ISSN 2383-630X(Print) / ISSN 2383-6296(Online)
Journal of KIISE, Vol. 52, No. 4, pp. 341-349, 2025. 4
https://doi.org/10.5626/JOK.2025.52.4.1

자원 공유에 따른 이상치 전파를 활용한 마이크로서비스 시스템 결함의 근본 원인 분석

(Root Cause Analysis for Microservice Systems Using Anomaly Propagation by Resource Sharing)

박 준 호 †　　황 지 영 ††
(Junho Park)　(Joyce Jiyoung Whang)

요 약 마이크로서비스 시스템에서 발생하는 결함의 근본 원인을 분석하는 작업은 다수의 리소스가 동적으로 변화하는 특성 때문에 어려운 과제로 남아 있다. 본 논문에서는 이를 해결하기 위해 리소스 간 상호작용과 이에 따른 이상치 전파를 반영한 근본 원인 분석 모델인 AnoProp을 제안한다. AnoProp은 회귀 모델을 통해 메트릭의 이상치를 측정하고, 이러한 이상치의 전파율을 기반으로 리소스의 오류 원인 여부를 평가하는 두 가지 핵심 기법을 통합하여 설계되었다. 실제 마이크로서비스 시스템에서 수집한 Online Boutique 데이터셋을 활용한 실험 결과, AnoProp은 다양한 평가 지표에서 기존 모델을 상회하는 성능을 기록하였으며, 모델의 설계가 근본 원인 종류별로 균형 잡힌 성능을 제공함을 입증하였다. 본 연구는 AnoProp을 통해 마이크로서비스 환경에서 시스템 안정성을 강화하고 운영 효율성을 높일 수 있는 실질적 가능성을 제시한다.

키워드: 마이크로서비스 시스템, 근본 원인 분석, 이상치 전파, 자원 공유 의존성, 인과 관계 그래프, 리소스 그래프

Abstract Identifying root causes of failures in microservice systems remains a critical challenge due to intricate interactions among resources and propagation of errors. We propose AnoProp, a novel model for root cause analysis to address challenges by capturing inter-resource interactions and the resulting propagation of anomalies. AnoProp incorporates two core techniques: the anomaly score measurement for metrics using regression models and the root cause score evaluation for resources based on the propagation rate of these anomalies. Experimental results using an Online Boutique dataset demonstrated that AnoProp surpassed existing models across various evaluation metrics, validating its ability to provide balanced performance for different types of root causes. This study underscores the potential of AnoProp to enhance system stability and boost operational efficiency in microservice environments.

Keywords: microservice system, root cause analysis, anomaly propagation, resource-sharing dependency, causality graph, resource graph

· 본 연구는 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행하였습니다(RS-2025-00559066, 2022R1A2C4001594)

† 비 회 원 : 한국과학기술원 전산학부 학생
jhpakk1024@kaist.ac.kr
†† 종신회원 : 한국과학기술원 전산학부 교수(KAIST)
jjwhang@kaist.ac.kr
(Corresponding author임)

논문접수 : 2025년 1월 17일
(Received 17 January 2025)
논문수정 : 2025년 2월 18일
(Revised 18 February 2025)
심사완료 : 2025년 2월 21일
(Accepted 21 February 2025)

정보과학회논문지 제52권 제4호(2025. 4)

1. 서 론

마이크로서비스 시스템(microservice system)은 대규모의 복잡한 컴퓨터 시스템을 독립적으로 실행되는 다수의 작은 리소스(resource)로 분할하여 관리한다[1]. 이는 사용자에게 가상화된 리소스 단위로 서비스를 제공하여 실제 물리적 장비에 의존하지 않는 유연한 개발과 배포를 가능하게 하며, 시스템의 상황에 따라 자원을 유동적으로 할당함으로써 확장성이 높은 서비스 환경을 구축한다. 이러한 특징으로 인해 마이크로서비스 시스템은 사용자의 다양한 요구사항에 효과적으로 대응하며, 여러 산업에서 널리 사용되고 있다.

그러나 리소스 간의 복합적인 연결 구조로 인해, 한 리소스에서 발생한 오류가 다른 리소스에 영향을 미쳐 시스템 전체에 결함을 초래할 수 있으며, 이는 결함의 원인을 규명하는 과정을 더욱 복잡하게 만든다. 이러한 문제를 해결하기 위해, 근본 원인 분석(root cause analysis) 연구는 시스템에서 발생한 결함의 근본 원인을 파악하는 것을 목표로 한다[1]. 근본 원인 분석 작업은 시스템의 다양한 정보를 실시간으로 수집하고, 결함이 탐지되면 이를 분석하여 해당 결함의 원인을 예측한다. 분석 결과는 전문가가 결함을 해결하거나 대책을 수립하는 데 중요한 자료로 활용되며, 이러한 과정을 통해 시스템의 운영 효율성을 높이고 서비스의 신뢰성을 강화할 수 있다.

기존 연구들은 마이크로서비스 내의 애플리케이션(application)들이 각각 독립적으로 실행된다는 가정을 기반으로 한다. 그러나 리소스를 공유하는 애플리케이션 간에는 상호작용이 존재할 수 있다. 예를 들어, 여러 애플리케이션이 동일한 컴퓨팅 자원을 공유하는 경우, 한 애플리케이션이 과도하게 자원을 사용하면 다른 애플리케이션의 자원 사용량이 상대적으로 감소할 수 있다. 이러한 리소스 공유의 영향을 간과할 경우, 분석 과정에서 오차가 발생해 정확한 원인 분석이 어려워질 수 있다.

이를 해결하기 위해, 본 논문에서는 리소스를 공유하는 애플리케이션 간 상호작용에 따른 리소스 간 이상 전파를 반영한 근본 원인 분석 모델인 AnoProp(root cause analysis by considering Anomaly Propagation across resources)을 제안한다. AnoProp은 동일한 리소스를 공유하여 발생하는 영향을 고려한 이상치 측정 기법과, 리소스 간 이상치 전파율을 기반으로 리소스의 오류 여부를 평가하는 근본 원인 평가 기법으로 구성된다. 이를 통해 AnoProp은 리소스 간 상호작용에서 발생하는 복잡한 영향을 효과적으로 모델링하여 기존 모델 대비 높은 정확도와 신뢰성을 제공한다.

2. 관련 연구

마이크로서비스에서의 근본 원인 분석은 정보의 유형과 분석 수준에 따라 여러 범주로 구분된다. 마이크로서비스에서 주로 사용하는 정보는 메트릭(metric), 로그(log), 트레이스(trace)가 있다[1]. 메트릭은 자원 사용량, 네트워크 패턴 등 시스템에 관한 정보를 나타내는 데이터로, 이를 활용한 연구들은 주로 메트릭 간의 인과 관계를 나타내는 인과 관계 그래프(causality graph)를 구축하여 각 메트릭의 이상(anomaly) 발생 여부를 분석한다. 예를 들어, Microscope[2]는 인과 관계 그래프 상에서 깊이 우선 탐색(DFS)을 수행하여 이상 메트릭을 탐색한다. 로그는 시스템 내에서 발생한 사건을 기록한 데이터로, 시스템의 상태와 변화를 상세히 파악할 수 있다. 가령, SherLog[3]은 오류 메시지가 포함된 로그 메시지의 경로를 추적하고, 각 경로에서의 변숫값 변화를 분석하여 오류의 원인을 진단한다. 트레이스는 사용자의 요청이 서비스 간에 전달되는 과정을 시각화한 데이터로, 서비스 호출 관계와 대기 시간 등을 분석하여 오류가 발생한 서비스나 경로를 식별한다. 예를 들어, TraceRCA[4]는 이상 트레이스가 통과하는 비율을 기준으로 각 마이크로서비스의 신뢰도를 계산하여 오류의 원인을 특정한다. 로그나 트레이스는 오류에 대한 세부적인 분석에 유용하지만, 높은 수준의 관측 가능성(observability)이 필요하다는 제한이 있다. 반면, 메트릭은 상대적으로 저비용으로 수집할 수 있으며, 높은 관측 가능성을 요구하지 않으면서도 시스템 전반에 대한 포괄적인 정보를 제공할 수 있다.

이러한 정보에서 이상이 발생한 부분을 탐지하는 것을 넘어, 그 배후에 있는 원인 리소스를 탐색하는 연구들도 진행되고 있다. 이는 직접적인 관측이 가능한 메트릭, 로그, 트레이스와 달리 간접적으로 리소스 간 관계를 분석해야 하므로 보다 복잡한 추론 과정이 요구된다. TrinityRCL[5]은 메트릭과 리소스를 정점으로 하는 인과 관계 그래프를 생성하여 결함 원인 리소스를 파악한다. 그러나 이 접근법은 리소스 유형에 따라 그래프 설계 방식이 상이하며, 일부 리소스 유형만을 고려하는 한계가 있다. LatentScope[6]은 메트릭 정보를 기반으로 관련 리소스의 근본 원인 여부를 평가하는 방법을 제안하였으나, 리소스를 공유하는 메트릭 사이의 상호작용을 충분히 반영하지 못하는 한계가 있다.

3. 기초 개념 및 문제 정의

3.1 마이크로서비스의 기본 리소스

본 논문에서는 마이크로서비스의 실행과 관리를 지원

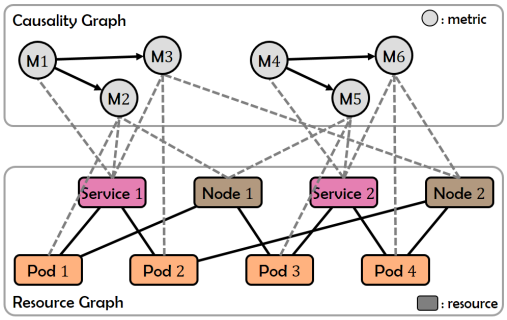

그림 1 리소스 그래프 및 인과 관계 그래프
Fig. 1 Resource Graph and Causality Graph

하는 플랫폼인 쿠버네티스(Kubernetes)에서 사용하는 파드(Pod), 서비스(Service), 노드(Node)를 기반으로 마이크로서비스 시스템의 기본 리소스를 정의한다[7]. 파드는 애플리케이션이 실행되는 가장 기본적인 배포 단위이다. 서비스는 트래픽 관리를 담당하며 여러 파드에 대한 네트워크 접근을 지원한다. 노드는 파드가 실행될 수 있는 컴퓨팅 자원을 제공하는 역할을 한다. 이러한 리소스들은 각각 실행 단위(파드), 통신 관리(서비스), 실행 환경(노드)이라는 세 가지 핵심적인 역할을 통해 마이크로서비스 시스템의 기본 구조를 형성하며, 플랫폼에 의존하지 않는 범용적인 개념으로 확장할 수 있다[8].

그림 1의 리소스 그래프(resource graph)는 마이크로서비스 시스템 내 리소스 간의 관계를 시각적으로 나타낸다. 각 파드는 네트워크 접근을 관리하는 하나의 서비스와 실행 환경을 제공하는 하나의 노드에 연결되며, 서비스와 노드는 각각 다수의 파드와 연결된다.

3.2 메트릭

메트릭은 마이크로서비스 시스템 성능에 관한 정량적 정보를 일정 간격으로 수집한 데이터이다[1]. 메트릭은 CPU 사용률, 메모리 사용량, 요청 지연 시간, 에러율 등 시스템의 주요 성능 지표를 모니터링하여 시스템의 상태를 평가하고 문제를 진단하는 데 기여한다.

메트릭은 성능 안정성 엔지니어링(site reliability engineering)[9]에서 제안한 네 가지 주요 신호(golden signals)인 트래픽(traffic), 포화(saturation), 지연(latency), 에러(error) 유형으로 분류될 수 있다. 트래픽 유형은 사용자가 시스템에 요구하는 수요를 측정하는 지표로, 요청 수나 데이터 전송량과 같은 항목을 모니터링한다. 포화 유형은 시스템 자원의 사용량을 나타내며, 파드와 관련된 메트릭이 포화 유형에 포함된다. 지연 유형은 요청을 처리하는 데 소요된 시간을 측정하며, 에러 유형은 요청 실패와 같은 예외 상황의 발생 빈도를 추적한다.

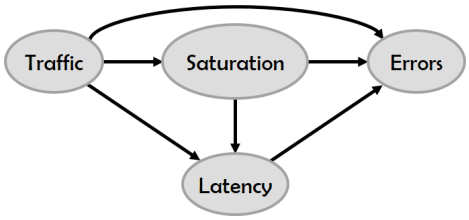

그림 2 메트릭 유형 간 인과적 의존성
Fig. 2 Causal Dependency between Metric Types

이 네 가지 메트릭 유형은 각각의 의미적 특성으로 인해 인과적 의존성(causal dependency)을 형성하며, 이러한 관계는 그림 2에 명시되어 있다. 이를 기반으로, 메트릭 간의 인과적 의존성을 간선으로 나타내어 그림 1의 인과 관계 그래프를 구성할 수 있다.

메트릭은 서비스를 통해 수집되며, 서비스와 관련된 항목을 모니터링한다. 이 중 일부 메트릭은 서비스가 관리하는 파드와 연관되어 있으며, 이러한 메트릭은 파드가 연결된 노드와도 연관되어 있다. 이에 따라 각 메트릭은 서비스와만 연관된 메트릭과 파드, 서비스, 노드와 모두 연관된 메트릭으로 구분할 수 있다. 또한, 파드와 연관된 메트릭은 모두 포화 유형에 속하며, 반대로 트래픽, 지연, 에러 유형의 메트릭들은 모두 서비스와만 연관된다.

3.3 개입 탐지

CIRCA[10]가 제안한 개입 탐지(intervention recognition) 기법은 시스템의 결함을 오류의 개입으로 모델링하고, 오류 개입 전후에 관측된 메트릭 간 인과 관계 변화를 분석하여 개입의 근본 원인을 규명하는 방법이다. 일반적으로 오류 발생 이전에는 인과 관계에 따라 메트릭이 일정한 경향성을 유지하지만, 오류가 개입되면 이러한 인과적 의존성이 변형되거나 단절되는 현상이 발생한다. CIRCA는 이를 효과적으로 포착하기 위해 각 메트릭에

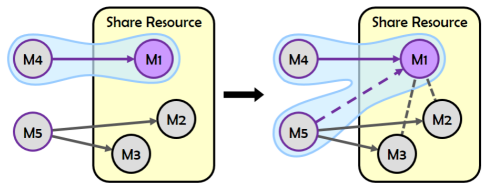

그림 3 자원 공유를 고려한 메트릭 값 예측
Fig. 3 Metric Value Prediction by Considering Resource-Sharing

대해 독립적인 회귀 모델을 학습하는 방식을 사용한다. 그림 3 (a)와 같이, 인과 관계 그래프상에서 각 메트릭에 인과적 영향을 미치는 메트릭을 부모 메트릭으로 정의하고, 부모 메트릭 값을 입력으로 사용하여 해당 메트릭 값을 예측하는 회귀 모델을 구축한다. 이 회귀 모델은 오류 발생 이전 시점의 데이터를 기반으로 학습된 후, 오류 발생 이후 시점에 대한 메트릭 값을 예측한다. 이 예측값과 실제 관측값 간의 차이로 해당 메트릭의 이상치가 정의되며, 이를 통해 각 메트릭의 이상 발생 여부가 평가된다. 이러한 접근법은 오류 개입으로 인한 시스템의 비정상 상태를 정량적으로 감지할 수 있도록 한다.

4. AnoProp 모델

파드에서 실행되는 애플리케이션은 서비스가 통신을 관리하며 노드로부터 컴퓨팅 자원을 제공받기 때문에, 동일한 서비스나 노드를 공유하는 파드 간에는 상호작용이 발생할 수 있다. 이러한 상호작용은 동일한 리소스를 공유하는 메트릭 간의 의존성으로 나타난다. 본 논문에서는 이를 반영하여, 개입 탐지를 기반으로 동일한 리소스를 공유하는 메트릭 간의 의존성을 효과적으로 모델링하는 근본 원인 분석 모델인 AnoProp을 제안한다.

4.1 자원 공유를 고려한 이상치 측정

AnoProp은 기존 접근법과 마찬가지로 각 메트릭의 이상치를 측정한 후, 이를 분석하여 각 리소스의 결함 원인 여부를 평가한다. 기존의 이상치 측정 기법은 인과 관계 그래프를 바탕으로 회귀 모델을 학습하여 메트릭 값을 예측한다. 이 과정은 다음 식으로 표현된다.

$$\bar{v}_{\mathrm{M}}^{(t)} = f_{\mathrm{M}}\left(pa(\mathrm{M})^{(t)}\right)$$

여기서 $pa(\mathrm{M})^{(t)}$은 메트릭 M의 부모 메트릭들이 시간 t에서 가지는 값을 의미하며, f_{M}은 기존 M의 회귀 모델을 나타낸다. 모델을 통해 예측된 M의 값은 $\bar{v}_{\mathrm{M}}^{(t)}$로 표현된다. 그러나 이러한 접근법은 인과 관계 그래프가 다른 유형의 메트릭 사이에 간선을 생성하는 방식으로 설계되기 때문에, 동일 유형의 메트릭 사이에 존재하는 의존성을 고려하지 못한다. 특히, 노드와 연관된 메트릭들은 모두 포화 유형에 속하므로, 같은 노드를 공유하는 메트릭 간의 상호작용을 반영하지 못하는 문제가 있다.

이를 극복하기 위한 방안으로, 동일한 리소스를 공유하는 메트릭들의 값을 회귀 모델의 입력으로 사용하는 접근법이 고려될 수 있다. 그러나 동일 리소스를 공유하는 메트릭들은 서로 간에 영향을 미치는 상호 의존성(mutual dependency)이 존재할 수 있으며, 이는 예측 대상 메트릭의 정보가 입력 데이터에 내포되는 결과를 초래한다. 이에 따라 시스템에 오류가 발생한 상황에서도 회귀 모델이 지나치게 정확한 예측을 수행하여, 실제로 이상이 발생한 메트릭의 이상치가 과소평가될 수 있다.

이를 해결하기 위해, 본 논문에서는 각 메트릭 값을 예측할 때 부모 메트릭 값을 사용하는 기존 접근법을 확장하여, 동일한 리소스를 공유하는 메트릭들의 영향을 이들의 부모 메트릭 값을 통해 모델링하였다. 이러한 회귀 모델의 구조는 그림 3 (b)에 나타나 있으며, 예측 과정은 다음과 같은 식으로 표현된다.

$$\bar{v}_{\mathrm{M}}^{(t)} = f'_{\mathrm{M}}\left(pa(\mathrm{M})^{(t)}, pa(res(\mathrm{M}))^{(t)}\right)$$

여기서 f'_{M}는 본 논문에서 제시하는 M의 새로운 회귀 모델이다. 또한, $res(\mathrm{M})$는 M과 동일한 리소스를 공유하는 메트릭들을 의미하며, $pa(res(\mathrm{M}))^{(t)}$은 이들의 부모 메트릭들이 시간 t에서 가지는 값을 나타낸다. 이를 통해 이상치가 과소평가 되는 문제를 해결하는 동시에, 동일한 리소스를 공유하는 메트릭들이 예측 대상 메트릭에 미치는 영향을 효과적으로 반영할 수 있다. 이후, 회귀 모델을 통해 얻은 예측값과 실제 관측값의 차를 해당 메트릭의 이상치 $AS_{\mathrm{M}}^{(t)}$로 정의한다.

$$AS_{\mathrm{M}}^{(t)} = v_{\mathrm{M}}^{(t)} - \bar{v}_{\mathrm{M}}^{(t)}$$

여기서 $v_{\mathrm{M}}^{(t)}$는 M의 시간 t에서의 실제 관측값을 나타낸다. $AS_{\mathrm{M}}^{(t)}$는 학습 데이터가 정규 분포를 따른다는 가정하에 각 메트릭별로 표준화되었으며, 이를 통해 메트릭 간 비교 가능성을 확보하였다.

4.2 이상치 전파율 측정을 통한 근본 원인 평가

리소스의 결함 원인 여부를 평가하기 위해, 관련 메트릭 간 이상치 전파 정도를 기반으로 각 리소스의 근본 원인 점수(root cause score)를 계산한다. 두 메트릭 간의 이상치 전파율은, 한 메트릭의 이상치를 입력으로 다른 메트릭의 이상치를 예측하는 회귀 모델을 통해 측정할 수 있다. 메트릭 Mi의 이상치로 메트릭 Mj의 이상치를 예측하는 회귀 모델 $g_{\mathrm{M}i,\mathrm{M}j}$는 다음과 같이 정의된다.

$$\overline{AS}_{\mathrm{M}j,\mathrm{M}i}^{(t)} = g_{\mathrm{M}i,\mathrm{M}j}\left(AS_{\mathrm{M}i}^{(t)}\right)$$

여기서 $\overline{AS}_{\mathrm{M}j}^{(t)}$는 시간 t에서 Mj의 이상치에 대한 예측값이다. 회귀 모델이 이상치를 정확히 예측할수록, Mi의 이상이 Mj로 전파되었음을 의미한다. 이에 따라 Mi에서 Mj로의 이상치 전파율은 다음과 같이 정의된다.

$$L_{\mathrm{M}i,\mathrm{M}j} = 1 - \frac{\max_t \left| AS_{\mathrm{M}j}^{(t)} - \overline{AS}_{\mathrm{M}j,\mathrm{M}i}^{(t)} \right|}{\max_t \left| AS_{\mathrm{M}j}^{(t)} \right|}$$

특정 리소스와 연관된 메트릭에서 다른 리소스와 연관된 메트릭으로 전파되는 이상치의 비율이 높으면, 해당 리소스에서 오류가 발생했을 가능성이 높다고 판단

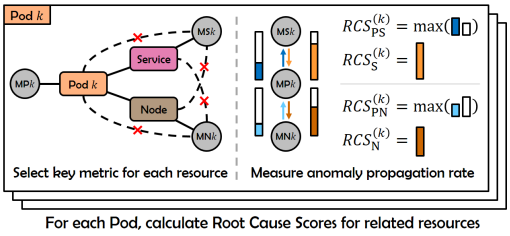

그림 4 이상치 전파율 기반 근본 원인 점수 계산
Fig. 4 Anomaly Propagation Rate-Based Root Cause Score Estimation

할 수 있다. 이를 바탕으로 근본 원인 점수를 산출하는 방식을 제안하며, 이 과정은 그림 4에 제시된 바와 같이 각 파드에 대해 수행된다. 먼저 'Pod k'에 대해, 관련 리소스별로 대표 메트릭(key metric)을 선정하여 분석 과정의 효율성을 향상하였다. 파드의 경우에는 연관된 메트릭 중, 전체 시간 동안 기록된 이상치 값 중에서 가장 높은 값을 가진 메트릭을 파드의 대표 메트릭으로 선정한다. 이렇게 선택된 파드의 대표 메트릭을 MPk로 정의한다. 'Pod k'와 연결된 서비스와 노드의 경우에는, 해당 리소스만의 영향을 반영하기 위해, MPk와 해당 리소스 외의 다른 리소스를 공유하지 않는 메트릭 중에서 대표 메트릭을 선정한다. 파드의 대표 메트릭 선정 방식과 비슷하게, 이들 중 가장 높은 이상치 값을 가진 메트릭을 각각의 대표 메트릭으로 선택한다. 이렇게 선택된 서비스와 노드의 대표 메트릭을 각각 MSk와 MNk로 정의한다. 이후, 대표 메트릭 간 이상치 전파율을 바탕으로 각 리소스의 근본 원인 점수를 계산한다. 서비스 및 노드의 경우, 해당 리소스의 대표 메트릭 이상치로 파드의 대표 메트릭 이상치를 정확히 예측하는 정도인 정방향 예측 정확성(forward prediction accuracy)에 비례한 근본 원인 점수를 부여한다. 서비스와 노드 각각이 받는 근본 원인 점수 $RCS_{\mathrm{S}}^{(k)}$와 $RCS_{\mathrm{N}}^{(k)}$는 다음과 같이 정의된다.

$$RCS_{\mathrm{S}}^{(k)} = \max_t \left| AS_{\mathrm{MP}k}^{(t)} \right| \cdot L_{\mathrm{MS}k,\mathrm{MP}k}$$

$$RCS_{\mathrm{N}}^{(k)} = \max_t \left| AS_{\mathrm{MP}k}^{(t)} \right| \cdot L_{\mathrm{MN}k,\mathrm{MP}k}$$

반면, MPk는 서비스 및 노드와도 연관된 메트릭이기 때문에, MSk나 MNk의 이상치로 MPk의 이상치를 예측하는 작업이 그 반대의 작업보다 상대적으로 더 쉬울 수 있다. 따라서 파드의 근본 원인 점수 계산은 두 가지 요소를 고려한다: (1) 파드의 대표 메트릭 이상치로 서비스나 노드의 대표 메트릭 이상치를 정확히 예측한 정도인 정방향 예측 정확성과 (2) 서비스나 노드의 대표 메트릭 이상치로 파드의 대표 메트릭 이상치를 정확히 예측하지 못한 정도인 역방향 예측 부정확성(reverse prediction inaccuracy). 이 두 값 중 최댓값을 선택하여 파드에게 점수를 부여한다. 파드가 서비스와 노드 각각으로부터 받는 근본 원인 점수 $RCS_{\mathrm{PS}}^{(k)}$와 $RCS_{\mathrm{PN}}^{(k)}$는 다음과 같이 정의된다.

$$RCS_{\mathrm{PS}}^{(k)} = \max_t \left| AS_{\mathrm{MP}k}^{(t)} \right| \cdot \max(L_{\mathrm{MP}k,\mathrm{MS}k}, 1 - L_{\mathrm{MS}k,\mathrm{MP}k})$$

$$RCS_{\mathrm{PN}}^{(k)} = \max_t \left| AS_{\mathrm{MP}k}^{(t)} \right| \cdot \max(L_{\mathrm{MP}k,\mathrm{MN}k}, 1 - L_{\mathrm{MN}k,\mathrm{MP}k})$$

마지막으로, 그림 5와 같이 리소스별로 계산된 근본 원인 점수들을 종합하여 최종 근본 원인 점수를 산출한다. 파드의 경우, 결함 발생 시 연결된 서비스와 노드로 이상이 전파되었을 것이라는 가정을 바탕으로, 이들에서 계산된 점수 중 최솟값을 선택한다. 반면, 서비스와 노드는 다수의 파드와 연결되어 있어 결함 발생 시 이상이 모든 파드로 전파되지 않을 가능성이 있으므로, 연결된 모든 파드에서 계산된 점수 중 최댓값을 선택한다. 또한, 서비스의 경우에는 서비스와만 연관된 메트릭들의 이상치도 포함하여 최댓값을 계산한다. 이렇게 계산된 최종 근본 원인 점수를 바탕으로, 가장 높은 점수를 가진 리소스를 결함의 원인으로 예측한다.

5. 실 험

5.1 데이터셋 및 실험 환경 설정

본 논문에서는 AnoProp의 근본 원인 분석 성능을 평

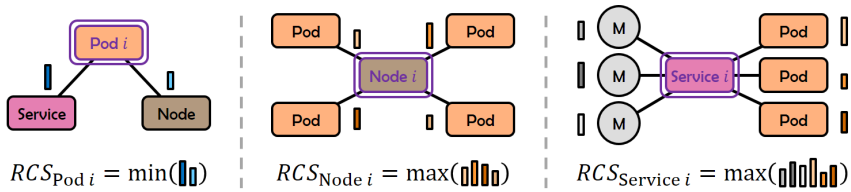

그림 5 리소스별 최종 근본 원인 점수 합산
Fig. 5 Final Root Cause Score Aggregation for Each Resource

가하기 위해 Google에서 제공하는 마이크로서비스 데모 시스템인 Online Boutique 테스트베드[11]에서 수집한 데이터셋[6]을 활용하여 실험을 수행하였다. 이 테스트베드는 총 32개의 파드, 11개의 서비스, 2개의 노드로 구성된 온라인 부티크 매장을 시뮬레이션하며, 시스템을 모니터링하는 1,976개의 메트릭을 포함한다. 해당 테스트베드에서 결함 주입(fault injection)을 통해 오류를 발생시켜 총 88개의 결함 사례를 생성하였다. 각 사례에서 메트릭은 1분 간격으로 수집되며, 결함 탐지 120분 전부터 10분 전까지의 데이터는 정상 상태 데이터로, 결함 탐지 전후 10분 동안의 데이터는 오류 발생 데이터로 사용된다.

AnoProp의 성능을 평가하기 위해, 다음의 여덟 가지 메트릭 기반 근본 원인 분석 모델을 기준(baseline) 모델로 사용하여 실험을 수행하였다.

- MonitorRank[12]: 인과 관계 그래프에서 Personalized PageRank(PPR) 알고리즘을 수행.
- CloudRanger[13]: 인과 관계 그래프에서 2차 random walk 알고리즘을 수행.
- RandomWalk[14]: 인과 관계 그래프와 리소스 그래프에서 각각 random walk 알고리즘을 수행.
- Microscope[2]: 인과 관계 그래프에서 DFS를 수행.
- MicroCause[15]: 인과 관계 그래프에서 시간 정보를 반영한 2차 random walk 알고리즘을 수행.
- CIRCA[10]: 인과 관계 그래프에서 개입 탐지 기반 회귀 모델을 활용해 이상 메트릭을 식별.
- TrinityRCL[5]: 멀티모달(multimodal) 데이터로 구축된 그래프에서 random walk 알고리즘을 수행.
- LatentScope[6]: CIRCA를 통해 메트릭의 이상치를 측정하고 이를 기반으로 이상 리소스를 식별.

RandomWalk, TrinityRCL, LatentScope을 제외한 기준 모델들은 이상 메트릭을 찾는 작업을 수행하므로, 이를 결함의 원인 리소스를 찾는 작업으로 확장할 필요가 있다. 이를 위해 각 리소스와 연관된 메트릭의 이상치 중 최댓값을 해당 리소스의 근본 원인 점수로 정의하고, 이를 바탕으로 결함의 원인 리소스를 예측하도록 하였다.

기준 모델인 CIRCA와 LatentScope, 그리고 본 논문에서 제안하는 AnoProp은 모두 회귀 모델을 활용하여 이상치를 측정하는 접근 방식을 따른다. 이들 모델에서는 Ridge 회귀를 기반으로 학습이 이루어지며, 정규화 강도(regularization strength)인 α(alpha) 값은 기본값인 1.0으로 설정하였다.

성능 평가는 기존 근본 원인 분석 연구에서 자주 활용되는 지표인 Hit@k와 MRR(Mean Reciprocal Rank)을 사용하여 수행되었다[6,16]. 두 지표는 모두 0에서 1 사이의 값을 가지며, 값이 클수록 모델의 성능이 우수함을 나타낸다. 또한, MRR의 변형 지표인 MRR@k를 사용하여 상위 k개의 예측에 정답이 포함되는 경우에만 reciprocal rank를 계산하고, 그렇지 않을 때는 0으로 설정하였다. 이는 실제 응용에서 상위 예측 후보에 정답을 포함하는 것이 중요하다는 점을 반영한 지표이다[6]. 또한, 전체 데이터셋에 대한 결과를 나타내는 Micro와, 근본 원인 종류별 성능의 평균을 측정하는 Macro 평가 방식을 함께 사용하여 근본 원인 종류별로 성능이 균등하게 나타나는지를 확인하였다. 각 평가 지표에 대해 가장 우수한 결과는 굵은 글씨로, 두 번째로 우수한 결과는 밑줄로 표시하였다.

5.2 근본 원인 예측 실험

표 1은 Online Boutique 데이터셋에서 AnoProp과 기준 모델들의 근본 원인 분석 성능을 비교한 결과를 제시한다. 실험 결과, AnoProp은 모든 평가 지표에서 가장 우수한 성능을 기록하였으며, 특히 Hit@1 지표에서는 두 번째로 높은 성능을 보인 LatentScope과 비교하여 상당한 성능 차이를 보였다. 또한, AnoProp과 LatentScope

표 1 근본 원인 예측 실험 결과
Table 1 Root Cause Prediction Results

	Micro				Macro			
	Hit@1	Hit@5	MRR	MRR@5	Hit@1	Hit@5	MRR	MRR@5
MonitorRank	0.1724	0.5632	0.3146	0.2816	0.1923	0.4313	0.2983	0.2587
CloudRanger	0.1707	0.6849	0.3906	0.3703	0.1740	0.7452	0.3936	0.3786
RandomWalk	0.1379	0.4912	0.2902	0.2498	0.1273	0.5795	0.3210	0.2841
Microscope	0.0920	0.5977	0.3123	0.2739	0.1026	0.6404	0.3495	0.3118
MicroCause	0.1724	0.7241	0.4101	0.3771	0.1862	0.7908	0.4151	0.3851
CIRCA	0.2159	0.8636	0.4633	0.4393	0.2804	0.8847	0.4537	0.4344
TrinityRCL	0.0517	0.1264	0.1169	0.0795	0.1189	0.2809	0.2185	0.1775
LatentScope	0.3864	0.9091	0.6060	0.5915	0.4615	0.9159	0.6569	0.6447
AnoProp	**0.4625**	**0.9205**	**0.6367**	**0.6140**	**0.5079**	**0.9287**	**0.6717**	**0.6534**

의 평균 실행 시간은 각각 8.39초 및 4.04초로 측정되었으며, 이는 AnoProp이 LatentScope보다 두 배 많은 회귀 모델을 활용하여 이상치 전파를 더욱 정교하게 모델링하기 때문이다. 이러한 결과는 AnoProp이 동일한 리소스를 공유하는 메트릭 간의 연관성과 이상치 전파를 효과적으로 모델링함으로써, 기존 모델 대비 근본 원인 분석 문제를 더욱 정밀하게 해결할 수 있음을 입증한다.

5.3 변형 실험

AnoProp의 설계 유효성을 평가하기 위해, AnoProp의 일부를 변형한 모델들을 사용하여 변형 실험을 수행하였다. 원본 AnoProp에서는 리소스의 근본 원인 점수를 계산할 때, 파드에 대해서만 정방향 예측 정확성뿐만 아니라 역방향 예측 부정확성을 함께 고려하였다. 이러한 근본 원인 점수 산출 방식이 성능에 미치는 영향을 분석하기 위해, 파드의 근본 원인 점수 산출 과정에서 역방향 예측 부정확성을 제거하거나, 서비스 및 노드의 근본 원인 점수 산출 과정에서도 역방향 예측 부정확성을 추가하는 두 가지 변형 기법을 원본과 비교하였다.

- 파드의 점수 산출 과정에서 역방향 예측 부정확성 제거(remove reverse prediction inaccuracy from Pods): 이 경우 파드가 서비스와 노드 각각으로부터 받는 점수 $RCS_{\mathrm{PS}}^{(k)}$와 $RCS_{\mathrm{PN}}^{(k)}$는 다음과 같이 정의된다.

$$RCS_{\mathrm{PS}}^{(k)} = \max_t \left| AS_{\mathrm{MP}k}^{(t)} \right| \cdot L_{\mathrm{MP}k,\mathrm{MS}k}$$

$$RCS_{\mathrm{PN}}^{(k)} = \max_t \left| AS_{\mathrm{MP}k}^{(t)} \right| \cdot L_{\mathrm{MP}k,\mathrm{MN}k}$$

- 서비스 및 노드의 점수 산출 과정에서 역방향 예측 부정확성 추가(add reverse prediction inaccuracy for Service and Node): 이 경우 서비스와 노드 각각의 점수 $RCS_{\mathrm{S}}^{(k)}$와 $RCS_{\mathrm{N}}^{(k)}$는 다음과 같이 정의된다.

$$RCS_{\mathrm{S}}^{(k)} = \max_t \left| AS_{\mathrm{MP}k}^{(t)} \right| \cdot \max(L_{\mathrm{MS}k,\mathrm{MP}k}, 1 - L_{\mathrm{MP}k,\mathrm{MS}k})$$

$$RCS_{\mathrm{N}}^{(k)} = \max_t \left| AS_{\mathrm{MP}k}^{(t)} \right| \cdot \max(L_{\mathrm{MN}k,\mathrm{MP}k}, 1 - L_{\mathrm{MP}k,\mathrm{MN}k})$$

또한, AnoProp은 리소스를 공유하는 메트릭들의 영향을 이들의 부모 메트릭으로 모델링함으로써, 메트릭의 이상치가 과소평가되는 문제를 해결하였다. 이를 검증하기 위해, 리소스를 공유하는 메트릭들을 직접 회귀 모델의 입력으로 사용하는 변형 기법을 원본과 비교하였다.

- 메트릭 값 예측 과정에서 리소스를 공유하는 메트릭들의 값을 직접 입력으로 사용(directly use resource-sharing metrics as inputs): 이 경우 메트릭 M의 변형 회귀 모델 f''_{M}은 다음과 같이 정의된다. 여기서 $res(\mathrm{M})^{(t)}$는 M과 동일한 리소스를 공유하는 메트릭들이 시간 t에서 가지는 값을 의미한다.

$$\bar{v}_{\mathrm{M}}^{(t)} = f''_{\mathrm{M}}(pa(\mathrm{M})^{(t)}, res(\mathrm{M})^{(t)})$$

표 2는 변형 실험의 결과를 보여준다. 원본 AnoProp은 대부분의 지표에서 다른 변형 기법들을 상회하는 성능을 보였으며, 특히 Hit@1 지표에서 두드러진 성능 차이를 나타냈다. 이는 원본 AnoProp이 리소스 유형별 특성을 효과적으로 반영한 설계임을 시사한다. 또한, 근본 원인 점수 평가 기법의 두 가지 변형 모델 모두 기준 모델 중 가장 높은 성능을 보였던 LatentScope과 비교했을 때 비슷하거나 더 높은 성능을 기록하여, 이상치 전파율을 활용해 근본 원인 점수를 평가하는 접근법이 근본 원인 예측에 적합함을 입증하였다.

리소스를 공유하는 메트릭을 직접 회귀 모델의 입력으로 사용하는 변형 모델의 경우에는, 리소스 공유의 영향을 고려하지 않는 CIRCA보다도 낮은 성능을 나타냈다. 이는 리소스를 공유하는 메트릭들의 영향을 반영할 때, 상호 의존성을 제거하는 과정이 정확한 이상치 측정에 필수적임을 보여준다.

5.4 소거 실험

AnoProp을 구성하는 두 기법인 이상치 측정 기법과 근본 원인 평가 기법의 성능 기여도를 분석하기 위해, 두 기법 중 하나를 제거하고 LatentScope의 기법으로

표 2 변형 실험 결과
Table 2 Modification Experiment Results

	Micro				Macro			
	Hit@1	Hit@5	MRR	MRR@5	Hit@1	Hit@5	MRR	MRR@5
CIRCA	0.2159	0.8636	0.4633	0.4393	0.2804	0.8847	0.4537	0.4344
LatentScope	0.3864	0.9091	0.6060	0.5915	0.4615	0.9159	0.6569	0.6447
Remove reverse prediction inaccuracy from Pods	0.4170	0.8977	0.6017	0.5836	0.4716	0.8969	0.6408	0.6245
Add reverse prediction inaccuracy to Services & Nodes	0.4170	0.9091	0.6222	0.6045	0.4989	0.9097	**0.6756**	**0.6596**
Directly use resource-sharing metrics as inputs	0.2102	0.7727	0.4215	0.3620	0.2157	0.8174	0.4456	0.3960
AnoProp	**0.4625**	**0.9205**	**0.6367**	**0.6140**	**0.5079**	**0.9287**	0.6717	0.6534

표 3 소거 실험 결과
Table 3 Ablation Experiment Results

	Micro				Macro			
	Hit@1	Hit@5	MRR	MRR@5	Hit@1	Hit@5	MRR	MRR@5
LatentScope	0.3864	0.9091	0.6060	0.5915	0.4615	0.9159	0.6569	0.6447
AnoProp	**0.4625**	0.9205	**0.6367**	0.6140	**0.5079**	0.9287	0.6717	0.6534
Only use anomaly measurement technique of AnoProp	0.4091	**0.9318**	0.6249	**0.6150**	0.4810	**0.9354**	**0.6722**	**0.6630**
Only use root cause score evaluation technique of AnoProp	0.4460	0.9091	0.6282	0.6051	0.4666	0.9221	0.6503	0.6320

대체하는 소거 실험을 수행하였다.

- AnoProp의 이상치 측정 기법만 적용(only use anomaly measurement technique of AnoProp)
- AnoProp의 근본 원인 평가 기법만 적용(only use root cause score evaluation technique of AnoProp)

표 3은 소거 실험의 결과를 나타낸다. 원본 AnoProp은 전체적으로 가장 높은 성능을 기록하였으며, 두 기법 중 하나를 제거하면 원본 설계 대비 성능이 감소하였다. 이는 두 기법이 결합하여 사용될 때 더욱 효과적임을 시사한다. 한편, 하나의 기법만을 적용한 경우에도 여전히 기준 모델보다 높은 성능을 보였으며, 이는 각 기법이 단독으로도 모델의 성능에 크게 기여함을 보여준다.

6. 결론 및 향후 연구

본 논문에서는 마이크로서비스의 근본 원인 분석을 위해, 리소스 간 상호작용과 이상치 전파를 통합적으로 고려하는 모델인 AnoProp을 제안하였다. AnoProp은 리소스로 인한 메트릭 간 연관성을 모델링함으로써, 결함의 원인 리소스를 효과적으로 포착하였다. 실제 마이크로서비스 결함 데이터셋을 활용한 실험에서 AnoProp은 기준 모델과 비교하여 우수한 성능을 보였으며, 결함 원인 종류별로 균형 잡힌 성능을 제공하였다.

향후 연구에서는 보다 다양한 데이터셋을 활용하여 모델의 일반화 성능을 면밀히 검증하는 과정이 필요하다. 본 연구에서는 공개 데이터셋의 제한으로 인하여 특정 테스트베드 환경에서 수집된 데이터셋만을 활용하였으나, 실시간으로 변화하는 실제 산업 환경에서도 일관된 성능을 보장하기 위해서는 다양한 실사용 사례에서의 평가가 요구된다. 이를 위해, 실제 운영 환경에서 수집된 마이크로서비스 데이터를 활용한 추가 실험이 중요한 과제가 될 것이다.

또한, 그래프 구조를 더욱 효과적으로 활용하는 GNN (Graph Neural Networks)과 같은 최신 기술을 접목하는 가능성을 고려할 수 있다. DéjàVu[17]는 지도 학습(supervised learning)을 통해 오류가 발생한 리소스를 탐색하는 기계학습 모델을 제시한 바 있다. 그러나 이는 대규모 학습 데이터가 필요하며, 실시간으로 변화하는 마이크로서비스 환경에 즉각적으로 적용하기 어렵다는 한계가 있다. 이에 따라, 새로운 그래프 구조에도 유연하게 적용 가능한 방법론이 필요하며, 이를 위해 지식 그래프와 같은 외부 지식을 활용하는 방안이 유용할 수 있다[18]. 나아가, 근본 원인 분석에 적합한 형태로 그래프 구조를 확장하거나[19], 이를 시공간 그래프(spatio-temporal graph)에 대입하여 문제를 해결하는 접근도 고려될 수 있다[20]. AnoProp은 이러한 연구의 기반이 되어 마이크로서비스 시스템의 안정성을 강화하고 운영 효율성을 향상하는 데 기여할 것으로 기대된다.

References

[1] J. Soldani, and A. Brogi, "Anomaly Detection and Failure Root Cause Analysis in (Micro) Service-Based Cloud Applications: A Survey," *ACM Computing Surveys*, Vol. 55, No. 3, pp. 1-39, Feb. 2022.

[2] J. Lin, P. Chen, and Z. Zheng, "Microscope: Pinpoint Performance Issues with Causal Graphs in Micro-service Environments," *Proc. of Service-Oriented Computing: 16th International Conference,* pp. 3-20, 2018.

[3] D. Yuan, H. Mai, W. Xiong, L. Tan, Y. Zhou, and S. Pasupathy, "SherLog: Error Diagnosis by Connecting Clues from Run-time Logs," *Proc. of the 15th International Conference on Architectural support for programming languages and operating systems,* pp. 143-154, 2010.

[4] Z. Li, J. Chen, R. Jiao, N. Zhao, Z. Wang, S. Zhang, Y. Wu, L. Jiang, L. Yan, Z. Wang, Z. Chen, W. Zhang, X. Nie, K. Sui, and D. Pei, "Practical Root Cause Localization for Microservice Systems via Trace Analysis," *Proc. of the 2021 IEEE/ACM 29th International Symposium on Quality of Service,* pp. 1-10, 2021.

[5] S. Gu, G. Rong, T. Ren, H. Zhang, H. Shen, Y. Yu, X. Li, J. Ouyang, and C. Chen, "TrinityRCL: Multi-Granular and Code-Level Root Cause Localization

Using Multiple Types of Telemetry Data in Microservice Systems," *IEEE Transactions on Software Engineering*, Vol. 49, No. 5, pp. 3071-3088, Feb. 2023.
[6] Z. Xie, S. Zhang, Y. Geng, Y. Zhang, M. Ma, X. Nie, Z. Yao, L. Xu, Y. Sun, W. Li, and D. Pei, "Microservice Root Cause Analysis With Limited Observability Through Intervention Recognition in the Latent Space," *Proc. of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining,* pp. 6049-6060, 2024.
[7] M. Luksa. *Kubernetes in Action,* Simon and Schuster, 2017.
[8] A. Jindal, V. Podolskiy, and M. Gerndt, "Performance Modeling for Cloud Microservice Applications," *Proc. of the 2019 ACM/SPEC International Conference on Performance Engineering,* pp. 25-32, 2019.
[9] B. Beyer, C. Jones, J. Petoff, and N. R. Murphy. *Site Reliability Engineering: How Google Runs Production Systems,* O'Reilly Media, Inc, 2016.
[10] M. Li, Z. Li, K. Yin, X. Nie, W. Zhang, K. Sui, and D. Pei, "Causal Inference-Based Root Cause Analysis for Online Service Systems with Intervention Recognition," *Proc. of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 3230-3240, 2022.
[11] GoogleCloudPlatform. (2024, Nov. 9). Online Boutique (v0.10.2) [Online]. Available: https://github.com/GoogleCloudPlatform/microservices-demo
[12] M. Kim, R. Sumbaly, and S. Shah, "Root cause detection in a service-oriented architecture," *ACM SIGMETRICS Performance Evaluation Review,* Vol. 41, No. 1, pp. 93-104, Jun. 2013.
[13] P. Wang, J. Xu, M. Ma, W. Lin, D. Pan, Y. Wang, and P. Chen, "CloudRanger: Root Cause Identification for Cloud Native Systems," *Proc. of the 2018 18th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing,* pp. 492-502, 2018.
[14] J. Weng, J. H. Wang, J. Yang, and Y. Yang, "Root Cause Analysis of Anomalies of Multitier Services in Public Clouds," *IEEE/ACM Transactions on Networking,* Vol. 26, No. 4, pp. 1646-1659, Jun. 2018.
[15] Y. Meng, S. Zhang, Y. Sun, R. Zhang, Z. Hu, Y. Zhang, C. Jia, Z. Wang, and D. Pei, "Localizing Failure Root Causes in a Microservice Through Causality Inference," *Proc. of the 2020 IEEE/ACM 28th International Symposium on Quality of Service,* pp. 1-10, 2020.
[16] D. Liu, C. He, X. Peng, F. Lin, C. Zhang, S. Gong, Z. Li, J. Ou, and Z. Wu, "MicroHECL: High-Efficient Root Cause Localization in Large-Scale Microservice Systems," *Proc. of the 2021 IEEE/ACM 43rd International Conference on Software Engineering: Software Engineering in Practice,* pp. 338-347, 2021.
[17] Z. Li, N. Zhao, M. Li, X. Lu, L. Wang, D. Chang, X. Nie, L. Cao, W. Zhang, K. Sui, Y. Wang, X. Du, G. Duan, and D. Pei, "Actionable and interpretable fault localization for recurring failures in online service systems," *Proc. of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pp. 996-1008, 2022.
[18] J. Lee, C. Chung, and J. J. Whang, "InGram: Inductive Knowledge Graph Embedding via Relation Graphs," *Proc. of the 40th International Conference on Machine Learning*, pp. 18796-18809, 2023.
[19] C. Chung, J. Lee, and J. J. Whang, "Representation Learning on Hyper-Relational and Numeric Knowledge Graphs with Transformers," *Proc. of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining,* pp. 310-322, 2023.
[20] J. Choi, H. Kim, M. An, and J. J. Whang, "SpoT-Mamba: Learning Long-Range Dependency on Spatio-Temporal Graphs with Selective State Spaces," *Proc. of the 3rd International Workshop on Spatio-Temporal Reasoning and Learning co-located with the 33rd International Joint Conference on Artificial Intelligence*, 2024.

박 준 호

2023년 GIST 전기전자컴퓨터공학부 졸업(학사). 2025년 KAIST 전산학부 졸업(석사). 2025년~현재 그래파이 Graph 팀. 관심분야는 그래프 기계학습, 마이크로서비스 근본 원인 분석, 데이터 마이닝

황 지 영

2010년 이화여자대학교 컴퓨터공학과 졸업(학사). 2015년 텍사스 오스틴 대학교(The University of Texas at Austin) 컴퓨터 과학과 졸업(박사). 2016년 3월~2020년 6월 성균관대학교 소프트웨어학과 조교수(빅데이터 연구실 운영), 2020년 7월~2023년 8월 KAIST 전산학부 조교수, 2023년 9월~현재 KAIST 전산학부 부교수(빅데이터 지능 연구실 운영). 관심분야는 그래프 기계학습, 데이터 마이닝, 빅데이터 분석