ÇİZGE BOYAMA PROBLEMLERİ ÜZERİNE KAPSAMLI BİR ARAŞTIRMA

A COMPREHENSİVE RESEARCH ON GRAPH COLORİNG PROBLEMS

Murat Erşen ÜNAL

Dr. Öğr. Üyesi Talha ARIKAN
Tez Danışmanı

Hacettepe Üniversitesi
Lisansüstü Eğitim-Öğretim ve Sınav Yönetmeliğinin
Matematik Anabilim Dalı için Öngördüğü
YÜKSEK LİSANS TEZİ olarak hazırlanmıştır.

2025

ÖZET

ÇİZGE BOYAMA PROBLEMLERI ÜZERINE KAPSAMLI BIR ARAŞTIRMA

Murat Erşen Ünal

Yüksek Lisans, MATEMATİK Bölümü

Tez Danışmanı: Dr. Talha ARIKAN

Haziran 2025, 101 sayfa

Çizge teorisi, yüksek teorik öneminden dolayı ve pratikte uygulama alanlarının sık olması sebebiyle matematiğin önemli araştırma alanlarından biridir. Özel olarak çizge boyama probleminin inceleneceği bu tezde; öncelikle bazı temel kavramlar verilecektir. Bunun yanı sıra çizge boyama problemi ile ilgili olarak var olan teorik alt/üst limitler ile ilgili önemli sonuçlar sunulacaktır.

Ayrcıca karmaşıklık teoresinin önemli kavramları ve temelleri tanıtılacak olup bunlar yardımıyla P ve NP probleleri formel ve informel olarak tanıtılacaktır. Bunlarla beraber NP-Tam problem sınıfı verilecek ve Cook-Levin Teoremi yardımıyla bu sınıfın boş olmadığı gösterilecektir. Bu sayede verilen herhangi bir çizgenin verilmiş bir k doğal sayısı için k renk ile uygun bir biçimde boyanıp boyanamadığını belirlemenin bir NP-tam problem olduğunu göreceğiz.

Bu problemi polinom zamanda çözen algoritmalar mevcut değildir fakat literatürde bilinen ve optimal olmasa dahi verilen çizgelerin boyanmalarına yönelik bazı önemli algoritmalar vardır. Bunlardan Greedy, DSatur, HEA ve TabuCol algoritmaları detayları ile sunulucak ve karşılaştırmaları özgün olarak rasgele üretilen çizgeler üzerinden yapılacaktır.

Çizge boyama problemi gerçek hayatta bazı uygulamalara da sahiptir. Bu uygulamalarla ilgili çeşitli detaylar sunulacatır. Bu kapsamda en önemli ve yaygın uygulaması olan sınav takvimi oluşturmak adına özgün bir tasarım yapılarak Greedy algoritması mantığıyla Python programlama dilinde bir yazılım gerçeklenmiştir.

Sonuç olarak bu tez çizge boyama ve karmaşıklık teoreisi üzerine nitelikli bir Türkçe kaynak olacaktır. Ayrıca yazılan bu program yardımıyla "Hacettepe ¨Üniversitesi, Fen Fakültesi, Matematik Bölümü" hizmetine sunulacaktır.

Anahtar Kelimeler: Çizge Boyama, NP-Tam, Cook-Levin, Algoritmalar, Kromatik Sayı

ABSTRACT

A COMPREHENSIVE RESEARCH ON GRAPH COLORING PROBLEM

Murat Erşen ÜNAL

Master of Science, Department of MATHEMATICS

Supervisor: Dr. Talha ARIKAN

June 2025, 101 pages

Graph theory is one of the most significant research areas in mathematics due to its wide range of practical applications. In this thesis, we focus specifically on the graph coloring problem. First, we introduce some fundamental concepts. Subsequently, we present key theoretical results concerning existing lower and upper bounds related to graph coloring.

Additionally, we discuss essential concepts and foundations of computational complexity theory. Using these, we formally and informally introduce the classes of P and NP problems. Furthermore, we define the class of NP-complete problems and demonstrate, via the Cook-Levin Theorem, that this class is non-empty. Consequently, we establish that determining whether a given graph can be properly colored with a given number of colors k (where k is a natural number) is an NP-complete problem.

Although no known polynomial-time algorithm solves this problem optimally, several heuristic algorithms exist in the literature for graph coloring. Among these, we examine the Greedy, DSatur, HEA, and TabuCol algorithms in detail and provide an original comparative analysis based on randomly generated graphs.

The graph coloring problem also has real-world applications. We discuss various practical implementations, with a particular focus on examination scheduling, one of its most prominent and widely used applications. In this context, we develop an original design and implement a software solution in Python using Integer Linear Programming (ILP).

In conclusion, this thesis serves as a comprehensive Turkish-language resource on graph coloring and computational complexity. Moreover, the developed software will be made available for use by the Department of Mathematics, Faculty of Science, Hacettepe University.

Keywords: Graph Coloring, NP-complete,Cook-Levin,Algorithms,Chromatic Number

TEŞEKKÜR

Beni var eden babam; Mehmet Recai ÜNAL'a, tez çalışmam boyunca bana yol gösteren danışmanım; Dr. Talha ARIKAN'a, değerli hocalarım Prof. Dr. Bülent SARAÇ'a ve Prof. Dr. Nurettin IRMAK'a, değerli katkılarından dolayı Mustafa SOLMAZ'a, özel olarak Sayın Enes Can ARSLAN'a,
ve tabii ki ailem: Şeyda ÜNAL'a,Gülşen ÜNAL'a,Ruşen ÜNAL'a, bu çalışmayı yapmamı sağladıkları için
SONSUZ KEZ TEŞEKKÜR EDERİM.

İÇİNDEKİLER

Sayfa

ÖZET i

ABSTRACT iii

TEŞEKKÜR v

ŞEKİLLER viii

SİMGELER VE KISALTMALAR ix

1. GİRİŞ 1

1.1 Tezin İçeriği 2

1.2 Katkı 3

1.3 Organizasyon 3

2. ÇİZGE BOYAMASINA GİRİŞ 5

2.1 Çizge Boyama 5

2.2 Kromatik Sayı 10

2.2.1 Üst Limitler 11

2.2.2 Alt Limitler 16

3. Hesaplanabilirlik ve Karmaşıklık Teorisine Giriş 22

3.1 Karar Problemleri ve Makineler 22

3.1.1 Alfabeler 23

3.1.2 Sonlu Otomatalar 24

3.1.3 Turing Makineleri 30

3.2 Karmaşıklık Sınıfları ve Cook-Levin Teoremi 35

4. Çizge Boyama Problemi ve NP-Tamlık 50

4.1 Temel Tanımlar 50

4.2 SAT Probleminin 3-SAT Problemine İndirgenmesi 51

4.3 3-SAT Probleminin 3-Renklendirme Problemine İndirgenmesi 52

4.4 k-Renklendirme Probleminin 3-Renklendirme Problemine İndirgenmesi 55

4.5 Sonuç 56

5. Çizge Boyama Problemi ve Bazı Algoritmalar 58

5.1 Tam Sayılı Programlama Algoritması 59
5.2 Greedy Algoritması 61
5.3 DSatur 63
5.4 Tabu-Col 64
5.5 HEA (Hibrit Evrimsel Algoritma) 66
5.6 Sonuç ve Değerlendirme 68
5.7 Çizge Boyama Problemi İçin Bazı Uygulama Örnekleri 70
5.7.1 Ders-Sınav Programı Hazırlama 70
5.7.2 Oturma Planı Hazırlama 70
6. Sınav Programı Hazırlama Algoritması 72
6.1 Uygulamanın Tasarımı ve Çalışma Prensibi 72
6.1.1 classroom_updater.py 73
6.1.2 course_updater.py 73
6.1.3 olamaz.py 73
6.1.4 gun.py 73
6.1.5 schedule.py 74
6.1.6 main.py 74
7. Tartışma ve Sonuç 75
7.1 Kenar Boyama 76
7.2 Tartışmalar 79
8. EKLER 82
8.1 EKLER-A 82

ŞEKİLLER

Sayfa

Şekil 2.1 : Çizge örneği 6
Şekil 2.2 : Çizge Boyama Örneği 7
Şekil 2.3 : Kenar üzerine kasılma 8
Şekil 3.1 : 4 ile bölünebilme kuralını tanıyan otomata 27
Şekil 3.2 : 01 ile biten dilleri bulan kararsız otomata 29
Şekil 3.3 : Kararlı Turing Makinesi 31
Şekil 3.4 : Kararsız Turing Makinesi 35
Şekil 4.1 : 3-SAT – 3-Renk denkliğini göstermede kullanılan taban çizge 53
Şekil 4.2 : 3-SAT - 3-Renk denkliğini göstermede kullanılan tabana eklenecek çizge 54
Şekil 5.1 : Greedy Algoritmasının Sudo kodu 61
Şekil 5.2 : Aynı çizgenin 2 farklı sıralamayla boyanması. 62
Şekil 5.3 : DSatur algoritmasına bir örnek. 63
Şekil 5.4 : DSatur algoritmasını sudo kodu. 64
Şekil 5.5 : 100 Köşeli Karşılaştırma 69
Şekil 5.6 : 300 Köşeli Karşılaştırma 69
Şekil 7.1 : c_0 ve c_l boyamaları. 80
Şekil 7.2 : Başka bir çizgenin kenar dönüşümü olmayan bir çizge. 80

SİMGELER VE KISALTMALAR

NP	:	Kararsız Polinomsal Zaman Problemler
P	:	Polinomsal Zaman Problemler
KTM	:	Kararsız Turing Makinesi

1. GİRİŞ

Çizge teorisi, modern matematiğin en yaygın uygulamalı dallarından biri olarak, hem teorik hem de pratik açıdan büyük önem taşımaktadır. 18. yüzyılda Leonhard Euler'in Königsberg köprüleri problemiyle ortaya çıkan çizge teorisi, günümüzde kombinatoryal matematiğin en önemli alt alanlarından biridir. Çizge teorisi, nokta ve kenar gibi basit elemanlarla karmaşık ilişkileri modelleyerek, ağ yapıları, optimizasyon problemleri, algoritma tasarımı ve veri analizi gibi birçok alanda kritik çözümler sunmaktadır. Bu teorinin önemi, sadece soyut matematiksel kavramları anlamamızı derinleştirmekle kalmayıp, bilgisayar bilimi, sosyal ağ analizi, biyoinformatik, ulaşım planlaması ve ekonomi gibi disiplinlerde pratik uygulamalar geliştirmemize olanak sağlamasından kaynaklanmaktadır. Çizge teorisindeki en ilgi çekici konulardan birisi ise çizge boyama problemleridir. Çizge boyama problemi, kombinatoryal optimizasyon alanında temel bir araştırma konusu olmakla birlikte, zaman çizelgeleme, kaynak tahsisi, harita boyama ve bilgisayar ağlarında frekans atama gibi kritik uygulamalara da çeşitli çözümler sağlamaktadır.

Aralarında kenar bulunan köşelerin farklı renge boyanması suretiyle bütün köşelerin boyanmasına basit bir çizgenin uygun boyanması denir. Bir çizgenin uygun boyanmasına olanak veren en az renk sayısı ise o çizgenin kromatik sayısı olarak adlandırılır. Özel olarak yüzeysel çizgeler (haritalar) için kromatik sayı 4'tür [1].

1852'de Francis Guthrie bir haritanın komşu bölgeler farklı renk olmak üzere 4 renk kullanılarak boyanabileceğini fark etmiş ve bu problem Francis Guthrie'nin abisi Frederick Guthrie tarafından ünlü matematikçi Augustus De Morgan'a iletilmiştir. Morgan, Arthur Cayley ve başka birçok matematikçiyle problemi tartıştıktan sonra, problemi 1878'de Londra Matematik Topluluğu'na (London Mathematical Society) sunmuştur. Fakat bu problem yaklaşık 100 yıl sonra 1976 yılında Kenneth Appel ve Wolfgang Haken tarafından bilgisayar yardımıyla kanıtlanabilmiştir. Bu sonuç, bilgisayar yardımıyla kanıtlanan ilk teorem olarak tarihe geçmiştir [1].

Bugün 4-renk problemi, bu tezin araştırma konusu olan çizge boyama probleminin özel bir hali olarak kabul edilirken, çizge boyama teorisi kendisini çok daha geniş bir spektrumda tanımlayarak liste boyama, eşlik boyama, toplam boyama gibi çeşitli varyasyonlarıyla günümüzde aktif araştırma alanlarından biri haline gelmiştir.

1.1 Tezin İçeriği

Bu tez çizge boyama problemlerinin geçtiğimiz yüzyılda ele alınış biçimini, verilen bir çizgenin kromatik sayısının bilinen bazı alttan ve üstten limitlerini ve çizge boyama probleminin NP-tam bir problem olması ile ilişkili konuları ele almaktadır. Ayrıca bazı önemli çizge boyama algoritmaları da bu tez kapsamında incelenip karşılaştırılacaktır. Bunun yanında, çizge boyama problemlerinin gerçek hayatta çözüm ürettiği bazı problemlerden kısaca bahsedilecektir. Bu bağlamda, biz de bu çalışmanın bir çıktısı olarak, Bölüm 6'da Hacettepe Üniversitesi Fen Fakültesi Matematik Bölümü için yaptığımız sınav programı uygulamasını sunacağız.

Çizge boyama probleminin çok sayıda ve kullanışlı uygulaması bulunmasının yanı sıra, bu problemi teorik açıdan önemli kılan temel faktör, problemin NP-tam sınıfında yer almasıdır. Verilen herhangi bir çizgenin kromatik sayısını polinom zamanda hesaplayabilen bir algoritmanın varlığının ispatlanması durumunda, bilgisayar biliminin en önemli açık problemlerinden biri olan

$$P \stackrel{?}{=} NP$$

sorusunun da cevabı verilmiş olacaktır. Bu kapsamlı problemin derinlemesine anlaşılabilmesi için gerekli olan otomat teorisi ve ilgili konular Bölüm 3'te ayrıntılı olarak ele alınmıştır. P ve NP karmaşıklık sınıfları arasındaki ilişkinin henüz tam olarak belirlenmemiş olması nedeniyle, çizge boyama probleminin NP-tam karakteristiği bu problemin en belirleyici özelliğini oluşturmaktadır.

1.2 Katkı

Türkçe literatürde, Polinom-Zaman (P) ve Rastgele Polinom Zaman (NP) gibi kavramları anlatan yeterli kaynak olmadığını düşündüğümüzden ve çizge boyama problemlerine olan yaygın ilgiden dolayı bu alanda kapsamlı bir çalışma yapmaya çalıştık.

Ayrıca bu çalışmanın 2. Bölümünde anlatılan, kromatik sayının bazı alttan ve üstten teorik limitleri konularını ele alan Türkçe bir kaynak olmaması da bu tezi önemli kılan unsurlardan biridir. Bunun yanında $P \stackrel{?}{=} NP$ probleminin anlaşılır ve sistematik bir şekilde anlatan Türkçe bir kaynak bulamadık. Renk boyama probleminin NP-Tam'lığını 3. ve 4. Bölümlerde adım adım göstererek bu eksikliği gidermeyi hedefledik.

5. Bölüm'de ise çizge boyamak için kullanılan algoritmaların çalışma mekanizmalarını inceleyep bu algoritmaları kendi hazırladığımız bir simülasyonda test edip bu testlerin sonuçlarını, farklı algoritmaları karşılaştırabileceği bir görsel olarak sunduk. 6. Bölüm'de ise bir gerçek hayat problemini özel bir çizge boyama problemine dönüştürüp çözmeye çalıştık. 5. ve 6. Bölümlerde yapılan bu çalışmalar ile Türkçe literatüre, çizge boyama algoritmalarını irdeleyen ve çizge boyama algoritmalarının problem çözmek için nasıl kullanılacağını örnekleyen bir kaynak sunmayı hedefledik.

1.3 Organizasyon

Bu çalışmanın ikinci kısmında, çizge boyama problemi ve bazı önemli kavramlar tanıtılmıştır. Ardından verilen herhangi bir çizgenin kromatik sayısının bazı teorik alttan ve üstten limitleri gösterilmiştir.

Çalışmanın üçüncü bölümünde, çizge boyama probleminin teorik önemini anlamak amacıyla otomat teorisinin temel kavramlarına yer verilmiştir. P ve NP gibi karmaşıklık sınıfları, ancak sağlam bir otomat teorisi altyapısı üzerine kurulduğunda tam olarak kavranabilir. Bu nedenle, çizge boyama probleminin algoritmik analizini yapmadan önce bu teorik temellerin atılması zorunlu görülmüştür. Her ne kadar bu bölüm doğrudan çizge boyama ile ilgili

içerik sunmasa da, burada ortaya konan tanımlar ve kanıtlar sonraki bölümlerin temelini oluşturmaktadır.

Dördüncü bölümde ise üçüncü bölümde tanımlanan ve kanıtlanan kavramlar kullanılarak çizge boyama probleminin NP-tam sınıfında olduğu gerçeğine, çizge boyama probleminin meşhur SAT problemine denk olduğu gösterilerek ulaşılmıştır.

Beşinci bölümde çizge boyama problemini çözmek için kabul gören bazı önemli algoritmalara detaylıca değinilmiştir. Bu algoritmların çalışma prensipleri ve çalışma zamanları detaylıca ele alınmıştır. Ayrıca bu bölümde algoritmaların performansları da değerlendirilmiştir.

Altıncı bölümde ise; bir gerçek hayat problemi olan sınav programı düzenlemeyi, çözmek için bu tez bağlamında tasarlanan uygulamanın tasarım ve çalışma süreçleri detaylıca anlatılmıştır.

2. ÇİZGE BOYAMASINA GİRİŞ

Bu bölümde çizge boyama, kromatik sayı gibi kavramlar tanımlanmıştır. Ayrıca verilen bir çizgenin kromatik sayısı için bilinen önemli alt ve üst limitleri gösterilmiştir.

2.1 Çizge Boyama

Farklı türlerde çizgeler vardır ancak bizim bu çalışmada inceleyeceğimiz çizgeler basit, yönsüz çizgeler olacaktır. Bir çizgeyi köşeler ve bu köşeler arasındaki kenarlar olarak tanımlayacağız. Bir kenar iki köşeyi birbirine bağlar ve basit yönsüz çizgelerde kenarın bir yönü, değeri veya şekli yoktur sadece bu kenarın olup olmaması ile ilgilenilir. Ayrıca bu çalışmada aksi belirtilmedikçe bir çizge dendiği zaman okurun aklına bağlı bir çizge gelmelidir, yani bir köşeden başlayıp kenarlar boyunca hareket ederek bütün köşere gidilebilen çizgeleri inceliyoruz.

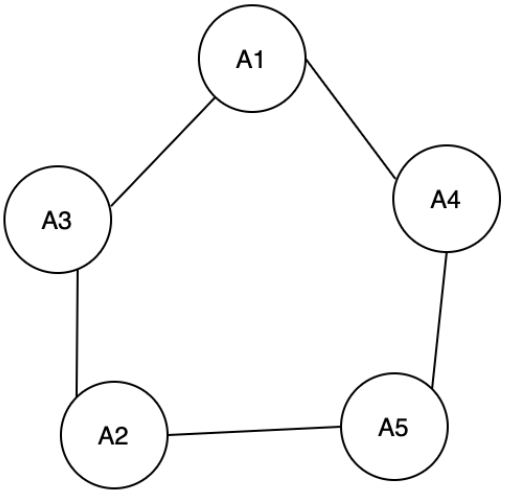
Basit yönsüz çizgeleri günlük hayattaki birçok olguyu modellemek için kullanabiliriz. Örneğin bir okulda A1, A2, A3, A4 ve A5 dersleri olsun. Bu derslerin hangilerinin ortak öğrenciler tarafından alındığını bir çizge kullanarak modelleyebiliriz. Bu dersler oluşturacağımız çizgenin köşeleri olacak. Yani çizgemizde A1, A2, A3, A4 ve A5 köşeleri olacak. Ve ortak öğrenci içeren iki ders arasına bir kenar koyacağız. A1'in A3 ve A4 ile ortak öğrencisi, A2'nin A3 ve A5 ile ortak öğrencisi, A4'ün A5 ile ortak öğrencisi varsa oluşturacağımız çizge Şekil 2.1 deki gibi olur.

Şimdi basit ve yönsüz bir çizgenin genel tanımını verelim.

Tanım 2.1.1. Bir çizgeyi $G(V, E)$ olarak ifade edeceğiz, burda G çizgenin ismi, V çizgenin köşeler kümesi ve E ise çizgenin kenarlar kümesidir.

Bazı durumlarda çizgelerin parameterelerini ayırt edebilmek için G çizgesinin köşeleri ve kenarları sırasıyla $V(G)$ ve $E(G)$ olarak gösterilir.

Tanım 2.1.2. $G(V, E)$ bir çizge olmak üzere $\forall u, v \in V$ için u köşesinden v köşesine kenarlar boyunca hareket ederek gidilebiliyorsa, $G(V, E)$ çizgesine bağlı çizge denir.



Şekil 2.1 : Çizge örneği

Burda E'nin V'nin ikili alt kümelerinden oluşan bir küme olduğunu kolayca gözlemlenebilir. $u, v \in V$ olmak üzere; u ve v köşeleri arasındaki kenarı uv olarak ifade edeceğiz. Örneğin Şekil 2.1'de verilen çizgenin köşeler kümesi $V = \{A1, A2, A3, A4, A5\}$ ve kenarlar kümesi $\{A1A3, A3A2, A2A5, A5A4, A4A1\}$ olur.

Yukarıda 5 dersi ortak öğrenci içerme durumuna göre basit çizgeler kullanarak modelledik. Artık çizge boyama probleminin tanımını verebiliriz. Ama öncesinde, çizge boyama probleminin neleri modellemek için kullanılabileceğini bir örnek ile aktaracağız.

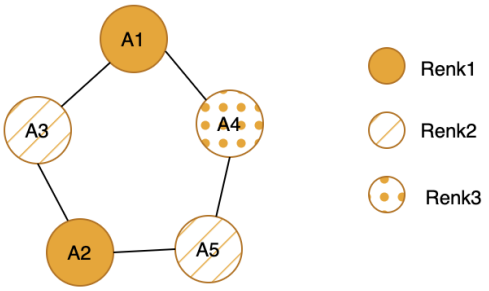
Şekil 2.1'de modellediğimiz dersleri ele alalım. Bu derslerin bazıları aynı öğrenciler tarafından alınmaktadır. Örneğin hem A1 hem A3 dersini alan öğrenciler var. Bu yüzden A1 ve A3 derslerinin sınav saatleri kesişmemelidir. Ortak öğrencisi olan derslerin sınav saatleri kesişmeyecek biçimde her derse bir sınav saati atanması bir çizge boyama problemi olarak görülebilir.

Daha genel olarak aşağıdaki tanımları verebiliriz.

Tanım 2.1.3. Verilen basit bir çizgenin komşu olan her köşesi farklı renkte olacak şekilde boyanmasına, uygun boyama denir.

Tanım 2.1.4. Bir çizgenin köşelerinin k renk kullanılarak boyanmasına ise k-boyama adı verilir.

Şekil 2.2'de bir çizgenin 3 renk kullanılarak uygun boyanması gösterilmiştir. Okur dikkat ederse bu şekilde gösterilen çizgenin 3'den daha az renkle uygun boyanamayacağını görecektir.



Şekil 2.2 : Çizge Boyama Örneği

Bu noktada çok bariz olan bir gözlem vardır; her çizge nokta sayısı kadar renkle boyanabilir. Burada önemli olan verilen bir çizgenin en az kaç renk ile boyanabileceğini veya verilen bir k doğal sayısı için k-renkle uygun bir boyama olup olmayacağını belirleyebilmektir.

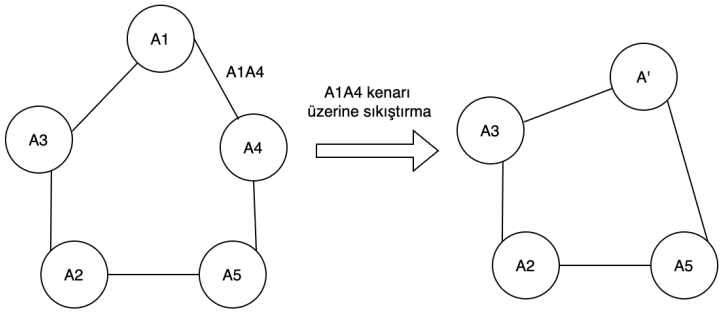
Bir çizgenin k farklı renkle kaç farklı biçimde boyanacağı ise o çizgenin *kromatik polinomu* olarak tanımlanır. Yani $P(G, k)$, G çizgesinin k renkle kaç farklı şekilde uygun boyanacağını ifade eder. Fakat öncelikle bir G çizgesinin k farklı renkle kaç farklı biçimde boyanacağının sayısının k 'nın polinomu türünden bir fonksiyon olduğunu göstermemiz gerekiyor, bunu bir teorem yardımıyla yapacağız. Böylece okur hem farklı boyamanın ne olduğunu hem de kromatik polinom adı ile tanımlanan bu göstergenin gerçekten bir polinom olduğunu kavrayabilecektir. Ancak bunu göstermeden aşağıda vereceğimiz önemli tanımlara ihtiyacımız var. Daha sonra bir önermeyi kanıtlayıp, asıl teoremi yani $P(G, k)$'nin polinomsal bir fonksiyon olduğunu göstereceğiz.

Tanım 2.1.5. G çizgesinin kromatik polinomu, $P(G, k)$, G çizgesinin k farklı renkle kaç farklı biçimde boyanabileceğini belirtir.

Tanım 2.1.6. $G(V, E)$ bir çizge ve $e \in E$ olmak üzere; $G - e$ çizgesi, G çizgesinden e kenarının çıkarılmasıyla elde edilen çizgedir.

Tanım 2.1.7. $G(V, E)$ bir çizge, $e \in E$ ve $u, v \in V$ için e kenarı u ve v köşeleri arasında olsun. O zaman G çizgesinin e kenarı üzerine kasılması, kısaca G/e ile gösterilsin, G çizgesinden e kenarının silinmesi, u ve v köşelerinin birleştirilip w köşesinin oluşturulması ve w köşesinden u ve v nin komşu olduğu bütün köşelere kenar çizilmesiyle elde edilen çizgedir.

Aşağıdaki şekilde, Şekil 2.1 de verilen çizgenin A1 ve A4 arasındaki kenar üzerine kasılması ile elde edilen çizge gösterilmiştir. Burada yeni köşe A′ olarak belirtilmiştir.



Şekil 2.3 : Kenar üzerine kasılma

Önerme 2.1.8. $G(V, E)$ bir çizge ve $P(G, k)$ bu çizgenin kromatik polinomu olsun. $e \in E$ olmak üzere her $G - e$ çizgesi ve her $k \in \mathbb{N}$ için;

$$P(G - e, k) = P(G, k) + P(G/e, k)$$

eşitliği sağlanır.

Kanıt. $P(G - e, k)$ sayısı $G - e$ çizgesinin k farklı renkle kaç farklı şekilde uygun boyanacağının sayısı olduğunu hatırlayalım. $e \in E$ kenarı $u, v \in V$ olmak üzere, u ve v köşeleri arasındaki kenar olsun. O halde bu farklı boyamaları;

- G çizgesindeki uygun olan boyamalar,
- G çizgesinde uygun olmayan boyamalar,

olmak üzere ikiye ayırabiliriz. Yani; B, u ve v dışında bütün komşuların farklı renkle boyandığı uygun olmayan bütün k-boyamalarının sayısını göstermek üzere

$$P(G-e,k) = P(G,k) + B$$

eşitliği sağlanır.

$P(G/e,k)$ ise u ve v 'nin aynı renge boyandığı kalan bütün komşu köşelerin farklı renge boyandığı k-boyamaların sayısı olduğundan, her $G(V,E)$ ve $e \in E$ için;

$$P(G-e,k) = P(G,k) + P(G/e,k)$$

eşitliği tüm k doğal sayıları için doğru olup ispat tamamlanır. □

Şimdi önemli teoremi ifade edebiliriz.

Teorem 2.1.9. Verilen bir G çizgesi için $P(G,k)$ değeri k'nın polinomu cinsinden bir fonsiyon olarak ifade edilir.

Kanıt. Bu teoremi kanıtlamak için verilen G çizgesinin kenar sayısı üzerine tümevarım yapacağız. İlk olarak G çizgesinin n tane köşesi ve 0 adet kenarı olsun. Bu durumda G çizgesini her köşeyi k farklı renkle boyayarak toplamda k^n farklı şekilde boyayabiliriz.

Şimdi kabul edelim ki, kenar sayısı n den az olan herhangi bir çizge H için $P(H,k)$ k'nın bir polinomu olsun. G, n tane kenarı olan bir çizge olsun, Önerme 2.1.8'den biliyoruz ki

$$P(G-e,k) = P(G,k) + P(G/e,k)$$

eşitliği sağlanır. $G - e$ ve G/e çizgelerinin kenar sayısı G çizgesinin kenar sayısından bir azdır yani $n - 1$ tane kenara sahiptirler. Dolayısıyla $P(G - e, k)$ ve $P(G/e, k)$ fonksiyonları tümevarım hipotezimiz gereği k'nın birer polinomudurlar. Sonuç olarak iki polinomun toplamı da bir polinom olacağından $P(G, k)$ de k'nın bir polinomu olacaktır. □

Şimdiye kadar uygun çizge boyamanın ne olduğundan ve kromatik polinomdan söz ettik. Ancak biz bu çalışmada genel olarak bir çizgenin en az kaç farklı renkle uygun boyanabileceği üzerine çalışacağız. Bir çizgenin en az kaç farklı renkle uygun boyanabileceği o çizgenin *kromatik sayısı* adı verilir.

Tanım 2.1.10. Verilen bir $G = (V, E)$ çizgesinin uygun boyanmasını sağlayan en az renk sayısına o çizgenin kromatik sayısı denir ve $\chi(G)$ ile gösterilir.

Bir çizgenin kromatik sayısı ile kromatik polinomunun 0 değerinden büyük olduğu ilk pozitif tam k değeri aynıdır. Bu eşitlik kromatik sayının ve kromatik polinomun tanımlarından kolayca görülebilmektedir.

2.2 Kromatik Sayı

Bir çizgenin kromatik sayısının ne olduğundan bahsettik. Tanımından görüleceği üzere çizge boyama problemlemlerinde kromatik sayı çok önemli bir parametredir. Bu bölümde ise bir çizgenin kromatik sayısının alt ve üst limitlerine değineceğiz. Verilen bir çizgenin kromatik sayısının bulunması kolay bir problem değildir, hatta çizge yeterince büyükse oldukça zor bir problem olacaktır. Üçüncü bölümde kromatik sayı bulma probleminin zorluk derecesi detaylı olarak incelenecektir. Bu alt bölümde sadece verilen bir çizgenin kromatik sayısının bazı alt ve üst limitlerine değineceğiz. Bu sonuçlara geçmeden önce aşağıdaki tanımlara ihtiyaç olacaktır.

Tanım 2.2.1. Bir $G(V, E)$ çizgesinde bir köşenin derecesi o noktaya bağlı toplam kenar sayısıdır ve $v \in V$ olmak üzere v köşesinin derecesi Δv ile gösterilir.

Tanım 2.2.2. Bir $G(V, E)$ çizgesinin derecesi G çizgesinde bulunan en yüksek dereceli köşenin derecesine eşittir ve $\Delta(G)$ ile gösterilir. Yani

$$\Delta(G) = \max(\{\Delta v : v \in G\}).$$

Bir sonraki alt bölümde literatürde bilinen çeşitli üst sınırlar ile ilgili sonuçlar verilecektir.

2.2.1 Üst Limitler

En bariz üst sınır bir çizgedeki köşe sayısıdır. Bir çizgenin bütün köşelerini farklı renge boyarsak, komşu olan bütün köşeleri de farklı renge boyamış oluruz. Yani bir çizgenin kromatik sayısı en fazla o çizgedeki köşe sayısı kadardır. Ancak bu çok barizdir ve üzerine konuşulacak pek bir şey yoktur. Ayrıca yeterince çok köşeye sahip bir çizge için pratikte bir anlam ifade etmeyecektir.

Bir çizgenin derecesi de kromatik sayısına bir üst sınır olabilir. Aşağıdaki teorem bu gerçeği vermektedir.

Teorem 2.2.3. Her $G(V, E)$ çizgesi için $\chi(G) \leq \Delta(G) + 1$ eşitsizliği sağlanır.

Kanıt. $n = \Delta(G) + 1$ olsun ve renk kümemiz $C = \{c_1, c_2, \cdots, c_n\}$ olsun. Şimdi sadece C kümesindeki renkleri kullanarak G çizgesinin uygun bir şekilde boyanabileceğini göstereceğiz. Böylece $\chi(G) \leq \Delta(G) + 1$ eşitsizliğini göstermiş olacağız. G çizgesinin köşelerini herhangi bir sıraya dizelim. Örneğin; $a = |V|$ olmak üzere

$$v_1, v_2, \cdots, v_a$$

sıralamasını ele alalım. Bu sırayla her köşeyi, C kümesinden seçeceğimiz renklerle şöyle boyayabiliriz;

Bir v köşesi komşularının boyanmadığı C kümesindeki en küçük indeksli renkle boyanır. Çizgede en çok komşusu olan köşenin $n - 1$ komşusu olduğundan bu boyanma sırasında

bütün köşeler için C kümesinden uygun bir renk seçilebilecektir. Yani C kümesindeki renkler ile verilen çizge için bir uygun boyama mümkündür. Böylece G çizgesi $\Delta(G)+1$ tane farklı renk kullanılarak uygun bir biçimde boyanabilecek olup ispat tamamlanır. □

Bazı özel durumlarda bu eşitsizliktan daha güçlü olan bir eşitsizliği verebiliriz. Fakat bunun için öncelikle bu özel tipteki çizgelerin tanımlarını verelim.

Tanım 2.2.4. $G(V,E)$ bir çizge olsun, $\forall\, u,v \in V$ için $u \neq v \Rightarrow uv \in E$ ise G' ye tam çizge denir. Ayrıca n tane köşesi olan tam çizgeler K_n ile gösterilir.

Tanım 2.2.5. Bütün köşelerinin derecesi 2 olan ve tek sayıda köşe içeren çizgelere tek tekerlek adı verilir.

Tanım 2.2.6. Bütün köşelerinin derecesi 2 olan ve çift sayıda köşe içeren çizgelere ise çift tekerlek adı verilir.

Tanım 2.2.7. İki köşesinin derecesi 1 olan ve derecesi en fazla 2 olan bağlı çizgeye patika denir.

Tanım 2.2.8. $G(V,E)$ bağlı bir çizge olsun ve $\forall u,v \in V$ için $|uv|$ değeri u köşesinden v köşesine giden ve en az köşe içeren patikanın toplam köşe sayısıdır.

Tanım 2.2.9. $G(V,E)$ bir çizge olmak üzere herhangi bir $U \subseteq V$ ve

$$F = \{ab \mid a,b \in U \text{ ve } ab \in E\} \subseteq E$$

kenarlar kümesi için $H(U,F)$ çizgesine $G(V,E)$ çizgesinin bir alt çizgesi denir.

Şimdi aşağıdaki önemli sonucu verebiliriz.

Teorem 2.2.10 (Brook Teoremi)**.** $G(V,E)$ bir çizge olsun. Eğer G tam veya tek tekerlek tipinde değil ise;

$$\chi(G) \leq \Delta(G)$$

eşitsizliği sağlanır.

Brook Teoreminin kanıtında ihtiyacımız olan birkaç tanımı verelim.

Tanım 2.2.11. Verilen bir G çizgesinden hangi $k-1$ köşe ve o köşeye bağlı bütün kenarlar silinirse silinsin eğer G hala bağlı kalıyorsa G'ye k-bağlı çizge adı verilir.

Tanım 2.2.12. $G(V,E)$ bir çizge olsun. $U \subseteq V$ olmak üzere;

$$\forall v_1, v_2 \in U \quad \text{için} \quad v_1v_2 \in E$$

oluyor ise köşeleri U kümesinden gelen G çizgesinin alt çizgesine bir klik denir.

Tanım 2.2.13. $G(V,E)$ bir çizge olsun. G çizgesinden elde edilen en büyük kliğin köşe sayısına G'nin klik sayısı denir ve $\omega(G)$ ile gösterilir.

Brook Teoremini aşağıdaki iki önermeyi kanıtladıktan sonra kanıtlayabileceğiz.

Önerme 2.2.14. Herhangi bir $G(V,E)$ çizgesi için $V = \{v_1, v_2, ..., v_n\}$ kümesi köşeler kümesi olsun. Bu köşeler;

$$\left|v_{j_n}v_{j_j}\right| \leq \left|v_{j_n}v_{j_i}\right| \forall i,j \text{ öyle ki; } i \leq j$$

eşitsizliği sağlanacak şekilde $V = \{v_{j_1}, v_{j_2}, v_{j_3}, ...v_{j_n}\}$ olarak sıralanabilir.

Kanıt. $G(V,E)$ çizgesi bağlı olduğundan

$$\forall u, v \in V \text{ için } |uv| < \infty$$

olur. Şimdi herhangi bir $u \in V$ alalım ve $\forall u' \in V$ için $|uu'|$ değerini hesaplayalım. Bu değerleri küçükten büyüğe sıralayacak şekilde belirleyen u' köşelerini sıraladığımız durumda, istenilen

$$V = \{v_{j_1}, v_{j_2}, v_{j_3}, ...v_{j_n}\}$$

sırasını elde ederiz ve ispat tamamlanır. □

Önerme 2.2.15. Derecesi en az 3 olan ve tam çizge olmayan herhangi bir 2-bağlı G çizgesinde, $xy, xz \in E$, $yz \notin E$ ve $G - y - z$ bağlı olacak şekilde x, y ve z köşeleri vardır.

Kanıt. Bu önermeyi kanıtlarken G için 2 farklı durumu inceleyeceğiz.

Birinci durumda G'nin 3-bağlı olduğunu ikinici durumda G'nin $3-$bağlı olmadığını varsayacağız.

İlk olarak kabul edelim ki G 3-bağlı bir çizge olsun. G tam çizge olmadığından $yz \notin E$ olacak şekilde $y, z \in V$ iki köşe vardır. G bağlı çizge olduğundan y köşesinden z köşesine giden en kısa patika vardır.

$$y = x_1, x_2, x_3, \ldots, x_n = z$$

y köşesinden z köşesine giden en kısa patika olsun. Bu durumda $yx_2, x_2x_3 \in E$ olduğu açıktır, ayrıca bu yol en kısa yol olduğundan $yx_3 \notin E$ olduğu da aşikardır. Ayrıca G 3-bağlı olduğundan $G - y - x_3$ çizgesi de bağlı bir çizge olacaktır. Böylelikle G'nin 3-bağlı olduğu durum için önerme kanıtlanmış olur.

Şimdi ise G'nin 3-bağlı olmadığı durumu göz önüne alalım. $G(V, E)$ çizgesi 2-bağlı ama 3-bağlı olmayan bir çizge olsun. Dolayısıyla $u, v \in V$ olmak üzere $G - u - v$ bağlı olmayacak şekilde bir u, v ikilisi vardır. Şimdi $G - v$ çizgesine bakalım; bu çizgeden u köşesini çıkarırsak G çizgesi en az 2 farklı çizgeye ayrılır. Bunlara sırasıyla C_1 ve C_2 altçizgeleri diyebiliriz. C_1 çizgesinden hiçbir köşe C_2 çizgesinden bir köşeyle bağlı değildir. Fakat $G - v$ çizgesi bağlı olduğundan C_1 ve C_2 çizgelerinin en az birer köşeleri u köşesine bağlıdır. Yani $a_1u \in E$, $a_2u \in E(G)$ ve $a_1a_2 \notin E(G)$ olacak şekilde $a_1 \in V(C_1)$ ve $a_2 \in V(C_2)$ köşeleri mevcuttur. Şimdi $G - a_1 - a_2$ çizgesinin bağlı olduğunu gösterirsek önermeyi kanıtlamış oluyoruz. G çizgesi 2-bağlı olduğundan $G - a_1$ ve $G - a_2$ çizgeleri bağlı çizgelerdir. C_1 ve C_2 alt çizgeleri arasında kenar olmadığından ve G çizgesi 2-bağlı olduğundan a_1 ve a_2 köşelerinin silinmesi G çizgesinde ayrık bir parça yaratamaz. Böylelikle ispat tamamlanmış olur. □

Artık Brook Teoremi'nin ispatını verebiliriz.

Brook Teoremi'nin ispatı. Bu ispatta çelişki bulma yöntemini kullanacağız. Özel olarak $\Delta(G) = 0$ ve $\Delta(G) = 1$ durumlarında G sırasıyla K_1 ve K_2'ye eşit olacağından bu durumlar teoremin hipotezlerini sağlamamaktadır. Dolayısıyla bu durumlarda incelenmesi gereken bir gerçek yoktur.

Kabul edelim ki $\Delta(G) = 2$ olsun. Bu durumda G tek tekerlek, çift tekerlek veya patika olabilir. Eğer G tek tekerlek ise teorem tarafından içerilmemektedir, çift tekerlek veya patika ise herhangi bir köşeden başlayarak herhangi bir yönde hareket edip her köşeyi bir önceki boyalı köşenin boyanmadığı renge boyayarak 2 farklı renge boyayabiliriz. Böylelikle bu durum için de ispatlanmış olur.

Şimdi ise $\Delta(G) \geq 3$ durumunu göz önüne alalım. Kabul edelim ki $H(V', E')$ çizgesi $\Delta(H) \geq 3$ olmak üzere teoremi sağlamayan en küçük ters örnek olsun. Yani H çizgesi tek tekerlek veya tam çizge olmasın ayrıca

$$\chi(H) = \Delta(H) + 1$$

olsun. Eğer H 2-bağlı değilse; $H - u$ çizgesi bağlı olmayacak şekilde en az bir $u \in V(H)$ vardır. $H - u$ bağlı olmadığı için en az iki farklı ve ayrık çizgeden oluşmaktadır. Bunlara C_1 ve C_2 çizgeleri diyelim. $\Delta(C_1) \leq \Delta(H)$ ve $\Delta(C_2) \leq \Delta(H)$ olduğundan ve H çizgesi Brook teoreminin en küçük ters örneği olduğundan C_1 ve C_2 çizgelerini sırasıyla $\Delta(C_1)$ ve $\Delta(C_2)$ tane renkle boyayabiliriz. Ayrıca boyadığımız C_1 ile C_2 çizgeleri arasında kenar olmadığından u köşesini de $\Delta(H) - \max(\Delta(C1), \Delta(C_2))$ renkle boyayabileceğimizden H çizgesini $\Delta(H)$ farklı renkle uygun bir şekilde boyayabiliriz. Ama bu durum H'nin teoreminin ters örneği olduğuyla çelişeceğinden, H 2-bağlı olmak zorundadır.

Diğer taraftan $V(H) = \{v_1, v_2, ..., v_n\}$ olsun. H 2-bağlı ve $\Delta(H) \geq 3$ olduğundan Önerme 2.2.14 ve 2.2.15'u kullanarak; H'nin köşelerinin sıralamasını $v_1v_2 \notin E(H)$, $v_1v_n, v_2v_n \in E(H)$ ve $3 \leq i < j < n$ olmak üzere; $|v_nv_i| \geq |v_nv_j|$ olacak şekilde belirleyebiliriz. Şimdi H'nin köşerini sırasıyla $R = \{r_1, r_2, ..., r_{\Delta(H)+1}\}$ renklerini kullanarak her köşeyi kendi

sırasında R kümesindeki en düşük indeksli renkle boyayacağız. v_1 ve v_2 köşeleri arasında kenar olmadığından her ikisini de r_1 rengiyle boyayabiliriz. $\forall\ 2 < j < n$ için v_j köşesinin en az bir komşusunun $\{v_{j+1}, v_{j+2}, v_{j+3}, \ldots, v_n\}$ kümesinde olduğu yukarıda yapılan sıralamayla garanti altına alınmıştır. Dolayısıyla $\forall\ 1 \leq j < n$ için v_j köşesi R kümesinin indeksi $\Delta(H)$'ye kadar olan elemanlarından biriyle boyanabilecektir. v_n'in iki komşusu (v_1, v_2) aynı renkte boyandığı için ve en fazla $\Delta(H)$ komşusu olduğu için v_n köşesini de R kümesinin indeksi $\Delta(H)$'ye kadar olan elemanlarından biriyle boyayabiliriz. Dolayısıyla H yi en fazla $\Delta(H)$ renk kullanarak boyanabilir.

Böylelikle teoreminin olabilecek en küçük karşıt örneğini kullanarak bir çelişki elde etmiş olduk. Bu şekilde ispat tamamlanmış olur. □

2.2.2 Alt Limitler

Bir önceki bölümde verilen bir $G(V, E)$ çizgesi için bu çizgenin kromatik sayısı $\chi(G)$'nin bazı üst sınırlarını inceledik. Bu bölümde ise verilen bir G çizgesi için $\chi(G)$ sayısının alt sınırlarının en göze çarpanını inceleyeceğiz ve bu sınırla ilgili ilginç bir teorem kanıtlayacağız. Bu teoremi kanıtlarken rastgele çizgeler konusunu da giriş yapmış olacağız. Rastgele çizgeleri daha sonra 4. ve 5. Bölümlerde de kullanacağız.

Bir önceki bölümde verilen bir $G(V, E)$ çizgesinin klik sayısını tanımlayıp $\omega(G)$ olarak göstermiştik. Hatırlayacak olursak eğer $U \subseteq V$ olmak üzere $\forall a, b \in U$ ve $a \neq b \implies ab \in E$ sağlanıyor ise U çizgesine G çizgesinde bir klik olduğunu belirtmiştik. Tanımlardan rahat bir şekilde gözlemlenebileceği gibi herhangi bir $G(V, E)$ çizgesi için

$$\chi(G) \geq \omega(G)$$

eşitsizliği geçerlidir. Fakat verilen bir $G(V, E)$ çizgesinin $\omega(G)$ değerini belirlemek aynı çizgenin $\chi(G)$ değerin belirlemekten daha kolay bir problem değildir. Burda asıl problem:

herhangi bir $G(V, E)$ çizgesi için:

$$f(\omega(G)) \geq \chi(G)$$

eşitsizliğini sağlayan bir f fonksiyonun olup olmadığını belirlemektir.

Özetle bir G çizgesi için $\omega(G)$ değeri, $\chi(G)$ değeri için bir alt sınırdır, peki $\omega(G)$ değerini bildiğimiz bir G çizgesinin $\chi(G)$ değerinin olabileceği en yüksek değere dair bir şey söyleyebilir miyiz? Bu soru için akla gelen ilk cevap "evet" gibi gözükmektedir. Fakat Paul Erdös tarafından kanıtlanan aşağıda vereceğimiz teorem sezgilerin yanıldığını göstermektedir.

Erdös'ün $\omega(G)$ değerinin $\chi(G)$ değerine dair bir üst limit belirtemeyeceğini gösterdiği teoremi vermeden önce bu teoremde kullanılan birkaç tanımı ve önermeyi verelim.

Tanım 2.2.16. $G(V, E)$ bir çizge ve $a, b, c \in V$ olmak üzere, $ab, ac, bc \in E$ ise $\{a, b, c\}$ kümesine G çizgesinde köşeleri a, b, c olan üçgen adı verilir.

Tanım 2.2.17. Herhangi bir G çizgesi için eğer

$$\omega(G) < 3$$

ise G çizgesine üçgensiz bir çizge denir.

Tanım 2.2.18. $G_{n,p}$ çizesi n adet köşesi olan ve herhangi iki köşesi arasında sabit bir p olasılıkla kenar olan rastgele çizge olarak tanımlanır.

Burada tanımlanan rastgele çizgeleri 5. Bölümde çizge boyama algoritmlarını karşılaştırırken kullanacağız. Rastgele çizgeler için daha detaylı bilgilere [2] referansından ulaşılabilir.

Tanım 2.2.19. $G(V, E)$ bir çizge ve $U \subseteq V$ olmak üzere, eğer

$$\forall a, b \in U \implies ab \notin E$$

şartı sağlanıyor ise U'ya G çizgesinde bir sabit küme denir.

Önerme 2.2.20. $G(V, E)$ bir çizge olsun ve $|V| = n$ olsun. k bir doğal sayı olmak üzere eğer G'nin $\lceil \frac{n}{k} \rceil$ büyüklüğünde bir sabit kümesi yoksa $\chi(G) > k$ olacaktır.

Kanıt. Önermeyi tersini kullanarak ispatlayacağız. Yani eğer $\chi(G) \leq k$ ise G'nin $\lceil \frac{n}{k} \rceil$ büyüklüğünde bir sabit kümesi olduğunu göstereceğiz. G'nin boyanmasında aynı renkle boyanan köşelerin oluşturduğu kümenin bir sabit küme olduğu açıktır. Dolayısıyla G'nin $\lceil \frac{n}{\chi(G)} \rceil$ büyüklüğünde bir sabit kümesi vardır. Ayrıca

$$\left\lceil \frac{n}{\chi(G)} \right\rceil \geq \left\lceil \frac{n}{k} \right\rceil$$

olduğundan, G'nin $\lceil \frac{n}{k} \rceil$ büyüklüğünde bir sabit kümesi olup ispat tamamlanır. □

Önerme 2.2.21. X bir rastgele değişken ve $t \in \mathbb{R}$ olsun. Eğer

$$E(X) \leq t$$

ise $Pr(X \leq t) > 0$ olur.

Olasılık teorisinde sıkça kullanılan aşağıdaki önermeyi hatırlayalım.

Önerme 2.2.22 (Markov Eşitsizliği)**.** X negatif olmayan bir rastgele değişken olsun. O halde

$$Pr(X \geq t) \leq \frac{E(X)}{t}$$

eşitsizliği sağlanır.

Yukarıda verilen önermelerin ispatları ve detaylar için [3] referansı incelenebilir.

Şimdi Erdös'ün bahsettiğimiz teoremini verebiliriz.

Teorem 2.2.23. Her $k \geq 1$ tam sayısı için $\chi(G) > k$ olacak şekilde bir $G(V, E)$ üçgensiz çizgesi vardır.

Kanıt. Verilen her k pozitif tam sayısı için $\chi(G) > k$ olacak şekilde üçgensiz bir $G(V, E)$ çizgesi oluşturacağız. Önerme 2.2.20'de gösterildiği üzere, $n = |V|$ olmak üzere G'nin büyüklüğü $\lceil \frac{n}{k} \rceil$ olan bir sabit kümesi yoksa

$$\chi(G) > k$$

eşitsizliğini göstermiş oluruz.

Şimdi $p = n^{\frac{-2}{3}}$ ve n, bir $2k$'nın bir katı olmak üzere; $G'_{n,p}$ rastgele çizgesini tanımlayalım. Bu rastgele G' çizgesi üzerine iki tane rastgele değişken tanımlayacağız. İlk olarak I rastgele değişkeni G' rastgele çizgesinde köşe sayısı $\lceil \frac{n}{2k} \rceil$ olan sabit kümelerin sayısını belirtsin. Dolayısıyla

$$\begin{aligned} E(I) &= \binom{n}{\lceil \frac{n}{2k} \rceil} \cdot (1-p)^{\binom{\lceil \frac{n}{2k} \rceil}{2}} \\ &< \binom{n}{\lceil \frac{n}{2k} \rceil} \cdot (1-p)^{\binom{\frac{n}{2k}}{2}} \end{aligned}$$

eşitsizliği elde edilir. p pozitif olduğundan $1 - p < e^{-p}$ olacaktır. Ayrıca $\binom{n}{\lceil \frac{n}{2k} \rceil} < 2^n$ olduğundan

$$E(I) < 2^n \cdot e^{-p \cdot \frac{\frac{n}{2k} \cdot (\frac{n}{2k} - 1)}{2!}} = 2^n \cdot e^{-p \cdot \frac{n \cdot (n-2k)}{8 \cdot k^2}}$$

olur. Şimdi $p = n^{\frac{-2}{3}}$ değerini yerine yazarsak

$$E(I) < 2^n \cdot e^{\frac{-n^{\frac{4}{3}}}{16 \cdot k^2}}$$

eşitsizliği elde edilir. Şimdi $m \in \mathbb{N}$, $(\frac{2}{e})^m < \frac{1}{2}$ eşitsizliği sağlayan en küçük doğal sayı ve $n \geq max(m, 2^{12} \cdot k^6)$ olsun. O halde

$$E(I) < \frac{1}{2}$$

olur. Markov eşitsizliğinden;

$$Pr(I \geq 1) = Pr(I > 0) < \frac{1}{2} \tag{1}$$

elde edilir. (Burda I'nın negatif olmayan bir rastgele değişken olduğu gözden kaçmamalıdır.)

Şimdi kanıtımızda kullanacağımız diğer rastgele değişkeni tanımlayabiliriz. T rastgele değişkeni $G'_{n,p}$ çizgesindeki üçgenlerin sayınını belirtsin. Dolayısıyla

$$E(T) = \binom{n}{3} \cdot p^3 < \frac{n^3}{3!} \cdot n^{\frac{-2}{3}3} = \frac{n}{6}$$

eşitsizliği sağlanır.

Markov eşitsizliğinden

$$Pr(T \geq \frac{n}{2}) < \frac{1}{3} \tag{2}$$

durumu elde edilir. (1) ve (2) eşitsizlikleri kullanılarak

$$Pr(T \geq \frac{n}{2}) + Pr(I \geq 1) < 1$$

eşitsizliği elde edilir. Sonuç olarak

$$Pr(I = 0 \cap T < \frac{n}{2}) > 0$$

olur. Öyleyse $|V''| = n$ olmak üzere öyle bir $G''(V'', E'')$ çizgesi vardır ki; G'' çizgesindeki üçgen sayısı $\frac{n}{2}$'den azdır ve G'' çizgesinde $\lceil \frac{n}{2k} \rceil$' büyüklüğünde sabit bir küme yoktur.

Son olarak bu G'' çizgesinden; bütün üçgenlerden en az bir köşe olmak koşuluyla $\frac{n}{2}$ köşe ve bu köşeleri içeren bütün kenarları silip $G(V, E)$ çizgesini elde edelim. $|V| = \frac{n}{2}$ olduğu açıktır. Ayrıca G çizgesinde üçgen ve $\lceil \frac{n}{2k} \rceil$ büyüklüğünde sabit bir küme yoktur. Dolayısıyla $\chi(G) > k$ olup ispat tamamlanır. □

Bu bölümde çizge boyamanın ne olduğunu, verilen bir çizgenin kromatik polinomunun ve kromatik sayısının ne olduğunu tanımladık. Ayrıca verilen bir çizgenin kromatik sayısı için birkaç tane önemli alt ve üst sınır sonuçlarını inceledik. Bir sonraki bölümde ise çizge boyama probleminin önemini anlamak adına otomata teori ve algoritmik kompleksiti konularını ele alacağız. Yine bir sonraki bölümde P ve NP gibi kavramları tanımlayacak ve meşhur $P = NP$ sorusunun ne anlama geldiğini ve çizge boyama probleminin bu anlamda nasıl bir öneme sahip olduğunu göreceğiz.

3. Hesaplanabilirlik ve Karmaşıklık Teorisine Giriş

Verilen herhangi bir çizgenin $k \in \mathbb{N}$ olmak üzere k renk kullanılarak uygun boyanıp boyanamayacağını belirlemek bir NP-Tam bir problemdir. 4. Bölümde, verilen bir $G(V, E)$ çizgesinin verilen bir k pozitif tam sayısı için k renk kullanılarak uygun bir şekilde boyanıp boyanamayacağını bulma probleminin NP-Tam olduğu detaylıca gösterilecektir. Bu bölümde ise çizge boyama probleminin zorluk derecesini anlamak için gerekli olan hesaplanabilirlik ve karmaşıklık kuramına dair temel kavramlar tanıtılacaktır. Amaç, okuyucunun çizge boyamanın neden NP-Tam bir problem olduğunu 4. Bölümde anlayabilmesini sağlamak ve bu konu hakkında gerekli olan tüm ön bilgileri öğrenmesidir.

Bu amaçla bu bölüm sırasıyla; temel dil teorisi, karar problemlerinin tanıtımı, temel sonlu otomata teorisi, Turing makinelerinin tanımı, karmaşıklık sınıfları ve son olarak algoritmik kompleksiti teorisinin en temel teoremlerinden biri olan Cook-Levin Teoremi'ni içerecek şekilde yapılandırılmıştır. Böylece algoritmik karmaşıklık konusunu hiç bilmeyen bir okur bu bölümün sonunda polinom zaman algoritmalar ve kararsız-polinom zaman algoritmalar gibi iki temel kavramı kavrayabilecektir. 4. Bölümde ise bu bölümde aktarılan kavramları kullanarak çizge boyama probleminin NP-Tam olduğunu kanıtlayacağız ve diğer önemli problemlerle olan ilişkilerini göreceğiz.

3.1 Karar Problemleri ve Makineler

Günümüzde bilgisayarları, bilgisayarın tasarım aşamasında tanımlanmış prosedürleri kullanarak farklı türden problemleri çözmek için kullanabilmekteyiz. Örneğin optimizasyon problemlerini çözmek ve incelemek bilgisayar programlarının en önemli görevlerindendir ve optimizasyon problemlerinin önemini kavramak sezgisel olarak kolaydır. Biz bu bölümde karar problemleriyle ilgileneceğiz. Okur herhangi bir karar problemini şu şekilde düşünebilir: verilen bir karar probleminin cevabı ya evettir ya hayırdır. Tabii bu noktada belirtmeliyiz ki bu problemler objektif problemler yani bir cevabı olan problemler olmalıdır. Bir cevabının olmasından kastımız, probleminin cevabının sorulan kişiye, yere, zamana veya sorudan başka

bir şeye bağlı olmamasıdır. Örneğin, ”Bal güzel midir?” sorusu bu bağlamda gerçek bir karar problemi değildir. Çünkü sorunun cevabı kişiye ya da bazı durumlara göre farklılık gösterebileceğinden kesin bir cevabı yoktur. Ama “2 sayısı 3 sayısından büyük müdür?” sorusu gerçek bir karar problemidir. Çünkü soruda verilen her şeyin net bir tanımı olduğu gibi sorunun net ve kesin bir cevabı da vardır. Karar problemleri birçok fenomeni ifade etmek için kullanılabilir ve karar problemlerinin çözümü başka birçok problemin de çözümünün bulunmasında yardımıcı olur. Örneğin verilen bir optimizasyon probleminin optimum değerine “Bu optimum değer herhangi bir x değerinden büyük mü?” gibi karar problemleriyle yaklaşabiliriz.

Yukarıda bir problemin, karar problemi olarak kabul edilebilmesi için gerçek bir cevabının olması gerektiğini belirtmiştik. Şimdi biraz daha temel bir durumdan bahsedelim: Bir problemin var olması için önce o problemi yazılı olarak ifade edebilmemiz gerekir. Bir problemi yazmanın formel tanımı oldukça teknik ve uzundur. Bu nedenden ötürü bu bölümde çok formel bir dil kullanmayacağız. [4] referansından bu konu hakkındaki teknik detaylar incelenebilir.

3.1.1 Alfabeler

Yazılı dillerin en temel öğeleri harflerdir. Bütün diğer öğeler harfler vasıtasıyla üretilir. Bir karar problemini de yazarken harfleri kullanacağız. Tabii aynı problemler farklı dillerde farklı alfabelerle farklı şekilde ifade edilebilir. Bir problemi farklı dillerde ifade etmenin, eğer diller makul ise, o problemin çözülebilirliği üzerine etkisi yoktur [4]. Dolayısıyla bir problemi kendi oluşturduğumuz bir dille ifade edip o problemin çözümünü de yine aynı dil üzerinden yapmak, o problemin çözülebilirliğini kavrayabilmek açısından yeterli olacaktır.

Modern dijital bilgisayarlar farklı seviyelerde farklı diller kullanmaktadır. Ancak en temelde, bilgisayarlar devrelerden oluşmaktadır ve bu devreler de transistörlerden oluşmaktadır. Transistörleri açık veya kapalı durumda olan elektrik anahtarları olarak düşünebiliriz. Bütün bu fiziksel alt yapı bilgisayarların bir dil oluşturmak için gerek duyduğu alfabeye zemin hazırlamaktadır. Modern bilgisayarların en temel dili $\{0, 1\}$ harflerinden oluşur. Burada

değinmekte fayda görüyoruz ki anlamlı bir alfabe en az 2 harften oluşmak durumundadır, tek harften oluşan bir alfabe çok anlamlı değildir [4]. Ayrıca bir dilde az veya çok harf olması o dilin anlatım kabiliyetiyle ilişkili değildir. Makul olan bütün dillerin anlatım kapasitelerinin eş değer olduğunu kabul etmekte bir mahsur yoktur [4].

Sonuç olarak bir dilin en temel yapı taşı *harflerdir* ve bir dilde var olan bütün harfler o dilin *alfabesini* oluşturur. Bir karar problemini, makul bir dilde yazılı bir biçimde ifade edip o yazılı biçimi girdi olarak alan ve çıktı olarak ise karar probleminin cevabını, yani (Evet/Hayır), veren bir mekanizma geliştireceğiz. Daha önce de belirttiğimiz gibi bir problem farklı dillerde farklı şekilde ifade edilebilir dolayısıyla aynı problemi farklı dillerde çözmek için farklı mekanizmalar gereklidir, ancak yine daha önce belirttiğimiz üzere makul dilleri eş değer saymamızda bir mahsur yoktur. Yani bir problemi A makul dilinde ifade edip çözebilen bir mekanizma kurabiliyorsak aynı problemi başka bir B makul dilinde de ifade edip çözen başka bir mekanizma kurabileceğimizi varsayabiliriz [4]. Burada problem dediğimizde okurun aklına özel bir problem yerine bir problem kümesi gelmelidir. Yani verilen herhangi bir $G(V, E)$ çizgesi için G çizgesinin k renk kullanılarak uygun boyanıp boyanamayacağı bir problemdir. Özel olarak verilen bir $G(V, E)$ çizgesinin, örneğin tek teker veya çift teker yapısında bir çizgenin, k tane renk kullanılarak uygun boyanıp boyanamayacağı ise o problemin özel bir elemanıdır.

Yukarıda bir dilin sezgisel tanımından bahsettik. Bir sonraki alt bölümde ise dillerin tanımını otomataları ve alfabeleri kullanarak vereceğiz.

3.1.2 Sonlu Otomatalar

Bu bölümde otomataları öncelikle örnekler vasıtasıyla inceleyecek ardından formel olarak tanımlayacağız. Şekil 3.1'de bir otomatanın görseli görülmektedir. Bu otomata q_0 konumunda başlamaktadır ve girdi olarak 0 ve 1 harflerinden oluşan bir dizi kabul etmektedir. Bir otomata bulunduğu konum ve girdisinin incelediği elemanına göre bir sonraki konumunu güncellemektedir. Bu güncelleme yöntemi oklarla gösterilmiştir. Örneğin q_2 konumunda

olan bir otomata; 1 harfi geldiğinde q_0 konumuna gidecek, 0 harfi geldiğinde ise q_2 konumunda kalacaktır. Girdinin her elemanı incelendiğinde otomatanın bulunduğu konum ise o girdiye verdiği cevap olarak adlandırılır. Örneğin şekildeki otomataya 01001 dizisini girdi olarak verdiğimizde otomata aşağıda adım adım anlatıldığı şekilde hareket edip cevap verecektir. Burada dikkat edilmelidir ki okumayı soldan sağa doğru yapıyoruz. Bu durum farklı kaynaklarda farklılıklar göstermektedir. Fakat genel durumu değiştiren herhangi bir yargı yoktur, sadece sıralamanın tersten yapılması olasıdır. Yukarıda belirtildiği üzere makul dillerin sıralama farklılığı ile oluşturulduğu düşünülebilir.

1. otomata q_0 konumunda başlayacaktır.
2. Girdinin ilk elemanı 0 harfi olduğu için otomata q_0 konumundan q_1 konumuna okla belirtildiği gibi gidecektir.
3. otomata artık q_1 konumundadır.
4. Girdinin sonraki elemanı 1 harfi olduğundan otomata q_1 konumundan q_0 konumuna gidecektir.
5. otomata artık q_0 konumundadır.
6. Girdinin sonraki elemanı 0 harfi olduğundan otomata q_0 konumundan q_1 konumuna gidecektir.
7. otomata artık q_1 konumundadır.
8. Girdinin sonraki elemanı 0 harfi olduğundan otomata q_1 konumundan q_2 konumuna gidecektir.
9. otomata artık q_2 konumundadır.
10. Girdinin sonraki elemanı 1 harfi olduğundan otomata q_2 konumundan q_0 konumuna gidecektir.
11. otomata artık q_0 konumundadır.

12. Girdinin bütün elemanları incelendiğinden ve otomata q_0 konumunda olduğundan otomatanın girdiye verdiği cevap q_0 olacaktır.

Otomataya verdiğimiz girdiyi karar probleminin yazılı hali, otomatanın verdiği cevabı ise otomatanın problem için bulduğu cevap olarak düşünebiliriz. Şekil 3.1'de görülebileceği gibi q_0 ve q_1 konumları tek çemberden oluşmaktayken q_2 konumu çift çemberden oluşmaktadır. Bunu şöyle düşünebiliriz; eğer otomata tek çemberden oluşan bir konumu cevap olarak veriyorsa soruya "Hayır" cevabını veriyor, çift çemberden oluşan bir konumu cevap olarak veriyorsa soruya "Evet" cevabını veriyordur. Bir otomata sonlu sayıda tek veya çift çemberli konumlardan oluşabilir. Burda önemli olan bir otomatanın her konumu için ve kabul ettiği alfabenin her harfi için gideceği yeni konumun tanımlanmış olmasıdır. Ayrıca kolayca anlaşılabileceği gibi otomata farklı uzunlukta girdileri de kabul edebilmektedir.

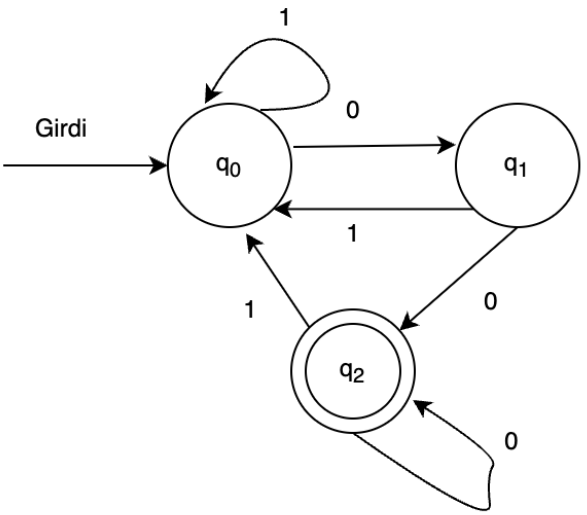
Aslında Şekil 3.1 görselleştirdiğimiz otomata verilen bir doğal sayının 4 ile bölünüp bölünemeyeceğinin cevabını vermektedir. Girdi olarak da doğal sayının 2'lik tabanda yazılımını kabul etmektedir. Bu örneği vermemizin bir sebebi de okura otomataların karar problemlerini çözmek için nasıl kullanılabileceğini aktarmak.

Artık otomataların formel tanımını verebiliriz.

Tanım 3.1.1. Bir $D = (Q, \Sigma, \delta, q_0, F)$ 5'lisi,

- Q: Konumlar kümesi,
- q_0: Başlangıç konumu,
- Σ: Alfabe,
- $F \subseteq Q$: Evet cevaplı konumlar,
- δ: $Q \times \Sigma \to Q$, geçiş fonksiyonu,

olmak üzere kararlı bir otomata olarak adlandırılır.



Şekil 3.1 : 4 ile bölünebilme kuralını tanıyan otomata

Örneğin Şekil 3.1'de gösterilen otomatanın: Konumlar kümesi $Q = \{q_0, q_1, q_2\}$ kümesi, alfabesi $\Sigma = \{0, 1\}$ kümesi, başlangıç konumu q_0, evet cevaplı konumları $F = \{q_2\}$ kümesidir. Geçiş delta fonksiyonu ise aşağıdaki tabloda belirtilmiştir.

Anlık Konum \Giren harf	0	1
q_0	q_1	q_0
q_1	q_2	q_0
q_2	q_2	q_0

Şimdi Şekil 3.1 deki otomatayı inceleyelim. Örneğin bu otomataya 1100 girdisini verirsek, otomata, ilk konum başlangıç konumu olmak üzere sırasıyla: q_0, q_0, q_0, q_1, q_2 konumlarında bulunur ve q_2 konumu girdiye verdiği cevap konumu olur. Veya otomataya 00 girdisini verirsek, otomata ilk konum başlangıç konumu olmak üzere sırasıyla : q_0, q_1, q_2 konumlarında olur. Okurun da fark edebileceği üzere bu otomata, son iki harfi 0 olan bütün girdilere "Evet" cevabını verirken son iki harfi 0 olmayan bütün girdilere de "Evet" cevabını vermiyor. 4'e bölünen bütün sayıların ikilik tabanda son iki hanesi 0'dır. Dolayısıyla bu otomata 4'e bölünen sayıların 2'lik tabandaki yazılışlarına "Evet" cevabı veriyor, yani bir sayının 4'e bölünüp bölünemeyeceğini belirleyebiliyor. Bu örneği otomataları okurun kafasında daha anlaşılır kılmak için paylaştık.

Şimdi bazı önemli kavramların tanımlarını verelim.

Tanım 3.1.2. Σ sonlu bir alfabe olmak üzere, Σ kümesinin elemanlarıyla üretilen her sonlu diziye Σ alfabesinden üretilen sonlu kelime denir.

Tanım 3.1.3. Σ sonlu bir alfabe olmak üzere, Σ^* kümesi Σ alfabesiyle üretilen bütün sonlu kelimelerin kümesi olarak tanımlanır.

Tanım 3.1.4. Σ sonlu bir alfabe ve $L \subseteq \Sigma^*$ olmak üzere, L'ye Σ alfabesiyle üretilen bir dil denir.

Tanım 3.1.5. $D = (Q, \Sigma, \delta, q_0, F)$ bir kararlı otomata ve $k \in \Sigma^*$ olsun. $D(k)$ değeri D otomatanının k girdisine verdiği cevap olarak adlandırılır.

Tanım 3.1.6. $D = (Q, \Sigma, \delta, q_0, F)$ bir kararlı otomata ve $L \subseteq \Sigma^*$ olsun. $a \in \Sigma^*$ olmak üzere, $D(a) \in F \iff a \in L$ ise L'ye D otomataıyla oluşturalan bir dil veya D otomatanının tanıdığı bir dil denir.

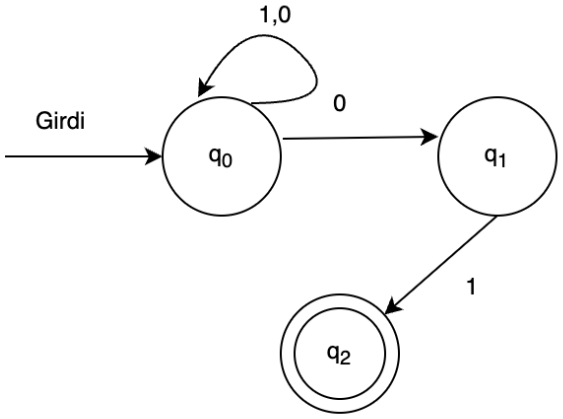
Şimdi kararsız sonlu otomataları tanımlayacağız. Kararsız sonlu otomatalar; kararlı sonlu otomatalarla aynı dilleri tanımaları, aynı dili ayırt eden kararlı otomatalara göre daha kolay görselleştirilebilmeleri ve sezgisel olarak anlaması karışık olan kararsız Turing makinelerinin anlaşılması için bir alt yapı kurması özelliklerinden dolayı önemli mekanizmalardır.

Tanım 3.1.7. Bir $N = (Q, \Sigma, \delta, q_0, F)$ 5'lisi

- Q: Konumlar kümesi,
- q_0: Başlangıç konumu,
- Σ: Alfabe,
- $F \subseteq Q$: Evet cevaplı konumlar,
- δ: $\delta(Q)$, Q'nun kuvvet kümesi olmak üzere $Q \times \Sigma \to \delta(Q)$ ile tanımlanan geçiş fonksiyonu

olmak üzere kararsız otomatadır.

Tanımından da görüleceği üzere kararsız otomatalar bir girdiyi incelerken birden fazla konumda olabilir bu durumda, bulunduğu bütün konumlar için girdinin sıradaki elemanına ve geçiş fonksiyonuna göre yeni konumlara (birden fazla konum olabilir) geçer veya o konumdan silinebilir (boş kümenin her kuvvet kümesinin elemanı olduğu gerçeğinden dolayı). Girdinin bütün harfleri incelendiğinde kararsız otomata Evet cevaplı herhangi bir konumda bulunuyorsa o girdiye "Evet" cevabını verir aksi durumda ise "Hayır" cevabını verir. Şekil 3.2'de $\Sigma = \{0, 1\}$ alfabesiyle çalışan ve sonu 01 ile biten kelimeleri bulan bir kararsız otomata gösterilmiştir. Okur aynı dili kabul eden kararlı bir otomata tasarlarsa, kararsız otomataları görselleştirmenin çok daha kolay olduğunu görebilecektir. Kararlı otomatalar ve kararsız otomataların aynı dilleri kabul ettiğinin kanıtı, otomatalar, Diller ve Gramerler hakkında daha fazla bilgi sahibi olmak isteyen okuyucular Peter Linz'in "An Introduction to Formal Languages and Automata" [5] isimli kitabını inceleyebilirler .



Şekil 3.2 : 01 ile biten dilleri bulan kararsız otomata

Kararsız otomataların çalışmasını daha iyi kavramak adına Şekil 3.2'de gösterilen kararsız otomataya "1001" girdisini verip adım adım inceleyelim:

1. İlk adımda otomata q_0 konumundadır
2. 1 harfi sonucu otomata q_0 konumuna geçer

3. q_0 konumundaki otomataya, 0 harfi girdiğinde otomata hem q_0 konumuna hem q_1 konumuna geçer.

4. otomata şu anda q_0 ve q_1 konumlarındadır.

5. Sonraki adımda girdinin 3.harfi olan 0 harfi; otomatayı q_0 konumundan, q_0 ve q_1 konumlarına götürürken q_1 konumundan bir yere götürmez.

6. otomata hem q_0 hem q_1 konumundadır. (otomatanın q_1 konumunda olmasının nedeni bir önceki adımda q_0'dan q_1 gelmiş olmasıdır,bir önceki adımda bir yere götürmeyen q_1 konumu yok olmuştur.)

7. Girdinin son harfi olan 1 harfi otomataya girdiğinde otomata q_0 konumundan q_1 konumuna, q_1 konumundan ise q_2 konumuna geçer.

8. otomata q_2 ve q_0 konumlarındadır ve girdi bitmiştir.

9. otomatanın son konumlarından biri evet cevaplı konum olduğu için, yani q_2, otomata girdiye evet cevabını verir.

Şimdi, hesaplanabilirlik kuramının temel yapı taşlarından biri olan ve modern kuramsal bilgisayar biliminin sınırlarını tanımlamada merkezi bir rol oynayan Turing makineleriyle ilgili detaylara geçebiliriz.

3.1.3 Turing Makineleri

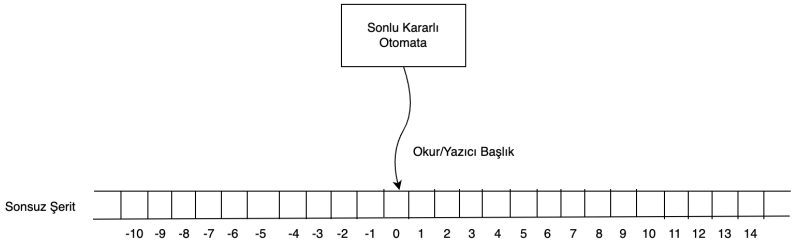
Bir önceki bölümde kararlı sonlu otomataları ve kararsız sonlu otomataları tanımladık. Ayrıca bunların aynı dilleri tanıdığına değindik. Mühendislikte kullanılan birçok dijital devre özelinde bir sonlu otomatadır. Sonlu kararlı otomatalar yapıları itibariyle CMOS transistörler kullanılarak elektronik devrelerinde kontrol merkezi olarak kullanılabilmektedir. Bu noktada kararsız sonlu otomatalar gerçek uygulamaları olan mekanizmalar değildirler. Kararsız otomatalar kararlı otomataları daha az karmaşık şekilde ifade etmek için tanımlanmış mekanizmalardır.

Bu noktada okurun aklına şu soru gelebilir: "Sonlu bir otomata bir alfabeyle oluşturulabilecek her dili tanıyabilir mi?". Bu sorunun cevabı maalesef hayırdır. Üstelik daha da sıkıntılı durum sonlu otomatalar bir alfabeyle oluşturulabilecek dillerin çok küçük bir kısmını tanıyabilirler. Örneğin $\Sigma = \{0, 1\}$ alfabemiz olsun.

$L = \{1^n 0^n : n \in \mathbb{N}$ ve $a \in \Sigma$ için $a^n :=$ n tane a'nın yanyana yazılmasıyla elde edilen dizi$\}$

dilleri genel olarak sonlu otomatalar tarafından tanınamazlar [5].

Kısacası, sonlu otomatalar yapılarının basitliği nedeniyle birçok dili tanıyamamaktadır. Ancak yapısı daha karışık olan Turing makineleri dil tanıma konusunda sonlu otomataların performansını geçmektedir. Şekil 3.3'de bir Turing makinesinin mekanizmasını görülebilir. Turing makineleri bir adet sonsuz şerit, bir adet sonlu otomata ve bir adet yazıcı/okuyucu başlıktan oluşur. Şekil 3.3'de de görülebileceği gibi sonsuz şerit üzerinde eksi sonsuzdan artı sonsuza kadar etiketlenmiş alanlar vardır. Bu alanların her birine bir harf yazılabilmektedir. En başta Turing makinesine verilecek girdi 0 numaralı alandan başlayarak sağa doğru yazılır. Turing makinelerinde kapsayıcılığı arttırmak açısından iki farklı alfabe kullanılır. Bunlardan birincisi girdilerin yazıldığı Σ alfabesi diğeri ise Turing makinesinin şeride yazdığı Π alfabesidir. Genel olarak $\Sigma \subset \Pi$ ilişkisi vardır. Turing makinesinin çalışma prensibi aşağıdaki gibi özetlenebilir:



Şekil 3.3 : Kararlı Turing Makinesi

1. Okur/Yazar başlık bulunduğu alanı okur ve okuduğu değer ve sonlu kararlı otomatanın bulunduğu konuma göre sonlu kararlı otomatanın bir sonraki konumunu belirler.

2. Okur/Yazar başlık bulunduğu alandaki harfe göre ve sonlu kararlı otomatanın bulunduğu konuma göre bulunduğu alana çalışma alfabesinden yazacağı harfi belirler.

3. Okur/Yazar başlık bulunduğu alandaki harfe göre ve sonlu kararlı otomatanın bulunduğu konuma göre bulunduğu alanı ya bir sağa ya bir sola taşımak koşuluyla Okur/Yazar başlığın sonraki konumunu belirler.

4. Turing makinesi otomatını 1.adımda belirlediği konuma getirir.

5. Turing makinesi Okur/Yazar başlığın bulunduğu kareye 2.adımda belirlediği harfi yazar.

6. Turing makinesi Okur/Yazar başlığını 3.adımda belirlediği şekilde hareket ettirir.

7. Eğer sonlu kararlı otomata evet ya da hayır konumlarına gelirse geldiği konuma göre Turing makinesi girdi için evet ya da hayır cevaplarını verir.

8. Eğer sonlu kararlı otomata evet ya da hayır konumlarına gelmezse gelene kadar aynı şekilde çalışmaya devam eder.

Turing makinelerinin adımlar arasındaki geçişleri otomatalarda olduğu gibi bir adet transfer fonksiyonu ile ifade edilebilir. Buradaki önemli çıkarımlardan biri şudur, sonlu otomatalar girdinin boyutu (yani harf sayısı) adımda cevap vermektedir ancak Turing makineleri için bu durum geçerli değildir. Turing makinelerinin cevaba ulaştığı adım sayısı girdinin boyutuna ve girdinin kendisine bağlıdır ayrıca bir Turing makinesi sonsuz bir döngüye de girebilir.

Bir problem kümesini düşünelim, örneğin sudoku bulmacasını ele alalım.. Sudoku 10×10 luk bir karede oynanabileceği gibi 100×100 lük bir karede de oynanabilir. İkisi de aynı problem olmasına rağmen açık bir şekilde görüleceği üzere 100×100 lük sudoku daha büyük olduğu için bu sudokuyu çözmek daha çok zaman alacaktır. Bundan dolayı farklı problemlerin zorluk düzeylerini karşılaştırırken problemin büyüklüğünü de hesaba katmak gerekmektedir. Bu bağlamda bir Turing makinesine verilen bir girdi için, Turing makinesinin o girdiye bir cevap vermesi için geçen adım sayısının girdinin büyüklüğüyle nasıl

bağlantılı olduğu önemlidir. Örneğin n uzunluğunda verilen bir problem için Turing makinesinin o problemi çözmek için attığı adım sayısı n'nin polinom cinsinden bir fonksiyonuyla sınırlanmış ise o problem polinom zaman problemdir. Turing makinesinde polinom zaman olan bir problem modern bilgisayarlarda da polinom zamandır ve modern bilgisayarlarda polinom zaman olan bir problem Turing makinesinde de polinom zamandır [6].

Genel olarak bir karar problemi polinom zamansa o problem modern bilgisayarlar yardımıyla büyüklüğüne göre makul bir hızda çözülebilir.

Artık formel tanımları verebiliriz.

Tanım 3.1.8. $T = (\Pi, \Sigma, b, Q, q_0, q_y, q_n, \delta)$ 8'lisi

- Π: Turing makinesinin şeride yazığı alfabe,
- Σ: Girdilerin yazıldığı alfabe,
- $b \in \Pi - \Sigma$, olacak şekilde bir boşluk harfi,
- Q: Turing makinesinin sonlu kararlı otomatanının konumlar kümesi,
- $q_0 \in Q$, olacak şekilde başlangıc konumu,
- $q_y \in Q$, olacak şekilde evet cevaplı girdilerin son konumu,
- $q_n \in Q$, olacak şekilde hayır cevaplı girdilerin son konumu,
- $\delta : Q - \{q_y, q_n\} \times \Pi \to Q \times \Pi \times \{-1, +1\}$ olacak şekilde geçiş fonksiyonu,

olmak üzere kararlı bir Turing makinesidir.

otomatalarda olduğu gibi Turing makinelerinde de kararlı ve kararsız ayrımı vardır. Ve yine otomatalarda olduğu gibi Turing makinelerinde de kararlı ve karasız makineler aynı dilleri tanıyabilmektedir [6]. Fakat otomatalarda bir girdiye verilen cevaba o girdinin uzunluğu kadar adımda ulaşılırken, Turing makinelerinde durumun böyle olmadığından bahsetmiştik. Bir problemin kararlı ve kararsız Turing makinelerinde çözüldüğü adım sayısı o problemin

zorluk sınıfını belirlemede kullanılır. Genel olarak Turing makineleri; problemlerin zorluk sınıflarını tanımlamak için kullanılan, bilgisayar biliminde önemli yere sahip olgulardır.

Kararsız Turing makineleri için birçok eş değer tanım vardır ama biz burada Garey tarafından verilen tanımını kullanacağız [5].

Şekil 3.4 de bir kararsız Turing makinesinin mekanizması gösterilmiştir.

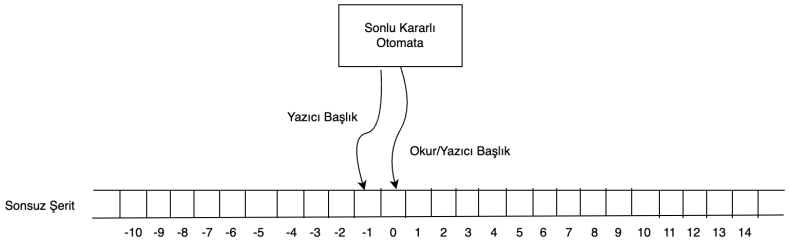
Genel olarak kararsız Turing makinelerinin çalışma prensibi aşağıdaki gibi özetlenebilir:

- Yazıcı başlık -1 numaralı kareden başlar ve her adımda rastgele ya bulunduğu kareye makinenin alfabesinden bir harf yazar ya da yazmaz.
- Eğer Yazıcı başlık bulunduğu kareye bir harf yazmazsa kendini kapatır ve makine aynı şerit üzerinden kararlı Turing makinesi adımlarını izleyerek devam eder.
- Yazıcı başlık bulunduğu kareye bir harf yazarsa bir sola kayar ve döngüye aynı şekilde devam eder.

Şöyle bir analoji yapmak yanlış olmaz, kararsız bir Turing makinesi verilen bir problem için yazılabilecek bütün cevapları yazıp ardından yazdığı her cevap için Turing makinesi gibi davranıp cevabın doğruluğunu kontrol eder. Eğer doğru bir cevap bulursa verilen girdiye "Evet" yanıtını verir, doğru bir cevap bulamazsa verilen girdiye "Hayır" yanıtını verir.

Kararsız Turing makinesinde yazıcı başlık sonsuza kadar yazmaya devam edebilir. Dolayısıyla Kararsız Turing makineleri sezgisel olmayan, anlaması güç ve hiçbir şekilde gerçeklenemeyecek mekanizmalardır. Ancak kararsız Turing makineleri kullanılarak NP zorluk sınıfı tanımlanmaktadır.

Bir sonraki alt bölümde, önce yukarıda değindiğimiz karmaşıklık sınıflarını ayrıntılı biçimde tanımlayacak, ardından bu sınıfların karakterizasyonunda önemli bir rol oynayan Cook-Levin Teoremi'ni ele alacağız.



Şekil 3.4 : Kararsız Turing Makinesi

3.2 Karmaşıklık Sınıfları ve Cook-Levin Teoremi

Bir önceki alt bölümde bir karar probleminin ne olduğuna ve ne olmadığına değinmiştik. Ayrıca bu problemleri alfabeler kullanarak ifade etmekten bahsetmiştik. Eğer bir karar problemini yazarsak bu yazılı formları Evet/Hayır cevabı veren mekanizmalara girdi olarak verip cevap alabiliriz.

Örneğin; verilen herhangi bir $G(V, E)$ çizgesinin, bir k pozitif tam sayı sabiti için, k farklı renk kullanılarak uygun biçimde boyanıp boyanamayacağını bulan bir mekanizma geliştirmek istiyoruz. Bir Σ alfabesini kullanarak, her $G(V, E)$ çizgesini yazabileceğimizi kabul edebiliriz.

$$L = \{G(V, E) : G(V, E), k \text{ renk kullanılarak uygun şekilde boyanabilir}\}$$

kümesini tanımlayalım. Tanım gereği, $L \subset \Sigma^*$ olduğundan, L, Σ alfabesiyle oluşturulan bir dildir. Eğer biz L dilini tanıyan bir mekanizma oluşturursak verilen herhangi bir $G(V, E)$ çizgesinin k renk kullanılarak uygun şekilde boyanıp boyanamayacağını belirleyen bir mekanizma oluşturmuş oluruz.

Turing makineleri dil tanıma konusunda otomatalardan çok daha fazla kabiliyetli olduğundan karışık dilleri tanımak için Turing makineleri kullanılır. Daha önceden kararlı ve kararsız Turing makinelerini tanımlamıştık. Ayrıca otomataların aksine bir Turing makinesinin verilen bir girdi için girdinin uzunluğu adımda cevap vermek zorunda olmadığından bahsetmiştik. Şimdi karar problemleri için iki adet zorluk sınıfını Turing makinelerinin yardımıyla tanımlacağız.

Tanım 3.2.1. $\Theta = \{A : A$ bir karar problemi$\}$ kümesi bütün karar problemlerinin kümesi olarak tanımlanır.

Tanım 3.2.2. $A \in \Theta$ olmak üzere; $|A|_e$, A probleminin e makul dilinde kaç harf kullanılarak yazılabileceğini göstersin.

Daha önceden makul dillerin birbirine denk olduğunu varsayabileceğimizi söylemiştik, bu sebeple $|A|_e$ yerine kısaca $|A|$ kullanabiliriz.

Şimdi karmaşıklı teorisinde kullanılan iki önemli tanımı verelim.

Tanım 3.2.3. P kümesi, A karar problemini $|A|$'nın polinomu türünden bir fonksiyonla sınırlanmış adımda tanıyan bir kararlı Turing makinesi olan tüm karar problemlerin kümesi olarak tanımlanır.

Yukarıdaki tanımı matematiksel olarak

P=$\{A \in \Theta :$ A'yı $|A|$'nın polinomu türünden bir fonksiyonla sınırlanmış adımda tanıyan, kararlı bir Turing makinesi var. $\}$

şeklinde ifade edebiliriz.

Tanım 3.2.4. NP kümesi, A karar problemini $|A|$'nın polinomu türünden bir fonksiyonla sınırlanmış adımda tanıyan bir kararsız Turing makinesi olan tüm karar problemlerinin kümesi olarak tanımlanır.

Bu tanımı ise matematiksel olarak

NP=$\{A \in \Theta :$ A'yı $|A|$'nın polinomu türünden bir fonksiyonla sınırlanmış adımda tanıyan, kararsız bir Turing makinesi var. $\}$

olarak ifade edebiliriz.

Okurun kafasında NP ve P kümelerini sezgisel olarak oturtmak için şu iki örneği vereceğiz. Bir sudoku çözerken boş karelere gelebilecek sayıları teker teker deneyip diğer kareleri

dolduruyoruz. Eğer çelişki bulursak dönüp doldurduğumuz kareleri silip yerlerine başka sayılar yazarak çelişki bulmadan ilerlemeye devam ediyoruz. Yani deneme yanılma yapıyoruz, deneme yanılma yaparak çözebileceğimiz problemler NP kümesindeki problemler olarak düşünülebilir. Şimdi $x^2 - 1 = 0$ polinomunun köklerini bulurken de x'in alacağı değerler üzerine deneme yanılma yapabilir polinomu böyle çözmeye çalışabiliriz. Ancak genel olarak polinomların çözümlerini bulmamızı sağlayan formüller vardır, yani deneme yanılma yapmadan daha hızlı bir şekilde bir polinomun köklerini belirleyebiliriz. Deneme yanılma yapmak zorunda kalmadan çözebileceğimiz problemler, bir formülü veya kısa yolu olan problemler ise P kümesindeki problemlerdir. Anlaşılacağı üzere $P \subset NP$ aşikar bir durumudur. Fakat bu kapsamanın tersi için genel bir şey söyleyemeyiz.

En temel düzeyde "$P = NP$ midir?" sorusu, her doğruluğu verimli bir şekilde denetlenebilen problemin aynı zamanda verimli bir şekilde çözülebilir olup olmadığını sorgular. Eğer $P = NP$ ise, bu durum tüm karar problemlerini polinom zamanda çözen bir algoritmanın var olduğu, ancak bizim henüz bu algoritmaların bazılarını keşfetmemiş olduğumuz anlamına gelir. Buna karşılık $P \neq NP$ ise, bazı problemlerin doğası gereği yalnızca olası tüm çözümleri deneyerek çözülebileceğini, yani bu problemler için genel geçer verimli bir çözüm yolunun bulunmadığını ifade eder.

Teknik bir bakışla her kararlı Turing makinesi aynı zamanda yazıcı başlığı bir şey yazmayan bir kararsız Turing makinesi olduğundan, $P \subseteq NP$ olduğu açıktır. Bu noktada P ve NP kümeleri arasındaki ilişkiyi okura daha iyi bir şekilde aktarabilmek için Cook-Levin teoremini kanıtıyla beraber vereceğiz. Bunun için öncelikle polinomsal zamanda indirgeme ve SAT problemini tanımlamamız gerekmektedir.

Tanım 3.2.5 (Polinomsal Zamanda İndirgenme)**.** Polinomsal zamanda indirgeme, A problemine ait herhangi bir x girdisi için, B problemine ait bir $f(x)$ girdisi üretmek üzere tanımlanmış bir indirgeme fonksiyonu f olduğu anlamına gelir. Bu f fonksiyonu aşağıdaki özellikleri sağlamalıdır:

1. $x \in A \Rightarrow f(x) \in B$, yani x girdisi A problemini evet cevabına götürüyorsa, $f(x)$ girdisi de B problemini evet cevabına götürmelidir.

2. f fonksiyonu kararlı bir Turing makinesi tarafından polinomsal zamanda hesaplanabilir olmalıdır.

Bir karar problemi A, başka bir karar problemi B'ye *polinomsal zamanda indirgenebilir* ise, bu durum $A \leq_p B$ şeklinde gösterilir.

Dolayısıyla, $A \leq_p B$ ise, A problemini polinom zamanında çözmek için B problemini polinom zamanda çözmek yeterlidir ve A problemini, B problemine dönüştürüp çözme işlemi polinom zamanda olacağından verimli olarak kabul edilir.

Tanım 3.2.6. NP-Tam kümesi, NP kümesinin içindeki bütün problemlerin polinomsal zamanda indirgenebileceği problemlerin kümesidir.

Bu tanımı ise matematiksel olarak

$$NP\text{-Tam} = \{A \in NP : \forall B \in NP, B \leq_p A\} \subset NP$$

olarak ifade edebiliriz.

Bu kümenin boş olmadığı ilk defa Cook-Levin Teoremiyle gösterilmiştir [4]. Ardından Richard Karp 21 problemin daha NP-Tam kümesinde olduğunu kanıtlamıştır [7].

Bununla birlikte, henüz NP sınıfındaki tüm problemlerin polinomsal zamanda indirgenebileceği ortak bir problemin varlığını,yani NP-Tam kümesinin boş olmadığını, göstermiş değiliz. Ancak bu noktada, okurun NP-Tam kavramını anlamasının gerekli olduğunu düşündüğümüz için, bu alt bölümde NP-Tamlığın tanımına yer verdik. İzleyen kısımlarda ise Cook-Levin Teoremi aracılığıyla NP-Tam sınıfının boş olmadığını gösterecek ve çizge boyama probleminin NP-Tam bir problem olduğunu kanıtlayacağız.

Polinomsal zamanda indirgenme kavramı, özellikle NP-Tamlık teorisinde kritik öneme sahiptir. Bir problemin NP-Tam olduğunu göstermek için, NP-Tam olduğu bilinen başka bir probleme polinomsal zamanda indirgemenin mevcut olduğunu göstermek yeterli olacaktır.

Bazı NP-Tam problemlerin bazı problemlere polinomsal zamanda indirgenebildiği gösterilmişken, o problemlerin NP-Tam problemlere polinomsal zamanda indirgenebileceği gösterilememiştir. Bu tür problemlerden oluşan kümeye NP-Zor denir. Gezici Satıcı,Durma(Halting) problemleri en meşhur NP-Zor problemlerdendir. NP-Zor problemler en az NP-Tam problemler kadar zor olan problemlerdir.

Cook-Levin Teorem'inin ifadesini ve ispatını verebilmek için aşağıda vereceğimiz bazı önemli kavramlara ihtiyacımız vardır.

Tanım 3.2.7. $A = \{x_1, x_2, ..., x_n\}$ 0 veya 1 değerlerini alan mantık değişkenlerimizin kümesi olsun (bilateral) ve $\forall x_i \in A$ için x_i', x_i bilateralinin değili olsun. O zaman sonlu bir $M \in \mathbb{N}$ için $1 \leq j \leq M$ ve $x_j \in A$ olmak üzere $l_j \in \{x_j, x_j'\}$ olsun. O zaman l_j'lerin birbirleriyle mantık işlemleri altında (yani $\wedge$ veya $\vee$) birleştirildiği formüllere Boolean formülleri denir.

Tanım 3.2.8. Her j için $l_{j_1}, l_{j_2}, ...l_{j_k}$, birer biliteral olmak üzere,

$$C_j = l_{j_1} \vee l_{j_2} \vee \cdots \vee l_{j_k}$$

formülü k-madde (clause) olarak tanımlanır.

Tanım 3.2.9. $k, m \in \mathbb{N}$ olmak üzere ve her C_i, m-madde olmak üzere

$$C = C_1 \wedge C_2 \cdots \wedge C_k$$

formundaki her Boolean formüle m-CNF(conjuctive normal form) denir.

Tanım 3.2.10. k bir doğal sayı ve her C_i en fazla k-madde ise

$$C_1 \wedge C_2 ... \wedge C_n$$

formundaki boolean formüllerine CNF formunda formül denir.

Boolean formüller için çok önemli bir yere sahip teoremi ifade edelim.

Teorem 3.2.11. Her Boolean formülü CNF formunda yazılır.

Bu teoremin kanıtı için "Logic for Computer Science" [8] isimli kitap incelenebilir.

Tanım 3.2.12 (Boolean Tatmin Edilebilirlik (Boolean Satisfiability) Problemi (SAT))**.** SAT, CNF formunda verilen bir Boolean formülünün sonuç olarak 1 değerini alıp alamayacağını belirlemeye yönelik bir karar problemidir. Yani herhangi bir

$$\phi = C_1 \wedge C_2 \wedge \cdots \wedge C_m$$

olsun ve her C_i bir $k \in \mathbb{N}$ için k-madde olsun. Ayrıca $\{x_1, x_2, ..., x_n\}$ kümesi ϕ formülünü oluşturan biliterallerin kümesi olsun. O zaman öyle bir atama $\tau : \{x_1, x_2, \ldots, x_n\} \to \{0, 1\}$ var mıdır ki, ϕ formülü 1 değerini alsın? Bu soru SAT problemidir. Yani, SAT problemi CNF formunda verilmiş bir önermenin 1 değerini alıp alamayacağını kontrol eder.

Eğer en az bir tane böyle atama varsa, ilgili formül *doğrulanabilirdir (satisfiable)* olarak adlandırılır. Aksi halde formül *doğrulanamaz (unsatisfiable)* olarak kabul edilir.

SAT problemi, karmaşıklık kuramında merkezi bir rol oynar. SAT problemi, NP-Tam olarak kanıtlanan ilk problemdir. 1971 yılında Stephen Cook ve Leonid Levin tarafından kanıtlanan Cook-Levin Teoremi, SAT probleminin NP-Tam olduğunu göstermektedir [4]. Bölüm 3.1'in sonunda yaptığımız analojiye göre SAT probleminin NP'de olduğunu varsayabiliriz çünkü verilen bir boolean formülü ve çözüm adayı için çözümün doğruluğunu deneme yanılma yaparak kontrol edecek bir Turing makinesi vardır. Yani SAT $\in NP$ olduğu açıktır.

Teorem 3.2.13 (Cook-Levin, 1971)**.** Boolean tatmin edilebilirlik problemi (SAT) NP-Tamdır.

Yukarıdaki ifade bir başka deyişle, tüm NP problemleri SAT problemine polinom zamanda indirgenebilir, şeklinde ifade edilebilir.

Kanıt. Tüm NP problemlerinin SAT problemine polinomsal zamanda indirilebileceğini yani herhangi bir $L \in NP$ için $L \leq_p$ SAT olduğunu göstermeliyiz.

Şimdi öyle bir f fonksiyonu inşa edeceğiz ki bu fonksiyon:

1. L problemi üzerime tanımlanacak ve Turing makineleri kümesinden Boolean formüllerine gidecek,
2. Polinomsal zamanda çalışan bir Turing makinesiyle hesaplanabilecek,
3. Son olarak çıktısının doğrulanabilir olması için gerek ve yeter şart girdisinin cevabının Evet olması olacaktır,

şartlarını sağlamalıdır.

Şimdi L herhangi bir NP problemi ve n, L probleminin uzunluğu olsun (yani makul bir dilde L'yi ifade etmek için gereken harf sayısı). NP kümesinin tanımı gereği, L problemine $p(n)$ polinom olmak üzere $p(n)$ zamanda doğru cevap veren bir KTM (kararsız Turing makinesi) vardır. Dolayısıyla başlangıçta şeridinin $-p(n)$'inci karesinden n'inci karesine kadar harf olan(boşluk dahil) ve L'yi $p(n)$ zamanda çözen bir ,$M(\Pi, \Sigma, Q, q_0, q_y, q_n, \delta)$, Turing makinesi vardır. Şimdi $r = |Q| - 1$, $v = |\Pi| - 1$,$q_1 = q_y$ ve $\Pi = \{s_0, s_1, ..., s_m\}$, s_0 boş karakter ve L problemimizin M makinesine verildiği girdi, $s_1 s_2 .. s_n$ olmak üzere M makinesi için aşağıda gösterilen değişkenleri(bilateralleri) tanımlayalım.

Değişkenler \Özelikler	Aralık	Anlamı
$Q[i,k]$	$0 \leq i \leq p(n)$ $0 \leq k \leq r$	i'inci adımda M, q_k konumundadır.
$H[i,j]$	$0 \leq i \leq p(n)$ $-p(n) \leq j \leq p(n)+1$	i'inci adımda M'nin okur-yazar başlığı sonsuz şeridin j'inci karesindedir
$S[i,j,k]$	$0 \leq i \leq p(n)$ $-p(n) \leq j \leq p(n)+1$ $0 \leq k \leq v$	i'inci adımda, şeridin j'inci karesinde s_k harfi vardır.

Tablo 3.1 Bir Turing makinesini değişkenler ile ifade etme

Öncelikle belirtmekte fayda var ki eğer M makinesi L girdisi için $p(n)$ adımdan daha kısa sürede bitiş konumlarına ulaşıyorsa **kalan adımlar** için $Q[i,j]$, $H[i,j]$ ve $S[i,j,k]$ değişkenlerini sabit kabul edebiliriz. Şimdi L probleminin cevabı "Evet" tir ifadesi ile;

L problemini M makinesine girdi olarak verdiğimizde, M makinesinin bu girdiyle Turing makinesi adımlarıyla en fazla $p(n)$ adımda q_y konumunda son bulması ifadesi denktir. Şimdi böyle bir M makinesinin Tablo 3.1'de belirtilen değişkenlerini kullanarak bir SAT formülü inşa edeceğiz öyle ki L'nin cevabı "Evet" olacaktır ancak ve ancak inşa ettiğimiz SAT doğrulanabilir olacaktır.

Madde Grubu	Özellikler
G_1	Her adımda, M makinesi tam olarak 1 konumdadır.
G_2	Her adımda M makinesinin okur-yazar başlığı tam olarak bir karededir.
G_3	Her adımda şeridin her karesi en fazla bir harf içermektedir.
G_4	0'ıncı adımda makine verilen girdiye göre başlangıç ayarlarındadır.
G_5	$p(n)$'inci adımda makine q_y konumundadır.
G_6	Her $0 \leq i \leq p(n)$ için M makinesinin $i+1$'inci adımdaki durumuna, M makinesinin i'inci adımdaki durumundan δ fonksiyonuna göre 1 adımda gelinebilir.

Tablo 3.2 M makinesinin varlığını SAT problemine dönüştüreceğimiz madde grupları

Tablo 3.2'de belirttiğimiz grupları kullanarak bahsettiğimiz SAT formülünü oluşturacağız. Bu gruplar Tanım 3.2.8'de verdiğimiz maddelerin gruplarıdır, yani her bir grup maddelerden oluşmakta ve formül içindeki görevi, o gruba karşılık gelen özelliği garanti etmektir.

Amacımız bir $L \in NP$ için L'yi çözen bir M KTM'si yapmanın polinomsal olarak bir SAT problemine indirilebileceğini göstermek. Tablo 3.1'deki değişkenleri kullanarak M'yi ifade edeceğiz, Tablo 3.2'deki madde gruplarını kullanarak ise bir SAT formülü oluşturacağız. Bu oluşturduğumuz SAT formülü şunu sağlayacak: L probleminin cevabı "Evet" olacaktır ancak ve ancak oluşturduğumuz SAT formülü tatmin edilebilir. Eğer bunu sağlayan bir SAT formülünü n'ye göre polinom zamanda oluşturabilirsek, SAT'ın NP-Tam olduğunu göstermiş oluruz.

G_1, G_2, G_3, G_4 ve G_5 gruplarını:

1.
$$G_1 = \left(\bigwedge_{\substack{0 \le i \le p(n) \\ 0 \le j < j' \le r}} (Q'[i,j] \vee Q'[i,j']) \right) \wedge \bigwedge_{i=0}^{p(n)} \left(\left(\bigvee_{0 \le j \le r} Q[i,j] \right) \right)$$

2.
$$G_2 = \left(\bigwedge_{\substack{0 \le i \le p(n) \\ -p(n) \le j < j' \le p(n)+1}} \big(H'[i,j] \vee H'[i,j']\big) \right) \wedge \left(\bigwedge_{i=0}^{p(n)} \bigvee_{-p(n) \le j \le p(n)+1} H[i,j] \right)$$

3.
$$G_3 = \left(\bigwedge_{i=0}^{p(n)} \left(\bigwedge_{j=-p(n)}^{p(n)+1} \left(\bigvee_{0 \le k \le v} S[i,j,k] \right) \right) \right) \wedge \bigwedge_{\substack{0 \le i \le p(n) \\ -p(n) \le j \le p(n)+1 \\ 0 \le k < k' \le v}} \big(S'[i,j,k] \vee S'[i,j,k']\big)$$

4. M makinesine verdiğimiz girdi $s_{k_1} s_{k_2} \dots s_{k_n}$ olmak üzere
$$G_4 = \left(\bigwedge_{0 < i \le n} S[0,i,i] \right) \wedge \left(\bigwedge_{n+1 \le i \le p(n)+1} S[0,i,0] \right) \wedge \big(Q[0,0] \wedge H[0,1]\big)$$

5.
$$G_5 = Q[p(n),1]$$

olarak tanımlayalım.

Şimdi bütün bu G_i gruplarına karşılık gelen özelliklerin sağlandığını göstereceğiz.

G_1'in 1 değerini alabilmesi için:

$$\bigwedge_{\substack{0\leq i\leq p(n)\\ 0\leq j<j'\leq r}} \left(Q'[i,j] \vee Q'[i,j']\right)$$

ve

$$\left(\bigwedge_{i=0}^{p(n)} \left(\bigvee_{0\leq j\leq r} Q[i,j]\right)\right)$$

ifadeleri ayrı ayrı 1 değerini almalıdır. İkinci ifade M'nin, i'inci adımda en az bir konumda olmasını garanti ederken birinci ifade ise M'nin, i'inci adımda iki farklı konumda olmamasını garanti ediyor dolayısıyla G_1, Tablo 3.2'de belirtilen özelliği garanti etmiş olur.

G_2 madde grubunun 1 değerini alabilmesi için

$$\left(\bigwedge_{\substack{0\leq i\leq p(n)\\ -p(n)\leq j<j'\leq p(n)+1}} \left(H'[i,j] \vee H'[i,j']\right)\right)$$

ve

$$\left(\bigwedge_{i=0}^{p(n)} \bigvee_{-p(n)\leq j\leq p(n)+1} H[i,j]\right)$$

ifadelerinin ayrı ayrı 1 olması gerekmektedir. İlk ifadenin 1 olması, her i adımı için okur/yazar başlığın en fazla bir karede olmasını garanti ederken, ikinci ifadenin 1 olması ise her i adımı için okur/yazar başlığının en az bir karede olduğunu garanti etmektedir.

G_3'ün 1 değerini alabilmesi için

$$\left(\bigwedge_{i=0}^{p(n)} \left(\bigwedge_{j=-p(n)}^{p(n)+1} \left(\bigvee_{0\leq k\leq v} S[i,j,k]\right)\right)\right)$$

ve

$$\bigwedge_{\substack{0\leq i\leq p(n)\\ -p(n)\leq j\leq p(n)+1\\ 0\leq k<k'\leq v}} \left(S'[i,j,k] \vee S'[i,j,k']\right)$$

ifadelerinin her biri 1 değerini almalıdır. İlk ifadenin 1 olması her i adımı için $-p(n)$ ile $p(n)+1$ arasındaki her konumda en az 1 harf olmasını garanti ederken, ikinci ifade ise her i adımı için $-p(n)$ ile $p(n)+1$ arasındaki her karenin en fazla 1 harf olmasını garanti etmektedir.

G_4 madde grubunun 1 değerini alabilmesi için $Q[0,0]$, $H[0,1]$ ifadelerinin 1 olması gerekmektedir. Bu ifadelerin 1 olması sırasıyla makinenin başlangıç konumunda ve okur/yazar başlığın başlangıç karesinde olduğunu garanti etmektedir. Ayrıca

$$\left(\bigwedge_{0<i\leq n} S[0,i,i]\right)$$

ve

$$\left(\bigwedge_{n+1\leq i\leq p(n)+1} S[0,i,0]\right)$$

ifadelerinin de 1 olması gerekmektedir. Bu ifadeler de sırasıyla; 0'ıncı adımda ilk n karenin girdinin kendisi, n'den büyük karelerde ise boş karakter (yani k_0) olmasını garanti etmektedir.

Son olarak G_5'in değerinin 1 olması ise $p(n)$'inci adımda M makinesinin kabul konumunda (yani q_y konumunda) olduğunu garanti etmektedir.

G_6 grubundaki maddeleri ise 2 alt gruba ayıracağız. Bu iki alt grup sırasıyla aşağıdaki iki koşulu garanti etme işlevine sahip olacak:

1. i'inci adımda okur/yazar başlık sonsuz şeridin j'inci karesinde değilse, j'inci kare; i'inci adımdan $i+1$'inci adıma geçerken değişmez.

2. i'inci adımdan $i+1$'inci adıma geçerken aşağıdaki maddeler M'nin transfer fonksiyonuna göre değişecek:

- Okur/Yazar başlığın bulunduğu kare
- Okur/Yazar başlığın bulunduğu karedeki harf
- M'nin bulunduğu konum

G_6'nın birinci grubunu, kısaca $G_{6,1}$, aşağıdaki gibi oluşturacağız:

$$G_{6,1} = \bigwedge_{\substack{0\leq i\leq p(n) \\ -p(n)\leq j\leq p(n)+1 \\ 0\leq l\leq v}} \big(S'[i,j,l] \vee H[i,j] \vee S[i+1,j,l]\big)$$

Ve G_6'nın ikinci grubunu, kısaca $G_{6,2}$, oluşturmak için devam eden adımları takip edeceğiz. Eğer bir M makinesi $p(n)$'inci adımdan önce q_y veya q_n konumlarından birine gelmişse makinenin durumu sabit kalacağından, bir başka deyişle makine duracağında, ikinci grubu inşa ederken tanımlayacağımız Δ, k' ve l' değelerini makinenin durup durmaması durumuna göre iki farklı şekilde tanımlayacağız. $G_{6,2}$ uzun bir ifade olduğu için bu ifadeyi anlaşılır parçalar halinde yazmak adına üç ayrı parçaya böleceğiz ve bu üç paraçayı "ve" kapılarıyla birleştirip $G_{6,2}$ elde edeceğiz.

$$G_{6,2,1} = \left(\bigwedge_{\substack{0\leq i\leq p(n) \\ -p(n)\leq j\leq p(n)+1 \\ 0\leq l\leq v \\ 0\leq k\leq r}} \big(S'[i,j,l] \vee H'[i,j] \vee Q'[i,k] \vee H[i+1,j+\Delta]\big) \right)$$

$$G_{6,2,2} = \left(\bigwedge_{\substack{0\leq i\leq p(n) \\ -p(n)\leq j\leq p(n)+1 \\ 0\leq l\leq v \\ 0\leq k\leq r}} (S'[i,j,l] \vee H'[i,j] \vee Q'[i,k] \vee Q[i+1,k']) \right)$$

$$G_{6,2,3} = \left(\bigwedge_{\substack{0 \le i \le p(n) \\ -p(n) \le j \le p(n)+1 \\ 0 \le l \le v \\ 0 \le k \le r}} (S'[i,j,l] \vee H'[i,j] \vee Q'[i,k] \vee S[i+1,j,l']) \right)$$

yukarıda kullanılan Δ, k' ve l', M makinesinin durduğu durumda $\Delta = 0$, $k' = k$ ve $l' = l$ şeklinde tanımlanır. M makinesinin durmadığı durumda ise $\delta(q_k, s_l) = (q_{k'}, s_{l'}, \Delta)$ şeklinde tanımlanır.

Daha önceden de belirttiğimiz üzere G_6 maddesini

$$G_6 = G_{6,1} \wedge G_{6,2,1} \wedge G_{6,2,2} \wedge G_{6,2,3}$$

ile tanımlayalım. $G_{6,1}$ maddesinin değerinin 1 olması yukarıda bahsettiğimiz birinci koşulu, $G_{6,2,1} \wedge G_{6,2,2} \wedge G_{6,2,3}$ önermesinin değerinin 1 olması ise yukarıda bahsettiğimiz ikinci koşulu garanti edecektir.

Son olarakta S ifadesi

$$S = \bigwedge_{i=1}^{6} G_i$$

formülü ile tanımlayalım.

Bütün G_i'lerin 1 olması yukarıda bahsedilen özellikleri garanti edeceği için:

$$L \text{ probleminin cevabı “Evet” olur } \iff S \text{ SAT problemi doğrulanabilirdir.}$$

Sonuç olarak herhangi bir NP problemi L için bir Boolean formülü oluşturduk. Bu formül, n'nin polinomu cinsinsinden sayıda madde ve biliteral içerdiği kolayca tanımlardan görülebilir. Bu maddeleri ve biliteralleri bağlamak ise sabit zaman alacağından [4] bu formülü polinom zamanda oluşturmuş oluruz. Böylelikle herhangi bir NP problemi SAT problemine polinomsal zamanda indigenmiş olup ispat tamamlanır. □

SAT probleminin NP-Tam olduğunun kanıtlanmasının ardından Richard Karp 21 adet farklı probleme SAT probleminin polinomsal zamanda indirgenebileceğini gösterip böylelikle bu problemlerin de birer NP-Tam problem olduklarını kanıtladı [7]. Tezimizin konusu olan çizge boyama problemi de bu problemlerden birisidir. Yani verilen herhangi bir çizgenin verilen bir $k \geq 3$ doğal sayısı için k renkle uygun boyanıp boyanamayacağı problemi bir NP-Tam problemdir. 4. Bölümde bu gerçeğin ispatı sunulacaktır. [7] kaynağında sunulan çizge teorisindeki diğer meşhur NP-Tam problemlerin bazıları şunlardır:

- **Kliklerle Kaplama Problemi**: Verilen bir $G = (V, E)$ çizgesi ve pozitif bir tam sayı $K \leq |E|$ için, öyle bir $m \leq K$ sayısı ve V'nin $V_1, V_2, \ldots, V_m \subseteq V$ alt kümeleri var mıdır ki:

  1. Her V_i, G üzerinde bir tam alt çizgeyi temsil etsin,
  2. Her $ab \in E$ kenarı için, a ve b aynı alt küme V_i içinde yer alsın.

  Başka bir deyişle, G üzerindeki kenar kümesini, en fazla m tane tam alt çizgenin birleşimi olarak ifade etmek mümkün müdür?

- **Klik Problemi**: $G(V, E)$ bir çizge ve $K \leq |V|$ pozitif tam sayı olsun. G'de K tane köşe içeren bir klik var mıdır?

- $G(V, E)$ bir çizge ve K pozitif tam sayı olsun. Öyle bir, bire bir

  $$f : V \to \{1, 2, ...|V|\}$$

  fonksiyonu var mıdır ki

  $$\sum_{uv \in E} |f(u) - f(v)| \leq K$$

  olsun?

NP ve P kümelerini okura daha iyi aktarmak adına bu tezde genel geçer bilinen tanımlardan daha farklı bir şekilde ele aldık. Genel olarak P ve NP kümesinin elemanları sadece karar

problemlerinden oluşmak zorunda değildir. Yani eğer bir problemi kararlı bir Turing makinesi, girdisinin boyutunun polinomsal fonksiyonu kadar adımda çözüyorsa bu problem P kümesindedir. Benzer şekilde eğer bir problemi kararsız bir Turing makinesi, girdisinin boyutunun polinomsal fonksiyonu kadar adımda çözüyorsa bu problem NP kümesindedir. Dikkatle incelenirse, Cook-Levin teoremi kapsamında SAT probleminin NP-tamlığı sonucunun, söz konusu P ve NP sınıflarının tanımlarını da kapsadığı görülecektir.

Bir sonraki bölümde çizge boyama probleminin bir NP-tam problem olduğu detayları ile ele alınacaktır.

4. Çizge Boyama Problemi ve NP-Tamlık

Teknik ayrıntılara geçmeden önce, ilerleyen kısımlarda kullancağımız bazı önemli temel tanımlar ve kavramsal çerçeve sunulacaktır.

4.1 Temel Tanımlar

Bir önceki bölümde NP ve P sınıflarının ve SAT probleminin tanımlarını vermiştik. Ayrıca SAT probleminin NP-Tam sınıfında olduğunu Cook-Levin teoremi ile kanıtlamıştık. Bu bölümde verilen herhangi bir SAT probleminin çözümünün olup olmadığını bulmanın, herhangi bir $k \geq 3$ için verilen bir $G(V, E)$ çizgesinin k renk kullanılarak uygun boyanıp boyanamayacağına polinomsal anlamda eşit olduğunu, yani polinomsal zamanda birbirlerine indigenebileceğini, göstereceğiz. Böylelikle çizge boyama probleminin de bir NP-Tam problem olduğunu göstermiş olacağız. Çizge boyama probleminin SAT problemine polinomsal zamanda indirgenebileceği çok açıktır. Dolayısıyla biz, SAT probleminin çizge boyama problemine polinomsal zamanda indirgenebileceğini göstereceğiz.

Bu gerçeği aşağıda verilen sıralamayı göz önüne alarak göstereceğiz:

1. Verilen herhangi bir SAT problemi 3-SAT problemine polinomsal zamanda indirgenir.
2. Verilen herhangi bir 3-SAT problemi 3-Renklendirme problemine polinomsal zamanda indirgenir.
3. Verilen herhangi bir 3-Renklendirme boyama problemi k-Renklendirme boyama problemine polinomsal zamanda indirgenir.

Görüleceği üzere burada yeni bazı problemlerin isimlerini vermiş olduk. Şimdi yukarıda bahsedilen tüm bu problemlerin tanımlarını aşağıda sırayla verelim. Bu problemler ile ilgili daha fazla detay için [9] kaynağı incelenebilir.

- **SAT Problemi**: Verilen bir CNF formatındaki Boolean formülünün 1 değerini alıp alamayacağını belirleme.
- **3-SAT Problemi**: Verilen bir 3-CNF formatındaki Boolean formülünün 1 değerini alıp alamayacağını belirleme.
- **3-Renklendirme Problemi**: Verilen bir çizgenin köşelerinin 3 renk ile uygun boyanıp boyanamayacağını belirleme.
- k**-Renklendirme Problemi**: Verilen bir çizgenin köşelerinin k renk ile uygun boyanıp boyanamayacağını belirleme.

4.2 SAT Probleminin 3-SAT Problemine İndirgenmesi

Teorem 4.2.1. SAT problemi, 3-SAT problemine polinomsal zamanda indirgenebilir.

Kanıt. Genelliği bozmadan verilen bir SAT probleminin formülü

$$\Phi = C_1 \wedge C_2 \wedge \cdots \wedge C_n$$

olsun. Burada her C_i herhangi bir k doğal sayısı için

$$C_i = x_1 \vee x_2 \vee \cdots \vee x_k$$

formunda olacaktır. Eğer $k = 1$ ise yani $C_i = x_1$ tipinde ise iki yeni d_1 ve d_2 Boolean değişkenleri tanımlayarak

$$C_i = (x_1 \vee d_1 \vee d_2) \wedge (x_1 \vee d_1' \vee d_2) \wedge (x_1 \vee d_1 \vee d_2') \wedge (x_1 \vee d_1' \vee d_2')$$

eşitliğini kullanıp C_i'yi 3 tane biliteral içeren maddeler cinsinden yazabiliriz. Eğer $k = 2$ ise yani $C_i = (x_1 \vee x_2)$ tipinde ise bir d Boolean değişkeni tanımlayıp

$$C_i = (x_1 \vee x_2 \vee d) \wedge (x_1 \vee x_2 \vee d')$$

eşitliğini kullanıp C_i'yi 3 tane biliteral içeren maddeler cinsinden yazabiliriz. Son olarak eğer $k > 3$ ise $x_z = x_3 \vee x_4 \vee ... \vee x_k$ değişkenini tanımlayarak

$$C_i = x_1 \vee x_2 \vee x_z$$

şeklinde ifade edebiliriz.

Sonuç olarak her SAT formülünü o SAT formülüne denk 3-SAT formülüne dönüştürebiliriz. Ayrıca bu gösterilen bir Boolean değişkeni sabit bir zamanda oluşturabileceğimiz için SAT formülünü 3-SAT formülüne polinomsal zamanda dönüştürürebiliriz. Böylelikle ispat tamamlanmış olur. □

4.3 3-SAT Probleminin 3-Renklendirme Problemine İndirgenmesi

Teorem 4.3.1. 3-SAT problemi, 3-Renklendirme problemine polinomsal zamanda indirgenebilir.

Kanıt. Verilen herhangi bir 3-SAT formülü için bir çizge oluşturup eğer oluşturduğumuz bu çizge 3 renk ile uygun bir şekilde boyanabilen bir çizge oluyor ise verilen 3-SAT formülünün sağlanabileceğini, diğer taraftan oluşturduğumuz çizge 3 renk kullanılarak boyanamıyorsa verilen 3-SAT formülünün sağlanamadığını göstereceğiz.

Genelliği bozmadan kabul edelim ki

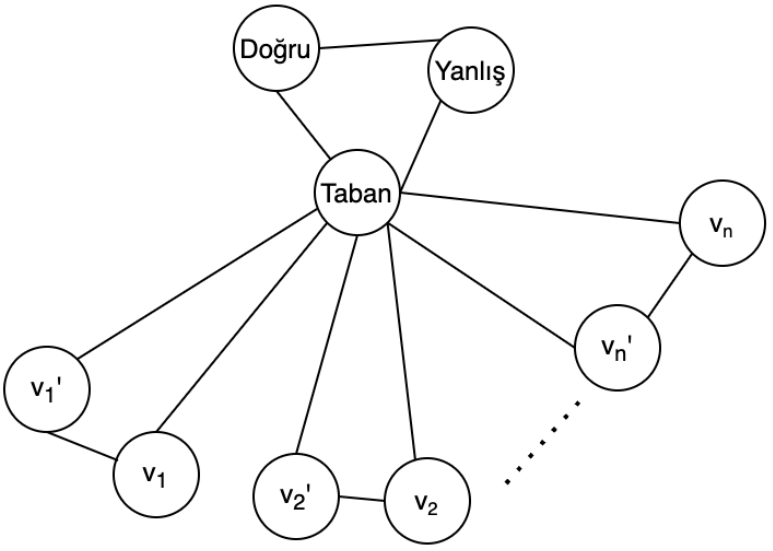
$$C_1 \wedge C_2 \wedge C_3 ... \wedge C_m \tag{1}$$

tipinde bir formülümüz olsun. Burda her C_i, tam olarak 3 adet literalin "veya" kapılarıyla bağlanması sonucunda oluşmaktadır. Örneğin; $i \in \{1, 2, \ldots, m\}$ için her bir madde

$$C_i = v_{i_1} \vee v_{i_2} \vee v_{i_3}$$

tipindedir. Burada $\{v_1, v_2, v_3, \ldots, v_n\}$ kümesi C_i maddelerinde kullanılan tüm literallerin (yani mantık değerlerinin) kümesi olsun.

Bu v_i literalleri oluşturacağımız çizgenin köşelerinin bazıları olacak. İlk olarak Şekil 4.1'de görüldüğü gibi bir çizge oluşturalım. Daha sonra uygun mantıkla bu çizgeyi büyüteceğiz.



Şekil 4.1 : 3-SAT – 3-Renk denkliğini göstermede kullanılan taban çizge

Çizgemizi "Doğru, Yanlış, Taban" isimleri ile adlandırdığımız 3 renk ile boyamaya çalışacağız. Daha sonra 3-Sat formülündeki her bir v_i literaline çizgede boyandığı rengin değerini vereceğiz, yani "Doğru" ise 1 "Yanlış" ise 0. Kolayca görüleceği üzere Şekil 4.1 de oluşturduğumuz çizge bize her bir v_i literalinin "Doğru" ya da "Yanlış" renklerinden biriyle boyanmasını garanti ediyor. Ayrıca Şekil 4.1'de oluşturduğumuz çizge bize her v_i literalinin değilinden farklı renkte olmasını ve her v_i' literalinin de "Doğru" ya da "Yanlış" renklerinden biriyle boyanmasını garanti eder. Bu durum önemlidir çünkü C_i maddeleri v_i'lerin değillerini de içerebilir.

Şu ana kadar her v_i ve v_i' literalerinin "Doğru" ya da "Yanlış" renklerinden biriyle boyanmasını ve her v_i ve v_i''nin farklı renklerle boyanmasını garantilemiş olduk. Unutmayalım ki

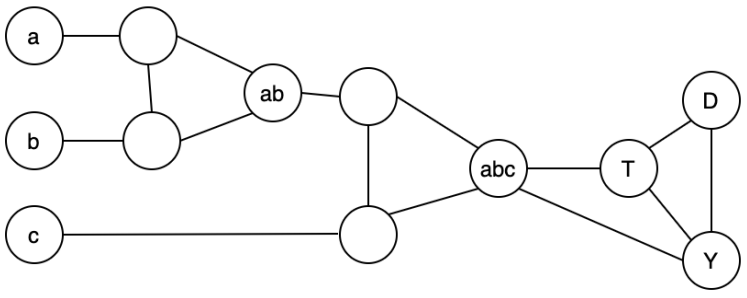
amacımız öyle bir çizge oluşturmak ki, eğer oluşturduğumuz çizge 3 renk ile uygun boyanabiliyorsa bize verilen (1) formülü 1 değerini alabiliyor boyanamıyorsa bu formül 1 değerini alamıyor olmalıdır.

Şimdi her C_i maddesi için C_i'nin 1 değerini alabileceğini garanti altına almaya çalışacağız. Unutulmamalıdır ki formüldeki her C_i maddesi

$$C_i = a \vee b \vee c$$

formundadır. Burdaki a, b ve c yukarıda bahsettiğimiz $v_1, v_2, ..., v_n$ literallerinden veya onların değillerinden biridir.

Bu aşamadan sonra Şekil 4.1'de verilen çizgeyi her C_i maddesi için genişletip, en son oluşan çizgenin 3 renk ile boyanabilmesiyle verilen 3-SAT formülünün 1 değerini alabilmesinin eş değer olduğunu göreceğiz.



Şekil 4.2 : 3-SAT - 3-Renk denkliğini göstermede kullanılan tabana eklenecek çizge

Burada 3-SAT formülende verilmiş her C_i için Şekil 4.2 deki yapıyı Şekil 4.1'e ekleyeceğiz. Şekil 4.2'de verilen çizgedeki "T, D, Y" köşeleri Şekil 4.1'de verilen çizgenin "Taban, Doğru, Yanlış" köşeleridir. Ayrıca Şekil 4.2'deki "a, b, c" köşeleri, karşılık gelen C_i'nin literallerinin Şekil 4.1'deki uygun karşılıklarıdır. Yani her bir C_i için Şekil 4.2'yi, Şekil 4.1'e uygun bir şekilde eklerken; Şekil 4.2'deki "a, b, c, T, D, Y" köşelerini yeniden oluşturmadan Şekil 4.2'deki kalan köşeleri ve kenarları Şekil 4.1'de verilen çizgeye "a, b, c, T, D, Y" noktalarından monteliyoruz.

Son olarak Şekil 4.2'deki çizgedeki "a, b, c" köşeleri Şekil 4.1'deki çizgenin literalleri olduğundan bunların "Doğru" ya da "Yanlış" renkleriyle boyanacakları kesindir. Şimdi dikkatli incelersek Şekil 4.2'deki "ab" köşesisinin her zaman için $a \vee b$ rengiyle boyanabileceğini aynı şekilde, "abc" köşesinin ise her zaman için $a \vee b \vee c$ rengiyle boyanabileceğini gözlemlenir. Ayrıca "abc" köşesinin "Taban" ve "Yanlış" renkleriyle boyanan köşelere komşu olması "abc"'nin "Doğru" rengine boyanmasını mecbur kılmaktadır.

Yukarıda anlatıldığı ve Şekil 4.2'de gösterildiği gibi Şekil 4.1'deki çizgeyi her C_i için genişleterek G çizgesini elde edelim. O halde verilen 3-SAT formülü 1 değerini alabilmesi için gerek ve yeter şart bu G çizgesinin 3 renk ile uygun boyanabilmesi olacaktır. Taban çizgeyi oluşturmak, 3-SAT probleminde kullanılan toplam biliteral sayısıyla doğru orantılı zaman alacağından, yine aynı şekilde her bir eklenti sabit zaman alacağından ve sonuç olarak toplamda 3-SAT formülündeki biliteral sayının polinomsal cinsinden eklenti yapılacağı için bu indirgeme polinomsal zamanda olacaktır. Böylelikle ispat tamamlanmış olur. □

4.4 k-Renklendirme Probleminin 3-Renklendirme Problemine İndirgenmesi

Teorem 4.4.1. Her 3-Renklendirme problemi, k-Renklendirme problemine polinomsal zamanda indirgenebilir.

Kanıt. Bu ifadenin ispatı görece daha basittir. Verilen bir G çizgesinin 3-renk ile boyanıp boyanamayacağını belirlemek istiyoruz. H çizgesini, K_{k-3} çizgesini G çizgesinin her köşesiyle birleştirerek elde edelim. O halde:

H çizgesi k renk ile uygun boyanabilir $\iff$ G çizgesi 3 renk ile uygun boyanabilir

ifadesi elde edilecektir. Bu indirgemenin polinomsal zaman indirgemesi olduğu açık olup ispat tamamlanır. □

Son olarak aşağıdaki alt bölümde yukarıda yaptıklarımız özetlenmiştir.

4.5 Sonuç

Teorem 4.5.1. k-Renklendirme Problemi NP-tamdır.

Kanıt. Önceki bölümlerde sırasıyle şunları gösterdik

- Verilmiş herhangi bir SAT A problemi için polinomsal zamanda elde edilebilen öyle bir 3-SAT B problemi vardır ki:

$$A\text{, 1 değerini alabilir} \iff B\text{, 1 değerini alabilir}$$

- Verilmiş herhangi bir 3-SAT B problemi için polinomsal zamanda elde edilebilen öyle bir G çizgesi vardır ki:

$$B\text{, 1 değerini alabilir} \iff \text{G çizgesi 3 renkle uygun boyanabilir}$$

- Verilmiş herhangi bir G çizgesi ve $k \geq 3$ tamsayısı polinomsal zamanda öyle bir H çizgesi elde edilebilir ki:

$$H \text{ çizgesi } k \text{ renkle uygun boyanabilir} \iff G \text{ çizgesi 3 renkle uygun boyanabilir}$$

Ayrıca Cook-Levin Teoremi'nden biliyoruz ki SAT problemi bir NP-Tam problemdir. Yukarıdaki denkliklerden ve Cook-Levin teoremi göz önüne alındığında, verilen herhangi bir $G(V, E)$ çizgesinin k renk ile uygun boyanıp boyanamayacağı problemi bir NP-Tam problemdir. □

Not edilmelidir ki yukarıdaki denklikler aynı zamanda 3-SAT ve 3-renklendirme problemlerinin de birer NP-Tam problem olduğunu göstermektedir.

Özetle çizge boyama problemleri hesaplanabilirlik teorisinde önemli bir yere sahiptirler. Verilen herhangi bir çizgenin k-renk ile uygun bir biçimde boyanıp boyanamayacağını polinom zamanda belirleyebilirsek, verilen herhangi bir SAT probleminin 1 değerini alıp almayacağını da polinom zamanda belirleyebilmiş oluruz. Böylelikle NP kümesindeki her problemin cevabını polinom zamanda belirleyebiliriz. Bir başka deyişle $NP = P$ eşitliği gösterilmiş olur. Fakat günümüzde halen verilen herhangi bir çizgenin polinomsal zamanda uygun bir şekilde boyanabileceğiniz gösteren algoritmalar mevcut değilidir. Hatta böyle bir algoritmanın var olabileceği bile belirsizdir.

Fakat pratikte çizge boyamanın birçok alanda uygulaması bulunduğundan verilen çizgelerin kaç renkle boyanabileceğini belirlemek (optimal olmasa dahi) önem arz etmektedir. Bir sonraki bölümde literatürde bilinen ve pratikte sıklıkla kullanılan önemli bazı algoritmalar ele alınacaktır.

5. Çizge Boyama Problemi ve Bazı Algoritmalar

Bu bölümde verilen bir $G(V, E)$ çizgesini uygun bir şekilde boyayan bazı algoritmaları inceleyeceğiz. Bölüm 5.1'de inceleyeceğimiz Tam Sayılı Programlama algoritması verilen bir çizgenin tam olarak kromatik sayısını belirler. Ancak bilinmektedir ki Tam Sayılı Programlama algoritması polinom zamanda çalışmaz, yani örneğin 10000 köşesi olan ortalama bir büyüklükteki bir çizgenin kromatik sayısını hesaplaması binlerce yıl sürebilir. Bu bağlamda Bölüm 5.1'de inceleyeceğimiz Tam Sayılı Programlama algoritması, diğer bölümlerde inceleyeceğimiz algoritmalardan ayrışmaktadır ve bu algortimaya "Değerlendirme ve Karşılaştırma" bölümünde değinmeyeceğiz. Diğer bölümlerde inceleyeceğimiz algoritmaların hepsi polinom zamanda çalışmaktadır.

Bölüm 5.2, 5.3, 5.4 ve 5.5'te inceleyeceğimiz algoritmaların amacı, bir çizgenin kromatik sayısını tam olarak belirlemek değil, bir çizgeyi uygun bir biçimde mümkün olan en az renkle boyamaktır. Bir başka deyişle bu boyamalar en optimal boyamalar olmak zorunda değildirler. Bu bölümlerde anlatacağımız algoritmaların hepsi polinom zamanda çalışmaktadır.

Farklı algoritmaları kıyaslarken ; bu algoritmaların aynı çizgeleri kaç renk kullanarak boyadıklarını karşılaştıracağız. Bu karşılaştırmayı 2. Bölümde tanımladığımız $G_{n,p}$ rastgele çizgeler yardımı ile yapacağız.

Sırasıyla her bir algoritmanın çalışma metodolojisine ve çalışma zamanına değineceğiz. Literatürde daha başka algortimaların da mevcut olduğu unutulmamalıdır [10].

Bu bölümün sonunda farklı n ve p değerleri için oluşturacağımız $G_{n,p}$ rastgele çizgeleri üzerinde Bölüm 5.3, 5.4 ve 5.5'te ele aldığımız 3 algoritmayı test edip performanslarını karşılaştıracağız. Ayrıca bu bölümde kullanılan bütün kaynak kodları Ekler-A bölümünde verilmiştir.

Öncelikle ilerleyen alt bölümlerde vereceğimiz algoritmlarında bazılarında ihtiyacımız olan tanımları verelim.

Tanım 5.0.1. $G(V, E)$ bir çizge ve $k \in \mathbb{N}$ olsun. G çizgesinin düzensiz k-boyanması, her köşenin bu köşeye komşu olan köşelerin farklı renge boyanması şartı aranmadan k renk kullanılarak boyanmasıdır.

Tanım 5.0.2. Bir $G(V, E)$ çizgesinin uygun veya düzensiz k-boyanması: $0 \leq i, j \leq k$ ve $i, j \in \mathbb{N}$, için,

$$i \neq j \implies S_i \cap S_j = \emptyset \quad \text{ve} \quad \bigcup_{i=0}^{k} S_i = V$$

olmak üzere $S = \{S_1, S_2, ..., S_k\}$ köşelerin parçalanışı kümesiyle ifade edilir. Eğer S'bir uygun boyamaysa $\forall A \in S$ ve $\forall u, v \in A$ için $uv \notin E$ olacaktır.

Yukarıdaki tanımda betimlenen S_i kümesi, i rengi ile boyanan köşelerin kümesi olacaktır.

Önceki açıklamaların ışığında, artık ele alınacak algoritmaları sırayla ve sistematik bir şekilde sunabiliriz.

5.1 Tam Sayılı Programlama Algoritması

Bu bölümde kullanılan algoritma bir Tam Sayı Optimizasyon algoritmasıdır. Bu algoritma optimizasyon problemlerinde sıklıkla karşılaşılan bir algoritmadır. Bir çizgenin en az kaç renkle uygun bir biçimde boyanacağını belirlemek bir optimizasyon problemi olacağından bu algoritma yardımıyla bu sorunun cevabı belirlenebilir.

Şimdi verilen bir $G(V, E)$ bir çizgesi ve $V = \{v_1, v_2, ..., v_n\}$ köşeleri için bilinen bir uygun boyamayı $X_{n\times n}$ ve $Y_{n\times 1}$ matrislerini aşağıdaki koşulları sağlayacak şekilde oluşturarak ifade edebiliriz.

1. $1 \leq i, j \leq n$ için

$$x_{ij} = \begin{cases} 1 \text{ eğer } v_i, j \text{ rengine boyanmışsa} \\ 0 \text{ eğer } v_i, j \text{ rengine boyanmamışsa} \end{cases}$$

2. $1 \leq i \leq n$ için

$$y_{i1} = \begin{cases} 1 \text{ eğer i rengine boyanmış en az bir köşe varsa} \\ 0 \text{ eğer i rengine boyanmış köşe yoksa} \end{cases}$$

X ve Y matrisleri ile $G(V, E)$ çizgesi için aşağıdaki tam sayı optimizasyon problemini ele alalım:

$$\forall v_i v_l \in E, \forall j \in \{1, 2, .., n\} \quad \text{ve } \forall x_{ij}, y_{j1} \in \{0, 1\} \quad \text{için} \quad x_{ij} + x_{lj} \leq y_{j1}$$

ve

$$\forall v_i \in V \quad \text{için} \quad \sum_{j=1}^{n} x_{ij} = 1,$$

kısıtları sağlanmak üzere;

$$\sum_{j=1}^{n} y_{j1}$$

toplamının minimum değeri kaçtır?

Yukarıdaki tamsayı optimizasyon probleminde, hesaplanan bu minimum değer, X ve Y matrisleri için 1. ve 2. koşularını da sağlayacağından G çizgesinin kromatik sayısı olacaktır. Ancak bu algoritma maalesef polinom zamanda çalışmamaktadır [10]. Diğer bir deyişle tam sayılı programlama algoritması da bir NP problemdir.

Fakat yukarıdaki tam sayı optimizasyon probleminde X ve Y matrislerinin elemanlarının 0 veya 1 olma şartını, hatta tam sayı olması durumunu, kaldırırsak bu optimizasyon problemi doğrusal programlamayla polinom zamanda çözülebilir. Böylece elde edeceğimiz değer tam sayı optimizasyon problemine bir alt sınır olacağından, G çizgesinin kromatik sayısı için bir alt sınır bu algoritma sayesinde belirlenebilir. Fakat bu alt sınır genel bir alt sınır değil verilen çizgeye özel bir alt sınır olacağından matematiksel olarak genellemek mümkün olmayacaktır.

5.2 Greedy Algoritması

Greedy algoritması verilen bir $G(V, E)$ çizgesini aşağıda anlatıldığı şekilde boyayan algoritmadır.

$C = \{c_1, c_2, c_3, ...\}$ kümesi renk kümesi olmak üzere, V kümesinin elemanları rastgele bir $v_1, v_2, \ldots, v_n$ sırasına dizilir. Sonra sırayla her bir v_i köşesi; ona komşu olan ve daha önce boyanmış olan köşelerin boyandığı renklerden farklı bir renk olmak suretiyle C kümesindeki en küçük indisli renkle boyanır.

Görüleceği üzere akla ilk gelen mantıkla çalışan en ilkel boyama algoritmasıdır.

Şekil 5.1'de Greedy algoritmasının sudo kodu mevcuttur.

```
    GREEDY (S ← ∅, π)
(1) for i ← 1 to |π| do
(2)       for j ← 1 to |S|
(3)             if (S_j ∪ {π_i}) Bağımsız kümeyse
(4)                   S_j ← S_j ∪ {π_i}
(5)                   break
(6)             else j ← j + 1
(7)       if j > |S| then
(8)             S_j ← {π_i}
(9)             S ← S ∪ S_j
```

Şekil 5.1 : Greedy Algoritmasının Sudo kodu

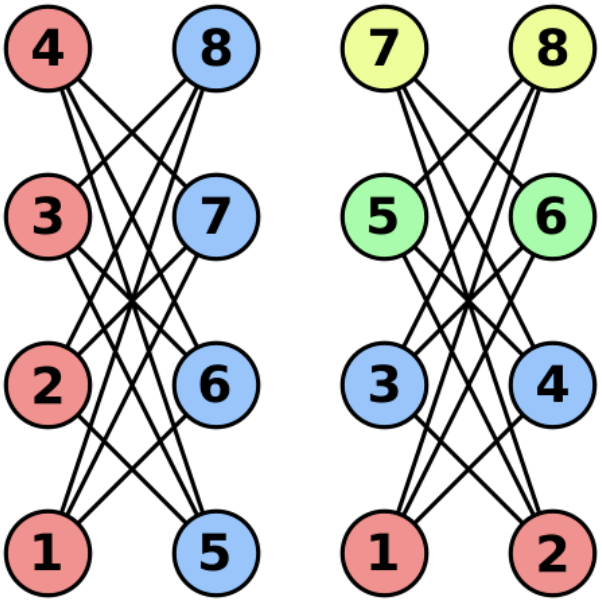
Modern bilgisayarların n eleman içeren bir kümeyi rastgele bir sıraya dizemesi sabit c zaman alır, algoritmanın asimptotik çalışma zamanını incelerken bu c sabitini göz ardı edebiliriz.[10] Algoritma i'inci adımda v_i köşesini boyamak için ilk $i-1$ köşenin her birinin hangi renge boyandığını kontrol etmek durumundadır. Ayrıca bu kontrollerin her biri de yine sabit c' zamanında olacaktır. Genel olarak bir algoritmanın asimptotik çalışma zamanını

incelerken; sabitlerin hepsini 1 olarak kabul edebiliriz. Bu yüzden i'inci köşenin boyanmasının alacağı zaman $i-1$ olacaktır (aslında unutulmamaldır ki $(i-1)$'in bir sabit katı).[10] Dolayısıyla, Greedy algoritmasının çalışma zamanı

$$\sum_{i=0}^{n-1} i = O(n^2)$$

olarak hesaplanır. Yani Greedy algoritması, n köşeli bir çizge üzerinde asimptotik olarak n^2 zamanda çalışır.

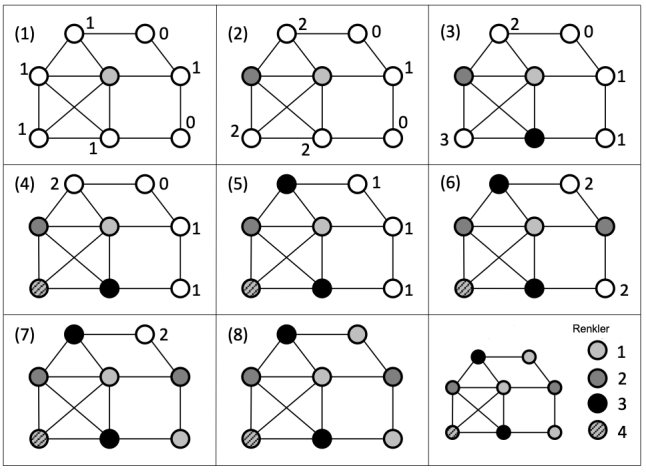
Greedy algoritması yukarıda bahsettiğimiz formuyla çok tercih edilmez, bunun nedeni aşağıda Şekil 5.2'de sunduğumuz üzere 8 köşeli küçük bir çizgede bile iki farklı sıralama kullanılması durumunda birinde 2 farklı renk kullanarak çözüm verirken diğerinde 4 farklı renk kullanarak bir çözüm vermektedir. Bu fark daha büyük çizgelerde çok daha fazla olabileceğinden Greedy algoritması çok tercih edilen bir algoritma değildir.



Şekil 5.2 : Aynı çizgenin 2 farklı sıralamayla boyanması.

5.3 DSatur

DSatur algoritması mekanizması itibariyle Greedy algoritmasına benzer. Farklı olarak DSatur algoritmasında köşelerin sıralanması en başta olmaz, her köşe boyandıktan sonra bir sonraki boyanacak köşe belirlenir. Boyamanın herhangi bir adımında boyanmamış bir köşenin *doygunluk sayısı* o ana kadar boyanmış komşularının sayısı olarak belirlenir. Bir sonraki adımda boyanacak köşe ise boyanmamış ve en yüksek doygunluk sayısına sahip köşedir. Algoritmanın devam eden aşamasında Greedy algoritmasında olduğu gibi her köşe boyanan komşularından farklı bir renkle boyanmak suretiyle renkler kümesindeki en küçük indisli renkle boyanır. DSatur algoritmasına ait bir boyama Şekil 5.3'te gösterilmiştir. Verilen şekildeki 8 köşeli çizge 8 adımda 4 renk ile boyanmıştır ve her adımda bütün köşelerin doygunluk sayısı gösterilmiştir. DSatur algoritmasının sudo kodu Şekil 5.4'te verilmiştir.



Şekil 5.3 : DSatur algoritmasına bir örnek.

Özetlemek gerekir ise DSatur algoritması aşağıdaki şekilde çalışır:

1. Çizgedeki her köşenin doygunluk sayısı başlangıçta 0 olarak ayarlanır.
2. Çizgedeki en yüksek dereceli köşe, eğer birden fazlaysa herhangi biri rastgele seçilerek, renkler kümesindeki en küçük indisli renkle boyanır.
3. Boyanmayan köşelerin doygunluk sayısı, boyanmış komşularının sayısı olarak değiştirilir.

DSATUR $(\mathcal{S} \leftarrow \emptyset, X \leftarrow V)$

```
(1)  while X ≠ ∅ do
(2)      Choose v ∈ X
(3)      for j ← 1 to |S|
(4)          if (S_j ∪ {v}) Bağımsız kümeyse
(5)              S_j ← S_j ∪ {v}
(6)              break
(7)          else j ← j + 1
(8)      if j > |S| then
(9)          S_j ← {v}
(10)         S ← S ∪ S_j
(11)     X ← X − {v}
```

Şekil 5.4 : DSatur algoritmasını sudo kodu.

4. En yüksek doygunluk sayısına sahip köşe seçilir. Eğer bir eşitlik durumu olursa daha yüksek dereceli köşe seçilir. Yine eşitlik olursa en yüksek doygunluk ve dereceli köşelerden biri rastgele seçilir.

5. Seçilen köşe renkler kümesinden boyanabileceği en küçük indisli renkle boyanır.

DSatur algoritması bazı durumlarda optimal çözümleri verir. Örneğin aşağıda vereceğimiz teorem bu gerçeğin önemli örneklerinden biridir [10].

Teorem 5.3.1. DSatur algoritması kromatik sayısı 2 olan bir çizgeyi 2 renk kullanarak boyar.

5.4 Tabu-Col

Tabu-Col algortiması ilk olarak 1987'de ortaya atılmıştır. Çeşitli varyasyonları mevcuttur. Bizim burada inceleyeceğimiz varyasyonu "Geliştirilmiş Tabu-Col" algoritması olarak bilinen varyansyonudur. Detaylar için [10] kaynağı incelenebilir.

Bu algoritma çalışma mantığı aşağıdaki gibidir:

1. İlk olarak bir l doğal sabiti belirlenir.

2. Ardından $V = \{v_1, v_2, ..., v_n\}$ köşeleri ile verilen bir $G(V, E)$ çizgesini DSatur algoritmasıyla k renk kullanarak boyar,eğer herhangi bir adımda bir köşe k renkten biriyle komşularından farklı renkle boyanamıyorsa, o köşe rastgele bir renkle boyanır. Böylece G çizgesinin düzensiz k-boyaması

$$S = \{S_1, S_2, ..., S_k\}$$

elde edilir.

3. Komşu olan iki köşenin aynı renge boyanması *çakışma durumu* olarak adlandırılır. Elde eldilen bu S boyaması için çakışma sayısı, toplam çakışma durumu olarak tanımlanır ve bu sayı $f(S)$ ile gösterilir.

4. Algoritma, bu k sayısı için $n \times k$ boyutundaki sıfır matrisi $T_{n\times k}$ ile $1 \leq i \leq n$, $1 \leq j \leq k$ için

$$c_{ij} := v_i \text{ köşesinin } S_j \text{ rengi ile boyanmış komşularının sayısı}$$

kuralı ile belirlenen $n \times k$ boyutundaki $C_{n\times k}$ matrisini tanımlar.

5. Algoritma döngünün ilk adımında $S' = \{S'_1, S'_2, ..., S'_k\}$ boyamasını şu şekilde elde eder: Her v_v köşesi,bu köşe S_i rengine boyanmış olsun, için C matrisinde $c_{vj} - c_{vi}$, değerini minimum yapan j'yi bulur. Ve bu işlemi her satır için yapar ve bu satırların herbirinde bulunan en küçük j'yi bulur.

6. Bulduğu en küçük j'ye, j', bulunduğu satıra da v' diyelim.

7. O zaman algoritma: $S_{i'}$ rengine boyanan $v_{v'}$ köşesinin rengini $S_{j'}$ rengine değiştirip yeni S' boyamasını elde eder. Ve $f(S') = f(S) + c_{v'j'} - c_{v'i'}$ olarak hesaplanır.

8. Bunun ardından en başta belirlenen bir l sabiti ve döngünün bulunduğu adım,t, için $T_{v'i'} = t + l$ olarak güncellenir. Bu $v_{v'}$ köşesinin $t + l$'inci adıma kadar i' rengini alamayacağı anlamına gelir ve bu durumlar 5. adımda yok sayılır.

9. Ve yine v''nin her komşu köşesi u için: $C_{ui'} = C_{ui'} - 1$ ve $C_{uj'} = C_{uj'} + 1$ güncellenerek C matrisi güncellenmiş olur.Ve algoritma bir sonraki adıma geçer.

Eğer herhangi bir adımın sonunda toplam çakışma sayısı 0 olursa, algoritma k değerini bir düşürerek $k - 1$ değerine göre elde ettiği son boyama ile 5.adımdan devam eder.

Tabu-Col algoritması herhangi bir k değerine ulaşılınca veya bir döngüde yapılabilecek maksimum adım sayısına ulaşınca,bu değer kullanıcı tarafından belirlenir, son bulur.

5.5 HEA (Hibrit Evrimsel Algoritma)

Hibrit Evrimsel Algoritma 1999 yılında Phillep Gao ve Jin Kao Hao tarafından geliştirilmiştir [10].

Boyanmak üzere bir $G(V, E)$ çizgesi verilsin. Bir k doğal sayısı belirleyelim. Verilen bu G çizgesini, DSatur algoritmasını kullanarak farklı şekillerde k renk kullanarak boyamaya çalışacağız. Aslında DSatur algoritması derecesi en yüksek köşeden boyamaya başlar. Fakat burada amacımız farklı boyamalar elde etmek olduğundan boyanılacak ilk köşe rastgele belirlenir. Ayrıca unutulmamalıdır ki DSatur algoritması verilen G çizgesini k renkte boyamayı garanti etmez. Yani boyama sırasında bir köşe k renkten biriyle komşularından farklı olacak şekilde boyanamayabilir. Böyle bir durumda o köşe rastgele bir renkle boyanacak. Böylece G çizgesinin bir k-düzensiz boyanması elde edilecektir.

Yukarıda anlatılan metodla G çizgesinin daha önceden belirlenen bir N doğal sayısı için N farklı k-düzensiz boyamalarını elde edelim. Ve bu boyamalarla *popülasyon* adını verilen kümeyi oluşturalım. Şimdi bu popülasyon kümesindeki *ebeveyn* boyamalar olarak adlandırılan boyamaları aşağıdaki örnekte anlatacağımız gibi çiftleyerek *çocuk* boyamalar olarak adlandırılan yeni boyamalar elde edeceğiz.

Örneğin, populasyondan iki adet ebeveyn boyama alacağız. Bu ebeveyn boyamalarla bir adet çocuk boyama elde edeceğiz. Ardından aldığımız iki ebeveyn boyamayı popülasyondan çıkartıp onların yerine daha sonra basit bir örneğini sunacağımız mantıkla elde ettiğimiz çocuk boyamayı koyacağız. Normalde; evrimsel algoritmalarda, çocuk algoritmalar ebeveyn algoritmalarının kötü performans göstereninin yerine konur ancak yukarıda anlattığımız gibi HEA'da çocuk algoritma iki ebeveyninin yerine populasyona girer [10]. Bu anlamda bu özellik bu tür yapıdaki algoritmalarda bir yenilik olarak görülebilir.

Bir popülasyonun içinde yalnızca bir boyama kaldığında; o boyamada bir çakışmanın içinde olan bütün köşelerin renkleri silinir ve kalan boyama DSatur algoritmasına sokularak uygun bir boyama elde edilir. Bu elde edilen boyama HEA'nın ulaştığı nihai uygun boyamadır. Tablo 5.1'de, verilen bir S_1 ve S_2 ebeveyn boyamlarından, S' çocuk boyamanın elde edilmesi gösterilmiştir.

S_1	S_2	S'
$\{v_1, v_2, v_3\}$, $\{v_4, v_5, v_6, v_7\}$, $\{v_8, v_9, v_{10}\}$	$\{v_3, v_4, v_5, v_7\}$, $\{v_1, v_6, v_9\}$, $\{v_2, v_8, v_{10}\}$	$\{\}$
$\{v_1, v_2, v_3\}$, $\{v_8, v_9, v_{10}\}$	$\{v_3\}$, $\{v_1, v_6, v_9\}$, $\{v_2, v_8, v_{10}\}$	$\{v_4, v_5, v_6, v_7\}$
$\{v_1, v_3\}$, $\{v_9\}$	$\{v_3\}$, $\{v_1, v_9\}$	$\{v_4, v_5, v_6, v_7\}$, $\{v_2, v_8, v_{10}\}$
$\{v_9\}$	$\{v_9\}$	$\{v_4, v_5, v_6, v_7\}$, $\{v_2, v_8, v_{10}\}$, $\{v_1, v_3\}$
$\{\}$	$\{\}$	$\{v_4, v_5, v_6, v_7\}$, $\{v_2, v_8, v_{10}, v_9\}$, $\{v_1, v_3\}$

Tablo 5.1 HEA için çiftleme örneği.

Tablo 5.1'de gösterilen çiftleme örneği aşağıda anlatıldığı gibi çalışmaktadır:

1. DSatur algoritmasıyla yukarıda anlatıldığı gibi farklı boyamalarla elde edilen popülasyondan rastgele iki boyama olarak S_1 ve S_2 ebeveyn olarak seçilmiş olsun ve Tablo 5.1'in ilk satırına, yeni bir çocuk boyama, S' ,elde etmek için eklensin.

2. İkinci satırda, ebeveyn boyamalardan en büyük renk sınıfı, beraberlik durumunda rastgele olacak şekilde, bulunduğu ebeveynden silinip, o renk sınıfını oluşturan köşeler diğer ebeveynden de silinerek, S' çocuk boyamasına eklenmiştir.

3. Ebeveynlerin kalan hallerindeki en büyük renk sınıfı bir önceki adımda anlatıldığı gibi silinip çocuk boyamaya eklenmiştir.

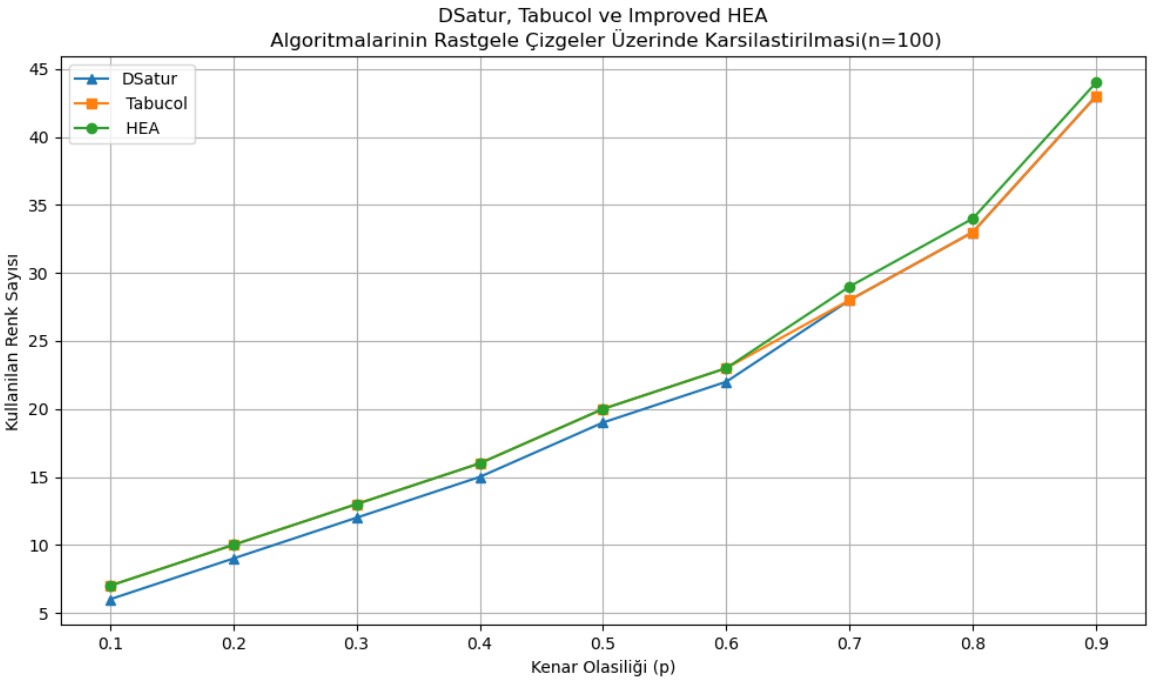
4. Bu adımlar ebeveyn boyamalarda bir köşe kalmayıncaya kadar devam ettirilir. Nihai olarak çocuk boyama S' elde edilmiş olur.

5.6 Sonuç ve Değerlendirme

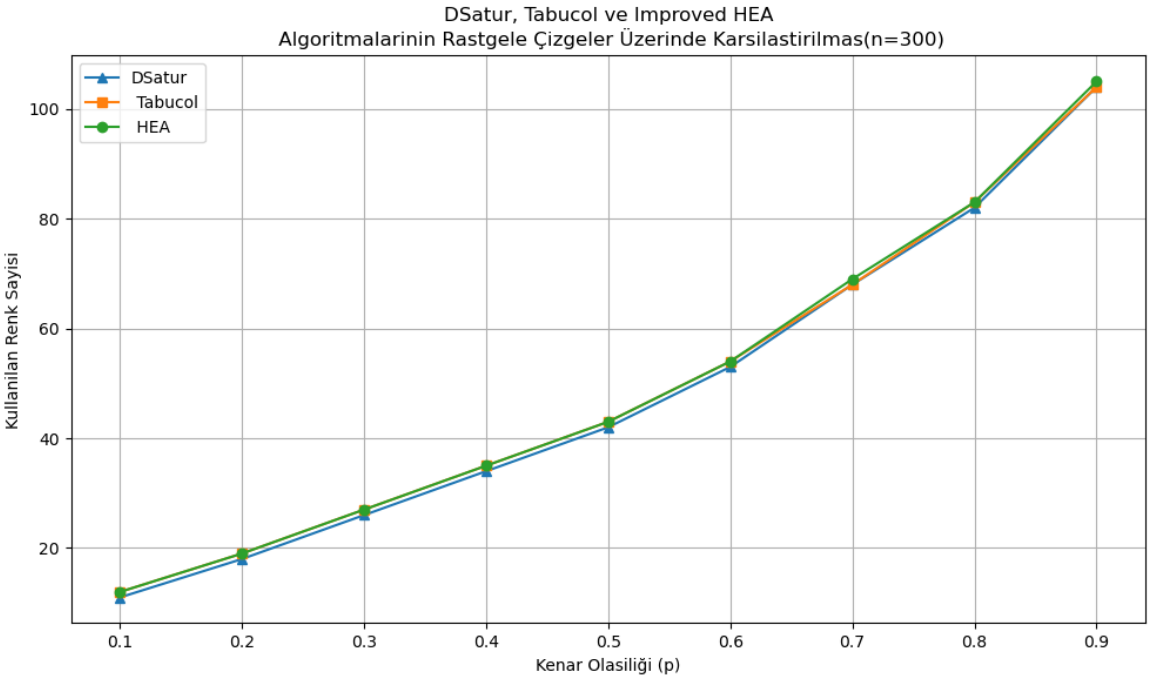
Bu bölümde HEA, Tabu-Col ve DSsatur algoritmalarını rastgele çizgeler üzerinde farklı kenar olasıklıkları, p, değerleri ve farklı köşe sayıları için karşılaştıracağız.

Şekil 5.5'te 100 köşeli rastgele çizgeler için algoritmaların aynı çizgeleri kaç farklı renk kullanarak boyadığı gösterilmiştir. Şekil 5.6'da ise 300 köşeli rastgele çizgeler için karşılaştırma gösterilmiştir. Bu grafikleri çizmek için kullanılan kodlar Ekler-A bölümüne eklenmiştir. Genel olarak; kod çalıştığında x eksenindeki her bir p değeri için, verilen bir n sayısı(örneklerde 100 ve 300 değerlerini kullandık) tane köşe içeren rastgele çizgeler oluşturularak, algoritmaların oluşturulan bu çizgeleri kaç farklı renkle boyadığı y ekseninde gösterilmiştir.

Grafiklerden de görebileceği üzere 100 köşeli ve 300 köşeli çizgelerde algoritmalar genel olarak benzer performanslar sergilediğinden ve DSatur en hızlı çalışan algoritma olduğundan bir çizge boyama problemi için kullanılacak en makul algoritmanın DSatur olduğunu düşünüyoruz.



Şekil 5.5 : 100 Köşeli Karşılaştırma



Şekil 5.6 : 300 Köşeli Karşılaştırma

5.7 Çizge Boyama Problemi İçin Bazı Uygulama Örnekleri

Önceki bölümlerde de bahsettiğimiz gibi çizge boyama problemi NP-Tam olduğu için çözülebileceği zaman bakımından birçok önemli probleme denktir. Yalnızca bu durum bile, çizge boyama problemini araştırmaya değer çok önemli bir problem yapmaktadır. Bunun yanında çizge boyama problemleri günlük hayatta birçok yerde farkında olmadan karşımıza çıkmaktadır. Bu bölümde günlük hayatta karşımıza çıkan çizge boyama problemi örneklerinden bazılarına kısaca değinerek bu konuyu somutlaştırmayı hedefliyoruz.

5.7.1 Ders-Sınav Programı Hazırlama

Okullarda aynı öğrencisi olan derslerin sınav veya ders saatleri çakışabilmektedir. Bu noktada bir ders programı veya sınav programı hazırlamanın en zor tarafı; ortak öğrencisi olan derslerin, ders saatlerinin veya sınav saatlerinin kesişmeyecek saat aralıklarında olmak şartıyla her ders için ders saati veya sınav saati belirlemektir.

Bölüm 2'de de bahsettiğimiz gibi dersleri bir çizgenin köşeleri olarak belirleyip, iki köşe arasındaki kenarın varlığını da, iki ders arasında ortak öğrenci olduğunu belirtmek üzere tanımlarsak, oluşturacağımız çizgeyi uygun bir şekilde boyamak ile ortak öğrencisi olan derslerin ders ve sınav saatlerinin kesişmemesini sağlayacak bir program yapmak eş değer problemler olacaktır.

Bölüm 6'da, önceki alt bölümlerde ele alınan Greedy Algoritması kullanılarak, ortak öğrencisi bulunan derslerin sınav saatlerini, kesişmeyecek şekilde ayarlayan ve başka kriterleri de kontrol eden bir sınav programı oluşturucusu yapacağız.

5.7.2 Oturma Planı Hazırlama

Organizasyonlarda oturma planını belirlemek oldukça karışık olabilir. Oturma planları genel olarak; kişilerin kendi aralarındaki ilişkileri, kişiler arasındaki hiyerarşik ilişkileri, masaların kapasiteleri vb. gibi birçok durum göz önüne alınarak yapılır.

Örneğin, her masanın en fazla 6 kişi alabildiği ve n kişinin katılıdığı bir organizasyon için $1 \leq i, j \leq n$ olmak üzere

$$w_{ij} := \text{eğer } i \text{ kişisi ,} j \text{ kişisi ile oturabiliyorsa 0, oturamıyorsa 1}$$

şeklinde tanımlanan bir $W_{n\times n}$ matrisini oluşturalım. Bu W matrisinin simetrik olduğu açıktır. Şimdi bu W matrisini kullanarak n kişi için şöyle bir çizge oluşturacağız:

- Her insan bir köşe tarafından temsil edilmek üzere, eğer w_{ij} değeri 1 ise i ve j kişileri arasında bir kenar olsun, yani komşu köşeler olsunlar, 0 ise bu köşeler arasında bir kenar olmasın.

Eğer oluşturduğumuz bu çizgeyi uygun bir şekilde boyarsak ve her renk grubunu ortak bir masaya oturtursak, farklı masalarda oturması gereken herkesi farklı masalara oturtmayı garanti etmiş oluruz. Fakat bu yaklaşımla masaların kapasitelerinin aşılmayacağını garanti etmiş olmayız.

Bu durumda bu örnek çizge boyama probleminin özel bir halidir. *A Guide to Graph Coloring* kitabının 6. Bölümünde bu özel durum detaylı bir şekilde incelenmiştir. [10] Bu özel durumu incelemek isteyen okurlarımız bu kaynağın bahsi geçen bölümünü inceleyebilir.

6. Sınav Programı Hazırlama Algoritması

Bir önceki bölümde görüldüğü üzere çizge boyama probleminin akademik bir birime en uygun uygulaması sınav veya ders programı hazırlanması olabilir. Bu kapsamda, bu bölümde tezin bir çıktısı olarak "Hacettepe Üniversitesi, Fen Fakültesi, Matematik Bölümü" adına python programlama dilinde hazırladığımız uygulamayı anlatacağız.

Bu uygulama:

- Ortak öğrencisi olan derslerin sınav saatlerinin farklı saat dilimlerinde olması,
- Aynı dönemin zorunlu derslerinin sınavlarının farklı günlerde olması,
- Bir sınav için mümkün olan en az sayıda derslik kullanılması,

amaçlarına göre hazırlanmıştır.

Oluşturduğumuz model, bu amaçları aşağıda listelemiş olduğumuz kısıtlara göre sağlayan bir modeldir:

- Her dersin ismi ve o dersi alan öğrenci sayısı manuel olarak girilebilecektir.
- Aynı saatte olmaması gereken sınavlar manuel olarak seçilebilecektir.
- Aynı gün olmaması gereken sınavlar manuel olarak seçilebilecektir.
- Dersliklerin sayısı ve kapasitesi manuel olarak ayarlanabilecektir.
- Sınav periyodları aksi belirtilmedikçe sabit olacak, aksi durumlar manuel olarak belirlenebilecektir.

6.1 Uygulamanın Tasarımı ve Çalışma Prensibi

Uygulama 6 ana mekanizmadan oluşmaktadır. Uygulamaya, https://github.com/muratersen/SinavTakvimiYapici, adresinden erişilebilir. Şimdi bu mekanizmaları detaylıca inceleyeceğiz.

6.1.1 classroom_updater.py

Bu dosya kullanıcıya; sınava girilebilecek salonların isimlerini ve kapasitelerini girebileceği bir arayüz sağlar. Girilen bilgiler aynı klasörde classroom_data.csv dosyasında python programlama dilindeki sözlük formatında saklanır.

6.1.2 course_updater.py

Bu dosya kullanıcıya; derslerin isimlerini ve dersi alan öğrenci sayılarını girebileceği bir arayüz sağlar. Girilen bilgiler aynı klasörde data_dict.csv dosyasında yine sözlük formatında saklanır.

6.1.3 olamaz.py

Bu dosya kullanıcıya; hangi derslerin sınavlarının çakışamayacağını girebileceği bir arayüz sunar. Girilen bilgiler aynı klasörde relations.csv dosyasında benzer şekilde sözlük formatında saklanır. relations.csv'de anahtarlar (keys) ders isimleriyken değerler (values) ders isimlerinden oluşan listelerdir. Bu durum, keylerin valuelerle çakışamayacağı anlamına gelir.

6.1.4 gun.py

Bu dosya kullanıcıya; hangi derslerin sınavlarının farklı gün olması gerektiği bilgisini girebileceği bir arayüz sunar. Girilen bilgiler aynı klasörde gun.csv dosyasında sözlük formatında saklanır. gun.csv'de anahtarlar ders isimleriyken değerler ders isimlerinden oluşan listelerdir. Bu durum, keylerin valuelerle aynı gün olamayacağı anlamına gelir.

6.1.5 schedule.py

Bu dosya kullanıcıya; özel bir dersin özel bir zaman diliminde veya özel bir uzunlukta yapılacağı bilgisini girebilmesi için bir arayüz sunar. Girilen bilgiler aynı klasörde schedule.csv dosyasında sözlük formatında saklanır.schedule.csv'de anahtarlar ders isimleriyken değerler sırasyıyla sınavın yapılacağı gün, sınavın başlama saati, sınavın bitiş saati, sınavın yapılacağı hafta değerlerinden oluşan bir listedir.

6.1.6 main.py

Bu program yukarıda bahsedilen 5 program tarafından oluşturulan 5 dosyayı kullanarak çıktı olarak bir sınav takvimi vermektedir. Verilen sınav takviminde; her sınavın yapılacağı tarih, başlama ve bitiş saatleri açıkça belirtilmiştir.

7. Tartışma ve Sonuç

Bu çalışmada basit yönsüz çizgelerin boyanmasını incelenmiştir. Bu bölümde ise farklı biçimde tanımlanan boyama problemlerinden bahsedeceğiz. İlk olarak liste boyama problemini ardından kesirli boyama problemini tanıtacağız. Son olarak kenar boyama problemini tanıtıp bu problemi biraz daha detaylı bir şekilde inceleyip Vizing Teoremini kanıtlayacağız ve çizge boyama problemiyle ilişkilendirmeye çalışacağız. Vizing Teoremini ilk kez gördüğümüzde doğru dönüşümleri tanımlayarak verilen herhangi bir çizgenin kromatik sayısını en fazla 1 hatayla bulabileceğimizi düşünmüştük. Ancak önemli bir engelden dolayı bunun doğru olmadığını gördük. Bu durumu detaylı bir şekilde kenar boyama alt başlığı altında inceledik. İlk olarak liste boyama ve kesirli boyama problemlerini tanımlayalım.

Tanım 7.0.1. $G(V, E)$ bir çizge olsun. G'nin her köşesine k **farklı** renk içeren bir liste verilsin. Her köşe kendi listesinden bir renkle boyanmak koşuluyla, listelerdeki renkler ne olursa olsun, G çizgesinin uygun boyanmasına k-liste boyanması, G çizgesine de k-liste boyanabilir bir çizge denir.

Verilen bir G çizgesinin k-liste boyanmasını mümkün yapan en küçük k değerine ise o çizgenin liste kromatik sayısı denir ve $\chi_l(G)$ olarak gösterilir. Bütün listelerdeki renkler aynı olursa çizge boyama problemi elde edileceğinden $\chi(G) \leq \chi_l(G)$ eşitsizliği kolayca görülebilir. Verilen bir çizgenin liste kromatik sayısını bulmak yine bir NP-Tam bir problemdir [11]. Liste boyanması problemi, frekans/kanal ayarlamalarında sıklıkla kullanılmaktadır [11].

Bazı önemli tanımları verelim.

Tanım 7.0.2. Verilen bir $G(V, E)$ çizgesinin her köşesinin b renk kullanılarak komşu olan köşelerin ortak renk içermeden boyanmasına b-katlamalı boyama denir. Bir G çizgesinin b-katlamalı boyanmasını mümkün kılan en küçük toplam renk sayısı ise $\chi_b(G)$ olarak gösterilir.

Tanım 7.0.3. Verilen bir $G(V, E)$ çizgesi için;

$$\lim_{b\to\infty} \frac{\chi_b(G)}{b}$$

ifadesi G çizgesinin kesirli kromatik sayısı olarak tanımlanır ve $\chi_f(G)$ olarak gösterilir.

Yukardaki tanımda verdiğimiz limit gerçekten yakınsaktır çünkü:

$$\chi_{a+b}(G) \leq \chi_a(G) + \chi_b(G)$$

eşitsizliği her G çizgesi ve a, b doğal sayıları için doğrudur [11]. Sonuç olarak $\chi_f(G) \leq \chi(G)$ eşitsizliği sağlanmaktadır.

7.1 Kenar Boyama

Bu bölümde kenar boyama problemini tanıtıp Vizing Teoremini kanıtlayacağız. Ardından bu teorem üzerine önemli bir tartışma sunacağız.

Çizge boyama probleminde verilen bir çizgenin köşelerini, komşu olan köşeleri farklı renge boyamak koşuluyla boyamıştık. Kenar boyamada ise verilen bir çizgenin kenarlarını, ortak köşesi olan kenarları farklı renkle boyamak koşuluyla, boyayacağız. Bu tanıma göre bir çizgenin kenarlarının boyanmasına olanak veren en az renk sayısına ise o çizgenin kenar kromatik sayısı adı verilir. Bir $G(V, E)$ çizgesinin kenar kromatik sayı $\chi_k(G)$ olarak ifade edilir. Şimdi kenar boyamanın formel tanımını ve ardından α/β yolu tanımını verip ardından Vizing Teoremini kanıtlayacağız.

Tanım 7.1.1. $G(V, E)$ çizgesinin kenarlarının, ortak köşesi olan kenarları farklı renkle boyamak koşuluyla, boyanmasına kenar boyama denir. Bir çizgenin kenar boyanmasına olanak veren en küçük renk sayısına ise o çizgenin kenar kromatik sayısı denir ve $\chi_k(G)$ olarak gösterilir.

Tanım 7.1.2. $G(V, E)$ çizgesinin kenarları uygun bir şekilde boyanmış olsun. $x \in V$ ve α, β iki farklı renk olmak üzere, x köşesinden başlayıp dönüşümlü olarak α ve β renkleriyle boyanmış kenarları takip ederek giden patikaya x çıkışlı α/β-patika adı verilir. Bu patikadaki kenar sayısına ise patikanın uzunluğu denir.

Burda kolayca gözlemlenebilecek iki özellik vardır: [12]

- α/β patikaların uzunluğu 0 olabilir. Yani böyle bir patika olmayabilir.
- Uygun bir kenar boyaması için her köşeden çıkan α/β patikası var ise biriciktir.

Tanım 7.1.3. $G(V, E)$ çizgesinin kenarları

$$C = \{c_1, c_2, ..., c_{\chi_k(G)}\}$$

renkleri kullanılarak uygun bir şekilde boyansın. $x \in V$ ve $c_i \in C$ olmak üzere; x köşesinden çıkan c_i rengiyle boyanan bir kenar yoksa, c_i rengi x köşesinde eksiktir denir.

Teorem 7.1.4 (Vizing Teoremi)**.** $G(V, E)$ bir çizge olsun. O halde

$$\Delta G \leq \chi_k(G) \leq \Delta G + 1$$

eşitsizlikleri sağlanır.

Kanıt. $G(V, E)$ bir çizge olsun. Tanımlar gereği $\Delta G \leq \chi_k(G)$ eşitsizliği açıktır. $\Delta G + 1 \geq \chi_k(G)$ eşitsizliğini G'nin kenar sayısı üzerine tümevarımla kanıtlayacağız.

1. Eğer G'nin hiçbir kenarı yoksa, $\Delta G = 0$ olur ve G'nin kenarları 0 renk kullanılarak boyanabilir. Dolayısıyla teorem 0 kenarı olan çizgeler için doğrudur.
2. Şimdi, $m \in \mathbb{N}$ olmak üzere teoremin kenar sayısı m'ye kadar olan bütün çizgelerde doğru olduğunu varsayıp, $m + 1$ kenarlı çizgelerde de doğru olduğunu göstermeye çalışacağız.
3. $G(V, E)$ kenar sayısı $m + 1$ olan bir çizge olsun. Bir önceki adımı kullanarak, $\Delta G \leq \chi_k(G) \leq \Delta G + 1$ eşitsizliğini göstermeye çalışacağız. $xy \in E$ olmak üzere $G' = G - xy$ olsun. 2. adımdan G''nin kenarlarının, en fazla $\Delta G' + 1$ renk kullanılarak boyanabileceğini biliyoruz. G''nin kenarlarını

$$R = \{r_1, r_2, ..., r_{\Delta G'+1}\}$$

renklerini kullanarak c_i boyamasıyla şöyle boyayalım:

$$c_i : \ E \to R \text{ öyle ki aynı köşeyi içeren kenarlar farklı renk olacak}$$

G''de her köşede R kümesinden en az bir renk eksiktir. Şimdi herhangi bir c_i boyaması ve $x, y \in V$ için

$$\alpha, \beta \in \{r_1, r_2, ..., r_{\Delta G'+1}\}$$

sırasıyla x, y köşelerindeki eksik renkler olsunlar. Bu durumda:

$$\chi_k(G) > \Delta G + 1 \iff \ y\text{'den başlayan } \alpha/\beta \text{ patikası } x\text{'de biter.}$$

olması gerekir. Öncelikle bu denkliği gösterelim:

($\Rightarrow$) Eğer y'den çıkan α/β patikası x'de bitmiyorsa, y'de β eksik olduğundan bu patikadaki α ve β'ları yer değiştirirsek G''deki kenar boyamasının **uygunluğunu** koruruz ve y'de α eksik olmuş olur. Şimdi hem x'te hem y'de α eksik olacağından, sildiğimiz xy kenarını α'ya boyayarak G'nin en fazla $\Delta G + 1$ renkle boyanabileceğini gösteririz.

($\Leftarrow$) Diğer taraftan G çizgesinin kenarları en fazla $\Delta G + 1$ renk kullanılarak boyanabiliyorsa; G''nin uygun bir boyanmasında x ve y köşelerinde eksik olan aynı renk vardır. $\alpha = \beta$ seçersek y'de başlayan α/β patikası x'te bitmemiş olur. Sonuç olarak ilgili denkliği ispatlamıış oluruz.

Amacımız bu denkliği kullanarak G'nin kenarlarının $\Delta G + 1$ renk ile boyanabildiğini göstermek. Şimdi $xy_0 \in E$ olsun. c_0 da; $G - xy_0$'ın $R = \{r_1, r_2, ..., r_{\Delta(G-xy_0)+1}\}$ renklerinin kullanıldığı bir uygun kenar boyaması olsun. $\alpha \in R$ olmak üzere c_0 boyamasında, α x'te eksik olsun. $y_0, y_1, ..., y_k$ köşe dizisi de x'in komşularından oluşan ve $\forall 0 < i \leq k$ için $c_0(xy_i)$ renginin y_{i-1} köşesinde eksik olma şartını sağlayan en uzun

dizi olsun. $c_1, c_2, ..., c_k$, uygun kenar boyamalarını sırasıyla $G - xy_i$ çizgeleri için şöyle tanımlayacağız.

$$\text{her } 0 \leq j < i \text{ için } c_i(xy_j) = c_0(xy_{j+1})$$

$$c_i(xy_i) \text{ tanımsız}$$

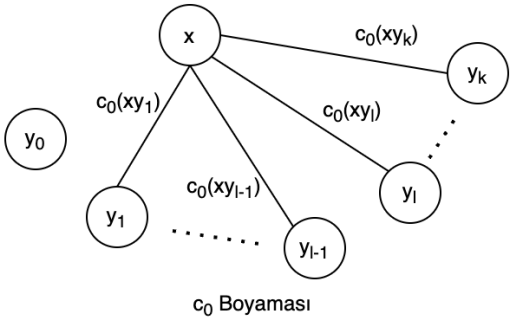
$$\text{diğer durumlarda } c_i(e) = c_0(e)$$

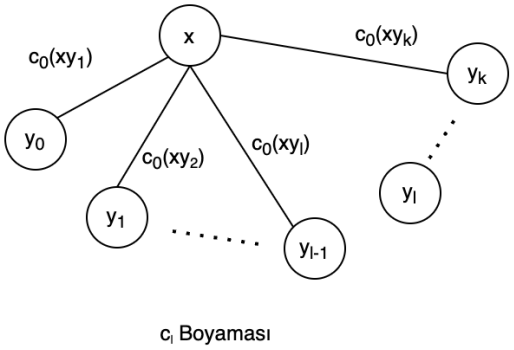
Bu boyamaların görselleştirimiş hali Şekil 7.1 de sunulmuştur. Dikkat edilirse her c_i için x'deki eksik renklerin aynı olduğu gözükür. β, c_0'a göre y_k'daki eksik renk olsun. Şimdi c_k'ya göre y_k'dan çıkan α/β patikasına bakalım ve buna P patikası diyelim, daha önceden ispatlandığı üzere bu patika x'te bitmek zorunda aksi halde G'yi $\Delta G + 1$ renk kullanarak boyamış oluruz. Eğer, c_k'ya göre y_k'dan çıkan α/β patikası x'te bitiyorsa bu patikanın son kenarı β rengiyle boyanmıştır. $\{y_i\}$ dizisi maksimal olduğundan bir $0 < l \leq k$ için $c_0(xy_l) = c_{l-1}(xy_l) = c_k(xy_{l-1}) = \beta$ olur. Dolayısıyla y_{l-1} köşesi P patikasının içindedir. Şimdi c_{l-1} boyamasına göre y_{l-1}'den çıkan α/β patikasına, kısaca P^* a, bakalım. c_i lerde sadece x'e bağlı kenarların rengi değiştiğinden P'deki ve P^*'daki kenarlar y_k ile y_{l-1} arasında aynı renge boyanmış aynı kenarlar olacaktır. P^* patikası y_k köşesine α rengiyle gelecektir ve β rengi y_k'da eksik olduğundan P^* patikası y_k'da bitecektir. Sonuç olarak G çizgesinin kenarları, c_{l-1} boyamasına göre $\Delta G + 1$ tane renk kullanılarak boyanabilecektir.

□

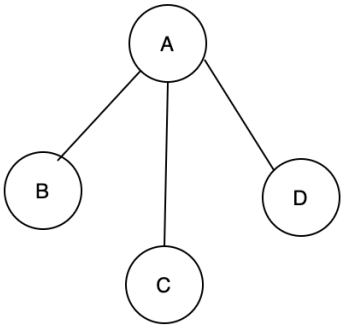
7.2 Tartışmalar

Verilen bir $G(V, E)$ çizgesinin kenar boyama problemini başka bir $H(U, F)$ çizgesinin köşe boyama problemine dönüştürebiliriz. G çizgesinin kenarları H çizgesinin köşeleri olacak şekilde, ve G çizgesindeki kenarların ortak köşe içermesi durumunu H çizgesinde karşılık gelen köşelerin arasına kenar koyarak ifade edersek, bu dönüşüme kenar dönüşümü diyelim,





Şekil 7.1 : c_0 ve c_l boyamaları.



Şekil 7.2 : Başka bir çizgenin kenar dönüşümü olmayan bir çizge.

$\chi_k(G) = \chi(H)$ olur. Hatta Vizing teoremini de kullanırsak, $\Delta G \leq \chi(H) \leq \Delta G + 1$ olur. Yani verilen bir H çizgesi için öyle bir G çizgesi bulalım ki bu G çizgesinin kenar dönüşümü H çizgesi olsun. Böylece $\chi(H)$'yi G çizgesinin derecesini kullanarak rahatça 1 hatayla bulabiliriz. Ancak tersini düşünürsek ne yazık ki her çizge başka bir çizgenin kenar dönüşümü

olarak yazılamaz hatta yeterince büyük çizgelerde neredeyse hiçbir çizge başka bir çizgenin kenar dönüşümü olarak yazılamaz [13]. Bir başka deyişle köşe sayısı büyüdükçe kenar dönüşümü olarak yazılabilecek çizge sayısı yeterince az kalacaktır. Örneğin Şekil 7.2'de bir başka bir çizgenin bir kenar dönüşümü olarak ifade edilemeyecek bir çizge gösterilmiştir. Eğer Şekil 7.2'de gösterilen çizge bir G çizgesinin kenar dönüşümü olsaydı, G çizgesi A, B, C ve D kenarlarından oluşurdu. A ve B, C, D kenarları arasında bir ortak köşe ve B, C, D kenarları arasında bir ortak köşe olmazdı. Bu durum imkansız olacağından böyle bir G çizgesi olamaz. Dolayısıyla Şekil 7.2'de gösterilen çizge bir başka çizgenin kenar dönüşümü değildir.

Bu tez çalışması, çizge teorisinin önemli bir problemi olan çizge boyama üzerine kapsamlı bir inceleme sunarak Türkçe literatüre hem teorik hem de pratik katkılar sağlamaktadır. Çizge teorisi üzerine temel kavramları tanımlayarak başladığımız bu tezde, karmaşıklık teorisi, P, NP ve NP-Tam kümelerinin tanımı, çizge boyama probleminin NP-Tam olduğunun kanıtı, çizge boyama problemi için kullanılan algoritmaların incelenmesi ve karşılaştırılması, farklı türlerde çizge boyama problemleri konuları ele alınmıştır.

Ayrıca, geliştirilen Python tabanlı sınav takvimi uygulaması, ile çizge boyama probleminin pratik bir yansıması olarak Hacettepe Üniversitesi Matematik Bölümü'nün ihtiyaçlarına somut bir çözüm sunulmuştur. Bu çalışmanın matematiğin çizge teorisi ve karmaşıklık teorisi alanlarında Türkçe literatüdeki eksikliği bir miktar gidereceğini düşünmekteyiz.

8. EKLER

8.1 EKLER-A

```
import networkx as nx
import matplotlib.pyplot as plt
import random

# ----------------------------
# Coloring Algorithm Helpers
# ----------------------------

def get_conflicts(G, coloring):
    return sum(1 for u, v in G.edges() if coloring.get(u) == coloring.get(v))

def initial_coloring(G, k):
    return {node: random.randint(0, k - 1) for node in G.nodes()}

# ----------------------------
# DSatur Algorithm
# ----------------------------

def dsatur_coloring(G):
    coloring = {}
    saturation = {v: 0 for v in G.nodes()}
    degrees = dict(G.degree())
    while len(coloring) < len(G.nodes()):
        uncolored = [v for v in G.nodes() if v not in coloring]
        max_sat = max(saturation[v] for v in uncolored)
        candidates = [v for v in uncolored if saturation[v] == max_sat]
        next_vertex = max(candidates, key=lambda v: degrees[v])
        used_colors = {coloring[n] for n in G.neighbors(next_vertex) if n in coloring}
        color = 0
        while color in used_colors:
            color += 1
        coloring[next_vertex] = color
        for n in G.neighbors(next_vertex):
            if n not in coloring:
                neighbor_colors = {coloring[m] for m in G.neighbors(n) if m in coloring}
                saturation[n] = len(neighbor_colors)
    return len(set(coloring.values()))

# ----------------------------
```

```
# Improved Tabucol Algorithm
# ---------------------------

def improved_tabucol(G, k, max_iter=1000, tabu_tenure=7):
    coloring = initial_coloring(G, k)
    tabu_list = {}
    best_conflicts = get_conflicts(G, coloring)

    for iteration in range(max_iter):
        if best_conflicts == 0:
            return k
        conflict_edges = [(u, v) for u, v in G.edges() if coloring[u] == coloring[v]]
        u, v = random.choice(conflict_edges)
        node = u if random.random() < 0.5 else v

        min_conflict = float('inf')
        best_color = coloring[node]
        for color in range(k):
            if color == coloring[node]:
                continue
            temp_coloring = coloring.copy()
            temp_coloring[node] = color
            conflicts = get_conflicts(G, temp_coloring)
            if (node, color) not in tabu_list or conflicts < best_conflicts:
                if conflicts < min_conflict:
                    min_conflict = conflicts
                    best_color = color

        coloring[node] = best_color
        tabu_list[(node, best_color)] = iteration + tabu_tenure

        current_conflicts = get_conflicts(G, coloring)
        if current_conflicts < best_conflicts:
            best_conflicts = current_conflicts

    return k + 1

# ---------------------------
# Improved HEA Algorithm
# ---------------------------

def improved_hea_coloring(G, k, population_size=20, max_iter=200):
    population = [initial_coloring(G, k) for _ in range(population_size)]

    def crossover(p1, p2):
```

```
        return {v: p1[v] if random.random() < 0.5 else p2[v] for v in G.nodes()}

    def mutate(c):
        v = random.choice(list(G.nodes()))
        c[v] = random.randint(0, k - 1)

    best = min(population, key=lambda c: get_conflicts(G, c))
    best_score = get_conflicts(G, best)

    for _ in range(max_iter):
        new_pop = []
        for _ in range(population_size):
            p1, p2 = random.sample(population, 2)
            child = crossover(p1, p2)
            if random.random() < 0.2:
                mutate(child)
            new_pop.append(child)
        population = sorted(new_pop, key=lambda c: get_conflicts(G, c))[:population_size]
        candidate = population[0]
        score = get_conflicts(G, candidate)
        if score < best_score:
            best_score = score
            best = candidate
        if best_score == 0:
            return k

    return k + 1

# ----------------------------
# Simulation and Plotting
# ----------------------------

p_values = [round(0.1 * i, 1) for i in range(1, 10)]
results_dsatur, results_tabucol, results_hea = [], [], []

for p in p_values:
    G = nx.erdos_renyi_graph(300, p)
    k_dsatur = dsatur_coloring(G)
    results_dsatur.append(k_dsatur)
    results_tabucol.append(improved_tabucol(G, k_dsatur))
    results_hea.append(improved_hea_coloring(G, k_dsatur))

# Plotting
plt.figure(figsize=(10, 6))
plt.plot(p_values, results_dsatur, marker='^', label='DSatur')
```

```
plt.plot(p_values, results_tabucol, marker='s', label=' Tabucol')
plt.plot(p_values, results_hea, marker='o', label=' HEA')
plt.xlabel('Kenar Olasili i (p)')
plt.ylabel('Kullanilan Renk Sayisi')
plt.title('Algoritmalarinin Rastgele  izgeler  zerinde  Karsilastirilmas(n=300)')
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```

KAYNAKÇA

[1] Thomas L. Saaty Paul C. Kainen. *The Four-Color Problem: Assaults and Conquest*. Dover Publications, **1986**. ISBN 0486676855.

[2] Bruce Reed Michael Molloy. *Graph Colouring and the Probabilistic Method*. Springer, **2002**. ISBN 9783642040153.

[3] Sheldon Ross. *A First Course in Probability*. Pearson, **2012**. ISBN 9780321794772.

[4] Michael R. Garey and David S. Johnson. *Computers And Intractability A Guide to Theory of NP-Completeness*. W. H. Freeman and Company, **1979**. ISBN 0-7167-1044-7.

[5] Peter Linz. *An Introduction to Formal Languages and Automata*. 9781284077247, **2016**. ISBN 9781284077247.

[6] Michael Sipser. *Introduction to the Theory of Computation*. Cengage Learning, **2012**. ISBN 113318779.

[7] Richard M.Karp. *Complexity of Computer Computations (Proceedings of a Symposium on the Theory of Computing)*. Plenum Press, **1972**. ISBN 0306307073.

[8] Jean H. Gallier. *Logic for Computer Science: Foundations of Automatic Theorem Proving*. Dover Publications, **2015**. ISBN 0486780606.

[9] Boaz Barak Sanjeev Arora. *Computational Complexity: A Modern Approach*. Cambridge University Press, **2009**. ISBN 0521424267.

[10] R.M.R. Lewis. *A Guide to Graph Colouring*. Springer, **2016**. ISBN 9783319257303.

[11] Marek Kubale ve diğerleri. *Graph Colorings*. American Mathematical Society, **2004**. ISBN 9780821834589.

[12] Reinhard Diestel. *Graph Theory*. Springer, **2000**. ISBN 0387989765.

[13] Gordon Godsil, Chris ve Royle. *Algebraic Graph Theory*. Springer, **2013**. ISBN 9781461301639.