ChatGPT

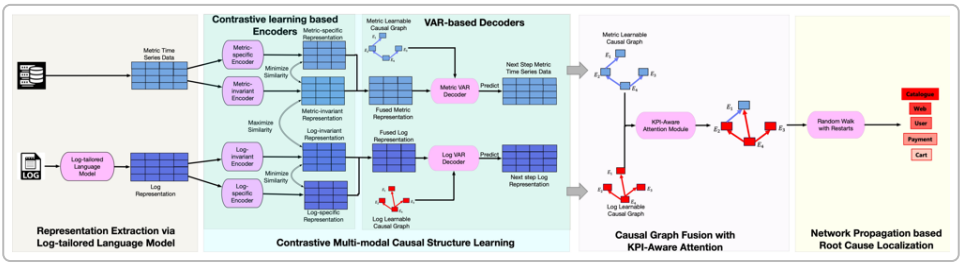
"Multi-modal Causal Structure Learning and Root Cause Analysis" Makalesinin Detaylı İncelemesi

Makalenin Genel Konusu ve Temel Amacı

Bu makale, karmaşık bilgisayar sistemlerinde **kök neden analizi** (Root Cause Analysis, RCA) problemini ele almaktadır. RCA, büyük ölçekli mikroservis mimarileri gibi kompleks sistemlerde meydana gelen arızaların kaynağını hızlıca tespit etmeyi amaçlar. Mevcut sistemlerin sürekliliğini sağlamak için metrik (ör. CPU bellek kullanımı gibi sayısal zaman serileri) ve log kayıtları (sistem günlükleri) gibi çeşitli veri türleri sürekli izlenir [1]. Ancak geçmişteki veri odaklı RCA yöntemleri genellikle **tek bir veri moduna** (ör. sadece metrikler) dayanarak nedensel veya bağımlılık grafikleri oluşturmuş ve bu grafikler üzerinden geriye doğru izleme ile arızanın nedenini bulmaya çalışmışlardır [2]. Sadece tek modlu (tek kaynaklı) veriye dayanmak, bazı arıza türlerinin gözden kaçmasına veya eksik analiz edilmesine yol açabilmektedir. Örneğin, veritabanı sorgu hatası veya kullanıcı giriş hatası gibi sorunlar, yalnızca metrik verilerine bakılarak anlaşılamaz; bunların kök nedeni genellikle log mesajlarında saklıdır. Tersine, disk dolması gibi bazı sorunlarda hem metrik veriler (ör. disk kullanım oranı) hem de log mesajları birlikte ipucu verir [3]. Bu nedenle **çok modlu veri** (hem metrik hem log) kullanarak daha bütüncül bir RCA yapmak önem taşır [4].

Makalenin temel amacı, **çok modlu nedensel yapı öğrenimi** yaklaşımıyla RCA performansını artırmaktır. Yazarlar, **MULAN** adını verdikleri birleşik bir çerçeve önererek, sistem metrikleri ile sistem loglarını birlikte analiz edip KPI (Key Performance Indicator – Temel Performans Göstergesi, ör. sistem gecikmesi gibi bir hedef metrik) ile en ilgili sistem bileşenlerini (mikroservisleri) nedensel bir grafik üzerinde belirlemeyi hedeflemektedir [2] [5]. Bu kapsamda makale üç ana zorluğa çözüm getiriyor: (C1) **Log verilerinin temsil öğrenimi** – serbest biçimli metin loglarından anlamlı bir özellik çıkarımı yapmak, (C2) **Çok modlu nedensel ilişki öğrenimi** – iki farklı veri modundan gelen bilgileri hem ortak (moddan bağımsız) hem de mod-özel şekilde yakalamak, ve (C3) **Mod güvenilirliğinin değerlendirilmesi** – gürültülü veya düşük kaliteli bir veri modunun analizdeki olumsuz etkisini azaltmak [6] [7]. Makalede sunulan yöntem, bu zorlukları aşmak için derin öğrenme tabanlı bir mimari, büyük dil modeli kullanımı, karşıt-öğrenme (contrastive learning) teknikleri ve grafik tabanlı nedensellik öğrenimi bileşenlerini bir araya getiriyor. Sonuç olarak, önerilen **MULAN** yöntemiyle tek modlu yaklaşımların eksik kaldığı durumlarda daha yüksek doğrulukla kök neden tespiti yapılabildiği ve hizmet kesintilerinin daha hızlı giderilebileceği gösterilmiştir [2] [8].

Önerilen Yöntemin Mimarisi ve Teknik Detayları



Şekil 1: Önerilen MULAN çerçevesinin genel mimarisi [9]. Yöntem, dört ana modülden oluşmaktadır: (1) Log-tailored dil modeli ile log verilerinden temsil çıkarma, (2) Karşıt-öğrenme tabanlı çok modlu nedensel yapı öğrenimi (moddan bağımsız ve mod-spesifik özellik çıkarımı), (3) KPI-farkındalıklı dikkat mekanizması ile nedensel grafik birleştirme (farklı modlardan gelen grafiklerin güven ağırlıklı füzyonu) ve (4) Ağ yayılımı tabanlı kök neden belirleme (birleşik graf üzerinde random walk ile en olası neden düğümlerinin bulunması). Her modülün detayları aşağıda açıklanmaktadır. [9] [10]

1. Log Verilerinin Dil Modeli ile Temsil Çıkarımı

İlk modül, ham sistem loglarını uygun bir biçimde sayısal temsillere dönüştürerek **zaman serisi verisi** haline getirmektir [11]. Serbest biçimli metin loglarının dil bilgisi kurallarından yoksun olması ve birç**ok özel simge içermesi**, onları doğrudan standart dil modelleriyle işlemede zorluk çıkarır [12]. Bu nedenle yazarlar, log verilerine özel uyarlanmış bir dil modeli yaklaşımı önermektedir (Log-tailored Language Model) [13]. Bu yaklaşım üç aşamada gerçekleştirilir:

- **Aşama 1 (Önişleme - Log Şablonlarının Çıkarılması):** Ham log metinleri, örüntü tabanlı bir log ayrıştırıcı ile işlenir. Makalede, yaygın bir araç olan **Drain** log ayrıştırıcısı kullanılarak benzer yapıya sahip log mesajları için **log şablonları** (template) çıkarılmaktadır [14]. Her log satırı, içeriğine göre bir şablon kategorisine atanır. Böylece yapılandırılmamış metin yerine, sistemde gözlenen birkaç temel log şablonu elde edilir.

- **Aşama 2 (Log Dizisi Oluşturma):** Tüm log kayıtları, sabit uzunlukta zaman pencerelerine bölünür. Her bir mikroservis veya sistem bileşeni için, her zaman penceresinde o bileşene ait bir **log dizisi** oluşturulur [15]. Bu dizide, ilgili zaman aralığında o bileşenden üretilen **benzersiz log şablonları**, gerçekleşme sırasına göre sıralanır [15]. Örneğin, 5 dakikalık bir pencere içinde bir veritabanı servisi 3 çeşit log mesajı üretmişse, o pencere için veritabanı servisine ait log dizisi [ŞablonA, ŞablonB, ŞablonC] şeklinde olur. Metin verisini işlemede verimli olması için, her log şablonu bir token (özel bir kelime gibi) kabul edilmiştir; yani log dizilerini geleneksel cümleler gibi düşünürsek, her "kelime" bir log şablonudur [15]. **Log frekansı** da önemli bir bilgidir: Belirli bir pencerede aynı şablondan çok sayıda üretilmesi, anormal bir durumu işaret edebilir. Bu nedenle, her log şablonunun hemen ardından o penceredeki tekrar sayısı da diziye eklenir [16]. Örneğin dizimiz aslında ["ŞablonA", 5, "ŞablonB", 2, "ŞablonC", 1] gibi olacaktır (5 kez A tipi log, 2 kez B tipi, 1 kez C tipi gibi). Böylece log dizilerinin uzunluğu makul seviyede tutulur ve frekans bilgisi de temsil edilmeye dahil edilir. Ayrıca, her log dizisine ilişkin bir **anomali skoru** elde edilir: Bunun için mevcut log anomali tespit algoritmaları (ör. **OC4Seq** ya da **DeepLog**) kullanılarak, her pencere için logların ne derece anormal olduğu sayısal olarak skorlanır [17]. Bu anomali skoru ilgili pencerenin "etiketi" olarak kullanılacaktır.

- **Aşama 3 (Dil Modeli ile Temsil Öğrenimi):** Her bir log dizisi, önceden eğitilmiş bir dil modeli (örneğin BERT türevi bir transformer modeli) üzerinde işlenerek bir sabit boyutlu vektör ile temsil edilir [18]. Yazarlar, dil modelini *regresyon tabanlı* bir yaklaşımla ince ayar yapmaktadır: Modelin çıktısı, ilgili log dizisinin anomali skorunu tahmin etmeye çalışır [19]. Bu amaçla dil modelinin sonuna tek katmanlı bir **çok katmanlı algılayıcı** (MLP) eklenmiş ve **[CLS] özel tokeninin çıktısı** kullanılarak anomali skorunu veren bir regresyon hedefi tanımlanmıştır [18]. Kayıp fonksiyonu (Loss) olarak tahmin edilen skor ile gerçek anomali skoru arasındaki hatayı minimize eden bir yöntem (örn. ortalama kareler hatası) kullanılır (denklem 3 ile gösterilmiştir) [19]. Böylece model, log dizisinin içeriğini öyle bir vektör uzayına yerleştirir ki bu vektörden anomali düzeyi çıkarılabilir hale gelir. Eğitim tamamlandığında, dil modelinin ürettiği **[CLS] tokenine ait vektör** ilgili log dizisinin (yani bir servis için belli bir zaman aralığındaki logların) temsili olarak alınır [18]. Bu işlem her servis ve her zaman penceresi için yapılarak, tüm sistem için **zaman serisi biçiminde log temsilleri** elde edilir. Gerekirse bu yüksek boyutlu dil modeli çıktı vektörleri, **Temel Bileşen**

Analizi (PCA) gibi tekniklerle boyut indirgemeye tabi tutulabilir (örneğin makalede boyut birkaç yüzden ~50'ye düşürülmüştür) [20]. Sonuç olarak, m adet sistem bileşeni varsa, her biri için her zaman adımında bir log temsil vektörü ($\mathbf{h}^{log}$) ve benzer şekilde her bir metrik için de o anda bir metrik değeri (veya metrik özelliği $\mathbf{h}^{metric}$) bulunacaktır [21]. Bu veriler sonraki adıma beslenir.

2. Karşıt-Öğrenme ile Çok Modlu Nedensel Yapı Öğrenimi

Metrik zaman serileri (nümerik veriler) ile log tabanlı temsil serileri (metin kaynaklı özellikler) elde edildikten sonra, ikinci modül bu **iki modun ortak nedensel yapısını** öğrenmeye odaklanır. Amaç, hem **moddan bağımsız** (iki veride ortak) hem de **moda özel** (her veriye özgü) bilgi bileşenlerini yakalamaktır [22]. Bu doğrultuda yazarlar, her iki veri modu için paralel birer **kodlayıcı (encoder)** ve **çözümleyici (decoder)** mimarisi kurgulamıştır [23] [24]. Ana bileşenler şöyle:

- **Graf Tabanlı Kodlayıcılar:** Metrik veriler ve log temsilleri, her ikisi de sistemdeki bileşenleri düğümler olarak içeren bir grafik yapısı üzerinde modellenir. Yani servisler arası nedensel etkileşimleri yakalayabilmek için bir **öğrenilebilir komşuluk matrisi (adjacency matrix)** tanımlanır; başlangıçta bu matris bilinmiyor ve eğitim sürecinde öğreniliyor [25]. Yazarlar, her iki mod için de birer **GraphSAGE** tabanlı grafik sinir ağı kullanıyorlar [26]. GraphSAGE, düğümlerin kendi komşularının özelliklerini öğrenerek düğüm temsilini güncelleyen bir grafik öğrenme modelidir. Burada iki tür GraphSAGE encoder var: **Moddan bağımsız (invariant) encoder** ve **moda özgü (specific) encoder** [23]. İnvariant encoder, her mod verisinden gelen girdileri işleyerek modlar arasında ortak olan yönleri yakalamaya çalışırken; specific encoder ise her modun kendine has, diğerinde bulunmayan bilgi yönlerini modele dahil eder. Bu encoder'ların çıktıları şu şekilde özetlenebilir: Her bir mod için her bir düğüm (servis) için bir **modality-invariant representation** (z^{inv}) ve bir **modality-specific representation** (z^{spec}) elde edilir [27]. Örneğin metrik veriler için $z^{metric, inv}$ ve $z^{metric, spec}$; log verileri için $z^{log, inv}$ ve $z^{log, spec}$ hesaplanır. İnvariant temsil, metrik ve log verisinin ortak paternlerini yakalarken; spesifik temsil, sadece o modda görülen özel sinyalleri taşır.

- **Karşıt-öğrenme (Contrastive) Düzenlemeleri:** Modlar arası uyumu sağlamak ve gereksiz tekrarları önlemek adına iki ek kısıt uygulanır. İlki, **moddan bağımsız temsillerin birbirine benzer kılınması**dır. Metrik ve log için elde edilen invariant temsil vektörleri karşılaştırılır; aralarındaki **benzerlik (cosine benzerliği)** olabildiğince artırılmaya çalışılır [28]. Bu, her bir servis düğümü için "ortak" nedensel faktörlerin her iki modda da benzer şekilde temsil edilmesini sağlar. Yazarlar bunu sağlamak için bir **karşıt öğrenme kaybı** tanımlamışlardır: İki modun invariant vektörleri arasındaki bilgi ortaklığını maksimize eden bir düzenleyici terim (denkle**m 5**, cosine benzerliğiyle ölçülüyor) optimizasyon hedeflerine eklenir [29]. İkinci kısıt ise, **invariant ve spesifik temsillerin ayrıştırılması**dır. Her modun kendi z^{inv} ve z^{spec} vektörlerinin içerik olarak örtüşmemesi istenir. Bunun için **ortogonallik kısıtı** uygulanır: z^{inv} ile z^{spec} arasındaki iç çarpım benzeri bir ölçütü minimize eden bir terim (denklem 6) kayıp fonksiyonuna eklenir [30]. Böylece model, ortak bilgiyi ve mod-özel bilgiyi ayrı eksenlere iterek aynı bilgiyi iki kez temsil etmemeyi öğrenir.

- **Nedensel Bağlantı (Kenar) Tahmini:** Sadece temsillerin öğrenilmesi yetmez; bu temsillerin gerçekten **nedensel grafiği** yansıtacak biçimde olmasını sağlamak gerekir. Bunun için yazarlar, her bir potansiyel **kenar (düğüm çifti)** için bir tahmin mekanizması eklemişlerdir [31]. Öğrenilen düğüm temsilleri kullanılarak, iki servis arasındaki etkileşimin var olup olmadığını tahmin eden bir **MLP sınıflandırıcı** tanımlanır. Özellikle, iki düğümün (servisin) temsil vektörleri birleştirilerek bir kenar vektörü oluşturulur ve bunun üzerinden sigmoid aktivasyonlu bir tek katmanlı ağ ile "bu iki düğüm arasında nedensel bir ilişki var mı?" sorusu yanıtlanır [32]. Bu tahminin hatası da

(kenar var/yok hatası) eğitim kaybına **kenar kaybı (edge loss)** olarak eklenir. Bu sayede model, temsilleri öğrenirken aynı zamanda bu temsillerin oluşturduğu grafiğin anlamlı (gerçek nedensel ilişkilere yakın) olmasını sağlamaya çalışır. Özellikle bu kenar tahmini, sistemdeki servis düğümleri ile KPI düğümü arasındaki ilişkileri de içerir (grafikte KPI da bir düğüm olarak yer alır) [33]. Sonuç olarak, eğitim süreci boyunca model hem temsilleri iyileştirir hem de altında yatan en olası nedensel grafiği (komşuluk matrisi A'yı) öğrenir [25].

- **VAR Tabanlı Çözümleyiciler:** Kodlayıcılar sonunda, her iki mod için de düğümlerin belirli bir andaki gizli temsil vektörleri ve ilişkisel bir komşuluk yapısı elde edildi. Sıradaki adım, bu yapıdan hareketle sistemin dinamik davranışını modellemektir. Bunun için her mod (metrik ve log) verisi üzerinde ayrı ayrı birer **Vektör Otoregresyon (VAR)** tabanlı **decoder** (çözümleyici) kullanılmıştır [24]. VAR modeli, bir zaman serisinin kendi geçmiş değerlerine (ve burada diğer ilgili serilere) bakarak gelecekteki değerini tahmin etmeye çalışır. Makaledeki formülasyona göre, önceki p adımlık veriden sonraki adım tahmin edilir (denklem 8) [24]. GraphSAGE modeli burada da gömülü kullanılır; her bir modun decoder'ı, kendi öğrenilmiş nedensel komşuluk yapısını (learnable causal graph) kullanarak, t zaman adımında tüm düğümlerin temsilinden $t+1$ adımındaki değeri öngörür. Metrik decoder, metrik zaman serilerinin geleceğini; log decoder ise log temsillerinin (örneğin anomali skorlarının) geleceğini tahmin etmeye çalışır. Decoder'ların eğitimi, tahmin ettikleri değerler ile gerçek değerler arasındaki hata (örn. VAR tahmin hatası) üzerinden yapılır. Bu sayede her mod için **öğrenilmiş nedensel grafik** yapıları elde edilir: Biri metrik veriden türetilmiş grafik (A_{metric}), diğeri log veriden türetilmiş grafik (A_{log}). Bu grafikler, aslında ilgili moddaki değişkenlerin (servislerin ve KPI'nın) etkileşimini gösteren yönlü ilişki ağlarıdır.

3. KPI-Duyarlı Dikkat Mekanizması ile Nedensel Graf Birleştirme

Elde edilen iki ayrı grafiği (metrik tabanlı ve log tabanlı nedensel grafikleri) uygun şekilde birleştirerek tek bir **birleşik nedensel grafik** elde etmek gerekir. Basitçe bu grafiklerin adjacency matrislerini toplayıp birleştirmek doğru değildir; çünkü bu durumda çok yoğun (her yerde bağlantı olan) veya döngülü graf yapıları ortaya çıkabilir [34]. Ayrıca üçüncü zorluk olarak belirtilen **düşük kaliteli mod** sorunu da burada devreye girer: Eğer iki veri modundan biri çok gürültülü veya hatalı bilgiler içeriyorsa, her ikisine eşit güvenmek sonucu bozabilir [35]. Yazarlar bu nedenle **KPI-farkındalıklı bir dikkat (attention) mekanizması** tasarlamışlardır [36]. Bu mekanizma, her bir modun ne derece güvenilir olduğuna dair bir ağırlık hesaplar.

Öncelikle, her bir mod için her bir servis düğümünün ham özellikleri ile KPI arasındaki **zaman serisi korelasyonu** ölçülür. Spesifik olarak, **çapraz korelasyon (cross-correlation)** analizi ile belli bir maksimum gecikmeye (L time lag) kadar, ilgili servis metriği ile KPI metriği arasındaki benzerlik değerlendirilir [37]. Denklem 9'da tanımlandığı üzere, $corr_{X,Y}$ ifadesi, örneğin bir servis X ile KPI (Y) arasında en yüksek korelasyon değerini (mutlak veya belirli bir lag değerinde) temsil eder [37]. Bu işlem hem metrik verisi için (servisin metrik serisi vs KPI serisi) hem de log verisi için (servisin log anomali skoru serisi vs KPI) yapılır. Elde edilen korelasyon katsayıları, ilgili mod verisinin KPI ile ne kadar senkronize hareket ettiğinin bir göstergesidir; dolayısıyla bir nevi **mod kalite göstergesi** olarak yorumlanabilir [38]. Örneğin, eğer bir moddaki (metrik ya da log) en önemli birkaç servisin sinyalleri KPI ile güçlü korelasyon gösteriyorsa, o modun arızayı yakalamada yüksek kaliteli olduğunu düşünebiliriz. Tam tersi, bir moddaki hiçbir servis sinyali KPI ile anlamlı ilişki göstermiyorsa, o mod gürültülü veya düşük kaliteli olabilir.

İkinci adımda, her iki mod için bulunan bu korelasyon skorlarına bir **softmax normalizasyonu** uygulanır ve **mod ağırlıkları** elde edilir (denklem 10) [39]. Softmax işlemi, metrik mod ve log mod için bulunan korelasyon ölçülerini karşılaştırarak toplamı 1 olacak şekilde birer önem değeri çıkarır. Örneğin,

metrik veri daha belirgin sinyaller içeriyorsa ağırlığı $\alpha \approx 0.8$ ve log verisi için $\alpha \approx 0.2$ gibi bir dağılım olabilir. Yazarlar, bu mekanizmanın etkinliğini makalede göstermiştir: Gerçek vakalarda yüksek kaliteli moda yüksek, düşük kaliteli olana düşük ağırlık verildiği gözlemlenmiştir [40].

Son olarak, hesaplanan bu **mod önem ağırlığı** kullanılarak iki grafik birleştirilir. Metrik grafının tüm kenar ağırlıkları (adjacency) kendi mod ağırlığı ile çarpılır, log grafının kenarları da kendi ağırlığı ile çarpılır ve sonra matrisler toplanır (denklem 11) [41]. Basitçe, güvenilen modun ilişkileri daha baskın olacak şekilde grafikler iç içe geçirilir. Bu işlem sonucunda **fused adjacency matrix** denilen, her bir kenarın ağırlığının ilgili modların katkısına göre belirlendiği birleşik bir nedensel komşuluk matrisi elde edilir [42]. Bu birleşik graf yine sistem servisleri ve KPI düğümünü içerir, yani sorunun tüm alanını kapsar.

Birleştirme sonrasında, elde edilen grafın **seyrekliği ve çevrimsizliği** (yani bir DAG – Directed Acyclic Graph olması) sağlanmalıdır. Modelin öğrenme hedefinde, cezalandırıcı bir **sparsity (seyreklik) terimi** ve NOTEARS yaklaşımlarından uyarlanan bir **acyclicity terimi** de bulunmaktadır [43]. Yazarlar, Pamfil et al. (2020) çalışmasına referansla, yönlü grafiklerde döngü olmamasını sağlamak için **iz (trace) log determinant** fonksiyonunu kullanmaktadır (bu fonksiyon, adjacency matrisinin üstel formuyla birlikte, graf çevrimsiz ise belirli bir koşulu sağlar) [43]. Kaynak makalede denklem 12, modelin nihai optimizasyon amacını göstermektedir: VAR tahmin hatası, karşıt-öğrenme (invariant benzerliği) kaybı, ortogonallik kaybı, kenar tahmin kaybı, mod ağırlık mekanizması ve DAG kısıtı gibi terimler, belirli hiperparametre ağırlıklarıyla toplanarak toplam kaybı verir [44]. Eğitim sürecinde bu toplam kayıp minimize edilerek model parametreleri (dil modeli, GraphSAGE encoder/decoder'ları, adjacency matrisi vb.) öğrenilir [44].

4. Ağ Yayılımı Tabanlı Kök Neden Belirleme

Son modül, elde edilen birleşik nedensel grafi kullanarak **kök neden** olabilecek düğümleri tespit etmektir. Bir arıza durumunda, graf üzerindeki etkiler genellikle **kaynak noktadan (kök neden olan servisten)** başlayıp zamanla komşu bileşenlere doğru yayılır [45]. Bu, arıza anında KPI'nın doğrudan bağlı olduğu servis düğümünün her zaman asıl sorun kaynağı olmayabileceği anlamına gelir – belki de birkaç adım öteden gelen bir sorun, KPI'nın bağlı olduğu servisi etkileyerek dolaylı bir etki yaratmıştır. Dolayısıyla graf üzerinde bir etki yayılım analizi yapmak gerekir.

Yazarlar bu amaçla **yeniden başlatmalı rastgele yürüyüş (Random Walk with Restart, RWR)** yöntemini kullanmıştır [46]. Bu, graf teorisinde bir düğümden rastgele yürüyüşler yaparak önemli düğümleri puanlamaya yarayan bir algoritmadır. Özellikle, birleşik nedensel graf için bir **geçiş olasılık matrisi** P oluşturulur (denklem 13) [47]. Bu matris, P_{ij} entry'sinin, graf üzerinde i düğümünden j düğümüne bir adımda geçiş olasılığını ifade edecek şekilde normalize edilir. Ardından, başlangıçta **KPI düğümüne** yüksek bir olasılık verilir ve rastgele yürüyüş süreci başlatılır. Rastgele yürüyüş, iteratif olarak $v(t) = (1-r) \cdot P^T \cdot v(t-1) + r \cdot v(0)$ formülüne göre güncellenir (denklem 14) [48]. Burada $v(t)$ tüm düğümlerin t. adımdaki olasılık dağılımıdır, r yeniden başlama (reset) olasılığıdır (her adımda r oranında tekrar KPI düğümünden başlama ihtimali), $v(0)$ ise başlangıç dağılımıdır (genellikle KPI düğümünde 1, diğerlerinde 0 şeklinde başlatılır). Bu denklemin her adımda uygulanmasıyla, KPI'dan başlayıp graf üzerinde dolaşan ve tekrar KPI'ya dönme ihtimali olan bir "yürüyüşçü" modellenir. Bu yürüyüş, grafik yapısındaki önemli düğümlerde daha fazla "oyalanma" eğilimindedir. Yeterli adım sonunda ($v(t)$ vektörü yakınsadığında), her bir düğüm için bir istasyoner olasılık elde edilir [49]. Bu değer, o düğümün KPI ile bağlantılı arıza yayılımındaki önemini ifade eder. Son olarak, düğümler bu olasılık skorlarına göre sıralanır ve en yüksek skora sahip olanlar **kök neden adayı** olarak seçilir [50]. Gerçek kullanımda genellikle en tepedeki birkaç düğüm (örneğin ilk 1, 3 veya 5 düğüm) potansiyel arıza nedenleri olarak operatöre sunulur.

Deneysel Kurulumlar ve Değerlendirme

Veri Setleri ve Deney Ortamı

Önerilen yöntemin etkinliği, **üç gerçek dünya mikroservis veri seti** üzerinde kapsamlı deneylerle değerlendirilmiştir. Bu veri setleri şunlardır [51]:

- **Product Review:** Çevrimiçi ürün yorum sistemi olarak tasarlanmış bir mikroservis uygulamasından toplanmıştır. Bulut sunucular üzerinde çalışan bu sistemde, Mayıs 2021 – Aralık 2021 arasında meydana gelen 4 farklı sistem arızası olayına ait veriler içerir [51].
- **Online Boutique:** Bir e-ticaret sitesi şeklinde tasarlanan mikroservis sisteminden elde edilmiştir. Bu sistemde 5 farklı arıza senaryosu gözlemlenmiştir [52].
- **Train Ticket:** Tren bileti rezervasyonu sağlayan bir mikroservis sistemine ait verilerden oluşur. Bu veri setinde de 5 farklı arıza durumu kayıt altına alınmıştır [53].

Her üç veri seti de **iki modaliteyi** içerir: (1) Zaman serisi şeklinde **sistem metrik verileri** (CPU kullanımı, bellek, gecikme vb. sayısal ölçümler) ve (2) aynı zaman aralıklarında toplanmış **sistem log (günlük) verileri** (mikroservislerin ürettiği metin log kayıtları). Bu sayede yöntemin, hem tekil modlar üzerinde hem de çok modlu senaryoda test edilmesi mümkün olmuştur.

Deneyler, Ubuntu 18.04 işletim sistemi yüklü bir sunucuda, Intel Xeon Silver 4110 işlemci ve dört adet NVIDIA GTX 2080 (11 GB) GPU kullanılarak gerçekleştirilmiştir [54]. Bu güçlü donanım, büyük dil modeli içeren log temsil öğrenimi ve GraphSAGE tabanlı grafik işlemlerini makul sürede tamamlamak için gereklidir.

Değerlendirme Metrikleri ve Karşılaştırmalar

Değerlendirme Metrikleri: Model performansını ölçmek için RCA literatüründe yaygın olarak kullanılan **üç** metrik tercih edilmiştir [55] [56]:

- **Precision@K (PR@K):** Her bir arıza vakasında modelin ilk K tahmininin ne kadar isabetli olduğunu gösterir. Örneğin $K=3$ için, gerçek kök neden(ler)in modelin ilk 3 tahmini içinde bulunma oranıdır [57]. PR@K değeri 1.0 ise model her seferinde doğru nedeni ilk 3 içinde buluyor demektir.
- **Mean Average Precision@K (MAP@K):** Bütün arıza vakalarını hesaba katarak, her bir vaka için Precision@K değerlerinin ortalamasını alan bir metriktir [58]. Kısaca, ilk K tahmin listelerinin genel başarısını özetler; değerin yüksek olması modelin ilk K tahminde genel olarak tutarlı biçimde doğru nedenleri yakaladığını gösterir.
- **Mean Reciprocal Rank (MRR):** Modelin tahmin sıralamasının kalitesini ölçer [59]. Her bir arıza için doğru nedenin kaçıncı sırada tahmin edildiğine bakar ve bunun tersinin ortalamasını alır. Örneğin gerçek neden bir vakada 1. sırada, bir diğerinde 2. sırada bulunduysa, bu vakalar için reciprocal rank değerleri 1 ve 1/2 olur; bunların ortalaması MRR'yi verir. MRR değeri 1.0 olması, her arıza için doğrunun ilk sırada bulunduğunu (mükemmel sıralama) gösterir.

Bu metrikler sayesinde model, yalnızca en tepede doğru nedeni bulup bulmadığıyla kalmayıp genel sıralama başarısı ve ilk birkaç tahmindeki isabet oranı açısından kapsamlı değerlendirilmiştir.

Karşılaştırılan Yöntemler: Yazarlar, MULAN'ın performansını anlamak için altı farklı RCA ve nedensel keşif yöntemiyle karşılaştırma yapmıştır [60] [61]. Bu yöntemler ve kısa açıklamaları şöyledir:

- **PC Algoritması (Burr, 2003):** İstatistiksel bağımsızlık testleri ile nedensel grafın iskeletini ortaya çıkaran klasik bir **kısıt-tabanlı** nedensel keşif algoritmasıdır [62]. Sadece metrik verilerle çalışır, çok modlu desteği yoktur.
- **DynoTYEARS (Pamfil et al., 2020): Dynamic NOTEARS** olarak da bilinen bu yöntem, vektör otoregresyon (VAR) modellerini kullanarak **dinamik Bayesyen ağ** yapıları öğrenir [63]. Zaman serilerindeki nedensel ilişkileri DAG (Directed Acyclic Graph) formunda bulmaya çalışır.
- **C-LSTM (Tank et al., 2022):** Zaman serilerinde **non-lineer Granger nedenselliğini** yakalamak için LSTM (Long Short-Term Memory) ağlarını kullanan bir modeldir [64]. Metrikler arası karmaşık ilişkiyi öğrenerek nedensel bağlantılar kurar.
- **GOLEM (Ng et al., 2020):** NOTEARS yönteminin bir türevi olup, yönlü çevrimsiz graf (DAG) kısıtını yumuşatarak sürekli bir skor fonksiyonu üzerinden optimizasyon yapan bir yaklaşımdır [65]. Böylece DAG uzayında daha verimli arama yaparak nedensel yapıyı öğrenir.
- **REASON (Wang et al., 2023):** Mikroservis sistemleri için önerilmiş, **hiyerarşik grafik sinir ağları** tabanlı bir RCA yöntemidir [66]. Hem servis içi (intra-level) hem servisler arası (inter-level) etkileşimleri yakalayıp iki katmanlı bir nedensel ağ kurar. Tek modlu metrik veriyle çalışır.
- **Nezha (Yu et al., 2023):** RCA problemini çok modlu olarak ele alan mevcut nadir yaklaşımlardan biridir [61]. Metrik ve log verilerindeki **anormal örüntüleri tespit etmeye** odaklanır; örneğin her modda sıra dışı davranışları saptayarak bunların kesişimine göre kök nedeni bulmaya çalışır. Ancak iki modun verisini birlikte işlemesine rağmen, modlar arası temsil öğrenimi yapmaz.

Karşılaştırma adil olması için, tek modlu tasarlanmış yöntemler (PC, DynoTears, C-LSTM, GOLEM, REASON) hem **metrik-verisi-ile** hem **log-verisi-ile** ayrı ayrı eğitilip test edilmiştir [67]. Ayrıca, bu yöntemlerin çok modlu senaryoda nasıl performans vereceğini görmek için, yazarlar log verilerini kendi yöntemleriyle zaman serisine çevirip (3.1'deki dil modeli aracılığıyla) metrik veri setine eklemişler ve her bir yöntemin böylece "metrik+log" verisiyle de sonuçlarını raporlamışlardır [68]. Bu sayede mulan dışındaki yöntemlerin çok modlu veriden ne kadar faydalanabildiği de değerlendirilmiştir.

Elde Edilen Sonuçların Yorumlanması ve Güvenilirliği

Genel Performans: Deneysel sonuçlar, çok modlu yaklaşımın RCA başarısını belirgin biçimde artırdığını ortaya koymaktadır. Tek modlu senaryolarda (sadece metrik veya sadece log verisiyle) elde edilen sonuçlar ile çok modlu senaryolar kıyaslandığında, hemen hemen tüm karşılaştırma yöntemlerinin bile çok modlu veriden fayda gördüğü raporlanmıştır [69]. Örneğin, Product Review veri setinde bazı yöntemler tek bir mod kullanıldığında hiçbir doğru tahmin yapamazken, log verisi eklendiğinde veya birden fazla mod beraber kullanıldığında başarıları artmıştır [67] [70]. Bu, sistem arızalarının izlerini farklı kaynaklarda bıraktığı ve tek kaynağa bakmanın yetersiz kaldığı tezini doğrular niteliktedir.

Önerilen **MULAN** modeli ise **her üç veri setinde de tüm rakiplerini tutarlı bir şekilde geride bırakmıştır** [70]. Ayrıntılı tablo sonuçlarına göre (Tablo 2, 3 ve 4), MULAN özellikle üst sıralarda doğru nedeni bulma konusunda (MRR ve Precision@K) üstün performans sergilemektedir. Örneğin, **Product Review** veri setinde MULAN, tüm arıza durumlarında doğru servisi *ilk sırada* bularak $MRR = 1.0$ gibi mükemmel bir değere ulaşmıştır. İkinci en iyi yöntem olan REASON ise $MRR \approx 0.875$ değerinde kalmıştır; aradaki fark yaklaşık **%12.5**'lik bir iyileşmeye denk gelmektedir [71]. Yine bu veri setinde, Precision@1 değeri (her arızada doğru nedeni ilk tahminde bulma oranı) MULAN için %100 iken en iyi diğer yöntemde %75 civarındadır [71]. **Online Boutique** veri setinde de benzer şekilde MULAN $MRR = 0.900$ değeriyle, en yakın rakibi olan Nezha'yı ($MRR \approx 0.767$) belirgin bir farkla aşmıştır [72]. Yazarlar bunu yaklaşık **%13.2'lik** bir iyileşme olarak not etmişlerdir. Precision@3 ve MAP@5 gibi diğer metriklerde de MULAN, Nezha'ya göre %8–%13 arası iyileşme sağlamıştır [71]. **Train Ticket** veri seti,

genel olarak en zorlayıcı olanıdır; tüm yöntemlerin başarı değerleri diğerlerine kıyasla daha düşüktür. Bu sette dahi MULAN en yüksek MRR değerine (0.381) ulaşmış, onu 0.31–0.32 bandındaki diğer yöntemler izlemiştir [73]. Özellikle C-LSTM ve REASON yöntemleri ~0.31 MRR elde ederken, MULAN yaklaşık **%20 daha yüksek** bir skorla birinci olmuştur. Bu tutarlı üstünlük, önerilen yaklaşımın çok modlu veriden etkin biçimde faydalanarak genel başarımı artırdığını göstermektedir [70].

KPI-Duyarlı Mekanizmanın Etkisi: Makalede sunulan **vaka çalışması (case study)** sonuçları, modelin düşük kaliteli modlara karşı dayanıklılığını somut olarak ortaya koymaktadır. Product Review sistemi üzerinde yapılan bu özel deneyde, yapay olarak bir metrik verisi "düşük kaliteli" (ör. çok gürültülü veya bilgi içermeyen) hale getirilmiş ve modellerin performansı incelenmiştir. Sonuçlar göstermiştir ki, tek moda dayalı yöntemlerin performansı böyle bir durumda belirgin şekilde düşmektedir; hatta bazı yöntemler düşük kaliteli metrik verisiyle neredeyse rastgele tahmin düzeyine gerilemiştir [74]. Ancak **MULAN**, log verisinin sağladığı ek bilgiyi ve KPI-farkındalıklı dikkat mekanizmasını kullanarak **performansını büyük ölçüde koruyabilmiştir** [75]. Bu senaryoda MRR değeri, yüksek kaliteli veri senaryosuna yakın kalmış ve rakip yöntemlerin aksine ciddi bir bozulma göstermemiştir [76]. İncelemeler, MULAN'ın dikkat modülünün düşük kaliteli mod için ağırlığı düşürdüğünü, yüksek kaliteli moda ise ağırlığı artırdığını ortaya koymuştur. Nitekim Şekil 3(c)'de, yüksek kaliteli metrik için hesaplanan ağırlık w_1 log ile birleştiğinde (yüksek kaliteli mod durumunda) belirgin şekilde yüksekken; düşük kaliteli metrik için ağırlık w_3 çok düşüktür ve model log verisine (w_4) nispeten daha fazla önem vermiştir [77] [74]. Bu adaptif ağırlıklandırma sayesinde, model gürültülü verinin tuzağına düşmeden doğru sinyallere odaklanabilmiştir. Bu bulgu, önerilen dikkat mekanizmasının ve genel olarak çok modlu yaklaşımın pratik sistemlerdeki güvenilirliğine işaret etmektedir.

Ablasyon Analizi: Yazarlar, önerilen mimarinin her bir bileşeninin katkısını değerlendirmek amacıyla bir **ablasyon çalışması** da yapmışlardır. MULAN modelinden belirli bileşenler tek tek çıkarılarak performans düşüşü ölçülmüştür [78]. Örneğin, VAR tabanlı decoder'lar devre dışı bırakıldığında (sadece anlık nedensel ilişkilere bakan model, **MULAN-V**), Product Review setinde MRR 1.0'dan 0.875'e gerilemiştir [79] [78]. Benzer şekilde, modlar arası orthogonallik kısıtı kaldırıldığında (**MULAN-O**), moddan bağımsız bilgi öğrenimi kapatıldığında (**MULAN-N**) veya kenar tahmin loss'u çıkarıldığında (**MULAN-E**), tüm veri setlerinde azalmalar görülmüştür [80] [81]. Özellikle **kenar çıkarma** durumu (MULAN-E), Online Boutique setinde MRR'yi 0.9'dan 0.667'ye düşürerek ~%23'lük ciddi bir performans kaybına yol açmıştır [82]. **Moddan bağımsız bilgi (node loss) olmadan** model (MULAN-N), Product Review'de ~%18 daha kötü sonuç vermiştir [82]. Bu deneyler, her bir bileşenin (VAR zaman bağımlılık modeli, invariant/specific ayrımı, kenar tahmin düzenleyicisi vb.) bütünsel başarıya önemli katkı yaptığını kanıtlamaktadır. Modelin mimarisindeki seçimlerin her biri ayrı ayrı doğrulanmış olup, herhangi birinin eksikliğinin performansı anlamlı derecede gerilettiği gösterilmiştir [83]. Dolayısıyla, sonuçların güvenirliği açısından bakıldığında, yazarlar sadece yüksek skorlar almakla kalmamış, aynı zamanda neden bu skorların elde edildiğini de açıklayacak deneyler sunmuşlardır.

Hiperparametre Duyarlılığı: Makalede, modelin bazı kritik hiperparametrelerine (ör. VAR modeli gecikme adımı p, loss fonksiyonundaki λ ağırlıkları, sparsity katsayısı vb.) karşı duyarlılığı da incelenmiştir [84] [85]. Product Review verisinde yapılan taramalarda, VAR gecikme süresinin daha büyük seçilmesinin performansı artırdığı görülmüştür (örneğin $p=4$ iken MRR en yüksek değerine ulaşmıştır) [86]. Bu durum, sistemdeki servisler arası etkilerin birkaç zaman adımı ötelenebileceğini ve bunları yakalamak için VAR penceresinin yeterince geniş tutulmasının önemli olduğunu gösterir. Kontrastif öğrenme için kullanılan λ ağırlığının 1 civarında optimum olduğu, daha yüksek değerlerde modelin fazla kısıtlanıp performans kaybettiği raporlanmıştır [87]. Benzer şekilde orthogonallik ve edge loss ağırlıklarının da belirli optimum değerleri vardır (her ikisi için de 1.0 uygun bulunmuştur) [88]. Aşırı düşük sparsity cezasının ($\lambda_{sparsity}$) grafikte gereğinden fazla bağlantıya izin verip performansı düşürdüğü, ancak çok yüksek cezanın da fayda sağlamadığı belirtilmiştir (orta düzey bir ceza en iyisidir, ör. 0.001) [89]. Tüm bu analizler, sonuçların rastlantısal olmadığına, modelin uygun

ayarlandığında tutarlı şekilde yüksek performans verdiğine işaret etmektedir. Yazarlar ayrıca bu hiperparametre incelemelerinde optimal değerleri belirleyip, ana deneylerinde bu değerleri kullanmışlardır (Şekil 4'te kırmızı kesik çizgi ile işaretlenmiş ayarlar) [90].

Sonuçların Doğruluğu ve Güvenilirliği: Makalenin bulguları, gerçek sistemlerden toplanmış verilerle doğrulandığı için pratik açıdan değerlidir. Her ne kadar kullanılan arıza örnekleri sınırlı sayıda olsa da (toplam 14 arıza vakası), her vaka gerçek bir servis kesintisi senaryosunu temsil etmektedir. Elde edilen yüksek başarı, özellikle karmaşık mikroservis sistemlerinde geleneksel yöntemlerin kaçırabileceği kök nedenlerin, çok modlu yaklaşım sayesinde yakalanabildiğini göstermektedir. Sonuçların güvenilirliği, farklı metriklerle benzer üstünlükler elde edilmesiyle pekişmektedir: Yani sadece bir tek ölçütte değil, Precision@K, MAP ve MRR'in tamamında MULAN genelde en iyidir. Ayrıca, yöntemin bileşenlerine dair yapılan derinlemesine analizler, modelin **neden iyi çalıştığını** anlaşılır kılmaktadır – bu da yönteme olan güveni artıran bir faktördür. Kısacası çalışma, önerilen yöntemin hem **etkili** (yüksek başarımlı) hem de **anlaşılır ve doğrulanabilir** olduğunu ortaya koymaktadır.

Potansiyel Uygulamalar ve Yapay Zeka Alanındaki Etkileri

Bu çalışmanın önerdiği çok modlu RCA yaklaşımının başlıca uygulama alanı, **BT sistemleri operasyonları (AIOps)** ve **hizmet izleme** (monitoring) alanlarıdır. Mikroservis tabanlı mimariler günümüzde e-ticaret, çevrimiçi hizmetler, bulut uygulamaları, IoT platformları gibi birçok alanda kullanılmaktadır. Bu sistemlerde oluşan hataların kök nedenini hızla bulmak, kesinti sürelerini azaltmak ve kullanıcı memnuniyetini korumak için kritik önemdedir [1]. MULAN çerçevesi, bu tip sistemlerde log kayıtları ile metrik ölçümleri bir arada değerlendirerek **otomatik arıza teşhisi** yapabilecek akıllı bir asistan gibi düşünülebilir. Örneğin, büyük bir e-ticaret sitesinde aniden müşteri işlemleri yavaşladığında (KPI: yüksek gecikme), sistem yöneticisi MULAN gibi bir modeli çalıştırarak aynı anda sunucu metriklerini ve uygulama loglarını analiz ettirip sorunun kaynağını (örn. spesifik bir mikroserviste bellek sızıntısı veya veritabanı bağlantı hatası) hızlıca tespit edebilir. Bu, geleneksel manuel arıza arama süreçlerine göre çok daha hızlı bir çözüm sunar ve büyük ölçekli sistemlerde insanın gözden kaçırabileceği detayları yakalayabilir.

Bunun yanı sıra, önerilen yaklaşım **genel anlamda çok modlu yapay zeka** araştırmaları için de önemli bir örnek teşkil etmektedir. Doğal dil işlemedeki gelişmiş dil modellerini (LLM) sistem loglarına uygulayarak, yapılandırılmamış metin verisini sayısal nedensel analiz ile buluşturmuştur. Bu sayede **NLP ile zaman serisi analizi** sıra dışı bir kombinasyon içinde kullanılmıştır. Makale, literatürde çoğunlukla tek modlu kalan RCA problemine ilk defa gerçekten entegre bir çok modlu çözüm önermesi açısından yenilikçidir [91] [92]. Bu yaklaşım, yapay zeka alanında **farklı veri modlarının birlikte yorumlanması** temasına katkı yapmaktadır. Örneğin, gelecekte bu fikir sağlık alanında (hastane cihaz verileri + doktor notları ile teşhis nedenselliği), endüstriyel IoT'de (sensör ölçümleri + bakım logları ile arıza analizi) gibi farklı alanlara uyarlanabilir. Yöntemin genel formülasyonu, bir modun zayıf kaldığı durumda diğer modun desteğiyle doğru sonuca ulaşma fikrine dayanır; bu fikir birçok alanda geçerli olabilir.

MULAN'ın bir diğer önemli katkısı, **nedensel keşif** yöntemlerine getirdiği bakıştır. Geleneksel yöntemler, verilmiş veri setinden bir nedensel yönlü grafik çıkarmaya odaklanırken, MULAN hem veriyi temsil uzayında dönüştürmekte (örn. logları dil modeli ile vektörleştirmek), hem de grafik öğrenimini bu temsil üzerinde entegre bir optimizasyon problemi olarak ele almaktadır. Ayrıca, **random walk ile kök neden skorlama** yeniliği, nedensel grafik elde edildikten sonra ondan faydalanmayı kolaylaştıran bir araçtır. Bu sayede model sadece "grafiği bulmakla" kalmıyor, o grafiği kullanarak pratik bir çıktı (kök neden listesi) üretiyor. Bu uçtan uca yaklaşım, yapay zeka sistemlerinin tasarımında **sonuç-odaklı** (goal-driven) tasarımın güzel bir örneğidir.

Çalışmanın yapay zeka araştırma alanına bir diğer etkisi de **karşıt-öğrenme ve çok modlu temsil** konularını nedensel yapıyla birleştirmesidir. Yazarlar, modality-invariant ve modality-specific kavramlarını tanımlayarak, çok modlu veride ortak olan ve olmayan yönleri ayrıştırmanın önemini vurgulamışlardır [29] [93]. Bu fikir, ileride farklı modaliteler arasında bilgi aktarımı yapmak isteyen diğer modeller için de uygulanabilir. Örneğin, görüntü ve metin modalitelerini kullanan sistemlerde de benzer invariant/spesifik ayrımları yapmak performansı artırabilir. Bu çalışma, çok modlu öğrenmede salt erken veya geç birleştirme yerine, **ortak temsil öğrenimi** gibi daha ince ayarlara girmenin getirilerini göstermiştir.

Son olarak, makalenin bir bölümü geleceğe dönük potansiyel geliştirmelere değinmektedir. Özellikle **gerçek zamanlı (online) kök neden analizi** vurgulanmıştır [94]. Şu anki yöntem, toplu veriler üzerinde çalışmaktadır (yani arıza sonrası verilerle). Gelecekte, streaming veriye uyarlanarak sürekli akan log ve metrik verisi üzerinde anlık anomali tespiti ve kök neden bulma gerçekleştirilebilir. Bu, yapay zeka uygulamalarının **kesintisiz izleme** ve **anlık karar destek** sistemleri yönünde ilerlediğini gösterir. MULAN çerçevesi, uygun optimizasyonlarla gerçek zamanlıya yaklaştırılırsa büyük veri merkezlerinde otomatik uyarı ve teşhis sistemleri kurulabilecektir. Bu da insan müdahelesini en aza indirip, otonom yönetilen sistemlere doğru bir adım anlamına gelir.

Özetle, "Multi-modal Causal Structure Learning and Root Cause Analysis" makalesi, yapay zeka alanında **çok modlu veri analizi, büyük dil modellerinin sistem yönetimine uygulanması, nedensel grafik öğrenimi ve grafik tabanlı arama** kavramlarını bir araya getiren öncü bir çalışmadır. Mikroservis dünyasında pratik bir probleme çözüm getirirken, kullanılan tekniklerle geniş bir etki potansiyeline sahiptir. Bu yaklaşım, gelecekte daha güvenilir, akıllı ve hızlı teşhis koyabilen otonom sistemlerin geliştirilmesinde bir kilometre taşı olma niteliği taşımaktadır. Makalenin The Web Conference 2024'de kabul edilmiş olması da, bu katkının akademik camiada takdir gördüğünü ve benzer çalışmalar için bir teşvik oluşturduğunu göstermektedir [95]. Çok modlu RCA konusunun henüz keşfedilmemiş pek çok yönü olduğu düşünüldüğünde, bu çalışma hem bir temel sunmakta hem de gelecek araştırmalar için yeni sorular ortaya koymaktadır.

Kaynaklar: Makalede belirtilen referanslar ve deney sonuçları doğrudan alıntılanarak yukarıda sunulmuştur. Bunlar arasında özellikle arXiv'de yayınlanan orijinal makalenin sunduğu tablolar ve şekiller kaynak olarak kullanılmıştır. Aşağıdaki bağlantılar, ilgili içeriklerin alındığı yerlere işaret etmektedir:

[2] [3] Makalenin özeti ve giriş bölümünden alınan, çok modlu RCA ihtiyacını ve önerilen yaklaşımı açıklayan satırlar.
[9] Şekil 1'in başlığından, modelin dört ana modülünü tanımlayan ifade.
[14] [17] Log verilerinin işlenmesi ve anomali skoru etiketlemesi adımlarının anlatıldığı kısımlar.
[29] [30] Karşıt öğrenme (contrastive) ve ortogonallik kısıtlarının tanımlandığı formüller ve açıklamalar.
[36] [39] KPI-farkındalıklı attention mekanizması ve mod önem ağırlıklarının hesaplanmasını açıklayan bölümler.
[47] [48] Random walk with restart yönteminin formülasyonunu ve kök neden sıralamasını ifade eden denklemler.
[51] Kullanılan üç veri setinin tanıtıldığı satırlar.
[60] [61] Karşılaştırılan yöntemlerin (PC, DynoTears, C-LSTM, GOLEM, REASON, Nezha) listelendiği ve açıklandığı kısım.
[70] Genel performans değerlendirmesi: çok modluluğun faydası ve MULAN'ın üstünlüğünün vurgulandığı yerler.
[75] Düşük kaliteli mod durumunda MULAN'ın performansının düşmediğini belirten vaka çalışması bulguları.
[82] Ablasyon çalışmasından, model bileşenlerinin çıkarılmasının performansı düşürdüğünü gösteren

sonuçlar.
91 92 İlgili çalışmalardan farklar ve makalenin çok modlu RCA alanındaki yeniliğini vurgulayan cümleler.
96 Sonuç bölümünden, çalışmanın özetini ve geleceğe dönük planlarını aktaran kısım.

---

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29
30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58
59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87
88 89 90 91 92 93 94 96 [2402.02357] Multi-modal Causal Structure Learning and Root Cause Analysis
https://ar5iv.labs.arxiv.org/html/2402.02357

95 [2402.02357] Multi-modal Causal Structure Learning and Root Cause Analysis
https://arxiv.org/abs/2402.02357