T.C.

SELÇUK ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

GRAFLARDA DÜĞÜM BOYAMA PROBLEMİ İÇİN KURBAĞA SIÇRAMA ALGORİTMASI TABANLI BİR YAKLAŞIM

Murat ASLAN

YÜKSEK LİSANS TEZİ

Bilgisayar Mühendisliği Anabilim Dalını

Ocak-2017
KONYA

TEZ KABUL VE ONAYI

Murat ASLAN tarafından hazırlanan "Graflarda Düğüm Boyama Problemi İçin Kurbağa Sıçrama Algoritması Tabanlı Bir Yaklaşım" adlı tez çalışması ...03/01/2017... tarihinde aşağıdaki jüri tarafından oy birliği / ~~oy çokluğu~~ ile Selçuk Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı'nda YÜKSEK LİSANS TEZİ olarak kabul edilmiştir.

Jüri Üyeleri	**İmza**
Başkan Doç. Dr. Halife KODAZ	.....
Danışman Yrd. Doç. Dr. Nurdan BAYKAN	.....
Üye Yrd. Doç. Dr. Onur İNAN	.....

Yukarıdaki sonucu onaylarım.

Prof. Dr. Mustafa YILMAZ
FBE Müdürü

Bu tez çalışması ÖYP Kurum Koordinatörlüğü tarafından 2016-ÖYP-092 no'lu proje ile desteklenmiştir.

TEZ BİLDİRİMİ

Bu tezdeki bütün bilgilerin etik davranış ve akademik kurallar çerçevesinde elde edildiğini ve tez yazım kurallarına uygun olarak hazırlanan bu çalışmada bana ait olmayan her türlü ifade ve bilginin kaynağına eksiksiz atıf yapıldığını bildiririm.

DECLARATION PAGE

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Murat ASLAN

03.01.2017

ÖZET

YÜKSEK LİSANS

GRAFLARDA DÜĞÜM BOYAMA PROBLEMİ İÇİN KURBAĞA SIÇRAMA AGORİTMASI TABANLI BİR YAKLAŞIM

Murat ASLAN

Selçuk Üniversitesi Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

Danışman: Yrd. Doç. Dr. Nurdan BAYKAN

2017, 111 Sayfa

Jüri
Doç. Dr. Halife KODAZ
Yrd. Doç. Dr. Nurdan BAYKAN
Yrd. Doç. Dr. Onur İNAN

"Bir haritadaki komşu bölgeler farklı renkte boyandığında, en az kaç farklı renk kullanılarak bu harita boyanabilir" problemi olarak ortaya çıkan Graf Boyama Problemi (GBP), günümüzde harita boyama, zamanlama ve çizelgeleme problemi gibi birçok farklı gerçek hayat probleminin çözümünde kullanılmaktadır. Graflarda düğüm boyama ya da kavis boyama probleminde, bir grafı mümkün olan en az sayıda renk kullanarak boyamak, o graf için minimum rengi ifade etmekte ve kromatik sayı olarak adlandırılmaktadır. Graftaki düğüm ve kavis sayısı arttıkça problemin karmaşıklığı da artmaktadır. Bundan dolayı her algoritmasının çalışma süresi farklılık gösterebilir ve her algoritma kromatik sayıya ulaşamayabilir. Doğrudan GBP'nin çözümü için literatürde birçok açgözlü algoritma geliştirilmiştir. Bununla birlikte birçok metasezgisel algoritma GBP'ye uygulanmıştır. Kurbağa Sıçrama Algoritması (KSA), kurbağaların en az hareketle kaliteli besin kaynağına ulaşmak için yaptıkları hareketlerin modellenmesi ile geliştirilen bir algoritmadır. KSA'da bilgi paylaşımının hem mempleks olarak adlandırılan grup içinde hem de bir iterasyon sonunda tüm kurbağalarla yapılması gerek yerel arama uzayında gerekse de genel arama uzayında daha fazla çözüm arama imkânı vermektedir. Yapılan bu çalışma kapsamında KSA'da bazı değişiklikler yapılarak GBP'ye uygulanmıştır. GBP'nin kendine özgü yapısından dolayı KSA'nın saf halinin GBP'ye uygulanması iyi çözümler üretememiştir. Bundan dolayı yapılan değişikliklerle KSA'nın GBP için uygulanabilir bir yöntem olması sağlanmıştır. Bu kapsamda KSA'nın yerel arama mekanizması yeniden düzenlenmiştir. Alt mempleks içindeki en kötü kurbağanın yeni konumu Genetik Algoritmanın (GA) çaprazlama ve mutasyon operatörlerinden faydalanılarak hesaplanmaktadır. Çaprazlama operatörü alt mempleksteki en kötü kurbağanın, yerel ya da genel en iyi kurbağaya yakınsamasını sağlamaktadır. Mutasyon operatörü ise çaprazlama işleminde konumunu değiştiremeyen alt mempleksteki en kötü kurbağanın genlerinin değerlerini değiştirerek, en kötü kurbağa için yeni bir konum oluşturmaktadır. Ayrıca memetik evrim sonunda mempleksteki en iyi kurbağaya düğüm komşuluğu algoritmalarından (DKA) Chain yöntemi uygulanmaktadır. Chain yöntemi komşuları ile aynı renge sahip düğümlerin renklerini, düğümlerin komşuluklarını dikkate alarak değiştirmektedir. Böylece mempleks içindeki en iyi kurbağanın konumunu daha iyi bir noktaya götürmektedir. Algoritmanın çalışması esnasında, düzgün boyama kuralına uygun olarak sonuca ulaşıldığında, bu sonuç o ana kadarki en iyi çözüm olarak kaydedilmektedir. Ancak daha az renk kullanarak grafın boyanıp boyanamayacağını belirlemek amacıyla, renk sayısı bir azaltılarak, popülasyon yeniden üretilmektedir. Bunun için popülasyondaki bireylerden üst sınır (maksimum renk değeri) renk değerine sahip olan bireylerin renk değerleri bir azaltılmakta ve bireylerde güncelleme yapılmaktadır. Böylelikle daha az renk

kullanarak, graf boyama işlemi tekrarlanmaktadır. Popülasyonda yapılan bu değişiklik daha iyi çözümlerin bulunmasını hızlandırmıştır. Bu kapsamda ayrıca kromatik sayısı bilinmeyen graflar için de literatürde bulunan en iyi değerlerden daha iyi sonuçlar bulunması hedeflenmiştir. Önerilen algoritma DIMACS tarafından sunulan Benchmark grafları üzerine uygulanarak sonuçlar değerlendirilmiştir. Deneysel sonuçlara göre önerilen yöntem (Modifiye KSA-MKSA) saf KSA'ya göre daha başarılı sonuçlar vermiştir. Ayrıca MKSA, literatürde bulunan algoritmalarla da karşılaştırılmış ve sonuçların başarılı olduğu gösterilmiştir.

Anahtar Kelimeler: Genetik algoritma**,** Graf boyama problemi, Kromatik sayı, Kurbağa sıçrama algoritması.

ABSTRACT

MS THESIS

AN APPROACH BASED ON SHUFFLED FROG LEAPING ALGORITHM FOR VERTEX COLORING PROBLEM IN GRAPHS

Murat ASLAN

THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCE OF SELÇUK UNIVERSITY
THE DEGREE OF MASTER OF SCIENCE IN COMPUTER ENGINEERING

Advisor: Asst. Prof. Dr. Nurdan BAYKAN

2017, 111 Pages

Jury
Assoc. Prof. Dr. Halife KODAZ
Asst. Prof. Dr. Nurdan BAYKAN
Asst. Prof. Dr. Onur İNAN

The Graph Coloring Problem (GCP), which emerged as a problem such that "When adjacent regions in a map are colored with different colors, at least how many different colors can be used for this map? ", is nowadays being used to solve many different real-world problems such as map coloring, timetabling and scheduling problems. In the problem of vertex or edge coloring in graphs, coloring the graph using the least possible number of colors expresses minimum color for that graph and is called the chromatic number. As the number of vertices or edges in a graph increases, the complexity of the problem also increases. Because of this, each algorithm can not find the chromatic number of the problems and may also be different in their executing times. Various greedy algorithms have been developed in the literature for the directly solution of the GCP. In addition, many metaheuristic methods are applied to GCP. The Shuffled Frog Leaping Algorithm (SFLA) is an algorithm that is developed by the modeling of the movements of frogs in order to achieve high quality food source with minimal movement. It enables to find more solutions in both local and global search space because of that the sharing of information in the SFLA is done with all the frogs at the end of iteration and also the frogs in the group called memeplex. In this study, some changes were made to SFLA and applied to GCP. Due to the peculiar nature of the GCP, the implementation of the original SFLA into the GCP can not produce efficient solutions. Therefore, the implementations have made it possible for the SFLA to be available method for the GCP. For this reason, the SFLA's local search mechanism has been rearranged. The new position of the worst frog in the sub memeplex was calculated by using the crossover and mutation operators of the Genetic Algorithm (GA). The crossover operator ensures that the worst frog in the sub memeplex is close to the local or global best frog. The mutation operator creates a new position for the worst frog in the sub memeplex by changing the values of their genes. In addition, at the end of memetic evolution, the Chain method from vertex neighborhood algorithms (VNS) is applied to the best frog in the memeplex. The Chain method changes the colors of the nodes which have the same color with their neighbors, taking into consideration the neighborhoods of the nodes. This leads to a better position of the best frog in the memeplex. When the result of the algorithm is reached in accordance with the proper staining rule; this result is recorded as the best solution. However, the population is regenerated by reducing the number of colors by one in order to determine whether the graph can be colored using less color. For this reason, the color values of the individuals with the upper limit of color value (maximum color value) of the individuals in the population are reduced and updated. Thus, the graph coloring process is repeated using less color. This change in the population accelarates the process of finding better solutions. In this case, it

is aimed to find better solutions than the current best solutions in the literature for graphs whose chromatic numbers is unknown. The proposed algorithm was applied on the Benchmark graphs presented by DIMACS and the results were evaluated. According to the experimental results, the proposed method (Modified SFLA-MSFLA) is more successful than the original SFLA. In addition, MSFLA is also compared with the algorithms in the literature and the results are shown to be successful.

Keywords: Chromatic number, Genetic algorithm, Graph coloring problem, Shuffled frog leaping algorithm.

ÖNSÖZ

Tez Çalışmam boyunca gösterdiği katkı ve yardımlarından dolayı danışman hocam Sayın Yrd. Doç. Dr. Nurdan (Akhan) BAYKAN'a ve benden desteklerini esirgemeyen Selçuk Üniversitesi Mühendislik Fakültesi Bilgisayar Mühendisliği Bölümü'nün tüm öğretim elemanlarına teşekkür etmeyi bir borç bilirim.

Çalışmalarım boyunca benden maddi ve manevi desteklerini asla esirgemeyen üzerimde büyük hakları olan aileme en içten teşekkürlerimi sunarım.

Murat ASLAN
KONYA-2017

İÇİNDEKİLER

SİMGELER VE KISALTMALAR

Simgeler

R	Renk kümesi
Deg	Düğüm dereceleri
E	Kavis
G	Graf
V	Düğüm
$W(e_i)$	Ağırlıklı graf
G'	Alt graf
$\delta\,(\mathrm{G})$	Minimum düğüm derecesi
$\Delta(\mathrm{G})$	Maksimum düğüm derecesi
$\chi(\mathrm{G})$	Kromatik sayı
σ	Düğüm derecesi

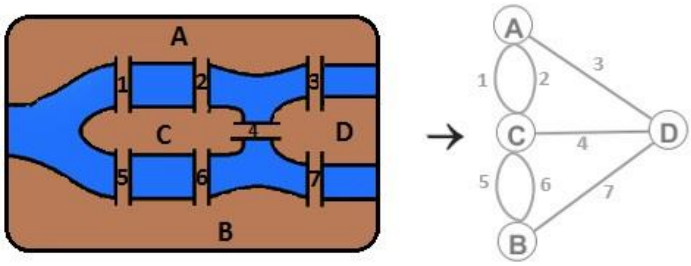
Kısaltmalar

GKA	Guguk Kuşu Algoritması
DSATUR	Degree of Saturation
FF	First Fit
GA	Genetik Algoritma
MGKA	Modifye edilmiş Guguk Kuşu Algoritması
KKA	Karınca Kolonisi Algoritması
RLF	Recursive Largest First
TB	Tavlama Benzetimi
TA	Tabu Arama
WP	Welsh ve Powell
KSA	Kurbağa Sıçrama Algoritması
MKSA	Modifye Edilmiş Kurbağa Sıçrama Algoritması
YAK	Yapay Arı Kolonisi
YA	Yerçekimi Arama
LDO	Largest Degree Ordering
IDO	Incidence Degree Ordering
GBP	Graf Boyama Problemi

1. GİRİŞ

Graf teorisi bir problemin, düğüm (tepe) ve kavisler (ayrıt) ile gösterilmesidir. Bir grafta düğümler kümesi ile kavisler kümesi ayrık iki kümedir ve $G(V, E)$ olarak gösterilir (Biggs ve ark., 1976; West, 2001). Şekil 1.1'de bir graf örneği verilmiştir. $V = \{A, B, C, D\}$ düğümler kümesini, $E = \{1,2,3,4,5,6,7\}$ ise bu düğümleri birbirine bağlayan ayrıtlar kümesidir.

Graf teorisi kavramı 1736 tarihinde ünlü Matematikçi Leonhard Euler tarafından önerilmiştir. Leonhard Euler, graf teorisini ünlü Königsberg'in yedi köprüsü problemini çözmek için kullanmıştır. Königsberg'in yedi köprüsü problemi, Königsberg şehrinin köprülerinden esinlenerek ortaya çıkan bir matematik problemidir. Şekil 1.1'de Königsberg şehrindeki kara bağlantılarını gösteren köprüler verilmiştir. Bu problem için kara parçaları düğümler, köprüler ise kavisler ile gösterilmiştir. Königsberg probleminde düğüm sayısı dört, kavis sayısı ise yedidir. Leonhard Euler, Königsberg şehri için "Bütün köprülerden sadece bir kez geçmek şartı ile tüm kara parçaları gezilebilir mi ?" sorusunun cevabını bulmaya çalışmıştır (Biggs ve ark., 1976).



Şekil 1.1. Königsberg köprüsü problemi

Graf ve graf algoritmaları, bilgisayar tabanlı uygulama ve problemlerin çözümünde sıklıkla kullanılmaktadır. Mühendislik uygulamalarından tıp alanına kadar birçok bilim dalında karşılaşılan problemlerin çözümü için graf teorisi kullanılabilmektedir (Gross ve Yellen, 1998).

Dört Renk Problemi olarak bilinen graf renklendirme problemi ünlü İngiliz matematikçi Francis Gutrie tarafından ortaya atılmıştır (Fritsch ve Fritsch, 1998). Bu teoremin doğrudan uygulama alanlarından birisi harita boyanmasıdır. Francis Gutrie, Dört Renk Problemini "Bir haritadaki komşu bölgeler farklı renkte boyandığında, en az kaç farklı renk kullanılarak ilgili harita boyanabilir?" olarak tanımlamıştır. Bu

problemde sadece bir noktada birleşen bölgelerin komşu olmadığı varsayılmıştır (Fritsch ve Fritsch, 1998; Betin ve Erhan, 2013). Sonuç olarak haritaların en az dört renk kullanarak boyanabileceği fark edilmiştir. Dört Renk Problemi, Kenneth Appel ve Wolfgang Haken (1976) tarafından çözüme kavuşturulmuştur (Berkman ve ark., 1991).

Graf boyama algoritmaları (GBA), düğüm (tepe) ve kavis (ayrıt) boyama olarak 2 farklı şekilde olabilmektedir (Gross ve Yellen, 1998). Her iki boyama algoritmasında da amaç tüm grafı hatasız bir şekilde boyamaktır. Düğüm boyamada amaç, grafta bulunan düğümlerden komşu olan düğümlerin farklı renkler ile boyanmasıdır. Kavis boyama da ise amaç, komşu kavislerin aynı renk olmayacak şekilde farklı renkler kullanılarak boyanmasıdır. Graflarda komşu düğümler, aralarında en az bir adet bağlantı yani kavis olan düğümlerdir. Şekil 1.1'de A düğümü ile D düğümü arasında en az bir bağlantı olduğu için komşu düğümdürler. Ancak A ile B düğümü arasında herhangi bir kavis olmadığı için komşu düğüm değildirler.

$G = (V, E)$ olarak verilen yönsüz bir graf için; V düğüm kümesini E ise kavis kümesini temsil eder. Graflarda, düğüm boyama probleminde amaç, komşu düğümler aynı renk olmayacak şekilde, tüm düğümleri boyamak için kullanılan renk sayısını en az olacak şekilde bulmaktır. $V = \{v_1, v_2, \ldots\ldots, v_n\}$ düğüm kümesi ve $R = \{1,2, \ldots., k\}$ ise düğümleri boyamak için kullanılan renk kümesi olsun. Kavis kümesi ise $E = \{e_{ij} | v_i \; ve \; v_j \; düğümleri \; arasındaki \; kavis\}$ olarak tanımlanmaktadır. Birbiri ile bağlantısı olan iki düğümün aynı rengi almaması koşulu ile boyanan grafa "k renkli graf" denir (Ge ve ark., 2010). Bir grafı mümkün olan en az sayıda renk kullanarak boyamak, o graf için minimum rengi ifade etmekte ve "kromatik sayı" olarak adlandırılmaktadır. Kromatik sayı χ(G) ile gösterilmektedir (Welsh ve Powell, 1967; Díaz ve Zabala, 1999).

Mühendislik uygulamalarından birçok gerçek dünya problemine kadar karşılaşılan problemlerin çözümü için graf boyama algoritmaları kullanılabilmektedir (Gross ve Yellen, 1998). Graf boyama problemi (GBP) (Graph Coloring Problem-GCP), baskı devre kartlarının test problemi (printed circuit board testing) (Garey ve ark., 1976), frekans tahsisi problemi (Hale, 1980), yazmaç özgüleme (Chaitin ve ark., 1981; Chow ve Hennessy, 1990), harita boyama (Gwee ve ark., 1993), kanal yönlendirme (Channel routing) (Sen Sarma ve ark., 1994), zamanlama ve çizelgeleme problemi (Dowsland ve Thompson, 2005; Chmait ve Challita, 2013), sudoku problemi (Ono ve ark., 2009) gibi farklı problemlerin çözümünde kullanılmaktadır.

Graf boyama probleminin çözümünde sezgisel ve metasezgisel yöntemler kullanılabilmektedir (Ge ve ark., 2010). GBP, graf teorisinde çözülmesi zor yani NP-zor

(NP-hard) problemlerden birisidir. Böyle nitelendirilmesinin iki önemli sebebi vardır. Birinci sebep uygulama alanlarının zorluğu ve çeşitliliği, ikincisi ise problemin farkı seviyelerde daha zor hesaplama gerektirmesidir (Garey ve Johnson, 1979). Çünkü graftaki düğüm ve kavis sayısı arttıkça problemin karmaşıklığı da artmakta ve uygulanan algoritmaların performans başarıları da değişebilmektedir. Ayrıca bunlar ile birlikte herhangi bir graf boyama problemi için farklı çözüm yöntemlerinin olması NP-zor olan bu problemin NP-tam (NP-completeness) olarak nitelendirilmesinde etkilidir (Mahmoudi ve Lotfi, 2015).

GBP'nin çözümü için literatürde önerilen algoritmalardan sezgisel yöntemlerde, açgözlü (greedy) yapılar kullanılarak graf boyama problemine çözüm bulunmaya çalışılmıştır (Mahmoudi ve Lotfi, 2015). Açgözlü algoritmalarda, algoritma sadece belirlenen bir çözüm yönteminden yola çıkarak probleme çözüm bulmaya çalışır. Yani probleme genel bir bakış ile yaklaşmadığından, çözüm uzayındaki farklı olasılıklar gözetilmez. Bundan dolayı düğüm sayısının ve kavis karmaşıklığının yoğun olmadığı graflar için oldukça etkili bir çözüm sunmuşlardır. Ancak problemin karmaşıklığı arttıkça çözüm uzayında farklı çözümlerin olabileceği düşünüldüğünde metasezgisel yöntemlerin iyi çözümler bulacağı düşünülebilir. First Fit (FF), Incidence Degree Ordering (IDO) ve Largest Degree Ordering (LDO) (Al-Omari ve Sabri, 2006), Welsh ve Powell (WP) (Welsh ve Powell, 1967) , Degree of Saturation (DSATUR) (Brélaz, 1979) ve Recursive Largest First (RLF) (Leighton, 1979) algoritmaları birer açgözlü sezgisel algoritmadır. Bu algoritmalar birçok sezgisel ve metasezgisel algoritmanın yapılarında kullanıldıkları için önemli temel algoritmalardır.

Optimizasyon temelli problemlerin çözümünde metasezgisel yöntemler oldukça etkili yöntemlerdir. Metasezgisel yöntemler birçok farklı probleme, problemin yapısına göre uyarlanabilirler. Metasezgisel algoritmalar arı (Karaboğa, 2011), kurt (Liu ve ark., 2011), kurbağa (Eusuff ve Lansey, 2003), karınca (Dorigo ve ark., 1991), algler (Uymaz, 2015), kuş ve balık sürüsü hareketleri (Eberhart ve Kennedy, 1995) gibi birçok canlının doğal hareketlerinden ya da yer çekimi (Rashedi ve ark., 2009) gibi doğada gerçekleşen olaylardan esinlenerek oluşturulan algoritmalardır. GBP'nin çözümü için farklı metasezgisel algoritmalar kullanılmaktadır. Tabu Arama (Tabu Search-TS) algoritması (Hertz ve de Werra, 1987), Tavlama Benzetimi Algoritması (Simulated Annealing-SA) (Chams ve ark., 1987), Genetik Algoritma (GA) (Gwee ve ark., 1993), Karınca Kolonisi Algoritması (KKA) (Ant Colony Optimization-ACO) (Ahn ve ark., 2003), Parçacık Sürü Optimizasyonu algoritması (Particle Swarm Optimization-PSO)

(Qin ve ark., 2011), Guguk Kuşu Algoritması (GKA) (Cuckoo Optimization Algorithm-COA) (Mahmoudi ve Lotfi, 2015), Yerçekimi Algoritması (YA) (Gravitational Search-GS) (Ruiz ve Romay, 2011), Kültürel (Cultural) algoritması (Abbasian ve ark., 2011) gibi algoritmalar graf boyama problemi için kullanılan bazı metasezgisel algoritmalardır.

Bu çalışmada, graf boyama probleminin çözümü için kurbağa sıçrama algoritmasına dayalı bir algoritma önerilmiştir. Kurbağa sıçrama algoritması (KSA) popülasyon tabanlı bir optimizasyon algoritmasıdır. KSA kurbağaların doğal yaşam alanında yiyecek arama davranışları üzerine geliştirilmiş memetik bir yaklaşımdır. KSA'da amaç, en iyi besin kaynağına yakın olan kurbağanın konum bilgilerini kullanarak diğer kurbağaların da bu besin kaynağına doğru hareket etmesini sağlamaktır (Eusuff ve Lansey, 2003). Algoritma memetik evrimleri kullanarak yerel arama uzayında bir bireyden diğer bir bireye bilgi geçişini sağlar. KSA'nın her aşamasında amaç en kötü kurbağanın arama uzayında daha iyi bir konuma hareket ettirilmesini sağlamaktır. En kötü kurbağanın konum güncellemesi ya yerel en iyi kurbağaya ya da genel en iyi kurbağaya göre yapılmaktadır. KSA'nın yerel arama yapısı PSO algoritmasına benzerlik gösterir. Algoritmada öncelikle en kötü kurbağa için yereldeki kurbağaya göre konum güncellemesine gidilir. Çünkü bu yerel arama yapısı sayesinde bir kurbağanın kendi çevresindeki en iyi kurbağanın konumuna doğru hareket etmesi sağlanmış olur. Ancak yerel kurbağanın bulunduğu konum kötü birey için daha iyi bir konum değilse, bu durumda genel en iyi kurbağanın konum bilgilerine göre alt mempleks grubu içindeki en kötü kurbağanın yeni konumu belirlenir. Eğer hala en kötü kurbağa daha iyi bir noktaya ulaşamamışsa, bu kurbağa arama uzayında rastgele olarak yeni bir konuma yerleştirilir. Böylece arama uzayı genişletilmiş olur. Saf KSA'yı GBP'ye doğrudan uygulamak GBP'nin yapısından dolayı bizi iyi çözümlere götürememiştir. Bundan dolayı bu çalışmada Saf KSA'da bazı değişikliklere gidilmiştir. En kötü kurbağanın konum güncellemesi aşamasında genetik algoritmanın çaprazlama ve mutasyon operatörlerinden faydalanılmıştır. Üzerinde gezinilen mempleks için, memetik evrim aşaması bittikten sonra mempleks içindeki en iyi kurbağaya DKA algoritmasının komşuluk arama mekanizmalarından olan Chain metodu uygulanmıştır. Böylece GBP için çözüm üretecek bir algoritma önerilmesi amaçlanmıştır.

Çalışmanın ikinci bölümünde hem graf boyama algoritmaları hem de kurbağa sıçrama algoritması ile ilgili literatür araştırması yapılmıştır. GBP için yapılan çalışmalar ve bu çalışmaların uygulamaları hakkında bilgiler verilmiştir. GBP'nin

önemine değinilmiştir. Ayrıca KSA ile ilgili yapılan çalışmalar hakkında bilgiler verilmiştir.

Çalışmanın üçüncü bölümünde graf ve graf boyama problemi ile ilgili detaylı bilgiler verilmiştir. GBP için literatürde verilen bazı algoritmalarının çalışma adımları ile saf KSA adımları verilmiş ve detaylandırılmıştır. Daha sonra GBP için neden KSA'da değişikliğe gidilmesine ihtiyaç duyulduğu ile ilgili açıklamalar yapılmış ve önerilen algoritma ile ilgili bilgiler detaylı olarak verilmiştir.

Çalışmanın dördüncü bölümünde, önerilen algoritmanın DIMACS tarafından sunulan bazı Benchmark grafları üzerinde test edilmesi sonucunda elde edilen sonuçlar verilmiştir. DIMACS tarafından sunulan Benchmark grafları graf boyama problemini algoritmalar üzerinde test etmek için kullanılan graflardır Elde edilen sonuçlar, literatürdeki benzer algoritmalar ile kıyaslanarak yorumlanmıştır.

Çalışmanın son bölümünde çalışmadan elde edilen çıkarımlar ve öneriler sunulmuştur.

2. KAYNAK ARAŞTIRMASI

Dört Renk Problemi olarak bilinen graf boyama problemi ünlü matematikçi Francis Gutrie tarafından ortaya atılmıştır (Fritsch ve Fritsch, 1998). F. Gutrie, İngiltere haritasını dört renk ile boyamaya çalışmıştır. Kendisi teoremini ispatlamadan vefat etmiştir. Gutrie'nin kardeşi problemi Londra Üniversitesindeki matematik hocası Augustus De Morgan'a iletmiştir. 1852'de De Morgan problem eline ulaştıktan sonra problemi İrlandalı bilim adamı William Rowan Hamilton'a bir mektup göndererek iletmiştir. Bu mektup şu anda Dublin'deki Trinity Üniversitesinde tutulmaktadır (Fritsch ve Fritsch, 1998).

Graflarda düğüm boyama probleminde, bir grafı mümkün olan en az sayıda renk kullanarak boyamak, o graf için minimum rengi ifade etmektedir. Buna "kromatik sayı" denir ve $\boldsymbol{\chi(G)}$ olarak gösterilir. Burada G, grafı temsil etmektedir (Welsh ve Powell, 1967; Díaz ve Zabala, 1999).

GBP, graf teorisinde NP-zor problemlerden birisidir. Böyle nitelendirilmesinin iki önemli sebebi vardır. Birinci sebep uygulama alanlarının zorluğu ve çeşitliliği, ikincisi ise problemin farkı seviyelerde daha zor hesaplama gerektirmesidir (Garey ve Johnson, 1979). Ayrıca bunlar ile birlikte herhangi bir graf boyama problemi için farklı çözüm yöntemlerinin olması NP-zor olan bu problemin NP-tam olarak nitelendirilmesinde etkilidir (Mahmoudi ve Lotfi, 2015). X olarak tanımlanan bir problem, Y olarak tanımlanan problemi iyi çözen bir algoritmadan faydalanarak elde edilen başka iyi bir algoritma ile çözülebiliyorsa, bu durumda X'in Y'ye indirgenebileceği söylenir. Yani X problemi Y probleminin özel bir durumu olduğu için Y problemini çözen algoritma kullanılarak X problemi çözülebilir. Eğer NP sınıfındaki bir problem Y probleminin özel durumlarına indirgenebiliyorsa, Y problemi NP-zor olarak tanımlanır. Bununla birlikte Y'nin kendisi de NP sınıfında bulunuyorsa bu durumda Y'ye NP-tam problem denir (Nuriyev ve Sadıgova, 2002). Örneğin n düğümlü bir graftaki düğümler boyanırken, birinci düğüm için diğer tüm düğümler ile olan ilişkisi kontrol edilmelidir. Bunun için $(n-1)$ tane durum söz konusudur. İkinci düğüm için $(n-2)$ tane durum söz konusudur. Tüm ihtimallerin gözetilmesi ile grafın hatasız bir şekilde boyanabilmesi, düğüm sayısının artış göstermesi ile artacak ve çözülmesi gittikçe zorlaşacaktır. GBP probleminin başka algoritmaların rehberliğinde metasezgisel algoritmalar ile çözülmesi bu problemi NP-tam yapmaktadır.

Welsh ve Powell (1967) graf boyama problemi için açgözlü bir algoritma önermişlerdir. Önerdikleri algoritmada, her çalışma adımında mümkün olan en fazla sayıda düğüm, aynı renk ile boyanmaya çalışılmaktadır. Bu algoritmaya göre boyanmamış her düğüm için diğer düğümler ile olan bağlantı sayısı bir dizide tutulmaktadır. Bu dizideki düğümler bağlantı sayılarına göre büyükten küçüğe doğru sıralanmakta, daha sonra bu diziden sırayla boyanmamış bir düğüm seçilmekte ve boyanmaktadır. Boyanan düğümün komşu düğümleri aynı renk ile boyanamayacağı için bu düğümler diziden silinir. Kalan düğümler için boyama işlemi tekrarlanır. Dizide boyanabilecek düğüm kalmadığında, boyama işlemi için yeni bir renk oluşturulur ve boyanmamış düğümler için algoritma tekrar çalışmaktadır (Welsh ve Powell, 1967).

Dört Renk Problemi, Francis Gutrie tarafından ortaya atılmış, ancak 1976 yılında Kenneth Appel ve Wolfgang Haken tarafından çözüme kavuşturulmuştur (Berkman ve ark., 1991). K.Appel ve W. Haken, Dört Renk Problemini bilgisayar tabanlı sistem kullanarak çözmüşlerdir. Bütün olasılıkların kontrol edilmesi için bilgisayar yaklaşık olarak 1200 saat (50 gün) çalışmıştır. Bu problem dünyanın ilk bilgisayar destekli matematik ispatı olarak bilinmektedir (Betin ve Erhan, 2013).

Garey ve Johnson (1976) baskı devre kartları için bir yöntem önermişlerdir. Çalışmalarında baskı devre kartlarının basılması aşamasında, baskı devre kartları için mümkün olan en kısa devre yapısını elde etmek için problemi graf boyama problemi olarak tanımlamışlardır (Garey ve ark., 1976).

F.T. Leighton (1979) graf problemini çözmek için yeni bir algoritma önermiştir. Bu algoritma RLF olarak bilinen Recursive Largest First algoritmasıdır. RLF algoritması da sezgisel bir algoritmadır. Bu algoritma da graf boyama problemi için oldukça etkili bir algoritmadır (Leighton, 1979). Ayrıca birçok metasezgisel algoritmalarda ilk renklendirme işlemi için kullanılan algoritmalardan biridir.

Bir başka çalışmada ise Brélaz (1979) düğüm boyama işlemi için yeni bir algoritma önermiştir. Bu algoritma DSATUR algoritması olarak bilinmektedir. DSATUR algoritmasında renklendirme işlemi için seçilecek ilk düğüm her zaman için en çok düğüm ile bağlantısı olan düğüm yani düğüm derecesi en büyük olan düğümdür. Daha sonra, algoritma içerisinde her adımda boyanacak düğüm için bağlantılı olduğu düğümlerden farklı renge sahip komşu sayısı yani derecesi diğer düğümlerden büyük olan düğüm seçilerek boyama işlemi yapılmaktadır. Tüm düğümler boyanıncaya kadar bu işlem tekrarlanmaktadır (Brélaz, 1979). Ono ve ark. Sudoku problemini düğüm boyama problemi olarak tasarlamış ve yaptıkları çalışma D. Brélaz'ın önerdiği

DSATUR algoritmasını kullanarak sudoku problemi için çözüm sunmuşlardır (Ono ve ark., 2009).

Düğüm boyama için kullanılan bir diğer algoritma First Fit (FF) algoritmasıdır. Düğüm kümesi V = $\{v_1, v_2, \ldots\ldots, v_n\}$ ve renk kümesi R = {1,2, … , k} olarak tanımlanan bir G grafı için FF algoritması, düğümler kümesindeki birinci düğümden başlayarak, sonuncu düğüme varıncaya kadar her düğümü sırasıyla boyamaktadır. Boyama işlemi sırasında boyanacak düğüme renk verme işlemi sırasında, renk kümesinde bulunan ilk renk verilmeye çalışılır. Eğer bu renk, boyanacak düğümün komşu düğümlerinden herhangi birine verilen renk ise bu durumda renk kümesinde bulunan bir sonraki renk bu düğüme verilmeye çalışılır. Bu işlem uygun boyama kuralına bağlı olarak, boyanacak tüm düğümlere bir renk verilinceye kadar devam eder. Böylece tüm düğümler boyandığından renk kümesinden en az sayıda renk kullanılmış olur. Algoritma basit ve oldukça hızlı çalışmaktadır (Gebremedhin, 1999; Ge ve ark., 2010).

Yapılan bir başka çalışmada, radyo frekans tahsisi probleminin çözümü için graf boyama kullanılmıştır. Çalışmada, radyo frekans tahsisi probleminin graf boyama algoritmaları ile çözülmesinin sebebi GBP ile frekans tahsisi arasında sıkı bir bağlantı olmasıdır. Graf boyama, radyo frekans tahsisi problemi için iyi sonuçlar vermiştir (Hale, 1980). Glass ve Bennett (2003) yaptıkları çalışmada frekans tahsisi probleminin çözümü için yeni bir yaklaşım önermişlerdir. Bu çalışmada GA'yı Tabu arama algoritması ile birlikte kullanmışlardır. Algoritmayı güçlü kılmak için GA'nın yeni çözüm bulamadığı noktada Tabu Arama algoritması devreye girerek iyi çözümlerin bulunmasını sağlamıştır (Glass ve Prügel-Bennett, 2003).

Chaitin ve ark. (1981) yaptıkları çalışmada bilgisayar yazmaç özgüleme (computer register allocation) problemi için graf boyama kullanmışlardır. Herhangi bir program çalışırken, bu çalıştırılan programda kullanılan değişkenler, ifadeler register'larda tutulmaktadır. Yazmaç özgülemede amaç, hafızayı en iyi şekilde kullanmaktır. Bunu yapmak için de aynı bellek alanına ihtiyaç duymayan problemler aynı register'i kullanarak çözülebilmektedir. Bu problem çözülürken programdaki her bir register düğüm olarak tasarlanmıştır. Bu yaklaşım ilk kez Chaitin ve ark. (1981) yapmış oldukları çalışmada kullanılmıştır (Chaitin ve ark., 1981). Daha sonra Chow ve Hennessy (1990) yaptıkları çalışmada yazmaç özgüleme problemi için önceliğe dayalı bir graf boyama algoritması önermişlerdir (Chow ve Hennessy, 1990).

Zaman ve çizelgeleme problemlerinin çözümünde ve ders programı probleminde de düğüm boyama sıklıkla kullanılmaktadır. Bunun birinci nedeni ders programı

problemindeki özelliklerin çeşitlilik göstermesi ve çok fazla kıstasın olmasıdır. İkinci neden ise eğitim sisteminin sürekli değişmesidir. Bu sebeplerden dolayı zamanlama problemleri için uygulanan çözümün güncellenmesi ve isteklerimize cevap verecek bir uygulamanın sunulması gerekmektedir (de Werra, 1985). Dowsland ve Thompson (2005) yaptıkları çalışmada sınav hazırlama programı problemini graf boyama problemi olarak tanımlamışlardır. Daha sonra bu problemi Karınca Kolonisi Algoritması (KKA) (Ant Colony Optimization-ACO) ile çözmüşlerdir (Dowsland ve Thompson, 2005). GBP için KKA kullanılarak graf renklendirme işlemi yapılırken, metasezgisel algoritmaya verilmiş olan graf, ön renklendirmeden geçirilerek verilmiştir (Ahn ve ark., 2003; Salari ve Eshghi, 2005; Plumettaz ve ark., 2010). Kılıçaslan (2007), Welsh ve Powell'ın sezgisel algoritmasını kullanarak Beykent Üniversitesi için sınav programı hazırlama uygulaması geliştirmiştir (Kılıçaslan, 2007). Son yıllarda iletişim hattındaki bağlantıyı güçlendirmek için gökkuşağı bağlantı sayısı veya tek renkli bağlantılı boyama çalışmaları yapılmıştır. Çolakoğlu (2012) yaptığı çalışmada graflarda gökkuşağı bağlantı sayısı ile tek renkli bağlantılı boyama sayısını hesaplayan bir algoritma önermiştir (Çolakoğlu, 2012).

Tabu Arama (TA) (Tabu Search-TS) algoritması bir yerel arama algoritmasıdır. Tabu arama algoritması rastgele bir çözüm ile başlar, daha sonra ise her bir adımda önceki adımlarda elde edilen sonuçlara göre yeni çözümler üretir (Porumbel ve ark., 2010). Bundan dolayı arama yaptığı alandaki en iyi çözümü bulur. Tabu aramada tabu listeleri tutulur. Tabu listesi uğranılmaması gereken duruma ceza vererek, çözüm uzayında yerel çözüme takılmanın önüne geçer. Tabu listesi büyüdükçe çözüme ulaşmak için harcanan süre de artar (Ruiz, 2012). Hertz ve Werra (1987) yaptıkları çalışmada TABUCOL algoritmasında minimum değer üreten bir fonksiyon kullanarak renklendirme işlemi yapmışlardır. Algoritma kullandığı ceza fonksiyonu sayesinde yanlış boyanan düğümleri tespit ederek çakışmaları gidermektedir. TABUCOL, GBP için iyi bir renklendirme sunmuştur (Hertz ve de Werra, 1987). TABUCOL algoritması birçok algoritmanın yerel arama mekanizması olarak kullanılmaktadır (Yılmaz, 2011).

Graf boyama problemi için kullanılmış olan ilk yöntemlerden biri yerel arama yöntemi olan Tavlama Benzetimi (TB) (SA: Simulated Annealing) algoritmasıdır. Chams ve ark. (1987) yaptıkları çalışmada TB algoritmasını uygulamadan önce grafı sezgisel bir algoritma ile renklendirmişlerdir. Daha sonra TB algoritmasının çalıştırıldığı her adımda sadece bir düğümün rengi değiştirilmiştir. Oluşan yeni grafın komşulukları incelendikten sonra graf düzgün bir renklendirmeye sahipse, bir önceki

renk kümesine göre daha iyi bir çözüm kümesi elde edilmiştir. Eğer graf düzgün boyanmamışsa o zaman tavlama algoritmasındaki sıcaklık kısıtı devreye girmekte ve kötü çözümün seçilme olasılığı azaltılmaktadır (Chams ve ark., 1987). Yılmaz (2011) yaptığı çalışmada Geri İzleme Algoritmasına dayalı TB Algoritmasını (GİTB) (Simulated Annealing with Backtrackıng-SABT) kullanarak graf boyama problemi için yeni bir yaklaşım önermiştir. GİTB algoritmasının çözülmesi zor graflar için etkili bir çözüm sunduğunu belirtmiştir (Yılmaz, 2011). Bir başka çalışmada ise Chmait ve Challita (2013) TB algoritmasını KKA algoritması ile birlikte kullanmışlardır. Yapılan çalışmada zaman ve çizelgeleme problemi için etkili bir çözüm sunulmuştur (Chmait ve Challita, 2013).

Bir diğer yerel arama algoritması ise Yinelemeli Yerel Arama (YYA) (Iterated Local Search-ILS) algoritmasıdır. Bu algoritma, DSATUR'dan gelen renk sayısını referans kullanarak bir çözüm üretmektedir. Daha sonra yerel arama yapısını kullanarak başlangıçta oluşturulan çözümdeki yanlış boyanan düğümleri düzeltmeye çalışmaktadır. Eğer algoritma bir çözüm bulursa kullanılan renk sayısı bir azaltılarak yerel arama algoritması tekrar çalıştırılır (Chiarandini ve Stützle, 2002). Bir başka YYA tabanlı çalışmada Caramia ve Dell'Olmo (2008) YYA algoritmasını iki farklı işlevsel özellikteki algoritmayla birlikte birleştirerek yeni bir sezgisel yaklaşım geliştirmişlerdir. Bu işlevlerden biri, iki açgözlü algoritmanın altyapısını kullanarak oluşturulmuştur. Diğeri ise komşuluk araması yaparak yanlış renklendirilmiş düğümlerin düzgün renklendirilmiş hale getirilmesi için kullanılmıştır. Çalışmanın Benckmark grafları üzerinde test edildiği ve başarılı sonuçlara ulaşıldığı söylenmiştir (Caramia ve Dell'Olmo, 2008).

FOO-PARTIALCOL algoritması GBP için oluşturulmuş tabu arama algoritması tabanlı bir algoritmadır. Ancak tabu algoritmasından farklı olarak kısmi bir çözüm ile başlar ve mevcut çözüm uzayını geliştirmeye çalışır. Algoritma reaktif tabu yapısını (reactive tabu tenure) kullanır. Reaktif tabu yapısı, tabu listesinin boyutunu ifade eder. İlgili çalışmada statik, dinamik ve reaktif olmak üzere üç çeşit tabu listesi yapısı kullanılmıştır. Bu üç farklı tabu listesi temel alınarak otomatik değişebilen FOO-şema (FOO-scheme) isimli yeni bir yaklaşım önerilmiştir. Bu yapı sayesinde klasik TABUCOL algoritmasından daha iyi performans elde ettiklerini belirtmişlerdir (Blöchliger ve Zufferey, 2008).

Bir başka çalışmada sparse graflarını boyamak için Açgözlü Rasgele Adaptif Arama Prosedürü (ARAAP) (Greedy Randomized Adaptive Search Procedure-GRASP)

algoritması geliştirilmiştir (Laguna ve Martí, 2001). Sparse graf, bir graftaki kavis sayısının o graftaki muhtemel kavis sayısından oldukça az olması durumudur (Lee ve Streinu, 2008). Yani algoritma kavis yoğunluğu az olan graflar için geliştirilmiştir. Algoritma temel olarak iki aşamadan oluşur. Temel aşamada başlangıç renklendirmesi için RLF algoritmasını, sonraki aşamada ise bulunan sonucu daha iyi bir çözüme götürmek için ise yerel arama metodu kullanmışlardır (Laguna ve Martí, 2001).

Avanthay ve ark. (2003), graftaki düğümleri boyarken tek bir komşuluk algoritması yerine farklı komşuluk algoritmalarının kullanılması durumunda problemin yerel bir çözüme takılma ihtimalinin düşürüleceği ve daha iyi çözümler bulunabileceği düşüncesiyle Mladenović ve Hansen (1997) tarafından literatüre kazandırılmış olan Değişken Komşuluk Arama (DKA) (Variable Neighborhood Search-VNS) algoritmasını GBP üzerine uygulamışlardır. DKA'da üç farklı komşuluk algoritmaları geliştirilmiştir. Düğüm komşulukları algoritmasında çakışan düğümlerin renkleri değiştirilir, renk (class) komşulukları algoritmasında ise yanlış renkle boyanmış düğümlerin tamamı ya da bazılarının renkleri değiştirilir, Artmayan Komşuluk (Non-increasing Neighborhoods) algoritmasında ise graftaki çakışmaların sayısının artmaması koşulu ile bazı düğümlerin renklerinin değiştirilmesi ile yeni çözüm kümeleri oluşturulur. Yazarlar, DKA algoritmasının graf boyama problemi için etkin bir yöntem sunduğunu söylemişlerdir (Avanthay ve ark., 2003). Değişken Alan Arama (DAA) (Variable Space Search-VSS) algoritması GBP için kullanılmış bir diğer yerel arama algoritmasıdır. Bu algoritma DKA algoritmasının yapısını kullanmaktadır. DKA'dan farklı olarak birden fazla arama uzayı ve birden fazla amaç fonksiyonu kullanır. DAA algoritmasında bir graf için çözüm uzayında çözümler aranırken yerel bir çözüme takıldığında kullandığı komşuluk algoritmasını değiştirerek, farklı çözümlere ulaşılmasını sağlamış olur. Böylece yerel bir çözüme takılmak daha zor olacaktır (Hertz ve ark., 2008).

Graf boyama problemi için birçok açgözlü ve yerel arama algoritması geliştirilmesine rağmen, bu metotların büyük random graflar için düşük bir performans sergilemesinden dolayı farklı çözüm arayışlarına gidilmiştir (Galinier ve Hertz, 2006). Popülasyon tabanlı algoritmalar ve evrimsel hibrit algoritmalar, GBP için yeni çözüm önerileri getirmiştir. Graf boyama problemi için farklı metasezgisel algoritmalar kullanılmıştır. Bu alanda en çok kullanılan algoritmalardan biri genetik algoritmadır. Graf boyama problemi kombinasyonel bir problem olduğu için genetik algoritma bu alanda oldukça iyi sonuçlar vermiştir (Shen, 2003). Harita boyama problemleri için graf

boyama algoritmaları kullanılabilmektedir. Gwee ve ark. (1993) yaptıkları çalışmada genetik algoritma (GA) ile ABD haritasındaki tüm şehirleri dört renk kullanarak boyamışlardır (Gwee ve ark., 1993).

Fleurent ve Ferland (1996), yaptıkları çalışmada genetik algoritma ile bazı komşuluk arama algoritmalarını birlikte kullanarak iyi sonuçlar elde etmeyi hedeflemişlerdir. Yaptıkları çalışmada Tabu Arama için kısa süreli hafıza stratejilerini, genetik operatörler için ise uzun vadeli stratejik işlevleri yerine getirebilecek bir algoritma sunmuşlardır. Yaptıkları çalışmada GBP için iyi bir çözüm sunulmuştur

Ali ve ark. (1999) GBP için GA'ya dayalı bir çözüm sunmuşlardır. Yaptıkları çalışmada her bir çözüm kümesi popülasyondaki bir birey olarak tanımlanmıştır. Güçlü kodlama ve GA'nın farklı mutasyon ve çaprazlama operatörleri sayesinde iyi çözümlere ulaşıldığı söylenmiştir (Ali ve ark., 1999). Tagawa ve ark. (1999) yerel aramanın avantajları ile genel aramada etkin olan genetik algoritmanın avantajlarını kullanarak genetik tabanlı hibrid bir algoritma önermişlerdir. Çalışmada yeni bir çaprazlama yöntemi olan Harmonik çaprazlama kullanılmıştır. Yerel arama algoritmasını ise bulunan en iyi çözümlerin etrafındaki yerel en iyi çözüme ulaşmak için kullanmışlardır (Tagawa ve ark., 1999). Ray ve ark. (2010) GBP için evrimsel bir algoritma olan GA ile yeni bir yaklaşım önermişlerdir. Bu çalışmada özel bir özelliğe dayalı çift noktalı mutasyon adlı yeni bir operatör tanımlamışlardır. Bu özellik sayesinde GA'nın performansının fark edilir şekilde arttığını söylemişlerdir. Algoritma büyük ölçekli graflar üzerinde test edilmiş ve benzer algoritmalara göre daha iyi sonuçlar elde edildiği belirtilmiştir (Ray ve ark., 2010). Lü ve Hao (2010) GBP için MACOL adında memetik bir algoritma önermişlerdir. Bu algoritma, çok bireyli adaptif çaprazlama (multi-parent crossover - AMPaX), mesafe ve kaliteye bağlı güncelleme gibi birkaç özellik içermektedir (Lü ve Hao, 2010). Sethumadhavan ve Marappan (2013), yaptıkları çalışmada GA'nın çaprazlama ve mutasyon operatörleri için yeni yaklaşımlar önererek GBP için yeni bir çözüm üretmişlerdir. Yapılan çalışmada tek bireydeki hatalı boyanmış düğümler için yeni bir çaprazlama ile hatalı boyanan düğüm için yeni bir mutasyon operatörü önermişlerdir. Önerilen çaprazlama ve mutasyon operatörleri sayesinde, graflardaki düğümleri boyamak için kullanılan renk sayısı azaltılmıştır (Sethumadhavan ve Marappan, 2013).

Croitoru ve ark. (2002) yaptıkları çalışmada seçilecek düğümün sırası ile düğüme verilecek renk arasındaki ilişki için yeni bir evrimsel yaklaşım önermişlerdir. Bu yaklaşımda simetrik bir çözüm uzayı bulunmaktadır ve bu çözüm uzayında evrimsel

algoritma ile diğer arama tekniklerinin birlikte kullanılması ile çözüm aranmıştır. Yapılan çalışmanın GBP için etkin bir çözüm sunduğu belirtilmiştir (Croitoru ve ark., 2002).

Costa ve Hertz (1997), GBP için KKA'ya dayalı bir yaklaşım sunmuşlardır. Yapılan çalışmada KKA, DSATUR ve RLF algoritmaları ile birlikte kullanılmıştır. Çalışmada önerilen algoritmalar rastgele oluşturulan graflar üzerinde test edilerek başarıları karşılaştırılmıştır (Costa ve Hertz, 1997). Ahn ve ark. (2003), GBP için Modifiye edilmiş KKA önermişlerdir. Yaptıkları çalışmada ANTCOL algoritmasını XRLF algoritması ile kombine ederek ANT_XRLF algoritmasını geliştirmişlerdir. ANT_XRLF algoritması, RLF algoritmasının rehberliğinde geliştirilmiştir. Daha sonra ANT_XRLF algoritması ANT_Random, ANT_LF, ANT_SL, ANT_DSATUR ve ANT_RLF algoritmaları ile karşılaştırılarak sonuçlar verilmiştir (Ahn ve ark., 2003).

Ge ve ark. (2010), Kaotik Karınca sezgisel yaklaşımını (KKS) (Chaotic Ant Swarm-CAS) temel açgözlü bir algoritma olan First Fit (FF) algoritması ile kombine ederek GBP için yeni bir yaklaşım önermişlerdir. Önerdikleri CASCOL algoritması sürü zekâsının karıncaların kaotik davranışları ile birleştirilmesi sonucu oluşturulan karınca kolonisi tabanlı bir algoritmadır. CASCOL algoritmasının GBP için etkin bir yöntem sunduğunu belirtmişlerdir (Ge ve ark., 2010).

GBP için uygulanan bir diğer algoritma da bal arılarının arasındaki ilişkiden esinlenerek oluşturulan Bal Arılarında Evlilik (BAE) (Marriage in honey Bees-MBO) algoritmasıdır. BAE algoritması farklı sezgisel algoritmaların karışımı bir algoritmadır. Bessedik ve ark. (2011), önerdikleri BEESCOL algoritmasında bir işçi arı yerel arama, tabu arama ya da önerdikleri karınca kolonisi tabanlı IACSCOL yapılarını kullanmışlardır. İşçi arılar başlangıç çözümü ile çaprazlama yapılan aşamalarda kullanılmaktadır. BEESCOL algoritmasında bir ya da birkaç kraliçe arı rastgele ya da graf boyama problemi için kullanılan algoritmalardan olan RLF ya da DSATUR algoritmaları ile üretilmiştir (Bessedik ve ark., 2011).

Faraji ve Javadi (2011) arıların davranışlarından (BEECOL) esinlenerek GBP için yeni bir yaklaşım önermişlerdir. Başlangıç çözümü için kâşif arılar kullanılmıştır. Daha sonra oluşturulan çözümlerin maliyet hesabı yapılmıştır. Bu çözümlerden iyi olanlar işçi arılar ile değiştirilmiştir. İşçi arılar bu bilgileri yaptıkları sallanma dansı ile tüm arılar ile paylaşmaktadırlar. Oluşan yeni çözüme komşuluk araması uygulanmış ve arının mevcut bilgileri güncellenmiştir. Önerdikleri algoritma GBP için başarılı sonuçlar vermiştir (Faraji ve Javadi, 2011).

Consoli ve ark. (2013), GBP için Karınca Kolonisi Algoritması (KKA) ve Yapay Arı Kolonisi (YAK) algoritmalarıyla oluşturulmuş KKA-GBP (AS-GCP) ve YAK-GBP (ABC-GCP) olarak adlandırdıkları iki yeni yaklaşım önermişlerdir. Bunlardan ilki temel olarak açgözlü çaprazlama olarak adlandırılan GPX yaklaşımını kullanmaktadır. Yerel arama mekanizmasını ise karıncaların feromon yapısı ile birleştirmişlerdir. Diğer yaklaşımda ise mutasyon operatörü, GPX'in geliştirilmiş bir versiyonu ile yerel aramanın sıcaklık mekanizmasını kullanmıştır. İki algoritma da GBP için etkin birer çözüm sunmaktadır (Consoli ve ark., 2013).

Standart PSO algoritması sürekli problemlerin çözümü için geliştirilmiştir. Qin ve ark. (2011) yapmış oldukları çalışmada PSO algoritmasını ayrık bir problem olan GBP için uyarlamışlardır. PSO algoritmasını ayrık probleme uygulamak için PSO'nun standart aritmetik operatörlerini yeniden tanımlamışlardır. PSO'da parçacığın yeni değeri ile bir önceki değeri arasındaki değişim için bir algoritma tanımlamışlardır. Daha sonra oluşturulan ayrık PSO algoritmasını yerel bir algoritma ile hibrid bir şekilde kullanmışlardır (Qin ve ark., 2011).

Fister ve Brest (2011) yapmış oldukları çalışmada Evrimsel Differansiyel Algoritmasını GBP'ye uyarlamak için bu algoritmayı kendinden adaptif (self-adaptive) hibrid bir şekilde tanımlamışlar ve graflar üzerine uygulamışlardır. (Fister ve Brest, 2011).

Abbasian ve ark. (2011) yapmış oldukları çalışmada Sıralı Sezgisel Graf Boyama Algoritmasını (SSGBA) (Sequential Graph Coloring Heuristic Algorithm-SGCHA), metasezgisel bir algoritma olan Kültürel Algoritma (KA) (Cultural Algorithm-CA) ile birlikte kullanarak GBP için bir algoritma önermişlerdir. Bu algoritma iki aşamadan oluşmaktadır. Bu aşamalar tahmin aşaması (EP) ile geliştirme ya da iyileştirme (IP) aşamasıdır. EP aşamasında SSGBA algoritmasını kullanarak boyanan graf için kullanılabilecek renk sayısında bir üst sınır belirlenmiştir. Daha sonra IP aşamasında popülasyondaki bireyler bu renk sayısı üzerinden oluşturulmuştur. Mevcut bireyleri iyileştirmek için GA'nın mutasyon ve çaprazlama operatörlerinden faydalanılmıştır. Önerilen algoritmayı bazı Benchmark grafları üzerinde uygulayarak test etmişlerdir (Abbasian ve ark., 2011).

Hsu ve ark. (2011) GBP'nin çözümü için Modifiye turbülent PSO (Modified Turbulent Particle Swarm Optimization-MTPSO) algoritmasını önermişlerdir. Bu algoritma graf boyama problemindeki planar graflar üzerine uygulanmıştır (Hsu ve ark.,

2011). Planar graf, düğümleri bağlayan kavislerden herhangi ikisi birbirini kesmeyecek şekilde çizilebilen özel bir tür graftır (West, 2001).

Ruiz ve Romay (2011) yaptıkları çalışmada GBP için Yerçekimi Algoritması (YA) (Gravitational Search-GS) tabanlı yeni bir yaklaşım önermişlerdir. Yerçekimi algoritması Newton'un yerçekimi yasası ile Reynolds'un parçacık sürü algoritmalarından esinlenerek ortaya çıkmış bir algoritmadır (Reynolds, 1987). Aslında bakıldığında sürü hareketi bireysel hareketleri kısıtlar, ama bu durum karmaşık problemlerin çözümüne yardımcı olmaktadır. Yapılan çalışmada ajanların (agent's) davranışları GBP'nin çözümü için tasarlanmıştır. Yerçekimi algoritmasındaki ajanlar parçacıklar gibi hareket ettirilmişlerdir. Bu çalışmada kullanılan YA algoritmasında bulunan ajanlar düğümlere verilen renklerden sorumludurlar. Ajanların sahip olduğu dost ya da düşman bilgisine göre graftaki düğümlerin renkleri belirlenmiştir. Önerilen algoritma Benchmark grafları üzerinde test edilmiştir (Ruiz ve Romay, 2011).

Mahmoudi ve Lotfi (2016), Modifiye edilmiş Guguk Kuşu Algoritmasını (MGKA) (Modified Cuckoo Optimization Algorithm-MCOA) kullanarak Benchmark grafları için etkili bir yöntem sunmuşlardır. Graf boyama problemi kombinasyonel bir problem olduğu için arama uzayı ayrıktır. Ancak Guguk Kuşu Algoritması (GKA) (Cuckoo Optimization Algorithm-COA) sürekli problemlerin çözümü için geliştirilmiştir (Yang ve Deb, 2009). Mahmoudi ve Lotfi, GKA algoritmasını GBP'ye uygun hale getirmek için algoritmanın bazı parametrelerinde değişiklikler yapmışlardır. GKA algoritmasının standart aritmetik operatörleri ile göç operatörü ayrık uzay için yeniden tanımlanmıştır. Bu çalışmada ilk olarak açgözlü bir algoritma ile graflar renklendirilerek, başlangıç çözüm kümeleri oluşturulmuştur. Bu çözümler anne guguk kuşu (mother cuckoo) olarak kabul edilmiş ve yumurtalama algoritması (lay egg) kullanılarak çocuk guguk kuşu (child cuckoo) çözümleri oluşturulmuştur. Son olarak bulunan çözümlere Tabu Arama algoritması uygulanarak yereldeki en iyi çözüme ulaşılmaya çalışılmıştır. Sonuç olarak MGKA, sezgisel yöntem kullanarak elde edilen ilk renklendirme işleminden sonra gelen renk sayısını düzgün renklendirme kuralına bağlı olarak azaltmaya çalışmıştır (Mahmoudi ve Lotfi, 2015).

J. Agrawal ve S.Agrawal (2015) GBP'nin çözümü için PSO algoritmasına dayalı yeni bir yaklaşım önermişlerdir. PSO algoritması oldukça basit ve etkili bir optimizasyon tekniğidir. Ancak PSO'nun en büyük dezavantajı yerel optimuma takılma ihtimalinin yüksek olmasıdır. Bundan dolayı yapılan çalışmada bu dezavantajın üstesinden gelmek için hızlandırılmış PSO (HPSO) (Acceleration Based Particle Swarm

Optimization-APSO) yaklaşımı önerilmiştir. HPSO algoritmasını ve standart PSO algoritmasını DIMACS'ın graf örnekleri üzerinde test ederek karşılaştırmışlardır (Agrawal ve Agrawal, 2015).

Kurbağa Sıçrama Algoritması (KSA) (Suffled Frog Leaping Algorithm- SFLA) ilk olarak su dağıtım şebekesi probleminin çözümünde kullanılmıştır (Eusuff ve Lansey, 2003). Eusuff ve Lansey (2003) yapmış oldukları çalışmada kurbağaların doğadaki hareketlerinden esinlenerek memetik tabanlı yeni bir yaklaşım önermişlerdir. Algoritma memetik evrimleri kullanarak yerel arama uzayında bir bireyden diğer bir bireye bilgi geçişini sağlar. KSA'nın yerel arama yapısı PSO algoritmasına benzerlik gösterir. Çalışmadaki karıştırma (shuffling) yapısı sayesinde yerel aramada bireyler arasında gerçekleşen bilgi değişimi ile genel en iyiye ulaşılması sağlanmıştır (Eusuff ve Lansey, 2003).

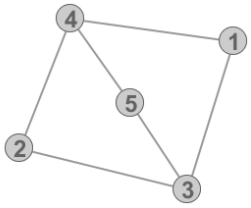
Chittineni ve ark. (2011) Saf KSA'nın memetik evrim sürecinde en kötü kurbağanın konumunu güncellemek için kullanılan yapıda değişikliğe gitmişler ve en kötü kurbağanın yerine rastgele bir kurbağa oluşturmak yerine en iyi kurbağayı yerleştirmişlerdir. Saf KSA'da yapılan bu değişiklik sonucu oluşturulan yeni KSA'yı kümeleme problemine uygulayarak performans karşılaştırması yapmışlardır. (Chittineni ve ark., 2011). Karakoyun (2015) yapmış olduğu çalışmada Saf KSA'nın memetik evrim aşamasında maliyet açısından en kötü kurbağanın konumunun güncellenmesi sırasında sıçrama miktarının belirlenmesi için Adaptif Kurbağa Sıçrama Algoritmasını (AKSA) önermiştir. Deneysel sonuçlara göre AKSA'nın kümeleme problemlerinin çözümü için etkin bir yöntem olduğu gösterilmiştir (Karakoyun, 2015).

KSA kullanarak farklı alanlarda çok farklı çalışmalar gerçekleştirilmiştir. Panda ve ark. (2014) KSA'yı çok katmanlı yapay sinir ağlarının (YSA) (Artificial Neural Network-ANN) eğitilmesi sürecinde kullanmışlardır (Panda ve ark., 2014). Luo ve ark. (2014) büyük ölçekli kaynak tahsisi problemini çözerken hem maliyeti azaltacak hem de çevresel olumsuzlukları düşürecek bir çözüm üretmek amacıyla KSA temelli modifiye edilmiş kurbağa sıçrama algoritmasını önermişlerdir. Mehta ve Banati (2014) çalışmalarında demografik probleminin çözümü için KSA'ya dayalı bir filtreleme yöntemi ortaya koymuşlardır (Mehta ve Banati, 2014). Samuel ve Rajan (2015) ise yaptıkları çalışmada uzun vadeli jeneratör bakım planlaması probleminin çözümü için hibrid PSO tabanlı GA ile hibrid PSO tabanlı KSA'yı önermişlerdir (Samuel ve Rajan, 2015).

3. MATERYAL VE YÖNTEM

3.1. Graflar Hakkında Temel Tanımlamalar

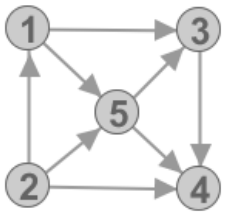
Graf (Çizge Kuramı) teorisi, düğüm (tepe) olarak adlandırılan noktalar ve her biri bu noktaları veya sadece noktanın kendisini birleştiren ve kavis (kenar, ayrıt) olarak adlandırılan çizgiler topluluğudur (Biggs ve ark., 1976; Buckley ve Harary, 1990; West, 2001). Örneğin Türkiye haritasındaki her bir şehir düğümleri, şehirleri birbirine bağlayan yollar da kavisleri temsil etmektedir. $G(V,E)$ olarak verilen bir G grafı için n düğüm sayısı, m ise kavis sayısı olsun. Bu durumda düğümler $V = \{v_1, v_2, \ldots\ldots, v_n\}$ ve kavisler de $E = \{e_1, e_2, \ldots\ldots, e_m\}$ olarak tanımlanır. Kavisler iki düğüm arasındaki bağlantıyı temsil etmektedir. Eğer iki düğüm arasında en az bir bağlantı söz konusu ise bu düğümlere ***"komşu düğüm"*** denilmektedir (West, 2001). Şekil 3.1'de $n=5$ olan bir graf örneği verilmiştir.



Şekil 3.1. 5 düğümlü graf

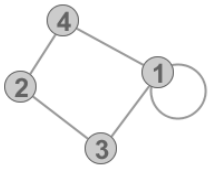
Yönsüz Graf: Bir $G(V,E)$ grafında bulunan bir kavis için başlangıç ve bitiş düğümünün hangi düğümler olduğu belirtilmemiş ise bu kavis yönsüzdür. Eğer tüm kavisler böyle ise bu grafa ***"yönsüz graf"*** denir (Skiena, 1990). Şekil 3.1'de verilen graf bir yönsüz graftır.

Yönlü Graf: Bir $G(V,E)$ grafındaki düğümlerin birbiri ile bağlantısını sağlayan kavisler yönlü ise, yani başlangıç ve bitiş düğümleri belirlenmişse bu grafa ***"yönlü graf"*** denir (Skiena, 1990). Şekil 3.2'de verilen graf yönlü grafdır.



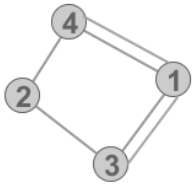
Şekil 3.2. Yönlü graf

Döngü (Bukle): Bir $G(V,E)$ grafında bulunan ve başlangıç ve bitiş düğümü aynı olan kavise ***"döngü (loop)"*** denir (West, 2001). Şekil 3.3'te verilen graf örneğinde 1 no'lu düğümde döngü bulunmaktadır ve bu düğüm bir bukle düğümdür.



Şekil 3.3. Döngü içeren graf

Paralel Kavis: Bir grafta aynı düğüm çifti arasında iki veya daha çok kavis varsa buna ***"çoklu ya da paralel kavis"*** denir (West, 2001). Şekil 3.4'te verilen $v_1$ve v_3 ile v_1 ve v_4 düğümleri çoklu kavisler ile birbirine bağlıdır.

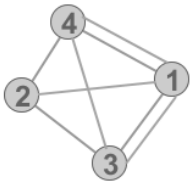


Şekil 3.4. Paralel kavis içeren graf

Komşu Düğüm: Bir $G(V,E)$ grafında, V düğümler kümesini temsil etmektedir. $V = \{v_1, v_2, \ldots\ldots, v_n\}$ olarak gösterilir. v_1 ve v_2 düğümleri eğer herhangi bir kavis ile birbirine bağlı ise bu düğümlere ***"komşu düğüm"*** denir (West, 2001). Şekil 3.4'te verilen v_3 ve v_2 nolu düğümler komşu düğüm iken, v_3 ve v_4 nolu düğümleri bağlayan bir kavis olmadığı için komşu düğüm değillerdir.

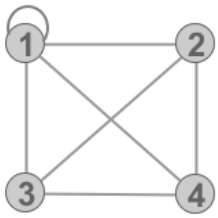
Basit Graf: Herhangi bir döngü veya paralel kavis içermeyen yönsüz graflara ***"basit graf"*** denir (West, 2001). Şekil 3.1'de verilen graf basit graftır.

Çoklu Graf: Yönsüz ve paralel kavis içeren, ancak döngü içermeyen graflara **"*çoklu graf*"** denir (West, 2001). Şekil 3.5'te verilen graf bir çoklu graftır.



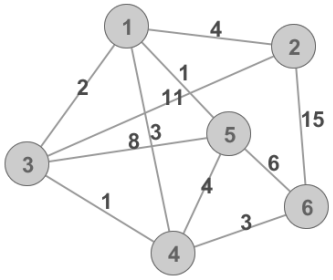
Şekil 3.5. Çoklu graf

Sahte Graf (Pseudo Graf): Paralel kavis veya döngü (loop) içeren graflara ***"sahte graf"*** denir (Harary, 1994; Zwillinger, 2002). Şekil 3.6'da verilen graf bir sahte graftır.



Şekil 3.6. Pseudo graf

Ağırlıklı Graf: Bir $G(V,E)$ grafı için, kavisler kümesi $E = \{e_1, e_2, \ldots\ldots, e_m\}$ olarak tanımlanır. Her bir kavisin sayısal bir değere sahip olduğu graf ***"ağırlıklı graf"*** olarak tanımlanır (Weisstein, 2015). Her bir ağırlıklı kavis $w(e_m)$ ile gösterilir. Şekil 3.7'de ağırlıklı graf gösterilmiştir.



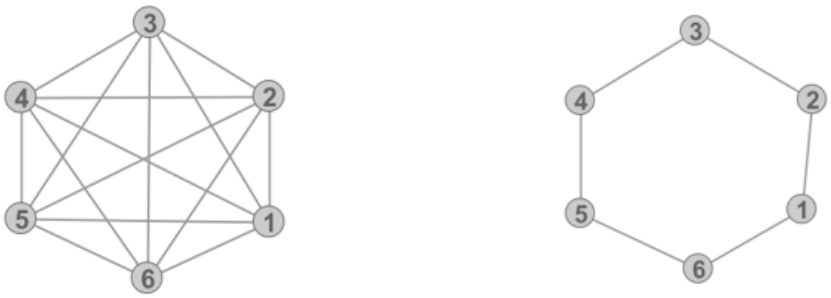
Şekil 3.7. Ağırlıklı graf

Tam Graf: Verilen bir $G(V,E)$ grafı için eğer her bir düğüm çoklu kavis olmadan graftaki diğer tüm düğümlere bağlanmışsa ve grafta hiç döngü yoksa bu grafa ***"tam graf"*** denir (Gross ve Yellen, 1998). Şekil 3.8'de 4 düğümlü ve 6 düğümlü tam graflara ait örnekler verilmiştir.

(a) 4 düğümlü tam graf (b) 6 düğümlü tam graf

Şekil 3.8. Tam graf örmekleri

Alt Graf (Sub Graf): $G(V,E)$ olarak verilen bir G grafı için V düğüm kümesi ve E kavis kümesi olmak üzere, $V_t \subseteq V$ ve $E_k \subseteq E$ şartını sağlayan $G'(V_t, E_k)$ grafı çıkarılabiliyorsa, bu grafa ***"alt graf"*** denir ve G′ ile gösterilir (Harary, 1994; West, 2001). Bir graftan birden fazla farklı alt graflar çıkarılabilir. Şekil 3.9'da G ve çıkarılabilecek örnek bir G′ grafı verilmiştir. G′ grafı G grafının bir alt grafıdır.



(a) G grafı (b) G grafına ait G′ grafı

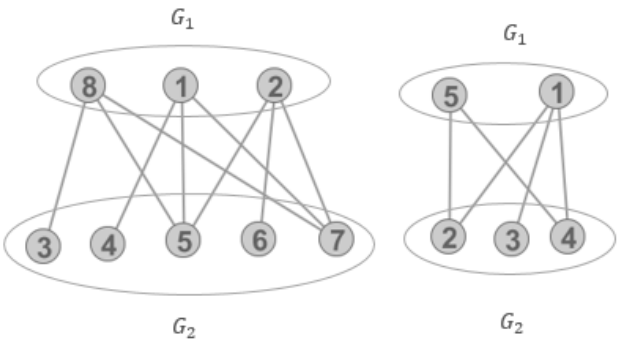
Şekil 3.9. G grafı ve G′ grafı

Bipartite Graf: $G(V,E)$ olarak verilen bir graf için V düğümler kümesidir. Bir grafı oluşturan düğümler iki ayrı kümeye ayrılabiliyorsa buna ***"bipartite graf"*** denir. Bu ayırma işleminde izlenecek yol; bir kavis ile birbirine bağlanabilecek durumda olan düğümleri aynı küme içerisine yerleştirmemektir. Yani bağlantısız düğümleri iki ayrı kümede tutarak, grafı iki parçaya ayırabiliriz (Gross ve Yellen, 1998). Şekil 3.10'da 8 düğümlü ve 5 düğümlü bipartite graflar verilmiştir. G, bir bipartite graf ise, aşağıdaki şartları sağlamalıdır:

1) $V(G) = V(G_1) \cup V(G_2)$

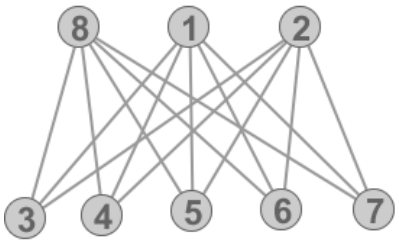
2) $|V(G_1)| = m, |V(G_2)| = n \; ise \; |V(G)| = m + n$

3) $V(G_1) \cap V(G_2) = \emptyset$



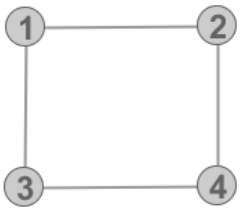
Şekil 3.10. Bipartite graflar

Tam Bipartite Graf : $G(V,E)$ olarak verilen bir bipartite graf için eğer ayrı iki ayrı kümede bulunan düğümlerin her biri diğer kümedeki tüm düğümler ile bağlantılı ise buna ***"tam bipartite graf"*** denir (Gross ve Yellen, 1998). V düğümler kümesi olarak tanımlanmıştır. Şekil 3.11'da verilen 8 düğümlü graf için $V_A = \{v_1, v_2, v_8\}$ ve $V_B = \{v_3, v_4, v_5, v_6, v_7\}$ olacak şeklinde iki ayrı kümeye ayrılabilmektedir. Şekil 3.11'de verilen graf için V_A kümesindeki her bir düğüm, V_B kümesinde bulunan tüm düğümler ile bağlantılıdır.



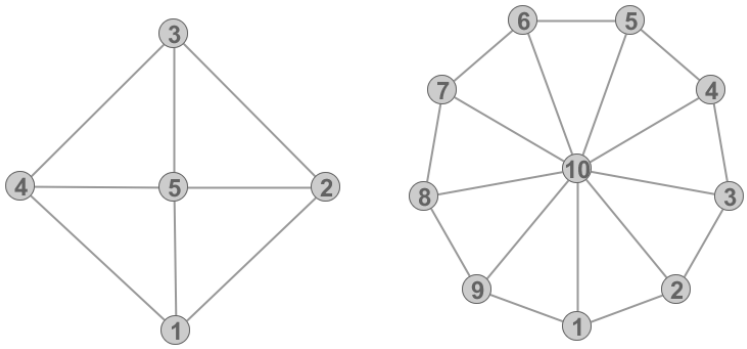
Şekil 3.11. Tam biparite graf

Çember Graf: Bir $G(V,E)$ grafı için düğümlerin çember yapısında birbiri ile bağlantılı olmasına ***"çember graf"*** denir. Çember graf C_n olarak tanımlanır. n graftaki düğüm sayısıdır. Çember graf düğüm sayısının tek ya da çift olması durumuna göre tek çember graf ya da çift çember graf olarak adlandırılır (Gross ve Yellen, 1998). Şekil 3.12'de verilen 4 düğümlü graf bir çift çember graf örneğidir.



Şekil 3.12. Çember graf

Tekerlek Graf (Wheel Graph): Çember grafa, ortaya yeni bir düğüm eklenerek tekerlek graf oluşturulabilir. Tekerlek graf W_n olarak gösterilir. n graftaki düğüm sayısıdır (Harary, 1994; Gross ve Yellen, 1998). Tekerlek graf $K_1 + C_{n-1}$ olarak da tanımlanabilir. K_1 tek düğümlü grafı, C_{n-1} ise $n-1$ düğümlü çember graftır. Şekil 3.13'de verilen graflar tekerlek graf örnekleridir.



(a) 5 düğümlü tekerlek graf (b) 10 düğümlü tekerlek graf

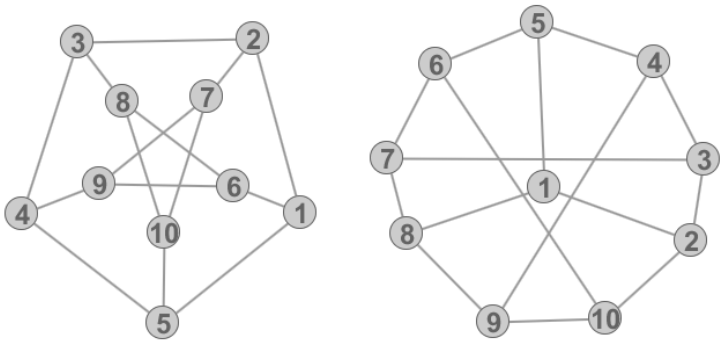
Şekil 3.13. Beş düğümlü ve on düğümlü tekerlek graflar

Şekil 3.13'de verilen tekerlek graflar için 5 düğümlü tekerlek graftan 5.düğüm, 10 düğümlü graftan ise 10.düğüm çıkarılırsa oluşacak olan graf, çember graftır.

Yıldız Graf: Yıldız grafta kavislerin bağlantısı yıldız görünümündedir. n düğümlü bir $G(V,E)$ grafı için yıldız graf S_n ile gösterilir. Yıldız grafta merkez düğümün derecesi $n-1$ iken, diğer tüm düğümlerin dereceleri daima 1'dir (Harary, 1994).

r*-Regüler Graf:** Verilen bir $G(V,E)$ grafında bulunan düğümlerin derecesi aynı ve r gibi bir sayıya eşitse, bu graf düzenli bir dağılım göstermiştir. Tüm düğümlerin derecesi r olduğundan, bu grafa ***"regüler graf" ya da "***r-regüler***" grafı denir (Buckley ve Harary, 1990).

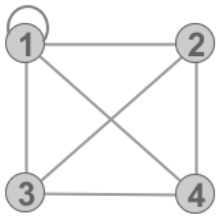
Petersen Graf: Petersen grafı 1989'da Julius Petersen tarafından bulunmuştur. Petersen grafı 10 düğüm ve 15 kavisten oluşmaktadır (West, 2001). Petersen grafında tüm düğümler sadece üç renk kullanılarak boyanabilmektedir. Petersen grafı aynı zamanda $3-regüler$ graftır. Şekil 3.14'de Petersen grafının farklı çeşitleri verilmiştir.



Şekil 3.14. Petersen Graf örnekleri

Düğüm Derecesi: Bir $G(V,E)$ grafında $V=\{v_1, v_2, \ldots . v_n\}$ olmak üzere, V düğümler kümesini göstermektedir. v_1 düğümünün derecesi, o düğüme bağlı olan toplam kavis

sayısıdır ve $deg(v_1)$ ile gösterilir (Buckley ve Harary, 1990). Düğüm derecesi hesaplanırken aksi belirtilmemiş ise döngüler düğüm derecesini iki arttırırlar.



Şekil 3.15. Beş düğümlü G grafı

Şekil 3.15'te beş düğümlü bir G grafı verilmiştir. G grafında v_1, v_2, v_3 ve v_4 düğümleri bulunmaktadır. v_1 düğümünün düğüm derecesi, v_1 düğümüne bağlı olan toplam kavis sayısıdır. Biri döngü olmak üzere v_1 düğümüne dört kavis bağlantısı vardır. Döngüler düğüm derecesini iki arttırdığından dolayı v_1 düğümünün düğüm derecesi $deg(v_1) = 5$'tir.

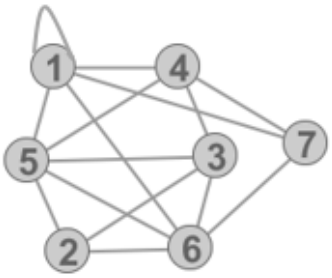
Minimum Düğüm Derecesi: Bir $G(V, E)$ grafının en küçük düğüm derecesine "**minimum düğüm derecesi**" denir ve $\delta(G)$ ile gösterilir (Skiena, 1990).

Maksimum Düğüm Derecesi: Bir $G(V, E)$ grafının düğüm derecelerinin en büyüğüne grafın "**maksimum düğüm derecesi**" denir ve $\Delta(G)$ ile gösterilir (Skiena, 1990).

Komşuluk Matrisi: Komşuluk matrisi aynı zamanda "**bağlantı matrisi**" olarak da bilinmektedir. Komşuluk matrisi komşu düğümlerin kavis bağlantılarına göre belirlenmektedir (West, 2001). $G = (V, E)$ verilen graf için düğüm kümesi $V = \{v_1, v_2, \ldots . v_n\}$ olarak tanımlanmaktadır. G grafı için komşuluk matrisi eşitlik (3.1)'e göre oluşturulmaktadır. Şekil 3.16'da verilen yönsüz graf için komşuluk matrisi Çizelge 3.1'de verilmiştir. G grafı için komşuluk matrisi; a_{ij}, v_i ve v_j'yi bağlayan kavis sayısı olmak üzere, bu ilişki A matrisi üzerinde gösterilmiştir. Yönsüz bir grafta komşuluk matrisi v_i ve v_j'yi birbirine bağlayan kavislerin sayısını gösterdiğinden dolayı a_{ij}ve a_{ji} hücrelerindeki sayısal değerler birbirine eşit olmalıdır. Yani komşuluk matrisi simetrik olmalıdır. Komşuluk matrisi ile herhangi bir düğümün derecesi kolayca belirlenebilir. Örneğin v_1 düğümü için düğüm derecesi hesaplanırken, bu düğüm için döngü yoksa bu düğümün derecesi A matrisinin birinci satırındaki değerlerin toplamıdır. Her bir döngü dereceyi iki kat arttırdığından birinci düğüm için düğüm derecesi hesaplanırken hesaplama işlemi doğru yapılmalıdır. Komşuluk matrisinin birinci satırındaki değerler

toplanırken komşuluk matrisinde birinci düğümün döngü içeren bağlantı değerinin iki katı alınarak hesaplama yapılmaktadır. (West, 2001).

$$A = \begin{cases} 1, \text{Eğer } v_i \; ile \; v_j \; arasında \; kavis \; varsa \\ 0, \text{Eğer } v_i \; ile \; v_j \; arasında \; kavis \;\; \text{yoksa} \end{cases} \tag{3.1}$$



Şekil 3.16. Graf örneği

Çizelge 3.1. G grafının komşuluk matrisi

A	v_1	v_2	v_3	v_4	v_5	v_6	v_7
v_1	1	0	0	1	1	1	1
v_2	0	0	1	0	1	1	0
v_3	0	1	0	1	1	1	0
v_4	1	0	1	0	1	0	1
v_5	1	1	1	1	0	1	0
v_6	1	1	1	0	1	0	1
v_7	1	0	0	1	0	1	0

Şekil 3.16'de verilen G grafı döngü içeren yönsüz ve ağırlıksız bir graftır. Ağırlıksız olduğu için, her kavisin ağırlığı birbirine eşittir. Örneğin birinci düğüm için düğüm derecesi komşuluk matrisine göre hesaplanacak olursa; bu düğümün derecesi:

$$Deg(v_1) = 2 * a_{1,1} + a_{1,2} + \; a_{1,3} + a_{1,4} + a_{1,5} + a_{1,6} + a_{1,7}$$

olarak hesaplanmaktadır. Birinci düğümde döngü (loop) olduğu için $a_{1,1}$ hücresinin iki katı alınmıştır. Sonuç olarak $Deg(v_1) = 6$ olarak bulunmaktadır.

3.2. Graf Boyama Problemi

Dört Renk Problemi olarak ortaya çıkan graf boyama problemi ünlü matematikçi Francis Gutrie tarafından ortaya atılmıştır. F. Gutrie İngiltere haritasını dört renk ile boyamaya çalışmıştır. Teoremini ispatlayamadan vefat etmiştir. Gutrie'nin kardeşi problemi Londra Üniversitesindeki matematik hocası Augustus De Morgan'a iletmiştir. 1852 yılında De Morgan problem eline ulaştıktan sonra problemi İrlandalı bilim adamı William Rowan Hamilton'a bir mektup göndererek iletmiştir. Bu mektup şu anda Dublin'deki Trinity Üniversitesinde tutulmaktadır (Fritsch ve Fritsch, 1998).

Graf boyama işlemi düğüm boyama ya da kavis boyama olmak üzere iki şekilde yapılabilmektedir (Gross ve Yellen, 1998). Düğüm boyamada amaç, komşu düğümlerin aynı rengi almayacak şekilde tüm düğümlerin boyanması iken; kavis boyamada amaç komşu kavislerin aynı rengi almayacak şekilde boyama yapılmasıdır. Yapılan bu çalışmada graf boyama problemi için düğüm boyama kullanılmıştır.

$G = (V, E)$ yönsüz bir graf olsun. V düğüm kümesini, E ise kavis kümesini temsil etmektedir. Bir graftaki düğümler, komşu olan düğümler aynı rengi almayacak şekilde boyanmak şartı ile en az sayıda renk kullanarak boyama işlemi graf boyamada ulaşılmak istenen hedeftir. Yani bir graf için k renk kullanmışsak, bu graf $(k-1)$ renk ile boyanamamalıdır (West, 2001). Graf boyamada komşu iki düğüm aynı rengi alırsa "**çakışma**" (conflict) var demektir (Gross ve Yellen, 1998).

Bir G grafındaki tüm düğümlerin boyanması şartı ile bu düğümleri boyamak için kullanılabilecek minimum farklı renk sayısına "**kromatik sayı**" denir ve $\chi(G)$ ile gösterilir (Welsh ve Powell, 1967; Díaz ve Zabala, 1999). Ancak düğümler boyanırken grafta çakışma olmamalıdır. Yani iki komşu düğüm aynı rengi almamalıdır (Skiena, 1990; Gross ve Yellen, 1998).

Hiç kavis bulunmayan bir graf için kromatik sayı $\chi(G) = 1$, bipartite bir graf için ise kromatik sayı $\chi(G) = 2$'dir (Gross ve Yellen, 1998). Bir grafın kromatik sayısını hesaplamak için farklı yöntemler vardır. Kromatik sayı $\chi(G)$ hesaplamakta kullanılan bazı yöntemler bu çalışmada detaylı olarak açıklanmıştır.

- **Üst Sınır (Upper Bound):** k renk ile boyanabilen bir G grafı için kromatik sayı $\chi(G) \leq k$ olmak zorundadır (Gross ve Yellen, 1998).

- **Alt Sınır (Lower Bound):** Verilen bir G grafının alt grafı olan G' grafı k renk ile boyanabiliyorsa G grafı için kromatik sayı $\chi(G) \geq k$ olur (Gross ve Yellen, 1998).
- Verilen bir G grafı için k komşu düğüm sayısını temsil ediyorsa, G grafı için kromatik sayı $\chi(G) \geq k$ olur (Gross ve Yellen, 1998).
- **Klik (Clique):** Bir G grafının içinden tam graflar çıkarılabiliyorsa, bu alt tam graflarının her birine klik denir. Alt tam graflardan düğüm sayısının en çok olduğu alt tam graftaki düğüm sayısına maksimum klik denir ve $\omega(G)$ olarak gösterilir. Böyle bir G grafı için kromatik sayı $\chi(G) \geq \omega(G)$ olarak kabul edilir (Gross ve Yellen, 1998).
- Verilen bir G grafı için $\Delta(G)$ maksimum düğüm derecesi iken maksimum düğüm derecesine sahip düğüm ise en fazla komşuluğu bulunan düğümdür. Açgözlü bir algoritma ile graf boyanırken kullanılacak renk sayısı $\Delta(G) + 1$'den fazla olamaz. Yani böyle bir G grafı için kromatik sayı $\chi(G) \leq \Delta(G) + 1$ 'dir (Gross ve Yellen, 1998).
- **Brook'un Teoremi (Brooks Theorem):** Verilen bir G grafı tam graf ya da düğüm sayısı çift olan çember graf değilse bu graf için kromatik sayı $\chi(G) = \Delta(G) + 1$ 'dir (Brooks, 1941; Gross ve Yellen, 1998).
- **Polinomal Kromatik Sayı:** $P(G, k)$ olarak tanımlanır. G grafı, k ise renk sayısını gösterir. Polinomal kromatik sayı k renk sayısı üzerinden, G grafı için mümkün olan toplam çözüm sayısını temsil etmektedir (Gross ve Yellen, 1998).

3.3. Graflar için Temel Düğüm Boyama Yöntemleri

Bir grafın düğümleri, komşu iki düğüm aynı rengi almayacak şekilde boyandığında graf doğru bir şekilde boyanmış olmaktadır. Ancak belli şartlara bağlı olarak graftaki düğümler boyandığında ise aynı graf için daha az sayıda renk kullanılarak, bu grafın boyanıp boyanamayacağı, graf boyama problemin temel noktasıdır. Bir G grafı için kromatik sayı $\chi(G) = 3$ verilmişse, bu durumda G grafındaki düğümler 3 renkten daha az sayıda renk ile boyanamamalıdır. GBP için farklı sezgisel ve metasezgisel yöntemlerin geliştirilmesi, ayrıca her algoritmanın çalışma süresinin farklılık göstermesi ve her algoritmanın farklı sonuçlar bulması, graf boyama

probleminin NP-tam problem olarak nitelendirilmesinde etkilidir (Mahmoudi ve Lotfi, 2015)

Düğüm boyama problemi için farklı yöntemler geliştirilmiştir. Temel yöntemler düğüm sayısının az ve kavis bağlantılarının karmaşık olmadığı graflar için hızlı ve iyi sonuçlar vermektedir. Çünkü bu algoritmalar belirli kurallara göre dar bir çözüm uzayında çalışmaktadırlar. Ancak graftaki düğüm sayısı arttıkça veya kavis bağlantılarının karmaşıklığı arttıkça problemin karmaşıklığı da arttığı için Mahmoudi ve Lotfi (2015) metasezgisel yöntemlerin daha iyi sonuçlar verebileceğini söylemişlerdir (Mahmoudi ve Lotfi, 2015).

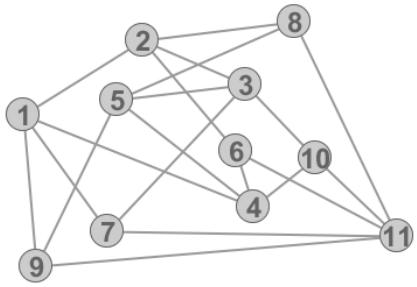
3.3.1. First Fit Algoritması (FF)

Düğüm boyama için kullanılan temel algoritmalardan biri First Fit (FF) algoritmasıdır. FF algoritmasına göre bir graftaki düğümler boyanırken boyama işlemi çok hızlı bir şekilde olmaktadır (Al-Omari ve Sabri, 2006). Düğüm kümesi $V = \{v_1, v_2, \ldots\ldots, v_n\}$, renk kümesi ise $R = \{r_1, r_2, \ldots\ldots, r_k\}$ olarak tanımlanan bir G grafı için FF algoritması, düğümler kümesindeki birinci düğümden başlayarak, sonuncu düğüme varıncaya kadar her düğümü, düğümler kümesinde bulunan sırasına göre boyamaktadır. Boyama işlemi sırasında boyanacak düğüm için en uygun renk seçilerek (boyanacak düğümün komşularının sahip olduğu renklerin dışındaki renklerden biri), boyama işlemi gerçekleştirilir. Böylece tüm düğümler boyandığından, renk kümesinden en az sayıda renk kullanılmış olur. Algoritma basit ve oldukça hızlı çalışmaktadır (Gebremedhin, 1999; Ge ve ark., 2010). FF algoritmasının çalışma adımları şöyledir:

- **Adım 1:** Bir renk kümesi oluşturulur (Başlangıçta renk kümesi boştur).
- **Adım 2:** İlk düğüm başlangıç düğümü olarak seçilir. İlk düğüm için ilk renklendirme yapılır ve renk kümesine ilk düğüme verilen renk eklenir.
- **Adım 3:** Düğümler kümesinde bulunan boyanmamış düğümlerden bir sonraki düğüm, boyama işlemi için seçilir.
- **Adım 4:** Seçilen düğüm için komşuluk matrisindeki komşuluklar dikkate alınarak renk kümesindeki mevcut renklerden biri seçilir. Eğer renk kümesinde bulunan renkler, seçilen düğümü boyamak için uygun değilse yeni bir renk tanımlanır. Seçilen düğüme yeni renk verilir ve yeni renk, renk kümesine eklenir.

- **Adım 5:** Tüm düğümler boyanıncaya kadar üçüncü adıma geri dönülür.

Hem FF algoritması hem de diğer temel düğüm boyama algoritmaları ile düğüm boyama işlemine ait örnekleri gerçekleştirmek için Benchmark graflarından *myciel3.col* grafı kullanılmıştır. *myciel3.col* grafı 11 düğüm, 20 kavisten oluşmaktadır. *myciel3.col* Benchmark grafını boyamak için kullanılabilecek en az renk sayısı dört olarak bilinmektedir. Yani bu graf için kromatik sayı $\chi(G) = 4$'tür. Şekil 3.17'de *myciel4.col* grafı, Çizelge 3.2'de ise bu grafa ait komşuluk matrisi verilmiştir.



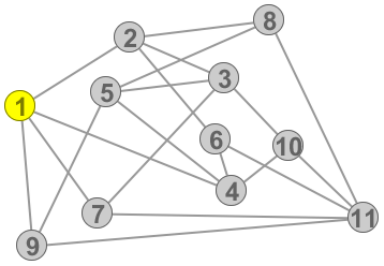
Şekil 3.17. *myciel3.col* grafı

Çizelge 3.2. *myciel3.col* grafına ait komşuluk matrisi

A	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}
v_1	0	1	0	1	0	0	1	0	1	0	0
v_2	1	0	1	0	0	1	0	1	0	0	0
v_3	0	1	0	0	1	0	1	0	0	1	0
v_4	1	0	0	0	1	1	0	0	0	1	0
v_5	0	0	1	1	0	0	0	1	1	0	0
v_6	0	1	0	1	0	0	0	0	0	0	1
v_7	1	0	1	0	0	0	0		0	0	1
v_8	0	1	0	0	1	0	0	0	0	0	1
v_9	1	0	0	0	1	0	0	0	0	0	1
v_{10}	0	0	1	1	0	0	0	0	0	0	1
v_{11}	0	0	0	0	0	1	1	1	1	1	0

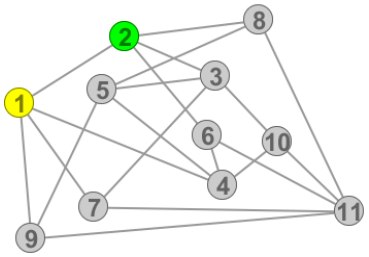
Şekil 3.17'de verilen graf ve Çizelge 3.2'de verilen komşuluk matrisine göre graftaki düğümler FF algoritması ile boyanacak olursa boyama işlemi sırasında algoritmanın çalışma adımları şunlardır:

Adım 1: Renk kümesi boş olduğu için ilk düğüme ilk renk verilir ve renk kümesine bu renk eklenir.



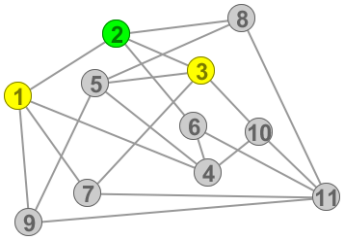
Şekil 3.18. FF algoritması birinci adım

Adım 2: İkinci düğüm için komşuluk matrisi kontrol edilir. Eğer bu düğüm önceki düğüm ile komşu değilse ilk renk bu düğüme de verilir, ancak bu iki düğüm arasında bir komşuluk söz konusu ise bu düğüme ikinci renk verilir ve ikinci renk, renk kümesine eklenir. Şekil 3.18 incelendiğinde v_1 ile v_2 komşu düğüm olduğundan v_2 düğümüne yeni bir renk verilmiştir. Şekil 3.19'da ikinci adım sonunda boyanan grafın son durumu verilmiştir.

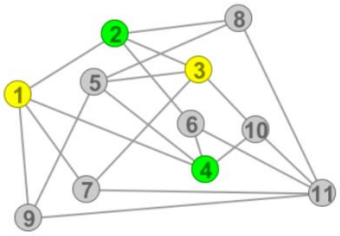


Şekil 3.19. FF algoritması ikinci adım

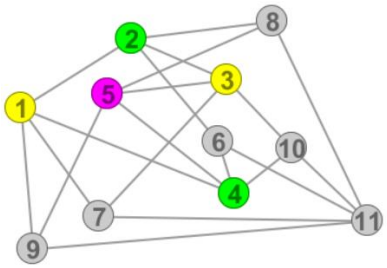
Tüm düğümler boyanıncaya kadar FF algoritmasının çalışma adımları tekrarlanır. Şekil 3.20'de algoritmanın çalışma adımlarına göre sırasıyla boyanan düğümler ve bu düğümlere verilen renkler gösterilmiştir. FF algoritması düğümleri, düğümler kümesindeki sırasına göre boyadığı için hızlı bir şekilde düğümleri boyamaktadır. Ancak bu avantajının yanında en büyük dezavantajı, düğümlerin özelliklerine bakmadığı için düğüm sayısı çok olan graflar için kullandığı renk sayısının da olması gerekenden daha fazla çıkmasıdır.



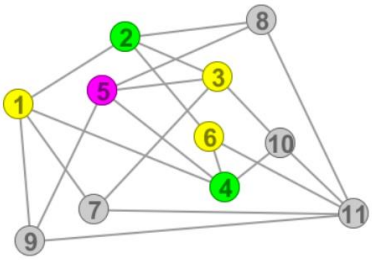
(a) FF algoritması üçüncü adım



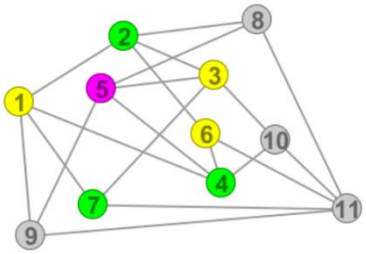
(b) FF algoritması dördüncü adım



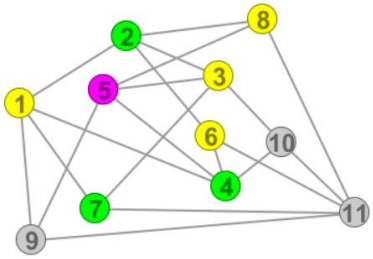
(c) FF algoritması beşinci adım



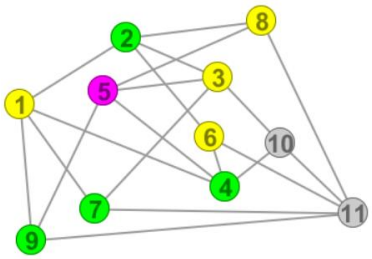
(d) FF algoritması altıncı adım



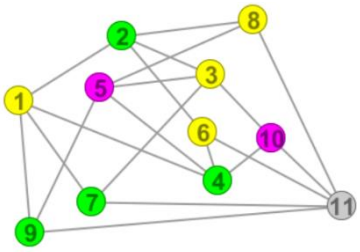
(e) FF algoritması yedinci adım



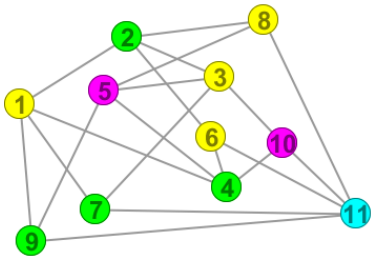
(f) FF algoritması sekizinci adım



(g) FF algoritması dokuzuncu adım



(h) FF algoritması onuncu adım



(i) FF algoritması onbirinci adım

Şekil 3.20. FF algoritmasına göre düğümlerin boyanma sırası ve renkleri

3.3.2. Largest Degree Ordering Algoritması (LDO)

Düğüm kümesi $V = \{v_1, v_2, \ldots\ldots, v_n\}$, renk kümesi $R = \{r_1, r_2, \ldots\ldots, r_k\}$ olarak tanımlanan bir G grafı, LDO algoritmasına göre boyanırken, boyama işleminin her bir adımında seçilecek olan düğüm, düğüm derecesi en büyük olan boyanmamış düğümdür (Al-Omari ve Sabri, 2006). Algoritmanın çalışma adımları:

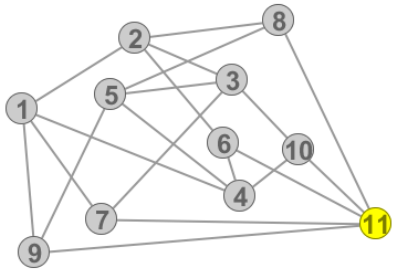
- **Adım 1:** Bir renk kümesi oluşturulur (Başlangıçta renk kümesi boştur). Her düğüm için düğüm derecesi hesaplanır ve $Deg(v_i)$ dizisinde tutulur. ($i = 1,2, \ldots, n$).
- **Adım 2:** $Deg(v_i)$ dizisinden düğüm derecesi en büyük olan boyanmamış düğüm renklendirme işlemi için seçilir.
- **Adım 3:** Seçilen düğüm, renk kümesinde bulunan renkler ile boyanmaya çalışılır. Eğer renk kümesi boş ya da renk kümesinde bulunan renkler seçilen düğümü boyamak için uygun değilse (seçilen düğümün komşularının sahip olduğu renkler ise) yeni bir renk tanımlanır. Tanımlanan yeni renk, hem renk kümesine eklenir hem de seçilen düğüme verilir.
- **Adım 4:** Tüm düğümler boyanıncaya kadar ikinci adıma geri dönülür.

myciel3.col grafı LDO algoritması ile boyanacak olursa boyama işlemi sırasında algoritmanın çalışma adımları şunlardır:

Çizelge 3.3. $myciel3.col$ grafına ait düğüm dereceleri

Düğümler (V)	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}
Düğüm Dereceleri (σ)	4	4	4	4	4	3	3	3	3	3	5

Adım 1: Çizelge 3.3'de $myciel3.col$ grafındaki her bir düğüme ait düğüm dereceleri verilmiştir. Düğüm derecesi en büyük olan boyanmamış düğüm v_{11} düğümü olduğundan, ilk olarak bu düğüm, boyama işlemi için seçilmiştir. Renk kümesi boş olduğu için v_{11} düğüme ilk renk verilip, bu renk, renk kümesine eklenmiştir. Şekil 3.21'de ilk boyama işlemi sonunda boyanan graf verilmiştir. Çizelge 3.4'te ise boyanabilecek düğümlere ait düğüm dereceleri tablosu verilmiştir.

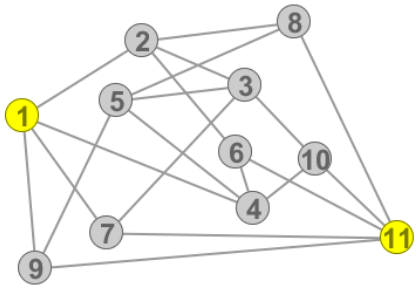


Şekil 3.21. LDO algoritması birinci adım

Çizelge 3.4. Boyanabilecek düğümlere ait düğüm dereceleri tablosu – 1

Düğümler (V)	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}
Düğüm Dereceleri (σ)	4	4	4	4	4	3	3	3	3	3	5

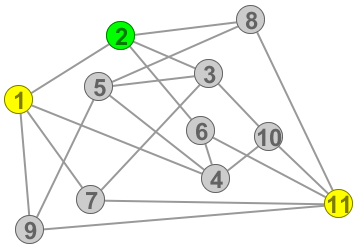
Adım 2: Çizelge 3.4'deki düğüm dereceleri tablosu kontrol edilir. Düğüm derecesi en büyük olan boyanmamış düğüm seçilir. v_1, v_2, v_3, v_4, v_5 düğümleri, düğüm dereceleri eşit ve en büyük düğüm derecesi olan boyanmamış düğümlerdir. Düğüm derecesi en büyük olan boyanmamış düğüm birden fazla olduğundan, boyama işlemi için seçilecek olan düğüm, düğüm sırasına göre boyanmamış ilk düğümdür. Buna göre, ikinci adımda v_1 düğümü boyama işlemi için seçilmiştir. Çizelge 3.2'de verilen komşuluk matrisi kontrol edilmiş ve v_1 düğümünün komşuları tespit edilerek, v_1 düğümüne uygun renk verilmiştir. v_1 düğümü v_{11} düğümü ile komşu olmadığı için renk kümesinde bulunan ilk renk v_1 düğümüne de verilmiştir. Şekil 3.22'de ikinci adım sonunda boyanan graf ve Çizelge 3.5'te ise boyanabilecek düğümler ait düğüm dereceleri tablosu verilmiştir.



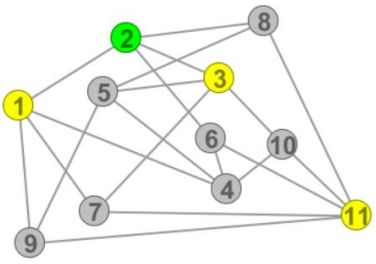
Şekil 3.22. LDO algoritması ikinci adım

Çizelge 3.5. Boyanabilecek düğümlere ait düğüm dereceleri tablosu – 2

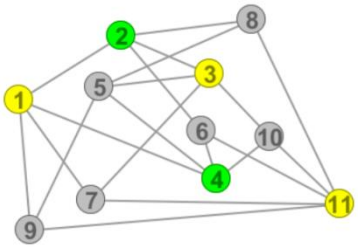
Düğümler (V)	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}
Düğüm Dereceleri (σ)	4	4	4	4	4	3	3	3	3	3	5



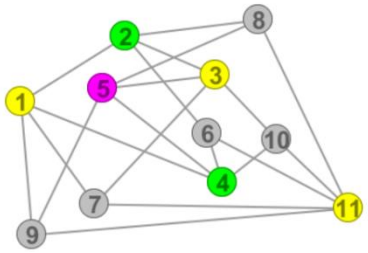
(a) LDO algoritması üçüncü adım



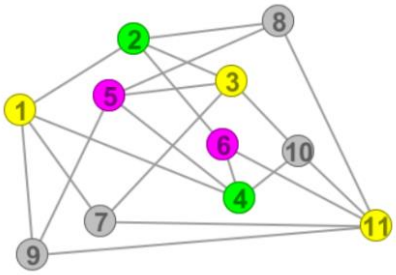
(b) LDO algoritması dördüncü adım



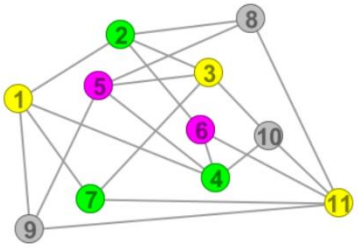
(c) LDO algoritması beşinci adım



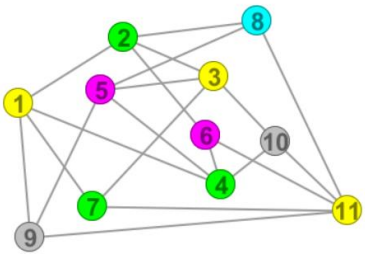
(d) LDO algoritması altıncı adım



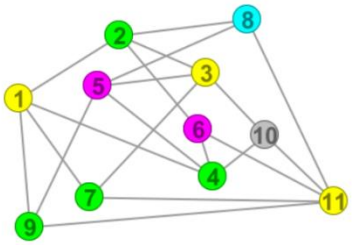
(e) LDO algoritması yedinci adım



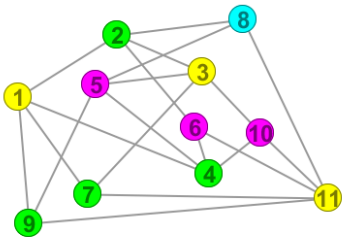
(f) LDO algoritması sekizinci adım



(g) LDO algoritması dokuzuncu adım



(h) LDO algoritması onuncu adım



(i) LDO algoritması onbirinci adım

Şekil 3.23. LDO algoritmasına göre düğümlerin boyanma sırası ve renkleri

Tüm düğümler boyanıncaya kadar LDO algoritmasının çalışma adımları tekrarlanmaktadır. Şekil 3.23'de algoritmanın çalışma adımlarına göre sırasıyla boyanan düğümler ve bu düğümlere verilen renkler gösterilmiştir. LDO algoritması düğümleri boyarken, düğüm derecelerini dikkate aldığı için FF algoritmasına göre daha iyi bir strateji sunmaktadır.

3.3.3. Welsh ve Powell Algoritması (WP)

Welsh ve Powell (1967) graf boyama problemi için açgözlü sezgisel bir algoritma önermişlerdir. Bu algoritmanın en önemli noktası, boyama işleminin düğüm derecelerine göre yapılmasıdır. Öncelikle düğümler, düğüm derecelerine göre büyükten küçüğe doğru sıralanmaktadır. Tüm düğümler boyanıncaya kadar, tanımlanan her yeni renk bu diziden boyanmamış en büyük dereceli düğüme verilmektedir. Düğüm kümesi $V = \{v_1, v_2, \ldots\ldots, v_n\}$, renk kümesi $R = \{r_1, r_2, \ldots\ldots, r_k\}$ olarak tanımlanan bir G grafı için WP algoritmasının çalışma adımları;

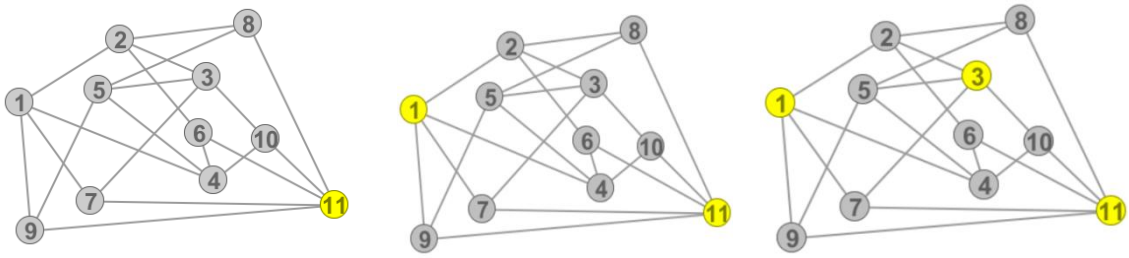
- **Adım 1:** Her düğüm için düğüm derecesi hesaplanır ve $Deg(v_i)$ dizisinde tutulur. ($i = 1,2, \ldots, n$).
- **Adım 2:** $Deg(v_i)$ dizisinden düğüm derecesi en büyük olan boyanmamış düğüm için renk kümesindeki ilk renk aktif renk olarak seçilir.
- **Adım 3:** Seçilen düğüm aktif olan renk ile boyanır. Daha sonra boyanmış düğüme komşu olmayan düğümler komşuluk matrisinden tespit edilir $V' = \{v'_1, v'_2, \ldots\ldots, v'_n\}$. Burada V', boyanmış düğüme komşu olmayan düğümler kümesidir.

  V' kümesindeki düğümlerden, düğüm derecesi en büyük olan düğüm, aktif renk ile boyanır ve bu düğüme komşu olan düğümler V' kümesinden silinir. V' kümesindeki tüm düğümler boyanıncaya kadar bu adım tekrarlanır.
- **Adım 4:** Boyanmamış düğüm varsa renk kümesinde bulunan bir sonraki renk aktif renk olarak seçilir ve ikinci adıma geri dönülür.

Welsh ve Powell algoritması Şekil 3.17'de verilen *myciel3.col* grafına uygulanırsa, boyama işlemi sırasında algoritmanın çalışma adımları şunlardır:

Adım 1: Çizelge 3.3'de *myciel3.col* grafındaki düğümlere ait düğüm dereceleri verilmiştir. Düğüm derecesi en büyük olan boyanmamış düğüm v_{11} düğümü

olduğundan, bu düğüm ilk boyama işlemi için seçilmiştir. v_{11} düğümü renk kümesinde bulunan aktif renk olan r_1 rengi ile boyanmıştır. WP algoritmasına göre aktif renk ile boyanabilecek düğümler kümesi olan V′ kümesi oluşturulmaktadır. Komşuluk matrisi incelendiğinde v_1, v_2, v_3, v_4, v_5 düğümlerinin v_{11} düğümü ile komşu olmadığı tespit edilmiştir. V′ kümesinde bulunan düğümlerden, düğüm derecesi en büyük olan düğüm boyama işlemi için seçilir. Ancak V′ kümesinde düğüm derecesi en büyük olan birden fazla düğüm varsa, düğüm numarasına göre ilk düğüm boyama işlemi için seçilir ve aktif renk ile boyanır. Daha sonra bu düğüm ve bu düğüme komşu olan düğümler V′ kümesinden silinir. Çünkü birbirine komşu olan düğümlerden sadece biri aktif renk ile boyanabilir. Çizelge 3.3'te verilen düğüm dereceleri tablosu incelendiğinde, V′ kümesinde bulunan düğümlerden v_1 düğümü aktif renk ile boyanacak düğüm olarak seçilir ve aktif renk ile boyanır. Daha sonra bu düğüme komşu olan v_2 ve v_4 düğümleri ile v_1 düğümü V′ kümesinden silinir. V′ kümesinde kalan v_3 ve v_5 düğümlerinden v_3 düğümü boyama işlemi için seçilir ve aktif renk ile boyanır. v_3 ve v_5 düğümleri komşu olduğundan, v_5 düğümü de V′ kümesinden silinir. V′ kümesinde boyanabilecek düğüm kalmadığı için renk kümesinde bulunan renklerden bir sonraki renk, aktif renk olarak seçilir. Bir sonraki adım için boyanmamış düğümlerden düğüm derecesi en büyük olan düğüm boyama işlemi için seçilir. Şekil 3.24'te birinci adım sonunda aktif renk ile boyanan düğümler verilmiştir.



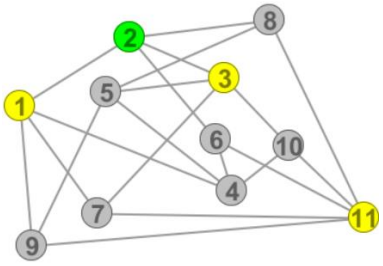
(a) Aktif renk ile boyanan ilk düğüm

(b) Aktif renk ile boyanan ikinci düğüm

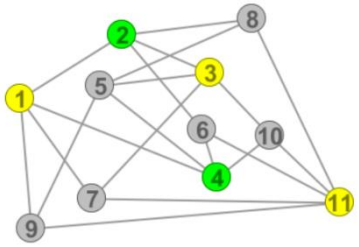
(c) Aktif renk ile boyanan üçüncü düğüm

Şekil 3.24. WP algoritması birinci adım

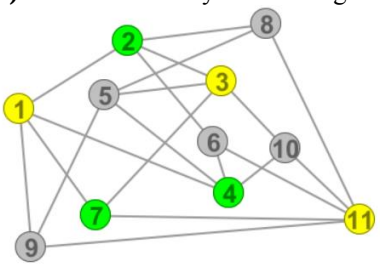
Adım 2: Birinci adımda yapılan işlemler tekrarlandığında aktif renk olan ikinci renk ile boyanabilecek düğümler belirlenmiş ve boyanmıştır. İkinci adım sonunda v_2, v_4, v_7, v_9 düğümleri ikinci renk ile boyanmıştır. Şekil 3.25'te ikinci adım sonunda boyanan düğümlerin düğüm sırası verilmiştir.



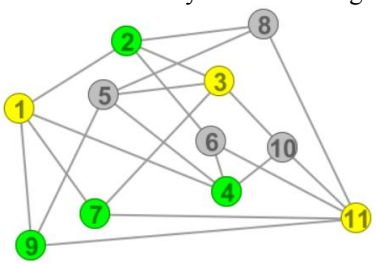
(a) Aktif renk ile boyanan ilk düğüm



(b) Aktif renk ile boyanan ikinci düğüm



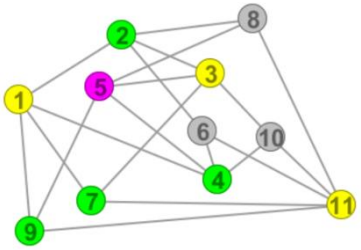
(c) Aktif renk ile boyanan üçüncü düğüm



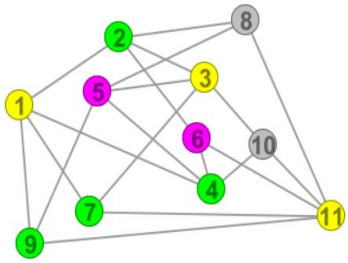
(d) Aktif renk ile boyanan dördüncü düğüm

Şekil 3.25. WP algoritması ikinci adım

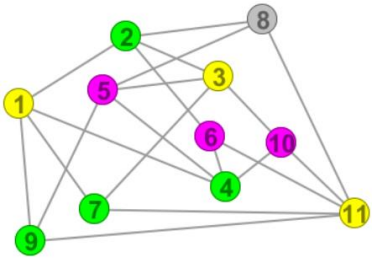
Adım 3: Şekil 3.26'da algoritmanın çalışma adımlarına göre sırasıyla v_5, v_6, v_{10} düğümleri boyanmıştır.



(a) Aktif renk ile boyanan ilk düğüm



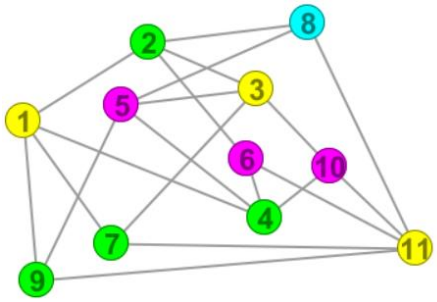
(b) Aktif renk ile boyanan ikinci düğüm



(c) Aktif renk ile boyanan üçüncü düğüm

Şekil 3.26. WP algoritması üçüncü adım

Adım 4: Son adımda ise v_8 düğümü WP algoritmasına göre boyanmıştır. Şekil 3.27'de WP algoritmasına göre boyanmış olan tüm graf verilmiştir.



Şekil 3.27. WP algoritması son adım

3.3.4. Incidence Degree Ordering Algoritması (IDO)

Düğüm kümesi $V = \{v_1, v_2, \ldots\ldots, v_n\}$, renk kümesi $R = \{r_1, r_2, \ldots\ldots, r_k\}$ olarak tanımlanan bir G grafı için IDO algoritmasının çalışma adımları;

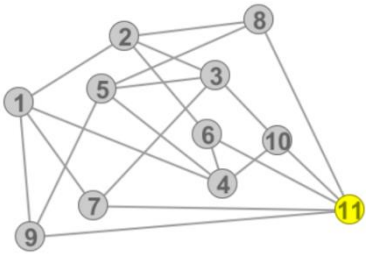
- **Adım 1:** Her düğüm için düğüm derecesi hesaplanır $Deg(v_i)$ dizisinde tutulur. $(i = 1,2, \ldots, n)$. Başlangıçta renk kümesinde sadece bir renk vardır.
- **Adım 2:** Düğüm derecesi en büyük olan düğüm, ilk renklendirme işlemi için seçilir. Seçilen düğüme ilk renk verilir.
- **Adım 3:** Boyanmamış düğümlerden, boyanan komşu sayısı en fazla olan düğüm, bir sonraki renklendirme işlemi için seçilir. Bu şartı sağlayan birden fazla düğüm varsa, bu düğümlerden düğüm derecesi en büyük olan düğüm bir sonraki boyama işlemi için seçilir.
- **Adım 4:** Seçilen düğüm renk kümesinde bulunan renkler ile boyanmaya çalışılır. Eğer renk kümesinde bulunan renkler, seçilen düğümü boyamak için uygun değilse yeni bir renk tanımlanır. Tanımlanan yeni renk hem renk kümesine eklenir, hem de seçilen düğüme verilir.
- **Adım 5:** Tüm düğümler boyanıncaya kadar üçüncü adıma geri dönülür.

IDO algoritması Şekil 3.17'de verilen *myciel3.col* grafına uygulanırsa, boyama işlemi sırasında algoritmanın çalışma adımları şunlardır:

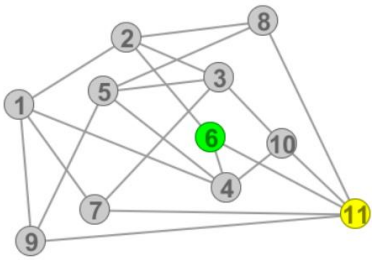
Çizelge 3.3'de *myciel3.col* grafındaki düğümlere ait düğüm dereceleri verilmiştir. Düğüm derecesi en büyük olan boyanmamış düğüm v_{11} düğümü olduğundan, bu düğüm ilk boyama işlemi için seçilmiştir. Renk kümesinde başlangıç aşamasından sadece bir renk vardır. v_{11} düğümü renk kümesinde bulunan r_1 rengi ile boyanmıştır. Şekil 3.28(a)'da birinci adım sonunda boyanan graf verilmiştir. Boyanmış olan v_{11} düğümünün komşuları komşuluk matrisinden bulunmuş ve bu düğümler için boyanmış komşu sayıları hesaplanmıştır. Çizelge 3.6'da IDO algoritmasına göre seçilecek olan düğümlerin boyanmış komşu sayıları verilmiştir.

Çizelge 3.6. Boyanmamış her düğümün boyanmış komşu sayısı ve düğüm dereceleri

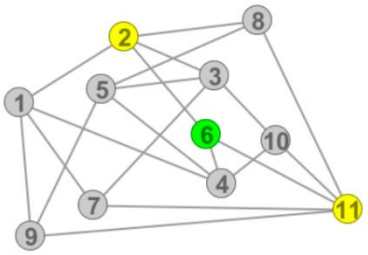
Düğümler (V)	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}
Düğüm Dereceleri (σ)	4	4	4	4	4	3	3	3	3	3	5
Boyanan komşu sayısı	0	0	0	0	0	1	1	1	1	1	



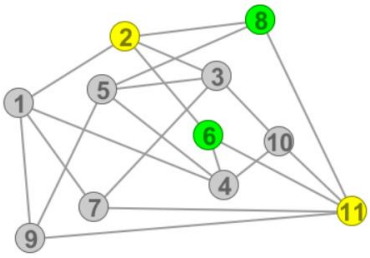
(a) IDO algoritması birinci adım



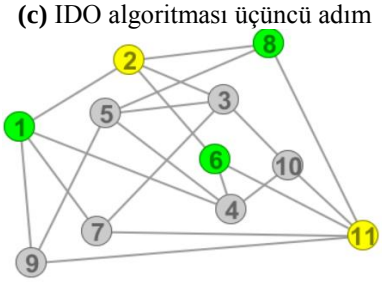
(b) IDO algoritması ikinci adım



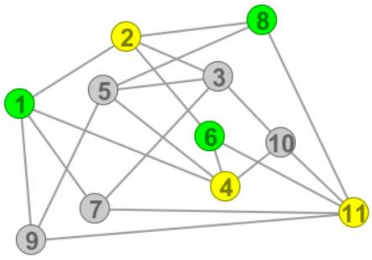
(c) IDO algoritması üçüncü adım



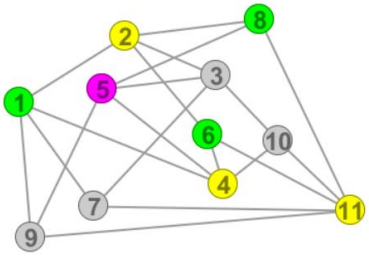
(d) IDO algoritması dördüncü adım



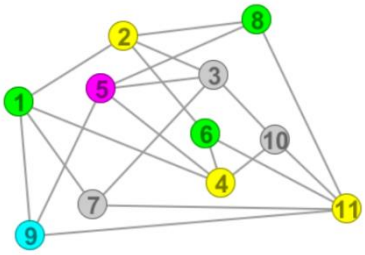
(e) IDO algoritması beşinci adım



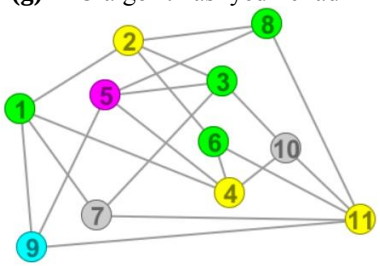
(f) IDO algoritması altıncı adım



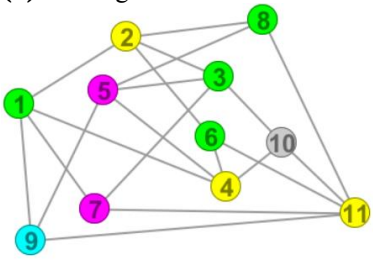
(g) IDO algoritması yedinci adım



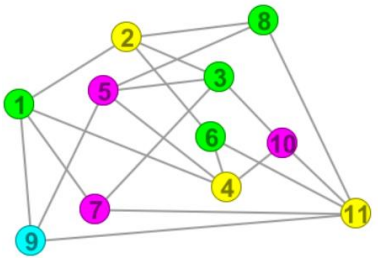
(h) IDO algoritması sekizinci adım



(i) IDO algoritması dokuzuncu adım



(j) IDO algoritması onuncu adım



(k) IDO algoritması onbirinci adım

Şekil 3.28. IDO algoritmasına göre düğümlerin boyanma sırası ve renkleri

Çizelge 3.6'da verilen bilgilere göre $v_6, v_7, v_8, v_9, v_{10}$ düğümlerinin birer tane boyanmış komşusu bulunmaktadır. Bu durumda bu düğümlerden düğüm derecesi en büyük olan düğüm boyama işlemi için seçilir. Bu şartı sağlayan birden fazla düğüm olması durumunda ise düğüm sırasına göre ilk düğüm boyanma işlemi için seçilir. Bu şartı sağlayan düğüm v_6 düğümü olduğu için bu düğüm boyama işlemi için seçilmiştir. v_6 düğümü ile v_{11} düğümü, komşu düğüm olduklarından dolayı v_6 düğümüne aynı renk verilemeyecektir. Bundan dolayı v_6 düğümüne renk kümesinde bulanan r_2 rengi verilmiştir. Şekil 3.28 (b)'de ikinci adım sonunda boyanan graf verilmiştir. Tüm düğümler boyanıncaya kadar IDO algoritmasının çalışma adımları tekrarlanmaktadır. Şekil 3.28'da algoritmanın çalışma adımlarına göre sırasıyla boyanan düğümler ve bu düğümlere verilen renkler gösterilmiştir.

3.3.5. Degree of Saturation Algoritması (DSATUR)

DSATUR algoritması Daniel Brélaz tarafından önerilmiştir. Bu algoritmaya göre bir graftaki düğümler boyanırken, düğümler dinamik bir şekilde boyanmaktadır. Bu algoritmada boyanacak ilk düğüm seçilirken, düğüm derecesi en büyük olan düğüm seçilmektedir. Sonraki aşamalarda ise kalan düğümler boyanırken *saturasyon* (saturation) denilen algoritmanın sürekli güncellenen yapısı kullanılmaktadır. *Saturasyon*'da bir düğümün farklı renk ile boyanmış komşularının sayısı tutulmaktadır. Tüm düğümler boyanıncaya kadar, *Saturasyon* değeri en büyük olan boyanmamış düğüm, boyanma işlemi için seçilmektedir. Düğüm kümesi $V = \{v_1, v_2, \ldots\ldots, v_n\}$, renk kümesi $R = \{r_1, r_2, \ldots\ldots, r_k\}$ olarak tanımlanan bir G grafı için DSATUR algoritmasının çalışma adımları;

- **Adım 1:** Her düğüm için düğüm derecesi hesaplanır ve $Deg(v_i)$ dizisinde tutulur. $(i = 1,2, \ldots, n)$.
- **Adım 2:** Düğüm derecesi en büyük olan düğüm ilk renklendirme işlemi için seçilir. Seçilen düğüme ilk renk verilir.
- **Adım 3:** Boyanmamış düğümlerden, farklı renk ile boyanmış komşu sayısı en çok olan düğüm bir sonraki boyama işlemi için seçilir. Bu şartı sağlayan birden fazla düğüm varsa, bu düğümlerden düğüm derecesi en büyük olan düğüm bir sonraki boyama işlemi için seçilir. Eğer düğüm derecesi en büyük olan düğüm

sayısı da birden fazla ise bu durumda düğümler kümesindeki düğüm sırasına göre ilk sırada bulunan boyanmamış düğüm boyama işlemi için seçilir.

- **Adım 4:** Seçilen düğüm renk kümesinde bulunan renkler ile boyanmaya çalışılır. Eğer renk kümesinde bulunan renkler seçilen düğümü boyamak için uygun değilse (seçilen düğümün komşularının sahip olduğu renkler ise) yeni bir renk tanımlanır. Tanımlanan yeni renk hem renk kümesine eklenir, hem de seçilen düğüme verilir.
- **Adım 5:** Tüm düğümler boyanıncaya kadar üçüncü adıma geri dönülür.

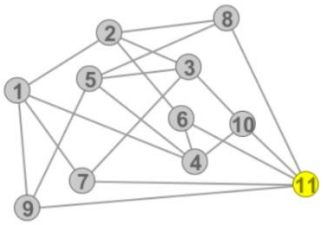
DSATUR algoritması Şekil 3.17'de verilen *myciel3.col* grafına uygulanırsa, boyama işlemi sırasında algoritmanın çalışma adımları şunlardır:

Çizelge 3.3'de *myciel3.col* grafındaki düğümlere ait düğüm dereceleri verilmiştir. Düğüm derecesi en büyük olan boyanmamış düğüm v_{11} düğümü olduğundan, v_{11} düğümü renk kümesindeki ilk renk olan r_1 rengi ile boyanmıştır. Şekil 3.29(a)'da birinci adım sonunda boyanan graf verilmiştir. Boyanmış olan v_{11} düğümünün komşuları komşuluk matrisinden bulunup, bu düğümler için farklı renk ile boyanmış komşu sayıları hesaplanmıştır.

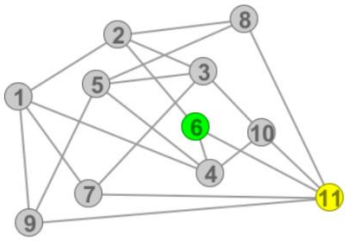
Çizelge 3.7. Boyanmamış her düğümün farklı renk ile boyanmış komşu sayısı ve düğüm dereceleri

Düğümler (V)	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}
Düğüm Dereceleri (σ)	4	4	4	4	4	3	3	3	3	3	5
Farklı renk ile boyanan komşu sayısı	0	0	0	0	0	1	1	1	1	1	

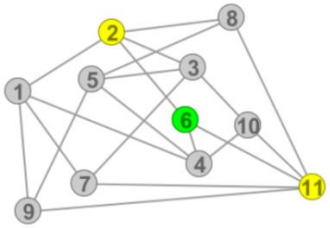
Çizelge 3.7'de DSATUR algoritmasına göre seçilecek olan düğümlerin farklı renk ile boyanmış komşu sayıları verilmiştir. $v_6, v_7, v_8, v_9, v_{10}$ düğümlerinin birer tane farklı renk ile boyanmış komşusu bulunmaktadır. Bu durumda bu düğümlerden düğüm derecesi en büyük olan düğüm boyama işlemi için seçilir. Bu şartı sağlayan birden fazla düğüm olması durumunda ise düğüm sırasına göre ilk düğüm boyama işlemi için seçilir. Bu şartı sağlayan düğüm v_6 düğümüdür ve v_6 düğümü, v_{11} düğümü ile komşu düğüm olduğundan dolayı v_6 düğümüne renk kümesinde bulanan r_2 rengi verilmiştir. Tüm düğümler boyanıncaya kadar DSATUR algoritmasının çalışma adımları tekrarlanmaktadır. Şekil 3.29'da algoritmanın çalışma adımlarına göre sırasıyla boyanan düğümler ve bu düğümlere verilen renkler gösterilmiştir.



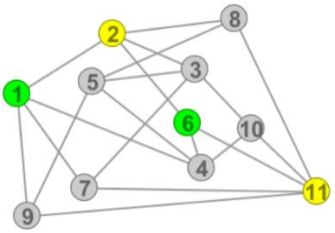
(a) DSATUR algoritması birinci adım



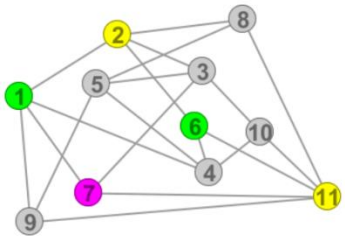
(b) DSATUR algoritması ikinci adım



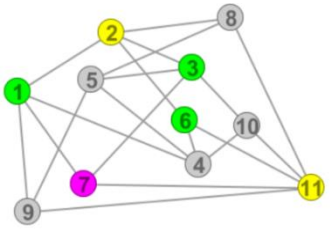
(c) DSATUR algoritması üçüncü adım



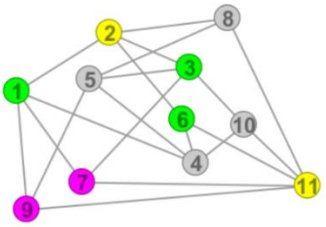
(d) DSATUR algoritması dördüncü adım



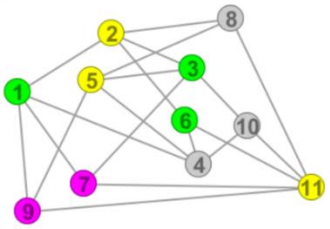
(e) DSATUR algoritması beşinci adım



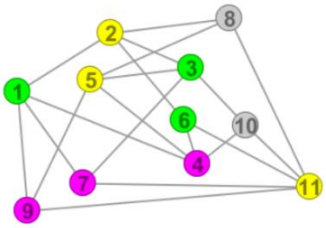
(f) DSATUR algoritması altıncı adım



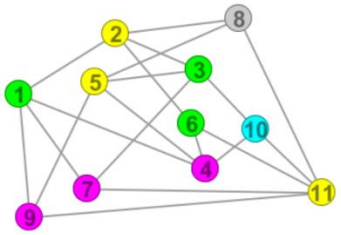
(g) DSATUR algoritması yedinci adım



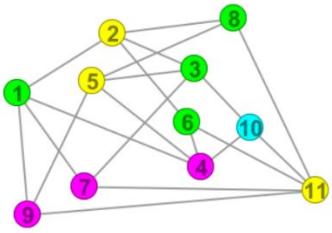
(h) DSATUR algoritması sekizinci adım



(i) DSATUR algoritması dokuzuncu adım



(j) DSATUR algoritması onuncu adım



(k) DSATUR algoritması onbirinci adım

Şekil 3.29. DSATUR algoritmasına göre düğümlerin boyanma sırası ve renkleri

3.3.6. Recursive Largest First Algoritması (RLF)

RLF algoritmasının en önemli özelliği düğümler boyanırken kullandığı rekürsif yapıdır (Leighton, 1979). Bu rekürsif yapı sayesinde minimum renk kullanılarak düğümler boyanmaya çalışılmaktadır. Düğüm kümesi $V = \{v_1, v_2, \ldots\ldots, v_n\}$, renk kümesi $R = \{r_1, r_2, \ldots\ldots, r_k\}$ olarak tanımlanan bir G grafı için RLF algoritmasının çalışma adımları:

- **Adım 1:** Her düğüm için düğüm derecesi hesaplanır ve $Deg(v_i)$ dizisinde tutulur. $(i = 1,2, \ldots, n)$. Düğüm derecesi en büyük olan düğüm ilk renklendirme işlemi için seçilir. Renk kümesindeki ilk renk aktif renk olarak seçilir.
- **Adım 2:** Seçilen düğüm aktif renk ile boyanırken, bu düğüme komşu olan düğümler aktif renk ile boyanamaz. Ancak bu düğüme komşu olmayan düğümler aktif renk ile boyanabilir. Bu işlem sırasında takip edilmesi gereken adımlar:
- V kümesinden boyama işlemi için seçilen v_{i} düğümüne komşu olan düğümler tespit edilir $U = \{u_1, u_2, \ldots\ldots, u_t\}$.
- v_{i} düğümüne komşu olmayan düğümler de tespit edilir ve V' kümesine konulur. V' kümesi boşalıncaya kadar, V' kümesindeki düğümler üzerinde gezilir. Bu işlem sırasında komşuluk matrisi üzerinde gezilerek, V' kümesinde bulunan düğümlerin her biri için U kümesinde bulunan düğümler ile olan komşulukları kontrol edilir. En fazla komşuluğu olan düğüm, aktif renk ile boyanır.
- Boyanan düğüme komşu olan düğümler V' kümesinden silinerek U kümesine eklenir.
- V' kümesi boşalıncaya ikinci adıma dönülür.
- **Adım 3:** Eğer boyanmamış düğüm varsa renk kümesinde bulunan renklerden bir sonraki renk aktif renk olarak seçilir. Eğer tüm düğümler boyanmış ise program sonlandırılır.
- **Adım 4:** Boyanmamış düğümlerden düğüm derecesi en büyük olan boyanmamış düğüm boyama işlemi için seçilir. Eğer bu şartı sağlayan birden fazla düğüm varsa bu düğümlerden düğümler kümesinde bulunan ilk sıradaki düğüm boyama işlemi için seçilir ve ikinci adıma geri dönülür.

RLF algoritması Şekil 3.17'de verilen *myciel3.col* grafına uygulanırsa, boyama işlemi sırasında algoritmanın çalışma adımları şöyledir:

Adım 1: Çizelge 3.3'te *myciel3.col* grafındaki düğümlere ait düğüm dereceleri verilmiştir. Düğüm derecesi en büyük olan boyanmamış düğüm v_{11} düğümü olduğundan, bu düğüm ilk boyama işlemi için seçilmiştir. v_{11} düğümü renk kümesinde bulunan aktif renk olan r_1 rengi ile boyanmıştır. Şekil 3.30(a)'da ilk boyama işlemi sonunda boyanan graf verilmiştir.

Çizelge 3.8. V′ ve U kümeleri

V′ kümesi	v_1	v_2	v_3	v_4	v_5
***U* kümesi**	v_6	v_7	v_8	v_9	v_{10}

Daha sonra RLF algoritmasının çalışma adımlarına göre aktif renk ile boyanabilecek düğümler ve boyanamayacak düğümler komşuluk matrisinden tespit edilmiştir. Çizelge 3.8'de V′ ve U kümeleri verilmiştir. v_{11} düğümüne komşu olmayan düğümlerin kümesi $\mathrm{V}' = \{v_1, v_2, v_3, v_4, v_5\}$ olarak verilmiştir. Ayrıca v_{11} düğümüne komşu olan düğümlerin kümesi olan U kümesi de tespit edilmiştir. U kümesi $v_6, v_7, v_8, v_9, v_{10}$ düğümlerinden oluşur. Aktif renk ile boyanacak düğümü tespit etmek için V′ kümesinde bulunan her düğüm için U kümesinde bulunan düğümler ile komşulukları incelenir. V′ kümesinde bulunan düğümlerden en fazla komşuluğa sahip olan düğüm, boyama işlemi için seçilir. Eğer bu şartı sağlayan birden fazla düğüm varsa, bu düğümlerden düğüm derecesi en büyük olan düğüm, boyama işlemi için seçilir. Çizelge 3.9 incelendiğinde V′ kümesinde bulunan düğümlerin hepsinin aynı sayıda komşuluğu bulunmaktadır. Bundan dolayı düğüm derecesi en büyük olan ilk düğüm v_1 düğümü boyama işlemi için seçilir ve aktif renk ile boyanır. Şekil 3.30(b)'de aktif renk ile boyanan ikinci düğüm verilmiştir.

Çizelge 3.9. Aktif renk ile boyanabilecek düğümler

V′ kümesi	v_1	v_2	v_3	v_4	v_5
Düğüm Dereceleri	4	4	4	4	4
***U* kümesi ile komşuluk sayısı**	2	2	2	2	2

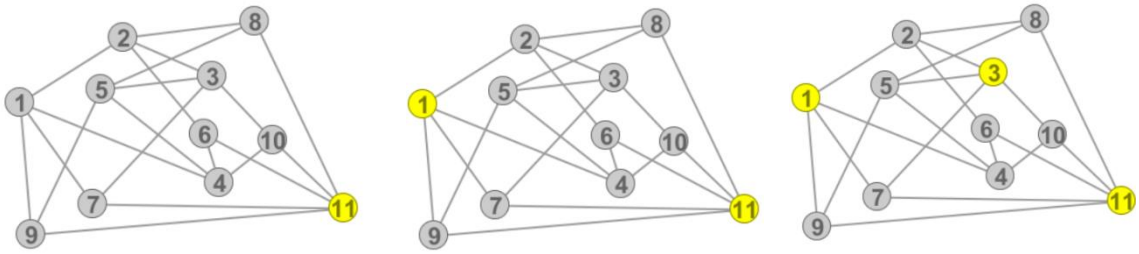
Komşu düğümler aynı rengi alamayacağı için v_1 düğümüne komşu olan düğümler olan v_2 ve v_4 düğümleri V′ kümesinden silinerek U kümesine eklenir. Çizelge

3.10'da güncel V′ ve U kümelerinde bulunan düğümler verilmiştir. V′ kümesinde bulunan düğümlerden U kümesinde bulunan düğümler ile en fazla komşuluğa sahip olan düğüm tespit edilir. v_3 ve v_5 düğümlerinin komşuluk sayıları ve düğüm dereceleri eşit olduğu için düğümler kümesinde düğüm numarası küçük olan v_3 düğümü aktif renk ile boyanır.

Çizelge 3.10. V′ ve U kümeleri

V′ kümesi	v_3	v_5					
***U* kümesi**	v_2	v_4	v_6	v_7	v_8	v_9	v_{10}

v_3 ile v_5 düğümleri komşu düğüm olduğundan, v_5 düğümü V′ kümesinden silinir. Böylece V′ kümesinde boyanabilecek düğüm kalmadığından bir sonraki adıma geçilir. Şekil 3.30(c)'de aktif renk ile boyanan üçüncü düğüm verilmiştir. Bir sonraki aşama için aktif renk olarak r_2 rengi seçilir. Boyanmamış düğümlerden düğüm derecesi en büyük olan düğüm boyama işlemi için seçilir ve RFL algoritmasının adımları tekrarlanır.



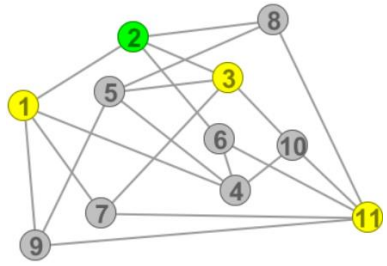
(a) Aktif renk ile boyanan ilk düğüm

(b) Aktif renk ile boyanan ikinci düğüm

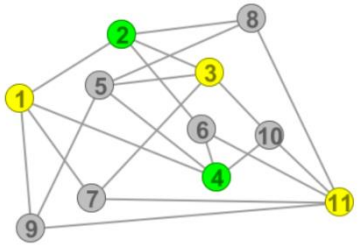
(c) Aktif renk ile boyanan son düğüm

Şekil 3.30. RLF algoritması birinci adım

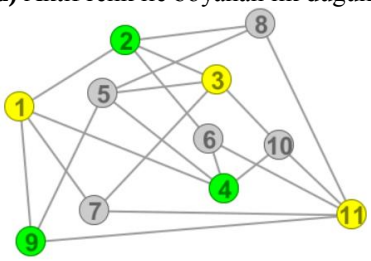
Adım 2: Birinci adımda yapılan işlemler tekrarlandığında aktif renk olan ikinci renk ile boyanabilecek düğümler belirlenmiş ve boyanmıştır. İkinci adım sonunda v_2, v_4, v_7, v_9 düğümleri ikinci renk ile boyanmıştır. Şekil 3.31'de ikinci adım sonunda boyanan düğümlerin düğüm sırası verilmiştir.



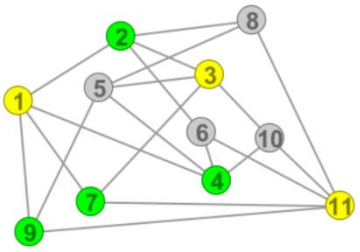
(a) Aktif renk ile boyanan ilk düğüm



(b) Aktif renk ile boyanan ikinci düğüm



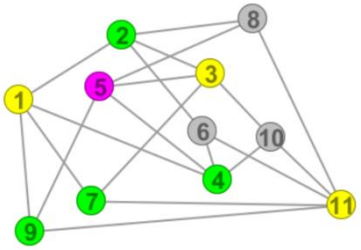
(c) Aktif renk ile boyanan üçüncü düğüm



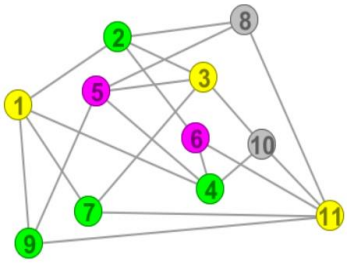
(d) Aktif renk ile boyanan dördüncü düğüm

Şekil 3.31. RLF algoritması ikinci adım

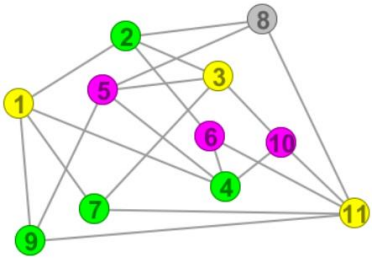
Adım 3: Şekil 3.32'de algoritmanın çalışma adımlarına göre sırasıyla v_5, v_6, v_{10} düğümleri boyanmıştır.



(a) Aktif renk ile boyanan ilk düğüm



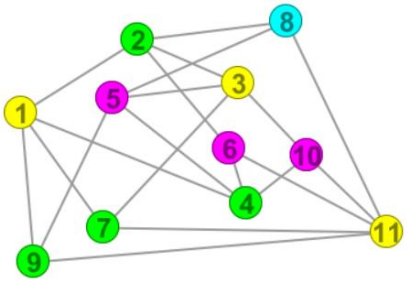
(b) Aktif renk ile boyanan ikinci düğüm



(c) Aktif renk ile boyanan üçüncü düğüm

Şekil 3.32. RLF algoritması üçüncü adım

Adım 4: Son adımda ise v_8 düğümü RLF algoritmasına göre boyanmıştır. Şekil 3.33'te RLF algoritmasına göre boyanmış olan tüm graf verilmiştir.



Şekil 3.33. RLF algoritması son adım

3.4. Metasezgisel Yöntemler

Bir probleme optimizasyon problemi olarak yaklaşıldığında, bundan çıkarılacak sonuç bu problem için eldeki verilerden yola çıkarak mevcut çözüm uzayında (bütün çözümler arasında) en iyi çözüme ulaşabilmektir. Belirli kısıtları sağlamak suretiyle, eldeki verilerin rehberliğinde bilinmeyen değerlerin bulunmasına giden yol olarak tanımlanabilen herhangi bir problem, optimizasyon problemi olarak tanımlanabilir (Murty, 2003).

Metasezgisel algoritmalar, hedeflenen amaca ulaşmak için doğadaki hareketlerden esinlenerek oluşturulan algoritmalardır. Metasezgisel algoritmalar arama uzayındaki kesin çözüme ulaşmayı garanti etmezler. Ancak çözüm uzayındaki tüm çözümleri rehber edindiği için kesin çözüme yakın bir çözüme ulaşmayı garanti etmektedirler (Karaboğa, 2011).

Temeli biyolojiye dayanan Genetik Algoritma (GA), genler arasındaki bilgi paylaşımı sonucunda yeni nesillerin oluşmasını sağlamaktadır. Genlerin bir araya gelmesi ile biyolojideki karşılığı olan kromozomlar oluşur. GA'da her bir kromozom bir bireyi temsil etmektedir. Bireyler ise popülasyon dediğimiz bireylerin bir araya geldiği topluluğu ifade etmektedir. Popülasyon içindeki bireylerin genleri arasındaki bilgi paylaşımı sayesinde yeni nesiller oluşmaktadır (Angeline, 1995). Genetik algoritma, çeşitlilik sağlayarak bilinmeyen çözümlere ulaşılmasına katkıda bulunmaktadır.

Sürü zekâsına dayalı algoritmalar ise, grup içinde rehberlik eden bireyler sayesinde, toplu hareket ederek hedefe daha kolay ulaşabilmektedirler. Popülasyon içindeki bireyler, popülasyon içindeki en iyi bireyin sahip olduğu bilgiyi, kendi tecrübesi ile harmanlayarak ileride oluşabilecek problemlerin çözümü için kendisine rehberlik edecek bir araç olarak kullanmaktadırlar (Akyol ve Alataş, 2012). Örneğin; sürü içindeki bir birey, iyi bir besin kaynağı bulduğunda, bu bilgiyi diğer bireyler ile paylaşması sayesinde sürü içindeki diğer bireylerin de bu besin kaynağına hareket etmelerini sağlamış olmaktadır.

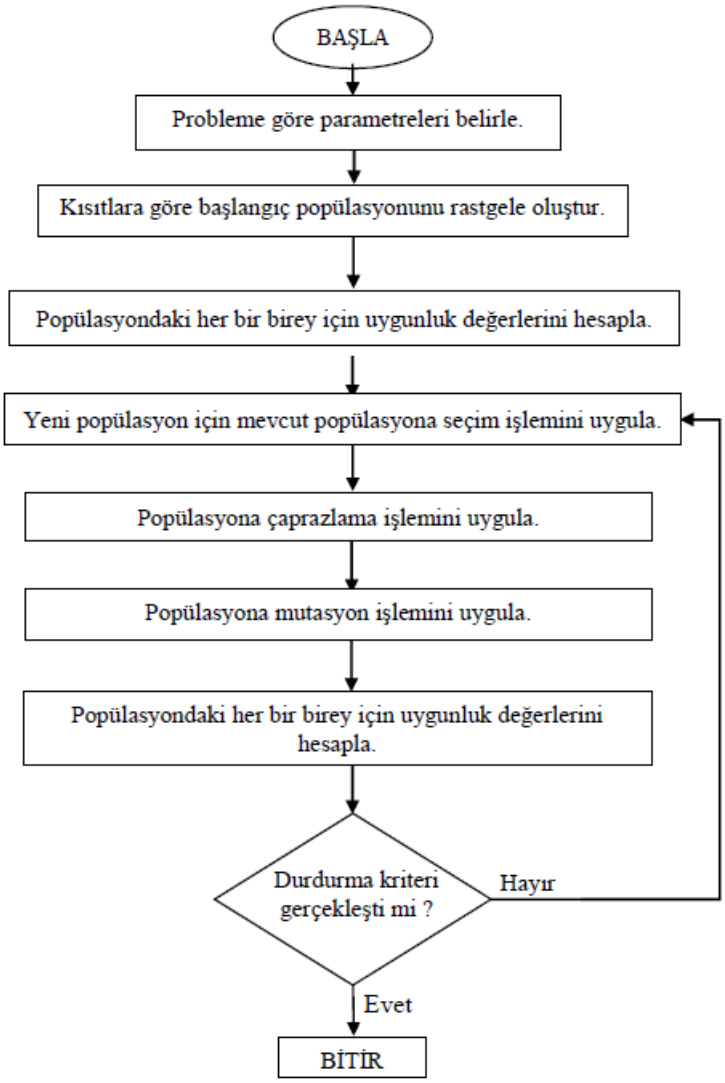
Metasezgisel yöntemler; biyoloji, kimya ve fizik temelli, sürü zekâsına dayalı, sosyal davranışlara dayanan ya da müzik tabanlı olabilirler. Bununla birlikte bu algoritmalardan bazılarının birlikte kullanılması ile oluşturulmuş hibrid algoritmalar da vardır (Alataş, 2007; Akyol ve Alataş, 2012). Ancak hepsinin genel amacı, eldeki mevcut bilgileri en iyi şekilde kullanarak, en iyi çözümlere ulaşabilmektir.

3.4.1. Genetik Algoritma (GA)

Genetik algoritma, ilk olarak John Holland (1975) tarafından uygulanmıştır (Holland, 1975). Daha sonra öğrencisi David Goldberg, GA'yı gaz boru hattının kontrolünü içeren bir problemin üzerine uygulamıştır (Golberg, 1989). Genetik algoritma hem sürekli hem de ayrık problemlerin çözümünde kullanılabilmektedir. Karmaşık amaç fonksiyonu parametrelerini, yerel minimum ya da maksimuma takılmadan optimize edebilmesi gibi avantajları vardır (Çunkaş, 2006).

Genetik algoritmada geniş arama algoritmalarının aksine en iyiyi seçmek için tüm farklı durumlar üretilmez. Bu özelliğinden dolayı en iyi çözüme ulaşmayı garanti etmez. Genetik algoritma, mevcut koşullara uyum sağlayan bir algoritmadır. Yani, bir konuda hiç bilgisi olmamasına rağmen, problemi ve bilgiyi öğrenme ve toplama yeteneğine sahiptir (Çunkaş, 2006).

Genetik algoritma en iyi çözüme ulaşmak için arama uzayında birçok farklı noktadan çözüm aramaya başlar. Bundan dolayı başlangıç popülasyonu oldukça önemlidir. GA'da başlangıç popülasyonu genellikle rastgele oluşturulmaktadır (Reeves ve Rowe, 2002; Kaya, 2012). Başlangıç popülasyonu problemin kısıtlarına göre oluşturulduktan sonra, her bireyin uygunluk değeri amaç fonksiyonuna göre hesaplanır. Daha sonra genetik algoritmanın seçim yöntemleri kullanılır. Seçim yöntemleri sayesinde popülasyon içindeki bazı bireyler seçilerek, bunlar yeni nesile (popülasyona) aktarılır. Oluşan yeni popülasyon için genetik algoritmanın evrimsel adımları uygulanmaktadır. Seçim işleminden sonra bireyler çaprazlama işlemine tabi tutulmaktadır. Çaprazlama işleminden sonra bireylere mutasyon işlemi uygulanır. Bu işlemlerden sonra oluşan yeni popülasyon için uygunluk değerleri yeniden hesaplanır. Uygunluk değerleri hesaplandıktan sonra seçim yöntemleri tekrar kullanılarak, bir sonraki nesil için bireyler arasından tekrardan seçim yapılır ve GA adımları tekrar uygulanır (Reeves ve Rowe, 2002). GA süreçleri, durdurma şartları sağlanıncaya kadar tekrarlanır. Şekil 3.34'de GA'ya ait temel akış diyagramı verilmiştir. GA'da problemin yapısına uygun olacak şekilde genellikle bu adımlar kullanılmaktadır.



Şekil 3.34. Genetik algoritma akış diyagramı

3.4.2. Kurbağa Sıçrama Algoritması (KSA)

Kurbağa sıçrama algoritması (KSA) (SFLA: Suffled Frog Leaping Algorithm) ilk olarak su dağıtım şebekesi probleminin çözümünde kullanılmış olan memetik tabanlı bir optimizasyon algoritmasıdır (Eusuff ve Lansey, 2003). Genetik biliminde gen kavramı, nesiller arasında bilgi taşıma görevi yapmaktadır. Sosyal yaşamdaki mem kavramı, işte bu gen kavramına karşılık olarak gelmektedir. Nasıl ki kalıtsal özellikler genler ile nesiller arasında aktarılıyorsa, bireylerin sosyal yaşamdan edindiği tecrübe ve davranışlar da, mem dediğimiz kavram aracılığıyla diğer bireylere aktarılabilmektedir (Karakoyun, 2015). Memler, bireylerin birbirini taklit etmesi yolu ile diğer bireylere aktarılmaktadır. Popülasyon içindeki bireyler mem bilgisini birbirleriyle paylaşırken,

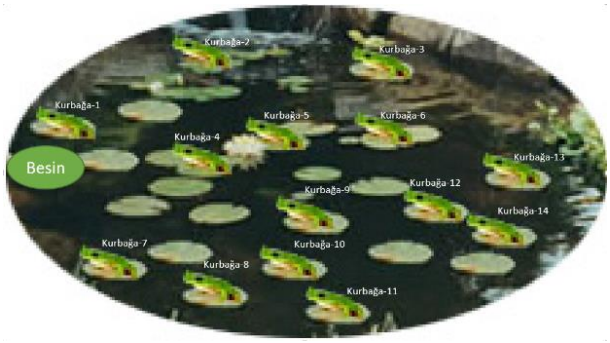
mem (bilgi) kalitesi yüksek olan bireyler popülasyondaki diğer bireyleri yönlendirmektedir. KSA'da amaç; kurbağa topluluğu içindeki kurbağaların kaliteli besin kaynağına, en az hareketle ulaşmasını sağlamaktır (Eusuff ve ark., 2006).

3.5. Graf Boyama Problemi için Önerilen Yöntem

3.5.1. Kurbağa Sıçrama Algoritması (KSA)

Kurbağa sıçrama algoritması (KSA), ilk olarak su dağıtım şebekesi probleminin çözümünde kullanılmıştır (Eusuff ve Lansey, 2003). Eusuff ve Lansey (2003) yapmış oldukları çalışmada kurbağaların doğadaki hareketlerinden esinlenerek memetik tabanlı yeni bir yaklaşım önermişlerdir. Kurbağa sıçrama algoritması doğal memetiklerden esinlenen popülasyon tabanlı bir metasezgisel algoritmadır. Algoritma, memetik evrimleri kullanarak yerel arama uzayında bir bireyden diğer bir bireye bilgi geçişini sağlamaktadır. KSA'nın yerel arama yapısı PSO algoritmasına benzerlik gösterir. Eusuff ve ark. (2006) yaptıkları çalışmada KSA'da bulunan karıştırma (shuffling) yapısı sayesinde yerel aramada bireyler arasında gerçekleşen bilgi değişimi ile genel en iyiye ulaşmayı sağladığını belirtmişlerdir (Eusuff ve ark., 2006).

Her metasezgisel algoritmada olduğu gibi KSA'da da amaç iyi çözümlere ulaşabilmektir. Kurbağaların yaşam alanları; genellikle göl, dere ve gölet gibi su kaynaklarının olduğu yerlerdir (Eusuff ve ark., 2006). Şekil 3.35'de kurbağaların örnek bir yaşam alanı verilmiştir. Şekil 3.35'de de görüldüğü gibi kurbağalar bulundukları ortamda farklı noktalarda konumlanmaktadırlar. Bu kurbağaların, her biri "**memetik**" olarak nitelendirilen besin kaynağı bilgisine sahiptir. Kurbağalar birbiriyle iletişim kurarak bu bilgiyi birbirleri ile paylaşırlar. Böylece kurbağalar hangi besin kaynağı daha kaliteli ise o noktaya doğru hareket ederler. Şekil 3.35'de verilen kurbağalardan birinci kurbağanın sahip olduğu mem bilgisinin daha kaliteli olduğunu düşünürsek diğer kurbağaların o kurbağanın bulunduğu konuma doğru hareket etmesi gerekir.



Şekil 3.35. Kurbağaların doğal yaşam alanı

Eusuff ve Lansey (2003) yaptıkları çalışmada, kurbağaların besin kaynağına ulaşmak için yaptıkları hareketleri modelleyerek bir algoritma önermişlerdir. Kurbağalar en az hareketle, besin kaynağının en çok olduğu yere ulaşmak isterler (Eusuff ve Lansey, 2003; Karakoyun, 2015). Popülasyon tabanlı algoritmalarda olduğu gibi KSA'da da her bir kurbağa bir bireyi temsil eder. Yani her bir kurbağa bir çözüme karşılık gelmektedir. Bir kurbağanın sahip olduğu bilgi, problemin yapısına uygunluk göstermek durumundadır. Bundan dolayı problemin sahip olduğu özelliklere göre bir kurbağa çözümü oluşturulur. Her bir kurbağa bir çözümü temsil ettiğinden, popülasyonda kaç kurbağa (birey) varsa o kadar çözüm var demektir. Ancak KSA'da başlangıç popülasyonu problemin yapısına uygun olarak rastgele bir şekilde oluşturulur. Bu rastgele yapısından dolayı her kurbağanın sahip olduğu bilgi doğal olarak birbirinden farklılık gösterir. Başlangıç popülasyonundaki bu çeşitlilik her bir kurbağanın farklı bir konuma yerleşmesini ve çevresindeki besin kaynaklarını genel bir uzayda keşfetmesi açısından önemlidir (Eusuff ve ark., 2006).

Metasezgisel algoritmalarda başlangıç popülasyonu oluşturulduktan sonra her bir bireyin sahip olduğu çözümün kalitesini tespit etmek için maliyet hesabının yapılması gerekmektedir. Bundan dolayı KSA'da da her bir kurbağa için maliyet hesabı yapılır. Böylece her bir kurbağanın sahip olduğu çözümün kalitesi tespit edilmiş olur. KSA'da popülasyon "**mempleks"** olarak adlandırılan gruplardan oluşur. Popülasyondaki toplam kurbağa sayısı mempleks grup sayısı ile her grup içinde bulunan kurbağa sayısının çarpımı olarak hesaplanır. Şekil 3.36'da hem popülasyon hem de mempleks yapısı gösterilmiştir. KSA'da memetik evrim mempleks üzerinde gerçekleştiğinden dolayı, oluşturulan popülasyon parametre aşamasında belirlenen gruplara dağıtılmaktadır. Uygunluk değerleri hesaplanmış olan bireyler eşitlik (3.2)'de verilen denkleme göre memplekslere dağıtılmaktadır. Böylece her memplekste hem iyi

ve hem de kötü çözümlerin eklenmesi sağlanmış olur. Ancak eğer popülasyon mempleklere rastgele olarak dağıtılsaydı bu durumda iyi çözümler bir tarafta, kötü çözümler bir tarafta kümelenebilir ve yerel optimum değere takılma söz konusu olabilirdi. Memetik evrim aşamasında oluşturulan her bir mempleks grubu, diğer mempleks gruplarından bağımsız olarak kendi grupları içerisinde birbirlerine bilgi aktarımı gerçekleştirirler. Her mempleks grubu içerisinde belirlenen iterasyon sayısı kadar yapılan çevrim sayesinde, her çevrimde alt grup içindeki en kötü çözüme sahip kurbağanın daha iyi çözümlere gitmesi için sahip olduğu mem bilgisi güncellenir. Birbirinden bağımsız mempleks gruplarının olması ve her grup içinde hem iyi hem de kötü çözüme sahip kurbağaların olması kaliteli çözümlerin oluşmasını sağlamaktadır. (Eusuff ve ark., 2006).

Memetik evrim aşamasının her iterasyonunda, mempleks içindeki belirli bir sayıda kurbağa memetik evrime dâhil edilir. Kurbağalardan hangilerinin memetik evrim aşamasında seçileceğine karar vermek için üçgensel olasılık dağılımına göre her bir kurbağaya, sahip olduğu bilginin kalitesine göre bir seçilme olasılığı verilmektedir. Daha sonra belirlenen sayıda kurbağa rulet tekerleği mantığına göre alt gruba seçilmektedir. Alt grup için seçilecek bireyler belirlenirken üçgensel olasılık dağılımı ile rulet tekerleğinin kullanılmasının sebebi; hem kötü çözümlere hem de iyi çözümlere seçilme şansı vermek içindir. Çünkü bazen kötü çözümlerdeki mem bilgisi bizi daha iyi çözümlere götürebilir. Şayet hep iyi çözümler üzerinde işlem yapılsaydı çözüm uzayı daraltılmış ve bunun sonucunda da yerel bir çözüme takılma ihtimali artmış olurdu (Eusuff ve ark., 2006).

Memetik evrim aşaması her bir mempleks grubu için tamamlandıktan sonra gruplara ayrılan kurbağalar tekrar tek bir popülasyon altında toplanır. Her bir kurbağanın uygunluk değeri tekrar hesaplanır ve popülasyon, uygunluk değeri en iyi kurbağadan en kötü kurbağaya doğru sıralanır. Kurbağaların grup dışına çıkarılarak tek bir çözüm uzayında toplanmasının sebebi; gruptaki kurbağaların genel uzayda bilgi alışverişi yapmalarını sağlamak içindir. Böylece kurbağalar tek bir popülasyonda birleştirildikten sonra ilk iterasyon tamamlanmış olur. Sonraki iterasyon için kurbağalar tekrar mempleks gruplarına dağılacaktır. Bu şekilde ilk iterasyonda A grubunda olan bir kurbağa ikinci iterasyonda B grubuna dâhil olabilir. Çünkü birinci iterasyon sonunda her bir grup içinde bulunan kurbağaların çözüm kalitesi değişmekte, bir grupta bulunan iyi çözümlerin kötü gruplara dağılması ile birlikte, o grupların içinde bulunan kötü çözümlerin daha iyi çözümlere ulaşması sağlanabilmektedir (Eusuff ve ark., 2006).

KSA'da bilgi paylaşımı hem mempleks grubu içinde hem de iterasyon sonunda tüm kurbağalarla yapılır. Bu da hem yerel hem de genel aramanın avantajlarından faydalanıldığını gösterir (Jonnalagadda ve Mallesham, 2013). Eusuff ve ark. (2006) çalışmalarında kurbağa sıçrama algoritmasının adımlarını belirlemişlerdir (Eusuff ve ark., 2006). KSA'nın çalışma adımları:

Adım 1 (Parametre ve kısıt belirleme): Algoritmanın parametre ve kısıtlarının belirlendiği aşamadır. KSA'da kullanılan parametreler ve özellikleri aşağıda verilmiştir:

- $\boldsymbol{m}$: KSA'daki mempleks grup sayısı
- $\boldsymbol{n}$: Her bir mempleks grubu içinde bulunan kurbağa sayısı
- $\boldsymbol{d}$: Problem boyutu
- $\boldsymbol{F}$: Popülasyondaki toplam kurbağa sayısını temsil eder. Popülasyondaki toplam kurbağa sayısı mempleks grup sayısı ile her grup içinde bulunan kurbağa sayısının çarpımı ($F = m \times n$) olarak hesaplanır.
- $\boldsymbol{q}$: Memetik evrim aşamasında alt grup için seçilecek birey sayısını temsil eder. Memetik evrim aşamasının her iterasyonunda alt grup içinde bulunan en kötü kurbağa değişime uğrar.
- $\boldsymbol{S}$: Kurbağa için adım büyüklüğüdür. Pozitif ya da negatif yönde ilerleme olabilir.
- $\boldsymbol{S_{max}}$: Memetik evrim aşamasında en kötü kurbağanın sahip olduğu bilgi güncellenirken, bu kurbağada meydana gelecek değişimin ne kadar olacağının belirlendiği maksimum adım büyüklüğüdür.
- $\boldsymbol{N}$: Memetik evrim aşamasında her mempleks grubu için memetik evrim döngü sayısıdır.
- $\boldsymbol{I}$: Kurbağa sıçrama algoritmasında algoritmanın sonlandırma kriteri olan iterasyon sayısıdır.

Adım 2 (Popülasyon oluşturma): Problemde belirlenen kısıtlara bağlı olarak F adet kurbağa rastgele oluşturulur. Popülasyondaki her bir kurbağa d adet karar değişkenine bağlı olarak oluşturulur. Burada d problemin boyutunu temsil etmektedir. Örneğin graflarda bir problemin boyutu düğüm sayısına göre belirlenmektedir. Oluşturulan her bir kurbağa çözüm uzayında $U = \{U_1, U_2, U_3, \dots \dots, U_i\}$ olarak gösterilir, $(i = 1,2, \dots, F)$.

Adım 3 (Uygunluk değeri hesaplama): Her bir kurbağa için maliyet (uygunluk) değeri hesaplanır ve $f = \{f_1, f_2, f_3 \dots, f_i\}$ kümesinde tutulur. Maliyet (fitness) değeri problemin yapısına bağlı olarak hangi bireyin en iyi birey olduğunun bilinmesi için önemlidir.

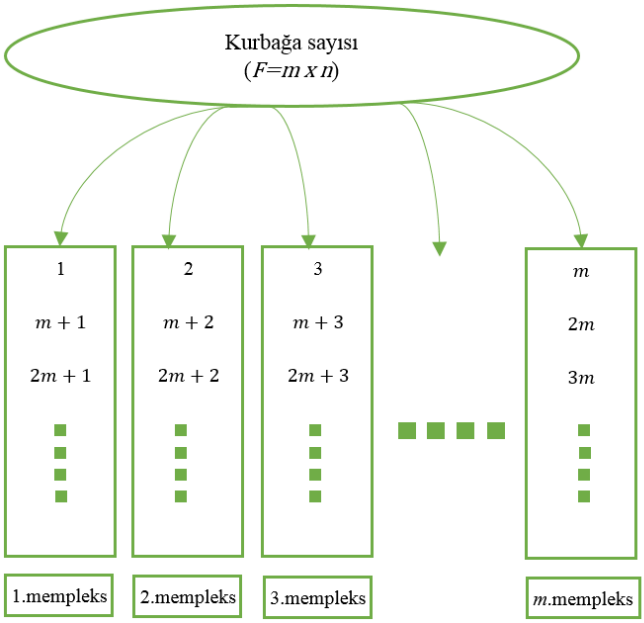
Uygunluk değerleri hesaplanan kurbağalar, en iyi uygunluk değerine sahip kurbağadan en kötü uygunluk değerine sahip kurbağaya doğru sıralanır. Uygunluk değerlerine göre sıralanan kurbağalar X kümesine konulur ve $X = \{P_1, P_2, P_3, \dots, P_i\}$ olarak gösterilir. Yani X kümesinde bulunan P_1 kurbağası popülasyondaki en iyi maliyet değerine sahip olan kurbağadır. P_i ise popülasyondaki en kötü kurbağayı temsil eder. KSA'da genel en iyi kurbağa P_x olarak gösterilir. P_x tüm popülasyondaki en iyi uygunluk değerine sahip olan kurbağadır.

Adım 4 (Mempleks dağılımı): Kurbağalar uygunluk değerlerine göre sıralandıktan sonra parametre aşamasında belirlenen m ve n değerlerine göre mempleks gruplarına dağıtılırlar. m mempleks sayısını, n ise her bir memplekste bulunan kurbağa sayısıdır. Popülasyondaki kurbağalar m adet mempleks grubuna dağıtılırken, bu dağıtma işlemi belli bir kural çerçevesinde gerçekleştirilmektedir. Örneğin 3 adet mempleks ve 9 adet kurbağamız varsa dağıtma işlemi şu şekilde olur;

- Uygunluk değerlerine göre sıralanan ve X kümesinde bulunan kurbağalardan en iyi birinci kurbağa olan P_1 kurbağası ilk mempleks grubuna dâhil edilir. İkinci en iyi kurbağa olan P_2 kurbağası ikinci mempleks grubuna, üçüncü en iyi kurbağa olan P_3 kurbağası üçüncü mempleks grubuna dâhil edilir. Bu kural çerçevesinde tüm kurbağalar mempleks gruplarına dağıtılmaktadır.
- Sonuç olarak ilk mempleks grubunda P_1, P_4 ve P_7 kurbağaları, ikinci mempleks grubunda P_2, P_5 ve P_8 kurbağaları, son olarak üçüncü grupta ise P_3, P_6 ve P_9 kurbağaları bulunur. Yukarıda KSA'ya göre popülasyondaki kurbağaların mempleks gruplarına dağıtılması işlemi için örnek bir dağıtma işlemiş verilmiştir. Bu örnekte de görüldüğü gibi kurbağalar belirli bir kurala göre gruplara dağıtılmaktadır. Kurbağalar mempleks gruplarına eşitlik (3.2)'de verilen kurala göre dağıtılmaktadır.

$$Y_k = \{ X(j)_k | X(j)_k = X(k + m * (j - 1)\} \quad (3.2)$$

Eşitlik (3.2)'de $(k = 1, \dots, m)$'ye kadar ve $(j = 1, \dots, n)$'ye kadar verilmiştir. Böylece adil bir dağılıma göre kurbağalar mempleklere dağıtılmış olmaktadır. Şekil 3.36'da popülasyondaki kurbağaların mempleklere dağıtılması işlemi verilmiştir.



Şekil 3.36. Kurbağaların mempleks gruplarına dağıtılması işlemi

Adım 5 (Memetik evrim): Bir önceki aşamada eşitlik (3.2)'e göre kurbağaların mempleklere dağıtılma işlemi gerçekleşmiştir. Bu aşamada ise, parametrelerin belirlenmesi aşamasında belirlenmiş olan N adet iterasyon kadar, her mempleks grubu için memetik evrim döngüsü gerçekleşmektedir. KSA'daki en önemli aşamalardan biri memetik evrim aşamasıdır. Çünkü bu aşama KSA'nın yerel arama işlemlerinin gerçekleştiği aşamadır. Memetik evrim aşamasının her bir iterasyonunda, alt mempleks grubuna seçilmiş olan en kötü kurbağanın bilgileri ya yerel ya da genel en iyi kurbağanın sahip olduğu mem bilgilerine göre güncellenmektedir. Ancak yerel ya da genel kurbağanın paylaştığı bilgiler en kötü kurbağa için daha iyi bir çözüm oluşturmuyor ise bu durumda en kötü kurbağanın yerine rastgele yeni bir çözüm üretilmektedir. Kurbağa sıçrama algoritmasının memetik evrim aşamasında gerçekleştirilen işlemler şu şekilde sıralanabilir;

- **Adım 5.1:** Memetik evrim aşaması her mempleks grubu için N adet çevrim olarak gerçekleştirileceği için iç içe iki döngü içermektedir. Birinci döngü ($im = 1, \ldots, m$) kadar olan üzerinde işlem yapılan mempleks grubunu tutan sayaçtır. İkinci döngü ise ($iN = 1, \ldots, N$) kadar olan iterasyon sayısını tutan sayaçtır.
- **Adım 5.2:** KSA'daki memetik evrim aşaması en kötü kurbağaların sahip olduğu bilgileri güncelleyerek en kötü kurbağalar için daha iyi çözümler bulmak, bunun sonucunda da algoritmanın o aşamaya kadar bulduğu genel çözümden daha iyi bir

çözüme ulaşmak için gerçekleştirilir. KSA'da en kötü kurbağa güncellenirken üç durum söz konusudur;

- **Birinci durum:** En kötü kurbağa $(P_w: P_{worst})$ yerel en iyi kurbağaya $(P_b: P_{best})$ göre güncellenir.
- **İkinci durum:** Birinci durumda güncelleme olmaz ise; en kötü kurbağa genel en iyi kurbağaya (P_x) göre güncellenir.
- **Üçüncü durum:** İkinci durumda güncelleme olmaz ise; en kötü kurbağa yerine rastgele yeni bir kurbağa oluşturulur.

En kötü kurbağa güncellenirken bu üç seçenekten sadece biri uygulanmaktadır. En kötü kurbağanın iyileştirilmesi aşamasında eğer en kötü kurbağa sürekli olarak genel en iyi kurbağaya benzetilmeye çalışılırsa dar bir arama uzayında çözümler aranmaktadır. Bunun sonucunda ise yerel bir çözüme takılma ihtimali yükselmiş olur. KSA en kötü kurbağayı güncellerken hem yerel arama kullanma şansının olması hem de genel arama kullanma şansının olması çözüm kalitesini arttırmaktadır. Memetik evrim aşamasında bir memplekste N adet çevrimin her birinde daha önce belirlenen q adet kurbağa, üçgensel olasılık dağılım fonksiyonuna göre belirlenmiş olan seçilme olasılıklarına göre rulet tekerleği ile alt gruba seçilmektedir. Bu durumda en iyi kurbağaların alt mempleks grubuna seçilme olasılıkları daha yüksektir. Alt mempleks içindeki en kötü kurbağa (P_w) öncelikli olarak alt mempleks içindeki en iyi çözüm olan yerel en iyi kurbağa (P_b) ile bilgi paylaşımı gerçekleştirir. P_b ile P_w arasında gerçekleşen bilgi paylaşımı sonunda oluşan yeni P_w için uygunluk değeri hesaplanır. Eğer P_w'nin yeni uygunluk değeri bir önceki P_w'den daha iyi bir sonuç değilse, bu durumda alt grup içindeki en kötü kurbağa (P_w) ile popülasyondaki en iyi kurbağa olan genel en iyi kurbağa (P_x) arasında memetik evrim gerçekleştirilir. Şayet yeni oluşan kurbağanın uygunluk değeri bir önceki P_w'nin uygunluk değerinden daha iyi değilse bu durumda en kötü kurbağanın yerine rastgele yeni bir kurbağa eklenir. Memetik evrim aşamasının her bir çevrimi için alt gruba seçilecek kurbağalardan uygunluk değeri en iyi olan kurbağa en büyük olasılık değerine, en kötü kurbağa da en kötü olasılık değerine sahiptir. Mempleks içindeki her bir kurbağanın sahip olduğu olasılık değeri eşitlik (3.3)'de verilen üçgensel dağılım fonksiyonuna göre belirlenmektedir.

$$P_i = \frac{2(n+1-i)}{n(n+1)} \qquad i = 1, \ldots . n \qquad (3.3)$$

Eşitlik (3.3)'te verilen fonksiyona göre mempleks içindeki en iyi uygunluk değerine sahip kurbağanın seçilme olasılığı en yüksek, en kötü uygunluk değerine sahip kurbağanın seçilme olasılığı ise en düşüktür. Mempleks içindeki birinci kurbağanın alt memplekse seçilme olasılığı:

$$P_1 = 2/(n+1)$$

şeklinde olur ve bu kurbağanın alt memplekse seçilme olasılığı en yüksektir. Mempleks içindeki en son kurbağanın alt memplekse seçilme olasılığı ise:

$$P_n = 2/n(n+1)$$

şeklinde olur ve bu kurbağanın alt memplekse seçilme olasılığı en düşüktür. Mempleks grubu içinde n adet kurbağa olduğu için bu n adet kurbağanın seçilme olasılıkları toplamı birdir (1). Bundan dolayı her bir kurbağanın seçilme olasılığı uygunluk değerine göre 0 ile 1 arasında dağıtılır. n adet kurbağa için seçilme olasılıkları belirlendikten sonra, bu kurbağalar arasından birbirinden farklı q adet kurbağa seçilme olasılıklarına göre rulet tekerleği ile alt mempleks grubuna seçilir.

- **Adım 5.3:** Daha önceki aşamada belirtildiği gibi öncelikli olarak alt mempleks içindeki en kötü kurbağa (P_w) ile en iyi kurbağa (P_b) arasında bilgi paylaşımı gerçekleştirilerek en kötü kurbağa iyileştirilmeye çalışılır. En kötü kurbağanın konum güncellemesi çözüm uzayında ileriye doğru ya da geriye doğru bir sıçrama işlemi ile gerçekleştirilir. Aşağıda verilen eşitliklerden eşitlik (3.4)'de ileri (pozitif) yönde gerçekleşecek adım büyüklüğü, eşitlik (3.5)'te ise geri (negatif) yönde gerçekleşecek adım büyüklüğü formülleri verilmiştir.

$$S^i = min\{int[rand * (P_b^i - P_w^i)], S_{max}^i\} \qquad (3.4)$$

$$S^i = max\{int[rand * (P_b^i - P_w^i)], -S_{max}^i\} \qquad (3.5)$$

Yukarıda verilen formüllere göre elde edilen S^i değerine göre en kötü kurbağa ileri yönde ya da geri yönde hareket etmiş olur. $i = (1, \ldots . , d)$ olmak üzere i; üzerinde işlem yapılan problemin i. boyutunu temsil etmektedir. Graf boyama problemi için graftaki düğüm sayısı, problemin boyutu (d) olarak tanımlanmaktadır. P_w ile P_b kurbağaları arasında bilgi geçişi sırasında her bir boyut kendi arasında memetik bilgi geçişini sağlamaktadır. Örneğin beş düğümden oluşan

bir graf için mempleks içindeki alt memplekste bulunan kurbağalardan P_w ile P_b kurbağaları arasında bilgi geçişi yapılırken, kurbağanın birinci boyutunda bulunan birinci düğümler kendi arasında, ikinci düğümler kendi aralarında olacak şekilde beş düğüm için her kurbağanın i. boyutları arasında bilgi paylaşımı yapılır. Formüllerde verilen $rand$ değeri [0, 1] aralığında üretilen bir sayıdır. S_{max} değeri ise parametrelerin belirlenmesi aşamasında belirlenmiş olan maksimum sıçrama miktarıdır. Yani en kötü kurbağanın her bir boyutu için belirlenen maksimum değişimi ifade eder. Örneğin $d = 5$ olarak belirlenen bir problem için en kötü kurbağanın birinci boyutunun uğrayacağı değişim pozitif yönde ise S_{max} değerinden büyük, negatif yönde ise $-S_{max}$ değerinden küçük olamaz. P_w kurbağasının her bir boyutu için S^i adım büyüklüğü (değişim miktarı) hesaplandıktan sonra bulunan S değerleri en kötü kurbağanın ilgili boyutlarına eklenir. Böylece en kötü kurbağa için yeni bir konum elde edilmiş olur. Eşitlik (3.6)'da en kötü kurbağanın yeni konumunun nasıl hesaplandığını gösteren eşitlik verilmiştir.

$$P_{w\_yeni} = P_w + S \tag{3.6}$$

En kötü kurbağa için elde edilen yeni konum olan P_{w_yeni} konumu, problem için belirlenmiş olan sınır değeri aralıkları arasında ise P_{w_yeni} için maliyet değeri hesaplanır. En kötü kurbağanın yeni maliyet değeri f_{yeni}, ilk maliyet değeri olan f_w değerinden daha iyi ise en kötü konumdaki kurbağanın P_w konumu P_{w_yeni} ile değiştirilir. Ayrıca yeni uygunluk değeri olan f_{yeni}, f_w'nin yerine eklenir. En kötü kurbağanın mevcut konumu güncellendiği için Adım 5.6'ya geçilir. Aksi durumda, P_{w_yeni}'nin uygunluk değeri, ilk uygunluk değeri olan f_w'den daha kötü bir uygunluk değerine sahiptir ya da yeni kurbağanın herhangi bir boyutundaki bir değer, problem için belirlenmiş olan sınır aralıklarının dışındadır. Bu durumda en kötü kurbağa güncellenemeyeceği için Adım 5.4' e geçilir.

- **Adım 5.4:** En kötü kurbağanın iyileştirilmesi aşamasında alt mempleks içindeki yerel en iyi kurbağanın (P_b) kullanılması sonucunda daha iyi bir çözüm bulunamadığından dolayı, en kötü kurbağanın iyileştirilmesi için genel en iyi kurbağa (P_x) kullanılır. Adım 5.3'te alt memplekseteki en kötü kurbağa ile en iyi

kurbağa arasında gerçekleştirilen işlemler bu adımda alt mempleks içinde bulunan en kötü kurbağa ile genel en iyi kurbağa arasında gerçekleştirilir.

$$S^i = min\{int[rand * (P_x^i - P_w^i)], S_{max}^i\} \quad (3.7)$$

$$S^i = max\{int[rand * (P_x^i - P_w^i)], -S_{max}^i\} \quad (3.8)$$

Eşitlik (3.7) ve eşitlik (3.8)'de verilen denklemlere göre S^i değişim miktarının belirlenmesi için en kötü kurbağa ile genel en iyi kurbağa (P_x) kullanılmıştır. Daha sonra problemin her bir boyutu için bulunan S^i değişim miktarı, eşitlik (3.6)'de verilen denkleme göre, en kötü kurbağanın ilk konumuna eklenerek, en kötü kurbağanın bilgileri güncellenir. Oluşan yeni kurbağa için maliyet hesabı yapılır. En kötü kurbağa üzerinde yapılan değişiklikler sonucunda oluşan kurbağanın f_{yeni} maliyet değeri, en kötü kurbağanın değişim öncesindeki f_w maliyet değerinden daha iyi bir maliyet değerine sahipse P_w'nin konumu P_{w_yeni} ile değiştirilir ve eski uygunluk değeri olan f_w de f_{yeni} ile değiştirilir. Daha sonra Adım 5.6'ya geçilir. Ancak en kötü kurbağa üzerinde yapılan değişiklikler sonucu oluşan P_{w_yeni}, P_w' den daha iyi bir çözüm değilse ya da P_{w_yeni}'nin herhangi bir boyutundaki bir değer problem için belirlenmiş olan sınır değeri aralıklarının dışındaysa Adım 5.5'e geçilir.

- **Adım 5.5:** Eğer en kötü kurbağanın güncellenmesi aşamasında Adım 5.3 veya Adım 5.4'e göre en kötü kurbağa güncellenemiyorsa, en kötü kurbağanın yerine problemin kısıtlarına uygun olarak rastgele yeni bir kurbağa eklenir. Yani alt mempleksin o çevrimindeki en kötü P_w'nin yerine, oluşturulan yeni kurbağa eklenir. Daha sonra bu kurbağa için maliyet hesabı yapılır. f_w'in yerine yeni f_{yeni} maliyet değeri eklenir.
- **Adım 5.6:** Alt mempleks içindeki en kötü kurbağanın konum ve maliyet değerleri güncellendikten sonra, alt mempleks içindeki kurbağalar üzerinde evrimsel döngünün gerçekleştirildiği, mempleks içindeki konumlarına taşınırlar. Daha sonra mempleks içindeki kurbağalar uygunluk değerlerine göre en iyi kurbağadan en kötü kurbağaya doğru olacak şekilde sıralanırlar. Böylece üzerinde işlem yapılan mempleks için birinci evrimsel döngü tamamlanmış olur.
- **Adım 5.7:** Üzerinde işlem yapılan mempleks için evrimsel döngü sayacı bir attırılır $(iN = iN + 1)$. Böylece evrimsel döngüdeki bir iterasyon tamamlanmış olur

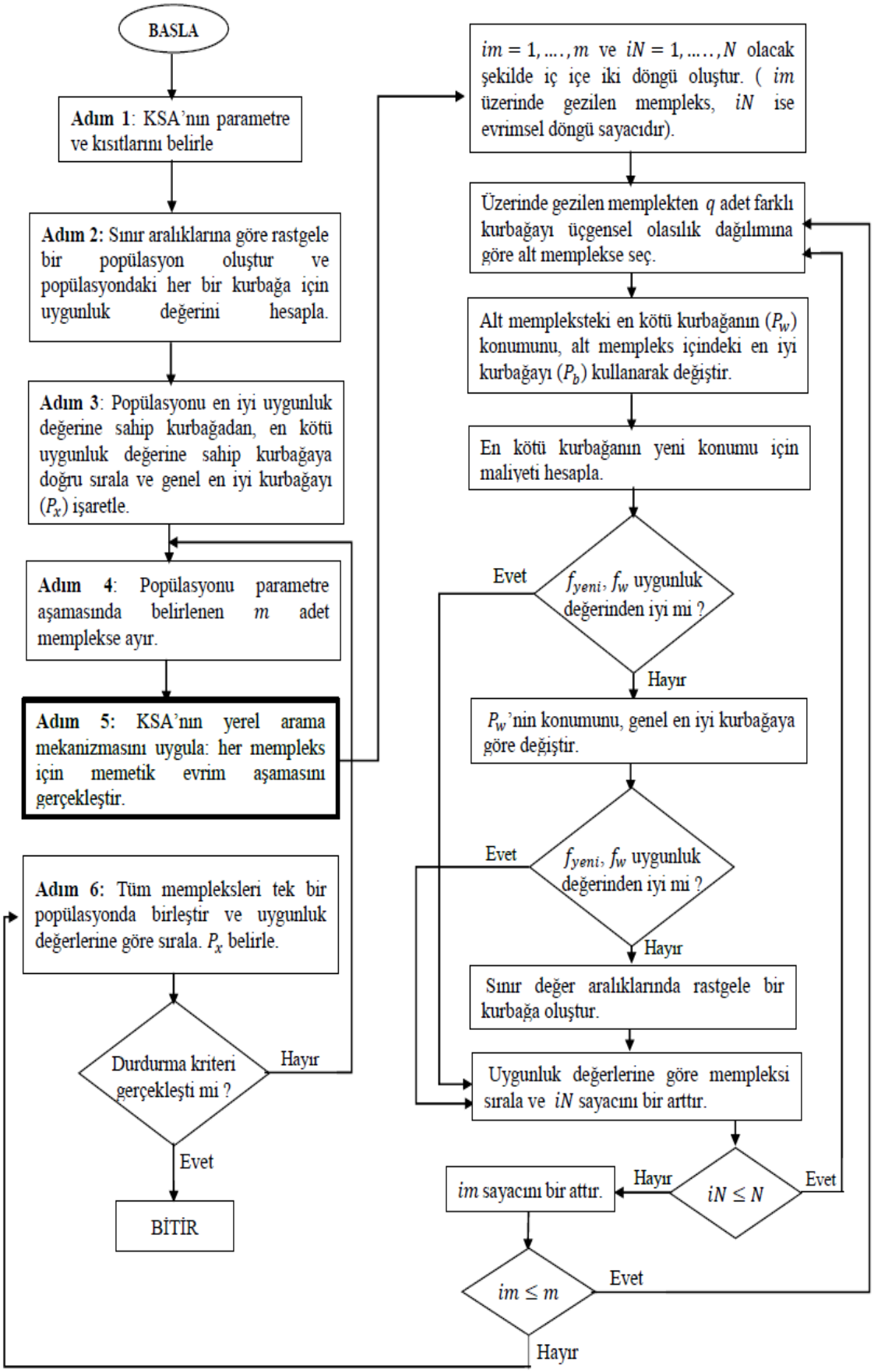
ve Adım 5.8'e geçilir. Eğer $iN \leq N$ ise üzerinde işlem yapılan mempleks için evrimsel döngü sayısı tamamlanmamış olduğundan Adım 5.2'ye geri dönülür.

- **Adım 5.8:** Eğer bu adıma geçilmiş ise üzerinde işlem yapılan mempleks için evrimsel döngü tamamlanmış demektir. Bu durumda bir sonraki memplekse geçilerek, bu mempleks üzerinde evrimsel döngü işlemlerini gerçekleştirmek için $(im = im + 1)$ olacak şekilde im sayacı bir arttırılır.
- **Adım 5.9:** im sayacı kontrol edilir. Eğer $im \leq m$ ise Adım 5.2'ye gidilerek $im.$ mempleks için evrimsel döngü aşaması yeniden başlatılır. Ancak $im > m$ olması durumunda tüm mempleksler için evrimsel döngüler tamamlanacağı için yerel arama aşaması sonlandırılarak Adım 6'ya geçilir.

Adım 6 (Yeni popülasyon üretme): KSA'nın yerel arama mekanizması olan Adım 5 tamamlandıktan sonra mempleklerdeki kurbağalar tek bir popülasyonda tekrardan bir araya getirilir. Bu işlem için Y mempleks kümesinde bulunan her bir mempleks içindeki n adet kurbağa X kümesinde bir araya getirilir. Daha sonra X kümesinde bulunan kurbağalar uygunluk değerlerine göre sıralanır. Bu işlem için kurbağalar en iyi uygunluk değerine sahip kurbağadan en kötü uygunluk değerine sahip kurbağaya doğru popülasyon içinde sıralanmaktadırlar. X kümesinde bulunan birinci kurbağa popülasyon içindeki en iyi uygunluk değerine sahip kurbağa olduğu için bu kurbağa genel en iyi kurbağa (P_x) olarak kaydedilir. Daha sonra Adım 7'ye geçilir.

Adım 7 (Sonlandırma): I iterasyon sayacı $I = I + 1$ olacak şekilde bir arttırılır. Eğer maksimum iterasyon sayısına ulaşılmışsa ya da programı sonlandıracak durdurma kriterleri sağlanmışsa program sonlandırılır, aksi durumda Adım 4'e geri dönülür.

KSA'nın çalışma adımları yukarıda verilmiştir. Şekil 3.37'de ise algoritmanın çalışma adımlarını gösteren akış diyagramı verilmiştir.



Şekil 3.37. Kurbağa sıçrama algoritmasının akış diyagramı

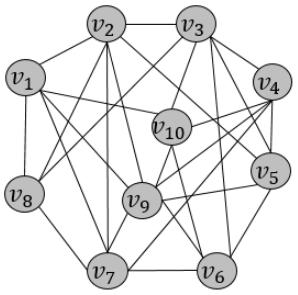
3.5.2. KSA'nın graf boyama problemine uygulanması

Kurbağa sıçrama algoritmasının (KSA), graf boyama problemine (GBP) uygulanması için probleminin yapısına uygun olarak düzenlenmesi gerekmektedir. KSA'nın popülasyonundaki her bir kurbağa d adet karar değişkenine bağlı olarak oluşturulur. GBP için d düğüm sayısına karşılık gelmektedir. Yani sınır değer aralıklarına göre rastgele oluşturulan popülasyondaki bir kurbağanın boyutu (uzunluğu) düğüm sayısına eşittir. KSA'nın amacı düzgün boyama kuralına uygun olarak GBP için kullanılan renk sayısını azaltmak ve en iyi çözümü bulmaktır. Bunun için de rastgele oluşturulan kurbağaları birbiri ile etkileşime sokarak, en iyi çözüm kümesini bulmaya çalışmaktadır.

KSA'nın, GBP'ye uygulanması Şekil 3.38'de verilen graf örneği üzerinde verilmiştir. Bu grafı boyamak için kullanılan renk sayısı, başlangıç aşamasında 5 renk olarak alınmıştır. Grafın kromatik sayısı dörttür. Ancak, bu graf için RLF algoritması sonucunda çıkan renk sayısı 5'tir. Tez kapsamında önerilen algoritmada ön işlem olarak RLF algoritması kullanıldığından, başlangıç değeri olarak RLF sonucu alınmıştır. KSA'nın parametreleri ise şu şekilde belirlenmiştir:

- Mempleks sayısı $m = 1$, mempleks grubu içinde bulunan kurbağa sayısı $n = 5$,
- Memetik evrim aşamasında alt grup için seçilecek birey sayısı $q = 5$,
- Kurbağada meydana gelecek değişimin ne kadar olacağının belirlendiği maksimum adım büyüklüğü $S_{max} = 2$ ve $rand = 0.7$ olarak alınmıştır.

Belirlenen kısıtlara göre beş kurbağadan oluşan popülasyon $renk\ kısıtı = (1, \dots, 5)$ olacak şekilde rastgele olarak olarak oluşturulur. KSA'da tek mempleks olması ve alt memplekse seçilen kurbağa sayısı, popülasyon sayısına eşit olduğundan dolayı, tüm işlemler tek mempleks içinde gerçekleşmektedir. Rastgele oluşturulan popülasyon için maliyet hesabı yapılmalıdır. Popülasyon içindeki kurbağalar uygunluk değerine göre sıralanır. Uygunluk değerine göre sıralanan mempleks içindeki ilk kurbağa yerel en iyi kurbağa olarak işaretlenir. Algoritmada tek bir mempleks kullanıldığı için yerel en iyi kurbağa (P_b) aynı zamanda genel en iyi kurbağadır (P_x). Mempleks içindeki son kurbağa ise en kötü kurbağa (P_w) olarak işaretlenir.



Şekil 3.38. 10 düğümlü bir graf

Çizelge 3.11. Parametrelere göre oluşturulan rastgele popülasyon

$Kurbağa\ (1)\ =$	1	2	3	1	2	4	5	4	5	3
$Kurbağa\ (2)\ =$	5	5	3	2	1	3	4	2	1	1
$Kurbağa\ (3)\ =$	4	5	3	5	2	4	3	1	3	2
$Kurbağa\ (4)\ =$	3	4	3	1	2	5	5	4	5	3
$Kurbağa\ (5)\ =$	1	3	2	1	2	2	5	4	4	3

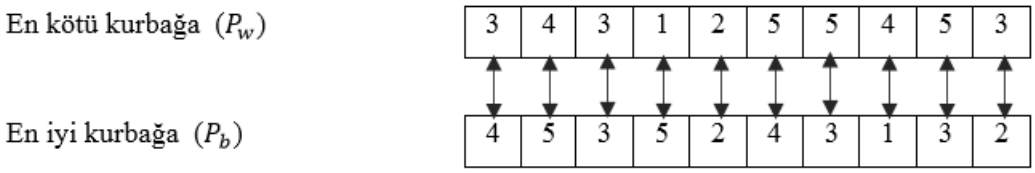
Çizelge 3.11'de 10 düğümlü graf için beş bireyden oluşan rastgele bir popülasyon verilmiştir. Her bir kurbağa bir çözümü temsil etmektedir. Kurbağanın boyutu ise düğüm sayısına eşittir. Her bir düğüme verilen renk, komşulukları gözetilmeksizin rastgele olarak dağıtılmıştır. Her kurbağa için maliyet değeri hesaplanmalı ve en kötü kurbağa ile en iyi kurbağa tespit edilmelidir. Bir kurbağa için uygunluk değeri hesaplanırken, o kurbağa için kullanılan renk sayısı ile her düğüm için aynı renk ile boyanan komşu düğümlerin sayısı toplanmıştır. Çizelge 3.12'de kurbağaların uygunluk değerleri verilmiştir. Mempleks içindeki ilk kurbağa için uygunluk hesabı şöyledir:

Birinci kurbağada kullanılan farklı renk sayısı hesaplanır. $Kurbağa(1)'de\ kullanılan\ farklı\ renk\ sayısı = 5$, daha sonra her bir düğüm için tek tek komşu aynı renge sahip komşuları hesaplanır. v_1 düğümüne komşu olan düğümler v_2, v_7, v_8, v_9 ve v_{10} düğümleridir. Bu düğümlerden hiç biri v_1 düğümü ile aynı renge sahip değildir. Bundan dolayı v_1 için $maliyet = 0$'dır. v_2 düğümüne komşu olan düğümler v_1, v_3, v_5, v_7, v_8 ve v_9 düğümleridir. v_2 düğümü ile aynı renge sahip düğüm sadece v_5 düğümüdür. Tüm düğümler için maliyetler hesaplandıktan sonra bu maliyetler, o birey için kullanılan farklı renk sayısı ile toplanarak toplam maliyet bulunur. $Kurbağa(1)$ için maliyet $f_1 = farklı\ renk\ sayısı + tüm\ düğümlerin\ maliyeti$'dir. $f_1 = 11$ olarak hesaplanır.

Çizelge 3.12. Kurbağaların uygunluk değerleri

Birinci kurbağanın uygunluk değeri	→	$f_1 = 11$
İkinci kurbağanın uygunluk değeri	→	$f_2 = 13$
Üçüncü kurbağanın uygunluk değeri	→	$f_3 = 7$
Dördüncü kurbağanın uygunluk değeri	→	$f_4 = 17$
Beşinci kurbağanın uygunluk değeri	→	$f_5 = 11$

Çizelge 3.12'e göre en iyi uyguluk değerine sahip kurbağa üçüncü kurbağadır. Bu kurbağa yerel en iyi kurbağa (P_b) olarak işaretlenir. En kötü kurbağa ise dördüncü kurbağadır ve yerel en kötü kurbağa (P_w) olarak işaretlenir.



Şekil 3.39. En iyi ve en kötü kurbağaların konumları

Daha sonra alt mempleks içinde bulunan en iyi ve en kötü kurbağa kullanılarak, en kötü kurbağanın konumu değiştirilir. En kötü kurbağanın güncellenmesi işlemi gerçekleştirilirken P_b ile P_w kurbağalarının her boyutu (her düğüm çifti) kendi arasında memetik aşamayı gerçekleştirir. Kurbağaların her bir boyutu arasındaki S^i adım büyüklükleri belirlenir. S^i adım büyüklükleri eşitlik (3.4) ve eşitlik (3.5)'e göre hesaplanır. Daha sonra bulunan S^i adım büyüklükleri eşitlik (3.6)'ya göre en kötü kurbağanın sahip olduğu bilgilere eklenir. Böylece en kötü kurbağa için yeni bir konum oluşmuş olur. Şekil 3.39'da en kötü ve en iyi kurbağalar verilmiştir. Çizelge 3.13'te ise her bir boyut için S^i adım büyüklükleri, bunların hesaplanması işlemi verilmiştir ve en kötü kurbağanın her bir boyutunun yeni konumları verilmiştir. İşlem sonunda en kötü kurbağanın yeni konumu (P_{w_yeni}) sınır değer aralığında olduğu için maliyet hesabı yapılmıştır. P_{w_yeni}'nin uygunluk değeri $f_{yeni} = 16$ olarak bulunmuştur. f_{yeni}, P_w'nin uygunluk değerinden (başlangıçta 17 idi) daha iyi bir uygunluk değerine sahip olduğu için en kötü kurbağanın konumu, yeni konum ile değiştirilir. Böylece üzerinde işlem yapılan mempleks için bir çevrim tamamlanmış olur.

Çizelge 3.13. En kötü kurbağanın konum güncellenmesi işlemi

$P^i_{w\_yeni}$	S^i			+	P^i_w	=	$P^i_{w\_yeni}$
$P^1_{w\_yeni}$	$S^1 = min(int[0.7 * (4-3)], 2)$	=	1	+	3	=	4
$P^2_{w\_yeni}$	$S^2 = min(int[0.7 * (5-4)], 2)$		1		4		5
$P^3_{w\_yeni}$	$S^3 = min(int[0.7 * (3-3)], 2)$		0		3		3
$P^4_{w\_yeni}$	$S^4 = min(int[0.7 * (5-1)], 2)$		2		1		3
$P^5_{w\_yeni}$	$S^5 = min(int[0.7 * (2-2)], 2)$		0		2		2
$P^6_{w\_yeni}$	$S^6 = max(int[0.7 * (4-5)], -2)$		-1		5		4
$P^7_{w\_yeni}$	$S^7 = max(int[0.7 * (3-5)], -2)$		-1		5		4
$P^8_{w\_yeni}$	$S^8 = max(int[0.7 * (1-4)], -2)$		-2		4		2
$P^9_{w\_yeni}$	$S^8 = max(int[0.7 * (3-5)], -2)$		-1		5		4
$P^{10}_{w\_yeni}$	$S^{10} = max(int[0.7 * (2-3)], -2)$		-1		3		2

KSA, graf boyama problemine doğrudan uygulanmak istenirse, yukarda verilen örnek çalışmada olduğu gibi uygulanabilir. Ancak GBP için düğümlere verilen renkler önemli olduğu için boyutlar arasında gerçekleşen bilgi alışverişi neticesinde, üzerinde işlem yapılan düğümün rengi değişmekte ve bu değişim doğrudan o düğüme komşu olan düğümleri de etkilemektedir. Bundan dolayı KSA'yı düzgün boyama kuralına bağlı kalarak doğrudan GBP'ye uygulamak, düğüm sayısının az olduğu graflar için doğru sonuçlar verebilirken, nispeten düğüm sayısının gittikçe arttığı graflar için çözüm bulamayabilmektedir. Çünkü bir düğümün renginin değişmesi doğrudan komşu düğümlerini de etkilemekte ve çözüm bulmak zorlaşmaktadır.

KSA'nın yerel arama mekanizması olan Adım 5 üzerinde bazı değişiklikler yapmak KSA'nın, GBP için iyi çözümler bulmasını sağlayabilir. KSA'nın genel araması oldukça etkili olduğundan dolayı, yapılan değişiklikler neticesinde GBP için etkili bir yöntem geliştirilebilir.

KSA'nın yerel arama mekanizmasında en kötü kurbağanın konumu güncellenirken kullanılan eşitliklerde değişikliğe gidilmiştir. Yapılan çalışmada en kötü kurbağanın konumu güncellenirken, Genetik Algoritmanın (GA) çaprazlama ve mutasyon operatörlerinden faydalanılmıştır. Ayrıca üzerinde işlem yapılan mempleksin N memetik evrim sayısı tamamlandıktan sonra yerel en iyi kurbağaya komşuluk araması uygulanmıştır. Komşuluk aramasının uygulanmasının sebebi, yanlış boyanmış olan düğümlerin, komşularının sahip olduğu renkler de dikkate alınarak daha hızlı bir şekilde düzgün boyanmalarını sağlamaktır. Değişken Komşu Araması (DKA) (Variable Neighborhood Search-VNS) algoritması, Mladenović ve Hansen (1997) tarafından

literatüre kazandırılmış olan komşuluk araması algoritmasıdır (Mladenović ve Hansen, 1997). Avanthay ve ark. (2003), DKA algoritmasında bazı değişiklikler yaparak DKA'yı GBP üzerine uygulamışlardır. DKA algoritması içinde farklı komşuluk arama mekanizmaları içermektedir. Bu çalışmada kullanılan komşuluk arama yöntemi ise Avanthay ve ark. (2003) çalışmalarında kullandıkları düğüm komşulukları (vertex neighborhood) algoritmalarından Chain komşuluk araması metodudur (Avanthay ve ark., 2003).

KSA üzerinde yapılan değişikliklerden sonra, graf boyama problemi için uygulanabilir bir yöntem önerilmiştir.

3.5.3. Graf boyama problemi için önerilen KSA tabanlı algoritma

GBP için önerilen algoritmada iki aşama söz konusudur. Birinci aşama tahmin aşamasıdır. Bu aşamada açgözlü algoritmalardan RLF algoritması kullanılmıştır. Tahmin aşamasının kullanılmasının sebebi üzerinde işlem yapılacak graf için bir üst sınır belirlemektir. Çünkü düğüm sayısının 100'den fazla olduğu bir graf için başlangıç aşamasında oluşturulacak popülasyon için kullanılacak maksimum renk sayısının belirlenmesi gerekmektedir. Örneğin bir graf için oluşturulacak başlangıç popülasyonu düğüm sayısı kadar renk kullanılarak rastgele oluşturulacak olursa bu durumda bu grafı boyamak için kullanılacak renk sayısının optimize edilmesi daha çok işlem gerektirecektir. Bu da maliyeti doğal olarak arttıracaktır. Bunun için öncelikli olarak üzerinde işlem yapılan graf, bir açgözlü algoritma ile boyanmıştır. Böylece başlangıç popülasyonu oluşturulurken kullanılacak maksimum renk sayısı için bir kısıtlama getirilmiştir. Algoritmanın ikinci aşamasında ise KSA kullanılmıştır. Birinci aşamada elde edilen renk sayısı ile rastgele bir popülasyon oluşturulmuş, daha sonra rastgele oluşturulan popülasyona KSA uygulanmıştır. Popülasyondaki bireyler rastgele olarak oluşturulduğu için genellikle düğümler yanlış renkler ile boyanmış olmaktadır. KSA'da öncelikli amaç bireyleri, düzgün renklendirme kuralına uygun olarak iyileştirmektir. Düzgün bireylere ulaşıldıktan sonra kullanılan renk sayısını daha da azaltarak daha iyi çözümlerin bulunması amaçlanmıştır. Bunun için algoritmanın ikinci aşamasında, Saf KSA'nın yerel arama mekanizması olan Adım 5'te bazı değişikliklere gidilmiştir. Algoritmada, Adım 5 dışında değişikliğe gidilmemiş, Saf KSA'nın diğer adımları aynen uygulanmıştır. Adım 5'te yapılan değişiklikler aşağıda verilmiştir;

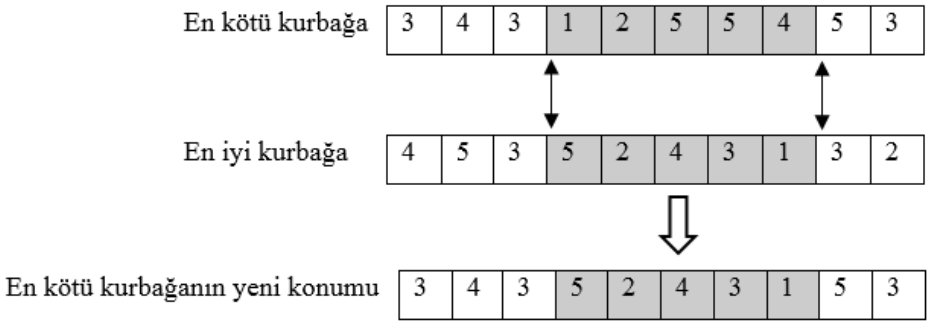
- En kötü kurbağanın konumunun güncellenmesi sırasında kullanılan eşitlik (3.4), (3.5), (3.6), (3.7) ve (3.8), önerilen algoritmada kullanılmamıştır. Bu eşitliklerin yerine genetik algoritmanın çaprazlama ve mutasyon operatörleri kullanılmıştır.
- Üzerinde gezinilen mempleks için, memetik evrim aşaması bittikten sonra mempleks içindeki en iyi kurbağaya DKA algoritmasının komşuluk arama mekanizmalarından olan Chain metodu uygulanmıştır. Chain metodu, kullanılan renk sayısını azaltmaz. Ancak rastgele oluşturulan bireylerin daha hızlı bir şekilde düzgün hale getirilmesini sağlar. Böylece KSA'nın daha etkin bir şekilde renk sayısını azaltması sağlanmıştır.

3.5.3.1. En kötü kurbağanın iyileştirilmesi aşaması

En kötü kurbağanın iyileştirilmesi aşamasında genetik algoritmanın çaprazlama ve mutasyon operatörleri kullanılmıştır. KSA'da en kötü kurbağanın konumu güncellenirken üç durum söz konusudur. Önerilen algoritmada ise en kötü kurbağanın konumu güncellenirken, GA kullanılmıştır. Buna göre; en kötü kurbağanın konumu yerel en iyi ya da genel en iyi kurbağaya göre güncellenecekse, genetik algoritmanın çaprazlama operatörü kullanılmıştır. Eğer, rastgele yeni bir birey oluşturulacaksa; bu durumunda da rastgele yeni bir birey oluşturmak yerine en kötü kurbağaya mutasyon uygulanmıştır.

GA'da çaprazlama operatörü için farklı yaklaşımlar geliştirilmiştir (Michalewicz, 1994; Adewuya, 1996). Bu yaklaşımlardan bazıları; basit çaprazlama, lineer çaprazlama, tek noktalı çaprazlama, iki noktalı çaprazlama, çok noktalı çaprazlama, uniform çaprazlama gibi yaklaşımlardır. Yapılan çalışmada en kötü kurbağanın konum güncellenmesi aşamasında, yerel en iyi ya da genel en iyi kurbağa ile en kötü kurbağanın konumu güncellenirken iki noktalı çaprazlama kullanılmıştır (Fleurent ve Ferland, 1996). İki noktalı çaprazlamada bireyin birinci geni ile son geni arasında rastgele iki nokta seçilmektedir. Daha sonra ise seçilen iki bireyin bu iki nokta arasındaki genleri birbiri ile değiştirilmektedir. Böylece yeni bireyler elde edilmektedir. GBP için ise; üzerinde işlem yapılan grafın birinci düğümü ile son düğümü arasında rastgele iki nokta seçilir. Daha sonra en kötü kurbağanın bu iki nokta arasında kalan boyutları, yerel en iyi ya da genel en iyi kurbağanın aynı noktalar arasında kalan boyutları ile değiştirilmektedir. Böylece en kötü kurbağa için yeni bir konum oluşturulmuştur.

Çizelge 3.11'de, 10 düğümlü bir graf için beş bireyden oluşan rastgele bir popülasyon verilmiştir. Çizelge 3.12'te ise bu kurbağaların uygunluk değerleri verilmiştir. En iyi uygunluk değerine sahip kurbağa üçüncü kurbağa, en kötü uygunluk değerine sahip kurbağa ise dördüncü kurbağadır. Graf 10 düğümden oluştuğundan dolayı, 1 ile 10 arasında rastgele iki nokta seçilir. Rastgele seçilen iki nokta 4. düğüm ile 8. düğüm olarak kabul edilmiştir. İki noktalı çaprazlama ile en kötü kurbağanın güncellenmesi işlemi Şekil 3.40'ta verilmiştir.



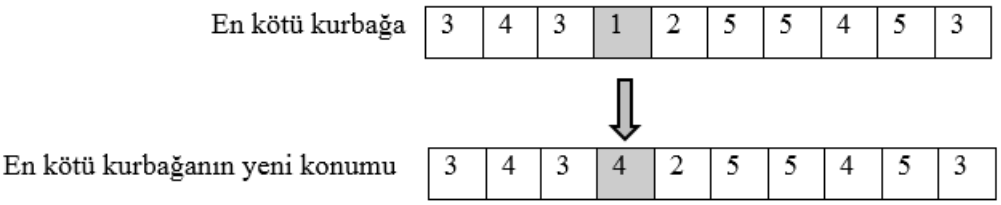
Şekil 3.40. İki noktalı çaprazlama ile en kötü kurbağanın güncellenmesi işlemi

Yerel en iyi ya da genel en iyi kurbağa, alt mempleks içindeki en kötü kurbağanın konumunu daha iyi bir konuma götürememişse, bu durumda en kötü kurbağaya mutasyon işlemi uygulanır. Genetik algoritmada mutasyonun kullanılmasının amacı popülasyon içinde çeşitliliği arttırmaktadır. Mutasyon bir bireyin kendi genleri üzerinde gerçekleştirilir. Mutasyon işlemi gerçekleştirilirken, bir bireyin bir ya da bir kaç geni değişime uğramaktadır (Adewuya, 1996; Kaya, 2012). Mutasyon işlemi için sıfır ile bir arasında [0,1] bir değer belirlenir. Bu değer genin mutasyona uğrama ihtimalini belirler. Mutasyon oranı bire (1) yaklaştıkça genin mutasyona uğrama ihtimali yükselmiş olur. Mutasyon işlemi bir bireyin tüm genleri için yapılabilir. Mutasyon işlemi şu şekilde gerçekleşir;

- **Adım 1:** Mutasyon oranı belirlenir.
- **Adım 2:** Bireyin birinci geni seçilir ($i = 1$).
- **Adım 3:** 0 ile 1 arasında rastgele bir sayı üretilir. ($rand$)
- **Adım 4:** Eğer $rand$ değeri, mutasyon oranından küçükse gen mutasyon işlemine tabi tutulur. Mutasyon işlemi sırasında mutasyona uğrayan gene, sınır değer aralıklarında rastgele bir değer atanır.

- **Adım 5:** $i = i + 1$ olacak şekilde bir attırılır. n toplam gen sayısı olmak üzere $i \leq n$ ise Adım 3'e geri dönülür. Değilse mutasyon işlemi tamamlanmış olur.

Fleurent ve Ferland (1996) yaptıkları çalışmada GBP için mutasyon operatörünü kullanmışlardır (Fleurent ve Ferland, 1996). Hindi ve Yampolskiy (2012) GBP için yaptıkları çalışmada mutasyon operatörünü uygularken problemin yapısına uygun olarak bazı değişiklikler yapmışlardır. İki farklı mutasyon işlemi uygulamışlardır. Birinci mutasyon işleminde mutasyona uğrayacak olan düğümün komşularının sahip olduğu renkleri tespit etmişlerdir. Daha sonra renk dizisindeki renklerden, komşu düğümlerin sahip olduğu renkleri çıkarmışlardır ve kalan renklerden birini rastgele bu düğüme vermişlerdir. İkinci mutasyon işleminde ise eğer mutasyona uğrayacak düğüm komşuları ile aynı renge sahipse, bu durumda bu düğüme renk dizisinde bulunan renklerden biri rastgele verilmiştir (Hindi ve Yampolskiy, 2012). Yapılan bu çalışmada da buna benzer bir mutasyon işlemi uygulanmıştır ve mutasyon işlemi Adım 4'te bazı değişiklikler yapılmıştır. Öncelikle mutasyona uğrayacak olan düğümün komşularının sahip olduğu renkler tespit edilmiştir. Daha sonra o graf için kullanılan renklerden, komşu düğümlerin sahip olduğu renkler çıkarılmıştır. Eğer renk dizisinde renk kalmışsa bu renklerden biri rastgele olarak bu düğüme verilmiştir. Ancak renk dizisindeki tüm renkler komşu düğümlerin sahip olduğu renkler ise, bu durumda düğüme renk dizisindeki renklerden biri rastgele olarak verilmiştir. Çizelge 3.11'te verilen popülasyondaki en kötü kurbağa olan dördüncü kurbağanın bir genine mutasyon işlemi uygulanarak, mutasyon işleminin nasıl gerçekleştirildiği Şekil 3.41'de verilmiştir. En kötü kurbağanın dördüncü genine mutasyon uygulanmıştır. Dördüncü düğümün rengi başlangıçta r_1'dir. Komşuluk matrisinden v_4 düğümün komşularına bakıldığında, $v_3, v_5, v_7, v_9, v_{10}$ düğümleri v_4 düğümüne komşu olan düğümlerdir. Bu düğümlerin sahip olduğu renkler ise $r_{2,} r_3$ ve r_5 renkleridir. Bu durumda renk dizisinde r_1 ve r_4 renkleri kalır. Bu iki renkten rastgele biri v_4 düğümüne verilir. v_4 düğümüne verilen renk r_4 rengidir. Ancak renk dizisinde v_4 düğümüne verilecek bir renk kalmasaydı, bu durumda beş renkten biri rastgele olarak bu düğüme verilirdi.



Şekil 3.41. En kötü kurbağanın bir boyutunun mutasyon ile güncellenmesi işlemi

3.5.3.2. Mempleks içindeki en iyi kurbağayı iyileştirme aşaması

Üzerinde gezilen mempleks için memetik evrim aşaması bittikten sonra mempleks içindeki en iyi kurbağaya DKA algoritmasının komşuluk arama mekanizmalarından olan Chain metodu uygulanmıştır. Chain metodu, kullanılan renk sayısını azaltmaz. Ancak rastgele oluşturulan bireylerin daha hızlı bir şekilde düzgün hale getirilmesini sağlar. Böylece KSA'nın daha etkin bir şekilde renk sayısını azaltması sağlanmıştır.

Chain metodu Avanthay ve ark.'ın (2003) çalışmalarında kullandıkları DKA düğüm komşuluk araması yöntemidir. Yapılan çalışmada da her mempleks içindeki en iyi kurbağaya Chain metodu uygulanmıştır. DKA, Chain metodu değiştirilmeden Avanthay ve ark. (2003) önerdiği şekilde, KSA'ya eklenmiştir. Chain metodu ile komşuluk araması şu şekilde gerçekleştirilir;

- Belirli bir döngü sayısı belirlenir. Her döngüde farklı bir düğüm üzerinde işlem yapılır. Döngü sonunda en iyi kurbağa için daha iyi bir çözüm oluşturulur.
- Öncelikle üzerinde işlem yapılan çözüm için komşuları ile aynı rengi almış olan düğümler tespit edilir ve bunlardan biri rastgele seçilir.
- Seçilen x düğümünün komşulukları da dikkate alınarak renk dizisindeki en uygun renk x düğümüne verilir. x düğümünün renginin değiştirilmesi ile muhtemelen hatalı boyanmış yeni düğümler oluşur. Daha sonra hatalı boyanan düğümlerden biri tekrar seçilir. Seçilen y düğümünün komşulukları da dikkate alınarak renk dizisindeki en uygun renk y düğümüne verilir. y düğümünün renginin değiştirilmesi ile hatalı boyanmış yeni düğümler oluşur. Daha sonra hatalı boyanan düğümlerden biri tekrar seçilir. Düğüm seçme ve renginin değiştirilmesi işlemi başka düğümler için de tekrar tekrar uygulanır.

- Chain metodu ile düğüme renk verme işlemi döngü sayısı kadar tekrarlanır. Burada dikkat edilmesi gereken husus bir düğümün rengi sadece bir kez değiştirilir. Çünkü algoritmanın her bir adımında seçilen düğümün renginin değiştirilmesi ile yanlış boyanan düğüm sayısı gitgide azalmaktadır.

3.5.3.3. Popülasyonun yeniden oluşturulması aşaması

Önerilen algoritmada başlangıç popülasyonunda bireyler açgözlü algoritmanın bulduğu üst sınır renk sayısı kullanılarak oluşturulmaktadır. Algoritmanın çalışması esnasında, düzgün boyama kuralına uygun olarak sonuca ulaşıldığında, bu sonuç o ana kadarki en iyi çözüm olarak kaydedilmekte ve üst sınır renk sayısı;

$$\text{üst sınır} = \text{üst sınır} - 1$$

olacak şekilde güncellenmektedir. Daha sonra popülasyondaki bireyler kontrol edilerek, bireylerde kullanılan üst sınır renk, son duruma göre güncellenmektedir. Algoritma düzgün bir çözüme ulaştığı anda renk sayısının azaltılması sadece GA'nın çaprazlama ve mutasyon operatörleri ile yapıldığında, daha az renk kullanarak çözüm bulmak zorlaşmaktadır. Bundan dolayı üst sınır renk sayısının algoritma içinde azaltılması algoritmanın daha etkin kullanılmasını sağlamaktadır. Popülasyon korunup, bireylerden sadece üst sınır renk değerine sahip genlerin bilgilerinin güncellenmesi işlemi yapıldığından dolayı, popülasyonun o ana kadar kazanmış olduğu tecrübe korunmaktadır.

3.5.3.4. Graf boyama problemi için maliyet (uygunluk) hesabı

Bir popülasyondaki bireylerin maliyeti hesaplanırken, her birey için maliyet ayrı ayrı hesaplanmaktadır. Graf boyama problemi için en düşük maliyete sahip birey en iyi bireydir. GBP için maliyet hesaplanırken, üzerinde işlem yapılan birey için her düğümün komşulukları incelenir ve aynı renge sahip komşu sayısının toplamı maliyeti doğrudan etkiler (Abbasian ve ark., 2011). Abbasian ve ark. (2011) yapmış oldukları çalışmada bir birey için maliyeti hesaplarken eşitlik (3.11)'de verilen denklemi kullanmışlardır.

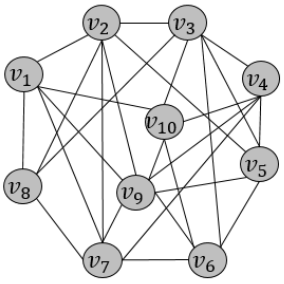
$$f(s) = \sum_{i \in V} \sum_{j \in adj_i} hata(i,j) \tag{3.9}$$

Eşitlik (3.9)'da verilen denklemde V düğümler kümesini temsil etmektedir. $i = (1, \ldots . n)$ olmak üzere n toplam düğüm sayısıdır. adj_i ise i düğümüne komşu olan düğümler kümesidir. Eşitlik (3.9)'e göre yapılan maliyet hesabında her düğümün kendi komşulukları incelenir ve hatalı boyanan düğümlerin sayısı $hata(i,j)$'e göre bulunur. Her düğüm için bulunan hata sayıları toplanarak, o birey için toplam maliyet hesaplanır.

Emami ve Lotfi (2013) yapmış oldukları çalışmada GBP için eşitlik (3.10)'da verilen maliyet fonksiyonunu kullanmışlardır (Emami ve Lotfi, 2013). Yapılan bu çalışmada da eşitlik (3.10)'de verilen maliyet fonksiyonu kullanılmıştır.

$$f(s) = \begin{cases} FarklıRenk(S), & Eğer \quad hata = 0 \\ \sum_{i \in V} \sum_{j \in adj_i} hata(i,j) * p \; + \; FarklıRenk(S), & Eğer \quad hata \neq 0 \end{cases} \tag{3.10}$$

Eşitlik (3.10)'da verilen $FarklıRenk(S)$ ifadesi; üzerinde işlem yapılan çözüm için kullanılan farklı renk sayısını verir. Eğer hatalı boyanan düğüm yoksa maliyet, o birey için kullanılan farklı renk sayısıdır. Ancak eğer hatalı boyanan düğüm varsa (komşularından en az bir tanesi ile aynı renge sahip düğüm), bu durumda maliyet hesabı için eşitlik (3.10)'de verilen ikinci eşitlik kullanılır. p ceza katsayısıdır. Toplam hata sayısının maliyete etkisini artırmak için kullanılır. İkinci eşitlikte toplam hata sayısı, kullanılan farklı renk sayısı ile toplanarak maliyet hesaplanmaktadır. Maliyet hesaplanırken kullanılan farklı renk sayısının maliyete eklenmesinin sebebi, daha düşük maliyetli bireylerin dikkate alınmasıdır. Çünkü eğer sadece hata sayısı dikkate alınırsa, hatalar giderildikten sonra sadece düzgün bir birey elde edilir. Ancak GBP için amaç, sadece düzgün bireyler elde etmek değil, aynı zamanda kullanılan renk sayısını da azaltmaktır. Bu amaçla önerilen çalışmada uygunluk değerlerinin hesaplanması için eşitlik (3.10) kullanılmıştır. $p = 1$ olarak alınmıştır. Çünkü cezanın büyük ya da küçük olması bütün maliyet hesaplarına etki edeceğinden, düşük maliyetli birey değişmeyecektir. Şekil 3.42'de verilmiş olan 10 düğümlü graf için oluşturulan rastgele S çözümü Çizelge 3.14'te verilmiştir.



Şekil 3.42. 10 düğümlü graf

Çizelge 3.14. Rastgele oluşturulan *S* çözümü

Düğümler	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}
S çözümü	1	2	3	1	2	4	5	4	5	3

Çizelge 3.14'te verilen S çözümü için maliyet hesabı yapılırken her düğüm için tek tek maliyet hesabı yapılır. Çizelge 3.15'de her bir düğümün komşu düğümleri ve maliyetleri verilmiştir.

Çizelge 3.15 incelendiğinde v_1 düğümünün renginin r_1 olduğu görülmektedir. Komşuluk matrisine göre komşu düğümleri v_2, v_7, v_8, v_9 ve v_{10}'dur. v_1 düğümü komşuları ile aynı renge sahip olmadığı için bu düğüm için maliyet $f_{v_1} = 0$ olur. S çözümü için kullanılan farklı renk sayısı $FarklıRenk(S) = 5$'tir. Eşitlik (3.10)'a göre toplam maliyet $f(s) = \left(f_{v_1} + f_{v_2} + \cdots + f_{v_{10}}\right) * p + FarklıRenk(S)$ hesabı yapılır. Çizelge 3.15'e göre düğümlerin toplam maliyeti 6 olarak hesaplanır. Bu durumda $f(s) = 6 * 1 + 5 => 11$ olur. ($p = 1$ olarak alınmıştır).

Çizelge 3.15. *S* çözümü için maliyet hesabı

v_i	v_i'nin Komşu Düğümleri	v_i'nin Maliyeti (f_{v_i})
v_1	v_2, v_7, v_8, v_9 ve v_{10}	$f_{v_1} = 0$
v_2	v_1, v_3, v_5, v_7, v_8 ve v_9	$f_{v_2} = 1$
v_3	v_2, v_4, v_5, v_6, v_8 ve v_{10}	$f_{v_3} = 1$
v_4	v_3, v_5, v_7, v_9 ve v_{10}	$f_{v_4} = 0$
v_5	v_2, v_3, v_4, v_6 ve v_9	$f_{v_5} = 1$
v_6	v_3, v_5, v_7, v_9 ve v_{10}	$f_{v_6} = 0$
v_7	v_1, v_2, v_4, v_6, v_8 ve v_9	$f_{v_7} = 1$
v_8	v_1, v_2, v_3 ve v_7	$f_{v_8} = 0$
v_9	$v_1, v_2, v_4, v_5, v_6, v_7$ ve v_{10}	$f_{v_8} = 1$
v_{10}	v_1, v_3, v_4, v_6 ve v_9	$f_{v_{10}} = 1$

4. DENEYSEL SONUÇLAR

Yapılan tez çalışmasında DIMACS tarafından sunulan Benchmark Grafları kullanılmıştır (Trick, 2016). Tez kapsamında bu graflar üzerinde önerilen algoritma test edilmiş ve deneysel sonuçlar verilmiştir.

4.1. DIMACS Benchmark Grafları ve Özellikleri

Yapılan çalışmada önerilen algoritma, DIMACS tarafından sunulan bazı Benchmark grafları üzerinde test edilmiştir. DIMACS tarafından sunulan Benchmark grafları, graf boyama problemini ilgili algoritmalar üzerinde test etmek için kullanılan graflardır (Johnson ve Trick, 1996). Çalışmada bu grafların kullanılmasının en önemli sebebi, algoritmaların performans karşılaştırması için belirli bir standart sunmasıdır.

DIMACS Benchmark graflarından bazıları çözülmesi zor, bazıları ise nispeten daha kolaydır. Benchmark graflarından bazılarının kromatik sayısı bilinirken bazılarının ise kromatik sayıları bilinmemektedir. Kromatik sayıları bilinmeyen graflar için literatürde bulunabilen en iyi sonuçlar verilmiştir (Yılmaz, 2011). Bu çalışmada kullanılan grafların kavis yoğunlukları denklem (4.1)'e göre hesaplanmıştır (Ruiz, 2012). Denklem (4.1)'de verilen bilgilerden E kavis sayısını, V ise düğüm sayısını göstermektedir.

$$D = \frac{2 * E}{\mathrm{V} * (\mathrm{V} - 1)} \qquad (4.1)$$

DIMACS tarafından sunulan graflar belirli bir formatta oluşturulmuştur. Şekil 4.1'de DIMACS tarafından sunulan graflar için dosya formatı verilmiştir. İlk satırdaki c karakterinden sonra gelen ifade graf hakkında yapılan açıklamayı içerir. İkinci satırdaki $p\ edge$ ifadesinden sonra gelen iki sayısal değerden birincisi düğüm sayısını, ikincisi ise kavis sayısını gösterir. Üçüncü satırdan itibaren son satıra kadar ise her satırda e ile başlayıp iki sayısal değer içeren ifadeler vardır. Bu sayısal değerler düğümleri temsil eder ve o satırda verilen iki düğüm arasında bir kavis olduğunu göstermektedir.

```
c açıklama: 1-FullIns grafı
p edge 30 100
e 1 2
e 1 4
e 1 11
e 1 13
e 2 3
e 2 10
e 2 12
e 3 6
```

Şekil 4.1. DIMACS grafları dosya formatı

$Myciel^*$ grafları Mycielski algoritmasına göre oluşturulan graflardır. Mycielski algoritması ile kromatik sayısı verilen bir graf için (graftaki düğümleri bağlayan kavislerin üçgen oluşturmaması koşulu ile) en az kaç düğüm gerektiği hesaplanır. Örneğin kromatik sayısı 3 olan bir graf için düğüm sayısı bulunurken $3 \times 2^{\chi(G)-2} - 1$ formülü kullanılmaktadır. Sonuç olarak $\chi(G) = 3$ olan bir graf için en az 5 düğüm gereklidir (Mycielski, 1955). $Myciel^*$ grafları M. Trick tarafından bu algoritmaya göre oluşturulmuştur (Johnson ve Trick, 1996).

$Insertion^*$ ve $Fullins^*$ grafları M.Caramia ve P.Dell'Olmo tarafından oluşturulan graflardır ve CAR grafları olarak bilinirler. Bu graflar Mycielski graflarının biraz daha zorlaştırılmış graflarıdır. Bu zorlaştırma işlemi grafın kavis yoğunluğu değişmemek şartı ile grafa yeni düğümler eklenerek yapılmıştır (Caramia ve Dell'Olmo, 2008).

Kitap Grafları (Book Graphs) ise Donald Knuth tarafından oluşturulmuştur. Donald Knuth yaptığı çalışmada beş kitap için graf oluşturmuştur. Kitaplardaki her bir karakter bir düğüm ile temsil edilmektedir. Kitapta geçen karakterlerin birbiri ile olan ilişkileri kavisler ile gösterilmiştir. Kitap içinde iki karakter birbiri ile karşılaştığı anda bu iki karakter (düğüm) arasında bir kavis çizilmektedir. Bu kitaplar: Tolstoy'un Anna Karenina (anna), Charles Dickens'ın David Copperfield (david), Homeros'un İlyada (homer), Mark Twain'nin Huckleberry Finn'in Maceraları (huck), ve Victor Hugo'nun Sefiller (jean) kitaplarıdır (Knuth, 1993; Mehrotra ve Trick, 1996).

Miles Grafları (Miles Graphs), Donald Knuth tarafından ABD haritasındaki belirli bir alan içindeki şehirleri göstermek için oluşturulmuştur. Miles graflarındaki her

bir düğüm bir şehri temsil etmektedir. Miles grafları belirli mesafede bulunan iki şehir arasında bir kavis çizilerek oluşturulmuştur (Knuth, 1993; Mehrotra ve Trick, 1996).

Oyun Graflarında (Game Graphs), bir futbol turnuvasında bir sezonda futbol takımları arasında oynanan maçlar tutulmaktadır. Bu grafta her bir düğüm bir takımı temsil etmektedir. Kavisler de sezon boyunca iki takımın kendi arasında oynadığı maçları göstermektedir (Knuth, 1993; Mehrotra ve Trick, 1996).

Queen grafları, nxn boyutlu satranç tahtası graflarıdır. Satranç oyununda iki vezirin aynı satır, sütun ya da köşegen üzerinde bulunmaması gerekir, aksi takdirde iki vezirden biri diğer veziri yer. Bundan dolayı iki vezirin birbirini yiyememesi için vezirlerin satranç tahtasındaki hareket alanları graf olarak tasarlanmıştır. $Queen$ graflarında matris üzerinde gezilirken seçili hücre (düğüm) kendi satır, sütun ya da köşegeninde bulunan hücre ile komşudur ve her komşu ile arasında bir kavis çizilmektedir. Böylece vezirlerin hareket alanları belirlenmiş olur (Mehrotra ve Trick, 1996).

David Johnson tarafından oluşturulan DSJ serisi grafları, çözülmesi zor Benchmark graflarıdır (Mahmoudi ve Lotfi, 2015). Bu graf türünde C ve R serisi alt graflar bulunmaktadır. $DSJC$ grafları random, $DSJR$ grafları ise random geometrik graflardır. $DSJC125.5$ grafı bir random graf örneğidir. Bu grafın ilk parametresi olan 125 değeri düğüm sayının 125 olduğunu, 5 değeri ise kavis yoğunluğun %50 olduğunu gösterir. Random geometrik graflar ise oluşturulurken, kare oluşturacak şekilde belirli sayıda düğüm oluşturulur, daha sonra belirli bir mesafede bulunan her düğüm çifti arasında bir kavis çizilir (Yılmaz, 2011). $r125\ ve\ r250$ grafları ise Michael Trick tarafından oluşturulan farklı random geometrik graflardır (Johnson ve Trick, 1996).

Leighton graflarında ise düğüm sayısı eşit ve 450'dir. $le450\_5a$ grafı bir Leighton grafıdır. Bu grafın ilk parametresi düğüm sayısını, ikinci parametresi ise bu graf için kromatik sayıyı göstermektedir (Leighton, 1979)

$qg.order$ grafları ise Carla Gomes tarafından oluşturulan Latin square (kare) problemi graflarıdır. Latin square probleminde, nxn boyutlu bir matriste herhangi bir kolon ya da satırda aynı renk tekrar etmemelidir (Gomes ve Shmoys, 2002).

mug^* grafları Kuzunori Mizuno tarafından oluşturulan graflardır (Mizuno ve Nishihara, 2008). Grafın karmaşıklığı değişmeyecek şekilde farklı sayıda düğüm ve kavis ile oluşturulmuşlardır. Ancak graf büyüdükçe graf için çözüm üretmek daha da zorlaşmaktadır (Ruiz, 2012).

Çizelge 4.1. Kullanılan DIMACS grafları ve özellikleri

Graf ismi	*V*	*E*	*Yoğunluk*	*Eniyi* $\chi(G)$	*Graf ismi*	*V*	*E*	*Yoğunluk*	*Eniyi* $\chi(G)$
1-Fullins_4	93	593	0,14	5	queen7_7	49	476	0,40	7
1-Fullins_5	282	3247	0,08	6	queen8_12	64	728	0,36	12
1-Insertions_4	67	232	0,10	5	queen8_8	96	1368	0,30	9
1-Insertions_5	202	1227	0,06	6	queen9_9	81	1056	0,33	10
1-Insertions_6	607	6337	0,03	7	queen10_10*	100	2940	0,59	11
2-Fullins_3	52	201	0,15	5	queen11_11	121	3960	0,55	11
2-Fullins_4	212	1621	0,07	6	queen12_12*	144	5192	0,50	13
2-Fullins_5	852	12201	0,03	7	queen13_13	169	6656	0,47	13
2-Insertions_4	149	541	0,05	5	queen14_14*	196	8372	0,44	16
2-Insertions_5	597	3936	0,02	6	queen15_15*	225	10360	0,41	16
3-Fullins_3	80	346	0,11	6	queen16_16*	256	12640	0,39	18
3-Fullins_4	405	3524	0,04	7	fpsol2.i1	496	11654	0,09	65
3-Fullins_5	2030	33751	0,02	8	fpsol2.i2	451	8691	0,09	30
3-Insertions_3	56	110	0,07	4	fpsol2.i3	425	8688	0,10	30
3-Insertions_4	281	1046	0,03	5	inithx.i1	864	18707	0,05	54
3-Insertions_5	1406	9695	0,01	6	inithx.i2	645	13979	0,07	31
4-Fullins_3	114	541	0,08	7	inithx.i3	621	13969	0,07	31
4-Fullins_4	690	6650	0,03	8	mulsol.i1	197	3925	0,20	49
4-Fullins_5	4146	77305	0,01	9	mulsol.i2	188	3885	0,22	31
4-Insertions_3	79	156	0,05	4	mulsol.i3	184	3916	0,23	31
4-Insertions_4	475	1795	0,02	5	mulsol.i4	185	3946	0,23	31
5-Fullins_3	154	792	0,07	8	mulsol.i5	186	3973	0,23	31
5-Fullins_4	1085	11395	0,02	9	zeroin.i1	211	4100	0,19	49
DSJC125.1*	125	736	0,09	5	zeroin.i2	211	3541	0,16	30
DSJC125.5*	125	3891	0,50	17	zeroin.i3	206	3540	0,17	30
DSJC125.9*	125	6961	0,90	44	le450_5c	450	9803	0,10	5
DSJC250.1*	250	3218	0,10	8	le450_5d	450	9757	0,10	5
DSJC250.5*	250	15668	0,50	28	le450_25a	450	8260	0,08	25
DSJR500.1*	500	3555	0,03	12	le450_25b	450	8263	0,08	25
Jean	80	254	0,08	10	r125.1	125	209	0,03	5
Anna	138	493	0,05	11	r125.1c	125	7501	0,97	46
david	87	406	0,11	11	r125.5	125	3838	0,50	36
homer	561	1629	0,01	13	r250.1	250	867	0,03	8
Huck	74	301	0,11	11	r250.1c	250	30227	0,97	64
games120	120	638	0,09	9	will199GPIA	701	7065	0,03	7
miles250	128	387	0,05	8	qg.order30	900	26100	0,06	30
miles500	128	1170	0,14	20	qg.order40	1600	62400	0,05	40
miles750	128	2113	0,26	31	mug100_1	100	166	0,03	4
miles1000	128	3216	0,40	42	mug100_25	100	166	0,03	4
miles1500	128	5198	0,64	73	mug88_1	88	146	0,04	4
myciel3	11	23	0,42	4	mug88_25	88	146	0,04	4
myciel4	20	71	0,37	5	abb313GPIA	1557	65390	0,05	9
myciel5	47	236	0,22	6	ash331GPIA	662	4185	0,02	4
myciel6	95	755	0,17	7	ash608GPIA	1216	7844	0,01	4
myciel7	191	2360	0,13	8	ash958GPIA	1916	12506	0,01	4
queen5_5	25	160	0,53	5	school1_nsh	352	14612	0,24	14
queen6_6	36	290	0,4	7	school1	385	19095	0,26	14

*kromatik sayı bilinmemektedir.

abb^*, ash^* *ve* $will^*$ grafları (Hossain ve Steihaug, 2008) tarafından sparse Jacobian matrislerini belirlemek için oluşturulan graflardır (Hossain ve Steihaug, 2008).

Yazmaç özgüleme problemi, bir gerçek dünya problemidir. Bu problemde amaç, hafızayı en etkin şekilde kullanmaktır. Bunu yapmak için de aynı bellek alanına ihtiyaç duymayan programlar için aynı register kullanılmaktadır. $fpsol^*, inithx^*, zeroin^*, mulsol^*$ grafları Gary Lewandowski tarafından oluşturulan yazmaç özgüleme problemi graflarıdır (Johnson ve Trick, 1996).

$school1$ ve $school1\_nsh$ grafları Gary Lewandowski tarafından oluşturulan sınıf çizelgeleme problemi graflarıdır (Johnson ve Trick, 1996).

Çizelge 4.1'de bu çalışma kapsamında kullanılan Benchmark grafları verilmiştir. Çizelge 4.1'in birinci sütununda grafın ismi, ikinci sütununda düğüm sayısı (V), üçüncü sütunda kavis sayısı (E), dördüncü sütunda kavis yoğunlukları, beşinci sütunda ise kromatik sayı ya da o graf için bilinen en iyi sonuç değerleri ($Eniyi/\chi(G)$) verilmiştir.

4.2. Önerilen Algoritmanın Önemi

Popülasyon tabanlı algoritmalarda, algoritmaların başarısı popülasyondaki bireylere bağladır. KSA da popülasyon tabanlı bir algoritma olduğu için başlangıç popülasyonu oldukça önemlidir. Çizelge 4.1'de bu çalışma kapsamında kullanılan DIMACS grafları verilmiştir. Bu grafların düğüm sayıları birbirinde farklıdır. En az düğüm sayısına sahip graf 11 düğümden oluşan $myciel3$ grafıdır, en çok düğüm sayısına sahip graf ise 4146 düğümlü $4\_Fullins\_5$ grafıdır. Bundan dolayı her graf için oluşturulacak başlangıç popülasyonu da farklılık gösterecektir. Popülasyondaki bir bireyin toplam uzunluğu (gen sayısı), üzerinde işlem yapılan grafın düğüm sayısı kadardır. Bir bireyin bir genine verilen değer ise o gen (düğüm) için renk bilgisini içerir. Ayrıca bir bireydeki genler birbirinden bağımsız değildir. Bir genin renk değerinin değiştirilmesi, o bireydeki diğer genleri de etkilemektedir. Gende meydana gelen değişim bireyi olumlu etkileyebileceği gibi olumsuz da etkileyebilir. Şekil 4.2'de bir gende meydana gelen değişim ile ilgili bir örnek verilmiştir.

İlk durum	1	3	5	4	2	1	5	3	2	1

Son durum	1	3	2	4	2	1	5	3	2	1

Şekil 4.2. Bir bireyin bir geninde meydana gelen değişim

Şekil 4.2'de verilen bireyin üçüncü gen değeri değiştirilmiştir. Genlerin sahip olduğu değerler, ilgili düğümün rengini ifade etmektedir. Grafın 3. düğümü ile 9. düğümünün komşu olduğu kabul edilirse, genlerde meydana gelen değişim bireyi olumsuz etkilemiştir. Bundan dolayı başlangıç popülasyonu oluşturulurken üzerinde işlem yapılacak graf için başlangıç renk aralığının ne olacağı, bireyler arasında bilgi değişimi sırasında nasıl bir yöntem uygulanacağı gibi sorunlar ortaya çıkmaktadır.

GBP'de düzgün renklendirme kuralına göre bir çözüm bulunması gerekir. Ayrıca bu çözüme ulaşmak için kullanılan renk sayısının da olabildiğince az olması gerekmektedir. Saf KSA literatürde verilen şekli ile GBP'ye uygulandığında, Saf KSA'nın GBP için uygun çözümler oluşturmadığı gözlemlenmiştir. Bundan dolayı Saf KSA'da bazı değişiklikler yapılmasına ihtiyaç duyulmuştur. Ayrıca tez kapsamında önerilen algoritmada başlangıç popülasyonun oluşturulması aşamasında, ilgili graf için kullanılacak renk aralığının belirlenmesi için de Leighton'nun literatüre kazandırmış olduğu RLF algoritması kullanılmıştır. RLF, GBP için geliştirilmiş olan dinamik bir graf boyama algoritmasıdır. RLF ve benzeri algoritmalar bir grafı mümkün olan en az sayıda renk kullanarak boyamayı garanti etmezler, ancak hızlı bir şekilde bir çözüm bulmayı garanti ederler. GBP için literatürde önerilen birçok metasezgisel algoritma açgözlü algoritmaların avantajlarından faydalanmıştır.

Saf KSA'da başlangıç popülasyonu, düğüm sayısı kadar renk ile oluşturulduğunda, tüm bireyler düzgün renklendirme kuralına göre oluşturulmuş olmaktadır. Algoritma başlangıçta kullanılan renk sayısını azaltamamıştır. Ancak saf KSA'nın başlangıç popülasyonu, açgözlü algoritmanın bulduğu renk sayısı kullanılarak oluşturulduğunda, Saf KSA'nın sadece bazı graflar için en iyi çözümleri bulduğu gözlemlenmiştir. Saf KSA'da bireyler arasında gerçekleşen bilgi alışverişinin hatalı boyanan düğümleri azalttığı, ancak düzgün renklendirme kuralına göre birçok graf için iyi çözümler bulamadığı gözlemlenmiştir. Başlangıç popülasyonu üst sınır renk sayısı ile oluşturulmuş Saf KSA'ya ait sonuçlar Çizelge 4.2'de verilmiştir. İterasyon sayısı 3000 olarak alınmıştır.

Çizelge 4.2 incelendiğinde başlangıç popülasyonu üst sınır renk kullanılarak oluşturulmasına rağmen saf KSA'nın yanlış boyanan düğümleri azaltmakta zorlandığı görülmektedir. Saf KSA sadece düğüm sayısı az olan Benchmark grafları için kromatik sayılara ulaşmıştır. Başlangıç popülasyonu düğüm sayısı kadar renk ile oluşturulmuş saf KSA'nın renk sayısını hiç azaltmadığı gözlemlenmiştir.

Çizelge 4.2. Saf KSA - üst sınır için elde edilen sonuçlar

Graf	Eniyi/χ(G)	Sonuç	İterasyon
1-Fullins_4	5	**5**	2397
1-Insertions_4	5	**5**	1644
2-Fullins_3	5	**5**	36
3-Fullins_3	6	**6**	876
3-Insertions_3	4	4	51
4-Insertions_3	4	4	264
myciel3	4	**4**	1
myciel4	5	**5**	12
myciel5	6	**6**	126
queen5_5	5	**5**	103
r125.1	5	**5**	1915
mug100_1	4	**4**	175
mug100_25	4	**4**	276
mug88_1	4	**4**	151

NOT: Diğer Benchmark grafları için çözüm bulunamamıştır.

KSA'nın GBP için uygulanabilir bir yöntem olabilmesi için saf KSA'da bazı değişiklikler yapılmıştır. Öncelikle KSA'nın yerel arama yapısında bazı değişikliklere gidilerek KSA_GA olarak adlandırılan bir yöntem önerilmiştir. KSA'nın yerel arama mekanizmasında alt mempleksteki en kötü kurbağanın güncellenmesi sırasında genetik algoritmanın çaprazlama ve mutasyon operatörlerinden faydalanılmıştır. Ayrıca memetik evrim sonunda mempleks içindeki en iyi kurbağaya komşuluk arama algoritması olan DKA'nın Chain yöntemi uygulanmıştır. KSA_GA, başlangıç popülasyonu düğüm sayısı kadar renk ile oluşturulmuş ve üst sınır renk sayısı ile oluşturulmuş olarak graflar üzerine uygulanmıştır. Yapılan testlerde, başlangıç popülasyonu düğüm sayısı kadar renk ile oluşturulmuş KSA_GA'nın renk sayısını oldukça azalttığı gözlemlenmiştir. Ancak başlangıç popülasyonu düğüm sayısı kadar renk kullanılarak oluşturulduğu için, yanlış boyanan düğüm olmamıştır ve Chain yönteminden faydalanılamamıştır. Chain metodu hatalı boyanan düğüm sayısını azaltmaya yönelik bir çalışmadır. Üzerinde işlem yaptığı bireyin bazı genlerinin değerlerini (düğümlerin renklerini), düğümlerin komşuluklarını da dikkate alarak değiştirmektedir. Hatalı boyanan düğüm olmadığı için algoritma sadece kullanılan renk sayısını azaltmaya çalışmıştır. Sonuç olarak, iterasyonlar ilerledikçe algoritmanın renk sayısını azaltmakta zorlandığı gözlemlenmiştir. Başlangıç popülasyonu açgözlü algoritmanın bulduğu üst sınır renk sayısı kullanılarak oluşturulduğunda hatalı boyanan düğüm oldukça fazla olduğundan KSA_GA, Chain yöntemini etkin bir şekilde kullanmış ve oldukça iyi çözümlere ulaşmıştır.

Çizelge 4.3. KSA_GA için elde edilen sonuçlar

Graf ismi	*Eniyi* $/\chi(G)$	KSA_GA düğüm sayısı		KSA_GA üst sınır	
		S	İter	S	İter
1-Fullins_4	5	**7**	2095	**5**	2
1-Fullins_5	6	24	1080	**6**	9
1-Insertions_4	5	**5**	2883	**5**	1
1-Insertions_5	6	17	1911	**6**	1
1-Insertions_6	7	46	1928	**7**	6
2-Fullins_3	5	**5**	874	**5**	1
2-Fullins_4	6	18	1635	**6**	2
2-Fullins_5	7	64	2171	**7**	38
2-Insertions_4	5	12	620	**5**	1
2-Insertions_5	6	41	2926	**6**	1
3-Fullins_3	6	**7**	2087	**6**	1
3-Fullins_4	7	30	2608	**7**	2
3-Fullins_5	8	140	2575	**8**	66
3-Insertions_3	4	**5**	587	**4**	1
3-Insertions_4	5	21	1432	**5**	1
3-Insertions_5	6	97	1645	**6**	2
4-Fullins_3	7	10	825	**7**	1
4-Fullins_4	8	49	2547	**8**	2
4-Fullins_5	9	281	1967	**9**	118
4-Insertions_3	4	7	672	**4**	1
4-Insertions_4	5	31	2449	**5**	1
5-Fullins_3	8	12	1982	**8**	1
5-Fullins_4	9	98	2468	**9**	2
DSJC125.1*	5	12	1096	6	3
DSJC125.5*	17	21	2465	20	1696
DSJC125.9*	44	**44**	596	**44**	152
DSJC250.1*	8	24	2852	10	5
DSJC250.5*	28	46	450	35	12
DSJR500.1*	12	38	1369	**12**	2
Jean	10	**10**	353	**10**	1
Anna	11	12	2310	**11**	1
david	11	**11**	795	**11**	1
homer	13	35	2947	**13**	1
huck	11	**11**	22	**11**	1
games120	9	11	772	**9**	1
miles250	8	10	796	**8**	1
miles500	20	**20**	223	**20**	1
miles750	31	**31**	112	**31**	2
miles1000	42	**42**	38	**42**	3
miles1500	73	**73**	5	**73**	1
myciel3	4	**4**	1	**4**	1
myciel4	5	**5**	4	**5**	1
myciel5	6	**6**	292	**6**	1
myciel6	7	9	2664	**7**	1
myciel7	8	18	1291	**8**	1
queen5_5	5	**5**	10	**5**	8
queen6_6	7	**7**	1714	**7**	28

Graf ismi	*Eniyi* $/\chi(G)$	KSA_GA düğüm sayısı		KSA_GA üst sınır	
		S	İter	S	İter
queen7_7	7	**7**	700	**7**	73
queen8_12	12	16	444	13	76
queen8_8	9	**11**	562	11	2
queen9_9	10	13	1707	12	4
queen10_10*	11	16	555	13	1
queen11_11	11	18	1510	14	2
queen12_12*	13	19	2941	15	3
queen13_13	13	23	2948	16	4
queen14_14*	16	26	1730	17	5
queen15_15*	16	29	2381	18	7
queen16_16*	18	31	2091	19	8
fpsol2.i1	65	69	1233	**65**	2
fpsol2.i2	30	47	2848	**30**	2
fpsol2.i3	30	46	2199	**30**	2
inithx.i1	54	79	2249	**54**	2
inithx.i2	31	60	2792	**31**	2
inithx.i3	31	59	1999	**31**	1
mulsol.i1	49	**49**	38	**49**	1
mulsol.i2	31	32	2512	**31**	2
mulsol.i3	31	33	514	**31**	1
mulsol.i4	31	35	2302	**31**	1
mulsol.i5	31	35	167	**31**	1
zeroin.i1	49	**49**	405	**49**	1
zeroin.i2	30	33	2352	**30**	1
zeroin.i3	30	31	1802	**30**	1
le450_5c	5	43	2641	6	252
le450_5d	5	43	2314	6	258
le450_25a	25	44	1648	**25**	14
le450_25b	25	45	1506	**25**	8
r125.1	5	8	1082	**5**	1
r125.1c	46	**46**	86	**46**	27
r125.5	36	**38**	1364	37	1254
r250.1	8	18	2039	**8**	1
r250.1c	64	**64**	474	**64**	152
will199GPIA	7	53	1597	**7**	40
qg.order30	30	79	2808	**30**	11
qg.order40	40	136	2397	**40**	19
mug100_1	4	7	1318	**4**	1
mug100_25	4	7	1415	**4**	1
mug88_1	4	6	458	**4**	1
mug88_25	4	6	2654	**4**	1
abb313GPIA	9	122	2814	11	232
ash331GPIA	4	48	2050	**4**	84
ash608GPIA	4	80	2994	5	15
ash958GPIA	4	124	2717	5	34
school1_nsh	14	49	1124	21	96
school1	14	52	1412	23	116

S: Sonuç, İter: İterasyon

Çizelge 4.3 incelendiğinde başlangıç popülasyonu düğüm sayısı kadar renk kullanılarak oluşturulduğunda da KSA_GA'nın oldukça iyi çözümlere ulaştığı

söylenebilir. Ancak başlangıç popülasyonu düğüm sayısı kadar renk kullanılarak oluşturulduğu için, yanlış boyanan düğüm olmamıştır. Renk azaltma işlemini GA'nın çaprazlama ve mutasyon operatörleri yapmıştır. Chain yöntemi kullanılmadığı için KSA_GA sonuca ulaşırken oldukça zorlanmış ve çok fazla iterasyona ihtiyaç duymuştur. Başlangıç popülasyonu açgözlü algoritmanın bulduğu renk sayısı üst sınır olarak kabul edilerek oluşturulduğunda KSA_GA'nın birçok graf için kromatik sayılara oldukça hızlı ulaştığı görülmektedir. KSA_GA, hem çaprazlama ve mutasyonu hem de Chain yöntemini etkin olarak kullandığı için çözüme hızlı gitmiştir. Hem GA'nın çaprazlama ve mutasyon operatörleri hem de Chain yöntemi grafta hatalı boyanan düğüm olduğu sürece etkin bir şekilde fayda sağlamaktadırlar. Ancak KSA_GA'nın genel olarak $queen, ash^{*}, school1, school1\_nsh \; ve \; random$ graflar için kromatik sayılara ulaşmadığı görülmektedir. Hem çaprazlama ve mutasyon operatörlerinin hem de Chain yönteminin etkinliğini arttırmak için geliştirilen KSA_GA'nın üzerinde bazı değişiklikler yapılarak bu tez kapsamında modifiye KSA (MKSA) yöntemi önerilmiştir. MKSA'da, KSA_GA'dan farklı olarak popülasyon üzerinde de bazı değişiklikler yapılmıştır.

Önerilen algoritma olan MKSA'da başlangıç popülasyonunda bireyler açgözlü algoritmanın bulduğu üst sınır renk sayısı kullanılarak oluşturulmaktadır. Örneğin Şekil 3.42'de verilen 10 düğümlü grafı, RLF beş renk kullanarak boyamıştır. Şekil 3.42'de verilen bu graf için başlangıç popülasyonu oluşturulurken üst sınır renk olarak RLF'nin bulduğu sonuç kullanılmıştır. Yani bireyler oluşturulurken renk aralığı olarak [1,üst sınır] kullanılmıştır. Şekil 4.3'te, Şekil 3.42'de verilen 10 düğümlü graf için [1,üst sınır] aralığında olmak şartı ile rastgele oluşturulan bir birey verilmiştir.

Birey	1	3	2	4	2	1	5	3	2	1

Şekil 4.3. Sınır değer aralıklarında rastgele oluşturulan bir birey

Önerilen algoritmada (Modifiye KSA – MKSA) başlangıç popülasyonundaki bireyler Şekil 4.3'te verilen bireye benzer şekilde oluşturulmaktadır. Algoritmanın çalışması esnasında, düzgün boyama kuralına uygun olarak sonuca ulaşıldığında, bu sonuç o ana kadarki en iyi çözüm olarak kaydedilmekte ve üst sınır renk sayısı;

$$\text{üst sınır} = \text{üst sınır} - 1$$

olacak şekilde güncellenmektedir. Daha sonra popülasyondaki bireyler kontrol edilerek, bireylerde kullanılan üst sınır renk, son duruma göre güncellenmektedir. Örneğin Şekil

4.3'te üst sınır renk beştir. Eğer algoritma bu birey için düzgün boyama kuralına uygun olarak bir çözüme ulaşırsa, beşinci renkle boyanmış düğümlerin (genlerin) rengi silinmekte, bunun yerine yeni üst sınır olan dördüncü renk verilmektedir. Böylece algoritma düzgün renklendirme kuralına uygun olarak dört renkli bir çözüm arayışına girmektedir. Sonlandırma kriteri sağlanıncaya kadar, her çözüme ulaşıldığında, üst sınır renk sayısı bir azaltılarak algoritma çalışmaya devam etmektedir. Böylece hem GA'nın çaprazlama ve mutasyon operatörleri hem de Chain komşuluk araması daha etkin kullanılmış olmaktadır. Çünkü Chain metodu eğer bireyde yanlış boyanan düğüm varsa çalışmaktadır. Algoritma düzgün bir çözüme ulaştığı anda renk sayısının azaltılması sadece GA'nın çaprazlama ve mutasyon operatörleri ile yapıldığında, daha az renk kullanarak çözüm bulmak zorlaşmaktadır. Bundan dolayı $üst\ sınır$ renk sayısının algoritma içinde azaltılması algoritmanın daha etkin kullanılmasını sağlamaktadır. Popülasyon korunup, bireylerden sadece üst sınır renk değerine sahip genlerin bilgilerinin güncellenmesi işlemi yapıldığından dolayı, popülasyonun o ana kadar kazanmış olduğu tecrübe korunmaktadır. Böylece çözüme gitmek hızlanmaktadır. MKSA, geliştirilen KSA_GA algoritmasının repopülasyon üretilen en gelişmiş şeklidir. Böylelikle KSA_GA algoritması kapsamında, eğer aç gözlü algoritmadan (RLF) elde edilen sonuca göre üretilen popülasyon ile sonuca ulaşıldıysa, renk sayısı 1 azaltılarak, daha iyi bir çözüm olup olmadığı araştırılmıştır. Böylelikle hem kromatik sayıyı bulmak daha hızlı olmuş, hem de literatürde kromatik sayısı bilinmeyen graflar için bilinen çözümlerden daha iyi çözüm araması yapılabilmiştir. Şekil 4.4(a)'da düzgün boyama kuralına göre ulaşılmış bir çözüm verilmiştir. Şekil 4.4(b)'de ise bu bireydeki üst sınır renk değerine sahip genlerin değerleri güncellenmiştir. Güncelleme işlemi ile birlikte bireyde hatalı boyanmış düğümler tekrar oluşmaktadır.

Düzgün birey	4	3	1	3	2	3	3	5	1	2

(a) Algoritmanın düzgün boyama kuralına uygun bulduğu çözüm

Bireyin yeni durumu	4	3	1	3	2	3	3	4	1	2

(b) Yanlış boyanan düğümlerin bulunduğu yeni durum

Şekil 4.4. Üst sınır değerine göre bireydeki değişim

Önerilen algoritmada hem açgözlü algoritmanın hem de Saf KSA'nın avantajlarından faydalanılmıştır. Ayrıca genetik algoritmanın çaprazlama ve mutasyon

operatörlerinden de faydalanılmıştır. Algoritmanın başarısını arttıran bir diğer özellik ise algoritmanın çalışması esnasında repopülasyon işleminin yapılmasıdır. Ayrıca komşuluk arama algoritması olan DKA'nın Chain yönteminin de algoritma içinde kullanılması sayesinde düğümlerin renkleri komşuluklar dikkate alarak değiştirilmekte ve sonuca yakınsama hızlanmaktadır.

4.3. Önerilen Algoritmanın Parametrelerinin Belirlenmesi

Parametreler belirlenirken KSA için literatürdeki çalışmalarda sıklıkla kullanılan değerler kullanılmıştır. Önerilen MKSA algoritmasında memetik evrim aşamasında en kötü kurbağanın güncellemesi sırasında, mutasyon işlemine tabi tutulacak genin mutasyona uğrayıp uğramama durumu için farklı seçenekler kullanılmıştır. Farklı mutasyon değerleri için algoritmanın başarı durumları çizelgelerle verilmiştir. Böylece farklı mutasyon değerlerinde algoritmanın yakınsama durumları gözlemlenmiştir. Önerilen algoritma için parametreler Çizelge 4.4'te verilmiştir.

Çizelge 4.4 Önerilen algoritma için parametreler

Parametre	Açıklama	Değer
F	Popülasyon sayısı	50
m	Mempleks sayısı	5
n	Memleksteki kurbağa sayısı	10
q	Alt memplekstekı kurbağa sayısı	$n * 0.5$
N	Memtik evrim döngü sayısı	50
I	Algoritma iterasyon sayısı	3000
R	Bir gen için mutasyon oranı	$[0.10, 0.25, 0.50, 0.75, 1]$
C	Chain metodu için döngü sayısı	20

Çizelge 4.4'te, önerilen algoritma için bu çalışmada kullanılan parametreler verilmiştir. Bu parametrelerden sadece R parametresi için beş farklı değer kullanılmıştır. Böylece farklı mutasyon değerlerinde algoritmanın sonuca ulaşmadaki etkisi gözlemlenmiştir. Algoritma her graf için 20 kez çalıştırılmıştır. 20 çalışma sonunda farklı mutasyon değerleri için bulunan sonuçlar Bölüm 4.4'te çizelgelerle verilmiştir. Önerilen algoritma bazı Benchmark graflarında her mutasyon oranı için ayrı ayrı yapılan 20 çalışmanın (run) tamamında aynı sonuca ulaşmıştır. Ancak önerilen algoritma bazı Benchmark graflarında her mutasyon oranı için ayrı ayrı yapılan 20 çalışmanın (run) tamamında aynı sonuca ulaşamamıştır. Bundan dolayı bu farklı sonuçlar için ayrıca tablolar verilmiştir. Böylece algoritmanın her mutasyon oranı için

ayrı ayrı yapılan 20 çalışma için bulduğu sonuçlar ve bu sonuçlara kaç kez ulaşıldığı gözlemlenebilmektedir.

4.4. Farklı Mutasyon Oranları için Sonuçlar

Her graf sınıfı için algoritma sonuçları ayrı ayrı verilmiştir. Çizelge 4.5'te *Myciel* grafları için sonuçlar verilmiştir. Her mutasyon değeri için algoritma 20'şer kez çalıştırılmıştır.

Çizelge 4.5. Myciel grafları için sonuçlar

Graf Adı	Eniyi/χ(G)	Mutasyon Oranına Göre Sonuca Ulaşılan İterasyon Değerleri					Çalışma Sayısı (20)
		0.10	0.25	0.50	0.75	1.0	
myciel3	4	1	1	1	1	1	20/20
myciel4	5	2	1	1	1	1	20/20
myciel5	6	5	2	1	1	1	20/20
myciel6	7	13	7	3	1	1	20/20
myciel7	8	7	5	4	3	1	20/20

Myciel graflarında düğüm sayısı az olduğu için algoritma her mutasyon değeri için kromatik sayıya ulaşmıştır. Çizelge 4.5'te algoritmanın 20 çalışma sonunda çözüme ulaştığı en iyi iterasyon değerleri verilmiştir. Çizelge 4.5 incelendiğinde *myciel*7 grafı düğüm sayısı en fazla olan *Myciel* grafıdır. Bu graf için bulunan sonuçlara göre mutasyon oranının yüksek olması, graf için çözüm bulmayı hızlandırmıştır. Bundan dolayı düğüm sayısının artması durumunda mutasyon oranının yüksek seçilmesi sonuca yakınsamayı hızlandırabilmektedir.

Çizelge 4.6. Books grafları ile games120 grafı için sonuçlar

Graf Adı		Eniyi/χ(G)	Mutasyon Oranına Göre Sonuca Ulaşılan İterasyon Değerleri					Çalışma Sayısı (20)
			0.10	0.25	0.50	0.75	1.0	
Books Grafları	Jean	10	4	2	1	1	1	20/20
	Anna	11	2	1	1	1	1	20/20
	David	11	9	3	2	1	1	20/20
	Homer	13	4	3	1	1	1	20/20
	Huck	11	4	1	1	1	1	20/20
Games Grafı	Games120	9	1	1	1	1	1	20/20

Önerilen algoritma Çizelge 4.6'da verilen grafların hepsi için en iyi çözüme hızlı bir şekilde ulaşmıştır. Books graflarından *homer* grafının düğüm sayısı 561'tir. Buna

rağmen önerilen algoritma 0.10 ve 0.25 mutasyon oranı dışındaki mutasyon oranlarında birinci iterasyonun sonunda çözüme ulaşmıştır.

Çizelge 4.7'de Donald Knuth tarafından oluşturulan Miles grafları için önerilen algoritmanın bulduğu sonuçlar verilmiştir. Miles graflarında düğüm sayısı 128'tir. Ancak her graf için kavis sayısı ve yoğunlukları farklıdır. Miles graflarının kromatik sayıları bilinmektedir. Ayrıca Miles grafları çözülmesi kolay graflardır. Önerilen algoritma farklı mutasyon oranlarında hızlı bir şekilde sonuca ulaşabilmiştir. Algoritma farklı mutasyon oranlarında 20'şer kez çalıştırılmış, bulunan en iyi iterasyon değerleri Çizelge 4.7'de verilmiştir.

Çizelge 4.7. Miles grafları için sonuçlar

Graf Adı	Eniyi/χ(G)	Mutasyon Oranına Göre Sonuca Ulaşılan İterasyon Değerleri					Çalışma Sayısı (20)
		0.10	0.25	0.50	0.75	1.0	
miles250	8	1	1	1	1	1	20/20
miles500	20	6	1	1	1	1	20/20
miles750	31	12	3	4	2	2	20/20
miles1000	42	15	6	4	4	3	20/20
miles1500	73	4	2	1	1	1	20/20

Çizelge 4.8'de $queen$ grafları için bulunan sonuçlar verilmiştir. Önerilen algoritma $queen5\_5, 6\_6, 7\_7, 8\_8, 9\_9$ ve 8_12 grafları için kromatik sayı değerlerine ulaşmıştır. $queen10\_10,14\_14$ ve 16_16 için ise literatürde bilinen en iyi sonuçlara ulaşmıştır. Ancak diğer $queen$ grafları için $Eniyi/\chi(G)$ değerlerine ulaşılmamıştır. Algoritma farklı mutasyon değerleri için ayrı ayrı 20 kez çalıştırılmış ve bulunan en iyi iterasyonlar Çizelge 4.8'de verilmiştir. Çizelge 4.8 incelendiğinde mutasyon oranının arttırılmasının sonuca ulaşmayı hızlandırdığı görülmektedir. Çünkü mutasyona uğrayan gen genellikle çözümü iyileştirmektedir. Böylece mutasyon işlemi sonunda hatalı boyanan düğüm sayısı genellikle azalmaktadır. Önerilen algoritma farklı mutasyon değerlerinde $queen6\_6, 7\_7,$ $8\_8, 9\_9, 10\_10$ ve 16_16 dışındaki graflar için 20 çalıştırmanın tamamında aynı sonuçlara ulaşmıştır. Ancak bu graflar için mutasyon oranının değiştirilmesi başarı sayısını ve ulaşılan renk sayısını etkilemiştir. Çizelge 4.9'da bu graflar için sonuçlar detaylandırılmıştır.

Çizelge 4.8. *queen* grafları için sonuçlar

Graf Adı	Eniyi/χ(G)	MKSA	Mutasyon Oranına Göre Sonuca Ulaşılan İterasyon Değerleri					Çalışma Sayısı (20)
			0.10	0.25	0.50	0.75	1.0	
queen5_5	5	**5**	6	6	5	6	8	20/20
queen6_6	7	**7**	4	5	12	4	6	--
queen7_7	7	**7**	4	9	12	14	9	--
queen8_12	12	**12**	5	4	3	3	4	20/20
queen8_8	9	**9**	8	12	6	12	12	--
queen9_9	10	**10**	20	15	10	18	20	--
queen10_10*	11	**11**	340	112	2906	691	175	--
queen11_11	11	13	7	5	6	6	5	20/20
queen12_12*	13	14	10	10	9	7	11	20/20
queen13_13	13	15	19	13	11	13	13	20/20
queen14_14*	16	**16**	27	16	21	18	15	20/20
queen15_15*	16	17	26	24	26	22	40	20/20
queen16_16*	18	**18**	35	42	38	34	35	--

-- 20 çalışma sonunda ulaşılan renk sayısı ile bu renk sayılarına ulaşma sıklıkları farklılık gösterdiği için bu graflar için sonuçlar Çizelge 4.9'da verilmiştir.

Çizelge 4.9 incelendiğinde mutasyon oranının $0.75\ ve\ 1.0$ seçilmesi $queen6\_6$ grafı için 20 çalıştırmanın tamamında $\chi(G)$'e ulaşıldığını görülmektedir. $queen7\_7$ grafı için ise MKSA $\chi(G), \chi(G)+1$ ve $\chi(G)+2$ sonuçlarına ulaşmıştır. $queen8\_8$ grafı için MKSA grafı 9 renk, $queen9\_9$ için ise 10 renk kullanarak boyayabilmiştir. MKSA $queen10\_10$ grafı için bilinen en iyi değer olan 11 renge ulaşabilmiştir. Ancak Çizelge 4.9 incelendiğinde bu sonuca ulaşma sıklığının düşük olduğu görülmektedir. MKSA'nın $queen10\_10$ grafı için 11 renk değerine mutasyon oranı 1.0 seçildiğinde 5 kez ulaştığı görülmektedir. $queen10\_10$ grafı için en iyi mutasyon oranın 1.0 olduğu görülmektedir. $queen16\_16$ grafı için mutasyon oranı $0.10\ ya\ da\ 0.25$ seçildiğinde MKSA 2 kez 19 renk değerine 18 kez de 18 renk değerine ulaşmıştır. Diğer mutasyon oranları için MKSA'nın 20 çalışmanın tamamında bilinen en iyi değer olan 18 renge ulaştığı görülmektedir.

Çizelge 4.9. *queen* graflarında farklı sonuçlar için başarı sıklıkları

Graf Adı	MKSA	Mutasyon Oranına Göre 20 Çalışma (run) Sonunda Bulunan Sonuç ve Sonuca Ulaşma Sıklıkları				
		0.10	0.25	0.50	0.75	1.0
queen6_6	7	19	18	19	20	20
	8	1	2	1	X	X
queen7_7	7	9	8	7	12	11
	8	9	12	13	8	8
	9	2	X	X	X	1
queen8_8	9	15	13	16	16	17
	10	5	7	4	4	3
queen9_9	10	11	12	13	13	17
	11	9	8	7	7	3
queen10_10*	11	1	3	1	1	5
	12	19	17	19	19	15
queen16_16*	18	18	18	20	20	20
	19	2	2	X	X	X

X: 20 çalışma içinde hiç görülmedi.

Çizelge 4.10'da Leighton grafları için önerilen algoritmanın bulduğu sonuçlar verilmiştir. Leighton graflarında düğüm sayıları eşit ve 450'dir. Ancak bu grafların kavis yoğunlukları farklıdır. Leighton grafları çözülmesi zor graf olarak tanımlanırlar. Önerilen algoritma $le450\_25a\ ve\ le450\_25b$ grafları için $\chi(G)$ sonuçlarına her mutasyon oranı için ulaşmıştır. Ayrıca $le450\_5c\ ve\ le450\_5d$ grafı için tahmin aşamasında bulunan üst sınır renk ile oluşturulmuş popülasyonu optimize ederken zorlanmıştır. Her 2 graf için farklı mutasyon oranlarında sonuca ulaşma sayıları farklılık göstermiştir. Bundan dolayı popülasyon, açgözlü algoritmanın bulduğu renk sayısı bir attırılarak yeniden oluşturulmuş ve algoritmanın farklı mutasyon oranlarında sonuca yakınsama sayıları Çizelge 4.10'da verilmiştir. Çizelge 4.11'de ise $le450\_5c\ ve\ le450\_5d$ grafları için farklı mutasyon oranları için 20 çalışma sonunda başarılı sonuca yakınsama sayıları verilmiştir Önerilen algoritmanın kavis bağlantı karmaşıklığı fazla olan graflar için çözüm bulmakta zorlandığı söylenebilir.

Çizelge 4.10. Leighton grafları için sonuçlar

Graf Adı	Eniyi/χ(G)	MKSA	Mutasyon Oranına Göre Sonuca Ulaşılan İterasyon Değerleri					Çalışma Sayısı (20)
			0.10	0.25	0.50	0.75	1.0	
le450_5c	5	6	273	317	251	241	252	--
le450_5d	5	6	275	272	274	230	258	--
le450_25a	25	**25**	14	24	15	16	14	20/20
le450_25b	25	**25**	10	9	5	7	8	20/20

--: Sonuca ulaşma sayıları farklılık göstermektedir.

$le450\_5c$ grafı için Çizelge 4.11 incelendiğinde 0.10 mutasyon oranında düzgün renklendirme kuralına göre 20 çalışmanın sadece 6 tanesinde sonuca ulaşılmıştır. Yani algoritma 14 kez düzgün renklendirme kuralına uygun bir çözüm bulamamıştır. Ancak mutasyon oranı 0.75 ya da 1.0 olarak seçildiğinde 20 çalışmanın tamamında algoritma grafı düzgün renklendirmiştir. $le450\_5d$ grafı içinde benzer bir durum söz konusudur.

Çizelge 4.11. le450_5c *ve* le450_5d için sonuca ulaşma sayıları

Graf Adı	Mutasyon Oranına Göre 20 Çalışma (run) Sonunda Sonuca Ulaşma Sıklıkları				
	0.10	0.25	0.50	0.75	1.0
le450_5c	6	11	15	20	20
le450_5d	5	8	17	18	18

Çizelge 4.12'de random graflar için önerilen algoritmanın bulduğu sonuçlar verilmiştir. $DSJC125.1$ *ve* DSJC125.9 grafları dışındaki graflar için MKSA bilinen en iyi sonuçlara ulaşamamıştır. Algoritmanın farklı mutasyon oranları için 20'şer kez çalıştırılması durumunda sonucun bulunduğu en iyi iterasyon değerleri Çizelge 4.12'de verilmiştir. Ancak algoritmanın farklı mutasyon değerleri için bulduğu renk sayıları ve bu renk sayılarına ulaşma sıklıkları farklılık gösterdiği için Çizelge 4.12'de bu graflar için bulunan sonuçlar detaylandırılmıştır.

Çizelge 4.12. Random grafları için sonuçlar

Graf Adı	Eniyi/χ(G)	MKSA	Mutasyon Oranına Göre Sonuca Ulaşılan İterasyon Değerleri					Çalışma Sayısı (20)
			0.10	0.25	0.50	0.75	1.0	
DSJC125.1*	5	5	2188	1254	1210	2434	632	--
DSJC125.5*	17	18	27	34	25	60	35	--
DSJC125.9*	44	44	42	34	56	57	322	--
DSJC250.1*	8	9	19	19	16	14	24	20/20
DSJC250.5*	28	31	257	101	137	73	69	--

--: Sonuca ulaşma sayıları farklılık göstermektedir.

Çizelge 4.13'de Random graflar için algoritmanın farklı mutasyon oranları için başarı sayıları ve buldukları sonuçlar verilmiştir. Random grafların kromatik sayıları bilinmemekte, bu graflar için literatürde bilinen en iyi sonuçlar bulunmaktadır. Çizelge 4.13'e göre MKSA'nın $DSJC125.1$ *ve* $DSJC125.9$ gafları için bilinen en iyi sonuçlara ulaştığı görülmektedir. Ancak MKSA'nın $DSJC125.1$ grafı için bilinen en iyi sonuç olan 5 değerine çok az ulaştığı görülmektedir. MKSA $DSJC125.1$ garfı için genellikle 6 renk değerine ulaşmıştır. $DSJC125.5$ grafı için MKSA 18 renk ve 19 renk değerlerine

ulaşmıştır. Mutasyon oranı 1.0 olarak seçildiğinde MKSA 18 kez 18 renk sonucuna ulaşmışken, 2 kez de 19 renk değerine ulaşmıştır. Diğer mutasyon oranları için de benzer bir durum söz konusudur. Ancak mutasyon oranı düştükçe MKSA'nın $DSJC125.5$ grafı için 18 renk değerine ulaşmakta zorluk çektiği ve başarı sayısının azaldığı görülmektedir. Mutasyon oranı 0.1 seçildiğinde Çizelge 4.13 incelendiğinde MKSA'nın $DSJC125.5$ grafı için 10 kez 18 renk değerine 10 kezde 19 renk değerine ulaştığı görülmektedir.

Çizelge 4.13. Random graflarda farklı sonuçlar için başarı sıklıkları

Graf Adı	MKSA	Mutasyon Oranına Göre 20 Çalışma (run) Sonunda Bulunan Sonuç ve Sonuca Ulaşma Sıklıkları				
		0.10	0.25	0.50	0.75	1.0
DSJC125.1*	5	2	3	1	1	5
	6	18	17	19	19	15
DSJC125.5*	18	10	13	15	12	18
	19	10	7	5	8	2
DSJC125.9*	44	7	10	10	9	4
	45	13	9	10	10	11
	46	X	1	X	1	5
DSJC250.5*	31	4	16	15	16	20
	32	16	4	5	4	X

X: 20 çalışma içinde hiç görülmedi.

Çizelge 4.13'e göre MKSA'nın $DSJC125.9$ grafı için üç farklı renk sayısı ile sonuç bulduğu görülmektedir. $DSJC125.9$ grafı için mutasyon oranı arttıkça bilinen en iyi sonuca ulaşma başarısının azaldığı gözlemlenmiştir. $DSJC250.5$ grafı için ise MKSA bilinen en iyi değer olan 28 renk değerine ulaşamamıştır. Mutasyon oranı 1.0 seçildiğinde MKSA 20 çalışmamın tamamında 31 renk değerine ulaşmıştır.

Çizelge 4.14'te random geometrik graflar için önerilen algoritmanın bulduğu sonuçlar verilmiştir. $r125.5\ ve\ r250.1c$ grafları dışındaki graflar için, farklı mutasyon oranları için önerilen algoritma 20 çalıştırmanın tamamında kromatik sayılara ulaşabilmiştir. Çizelge 4.14'de her mutasyon oranı için en iyi sonuca kaçıncı iterasyonda ulaşıldığı verilmiştir. $r125.5$ grafı için en iyi çözüme sadece mutasyon oranı 0.10 seçildiğinde ulaşılabilmiştir. Diğer mutasyon oranları için algoritmanın bulduğu en iyi çözüm $37\ ya\ da\ 38$ renk olmuştur. $r250.1c$ grafı için ise mutasyon oranı 1.0 seçildiğinde MKSA kromatik sayıya ulaşamamıştır. MKSA'nın $r250.1c$ grafı için bulduğu sonuçlar $65\ ya\ da\ 66$ renk olmuştur. Çizelge 4.15'te $r125.5\ ve\ r250.1c$ grafları için farklı mutasyon oranları için sonuçlar detaylandırılmıştır.

Çizelge 4.14. Random geometrik graflar için sonuçlar

Graf Adı	Eniyi/χ(G)	MKSA	Mutasyon Oranına Göre Sonuca Ulaşılan İterasyon Değerleri					Çalışma Sayısı (20)
			0.10	0.25	0.50	0.75	1.0	
r125.1	5	**5**	1	1	1	1	1	20/20
r125.1c	46	**46**	11	12	11	15	17	20/20
r125.5	36	**36**	1096	X	X	X	X	--
r250.1	8	**8**	2	42	1	1	1	20/20
r250.1c	64	**64**	75	69	72	78	X	--
DSJR500.1*	12	**12**	5	14	7	4	2	20/20

--: Algoritmanın bulduğu diğer sonuçlar çizelge 4.15'te verilmiştir.
X: En iyi sonuca ulaşamamıştır.

Çizelge 4.15 incelendiğinde MKSA'nın $r125.5$ grafı için farklı mutasyon oranlarında üç farklı renk sayısına ulaştığı görülmektedir. MKSA'nın kromatik sayıya sadece 0.1 mutasyon oranında ulaştığı gözlemlenmiştir. Farklı mutasyon oranları için MKSA genellikle 37 renk değerine ulaşmıştır. Ancak mutasyon oranın yükselmesi bu graf için başarı sayısını azaltmıştır. Çünkü mutasyon oranı 0.75 ya da 1.0 olarak seçildiğinde MKSA genellikle 38 renk değerine ulaşabilmiştir. $r250.1c$ grafında da mutasyon oranının yükselmesi başarıyı ciddi bir şekilde azaltmıştır. Elde edilen sonuçlar değerlendirildiğinde random geometrik graflar için düşük mutasyon oranının başarıyı arttırdığı söylenebilir.

Çizelge 4.15. Random graflarda farklı sonuçlar için başarı sıklıkları

Graf Adı	MKSA	Mutasyon Oranına Göre 20 Çalışma (run) Sonunda Bulunan Sonuç ve Sonuca Ulaşma Sıklıkları				
		0.10	0.25	0.50	0.75	1.0
r125.5	36	6	X	X	X	X
	37	14	19	18	11	4
	38	X	1	2	9	16
r250.1c	64	20	20	20	18	X
	65	X	X	X	X	5
	66	X	X	X	2	15

X: 20 çalışma içinde hiç görülmedi.

$Insertion^*$ ve $Fullins^*$ grafları Mycielski graflarının biraz daha zorlaştırılmış graflarıdır. Bu zorlaştırma işlemi grafın kavis yoğunluğu değişmemek şartı ile grafa yeni düğümler eklenerek yapılmıştır. Çizelge 4.16'da $Insertion^*$ ve $Fullins^*$ grafları için önerilen algoritmanın bulduğu sonuçlar verilmiştir. İ$nsertion^*$ ve $Fullins^*$ grafları için düğüm sayısı arttıkça önerilen algoritmanın çözüme ulaşmakta zorlandığı gözlemlenmiştir. Mutasyon oranı arttıkça MKSA'nın $Insertion^*$ ve $Fullins^*$grafları için daha hızlı çözüme gittiği gözlemlenmiştir. MKSA farklı mutasyon oranları için

20'şer kez çalıştırılmıştır. 4-$Fullins\_5$ dışındaki graflar için her mutasyon oranı için ayrı ayrı 20 çalışmanın tamamında başarıya ulaşılmıştır. Mutasyon oran 1.0 seçildiğinde çözüme yakınsamamın daha hızlı olduğu gözlemlenmiştir.

Çizelge 4.16. $Insertion^{*}$ *ve* $Fullins^{*}$ graflar için sonuçlar

Graf Adı	Eniyi/χ(G)	MKSA	Mutasyon Oranına Göre Sonuca Ulaşılan İterasyon Değerleri					Çalışma Sayısı (20)
			0.10	0.25	0.50	0.75	1.0	
1-Fullins_4	5	**5**	9	6	8	3	2	20/20
1-Fullins_5	6	**6**	12	11	12	11	9	20/20
1-Insertions_4	5	**5**	5	2	1	1	1	20/20
1-Insertions_5	6	**6**	4	3	1	1	1	20/20
1-Insertions_6	7	**7**	16	15	12	10	6	20/20
2-Fullins_3	5	**5**	4	3	1	1	1	20/20
2-Fullins_4	6	**6**	6	4	3	3	2	20/20
2-Fullins_5	7	**7**	36	36	39	36	38	20/20
2-Insertions_4	5	**5**	3	2	1	1	1	20/20
2-Insertions_5	6	**6**	11	6	3	2	1	20/20
3-Fullins_3	6	**6**	6	3	2	1	1	20/20
3-Fullins_4	7	**7**	9	6	4	3	2	20/20
3-Fullins_5	8	**8**	58	72	72	58	66	20/20
3-Insertions_3	4	**4**	3	2	1	1	1	20/20
3-Insertions_4	5	**5**	4	2	1	1	1	20/20
3-Insertions_5	6	**6**	23	12	6	3	2	20/20
4-Fullins_3	7	**7**	2	1	1	1	1	20/20
4-Fullins_4	8	**8**	12	8	7	3	2	20/20
4-Fullins_5	9	**9**	375	640	643	351	118	--
4-Insertions_3	4	**4**	4	2	1	1	1	20/20
4-Insertions_4	5	**5**	6	3	1	1	1	20/20
5-Fullins_3	8	**8**	3	2	1	1	1	20/20
5-Fullins_4	9	**9**	16	10	7	4	2	20/20

--: 4-Fullins_5 grafı için algoritmanın bulduğu sonuçlar çizelge 4.17'de verilmiştir.

Tahmin aşamasında kullanılan açgözlü algoritma, 4-$Fullins\_5$ grafı için önerilen algoritmanın başlangıç popülasyonu oluşturulurken kullanılacak renk aralığı için bir üst sınır bulmuştur. Bu üst sınır kısıtına göre başlangıç popülasyonu oluşturulmuştur. Mutasyon oranı 1.0 olduğu durumda algoritma 4-$Fullins\_5$ grafı için 20 çalışmanın tamamında başarıya ulaşmakla birlikte, diğer mutasyon oranlarına kıyasla daha çabuk bu sonuca ulaştığı gözlemlenmiştir. Ancak mutasyon oranı 1.0 dışında bir değer olarak belirlendiğinde MKSA'nın başarılı sonuç sayısı azalmıştır. Çizelge 4.17'de 4-$Fullins\_5$ grafı için farklı mutasyon oranlarında başarılı sonuca ulaşma sayıları verilmiştir.

Çizelge 4.17. 4-$Fullins\_5$ grafı için başarı sayıları

Graf Adı	MKSA	Mutasyon Oranına Göre 20 Çalışma (run) Sonunda Bulunan Sonuç ve Sonuca Ulaşma Sıklıkları				
		0.10	0.25	0.50	0.75	1.0
4-Fullins_5	9	9	9	10	12	20

Mug grafları çözülmesi kolay graflardır. Önerilen algoritma her mutasyon oranı için ayrı ayrı çalıştırılmıştır. Önerilen algoritmanın oldukça hızlı bir şekilde sonuca gittiği gözlemlenmiştir. Elde edilen sonuçlar Çizelge 4.18'de verilmiştir.

Çizelge 4.18. Mug grafları için sonuçlar

Graf Adı	Eniyi/χ(G)	MKSA	Mutasyon Oranına Göre Sonuca Ulaşılan İterasyon Değerleri					Çalışma Sayısı (20)
			0.10	0.25	0.50	0.75	1.0	
mug100_1	4	**4**	1	1	1	1	1	20/20
mug100_25	4	**4**	1	1	1	1	1	20/20
mug88_1	4	**4**	3	1	1	1	1	20/20
mug88_25	4	**4**	3	2	1	1	1	20/20

$qg.order30$ ve $qg.order40$ grafları Latin square problemi graflarıdır. Çizelge 4.19'da bu graflar için farklı mutasyon oranları için bulunan sonuçlar verilmiştir. Önerilen algoritma Latin square problemi grafları için en iyi sonuçlara hızlı bir şekilde ulaşabilmiştir.

Çizelge 4.19. Latin square grafları için sonuçlar

Graf Adı	Eniyi/χ(G)	MKSA	Mutasyon Oranına Göre Sonuca Ulaşılan İterasyon Değerleri					Çalışma Sayısı (20)
			0.10	0.25	0.50	0.75	1.0	
qg.order30	30	**30**	14	12	11	11	11	20/20
qg.order40	40	**40**	21	19	18	18	19	20/20

abb^*, ash^* ve $will^*$ grafları sparse Jacobian matrislerini belirlemek için parçalı sütunlarda matris bölme probleminden elde edilmişlerdir. abb^* ve ash^* grafları çözülmesi zor Benchmark graflarıdır. Bu graflarda düğüm sayısı oldukça fazladır. Bundan dolayı önerilen algoritma sonuca ulaşmak için oldukça zorlanmıştır. ash^* graflarının düğüm sayısı 500'ten fazladır ve bu graflar için kromatik sayısı 4'tür. Genetik algoritmanın çaprazlama ve mutasyon işlemleri hatalı boyanan düğümleri düzgün hale getirmek için tek başına yeterince etkin olamamıştır. Çalışmanın yapısında kullanılan komşuluk arama algoritması olan Chain metodu sayesinde algoritma başarılı sonuçlara ulaşabilmiştir. abb^*, ash^* ve $will^*$ grafları için bulunan sonuçlar Çizelge

4.20'de verilmiştir. Farklı mutasyon oranlarında önerilen algoritmanın $abb313GPIA$ grafı için bulabildiği en iyi değer 11 olabilmiştir. Çizelge 4.20'de farklı mutasyon oranlarında, sonucun bulunduğu en iyi iterasyon değerleri verilmiştir. Çizelge 4.21'de ise farklı mutasyon oranlarında algoritmanın bulduğu renk sayıları ve bu sonuçlara ulaşma sıklıkları verilmiştir.

Çizelge 4.20. abb^*, ash^* *ve* $will^*$ grafları için sonuçlar

Graf Adı	Eniyi/χ(G)	MKSA	Mutasyon Oranına Göre Sonuca Ulaşılan İterasyon Değerleri					Çalışma Sayısı (20)
			0.10	0.25	0.50	0.75	1.0	
will199GPIA	7	**7**	42	42	36	47	40	--
abb313GPIA	9	11	242	246	281	271	232	--
ash331GPIA	4	**4**	62	74	85	90	84	20/20
ash608GPIA	4	**4**	85	111	132	108	92	--
ash958GPIA	4	**4**	1578	1494	1927	864	625	--

--: Her mutasyon oranı için ulaşılan sonuç farklılık gösterdiği için başarı sıklıkları çizelge 4.21'de verilmiştir.

Çizelge 4.21 incelendiğinde MKSA $will199GPIA$ grafı için mutasyon oranı $0.25\ ya\ da\ 1.0$ seçildiğinde 20 çalışmamın tamamında en iyi sonuca ulaşılmıştır. $abb313GPIA\ ve\ ash608GPIA$ grafları için ise genellikle kromatik sayılara ulaşılmıştır. $ash958GPIA$ grafı ise $ash608GPIA$ grafına göre daha zor bir graftır. MKSA'nın farklı mutasyon oranlarında bu graf için kromatik sayı olan 4 renk sayısına ulaşma sayısı oldukça düşüktür. MKSA, $ash958GPIA$ grafını genellikle 5 renk kullanarak renklendirebilmiştir.

Çizelge 4.21. abb^*, ash^* *ve* $will^*$ graflarında farklı sonuçlar için başarı sıklıkları

Graf Adı	MKSA	Mutasyon Oranına Göre 20 Çalışma (run) Sonunda Bulunan Sonuç ve Sonuca Ulaşma Sıklıkları				
		0.10	0.25	0.50	0.75	1.0
will199GPIA	7	19	20	19	19	20
abb313GPIA	11	20	15	16	20	17
ash608GPIA	4	18	19	19	18	19
	5	2	1	1	2	1
ash958GPIA	4	4	4	5	4	5
	5	16	16	15	16	15

Önerilen algoritma yazmaç özgüleme problemi grafları için de genellikle başarılı sonuçlar bulmuştur. Ancak önerilen algoritma düşük mutasyon oranlarında bazı yazmaç özgüleme problemi grafları için açgözlü algoritmanın bulduğu renk sayısını doğrulayacak bir çözüme ulaşamamış ve düğüm sayısı arttıkça daha düşük bir başarı

oranı sergilemiştir. Yazmaç özgüleme problemi grafları için önerilen algoritmanın başarı sonuçları Çizelge 4.22'de verilmiştir.

Çizelge 4.22. Yazmaç özgüleme problemi grafları için sonuçlar

Graf Adı	Eniyi/χ(G)	MKSA	Mutasyon Oranına Göre Sonuca Ulaşılan İterasyon Değerleri					Çalışma Sayısı (20)
			0.10	0.25	0.50	0.75	1.0	
fpsol2.i1	65	**65**	106	48	10	6	2	20/20
fpsol2.i2	30	**30**	X	1229	820	980	2	--
fpsol2.i3	30	**30**	X	804	513	595	2	--
inithx_i1	54	**54**	X	1453	716	36	2	20/20
inithx.i2	31	**31**	X	X	X	X	2	20/20
inithx.i3	31	**31**	X	X	X	X	1	20/20
mulsol.i1	49	**49**	8	4	3	1	1	20/20
mulsol_i2	31	**31**	46	19	9	3	2	20/20
mulsol.i3	31	**31**	64	16	7	3	1	20/20
mulsol.i4	31	**31**	47	26	8	4	1	20/20
mulsol.i5	31	**31**	64	21	8	4	1	20/20
zeroin.i1	49	**49**	27	12	9	3	1	20/20
zeroin.i2	30	**30**	80	54	10	4	1	20/20
zeroin.i3	30	**30**	89	35	9	5	1	20/20

--: Graflar için başarılı sonuca ulaşma sayıları çizelge 4.22'de verilmiştir.
X: Algoritma açgözlü algoritmanın bulduğu renk sayısını doğrulayacak bir çözüm bulamamıştır.

Çizelge 4.22 incelendiğinde, önerilen algoritmanın yazmaç özgüleme problemi grafları için yüksek mutasyon oranlarında çözüme daha çabuk yakınsadığı gözlemlenmiştir. Önerilen algoritmanın $inithx.i2$ ve $inithx.i3$ graflarını optimize etmekte zorlandığı görülmüştür. Mutasyon oranı 1.0 olarak alındığında bu graflar için başarılı sonuçlar bulunmuştur. Diğer mutasyon oranlarında önerilen algoritma yanlış boyanan düğüm sayısını azaltmasına rağmen, belli bir noktadan sonra çözüme gidememiştir. Bu da algoritmanın belli bir noktadan sonra yerel bir çözüme takıldığını göstermektedir. Önerilen algoritma $fpsol2.i2, fpsol2.i3\ ve\ inithx.i1$ grafları için mutasyon oranı 0.10 seçildiğinde uygun bir çözüme ulaşamamıştır. Çizelge 4.23'de farklı mutasyon oranlarında $fpsol2.i2$ ve $fpsol2.i3$ grafları için önerilen algoritmanın başarılı sonuca ulaşma sayıları verilmiştir.

Çizelge 4.23. $fpsol2.i2$ ve $fpsol2.i3$ grafları için başarı sayıları

Graf Adı	MKSA	Mutasyon Oranına Göre 20 Çalışma (run) Sonunda Bulunan Sonuç ve Sonuca Ulaşma Sıklıkları				
		0.10	0.25	0.50	0.75	1.0
fpsol2.i2	30	0	10	11	20	20
fpsol2.i3	30	0	14	15	20	20

Önerilen algoritma $school1\ ve\ school1\_nsh$ grafları için de genellikle başarılı sonuçlar bulmuştur. Çizelge 4.24 incelendiğinde MKSA'nın düşük mutasyon oranlarında daha başarılı olduğu görülmektedir. Çizelge 4.25'te ise farklı mutasyon oranlarında algoritmanın bulduğu renk sayıları ve bu sonuçlara ulaşma sıklıkları verilmiştir.

Çizelge 4.24. $school1\ ve\ school1\_nsh$ grafları için sonuçlar

Graf Adı	Eniyi/χ(G)	MKSA	Mutasyon Oranına Göre Sonuca Ulaşılan İterasyon Değerleri					Çalışma Sayısı (20)
			0.10	0.25	0.50	0.75	1.0	
school1_nsh	14	**14**	112	124	X	X	X	--
school1	14	**14**	100	99	984	X	X	--

--: Graflar için başarılı sonuca ulaşma sayıları çizelge 4.25'te verilmiştir.
X: En iyi sonuca ulaşamamıştır.

Çizelge 4.25 incelendiğinde, önerilen algoritma $school1\ ve\ school1\_nsh$ graflarını iki farklı renkle boyayabilmiştir. MKSA en iyi sonuçlarını düşük mutasyon oranları için bulmuştur. Mutasyon oranı arttıkça MKSA kromatik sayıya ulaşmakta zorlanmıştır. Mutasyon oranı 0.50'nin üstüne çıktığında algoritmanın bulduğu en iyi çözüm 15 renk olmuştur.

Çizelge 4.25. $school1\ ve\ school1\_nsh$ graflarında farklı sonuçlar için başarı sıklıkları

Graf Adı	MKSA	Mutasyon Oranına Göre 20 Çalışma (run) Sonunda Bulunan Sonuç ve Sonuca Ulaşma Sıklıkları				
		0.10	0.25	0.50	0.75	1.0
school1_nsh	14	13	10	X	X	X
	15	7	10	20	20	20
school1	14	19	16	2	X	X
	15	1	4	18	20	20

X: 20 çalışma içinde hiç görülmedi.

4.5. Literatürdeki Diğer Algoritmalar ile Sonuçların Karşılaştırılması

GBP çözümü için literatürde birçok algoritma sunulmuştur. Bunlardan bazılarının bulduğu sonuçlar önerilen algoritma ile karşılaştırılmıştır. Faraji ve Javadi (2011) arıların davranışlarından esinlenerek GBP için BEECOL olarak adlandırılan algoritmayı önermişlerdir. BEECOL algoritmasının başlangıç çözümünü oluştururken düğüm boyama (ColoringVertex) olarak adlandırdıkları açgözlü bir boyama algoritmasını kullanmışlardır. Daha sonra başlangıç çözümüne önerdikleri arı algoritmasını uygulamışlardır. Son olarak bu çözüme komşuluk araması uygulamışlardır

(Faraji ve Javadi, 2011). Bir diğer çalışmada ise Bui ve ark. (2008) GBP için KKA (Karınca Kolonisi Algoritması) dayalı ABAC (An Ant-Based Algorithm for Coloring Graph) yaklaşımını önermişlerdir. ABAC algoritması hem açgözlü algoritmanın avantajlarından hem de KKA'nın avantajlarından faydalanmıştır. Bui ve ark. öncelikle açgözlü bir algoritma ile üzerinde işlem yapacakları graf için kullanacakları renk sayısı için bir üst sınır (upper bound) belirlemişlerdir. Daha sonra üzerinde işlem yapılacak graf parçalara ayrılmıştır. Her parça için ABAC algoritması düzgün renklendirme kuralına uygun olarak bir çözüm bulduğu anda üst sınır renk sayısı bir azaltılarak, graf parçası için yeniden döngü başlatılmıştır. Durdurma kriteri sağlanıncaya kadar algoritma her uygun çözüm bulduğunda renk sayısını bir azaltarak çalışmaya devam etmiştir. Ancak üst sınır renk sayısı ile çözüm oluşturamadığı anda üst sınır renk sayısını bir arttırarak yeniden çözüm bulmaya çalışmıştır (Bui ve ark., 2008). Mahmoudi ve Lotfi (2015) GBP için MGKA'yı (Modifiye Guguk Kuşu Algoritması) önermişlerdir. Mahmoudi ve Lotfi yaptıkları çalışmada ilk olarak açgözlü bir algoritma ile üzerinde işlem yapacakları graf için anne guguk kuşu olarak adlandırılan başlangıç çözüm kümelerini oluşturmuşlardır. Daha sonra yumurtalama algoritmasını kullanarak popülasyondaki çocuk guguk kuşu olarak adlandırılan bireyleri anne guguk kuşu çözümünü kullanarak oluşturmuşlardır ve bu bireylere önerdikleri algoritmayı uygulamışlardır. Son olarak bireye yerel arama algoritması olan TA (Tabu Arama) algoritması uygulanmıştır (Mahmoudi ve Lotfi, 2015). Tez çalışması kapsamında önerilen algoritmanın başlangıç popülasyonu da açgözlü algoritmanın (RLF ile) bulduğu üst sınır renk ile rastgele oluşturulmuştur. Daha sonra MKSA'nın çalışma adımları uygulanmıştır. Son olarak mempleks içindeki en iyi kurbağaya komşuluk arama algoritması olan Chain metodu uygulanmıştır. Böylece hatalı boyanan düğümler hızlı bir şekilde azaltılmıştır. Çizelge 4.26'da önerilen algoritma ile diğer algoritmaların sonuçları verilmiştir.

Çizelge 4.26. Algoritmaların buldukları sonuçlar ve karşılaştırmaları

Graf Adı	Eniyi/χ(G)	ABAC (Bui ve ark., 2008)	BEECOL (Faraji ve Javadi, 2011)	MGKA (Mahmoudi ve Lotfi, 2015)	MKSA
1-Fullins_4	5	**5**	**5**	**5**	**5**
1-Fullins_5	6	**6**	**6**	**6**	**6**
1-Insertions_4	5	**5**	**5**	**5**	**5**
1-Insertions_5	6	**6**	**6**	**6**	**6**
1-Insertions_6	7	**7**	**7**	**7**	**7**
2-Fullins_3	5	**5**	**5**	**5**	**5**
2-Fullins_4	6	**6**	**6**	**6**	**6**
2-Fullins_5	7	**7**	**7**	**7**	**7**
2-Insertions_4	5	**5**	**5**	**5**	**5**
2-Insertions_5	6	**6**	**6**	**6**	**6**
3-Fullins_3	6	**6**	**6**	**6**	**6**
3-Fullins_4	7	**7**	**7**	**7**	**7**
3-Fullins_5	8	**8**	**8**	**8**	**8**
3-Insertions_3	4	**4**	**4**	**4**	**4**
3-Insertions_4	5	**5**	**5**	**5**	**5**
3-Insertions_5	6	**6**	**6**	**6**	**6**
4-Fullins_3	7	**7**	**7**	**7**	**7**
4-Fullins_4	8	**8**	**8**	**8**	**8**
4-Fullins_5	9	**9**	**9**	**9**	**9**
4-Insertions_3	4	**4**	**4**	**4**	**4**
4-Insertions_4	5	**5**	**5**	**5**	**5**
5-Fullins_3	8	**8**	**8**	**8**	**8**
5-Fullins_4	9	**9**	**9**	**9**	**9**
DSJC125.1*	5	6	6	6	**5**
DSJC125.5*	17	**18**	**18**	19	**18**
DSJC125.9*	44	**44**	**44**	45	**44**
DSJC250.1*	8	**9**	**9**	10	**9**
DSJC250.5*	28	**30**	**30**	33	31
DSJR500.1*	12	**12**	13	**12**	**12**
jean	10	**10**	**10**	**10**	**10**
anna	11	**11**	**11**	**11**	**11**
david	11	**11**	**11**	**11**	**11**
homer	13	**13**	**13**	**13**	**13**
huck	11	**11**	**11**	**11**	**11**
games120	9	**9**	**9**	**9**	**9**
miles250	8	**8**	**8**	**8**	**8**
miles500	20	**20**	**20**	**20**	**20**
miles750	31	**31**	**31**	**31**	**31**
miles1000	42	**42**	**42**	**42**	**42**
miles1500	73	**73**	**73**	**73**	**73**
myciel3	4	**4**	**4**	**4**	**4**
myciel4	5	**5**	**5**	**5**	**5**
myciel5	6	**6**	**6**	**6**	**6**
myciel6	7	**7**	**7**	**7**	**7**
myciel7	8	**8**	**8**	**8**	**8**
queen5_5	5	**5**	**5**	**5**	**5**
queen6_6	7	**7**	8	8	**7**

Graf Adı	Eniyi/χ(G)	ABAC (Bui ve ark., 2008)	BEECOL (Faraji ve Javadi, 2011)	MGKA (Mahmoudi ve Lotfi, 2015)	MKSA
queen7_7	7	**7**	8	**7**	**7**
queen8_12	12	**12**	**12**	13	**12**
queen8_8	9	**9**	10	10	**9**
queen9_9	10	**10**	11	11	**10**
queen10_10*	11	**11**	12	12	**11**
queen11_11	11	**13**	**13**	14	**13**
queen12_12*	13	**14**	**14**	15	**14**
queen13_13	13	**15**	**15**	16	**15**
queen14_14*	16	**16**	**16**	17	**16**
queen15_15*	16	**17**	18	18	**17**
queen16_16*	18	**18**	19	19	**18**
fpsol2.i1	65	**65**	**65**	**65**	**65**
fpsol2.i2	30	**30**	**30**	**30**	**30**
fpsol2.i3	30	**30**	**30**	**30**	**30**
inithx.i1	54	**54**	**54**	**54**	**54**
inithx.i2	31	**31**	**31**	**31**	**31**
inithx.i3	31	**31**	**31**	**31**	**31**
mulsol.i1	49	**49**	**49**	**49**	**49**
mulsol.i2	31	**31**	**31**	**31**	**31**
mulsol.i3	31	**31**	**31**	**31**	**31**
mulsol.i4	31	**31**	**31**	**31**	**31**
mulsol.i5	31	**31**	**31**	**31**	**31**
zeroin.i1	49	**49**	**49**	**49**	**49**
zeroin.i2	30	**30**	**30**	**30**	**30**
zeroin.i3	30	**30**	**30**	**30**	**30**
le450_5c	5	**5**	7	6	6
le450_5d	5	**5**	8	7	6
le450_25a	25	**25**	**25**	**25**	**25**
le450_25b	25	**25**	**25**	**25**	**25**
will199GPIA	7	**7**	**7**	8	**7**
qg.order30	30	**30**	31	**30**	**30**
qg.order40	40	**40**	41	41	**40**
mug100_1	4	**4**	**4**	**4**	**4**
mug88_1	4	**4**	**4**	**4**	**4**
mug88_25	4	**4**	**4**	**4**	**4**
abb313GPIA	9	**10**	**10**	12	11
ash331GPIA	4	4	5	5	**4**
ash608GPIA	4	5	5	5	**4**
ash958GPIA	4	5	5	5	**4**
r125.1	5	X	**5**	**5**	**5**
r125.1c	46	X	**46**	**46**	**46**
r125.5	36	X	**36**	37	**36**
r250.1	8	X	**8**	**8**	**8**
r250.1c	64	X	65	**64**	**64**
school1_nsh	14	**14**	**14**	**14**	**14**
school1	14	**14**	**14**	**14**	**14**

X: Algoritma grafı test etmemiştir.

Çizelge 4.26'ya göre dört algoritma da $Insertion^{*}\ ve\ Fullins^{*}$ grafları için kromatik sayılara ulaşmıştır. Random graflardan $DSJC125.1$ grafı için önerilen MKSA diğer algoritmalara göre daha iyi sonuca ulaşmıştır. $DSJC250.5$ dışındaki random graflar için ise için MGKA dışındaki algoritmalar aynı sonuçlara ulaşmışlardır. $DSJC250.5$ için ise en iyi sonuca ABAC ve BEECOL algoritmaları ulaşmışlardır. MKSA ise $DSJC250.5$ grafı için MGKA'dan daha iyi bir sonuca ulaşmıştır. Random geometrik graf olan $DSJR500.1$ grafı için BEECOL algoritması dışındaki algoritmalar kromatik sayıya ulaşabilmiştir. Algoritmalar $Myciel, Miles, Games120$ ve $Books$ grafları için kromatik sayılara ulaşmışlardır. Bu çalışma kapsamında önerilen MKSA ile Bui ve ark. önerdikleri ABAC algoritması $queen$ grafları için genel olarak kromatik sayılara ulaşabilmiştir. Çizelge 4.26'da ayrıca M. Trick tarafından oluşturulan R serisi random geometrik graflar için BEECOL, MGKA ve MKSA algoritmalarının sonuçları verilmiştir. ABAC algoritması bu grafları test etmemiştir. Çizelge 4.26 incelendiğinde, MKSA'nın kromatik sayılara ulaştığı görülmektedir. $r125.5$ grafı için en iyi sonuca BEECOL ve MKSA ulaşabilmiştir. MGKA ise bu graf için 37 sonucuna ulaşmıştır. Ancak $r250.1c$ grafı için MKSA ve MGKA kromatik sayıya ulaşmışken BEECOL algoritması daha kötü sonuç bulmuştur. Çizelge 4.26'ya göre $school1\ ve\ school1\_nsh$ grafları için ise tüm algoritmalar kromatik sayıya ulaşmışlardır.

Pal ve ark. (2012) yaptıkları çalışmada GBP için Tavlama Benzetimi (TB) (Simulated Annealing-SA) algoritmasına dayalı bir yaklaşım önermişlerdir. TB algoritmasına yanlış boyanan düğüm sayısını ve renk sayısını azaltacak bir metot eklemişlerdir. Başlangıç çözümünü düğüm sayısı kadar renk kullanarak rastgele oluşturmuşlardır. Daha sonra bu çözüme TB ve Modifiye Tavlama Benzetimi (MTB) (Modified Simulated Annealing approach for Graph Coloring Problem- MSAGCP) algoritmalarını uygulamışlardır. Hem orijinal TB algoritmasını hem de önerdikleri MTB algoritmasını bazı Benchmark grafları üzerine uygulamışlardır (Pal ve ark., 2012). Çizelge 4.27'de MKSA, TB ve MTB ait sonuçlar verilmiştir. Çizelge 4.27 incelendiğinde MKSA algoritmasının genel olarak daha başarı sonuçlara ulaştığı gözlemlenmiştir. Pal ve ark. yaptıkları çalışmaya göre TB algoritması genel olarak iyi sonuçlara ulaşamamıştır. Ancak önerdikleri MTB algoritması genel olarak iyi sonuçlara ulaşmıştır. Ancak MTB algoritması yazmaç özgüleme problemi grafları ve $miles$ grafları için genel olarak kromatik sayıya ulaşamamıştır. MTB algoritması $queen$

grafları için başarılı sonuçlara ulaşmıştır. MKSA ise *queen* grafları için genel olarak bilinen en iyi sonuçlara ulaşmıştır.

Çizelge 4.27. TB, MTB ve MKSA için sonuçlar

Graf Adı	Eniyi/χ(G)	TB (Pal ve ark., 2012)	MTB (Pal ve ark., 2012)	MKSA
1-Fullins_5	6	8	**6**	**6**
1-Insertions_5	6	9	7	**6**
2-Fullins_4	6	**6**	**6**	**6**
2-Insertions_4	5	6	**5**	**5**
3-Fullins_4	7	12	9	**7**
3-Insertions_4	5	**5**	**5**	**5**
4-Fullins_3	7	**7**	**7**	**7**
4-Insertions_4	5	7	6	**5**
5-Fullins_3	8	**8**	**8**	**8**
DSJC125.1*	5	10	6	**5**
jean	10	11	**10**	**10**
anna	11	13	**11**	**11**
david	11	**11**	**11**	**11**
homer	13	17	**13**	**13**
huck	11	**11**	**11**	**11**
games120	9	10	**9**	**9**
miles750	31	35	**31**	**31**
miles1000	42	50	45	**42**
miles1500	73	80	75	**73**
myciel6	7	8	**7**	**7**
queen5_5	5	7	**5**	**5**
queen7_7	7	9	**7**	**7**
queen9_9	10	14	11	**10**
queen10_10*	11	17	12	**11**
queen11_11	11	15	**11**	13
zeroin.i2	30	44	**30**	**30**
mulsol.i.1	49	58	52	**49**
mug100_25	4	**4**	**4**	**4**
mug88_25	4	**4**	**4**	**4**

Çizelge 4.28'de MKSA ile Abbasian ve ark. (2011) GBP için yaptıkları çalışmada buldukları sonuçlar verilmiştir. Abbasian ve ark. yapmış oldukları çalışmada Sıralı Sezgisel Graf Boyama Algoritmasını (SSGBA) (Sequential Graph Coloring Heuristic Algorithm-SGCHA), metasezgisel bir algoritma olan Kültürel Algoritma (KA) (Cultural Algorithm-CA) birlikte kullanarak GBP için bir algoritma önermişlerdir. Bu algoritma iki aşamadan oluşmaktadır. Çalışmanın ilk aşamasında SSGBA algoritmasını kullanarak üzerinde işlem yapılan grafın renk sayısı için bir üst sınır belirlemişlerdir. Daha sonra iyileştirme aşamasında popülasyondaki bireyleri bu renk sayısı üzerinden oluşturmuşlardır. Mevcut bireyleri iyileştirmek için GA'nın mutasyon ve çaprazlama operatörlerinden faydalanmışlardır (Abbasian ve ark., 2011). Çizelge 4.28 incelendiğinde hem MKSA'nın hem de Abbasian ve ark. önerdikleri algoritmanın

genel olarak başarılı sonuçlar bulduğu gözlemlenmiştir. $queen7\_7$ grafı için MKSA'nın Abbasian ve ark. önerdikleri algoritmaya göre daha iyi sonuç bulduğu gözlemlenmiştir.

Çizelge 4.28. Algoritmaların buldukları sonuçlar

Graf	Eniyi/χ(G)	SSGBA + KA (Abbasian ve ark., 2011)	MKSA
miles250	8	**8**	**8**
myciel3	4	**4**	**4**
myciel4	5	**5**	**5**
myciel5	6	**6**	**6**
queen5_5	5	**5**	**5**
queen6_6	7	**7**	**7**
queen7_7	7	8	**7**
jean	10	**10**	**10**
anna	11	**11**	**11**
david	11	**11**	**11**
huck	11	**11**	**11**
games120	9	**9**	**9**

5. SONUÇLAR ve ÖNERİLER

5.1. Sonuçlar

KSA, kurbağaların en az hareketle besin kaynağına ulaşmak için yaptıkları hareketlerin modellenmesi ile geliştirilen bir algoritmadır. KSA'da bilgi paylaşımının hem mempleks olarak adlandırılan grup içinde hem de bir iterasyon sonunda tüm kurbağalarla yapılması KSA'nın en büyük avantajlarındandır. Mempleks içinde bilgi paylaşımı yerel arama uzayında daha fazla arama yapma imkânı verir. Mempleks dışında da bilgilerin tüm kurbağalarla paylaşılması ise genel en iyi çözüme yakınsamayı arttırır. Yapılan bu çalışma kapsamında KSA, GBP üzerine uygulanmıştır. KSA ayrık bir optimizasyon algoritması olduğu için GBP'ye doğrudan uygulanabilir. Ancak yapılan çalışmada deneysel sonuçlar gösterdi ki; GBP'nin kendine özgü yapısından dolayı KSA'nın saf halinin GBP'ye uygulanması iyi çözümler üretememiştir. Bundan dolayı KSA'da bazı değişiklikler yapılarak KSA'nın GBP için uygulanabilir bir yöntem olması sağlanmıştır. KSA'nın yerel arama mekanizması yeniden düzenlemiştir. Bu kapsamda alt mempleks içindeki en kötü kurbağanın yeni konumu GA'nın çaprazlama ve mutasyon operatörlerinden faydalanılarak hesaplanmıştır. Ayrıca memetik evrim sonunda mempleksteki en iyi kurbağaya Avanthay ve ark. (2003), çalışmalarında kullandıkları düğüm komşuluğu algoritmalarından (DKA) Chain yöntemi uygulanmıştır. Önerilen algoritma DIMACS tarafından sunulan Benchmark Grafları üzerine uygulanarak sonuçlar değerlendirilmiştir.

Deneysel sonuçlar gösterdi ki; saf KSA üzerinde işlem yaptığı graf için düğüm sayısı kadar renk kullanarak oluşturduğu başlangıç çözümünde kullandığı renk sayısını azaltamamıştır. Saf KSA'nın başlangıç popülasyonu, RLF'nin bulduğu üst sınır renk sayısı ile oluşturulduğu zaman algoritmanın bazı Benchmark grafları için kromatik sayılara ulaştığı, ancak genel olarak etkisiz olduğu gözlemlenmiştir. Önerilen algoritmada başlangıç popülasyonunun düğüm sayısı kadar renk kullanılarak oluşturulması durumunda, birçok Benchmark grafı için kromatik sayılara ulaşıldığı gözlemlenmiştir. Ancak önerilen algoritma düğüm sayısı kadar renk ile oluşturulduğu için DKA'nın komşuluk arama yöntemi olan Chain metodunun avantajlarından faydalanılamamıştır. Düğüm sayısı kadar renk kullanılarak oluşturulan popülasyondaki bireyler düzgün boyama kuralına uygun olarak oluşturulduğu için hatalı boyanan düğüm olmamıştır. Çakışmaların (conflict) olmadığı bir birey için Chain metodu

uygulanamamaktadır. Çünkü Chain, hatalı boyanan düğümlerin sayısını azaltmaya yönelik bir yöntemdir. Başlangıç popülasyonu RLF'nin bulduğu üst sınır renk sayısı kullanılarak rastgele oluşturulması durumunda genellikle düğümler yanlış boyanmış olur. Bu durumda önerilen algoritma çaprazlama, mutasyon ve Chain metodunu etkin olarak kullanabilmiştir. Ayrıca algoritmanın çalışması esnasında, düzgün boyama kuralına uygun olarak sonuca ulaşıldığında, ü*st sınır* renk değeri algoritma içinde bir azaltılmakta ve popülasyondaki bireyler kontrol edilerek, bireylerde kullanılan üst sınır renk, son duruma göre güncellenmektedir. Popülasyonda yapılan bu değişiklik, çaprazlama, mutasyon ve Chain metodunun etkinliğini arttırmıştır. Yapılan değişiklikler başarılı sonuçlara ulaşmayı arttırırken, çözüme yakınsamayı da hızlandırmıştır. Önerilen algoritma bu çalışma kapsamında kullanılan Benchmark graflarının bazılarında RLF'nin bulduğu üst sınır renk sayısını azaltarak daha az sayıda renk ile uygun çözümlere ulaşmıştır. Bazı Benchmark grafları için ise RLF'nin bulduğu üst sınır renk sayısını doğrulayacak bir çözüm oluşturmakta zorlanmıştır. Bunun sebebi bu graflarda düğüm sayısının oldukça fazla olmasıdır. Örneğin *4-Fullins_5* grafı 4146 düğümden oluşmaktadır. Düğüm sayısının çok olduğu *4-Fullins_5* benzeri graflar için çaprazlama ve mutasyon operatörlerinin başarısı azaldığı için, önerilen algoritmanın da başarısı azalmaktadır.

5.2. Öneriler

KSA'da bilgi paylaşımının hem mempleks içinde hem de bir iterasyon sonunda tüm kurbağalarla yapılması hem yerel hem de genel arama uzayında çözüm arama imkânı verdiği için arama uzayı geniştir. Ancak bu avantajının yanında en büyük dezavantajı ise mempleksler birbirinden bağımsız olmasına rağmen bir mempleks için evrimsel döngü tamamlandıktan sonra sıradaki mempleks için evrimsel döngü başlamaktadır. Mempleksler birbirinden bağımsız olduğu için her mempleks için evrimsel döngünün aynı anda başlatılması, algoritmanın performansı arttıracağı düşünülmektedir. Paralel programlama kullanılarak her mempleks için evrimsel döngü aynı anda başlatılabilir. Böylece algoritmanın hızlanacağı düşünülebilir.

Memetik evrim aşamasında memetik evrime tabi tutulacak alt memplekse seçilecek bireyler için farklı yöntemler kullanılabilir. Örneğin 10 bireyden oluşan bir mempleks için bir evrimsel döngü sırasında bu 10 bireyin bir kısmı rulet tekerleğine göre alt memplekse seçilmekte ve alt memplekste bulunan en kötü kurbağa, alt

mempleksteki en iyi kurbağaya benzetilmeye çalışılmaktadır. Eğer en kötü kurbağa daha iyi çözüme gidememişse, bu durumda en kötü kurbağa genel en iyi çözüme göre konum bilgilerini güncellemeye çalışmaktadır. Hâlbuki alt memplekste bulunan en iyi kurbağa, mempleksteki en iyi kurbağa olmayabilir. Mempleksteki en iyi kurbağanın da evrimsel döngüye dâhil edilebileceği bir yöntem önerilebilir. Ayrıca alt memplekse seçilen bireyler için de farklı seçilme mekanizmaları kullanılabilir. Yapılacak değişikliklerin KSA'nın performansını arttırabileceği düşünülmektedir.

Gerçek dünya problemleri genel olarak birbiri ile çelişebilen birden fazla amacın eş zamanlı olarak optimize edilmesini gerektirir. Örneğin bir aracın yol tutuşunu artırmak için aracın ağırlığının arttırılması gerekir. Aracın ağırlığının arttırılması beraberinde harcanan yakıt miktarını arttırır. Bu problem için yol tutuşunu arttıran ve yakıt miktarını azaltan bir çözüme ihtiyaç vardır. İşte bunun gibi problemlerin çözümü için çok amaçlı optimizasyon algoritmalarına ihtiyaç vardır. Bu tarz problemlerinin çözümü için KSA'nın çok amaçlı bir modeli geliştirilebilir.

KAYNAKLAR

Abbasian, R., Mouhoub, M. ve Jula, A., 2011, Solving Graph Coloring Problems Using Cultural Algorithms, *FLAIRS Conference*.

Adewuya, A. A., 1996, New methods in genetic search with real-valued chromosomes, *Massachusetts Institute of Technology*.

Agrawal, J. ve Agrawal, S., 2015, Acceleration Based Particle Swarm Optimization for Graph Coloring Problem, *Procedia Computer Science*, 60, 714-721.

Ahn, S., Lee, S. ve Chung, T., 2003, Modified ant colony system for coloring graphs, *Information, Communications and Signal Processing, 2003 and Fourth Pacific Rim Conference on Multimedia. Proceedings of the 2003 Joint Conference of the Fourth International Conference on*, 1849-1853.

Akyol, S. ve Alataş, B., 2012, Güncel Sürü Zekâsı Optimizasyon Algoritmaları, *Nevşehir Üniversitesi Fen Bilimleri Enstitü Dergisi*, 1, 36-50.

Al-Omari, H. ve Sabri, K. E., 2006, New graph coloring algorithms, *American Journal of Mathematics and Statistics*, 2 (4), 739-741.

Alataş, B., 2007, Kaotik Haritalı Parçacık Sürü Optimizasyon Algoritmaları Geliştirme, *Fırat Üniversitesi*, Fen Bilimleri Enstitüsü,Elazığ, 157.

Ali, F. F., Nakao, Z., Tan, R. B. ve Chen, Y.-W., 1999, An evolutionary approach for graph coloring, *Systems, Man, and Cybernetics, 1999. IEEE SMC'99 Conference Proceedings. 1999 IEEE International Conference on*, 527-532.

Angeline, P. J., 1995, Evolution revolution: An introduction to the special track on genetic and evolutionary programming, *IEEE Expert: Intelligent Systems and Their Applications*, 10 (3), 6-10.

Avanthay, C., Hertz, A. ve Zufferey, N., 2003, A variable neighborhood search for graph coloring, *European Journal of Operational Research*, 151 (2), 379-388.

Berkman, A., Doğanaksoy, A. ve Keyman, E., 1991, Dört Renk Problemi, http://www.matematikdunyasi.org/arsiv/PDF_eskisayilar/91_1_7_10_DORTRENK.pdf: [Ziyaret Tarihi: 6 ocak 2016].

Bessedik, M., Toufik, B. ve Drias, H., 2011, How can bees colour graphs, *International Journal of Bio-Inspired Computation*, 3 (1), 67-76.

Betin, C. ve Erhan, İ., 2013, boyamak ne kadar zor olabilir ?, http://ebulten.library.atilim.edu.tr/sayi/2013-10?sayfa=4: [Ziyaret Tarihi: 6 ocak 2016].

Biggs, N., Lloyd, E. K. ve Wilson, R. J., 1976, Graph Theory, 1736-1936, Oxford University Press, p.

Blöchliger, I. ve Zufferey, N., 2008, A graph coloring heuristic using partial solutions and a reactive tabu scheme, *Computers & Operations Research*, 35 (3), 960-975.

Brélaz, D., 1979, New methods to color the vertices of a graph, *Communications of the ACM*, 22 (4), 251-256.

Brooks, R. L., 1941, On colouring the nodes of a network, *Mathematical Proceedings of the Cambridge Philosophical Society*, 37 (02), 194-197.

Buckley, F. ve Harary, F., 1990, Distance in graphs, Addison-Wesley Longman, p.

Bui, T. N., Nguyen, T. H., Patel, C. M. ve Phan, K.-A. T., 2008, An ant-based algorithm for coloring graphs, *Discrete Applied Mathematics*, 156 (2), 190-200.

Caramia, M. ve Dell'Olmo, P., 2008, Coloring graphs by iterated local search traversing feasible and infeasible solutions, *Discrete Applied Mathematics*, 156 (2), 201-217.

Chaitin, G. J., Auslander, M. A., Chandra, A. K., Cocke, J., Hopkins, M. E. ve Markstein, P. W., 1981, Register allocation via coloring, *Computer languages*, 6 (1), 47-57.

Chams, M., Hertz, A. ve De Werra, D., 1987, Some experiments with simulated annealing for coloring graphs, *European Journal of Operational Research*, 32 (2), 260-266.

Chiarandini, M. ve Stützle, T., 2002, An application of iterated local search to graph coloring problem, *Proceedings of the Computational Symposium on Graph Coloring and its Generalizations*, 112-125.

Chittineni, S., Godavarthi, D., Pradeep, A., Satapathy, S. C. ve Reddy, P. P., 2011, A modified and efficient shuffled frog leaping algorithm (MSFLA) for unsupervised data clustering, *International Conference on Advances in Computing and Communications*, 543-551.

Chmait, N. ve Challita, K., 2013, Using Simulated Annealing and Ant-Colony Optimization Algorithms to Solve the Scheduling Problem, *Computer Science and Information Technology*, 1 (3), 208-224.

Chow, F. C. ve Hennessy, J. L., 1990, The priority-based coloring approach to register allocation, *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 12 (4), 501-536.

Consoli, P., Collera, A. ve Pavone, M., 2013, Swarm intelligence heuristics for graph coloring problem, *2013 IEEE Congress on Evolutionary Computation*, 1909-1916.

Costa, D. ve Hertz, A., 1997, Ants can colour graphs, *Journal of the operational research society*, 48 (3), 295-305.

Croitoru, C., Luchian, H., Gheorghies, O. ve Apetrei, A., 2002, A new genetic graph coloring heuristic, *Computational Symposium on Graph Coloring and Generalizations COLOR*.

Çolakoğlu, Ö., 2012, Graf işlemleri altında yeni boyama ve birleştirilmişlik ölçümleri *Ege Üniversitesi*, Fen Bilimleri Enstitüsü, İzmir, 60.

Çunkaş, M., 2006, Genetik Algoritmalar ve Uygulamaları Ders Notları, http://www.horozerk.com/wp-content/uploads/genetic.pdf: [Ziyaret Tarihi: 15 Aralık 2015].

de Werra, D., 1985, An introduction to timetabling, *European Journal of Operational Research*, 19 (2), 151-162.

Díaz, I. M. ve Zabala, P., 1999, A Generalization of the Graph Coloring Problem.

Dorigo, M., Maniezzo, V. ve Colorni, A., 1991, The ant system: An autocatalytic optimizing process.

Dowsland, K. ve Thompson, J., 2005, Ant colony optimization for the examination scheduling problem, *Journal of the operational research society*, 56 (4), 426-438.

Eberhart, R. C. ve Kennedy, J., 1995, A new optimizer using particle swarm theory, *Proceedings of the sixth international symposium on micro machine and human science*, 39-43.

Emami, H. ve Lotfi, S., 2013, Graph Colouring Problem Based on Discrete Imperialist Competitive Algorithm, *arXiv preprint arXiv:1308.3784*.

Eusuff, M., Lansey, K. ve Pasha, F., 2006, Shuffled frog-leaping algorithm: a memetic meta-heuristic for discrete optimization, *Engineering Optimization*, 38 (2), 129-154.

Eusuff, M. M. ve Lansey, K. E., 2003, Optimization of water distribution network design using the shuffled frog leaping algorithm, *Journal of Water Resources Planning and Management*, 129 (3), 210-225.

Faraji, M. ve Javadi, H. H. S., 2011, Proposing a new algorithm based on bees behavior for solving graph coloring, *Int. J. Contemp. Math. Sciences*, 6 (1), 41-49.

Fister, I. ve Brest, J., 2011, Using differential evolution for the graph coloring, *Differential Evolution (SDE), 2011 IEEE Symposium on*, 1-7.

Fleurent, C. ve Ferland, J. A., 1996, Genetic and hybrid algorithms for graph coloring, *Annals of Operations Research*, 63 (3), 437-461.

Fritsch, R. ve Fritsch, G., 1998, The Four-Color Theorem: History, Topological Foundations, and Idea of Proof, translated by J. Peschke, Springer.

Galinier, P. ve Hertz, A., 2006, A survey of local search methods for graph coloring, *Computers & Operations Research*, 33 (9), 2547-2562.

Garey, M., Johnson, D. ve So, H., 1976, An application of graph coloring to printed circuit testing, *IEEE Transactions on circuits and systems*, 23 (10), 591-599.

Garey, M. R. ve Johnson, D. S., 1979, Computers and Intractability: A Guide to the Theory of NP-completeness, WH Freeman and Company, New York.

Ge, F., Wei, Z., Tian, Y. ve Huang, Z., 2010, Chaotic ant swarm for graph coloring, *Intelligent Computing and Intelligent Systems (ICIS), 2010 IEEE International Conference on*, 512-516.

Gebremedhin, A. H., 1999, Parallel graph coloring, *University of Bergen Norway*.

Glass, C. A. ve Prügel-Bennett, A., 2003, Genetic algorithm for graph coloring: exploration of Galinier and Hao's algorithm, *Journal of Combinatorial Optimization*, 7 (3), 229-236.

Golberg, D. E., 1989, Genetic algorithms in search, optimization, and machine learning, *Addion wesley*, 1989, 102.

Gomes, C. ve Shmoys, D., 2002, Completing quasigroups or latin squares: A structured graph coloring problem, *proceedings of the Computational Symposium on Graph Coloring and Generalizations*, 22-39.

Gross, J. L. ve Yellen, J., 1998, Graph theory and its applications, CRC press, p. 600.

Gwee, B.-H., Lim, M.-H. ve Ho, J.-S., 1993, Solving four-colouring map problem using genetic algorithm, *ANNES*, 332-333.

Hale, W. K., 1980, Frequency assignment: Theory and applications, *Proceedings of the IEEE*, 68 (12), 1497-1514.

Harary, F., 1994, Graph theory, Addison-Wesley, Reading, MA.

Hertz, A. ve de Werra, D., 1987, Using tabu search techniques for graph coloring, *Computing*, 39 (4), 345-351.

Hertz, A., Plumettaz, M. ve Zufferey, N., 2008, Variable space search for graph coloring, *Discrete Applied Mathematics*, 156 (13), 2551-2560.

Hindi, M. ve Yampolskiy, R. V., 2012, Genetic Algorithm Applied to the Graph Coloring Problem, *MAICS*, 60-66.

Holland, J. H., 1975, Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence, U Michigan Press, p.

Hossain, S. ve Steihaug, T., 2008, Graph coloring in the estimation of sparse derivative matrices: Instances and applications, *Discrete Applied Mathematics*, 156 (2), 280-288.

Hsu, L.-Y., Horng, S.-J., Fan, P., Khan, M. K., Wang, Y.-R., Run, R.-S., Lai, J.-L. ve Chen, R.-J., 2011, MTPSO algorithm for solving planar graph coloring problem, *Expert systems with Applications*, 38 (5), 5525-5531.

Johnson, D. S. ve Trick, M. A., 1996, Cliques, coloring, and satisfiability: second DIMACS implementation challenge, October 11-13, 1993, American Mathematical Soc., p.

Jonnalagadda, V. K. ve Mallesham, V. K. D., 2013, Bidding strategy of generation companies in a competitive electricity market using the shuffled frog leaping algorithm, *Turkish Journal of Electrical Engineering & Computer Sciences*, 21 (6), 1567-1583.

Karaboğa, D., 2011, Yapay Zekâ Optimizasyon Algoritmaları, Nobel Yayın Dağıtım**:** 244.

Karakoyun, M., 2015, Kurbağa sıçrama algoritmasının kümeleme problemlerine uygulanması, *Selçuk Üniversitesi*, Fen Bilimleri Enstitüsü , Konya, 95.

Kaya, İ., 2012, Genetik algoritmaların optimal güzergah belirlenmesine uygulanması, *Haliç Üniversitesi*, Fen Bilimleri Enstitüsü, İstanbul, 62.

Kılıçaslan, K., 2007, Graf Teorisine dayalı yerleştirme uygulamaları, *Beykent Üniversitesi* Fen Bilimleri Enstitüsü, İstanbul, 59.

Knuth, D. E., 1993, The Stanford GraphBase: a platform for combinatorial computing, Addison-Wesley Reading, p.

Laguna, M. ve Martí, R., 2001, A GRASP for coloring sparse graphs, *Computational optimization and applications*, 19 (2), 165-178.

Lee, A. ve Streinu, I., 2008, Pebble game algorithms and sparse graphs, *Discrete Mathematics*, 308 (8), 1425-1437.

Leighton, F. T., 1979, A graph coloring algorithm for large scheduling problems, *Journal of research of the national bureau of standards*, 84 (6), 489-506.

Liu, C., Yan, X., Liu, C. ve Wu, H., 2011, The wolf colony algorithm and its application, *Chinese Journal of Electronics*, 20 (2), 212-216.

Lü, Z. ve Hao, J.-K., 2010, A memetic algorithm for graph coloring, *European Journal of Operational Research*, 203 (1), 241-250.

Mahmoudi, S. ve Lotfi, S., 2015, Modified cuckoo optimization algorithm (MCOA) to solve graph coloring problem, *Applied Soft Computing*, 33, 48-64.

Mehrotra, A. ve Trick, M. A., 1996, A column generation approach for graph coloring, *informs Journal on Computing*, 8 (4), 344-354.

Mehta, S. ve Banati, H., 2014, Context aware filtering using social behavior of frogs, *Swarm and Evolutionary Computation*, 17, 25-36.

Michalewicz, Z., 1994, GAs: What are they?, In: Genetic algorithms+ data structures= evolution programs, Eds: Springer, p. 13-30.

Mizuno, K. ve Nishihara, S., 2008, Constructive generation of very hard 3-colorability instances, *Discrete Applied Mathematics*, 156 (2), 218-229.

Mladenović, N. ve Hansen, P., 1997, Variable neighborhood search, *Computers & Operations Research*, 24 (11), 1097-1100.

Murty, K. G., 2003, Optimization Models For Decision Making, vol. 1, Chapter 1, 1-18, http://www-personal.umich.edu/~murty/books/opti_model/junior-1.pdf: [Ziyaret Tarihi: 22 Eylül 2016].

Mycielski, J., 1955, Sur le coloriage des graphs, *Colloquium Mathematicae*, 161-162.

Nuriyev, U. G. ve Sadıgova, H. G., 2002, Hesaplanabilirlik, http://www.matematikdunyasi.org/arsiv/PDF_eskisayilar/2002_5_12_16_HESAPLANABILIR.pdf: [Ziyaret Tarihi: 19 aralık 2016].

Ono, S., Miyamoto, R., Nakayama, S. ve Mizuno, K., 2009, Difficulty estimation of number place puzzle and its problem generation support, *ICCAS-SICE, 2009*, 4542-4547.

Pal, A. J., Ray, B., Zakaria, N. ve Sarma, S. S., 2012, Comparative performance of modified simulated annealing with simple simulated annealing for graph coloring problem, *Procedia Computer Science*, 9, 321-327.

Panda, S., Sarangi, A. ve Panigrahi, S. P., 2014, A new training strategy for neural network using shuffled frog-leaping algorithm and application to channel equalization, *AEU-International Journal of Electronics and Communications*, 68 (11), 1031-1036.

Plumettaz, M., Schindl, D. ve Zufferey, N., 2010, Ant local search and its efficient adaptation to graph colouring, *Journal of the operational research society*, 61 (5), 819-826.

Porumbel, D. C., Hao, J.-K. ve Kuntz, P., 2010, A search space "cartography" for guiding graph coloring heuristics, *Computers & Operations Research*, 37 (4), 769-778.

Qin, J., Yin, Y.-x. ve Ban, X.-j., 2011, Hybrid discrete particle swarm algorithm for graph coloring problem, *Journal of Computers*, 6 (6), 1175-1182.

Rashedi, E., Nezamabadi-Pour, H. ve Saryazdi, S., 2009, GSA: a gravitational search algorithm, *Information sciences*, 179 (13), 2232-2248.

Ray, B., Pal, A. J., Bhattacharyya, D. ve Kim, T., 2010, An efficient ga with multipoint guided mutation for graph coloring problems, *International Journal of Signal Processing, Image Processing and Pattern Recognition*, 3 (2), 51-58.

Reeves, C. R. ve Rowe, J. E., 2002, Genetic algorithms-Principles and perspectives, A guide to GA Theory, *USA: Kluwer Academic Publisher, Norwell,MA*, 1-63.

Reynolds, C. W., 1987, Flocks, herds and schools: A distributed behavioral model, *ACM SIGGRAPH computer graphics*, 21 (4), 25-34.

Ruiz, I. C. R., 2012, Gravitational Swarm for Graph Coloring, *The University of the Basque Country*.

Ruiz, I. R. ve Romay, M. G., 2011, Gravitational swarm approach for graph coloring, In: Nature Inspired Cooperative Strategies for Optimization (NICSO 2011), Eds: Springer, p. 159-168.

Salari, E. ve Eshghi, K., 2005, An ACO algorithm for graph coloring problem, *2005 ICSC Congress on computational intelligence methods and applications*, 5 pp.

Samuel, G. G. ve Rajan, C. C. A., 2015, Hybrid: particle swarm optimization–genetic algorithm and particle swarm optimization–shuffled frog leaping algorithm for long-term generator maintenance scheduling, *International Journal of Electrical Power & Energy Systems*, 65, 432-442.

Sen Sarma, S., Mandal, R. ve Seth, A., 1994, Some sequential graph colouring algorithms for restricted channel routeing, *International journal of electronics*, 77 (1), 81-93.

Sethumadhavan, G. ve Marappan, R., 2013, A genetic algorithm for graph coloring using single parent conflict gene crossover and mutation with conflict gene removal procedure, *Computational Intelligence and Computing Research (ICCIC), 2013 IEEE International Conference on*, 1-6.

Shen, J. W., 2003, Solving the graph coloring problem using genetic programming, *Genetic Algorithms and Genetic Programming at Stanford*, 187-196.

Skiena, S. S., 1990, Implementing Discrete Mathematics: Combinatorics and Graph Theory with Mathematica, Addison-Wesley, Reading, MA, p.

Tagawa, K., Kanesige, K., Inoue, K. ve Haneda, H., 1999, Distance based hybrid genetic algorithm: an application for the graph coloring problem, *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*.

Trick, M., 2016, Graph Coloring and its Generalizations, http://mat.gsia.cmu.edu/COLOR08/INSTANCES/: [Ziyaret Tarihi: 15 şubat 2016].

Uymaz, S. A., 2015, Yeni bir biyolojik ilhamlı metasezgisel optimizasyon metodu: Yapay alg algoritması, *Selçuk Üniversitesi*, Fen Bilimleri Enstitüsü, Konya, 111.

Weisstein, E. W., 2015, "Weighted Graph." From MathWorld--A Wolfram Web Resource., http://mathworld.wolfram.com/WeightedGraph.html: [Ziyaret Tarihi: 15 aralık 2015].

Welsh, D. J. ve Powell, M. B., 1967, An upper bound for the chromatic number of a graph and its application to timetabling problems, *The Computer Journal*, 10 (1), 85-86.

West, D. B., 2001, Introduction to graph theory, Prentice hall Upper Saddle River, p.

Yang, X.-S. ve Deb, S., 2009, Cuckoo search via Lévy flights, *Nature & Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on*, 210-214.

Yılmaz, B., 2011, A novel meta-heuristic for graph coloring problem: simulated annealing with backtracking, *Yeditepe Üniversitesi*, Fen Bilimleri Enstitüsü, İstanbul, 89.

Zwillinger, D., 2002, CRC standard mathematical tables and formulae, CRC press, p.

ÖZGEÇMİŞ

KİŞİSEL BİLGİLER

Adı Soyadı : Murat ASLAN
Uyruğu : T.C
Doğum Yeri ve Tarihi : Nusaybin / 18.03.1988
Telefon :
Faks :
e-mail : mrtasln@hotmail.com, murataslan@selcuk.edu.tr

EĞİTİM

Derece	**Adı, İlçe, İl**	**Bitirme Yılı**
Lise :	Nusaybin (Y.D.A) Süper Lisesi	**2006**
Üniversite :	Selçuk Üniversitesi Bilgisayar Mühendisliği Bölümü	**2011**
Yüksek Lisans :	Selçuk Üniversitesi Bilgisayar Mühendisliği Bölümü	**2017**
Doktora :	-	

İŞ DENEYİMLERİ

Yıl	**Kurum**	**Görevi**
2013 - 2014	Şırnak Üniversitesi	Araştırma Görevlisi
2014 - (Devam ediyor)	Selçuk Üniversitesi	Araştırma Görevlisi

UZMANLIK ALANI

Graf Teorisi, Optimizasyon

YABANCI DİLLER

İngilizce = İleri Seviye

YAYINLAR

ASLAN, M., BAYKAN, N.A., "A Performance Comparison of Graph Coloring Algorithms", IJISAE, Sayı 4(Special Issue), syf. 1-7, 2016 (ICAT 2016 Konferansından seçilmiştir). DOI: 10.18201/ijisae.273053