**KOCAELİ ÜNİVERSİTESİ**
**FEN BİLİMLERİ ENSTİTÜSÜ**

**BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI**

**YÜKSEK LİSANS TEZİ**

# HEAD OUTPUT CONNECTED İLİŞKİLER İÇİN ÇİZGE TABANLI BİR KAVRAM KEŞİF SİSTEMİ

**NURAN PEKER**

**KOCAELİ 2017**

**KOCAELİ ÜNİVERSİTESİ**

**FEN BİLİMLERİ ENSTİTÜSÜ**

**BİLGİSAYAR MÜHENDİSLİĞİ**

**ANABİLİM DALI**

**YÜKSEK LİSANS TEZİ**

# HEAD OUTPUT CONNECTED İLİŞKİLER İÇİN ÇİZGE TABANLI BİR KAVRAM KEŞİF SİSTEMİ

**NURAN PEKER**

| | | |
|---|---|---|
| **Yrd. Doç. Dr. Alev MUTLU** | | |
| **Danışman,** | **Kocaeli Üniv.** | ..... |
| **Prof. Dr. Mübariz EMİNLİ** | | |
| **Jüri Üyesi,** | **Haliç Üniv.** | ..... |
| **Doç. Dr. Sevinç İLHAN OMURCA** | | |
| **Jüri Üyesi,** | **Kocaeli Üniv.** | ..... |



**Tezin Savunulduğu Tarih: 23.05.2017**

## ÖNSÖZ VE TEŞEKKÜR

Kavram keşfi, çok ilişkili veri tabanlarında hedef olarak belirlenen bir ilişkinin, veri tabanında arkaplan ilişkisi olarak tutulan diğer bazı ilişkiler cinsinden tanımlanmasıdır. Çizgelerin bu alandaki geçmişi çok eski olmamakla beraber, ilişkisel veri modelini göstermede kullanışlı olması ve bu alandaki diğer bazı yöntemlerin problemli yönlerine getirdiği çözüm önerileri, çizgelerin kullanımında artan bir eğilim göstermektedir. Head Output Connected, hedef kavrama ait en az bir çıkış değişkeninin, cümlenin gövde kısmında örneklenerek, ilişkili verilerin ifade edildiği cümle yapılarıdır. Bu tez çalışmasında, Head Output Connected ilişkiler için yol bulma tabanlı bir kavram keşif sistemi tanıtılmaktadır.

Bu tezin ortaya çıkmasında, değerli emeğiyle katkıda bulunan, tezin her aşamasında ilgi ve desteğini hissettiğim tez danışmanım Yrd. Doç. Dr. Alev Mutlu'ya sonsuz teşekkürlerimi sunarım. Eğitici rehberliği, hoşgörüsü ve sabrı için minnettarlığımı ayrıca ifade etmeyi borç bilirim.

Laboratuvarda geçirdiğim süre zarfında, odalarını benimle paylaşma nezaketini gösteren Arş. Gör. Ekin Ekinci ve Arş. Gör. Meltem Kurt Pehlivanoğlu'na ve yüksek lisans eğitimim boyunca hep zarif davanışlarına muhatap olduğum Kocaeli Üniversitesi Bilgisayar Mühendisliği Bölümü hocalarına teşekkür ederim.

Değerli zamanlarını ayırıp, tez savunmama katılmayı kabul eden Prof. Dr. Mübariz Eminli ve Doç. Dr. Sevinç İlhan Omurca'ya ayrıca teşekkürlerimi sunarım.

Son olarak, hayatım boyunca benden maddi, manevi desteğini esirgemeyen aileme ve bana duâlarıyla destek olan herkese teşekkür ederim.

Mayıs – 2017 Nuran PEKER

## İÇİNDEKİLER

ÖNSÖZ VE TEŞEKKÜR ..... i
İÇİNDEKİLER ..... ii
ŞEKİLLER DİZİNİ ..... iv
TABLOLAR DİZİNİ ..... v
SİMGELER VE KISALTMALAR DİZİNİ ..... vi
ÖZET ..... vii
ABSTRACT ..... viii
GİRİŞ ..... 1
1. GENEL BİLGİLER ..... 3
1.1. Tezin Amacı ..... 3
1.2. İlgili Çalışmalar ..... 3
1.2.1. TMP tabanlı yaklaşımlar ..... 4
1.2.1.1. Yukarıdan aşağıya sistemler (top-down) ..... 4
1.2.1.2. Aşağıdan yukarıya sistemler (bottom-up) ..... 6
1.2.1.3. Melez yaklaşımlar (hybrid) ..... 6
1.2.2. Çizge tabanlı yaklaşımlar ..... 7
1.2.2.1. Yol bulma tabanlı yaklaşımlar (path finding) ..... 7
1.2.2.2. Eşyapı tabanlı yaklaşımlar (substructure-based) ..... 8
1.2.2.3. Melez yaklaşımlar ..... 10
1.3.Tezin Organizasyonu ..... 12
2. KAVRAM KEŞİF SİSTEMLERİ ..... 13
2.1.Veri Madenciliği ..... 13
2.2. Çok İlişkili Veri Madenciliği ..... 16
2.3. Kavram Keşfi ..... 17
2.4. TMP Tabanlı Öğrenme ..... 21
2.4.1. Tahmin edici (predictive) TMP ..... 23
2.4.2. Tanımlayıcı (descriptive) TMP ..... 25
2.5. Çizge Tabanlı Öğrenme ..... 26
2.5.1. Eşyapı tabanlı öğrenme ..... 31
2.5.2. Yol bulma tabanlı öğrenme ..... 33
2.5.3. Melez öğrenme ..... 34
3. ÇİZGE TABANLI KAVRAM KEŞFİ ..... 36
3.1. Veri Gösterimleri ..... 36
3.2. Arama Teknikleri ..... 38
3.3. Kavram Tanımlayıcılarını Değerlendirme Yöntemleri ..... 41
3.4. Algoritmalar ..... 44
3.5. Özyineleme Kuralları ..... 46
4. ÖNERİLEN YÖNTEM ..... 48
4.1. Head Output Connected (HOC) Sınıfı ..... 48
4.2. Veri Seti ve Gösterimi ..... 49
4.3. Algoritma ..... 50
5. DENEYSEL BULGULAR ..... 62
5.1. Deney Ortamı ..... 62

5.2. Veri Setleri ..... 62
5.3. Performans Ölçümü ..... 64
6. SONUÇLAR VE ÖNERİLER ..... 68
KAYNAKLAR ..... 71
KİŞİSEL YAYIN VE ESERLER ..... 80
ÖZGEÇMİŞ ..... 81

## ŞEKİLLER DİZİNİ

Şekil 2.1. Bilgi keşif sürecinde veri madenciliği ..... 14
Şekil 2.2. Yerel plato problemi ..... 20
Şekil 2.3. Bir H hipotezi için tamlık ve tutarlılık durumları ..... 23
Şekil 2.4. a) Tahmin edici çıkarım b) tanımlayıcı çıkarım ..... 24
Şekil 2.5. Königsberg'in yedi köprüsü ..... 26
Şekil 2.6. Altçizge yapıları a) ana çizge b) bağlı altçizge c) kaynaklı altçizge d) ağaç yapısı e) yol yapısı ..... 28
Şekil 2.7. Sınıflandırılmış çizge madenciliği algoritmaları ..... 30
Şekil 3.1. Basit bir BFS uygulaması ..... 39
Şekil 3.2. Basit bir DFS uygulaması ..... 40
Şekil 3.3. (1) dizisi C++ kodu ..... 43
Şekil 4.1. Elti verisinin çizge ile gösterimi ..... 50
Şekil 4.2. Çizge oluşturma algoritması ..... 51
Şekil 4.3. Örnek elti çizgesi ..... 53
Şekil 4.4. Kavram tanımlayıcı çıkarım algoritması ..... 55
Şekil 4.5. HOC testi algoritması ..... 57
Şekil 4.6. Kavram tanımlayıcıları değerlendirme algoritması ..... 61

## TABLOLAR DİZİNİ

Tablo 1.1. Kavram keşif yöntemleri ..... 4
Tablo 1.2. Çizge tabanlı yaklaşımlar ..... 11
Tablo 2.1. Ata veri tabanı ..... 18
Tablo 2.2. Popüler çizge madenciliği alanları ..... 31
Tablo 4.1. Elti veri seti ..... 49
Tablo 4.2. Örnek hash tablosu ..... 55
Tablo 5.1. Fibonacci sayıları ..... 63
Tablo 5.2. Veri setleri ve deneysel ayarları ..... 64
Tablo 5.3. Family veri seti karşılaştırılması ..... 65
Tablo 5.4. Önerilen yöntemin performansı ..... 67

## SİMGELER VE KISALTMALAR DİZİNİ

### Kısaltmalar

| | | |
|---|---|---|
| BFS | : | Breadth-First Search (Enine Arama) |
| CLIP | : | Concept Learning From Inference Patterns (Sonuş Çıkarma Örüntülerinden Kavram Öğrenme |
| DFS | : | Depth-First Search (Derinine Arama) |
| DHDB | : | Deductive Hierarchical Database (Tümdengelim Hiyerarşik Veri Tabanı) |
| DLS | : | Depth-Limited Search (Derinlik Sınırlı Arama) |
| GBI | : | Graph-Based Induction (Çizge Tabanlı Çıkarım) |
| G-CDS | : | Graph-Based Concept Discovery System (Çizge Tabanlı Kavram Keşif Sistemi) |
| H-GCD | : | A Hybrid Graph-Based Method for Concept Rule Discovery (Melez Çizge Tabanlı Bir Kavram Kural Keşfi) |
| HOC | : | Head Output Connected (Baş Çıkış Bağlantılı) |
| ILP | : | Inductive Logic Programming (Tümevaran Mantık Programlama) |
| KDD | : | Knowledge Discovery from Databases (Veri Tabanlarından Bilgi Keşfi) |
| LGG | : | Least General Generalization (En Az Genel Genelleştirme) |
| MDL | : | Minimum Description Length (Minimum Tanımlama Uzunluğu) |
| MRDB | : | Multi Relational Database (Çok İlişkisel Veri Tabanı) |
| MRDM | : | Multi Relational Data Mining (Çok İlişkili Veri Madenciliği) |
| MDPF | : | Mode Directed Path Finding (Mod Yönlü Yol Blma) |
| RPBL | : | Relational Paths Based Learning (İlişkisel Yol Tabanlı Öğrenme) |
| RLGG | : | Relative Least General Generalization (Göreli En Az Genel Genelleştirme) |
| TMP | : | Tümevaran Mantık Programlama |
| VTBK | : | Veri Tabanlarında Bilgi Keşfi |

## HEAD OUTPUT CONNECTED İLİŞKİLER İÇİN ÇİZGE TABANLI BİR KAVRAM KEŞİF SİSTEMİ


## ÖZET

Bilgi çağı olarak adlandırılan günümüz dünyasında, teknolojik gelişmelere paralel olarak bilgiyi depolayan araçların ucuzlaması ve yaygınlaşması, artan miktarda bilginin depolanması ile sonuçlanmıştır. Hemen her alanda depolanan bu bilgiye hızlı ve güvenilir bir şekilde erişim, gerek bireysel gerek kurumsal bazda artan bir önem kazanmaktadır. Özellikle devlet kurumları, bankalar, hastaneler, okullar, şirketler vb. için bilgiyi saklama ve bilgiye hatasız bir şekilde erişme hayatî önem arz etmektedir. Bu tür kurumlar verileri büyük veri tabanlarında saklar. Veri tabanlarında saklanan bu bilgiye erişim söz konusu olduğunda, veri madenciliği kavramı ön plâna çıkmaktadır. Veri madenciliği, veri tabanlarından bilgi keşif sürecinin bir adımı olarak, bilgi yığınları arasından anlamlı ve kullanışlı bilginin elde edilmesi olarak tanımlanabilir.

Çok ilişkili veri madenciliği, birbiriyle ilişkili verilerin ayrı tablolarda tutulduğu veri tabanlarından, bilgiyi bulup çıkarma işlemidir. Bu alanda TMP tabanlı yaklaşımlar sıklıkla kullanılmaktadır. Çizgeler, bu tür verileri göstermeye çok elverişli yapılar olduğu için, bu alandaki kullanımları giderek artmaktadır.

Kavram keşfi, çok ilişkili veri madenciliğinin önemli uğraş alanlarından biridir. Kavram keşfinde, hedef kavram olarak adlandırılan ilişkinin, veri tabanında arkaplân bilgisi olarak tutulan diğer bazı ilişkiler cinsinden en iyi tanımının bulunması amaçlanır.

Bu tez çalışmasında, öncelikle çok ilişkili veri madenciliği temel hatlarıyla anlatılmakta ve kavram keşif probleminde yapılan TMP ve çizge tabanlı çalışmalar hakkında detaylı bilgiler verilmektedir. Daha sonra, çizge madenciliğinin yol bulma tabanlı yaklaşımı kullanılarak, Head Ooutput Connected (HOC) sınıfına ait kavram tanımlayıcıların bulunduğu bir kavram keşif sistemi tanıtılmaktadır.



**Anahtar Kelimeler:** Çizge, Destek, Güven, HOC, Kavram Keşfi.

## A GRAPH BASED CONCEPT DISCOVERY SYSTEM FOR HEAD OUTPUT CONNECTED RELATIONS


## ABSTRACT

In today's world, which is called information age, the cheaper and widespread use of tools that store information in parallel with technological developments has resulted in the storage of increasing amounts of information. This information, which is stored on almost every field, gradually gains fast and reliable access, both on an individual and institutional basis. Especially for government agencies, banks, hospitals, schools, companies, etc. it is vitally important to store and access the information without errors. Such institutions store data in large databases. In the case of this information which is stored in databases needs to accessed, the concept of data mining becomes crucial. Data mining, as a step in the knowledge discovery process from the databases, can be defined as the acquisition of meaningful and useful knowledge from the heaps of information.

Multi-relational data mining is the process of finding and extracting interrelated data from databases those are kept in separate tables. TMP based approaches are often used in this area. Since graphs are very convenient tools to show such data, their use in this area is continuously increasing.

Concept discovery is one of the most important study areas of multi-relational data mining. In concept discovery, it is aimed to find the best definition of the relation, which is called the target concept, in terms of some other relations held in the database as background knowledge.

In this thesis study, firstly multi-relational data mining is explained with basic lines and detailed information is given about TMP and graph-based studies in concept discovery problem. Then, using the path finding-based approach of graph mining, a concept discovery system is introduced in which concept identifiers belonging to the Head Output Connected (HOC) class.



**Keywords:** Graph, Support, Confidence, HOC, Concept Discovery.

# GİRİŞ

Hızla gelişen dünyamızda hayatın her alanına giren teknoloji, yapılan sistematik ve bilimsel çalışmaların bir sonucudur. Gerek bilimsel çalışmalarda, gerekse diğer alanlarda kullanılan bilgilerin üretilmesi, saklanması ve ihtiyaç duyulduğunda kolayca ulaşılabilmesi, bilginin büyük veri tabanlarında tutulmasını gerekli kılmıştır. Veri tabanı, herhangi bir konudaki verilerin tutulduğu dijital yapılardır. Bilgilerin güvenli bir şekilde saklanması kadar, saklı bilginin eksiksiz bir şekilde veri tabanından çıkarılması da önemlidir. Veri tabanlarında depolanan bilginin ortaya çıkarılması için veri madenciliği teknikleri kullanılır. Veri madenciliği, büyük ölçekli veri yığınları arasından ilk bakışta görülemeyen, fakat işe yarar anlamlı bilgi ve örüntülerin bulunup çıkarılma işlemi olarak tanımlanabilir.

Çok ilişkili veri tabanları (multi-relational database-MRDB [1]), veri tabanı içinde birbiriyle ilişkili verilerin, her bir ilişki için ayrı birer tabloda tutulduğu veri tabanlarıdır. Çok ilişkili veri madenciliği (multi-relational data mining-MRDM [2]) ise, ilişkisel veri tabanlarından ilk bakışta görülemeyen, örtük bilginin elde edilme sürecidir. Kavram keşfi (concept discovery), bu tür veri tabanlarında hedef ilişki (target relation) olarak adlandırılan bir ilişkinin, arkaplân bilgisi (background knowledge) olarak adlandırılan diğer ilişkiler cinsinden tanımlanması problemidir. Tümevaran Mantık Programlama (TMP) (Inductive Logic Programming-ILP [3]) ve çizge tabanlı yaklaşımlar bu alanda sıklıkla kullanılan yöntemlerdir.

TMP, mantık programlama ve makine öğrenmesinin kesişiminden oluşan bir araştırma alanıdır. Hem istatistik hem de mantık disiplinlerinin prensiplerini kullanır. TMP tabanlı sistemler tanımlayıcı (descriptive) ve tahmin edici (predictive) olmak üzere iki ayrı sınıftan oluşur. Tanımlayıcı TMP sistemlerinde, kullanıcı tanımlı bazı parametreler üzerinden örüntülerin bulunması hedeflenir. Tahmin edici TMP sistemlerinde ise geçmiş deneyimler ışığında öğrenilecek hedef kavramın bulunması sağlanır. Bu aynı zamanda kavram keşfi olarak adlandırılan süreçtir. Bu süreçte arama yönüne dayalı olarak TMP sistemleri, yukarıdan aşağıya (top-down) veya

aşağıdan yukarıya (bottom-up) şeklinde adlandırılır. Yukarıdan aşağıya sistemler, hem pozitif hem de negatif hedef örnekleri kapsayan genel bir varsayımla başlar ve bu varsayım tam (complete) ve tutarlı (consistent) oluncaya kadar özelleştirilir. Varsayımın tam olması, pozitif bütün hedef örnekleri kapsadığı; tutarlı olması ise hiçbir negatif hedef örneği kapsamadığı anlamına gelir. Aşağıdan yukarıya sistemler ise başlangıç varsayımı sadece tek bir hedef örneği açıklayan bir alt cümlecik (bottom clause) ile başlar ve bu cümlecik tam ve tutarlı oluncaya kadar genelleştirilir.

Çizge tabanlı yaklaşımlar, TMP sistemlerde var olan yerel plato (local plateau)[4] gibi bazı problemleri gidermesi ve ilişkisel veri modeli ile saklanan verileri göstermeyi kolaylaştırması gibi avantajlarından dolayı kavram keşif sürecinde TMP'ye alternatif olarak sunulan sistemlerdir. Kavram keşfinde çizge tabanlı sistemler, yol bulma tabanlı (path finding based) ve eşyapı tabanlı (substructure based) olmak üzere iki ana grupta ele alınabilir. Ancak son zamanlarda bu iki sistemin bazı yönlerini bir arada kullanan melez (hybrid) yaklaşımlar da mevcuttur. Yol bulma yaklaşımı, çizge içinde mevcut bir kavramın, onu oluşturan kavram kuralına ait sık ve sonlu uzunlukta yolların bulunduğu esasına dayanır. Eşyapı yaklaşımı ise çizgedeki sık ve benzer alt örüntülerin bir kavram kuralı oluşturduğu varsayımına dayanır [5].

Head Output Connected (HOC) ilişkiler, hedef kavrama ait en az bir çıkış değişkeninin, cümleciğin gövde kısmında bulunduğu geniş bir yüklemler (predicates) sınıfıdır. TMP tabanlı sistemler çeşitli problemlere ait alanlarda genel olarak etkili algoritmalara sahip olsa da, bu sistemlerin HOC sınıfına ait problemlere katkısı oldukça azdır. [6] bu tür problemlerin çözümüne katkı sağlamak için geliştirilen TMP tabanlı bir sistemdir. Çizge tabanlı sistemler içinden öneri geliştiren iki yaklaşım [7,8] olarak gösterilebilir.

Bu tez kapsamında, kavram keşif probleminde HOC ilişkiler için, yol bulma tabanlı yeni bir yaklaşım önerilmektedir.

# 1. GENEL BİLGİLER

## 1.1. Tezin Amacı

Çok ilişkili veri madenciliğinin sıklıkla çalışılan konularından biri, kavram keşif problemidir. Bu problemin çözümünde, uzun zaman TMP yöntemleri baskın bir şekilde kullanılmıştır. Ancak birçok alanda başarıyla uygulamaları bulunan bu sistemlerin verimlilik, ölçeklenebilirlik ve yerel plato gibi bazı sorunları bulunmaktadır [5].

Çizge tabanlı kavram keşif sistemleri, bu sorunlara çözüm sunmak için son yıllarda ilişkisel veri tabanlarından bilgi çıkarımı için yoğun bir şekilde çalışılan yöntemler arasındadır. Tez kapsamında HOC sınıfına giren ilişkisel bir veri modeli üzerinde, yol bulma tabanlı kavram keşif sistemleri için yeni bir yaklaşım sunulmakta ve bu yaklaşımın diğer bazı çizge tabanlı yaklaşımlarla karşılaştırmalı sonuçları anlatılmaktadır. Önerilen yaklaşımda kavram kuralına ait yollar, çizgenin oluşumuna eş zamanlı olarak çıkarılmakta, destek ve güven değerleri diğer yol bulma tabanlı yaklaşımlardan farklı bir yöntemle hesaplanmaktadır.

## 1.2. İlgili Çalışmalar

Bu kısmda kavram keşif problemi için yapılan belli başlı çalışmalar hakkında bilgiler verilmektedir. Öncelikle bu alandaki TMP ve çizge tabanlı yaklaşımların genel bir özeti Tablo 1.1'de verilmekte, devamında ise her bir yaklaşımın metodolojisi üzerinde durulmaktadır. Tablo 1.1'de TMP tabanlı yaklaşımlar, arama yönüne göre yukarıdan aşağıya, aşağıdan yukarıya ya da melez olarak sınıflandırılmaktadır. Çizge tabanlı yaklaşımlar ise, çizge içinde kavram kuralını bulmada izlenecek yöntem bakımından eşyapı, yol bulma ya da melez olarak sınıflandırılmaktadır.

Tablo 1.1'de ayrıca bu alanda yapılan TMP ve çizge tabanlı yaklaşımların, negatif veri kullanıp kullanmadıkları ve özyinelemeyi destekleyip desteklemedikleri de görülmektedir.

Tablo 1.1. Kavram keşif yöntemleri

| Yöntem Adı | Sistemi | Türü | Negatif Veri | Özyineleme |
|---|---|---|---|---|
| LINUS | TMP-tabanlı | Yukarıdan aşağıya | Yok | Yok |
| FOIL | TMP-tabanlı | Yukarıdan aşağıya | Var | Var |
| PROGOL | TMP-tabanlı | Yukarıdan aşağıya | Var | Var |
| ALEPH | TMP-tabanlı | Yukarıdan aşağıya | Var | Var |
| WARMR | TMP-tabanlı | Yukarıdan aşağıya | Var | Var |
| GOLEM | TMP-tabanlı | Aşağıdan yukarıya | Var | Yok |
| CIGOL | TMP-tabanlı | Aşağıdan yukarıya | Yok | Yok |
| $C^2D$ | TMP-tabanlı | Melez | Yok | Var |
| CRIS | TMP-tabanl | Melez | Yok | Var |
| Rel.Pathfinding | Çizge-tabanlı | Yol bulma tabanlı | Var | Var |
| RPBL | Çizge-tabanlı | Yol bulma tabanlı | Yok | Yok |
| MDPF | Çizge-tabanlı | Yol bulma tabanlı | Var | Yok |
| SUBDUE | Çizge-tabanlı | Eşyapı tabanlı | Var | Yok |
| GBI | Çizge-tabanlı | Eşyapı tabanlı | Var | Yok |
| CLIP | Çizge-tabanlı | Eşyapı tabanlı | Yok | Var |
| H-GCD | Çizge-tabanlı | Melez | Var | Yok |
| G-CDS | Çizge-tabanlı | Melez | Yok | Var |

### 1.2.1. TMP tabanlı yaklaşımlar

TMP tabanlı sistemler, yukarıdan aşağıya ve aşağıdan yukarıya olmak üzere temel iki yaklaşıma sahiptir, ancak son zamanlarda bu iki yaklaşımın özelliklerini birleştiren melez yaklaşımlar da literatürde yerini almıştır. Yukarıdan aşağıya sistemlerde hipotez alanı, özelleştirme (specialization) operatörleri kullanılarak taranır. Bu sistemlerde arama, negatif ve pozitif hedef örnekleri açıklayan en genel bir cümlecik ile başlar ve hiçbir negatif örnek kalmayıncaya kadar devam eder. Aşağıdan yukarıya sistemlerde ise genelleştirme (generalization) teknikleri kullanılır. Bu sistemlerde arama, verilen bir hedef örneği açıklayan belirli bir cümlecik ile başlar ve bu cümlecik için tam ve tutarlı bir genelleştirme yapılıncaya kadar devam eder.

#### 1.2.1.1. Yukarıdan aşağıya sistemler (top-down)

LINUS [9], önermesel mantık kullanan, bir özellik-değer öğrenme algoritmasına dayanır. Temel olarak üç adımdan oluşur. İlk adımda öğrenme problemi ilişkisel formdan, özellik-değer formuna dönüştürülür. İkinci adımda dönüştürülen bu öğrenme problemi, özellik-değer yötemleri kullanılarak çözülür ve son adımda elde edilen kural, tekrar ilişkisel forma dönüştürülür.

FOIL [10], mod deklarasyon kullanan top-down ilişkisel bir TMP sistemidir. Mod deklarasyon, hipotez alanındaki cümlelerin baş ve gövde kısımlarında bulunan

yüklemleri ve argümanların tipini tanımlayan bildirimdir. Mod deklarasyon ya modeh(n,atom) ya da modeb(n,atom) şeklinde olur. Buradaki n duyarlılık (recall) değerini ifade eder ve n≥1 ya da '*' olarak gösterilir; atom ise taban (ground) atomu –değişken barındırmayan- ifade eder. Duyarlılık, atom'u örneklemek için kullanılacak örnek sayısını sınırlandırmak amacıyla kullanılır. Bu değerin '*' olarak belirlenmesi olası bütün çözümlerin hesaba katılacağı anlamına gelir. Atomdaki terimler normal ya da yer işaretçi şeklinde bulunur. Bir normal terim ya sabit (constant) ya da parantez içinde terimlerin birleşimi olarak ifade edilen bir fonksiyon sembolü şeklinde olur. Yer işaretçi, tip bir sabit olmak üzere; +tip, -tip, #tip şeklinde olabilir. Burada '+' giriş değişkenlerini,'-' çıkış değişkenlerini, '#' ise sabit değerleri ifade eder. Eğer m bir mod deklarasyon ise a(m), yer işaretleri farklı değişkenlerle değiştirilen m atomunu ifade eder. m bir modeh ise işareti pozitif, m bir modeb ise işareti negatiftir [6].

FOIL, başlangıçta genel bir hipotezle başlar ve bilgi kazanç değerine (information gain value) göre arıtma (refinement) işlemi yaparak her bir literali gövdeye ekler. Negatif hedef örneklerini, kapalı dünya varsayımına [11] dayalı olarak başlangıçta üretir ve sonunda alakasız olan cümleler çözüm setinden çıkarılır. FOIL, özyinelemeli olarak kavram tanımlayıcılarının öğrenilmesine olanak tanır.

PROGOL [12], bir alt cümlecik (bottom clause) tarafından sınırlandırılan bir çizge boyunca arama yapan top-down TMP yaklaşımıdır. Arama alanını budamak (pruning) için mod deklarasyona ihtiyaç duyar. Arıtma esnasında ters gerektirme (inverse entailment) kullanır. PROGOL negatif örnekler içerir, özyinelemeli kural öğrenimini destekler ve gerçek dünya problemlerine uygulanabilir. PROGOL'de cümlecikleri değerlendirme işlemi, f değerine göre yapılır. f değeri aşağıdaki yöntemle hesaplanır.

$$f = p-(n+c+h) \tag{1.1}$$

Yukarıdaki ifadede p değeri, değerlendirmeye konu cümlecikten çıkarılabilecek pozitif ifadedeleri; n, cümlecikten çıkarılabilecek negatif ifadeleri; c, cümle uzunluğunun bir eksiğini; h ise cümleciği tamamlayan ilave atomları ifade eder.

ALEPH [13], arama stratejisi, arıtma operatörleri ve ters gerektirme açısından PROGOL'e benzer ancak değerlendirme ölçütlerinde daha karmaşık bir yapı kullanır. Ayrıca minimum destek (minimum support) ve minimum güven (minimum confidence) gibi daha fazla girdiye de olanak tanır.

WARMR [14], büyük veri tabanlarında, birliktelik kural madenciliğinde (association rule mining) sıklıkla kullanılan APRIORI [15] algoritmasını kullanır. APRIORI, enine arama (breadh-first search-BFS) algoritmasıyla benzer yapıda olup, k-öğe kümesinde (k-itemset) k-1. öğeye baktıktan sonra alt frekans örüntüsü yetersiz olan öğeleri budar ve kalan öğeler üzerinden arama yapar. WARMR, arama alanını sınırlamak için mod deklarasyon kullanır. Baştaki (head) hedef ilişkiyi bulmak için destek ve güven kriterlerine bakar. Tanımlayıcı bir TMP yaklaşımı olduğundan kavram tanımlayıcılar yerine sık örüntüleri (frequent patterns) bulur.

#### 1.2.1.2. Aşağıdan yukarıya sistemler (bottom-up)

GOLEM [16], arama alanını sınırlamak için göreli en az genel genelleştirme (relative least general generalization - rlgg) operatörlerini kullanan, bottom-up bir TMP yaklaşımıdır. Bir alt cümleciği (bottom clause) oluştururken, rastgele seçilen pozitif örneklerin rlgg'sini hesaplar ve kapsamı en geniş olanı çözüm cümlesi olarak sunar.

CIGOL [17], ters çözümleme (inverse resolution) kullanan etkileşimli bottom-up bir TMP yaklaşımıdır. Buradaki temel düşünce A, B'nin çocuğu ise; B, A'nın ebeveynidir ters gerektirme mantığından hareketle, A'yı içeren bütün arkaplân gerçeklerinden ters çözümleme ile bir kavram kuralı bulmaktır.

#### 1.2.1.3. Melez yaklaşımlar (hybrid)

$C^2D$ [18], baştaki hedef kavramı içeren bir dizi kavram kuralını bulmak için birinci derece mantık (first-order logic) kullanan bir TMP yaklaşımıdır. $C^2D$ algoritması, açıklanmamış ilk örneği seçerek başlar. Baş ve gövdede birer tane olmak üzere, iki pozitif literale sahip en genel cümlecikleri oluşturur ve kavram kural alanını (concept rule space), APRIORI tabanlı özelleştirme operatörleri ile arar. Maksimum derinliğe indiğinde güvenin belirlenen eşik değerinden daha düşük eşik değere sahip cümleler, çözüm setinden çıkarılır ve arama kuralı bütün kavram örnekleri için tekrarlanır.

CRIS [19], kavram tanımlayıcılarını bulmak için birliktelik kural madenciliği tekniklerini kullanan bir TMP yaklaşımıdır. CRIS, kavram tanımlayıcılarını değerlendirmek için minimum destek ve minimum güven değerlerini, arama alanını sınırlamak için ise maksimum kural tanımlayıcı uzunluğunu kullanır. Çoğu TMP sisteminin aksine mod deklarasyona ihtiyaç duymaz. Bunun yerine arama alanını budamak için de minimum destek ve minimum güven değerlerinden faydalanır. Bu değerleri hesaplamak için kavram tanımlayıcıları SQL cümlelerine çevirerek, veri seti üzerinde çalıştırır. CRIS, tüm hedef örnekleri eş zamanlı olarak ele aldığı için, hedef örnek sıralama probleminden etkilenmez.

### 1.2.2. Çizge tabanlı yaklaşımlar

Çizge tabanlı sistemler, çok ilişkili veri madenciliğinde kavram keşfi için kullanılan önemli yaklaşımlardandır. Temelde TMP 'nin büyük arama alanlarından kaynaklanan ölçeklenebilirlik ve verimlilik problemlerini çözmek için sunulan güçlü veri modelleridir [5]. Bu sistemler, temel olarak yol bulma tabanlı ve eşyapı tabanlı olmak üzere ikiye ayrılır. Ancak bu iki sistemin bazı özelliklerini birlikte kullanan melez yaklaşımlarda bulunmaktadır.

#### 1.2.2.1. Yol bulma tabanlı yaklaşımlar (path finding)

Relational Pathfinding [7] yol bulma tabanlı ilk yaklaşımlardan biridir. Bu yaklaşım, ilişkisel alanlarda (relational domains) hedef kavramı (target concept) karşılayan ilişki bağlantılarına ait sabit uzunlukta bir yol olduğu varsayımına dayanır. Temelde TMP tabanlı bazı yukarıdan-aşağı (top-down) sistemlerdeki yerel plato ve yerel maksimum problemlerinin çözümüne yardımcı olmak için tasarlanmıştır. Bu yaklaşımda veriler yönsüz çizgelerle gösterilir. Düğümler ilişkili argümanları (arguments), kenarlar ise bu düğümlere ait gerçeklerin (facts) değerlerini ifade eder.

Relational Paths Based Learning (RPBL) [20] diğer bir yol bulma tabanlı kavram keşif yaklaşımıdır. RPBL'de veriler yönsüz ve etiketli çizgelerle gösterilir. Düğümler arkaplân bilgisinden (background knowledge) gelen gerçekleri ifade eder, kenarlar ise ortak argümanlara sahip düğümleri birbirine bağlar. Çizgedeki yollar maksimum uzunluğu kullanıcı tanımlı olan enine arama algoritmasından faydalanılarak bulunur.

RPBL'nin geliştirilmiş formu [21] özyinelemeli olarak kavram tanımlayıcıların bulunmasına olanak tanır. Ayrıca alan teorileri de (domain theories) ele alır.

Mode Directed Path Finding (MDPF) [8], Richard ve Mooney [7]'ye ait pathfinding algoritmasının mod deklarasyon kullanılarak genişletilmiş bir formudur. Relational Pathfinding ve RPBL'den farkı, algoritmasının doymuş alt cümleciklere (saturated bottom clauses) uygulanması ve standart çizgeler yerine hiperçizgeler (hypergraphs) kullanılmasıdır. Doymuş cümlecikler ve hiperçizgeler arasındaki benzer özelliklerden faydalanılarak, doymuş cümleciklerdeki literaller hiperçizgelerde düğüm olarak gösterilir. Uygulamada hiperçizgeler, kaynak düğümler, her bir düğüm ya da literal için giriş/çıkış değişkenleri ve istenen maksimum derinliğe ait bilgiler kullanılarak oluşturulur. Algoritma bütün giriş değişkenlerine ait bağlantı yollarını tutan bir liste döndürür. Düğümler enine arama algoritmasına göre genişletilir ve her bir iterasyonda elde edilen değişken uygun bir listeye eklenir. Kavram tanımlayıcıları bulmak için bu listeler karşılaştırılır ve ortak yollar birleştirilir.

Graph-Based Concept discovery for n-Ary Relations [22], yöntem hedef örnekler ve arkaplân bilgilerini içeren ilişkisel bir veri tabanını, yönlü ve etiketli bir çizge yapısına dönüştürür. Çizge yapısı bir kök düğümle (root) başlar ve hedef örneklerin ait olduğu sınıf sayısı kadar düğüm oluşturur. Daha sonra bunları özel olarak etiketlenmiş bir kenar ile kök düğüme bağlar. Hedef örneklerden çıkan bütün yollar bulunur ve sık olarak bulunan yollar kavram tanımlayıcı olarak ifade edilir. Çizge dolaşımı ve budama işleminde birliktelik kural madenciliği (assosiation rule mining) ve çizge tabanlı yaklaşımlar birlikte kullanılır. Yöntem diğer güncel yöntemlerden, n-Ary (n-argümanlı, n>2 olabilir) ilişkiler kullanması, kavram tanımlayıcıları bulurken yol tabanlı sorgular yapması ve sayısal değerler alabilmesi açısından farklılık gösterir.

#### 1.2.2.2. Eşyapı tabanlı yaklaşımlar (substructure-based)

SUBDUE [23], yönlü ve etiketli çizge kullanır. Düğümler gerçeklere ait argümanları gösterirken, kenarlar bu argümanlara ait ilişki isimlerini tutar. SUBDUE başlangıçta her bir düğümün geçerli bir eşyapıya (substructure) sahip olduğu varsayımına dayanır. Her iterasyonda, her bir eşyapı bir kenar tarafından mümkün olan bütün yollarla genişletilir. Her bir genişlemenin sonunda bulunan eşyapılar başlangıç

çizgesini özetleme başarısına göre budanır. Bu başarı minimum tanımlama uzunluk (minimum description length-MDL) değerine göre belirlenir. Algoritma mümkün olan bütün eşyapılar dikkate alınıp en iyi eşyapı bulununcaya kadar arama işlemini sürdürür.

SUBDUECL [24], çizge gösterimi ve kullanılan temel fonksiyonlar açısından SUBDUE ile ortak yapıdadır ancak öğrenme ve değerlendirme ölçütleri farklılık gösterir. Pozitif ve negatif örnekleri ayırt etmek için çizge özetleme yaklaşımı yerine küme kapsama (set-covering) yaklaşımını kullanır. Eşyapıları değerlendirme işlemi, kapsamadığı pozitif hedef örnekler ve kapsadığı negatif hedef örnekler üzerinden yapılır.

Graph-Based Induction (GBI) [25], yönlü ve renkli çizge kullanır. Renkler gerçeklere ait özellikleri temsil eder. GBI, orijinal çizge boyutunu en aza indirmek için, düğümleri ikili olarak inceler, ortak örüntülere sahip olanları birleştirir ve aynı etikete sahip tek bir düğüm haline dönüştürür. Arama işleminin sonunda birleştirilen son eşyapı kavram (concept) olarak etiketlenir. GBI 'nın geliştirilmiş formunda [26], sadece ağaç yapısındaki çizgeler için değil aynı zamanda çok girişli ve çok çıkışlı düğüm yapısına ve döngüye sahip eşyapı tabanlı çizgeler için çözümler sunulmuştur.

Concept Learning From Inference Patterns (CLIP) [27], yönlü ve renkli çizge kullanır. Renkler, ilgili düğüme ait gerçeğin özellik değerlerini (attribute values of a fact) ifade eder. CLIP çoğu geleneksel kavram öğrenme yaklaşımının aksine, kavramları öğrenmek için pozitif/negatif örnekler yerine sonuç çıkarma örüntülerini (inference patterns) kullanır. İki adımdan oluşur. İlk adımda orijinal sonuç çıkarma örüntülerini renkli, yönlü çizgeye aktarır. İkinci adımda ise; bu yönlü çizge içinde sıklıkla bulunan tipik örüntülerin özetini çıkarır. Bunun için bu örüntülerin tutulduğu görünüm (view) isimli bir liste tutar. Örüntülerin benzerliğini kontrol etmek için çizge özdeşliğinden (graph identity) faydalanır. Kavram tanımlayıcıları bulmak için ise; sırasıyla her bir görünümü inceler ve örüntüleri düğümlere aktararak güncelleme işlemi yapar. En iyi örüntüler, her bir görünümde orijinal çizgeyi en iyi şekilde azaltan düğüm tarafından belirlenir.

#### 1.2.2.3. Melez yaklaşımlar

A Hybrid Graph-Based Method for Concept Rule Discovery (H-GCD) [28], yöntem yol bulma ve eşyapı tabanlı yaklaşımların birleşimidir. Diğer geleneksel çizge yapılarından farklı olarak çizge yapısında düğümler arkaplân bilgisinden gelen ilişkilerin argümanlarına ait bir dizi sabiti tutarken, kenarlar en az bir ortak argümana sahip bu düğüm çiftlerini birbirine bağlar. Kavram tanımlayıcılarını değerlendirme ve budama işlemi destek ve güven değerlerine göre yapılır. Yöntem kendinden önceki yöntemlerden farklı olarak veriyi başlangıç aşamasında ilişkisel veri tabanında tutar ve kavram tanımlayıcıları çizgeyi oluştururken çıkarır. Kavram tanımlayıcıları bulmada, bütün hedef örnekleri eşzamanlı olarak ele aldığı için, kavram tanımlayıcıları bulmada çıkabilecek bir karmaşıklığı önlemiş olur. Yöntem arama algoritması olarak derinlik sınırlı aramayı (depth-limited search-DLS) kullanır. Ayrıca yöntemin çizge yapısı kavram çıkarım sürecini, mod deklarasyona ihtiyaç duymayacak şekilde kolaylaştırır.

Graph-Based Concept Discovery System (G-CDS) [29], diğer bir melez yaklaşımdır. Yöntem girdi olarak hedef örnekler, arkaplân bilgisi, minimum destek, minimum güven ve maksimum kural uzunluğunu alır. Hedef örnekler ve arkaplân bilgisi başlangıçta ilişkisel veri tabanında tutulur. Her bir hedef örnek için ayrı bir çizge oluşturulur. Düğümler, gerçekleri tutarken; kenarlar, en az bir ortak argümana sahip olan düğümleri birbirine bağlar. G-CDS'de her bir hedef örnek için oluşturulan çizgelerden özet bir çizge elde edilir. Bunu yapmak için herbir çizgede gelen kenarı aynı etikete sahip olan düğümler, özdeş olarak adlandırılır ve tek bir düğüm olarak gösterilir. Özet çizge yönlü ve etiketlidir. Kural çıkarımı özet çizge üzerinde derinine arama (depth-first search-DFS) algoritması kullanılarak, maksimum kural uzunluk parametresine göre yapılır. Değerlendirme ve budama işlemi için ise destek ve güven değerlerine bakılır.

Tablo 1.2'de güncel çizge tabanlı yöntemler konusunda özet bilgilere yer verilmektedir. Tablodaki sütunlar çizgelerin türüne, yapısına, çizgelerde verilerin nasıl gösterildiğine, çizgelerin arama algoritmalarına, özyinelemeyi destekleyip desteklemediğine tekabül etmektedir.

Tablo 1.2. Çizge tabanlı yaklaşımlar

| Yöntem Adı | Çizge Türü | Çizge Yap. | Veri Gösterimi | Arm. Alg. | Özyineleme | n-Ary |
|---|---|---|---|---|---|---|
| Relatioanal Pathfinding [7] | Yol bulma tabanlı | Yönsüz | Düğüm-gerçek | BFS | Var | Var |
| RPBL [20] | Yol bulma tabanlı | Yönsüz, Etiketli | Düğüm-gerçek | BFS | Yok | Yok |
| MDPF [8] | Yol bulma tabanlı | Yönlü, Hiperçizge | Düğüm-doymuş alt cümlecikler | BFS | Yok | Var |
| Con.Disc. n-Ary Rel. [22] | Yol bulma tabanlı | Yönlü, Etiketli | İki düğüm-varlık ve özellik | BFS | Yok | Var |
| SUBDUE [23] | Eşyapı tabanlı | Yönlü, Etiketli, Çoklu | Düğüm-gerçeklerin argümanları | Kısıtlanmış Işın Arama | Yok | Yok |
| SUBDUECL [24] | Eşyapı tabanlı | Yönlü, Etiketli | Düğüm-gerçeklerin argümanları | Kısıtlanmış Işın Arama | Yok | Yok |
| GBI [25] | Eşyapı tabanlı | Renkli, Yönlü | Düğüm-gerçeklerin özellikleri | Fırsatçı Işın Arama | Yok | Yok |
| CLIP [27] | Eşyapı tabanlı | Renkli, Yönlü | Düğüm-gerçeklerin özellikleri | Işın Arama | Var | Yok |
| H-GCD [28] | Melez | Yönlü, Etiketli | Düğüm-ilişki argümanları | DLS | Yok | Yok |
| G-CDS [29] | Melez | Yönlü, Etiketli | Düğüm-gerçek | DFS | Var | Yok |
| Önerilen Yöntem | Yol bulma tabanlı | Yönlü | Düğüm-gerçek | DLS | Var | Var |

### 1.3. Tezin Organizasyonu

Bu tez altı bölümden oluşmaktadır. İlk bölümde, tez kapsamındaki konularla ilgili genel bilgiler verilmekte ve daha önce yapılan çalışmalardan bahsedilmektedir. İkinci bölümde, kavram keşfi ve veri madenciliği hakkında temel bilgiler verilmekte, çok ilişkili veri madenciliğinde sıklıkla kullanılan TMP ve çizge tabanlı öğrenmeden söz edilmektedir. Üçüncü bölümde kavram keşif alanında yapılan çizge tabanlı çalışmalar hakkında ayrıntılı bilgilere yer verilmektedir. Dördüncü bölümde önerilen yöntem tanıtılmakta, HOC sınıfı problemi ve yöntemin algoritması anlatılmaktadır. Beşinci bölümde yöntemin uygulandığı deney ortamı, kullanılan veri setleri tanıtılmakta ve yöntemin performans değerlendirmesi yapılmaktadır. Altıncı bölümde ise sonuçlar hakkında genel düşünceler ifade edilmekte, sonraki çalışmalar için öneriler sunulmaktadır.

# 2. KAVRAM KEŞİF SİSTEMLERİ

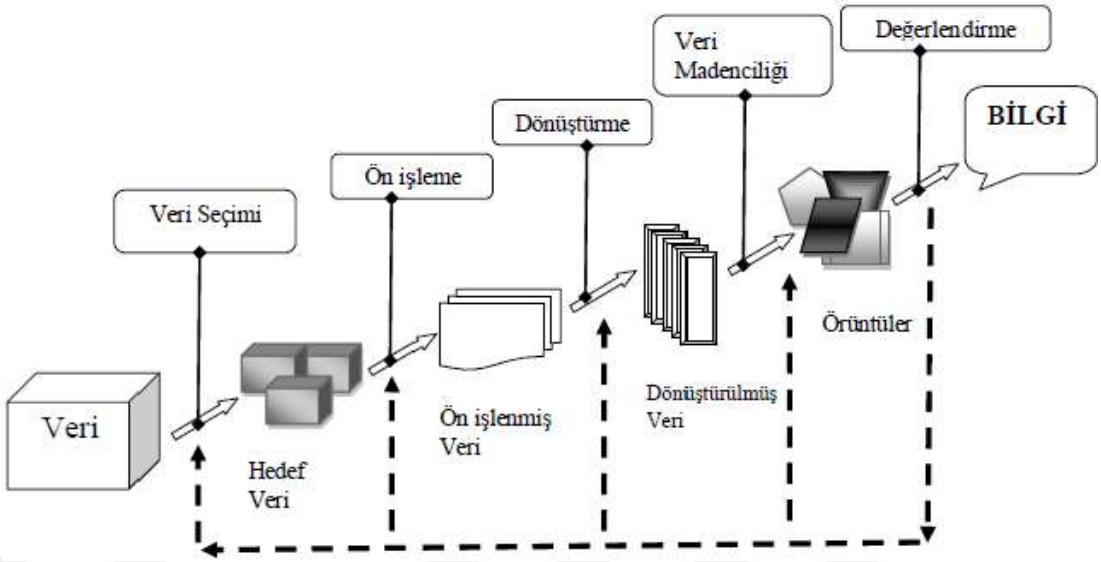
## 2.1. Veri Madenciliği

Hızla gelişen teknolojiye paralel olarak bilgisayarların bilgi saklama kapasiteleri de artmıştır. İlk bakışta herhangi bir anlam ifade etmeyen bu saklı ve ham verilerden çeşitli teknikler kullanarak anlamlı sonuçlar elde etmek, bilgisayar sistemlerini kullanan karar vericiler açısından oldukça önemlidir. Bu ham verileri bilgiye dönüştürmede veri madenciliği tekniklerinden faydalanılır.

Veri madenciliği, büyük veri yığınları arasından ilginç örüntü ve bilgileri elde etme sürecidir. Burada sözü geçen verilere ait kaynaklar; veri tabanları, veri ambarları, internet ya da bilgi akışının dinamik olarak gerçekleştiği diğer ortamlardır. Veriden bilgi madenleme (knowledge mining from data), bilgi çıkarımı (knowledge extraction), veri/örüntü analizi (data/pattern analysis), veri arkeolojisi (data archaeology), veri tarama (data dredging) gibi terimler de veri madenciliği yerine kullanılır [30].

Bilgi keşfi (knowledge discovery), mevcut verilerden daha önceden bilinmeyen değerli ve anlamlı bilgiyi elde etme sürecidir. Veri tabanlarından bilgi keşfi-VTBK (knowledge discovery from databases-KDD ) birçok kaynakta veri madenciliğine eş anlamda kullanılmasına rağmen aslında veri madenciliği, bu sürecin sadece bir aşamasıdır [31].

Şekil 2.1'de veri tabanlarından bilgi keşfine ait aşamalar görülmektedir. Bu aşamalarda, işleme tabi veri, gürültü ve tutarsız yapılardan ön işlem yoluyla arıtıldıktan sonra, farklı formda olanlar aynı forma dönüştürülür. Veri madenciliği, dönüştürülmüş bu veriden anlamlı örüntüler elde etmek için kullanılan çeşitli teknikleri kapsar. Diğer bir deyişle veri madenciliği, veriyi bilgiye dönüştürme esnasında, veriler arasındaki ilişkiyi, düzeni, kuralları ve özellikleri analiz edip ortaya çıkarmak için kullanılan değişik teknikleri içerir.




Şekil 2.1. Bilgi keşif sürecinde veri madenciliği [32]

VTBK sürecindeki başlıca adımlar [33] aşağıdaki gibi listelenmiştir;

• Veri Seçimi: Bu adım birkaç veri kümesini birleştirerek, sorguya uygun örneklem kümesini elde etmek için çalıştırılır.

• Ön işleme: Seçilen veri kümesinde yer alan hatalı verilerin çıkarılması, eksik ve gürültülü niteliklerin arındırılması için gerekli adımdır.

• Dönüştürme: Seçilen veri kümesinden ilgisiz niteliklerin atıldığı ve tekrarlı yapıların ayıklanıp verinin aynı forma sokulduğu adımdır. Bu aşama, seçilen veri madenciliği sorgusunun, çalışma zamanını iyileştirmesi bakımından önemlidir.

• Veri Madenciliği: Uygun bir veri madenciliği tekniğinin işletildiği ve örüntülerin bulunduğu aşamadır.

• Değerlendirme: Bulunan örüntülere ait bilginin; geçerlilik, yenilik, yararlılık basitlik gibi ölçütlere göre değerlendirilmesi aşamasıdır.

Temelde üç tür veri madenciliği modelinden bahsedilebilir: Sınıflama ve regresyon, kümeleme, birliktelik kuralları. Sınıflama ve regresyon tahmin edici model kapsamına girerken, kümeleme ve birliktelik kuralları tanımlayıcı model kapsamındadır [34].

Sınıflama (Classification) ve Regresyon (Regression): Sınıflama ve regresyon veri sınıflarını veya kavramları tanımlayan ve ayıran birer model bulma işlemidir. Model, verilerin bir kısmının eğitim amacıyla kullanılıp, analiz edilmesiyle çıkarılır. Bu model, sınıf etiketi belli olmayan verilerin, sınıf etiketini bulmada kullanılır.

Sınıflama, ayrık değerler için; regresyon ise sürekli değerler için kullanılır. İstatiksel bir metodoloji olan regresyon, ayrıca mevcut veriye dayalı olarak dağılımın tanımlanmasına da olanak tanır [30]. Sınıflama ve regresyon için, karar ağaçları (decision trees), yapay sinir ağları (artificial neural networks), k-en yakın komşu (k-nearest neighbor), naïve-bayes teknikleri sıklıkla kullanılır.

Kümeleme (Clustering): Kümeleme, bir veri kümesindeki kayıtların benzer özelliklerine göre farklı gruplara ayrıştırılmasıdır [35]. Sınıflamada yeni veriler önceden belirlenmiş etiketli sınıflara yerleştirilir. Kümelemede ise sınıf etiketi olan veriler başlangıçta mevcut değildir. Veriler, sınıf içi benzerliği en üst düzeye çıkarma ve sınıflararası benzerliği en alt düzeye indirme ilkesine dayalı olarak kümelere ayrılır. Yani, bir küme içindeki veriler birbirine kıyasla yüksek benzerlik gösterirken, diğer kümelerdeki verilerle oldukça farklı özellikler gösterir. Oluşturulan her küme, kuralların türetilebileceği bir veri sınıfı olarak görülebilir. Kümeleşme taksonominin oluşumunu, yani benzer olayları birlikte gruplayan sınıfların hiyerarşisinde gözlemlerin yapılmasını kolaylaştırır [30]. Hiyerarşik kümeleme, en yakın komşu algoritması, k-ortalama (k-means) algoritması kümeleme için kullanılan bazı yöntemlerdir.

Birliktelik Kuralları (Association Rules): Veri tabanındaki kayıtların birlikte gerçekleşme durumunu ele alan veri madenciliği yaklaşımıdır. Tıp, mühendislik, e-ticaret, eğitim, finans gibi birçok alanda kullanılan birliktelik kurallarının iyi bilinen örneklerinden biri, müşterilerin satın alma eğilimlerini belirlemek için uygulanan pazar sepeti analizidir. Bu uygulamada, yapılan alışverişlerde birlikte satın alınan ürünler üzerinden, müşterilerin satın alma davranışları analiz edilir. Bu durum, pazar yöneticilerine ürünleri daha etkin bir şekilde pazarlama avantajı sağlar. AIS, SETM, Apriori, FP-büyüme (FP-growth) algoritmaları, birliktelik kural madenciliğinde kullanılan algoritmalara örnek olarak verilebilir.

Veri tabanı yönetim sistemleri, yapay sinir ağları, istatistik, yapay öğrenme gibi birçok disiplinin kullanıldığı veri madenciliğinin; bankacılıktan [36] pazarlamaya [37], kriminolojiden [38] istihbarata [39], borsadan [40] telekomünikasyona [41], sağlıktan [42] endüstriye [43] kadar geniş bir uygulama sahası mevcuttur. Özellikle

tıp [44-47], genetik [48-50] ve biyoenformatik [51-53] alanlarında başarılı uygulamaları bulunmaktadır.

### 2.2. Çok İlişkili Veri Madenciliği

İlişkisel veri tabanları, verilerin, satır ve sütunlardan oluşan tablolarda saklandığı yapılardır. Tablolar arasındaki bağlantı, ilişkileri ifade edebilecek kurallar çerçevesinde oluşturulur. Bu ilişkiler sayesinde, verilere sistematik bir şekilde erişim mümkün olur.

Çoğu veri madenciliği yaklaşımı, verideki örüntüleri tek bir tabloda ararken çok ilişkili veri madenciliği yaklaşımları, birbiriyle ilişkili verilerin farklı tablolarda tutulduğu ilişkisel veri tabanlarından örüntüleri elde etmektedir [7]. Tek tabloda veri gösterimi karmaşık verileri depolamak için uygun olmadığından, çok ilişkisel veri tabanları bu tür verileri saklamak için bir çözüm olarak geliştirilmiştir. Verilerin çok-ilişkisel veri tabanlarında saklanması, ilişkili veri madenciliği tekniklerinin önemini arttırmıştır. İlişkisel öğrenme sistemleri iki farklı yaklaşım kullanır [54];

İlk yöntem üç aşamadan oluşur: ön işleme (pre-processing), hipotez oluşturma (hypothesis construction) ve rötuş (post-processing). Klasik veri madenciliği algoritmaları ilişkisel yapılar üzerinde çalışmadığı için ön işlem adımında veri, ilişkisel formdan önermesel forma dönüştürülür. Bu dönüştürme işlemi için uygun özellik-değer öğrenme yöntemlerinden biri kullanılır. Kural çıkarımından (rule extraction) sonra veri tekrar ilişkisel forma dönüştürülür. Bölüm 1'de anlatılan LINUS bu yöntemi kullanan TMP tabanlı bir çalışmadır. LINUS, veri dönüşümünde tümdengelim hiyerarşik veri tabanını (deductive hierarchical database-DHDB) kullanır. Veri ilişkisel DHDB'den önermesel forma dönüştürülür daha sonra ASSISTANT [55], NEWGEM [56] ve CN2 [57] gibi önermesel öğrenme algoritmalarından herhangi biri veriye uygulanır. Özellik-değer gösterim kısıtları, veri dönüşümü esnasında bilgi kaybı, önermesel örüntülerin, ilişkisel örüntüler kadar kolay anlaşılamaması gibi nedenlerle bu yöntem çok tercih edilebilir bir yaklaşım değildir.

İkinci yöntemde, özellik-değer öğrenme sistemi veri madenciliğinin her bir aşaması için geliştirilerek çok ilişkisel öğrenme sistemi içine entegre edilir. Geliştirilmiş veri

madenciliği sistemleri arkaplân bilgisi için veri girişi olarak birinci derece yüklem mantığını (first order predicate logic) kullanır. Hipotez çıkarımı Horn[1] cümlecikleri ile ifade edilir. Verilen arkaplân bilgilerinden mantıksal örüntüler çıkaran öğrenme sistemleri TMP'nin araştırma alanı içinde toplanmıştır. TMP tabanlı öğrenme sistemleri, veri madenciliğine ait tahmin (prediction), sınıflama (classification) ve birliktelik kural madenciliği (association rule mining) gibi geliştirilmiş makine öğrenmesi tekniklerini kullanır. TMP tabanlı FOIL önermesel kural çıkarım sistemi olan CN2 [58] 'nin geliştirilmiş bir formu olarak görülebilir. Ayrıca PROGOL kural çıkarımı için, AQ yaklaşımının [59] geliştirilmiş yapısı olarak düşünülebilir.

### 2.3. Kavram Keşfi

Kavram (concept), veri tabanlarındaki saklı ilişkiler kullanılarak bulunacak bir dizi örüntü olarak tanımlanabilir [60]. Kavram keşfi (concept discovery), hedef ilişki (target relation) olarak adlandırılan ilişkinin; arkaplân bilgisi (background knowledge) olarak adlandırılan diğer ilişkiler cinsinden tanımlanmasıdır [9]. İlişkisel veri tabanlarında kavram keşif süreci tahmin edici bir öğrenme (predictive learning) yaklaşımıdır. Bu süreçte geçmiş tecrübelerle elde edilen mevcut bilgilerden, öğrenilecek kavrama ait kurallar elde edilmesi hedeflenmektedir.

İlişkisel veri tabanlarından kavram keşfine ait temel terminoloji aşağıdaki gibi ifade edilebilir [19];

Kuram (Theory): Bir kurama ait en temel birimler değişken ve sabitlerdir. Değişken adları büyük harfle, sabit adları ise küçük harfle başlar.

A, B23 ifadeleri birer değişken; ab, nur ifadeleri ise birer sabit belirtir.

Atom (Atom): Bir yüklem sembolünü izleyen parantez içindeki n-üye'li (n-tuple) terimlerin oluşturduğu ifadedir.

ata(A,ayten) ifadesi yüklem sembolü ata olan 2 üyeli bir atom iken; plus(A, B, C) ifadesi yüklem sembolü plus olan 3 üyeli bir atomdur.

[1] Horn cümleciği, en fazla bir pozitif önerme içeren, pozitif ya da değillenmiş önermelerin ‘veya ’ bağlacı ile bağlanması sonucu oluşur.

Literal (Literal): Bir atomun normal formu $p(arg_1, ..., arg_n)$ şeklinde gösterilir ve pozitif literal olarak adlandırılır. Bu atomun değillenmiş formu olan $\neg p(arg_1, ...,arg_n)$ ifade ise negatif literal olarak adlandırılır.

Kavram kuralı (Concept rule): Kavram kuralı h← b şeklinde gösterilen bir birliktelik kuralıdır. Burada h kuralın baş (head) kısmını; b ise kuralın gövde (body) kısmını göstermektedir. Kuralın gövde kısmı - b -, birden fazla atom/literal'den oluşabilir.

Hedef ilişki (Target relation): Hedef ilişki öğrenilecek kavrama tekabül eden yüklem olarak tarif edilebilir. Hedef ilişkilere ait örnekler, bulunan örüntü tarafından doğru bir şekilde kapsanmalıdır. Kural formuna uyan bir örüntüde, hedef ilişki kuralın baş kısmında görünür. Ancak özyinelemeli kurallarda gövde kısmında da görülebilir.

Arkaplân ilişkisi (Background relation): Arkaplân ilişkisi, kavram keşif sürecine dâhil olan fakat hedef ilişkiden farklı diğer ilişkileri ifade eden yüklemdir. Kural formuna ait örüntüde, arkaplân ilişkisi genel olarak kural gövdesinde bulunur. Örneğin ata veri tabanına ait tabloda ata hedef ilişkiyi, anne ve baba arkaplân bilgisini ifade eder. Ata veri tabanına ait bir kavram kuralı;

ata(A,B)← baba(A,C),anne(C,B),bayan(C)

şeklinde tanımlanabilir.

Tablo 2.1. Ata veri tabanı

| Kavram örnekleri | Arkaplân gerçekleri |
|---|---|
| ata(sabihamelek,harun)<br>ata(sebahattin,memduh)<br>ata(ali,aysun)<br>ata(adalet,nur)) | anne(sabihamelek,şakir)<br>anne(gülseren,memduh)<br>baba(sebahattin,gülseren)<br>baba(şakir,harun)<br>baba(ali,ayten)<br>anne(ayten,aysun)<br>anne(adalet,ali)<br>baba(ali,nur)<br>bayan(sabihamelek)<br>bayan(gülseren) |

Cümlecik (Clause): Bir cümlecik evrensel formu $\forall(l_1 \vee l_2 \vee \ldots \vee l_n)$ şeklinde olan sayısal bir gösterimle ifade edilir. Cümlenin anlamı, bağlamından açıkça anlaşılabiliyorsa $\forall$ cümleden çıkarılabilir. Bu cümleciğin $h_1 \vee h_2 \vee \ldots \vee h_p \vee b_1 \vee b_2 \vee \ldots \vee b_r$ formundaki gösteriminde $h_i$'ler pozitif literalleri; $b_j$'ler negatif literalleri ifade eder. Bu cümlecik $h_1 \vee h_2 \vee \ldots \vee h_p \leftarrow b_1 \wedge b_2 \wedge \ldots \wedge b_r$ şeklinde de gösterilebilir. Burada $p \geq 0$ için $h_1 \vee h_2 \vee \ldots \vee h_p$ ifadesi cümleciğin baş kısmı; $r \geq 0$ için $b_1 \wedge b_2 \wedge \ldots \wedge b_r$ ifadesi ise cümleciğin gövde kısmı olarak adlandırılır.

$\forall$(kızı(A,B) $\vee \neg$ ebeveyn(B,A) $\vee$ bayan(A)) bir cümlecik belirtir. Bu cümleciği;

kızı(A,B) $\vee$ bayan(A)$\leftarrow$ebeveyn(B,A) şeklinde yazmak da mümkündür.

kızı(A,B) $\vee$ bayan(A) ifadesi cümleciğin baş kısmını; ebeveyn(B,A) ise cümleciğin gövde kısmını belirtir.

Belirli cümlecik (Definite clause): Belirli cümlecik sadece bir tane baş literal içeren cümledir. Gövde kısmı boş olan belirli cümlecik gerçek (fact) olarak adlandırılır. Baş kısmı boş olan cümlecik ise ret (denial) olarak ifade edilir.

kızı(A,B) $\leftarrow$ bayan(A) $\wedge$ ebeveyn (B,A) belirli bir cümleciği ifade eder.

Sorgu(Query): Bir sorgu $\exists(l_1 \wedge l_2 \wedge \ldots \wedge l_n)$ sayısal bağlayıcı ifade ile gösterilir. Sorgunun anlamı, cümleciğin bağlamından açıkça anlaşılıyorsa $\exists$ kullanılmaz.

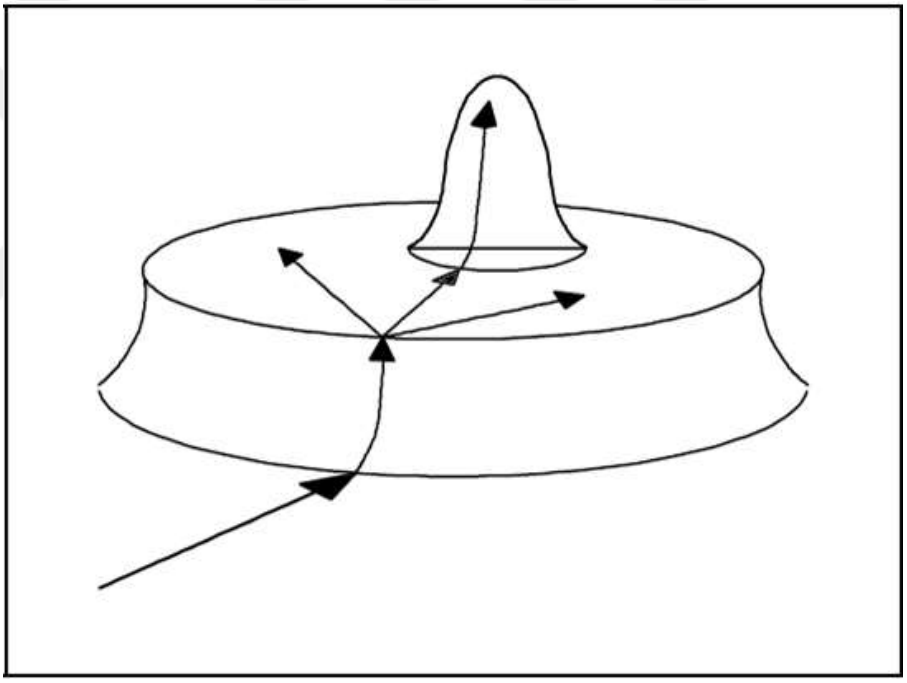
$\exists$(kızı(A,B) $\wedge$ ebeveyn(B,A)) ifadesi bir sorgu belirtir.

Kavram keşif sistemlerinin performansını değerlendirmek için iki ölçüt vardır: tamlık (completeness) ve tutarlılık (consistency). Bir kavram keşif sistemi, bütün pozitif hedef örnekleri kapsıyorsa sistem tam; hiç bir negatif hedef örneği içermiyorsa sistem tutarlı olarak adlandırılır. Gerçek dünya problemlerine ait veriler genelde gürültülü olduğu için, tamlık ve tutarlılığa ait kısıtlar mümkün olduğunca çok pozitif hedef örneğin ve mümkün olduğunca az negatif hedef örneğin kapsanması yönünde genişletilir.

Kavram keşif süreci iteratif bir şekilde ilerler. Her bir iterasyonda sınırlı sayıda hedef örneğe ait kavram tanımlayıcılar bulunur. Sistem kalan hedef örneklere ait kavram tanımlayıcılarını bulmak için yeniden başlar. Bu süreç, bütün hedef örnekler

kapsandığında, kapsanmamış hedef örnekler, belirlenmiş eşik değerin altına düştüğünde, ya da kalan hedef örneklerden çıkarılabilecek kavram tanımlayıcı kalmadığında son bulur.

TMP tabanlı çalışmalar, uzun süre kavram keşif probleminde başat rol oynamıştır. Biyoenformatik [61,62], mühendislik [63,64], doğal bilimler [65] ve doğal dil işleme [66,67] gibi alanlarda çalışmalar mevcuttur. Tepe tırmanma (hill-climbing) algoritması kullanan FOIL [10], FOCL [68], Forte [69] gibi TMP tabanlı yaklaşımlar, yerel plato problemiyle karşılaşmaktadır. Bu tür sitemlerde öğrenme süreci her bir adımda bir literal eklenmesiyle devam eder. TMP 'nin geleneksel arıtma operatörleri bu durumda kavram tanımlayıcılarını iyileştirmede etkisiz kalır.



Şekil 2.2. Yerel plato problemi [7]

Çizge tabanlı yaklaşımlar TMP'nin karşı karşıya kaldığı bu soruna çeşitli yöntemlerle çözüm sunmayı amaçlamaktadır. [7] bu problemin çözümünde birden fazla literali hipotez alanına eş zamanlı olarak eklemek suretiyle, kuralın sınıflama doğruluğunu geliştirmeyi amaçlamaktadır. Yine çizge tabanlı [27,70] gibi yaklaşımlar kavram keşif probleminde TMP tabanlı yaklaşımlarla aynı doğrulukta ancak verilerin daha doğal bir yolla gösterildiği ve sonuçların daha kolay yorumlanabildiği olumlu sonuçlar vermiştir.

Gerek TMP tabanlı yaklaşımlar, gerekse çizge tabanlı yaklaşımlar veri gösterimi, arama stratejisi gibi konularda temel alt bölümlere ayrılır. TMP, tanımlayıcı ve tahmin edici; çizge tabanlı yaklaşımlar ise, eşyapı tabanlı ve yol bulma tabanlı olmak üzere temel iki alt sınıftan oluşur.

## 2.4. TMP Tabanlı Öğrenme

TMP, hem makine öğrenmesi hem de mantık programlamanın tekniklerini kullandığı için bu iki disiplinin kesişimi olarak tanımlanabilir [3]. TMP'nin makine öğrenmesinin araç ve tekniklerini kullanmadaki amacı gözlemlerden hipotezler üretmek ve elde edilen tecrübelerden yeni bilgiler elde etmektir. Gösterim mekanizması için hesaplamalı mantık (computational logic) kullanmadaki amacı ise; klasik makine öğrenmesinin iki kısıtının üstesinden gelebilmektir [71]:

1. Sınırlı bilgi gösterimi kullanabilme olanağı (özellikle önermesel mantıkta)
2. Öğrenme sürecinde çok fazla miktardaki veri için arkaplân bilgisi kullanmadaki zorluklar

Mantık programlama, örneklerden sentez yapmaya dayandığı için; birçok mantık programını ifade etmede önermeler mantığı yetersiz kalır. Bu tür programlar için birinci derece yüklem mantığı kullanılır. Bundan dolayı TMP'nin birinci kısıtı açıklamadaki katkısı önemlidir. Arkaplân bilgisini kullanmadaki katkısı ise; yapay zekâ alanlarında, zekâ davranışını öğrenmede arkaplân bilgisinin kullanılmasının gerekli oluşudur [71].

TMP sistemler bazı kriterlere göre sınıflandırılabilir [72];

• Tek bir hedef kavramı mı yoksa birden çok hedef kavramı mı kullanır

• Hipotezi başlangıçta bütün eğitim örneklerini alarak mı bulur (grup öğrenme-batch learners); yoksa örnekleri birer birer mi ele alır (artış öğrenme-incremental learners)

• Sistem tarafından üretilen örneklerin sınıflandırılmasında kullanıcı aktif durumda mı (interactive learners) değil mi (non-interactive)

• Hedef kavram için kısmî bir tanımdan mı, bir başlangıç noktasından mı (scratch) hareket eder

• Hipotez için gösterim dili L nasıldır

Yukarıdaki kriterlere göre TMP sistemler, deneysel (empirical) ve etkileşimli (interactive) olarak ayrılabilir. Deneysel TMP'de aynı anda birden çok hedef kavramı öğrenmeye dayalı sistemler mevcut olsa da, büyük çoğunluğu büyük veri seti arasından tek bir kavramı öğrenmek üzere tasarlanmıştır. Bu sistemler grup öğrenmeye dayalı, etkileşimsiz, başlangıç noktasından (batch-learner, non-interactive, scratch) kavram öğrenmeye dayalıdır. FOIL [10], GOLEM [16], PROGOL [12] bu kategoriye giren örnekler olarak gösterilebilir.

Etkileşimli TMP sistemlerde ise, küçük veri setlerinden birden çok hedef yüklemin bulunması amaçlanır. Bu sistemler kısmî olarak oluşturulan hipotezin doğrulanması ve üretilen yeni örneklerin sınıflandırılması aşamasında kullanıcı deneyimine başvurur. MIS [73], MARVIN [74], CIGOL [17] bu alandaki çalışmalara örnek olarak verilebilir.

TMP kuramı, arkaplân bilgisi ve örneklem kümesini kullanarak tümevaran mantıksal çıkarım yapmayı amaçlar. Bu mantıksal çıkarıma ait terimeler aşağıdaki gibi açıklanabilir [71];

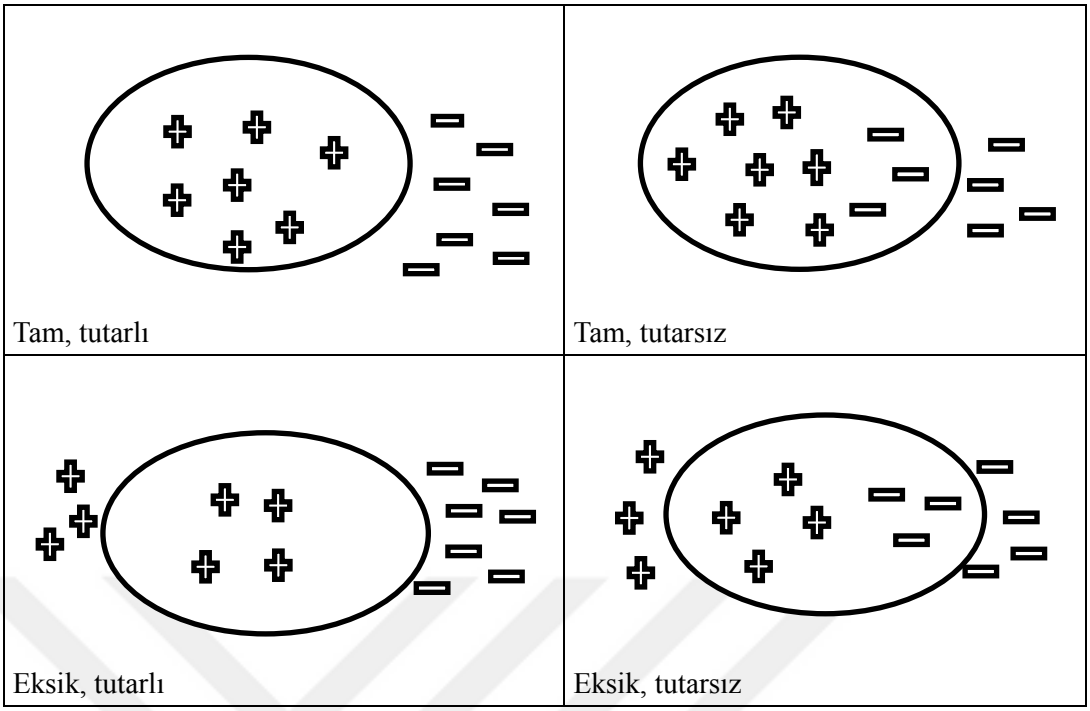
$E = E^+ \wedge E^-$ ifadesinde $E^+$ pozitif örnekleri, $E^-$ negatif örnekleri ifade etmek üzere;

Verilen B, arkaplân bilgisi ve E, örneklem kümesi için aşağıdaki durumlar sağlandığında H hipotezi elde edilir.

- Öncelikli karşılayabilirlik (prior satisfiability) : $B \wedge E^- \not\models \square$
- Sonraki karşılayabilirlik (posterior satisfiability) : $B \wedge H \wedge E^- \not\models \square$
- Öncelikli gereklilik (prior necessity) : $B \not\models E^+$
- Sonraki yeterlilik (posterior sufficiency) : $B \wedge H \models E^+$

Elde edilen H hipotezinin değerlendirilmesi için tamlık ve tutarlılık kriterleri kullanılır. Eğer hipotez bütün pozitif örnekleri içeriyorsa tam; hiçbir negatif örneği içermiyorsa tutarlı olarak değerlendirilir.

Şekil 2.3 pozitif (+) ve negatif (-) örnekleri içeren bir örneklem kümesine ait durumları göstermektedir [75].



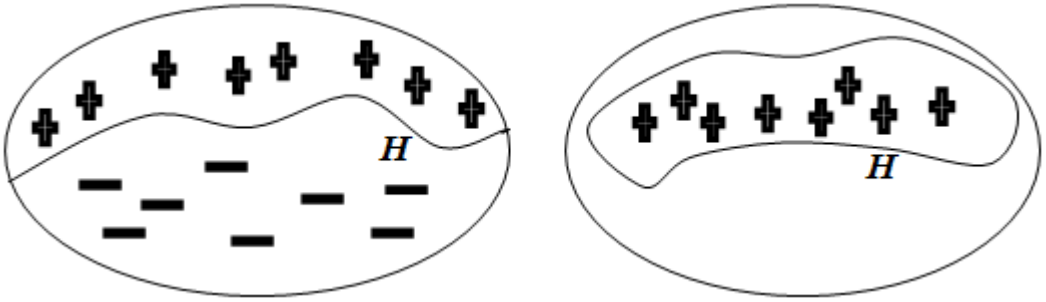

Şekil 2.3. Bir H hipotezi için tamlık ve tutarlılık durumları

Gerçek dünya uygulamalarına ait karmaşık verileri barındıran ilişkisel veri tabanlarından bilgi çıkarımı, TMP'nin çalışma alanlarından biridir. İlişkisel veri tabanı algortimaları doğrudan veri tabanları üzerinde çalışır [7]. TMP'nin öğrenme sistemleri bu alanda gösterim dili olarak birinci derece yüklem mantığını kullanır. Bu öğrenme sistemleri, mantıksal örüntüleri arkaplân bilgisine dayalı olarak bulur [76]. TMP sistemler, öğrenme teknikleri bakımından tahmin edici (predictive) ve tanımlayıcı (descriptive) olmak üzere iki temel sınıftan oluşur. Tahmin edici modelde öğrenilecek hedef bir kavram mevcutken, tanımlayıcı modelde verideki örüntülerin tanımlanması hedeflenmektedir [77].

### 2.4.1. Tahmin edici (predictive) TMP

Tahmin edici öğrenme, geçmiş deneyimlerle elde edilen bilgiler ışığında mevcut bilgilerden özel bir hedef kavramı elde etme süreci olarak tanımlanabilir. Tahmin edici veri madenciliğinde, verilen bir dizi örnek ya da gözlem, sonlu sayıdaki sınıflara ayrılır. Temelde iki ya da daha çok sınıftan biri pozitif diğerleri ise negatif olarak adlandırılır. Buradaki temel amaç bütün örneklerin doğru bir şekilde sınıflandırıldığı bir hipotez üretmektir.

Şekil 2.4 [78] (a)'da pozitif(+) ve negatif(-) iki örneklem kümesi verilmiş olsun. H hipotezi, pozitif ve negatif örnekleri birbirinden doğru bir şekilde ayırdığında, doğru bir hipotez olur [78].




(a) (b)

Şekil 2.4. a) Tahmin edici çıkarım b) tanımlayıcı çıkarım

Tahmin edici öğrenme yaklaşımı çerçevesinde; C hedef kavramı, E pozitif ve negatif örneklem sınıfını, B sonlu sayıdaki arkaplân gerçeğini, L kavram tanımlama dilini göstermek üzere;

Sonlu sayıdaki H cümleciği, pozitif bütün $E^+$'leri içeren (tam), negatif hiçbir $E^-$'yi içermeyen (tutarlı) arkaplân bilgisi B ile birlikte kavram tanımlama dili L'yi ifade eder. Bu durumda; H, B ve E'ye göre tam ve tutarlı olur [3].

Tamlık ve tutarlılık, bu tür problemlerin çözümünde hipotezlerin kalitesini ölçmek için kullanılan iki kriterdir. Bu terimlerin tanımından hareketle, hipotezlerin verilen örneklere bütünüyle uyması gerektiği sonucuna varılabilir. Çünkü arkapalan bilgisi ya da eğitim kavram örnekleri hatalı olabilir. Veya eğitim örnekleri, kavram içinde gizli genel düzeni yansıtmak konusunda yeterli olmayabilir [75].

Tahmin edici öğrenme yaklaşımının temel amacı, kavram tanımlayıcı örneklere en iyi uyan tam ve tutarlı bir hipotez elde etmektir. Hedef kavrama ait farklı her bir örüntü, hipotezde bir cümlecik olarak gösterilir. Kavram cümleciğine ait destek değeri, yapısal örüntüye uyan pozitif örnekler üzerinden hesaplanır.

Tahmin edici TMP sistemleri, hedef kavramı bulmak için hipotez alanını ya aşağıdan yukarıya ya da yukarıdan aşağıya tarar. Aşağıdan yukarıya yaklaşımlar, verilen bir pozitif örneği içeren en özel cümlecik ile başlar ve arkaplândaki bütün pozitif örnekleri içeren kavram tanımını buluncaya kadar hipotezi genelleştirir. Yukarıdan

aşağıya yaklaşımlar ise, kavrama ait bütün pozitif ve negatif örnekleri içeren en genel hipotezle başlar ve sonunda hipotezin sınırlarını hiçbir negatif örnek içermeyecek şekilde daraltır [76].

Tahmin edici TMP yaklaşımı herhangi bir sınıflama ya da tahmin etme problemine uygulanabilir. Örneğin LINUS [9], QuMAS [79]'ın öğrenme bileşenlerini geliştirerek önermesel öğrenme görevi içine entegre eden bir çalışma iken, [80] kimyasal bileşenlerin yapılarına göre kanserojen etkilerini tahmine dayalı bir çalışmadır. Yaygın olarak kullanılan tahmin edici TMP teknikleri C4.5 [81] gibi karar ağaç çıkarımı (decision tree induction) , CN2 [58] ve AQ [59] gibi kural çıkarımı (rule induction) teknikleridir.

### 2.4.2. Tanımlayıcı (descriptive) TMP

Tanımlayıcı veri madenciliği, tahmin edici veri madenciliğinden farklı olarak doğrudan bir hedef kavramı aramaz. Tanımlayıcı TMP, veri tabanlarından sık örüntüleri arar; kural sonucu, herhangi bir özellik ya da ilişki olabilir. Diğer bir deyişle, bir ürün almak ve bu ürünün kanser yapıcı etkisi gibi eylem-özellik değerlendirmesi yapmayı, ya da kişi-kişinin sahip olduğu özel genetik yapı arasındaki ilişkileri bulmayı hedefler [78].

Tanımlayıcı öğrenme sistemi, tahmin edici öğrenmenin aksine hedef kavramı etiketli örneklerden öğrenen, birliktelik kural öğrenme ve sık örüntü keşfi gibi tanımlayıcı öğrenme görevleri için sınıf etiketi kullanmaya ihtiyaç duymaz. Tanımlayıcı TMP, bir dizi gerçekten oluşan eğitim örneklerini, üye kural ve örüntüleri doğrulamak için kullandığı kapsam testlerini kullanarak sağladığı verilerden yorumlamaya [82] dayalı olarak öğrenme sağlar.

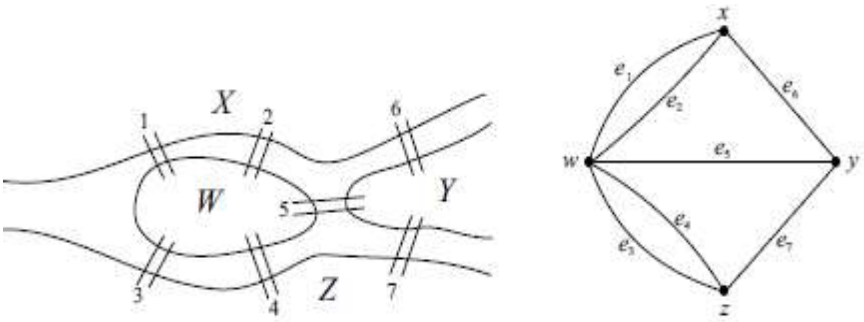
Tanımlayıcı TMP, verilen örneklerden mümkün olduğunca çok çıkarım yapmayı hedefler. Şekil 2.4 (b)'de, H hipotezi verilen bütün örnekleri karakterize edebilecek bir çıkarım yapmaktadır.

Tanımlayıcı TMP'nin temel amacı, kullanışlı, anlaşılabilir ve ilginç örüntüler bulmaktır. Örüntülerin gösterim dili ve ilginçliği, tanımlayıcı TMP'nin başarısında kullanılan temel kriterlerdir.

WARMR [14], CLAUDIEN [83], HR [84] iyi bilinen tanımlayıcı TMP yaklaşımlarıdır. WARMR, APRIORI kurallarını kullanarak herhangi bir özel hedefe ihtiyaç duymaksızın, örüntüleri bulmayı hedefler. CLAUDIEN, eğitim örneklerini tanımlayan sınıflama örneklerini üretmek için, pozitif örnekleri kullanarak karakteristik bir çıkarım sağlar. HR yapıcı dil yargısı (constructive language bias) üzerinden otomatik kuram formasyonunu (automated theory formation) kullanır. Arkaplân bilgisi olarak sunulan bir dizi sınıflama kuralından başlayarak mevcut bir kuralı geliştirmeyi veya mevcut iki kuralı birleştirmeyi amaçlar. Tahmin etme doğruluğunu yükseltmeyi amaçlayan PROGOL [12], FOIL [10] ve [85] gibi biyoenformatik alanında uygulanan geliştirilmiş tanımlayıcı TMP yaklaşımları da bulunmaktadır.

## 2.5. Çizge Tabanlı Öğrenme

Çizge teorisi, temeli Königsberg'in yedi köprüsüne dayanan bir matematik problemidir. Königsberg'deki Pregel Nehri kenti dört bölgeye ayırmakta ve bu bölgeler Şekil 2.5'te görüldüğü gibi yedi köprü ile birbirine bağlanmaktadır. Teorinin doğmasına konu olan problemde, köprülerden sadece bir kez geçmek ve bütün köprüleri dolaşmak koşuluyla gezinti yapmanın mümkün olup olmadığı sorusuna yanıt aranmaktadır. Matematikçi Euler 1736'da yaptığı çalışmada böyle bir gezintinin mümkün olmadığını göstermiş ve bu gezintiyi yapmaya olanaklı çizgelerin özelliklerini açıklamıştır.




Şekil 2.5. Königsberg'in yedi köprüsü [86]

Çizgeler, bilgisayar biliminde temel veri yapısı olarak kullanılır. Çizge teorisine ait temel terminoloji aşağıdaki gibi açıklanabilir [86];

Çizge (Graph): Bir G çizgesi, düğümler (vertices or nodes), bu düğümleri bağlayan kenarlar (edges) ve kenarların bağladığı düğümleri gösteren f fonksiyonundan oluşur ve aşağıdaki şekilde gösterilebilir;

$$G(V,E,f) = \begin{cases} V \text{ düğümler kümesi} \\ E \text{ kenarlar kümesi} \\ f \text{ bağıntı fonksiyonu } f : E \rightarrow V \text{x} V \end{cases}$$

Basit çizge (simple graph): Döngü ya da çoklu kenar içermeyen çizgedir.

Yol (path): Ardışık düğümler arasındaki bağlantıyı ifade eden basit bir çizge olarak tanımlanabilir. Bir G çizgesinde u düğümünden başlayıp v düğümüne kadar birbirini izleyen komşu düğümlerin oluşturduğu yol uv yolu olarak adlandırılır. Böyle bir yol n uzunluğuna sahip ise; bu yolun, n+1 tane düğümü, n tane de ardışık kenarı vardır.

Çevrim (cycle): Döngüsel sıralamada ilerleyen başlangıç ve bitiş düğümü aynı olan basit çizgedir.

Çoklu çizge (multi graph): Basit çizgelerin yeterli olmadığı durumlarda kullanılan, yönsüz ve döngü içermeen çizgedir.

Yönlü çizge (directed graph): Her bir kenarı sıralı bir düğüm çifti ile ilişkilendirilerek, yön tayin edilmiş çizgedir.

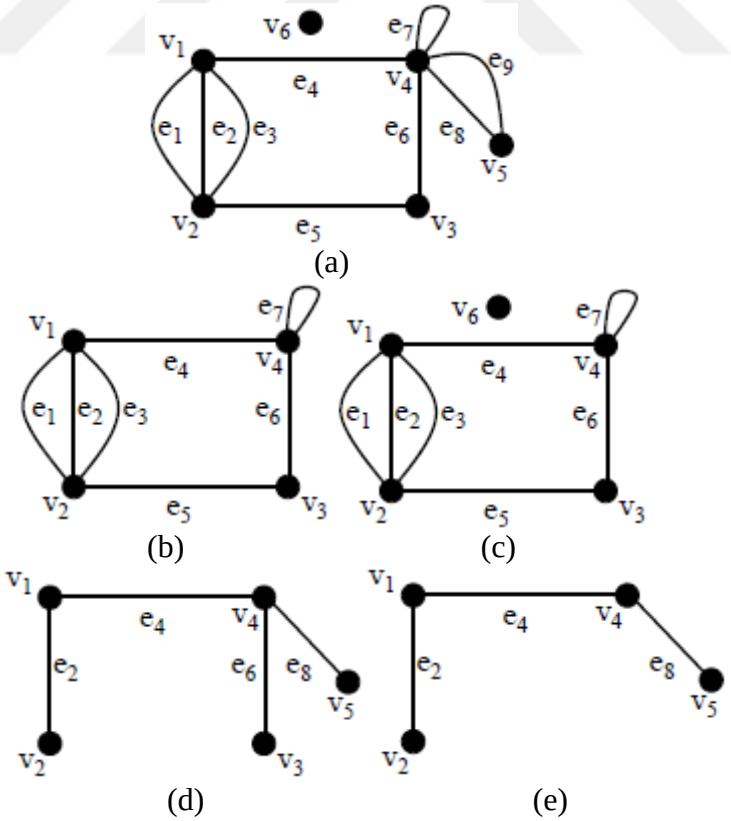
Sözde çizge (pseudo graph): Yönsüz, çoklu kenarı olan ve döngü içeren çizgedir. Bu çizgeler, yönsüz olarak gösterilen çizgelerin en temel biçimidir.

Ağırlıklı çizge (weighted graph): Her bir kenarına sayısal bir değer verilmiş çizgedir.

Altçizge (subgraph): H ve G çizgeleri verilmiş olsun. V (H) $\subseteq$ V (G) ve E(H) $\subseteq$ E(G) şartı sağlandığında, G çizgesinde e'nin bağladığı u ve v düğümleri için, e $\in$ E(H) şartı ve H çizgesinde u ve v'yi bağlayan e kenarı için u, v $\in$ V (H) şartı da sağlanıyorsa H çizgesi, G çizgesinin altçizgesidir denir. Diğer bir deyişle, H çizgesindeki kenar ilişkileri G çizgesindeki kenar ilişkileri ile aynı ise H, G'nin altçizgesidir.

Çizgeler yapısal verilerin gösteriminde oldukça kullanışlı veri yapılarıdır [87]. Bu özelliklerinden, ilişkisel veri modelindeki kullanımları giderek artmaktadır. Çizge tabanlı veri madenciliği, çizgelerin matematikteki uzun geçmişine dayalı olarak veri madenciliği alanında yeni prensipler sunmaktadır. Çizgeler, bu alanda birçok farklı algoritmaya sahip olsa da; kuramsal bazda beş temel faktörden bahsetmek mümkündür [88]:

Altçizge kategorileri (subgraph categories): Çizge tabanlı veri madenciliği algoritmaları, bir çizgeye ait genel yapıdan belirli özellikteki altçizgeleri bulmayı amaçlar. Bu altçizgeler, bağlı altçizge (connected subgrapg), kaynaklı altçizge (induced subgraph), ağaç (tree) ve yol (path) şeklinde olabilir. Bağlı altçizgelerde, bazı kenarlardan bütün düğümlere erişmek mümkündür. Kaynaklı altçizgeler, ana çizgenin bir alt kümesidir ve yapısındaki düğüm çiftlerini bağlayan kenarlar, ana çizgede aynı düğümleri bağlayan kenarlar ile özdeştir. Ağaç altçizgeleri döngü içermeyen yapılardır. Yol altçizgeleri ise düğümlerin herhangi bir dallanma içermediği yapılardır. Şekil 2.6'da altçizge yapıları görülmektedir.




Şekil 2.6. Altçizge yapıları a) ana çizge b) bağlı altçizge c) kaynaklı altçizge d) ağaç yapısı e) yol yapısı [88]

Altçizge eşbiçimciliği (supgraph isomorphism): Algoritmalar, farklı çizgelerdeki alt kümelere ait benzerlikleri ortaya çıkarmak için, altçizgelerdeki düğümler ve kenarlar arasında eşleme yöntemini kullanır. Aynı düğüm ve kenarlara sahip altçizgeler özdeş olarak adlandırılır. Çizge tabanlı veri madenciliğinde, altçizge eşbiçimciliği problemi, birden fazla çizgeyi kapsayacak şekilde genişletilir. Altçizge eşbiçimciliği, verilen çizgelerin topoojik olarak özdeş kısımlarına ait hesaplama ve eşleme temelini oluşturur.

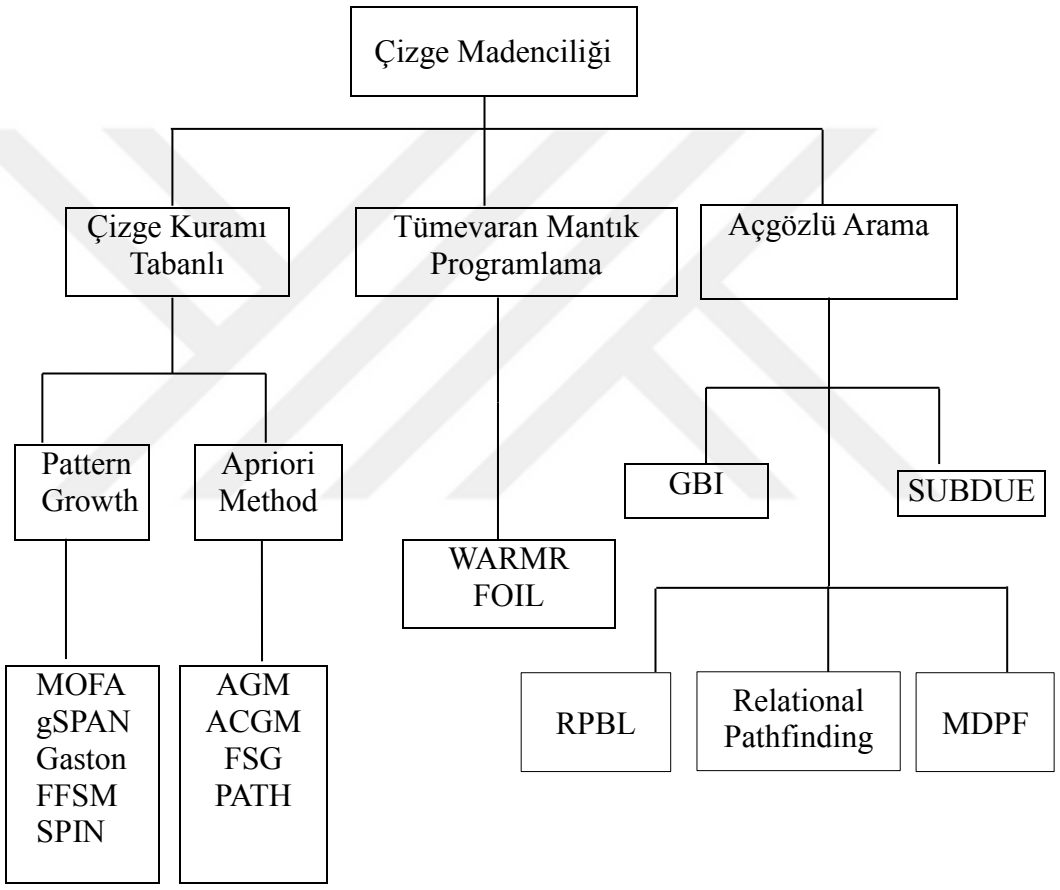
Madencilik ölçümleri (mining measures): Çizge madenciliğinde geleneksel veri madenciliği ölçümlerine benzer şekilde çeşitli ölçüm araçları kullanılır. Bu araçlar, amaca ve madencilik yaklaşımının kısıtlarına göre değişebilir. Çizge madenciliğinde en çok kullanılan ölçüm araçlarına; destek (support), bilgi kazancı (information gain), minimum tanımlama uzunluğu (minimum description length-MDL), F-1 ölçümü (F-1 measure) gibi araçlar örnek olarak verilebilir.

Çizge değişmezleri (graph invariants): Çizgelerin topolojik yapısına ait özellikleri sayısal olarak ifade eden değerler çizge değişmezleri olarak adlandırılır. Eğer iki çizge topolojik olarak özdeşse, aynı çizge değişmezlerine sahiptir ama bunun tersi geçerli değildir. Yani değişmezleri eşit sayıda olduğu için iki çizgenin özdeş olduğu söylenemez. Bir çizgedeki döngü sayısı, bir düğümden çıkan kenar sayısı birer çizge değişmezidir ancak bir kenara ait etiket değişmez olarak değerlendirilmez. Verilen çizge değişmezlerinin özdeş değerleri her zaman çizgelerin eşbiçimini içermez ancak, özdeş çizgeler her zaman tüm çizge değişmezlerinin aynı değerlerine sahiptir. Dolayısıyla, çizge değişmezlerinin kullanımı tam özdeş altçizge eşleme ve sayma ile eşdeğer değildir. Bununla birlikte, çizge değişmezleri, altçizge eşbiçimciliği problemini çözmek için arama alanını azaltmak için kullanılabilir.

Çözüm yöntemleri (solution methods): Çizge tabanlı veri madenciliğinde temel konu, altçizge özdeşliklerini elde edecek en uygun yöntemi bulmaktır. Güncel çalışmalarda kullanılan beş arama yönteminden bahsetmek mümkündür: Açgözlü arama (greedy search) yaklaşımı, TMP tabanlı arama yaklaşımı, tümevaran veri tabanı (inductive database) yaklaşımı, çekirdek fonksiyon tabanlı (kernel function based) yaklaşım ve matematiksel çizge kuramı tabanlı (mathematical graph theory based ) yaklaşım. Bu yaklaşımlar, aramanın tamlığı açısından tam (complete) ya da

sezgisel (heuristic) olabilir. Altçizge eşbiçimciliği problemine yaklaşımları açısından ise, doğrudan eşleme (direct matching) ve dolaylı eşleme (indirect matching) gibi iki ayrı kategoriye ayrılır. Dolaylı eşleme altyapı eşbiçimciliği problemine çözüm sunmaz ancak bazı benzerlik ölçütleri altında, altçizge benzerlik problemine çözüm sunabilir.

Şekil 2.7 [89]'de çizge madenciliği alanında yapılan bazı güncel çalışmalara ait arama yöntemlerinin sınıflandırılması görülmektedir.




Şekil 2.7. Sınıflandırılmış çizge madenciliği algoritmaları

Çizge madenciliği yöntemlerinin kullanıldığı birçok çalışma yapılmıştır. Biyoenformatik alanında protein moleküllerinin çıkarımı [90], mühendislik alanında yöneylem araştırması [91], ağ aramaları [92,93], değişik amaçlı sosyal ağ analizi [94,95], anomali bulma [96], Galois lattice'in [97] kullanıldığı kavram öğrenme [98], çizgelerin birinci derece mantık ile ilişkisini inceleyen [99,100] çalışmaları bunlardan bazılarıdır. Tablo 2.2 [101] çizge madenciliğinin kullanıldığı popüler alanları göstermektedir.

Tablo 2.2. Popüler çizge madenciliği alanları

| |
|---|
| Frequent subgraph mining (Sık altçizge madenciliği) |
| Correlated graph pattern mining (Korelasyonlu çizge örüntü madenciliği) |
| Optimal graph pattern mining (Optimal çizge örüntü madenciliği) |
| Approximate graph pattern mining (Yaklaşık çizge örüntü madenciliği) |
| Graph pattern summarization (Çizge örüntü özetleme) |
| Graph classification (Çizge sınıflama) |
| Graph clustering (Çizge kümeleme) |
| Graph indexing (Çizge indeksleme) |
| Graph searching (Çizge arama) |
| Graph kernels (Çizge çekirdekleri) |
| Link mining (Bağlantı madenciliği) |
| Web structure mining (Ağ yapısı madenciliği) |
| Work-flow mining (İş-akış madenciliği) |
| Biological network mining (Biyolojik ağ madenciliği) |

Çizgeler, son yıllarda yapılan çalışmalarla kavram keşif problemine de önemli katkılar sunmuştur. Çizge tabanlı kavram keşif sistemleri eşyapı tabanlı yaklaşımlar ve yol bulma tabanlı yaklaşımlar olmak üzere iki temel sınıfa ayrılır. Eşyapı tabanlı sistemlerde kavramlar sık altçizgelerde aranırken; yol bulma tabanlı yaklaşımlarda sık bulunan yollar üzerinde aranır. Son zamanlarda yapılan bazı çalışmalar, bu iki temel yaklaşımın bazı yönlerini birlikte kullanarak melez yöntemler ortaya koymuştur.

### 2.5.1. Eşyapı tabanlı öğrenme

Bir çizge yapısındaki sık gözlenen altçizgeleri bulmak, çizge madenciliğinin temelini oluşturur. Buradaki amaç etkili ve verimli bir şekilde altçizgeleri bulmak ve bu altçizgelerin, temel çizge içindeki sıklığını belirlemektir.

Eşyapı tabanlı kavram keşfinde, temel çizge yapısındaki veriyi en iyi şekilde özetleyen altçizgelerin bulunması hedeflenir. Verinin özeti olan bu yapılar, hedef kavrama ait kavram tanımlayıcı olarak kullanılır. Bu yapılar, verinin basit formlarla ifade edilmesinde ve yorumlanmasında, kullanıcılara kolaylık sağlaması açısından önemlidir.

SUBDUE [23] ve SUBDUECL [24] bu alanda iyi bilinen eşyapı tabanlı yaklaşımlardandır. SUBDUE, çizge yapısındaki ilginç eşyapıları MDL [102] prensiplerini kullanarak bulur. Sistem, verinin içindeki yapısal kavramları ifade edecek şekilde orijinal verinin özetlenmesi esasına dayanır. SUBDUE, benzer fakat özdeş olmayan eşyapı örüntülerini tanımlayan ve iki eşyapı arasındaki yaklaşık yakınlık ölçüsünü bulan eksik çizge eşleme (inexact graph match) yaklaşımını kullanır. SUBDUE 'nun geliştirilmiş formu olan SUBDUECL, SUBDUE'nun kavram öğrenme sürecine negatif örnekleri dâhil etme problemine çözüm sunmaktadır. Pozitif ve negatif örnekleri ayırt etmede öğreticili öğrenme (supervised learning) yaklaşımını kullanır.

GBI [25] ve CLIP [27] bu alanda yapılan diğer çalışmalardır. CLIP, bir çizge yapısında sıklıkla tekrarlanan yapıların kavrama ait önemli örüntülere işaret ettiği fikrine dayanır. Çizge boyutunu azaltmada ışın arama (beam search) algoritması kullanır ve her iterasyonda bulduğu önemli örüntüleri tek bir düğüm ile ifade eder. Yeni kavramları bulmak için sonuç çıkarım sürecini analiz ederek tipik örüntüleri bulur. CLIP, çizgeleri eşleme aracı olarak çizge eşbiçimciliği (graph isomorphism) yerine çizge özdeşliğini (graph identity) kullanır. Çizge özdeşliği, çizgedeki alt örüntüleri ararken incelediği düğüme ait mümkün olan bütün yolları eşlemek yerine düğümden çıkan kenarlara ait sıralı bir düzen olduğu varsayımından hareket eder [103]. CLIP, eşyapıları özetleme işleminde harcanan zamanı tahmin ederek hesaplama karmaşıklığını giderir.

GBI, CLIP'in kavram çıkarım sürecini kullanır. Çizge yapısında, özyinelemeli bir şekilde komşu iki düğümü birleştirerek (pairwise chunking) tipik örüntüleri bulmak üzere tasarlanmıştır. Düğümleri birleştirme işlemi esnasında herhangi bir bilgi kaybına yol açmaz. GBI, kullandığı açgözlü arama algoritmasına dayalı olarak

hesaplama zamanını oldukça azaltır. Çizge yapısının sıklığına dayalı olarak çeşitli değerlendirme fonksiyonları kullanır.

GBI, birçok düğümün aynı etiketi paylaştığı çizge yapıları için uygun değildir ancak her bir düğümün farklı etiket taşıdığı çizgelerde etkili sonuçlar vermektedir [26].

### 2.5.2. Yol bulma tabanlı öğrenme

Herhangi bir çizge yapısında, bir kavramın bulunması durumunda bu kavramı ifade eden kavram kuralını oluşturan literaller arasında sık ve sonlu uzunluktaki benzer yollar bulunmaktadır. Bu benzer yolları en etkili ve verimli şekilde bulmak yol bulma tabanlı yaklaşımların temelini oluşturur.

Relational Pathfinding [7], bu alanda yapılan öncü bir çalışmadır. Birinci derece mantık kullanılan yöntem, temelde TMP tabanlı yaklaşımların karşılaştığı yerel plato ve yerel maksimum problemlerine çizgelerin çözüm olabileceği fikrinden hareketle ortaya çıkmıştır. Relational pathfinding, yolları, incelediği pozitif örneğe bağlı sabitlerin oluşturduğu ardışık düğümleri genişleterek bulur. Sistem hedef kavramı bulmada oldukça kapsamlı bir algoritma kullanır ancak bunun pratikte iki önemli kazancı bulunur. Birincisi, bütün düğümleri eşzamanlı aradığı için hedef kavrama ulaştıracak yolların kesişimine ulaşmada geçilen toplam yol sayısını azaltır. İkincisi ise, çoğu anlamlı ilişkiyi tanımlamak için kısa yollar yeterli olduğu için arama derinliği kolayca sınırlandırılabilir.

MDPF [8], birinci derece kural öğrenme sistemlerine katkı sağlamak amacıyla, mod deklarasyon ile genişletilmiş pathfinding algoritmasını kullanarak çok ilişkili verilerde, nesneler arasındaki bağlantıları bulur. Bu bağlantılar arama alanında kavram kuralını ifade edecek yolları bulmak için kullanılır. MDPF, diğer yol bulma tabanlı algoritmalardan farklı olarak aşağıdan yukarıya bir TMP tabanlı kavram keşif sistemi olan doymuş alt cümecikleri kullanır. MDPF'de veri hiperçizgeler ile ifade edilir. Bu çizgelerdeki hiperyollar (hyperpaths) bulunacak hipotezlere tekabül eder. Mod deklarasyon kullanımı giriş/çıkış yüklem değişkenlerini belirlemede, doymuş alt cümlecikler ise arkaplân bilgisinde tutulan verilerin doğrudan kullanılmasıyla arama alanındaki uçların bağlanmasında kolaylık sağlar.

RPBL [21], yerel plato ve yerel maksimum problemlerine karşı pozitif örnekleri açıklamada ve hipotezleri elde etmede ilişkisel yolları kullanan eşzamanlı bir kapsama yaklaşımıdır. RPBL, hipotez alanında kavaram tanımlayıcılarını ifade eden yolları ve hipotezin performansını F1-ölçümünü (F1-measure) kullanarak bulur. RPBL'nin geliştirilmiş formunda [20], özyinelemeli kavram tanımlayıcılarını bulmak için hedef örnekler aynı zamanda arkaplân bilgisi olarak kullanılmaktadır. Alan teorilerini de (domain theories) öğrenme sürecine katmak için çizge yapısı alan teorisine uygun hale getirilmektedir.

Yol bulma tabanlı kavram keşif yaklaşımlarından [22], diğer yol bulma tabanlı yaklaşımlardan farklı olarak n-argümanlı (n-ary, n>2) ilişkileri ifade etmeye olanak tanır. Sistem hedef örnekler ve arkaplân bilgilerini tutan veri tabanındaki bilgileri bir çizge veritabanına aktararak çalışmaya başlar. Hedef örneklerden çıkan bütün yollar bulunur ve sık karşılaşılan yollar kavram tanımlayıcı olarak belirlenir. Zayıf ve sık karşılaşılmayan kavram tanımlayıcılar, başlangıç aşamasında eleme işleminden geçirildiği için çalışma zamanını iyileştirir ve daha yüksek doğruluk sağlar.

### 2.5.3. Melez öğrenme

Melez öğrenme yaklaşımında, hem yol bulma tabanlı hem de eşyapı tabanlı çizge madenciliği yaklaşımlarının bazı yönleri birlikte kullanılır.

H-GCD [28], hem eşyapı tabanlı hem de yol bulma tabanlı yaklaşımların tekniklerini bir arada kullanır. Eşyapı tabanlı yaklaşımlara benzer şekilde birbirine benzeyen düğümleri tek bir düğüm olarak gösterir. Yol bulma tabanlı yaklaşımlara benzer şekilde hedef örneklere ait argümanları bağlayan yolları kavram tanımlayıcı olarak ifade eder. Yöntem eşyapı tabanlı yaklaşımlardan farklı olarak benzer düğümleri gruplamak için çizge eşbiçimciliğini kullanmaz, bunun yerine özetlenmiş bir çizge yapısı oluşturur ve giriş verileri üzerinde doğrudan SQL sorgularını çalıştırır. Yol bulma tabanlı yaklaşımlardan farklı olarak, mevcut çizge üzerinde yol araması yapmak yerine, yol çıkarımını çizgeyi oluştururken yapar. Yöntem sadece ikili (binary) ilişkilere izin verir.

G-CDS [29], hem eşyapı tabanlı hem de yol bulma tabanlı yaklaşımların kombinasyonu olan melez bir yaklaşımdır. Hedef örnekler ve arkaplân bilgilerini

ilişkisel veri tabanında tutar. Her hedef örnek için ayrı bir çizge oluşturur ve bunlardan özet bir çizge çıkarır. Eşyapı tabanlı yaklaşımlara benzer şekilde birbirine benzeyen düğümleri sınıflar; kavram tanımlayıcılarını, yol bulma tabanlı yaklaşımların tekniklerini kullanarak özet çizge üzerinden çıkarır. Hem [28] hem [29]'da kural çıkarımı için maksimum kural uzunluk parametresi kullanılır. Değerlendirme işlemi ise destek ve güven değerlerine göre yapılır. Her iki yöntem, kavram tanımlayıcılarını doğru bir şekilde bulmada ve çalışma zamanını iyileştirmede olumlu sonuçlar elde etmiştir.

# 3. ÇİZGE TABANLI KAVRAM KEŞFİ

Çizge tabanlı kavram keşif sistemleri, veri gösterimi, arama teknikleri, kavram değerlendirme ölçütleri, kullandıkları algoritmalar gibi birçok açıdan birbirinden farklılık gösterir. Bu kısımda, çizge tabanlı kavram keşfinde öne çıkan çalışmalar detaylı bir şekilde incelenecektir.

## 3.1. Veri Gösterimleri

Relational Pathfinding [7] yaklaşımında, veriler yönsüz ve etiketli çizgelerle gösterilir. Düğümler gerçekleri turar. Kenarlar birbiriyle ilişkili gerçekleri oluşturan düğümleri bağlarken, ilişki adını taşıyacak şekilde etiketlenir. Yaklaşımda arkaplân gerçekleri ikili (binary) ya da tekli (unary) olarak gösterilirken; hedef kavram, çoklu (n-ary) ilişkilere izin verecek şekilde tasarlanmıştır.

RPBL [21]'de veriler, yönsüz ve etiketli çizgelerle gösterilir. Düğümler ikili gerçekleri tutarken, kenarlar ortak argümanlara sahip düğümleri birbirine bağlar. RPBL, kavram çıkarım sürecinde verileri ifade etmede aşağıda belirtilen beş yapıdan faydalanır:

- Structured Item Space (SIS): Arkaplân bilgilerinin tutulduğu etiketli bir çizgedir.
- Expanded SIS (E-SIS): SIS'in alan teorileri uygulanarak genişletilmiş formudur.
- Relational Path (RP): SIS içindeki pozitif hedef örnekelere ait argümanları bağlayan herhangi bir yoldur.
- Specialized RR (S-RP): RP'nin üzerindeki ikili gerçeklere ait argümanlara birli gerçeklerin eklenmesiyle düzenlenmiş formudur.
- Generalized S-RP (G-S-RP): S-RP üzerindeki gerçeklerin, birli ve ikili sabit argümanlarını değişken hale getirerek elde edilen, S-RP'nin en genel birinci dereceden Horn cümleciğidir.

MDPF [8], doymuş alt cümlecikler kullanan çizge tabanlı bir kavram keşif sistemidir. Çizge yerine; düğümlerden ve boş olmayan bir dizi düğümü tek bir hedef düğüme bağlayan hiperyaylardan (hyperarcs) oluşan yönlü ve etiketli hiperçizgeleri

kullanır. Doymuş alt cümleciklerin gövdesindeki her bir literal hiperçizgede bir düğüm olarak gösterilir. Literallerin argümanlarını bağlayan kenarlar giriş/çıkış mod deklarasyon kullanılarak oluşturulur. Bir hiperyay aşağıdaki şekilde tanımlanabilir [104];

D bir dizi düğümü, Y bir dizi hiperyayı göstermek üzere; yönlü bir H hiperçizgesi (D,Y) ikilisinden oluşur. Her bir hiperyay ise (K, h) sıralı ikilisinden oluşur. Burada K, $K \subseteq D$ olan kaynak düğümleri; h ise, $h \in D$ olan tek bir hedef düğümü gösterir.

Doymuş bir cümlecik bir hiperçizgenin içine iki adımda aktarılır. İlk olarak, alt cümlecik bir dizi literalden oluştuğu için her bir $L_i$ literali hiperçizgede bir düğüm olarak ifade edilir. İkinci olarak ise, bu $L_i$ düğümü girdi olarak diğer literal ve düğümlerden oluşan bir dizi argüman alır. $L_i$ düğümü $G_i$ gibi bir giriş değişkenine sahipse her bir hiperyay, $L_i$ ve $G_i$ 'nin oluşturduğu bir dizi literal ile gösterilir.

SUBDUE [23] ve SUBDUECL [24] veri gösteriminde aynı yöntemi kullanır. Bu yöntemde yönlü, etiketli ve iki düğümün birden çok kenar ile birbirine bağlanabildiği çoklu çizge (multi graph) kullanılır. Yöntemde düğümler argümanları tutarken, kenarlar bu argümanları bağlar ve ilişki adından sonra etiketlenir. Çizgesel gösterimde bir eşyapı, giriş çizgesine ait bir dizi düğüm ve kenardan oluşan bağlı bir altçizge olarak ifade edilir.

GBI [25] ve CLIP [27] renkli ve yönlü çizge kullanır. Her düğüm topolojik bilgi tutan bir kenar ve bir ya da daha fazla renk tutar. Renkler ilgili düğülere ait gerçeklerin özellik değerlerini ifade etmek için kullanılır. CLIP'teki düğümlerde iki renk kullanılır. Bunların ilki, verilerin değerini hesaplamak için kullanılan çıkarım kuralının tanımlayıcısına karşılık gelirken; ikincisi, değerin kendisine karşılık gelir. Birbirine bağlı iki düğüm incelenir aynı tipik yapıya sahip olanlar birleştirilir.

H-GCD [28], yönlü ve etiketli çizge kullanır. Düğümler arkaplân yapısında tutulan ilişkilerin argümanlarına ait bir dizi sabiti tutar. Kenarlar ise birbiriyle ilişkili bu düğüm çiftlerini bağlar ve arkaplânda tutulan ilişki adıyla etiketlenir. Yöntem üç tür düğüm kullanır: kaynak (source) düğüm, hedef (target) düğüm ve aracı (intermediate) düğümler. Kaynak düğüm, hedef örneklerin ilk argümanını tutarken; hedef düğüm ikinci argümanı tutar.

G-CDS [29], yönlü ve etiketli çizge kullanır. Düğümler hedef ve arkaplân ilişkilerini tutarken, kenarlar en az bir ortak argümana sahip düğümleri birbirine bağlar. Yöntem veriyi ilişkisel format içinde gösterir ve her bir hedef örnek için ayrı bir çizge oluşturur.

### 3.2. Arama Teknikleri

Çizge tabanlı kavram keşfinde, temel olarak açgözlü arama yaklaşımının (greedy search approach) sınıflandırılmış biçimi olan Enine Arama (BFS) ve Derinine Arama (DFS) yöntemleri kullanılır. Işın Arama (beam search) ve Derinlik Sınırlı Arama (depth-limited search-DLS) bu iki yöntemin biraz daha özelleşmiş halidir.

Açgözlü aramada (greedy search) her zaman mevcut durumdaki en iyi ve sonuca en yakın seçim göz önünde bulundurulur. Lokal olarak en uygun olan çözümün global bazda da en uygun olduğu varsayımından hareket eder. Algoritma geniş bir problem sahası için oldukça güçlü ve etkili bir yöntemdir. Çoğu problem için optimal çözümler sunsa da bu, bütün problemler için genelleştirilemez. Açgözlü yaklaşımın temellerini kullanan Enine Arama (breadth-first search-BFS) ve Derinine Arama (depth-first search-DFS) algoritmaları aşağıdaki gibi ifade edilebilir [105].

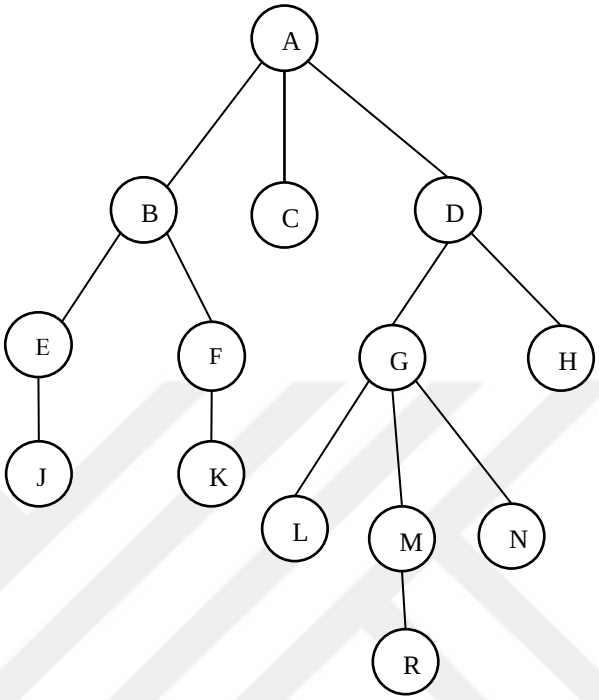
Enine Arama (breadth-first search-BFS): BFS, en basit çizge arama algoritmalarından biridir ve bu alanda diğer birçok algoritmaya da model teşkil etmiştir.

G = (V,E) gibi bir G çizgesi ve r gibi bir kök (root) düğüm verilmiş olsun. BFS, r'den ulaşılabilecek bütün düğümleri bulmak için, G'nin kenarlarını sistematik bir düzende kontrol eder ve her bir düğüm için en kısa mesafeyi hesaplar.

BFS algoritması, kök düğümden başlayarak ulaşabileceği bütün düğümleri içeren bir enine arama ağacı (breadth-first tree) oluşturur. G çizgesi içinde r'den ulaşabileceği herhangi bir d düğümü için bu ağaç üzerinde en az kenara sahip olan yol, r'den d'ye en kısa yol olarak ifade edilir. Algoritma hem yönlü hem de yönsüz çizgeler için kullanılabilir.

BFS algoritması adından da anlaşılabileceği gibi çizgede gezinti yaparken, bulunduğu seviyedeki mevcut bütün düğümlerin sınırına ulaşıncaya kadar aramayı

genişletir. Öncelikle seçilen bir kök düğümden aramayı başlatır ve k+1. seviyedeki düğümlere geçmeden önce k. seviyedeki bütün düğümleri ziyaret eder. Şekil 3.1'de yönsüz çizgede basit bir BFS uygulaması görülmektedir.




Şekil 3.1. Basit bir BFS uygulaması

BFS algoritması yukarıdaki şekilde A düğümünden aramaya başladığında, bir sonraki derinlikte önce B düğümünü, daha sonra sırasıyla aynı seviyedeki diğer düğümleri ziyaret eder. Arama işlemi bütün çizge için gerçekleştiğinde ziyaret edilen düğümler sırasıyla aşağıdaki gibidir.

A B C D E F G H J K L M N R

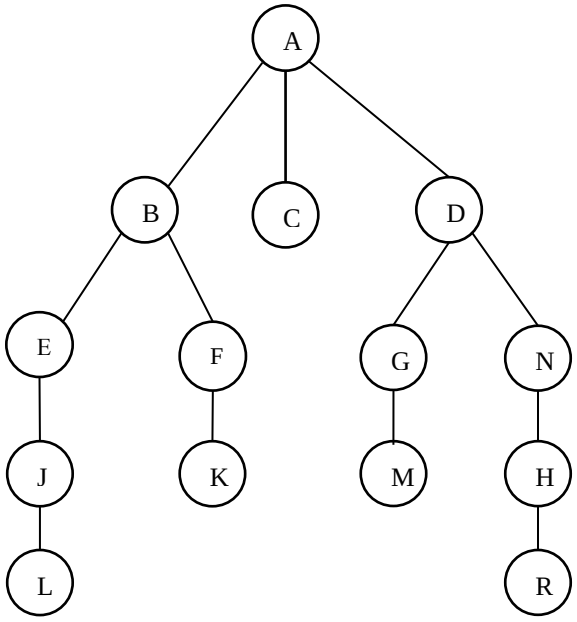
BFS 'de arama işlemi hedef düğüme ulaşılıncaya, ya da bütün çizge gezilinceye kadar devam eder. Algoritma tam ve optimal bir çözümü garanti eder. Bir çözümün bulunması durumunda bunun bulunması tamlık olarak ifade edilir. Optimal çözüm ise, birden fazla çözümün bulunması durumunda, algoritmanın en uygun olanına göre seçim yapmasıdır. Algoritmanın en büyük dezavantajı, büyük veri tabanlarında fazla bellek gereksinimi olarak görülebilir.

BFS, kuyruk (queue) veri yapısını kullanır. G = (V, E) çizgesinde V, düğümlerin sayısı; E, kenarların sayısı olmak üzere;

Zaman karmaşıklığı, en kötü durum için O (V + E) 'dir. O (E), giriş çizgesinin sık ya da seyrek oluşuna göre O (1) ve O ($V^2$) arasında değişir. Bellek karmaşıklığı ise komşuluk listesi (adjacency list) kullanıldığında O (V + E), komşuluk matrisi (adjacency matrix) kullanıldığında ise O ( $V^2$ ) 'dir.

Işın Arama algoritması, gezinti yaptığı çizgede ağaç yapısını BFS stratejisine göre oluşturur. Algoritma hafıza gereksinimini azaltacak şekilde En İyiyi Önce Ara (best-first search) algoritmasını iyileştirir. En İyiyi Önce Arama algoritması kısmî bütün çözümleri hesaba katarken, Işın Arama algoritması, sezgisel olarak belirlediği ve sonuca en yakın belirli sayıdaki kısmî çözümü hesaba katar.

Derinine Arama (depth-first search-DFS): DFS algoritması, bir çizgede arama yaparken mümkün olduğunca derine inme esasına dayanır. Çizge içinde r gibi bir kök düğümden aramaya başlar ve ziyaret ettiği her bir düğümün komşusunu ziyaret ederek aramayı derinlemesine sürdürür. Arama işlemi kök olarak seçilen düğümden herhangi bir dallanma kalmayıncaya kadar devam eder. Arama bittiğinde hala ziyaret edilmeyen düğümler kaldıysa, bunlar arasından yeni bir kök düğüm seçer ve aramaya yeniden başlar. Şekil 3.2'de basit bir DFS uygulaması görülmektedir.




Şekil 3.2. Basit bir DFS uygulaması

Şekil 3.2'de, A düğümünden başlayıp, sol alt ağaçtan devam eden bir DFS araması aşağıdaki gibi sonuçlanır.

A B E J L F K C D G M N H R

DFS, arama esnasında birden fazla kök düğüm kullandığı için oluşturduğu ağaç yapısı da birden fazla olur. Bu ağaç yapılarının birleşiminden derinine arama ormanı ( depth-first forest) oluşur. Bu sebeple DFS çizge yapısı, BFS 'den biraz farklı ifade edilir. V düğümleri, $E_\pi$ 'deki kenarlar ağaç yapılarındaki kenarları ifade etmek üzere; bir DFS çizgesi aşağıdaki şekilde tanımlanır. Buradaki $\pi$ değeri incelenen düğümün ziyaret edilip edilmediğine dair bilgileri tutar.

$G_\pi = (V, E_\pi)$

$E_\pi = \{ ( v.\pi, v) \; v \in V \text{ ve } v.\pi \neq NIL \text{ (boş küme)} \}$

DFS yığıt (stack) veri yapısını kullanır. G = ( V, E) çizgesinde V, düğümleri; E, kenarları göstermek üzere;

Zaman karmaşıklığı en kötü durum için O (V+E) 'dir. Bellek karmaşıklığı ise en kötü durum için O (V) olarak gösterilir. Buradan hareketle DFS 'nin bellek kullanımı açısından avantajlı olduğu söylenebilir. Ancak döngü içeren bir çizgede algoritmanın sonsuz döngüye girip optimal bir çözüm bulamaması, tahmin edilemeyen bir zaman karmaşıklığına yol açar.

DFS algoritmasının bu dezavantajı, arama alanında derinlik için bir sınır belirlenerek aşılabilir. Derinlik Sınırlı Arama bunun için kullanılan bir algoritmadır. Temel olarak DFS 'nin arama stratejisini kullanır. DFS 'den farkı, inilecek derinlik için tamsayı türünden bir sınır belirlenmesidir.

### 3.3. Kavram Tanımlayıcılarını Değerlendirme Yöntemleri

Çizge tabanlı kavram keşfinde, veri madenciliğinde olduğu gibi çeşitli değerlendirme yöntemleri kullanılır. Bunlardan iyi bilinen destek ve güven değerlerini [22,28,29,106] sistemleri kullanır. F1-ölçümünü kullanan [21,8], MDL 'yi kullanan [23] ve diğer bazı veri sıkıştırma yöntemlerini kullanan [25,107] gibi sistemler de mevcuttur.

Kavram keşfinde bir kavram kuralını değerlendirirken, kapsam (coverage) ve doğruluk (accuracy) kriterleri ön plâna çıkar. Kuralın içerdiği kavram örneği sayısı

kapsam olarak ifade edilir. Kural örüntüsünün doğru bir şekilde açıkladığı varlık sayısı ise doğruluk olarak ifade edilir. Kapsam destek (support) değerine; doğruluk ise güven (confidence) değerine tekabül eder.

Destek (support): Kurala uyan pozitif hedef örneklerin sayısının, hedef örneklerin toplam sayısına oranıdır. K gibi bir kavram kuralı, h⟵ b ile ifade edilsin. Destek h⟵ b aşağıdaki şekilde hesaplanır.

$$\text{destek}(h \longleftarrow b) = \frac{|\ h \longleftarrow b' \text{yi sağlayan h örnekleri}\ |}{|\ h' \text{yi sağlayan h örnekleri}\ |} \tag{3.1}$$

Güven (confidence): Kurala uyan pozitif hedef örneklerin sayısının, kural dahilinde gövdedeki ilişkilerden çıkarılabilecek örneklerin sayısına oranıdır. K gibi bir kavram kuralı, h⟵ b ile ifade edilsin. Güven h⟵ b aşağıdaki şekilde hesaplanır.

$$\text{güven}(h \longleftarrow b) = \frac{|h \longleftarrow b' \text{yi sağlayan h örnekleri}|}{|h' \text{yi sağlayan h örnekleri}|} \tag{3.2}$$

$F_1$-Ölçümü ($F_1$-measure): $F_1$-ölçümü, bilgi getirim (information retrieval) [108] sürecinin ölçümleri olan kesinlik (precision) ve duyarlılık (recall) değerlerinin harmonik ortalamasıdır. Kesinlik değeri, çıkarıma konu olan bilgi yapısından getirilen bilginin, ne oranda istenen bilgiye ait olduğunu ifade eder. Yani getirilen bilgideki pozitif sonuçların getirilen bütün bilgiye oranıdır. Duyarlılık değeri ise, istenen bilginin ne kadarının getirildiğini ifade eder. Yani getirilen bilgideki pozitif sonuçların, getirilmesi gereken pozitif sonuçlara oranıdır [109].

Genellikle kesinlik değeri p, duyarlılık değeri r ile gösterilir. Doğru pozitif (true positive) sonuçlar tp, yanlış pozitif (false positive) sonuçlar fp ile ifade edilmek üzere; kesinlik değeri p, aşağıdaki formülle hesaplanır.

$$p = \frac{tp}{tp+fp} \tag{3.3}$$

Doğru pozitif (true positive) sonuçlar tp, yanlış negatif (false negative) sonuçlar fn ile gösterilmek üzere; duyarlılık değeri r aşağıdaki formülle hesaplanır.

$$r = \frac{tp}{tp+fn} \tag{3.4}$$

$F_1$-ölçümü ise, p ve r değerlerinin harmonik ortalaması alınarak hesaplanır.

$$F_1=2\frac{p.r}{p+r} \quad (3.5)$$

MDL: MDL 'nin temel prensibi, bir veride düzenli bir yapı mevcutsa bu düzenli yapının, veriyi sıkıştırmak (data compress) suretiyle daha az sembol kullanılarak ifade edilebilme olanağına dayanır. Verideki düzen ne kadar fazlaysa veriyi sıkıştırma olanağı da o ölçüde fazla olur. Örneğin 0 ve 1'lerden oluşan 10.000 bit'lik üç veri dizisi aşağıdaki gibi verilmiş olsun [110].

(1) 00010001000100010001000.....00010001000100010001000100010001

(2) 01110100110100001010101O.....10101110101110110001011000010

(3) 11100111111010011011111.....01111011111011111100111101111

(1)'deki dizi 2500 tane 0001'in tekrarından oluşur. Bu dizinin yapısında belirli bir düzen bulunduğu için, diziden basit bir kural çıkarmak mümkündür. Çıkarılan bu kural sonraki veriler için de kullanılabilir. (2)'deki dizi bir paranın yazı-tura deneyi olsun. Buradaki veri rassal olarak oluştuğu için, belirli bir kuraldan söz etmek mümkün olmaz. (3)'deki dizide 1'lerin sayısı 0'ların sayısının tam dört katı olsun. (2) 'nin aksine buradaki veride bir kısım düzenli yapıdan bahsedilebilir; ancak bu düzen, belirli (deterministic) olmaktan çok istatiksel bir düzendir. Yine de mevcut düzenli yapılar dikkate alınıp, sonraki veriler için tahminde bulunmak mümkündür.

MDL'nin bakış açısıyla V verisinin düzenli yapıları baz alınarak kodlanmış bir V' verisi ile, her zaman V verisi yeniden oluşturulabilir. Herhangi bir programlama dilinde (1) dizisi, düzenli yapıları kullanılarak daha açık ve kısa bir biçimde yazılabilir. Örneğin C++ dilinde Şekil 3.3'teki gibi yazmak mümkündür.

```
for (int i = 0; i < 2500; i++)
    cout << "0001";
```

Şekil 3.3. (1) dizisi C++ kodu

Ancak (2) dizisindeki veriyi herhangi bir yöntemle kısaltmak mümkün olmadığı için, bu dizinin en kısa şekli yine kendisidir.

### 3.4. Algoritmalar

Relational Pathfinding [7]'de her bir hedef örnek için bir çizge oluşturulur. Bu çizgeler başlangıçta tek bir düğüme sahiptir. Sistem her bir çizge için sabitlerden (constant) oluşan bir son-değer (end-value) tutar. Hedef örneğe bağlı yolları bulmak için son-değer içindeki sabitler olası bütün yollar dikkate alınarak genişletilir. Bu genişleme bir kesişime (intersection) ya da sınıra ulaşıncaya kadar devam eder. Herhangi bir kesişim durumunda bu yol ilişkisi orijinal kurala eklenir. Bu yeni kural dâhilinde açılacak düğüm kalmadığında, bu genişletilmiş kural nihaî yol olarak sabitlenir. Birden fazla nihaî yolun bulunması durumunda doğruluğu en yüksek olan yol, kavram tanımlayıcı olarak seçilir.

RPBL [21], hedef ilişkiye ait yolları bulmak için başlangıçta boş olan $n_0$ adlı bir düğüm oluşturur ve bunu hedef düğümlere göre sınırlar. $n_0$'ı başlangıç kabul eden bir Tr ağacı oluşturarak bunu Open adlı bir listeye ekler. Closed ve Argument adında başlangıçta boş olan iki liste daha oluşturur. Open'dan ilk düğümü seçer, bu düğümü listeden kaldırır, Closed listesine ekler ve bunu n olarak adlandırır. n düğümü, hedef düğüm ise; Tr ağacında n ile $n_0$ arasındaki yolu çözüm kümesine ekler. n hedef düğüm değilse, Argument listesinde olmayan argümanları n düğümüne göre genişletir. n 'nin çocuk düğümlerinden oluşan M dizisini oluşturur. M'deki elemanları Open listesine ekler ve ilişkisel yolları (relational paths-RPs) döndürür.

MDPF [8]'de, baş literal n tane ayrı düğümden oluşur. Buradaki n, baş literalin sahip olduğu değişken sayısını ifade eder. Yollar bu kaynak düğümlerden başlanarak oluşturulur. Kaynak düğümlerin derinliği, başlangıçta 0 olarak alınır ve eklenen yeni düğümün giriş değişkenine göre, inilebilecek maksimum derinliğe kadar genişleme devam eder. Her bir kaynak düğüm için, mevcut derinlikte değişkenlerin yollarında kesişim olup olmadığı kontrol edilir. Kesişim varsa ilgili yollar birleştirilir ve kavram tanımlayıcıları buna göre bulunur.

SUBDUE [23] algortiması, belirli bir eşyapıyı çizgedeki tek bir düğüme eşleyerek başlar. Her bir iterasyonda en iyi eşyapı seçilir ve arama aşamalı olarak genişletilir. Algoritma en iyi eşyapıyı bulmak için, mevcut bütün eşyapıları dolaşana ya da belirlenen bir hesaplama sınırına ulaşana kadar aramayı sürdürür. Bulunan en iyi altçizge, giriş verisini sıkıştırmada kullanılır. Sıkıştırılan bu yapı, sonraki aşamada

giriş verisi olarak kullanılır. Algoritma, giriş verisinin tanımını buluncaya kadar işleme devam eder.

SUBDUECL [24] algoritması başlangıçta bütün eşyapıları tuttuğu ParentList adlı bir liste oluşturur. Bu listedeki eşyapılar bir düğüm ya da bir bir kenara ait olası bütün yollar kullanılarak genişletilir ve oluşan bu yeni eşyapılar ChildList adlı bir listede tutulur. Bu yapılar içinde, sınırlandırılmış ışın boyutu (beam size) altında - SUBDUECL beam search kullanır- en az bir pozitif hedef örneğe sahip olanlar BestList adlı listeye eklenir. Bir genişlemede herhangi bir pozitif örnek bulunamazsa ışın boyutu arttırılarak aramaya devem edilir.

GBI [25], giriş çizgesinde özdeş olan örüntüleri birleştirerek tek bir düğüm halinde yazar ve yeni bir etiket atar. Bu daraltılan çizge yapısında, bağlı iki düğümden oluşan düğüm çiftlerini çıkartır ve en tipik yapıya sahip olanları, birleştirilecek düğümler olarak seçer. Kavram tanımlayıcılar bu birleştirme işlemi sonucunda bulunur.

CLIP [27], üye örüntüleri tutmak için view ve pattern adlı iki dizi yapısı oluşturur. view, tek bir örüntüyü; pattern ise bir dizi örüntüyü tutar. Algoritma üç aşamadan oluşur: pattern modification, pattern combination, view selection. Pattern modification 'da ilk olarak bir view seçilir ve giriş çizgesi, seçilen bu view içindeki örüntülere göre yeniden oluşturulur. Her bir örüntü çizgede bir düğüm olarak gösterilir. Daha sonra bağlı diğer pattern 'ler view içine eklenir. Pattern combination 'da, güncellenmiş view 'lerdeki örüntüler olası bütün durumlar dikkate alınarak yeni bir view oluşturmak üzere birleştirilir. View selection aşamasında ise, her bir view içinden, çizgeyi en iyi özetleyen örüntüler seçilir. Kavram tanımlayıcılar bu örüntüler kullanılarak bulunur.

H-GCD [28] algoritması yedi aşamadan oluşur. Başlangıç (initialize) aşamasında kaynak ve hedef düğümlerinden oluşan iki düğümlü, bağlı olmayan bir çizge oluşturulur. Genişleme (expansion) adımında, ilgili gerçekleri (related facts) içeren düğümler çizgeye eklenir. Birleştirme (merge) adımında, aynı içeriğe sahip düğümler tek bir düğüm olarak gösterilir ve kenarlar buna göre yeniden düzenlenir. Değerlendirme ve budama (evaluation and pruning) adımında, mevcut kavram tanımlayıcıların destek ve güven değerleri SQL sorguları kullanılarak hesaplanır. Kesişim kontrol (check for intersection) adımında, kaynak ve hedef düğümler

arasında kesişim olup olmadığı kontrol edilir. Kesişim varsa, kesişen düğümlerin tuttuğu yol değerleri birleştirilerek, kaynak ve hedef düğümler arasındaki yol bulunur. Yol değişkenlerini güncelleme (update path variables) adımında, kavram tanımlayıcıların değişkenleri, içerdikleri sabitlerle tutarlı olacak -tekrarlı yapı yoldan çıkarılır- şekilde güncellenir. Kapsama (covering) adımında, hedef örneklerin çözüm cümleciği tarafından açıklanıp açıklanmadığına bakılır. Kapsama yeterliyse süreç sonlanır, değilse ilk adıma giderek yeniden başlar.

G-CDS [29] algoritması, ilk olarak her bir hedef ilişki için ayrı bir çizge oluşturur. Oluşturulan bu ayrı çizgeler karşılaştırılarak özet bir çizge çıkarılır. Bunun için her bir ayrı çizge üzerinde ataları aynı olan düğümler özet çizgede tek bir düğüm olarak gösterilir. Kural çıkarımı için, çizgedeki yollar derinine arama algoritmasıyla maksimum kural uzunluğu parametresine göre dolaşılır. Değerlendirme ve budama aşamasında, kural için çıkarılan yollar SQL sorgularına dönüştürülür ve kuralın destek ve güven değerleri hesaplanır. Değeri eşik değerinin üzerinde olan kural, çözüm kümesine eklenir. Son adımda, hedef örneklerin ne kadarının kapsandığına bakılır. Eğer henüz kapsanmamış hedef örneklerin sayısı, minimum_destek x hedef örnekler değerinden düşükse kavram çıkarım süreci sonlanır, değilse özet çizge olası çözümler için yeniden dolaşılır.

### 3.5. Özyineleme Kuralları

Özyinelemeli kavram tanımlayıcılar, hedef ilişkinin kavram kuralının gövdesinde yer aldığı kavram tanımlayıcılardır. Örneğin;

$e(A,B) \leftarrow e(B,C), e(C,A)$

cümleciği özyinelemeli bir kavram tanımlayıcıdır. Çünkü hedef kavram olan e (elti), aynı zamanda kuralın gövde kısmında da yer almaktadır.

Relational pathfinding [7] , eğitim verisini sadece pozitif örnekleri içerecek şekilde ayarlayıp, çözüm cümlelerine ait yüklemin genişletilmiş tanımı olarak pozitif örnekleri kullanmaktadır.

RPBL'nin geliştirilmiş formu [20] özyinelemeli kavram tanımlayıcıları bulabilmek için, hedef ilişkileri aynı zamanda arkaplân verisi olarak kullanmaktadır.

G-CDS yöntemi özyinelemeli kavram tanımlayıcılarını bulmak için same generation veri setini kullanmaktadır. Bu veri seti 344 tane aynı nesilden (same generation) olan gerçek aile bireylerini pozitif örnek olarak içermektedir. Ayrıca 64 tane arkaplân gerçeği, ailedeki ebeveyn (parental) ilişkisini oluşturmaktadır. G-CDS yönteminin bulduğu özyinelemeli kavram tanımlayıcılar aşağıda sıralanmaktadır.

$sg(A, B) \longleftarrow p(C, A), sg(C, D), p(D, B)$

$sg(A, B) \longleftarrow p(C, A), sg(C, D), p(D, A)$

$sg(A, B) \longleftarrow p(C, A), p(C, B)$

# 4. ÖNERİLEN YÖNTEM

Bu bölümde HOC sınıfına ait öğrenme problemleri için önerilen, yol bulma tabanlı bir kavram keşif yöntemi anlatılmaktadır. Yöntemde her bir hedef örnek için ayrı bir çizge oluşturulmakta, kavram tanımlayıcılar çizge üzerindeki yollar üzerinden çıkarılmaktadır. Destek ve güven değerleri, çizge üzerindeki yollar sayılarak hesaplanmaktadır. Yöntemin temel özellikleri aşağıdaki gibi listelenebilir:

- olası kavram tanımlayıcıları çizgeyi oluştururken çıkarma
- destek ve güven değerlerini çizgelere uygun şekilde tanımlama
- çoklu (n-arty) ilişkileri kullanabilme
- özyinelemeli kavram tanımlayıcıları çıkarma
- veriyi tek bir çizge yerine birden fazla çizgeyle gösterme
- bütün hedef örnekleri aynı anda dikkate alarak, hedef örneklerin diziliminin bulunacak kavram tanımlayıcıları etkilememesini sağlama

## 4.1. Head Output Connected (HOC) Sınıfı

$C = h \leftarrow b_1, b_2, .., b_n$ gibi bir belirli cümleciğin, HOC sınıfına dâhil olması için aşağıdaki şartları sağlaması gerekir [6];

Giriş/Çıkış (G/Ç) Tutarlılığı: C cümleciğinde, her bir gövde atomu $b_i$ için giriş değişkenleri; ya baş (head) giriş değişkenlerinin alt kümesi olmalı, ya da daha önceki gövde atomu $b_j$ içinde mevcut olmalıdır. $1 \leq j < i$ için C cümleciği, G/Ç tutarlılığına sahip olur.

Örneğin mod deklarasyonu, c(+int), a(+int,-int), b(+int, -int) şeklinde verilen bir C cümleciği için;

$c(X) \leftarrow a(X,Y), b(Y,Z)$

cümleciği G/Ç tutarlıdır. Ancak;

$c(X) \leftarrow a(X,Y), b(Z,Y)$

cümleciği G/Ç tutarlı değildir. Çünkü Z, b/2 için giriş değişkeni olmasına rağmen baş ilişki giriş değişkenlerine bağlantısı yoktur.

HOC: G/Ç tutarlılığına sahip ve bütün baş ilişki çıkış değişkenleri cümleciğin gövdesinde örneklenmiş bir C cümleciği, bir HOC cümledir. Diğer bir deyişle,

cümleciğin baş ilişki giriş değişkenleri ile baş ilişki çıkış değişkenlerini bağlayan literaller arasında zincirleme bir bağlantı mevcuttur. Örneğin;

c(X,Y) ← a(X,Z), b(Z,Y)

olarak verilen bir C cümleciği, c(+int,-int), a(+int,-int), b(+int,-int) şeklindeki mod deklarasyonu için, bir HOC cümledir. Çünkü X değişkenini Y değişkenine bağlayan literaller arasında, zincirleme bir bağlantı vardır.

### 4.2. Veri Seti ve Gösterimi

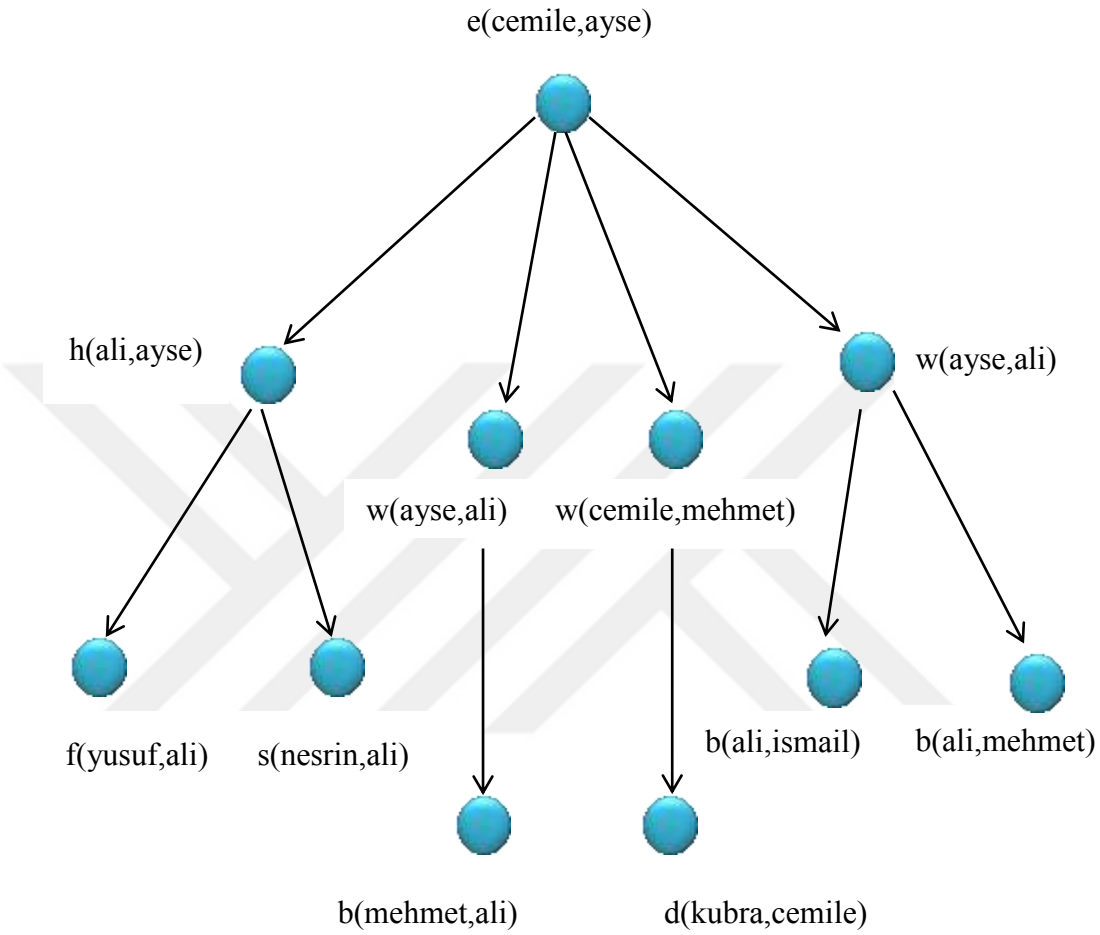
Tablo 4.1'de verilen elti veri setine ait örnekte e, elti'yi ifade etmektedir. Önerilen yöntemde elti, öğrenilecek hedef kavramı ifade eder. Tamamı pozitif kavram örneğinden oluşan elti veri setinde, Tablo 4.1'de üç tane hedef örnek verilmiştir.

Arkaplân verisinde tutulan akrabalık ilişkilerinden kardeşi (brother) b ile, karısı (wife) w ile, kocası (husband) h ile gösterilmektedir. Hedef ilişki olan elti/2, arkaplân bilgisi olarak tutulan kardeş (b), karı (w) ve koca (h) ilişkileri cinsinden tanımlanır.

Tablo 4.1. Elti veri seti

| Hedef Örnekler | Arkaplân Verisi | |
|---|---|---|
| e(cemile,ayse) | h(mehmet,cemile) | h(ali,ayse) |
| e(cemile,ayten) | b(mehmet,ali) | w(ayse,ali) |
| e(ayse,ayten) | w(cemile,mehmet) | w(ayten,ismail) |
| | b(ali,ismail) | b(ali,mehmet) |
| | m(cemile,dilek) | f(yusuf,ali) |
| | d(kubra,cemile) | b(mehmet,fatma) |

Önerilen yöntemde, ilişkisel formdaki veri yönlü çizge kullanılarak gösterilmektedir. Düğümler hedef ve arkaplân gerçeklerini tutarken, kenarlar en az bir ortak argümana sahip düğümleri birbirine bağlar. Şekil 4.1'de elti veri gösterimi görülmektedir.




Şekil 4.1. Elti verisinin çizge ile gösterimi

### 4.3. Algoritma

Yöntem girdi değeri olarak, veri ile birlikte kavram tanımlarını değerlendirmede kullanılan minimum destek (minimum support), min_sup; minimum güven (minimum confidence), min_conf ve maksimum kavram tanımlayıcı uzunluğu (maximum concept descriptor length), dmax, değerlerini alır. Yöntemin çıktı değeri ise, min_sup, min_conf kısıtlarını ve maksimum kavram tanımlayıcı uzunluğu dmax değerini sağlayan HOC sınıfına dahil kavram tanımlayıcılardır. Kavram tanımlayıcıları değerlendirme işlemi, destek ve güven değerlerine dayalı olarak yapılır. Yöntemin algoritması üç temel aşamadan meydana gelmektedir: çizge

oluşturma (graph construction), kavram tanımlayıcı çıkarımı (concept descriptor extraction), kavram tanımlayıcı değerlendirme (concept descriptor evaluation).

Çizge Oluşturma (graph construction): Bu aşamada, her bir hedef örnek için ayrı bir çizge oluşturulur. Hedef örnek, çizgenin kök düğümü olarak belirlenir ve bu kök düğümden başlanarak, gerçekleri tutan diğer düğümler, ilgili oldukları düğümlere eklenir. İki gerçek arasında en az bir ortak argüman varsa bunlar ilgili gerçek (related fact) olarak adlandırılır. Şekil 4.2 çizge oluşturma algoritmasını göstermektedir.

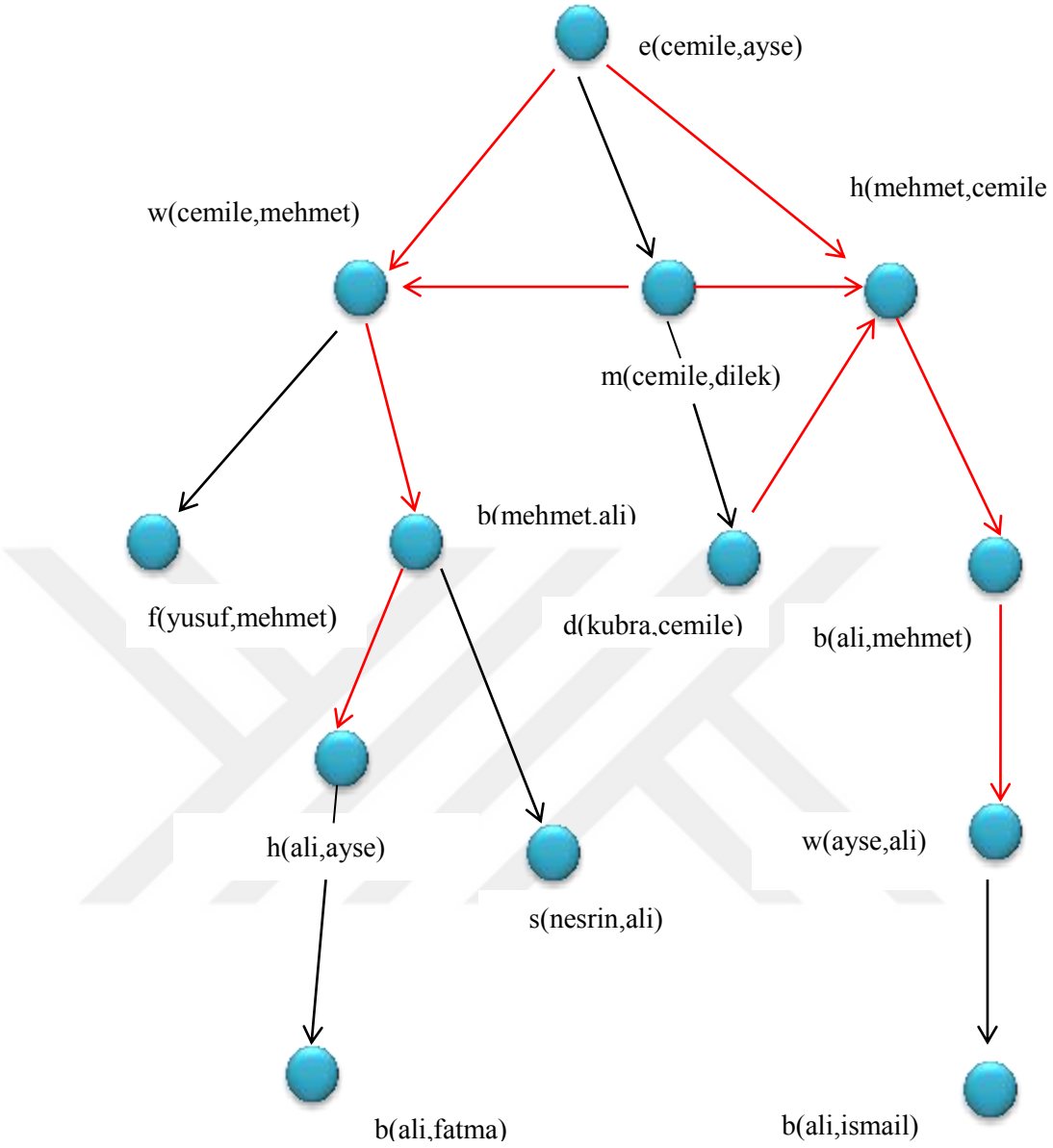
```
Başla: İlişkisel veri tabanı, D
Sağla: D 'yi, Gi çizgesine aktar
1: for (D 'deki her bir hedef örnek ti için) do
2: v = addVertex(Gi)
3: v.rn = ti
4: v.path.push(v)
5: toExpand.push(v)
6: for (d = 1; d<dmax; d++) do
7:       while (!toExpand.empty()) do
8:          v = toExpand.pop()
9:           r = getRelatedFacts(v.rn)
10:        for (i = 0; i< r.size(); i++) do
11:           it = find(Gi, r[i])
12:           if (it != NULL) then
13:              it.path=v.path
14:              it.path.push(it)
15:             addEdge(Gi, v, it)
16:             nodesAtLevel[i][d].push(it)
17:           else
18:              t = addVertex(Gi)
19:              t.rn = rfs[i]
20:              t.path=v.path
21:              t.path.push(t)
22:              addEdge(Gi, v, t)
23:              nodesAtLevel[i][d].push(v)
24:              toExpandTmp.push(t)
25:           end if
26:        end for
27:        toExpand = toExpandTmp
28:        toExpandTmp.clear()
29:     end while
30:   end for
31: end for
```

Şekil 4.2. Çizge oluşturma algoritması

Yöntemde çizgeler oluşturulurken paket özellikleri (bundled properties) kullanılmakta olup, rn özelliği düğümlerdeki ilişki bilgisini tutar. Bu özelliğin tuttuğu ilişki bilgisine, e(cemile,ayse), h(ali,ayse), w(cemile,mehmet) vb. örnek olarak verilebilir. Çizgedeki başlangıç düğümünden, gidilecek hedef düğüme kadarki yol bilgisini path özelliği tutar. Çizgeye yeni bir düğüm eklemek için, addVertex() fonksiyonu kullanılır. Mevcut iterasyonda genişletilecek düğümlerin listesini toExpand, sonraki iterasyonda genişletilecek düğümlerin listesini toExpandTmp tutar. İlgili bir gerçek için düğüm oluşturmada, düğümün çizgede mevcut olma durumuna bakılır. Bunun için, her bir seviyede düğümlerin tutulduğu nodesAtLevel dizisindeki düğümler kontrol edilir. Eğer düğüm çizgede mevcutsa, genişletilen düğümden yönlü bir kenar aracılığıyla, bu düğüme bağlantı sağlanır. Bunun için find() fonksiyonuna bakılır, fonksiyonun ikinci parametresi çizgede bir düğüm ile temsil ediliyorsa fonksiyon bir düğüm tanımlayıcı (vertex descriptor) döndürür. Düğüm çizgede mevcut değilse, çizgeye yeni bir düğüm olarak eklenir. Bu yeni düğümün verilerini oluşturmak için rn ve path özellikleri kullanılır. Genişleme esnasında çizgeye yeni bir düğüm eklendiğinde, bu düğüm toExpandTmp listesine eklenir. Öncelikle toExpand listesindeki bütün düğümler incelenir ve çizge buna göre düzenlenir. Daha sonra toExpandTmp listesindeki düğümler, toExpand listesine eklenir ve yeni bir iterasyon başlar. Bu genişleme dmax değeri kadar tekrar eder, dmax değerini aşan uzunluktaki yolların kullanılması gerekmez. Her bir iterasyonda, başlangıç düğümünden ulaşılabilecek düğümlere kadarki atlama sayısı nodesAtLevel dizisinde saklanır.

Maksimum kural uzunluğu parametresi kullanıldığı için, bu aşamada bütün veriyi çizgede göstermek gerekmez. Bunun yerine, sadece istenen uzunluktaki yollar için kavram tanımlayıcıların çıkarılabileceği kadar veri kullanılır.

Kavram Tanımlayıcı Çıkarımı (concept descriptor extraction): Bu aşamada ilk olarak çizgeden uzunluğu 1,2,…,dmax kadar olan yollar çıkarılır. Tez kapsamında elti veri seti için dmax değeri 3 olarak alınmaktadır. Elti veri seti 8 tane hedef örnekten oluşmakta, Şekil 4.3 bu hedef örneklerden elti(cemile,ayse) için oluşturulan örnek çizge üzerinde, kavram tanımlayıcıların çıkarımında kullanılan bazı yolları göstermektedir.




Şekil 4.3. Örnek elti çizgesi

Örnek çizgede e, elti; b, brother; h, husband; w, wife; m, mother; f, father; s, sister; d, daughter kavramlarını ifade etmektedir.

Yolları çıkarma işleminde, düğümlerin bilgisini tutan nodesAtLevel listesi ve her bir düğümün kendisini oluşturan farklı bütün yol bilgilerini tuttuğu, path özelliği kullanılır.

Örneğin Şekil 4.3'de, s(nesrin,ali) düğümüne ait path özelliğinde bu düğümü oluşturan yol bilgisi;

e(cemile,ayse), h(ali,ayse), b(mehmet,ali) ve

e(cemile,ayse), m(cemile,dilek), h(ali,ayse), b(mehmet,ali) şeklindedir.

Çizgede kırmızı oklarla gösterilen yollardan;

e(cemile,ayse) ← w(cemile,mehmet),b(mehmet,ali), h(ali,ayse)

e(cemile,ayse) ← h(mehmet,cemile),b(ali,mehmet),w(ayse,ali)

kavram kuralını ifade etmekte ve yolların destek ve güven değerlerinin hesaplanmasında kullanılmaktadır.

m(cemile,dilek) ← w(cemile,mehmet),b(mehmet,ali), h(ali,ayse)

m(cemile,dilek) ← h(mehmet,cemile),b(ali,mehmet),w(ayse,ali)

d(kubra,cemile) ← h(mehmet,cemile),b(ali,mehmet),w(ayse,ali)

cümleleri ise yolların güven değerinin hesaplanmasında kullanılan arkaplân gerçeklerini ifade etmektedir. Bir gerçeğin (fact) güven değerine katkı yapması için, bu gerçeğin çıkış kenarının (outgoing edge) izlediği yol ile kural cümlesindeki hedef kavrama ait çıkış kenarının izlediği yol, ortak olmalıdır. Diğer bir deyişle, bu gerçeğe ait gövde kısmındaki literallerin, kavram kuralının çıktığı cümleye ait hedef örneğin gövdesindeki literallerle aynı olması gerekir. Örneğin yukarıdaki cümlelerde, m(cemile,dilek) ve d(kubra,cemile) gerçekleri, hedef kavramı ifade eden elti(cemile,ayse) ile aynı gövde literallerine sahiptir.

Destek ve güven değerlerinin hesaplanması için, çizgeden çıkarılan yollardaki sabit argümanlar, değişkenlerle değiştirilir. Böylece çıkarılan bu yollar genelleştirilmiş bir forma dönüştürülür. Örneğin;

e(cemile,ayse) ← w(cemile,mehmet),b(mehmet,ali),h(ali,ayse) yoluna ait sabitlerin yerine değişkenler yazıldığında cümle aşağıdaki gibi ifade edilir.

e(A,B) ←w(A,C), b(C,D), h(D,B)

Genelleştirilmiş yol bilgileri ve bu yolların açıkladığı gerçekler için bir hash tablosu oluştulur. Şekil 4.4, bu aşamadaki algoritmayı göstermektedir.

```
Başla: Başlangıç düğümünden dmax kadar atlamada
ulaşılabilecek düğümler, nodesAtLevel[][]
Sağla: Genelleştirilmiş HOC kavram tanımlayıcılar
1: for (i = 0; i < #TargetInstances; i++) do
2:    for (j = 0; j < dmax; j++) do
3:       for (k = 0; k < nodesAtLevel[i][j].size(); k++) do
4:          s = generalizePath(G_i[nodesAtLevel[i][j][k]])
5:          if (isHOC(s)) then
6:          it = map.find(s)
7:          if (it == NULL) then
8:            map.insert(s, <i, nodesAtLevel[i][j][k]>)
9:           else
10:            map[it]→second.insert(s, <i, nodesAtLevel[i][j][k]>)
11:          end if
12:       end if
13:    end for
14:  end for
15: end for
```

Şekil 4.4. Kavram tanımlayıcı çıkarım algoritması

Tablo 4.2'de verilen örnek hash tablosunda genelleştirilmiş iki yol ve bu yolların açıkladığı gerçekler görülmektedir.

Tablo 4.2. Örnek hash tablosu

| Genelleştirilmiş Yollar | Gerçekler |
|---|---|
| e(A,B) ← w(A,C), b(C,D),h(D,B) | e(cemile,ayse)<br>e(ayse,fatma)<br>e(nalan,bedriye)<br>d(kubra,cemile)<br>m(cemile,dilek) |
| e(A,B) ← h(C,A), b(C,D),w(B,D) | e(cemile,ayse)<br>e(cemile,ayten)<br>e(cemile,fatma)<br>e(nalan,bedriye)<br>d(kubra,cemile) |

Hash tablosu < generalizedPath; paths > yapısındadır. Buradaki generalizedPath, genelleştirilmiş yol bilgilerini tutan string bir ifadedir. Paths ise, < graph_id; vertex > ikililerini tutan bir vektördür.

Hash tablosunda, değişkenler yollardaki sabit argümanların yerini aldığı için, farklı yollar aynı genelleştirilmiş yol içinde gösterilebilir. Tablo 4.2 incelendiğinde;

e(A,B) ← w(A,C), b(C,D), h(D,B)

yolunun üç ayrı elti çiftini ve iki tane de arkaplân gerçeğini açıkladığı görülmektedir. Daha önce açıklandığı gibi, hedef örneği açıklayan bir kavram kuralının aynı zamanda bir gerçeği açıklayabilmesi için, hedef örneğin gövdesindeki literaller ile, gerçeğin gövdesindeki literallerin aynı olması gerekir. Bu da, bulunan kavram kuralının sadece hedeflenen kavramı değil, hedef kavramla ilişkili diğer bazı kavramları da açıklayabildiği anlamına gelir.

Tabloda, e(cemile,ayse), hedef kavram; d(kubra,cemile) ve m(cemile,dilek) ise arkaplân bilgisinden gelen gerçekleri ifade eder.

Şekil 4.3'te verilen örnek çizge incelendiğinde bu gerçeklerin gövde kısımlarının aynı literallerden oluştuğu görülür. Bu da e(cemile,ayse) hedef kavramını açıklayan kavram kuralının, d(kubra,cemile) ve m(cemile,dilek) kavramlarını da açıkladığı anlamına gelir. Bu durum bulunan kavram kuralının güven değerini hesaplamada göz önünde bulundurulur. Diğer bir deyişle kuralın güvenilirliğinin test edilmesinde kullanılır. Tablo 4.2'den ayrıca;

e(A,B) ← h(C,A), b(C,D), w(B,D)

yolunun, dört ayrı elti çiftini ve bir tane de arkaplân gerçeğini açıkladığı sonucu çıkar.

Genelleştirilmiş yolların kavram kural çıkarımında kullanılması için HOC sınıfına dâhil olması gerekir. Genelleştirilmiş bir yolun HOC sınıfına dâhil olup olmadığını kontrol etmek için, oluşturulan hash tablosu kullanılır.

Şekil 4.5, hash tablosunda tutulan genelleştirilmiş yolların, HOC olup olmadığını test etmek için kullanılan algoritmayı göstermektedir.

```
Başla: Path, p
Sağla: True / False
1: for (i = 0; i < p[0].arguments.size(); i++) do
2:     s.insert(p[0].arguments[i].GeneralizedValue)
3: end for
4: for (i = 1; p.size() - 1; i++) do
5:     for (j = 0; j < p[i].arguments.size(); j++) do
6:         s1.insert(p[i].arguments[j].GeneralizedValue)
7:     end for
8:     s = s\s1 ∪ s1\s
9:     s1.clear()
10: end for
11: for (i = 0; i < p[p.size()].arguments.size(); i++) do
12:     s1.insert(p[p.size()].arguments[i].GeneralizedValue)
13: end for
14: if s\s1 == ∅ then
15:     return true
16: else
17:     return false
18: end if
```

Şekil 4.5. HOC testi algoritması

Algoritmada p değeri, HOC testine tabi tutulacak cümlenin uzunluğunu ifade eder. Bir cümlenin HOC olabilmesi için cümlenin gövdesinde bulunan bütün değişkenlerin bağımlı değişken olması gerekir.

e(A,B) ← w(A,C),b(C,D),h(D,B)

cümlesinde, e(A,B) ifadesi p[0] değerine; w(A,C) ifadesi ise p[1] değerine karşılık gelir. Başlangıç aşamasında p[1] değerine karşılık gelen ifadedeki A değişkeni, bağımlı; C değişkeni ise bağımsız değişkendir. Çünkü A değişkeni cümlenin baş (head) kısmında tanımlı olan A'ya karşılık gelmektedir. C değişkeni ise bu aşamada herhangi bir değişkene bağımlı değildir. Cümlenin p[2] değerine karşılık gelen ifadedeki C değişkeni ise bağımlıdır, çünkü bir önceki ifadedeki C değişkeni ile aynıdır. Bu ifadedeki D değişkeni ise bağımsız değişkendir. Cümlenin p[3] değerine karşılık gelen ifadede ise, hem D, hem de B bağımlı değişkendir. Algoritmada cümlenin yapısındaki literaller küme yapısından faydalanılarak karşılaştırılmakta ve son karşılaştırma işleminde kümeler arasındaki fark boş küme ise sonuç true dönmektedir. Bu da cümlenin HOC sınıfına dahil olduğu anlamına gelmektedir.

Örneğin yukarıdaki cümlenin HOC sınıfına ait olup olmadığı bulunmak istensin. S ve S1 gibi iki küme kullanılarak aşağıdaki adımlar takip edilir.

1. S = {A,B}
2. S1 = {A,C}
3. S = S\S1 ∪ S1\S
4. S = {B,C}
5. S1 = {C,D}
6. S = S\S1 ∪ S1\S
7. S = {B,D}
8. S1 = {B,D}
9. S = S\S1 ∪ S1\S
10. S = ∅

Son adımda karşılaştırma boş kümeye eşit olduğu için, cümledeki bütün değişkenlerin bağımlı değişken olduğu sonucu çıkar. Bu da bu cümlenin, HOC sınıfına dahil olduğu anlamına gelir. Örneğin aşağıdaki gibi bir cümle verilmiş olsun;

e(A,B) ← w(A,C),b(C,D),s(E,D)

S2 ve S3 gibi iki küme kullanılıp, yukarıdaki aşamalar izlendiğinde cümlenin HOC sınıfına ait olup olmadığı kolayca bulunabilir.

1. S2 = {A,B}
2. S3 = {A,C}
3. S2 = S2\S3 ∪ S3\S2
4. S2 = {B,C}
5. S3 = {C,D}
6. S2 = S2\S3 ∪ S3\S2
7. S2 = {B,D}
8. S3 = {E,D}
9. S2 = S2\S3 ∪ S3\S2
10. S2 = {B,E}
11. S2 ≠ ∅

Son adımda sonuç boş küme çıkmadığı için cümlenin gövde kısmında bağımsız değişkenlerin bulunduğu sonucuna varılır. Bu da, cümlenin HOC sınıfına dahil olmadığını göstermektedir.

Kavram Tanımlayıcı Değerlendirme (concept descriptor evaluation): Bu adımda yolların destek ve güven değerleri hesaplanır. Bir yolun destek değerini hesaplamak için, ikinci aşamada oluşturulan hash tablosu kullanılır. Genelleştirilmiş bir yolun destek kümesini bulmada, her bir v düğümü için yollardaki farklı $G_i$[v].path[0]'lerin sayısı hesaplanır. Buradaki $G_i$[v].path[0], elti veri seti için düşünüldüğünde e(cemile,ayse), e(bedriye,nalan),e(cemile,fatma) vb. hedef örnekleri tutan düğümleri ifade eder. Bir yolun destek değeri, hesaplanan destek kümesinin, hedef örnek sayısına bölünmesiyle bulunur. Tablo 4.2.'deki örnek hash tablosunda verilen bilgilere göre,

e(A,B) ← w(A,C), b(C,D), h(D,B)

yolu 3 tane hedef örneği kapsadığı için destek kümesi 3,

e(A,B) ← h(C,A), b(C,D), w(B,D)

yolu ise 4 tane hedef örneği kapsadığı için destek kümesi 4'tür. Verilen yollara ait destek değeri, elti veri setindeki toplam hedef örnek sayısının 8 olduğu bilgisi dâhilinde aşağıdaki şekilde hesaplanır:

$$\text{destek}_{e(A,B) \leftarrow w(A,C),\, b(C,D),\, h(D,B)} = \frac{3}{8} = 0{,}375$$

$$\text{destek}_{e(A,B) \leftarrow h(C,A),\, b(C,D),\, w(B,D)} = \frac{4}{8} = 0{,}50$$

Bir yolun güven değeri de yine çizge üzerindeki yollar kullanılarak bulunur. Belirli bir genelleştirilmiş yola katkısı olan düğümlere ait bilgiler hash tablosundan alınır. Çizgede her bir v düğümü için bu düğüme bağlı düğümler bulunur. $G_i$[v].path[0]'a ait argümanlar, ilgili düğüm tarafından gösterilen ilişkilerin argümanlarıyla değiştirilir. Bu değiştirilmiş yollar genelleştirilir ve orijinal genelleştirilmiş yollarla karşılaştırılır. Bu iki genelleştirilmiş yolun tam eşlenmesi, bu yolun ilgili gerçeği açıkladığı anlamına gelir ki bu gerçek, ilgili yolun güven kümesine katkı sağlar.

Şekil 4.3 incelendiğinde e(cemile,ayse)'ye ait kavram kuralının çıkarıldığı yolun, d(kubra,cemile) ve m(cemile,dilek) gerçeklerinin kullandığı yollar ile örtüştüğü görülmektedir. Çizgede v ile ifade edilen düğüm e(cemile,ayse), bu düğüme bağlı ilişkili düğüm ise m(cemile,dilek) olsun. Yukarıdaki bilgiden hareketle m(cemile,dilek) düğümünün tuttuğu argümanların yeri e(cemile,ayse) düğümünün argümanlarıyla değiştirildiğinde;

e(cemile,dilek) ← w(cemile,mehmet),b(mehmet,ali), h(ali,ayse)

cümlesi elde edilir. Bu cümle genelleştirildiğinde;

e(A,B) ← w(A,C), b(C,D), h(D,B)

şeklinde oluşan cümlenin, orijinal genelleştirilmiş cümle ile örtüştüğü görülür. Aynı durum d(kubra,dilek) düğümü için de söz konusudur. Bu da, bu gerçeklerin ilgili yolun güven kümesine katkı yapacağı anlamına gelir. Her bir yol için ister hedef örnek, ister arkaplân bilgisi olsun, yolun açıkladığı bütün gerçekler o yolun güven kümesine eklenir. Bir yolun güven değeri, destek kümesinin, bulunan bu güven kümesine bölünmesiyle hesaplanır. Örneğin Tablo 4.2'deki hash tablosunda verilen yolların güven değerleri hesaplanmak istensin.

e(A,B) ← w(A,C), b(C,D), h(D,B)

yolu için destek kümesi 3, güven kümesi 5;

e(A,B) ← h(C,A), b(C,D), w(B,D)

yolu için destek kümesi 4, güven kümesi 5 olduğu için bu yolların güven değeri aşağıdaki şekilde hesaplanır.

$$\text{güven}_{e(A,B) \leftarrow w(A,C),\, b(C,D),\, h(D,B)} = \frac{3}{5} = 0{,}60$$

$$\text{güven}_{e(A,B) \leftarrow h(C,A),\, b(C,D),\, w(B,D)} = \frac{4}{5} = 0{,}80$$

Kavram kuralını açıklayan yollara ait destek ve güven değerleri, bu yolların performansını değerlendirmek için kullanılır.

Şekil 4.6, destek ve güven değerlerinin hesaplanmasına ait algoritmayı göstermektedir.

```
Başla: Vector  < generalized, paths >, paths
Sağla: Sık ve güçlü kavram tanımlayıcılar
1: for (i = 0 i < paths.size(); i++) do
2:     p = paths[i]→second
3:     for (j = 0; j < p.size(); j++) do
4:         supSet.insert(G_p[j] →first[p[j] → second].rn)
5:         confSet.insert(G_p[j] →first[p[j] → second].rn)
6:         vC = getConnectedVertices()
7:         for (k = 0; k < vC.size(); k++) do
8:             G_p[j] →first[p[0]].rn.arguments = G_p[j] →first[vC[k]].rn.arguments
 9:             if (paths[i] →first == generalize(G_p[j] →first[vC[k]])) then
10:                 confSet.insert(G_p[j] →first[vC[k]].rn)
11:             end if
12:         end for
13:     end for
14:     support = supSet.size() / #TargetInstances
15:     confidence = supSet.size() / #confSet.size()
16:     if (support ≥ min_sup && confidence ≥  min_conf) then
17:         solution.push(paths[i] →first)
18:     end if
19: end for
```

Şekil 4.6. Kavram tanımlayıcıları değerlendirme algoritması

# 5. DENEYSEL BULGULAR

## 5.1. Deney Ortamı

Yöntemin uygulaması, Orta Doğu Teknik Üniversitesi Bilgisayar Mühendisliği Bölümü'ne ait NAR [111] adlı yüksek performanslı hesaplama sistemi üzerinde gerçekleştirilmiştir. NAR, 92 işlemci, her biri HP ProLiant BL460c olan toplam 368 çekirdeğe sahip 46 hesaplama düğümü,736 GB hafıza, 6,5 TB yerel disk alanı ve 6 TB ortak depolama alanından oluşmaktadır.

Yöntem, Scientific Linux v5.2 [112] 64-bit işletim sistemi üzerinde C++ dili ve g++ derleyici ile uygulanmıştır. Arama stratejisi olarak DLS algoritması tercih edilmiştir. Veri depolama ve yönetme aracı olarak MySQL, çizge yapısını oluşturmada boost_1_37_0 [113] kütüphanesi kullanılmıştır.

## 5.2. Veri Setleri

Önerilen yöntemin performansını ölçmek için, üç ayrı veri seti kullanılmıştır. Bu veri setlerinin ilki elti veri seti, ikincisi Fibonacci sayılarını ifade eden gerçeklerden oluşan ve tez kapsamında oluşturulan Fibonacci veri setidir. Üçüncü veri seti ise ikili akrabalık ilişkilerini tutan Family veri setidir.

Elti veri seti, elti/2 olarak gösterilen hedef kavram ve yine ikili olarak tutulan father, mother, brother, sister, son, doughter, husband ve wife ilişkilerinin kullanıldığı arkaplân bilgisini içeren toplam 224 örnekten oluşmaktadır.

Fibonacci sayıları, ilk iki sayıdan sonraki her bir sayının kendinden önceki iki sayının toplanması sonucunda oluşan bir sayı dizisidir. Bu durumda n. Fibonacci sayısı F(n), aşağıdaki şekilde ifade edilebilir;

$$F(n) = \begin{cases} 0 & n = 0 \\ 1 & n = 1 \\ F(n-1) + F(n-2) & n > 1 \end{cases} \quad (5.1)$$

F(n) ifadesinde, n'ye 0'dan başlamak üzere değerler verildiğinde, Fibonacci serisinin ilk 13 terimi Tablo 5.1'deki gibi bulunur.

Tablo 5.1. Fibonacci sayıları

| F(0) | F(1) | F(2 | F(3) | F(4) | F(5) | F(6) | F(7) | F(8) | F(9) | F(10) | F(11) | F(12) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 2 | 3 | 5 | 8 | 13 | 21 | 34 | 55 | 89 | 144 |

Fibonacci veri seti, 3-argümanlı plus (plus/3), 2-argümanlı predecessor (pred/2) ilişkilerini içeren arkaplân bilgisi ve 2-argümanlı fibonacci (fib/2) hedef ilişkisinden oluşur. Bu literallere ait argümanlar aşağıdaki gibi açıklanabilir;

pred(A, B) ← literalinde, B = A-1'i ifade eder.

plus(A, B, C) ← literalinde, C = A+B'yi ifade eder.

fib(A,B) ← literalinde ise B, A'nın değerine göre bulunan Fibonacci sayısını ifade eder. Bu ifadeye fib(7,13), fib(8,21), fib(9,34) örnek olarak gösterilebilir.

Family veri seti, wife, husband, mother, father, daughter, son, sister, brother, aunt, uncle, niece, nephew olarak tutulan hedef kavramlar ve bu kavramları ifade etmek için tutlan, married, parent, sibling, au ve gender arkaplân bilgilerinden oluşmaktadır. Kavram kurallarına ait bazı cümleler aşağıda listelenmiştir:

wife(A, B) ← fdt:gender(A, female), fdt:married(A, B)
husband(A, B) ← fdt:gender(A, male), fdt:married(A, B)
mother(A, B) ← fdt:gender(A, female), fdt:parent(A, B)
father(A, B) ← fdt:gender(A, male), fdt:parent(A, B)
daughter(A, B) ← fdt:gender(A, female), fdt:parent(B, A)
son(A, B) ← fdt:gender(A, male), fdt:parent(B, A)
sister(A, B) ← fdt:gender(A, female), sibling(A, B)
brother(A, B) ← fdt:gender(A, male), sibling(A, B)
aunt(A, B) ← fdt:gender(A, female), au(A, B)
uncle(A, B) ← fdt:gender(A, male), au(A, B)
niece(A, B) ← fdt:gender(A, female), au(B, A)
nephew(A, B) ← fdt:gender(A, male), au(B, A)

Veri setlerinin özellikleri Tablo 5.2'de verilmiştir. Minimum destek, minimum güven ve maksimum kural tanımlayıcı uzunluk parametreleri elti veri seti için [19], Family veri seti için [12] ve Fibonacci veri seti için [6]'dan alınmıştır.

Tablo 5.2. Veri setleri ve deneysel ayarları

| İsim | İlişki Sayısı | Örnek Sayısı | Min. Des. | Min. Güv. | Kural Uznl. |
|---|---|---|---|---|---|
| Elti | 9 | 224 | 0,2 | 0,6 | 3 |
| Family | 12 | 744 | 0,1 | 0,7 | 9 |
| Fibonacci | 5 | 30 | 1 | 1 | 5 |

### 5.3. Performans Ölçümü

Önerilen yöntemin performansı, diğer sistemler ile kapsam, çalışma zamanı, çıkarılan kavram tanımlayıcı sayısı ve kavram tanımlayıcıların uzunluğu açısından karşılaştırılmıştır.

Kapsam: Oluşturulan hipotezin yakaladığı örnek sayısının, verideki bütün hedef örnek sayısına oranıdır. Diğer bir deyişle bulunan destek değeridir.

Çalışma zamanı: Çalışma zamanı, çizge oluşturma ve kural çıkarımı için gerekli olan zamanı ifade eder.

Kavram tanımlayıcı sayısı: Yöntemin çıkardığı kavram tanımlayıcıların sayısıdır.

Kural uzunluğu: Kavram tanımlayıcıların gövdesindeki literallerin sayısını ifade eder.

Hedef örneklerle dolaylı bir şekilde ilişkili gerçekleri içeren elti veri seti, önerilen yöntemin performansını değerlendirmek için kullanılan ilk veri seti olmuştur. Yöntemin bu veri seti üzerindeki performansı H-GCD ile karşılaştırılmıştır. Fibonacci verisi, n-argümanlı ilişkiler üzerinde özyinelemeli kavram tanımlayıcılarını bulmada, yöntemin performansını değerlendirmek için kullanılmıştır. Yöntemin bu veri seti üzerindeki performansı ise [6]'daki çalışma ile karşılaştırılmıştır. Family veri setindeki bütün ilişkiler, akrabalık ifade eden farklı kategorik alanlardan oluşmaktadır. Yöntemin bu veri seti üzerinde, RPBL ve H-GCD ile olan karşılaştırmalı performansı Tablo 5.3'de verilmiştir. Tablodaki alanlar

kavram tanımlayıcılarını (K.T), ortalama kural uzunluğunu (O.K.U), kapsamı (Kap.) ve çalışma zamanını (Ç.Z) ifade etmektedir. Kapsam değeri yüzde (%), çalışma zamanı saniye cinsindendir.

Tablo 5.3. Family veri seti karşılaştırılması

| Veri Seti | Önerilen Yöntem | | | | RPBL | | | H-GCD | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | #K.T | O.K.U | Kap. | Ç.Z | #K.T | O.K.U | Kap. | #K.T | O.K.U | Kap. |
| Brother | 18 | 2 | 100 | 1,19 | 2 | 6 | 95 | 2 | 2.5 | 100 |
| Niece | 4 | 2 | 100 | 1,06 | 2 | 6 | 95 | 7 | 2 | 100 |
| Nephew | 4 | 2 | 100 | 1,06 | 2 | 6 | 95 | 7 | 2 | 100 |
| Mother | 4 | 2 | 100 | 1,44 | 2 | 6 | 95 | 7 | 2 | 100 |
| Daughter | 4 | 2 | 100 | 1,45 | 2 | 6 | 95 | 7 | 2 | 100 |
| Wife | 7 | 2 | 100 | 1,12 | 2 | 6 | 95 | 7 | 2 | 100 |
| Uncle | 4 | 2 | 100 | 1,14 | 2 | 6 | 100 | 9 | 2 | 100 |
| Aunt | 4 | 2 | 100 | 1,14 | 2 | 6 | 100 | 10 | 2 | 100 |
| Son | 4 | 2 | 100 | 1,45 | 2 | 6 | 100 | 10 | 2 | 100 |
| Father | 4 | 2 | 100 | 1,45 | 2 | 6 | 100 | 10 | 2 | 100 |
| Sister | 10 | 2 | 100 | 1,18 | 2 | 6 | 100 | 10 | 2 | 100 |
| Husband | 50 | 2 | 100 | 1,09 | 2 | 6 | 100 | 10 | 2 | 100 |

Deneysel sonuçlar H-GCD ile karşılaştırıldığında, önerilen yöntem, aynı uzunluğa sahip kuralları değişken sayıdaki kavram tanımlayıcılarla bulmaktadır. Örneğin önerilen yöntem brother verisi için 18, niece verisi için 4 tane kavram tanımlayıcı bulurken, H-GCD için bu sayılar 2 ve 7 olmuştur. Bu durum destek ve güven değerlerinin hesaplanma yöntemlerinden kaynaklanmaktadır. Örneğin veri setinde, p/2 olarak tutulan bir hedef ilişki ve ar1, ar2 gibi rasgele seçilmiş iki argüman bulunsun. H-GCD'de hedef ilişkinin argümanlarının bu ar1 ve ar2 değerleriyle yer değiştirmesi mümkündür. Ancak önerilen yöntemde bunun mümkün olması, bu iki değere sahip bir gerçeğin mevcudiyeti durumuna bağlıdır. Diğer bir deyişle, önerilen yöntemde bir hedef ilişkiye ait argümanlar, ancak ilişkili olduğu bir gerçeğin argümanları ile yer değiştirebilir.

Önerilen yöntem RPBL ile karşılaştırıldığında, farklı sayılarda kavram tanımlayıcı çıkarmaktadır. RPBL verinin yarısı için %95 diğer yarısı için %100 başarı oranıyla 2 kavram tanımlayıcı bulmaktadır. Önerilen yöntem bütün veri için %100 başarı oranıyla değişken sayıda kavram tanımlayıcı bulmaktadır. Az sayıdaki kavram tanımlayıcı ile bütün hedef örnekleri tanımlamak olumlu bir durum olduğu için, her iki yöntemin tam başarı sağladığ veriler için, RPBL'nin daha avantajlı sonuçlar elde ettiği söylenebilir.

Önerilen yöntem kavram tanımlayıcı uzunluğu açısından RPBL ile karşılaştırıldığında, daha kısa kavram tanımlayıcılar çıkarmaktadır. Çok sayıda literale sahip kavram tanımlayıcıların anlaşılması ve yorumlanması zordur ve aşırı öğrenme (overfitting) problemi açısından öznel bir karakter gösterir. Diğer bir deyişle bu tür kavram tanımlayıcılar, sistemin uygulandığı spesifik model üzerinde oldukça başarılı öğrenmeler sağladığı halde farklı veri grupları üzerinde aynı başarıyı sağlayamayabilir. Önerilen yöntem, aynı kesinlik ve duyarlılıktaki değerlerle daha basit kavram tanımlayıcılar çıkardığı için, bu açıdan önerilen yöntemin daha avantajlı olduğu söylenebilir.

Sistemler çalışma zamanı bakımından karşılaştırıldığında, önerilen yöntem ve H-GCD kavram tanımlayıcıları yaklaşık 1 saniyede bulurken, RPBL için bu süre 0,027 saniye olmaktadır. Önerilen yöntem ve H-GCD 2,66 GHz işlemci ve 16 GB hafızaya sahip ortamda, RPBL ise 1,6 GHz işlemci ve 2 GB hafızaya sahip ortamda uygulanmıştır. Önerilen yöntem RPBL'ye göre daha uzun sürede kavram tanımlayıcıları bulmaktadır. Yöntemin paralel uygulamasının yöntemin bu kısıtını aşmada olumlu katkılar sağlayacağı düşünülmektedir.

Önerilen yöntemin özyinelemeli ilişkiler ve çoklu ilişkiler ile olan uygulamaları, kullanıcı girişli Fibonacci veri seti üzerinde yapılmıştır. Yöntem, [6]'da 0,4 saniyede bulunan aşağıdaki kuralı yaklaşık 1 saniyede çıkarmaktadır.

fib(A,B)← pred(C,A), fib(C,D), pred(E,C), fib(E,F), plus(F,D,B)

Elti veri seti için H-GCD ile karşılaştırıldığında, önerilen yöntem, aşağıda örnekleri verilen çözüm cümlelerini 2 saniyede bulurken, H-GCD için bu süre 0,51 saniye olmaktadır.

e(A,B) ← h(C,A), b(C,D), h(D,B)

e(A,B) ← h(C,A), b(C,D), w(B,D)

e(A,B) ← w(A,C), b(C,D), h(D,B)

e(A,B) ← w(A,C), b(C,D), w(B,D)

Bunun sebebi H-GCD'nin bütün veri için tek bir çizge oluşturması, önerilen yöntemin ise her bir hedef örnek için ayrı bir çizge oluşturmasıdır. Bu ilk bakışta önerilen yöntemin olumsuz bir yönü olarak görülebilir ancak bu özelliği, yöntemi paralel bir şekilde oluşturmaya elverişli kılmaktadır. Sonraki bir çalışmada, farklı veri setleri üzerinde yöntemin paralel uygulamasının yapılması düşünülmektedir. Tablo 5.4 önerilen yöntemin, üç veri seti üzerindeki kapsam ve ortalama çalışma zamanını özetlemektedir.

Tablo 5.4. Önerilen yöntemin performansı

| Very Seti | Ort.Kural Uzunl. | Kapsam | Ort.Ç.Z(sn) |
|---|---|---|---|
| Elti | 3 | 100 | 2 |
| Family | 2 | 100 | 1,23 |
| Fibonacci | 5 | 100 | 1 |

# 6. SONUÇLAR VE ÖNERİLER

Veri tabanlarında birbiriyle ilişkili verileri bir arada bulundurma ihtiyacı, çok ilişkili veri tabanlarının ortaya çıkmasını gerekli kılmıştır. Çok ilişkili veri tabanlarında, veriler, farklı tablolarda tutulur. İlişkisel formdaki bu tür veriler için çok ilişkili veri madenciliği yaklaşımları, klasik veri madenciliği yaklaşımlarına ek olarak ortaya çıkmıştır.

Kavram keşfi çok ilişkili veri madenciliğinin temel problemlerinden biridir. Kavram keşfinde, hedef kavram ve bu hedef kavramla doğrudan ya da dolaylı olarak ilişkili olan diğer veriler, ilişkili bir veri tabanında, ayrı ayrı tablolarda tutulur. Kavram keşif probleminde, hedef kavramın, arkaplân verisi olarak tutulan diğer ilişkiler cinsinden tanımlanması amaçlanır. Bu alanda yapılan çalışmalar, arama yönü, arama tekniği, arama algoritması vb. açılardan farklı sistemlere sahiptir. Bu sistemlerin çeşitli avantaj ve dezavantajları mevcuttur.

Kavram keşif problemi uzun süre TMP tabanlı araştırmalara konu olmuştur. TMP ‘nin bu alandaki başarılı çalışmalarının yanı sıra, bazı kısıtları da bulunmaktadır. Temel olarak farklı alanlardaki çok ilişkili veri tabanları üzerinde oldukça başarılı sonuçlar veren TMP, büyük arama alanlarında, hem çalışma zamanı açısından hem de hafıza gereksinimi açısından verimlilik ve ölçeklenebilirlik sorunuyla karşılaşabilmektedir. Ayrıca HOC sınıfına ait problemlerin çözümünde etkili sonuçlar verememekte, birinci derece Horn cümleciklerinin kullanıldığı sistemlerde, yerel plato ve yerel maksimum noktalara takılma problemi yaşamaktadır.

Çizge tabanlı kavram keşif sistemleri, TMP ‘nin bu tür kısıtlarına çözüm bulmak için kullanılan alternatif yaklaşımlardır. Çizge yapısının ilişkili verileri göstermedeki uygunluğu ve TMP ‘nin mevcut kısıtlarının üstesinden gelmedeki başarısı bu alanda çizge madenciliği yaklaşımlarına hız kazandırmıştır. Kavram keşfinde, eşyapı tabanlı ve yol bulma tabanlı çizge madenciliği yaklaşımlarını kullanan, çeşitli öğrenme problemlerine ait başarılı çalışmalar yapılmıştır. Yine de bu alandaki çizge tabanlı sistemlerin dezavantaj olarak görülebilecek bazı yönlerinden söz etmek mümkündür.

Örneğin çizge izomorfizminin kullanıldığı eşyapı tabanlı yaklaşımlarda, benzer alt örüntüleri bulmak maliyetli arama algoritmalarını gerektirebilmektedir. Yol bulma tabanlı yaklaşımlarda ise, yol bilgilerini tutmak için karmaşık indeksleme yöntemleri kullanılmaktadır. Ancak son zamanlarda yapılan bazı çalışmalar, bu dezavantajları azaltmada oldukça başarılı sonuçlar elde etmiştir.

Bu tez çalışmasında, çok ilişkili bir veri tabanından, HOC sınıfına dahil kavram tanımlayıcı cümleleri çıkaran, yol bulma tabanlı bir kavram keşif yöntemi anlatılmıştır. Yöntemde güncel çizge tabanlı yaklaşımlardan farklı olarak, her bir hedef örnek için ayrı bir çizge oluşturulmuş ve maksimum kural uzunluğu parametresine göre, sadece kavram tanımlayıcıların elde edileceği kadar veri kullanılmıştır. Ayrıca yollar, çizge oluşturulurken çıkarılmış ve her düğüm kendisine ait yol bilgisini tuttuğu için, karmaşık indeksleme yöntemlerine ihtiyaç duyulmamıştır. Yolları değerlendirme ölçütü olarak, destek ve güven değerleri kullanılmıştır. Minimum destek ve minimum güven değerleri kullanıcı girişli olduğu için, çizgeden çıkan yolların sayısı bu parametrelere verilen değere göre değişmektedir.

Deneysel sonuçlar, yöntemin, farklı öğrenme problemleri için kavram tanımlayıcılarını, %100'lük bir oranla doğru bir şekilde çıkardığını göstermiştir. Yöntem, iyi bilinen yol bulma tabanlı bir sistem olan RPBL ile karşılaştırıldığında çalışma zamanı açısından RPBL daha iyi bir performans göstermiş, ancak bulunan kavram tanımlayıcıların uzunluğu dikkate alındığında, önerilen yöntem, yorumlanması ve anlaşılması daha kolay sonuçlar elde etmiştir. Kullanıcı deneyimi göz önüne alındığında, bu durum önerilen yöntemin olumlu bir sonucu olarak görülebilir.

Önerilen yöntem, elti veri seti için H-GCD ile karşılaştırıldığında aynı çözüm cümlelerini daha uzun sürede bulmuştur. Bunun nedeni her iki yöntemin veriyi çizgede göstermesindeki farklılıktır. H-GCD, bütün veri için tek bir çizge oluştururken, önerilen yöntem her hedef kavram için ayrı çizge oluşturmuştur. Ancak yöntemin paralel olarak uygulanabileceği düşünüldüğünde, bu yönünün bir avantaj olduğu söylenebilir.

Yöntemde, özyinelemeli kavram tanımlayıcılarını bulmak için, hedef örnekler arkaplân verisi olarak kullanılmış ve Fibonacci veri seti üzerinde yapılan deneylerde yöntemin, özyinelemeli kavram tanımlayıcıları da yine %100'lük bir oranla doğru bir şekilde bulduğu gözlenmiştir.

Önerilen yöntemin büyük bir kısıtı, sürekli alanda argümanları işleyememesi olarak görülebilir. Yapılması düşünülen bir çalışmada, bu tür veri setlerini işlemek için bir ayrıklaştırma yönteminin uygulanması uygun görülmektedir. Yine yapılması düşünülen bir başka çalışmada ise, önerilen yöntemin paralel uygulaması hedeflenmekte, bunun çalışma zamanı üzerinde olumlu sonuçlar üreteceği ve bu alandaki performansını önemli ölçüde arttıracağı düşünülmektedir. Önerilen yöntemde çizgeler üzerindeki işlemler birbirinden bağımsız olduğu için, yöntemin kolay bir şekilde paralel hale getirileceği ön görülmektedir.

## KAYNAKLAR


[1] Codd E. F., A Relational Model of Data for Large Shared Data Banks, *Communications of The ACM*, 1970, **13**(6), 377-387.

[2] Džeroski S., Multi-relational Data Mining: An Introduction, *SIGKDD Explorations*, 2003, **5**(1), 1-16.

[3] Muggleton S., Inductive Logic Programming, *New Generation Computing*, 1991, **8**(4), 295-318.

[4] Alphonse É., Osmani, A., On The Connection Between The Phase Transition of The Covering Test and The Learning Success Rate in ILP, *Machine Learning*, 2008, **70**(2-3), 135-150.

[5] Mutlu A., Improving Scalability and Efficiency of ILP-based and Graph-based Concept Discovery Systems, PhD Thesis, Middle East Technical University, Graduate School of Natural and Applied Science, Ankara, 2013, 338355.

[6] Santos J. C., Tamaddoni-Nezhad A., Muggleton S., An ILP System for Learning Head Output Connected Predicates, *Portuguese Conference on Artificial Intelligence*, Aveiro, Portugal, 12-15 October 2009.

[7] Richards B. L. and Mooney R. J., Learning Relations by Pathfinding, *The Tenth National Conference on Artificial Intelligence (AAAI)*, San Jose, California, USA, 12-16 July 1992.

[8] Ong I. M., de Castro Dutra, I., Page, D.,Costa, V. S., Mode Directed Path Finding, *European Conference on Machine Learning*, Porto, Portugal, 3-7 October 2005.

[9] Lavrač N., Džeroski S.,Grobelnik M., Learning Nonrecursive Definitions of Relations with LINUS, *European Working Session on Learning*, Porto, Portugal, 6-8 March 1991.

[10] Quinlan J. R., Learning Logical Definitions from Relations, *Machine Learning*, 1990, **5**(3), 239-266.

[11] Reiter R., On Closed World Data Bases, *Symposium on Logic and Data Bases*, Toulouse, France, 16-18 November 1977.

[12] Muggleton S., Tamaddoni-Nezhad A., QG/GA: A Stochastic Search for Progol, *Machine, Learning*, 2008, **70**(2-3), 121-133.

[13] Srinivasan A., The Aleph Manual, http://www.cs.ox.ac.uk/activities/machinelearning/Aleph/, (Ziyaret tarihi: 05 Eylül 2016).

[14] Dehaspe L., De Raedt L., Mining Association Rules in Multiple Relations, *International Conference on Inductive Logic Programming*, Prague, Czech Republic, 17-20 September 1997.

[15] Agrawal R., Mannila H., Srikant R., Toivonen H.,Verkamo A. I., Fast Discovery of Association Rules, *Advances in Knowledge Discovery and Data Mining*, 1996, **12**(1), 307-328.

[16] Muggleton S., Feng C., Efficient Induction of Logic Programs, *The First Conference on Algorithmic Learning Theory*, Tokyo, Japan, 8-10 October 1990.

[17] Muggleton S., Buntine W., Machine Invention of First Order Predicates by Inverting Resolution, *5th International Workshop on Machine Learning*, Ann Arbor, USA, 12-14 June 1988.

[18] Kavurucu Y., Senkul P.,Toroslu I. H, ILP-based Concept Discovery in Multi-Relational Data Mining, *Expert Systems with Applications*, 2009, **36**(9), 11418-11428.

[19] Kavurucu Y., Senkul P., Toroslu I. H, Concept Discovery on Relational Databases: New Techniques for Search Space Pruning and Rule Quality Improvement, *Knowledge-Based Systems*, 2010, **23**(8), 743-756.

[20] Gao Z., Zhang Z., Huang Z., Extensions to the Relational Paths Based Learning Approach RPBL, *ACIIDS*, Dong Hoi, Vietnam, 1-3 April 2009.

[21] Gao Z., Zhang Z., Huang Z., Learning Relations by Path Finding and Simultaneous Covering, *CSIE*, Los Angeles, USA, 31 March-2 April 2009.

[22] Abay N. C., Mutlu A., Karagoz P., A Graph-Based Concept Discovery Method for n-Ary Relations, *International Conference on Big Data Analytics and Knowledge Discovery*, Valencia, Spain, 2-4 September 2015.

[23] Holder L. B., Cook D. J., Djoko S., Substucture Discovery in the SUBDUE System, *KDD Workshop*, Seattle, USA, 31 July-01 August 1994.

[24] Gonzalez J.A., Holder L.B., Cook D.J., Graph-based Concept Learning, *FLAIRS Conference, AAAI Press*, Key West, USA, 21-23 May, 2001.

[25] Yoshida K., Motoda H., Indurkhya N., Graph-based Induction as A Unified Learning Framework, *Applied Intelligence*, 1994, **4**(3), 297-316.

[26] Matsuda T., Motoda H., Washio T., Graph-based Induction and Its Applications, *Advanced Engineering Informatics*, 2002, **16**(2), 135-143.

[27] Yoshida K., Motoda H., Clip: Concept Learning from Inference Patterns, *Artificial Intelligence*, 1995, **75**(1), 63–92.

[28] Mutlu A., Karagoz P., A Hybrid Graph-based Method for Concept Rule Discovery, *International Conference on Data Warehousing and Knowledge Discovery*, Prague, Czech Republic, 26-29 August 2013.

[29] Iğde M., Kavurucu Y., Mutlu A., Graph Representation of Relational Database for Concept Discovery, *Procedia-Social and Behavioral Sciences*, 2015, **195**, 1981-1989.

[30] Han J., Kamber M., Pei J., *Data Mining: Concepts and Techniques*, 3rd ed., Morgan Kaufmann, USA, 2011.

[31] Fayyad U., Piatetsky-Shapiro G., Smyth P., From Data Mining to Knowledge Discovery in Databases, *AI Magazine*, 1996, **17**(3), 37.

[32] Savaş S., Topaloğlu N., Yılmaz M., Veri Madenciliği ve Türkiye'deki Uygulama Örnekleri, *İstanbul Ticaret Üniversitesi Fen Bilimleri Dergisi*, 2012, **11** (21), 1-23.

[33] İlhan S., Küme Merkezlerinin Belirlenmesinde Yeni Bir Yöntem (IFART), Doktora Tezi, Kocaeli Üniversitesi, Fen Bilimleri Enstitüsü, Kocaeli, 2009, 259494.

[34] Özekes S.**,** Data Mining Models and Application Areas, *İstanbul Commerce University Journal of Science*, 2003, **3**, 65-82.

[35] Ramkumar G. D., Swami A. N., Clustering Data Without Distance Functions, *IEEE Data Eng. Bull.*, 1998, **21**(1), 9-14.

[36] Ince H., Aktan B., A Comparison of Data Mining Techniques for Credit Scoring in Banking: A Managerial Perspective, *Journal of Business Economics and Management*, 2009, **3**, 233-240.

[37] Ling C. X., Li C., Data Mining for Direct Marketing: Problems and Solutions, *KDD*, New York, USA, 27-31 August, 1998.

[38] Estivill-Castro V., Lee I., Data Mining Techniques for Autonomous Exploration of Large Volumes of Geo-referenced Crime Data, *6th International Conference on Geocomputation*, Brisbane, Australia, 24-26 September 2001.

[39] Xu J., Chen H., Criminal Network Analysis and Visualization, *Communications of the ACM*, 2005, **48**(6), 100-107.

[40] Al-Radaideh Q. A., Assaf A., Alnagi E., Predicting Stock Prices Using Data Mining Techniques, *The International Arab Conference on Information Technology* (ACIT'2013), Xian, China, 23-25 May, 2013.

[41] Masoud R., Ahmed T. M., Using Data Mining in Telecommunication

Industry: Customer's Churn Prediction Model, *Journal of Theoretical and Applied Information Technology*, 2016, **91**(2), 322.

[42] Durairaj M., Ranjani V., Data Mining Applications in Healthcare Sector: A Study, *Int. J. Sci. Technol. Res. IJSTR*, 2013, **2**(10), 29-35.

[43] Watson T. L., Bachu S., Evaluation of The Potential for Gas and CO2 Leakage Along Wellbores, *SPE Drilling & Completion*, 2009, **24**(01), 115-126.

[44] Zubi Z. S., Saad R. A., Using Some Data Mining Techniques for Early Diagnosis of Lung Cancer, *10th WSEAS International Conference on Artificial Intelligence, Knowledge Engineering and Data Bases*, Cambridge, UK, 20-22 February 2011.

[45] Bellazzi R., Zupan B., Predictive Data Mining in Clinical Medicine: Current Issues and Guidelines, *International Journal of Medical Informatics*, 2008, **77**(2), 81-97.

[46] Cios K. J., Moore G. W., Uniqueness of Medical Data Mining, *Artificial Intelligence in Medicine*, 2002, **26**(1), 1-24.

[47] Rhodes D. R., Yu, J., Shanker K., Deshpande N., Varambally R., Ghosh D., Chinnaiyan A. M., ONCOMINE: A Cancer Microarray Database and Integrated Data-Mining Platform, *Neoplasia*, 2004, **6**(1), 1-6.

[48] Perez-Iratxeta C., Bork P., Andrade M. A., Association of Genes to Genetically Inherited Diseases Using Data Mining, *Nature Genetics*, 2002, **31**(3), 316-319.

[49] Götz S., García-Gómez J. M., Terol J., Williams T. D., Nagaraj S. H., Nueda M. J., Conesa A., High-throughput Functional Annotation and Data Mining with The Blast2GO Suite, *Nucleic Acids Research*, 2008, **36**(10), 3420-3435.

[50] Kumar S., Tamura K., Nei M., MEGA3: Integrated Software for Molecular Evolutionary Genetics Analysis and Sequence Alignment, *Briefings in Bioinformatics*, 2004, **5**(2), 150-163.

[51] Futschik M. E., Sullivan M., Reeve A., Kasabov N., Prediction of Clinical Behaviour and Treatment for Cancers, *Applied Bioinformatics*, 2003, **2**, 53-58.

[52] Liu H., Li J., Wong L., Use of Extreme Patient Samples for Outcome Prediction from Gene Expression Data, *Bioinformatics*, 2005, **21**(16), 3377-3384.

[53] Wang J. T., Zaki M. J., Toivonen H. T., Shasha, D., Introduction to Data Mining in Bioinformatics, Editors: Wang J. T., Zaki M. J., Toivonen H. T., *Data Mining in Bioinformatic*, 1st ed., Springer Verlag, London, 3-8, 2005.

[54] İğde M., Graph Based Concept Discovery for Multi Relational Data Mining, A Master Thesis, Turkish Naval Academy, Naval Science end Engineering Institute Department of Computer Sience, İstanbul, 2015, 386948.

[55] Cestnik B., Kononenko I., Bratko I., ASSISTANT 86: A Knowledge Elicitation Tool for Sophisticated Users, *EWSL*, Bled, Yugoslavia, 13-15 May 1987.

[56] Mozetic I., NEWGEM: Program for Learning from Examples, Technical Documentation and User's Guide, *Department of Computer Science, University of Illinois*, UIUCDCS-F-85-949, 1985.

[57] Clark P., Boswell R., Rule Induction with CN2: Some Recent Improvements, *Machine Learning EWSL*, Porto, Portugal, 6-8 March 1991.

[58] Clark P., Niblett T., The CN2 Induction Algorithm, *Machine Learning*, 1989, **3**(4), 261-283.

[59] Michalski R. S., Mozetic I., Hong J., Lavrac N., The Multi-purpose Incremental Learning System AQ15 and Its Testing Application to Three Medical Domains, *Proc. AAAI*, Philadelphia, USA, 11-15 August, 1986.

[60] Huchard M., Hacene M. R., Roume C., Valtchev P., Relational Concept Discovery in Structured Datasets, *Annals of Mathematics and Artificial Intelligence*, 2007,**49**(1-4), 39-76.

[61] Srinivasan A., King R. D., Muggleton S. H., Sternberg M. J., Carcinogenesis Predictions Using Inductive Logic Programming, Editors: Lavrač N., Keravnou E.T., Zupan B., *Intelligent Data Analysis in Medicine and Pharmacology*, vol. 414, Springer US, Boston, 243-260, 1997.

[62] Clare A., King R. D., Predicting Gene Function in Saccharomyces Cerevisiae, *Bioinformatics*, 2003, **19**(suppl 2), ii42-ii49.

[63] Dehaspe L., Van Laer, W., De Raedt L., Applications of a Logical Discovery Engine, *4th International Workshop on Inductive Logic Programming*, Bonn, Germany, 12-14 September 1994.

[64] Klingspor V., Morik K. J., Rieger A. D., Learning Concepts from Sensor Data of a Mobile Robot, *Machine Learning*, 1996, **23**(2-3), 305-332.

[65] King R. D., Srinivasan A., Dehaspe L.,Warmr: A Data Mining Tool for Chemical Data, *Journal of Computer-Aided Molecular Design*, 2001, **15**(2), 173-181.

[66] Mooney R. J., Califf M. E., Induction of First-order Decision Lists: Results on Learning The Past Tense of English Verbs, *JAIR*, 1995, **3** , 1-24.

[67] Cussens J., Part-of-Speech Disambiguation Using ILP., *Oxford University Computing Laboratory*, Technical Report PRG-TR-25-96, 1996.

[68] Pazzani M. J., Brunk C. A., Silverstein G., A Knowledge-intensive Approach to Learning Relational Concepts, *8th International Workshop on Machine Learning*, Evanston, USA, 27-29 June 1991.

[69] Richards B. L., Mooney R. J., First-order Theory Revision, *8th International Workshop on Machine Learning*, Evanston, USA, 27-29 June 1991.

[70] Gonzalez J. A., Holder L. B., Cook D. J., Graph-based Relational Concept Learning, *19th. International Conference on Machine Learning*, Sydney, Australia, 08-12 July 2002.

[71] Muggleton S., De Raedt L., Inductive Logic Programming: Theory and Methods, *The Journal of Logic Programming*, 1994, **19**, 629-679.

[72] Poveda J. P., Borràs J. T., Inductive Logic Programming and Its Application to The Temporal Expression Chunking Problem, *LSI Department Technical Report*, PhD Thesis Programme on Artificial Intelligence (UPC), 2007.

[73] Shapiro E., *Algorithmic Program Debugging*, MIT Press, Cambridge, 1983.

[74] Sammut C., Banerji R., Learning Concepts by Asking Questions, Editors: Michalski S. R., Carbonell, G. J. M., Mitchell T., *Machine Learning: An Articial Intelligence Approach*, Vol. 2, Morgan Kaufmann, San Francisco, 67-191, 1986.

[75] Džeroski S., Lavrac N., *Inductive Logic Programming: Techniques and Applications*, Ellis Horwood, New York, 1994.

[76] Otero R., Tamaddoni-Nezhad A., *Inductive Logic Programming*, London Academic Press., 1992.

[77] Giannotti F., Manco G., Wijsen J., Logical Languages for Data Mining, Editors: Saake G., Meyden R., *Logics for Emerging Applications of Databases*, Springer Verlag, New York, 325-361, 2004.

[78] De Raedt L., Blockeel H., Dehaspe L., Van Laer W., Three Companions for Data Mining in First Order Logic, Editors: Džeroski S., Lavrač N., *Relational Data Mining*, Springer Berlin Heidelberg, 105-139, 2001.

[79] I. Mozetič., Learning of Qualitative Models, Editors: Bratko, I., Lavrač., N., *Progress in Machine Learning*, Sigma Press, 201-217, 1987.

[80] Srinivasan A., King R. D., Muggleton S. H., Sternberg M. J., The Predictive Toxicology Evaluation Challenge, *IJCAI*, Aichi, Japan, 23-29 August 1997.

[81] Quinlan J. R., *C4. 5: Programming for Machine Learning*, Morgan Kauffmann, San Francisco, USA, 1993.

[82] De Raedt L., Džeroski S., First-order JK-Clausal Theories are PAC-Learnable, *Artificial Intelligence*, 1994, **70**(1), 375-392.

[83] De Raedt L., Dehaspe L., Clausal Discovery, *Machine Learning*, 1994, **26**(23), 99-146.

[84] Colton S., Muggleton S., Mathematical Applications of Inductive Logic Programming, *Machine Learning*, 2006, **64**(1-3), 25-64.

[85] Jiang N., Colton S., Boosting Descriptive ILP for Predictive Learning in Bioinformatics, *International Conference on Inductive Logic Programming*, Santiago de Compostela, Spain, 24-27 August 2006.

[86] Dickson A., *Introduction to Graph Theory*, CRC Press, 2006.

[87] https://www.tutorialspoint.com/neo4j/neo4j_need_for_graph_databses.htm, (Ziyaret tarihi: 05 Nisan 2017).

[88] Washio T., Motoda H., State of The Art of Graph-based Data Mining, *Acm Sigkdd Explorations Newsletter*, 2003, **5**(1), 59-68.

[89] Gholami M., Salajegheh A., A Survey on Algorithms of Mining Frequent Subgraphs, *International Journal of Engineering Inventions*, 2012, **1**(5), 60-63.

[90] Kurumatani N., Monji H., Ohkawa T., Binding Site Extraction by Similar Subgraphs Mining from Protein Molecular Surfaces and Its Application to Protein Classification, *International Journal on Artificial Intelligence Tools*, 2014, **23**(03), 1460007.

[91] Romanowski C. J., Nagi R., Sudit M., Data Mining in an Engineering Design Environment: OR Applications from Graph Matching, *Computers & Operations Research*, 2006, **33**(11), 3150-3160.

[92] Cook D. J., Manocha N., Holder L. B., Using a Graph-based Data Mining System to Perform Web Search, *International Journal of Pattern Recognition and Artificial Intelligence*, 2003, **17**(05), 705-720.

[93] Kumar R., Raghavan P., Rajagopalan S., Sivakumar D., Tompkins A., Upfal E., The Web as a Grap, *ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems,* Dallas, USA, 15-18 May 2000.

[94] Bonchi F., Castillo C., Gionis A., Jaimes A., Social Network Analysis and Mining for Business Applications, *ACM Transactions on Intelligent Systems and Technology (TIST*), 2011, **2**(3), 22.

[95] Mukherjee M., Holder L. B., Graph-based Data Mining on Social Networks, *Workshop on Link Analysis and Group Detection,* Seattle, USA, 22-25 August 2004.

[96] Noble C. C., Cook D. J., Graph-based Anomaly Detection, *The Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Washington, USA, 24-27 August 2003.

[97] Wille R., Restructuring Lattice Theory: An Approach Based on Hierarchies of Concepts, *In Ordered Sets*, Springer Netherlands, 445-470, 1982.

[98] Liquiere M., Sallantin J., Structural Machine Learning with Galois Lattice and Graphs, *15th International Conference on Machine Learning*, Wisconsin, USA, 24-27 July 1998.

[99] Rensink A., Representing First-order Logic Using Graphs, *International Conference on Graph Transformation*, Rome, Italy, 28 September-2 October, 2004.

[100] Blockeel H., Witsenburg T., Kok J. N., Graphs, Hypergraphs, and Inductive Logic Programming, *MLG*, Firenze, Italy, 1-3 August 2007.

[101] Jiang C., Coenen F., Zito M., A Survey of Frequent Subgraph Mining Algorithms, *The Knowledge Engineering Review*, 2013, **28**(01), 75-105.

[102] Rissanen J., Stochastic Complexity in Statistical Inquiry, Vol. 15, *World Scientific*, USA, 1998.

[103] Cook D.J., Holder L. B., Substructure Discovery Using Minimum Description Length and Background Knowledge, *Journal of Artificial Intelligence Research*, 1994, **1**, 231-255.

[104] Ausiello G., Giaccio R., Italiano G. F., Nanni, U., Optimal Traversal of Directed Hypergraphs, *International Computer Science Institute*, Tecnical Report TR-92-073, 315-319, 1992.

[105] Thomas H., Cormen Leiserson C. E., Rivest R. L., Stein C., *Introduction to Algorithms*, 3rd ed., Cambridge: MIT Press.,USA, 590-610, 2009.

[106] Inokuchi A., Washio T., Motoda H., An Apriori-based Algorithm for Mining Frequent Substructures from Graph Data, *European Conference on Principles of Data Mining and Knowledge Discovery*, Lyon, France, 13-16 September 2000.

[107] Gonzalez J., Holder L. B., Cook D. J., Application of Graph-based Concept Learning to The Predictive Toxicology Domain, *5th European Conference PKDD*, Freiburg, Germany, 3-5 September 2001.

[108] Van Rijsbergen C. J., *Information Retrieval*, 2nd ed., Butterworth, 1979.

[109] Powers D. M., Evaluation: from Precision, Recall and F-measure to ROC, Informedness, Markedness and Correlation. *International Journal of Machine Learning Technology*, 2011, **2** (1), 37-63.

[110] Grünwald P., Model Selection Based on Minimum Description Length, *Journal of Mathematical Psychology*, 2000, **44**(1), 133-152.

[111] http://ceng.metu.edu.tr/hpc/index/, (Ziyaret tarihi: 01 Eylül, 2016).

[112] https://www.scientificlinux.org/, (Ziyaret tarihi: 01 Eylül 2016)

[113] http://www.boost.org/users/history/version_1_37_0.html, (Ziyaret tarihi: 01 Ekim 2016).

## KİŞİSEL YAYIN VE ESERLER

[1] **Peker N**., Mutlu A., A Graph-Path Counting Approach for Learning Head Output Connected Relations, *International Conference on Computational Collective Intelligence* (ICCCI), Halkidiki, Greece, 28-30 September 2016.

## ÖZGEÇMİŞ

Erzurum'da doğdu. İlk öğrenimini Erzurum'da, orta ve lise öğrenimini Kocaeli'de tamamladı. Kocaeli Üniversitesi Teknik-Eğitim Fakültesi Bilgisayar Öğretmenliği Bölümü'nden 2014 yılında mezun oldu. Aynı yıl başladığı Kocaeli Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı'ndaki Yüksek Lisans eğitimine devam etmektedir.