마다 Scatter, Gather, Reduce 순으로 수행된다. Scatter와 Reduce는 각 Vertex의 ID 순으로 실행되며, Gather의 경우 Target Vertex의 ID순으로 수행된다. 본 발명의 일 실시예에 따라 메모리에 유지되는 각 Vertex의 활성여부와 버텍스 ID를 확인하면, 현재 Superstep에서 앞으로 수행할 Vertex Program의 순서를 예측할 수 있다.

[0085] 도 8은 본 발명의 일 실시예에 따른 버텍스 사용 우선순위를 계산하기 위한 알고리즘의 일 예이다.

[0086] 도 8을 참조하면, 알고리즘에 따라 먼저 소스 버텍스 s와 타겟 버텍스 t의 gather를 입력하여, 소스 버텍스에서 타겟 버텍스 엣지에 대한 프로그램을 실행한 이후, 다음 s의 실행시점을 예측해서 우선순위를 업데이트 한다. t는 타겟 버텍스의 ID 순서대로 실행되고, s는 네트워크에 따라 임의대로 실행된다.

[0087] 도 8의 알고리즘의 첫번째 라인을 참조하면 알 수 있듯이 우선순위에 따라 앞으로 실행될 s를 소스 버텍스로 사용되지는지를 확인한다. 엣지가 (t,s)순으로 실행되기 때문에 s와 연결된 버텍스 중에서 t보다 큰 것이 있으면, s

가 다시 다른 엣지의 소스로 사용된다.

- [0088] 예를 들어, 만약 $s \rightarrow t'$ 에서 s 가 다시 소스로 사용된다고 하면, Line 2-4가 실행된다.
- [0089] 두번째 라인에서 s 가 $s \rightarrow t'$ 이외에 다른 엣지에서 타겟으로 사용되는 엣지(이하 $s' \rightarrow s$)가 있는지 체크한다.
- [0090] 만약 사용된다면, 세번째 라인에서, s 의 우선순위는 $s' \rightarrow s$ 와 $s \rightarrow t'$ 중에서 먼저 사용될 엣지의 타겟 ID(s 와 t' 중 더 작은값)로 업데이트한다.
- [0091] 만약 $s' \rightarrow s$ 엣지가 활성화 되어있지 않으면 네번째 라인에서(Line4)이 실행되어, t' 으로 우선순위를 업데이트한다.
- [0092] 활성화된 엣지의 경우, 타겟 버텍스의 상태를 triggered로 세팅한다.
- [0093] $s \rightarrow t$ 보다 나중에 실행될 엣지가 없다면($s \rightarrow t'$ 없다면), 다섯번째 라인과 여섯번째 라인6이 실행된다.
- [0094] s 가 다른 엣지에서 타겟으로 사용되고($s' \rightarrow s$), 활성화되어 있으면, s 의 우선순위를 $-s$ 로 업데이트한다.
- [0095] 다른 엣지에서 사용되지 않으면, s 는 앞으로 사용되지 않을 것이라고 예측하여, 우선순위는 최하값($-\infty$) 값을 갖도록 재설정한다.
- [0096] 이와 같이, 일 실시예에서는 Schedule-aware Cache를 채택하여, 먼저 사용할 Vertex 데이터의 우선순위를 높여 캐시에 저장하도록 한 후, 버텍스 프로그램이 완료될 때마다 버텍스 데이터의 우선순위를 재계산함으로써, 입출력 성능을 향상시킨다.
- [0097] 도 9는 본 발명의 일 실시예에 따른 Top-K Reliability Search 알고리즘의 일 예이다.
- [0098] Top-K Reliability Search 알고리즘은 Scatter()와 Gather(), Reduce() 등의 사용자 정의 함수를 채택할 수 있다. Scatter()와 Gather()는 각각의 샘플 그래프에서 분석할 때 사용된다. Visited는 버텍스가 연결되었는지를 확인하기 위한 것으로, 최초에 false로 설정되며, Scatter()를 통해 연결된 경우 버텍스의 false가 true가 되며, 연결된 버텍스 간의 트리거를 하게 된다.
- [0099] 이를 통해 모든 연결된 버텍스는 false에서 true로 변경되게 된다.
- [0100] Gvertex에서 reduce를 통해 visited 된 버텍스의 수를 카운트하여 패스카운트를 설정한다.
- [0101] 이후, TopkGlobal을 통해 패스카운트의 수를 미리 설정된 개수(top k)만큼을 추적할 수 있다.
- [0102] 이제까지 본 발명에 대하여 그 바람직한 실시예들을 중심으로 살펴보았다. 본 발명이 속하는 기술 분야에서 통상의 지식을 가진 자는 본 발명이 본 발명의 본질적인 특성에서 벗어나지 않는 범위에서 변형된 형태로 구현될 수 있음을 이해할 수 있을 것이다. 그러므로 개시된 실시예들은 한정적인 관점이 아니라 설명적인 관점에서 고려되어야 한다. 본 발명의 범위는 전술한 설명이 아니라 특허청구범위에 나타나 있으며, 그와 동등한 범위 내에 있는 모든 차이점은 본 발명에 포함된 것으로 해석되어야 할 것이다.

도면

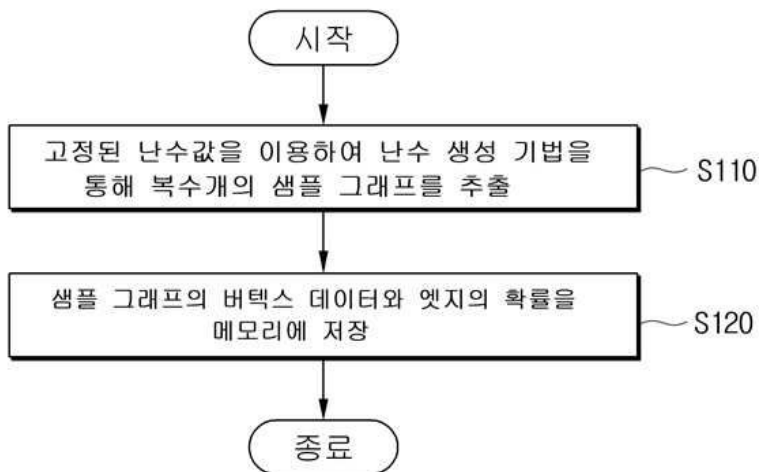
도면1



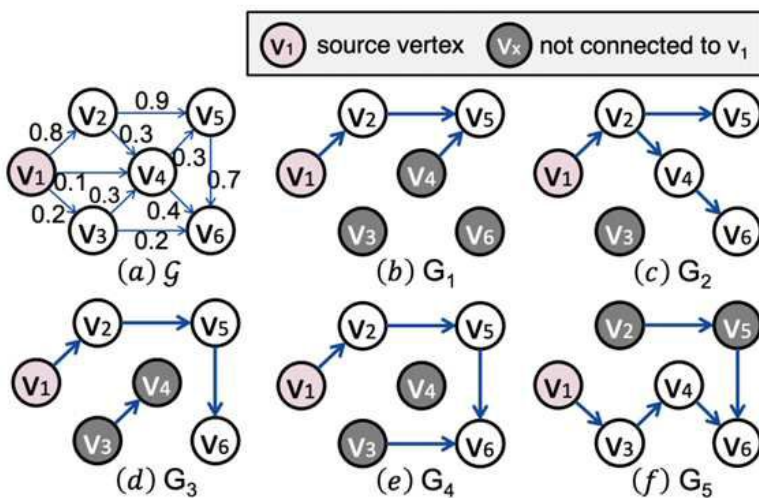
도면2



도면3



도면4



도면5



도면6

```

class Vertex {
    virtual void Init();
    virtual void Scatter();
    virtual void Gather(const Vertex src, const GEdge edge);
};
class GVertex {
    virtual void Reduce(const Vertex v);
    virtual void ReduceDone();
};
class Global {
    virtual void Reduce(GVertex gv);
    virtual void ReduceDone();
};
    
```

도면7

```

1 while active vertex exists do
2   foreach  $s \in \text{SampleNetworks}$  do
3     foreach  $v \in s.\text{ActiveVertices}()$  do
4        $v.\text{Scatter}()$ 
5     end
6   end
7   foreach  $s \in \text{SampleNetworks}$  do
8     foreach  $u \rightarrow v \in s.\text{Edges}()$  do
9       if  $u$  triggered  $v$  in  $u.\text{Scatter}()$  then
10         $v.\text{Gather}(u, u \rightarrow v)$ 
11      end
12    end
13    if active vertex not exists then
14      foreach  $gv \in \text{All GlobalVertices}$  do
15        foreach  $v \in gv.\text{SampleVertices}()$  do
16           $gv.\text{Reduce}(v)$ 
17        end
18         $gv.\text{ReduceDone}()$ 
19      end
20    end
21  end
22  foreach  $gv \in \text{All GlobalVertices}$  do
23     $\text{Global.Reduce}(gv)$ 
24  end
25  $\text{Global.ReduceDone}()$ 

```

도면8

Input: s and t are the source and target vertex of gather.

Output: updated priority of s

```

1 if  $s.\text{nbrLeastLarger}(t)$  exist then
2   if  $\text{isTriggered}(s)$  and  $t < s$  then
3      $s.\text{priority} = -\min(s, s.\text{nbrLeastLarger}(t))$ 
4   else  $s.\text{priority} = -s.\text{nbrLeastLarger}(t)$ ;
5 else
6   if  $\text{isTriggered}(s)$  and  $t < s$  then  $s.\text{priority} = -s$ ;
7   else  $s.\text{priority} = -\text{inf}$ ;
8 end

```

도면9

```

class TopkVertex: Vertex {
  bool visited = false;
  void Scatter() {
    if (visited==false) {
      visited = true;
      triggerGatherOnNeighbors();
    }
  }
  void Gather(const Vertex src, const GEdge edge) {
    ActivateSelf();
  }
};

class TopkGVertex: GVertex {
  int pathcount;
  void Reduce(const Vertex v) {
    if (v.visited) pathcount ++;
  }
};

class TopkGlobal: Global {
  pair<VertexId, float> topkVertices[K];
  void Reduce(GVertex gv) {
    InsertWithPriority(topkVertices, gv);
  }
};

```