Düzce Üniversitesi Bilim ve Teknoloji Dergisi

Araştırma Makalesi

Üniversite Ders Çizelgeleme Probleminin Genetik Algoritma ile Optimizasyonu[1]

Recep ÇOLAK [a,*], Tuncay YİĞİT [b]

[a] *Bilgisayar Teknolojileri Bölümü, UEMYO, Isparta Uygulamalı Bilimler Üniversitesi, Isparta, TÜRKİYE*
[b] *Bilgisayar Mühendisliği Bölümü, Mühendislik Fakültesi, Süleyman Demirel Üniversitesi, Isparta, TÜRKİYE*
** Sorumlu yazarın e-posta adresi: recepcolak@isparta.edu.tr*
DOI: 10.29130/dubited.1012132

ÖZ

Üniversitelerin her dönem başında yaptığı ders çizelgeleme problemi kombinatortal optimizasyon problemlerindendir. Çizelgeleme problemleri NP-Hard sınıfına giren ve çözümü zor problemlerdendir. Determinist bir yaklaşımla olası bütün ihtimallerin denenmesi gibi algoritmalarla çözüm mümkün olsa da çok zaman alıcı bir işlem olduğundan pratikte bu algoritmalar kullanılmamaktadır. Özelikle probleme ait veriler arttıkça ve çözülmesi gereken çok fazla kısıt olması durumunda çözüme ulaşmak daha da güçleşmektedir. Bu çalışmada ders çizelgeleme problemi çözülmesi gereken katı ve esnek kısıtlar olarak ele alınmıştır. Katı kısıtlar ders çakışması, derslik çakışması, kapasiteye uygun olmayan dersliğe şube atanması gibi kesin olarak çözülmesi gereken kısıtlardır. Esnek kısıtlar ise derslerin istenmeyen zaman dilimlerine atanması bir kısmı ihmal edilebilen kısıtlardır. Bu çalışmada probleme ait katı ve esnek kısıtlar belirlenmiş ve bu kısıtları ihlal edilen durumlara ceza puanları atanarak en az ceza puanına sahip çözümler aranmıştır. Problemin çözümü için çizelgeleme problemlerinde sıkılıkla kullanılan Genetik Algoritma kullanılmıştır. Yapılan testeler sonucunda Genetik Algoritma ile ders çizelgeleme probleminin kısa sürede çözülebildiği görülmüştür.

Anahtar Kelimeler: *Genetik algoritma, ders çizelgeleme, ders programı, sezgisel algoritmalar*

Optimization of University Course Scheduling Problem with Genetic Algorithm

ABSTRACT

The course scheduling problem that universities do at the beginning of each semester is one of the combinatorial optimization problems. Scheduling problems are NP-Hard problems and difficult to solve. Although it is possible to solve with algorithms such as trying all possible possibilities with a deterministic approach, these algorithms are not used in practice because it is a very time-consuming process. Especially when the data of the problem increases and there are too many constraints to be solved, it becomes more difficult to reach a solution. In this study, the lesson scheduling problem is considered as hard and soft constraints that need to be solved. Hard constraints are constraints that need to be resolved, such as course conflict, classroom conflict, assigning a branch to a classroom that is not suitable for capacity. Soft constraints, on the other hand, are constraints that can be neglected by assigning courses to undesirable time slots. In this study, the hard and soft constraints of the problem were determined and the solutions with the least penalty points were sought by assigning penalty points to the situations in which these constraints were violated. Genetic Algorithm, which is frequently used in scheduling problems, was used to solve the problem. As a result of the tests, it was seen that the course scheduling problem could be solved in a short time with the Genetic Algorithm.

Keywords: *Genetic algorithm, course scheduling, course timetabling, heuristic algorithm*

[1] ICAIAME 2021 konferansında sunulmuş olup, özet metin olarak basılmıştır.
Geliş: 19/10/2021, Düzeltme:30/11/2021, Kabul: 06/12/2021

I. GİRİŞ

Çizelgeleme problemleri; eğitim, lojistik, sağlık gibi birçok alanda karşılaşılan ve genelde alanda uzmanlaşmış kişilerin geçmiş tecrübesi ile çözülmektedir. Kombinatoryal optimizasyon problemlerinden olan çizelgeleme problemleri doğası gereği çözümü zor olan NP-Hard problemlerdir. Az sayıda veri ve koşul olduğunda probleme ait tam çözümler bulunabilirken veriler arttıkça makul bir sürelerde çözüm bulunamamaktadır. Çizelgeleme problemlerinde birden çok çözüm olup, tek ve tam bir çözüm yerine makul bir sürede çözüme yakın sonuçların elde edilmesi yeterli görülebilmektedir[1]. Çizelgeleme problemlerinin türevlerinden olan Üniversite ders çizelgeleme de derslik, ders saati ve öğretim elemanı kısıtlılığından dolayı çözümü zor bir optimizasyon problemidir [2]. Birçok araştırmacı farklı yöntemlerle problemi çözmeye çalışmışlardır [3]–[8]. Ders çizelgeleme problemini basitçe ders, öğretim elemanı ve derslik kaynaklarının istenen koşullar altında ve çakışmayacak şekilde aynı zaman periyoduna atanması olarak tanımlanabilir. Çakışmaların olmadığı bir ders çizelgesi mevcut çözümlerden biri olarak düşünülebilir. Pratikte ise sadece çakışmaların giderilmesi yeterli olmayıp başka birçok koşulunda sağlanması gerekebilir. Öğretim elemanı tercihi, derslerin atanabileceği derslik tipleri, şubeye bir günde atanması gereken maksimum ders saati, öğretim elemanının bir günde girebileceği maksimum ders sayısı, ders saati fazla olan bazı derslerin bölünerek atanmak istenmesi, uygulaması olan derslerde uygulama saatlerinin başka bir güne atanması gibi ve okula özgü başka kısıtların da dikkate alınması gerekmektedir. Problemin çözümü için gereken araştırma uzayı büyüdükçe ve atamalar için çok fazla koşulun ve kısıtların eklenmesi probleminin çözümünü daha da zorlaştırmaktadır [9]. Ders çizelgeleme problemi esnek ve katı kısıtlara ayrılabilen koşulları sağlaması gereken bir problemdir [10], [11]. Katı kısıtlar mutlaka sağlanması gereken ve sağlanmadığında çakışmaların olduğu ya da kaynakların yetersiz kaldığı kısıtlardır. Esnek kısıtlar ise sağlanmadığında da bir çakışma ya da kaynak yetersizliği olmayan ama ders çizelgesinin kalitesini etkileyen kısıtlardır. Bir öğretim elemanın istemediği bir saatte derse girmesi esnek bir kısıt iken, bir şubenin derslerinin çakışması katı kısıtlara örnek olarak verilebilir. Ders Çizelgeleme Problemi için yapılan birçok çalışmada, esnek ve katı kısıtları ihlal etmeyen ya da esnek kısıtların bir kısmını ihmal eden sonuçlar problemin çözümüm olarak görülmüş ve bu kısıtlara uyan çözümler aranmıştır [9], [10], [12]–[15]. Bu çalışmada problemin çözümü için çizelgeleme problemlerinde sıklıkla kullanılan Genetik Algoritma (GA) kullanılmıştır [1], [7], [9], [10][16]. Geliştirdiğimiz GA'da başlangıç rotaları rastgele oluşturulmuş ve GA operatörleri ile katı ve esnek kısıtların ihlal edildiği durumlar cezalandırılmıştır. GA'nın katı kısıtları çözmesi için katı kısıtlara esnek kısıtlardan daha büyük ceza puanları verilmiştir. Katı kısıtların esnek kısıtlara görece yüksek ceza puanları olması GA uygunluk fonksiyonunda katı kısıtların daha ağırlıklı olmasını sağlamış ve bu kısıtların iyileştirilmesi ile elde edilen yeni sonuçların daha iyi sonuçlar olarak algılanmasını sağlamıştır. Bu çalışma literatüre aşağıdaki katkıları sağlamıştır:

- GA'da kullanılan mutasyon işleminin yeni üretilen bireyler yerine tüm popülasyona uygulanmasının daha iyi sonuçlar verdiği gösterilmiştir.
- Derslerin ders parçalarına bölünerek atanması ile dersin farklı saatleri için farklı kısıtların nasıl çözülebileceği gösterilmiştir.

Çalışmanın bundan sonraki kısımlarında Bölüm 2'de literatür özeti, Bölüm 3'te problemin tanımı ve geliştirilen matematiksel model, Bölüm 4'te GA'nın probleme uygulanması son olarak ise elde ettiğimiz sonuçlar verilmiştir.

II. LİTERATÜR

Üniversite ders çizelgeleme problemine araştırmacılar graf boyama, tam sayılı doğrusal programlama, tam sayılı kısıt programlama, tabu arama, en yakın komşu arama, tavlama benzetimi, GA, karınca kolonisi gibi algoritmalarla çözüm aramışlardır[17]. Graf boyama algoritması ders çizelgeleme problemi için ilk kullanılan yöntemlerden birisidir. Graf boyama ile hem basit ve karmaşıklığı az olan veri kümeleri için [18] hem de ilerleyen yıllarda algoritmaya müdahaleler edilerek daha karmaşık olan veri kümelerine çözüm aranmıştır [19]. Tam sayılı doğrusal programlama da erken dönemde sıklıkla

kullanılmış algoritmalardan birisidir [20]. Tam sayılı doğrusal programlama algoritması da geliştirilerek farklı araştırmacılar tarafından problemin çözümünde kullanılmıştır [21]–[23]. Tam sayılı programlama ile yapılan çalışmalarda problemin önce matematiksel bir modeli çıkarılmış ve bu model üzerinden çözümler aranmıştır. Ders çizelgeleme gibi NP-hard problemlerin optimum çözümünü deterministtik yöntemlerle kabul edilebilir bir zaman da bulmak zor olduğundan bir çok araştırmacı sezgisel yöntemlere başvurmuştur [24]. Bu çalışmada ise problem sezgisel bir yöntem olan GA ile çözülmeye çalışılmıştır. Ders çizelgeleme problemini GA ile çözmeye çalışan ilk çalışmalardan birisini Colorni ve arkadaşları yapmışlardır. Çalışmalarında özellikle çaprazlama ve mutasyon işlemleri sonrası uygun olmayan çözümlerle karşılaşmışlardır[25]. Benzer şekilde Yiğit' te yaptığı çalışmada kromozom yapısının bozulmasından dolayı çaprazlama işleminden sonra tamir operatörü kullanmıştır [10]. Bizim yaptığımız denemelerde de aynı durumla karşılaşılmış ve benzer şekilde tamir operatörü kullanılmıştır. Problemi GA ile çözmeye çalışan birçok yazar kromozomları oluşturmak için rastgele bir atama işlemi uygulamıştır [26], [27]. Bizim çalışmamızda ise başlangıç rotalarının oluşturulmasında genetik çeşitlilik açısından bir rastgelelik olsa da tamamen random bir atama yoktur. Çalışmamızda başlangıç rotaları Insertion Heuristic Algoritmasına benzer bir yöntemle oluşturulmuştur[28]. Özellikle rotalama problemlerinde sıklıkla kullanılan bu yöntemde atama yapılmak istenen ders rastgele seçilerek ders programındaki boş bir konuma yerleştirilmeye çalışılmıştır. Yapılan bu yerleştirmenin istenilen kurallara uyması beklenmektedir. İlk atamanın bu şekilde yapılması sayesinde başlangıç kromozomlarının bazı kısıtlara uymasını garanti etmektedir. Bu çalışmada literatüre yaptığımız bir katkıda mutasyon işleminde olmuştur. Mutasyon işlemi GA'nın önemli bir operatörü olup algoritmanın lokal optimumlara takılmasını engellemek için kullanılır. GA ile çözüm arayan birçok çalışma da [29], [30] mutasyon işlemi çaprazlama sonunda üretilen bireyler için kullanılmıştır. Bir başka mutasyon işlemi ise Mahiba vd. tarafından uygulanmıştır, yazarlar yeni üretilen çocuk bireyler yerine uygunluk değeri kötü olan bireylere mutasyon uygulamışlardır [31]. Bizim çalışmamızda yukarıda bahsedilen iki ayrı mutasyon da test edilmiştir. Birinci uygulamada mutasyon işlemi sadece yeni oluşan çocuk bireylere uygulanmıştır. İkinci metotta ise mutasyon işlemi uygunluk değeri daha kötü olan bireylerin seçilme şansının daha yüksek olduğu bir olasılıkla popülasyonda ki bütün bireylere uygulanmıştır. Elde ettiğimiz sonuçlar ikinci tür mutasyon işleminin daha başarılı olduğunu göstermektedir.

III. PROBLEMİN TANIMI

Üniversite ders çizelgeleme problemi, öğrenci grubu(şube) için, dönemde açılmış derslerin uygun dersliklere okul tarafından önceden tanımlanmış zaman periyodları içinde atanmasıdır. Ders çizelgesi ataması yapılacak olan i adet şubeyi, $S = \{s_1, .., s_i\}$, atanması gereken j adet dersi, $D = \{d_1, .., d_j\}$, k adet dersliği $M = \{m_1, .., m_k\}$, l adet zaman periyodunu ise $P = \{p_1, .., p_l\}$ olarak tanımlayabiliriz. Bir günde 10 saat zaman periyodu olan örnek bir ders çizelgesi **Tablo 1**'de görülmektedir.

Tablo 1. *Örnek bir ders çizelgesi*

		Şube 1	Şube 2	….	Şube i_{maks}
Pazartesi	1	Ders Kodu; Öğretim elemanı kodu; Derslik kodu			
	2				
	…				
	10				
⋮	…				
Cuma	1				
	2				
	…				
	10				

Tablo 1'de iki boyutlu gibi görünen ders çizelgesinin tek bir hücresinde ders kodu, öğretim elemanı kodu ve derslik kodunun da olması ile aslında elimizde çözülmesi gereken, $Şube\ sayısı\ x\ Periyot\ sayısı\ x\ Ders\ sayısı\ x$ Öğ$retim\ elemanı\ sayısı\ x\ Derslik\ sayısı$ boyutund a bir problem vardır. Birçok okulda derslerin bazılarında hem teori hem de uygulama, laboratuvar, saha çalışması gibi birden fazla durumun olmasından dolayı atanması gereken dersler kümesi olan D yerine ders parçası kümesi DP'nin kullanılması önerilmektedir. DP, D kümesinden daha büyük olan ve D kümesinin elemanlarına ait ders parçaları kümesidir. $DP = \{ d_1^1, d_2^2, d_2^2, d_2^3 \ldots\ldots d_j^1, d_j^2 \}$ olarak gösterilebilir. Burada d_2^1, d_2^2, d_2^3 elemanları d_2 dersine ait ders parçalarıdır. Derslerin bu şekilde parçalara bölünmesi atanması gereken ders sayısının artmasından dolayı problem boyutunu artırsa da yapısal bir yaklaşım olduğundan birçok avantaj sağlamıştır. Bu avantajlar:

- Aynı dersin farklı parçaları farklı öğretim elemanına atanabilmektedir. Teori kısmına başka bir öğretim elemanı atanırken uygulama kısmına başka bir öğretim elemanı atanabilir.
- Aynı dersin farklı parçaları farklı dersliklere atanabilmektedir. Özellikle teori ve uygulaması olan derslerde dersin bir parçası teori olarak dersliğe, diğer parçası/ları ise laboratuvara atanabilir.
- Dersin farklı parçaları için farklı gün tercihi yapılabilir. Bu sayede dersin önce teorisinin daha sonra uygulamasının atanması sağlanabilir.

Ders ve ders parçası arasındaki ilişki Tablo 2 ve Tablo 3'te görülebilir.

Tablo 2. *Atanması gereken dersler*

Ders Kodu	Ders Adı	Teori	Uygulama
Mat101	Matematik 1	5	0
Tur101	Türk Dili 1	2	0
Bil101	Bilgisayar Programlama 1	3	2

Tablo 3. *Derslere ait ders parçaları*

Ders Parça Kodu	Ders Kodu	Ders Parça ID	Teori	Uygulama
Mat101-1	Mat101	1	3	0
Mat101-2	Mat101	2	2	0
Tur101-1	Tur101	1	2	0
Bil101-1	Bil101	1	3	0
Bil101-2	Bil101	2	0	2

Tablo 2'de dönemde atanması gereken örnek dersler, Tablo 3'te ise derslere ait ders parçaları görülmektedir. GA Tablo 3'teki tabloya göre atamaları gerçekleştirecektir. İki tablo incelendiğinde Mat101 kodlu ve 5 saat teorisi olan ders 2 parça şeklinde Mat101-1 ve Mat101-2 kodu ile atanacaktır. Tur101 kodlu dersi ise tek parça şeklinde Tur101-1 olarak ataması yapılacaktır. Uygulaması olan Bil101 kodlu dersin ise teori ve uygulaması iki ayrı parçaya bölünmüştür. Bil101-1 kodlu ders parçası derse ait teori kısmıdır ve bu ders parçası normal bir dersliğe atanabilir. Bil101-2 kodlu ders parçası ise derse ait uygulama kısmıdır bu parçaya ait kısıtlar oluşturulurken atamanın mutlaka laboratuvara yapılması seçilerek dersin bu kısmının laboratuvarda olması sağlanabilir. Örnekte de gösterildiği gibi aynı derse ait olsalar da her ders parçasına ayrı bir kod verildiğinden bu parçalarının her birisine farklı öğretim elemanları, farklı derslik tipleri ve farklı zaman tercihleri yapılabilmektedir.

Atanması gereken her ders parçası derse kayıtlı öğrenci sayısına uygun kapasiteye sahip bir dersliğe atanmalıdır. Bu durum şöyle şekilde ifade edilebilir. Her ders parçasını alan öğrenci sayısı $DP\_kapasite(n,i) = \{ dk_1^1, dk_2^1, dk_2^2, dk_2^i \ldots\ldots dk_n^1, dk_n^i \}$ ile ve her dersliğin kapasitesi $S\_kapasite(m) = \{s_{k1}, .., s_{km}\}$ ile tanımlanmış olsun. Bir ders parçasının bir dersliğe atanması için derslik kapasitesin ders parçası kapasitesinden büyük veya eşit olması gereklidir. $x_{n,m}^i$ karar

değişkenini, n. dersin i. parçasının m. dersliğe atanmış olması durumunda 1 aksi durumda ise 0 olduğunu varsayarsak. $x_{n,m}^{i} * S\_kapasite(m) \geq x_{n,m}^{i} * DP\_kapasite(n,i)$ koşulu mutlaka sağlanmalıdır. Üniversite ders çizelgeleme problemi temelde bir optimizasyon problemidir ve aynı veri kümesi için birden çok çözüm bulmak mümkündür. Bu çalışmada problem, istenmeyen durumların belirlenerek bu durumları ihlal etmeyen çözümlere ulaşmak şeklide ele alınmıştır. Literatürde bu istenmeyen durumlar mutlaka ihlal edilmemesi gereken durumlar olan katı kısıtlar ve ihlal edilirse kötüde olsa bir çözümün elde edildiği esnek kısıtlar olmak üzere iki ayrı gruba ayrılmıştır bu kısıtlar şunlardır [16].

Katı Kısıtlar

- Bir şubenin aynı zaman diliminde birden fazla dersi olamaz.
- Şubeye ait bütün dersler mutlaka bir zaman dilimine atanmalıdır.
- Öğretim elemanı aynı zaman diliminde birden fazla derse atanamaz.
- Bir dersliğe aynı zaman diliminde birden fazla ders parçası atanamaz.
- Ders parçaları kapasite açısından uygun dersliğe atanmalıdır.

Esnek Kısıtlar

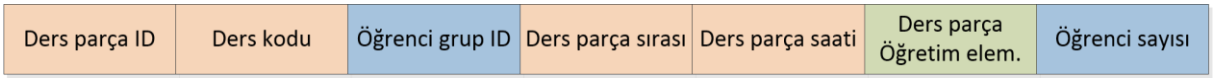
- Bir ders parçasının bütün saatleri aynı dersliğe atanmalıdır.
- Bir ders parçasının bütün saatleri aynı güne atanmalıdır.
- Ders parçası için atanmak istendiği derslikler seçilebilmelidir.
- Bazı derslerin günün belirli zamanında yapılmak istenmesinden dolayı ders parçalarının atanması için zaman tercihi yapılabilmelidir.
- Bir derse ait birden çok ders parçası var ise bu parçalar farklı günlerde olmalıdır.
- Şubeye ait ders programında boş saatler olmamalıdır.
- Öğretim elemanları ders programları için gün ve saat tercihinde bulunabilmelidir.
- Öğretim elemanı ders programında atamalar arasında boş saatler olmamalıdır.

IV. GENETİK ALGORİTMA VE PROBLEME UYGULANMASI

GA, Holland tarafından geliştirilen [32] rastlantısal bir yaklaşımla probleme çözüm arayan evrimsel bir algoritmadır [33]. GA probleme ait çözüm ya da çözümleri aynı anda çözüm uzayının birçok noktasında paralel olarak tarayabilmektedir. GA, Sezgisel bir yaklaşım olduğundan probleme ait kesin çözümleri bulamasa da çözüme yakın sonuçlara ulaşabilmesi NP-hard sınıfına giren çizelgeleme ve optimizasyon problemlerine makul bir sürede kabul edilebilir çözümler sunabilmektedir [34]. Birçok araştırmacı GA'nın ders çizelgeleme probleminin çözümünde başarılı sonuçlar verdiğini göstermiştir[35][36]. Ayrıca kodlanmasının ve uygulanmasının kolay olmasından dolayı bu çalışmada GA kullanılmıştır. Ders çizelgeleme problemi için geliştirdiğimiz GA'nın akış şeması **Şekil 2**'de görülmektedir.

A. GEN YAPISI

GA'nın en temel yapı taşı ve anlamlı bilgi içeren en küçük veri yapısı genlerdir. Canlılardaki genler ve dizilimi canlıya ait özellikleri ifade ettiği gibi probleme ait çözümde gen dizileri ile ifade edilmektedir. Her problemin kendine özgü yapısından dolayı farklı gen tipleri oluşturulabilmektedir. Bu çalışmada ele aldığımız problem için önerdiğimiz gen yapısı **Şekil 1**'de görülebilir.



Şekil 1. *Kullanılan gen yapısı*

Kullanılan gen yapısı aşağıda açıklanmıştır.

Ders parça ID: Derslerin birden çok parçaya bölünerek atama yapıldığı daha önce ifade edilmişti. GA atama için ders kodunu değil bu değeri kullanmaktadır. Buradaki ID değeri ile atanması istenen ders parçası ifade edilmektedir.

Ders Kodu: Ataması yapılan ders parçasının bağlı olduğu gerçek ders kodu değeridir. Gen yapısında yer almasının sebebi ise aynı derse ait ders parçalarının aynı gün içinde atanıp atanmadığını kontrol etmek içindir.

Öğrenci grup ID: Öğrenci gruplarını basitçe öğrenci şubeleri olarak da düşünülebilir. Bizim çalışmamızda öğrenci grup bilgisinin ayrı bir değer olarak tutulması ile farklı dersler için farklı şube sayıları tanımlanabilmektedir.

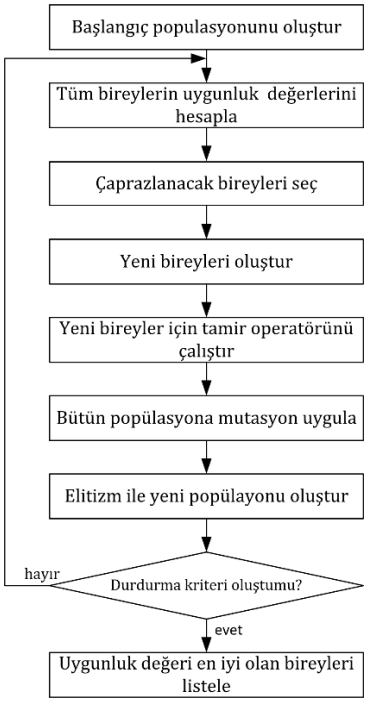
Ders parça sırası: Bir ders birden fazla parça olarak atanacağında önce hangi parçanın atanması gerektiğini tutan değerdir. Özellikle uygulaması olan derslerde önce teori sonra uygulama yapmak gereken durumlar için kullanılacaktır. Eğer bu durum önemli değil ise her ders parçası için aynı değer girilerek öncelik durumu ortadan kaldırılabilir.

Ders parça saati: Ders parçasının atanması gereken toplam ders saatidir. Oluşturulan gen ders planında bu değer kadar ardışık olarak yer alacaktır.

Ders parça Öğretim elem.: Ders parçasına atanana öğretim elemanı bilgisidir. Bu değer ile öğretim elemanın aynı anda başka bir dersinin atama yapılıp yapılmadı kontrol edilebilmektedir.

Öğrenci sayısı: Ders parçasını alan öğrenci sayısıdır. Derslerin kapasitesi uygun dersliklere atanması için kullanılmaktadır.

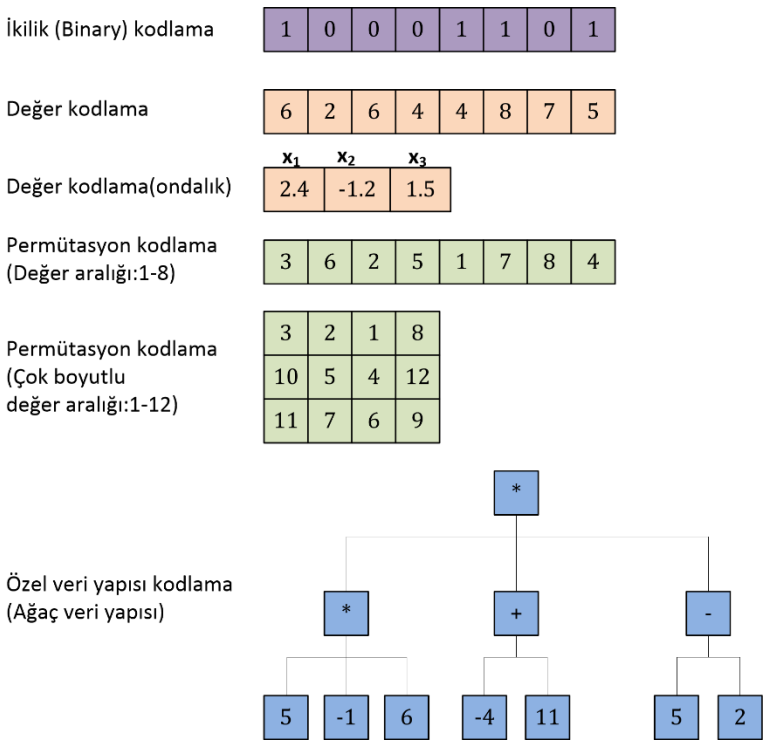
Geliştirilen uygulama yazılımında gen yapısı için bir sınıf tanımlanmıştır.



Şekil 2. *Ders çizelgeleme problemi için uygulanan GA'nın akış şeması*

B. KROMOZOM YAPISI

Kromozom temelde bir gen dizilimi olup probleme ait aday bir çözümü temsil etmektedir. Probleme göre tek boyutlu çok boyutlu ya da ağaç veri yapısı gibi özel bir yapıda olabilmektedir. Kromozomlar ikilik, değer, alfabetik, ondalık gibi farklı formatlarda kodlanabilmektedir. Rotalama ve atama problemi gibi seçenekler arasından belirli değerlerin seçildiği problemlerde ise permütasyon olarak kodlanabilmektedir. Şekil 3'te farklı şekillerde kodlanmış kromozom yapıları görülebilir.



Şekil 3. *Farklı şekillerde kodlanmış kromozomlar*

Bu çalışmada kromozom yapımız çok boyutlu permütasyon kodlama olarak uygulanmıştır. Kullanılan kromozom yapısının boyutları sabit olmayıp okuldaki ders saati periyoduna ve derslik sayısına göre değişebilmektedir. Haftada 5 gün, günde 10 saatlik bir zaman dilimi için ve toplam 12 adet derslik olduğu bir örnek te 10*50 boyutunda iki boyutlu bir kromozom yapısı elde edilecektir. Kullandığımız kromozom yapısı Şekil 4'te görülmektedir. Şekilde de görüldüğü gibi gün*günlük ders sayısı tek bir boyuta indirgenmiş bu durum bize 3 boyutlu bir dizi yerine 2 boyutlu bir dizi ile çalışmamızı sağlamıştır. Dizimizin bir boyutu derslikleri ifade eder iken diğer boyutu gün ve ders saatini birlikte barındıran periyot değerleridir. Bir periyoda ait ve gün ve saat değerleri aşağıdaki bağıntılarla bulunmuştur.

$$Gün\ değeri = \frac{Periyod}{Günlük\ ders\ saati}$$ (Kalansız bir bölme işlemi uygulanmıştır)

$$Ders\ saati\ değeri = Periyod\ \%\ Günlük\ ders\ saati$$ (% sembolü mod alma işlemini ifade etmektedir)

Şekil 4. *Problemin çözümü için önerilen kromozom yapısı*

Belirli bir gün ve belirli bir ders saatinden periyod değerine ulaşmak için ise aşağıdaki bağıntı kullanılmıştır.

$$Periyod\ değeri = Gün * Günlük\ ders\ saati + Ders\ saati$$

Yukarıdaki bağıntı için Salı günü 4. Ders saati için periyod değeri 1*10+4= 14 olarak bulunabilir.

C. BAŞLANGIÇ POPÜLASYONUNUN OLUŞTURULMASI

Başlangıç popülasyonu oluşturmak için dönemde açılmış bütün dersler probleme ait kısıtların bir kısmına uyacak şekilde rastgele olarak atanmıştır. İlk atamada ders parçalarının kapasite ve uygulama açısından uygun bir dersliğe atanması, derslerin başladığı derslikte sonlanması, ders parçasının aynı gün içinde başlayıp bitmesi, ders parçasının her saatinin ardışık olarak atanması kontrol edilerek oluşturulmuştur. İlk atama için kullanılan algoritma aşağıdadır.

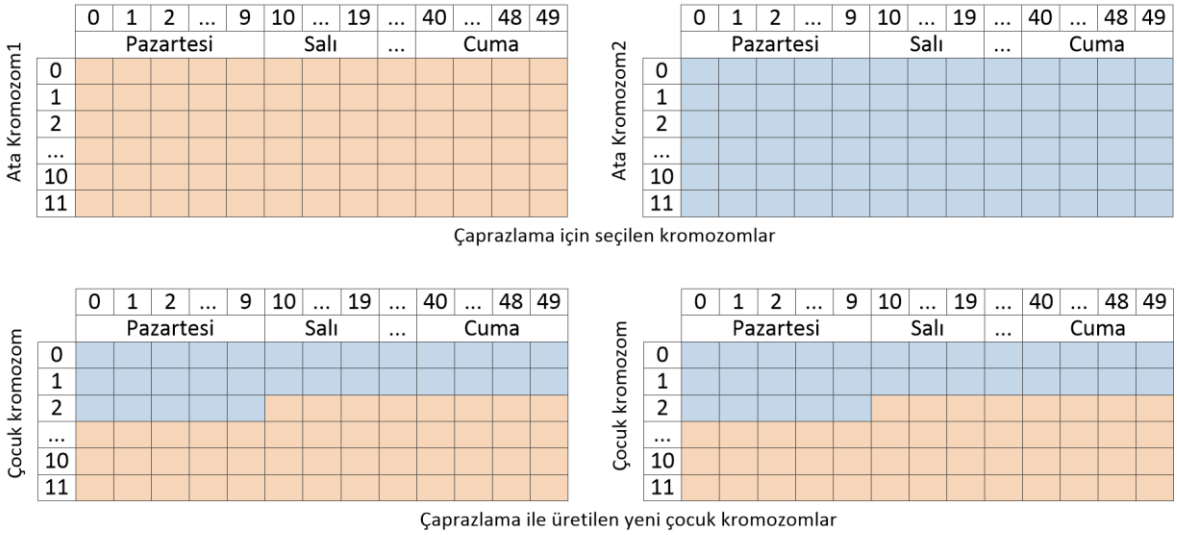
```
dersProgramı ← [derslik, gunSayisi*gunlukSaatSayisi] iki boyutlu dizi
bosKonumlar ←derslik* gunSayisi*gunlukSaatSayisi boyutunda bir boyutlu dizi
dersParcaList ← Atanması gereken ders parçalarını tutan bir boyutlu liste
dersParça //atama için seçilen ders parçası
indis //atama yapılacak konum
for (i=0;i<derslik*periyod)
        bosKonumlar[i]=i  //ders atanabilecek boş konumların listesi
end for
for (i=0;i<dersParcaList eleman sayısı) //bütün dersler yerleştirilinceye kadar dön
        dersParça =dersParcaList[randomSayı(0,dersParcaList eleman sayısı)]
        devamEt=true
        while (devamEt)
                indis=randomSayı(0,bosKonumlar eleman sayısı)
                derslik=boskonumlar[indis] / gunSayisi*gunlukSaatSayisi //tam sayı bölme işlemidir
                peryod=boskonum[indis] MOD  gunSayisi*gunlukSaatSayisi
                gun=peryod / gunlukSaatSayisi
                for (k=0;k<dersParça.atanabileceğiDerslikler)
                        if(dersParça.atanabileceğiDerslikler[k]==derslik)
                                devamEt=false //seçilen derslik uygun ise
                end for
                for (k=0;k<dersParça.dersSaati)
                        derslik2= boskonumlar[indis+k] / gunSayisi*gunlukSaatSayisi
                        peryod2=boskonum[indis+k] MOD  gunSayisi*gunlukSaatSayisi
                        gun2=peryod2 / gunlukSaatSayisi
                        if(derslik!=derslik2 OR peryod!=peryod2 OR gun!=gun2)
                                devamEt =true //kısıt ihlali var ise yeni bir konum ara
                end for
        end while  //ders parçasının bütün saatleri aynı derslik ve güne atanabiliyorsa döngünden çıkar
```

```
        for (k=0;k<dersParça.dersSaati)
                peryod=boskonum[indis+k] MOD  gunSayisi*gunlukSaatSayisi
                dersProgrami[derslik,peryod]=dersParca //ders parçası atandı
                bosKonumlar.RemoveAt(indis) //atanan konum boş konumlar listesinden silindi
        end for
        dersPArcaList.Remove(dersParça) // ataması yapılan ders tekrar atanmasın
end for
```

Algoritma incelendiğinde atama yapılabilecek boş konumların bir liste ile takip edildiği ve atama yapılacak konumların bu listedeki elemanlardan seçildiği görülmektedir. Atama yapılabilecek boş konumların bu şekilde takip edilmesi daha önce atama yapılmış bir konuma atama yapılmasını engellemekte ve aynı zamanda zaman karmaşıklığını da azaltmaktadır. Kullanılan yöntemde daha önce atama yapılmış bir noktanın tekrar seçilmesi engellendiğinden rastgele seçilen her noktanın boş olduğu garanti altına alınmıştır. Aksi durumda rastgele seçilen bir noktanın boş olma ihtimali ders programına yapılan ilk atamalarda yüksek iken dersler atandıkça bu ihtimal düşecektir. Atanacak konumlar rastgele seçildiğinden rastgele sayı üreteci boş bir nokta bulmak için birçok defa çalıştırılmak zorunda kalacaktır.

D. ÇAPRAZLAMA OPERATÖRÜ

Popülasyondaki mevcut çözümlerden yeni çözümler üretme işlemi çaprazlama ile yapılmaktadır. Çaprazlanarak elde edilen yeni çözüm çaprazlama için kullanılan ata bireylerin özelliklerini taşıyacak yeni bireyler oluşturulması hedeflenmektedir. Çaprazlama her zaman daha iyi yeni bireyler elde edilmesini garanti etmese de üretilen yeni bireylerin bazıları ata bireylerinden daha iyi çözümler oluşturabilmektedir. Çaprazlama için probleme, kromozom yapısına ve gen temsiline göre farklı yöntemler kullanılmaktadır [37]. Bu çalışmada tek noktadan ve iki noktadan çaprazlama operatörleri ayrı ayrı test edilmiş ve tek noktalı çaprazlama operatörü ile daha iyi sonuçlar elde edildiği gözlenmiştir. Kullanılan çaprazlama operatörünün uygulanması



Şekil 5. *Çaprazlama operatörünün uygulanması*

Şekil 5'te görülen çaprazlama işleminde 110 değeri rastgele seçilmiş çaprazlama noktasıdır. Seçilen bu nokta için her iki ata kromozomda da ders parçalarını bölmemesi gerekmektedir. Eğer seçilen nokta ders parçalarını bölüyorsa yeni bir nokta seçilmelidir ders parçalarının ortadan bölünmemesini sağlayan algoritma aşağıda görülebilir.

```
devamEt=true
indis //çaprazlama için seçilecek konum
ata1, ata2 //çaprazlama için seçilen kromozom1 ve kromozom2
while (devamEt)
```

```
        devamEt=false
        indis=randomSayi(0, derslik sayısı*günlük ders sayısı* gün sayısı)
        if (ata1[indis]== ata1[indis+1] OR ata2[indis]== ata2[indis+1])
                devamEt=true
end while
```

Çaprazlama işlemi sonunda ders parçalarının olması gerektiğinden daha fazla atandığı ya da hiç atama yapılmayarak kaybolduğu gözlenmiştir. Bu durumu düzeltmek için üretilen her yeni çocuk birey tamir operatörüne sokulmuştur. Tamir işleminde eksik genler ve fazla atanan genler bulunarak eksik fazla atanan genler kromozomdan silinmiş ve eksik genlerde ilk atama yöntemine göre kromozoma yerleştirilmiştir. Tamir işlemi her zaman başarılı olamamaktadır. Eğer çocuk kromozomun genleri tamir edilemez ise kromozom popülasyona eklenmemiştir.

E. MUTASYON İŞLEMİ

```
birey ← mutasyon için seçilen kromozomu ifade eder
while(true)
        while(true) //rastgele bir nokta seç null konum seçilirse seçime devam et
                konum1=randomSayi (0, derslikSayısı*gün sayısı* günlük ders sayısı) //seçilen ilk konum
                dersParca1=birey[konum1].dersParca //seçilen konumdaki ders parçasını bulur
                if(dersParca1!=null) break // boş bir saat seçilirse yeni konum seç
        end while
        i=0
        while(dersParça==birey[konum1-i]) //ders parçasının başlangıç konumunu buluncaya kadar döner
                i=i+1
        end while
        dersKonumu1=konum1-i //ders parçasının atandığı ilk konumu bulur
        dersSaati1=dersParça1.saat // dersin kaç saat olduğu bulur
        konum2=randomSayi (0, derslikSayısı*gün sayısı* günlük ders sayısı)
        dersParca2=birey[konum2].dersParca //seçilen konumdaki ders parçasını bulur
        i=0
        while(dersParça2==birey[konum2-i]) //yeni seçilen ders parçasının başlangıç konumunu bul
                i=i+1
        end while
        dersKonumu2=konum2-i //ders parçasının atandığı ilk konumu
        i=1
        while(dersParça2==birey[dersKonumu2+i]) //dersin kaç saat olduğu bulunuyor
                i=i+1
        end while
        dersSaati2==i // ders parçası2'nin kaç saat olduğu bulundu, null değer ise kaç saatlik boşluk olduğu
        bulundu
        if (dersSaati1==dersSaati2)
                for (i=0;i<dersSaati1) // iki ayrı konumdaki ders parçalarını yer değiştirerek mutasyon uygula
                        gecici=birey[konum1+0]
                        birey[konum1+0]= birey[konu2+0]
                        birey[konu2+0]=geçici
                end for
                return //mutasyon işlemi yapıldı algoritmayı bitir
        end if
end while
```

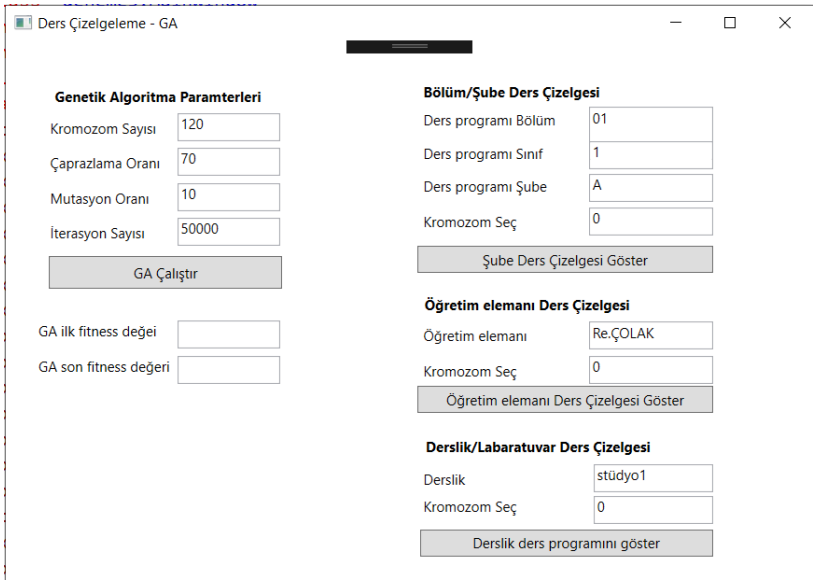
GA yapısı gereği lokal optimumlara takılması muhtemel bir algoritmadır. Bu durumun temel sebebi yeni bireylerin çaprazlama ile eski bireylerden elde edilmesidir. GA'nın lokal optimumlardan kurtulmasını sağlamak için mutasyon operatörü kullanılır [35]. Literatürdeki birçok çalışmada mutasyon operatörü yeni üretilen çocuk kromozomlara uygulanmıştır [29], [30]. Bu çalışmada ise mutasyon operatörü popülasyondaki bütün bireylere uygulanmıştır. Mutasyonun bu şekilde uygulanması popülasyondaki çeşitliliğin daha çok artmasını sağlamıştır. Mutasyonun tüm bireylere uygulanacak olması çözüme yakın olan kromozomların da bozulma ihtimalini ortaya çıkarmaktadır bunun önüne

geçmek için tamamen rastgele bir mutasyon uygulanmamış ve çözüme daha uzak bireylerin seçilme şansının yüksek olduğu rulet tekerine benzer bir yöntemler seçilen bireyler mutasyona uğramıştır. Bu şekilde diğer bireylere oranla daha iyi olan bireylerin genlerini olduğunu gibi diğer nesillere aktarma olasılığı yükselmiştir.

Bu çalışmada mutasyon işleminde yer değiştirme işlemi uygulanmıştır. Bu yer değiştirmede iki olasılık vardır bunlar ya yer değiştirecek ders parçalarının ders saatleri aynı olmalıdır ya da boş bir konuma bir ders parçası yer değiştirmelidir. Mutasyonun bu şeklide uygulanması ders parçalarının ortadan bölünme riskini de ortadan kaldırmaktadır. Kullandığımız mutasyon operatörüne ait algoritma yukarıda verilmiştir.

V. TESTLER VE DEĞERLENDİRME

Bu çalışmada üniversite ders programı çizelgelemesi için GA ile çözüm arayan bir uygulama geliştirilmiştir. Uygulamaya ait ekran görüntüsü **Şekil 6**'de görülmektedir. Uygulama Microsoft Visual Studio 2017'de WPF ve C# kullanılarak geliştirilmiştir. Uygulamanın kullandığı bütün veriler ve elde edilen ders çizelgeleri Microsoft SQL Sever 2018 de saklanmıştır.



Şekil 6. *GA ile ders çizelgeleme uygulama ekran görüntüsü*

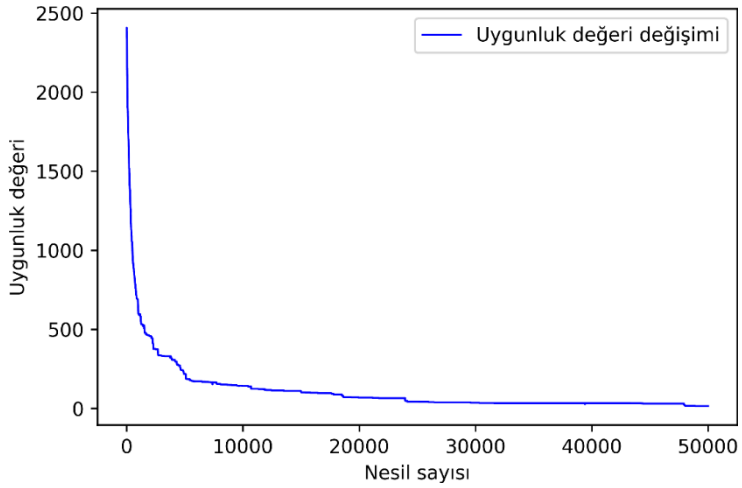
Şekil 6'da görüldüğü gibi geliştirilen GA parametreleri istenildiği gibi değiştirilebilmektedir. Geliştirilen GA, 95 bağımsız ders, 130 ders parçasına ait 276 ders saati, 40 ayrı öğretim elemanı, haftada 5 gün, günde 10 ders saati ve toplam 12 adet derslik olan örnek bir veri kümesi üzerinde çalıştırılmıştır. Testler İntel i7 9750H serisi bir CPU ve 16 GB belleğe sahip bir dizüstü bilgisayarda çalıştırılmıştır. Elde edilen test verileri Tablo 4'te görülebilir.

Tablo 4. *Test verileri ve elde edilen sonuçlar*

Test	KS	ÇO(%)	MO(%)	NS	İFD	SFD	KÇO(%)	TS (sn.)
1	80	80	5	50000	2409	98	95.91	3571
2	100	40	10	10000	2491	138	94.46	263
3	100	50	5	10000	2372	200	91.56	273
4	100	50	7	10000	2399	108	95.49	275
5	100	50	10	10000	2468	129	94.77	263
6	100	60	10	10000	2491	144	94.21	263
7	100	70	10	10000	2424	128	94.71	256
8	100	80	3	50000	2399	68	97.16	1485
9	100	80	5	10000	2376	149	93.72	300
10	100	80	7	50000	2259	63	97.21	1431
11	100	80	10	10000	2526	208	91.76	278
12	100	80	10	50000	2511	128	94.9	1404
13	100	90	5	50000	2432	265	89.1	1627
14	100	90	10	10000	2489	406	83.68	323
15	120	50	7	10000	2414	168	93.04	341
16	120	50	7	50000	2548	91	96.41	1628
17	120	60	7	50000	2390	91	96.19	1635
18	120	70	10	5000	2307	177	92,32	227
19	120	80	3	50000	2365	60	97.46	1811
20	120	80	5	50000	2485	55	97.78	1466
21	120	80	7	10000	2416	199	91.76	346
22	120	80	7	20000	2254	136	93.96	676
23	120	80	7	30000	2426	77	96.82	1034
24	120	80	7	40000	2477	62	97.49	1326
25	120	80	7	50000	2398	15	99.37	1708
26	150	50	5	10000	2511	148	94.1	426
27	150	50	5	20000	2501	124	95.04	828
28	150	50	7	10000	2270	97	95.72	427
29	150	50	10	10000	2447	97	96.03	402
30	150	80	7	50000	2394	59	97.53	2239

KS: Kromozom sayısı; **ÇO:** Çaprazlama oranı; **MO:** Mutasyon oranı; **NS:** Algoritmanın çalıştığı iterasyon sayısı; **İFD:** İlk nesil için en iyi çözüme sahip bireyin uygunluk değeri; **SFD:** Son nesil için en iyi çözüme sahip bireyin uygunluk değeri; **KÇO:** Kısıtların ne kadarının çözüme ulaştığını gösteren oran; **TS:** Algoritmanın saniye cinsinden çalışma süresi

GA doğası gereği rastsallık içerdiğinden aynı parametrelerle bile çalıştırılsa her seferinde farklı sonuçlar verebilmektedir. GA'nın bu durumundan dolayı her test verisi 10 defa denenmiş ve elde edilen en iyi sonuca ait veriler Tablo 4'te verilmiştir. Tablo incelendiğinde 25 nolu test verisinin oldukça başarılı olduğu ve kısıtların %99.37 sini çözdüğü görülmüştür. Şekil 7'de 25 nolu teste ait GA'nın uygunluk değerinin değişimi görülmektedir.



Şekil 7. *GA'nın uygunluk değeri değişimi*

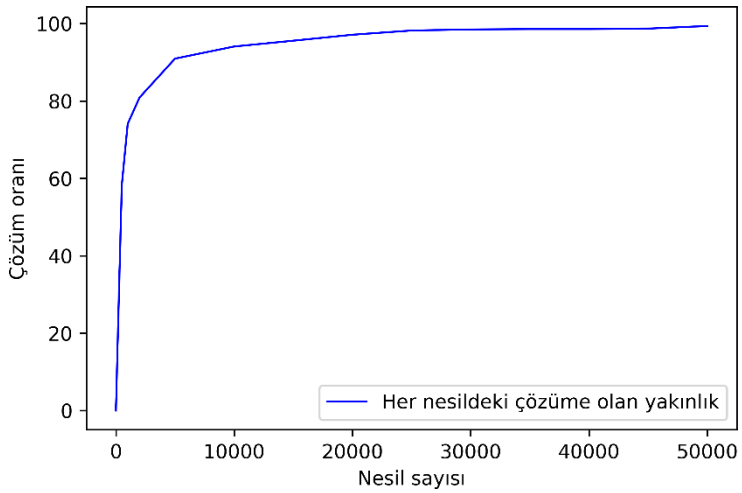
Şekil 7 incelendiğinde GA'nın başlangıçta hızlı bir iyileşme gerçekleştirerek kısıtları hızlı bir şekilde çözdüğü daha sonra ise iyileşmenin yavaşladığı gözlenmektedir. Tablo 5'te GA'nın çözüme hangi nesilde ne kadar yakınsadığını gösteren bir tablo görülmektedir.

Tablo 5. *GA'nın her nesilde çözüme olan uzaklığı ve geçen süre*

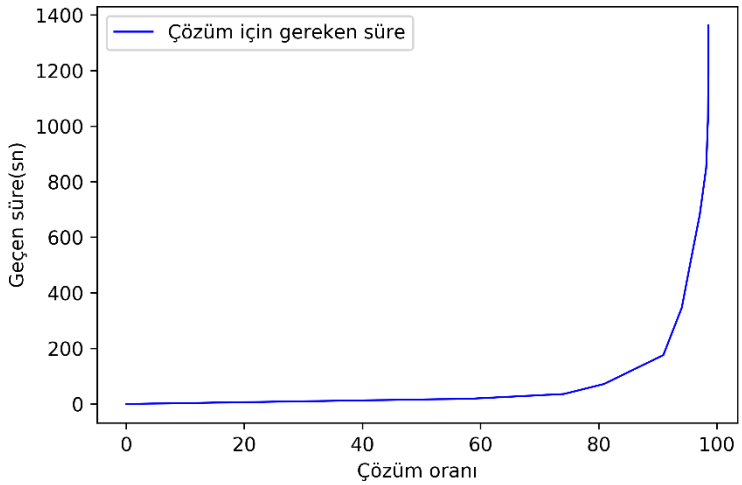
NS	FD	KÇO (%)	TS (sn.)
1	2398	0.00	0
500	1000	58.30	19
1000	621	74.10	36
2000	458	80.90	72
5000	217	90.95	176
10000	142	94.08	347
15000	106	95.58	514
20000	69	97.12	682
25000	43	98.21	850
30000	36	98.50	1019
35000	34	98.58	1191
40000	34	98.58	1362
45000	31	98.71	1535
50000	15	99.37	2368

NS: Nesil sayısı; **FD:** En iyi çözüme sahip bireyin uygunluk değeri

Tablo 5'te ki verilerin daha iyi anlaşılması için nesil sayısı-çözüm oranı arasındaki ilişkiyi gösteren grafik
Şekil ***8***'de, belirli bir çözüm oranına ulaşmak için gereken süreyi gösteren grafik
Şekil 9'da görülebilir.



Şekil 8. *GA'nın çözüme ne kadar yaklaştığı*



Şekil 9. *Çözüm için gereken süre*

VI. SONUÇLAR VE TARTIŞMA

Yapılan çalışma ile zor bir problem olan üniversite ders çizelgeleme problemine GA ile çözüm aranmıştır. İhlal edilen durumların cezalandırıldığı ve bu cezaların minimize edilmesine çalışan bir tasarım kullanılmıştır. Bu çalışmada ders yerine derslerin ders programına atanan parçaları dikkate alınmış ve bir ders birden fazla ders parçası olarak atanmıştır. Kullanılan kromozom yapısı derslik*periyod şeklinde iki boyutlu bir dizi olarak oluşturulmuştur. Kromozom yapısı sayesinde aynı dersliğe aynı anda birden fazla ders atanması tasarım ile çözülmüştür. İlk kromozom atamasında derslerin bölünmemesi, ders parçasının farklı günde ve farklı derslikte olmaması gibi durumlar da dikkate alındığından algoritmanın daha hızlı sonuca yaklaşmasını sağlamıştır. Kromozom sayısı, çaprazlama oranı ve mutasyon oranlarının farklı değerleri için farklı sonuçlar alınsa da yapılan bütün testlerde ilk atanan ders çizelgelerinde iyileştirmeler gözlenmiştir. Nesil sayısının artması daha fazla bölgenin aranmasını sağladığında her zaman sonucun küçükte olsa daha iyiye gitmesini sağlamıştır. Kromozom sayısının artmasının her zaman çözüme olumlu yansımadığı gözlenmiştir. Farklı kromozom sayıları için farklı çaprazlama ve mutasyon sayılarının çözüme daha yakın sonuçlar verebildiğini göstermiştir. Çözüme kavuşturulmak istenen veri kümesi ve kısıtlara göre en uygun kromozom sayısı, çaprazlama oranı ve mutasyon oranının değişiklik gösterdiği gözlenmiştir. Bizim veri kümemize göre

100-150 arası bir kromozom sayısı, 70-80 arası bir çaprazlama oranı ve 5-10 arası bir mutasyon oranının iyi sonuçlar verdiği söylenebilir. Ders çizelgeleme gibi gerçek hayat problemlerinde her zaman en iyi sonuca ulaşmak yerine makul bir sürede çözüme yakın sonuçlarda kabul edilebilmektedir. Problem bu açıdan incelendiğinde 5000 nesil ve 3 dakikadan az bir sürede kısıtların %90 nın çözülebildiği görülmüştür. Bundan sonraki çalışmalarda araştırmacılar kısıtları bir bütün olarak ele almak yerine veri kümesine göre daha zor ya da daha kolay çözülen kısıtları sınıflandırma yoluna giderek veri kümesine uygun kromozom tasarımı yapabilirler.

VII. KAYNAKLAR

[1] A. I. Diveev and O. V. Bobr, "Variational genetic algorithm for np-hard scheduling problem solution," *Procedia Computer Science*, vol. 103, pp. 52–58, 2017.

[2] A. Muklason, R. G. Irianti, and A. Marom, "Automated course timetabling optimization using tabu-variable neighborhood search based hyper-heuristic algorithm," *Procedia Computer Science*, vol. 161, pp. 656–664, 2019.

[3] K. Alomari, O. Almarashdi, A. Marashdh, and B. Zaqaibeh, "A new optimization on harmony search algorithm for exam timetabling system," *Journal Of Information And Knowledge Management*, vol. 19, no. 1, pp. 202009_1-202009_13, 2020.

[4] M. Chen, X. Tang, T. Song, C. Wu, S. Liu, and X. Peng, "A tabu search algorithm with controlled randomization for constructing feasible university course timetables," *Computers & Operations Research*, vol. 123, pp. 105007, 2020.

[5] A. Gülcü And C. Akkan, "Robust university course timetabling problem subject to single and multiple disruptions," *European Journal Of Operational Research*, vol. 283, no. 2, pp. 630–646, 2020.

[6] K. Patrick and Z. Godswill, "Greedy ants colony optimization strategy for solving the curriculum based university course timetabling problem," *British Journal Of Mathematics & Computer Science*, vol. 14, no. 2, pp. 1–10, 2016.

[7] E. A. Abdelhalim and G. A. El Khayat, "A utilization-based genetic algorithm for solving the university timetabling problem (uga)," *Alexandria Engineering Journal*, vol. 55, no. 2, pp. 1395–1409, 2016.

[8] M. W. Carter, "A comprehensive course timetabling and student scheduling system at the university of waterloo," *Practice and Theory of Automated Timetabling III. PATAT 2000*, 2000 pp. 64–82

[9] M. Shahvali Kohshori and M. Saniee Abadeh, "Hybrid genetic algorithms for university course timetabling," *International Journal of Computer Science Issues (IJCSI)*, vol. 9, no. 2, pp. 446-455, 2012.

[10] T. Yiğit, "Meslek liseleri haftalık ders çizelgelerinin genetik algoritmalar yardımıyla oluşturulması," *Gazi Üniversitesi Endüstriyel Sanatlar Eğitim Fakültesi Dergisi*, vol. 19, pp. 25–39, 2004.

[11] M. A. Cruz-Chávez, M. Flores-Pichardo, A. Martínez-Oropeza, P. Moreno-Bernal, And M. H. Cruz-Rosales, "Solving a real constraint satisfaction model for the university course timetabling problem: a case study," *Mathematical Problems İn Engineering*, vol. 2016, 2016.

[12] J. H. Obit, K. Y. Junn, and R. Alfred, "A performance comparison of metaheuristics search for university course timetabling problems," *Advanced Science Letters*, vol. 23, no. 11, pp. 11012–11015,

2017.

[13] K. Y. Junn, J. H. Obit, and R. Alfred, "A constraint programming approach to solving university course timetabling problem (uctp)," *Advanced Science Letters*, vol. 23, no. 11, pp. 11023–11026, 2017.

[14] M. Lindahl, A. J. Mason, T. Stidsen, and M. Sørensen, "A strategic view of university timetabling," *European Journal Of Operational Research*, vol. 266, no. 1, pp. 35–45, 2018.

[15] T. L. June, J. H. Obit, Y.-B. Leau, and J. Bolongkikit, "Implementation of constraint programming and simulated annealing for examination timetabling problem," *Computational Science and Technology*, pp. 175–184, 2019.

[16] R. Çolak, "Sezgisel algoritmalarla ders programı çizelegeleme probelmi çözümü," YL tezi, Bilgisayar Mühendisliği Bölümü, Süleyman Demirel Üniversitesi, Isparta, Türkiye, 2015.

[17] M. C. Chen, S. N. Sze, S. L. Goh, N. R. Sabar, and G. Kendall, "A survey of university course timetabling problem: perspectives, trends and opportunities," *IEEE Access*, vol. 9, pp. 106515–106529, 2021.

[18] D. De Werra, A. S. Asratian, and S. Durand, "Complexity of some special types of timetabling problems," *Journal Of Scheduling*, vol. 5, no. 2, pp. 171–183, 2002.

[19] E. K. Burke, B. Mccollum, A. Meisels, S. Petrovic, and R. Qu, "A graph-based hyper-heuristic for educational timetabling problems," *European Journal Of Operational Research*, vol. 176, no. 1, pp. 177–192, 2007.

[20] N. L. Lawrie, "An integer linear programming model of a school timetabling problem," *The Computer Journal*, vol. 12, no. 4, pp. 307–316, 1969.

[21] N. Boland, B. D. Hughes, L. T. G. Merlot, and P. J. Stuckey, "New integer linear programming approaches for course timetabling," *Computers & Operations Research*, vol. 35, no. 7, pp. 2209–2233, 2008.

[22] G. B. Bucco, C. J. Bornia-Poulsen, and D. L. Bandeira, "Development of a linear programming model for the university course timetabling problem," *Gestao E Producao*, vol. 24, no. 1, pp. 40–49, 2017.

[23] J. M. Maldonado-Matute, M. J. González Calle, and R. M. Celi Costa, "Development of a solution model for timetabling problems through a binary ınteger linear programming approach," *Intelligent Human Systems Integration*, vol. 1131, pp. 510–516, 2020.

[24] O. Kaynar ve A. Yurtsal, "Ders programıçizelgeleme probleminin genetik algoritma ile optimizasyonu," *Journal Of Information Systems And Management Research*, vol. 1, no. 1, pp. 9–14, 2019.

[25] A. Colorni, M. Dorigo, and V. Maniezzo, "A genetic algorithm to solve the timetable problem," *Politecnico Di Milano*, pp. 90-060, 1992

[26] M. A. Mohammed, M. Khanapi, A. Ghani, O. I. Obaid, and S. Mostafa, "A review of genetic algorithm application in examination timetabling problem," *Journal Of Engineering And Applied Sciences*, vol. 12, no. 20, pp. 5166–5181, 2017.

[27] S. Naseem Jat and S. Yang, "A guided search genetic algorithm for the university course timetabling problem," *4th Multidisciplinary International Conference On Scheduling : Theory And Applications (MISTA 2009), Dublin, Ireland*, 2009, pp. 180-191.

[28] W. Herbawi and M. Weber, “A genetic and ınsertion heuristic algorithm for solving the dynamic ridematching problem with time windows,” *Proceedings Of The 14th International Conference On Genetic And Evolutionary Computation*, New York, USA, 2012, pp. 385-392.

[29] A. A. Gozali, B. Kurniawan, W. Weng, and S. Fujimura, “Solving university course timetabling problem using localized island model genetic algorithm with dual dynamic migration policy,” *IEEJ Transactions On Electrical And Electronic Engineering*, vol. 15, no. 3, pp. 389–400, 2020.

[30] C. Akkan and A. Gülcü, “A bi-criteria hybrid genetic algorithm with robustness objective for the course timetabling problem,” *Computers And Operations Research*, vol. 90, pp. 22–32, 2018.

[31] A. A. Mahiba and C. A. D. Durai, “Genetic algorithm with search bank strategies for university course timetabling problem,” *Procedia Engineering*, vol. 38, pp. 253–263, 2012.

[32] J. H. Holland, “Genetic algorithms and the optimal allocation of trials,” *Siam Journal On Computing*, vol. 2, no. 2, pp. 88–105, 1973.

[33] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, And Machine Learning*, Addison-Wiley, 1989.

[34] G. Panchal and D. Panchal, “Solving np hard problems using genetic algorithm,” *International Journal Of Computer Science And Information Technologies*, vol. 6, no. 2, pp. 1824–1827, 2015.

[35] Babaei, H., Karimpour, J.,Hadidi, A. “A survey of approaches for university course timetabling problem, ” *Computers & Industrial Engineering*, vol. 86, pp. 43-59, 2015.

[36] Rezaeipanah, A., Matoori, S. S., and Ahmadi, G. “A hybrid algorithm for the university course timetabling problem using the improved parallel genetic algorithm and local search,” *Applied Intelligence*, vol.51, pp.467-492, 2021

[37] A. J. Umbarkar and P. D. Sheth, “Crossover operators in genetıc algorithms: a revıew,” *Ictact Journal On Soft Computing*, vol. 6 pp. 1083-1092, 2015.