metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

**İSTİNYE ÜNİVERSİTESİ**

**MÜHENDİSLİK VE DOĞA BİLİMLERİ FAKÜLTESİ**

**YAZILIM MÜHENDİSLİĞİ BÖLÜMÜ**

**YAZ492: BİTİRME TEZİ 2**

**İLK RAPOR**

Nisan 2022

**PROJE BAŞLIĞI**

Paralel Mimarilerde Seyrek Alt Üçgen Matris Çözümü için Graf Parçalaması ile Yük Dengelemesi

**PROJE YAZARI**

Abdülkadir Furkan Yıldız - 190701145

**DANIŞMAN**

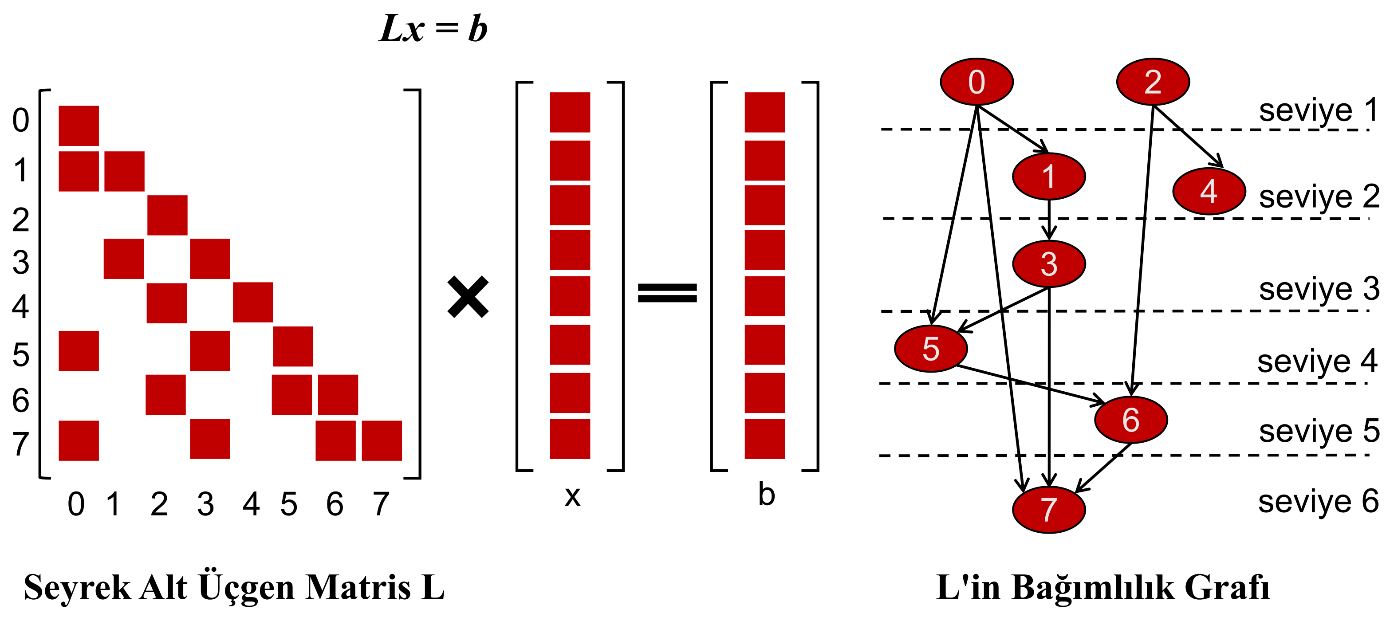
Dr. Öğr. Üyesi Buse Yılmaz

**ÖZET**

Chainbreaker paralel mimarilerde Seyrek Alt Uçgen Çözümü’nü optimize eden bir çerçevedir. Çerçevenin önemli bir modül yapılacak bağımlılık grafı dönüşümleri için stratejiler bütünü ve stratejilerin uygulanacağı seviyeleri belirleyen strateji koleksiyonu ve strateji seçme modülüdür. Projenin amacı, grafı aşağıdaki hedeflere ulaşılacağı şekilde dönüştürerek, seviyeler arasında ve içinde yük dengelemesini sağlayacak stratejileri ve uygulanacakları seviyeleri belirlemektir. Bu çalışmada, bir seyrek alt üçgen matrisin sınırlı paralellik gösteren parçalarının paralellik derecesini artırmak için bağımlılık grafını dönüştüren Chainbreaker çerçevesi için bir strateji koleksiyonu geliştirilecektir.

● Düşük paralellik derecesine sahip seyrek alt üçgen matris parçalarının paralelliğini graf dönüşümü ile artırarak, seyreklik yapısını daha homojen hale getirmek.

● Senkronizasyon noktalarına olan ihtiyacı azaltmak



Şekil 1. Seyrek Alt Üçgen Matrislerin Lx = b

**İÇİNDEKİLER**

GİRİŞ 1

LİTERATÜR TARAMASI 2

METOT VE YAPILACAK TESTLER 3

PROJE İLE İLGİLİ İLK RAPORDAN ARA RAPORA KADAR YAPILAN İŞLER 4

PROJE İLE İLGİLİ SONRAKİ RAPORA KADAR YAPILACAKLAR VE BEKLENEN SONUÇLAR 5

KAYNAKÇA 6

FİGÜR LİSTESİ

Tablo LİSTESİ

# TABLO LİSTESİ

Tablo 1: Cost Map’ın oluşturulması ve Budama işlemi 10

Tablo 2: Matrisin Budama İşleminden Sonraki Durumu 10

Tablo 3: Seviye Maliyeti ve Seviye Sayısında ki Değişiklik 13

# FİGÜR LİSTESİ

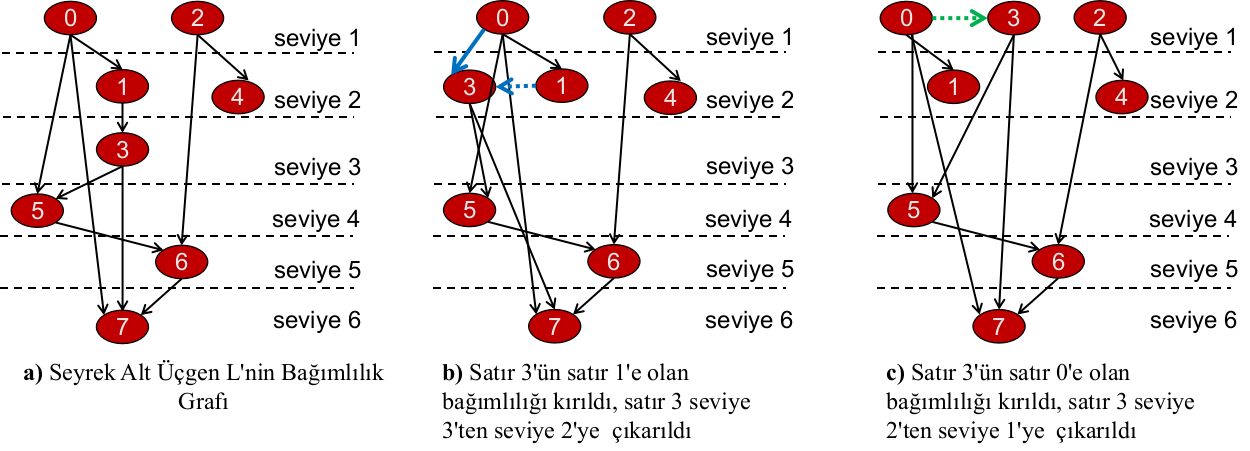
[Şekil 1: Seyrek Alt Üçgen Matrisi Lx = b](#_Toc504986067) 1

[Şekil 2: Yeniden Yazma Yöntemi](#_Toc504986067) 7

# GİRİŞ

Bu çalışmanın amacı Chainbreaker çerçevesinin efektif olarak Seyrek Alt Üçgen Çözümü için optimizasyon yapabilmesine olanak sağlayacaktır. Bunun için matrisin bağımlılık grafının mimarinin çekirdeklerine dengeli biçimde dağıtılması gerekmektedir.

Bu çalışma bağımlılık grafı üzerinde ki dönüşümleri modelleyerek bu dengelemeyi yapacaktır. Graf dönüşümü göz önüne alındığında, Chainbreaker’ın önerdiği yeniden yazma yöntemi ile matrisin paralellik derecesi az olan parçalarını dönüştürerek, bu parçaların paralellik derecesini artırmaktadır. Satır bağımlılıklarını esneterek ya da tamamen ortadan kaldırarak, daha çok çekirdek kullanımına olanak sağlar. Buna ek olarak, ince seviyeler bu yöntemler ile tamamen ortadan kaldırılarak senkronizasyon bariyeri ihtiyacını azaltmış olur.



Şekil 2. Yeniden Yazma Yöntemi

# LİTERATÜR TARAMASI

[1]’de Gauss Eliminasyonunda Markov Karar Sürecinin öğrenilmesi adına yapılan bir yaklaşımdan bahsedilmektedir. Burada özellikle Reinforcement Learning’in bir öğrenim şekli olan Q-Learning tekniği ele alınmıştır. Burada birden fazla Q-learning tekniği okuyucuya önerilmiştir. Bunlar minimum degree ordering, task scheduling and adaptive pivoting. Öğrenmeye dayalı bir algoritma olan bu Teknik ile seyrek çözücüde performansı artırmaya dayalı bir şekilde kullanılmıştır.

Bizim projemizde Seyrek alt üçgensel bir matrisler üzerinde işlem yaptığımızdan dolayı genel olarak burada Gauss Eliminasyonu ile yeniden yazma işlemlerini gerçekleştiriyoruz. [2] burada ki kitabın Chapter 3’ünde Sonlu Markov Karar Süreçlerini anlatarak bizlere bu konuda daha güzel örnekler üzerinden RL ile Markov’u beraber nasıl kullanabileceğimiz konusunda eğitmektedir.

[2]’nin Chapter 3’ünde ilk önce RL’in çalışma prensiplerinde Agent- Environment ilişkisinden bahsedilmektedir. Daha sonrasında ise Ödüller ve Cezalar üzerinden durulmaktadır. RL’de eğer istenilen işlemler sağlanıyorsa ödül sağlanmıyor ise ceza alarak ajanımız bir oyun oynar gibi eğitilmektedir. Burada verilen bir örnekte Çöp Toplama robotunun hareketleri üzerinden bir anlatım yapılmıştır. Robotumuzun şarjının 2 adet durumu vardır bunlar “high,low” ve robotumuz 3 tane işi yapabiliyor “search, wait, recharge”. Eğer robotumuzun pili “high” ise “search,wait” işlemlerini “low” ise 3 işlemide yapabiliyor, eğer “search ” işlemini yaparken şarjı biterse ceza alıyor, her topladığı çöp için ise bir adet ödül puanı alıyor. Robotun çalışma şekli Reinforcement Learning ile bu şekilde kuruluyor ve eğitiliyor. Burada Markov Karar Süreçlerini ise bu durumların koşullu olarak oluşma olasılıkları üzerinden hesaplanarak yapılıyor.

Biz ise bu örnekte olduğu gibi problemimizi bir agent-environment ilişki kurarak gerekli tüm hesaplamaları ve kurallarımızı bu agent üzerinden Markov Karar Süreçlerini kullanarak hesaplatmayı ve bu hesaplamaları kullanarak yeniden yazma işlemlerini gerçekleştirmeyi hedefliyoruz.

# METOT VE YAPILACAK TESTLER

Bu proje için geliştirdiğimiz en son algoritma 3 adet kritere sahip ve bu kriterler üzerinden gerekli işlemler yapılarak matrisimizi optimal hale getirmeyi amaçlamaktadır. Bu algoritma her bir düğümü yeniden yazma işlemini yaparken bakması gereken 3 adet kriter vardır bunlar:

1-avgCostPerLevel – Yani her yeniden yazdığımız seviyelerdeki maliyet değerimiz ortalama maliyet değerimizden büyük veya küçük mü ?

2-avgInPerRow – Ortalama maliyet değerimizden küçük seviyelerde ki ortalama düğüm sayısı. Bu değerden küçük mü büyük mü ?

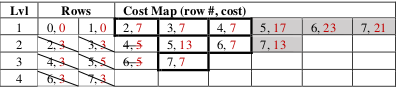
3-avgIndegrePerLevel – Ortalama maliyet değerimizden küçük seviyelerde giren düğüm sayılarının ortalaması. Bu değerden küçük mü büyük mü ?

Eğer bu değerlerden küçük ise bir üstte ki seviyeye gider ve orada bu kriterleri arar. Eğer bu kriterlerden herhangi birisi sağlanmaz ise o düğüm en son sağlanan seviyeye yazılır. Bu şekilde daha optimal bir matris elde etmek istenmektedir. Bu algoritma için lung\_2 isimli matris kullanılmaktadır.

# PROJE İLE İLGİLİ YAZ491’IN SON RAPORUNDAN YAZ492 İLK RAPORUNA KADAR YAPILAN İŞLER

En son ki raporda yazılı olan algoritma biraz geliştirilerek bu algoritma BAŞARIM 2022 konferansı için bildiri olarak gönderildi ve yakın zamanda bu bildiri yakın zamanda kabul görmüştür. Buraya gönderdiğimiz algoritma en son raporda ki algoritmanın biraz daha geliştirilmiş halidir. Bu algoritma genel olarak şu şekilde çalışmaktadır :

Algoritma ince seviyeler arasında yeniden yazma işlemini uygular. İnce seviyeler, toplam maliyeti avgLevelCost’tan daha küçük olan seviyelerdir. İnce seviyelerin en alt seviyesinden başlayarak, bu seviyeler yeniden yazdığımız seviyenin maliyeti avgLevelCost’a ulaşana kadar üst seviyelere yeniden yazılır. Bu nedenle, seviye sayısını azaltmak için kaynak seviyeleri silinir ve hedef seviyelerin maliyetleri avgLevelCost’a yüklenir. Kaynak ve hedef seviyelerine karar vermek için maliyetleri gösteren costMap adında bir tablo tutuyoruz.



Tablo 1. Cost Map’ın oluşturulması ve Budama işlemi



Tablo 2. Matrisin Budama İşleminden Sonraki Durumu

Bir satırın orjinal olarak bulunduğu seviyeden başlıyoruz ve her seviye 1’de, seviyeye kadar yeniden yazılmış gibi maliyetini hesaplıyoruz. Daha sonra bir budama işlemi ile bu seviyelerde ki suraların konumları kesinleşir.

Bu algoritmadan sonra daha efektif olması açısından yeni bir algoritma daha oluşturduk ve şuan bu algoritmanın koda dökülmesi aşamasındayız. Bu algoritmanın çalışma prensibi şu şekildedir :

Öncelikli olarak Ortalama Seviye Maliyeti (avg. level cost), Ortalama Giren Düğüm Derecesi (avg. indegree per row ), (Yeniden Yazılacak Seviyelerde) Seviye Başına Düşen Ortalama Düğüm Sayısı (avg. rows per level in levels to be rewritten) bu kriterler hesaplanır.

Daha sonrasında yeniden yazılacak seviyeler Ortalama Seviye Maliyeti’ne göre belirlenir. En alt seviyelerden başlayarak ;

a. Satırın düğüm derecesi, Ortalama Giren düğüm derecesinden küçük ise,

b. Yeniden yazılan seviyenin maliyeti, ortalama seviye maliyetinden küçük,

c. Yeniden yazılan seviyenin satır sayısı, Seviye başına düşen ortalama düğüm sayısından küçük ise,

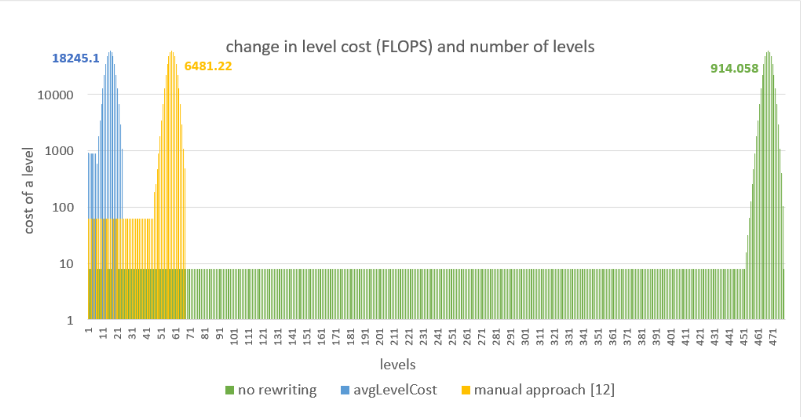
Bu kriterlerin hepsi sağlandığı sürece düğümü bir üst seviyeye taşırız. Kriterlerden herhangi biri sağlanmaz ise kriterlerin sağlandığı en son ki satıra yazılır.

# PROJE İLE İLGİLİ SONRAKİ RAPORA KADAR YAPILACAKLAR VE BEKLENEN SONUÇLAR

Final Raporuna kadar en son yazdığımız algoritmayı tamamlayı planlıyoruz daha sonrasında farklı veri setleri ile bu algoritmanın test edilmesini sağlayacağız. Bu algoritma dışında bu problemimizi Reinforcement Learning ve Markov Decision Pocess kullanarak nasıl çözebiliriz? vb. Soruların cevaplarını bulmayı planlıyoruz ve RL ve MDP’yi kullanarak bir algoritma kurmayı planlıyoruz. Bunlar üzerinden kurulan algoritmanın mantıksal olarak daha optimal bir yaklaşım ve sorunun çözümüne daha yakın olması amaçlıyoruz.

Bu problemin sonuçlarının en optimal şekilde olması istenmektedir. 1.Algoritma da aldığımız sonuçlar istenilen düzeyde olmadığı için 2.Algoritmayı yazma gereği duyduk. Aldığımız sonuçlar sonrasında 2.Algoritmayı yazma sebeplerimiz şunlardır:

* Sadece Ortalama Seviye Maliyeti üzerinden işlem yaptığımız için yeniden yazdığımız seviyelerde ki düğümlerin sayısının ortalamadan yüksek olması.
* 1.Algoritma da seviye sayımızda azalma oldu ama seviyelerde hem düğüm hemde ortalama maliyetleri arasında ciddi farklılıklar meydana geldi.



Tablo 3. Seviye Maliyeti ve Seviye Sayısında ki Değişiklik

Bu sebepler istenilen optimallik düzeyine ulaşmadığı için 2.Algoritmayı yazıp bu sebeplerinde ortadan kaldırılmasını amaçlıyoruz.

# KAYNAKÇA

Kaynakça Chicago formatında eklenmelidir.

|  |  |
| --- | --- |
| [1] | Yingshi Chen. "LearningThe Markov Decision Process In The Sparse Gaussian Elimination." arXiv: 2109.14929v1 30 Sep 2021 |
| [2] | Richard S. Sutton and Andrew G. Barto,. Reinforcement Learning. 2016 |
| [3] | Christian Schulz. "High Quality Graph Partitioning" Doktora Tezi, Karlsruher Instituts für Technologie, 2013. |
| [4] | John E. Savage and Markos G. Wloka. "Parallelism in Graph-Partitioning" Journal of Parallel and Distributed Computing 13(1991): 257-272. |
| [5] | Buse Yılmaz, Graph Transformation and Specialized code Generation For Sparse Triangular Solve(SpTRSV), 2103.11445v1, 21 Mart 2021 |
|  |  |