KÜTAHYA SAĞLIK BİLİMLERİ ÜNİVERSİTESİ
MÜHENDİSLİK VE DOĞA BÖLÜMLERİ FAKÜLTESİ

YAPAY ZEKA DERSİ

NÖBET ÇİZELGELEME PROBLEMLERİNİN GENETİK ALGORİTMALARLA OPTİMİZASYONU

Vize Rapor

Mustafa AKER

2118121039

25.04.2024

1 Giriş

Optimizasyon, bir sistemi veya süreci en iyi duruma getirmek için yapılan bir dizi işlem veya stratejidir. Genellikle belirli bir hedefin veya kriterin en iyi şekilde karşılanması için kaynakların en verimli şekilde kullanılmasını içerir

Çizelgeleme problemleri; çalışanların en düşük işgücü maliyetleri ile çalışma vardiyalarına atanmasının belirlenmesini, personel hizmet kalitesi ve çalışma yasalarıyla ilgili kısıtlamaların karşılanmasını konu alan problemlerdir.

Çizelgeleme problemlerinin uygulama alanlarının başında; hastaneler ,sağlık merkezleri,havalimanları,üniveristeler gibi bir cok kurum gelmektedir. Kurumların her türlü unsuru göz önünde bulundurması ve uygun bir çizelge oluşturabilmesi gerekir.

Çizelgelemedeki asıl amaç, daha az kaynakla ve daha az sürede, istenilen kriterlere uyacak biçimde problemin çözümüne ulaşmaktır. Bu sebeple probleminin en kısa sürede , istenilen hedef ve kriterlere uygun çözülmesini sağlayacak yollar aranacaktır.

2 Literatür Araştırması

[1] 90'ların sonlarında sezgisel arama yöntemleri hemşire çizelgeleme için de kullanılmaya başlanmıştır. Yamamura ve diğerlerinin (1993) çalışmaları, GA ile yapılan ilk hemşire çizelgeleme uygulamasıdır.

[2] Bailey ve diğerleri (1997), farklı beceri seviyelerine sahip personelin planlanması sorununa Benzetilmiş Tavlama ve GA uygulamıştır. Sonuçlar, optimal veya optimale yakın çözümler üretilebileceğini göstermektedir.

[3] Aikelin ve diğerlerinin (2004), ele aldıkları yaklaşım, hemşirelerin permütasyonlarına dayanan dolaylı bir kodlama ile programlar oluşturan sezgisel bir kod çözücü kullanmaktır. Sonuçlar, önerilen algoritmanın yüksek kaliteli çözümler sunabildiğini ve yakın zamanda yayınlanan bir Tabu Arama yaklaşımından daha hızlı ve daha esnek olduğunu ortaya koymaktadır.

[4] Duenas ve diğerleri (2009), hemşirelerin tercihlerini içeren çok amaçlı bir çizelgeyi ele almaktadır. Bu tercihler bulanık kümelerle modellenmiştir ve GA ile birleştirilmiş melez bir çözüm yöntemi kullanılmıştır. Sonuçlar, önerilen yaklaşımın iyi kalitede çözümler ürettiğini ve gerçek hayattaki problemlere uygulanabilir olduğunu göstermektedir.

[5] Balekar ve Mhetre (2013), hemşire çizelgeleme için GA yaklaşımının kullanıldığı çalışmaları toparlayan bir kaynakça sunmuştur.

[6] Leksakul ve Phetsawat çalışmasında elde edilen sonuçlara göre (2014) GA tarafından bulunan hemşire programı,mevcut programla karşılaştırıldığında aylık personel giderlerinde % 12 ve hemşire sayısında % 13 tasarruf görülmüştür.

[7] Kim ve diğerleri (2018), popülasyon büyüklüğü ve mutasyon oranı parametrelerini dikkate alarak Memetik Algıritma ve GA kıyaslamışlardır.

[8] Alfadilla ve diğerleri (2019), acil serviste hemşire çizelgelemesi ile ilgilenmişlerdir. GA hemşirelerin yerine getirilmemiş tercihlerini en aza indirmek için kullanılmaktadır.

[9] İnanç ve Şenaras (2020), evde bakımda hemşire çizelgeleme problemi için Genetik Algoritma kullanmışlardır.

[10] Küçük ve Deveci Kocakoç (2020) hemşirelerin en uygun çalışma saatlerini bulmak icin MATLAB programının GA aracından yararlanılmıştır

3 Metodoloji

Sezgisel algoritmalar genellikle en iyiye yakın olan çözüm yoluna hızlı ve kolay şekilde ulaştıklarından, çalışmada sezgisel algoritmalardan biri olan Genetik Algoritma (GA) kullanılacaktır. GA, tek çözüm değil birden fazla optimal çözüm elde etmesi, çok sayıda parametre ile çalışma imkanı olması, amaç fonksiyonunu geniş bir açıda araştırması nedeniyle yararlı bir yöntemdir.

3.1 Genetik Algoritma

Evrimsel hesaplamanın bir alt dalı olan genetik algoritmalar, yapay zekânın hızla gelişen alanlarından biridir. Bu metasezigel yaklaşımların esin kaynağı Darwin'in evrim teorisidir. Evrimsel hesaplama kavramı, 1960'lı yıllarda I.Rechenberg'in "Evrim Stratejileri" adlı çalışmasında ortaya atılmıştır. Takip eden 1970'li yıllarda da Michigan Üniversitesi'nden John Holand tarafından genetik algoritmalar bulunmuş ve öğrencileri ile meslektaşları yardımıyla geliştirilmiştir. Bunu 1975 yılında Holland'ın "Doğal ve Yapay Sistemlerde Uyum" adlı kitabını yayınlaması izlemiştir.

Genetik algoritmalar konusundaki esas gelişim ise, John Holland'ın doktora öğrencisi David E. Goldberg tarafından 1985 yılında hazırlanan "Gaz ve Boru Hatlarının Genetik Algoritma Kullanılarak Denetlenmesi" konulu doktora tezi ile sağlanmıştır. Ulusal Bilim Fonu tarafından verilen Genç Araştırmacı Ödülü'nü kazanan Goldberg, dört yıl sonra 1989 yılında yayımladığı "Makine Öğrenmesi, Arama ve Optimizasyon İçin Genetik Algoritma" adlı kitabı ile genetik algoritmaya yeni bir boyut kazandırmıştır. Bu çalışma günümüzde dahi genetik algoritma konusunda en kapsamlı referans olma özelliğindedir.

Genetik algoritma yöntemi, evrim teorisi esaslarına göre çalışarak, verilen bir sorun için en iyi çözüm veya çözümleri bulmaya yarar. Bu arayışı, karar değişkeni uzayındaki birçok başlangıç noktasından başlayarak, paralel işlemler dizisi ile en iyi yöne doğru gelişerek yapar. Karar uzayındaki bu noktalarda uygunluk derecelerinden başka bilgilere gerek yoktur. Toplumdaki noktaların paralel çalışarak en iyiye doğru gelişmesi rastgelelik ilkeleri ile sağlanmaktadır. Genetik algoritmanın esası doğal seçme ve genetik kurallarına dayanmaktadır. Bu kurallar ortama en fazla uyum sağlayan canlıların hayata devam etmesi ve uyum sağlayamayanların da elenmesi olarak algılanmalıdır.

Genetik algoritmalar doğadaki en iyinin yaşamasını gerektirmekte ve bunu belirleyen uygunluk işlevi, yeni çözümler üretmek için çaprazlama, kopyalama ve değiştirme gibi operatörleri kullanmaktadır. Genetik algoritmanın önemli özelliklerinden birisi de bir grup üzerinde çözümü araması ve bu sayede çok sayıda çözümün içinden en iyiyi seçmesidir.

Probleme ait en iyi çözümün bulunabilmesi için;

1. Bireylerin gösterimi doğru bir şekilde yapılmalı,
2. Uygunluk fonksiyonu etkin bir şekilde oluşturulmalı,
3. Doğru genetik işlemciler seçilmelidir,

3.2 Genetik Algoritmaların Diğer Yöntemlerden Farkı

Goldberg'e göre genetik algoritmayı diğer arama yöntemlerinden ayıran en belirgin özellikleri çözüm arama şeklinin farklı oluşudur.

1. Genetik algoritma, parametre setlerinin kodları ile ilgilenir, parametrelerin kendileri ile doğrudan ilgilenmez,
2. Genetik algoritmanın arama alanı, yığının veya populasyonun tamamıdır; tek nokta veya noktalarda arama yapmaz,
3. Genetik algoritmalarda, amaç fonksiyonu kullanılır, sapma değerleri veya diğer hata faktörleri kullanılmaz,
4. Genetik algoritmaların uygulanmasında kullanılan operatörler, stokastik yöntemlere dayanır, deterministik yöntemler kullanılmaz

Başka bir ifadeyle genetik algoritmalar doğal olayların gelişmesindeki genetik mekanik ilkelere göre çalışırlar.

- Genetik algoritmalar, çözümlemesinde karar değişkenlerini genetik sayı sistemine göre kodlayarak kullanır. Sayı sisteminde karar değişkenlerinin genleri topluca karar uzayında bir noktayı temsil eder.
- Genetik algoritmalarda bir nokta yerine aynı anda noktalar topluluğundan hareket edilir. Topluluğun genetik algoritmalar evrimi ile gelişmesi sonucunda en iyi çözüme ulaşılır. Evrim sırasında sistem yerel en iyiye takılmaz
- Genetik algoritmalar evrimi sırasında, karar değişkenlerinin belirttiği noktalardaki hedef fonksiyonu değerleri kullanılır. Türev ve integral işlemlerine gerek olmadığından başlangıç ve sınır şartları ile bazı klasik kabullerin yapılmasına gerek yoktur
- Genetik algoritmalar evrim işlemleri belirlilik değil kurallarına dayanır. Seçim işlemleri ihtimal ilkeleri ışığı altında yapılır. En iyi için sayılan klasik yöntemlerin dışında daha basit, fazla matematik gerektirmeyen ve objektif olan GA (genetik algoritma) yaklaşımları kullanılabilir. Genetik algoritmalardaki rastgele yaklaşımlar sonuç çözümün yerel en iyi çözümlere takılıp kalmamasını sağlar

3.3 Temel Kavramlar

[10] Genetik algoritmanın çalışmasında ve başarılı çözüm değerlerine ulaşılmasında temel kavramların iyi anlaşılmasının ve belirlenmisinin önemi büyüktür.

3.3.1 Gen

Kalıtsal molekülde bulunan ve organizmanın karakterlerinin tayininde rol oynayan kalıtsal birimlere denir. Yapay sistemlerde gen, kendi başına anlamlı birer genetik bilgi taşıyan en küçük yapı birimi olarak alınır.

Genetik algoritmanın kullanıldığı programlama yapısında bu gen yapıları programcının tanımlamasına bağlıdır. Bir genin içerdiği bilgi sadece ikili tabandaki sayıları içerebileceği gibi onluk taban ve onaltılık tabandaki sayı değerlerini de içerebilir. Dolayısıyla yazılan programa göre gen içeriği çok önem kazanmaktadır

3.3.2 Kromozom

Bir ya da birden fazla gen yapısının bir araya gelerek oluşturduğu problemin çözümüne ait tüm bilgiyi içeren diziye kromozom denir. Kromozomlar, alternatif aday çözümleri gösterirler

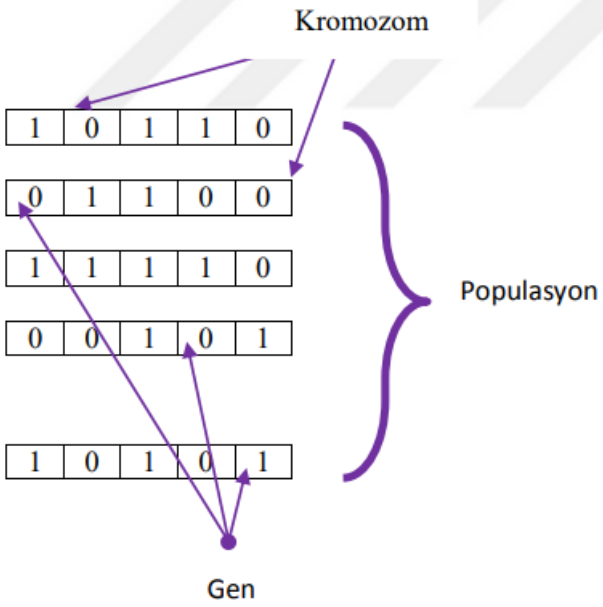
Kromozornlann bir araya gelmesiyle populasyon (yığın) oluşturulur. Yığındaki her bir bireye kromozom, kromozomdaki her bir bilgiye gen denir. Kromozomlar, üzerinde durulan problemin olası çözüm bilgilerini içermekledir Kromozom, GA yaklaşımında üzerinde durulan en önemli birim olduğu için bilgisayar ortamında iyi ifade edilmesi gerekir.

3.3.3 Populasyon

Populasyon, çözüm bilgilerini içeren kromozomların bir araya gelmesiyle oluşan olası çözüm yığınına denir. Yığındaki kromozoın sayısı sabit olup problemin özelliğine göre programlayıcı tarafından belirlenir .

Çözüm uzayı açısından populasyon büyüklüğünün küçük değerde olması çözüm uzayının küçük olmasını bu da aranan en iyi çözüm değerlerine ulaşılamamasına neden olmaktadır. Çözüm uzayının çok büyük değerde olması ise hem genetik algoritmanın etkinliğini azaltmakta hem de çözümün farklı noktalarda aranmasına neden olmaktadır .

Gen, Kromozom ve Populasyon Yapısı



3.4 Kodlama

İlk adım; problem için arama uzayını en iyi temsil eden kodlama yapısının seçilmiş olmasıdır. Kodlama biçimi, genetik algorimanın performansını oldukça önemli oranda etkiler; fakat programa bağlı olduğundan bütün problemler için geçerli en uygun kodlama biçimini söylemek imkânsızdır

3.4.1 İkili (Binary) Kodlama

En yaygın olarak kullanılan kodlama türüdür. Burada kromozomlar 0 ve 1 şeklinde gen değerlerinde kodlanırlar. Bu dizideki her bit, çözümün belli karakteristiğini temsil eder veya tüm dizi bir sayıyı temsil eder.

İkili Düzende Kodlama Yapısı

Kromozom 1	1 0 1 0 1 1 0 1 1
Kromozom 2	0 0 1 0 1 0 1 0 1

3.4.2 Sıralı (Permütasyon) Kodlama

Sıralı (permütasyon) kodlama tekniği genellikle gezgin satıcı, çizelgeleme ve sıralama, şebeke tasarımları gibi sıra takibi olan problemlere uygulanır.

	1	2	3	4	5	6	7	8	9	10	11	12
A_1:	1	2	3	4	5	6	7	8	9	0	4	5
A_2:	7	4	6	1	2	8	3	5	3	1	9	6
Kalıp	1	0	0	0	1	0	1	0	0	0	0	0
Çaprazlama Sonrası Oluşan Yeni Bireyler												
A^1_1:	1	7	2	4	5	6	3	8	9	0	4	5
A^1_2:	1	4	6	5	2	8	3	7	3	1	9	6

3.4.3 Değer (Alpha-Numeric Encoding) Kodlaması

Değer kodlama yönteminde ilgili parametre değerleri doğrudan alınır. Özel problemlerde alfa-sayısal ya da gerçel sayılar olarak kullanımı gerçekleştirilir

Değer kodlama yapısı

Kromozom A	0.123 1.234 2.345 4.567 5.678
Kromozom B	A B C D E F G H I J K L M N

3.5 Seçim

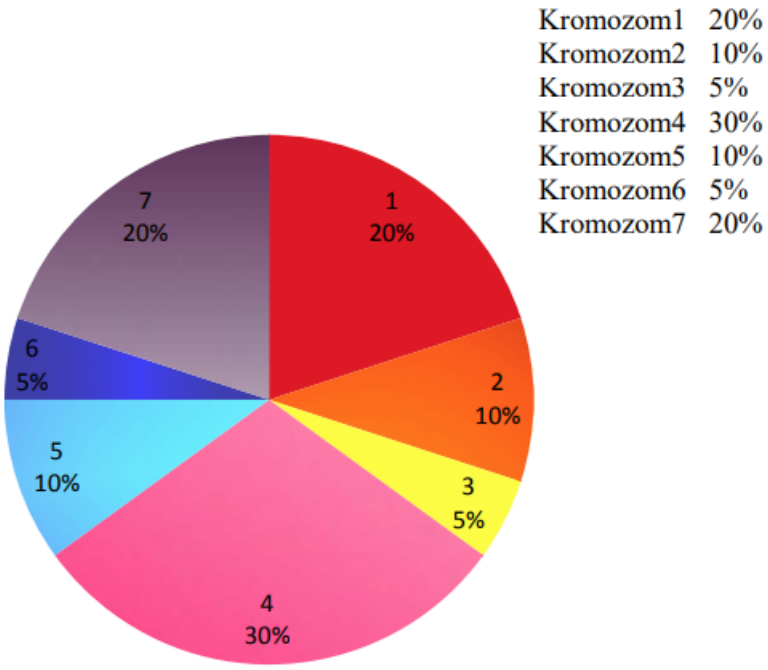
Goldberg'e göre, yeni nesiller için ebeveyn kromozomların belirlenmesi sürecinde önceki populasyondan gelen bazı kromozomların yeni populasyona aktırılması gerekmektedir. Burada önemli olan bu kromozomların nasıl seçileceğidir. Darvin evrim teorisine göre; hayatta kalan en iyi kromozom ebeveyn olarak yeni oğul kromozomları oluşturur. En iyi kromozom seçilmesinde birçok yöntem bulunmaktadır. Her bir kromozomun uygunluk değeri hesaplandıktan sonra uygunluğu yüksek olan kromozomların seçilmesi için geliştirilmiş değişik seçim yöntemleri bulunmaktadır

3.5.1 Rulet Tekeri Seçim Yöntemi

Genetik algoritmalarda kullanılan en basit ve en yaygın seçim mekanizması rulet tekerleği (çemberi) seçimidir.

Rulet Tekeri seçim operatöründe, bütün kromozomlar uygunluk değerlerine göre bir rulet etrafında dizilirler. Rulet üzerinde uygunluk değerlerine göre sıralanan kromozomlar rasgele olarak seçilirler. Bu şekilde her birey seçilmek için kendi uygunluk değerine göre bu rulet tekerinden bir pay almaktadır. Daha büyük alana sahip bireyin seçilme şansı daha fazla olacaktır. Bu metot yardımıyla kromozomlar istatistiksel yöntemler kullanılarak uygunluk fonksiyonu değerlerinin toplam uygunluk fonksiyonuna oranları ölçüsünde seçilirler. Ancak bu seçim yöntemi, uyum değeri büyük olan bireylerin seçilme olasılığı yüksek olduğu için hep aynı kromozomların seçilmesine neden olmaktadır. Bu da populasyon içindeki çeşitliliği etkileyerek sorun yaratır.

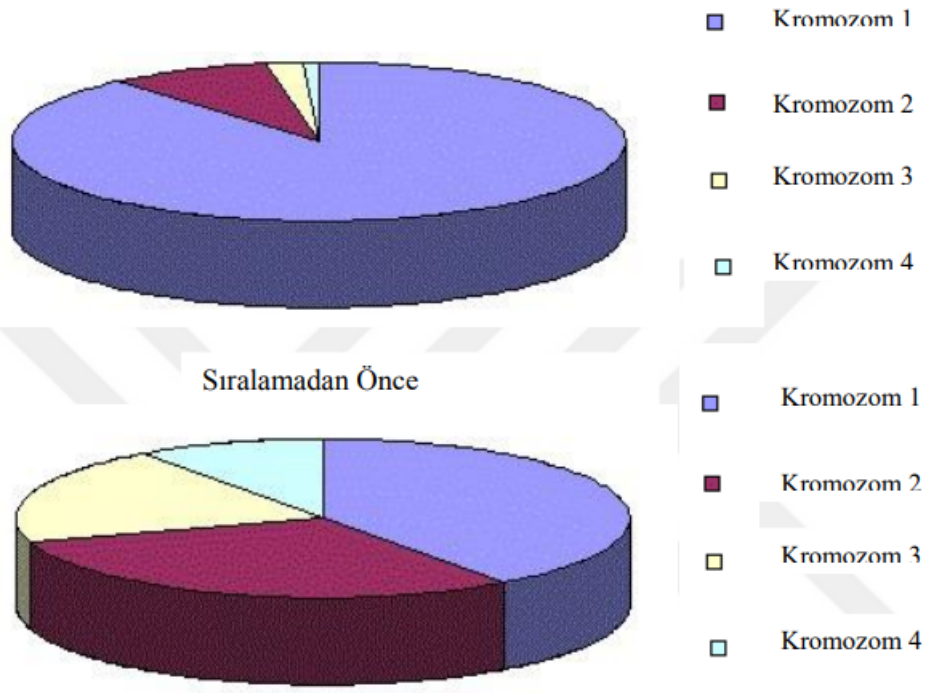
Rulet Tekerleği Seçme Operatörü



3.5.2 Sıralı (Rank) Seçim Yöntemi

Rulet seçimi eğer uyumluluk çok fazla değişiyorsa sorun çıkartabilir. Örneğin en iyi kromozomun uyumluluğu %90 ise diğer kromozomların seçilme şansı azalacaktır. Bunu önlemek için sıralı seçim kullanılabilir. Sıralı seçimde en kötü uyumlulukta olan kromozoma 1 değeri sonrakine 2 değeri verilir ve böylelikle seçilmede bunlara öncelik tanınmış olur. Bu şekilde onların da seçilme şansı artar. Fakat bu da çözümün daha geç yakınsamasına neden olabilir[11]

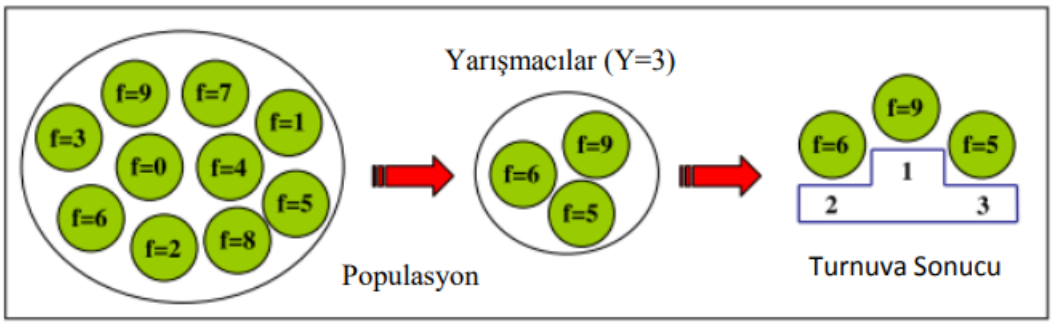
Sıralı Seçim



3.5.3 Turnuva Seçim Yöntemi

Bu seçim yönteminde, bireyler rasgele olarak gruplanır ve gruptaki bireyler aralarında seçim işlemi yapılmak üzere rekabete sokulur. Grup içinde en yüksek uygunluk değerine sahip olan birey, yeni nesli oluşturmak için ebeveyn bireylerden biri olarak seçilir. Bu işlem, toplam birey sayısına ulaşıncaya kadar devam eder. Bazı uygulamalarda grup büyüklüğü iki olarak seçilirken, bazılarında çok daha büyük gruplar oluşturulur [12]

Turnuva Seçim Yöntemi



Populasyondaki bireylerin uygunluk değerleri f ile bireyler arasından rasgele seçilen grup büyüklüğü Y ile gösterilmiştir

3.5.4 Sabit Durum (Kararlı Hal) Seçim Yöntemi

Sabit durum metodunda, her nesilde yalnızca birkaç birey yer değiştirir. Çoğunlukla çok düşük uygunluk değerine sahip bireyler, çaprazlama ve mutasyon yöntemleriyle yeniden üretilerek yeni nesilde yer alırlar. Sabit durumlu GA'lar daha çok kural tabanlı sistemlerde kullanılır

3.5.5 Seçkinlik (Elitizm) İşlemi

Mansfield'e göre üreme, çaprazlama ve mutasyon işlemleri sonrasında kuşakta bulunan en iyi uygunluk değerine sahip birey, sonraki kuşağa aktarılamayabilir. Bunu önlemek için bu işlemlerden sonra oluşan yeni kuşağa bir önceki kuşağın en iyi (elit) bireyi, yeni kuşaktaki herhangi bir birey ile değiştirilir.

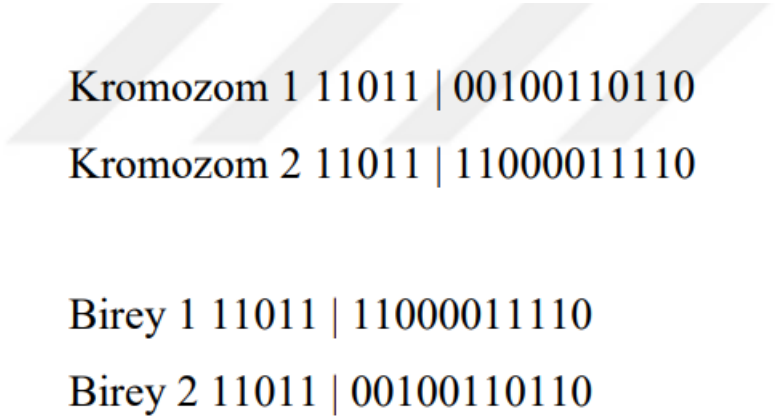
3.6 Genetik Operatörler

Genetik algoritmalarda belirli noktalardan sonra nesil çeşitliliği olmamaktadır. Bunun için dizilere çaprazlama (crossing over) ve değişim (mutation) operatörleri belirli yüzdelik oranlarıyla uygulanarak nesil çeşitliliği sağlanır. Böylece çözümün belirli noktalara gelip tıkanması önlenmiş olur

3.6.1 Çaprazlama

[13]

Çaprazlama ebeveynlerden bazı genleri alarak yeni bireyler oluşturma işlemidir. Burada amaç, eldeki nesilden farklı nesiller elde etmektir. Çaprazlama yapılacak konum rastgele seçilir. Oluşan yeni birey ebeveynlerin bazı özelliklerini almış ve bir bakıma ikisinin kopyası olmuştur. Daha iyi performans almak amacıyla değişik çaprazlamalar kullanılabilir



Çaprazlama operatörü, iki dizinin bir araya gelerek karşılıklı gen yapılarının değişimi ile yeni dizilerin oluşumunu sağlayan operatördür. Çaprazlanarak gen değişiminin yapılmasından önce dizilerin çaprazlamaya tutulma olasılığı belirlenmelidir. Literatürde bu oranın %50-%95 oranında uygulandığı görülmektedir.

Çaprazlamada bir diğer önemli unsur ise ne tür bir çaprazlamanın yapılacağıdır.

3.6.2 İkili kodlama Düzeninde Çaprazlama Yöntemleri

İkili kodlama düzeni için çaprazlama yöntemleri tek nokta, iki nokta ve üç nokta çaprazlama yöntemleri olarak sınıflandırılmıştı.

- **Tek Nokta Çaprazlama Operatörü:**
  Bu işlemde kromozomlar rastgele bir yerinden kesilir ve sonra ilgili genler ile yer değiştirilir. [13]

Tek Noktalı Çaprazlama

	1	2	3	4	5	6	7	8	9	10	11	12
A_1:	1	0	0	1	1	1	0	0	1	1	0	1
A_2:	1	1	0	0	0	0	1	1	1	0	0	0
Çaprazlama Sonrası Oluşan Yeni Bireyler												
A^1_1:	1	0	0	1	1	1	0	0	1	0	0	0
A^1_2:	1	1	0	0	0	0	1	1	1	1	0	1

- **İki Nokta Çaprazlama Operatörü:**
  Bazı durumlarda tek noktalı çaprazlama yöntemi yetersiz kalabilir ya da büyük parçalı blokların bozulması performansı düşürebilir. Bu sebeple iki noktalı çaprazlama yöntemi tercih edilebilir.İki noktalı çaprazlama operatöründe, rasgele iki nokta seçilir ve bu iki nokta arasında kalan bloklar kromozomlar arasında yer değiştirilir. Bu yöntem populasyondaki kromozomların performansını arttırabilir.[12]

İki Noktalı Çaprazlama

	1	2	3	4	5	6	7	8	9	10	11	12
A_1:	1	0	0	1	1	1	0	0	1	1	0	1
A_2:	1	1	0	0	0	0	1	1	1	0	0	0
Çaprazlama Sonrası Oluşan Yeni Bireyler												
A^1_1:	1	0	0	1	1	0	0	1	1	1	0	1
A^1_2:	1	1	0	0	1	1	0	0	1	0	0	0

3.6.3 Sıralı Kodlama Düzeninde Çaprazlama Yöntemleri

Çizelgeleme problemlerinde sıkça kullanılan sıralı (permütasyon) kodlama düzeninde yer alan çeşitli çaprazlama yöntemleri mevcuttur

- **Pozisyona dayalı çaprazlama operatörü (PBX):** bu yöntemde çaprazlama kalıp olarak sabit kalacak olan gen hücrelerini belirler. Kalıpla işaretlenen noktalar dizide sabit kalırken diğer noktalarda iki birey yer değiştirilerek yeni bireylerin üremesi sağlanır.[14]

Pozisyona Dayalı Çaprazlama

	1	2	3	4	5	6	7	8	9	10	11	12
A_1:	3	4	7	1	1	0	4	8	9	2	3	3
A_2:	0	0	1	4	7	2	8	9	2	1	0	0
Kalıp	1	1	1	0	0	0	1	1	0	0	1	0
Çaprazlama Sonrası Oluşan Yeni Bireyler												
A^1_1:	3	4	7	4	7	2	4	8	2	1	3	0
A^1_2:	0	0	1	1	1	0	8	9	9	2	0	3

- **Sıraya dayalı çaprazlama operatörü (OBX):** kalıp üzerindeki 1 değerleri çaprazlamada kullanılacak değerleri gösterir. Bu tür çaprazlama, kromozomu oluşturan karakterlerin sayı ve sıralarının önem taşıdığı durumlarda kullanılır.

Sıraya Dayalı Çaprazlama

	1	2	3	4	5	6	7	8	9	10	11	12
A_1:	1	2	3	4	5	6	7	8	9	0	4	5
A_2:	7	4	6	1	2	8	3	5	3	1	9	6
Kalıp	1	0	0	0	1	0	1	0	0	0	0	0
Çaprazlama Sonrası Oluşan Yeni Bireyler												
A^1_1:	1	7	2	4	5	6	3	8	9	0	4	5
A^1_2:	1	4	6	5	2	8	3	7	3	1	9	6

- **Kısmi eşleşmeli çaprazlama operatörü (PMX):**iki bireyden rastgele bir aralık belirlenir. Bu aralıktaki değerler yer değiştirilir

Kısmi Planlı Çaprazlama

	1	2	3	4	5	6	7	8
A_1:	2	8	6	4	5	7	1	3
A_2:	8	7	2	1	3	4	5	6
Çaprazlama								
A^1_1:	2	8	2	1	3	7	1	3
A^1_2:	8	7	6	4	5	4	5	6

Yer değiştirme sonunda dizide aynı olan değerler değiştirilen değerlerle tamamlanır.

	1	2	3	4	5	6	7	8
Oluşan Yeni Bireyler								
A^1_1:	6	8	2	1	3	7	4	5
A^1_2:	8	7	6	4	5	1	2	3

3.7 Mutasyon

[10]
Kromozomların genleri veya genleri oluşturan küçük birimleri üzerinde değişiklik yapılmasını sağlayan işlemcidir. Değişime uğratılacak kromozomun seçiminde, kromozomun değişime uğrama ihtimalini gösteren ve başlangıçta sabit olarak tanımlanan bir değişim oranı söz konusudur. Genetik algoritmalarda değişime tabi tutulacak kromozomların belirlenmesinde bazılarının istisna tutulması veya özellikle değişime uğratılması gibi özel stratejiler tanımlanabilir.

Amaç belli bir nesil sayısından sonra populasyon içerisindeki bireylerin gitgide birbirlerine benzemesine engel olmaktır. Çünkü bu durum çözüm uzayının daralmasına neden olmaktadır. Bireylere ne kadar çaprazlama operatörü uygulansa da belli bir nesil sayısından sonra birey çeşitliliği sağlanmamaktadır. Bu durumda bireyi oluşturan genlerden rasgele bir tanesi seçilir. Rasgele seçilen genin değeri değiştirilir. Böylelikle populasyon içindeki bireylerin çeşitliliğinin devamı sağlanmış olunur . Yapay sistemlerde mutasyon işlemi esnasında kromozomdaki gen sayısı değişmez sabit kalır. Doğal populasyondaki mutasyon oranı oldukça düşüktür. Mutasyon frekansının büyüklüğü GA'nın performansını etkilemektedir. Mutasyon işlemi bir tek kromozom üzerinde yapılır

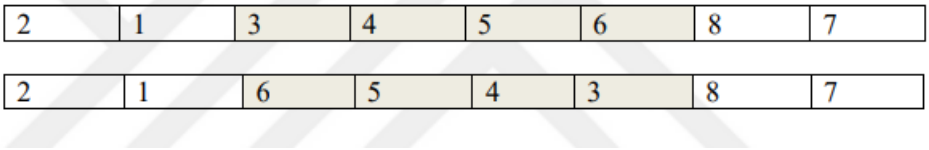
Mutasyon oranı, tıpkı çaprazlama oranında olduğu gibi mutasyonun olasılığını ifade eden bir orandır. Yine rasgele yöntemlerle kromozomun mutasyona uğrayıp uğramayacağı belirlenir ve buna göre mutasyon gerçekleştirilir. Mutasyon oranı genellikle çok düşük (0,01) olduğundan mutasyon işlemi fertlerde az görülür .

Mutasyonun sağladığı avantaj, problemin çözüm alanını araştırmada yön değişikliklerini sağlayarak araştırmanın kısır döngüye girmesini önlemektir. Mutasyon yöntemleri genel olarak beş farklı şekilde sınıflandırılmıştır

3.7.1 Ters Mutasyon

Bir bireyde rassal olarak iki pozisyon seçilir, bu iki pozisyondaki alt diziler ters çevrilir

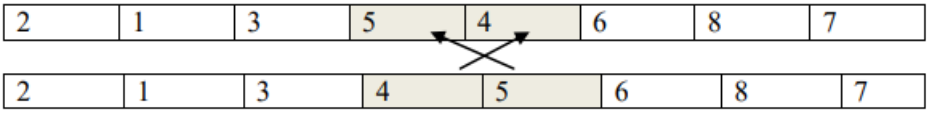
Ters Mutasyon



3.7.2 Komşu İki Geni Değiştirme

Rassal olarak iki komşu iş yer değiştirebilir.

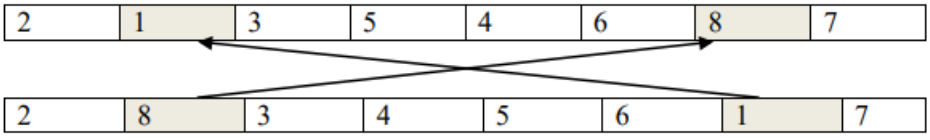
Komşu İki Geni Değiştirme



3.7.3 Keyfi İki Gen Değiştirme

Rassal olarak seçilen iki iş değiştirilebilir. Özel bir durum olarak, değiştirilebilen iki komşu işi bu mutasyon içerir

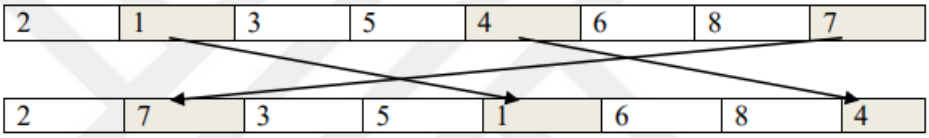
Keyfi İki Gen Değiştirme



3.7.4 Keyfi Üç Gen Değiştirme

Rassal seçilen üç iş keyfi olarak değiştirilir

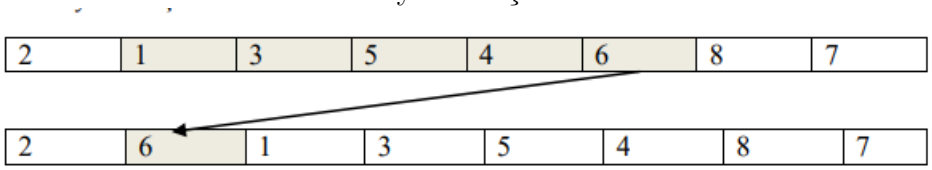
Keyfi Üç Gen Değiştirme



3.7.5 Araya Yerleştirme

Rassal olarak seçilen bir kaydırma noktasında kromozomdaki bir iş kaydırılır ve diğer bir pozisyona yerleştirilir. Komşu iki iş değiştirme yönteminin özel bir durumudur. Keyfi üç iş değiştirmeyle bir kesişime sahiptir

Araya Yerleştirme



3.8 Durdurma Kriteri

Üreme, çaprazlama ve mutasyon işlemlerinden sonra yeni bir nesil oluşmaktadır. Tüm bu işlemler sonsuz döngü içerisinde yapılır. Eğer bir durdurma kriteri belirlenmez ise bu süreç sonsuza kadar devam eder.

A. **Hesaplama zamanı kriteri:**
Bu yöntemde önceden bir hesaplama zamanı veya döngü sayısı belirlenmekte, bu zaman veya döngü sayısına ulaşıldığında durdurulmaktadır. Bu yöntemde belirlenen döngü sayısı gerektiğinden fazla ya da eksik olabilir.

B. **Optimizasyon hedefi kriteri:**
Önceden ulaşılması istenen amaç fonksiyonu değeri bilinmektedir. Uyum değeri bu değere ulaştığında algoritma durdurulmaktadır.

C. **Minimum iyileşme kriteri:**
Genetik algoritma problemlerinde bulunan en iyi çözümler önce hızlı daha sonra yavaş yavaş artış göstermektedir. Bulunan değerlerdeki iyileşme hızının giderek azalması ve sıfıra yaklaşması, artık daha fazla

iyileşme beklenmemesi gerektiğini gösterebilir. Çözüme harcanacak zaman ile çözümden beklenecek kalite arasında bir denge kurularak durdurma gerçekleştirilir[13]

3.9 Genetik Algoritmaların Çözümünde Takip Edilecek İşlem Adımları

1. Adım 1: Kullanıcının önceden tanımladığı kurallara göre genellikle rassal bir çözüm grubu seçilir veya kullanıcının kendisi ilk çözüm grubunu belirleyebilir. İlk çözüm grubuna başlangıç populasyonu denir.

2. Adım 2: Her bir kromozom için bir uygunluk değeri hesaplanır; bulunan uygunluk değerleri dizilerin çözüm kalitesini gösterir. Populasyonda yer alan en iyi uygunluk değerine sahip olan birey (kromozom), bir sonraki yeni nesile (populasyon) doğrudan değiştirilmeden aktarılır.

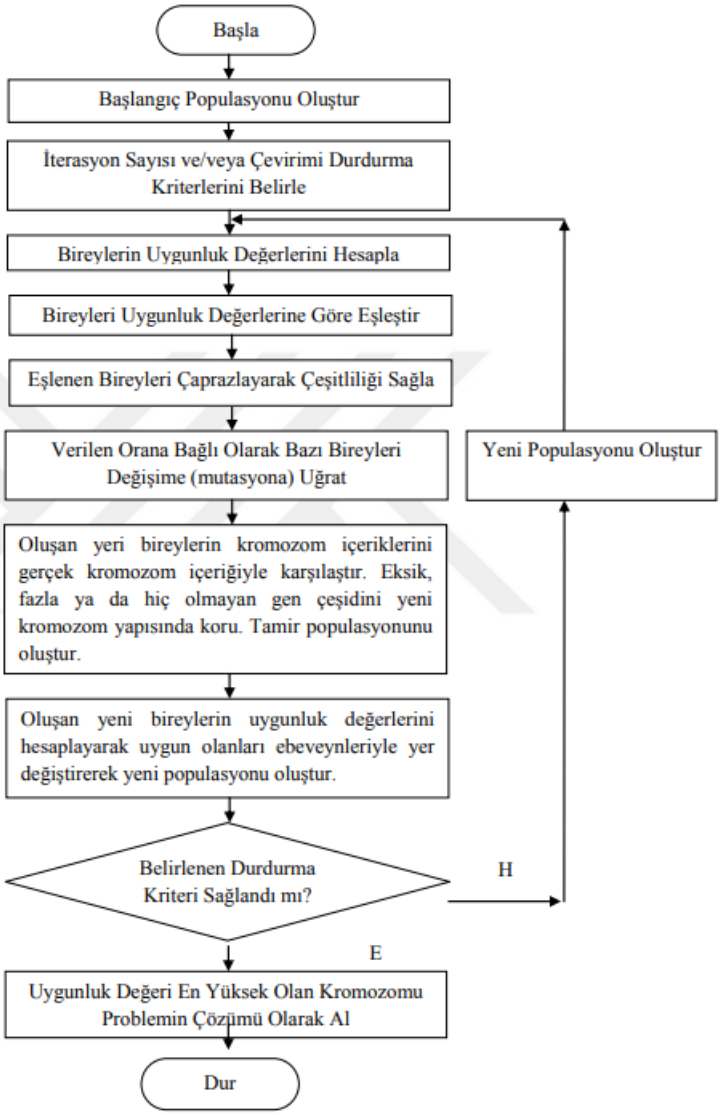
3. Adım 3: İki grup dizi (kromozom), belirli bir seçim yöntemine (uygunluk değerlerine) göre rassal olarak seçilir.

4. Adım 4: Seçilen iki kromozom için rassal olarak genetik operatörler kullanılarak çaprazlama işlemi gerçekleştirilir. Sonuçta yeni populasyonda yer alacak iki yeni birey (kromozom) oluşur. Çaprazlama, yeni populasyonda yer alacak birey sayısına ulaşılana dek sürer.

5. Adım 5: Yeni populasyondaki bireyler, rassal olarak mutasyon işleminden geçerler.

6. Adım 6: Önceden belirlenen nesil sayısı boyunca yukarıdaki işlemler sürdürülür. Eğer en büyük nesil sayısına ulaşılmamışsa Adım 2'ye dönülür. En büyük nesil sayısına ulaşınca işlem bitirilir. Uygunluk değeri en yüksek olan kromozom (çözüm) seçilir.

Genetik Algoritmanın Akış Diyagramı [13]



3.10 Yapılan Çalışmalar

Bu bölümden yukarıda anlatılan Genetik Algoritma kavramlarını anlaşılması icin basit bir uygulamalar yapılmıştır

3.10.1 Örnek-1

[15]
Uygulamada ki amac belirli bir bit dizisi (hedef gen) oluşturmaktır. Başlangıçta rastgele bir popülasyon oluşturulur ve ardından iteratif olarak en uygun genlere yaklaşmak için çaprazlama ve mutasyon işlemleri gerçekleştirilir.
Uygunluk fonksiyonu, hedef gene ne kadar yakın olduklarını belirler.

1. **İlk olarak, random modülünü ve genetik algoritmanın parametrelerini tanımlıyoruz:**

```
1   import random
2
3   # Genetik algoritmanın parametreleri
4   population_size = 10
5   gene_length = 8
6   mutation_rate = 0.1
7
8   # Hedef gen
9   target_gene = "10101010"
10
```

- Popülasyon boyutu, yani her bir neslin kaç bireyden oluşacağı.
- Her bir genin uzunluğu. Bu örnekte genler bit dizileri olarak temsil ediliyor.
- Mutasyon oranı, yani her bir genin mutasyona uğrama olasılığı.
- Hedef gen, yani genetik algoritmanın hedeflediği bit dizisi.

2. **'create population'**
   Fonksiyonu, başlangıç popülasyonunu oluşturur. Rastgele bit dizileri oluşturarak popülasyonu doldurur.

```
10
11  # Popülasyon oluşturma
12  def create_population():
13      population = []
14      for _ in range(population_size):
15          gene = ''.join(random.choice("01") for _ in range(gene_length))
16          population.append(gene)
17      return population
18
```

3. **'fitness'**
   Fonksiyonu, bir genin uygunluk değerini hesaplar. Uygunluk değeri, hedef gene ne kadar yakın olduğunu belirten bir skordur.

```
18
19  # Uygunluk değerini hesaplama
20  def fitness(gene):
21      score = 0
22      for i in range(len(gene)):
23          if gene[i] == target_gene[i]:
24              score += 1
25      return score
26
```

4. **rank population fonksiyonu**
   Popülasyonu uygunluk değerine göre sıralar. En yüksek uygunluk değerine sahip genler en üstte olacak şekilde sıralanır.

```
26
27   # Popülasyonu sıralama
28   def rank_population(population):
29       return sorted(population,key=lambda gene:fitness(gene),reverse=True)
30
```

5. **select parents fonksiyonu**
   Ebeveyn seçimi yapar. Bu örnekte turnuva seçimi kullanılmıştır; rastgele iki birey seçilir ve onlardan uygunluk değeri daha yüksek olanı ebeveyn olarak seçilir

```
31   # Ebeveyn seçimi (turnuva seçimi)
32   def select_parents(population):
33       parent1 = random.choice(population)
34       parent2 = random.choice(population)
35       return parent1, parent2
36
```

6. **crossover fonksiyonu**
   Çaprazlama işlemini gerçekleştirir. İki ebeveynin genlerini alır ve belirli bir noktadan çaprazlar, böylece iki yeni çocuk gen oluşturur.

```
# Çaprazlama (iki nokta çaprazlama)
def crossover(parent1, parent2):
    crossover_point1 = random.randint(0, gene_length - 1)
    crossover_point2 = random.randint(crossover_point1, gene_length)
    child1 = parent1[:crossover_point1] + parent2[crossover_point1:crossover_point2] + parent1[crossover_point2:]
    child2 = parent2[:crossover_point1] + parent1[crossover_point1:crossover_point2] + parent2[crossover_point2:]
    return child1, child2
```

7. **mutate fonksiyonu**
   Genlerde mutasyon gerçekleştirir. Her bir bitin mutasyona uğrama olasılığı mutation rate parametresi tarafından belirlenir.

```
44
45  # Mutasyon
46  def mutate(gene):
47      mutated_gene = ''
48      for bit in gene:
49          if random.random() < mutation_rate:
50              mutated_gene += '1' if bit == '0' else '0'
51          else:
52              mutated_gene += bit
53      return mutated_gene
54
```

8. **create new generation fonksiyonu**
   Bir sonraki nesli oluşturur. Ebeveynleri seçer, çaprazlama ve mutasyon işlemlerini uygular, ve yeni nesli oluşturur.

```
54
55  # Yeni nesil oluşturma
56  def create_new_generation(population):
57      new_generation = []
58      while len(new_generation) < population_size:
59          parent1, parent2 = select_parents(population)
60          child1, child2 = crossover(parent1, parent2)
61          child1 = mutate(child1)
62          child2 = mutate(child2)
63          new_generation.extend([child1, child2])
64      return new_generation[:population_size]
```

9. **genetic algorithm fonksiyonu**
   Genetik algoritmayı çalıştırır. Başlangıç popülasyonu oluşturur, her bir nesilde en iyi geni ve uygunluk değerini gösterir, ve hedef gene ulaşılana kadar yeni nesiller oluşturur.

```
65
66  # Genetik algoritma
67  def genetic_algorithm():
68      population = create_population()
69      generation = 1
70      while True:
71          population = rank_population(population)
72          best_gene = population[0]
73          print(f"Generation {generation}, Best gene: {best_gene}, Fitness: {fitness(best_gene)}")
74          if best_gene == target_gene:
75              print("Target gene found!")
76              break
77          population = create_new_generation(population)
78          generation += 1
79
```

10. **main**
    Genetic algorithm fonksiyonu çağrılır ve genetik algoritma çalıştırılır.

```
80  if __name__ == "__main__":
81      genetic_algorithm()
82
```

Bu adımların birleşimi, genetik algoritmanın çalışmasını sağlar. Başlangıçta rastgele bir popülasyon oluşturulur, her bir nesilde en iyi genler seçilir ve çaprazlama ve mutasyon işlemleri uygulanarak yeni nesiller oluşturulur. Bu süreç, hedef gene ulaşılana kadar devam eder.

```
Generation 1, Best gene: 10001100, Fitness: 5
Generation 2, Best gene: 00001010, Fitness: 6
Generation 3, Best gene: 10011000, Fitness: 5
Generation 4, Best gene: 10000111, Fitness: 4
Generation 5, Best gene: 10011000, Fitness: 5
Generation 6, Best gene: 10001001, Fitness: 5
Generation 7, Best gene: 10001010, Fitness: 7
Generation 8, Best gene: 11101010, Fitness: 7
Generation 9, Best gene: 11101000, Fitness: 6
Generation 10, Best gene: 11001000, Fitness: 5
Generation 11, Best gene: 11100000, Fitness: 5
Generation 12, Best gene: 11100000, Fitness: 5
Generation 13, Best gene: 10000010, Fitness: 6
Generation 14, Best gene: 10111011, Fitness: 6
Generation 15, Best gene: 10111011, Fitness: 6
Generation 16, Best gene: 10001111, Fitness: 5
Generation 17, Best gene: 10001110, Fitness: 6
Generation 18, Best gene: 10101100, Fitness: 6
Generation 19, Best gene: 00111011, Fitness: 5
Generation 20, Best gene: 01111010, Fitness: 5
Generation 21, Best gene: 11101001, Fitness: 5
Generation 22, Best gene: 10110010, Fitness: 6
Generation 23, Best gene: 10111010, Fitness: 7
Generation 24, Best gene: 10101010, Fitness: 8
Target gene found!
```

Bu çıktı, genetik algoritmanın her bir nesilindeki en iyi geni ve bu genin uygunluk değerini gösteriyor. Her bir satır, bir jenerasyonu temsil eder.

Örneğin, "Generation 1, Best gene: 10001100, Fitness: 5" satırı ilk jenerasyonun en iyi genini (10001100) ve bu genin uygunluk değerini (Fitness: 5) gösterir. Daha sonra, her jenerasyonda en iyi gen ve uygunluk değeri güncellenir.

Son jenerasyonda (Generation 24), hedef gen olan "10101010" elde edilmiştir ve uygunluk değeri 8'dir. Bu, genetik algoritmanın hedef gende yaklaştığını ve sonunda hedefi başarıyla bulduğunu gösterir.

3.11 Örnek-2

Bu bölümde Genetik Algoritma kullanılarak 'örnek-1'adlı uygulamadan esinlenerek yapılmıştır.Uygulamada ki amac hedef sayısı olan 1905'ı bulmaktır.

```python
import random
# Parametreler
populasyon_boyutu = 100
mutasyon_orani = 0.1
nesil_sayisi = 1000
# Hedef sayı
hedef_sayi = 1905

# Başlangıç popülasyonu oluşturma
def birey_olustur(uzunluk):
    return [random.randint(0, 9) for _ in range(uzunluk)]

# Uygunluk (fitness) hesaplama
def uygunluk_hesapla(birey):
    uygunluk = 0
    for i in range(len(birey)):
        uygunluk += abs(birey[i] - int(str(hedef_sayi)[i]))
    return uygunluk

# Çaprazlama
def caprazlama(ebeveyn1, ebeveyn2):
    caprazlama_noktasi = random.randint(1, len(ebeveyn1) - 1)
    cocuk1 = ebeveyn1[:caprazlama_noktasi] + ebeveyn2[caprazlama_noktasi:]
    cocuk2 = ebeveyn2[:caprazlama_noktasi] + ebeveyn1[caprazlama_noktasi:]
    return cocuk1, cocuk2

# Mutasyon
def mutasyon(birey, mutasyon_orani):
    for i in range(len(birey)):
        if random.random() < mutasyon_orani:
            birey[i] = random.randint(0, 9)
    return birey

```

```python

# Genetik algoritma
def genetik_algoritma(populasyon_boyutu, mutasyon_orani, nesil_sayisi):
    populasyon = [birey_olustur(len(str(hedef_sayi))) for _ in range(populasyon_boyutu)]

    for nesil in range(nesil_sayisi):
        populasyon = sorted(populasyon, key=lambda x: uygunluk_hesapla(x))
        en_iyi_birey = populasyon[0]
        print(f"Nesil {nesil + 1} - En İyi Birey: {''.join(map(str, en_iyi_birey))}, Uygunluk: {uygunluk_hesapla(en_iyi_birey)}")

        if uygunluk_hesapla(en_iyi_birey) == 0:
            print("Hedef sayı bulundu:", ''.join(map(str, en_iyi_birey)))
            return en_iyi_birey

        yeni_nesil = []

        for _ in range(populasyon_boyutu // 2):
            ebeveyn1 = random.choice(populasyon)
            ebeveyn2 = random.choice(populasyon)
            cocuk1, cocuk2 = caprazlama(ebeveyn1, ebeveyn2)
            cocuk1 = mutasyon(cocuk1, mutasyon_orani)
            cocuk2 = mutasyon(cocuk2, mutasyon_orani)
            yeni_nesil.extend([cocuk1, cocuk2])

        populasyon = yeni_nesil

    print("Belirli nesil sayısına ulaşıldı ancak hedef sayı bulunamadı.")
    en_iyi_birey = min(populasyon, key=lambda x: uygunluk_hesapla(x))
    print("En yakın sayı:", ''.join(map(str, en_iyi_birey)))
    return en_iyi_birey


# Genetik algoritma çalıştırma
genetik_algoritma(populasyon_boyutu, mutasyon_orani, nesil_sayisi)

```

- Parametreler: populasyon boyutu, mutasyon orani ve nesil sayisi gibi parametreler, genetik algoritmanın çalışma şeklini belirler.
- Hedef Sayı Belirleme: İlk olarak, hedef sayıyı 1905 olarak belirliyoruz.
- Başlangıç Popülasyonu Oluşturma: birey olustur fonksiyonu, belirli bir uzunluktaki bireyi rastgele oluşturur. Her bir birey, hedef sayıdaki rakamların sayısına sahip olacak şekilde 0 ile 9 arasında rastgele rakamlardan oluşur.
- Uygunluk (Fitness) Hesaplama: uygunluk hesapla fonksiyonu, bir bireyin hedef sayıya ne kadar uygun olduğunu hesaplar. Her bir rakamın hedef sayıdaki rakamla ne kadar uyumlu olduğunu belirlemek için mutlak değer farkları toplanır.

- Çaprazlama: caprazlama fonksiyonu, iki ebeveyn bireyden yeni çocuk bireyler üretir. Rastgele bir çaprazlama noktası belirlenir ve bu noktaya kadar olan kısımlar ebeveynler arasında değiştirilerek çocuklar oluşturulur.

- Mutasyon: mutasyon fonksiyonu, bir bireyin genlerinde belirli bir mutasyon oranıyla rastgele değişiklikler yapar. Her bir gen için belirli bir olasılıkla (mutasyon oranına bağlı olarak) yeni bir rastgele rakam atanır.

- Genetik Algoritma: genetik algoritma fonksiyonu, genetik algoritmayı uygular. Belirli bir populasyon boyutu ve nesil sayısı ile başlar. Her bir nesilde, populasyon uygunluklarına göre sıralanır ve en iyi birey ekrana yazdırılır. Eğer hedef sayı bulunursa veya belirli bir nesil sayısına ulaşılırsa döngüden çıkılır.

- Genetik Algoritma Çalıştırma: En son kısımda ise parametrelerle genetik algoritma çalıştırılır.

```
Nesil 1 - En İyi Birey: 1715, Uygunluk: 3
Nesil 2 - En İyi Birey: 0906, Uygunluk: 2
Nesil 3 - En İyi Birey: 3902, Uygunluk: 5
Nesil 4 - En İyi Birey: 1807, Uygunluk: 3
Nesil 5 - En İyi Birey: 1835, Uygunluk: 4
Nesil 6 - En İyi Birey: 1802, Uygunluk: 4
Nesil 7 - En İyi Birey: 3904, Uygunluk: 3
Nesil 8 - En İyi Birey: 0605, Uygunluk: 4
Nesil 9 - En İyi Birey: 1704, Uygunluk: 3
Nesil 10 - En İyi Birey: 1404, Uygunluk: 6
Nesil 11 - En İyi Birey: 1704, Uygunluk: 3
Nesil 12 - En İyi Birey: 0704, Uygunluk: 4
Nesil 13 - En İyi Birey: 1816, Uygunluk: 3
Nesil 14 - En İyi Birey: 1504, Uygunluk: 5
Nesil 15 - En İyi Birey: 0802, Uygunluk: 5
Nesil 16 - En İyi Birey: 2908, Uygunluk: 4
Nesil 17 - En İyi Birey: 1904, Uygunluk: 1
Nesil 18 - En İyi Birey: 0802, Uygunluk: 5
Nesil 19 - En İyi Birey: 2906, Uygunluk: 2
Nesil 20 - En İyi Birey: 2904, Uygunluk: 2
Nesil 21 - En İyi Birey: 1706, Uygunluk: 3
Nesil 22 - En İyi Birey: 2803, Uygunluk: 4
Nesil 23 - En İyi Birey: 2804, Uygunluk: 3
Nesil 24 - En İyi Birey: 1814, Uygunluk: 3
Nesil 25 - En İyi Birey: 3814, Uygunluk: 5
Nesil 26 - En İyi Birey: 2706, Uygunluk: 4
Nesil 27 - En İyi Birey: 1906, Uygunluk: 1
Nesil 28 - En İyi Birey: 1905, Uygunluk: 0
Hedef sayı bulundu: 1905
```

Bu çıktı, genetik algoritmanın her neslindeki en iyi bireyi ve uygunluk değerini göstermektedir.
Örneğin, "Nesil 1 - En İyi Birey: 1715, Uygunluk: 3" ifadesi, ilk neslin en iyi bireyinin 1715 olduğunu ve bu bireyin uygunluk değerinin 3 olduğunu belirtir. Uygunluk değeri, hedef sayıya ne kadar yakın olduğunu gösterir. Daha küçük bir uygunluk değeri, hedef sayıya daha yakın bir çözümü temsil eder.
Çıktıda görüldüğü gibi, 28. nesilde uygunluk değeri 0 olan bir birey bulunmuştur. Bu, hedef sayının tam olarak bulunduğunu gösterir. Sonuç olarak, "Hedef sayı bulundu: 1905" ifadesiyle belirtilir.

4 Veri Ve Sistem Tasarımı

Uygulamada kullanılan veriler; Ahsen KÜÇÜK'ün hazırlamış olduğu ve Prof. Dr. İpek DEVECİ KOCAKOÇ danişmanlık yaptığı "HEMŞİRE ÇİZELGELEME PROBLEMLERİNİN GENETİK ALGORİTMALARLA OPTİMİZASYONU VE BİR UYGULAMA" başlıklı yüksek lisans tez çalışmasında kullanılan Buca Seyfi Demirsoy Devlet Hastanesi, Koroner Yoğun Bakım servisinin Şubat 2015 çizelgeleridir.[10]

4.1 Kısıtlar

4.1.1 Zorunlu (sabit katı) kısıtlar

1. Hemşireler gündüz ve akşam olmak üzere iki vardiya şeklinde çalışmaktadır.

2. Gündüz vardiyasında çalışan gece nöbet tutamaz. Gece nöbetçisi gündüz çalışamaz. Bir hemşire bir gün içinde yalnızca bir vardiyada çalışılabilir

3. Denetleme ve kontrol işini yapan sorumlu hemşire (supervisor) gece nöbeti tutmaz. Sorumlu hemşire zaten ciddi bir iş yükü altındadır, gece nöbeti bu nedenle ona verilmez ve denetleme görevi aksatılmaz

4. Meslekte 30 yılını doldurmuş, 50 yaşın üstünde olan, sağlık sorunu olan ya da gebe olan hemşireler nöbet tutmaz.

5. kşam nöbetinde iki kişi çalışır

6. Her hemşire nöbetçi olmadığı gün 8 saat çalışır. Dolayısıyla haftada 40 ayda 160 saat çalışma zorunluluğu vardır. Bu kısıt hemşirenin zorunlu olarak doldurmak durumunda olduğu mesai saatlerini ifade eder

7. Akşam nöbeti tutanlara ertesi gün görev verilmez. Akşam 16 saatlik nöbetten çıkan hemşire ertesi sabah çalışmaya devam etmez

8. Hemşire aynı gün içinde yalnızca bir vardiyada çalışabilir ya da izinlidir. 2.kısıta benzer olan bu kısıt hemşire o gün içerisinde çalışmadıysa izinli olduğunu vurgular

9. Akşam vardiyasında nöbet tutan iki kişiden birinin uzman olmasına çalışılır

4.1.2 Esnek Kısıtlar

1. Nöbetçi personel nöbeti teslim alacak personel gelmeden ayrılamaz.

2. Nöbetler sağlık bakım hizmetlerinin onayı olmadan değiştirilemez. Elle yapılan çizelgelerde kimi zaman hemşireler kendi aralarında anlaşıp günlerini değiştirmeyi talep etmektedir. Değiştirilmesinde sakınca olmasa da bu işlem ancak sağlık bakım hizmetlerinin onayı ile yapılabilmektedir.

4.2 Amaç Fonksiyonu

Bu kısıtların sağlanması ile ulaşılmak istenen amaç çalışma sürelerinin minimizasyonu ve hemşireler için en iyi çalışma çizelgelerinin oluşturulmasıdır

4.3 Mevcut Çizelgele

Koroner yoğun bakım servisinde 14 hemşire bulunmaktadır. Aşağıdaki tablolarda Şubat ayında çalışan hemşirelerin toplam vardiya sayıları, her bir hemşirenin toplam gece nöbeti sayıları ve izin günleri incelenmiştir. Her gece vardiyasında en az 1 uzman hemşire olmasına ve hamile veya süt izninde olan hemşirelerin vardiya atamalarının sadece gündüz vardiyası şeklinde olmasına dikkat edilmiştir. Modelde çizelge uzunluğu 28 gündür. Hemşirelerden 4 tanesi verilen tarihler aralığında yıllık izin haklarını kullanmıştır.

Nöbet cizelgesinin anlaşılmasını kolaylaştırmak için matrisler ile ifade edilmiştir. Burada çalışılan günler 1 değerini, nöbet çıkışı olan ya da senelik izin olan günler 0 değerini almıştır.

4.3.1 Koroner Yoğun Bakım Matris Gösterimi

sabah	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
1	0	0	1	1	0	0	1	1	1	1	1	1	1	0	0	1	1	1	1	1	1	1	1	1	1	0	1	1
2	1	0	0	1	1	1	0	0	1	1	1	1	0	0	1	1	1	1	0	0	1	1	1	0	0	1	1	1
3	1	1	0	0	1	1	1	1	0	0	1	1	1	1	1	1	1	1	1	0	0	1	1	1	1	1	0	0
4	1	1	1	0	0	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1	0	0	1	1	1	1	0
5	1	1	1	1	1	0	0	1	1	1	1	0	0	1	1	1	1	1	0	0	1	1	1	1	0	0	1	1
6	1	1	1	1	1	1	1	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	1	0	0	1	1	0
7	1	1	1	1	1	1	0	0	1	1	0	0	1	0	0	1	1	1	1	1	1	1	1	0	0	1	1	1
8	1	1	0	0	0	0	1	1	1	1	1	1	1	1	0	0	1	1	1	1	1	1	0	0	1	1	1	1
9	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	1	0	0	1	1	0	0	1
10	1	1	1	1	1	1	1	0	0	0	0	1	0	0	0	0	0	0	1	1	0	0	0	0	1	1	0	0
11	0	0	1	0	0	1	1	1	1	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	1
12	1	0	0	1	1	0	0	1	0	0	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1	1	1	1
13	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1
14	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

gece	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
1	1	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0
2	0	1	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
3	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0
4	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	1
5	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0
6	0	0	0	0	0	0	0	1	0	1	0	1	0	0	0	1	0	1	0	1	0	0	0	1	0	0	0	1
7	0	0	0	0	0	0	1	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0
8	0	0	1	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	1	0	0	0	1	0	0
10	0	0	0	0	0	0	0	1	0	1	0	0	1	0	1	0	1	0	0	0	1	0	1	0	0	0	1	0
11	1	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
12	0	1	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

İzinler ve gerekli kısıtlamlar kullanılarak asağıda ki nöbet takvimi oluşturulmuştur

4.3.2 Koroner Yoğun Bakım Gece Nöbeti Listesi

Günler	Hemşireler	
1	1	11
2	2	12
3	3	8
4	4	11
5	1	8
6	5	12
7	2	7
8	6	10
9	3	12
10	6	10
11	7	11
12	5	6
13	2	10
14	7	1
15	8	10
16	9	6
17	4	10
18	9	6
19	2	5
20	3	6
21	10	12
22	4	9
23	8	10
24	7	6
25	11	5
26	9	1
27	3	10
28	4	6

4.3.3 Koroner Yoğun Bakım Yıllık İzin

13	19.02.2015	19.02.2015	1gün	20.02.2015
2	24.02.2015	25.02.2015	2 gün	26.02.2015
11	27.02.2015	27.02.2015	1 gün	28.02.2015
9	02.02.2015	13.02.2015	12 gün	14.02.2015

5 Sonuç

Bu aşamadan sonrasında genetik algoritmalar ile cizelgeleme problemi örnekleri yapılacak olup dolayısıyla genetik algorıtmaların kullanıldığı bir cok cizelgeleme problemi incelenecek ve cizelgeleme kısıtlamaları kullanılarak mevcut nöbet sisteminin optimize edilmesi calışılacaktır.

Kaynaça

[1] M. Yamamura, S. Kobayash, M. Yamagishi, and H. Ase, "Nurse scheduling by genetic algorithms," *Hiroaki Kitano: Genetic Algorithms*, vol. 2, pp. 89–125, 1993.

[2] R. Bailey, K. Garner, and M. Hobbs, "Using simulated annealing and genetic algorithms to solve staff-scheduling problems," *Asia-Pacific Journal of Operational Research*, vol. 14, no. 2, p. 27, 1997.

[3] U. Aickelin and K. A. Dowsland, "An indirect genetic algorithm for a nurse-scheduling problem," *Computers & operations research*, vol. 31, no. 5, pp. 761–778, 2004.

[4] A. Duenas, G. Y. Tütüncü, and J. B. Chilcott, "A genetic algorithm approach to the nurse scheduling problem with fuzzy preferences," *IMA Journal of Management Mathematics*, vol. 20, no. 4, pp. 369–383, 2009.

[5] S. S. Balekar and N. Mhetre, "Survey of genetic algorithm approach for nurse scheduling problem," *International Journal of Science and Research*, vol. 4, no. 6, pp. 55–62, 2013.

[6] K. Leksakul, S. Phetsawat, *et al.*, "Nurse scheduling using genetic algorithm," *Mathematical Problems in Engineering*, vol. 2014, 2014.

[7] J. Kim, W. Jeon, Y.-W. Ko, S. Uhmn, and D.-H. Kim, "Genetic local search for nurse scheduling problem," *Advanced Science Letters*, vol. 24, no. 1, pp. 608–612, 2018.

[8] N. Alfadilla, P. Sentia, D. Asmadi, *et al.*, "Optimization of nurse scheduling problem using genetic algorithm: a case study," in *IOP Conference Series: Materials Science and Engineering*, vol. 536, p. 012131, IOP Publishing, 2019.

[9] Ş. İnanç and A. E. Şenaras, "Solving nurse scheduling problem via genetic algorithm in home healthcare," in *Transportation, Logistics, and Supply Chain Management in Home Healthcare: Emerging Research and Opportunities*, pp. 20–28, IGI Global, 2020.

[10] A. Küçük and İ. D. KOCAKOÇ, "Hemşire çizelgeleme problemlerinin genetik algoritmalarla optimizasyonu ve bir uygulama," *Manisa Celal Bayar Üniversitesi Sosyal Bilimler Dergisi*, vol. 19, no. Armağan Sayısı, pp. 203–210, 2021.

[11] J. H. Holland, *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press, 1992.

[12] A. Elen, "Çizelgeleme probleminin sezgisel optimizasyon yaklaşımıyla çözümü," Master's thesis, Fen Bilimleri Enstitüsü, 2011.

[13] S. Biroğul, "Genetik algoritma yaklaşımıyla atölye çizelgeleme," *Gazi Üniversitesi Fen Bilimleri Enstitüsü Yüksek Lisans Tezi, Gazi Üniversitesi Fen Bilimleri Enstitüsü, ANKARA, Ocak*, 2005.

[14] E. Aydemir, "Atölye tipi çizelgeleme problemlerinin öncelik kuralı tabanlı genetik algoritma yaklaşımıyla simülasyon destekli optimizasyonu," Master's thesis, Fen Bilimleri Enstitüsü, 2009.

[15] miracöztürk, "Business intelligence specialist." `https://miracozturk.com/python-ile-siniflandirma-analizleri-genetik-algoritmalar/`. Erişim Tarihi: 15 Nisan 2024.