Tümevaran Mantık Programlama Tabanlı Kavram Keşif Sistemleri için Paralel bir Yöntem

Alev Mutlu[1], Pınar Şenkul[1], Yusuf Kavurucu[2]

1. Bilgisayar Mühendisliği Bölümü
Orta Doğu Teknik Üniversitesi
{mutlu,senkul}@ceng.metu.edu.tr

2. Deniz Harp Okulu
ykavurucu@dho.edu.tr

Öz

Tümevaran Mantık Programlama (TMP) tabanlı kavram keşif sistemlerinin bir çok alanda uygulaması bulunmakla birlikte bu yöntemlerin hala, büyük arama alanlarından kaynaklı, randıman ve ölçeklendirilebilirlik sorunları vardır. Paralelleştirme, TMP tabanlı kavram keşif sistemlerinde öğrenilen kavramların kalitelerini etkilemeden randıman ve ölçeklendirilebilirlik sorunlarına bir çözüm yöntemi olarak kullanılabilir. Bu çalışmada TMP tabanlı kavram keşif sistemlerinin arama alanı değerlendirmesi için paralel bir yöntem sunulmaktadır. Önerilen yöntem TPM tabanlı bir kavram keşif sistemi olan CRIS üzerinde uygulanmış ve deneysel sonuçları diğer bazı TMP tabanlı paralel kavram keşif sistemleriyle karşılaştırmalı olarak değerlendirilmiştir. Elde edilen sonuçlar önerilen yöntemin özellikle büyük verilerle çalışan ve pahalı sorgular işleten deneylerde daha başarılı olduğunu göstermiştir.

1. Giriş

Kavram keşfi (concept discovery) [1] hedef bir ilişki tanımının diğer ilişkiler cinsinden öğrenilmesi problemidir. Makine öğrenmesi ve mantık programlamanın kesişiminde yer alan Tümevaran Mantık Programlama'nın (TMP) (Inductive Logic Programming) [2] amacı verilen arkaplan bilgilerinin (background knowledge) ışığında eldeki örnekleri açıklayan genel kuramlar bulmaktır. Çok ilişkisel veri madenciliğinin ana problemlerinden biri olan kavram keşfi için TMP tabanlı yaklaşımlar sıklıkla kullanılmaktadır [3].

TMP tabanlı çok ilişkisel kavram keşfinde öğrenilen kurallar genellikle Horn yantümcesi (Horn clause) şeklinde ifade edilmektedir ve hedef kavram bu yantümcedeki olumsuz belirticidir. TMP tabanlı sistemlerin kullandığı özelleştirme ve genelleştirme operatörleri büyük arama alanları oluşturmaktadır. Kavram keşif probleminde arama alanını oluşturan hipotezlerin değerlendirilmesi pahalı bir işlem olduğu için bu tür sistemlerin büyük verilerde uygulanmasında sıkıntılar doğmaktadır. Bu sıkıntıları aşmak için literatürde farklı yöntemler önerilmiştir: etkin veri yapılarının kullanılması [4], sezgisel arama yöntemlerinin geliştirilmesi [5] ve bulanık mantık tabanlı yaklaşımların kullanılması [6] gibi. Diğer taraftan paralelleştime yöntemleri ise arama alanını kısıtlamadan ya da arama yöntemini değiştirmeden TMP tabanlı kavram keşif sistemlerinin randıman ve ölçeklendirilebilirlik sorunlarının aşılması için farklı bir yaklaşımdır.

Paralel sistemler bir ana işlemci ve birden çok işçi işlemciden oluşmaktadır. Problem ana işlemci tarafından alt problemlere bölünür, alt problemler işçi işlemcilere dağıtılır, orada işlenir ve kısmi sonuçlar ana işlemcide toplanarak ana problemin sonucu oluşturulur. Bu tür sistemler basit bir döngü içinde çalışmaktadır: işçi işlemci ana işlemciden işe başlama sinyali bekler, gelen işi işler, sonucunu ana işlemciye gönderir ve yeniden bekleme durumuna geçer [7].

TMP tabanlı kavram keşif sistemlerinin en pahalı aşaması arama alanının değerlendirilmesidir. Bu aşamada hipotezler genellikle SQL sorgu cümlelerine dönüştürülür ve veri tabanı sisteminde işletilir. Özellikle uzun hipotezler için bu sorgulamalar birden çok tablonun birleştirilmesini sonucu oluşan ara tablolar üzerinde yapılır ki bu sorgulamalar pahalı hesaplamalar gerektirir. Bu çalışmada TMP tabanlı kavram keşfi için paralel bir arama alanı değerlendirme yöntemi sunulmaktadır. Ana işlemci arama alanını oluşturan hipotezleri sorgu cümlelerine dönüştürmekte ve hesaplanması için işçi işlemcilere göndermektedir.

Önerilen yöntem ana-işçi işlemci ve dağıtık bellek tabanlı mimariler içindir. İşlemciler arasındaki haberleşme mesaj alışverişine dayanır. Önerilen yöntem parametrik yapıdadır. Kullanıcı arama alanının paralel olarak değerlendirilmesi için gereken en az sorgu sayısını ve ilk aşamada işçi işlemcilere gönderilecek sorgu sayısını belirleyebilmektedir. Mimari oluşturulurken işlemciler arası iletişime ayrılan sürenin ve son işlemcinin işlerini bitirmesi için bekleme süresinin en aza indirilmesi amaçlanmıştır. Ayrıca aynı sorgunun iki farklı işlemci tarafından işletilmemesi sağlanmıştır.

Önerilen yöntem TMP tabanlı bir kavram keşif sistemi olan CRIS [8, 9] üzerinde uygulanmıştır. Yapılan

deneysel çalışmalar önerilen yöntemin TMP tabanlı kavram keşif sistemlerinin çalışma zamanlarını iyileştirilmesinde ve ölçeklendirilmesinde başarılı olabileceğini göstermiştir. Elde edilen sonuçların literatürde var olan diğer çalışmlarala karşılaştırılabilecek nitelikte olduğu görülmüştür.

Bu çalışmanın ikinci kısmında tümevaran mantık programlama tabanlı kavram keşif problemi, üçüncü bölümünde CRIS anlatılmaktadır. Dördüncü bölümde TMP tabanlı kavram keşif sistemlerinin paralelleştirmesi için mevcut yöntemlerinden bazıları özetlenmektedir. Beşinci bölümde önerilen paralelleştirme yöntemi ve gerçekleştirim ortamı açıklanmaktadır. Altıncı bölümde deney ortamı sunularak deney sonuçları tartışılmaktadır. Son bölümde ise genel bir değerlendirme verilmektedir.

2. Tümevaran Mantık Programlama Tabanlı Kavram Keşfi

Tümevaran mantık programlama tabanlı kavram keşif sistemleri doğru ve yanlış örneklerden oluşan hedef kavram verilerini ve bu verilerle ilgili diğer arkaplan verilerini girdi olarak alır ve mümkün olduğunca çok doğru örneği (tam) (1) ve mümkün olduğu kadar az yanlış örneği açıklayan (tutarlı) (2) kuramları bulur. Bu tür sistemler kavram tanımlarını oluşturma yöntemlerine göre tabandan tepeye (bottom-up) [10] ve yukarıdan aşağıya (top-down) [11] sistemler olarak iki ana sınıfa ayrılabilir. İlk yaklaşımda sadece bir doğru örneği açıklayan özgül bir hipotez oluşturulur ve bu hipotez olabildiğince çok doğru örneği açıklayacak şekilde genelleştirilir. İkinci yaklaşımda ise tüm doğru ve yanlış örnekleri açıklayan genel bir hipotez oluşturulur ve bu hipotez yanlış örnekleri açıklamayacak şekilde özelleştirilir. Ayrıca her iki yaklaşımın yöntemlerini kullanan TMP tabanlı karma kavram keşif sistemleri mevcuttur.

$$tam(BF,H,E) = e \in E^{+} \left| BF \bigcup H \prec e \right. \quad (1)$$

$$tutarlı(BF,H,E) = e \in E^{-} \left| BF \bigcup H \prec \{\} \right. \quad (2)$$

Algoritma 1'de TMP tabanlı kavram keşif sistemleri için genel bir algoritma verilmiştir. 5. satırda verilen *refine* fonksiyonu kullanılan yönteme göre başlangıç hipotezini ya özelleştirir ya da genelleştirir. 6. satırdaki *evaluate* fonksiyonu hipotezin doğruluğunu ölçer. Eğer hipotez kullanıcı tarafından belirlenen kaliteyi sağlıyorsa, (satır 7), bu hipotezce açıklanan doğru hedef örnekler, hedef örnekler kümesinden silinir (satır 8) ve hipotez hedef kavram tanımları kümesine eklenir. Bu işlem doğru örnekler kümesi boşalana kadar ya da yeni bir hipotez üretilemeyene kadar devam eder.

```
Girdi: E: hedef kavram örnekleri, BF: arkaplan verileri
Çıktı: H: Tam ve tutarlı kavram tanımlamaları
1:  while E+ ≠ {} do
2:     Başlangıç hipotezini oluştur H'
3:     if H' ≠ {} then
4:        for all h ∈ H' do
5:           refine(h)
6:           evaluate(h, BF)
7:           if good(h) then
8:              cover(E+, h)
9:              H = H ⋃ h
10:          end if
11:       end for
12:    end if
13: end while
14: return H
```

Algoritma 1: Genel TMP tabanlı kavram keşf

Örnek olarak Tablo 1'de verilen *kız_çocuğu* (k_ç) hedef kavram ve *evebeyn* (e) ile *kadın* (k) arkaplan verileri için TMP tabanlı bir kavram keşif sisteminin

kız_çocuğu(X, Y) :- ebeveyn(Y, X), kadın(X)

kuramını bulması beklenir.

Tablo 1 kız_çocuğu veri kümesi

Hedef Kavram	Arkaplan Verileri	
k_ç(merve, ayşe)	e(ayşe, merve)	k(ayşe)
k_ç(emine, ömer)	e(ayşe, ömer)	k(merve)
	e(ömer, emine)	k(emine)

Temel verilerin karmaşıklığına bağlı olarak bulunan kavram tanımları farklı öğe boyutunda olabilir. Mesela *kız_çocugu* hedef kavramı için *anne*, *baba* ve *kadın* arkaplan verileri verilmişse, TMP tabanlı bir kavram keşif sisteminin

kız_çocuğu(X, Y) :- anne(Y,X), kadın(X)
kız_çocuğu(X, Y) :- baba(Y,X), kadın(X)

kuramlar kümesini bulması beklenir.

3. TMP-tabanlı bir Kavram Keşif Sistemi: CRIS

CRIS (Concept Rule Induction System), TMP tabanlı karma bir kavram keşif sistemdir. CRIS oluşturulan hipotezlerin uygunluğunu birlektelik kural madenciği (association rule mining) [12] yöntemlerini kullanarak değerlendirir. Bu bağlamda her hipotez için iki tane uygunluk değeri hesaplanır: destek (support) ve güven (confidence).

Bu değerlere göre bir hipotez üç farklı şekilde ele alınır:

1. Eğer hipotezin destek ve güven değerlerinden her ikisi de kullanıcı tarafından belirlenen asgari değerlerden yüksekse, bu hipotez sonuç kuramlar kümesine eklenir.
2. Eğer destek değeri asgari destek değerinden büyük ve güven değeri asgari güven değerinden küçükse bu hipotez bir sonraki aşamada özelleştirilir.
3. Eğer hipotezin destek değeri asgari destek değerinden küçükse hipotez arama alanından silinir.

CRIS dört ana bileşenden oluşmaktadır:

1. Genelleştirme: Bu aşamada en genel iki belirticili hipotezler oluşturulur. Belirticilerin argümanları değişken ya da sabit olabilir. Belirtici argümanının sabit bir değer olabilmesi için bu değerin ilgili atomik ilişkide en az

   *asgari destek * doğru örnek sayısı*

   kadar bulunması gerekir.
2. Özelleştirme: Bu aşamada *k* uzunluğundaki hipotezler özelleştirilerek *(k+1)* uzunluğundaki hipotezler oluşturulur. CRIS'te özelleştirme APRIORI [13] tabanlıdır. Buna göre özelleştirme sadece bir belirticiyle birbirinden ayrılan iki hipotezden ilkine onda eksik olan belirticinin eklenmesi ile elde edilir.
3. Filtreleme: Bu aşamada hipotezler oluşturulma sırasına göre teker teker SQL sorgularına çevirilir, destek ve güven değerlerinin hesaplanması için veri tabanı sistemine gönderilir. Hipotezler yukarıda anlatılan kriterlere göre değerlendirilir.
4. Kapsama Bulma: Bu aşama doğru kavram örneklerinden o aşamada bulunan kuramlarca açıklananlar silinir.

4. Literatür Özeti

Bu bölümde TMP tabanlı kavram keşif sistemleri için literatürde var olan bazı paralelleştirme yöntemleri özetlenmektedir.

TMP tabanlı kavram keşif sistemleri için önerilen paralelleştirme yöntemleri genel olarak (a) mimarileri: paylaşımlı bellek mimarisi / dağıtık bellek mimarisi; (b) paralel olarak işletilen kısımlar: paralel örtme / paralel hipotez oluşturma / arama alanının paralel değerlendirmesi bakımından farklılık gösterir.

TMP tabanlı kavram keşif sistemi CLAUDIEN [14] için paylaşımlı bellek tabanlı paralel sistem [15]'te sunulmuştur. Bu çalışmada hedef kavram örnekleri kümesi işlemciler arasında eşit olarak paylaştırılırken arka plan verilerin tamamı tüm işlemcilere dağıtılmıştır. İşçi işlemciler TMP tabanlı bir kavram keşfetme algoritması işletmekte ve ellerindeki verilere göre kısmi hipotezler oluşturmaktadır. Ana işlemci bu kısmi hipotezleri toplayarak asıl hipotezleri oluşturur.

TMP tabanlı kavram keşif sistemleri için dağıtık bellek mimarisine dayalı üç yaklaşım; paralel kapsama alanı testi (pct), veri paralel kural öğrenme (dprl) ve veri paralel TMP (dpilp) Fonseca ve arkadaşları tarafından [16]'da sunulmuştur. İlk yaklaşımda hedef kavram örnekleri işlemciler arasında eşit olarak dağıtılır ve işçi işlemciler kendilerine gönderilen sorguları bu veriler üstünde çalıştırır. Ana işlemci kısmi sonuçları toplayarak sorguların genel çözümünü hesaplar. *dprl* yaklaşımında ise işçi işlemcilere hedef kavramın yanlış örneklerin tamamı gönderilirken, hedef kavramın doğru örnekleri eşit olarak paylaştırılır. Her işlemci kendi verisi üzerinde kurallar bulur, bunları diğer işçi işlemcilere kendi verileri üzerinde sınaması için gönderir. Ana işlemci işçi işlemcilerden kavram kuralları için hesaplanan kısmı sonuçları toparlayarak oluşturulan bu kurallar için genel sonuçlar hesaplar. *dpilp* yaklaşımında ise hedef kavramın doğru ve yanlış örnekleri işçi işlemciler arasında eşit olarak dağıtılır, her işçi kendi verisi üzerimde TMP tabanlı bir kavram keşif algoritması işletir. Bulunan sonuçlar ana işlemcide toplanarak birleştirilir ve genel kuramlar oluşturulur.

Bu çalışmada önerilen yöntem *pct*, *dprl* ve *dpilp* sistemleri gibi dağıtık bellek mimarisi üzerinde çalışır. *CLAUDIEN* için önerilen paralel sistem gibi arkaplan verilerin tamamı ve ayrıca hedef kavam örnekleri tüm işlemcilere dağtılmıştır. *pct* yöntemine benzer olarak her işçi işlemci bazı sorgu cümlelerini kendi verisi üzerinde çalıştırır. Ancak önerilen yöntemde her işçi işlemci verilerin tamamına erişebildiği için, işçi işlemciler kendilerine gönderilen sorgunun ana sonucu hesaplamış olurlar.

5. Paralel CRIS

Bu bölüm üç alt kısımdan oluşmaktadır. İlk kısımda bu çalışmanın arkasındaki motivasyon, ikinci kısımda önerilen yöntem, son kısımda ise yöntemin gerçekleştirimi açıklanmıştır.

5.1. Motivasyon

CRIS'in çalışması incelendiğinde çalışma süresinin büyük kısmını arama alanın değerlendirilmesi aşamasının oluşturduğu gözlenmiştir. Tablo 2'de bazı veri kümeleri için CRIS'te arama alanı değerlendirme ve diğer işlemler için harcanan zaman yaklaşık yüzdelik oranlar olarak verilmiştir. Benzer şekilde, Aleph [10] sisteminde PTE veri kümesi için toplam çalışma zamanın yaklaşık %90'nının, Muta veri kümesi için ise yaklaşık %80'ninin arama alanı değerlendirmesi

aşamasında harcandığı Costa ve arkadaşlarının çalışmasında [17] rapor edilmiştir.

Bu verilerden de anlaşılabileceği gibi TMP tabanlı bir kavram keşif sistemin randımanı ve ölçeklendirilebilirliği büyük ölçüde arama alanı değerlendirme aşamasının performansının iyileştirilmesine bağlıdır.

Tablo 2 Bazı veri kümeleri için CRIS'in çalışma zamanı

Veri Kümesi	Arama alanı değerlendirme zamanı	Diğer işlemler
Muta	%84	%16
Mesh	%71	%29
PTE	%98	%2
PTE-5	%99	%1

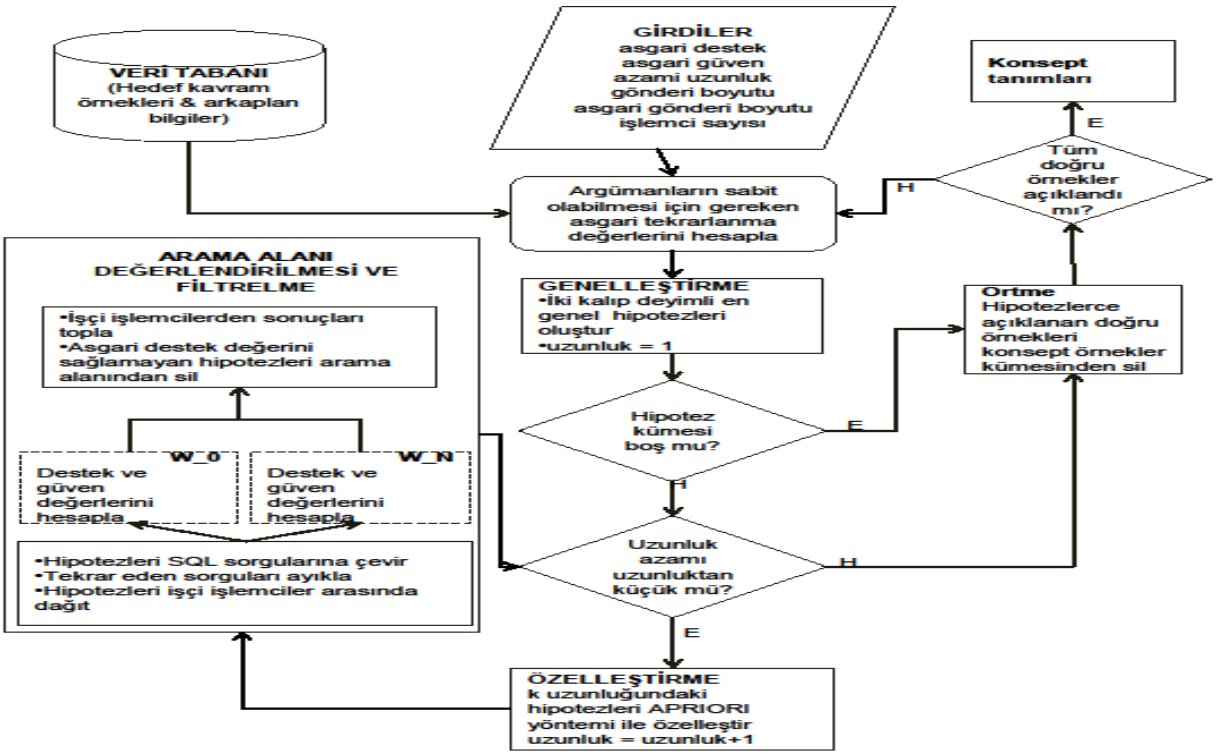
Arama alanı değerlendirmesi aşamasında hipotezler SQL seçme sorgularına dönüştürülür ve bu sorgular veri tabanında işletilir. Aynı veri kümesi üzerimde eşzamanlı çalışan iki seçme sorgusu birbirinin sonucunu etkilemediği için arama alanı değerlendirmesi paralel olarak yapılabilir.

5.2. Önerilen Paraleleştirme Yöntemi

Bu çalışmada önerilen TMP tabalı kavram keşif sistemleri için paralelleştirme yöntemi temel olarak arama alanı değerlendirme işleminin paralel olarak yapılmasına dayanır. Bu amaçla ana işlemci hipotezleri SQL sorgu cümlelerine çevirir ve bu sorguları işçi işlemcilere işletmesi için gönderir. Önerilen paraleleştirme yöntemi her işlemcinin kendisine ait belleği ve diski olan sistemler için tasarlanmıştır. İşlemciler arasındaki haberleşme mesajlaşmaya dayalıdır.

CRIS için önerilen paralelleştirme yöntemi Figür 1'de gösterilmiştir. Düz çizgilerle gösterilen işlemler ana işlemci tarafından yapılmakta, kesikli çizgilerle gösterilen işlemler ise işçi işlemciler tarafından yapılmaktadır.

CRIS algoritmasının argümanlarına ek olarak, bu algoritma kullanıcıdan *asgari gönderi boyutu*, *gönderi boyutu* ve *işlemci sayısı* parametrelerini almaktadır. Önerilen paralel mimaride arama alanı değerlendirme aşaması ancak oluşturulan sorgu sayısı asgari gönderi boyutu parametresinden fazlaysa paralel olarak yapılır.



Figür 1 Önerilen Paraleleştirme Algoritması

Farklı aşamalarda oluşturulan sorgu sayıları farklı olabileceği için önerilen mimaride bir çalıştırma içinde sorgu alanı değerlendirmesi bazı aşamalarda paralel bazı aşamalarda ise seri olarak yapılır. Benzer şekilde paralel olarak başlayan sorgu alanı değerlendirme aşaması kalan sorgu sayısı asgari gönderi boyutunun altına düşünce ana işlemci tarafından seri olarak yapılmaya devam eder. Bu yaklaşımlar sorgu sayısının çok az olduğu durumlarda işlemciler arasındaki haberleşme masrafının sorgu işletme masrafından fazla olma ihtimalini önlemek için geliştirilmiştir.

CRIS'te oluşturlan farklı hipotezler aynı SQL sorgularına denk gelebilir. CRIS mimarisinde bu sorun adres hesaplamalı tablolar kullanarak [18] ya da veri tabanı sistemlerinin sunduğu önbellekleme ile çözülebilir. İlk yaklaşımda sorgu cümlesi veri tabanında işletilmeden önce adres hesaplamalı tabloda aranır ve eğer sorgu tabloda mevcut ise sorgu sonucu tablodan çekilir. Eğer sorgu adres hesaplamalı tabloda yoksa, bu sorgu işletilmek üzere veri tabanı sistemine gönderilir, sonucu ile birlikte adres hesaplamalı tabloya eklenir. İkinci yaklaşımda ise tekrar eden sorguların ele alınması veri tabanı sisteminin sunduğu yeteneklere bırakılır. Önerilen paralel mimaride her işçi işlemcinin kendi adres hesaplamalı tablosunu oluşturması tekrar eden sorguların ancak aynı işçi işlemciye gönderilmesi durumunda faydalı olur. Ancak tekrar eden sorguların aynı işçi işlemciye gönderileceği kesin değildir. Önerilen paralel mimaride tekrar eden sorgu problemini çözmek için önce tüm hipotezler ana işlemci tarafından SQL sorgusuna çevirilir ve bu sorgulardan tekrar edenler ayrıklanır. Ana işlemci ilk olarak işçi işlemcilere gönderi boyutu kadar sorgu gönderir ve sorguların sonuçlarını beklemeye başlar. Ana işlemci sonucunu gönderen her işçi işlemciye azalan miktarda sorguyu işletilmeyen sorgu sayısı asgari gönderi boyutundan büyük olduğu sürece göndermeye devam eder. İşçi işlemcilere azalan miktarda sorgular gönderilerek en son işlemcinin işlemlerini bitirmesi için gereken bekleme zamanı en aza indirlmek amaçlanmaktadır. İşçi işlemcilerden sorgular ve sonuçları alındıkça bu sorgular ve sonuçları yeni bir adres hesaplamalı tabloya eklenir. Tüm sorgulamalar tamamlandıktan sonra ana işlemci hipotezlere denk gelen sorgu cümlelerini bu tabloda arayarak her hipotezin destek ve güven değerini belirler.

5.3. Gerçekleştirim

Yukarıda önerien yöntem CRIS algoritması üzerinde uygulanmıştır. Paralel algoritma C dilinde kodlanmıştır. İşlemciler arasındaki mesajlaşma için Boost [19] kütüphanesinde tanımlanan MPI fonksiyonları kullanılmıştır. Veri tabanı yönetim sistemi olarak MySQL [20] kullanılmıştır. Önerilen paralelleştirme yöntemi hiçbir şeyin paylaşılmadığı ortamlar için geçerli olduğundan veri tabanı yönetim sisteminin her işlemcide yüklenmiş ve verilerin hazır olması gerekmektedir.

6. Deneyler

Bu bölüm üç kısımdan oluşmaktadır. İlk kısımda deney ortamı açıklanmıştır. İkinci kısımda deneylerde kullanılan veri kümeleri açıklanmıştır. Son kısımda ise yapılan deneylerin sonuçları sunulmuş ve değerlendirilmiştir.

6.1. Deney Ortamı

Deneyler ODTÜ Bilgisayar Mühendisliği Bölümü Yüksel Başarımlı Hesaplama ortamında [21] gerçekleştirilmiştir. Bu hesaplama biriminin her ucu için donanımsal özellikleri şöyledir:

- 2 x Intel Xeon E5430 Quad-Core CPU
- 16 GB bellek
- 146 GB disk

Deney ortamı Scientific Linux V5.2 64-bit işletim sistemi üzerinde çalışmaktadır.

Deney ortamındaki kısıtlamalar nedeniyle sadece bir veri tabanı yönetim sistemi kurulmuş ve 8 adet nesnesi çalıştırılmıştır. Bu yapılandırmada her nesne için ayrı iletişim kapısı oluşturulmakta, bir nesne başka bir nesnenin bellek alanına erişememektedir.

6.2. Veri Kümeleri

Önerilen yöntem yedi farklı veri kümesi ile test edilmiştir. *Elti* ve *Dünür* [22] veri kümeleri aile ilişkileri ile ilgilidir ve Kinship [23] veri kümesi yapısındadır. *Eastbound* [24] veri kümesi trenler ve kapasiteleri hakkındadır. *Mesh* [25] veri kümesi çokgen yapıları oluşturan parçalarla ilgilidir. *PTE* [26] ve *Mutagenesis* [27] veri kümeleri biyokimyasal yapılar hakkındadır. İlk veri kümesinde kimyasalların kanserojen etkilerine, ikincide ise genlerde mutasyona yol açıp açmadığına dair bilgiler verilir. *PTE-5* veri kümesi *PTE* veri kümesinin kümelenmiş sayısal verilerle genişletilmiş halidir. Tablo 3 ve Tablo 4'te sırasıyla deneylerde kullanılan veri kümelerinin boyutları ve uygunluk değerleri verilmiştir.

Tablo 3 Veri kümelerinin boyutları

Veri Kümesi	Yüklem Sayısı	Örnek Sayısı
Dünür	9	224
Elti	9	224
Eastbound	12	196
Mesh	26	1749
Muta	8	279
PTE	32	29267
PTE-5	32	29267

Tablo 4 Veri kümeleri için deneylerde kullanılan uygunluk değerleri

Veri Kümesi	Asgari Destek	Asgari Güven
Dünür	0.3	0.7
Elti	0.3	0.7
Eastbound	0.1	0.1
Mesh	0.1	0.7
Muta	0.1	0.7
PTE	0.3	0.7
PTE-5	0.3	0.7

6.3. Deney Sonuçları

Önerilen yöntemin performans üzerindeki etkisini incelemek için yukarıda anlatılan 7 veri kümesi için 2, 4, 8 ve 16 işlemciden oluşan sistemlerde deneyler yapılmıştır. Tablo 5'te bu veri kümeleri için CRIS'in ve paralel yöntemin uygulandığı CRIS için 5 çalıştırmanın ortalaması ile oluşan çalışma zamanı listelenmiştir. Listelenen çalışma zamanlarına bakılınca önerilen yöntemin her veri kümesi için iyileştiği görülmüştür.

Tablo 5 5 çalıştırma üzerinden ortalama çalışma zamanı, zaman formatı [1]saniye.salise, [2]:dakika:saniye

Veri Kümesi	CRIS	İşlemci Sayısı			
		2	4	8	16
Dünür[1]	2.87	1.71	1.18	0.98	0.89
Elti[1]	2.49	2.14	1.47	1.16	0.99
Eastbound[1]	6.53	4.85	2.97	2.25	1.87
Mesh[1]	37.36	22.26	19.29	18.60	17.12
Muta[1]	11.98	7.78	5.91	5.48	4.3
PTE[2]	3:25	2:14	1:17	0:44	0:48
PTE-5[2]	36:13	15:20	5:50	4:21	4:19

Tablo 6'da işletilen toplam farklı sorgu sayısı ve ana işlemcinin oluşturduğu anahtar hesaplamalı tabloların boyutları verilmiştir.

Tablo 6 İşletilen sorgu sayısı ve ana işlemci tarafından kullanılan adres hespalamalı tablonun boyutu

Veri Kümesi	Sorgu Sayısı	Tablo boyutu ≈KB
Dünür	1775	28
Elti	2625	41
Eastbound	12245	2
Mesh	12715	75
Muta	12784	166
PTE	16242	184
PTE-5	112035	1163

Eastbound, Mesh, Muta ve PTE deneylerinde işletilen sorgu sayısı birbirine yakın olmasına karşın PTE deneyi diğerlerine nazaran çok daha uzun zaman almıştır. Bunun sebepleri ise

- Eastbound, Mesh ve Muta veri kümelerinin PTE'ye kıyasla çok daha küçük olması
- Eastbound kavramını açıklayan kuramların 2 belirticili, Mesh kavramını tanımlayanların ise 1 ve 2 belirticili iken PTE için 3 belirticili olmasıdır.

Figür 2 ve Figür 3'te sırasıyla farklı işlemciler için elde edilen hızlanma ve verimlilik sonuçları verilmiştir. Hızlanma (3) ve verimlilik (4) [28] paralel programların performansının değerlendirilmesinde kullanılan iki ölçüttür ve aşağıdaki gibi tanımlanırlar:

$$Hizlanma_p = \frac{T_1}{T_p} \tag{3}$$

$$Verimlilik_p = \frac{T_1}{T_p * p} \tag{4}$$

Denklem 3 ve 4'te verilen T_1, T_p ve p parametreleri sırasıyla seri programın çalışma zamanını, paralel programın p işlemci ile çalışma zamanını ve işlemci sayısını ifade etmektedir.

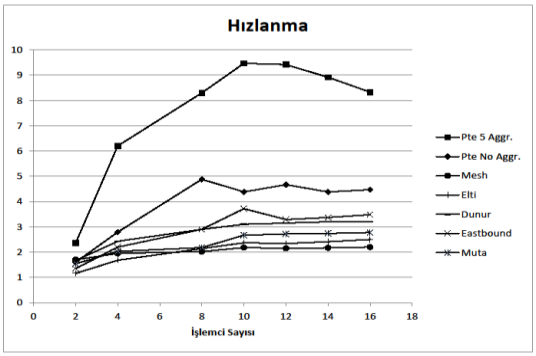
Tanım 1: Doğrusal hızlanma: p işlemci ile çalıştırılan bir programda hızlanmanın p olduğu durumdur. Bu bir paralel sistemden beklenebilecek ideal hızlanmadır.

Tanım 2: Süper doğrusal hızlanma: p işlemci ile çalıştırılıan bir programda hızlanmanın p'den büyük olduğu durumlardır.

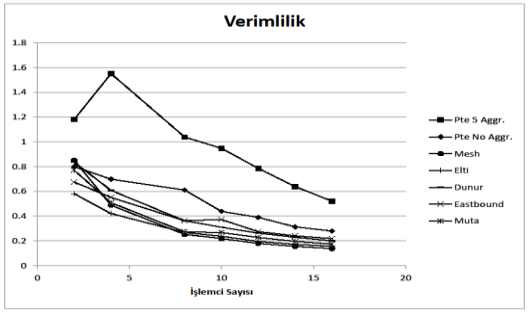
Hızlanma ölçütü paralel bir programın seri haline göre ne kadar hızlandığını gösterir. Verimlilik ise paralel bir çalıştırmada işlemcilerin ne kadar verimli kullanıldığını gösterir. Hızlanmanın işlemci sayısına her zaman eşit olduğu uygulamalar beklenebilecek en iyi yapılardır. Bu tür yapılarda verimlilik de her zaman 1'e eşit olur, ancak bunu sağlamak oldukça güçtür.

Figür 2 ve Figür 3'te görüldüğü gibi PTE-5 veri kümesi için 8 işlemciye kadar elde edilen hızlanma süper doğrusaldır. Diğer veri kümeleri için hızlanma değerleri elde edildiyse de bunlar kısmidir. PTE-5 veri kümesi diğerleri ile karşılaştılınca hem daha büyük hem de aralık karşılaştırmalı sayısal değerler de içerdiği için işletilen sorgular daha pahalıdır. Aynı şekilde PTE veri kümesi için elde edilen hızlanma da sorgu sayısı yakın olan Eastbound, Mesh ve Muta deneylerine göre daha büyüktür. Yukarıda da değinildiği gibi PTE veri kümesi

bunlara göre daha büyük ve oluşturulan hipotezler daha uzun ve dolayısıyla sorgular daha karmaşıktır. Bu sonuçlar önerilen yönetemin fazla sayıda ve pahalı sorgular işleten deneyler için kullanılabileceğini göstermektedir



Figür 2 Hızlanma Sonuçları



Figür 3 Verimlilik Sonuçları

.

Öte yandan 8'den fazla işlemci kullanılan deneylerde hızlanma artış oranında bir düşüş gözlenmektedir. Bunun sebebi ise 8'den fazla işlemci kullanılan deneylerde bir veri tabanı nesnesine birden fazla bağlantı yapılmasıdır. Tablo 7'de aynı veritabanı nesnesine birden fazla bağlantı yapıldığında PTE-5 veri kümesi için elde edilen hızlanmalar listelenmiştir. Bu değerler incelendiğinde aynı veri tabanı nesnesi yapılan bağlantı sayısının hızlanma üzerinde çok da etkili olmadığı görülebilmektedir.

Tablo 7 Aynı veritabanı nesnesine birden fazlabağlantı ile elde edilen hızlanmalar

Bağlantı Sayısı	Hızlanma
2	1.18
4	1.37
9	1.13

Diğer bir dizi deney de *gönderi boyutu* ve *asgari gönderi boyutu* parametrelerinin çalışma zamanı üzerindeki etkilerini incelemek için yapılmıştır. Gönderi boyutunun etkisini incelemek için 4 farklı gönderi boyutu ile deneyler yapışmıştır. Gönderi boytunun küçük olması işlemciler arasında iletişimin fazla olması anlamına gelir. Gönderi boyutunun büyük olması ise son işlemcinin işlemlerini bitirmesi için uzun süre beklenebileceğini gösterir. Tablo 8'de de görüldüğü gibi çok büyük ve çok küçük gönderi boyutları çalışma zamanını olumsuz etkilemektedir.

Tablo 8 Gönderi boyutu çalışma zamanı ilişkisi

Gönderi boyutu	Çalışma zamanı
20	8:32
100	5:14
200	4:42
1000	6:15

Tablo 9'de ise asgari gönderi boyutunun çalışma zamanı üzerindeki etkisi incelenmiştir. Asgari gönderi boyutunun 0 olması, büyüklüğüne bakılmaksızın arama alanını değerlendirmesinin paralel yapılacağı, 1000 olması ise sorgu sayısı 1000'den az olduğu sürece arama alanı değerlendirmesinin ana işlemci tarafından yapılacağı, diger işlemcilerin ise beklemede kalacağı anlamına gelir. Bu deneyde de uç değerlerin performansı kötü etkilediği görülmüştür.

Tablo 9 Asgari gönderi boytu çalışma zamanı

Asgari gönderi boyutu	Çalışma zamanı
0	5:43
50	5:36
200	4:55
1000	5:20

Asgari gönderi boyutu ve gönderi boyutu ile elde edilen bu sonuçlar Lowenthal'ın da çalışmasında [29] işaret ettiği gibi küçük gönderi boyutları boşta kalan işlemci zamanını azaltırken haberleşme masrafını artırmakta; büyük gönderi boyutları ise haberleşme masrafını azaltırken boşta kalan işlemci sayısını artırmaktadır tespiti ile aynı yönlüdür.

Tablo 10'da, önerilen yöntem PTE ve Mesh veri kümeleri için elde ettiği hızlanma *dpilp, pct* ve *dplr* isimli çalışmalarla karşılaştırılmıştır. Bu çalışmalar içinelde edilen hızlanma değerleri [16]'da rapor edilmiştir. Önerilen yöntem ve karşılaştırılan yöntemler farklı ortamlarda gerçekleştirilmiş ve sınanmış olsa dahi, karşılaştırma hızlanma değeri üzerinden yapıldığından ortam farklılıklarının etkisi önemli değildir.

Elde edilen değerler incelendiğinde önerilen yöntemin her iki deneyde de *pct* ve *dplr*'den daha başarılı olduğu göstermektedir. *dpilp* ile karşılaştırıldığında ise önerilen yöntem PTE veri kümesi için daha başarılı, Mesh için ise daha başarısızdır. Yukarıda da açıklandığı gibi PTE veri kümesi Mesh veri kümesine kıyasla çok daha büyüktür. Ayrıca PTE kavramı için bulunan kavram tanımları 3 uzunlukta iken Mesh için bulunanlar 1 ve 2 uzunluktadır. Bu sonuçlar önerilen yöntemin büyük veri kümeleri ve karmaşık sorgular için daha verimli çalıştığını göstermektedir.

Tablo 10 Diğer sistemlerle karşılaştırma

Veri Kümesi	İşmeci Sayısı	Önerilen yöntem	dpilp	pct	dplr
PTE	2	2.36	1.20	0.75	1.34
	4	6.2	3.23	0.42	2.31
	8	8.3	5.17	-	0.79
Mesh	2	1.21	2.75	0.76	0.38
	4	1.53	3.46	0.79	0.53
	8	1.41	4.35	0.80	1.18
	16	1.34	4.37	-	1.21

7. Sonuç ve Değerlendirme

Bu çalışmada yoğun şekilde veri tabanı sorgulaması yapan Tümevaran Mantık Programlama tabanlı kavram keşif sistemleri için bir paralelleştirme yöntemi önerilmiştir. Yöntem temel olarak ana işlemcinin sorguları oluşturması ve bu sorguları işletilmek üzere işçi işlemcilere dağıtmasına mantığına dayanmaktadır. Yapılan deneylerde gözlenmiştir ki küçük veri kümeleri ve kısa sorgulu deneyler için hızlanma elde edildiyse de asıl hızlanma büyük veri kümeleri ve pahalı sorguların işletildiği deneylerde elde edilmiştir. Yapılan deneyler önerilen yöntemin veri kümesi boyutu ve işletilen sorgu sayısı ile ölçeklenebildiğini göstermektedir. Bazı deneyler için süper doğrusal hızlanmalar elde edildiği görülmüştür. Ayrıca literatürde yayınlanan benzer bazı çalışmalar ile karşılaştırıldığında önerilen yöntemin büyük veri kümeleri için daha iyi sonuçlar ürettiği görülmüştür.

8. Kaynakça

[1] Quinlan, J. R. "Learning logical definitions from relations", *Machine Learning*, 5(3):239-266, 1990

[2] Muggleton, S., "Inductive Logic Programming", *The MIT Encyclopedia of the Cognitive Sciences,* 1999

[3] Dzeroski, S., "Multi-relational data mining: an introduction", *SIGKDD Explorations*, 5(1):1-16, 2003

[4] Mitra, S., Pal, S., "Data mining in soft computing framework: a survey", *IEEE Transactions on Neural Networks*, 13:3-14, 2002

[5] Mccreath, E., Sharma, A., "ILP with noise and fixed example size A bayesian approach", *International Joint Conferance on Artıficial Intelligence*, 1997

[6] Formica, A., "Semantic web search based on rough sets and fuzzy formal concept analysis", *Knowledge-based Systems*, 26:40-47, 2012

[7] Heymann, E., Senar, M. A., Luque, E., Livny, M., "Adaptive scheduling for master-worker applications on the computational grid", *IEEE/ACM International Workshop on Grid Computing,* 2000

[8] Kavurucu, Y., Şenkul, P., Toroslu, İ. H., "Concept discovery on relational databases: new techniques for search space pruning and rule quality improvement", *Konwledge-based Systems*, 23:743-756, 2010

[9] Kavurucu, Y., Şenkul, P., Toroslu, İ. H., "Confidence-based concept discovery in relational databases", *World Congress on Computer Science and Information Engineering*, 2009

[10] Srinivasan, A., "The Aleph manual", *http://www.comlab.ox.ac.uk/ctivities/machinelearning/*, 1999 son erişim Mart 2012

[11] Muggleton, S., Feng, C., "Efficient induction of logic programs", *Concferance on Algorithmic Learning Theory*, 1990

[12] Dehsape, L., Raedt, De L., "Mining association rules in multiple relations", *International Workshop on Inductive Logic programming*, 1997

[13] Agrawal, R., Mannila, H., Srikant, R., Toivonen H., Verkamo, A.I., "Fast discovery of association rules", *Advances in Knowledge Discovery and Data Mining*, 1996

[14] Raedt, L. D., Bruynooghe, M., "A theory of clausal discovery", *International Joint Conference on Artificial Intelligence*, 1993

[15] Dehaspe, L., Raedt, L. D., "Parallel inductive logic programming", *Familiarization Workshop on Statistics, Machine Learning and Knowledge Discovery in Databases*, 1995

[16] Fonseca, N. A., Silva, O., Costa, V. S., Camacho, R., "Strategies to parallelize ILP systems", *International Conference on Inductive Logic Programming* , 2005

[17] Costa, V. S., Srinivasan, A., Camacho, R., Blockeel, H., Demoen, B., "Query transformations for improving the efficiency of ILP systems", *Journal of Machine Learning Research*, 4:465-491, 2003

[18] Mutlu, A., Berk, M. A., Senkul,P., "Improving the time eciency of ILP-based multi-relational concept discovery with dynamic programming approach", *International Symposium on Computer and Information Sciences*, 2010

[19] http://www.boost.org, son erişim Mart 2012

[20] http: //www.mysql.com, son erişim Mart 2012

[21] http: //hpc.metu.edu.tr, son erişim Mart 2012

[22] Kavurucu, Y., Şenkul, P., Toroslu, İ. H., "Analyzing transitive rules on a Hybrid Concept Discovery System", *International Conference on Hybrid Artificial Intelligence Systems*, 2009
[23] Hinton, G., UCI Machine Learning Repository Kinship Data Set, *http:// archive.ics.uci.edu/ml/datasets/Kinship*, 1990.
[24] Michalski, R., Larson, J., "Inductıve inferenec of VL decision rules", *SIGART Bulletin*, 1977
[25] Dolsak, B., "Finite element mesh design system", *Knowledge-based Systems*, 15(8):315-322, 2002
[26] Srinivasan, A., King, R. D., Muggleton, S.H., Semtberg, M., "The predictive toxicology evaluation challenge", *International Joint Conference on Artificial Intelligence*, 1997
[27] A. Srinivasan, S. Muggleton, R. King, M. Sternberg, "Theories for mutagenicity: study of first-order and feature based induction", *Technical Report, PRG-TR-8-Oxford University Computing Laboratory*, 1995
[28] Eager, D., Zahorjan J., Lazowska, E., "Speedup versus efficiency in parallel systems", *IEEE Transactions on Computers*, 38:408-423, 1989
[29] Lowenthal, D. K., " Accurately selecting block size at runtime in pipelined programs", *International Journal of Parallel Programming*, 28:245-274, 2000