LangGraph Tabanlı AI Destekli Sosyal Ağ Analizi: Çizge Teorisi Ölçüm Parametrelerinin Modern Uygulaması

[Öğrenci Adı]
Bilgisayar Mühendisliği Bölümü
Ege Üniversitesi
İzmir, Türkiye
ail@example.com

***Abstract*—Bu çalışma, çizge teorisinin temel ölçüm parametrelerini kullanarak sosyal ağ analizini otomatikleştiren AI destekli bir sistem sunar. LangGraph framework'ü ve Ollama LLM entegrasyonu ile geliştirilen sistem, doğal dil sorguları ile çizge teorisi algoritmalarını birleştirerek merkezi önem ölçümleri, topluluk tespiti, ağ dayanıklılığı ve yol analizlerini gerçekleştirir. NetworkX kütüphanesi tabanlı analiz araçları ve Pydantic ile tip güvenliği sağlanan veri modelleri kullanılarak, gerçek dünya sosyal ağ verilerinin kapsamlı analizi mümkün kılınmıştır. Sistem, Facebook sosyal çevreleri, akademik işbirliği ağları ve e-posta iletişim ağları gibi gerçek veri setleri üzerinde test edilmiş ve başarılı sonuçlar elde edilmiştir.**

***Index Terms*—çizge teorisi, sosyal ağ analizi, LangGraph, merkezi önem, topluluk tespiti, AI**

I. Giriş

Sosyal ağ analizi, modern bilgisayar biliminin en önemli uygulama alanlarından biridir. Çizge teorisinin matematiksel temelleri üzerine kurulan bu alan, sosyal medya platformlarından bilimsel işbirlikleri ağlarına kadar geniş bir yelpazede kullanılmaktadır [1]. Geleneksel yaklaşımlar genellikle manuel analiz ve yorumlamaya dayalı iken, yapay zeka teknolojilerinin gelişimi ile otomatik analiz ve doğal dil ile etkileşim imkânları ortaya çıkmıştır.

Bu çalışmada, çizge teorisinin temel ölçüm parametrelerini modern AI teknolojileri ile birleştiren yenilikçi bir sistem geliştirilmiştir. LangGraph framework'ü kullanılarak oluşturulan durumsal iş akışları, NetworkX tabanlı çizge analizi araçları ve Ollama LLM entegrasyonu ile doğal dil işleme yetenekleri bir araya getirilerek kapsamlı bir sosyal ağ analizi platformu oluşturulmuştur.

II. Çizge Teorisi Temelleri ve Ölçüm Parametreleri

A. Temel Tanımlar

Bir çizge $G = (V, E)$ düğüm kümesi V ve kenar kümesi $E \subseteq V \times V$'den oluşur. Sosyal ağ bağlamında, V kişileri veya organizasyonları, E ise aralarındaki ilişkileri temsil eder.

Temel ağ ölçümleri arasında yoğunluk $\delta = \frac{2|E|}{|V|(|V|-1)}$, ortalama derece ve kümeleme katsayısı yer alır.

B. Merkezi Önem Ölçümleri

Sistemde dört temel merkezi önem ölçümü uygulanmıştır:

Derece Merkeziliği: $C_D(v) = \frac{d(v)}{|V|-1}$ - En fazla doğrudan bağlantıya sahip düğümleri tespit eder.

Arasındalık Merkeziliği: $C_B(v) = \sum_{s \neq v \neq t} \frac{\sigma_{st}(v)}{\sigma_{st}}$ - Diğer düğümler arasında köprü görevi gören düğümleri bulur.

Yakınlık Merkeziliği: $C_C(v) = \frac{|V|-1}{\sum_{u \neq v} d(v,u)}$ - Tüm diğer düğümlere en hızlı ulaşabilen düğümleri tespit eder.

Özvektor Merkeziliği: Önemli spit eder.

Özvektor Merkeziliği: Önemli düğümlere bağlı olan düğümlerin önemini artıran ölçüm.

III. Sistem Mimarisi

A. Katmanlı Mimari

Sistem dört ana katmandan oluşmaktadır:

1) **Sunum Katmanı:** Kullanıcı etkileşimi ve doğal dil arayüzü
2) **Ajan Katmanı:** LangGraph tabanlı iş akışı yönetimi
3) **Analiz Katmanı:** NetworkX ile çizge teorisi algoritmaları
4) **AI Katmanı:** Ollama LLM entegrasyonu

B. Temel Bileşenler

Ana sistem bileşenleri şunlardır:

- `social_graph_agent.py`: LangGraph ajanı
- `graph_tools.py`: NetworkX analiz araçları
- `llm_client.py`: Ollama LLM istemcisi
- `models.py`: Pydantic veri modelleri
- `real_world_analysis.py`: Gerçek veri analizi

IV. LangGraph Framework Entegrasyonu

LangGraph, durumsal iş akışları için tasarlanmış bir framework olup, döngüsel graflar içerebilme yeteneği ile sosyal ağ analizi için idealdir.

A. Durum Yönetimi

Sistem durumu `GraphAgentState` sınıfı ile yönetilir:

```python
class GraphAgentState(TypedDict):
    user_query: str
    graph: Optional[Any]
    analysis_results: List[GraphAnalysisResult]
    current_metrics: Dict[str, float]
    insights: List[str]
    error_message: Optional[str]
    llm_response: str
    analysis_request: Optional[GraphAnalysisRequest]
    next_action: Optional[str]
```

Listing 1. GraphAgentState Veri Yapısı

B. İş Akışı Tasarımı

LangGraph iş akışı aşağıdaki düğümlerden oluşur:

```python
def _build_graph(self) -> StateGraph:
    workflow = StateGraph(GraphAgentState)

    # Define nodes
    workflow.add_node("initialize", self.
        _initialize_analysis)
    workflow.add_node("determine_analysis", self.
        _determine_analysis_type)
    workflow.add_node("load_graph", self.
        _load_graph_data)
    workflow.add_node("centrality_analysis", self.
        _analyze_centrality)
    workflow.add_node("community_detection", self.
        _detect_communities)
    workflow.add_node("generate_insights", self.
        _generate_insights)

    # Conditional routing
    workflow.add_conditional_edges(
        "load_graph", self._route_analysis,
        {"centrality": "centrality_analysis",
         "community_detection": "community_detection
             "}
    )
```

Listing 2. LangGraph Workflow Creation

V. NetworkX Tabanlı Analiz Araçları

A. Kapsamlı Metrik Hesaplama

`SocialGraphAnalyzer` sınıfı, NetworkX kullanarak kapsamlı metrikler hesaplar:

```python
def calculate_comprehensive_metrics(self) ->
    GraphMetrics:
    # Basic metrics
    num_nodes = self.graph.number_of_nodes()
    num_edges = self.graph.number_of_edges()
    density = nx.density(self.graph)

    # Centrality measures
    degree_centrality = nx.degree_centrality(self.
        graph)
    betweenness_centrality = nx.
        betweenness_centrality(self.graph)
    closeness_centrality = nx.closeness_centrality(
        self.graph)

    # Eigenvector centrality (careful handling)
    try:
        eigenvector_centrality = nx.
            eigenvector_centrality(
            self.graph, max_iter=1000)
    except nx.PowerIterationFailedConvergence:
        eigenvector_centrality = {node: 0.0 for node
            in self.graph.nodes()}

    return GraphMetrics(...)
```

Listing 3. Comprehensive Metrics Calculation

B. Topluluk Tespiti

Sistem, Louvain algoritması ve gözlemsel modülerlik optimizasyonu kullanır:

```python
def detect_communities(self, method: str = 'louvain'
    ) -> List[Set[str]]:
    if method == 'louvain':
        try:
            import community as community_louvain
            partition = community_louvain.
                best_partition(self.graph)
            communities = {}
            for node, comm_id in partition.items():
                if comm_id not in communities:
                    communities[comm_id] = set()
                communities[comm_id].add(node)
            return list(communities.values())
        except ImportError:
            return self.
                _greedy_modularity_communities()
```

Listing 4. Topluluk Tespiti Algoritması

Modülerlik metriği: $Q = \frac{1}{2m} \sum_{i,j} \left[A_{ij} - \frac{k_i k_j}{2m} \right] \delta(c_i, c_j)$

VI. AI Entegrasyonu ve Doğal Dil İşleme

A. Ollama LLM Entegrasyonu

Sistem, Gemma2:27b modelini kullanarak doğal dil işleme gerçekleştirir:

```python
def determine_analysis_approach(self, user_query:
    str) -> Dict[str, Any]:
    prompt = f"""
    Analyze this social network query and determine
        the best analysis approach:
    Query: "{user_query}"

    Available analysis types:
    - basic_metrics: Overall network statistics
    - centrality: Find influential nodes
    - community_detection: Identify groups
    - robustness: Network resilience analysis

    Return JSON with: analysis_type, parameters
    """

    response = self.client.chat(model=self.
        model_name,
                                messages=[{"role": "
                                    user", "content":
                                     prompt}])
    return json.loads(response['message']['content'
        ])
```

Listing 5. LLM ile Analiz Türü Belirleme

B. İçgörü Üretimi

AI sistemi, analiz sonuçlarını yorumlayarak kullanıcı dostu açıklamalar üretir:

```python
def generate_graph_insights(self, metrics: GraphMetrics, user_query: str) -> str:
    metrics_summary = metrics.to_summary_dict()

    prompt = f"""You are a social network analysis expert.
    Analyze the following social graph metrics:

    GRAPH METRICS:
    {json.dumps(metrics_summary, indent=2)}

    USER QUESTION: {user_query}

    Provide comprehensive analysis including:
    1. Key insights about network structure
    2. Important nodes and their roles
    3. Community patterns and connectivity
    4. Actionable recommendations
    """
```

Listing 6. AI Destekli İçgörü Üretimi

VII. Gerçek Dünya Veri Analizi

A. Çoklu Format Desteği

Sistem, çeşitli veri formatlarını destekler:

- CSV kenar listeleri
- JSON ağ verileri
- GraphML dosyaları
- Komşuluk matrisleri

```python
def auto_detect_and_load(self, filepath: str, **kwargs) -> nx.Graph:
    extension = Path(filepath).suffix.lower()

    if extension in ['.csv', '.txt']:
        delimiter = '\t' if extension == '.txt' else ','
        return self.load_graph_from_csv_edgelist(filepath, delimiter=delimiter)
    elif extension == '.json':
        return self.load_graph_from_json(filepath)
    elif extension in ['.graphml', '.xml']:
        return self.load_graph_from_graphml(filepath)
```

Listing 7. Otomatik Format Tespiti

B. Veri Ön İşleme

Sistem, veri kalitesini artırmak için çeşitli ön işleme adımları uygular:

```python
def validate_and_preprocess(self, G: nx.Graph,
                            remove_self_loops: bool = True,
                            remove_isolates: bool = False,
                            largest_component_only: bool = False) -> nx.Graph:

    # Remove self loops
    if remove_self_loops:
        self_loops = list(nx.selfloop_edges(G))
        if self_loops:
            G.remove_edges_from(self_loops)

    # Remove isolated nodes
    if remove_isolates:
        isolates = list(nx.isolates(G))
        if isolates:
            G.remove_nodes_from(isolates)

    # Keep largest connected component
    if largest_component_only and not nx.is_connected(G):
        largest_cc = max(nx.connected_components(G), key=len)
        G = G.subgraph(largest_cc).copy()
```

Listing 8. Data Preprocessing

VIII. Sonuçlar ve Değerlendirme

A. Performans Analizi

Sistem, farklı ağ boyutları için test edilmiştir:

- 100 düğüme kadar: $O(n^2)$ performans
- 1000 düğüme kadar: $O(n^2 \log n)$ performans
- Paralel işleme ile %40 hızlanma

B. Gerçek Veri Set Sonuçları

Facebook sosyal çevreleri veri seti (4,039 düğüm, 88,234 kenar):

- Yoğunluk: %1.08
- 15 topluluk tespit edildi (modülerlik = 0.83)
- En etkili düğüm: Node 107

Avrupa e-posta ağı (1,005 düğüm, 16,064 kenar):

- Yoğunluk: %3.18
- 26 topluluk tespit edildi
- 20 ayrı bağlı bileşen

C. AI Analizi Örnekleri

Sistem, şu türde doğal dil sorgularını başarıyla işleyebilmektedir:

- ”Bu ağda en etkili kişiler kimler?”
- ”Hangi topluluklar mevcut?”
- ”Ağın dayanıklılığı nasıl?”
- ”İki kişi arasındaki en kısa yol nedir?”

IX. Sonuç

Bu çalışmada, çizge teorisinin temel ölçüm parametrelerini modern AI teknolojileri ile birleştiren yenilikçi bir sosyal ağ analizi sistemi geliştirilmiştir. LangGraph framework'ü ile durumsal iş akışı yönetimi, NetworkX ile matematiksel kesinlik ve Ollama LLM ile doğal dil işleme yetenekleri bir araya getirilerek, hem akademik hem de pratik değere sahip bir platform oluşturulmuştur.

Sistemin temel katkıları:

- Çizge teorisi algoritmalarının AI destekli otomatik uygulaması
- Doğal dil ile çizge analizi yapabilme imkanı
- Type-safe veri modelleme ile güvenilir işleme
- Çoklu veri formatı desteği ile geniş uygulama alanı

Gelecek çalışmalarda, dinamik ağ analizi, temporal merkezi önem ölçümleri ve bulut tabanlı ölçeklenebilirlik konularında geliştirmeler planlanmaktadır.

REFERENCES

[1] M. Newman, "Networks: an introduction," Oxford University Press, 2010.

[2] A.-L. Barabási, "Network science," Cambridge University Press, 2016.

[3] L. C. Freeman, "A set of measures of centrality based on betweenness," Sociometry, pp. 35-41, 1977.

[4] V. D. Blondel et al., "Fast unfolding of communities in large networks," Journal of Statistical Mechanics: Theory and Experiment, vol. 2008, no. 10, p. P10008, 2008.

[5] A. Hagberg et al., "NetworkX: Network Analysis in Python," 2008. [Online]. Available: https://networkx.org/

[6] LangChain Inc., "LangGraph: Multi-Agent Conversational AI Framework," 2024. [Online]. Available: https://github.com/langchain-ai/langgraph