ÇOK OTURUMLU SINAVLARDA

KİTAPÇIK OPTİMİZASYONU

Yüksek Lisans Tezi

Emine TUTSUN

Eskişehir, 2018

ÇOK OTURUMLU SINAVLARDA KİTAPÇIK OPTİMİZASYONU

EMİNE TUTSUN

YÜKSEK LİSANS TEZİ

ENDÜSTRİ MÜHENDİSLİĞİ ANABİLİM DALI

Danışman: Dr. Öğr. Üyesi Zehra KAMIŞLI ÖZTÜRK

Eskişehir

Anadolu Üniversitesi

Fen Bilimleri Enstitüsü

Mayıs, 2018

Bu tez çalışması BAP Komisyonunca kabul edilen 1706F390 no.lu proje kapsamında desteklenmiştir.

JÜRİ VE ENSTİTÜ ONAYI

Emine TUTSUN 'un "***Çok Oturumlu Sınavlarda Kitapçık Optimizasyonu*** " başlıklı tezi 21/05/2018 tarihinde aşağıdaki jüri tarafından değerlendirilerek "Anadolu Üniversitesi Lisansüstü Eğitim-Öğretim ve Sınav Yönetmeliği'nin ilgili maddeleri uyarınca, **Endüstri Mühendisliği** Anabilim dalında Yüksek Lisans Yeterlik tezi olarak kabul edilmiştir.

	Unvanı-Adı Soyadı	**İmza**
Üye (Tez Danışmanı)	**: Dr. Öğr. Üyesi Zehra KAMIŞLI ÖZTÜRK**	
Üye	**: Doç.Dr. Onur KAYA**	
Üye	**: Dr.Öğr.Üyesi Neslihan İYİT**	

.....

Enstitü Müdürü

ÖZET

ÇOK OTURUMLU SINAVLARDA KİTAPÇIK OPTİMİZASYONU

Emine TUTSUN

Endüstri Mühendisliği Anabilim Dalı

Anadolu Üniversitesi, Fen Bilimleri Enstitüsü, Mayıs, 2018

Danışman: Dr. Öğr. Üyesi Zehra KAMIŞLI ÖZTÜRK

Ders çizelgeleme ve sınav çizelgeleme problemleri eğitimsel zaman çizelgeleme problemlerinin alt kümesi olan ve akademik camialarda sıklıkla karşılaşılan optimizasyon problemleridir. Bu tarz problemler sınıf kapasitelerinin aşılmaması, kullanılan tüm sınıflara gözetmen atamasının yapılması, ortak derslere sahip öğrencilerin farklı sınav oturumlarına atanması gibi kısıtlardan oluşurken, amaç fonksiyonları da ders ataması yapılan günlerin en küçüklenmesi ya da sınav yapılacak günlerin en küçüklenmesi gibi işlemlerden oluşmaktadır. Bu çalışmada, her sınav döneminde dört oturumda gerçekleştirilen Anadolu Üniversitesi Açıköğretim Fakültesi sınavları için en az kitap türüne ulaşmayı amaçlayarak derslerin, sınav oturumlarına ve sınav kitaplarına yerleştirilme işlemi olan oturum düzeni oluşturulmuştur. Açıköğretim Fakültesi sistemi sınavlarının büyük öğrenci kitlesine sahip olması sebebiyle, çalışma sonunda elde edilecek ufak bir iyileşmenin ekonomik açıdan önemli katkı sağlayacağı öngörülmüştür. Bu çalışma kapsamında, sistemin tüm kısıtları araştırılarak 0-1 tam sayılı bir matematiksel model kurulup sonuçları incelenmiştir. Problemin karmaşıklığı dolayısıyla matematiksel model en iyi çözümü veremediğinden bir sezgisel algoritma geliştirilmiştir. Algoritmanın sonuçları mevcut sistemin sonuçları ile karşılaştırılıp sonuçlar yorumlanmıştır.

Anahtar Sözcükler: Optimizasyon, Matematiksel modelleme, Eğitimsel zaman çizelgeleme

ABSTRACT

BOOKLET OPTIMIZATION IN MULTIPLE SESSION EXAMS

Emine TUTSUN

Department of Industrial Engineering

Anadolu University, Graduate School of Science, May, 2018

Supervisor: Asst.Prof.Dr. Zehra KAMISLI OZTURK

Course scheduling and exam scheduling problems are optimization problems that are a subset of educational scheduling problems and frequently encountered in the academic community. These types of problems consist of constraints such as not exceeding class capacities, assigning supervisors to all classes used, assigning students with common courses to different examination sessions, while objective functions consist of minimizing the days of course assignment or minimizing the days of exams. In this study, aiming at reaching at least booklet for Anatolian University Open Education Faculty exams held in four sessions in each examination period, session planning which is the process of the placement of the lessons into the exam sessions and exam books has been established. Because of the large student population of the Open Education Faculty system examinations, it is predicted that a small improvement at the end of the study will make a significant contribution to the economy. For this process, all the constraints of the system are investigated and an 0-1 integer mathematical model is established and the results are examined. Because of the complexity of the problem, the mathematical model can not give the best solution, so an heuristic algorithm has been developed. The results of the algorithm are compared with the results of the current system and the results are interpreted.

Keywords: Optimization, Mathematical modeling, Educational timetabling

TEŞEKKÜR

Çalışmalarım süresince bilgi ve deneyimleriyle bana yol gösteren, değerli tez danışmanım Sayın Dr. Öğr. Üyesi Zehra Kamışlı Öztürk'e en içten teşekkürlerimi sunarım.

Bugünlere gelmemde nice emekleri olan ve her koşulda beni destekleyen sevgili annem Hafize Çokşen'e, sevgili babam Süleyman Çokşen'e ve sevgili teyzem Ayşe Kazak'a teşekkürü bir borç bilirim.

Çalışmalarım boyunca desteğini hiçbir zaman esirgemeyen ve her koşulda yanımda olan sevgili eşim Aşır Tutsun'a teşekkür ediyorum.

Aynı zamanda, çalışmalarım süresince bana destek veren BAUM'daki mesai arkadaşlarıma da teşekkürlerimi sunuyorum.

Emine TUTSUN

Mayıs 2018

07.06.2018

ETİK İLKE VE KURALLARA UYGUNLUK BEYANNAMESİ

Bu tezin bana ait, özgün bir çalışma olduğunu; çalışmamın hazırlık, veri toplama, analiz ve bilgilerin sunumu olmak üzere tüm aşamalardan bilimsel etik ilke ve kurallara uygun davrandığımı; bu çalışma kapsamında elde edilemeyen tüm veri ve bilgiler için kaynak gösterdiğimi ve bu kaynaklara kaynakçada yer verdiğimi; bu çalışmanın Anadolu Üniversitesi tarafından kullanılan "bilimsel intihal tespit programıyla tarandığını ve hiçbir şekilde "intihal içermediğini" beyan ederim. Herhangi bir zamanda, çalışmamla ilgili yaptığım bu beyana aykırı bir durumun saptanması durumunda, ortaya çıkacak tüm ahlaki ve hukuki sonuçlara razı olduğumu bildiririm.

İÇİNDEKİLER

TABLOLAR DİZİNİ

ŞEKİLLER DİZİNİ

1. GİRİŞ

Anadolu Üniversitesi Açıköğretim Fakültesi 1982 yılında, açık ve uzaktan eğitim alanında Türkiye'ye hizmet vermek için açılan ilk fakültedir. Zaman ilerledikçe ülke içinde hizmet vermenin yanı sıra, yurt dışı programlarının açılmasıyla birlikte dünya çapında hizmet verir hale gelmiştir. Her geçen yıl artan öğrenci sayısıyla Açıköğretim Fakültesi gittikçe büyümüş ve kendini öğrencilerin ihtiyaçları doğrultusunda sürekli geliştirmiştir. Bir anlamda artan öğrenci sayısı ve öğrencilerin ihtiyaçları sistemi sürekli gelişmeye açık hale getirmiştir.

Tez kapsamında, Anadolu Üniversitesi Açıköğretim sistemi sınavları için hazırlanan sınav oturum düzenlerinde, öğrencilere atanan kitap türü sayısının optimize edilmesi amaçlanmıştır. Burada optimizasyon işlemi sadece mevcut sistemdeki sınavların kitap türlerinin sayısının iyileştirilmesi olarak düşünülmemiş, yeni uygulanabilecek sınavlar da göz önünde bulundurularak bir çalışma tasarlanmıştır. Tez kapsamında seçilen bir yurt dışı programına ait bir eğitim dönemi dönem sonu sınavı verileri örneklem veri olarak alınmıştır. Kitap türü sayısının en küçüklenmesi hedeflenerek, kitapların sıfırdan oluşturulması ve derslerin sınav kitaplarına ve sınav oturumlarına en uygun şekilde yerleştirilmesi amaçlanmıştır. Bu problemin seçilme amacı, Anadolu Üniversitesi Açıköğretim sistemi Türkiye ve yurt dışı programlarına kayıtlı öğrenci sayısının çok fazla olması ve buna bağlı olarak ufak bir iyileştirmenin sağlayacağı katkının çok büyük ve önemli olacağının düşünülmesidir.

Bu tez kapsamında ele alınan kitapçık optimizasyonu problemi, eğitimsel zaman çizelgeleme problemleri altında ele alınmış ve literatürde bu alanda daha önce yapılan çalışmalar incelenmiştir. Zaman çizelgeleme problemlerinin alt kümesi olarak ele alınan eğitimsel zaman çizelgeleme problemleri, akademik çalışma ortamlarında sıklıkla karşılaşılan optimizasyon problemlerindendir. Bu problemler genellikle çok boyutlu yapıdadırlar ve öğrencilere derslerinin eksiksiz şekilde atanması, salon kapasitelerinin aşılmaması, ders/sınav ataması yapılan tüm salonlara gözetmen atanması ve bunlara benzer kısıtlardan oluşmaktadır.

Çalışmanın ikinci bölümünde literatür araştırmaları ele alınmıştır. Yapılan literatür araştırmasında zaman çizelgeleme, eğitimsel zaman çizelgeleme, ders zaman

çizelgeleme, sınav çizelgeleme problemleri ve bu problemler için yıllar içinde geliştirilen çözüm yaklaşımları incelenmiştir.

Üçüncü bölümde Anadolu Üniversitesi Açıköğretim sistemi ve sınavları hakkında bilgi verilmiş, devamında mevcut sistemde sınav oturum düzenlerinin nasıl hazırlandığı ve problemin tanımı konularına yer verilmiştir.

Dördüncü bölümde çok oturumlu sınavlarda kitapçık optimizasyonu problemi için kurulan matematiksel model incelenmiş ve GAMS yazılımı ile çözümünden elde edilen sayısal sonuçlar analiz edilmiştir.

Matematiksel modelin, problemin karmaşıklığı nedeniyle bu problemin çözümünde yetersiz kaldığı görülmüş ve bir sonraki adım olarak bir sezgisel algoritma geliştirilmiştir. Problem için geliştirilen sezgisel algoritma adımları ve algoritma sonuçlarının analizleri beşinci bölümde incelenmiştir.

Altıncı bölümde tez kapsamında yapılan tüm çalışmalar bir bütün olarak değerlendirilip elde edilen sonuçlar yorumlanmış ve gelecek çalışmalar için öneriler belirtilmiştir.

2. LİTERATÜR

Tez kapsamında ele alınan kitapçık optimizasyonu problemi, araştırmalarımız dâhilinde literatürde henüz birebir rastlanan bir çalışma değildir. Ancak bu çalışma, zaman çizelgeleme problemlerinin alt kümesi olan eğitimsel zaman çizelgeleme problemi olarak ele alınmış ve literatürde bu problemle ilgili yer alan çalışmalar incelenmiştir.

Bu bölümde, tez kapsamında ele aldığımız kitapçık optimizasyonu ile ilgili olarak zaman çizelgeleme problemleri, eğitimsel zaman çizelgeleme problemleri, ders zaman çizelgeleme problemleri, sınav zaman çizelgeleme problemleri ve bu problemler için geliştirilen çözüm yöntemleri hakkında yapılan çalışmalar yer almaktadır.

2.1 Zaman Çizelgeleme Problemleri

Zaman çizelgeleme problemleri optimizasyon problemlerinden bir tanesidir. Bu problem türlerinin hedefinde, belirlenen kurallar çerçevesinde olmak koşuluyla, olayları belirli bir zamana, en verimli olacak şekilde yerleştirmek söz konusudur. Hastahanelerde hemşirelerin ve diğer sağlık çalışanlarının nöbet çizelgelerinin oluşturulması, televizyon kanallarında yayın akışlarının belirlenmesi, spor müsabakalarının planlanması, eğitim kurumlarında ders ve sınav çizelgelerinin belirlenmesi gibi günlük hayatta sıkça karşılaşılan problemler, zaman çizelgeleme problemlerine örnek olarak verilebilir.

Zaman çizelgeleme problemleri ile birçok farklı alanda karşı karşıya gelinmektedir, ancak genellikle eğitim kurumlarında sınavların ve derslerin çizelgelenmesinde kullanılmaktadır. Bu problemlerin önemi çoğunlukla yönetimin, gözetmenlerin ve öğrencilerin tercihlerini göz önünde bulunduracak uygun bir zaman çizelgelemesi ortaya koymaktan kaynaklanmaktadır (Botsalı, 2000).

Diğer optimizasyon problemlerinde olduğu gibi zaman çizelgeleme problemleri de belli kısıtlar altında çözülür. Bu kısıtlar katı kısıt ve esnek kısıt olarak adlandırılan iki tür kısıttan oluşur. Katı kısıt olarak adlandırılan kısıtlar, modelde gerçekleştirilmesi zorunlu kısıtlar iken, esnek kısıtlar gerçekleştirilmeleri zorunlu olmayan ancak gerçekleşmeleri durumunda modele fayda sağlayan kısıtlardır. Burke vd. (2005) çalışmalarında, zaman çizelgeleme problemleri için katı ve esnek kısıtların tanımlamaları aşağıdaki gibidir:

- Katı kısıtlar herhangi bir durum altında mutlaka gerçekleşmesi gereken kısıtlardır. Örneğin, bir zaman çizelgelemesinde, büyük öğrenci grupları tarafından alınan iki ders

aynı zaman dilimine atanamaz. Katı kısıtların ihlal edilmediği zaman çizelgelemeleri uygun çözümler olarak adlandırılır.

- Esnek kısıtlar mümkün olduğunca gerçekleştirilmelidir. Örneğin, bir sınav çizelgelemesinde, ortak öğrenciler tarafından alınan derslerin zaman periyotlarına yayılması istenir, böylelikle öğrenciler birbirlerine çok yakın iki sınava girmek zorunda kalmazlar.

2.2 Eğitimsel Zaman Çizelgeleme Problemleri

Eğitimsel zaman çizelgeleme problemleri, zaman çizelgeleme problemlerinin alt kümesi olarak görülebilirler ve eğitim kurumlarındaki derslerin ve sınavların çizelgelenmesi işlemleri bu problem kümesinin elemanlarını oluşturur.

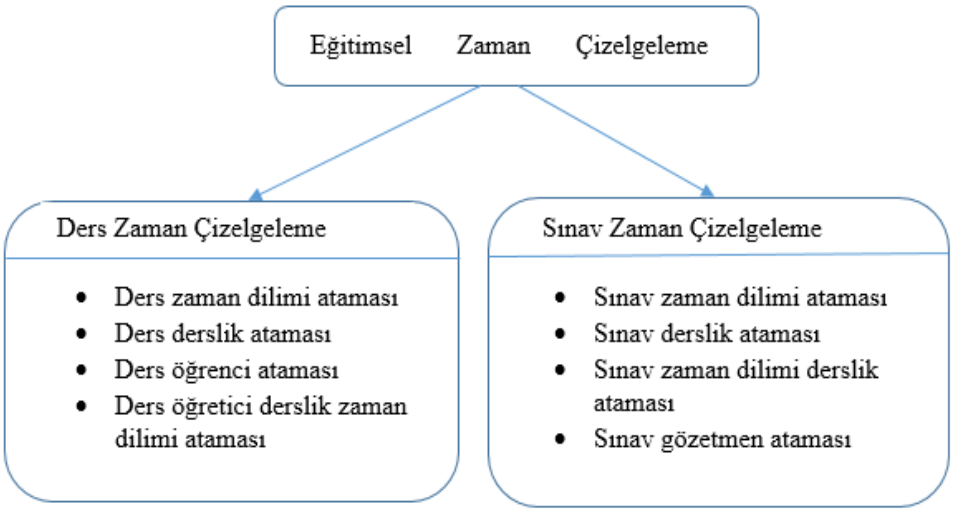
Uzun yıllar boyunca eğitimsel zaman çizelgeleme problemlerinin çözümü için çeşitli yollara başvurulmuştur. Ancak bu tip problemlere özgü geliştirilen ve birebir uygulanabilen tek bir yöntem mevcut değildir. İhtiyaç ve taleplerin kurumlara özgü olması, problem boyutlarının genellikle çok büyük olması ve kurumdan kurumda kısıtların ve amaçların farklılık göstermesi, tüm problem tipleri için uygulanabilecek tek bir çözüm yönteminin mevcut olmamasını açıklayan sebeplerdir.

Eğitimsel zaman çizelgeleme problemleri, yıllar içinde çeşitli kategorilere ayrılarak incelenmiştir. Literatürde bu konudaki çalışmalara bakıldığında en yaygın çalışmaların Burke vd. (2004), Schaerf (1999) ve Schaerf vd. (2001) olduğu görülmektedir.

Schaerf (1999)'in çalışmasına göre eğitimsel zaman çizelgeleme problemleri üç kısımdan oluşmaktadır:

- Okul zaman çizelgeleme (*school timetabling*): Öğretmenlerin aynı zaman diliminde iki derse girmesini engelleyecek şekilde, derslerin haftalık çizelgelerinin oluşturulması,
- Ders zaman çizelgeleme (*course timetabling*): Üniversitedeki derslerin haftalık olarak çizelgelerinin oluşturulması,
- Sınav zaman çizelgeleme (*exam timetabling*): Üniversitelerde büyük öğrenci kitleleri tarafından alınan derslerin çakışmasını engelleyecek şekilde, sınav çizelgelerinin oluşturulması

Öztürk (2010) yaptığı çalışmada, eğitimsel zaman çizelgeleme problemlerini ikiye ayırmış ve bu yaptığı sınıflandırmanın da hangi alt konularla ilgilendiğini belirtmiştir. Bu sınıflandırma Şekil 2.1'de gösterilmiştir.



Şekil 2.1 *Eğitimsel Zaman Çizelgeleme Problemi Sınıflandırması (Öztürk, 2010)*

2.2.1 Ders çizelgeleme problemleri

Ders çizelgeleme problemleri, çizelgeleme problemlerinin özel bir türüdür ve eğitimsel zaman çizelgeleme problemlerinin altında incelenir. Carter ve Laporte (1997) tarafından yapılan çalışmada çok boyutlu bir atama problemi olarak tanımlanan ders çizelgeleme, belirli kısıtlar altında olacak şekilde derslerin haftalık olarak dersliklere ve zaman periyotlarına atanması işlemidir.

Ders çizelgelemesi sadece birkaç genel prensip ile çözülemeyecek, karmaşık bir problem türüdür. Probleme dâhil olan tüm aktörlerin (yöneticiler, öğretim elemanları, öğrenciler) kendilerine özel amaçları vardır ve bu amaçlar genellikle birbiri ile çelişmektedir. Dersler, derslikler, zaman dilimleri, eğiticiler ve öğrenciler arasındaki karmaşık ilişkiler uygun bir çözüm bulabilmeyi zorlaştırmaktadır Lai vd. (2006).

Ders zaman çizelgeleme problemlerinin çözümleri katı ve esnek kısıtlar altında gerçekleşmektedir. Literatüre bakıldığında Burke ve Petrovic (2002), üniversite zaman çizelgelemeyi ele alarak yaptıkları çalışmada katı ve esnek kısıtları aşağıdaki gibi belirlemişlerdir.

Katı kısıtlar:

Tüm kaynaklar (öğrenci, öğretmen, ders) bir zaman diliminde sadece bir yerde bulunabilir. Bu kısıtın örnek bileşenleri de açıklamalarıyla birlikte aşağıda verilmiştir.

- Ders - Zaman dilimi: bir ders, sadece bir zaman dilimine atanabilir,
- Ders - Derslik: bir ders, sadece bir dersliğe atanabilir,
- Ders - Öğrenci - Zaman dilimi: bir ders bir öğrenciye sadece bir zaman diliminde atanabilir,
- Ders - Öğretici - Derslik - Zaman dilimi: bir öğretici, bir zaman diliminde, bir derslikte ve bir ders ile görevlendirilebilir,

Diğer katı kısıt ise her zaman dilimi için çizelgelemenin yapılabilmesi için gerekli kaynakların(sınıf, laboratuvar, öğretici) yeterli miktarda hazır olmasıdır.

Esnek kısıtlar:

Esnek kısıtlar ise, arzu edilen ancak kesinkes gerçekleşmesi gereken kısıtlar değildir. Gerçek dünya problemlerinde tüm esnek kısıtları sağlamak mümkün değildir, bazı esnek kısıt örnekleri aşağıda verilmiştir:

- Bir ders, zaman periyotları içinde belirli bir tanesine atanabilir,
- Bir ders, diğer bir dersin öncesinde veya sonrasında atanabilir,
- Öğrencilerin aynı gün içinde çok fazla ders almasını engellemek için derslerin hafta içine yayılacak şekilde atamaları yapılabilir,
- Öğreticiler tüm derslerini belirli bazı günlerde vermek ve boş günlere sahip olmak isteyebilirler ancak bu kısıt derslerin günlere yayılması kısıtı ile çakışmaktadır,
- Öğreticiler belirli dersliklerde ders vermeyi tercih edebilir.

Yukarıdaki katı ve esnek kısıtlardan da anlaşılacağı gibi bu tarz problemlerde çok fazla bileşen ve bu bileşenlerin etkileşimleriyle birlikte çok fazla katı kısıt da söz konusudur. Bu tarz durumlar ders çizelgeleme problemlerinin karmaşık yapısını açıklar niteliktedir. Çözümü kolaylaştırmak ve etkin sonuçlar almak için de ders çizelgeleme problemleri genellikle birden fazla aşamada, ders-derslik atama, ders-öğretici atama gibi alt problemler şeklinde ele alınıp çalışılmaktadır.

2.2.2 Sınav çizelgeleme problemleri

Sınav çizelgeleme problemleri basit bir tabirle, gerçekleştirilecek sınavların sınav periyotlarına yerleştirilme işlemidir. Bu işlem yapılırken de ortak öğrenciler tarafından alınan sınavların aynı zaman dilimine atanmaması ve sınıf kapasitelerinin aşmaması gibi kısıtlar göz önünde bulundurulur.

Burke vd. (1996) yılında İngiltere'deki 95 üniversiteye zaman çizelgeleme ile ilgili hazırladıkları bir anketi göndermişler ve 56 üniversiteden gelen cevaplara göre araştırmalarını gerçekleştirmişlerdir. Bu anketle birlikte problemlerin karmaşıklık, boyut, kısıt gibi yapıları, problemlerin nasıl çözülmesi gerektiği ve de problemlerin amaç veya amaçlarının ne olması gerektiği konularına cevap bulmayı amaçlamışlardır.

Sınav çizelgeleme problemlerinde amaç, sınavların çakışmasını engelleyecek şekilde sınavları en az zaman dilimine yerleştirmek olabileceği gibi, zaman dilimlerinin sayısını sabit tutarak eğiticilerin ve öğrencilerin ardışık katılacağı sınavların arasındaki süreyi mümkün olduğunca artırmak ya da büyük öğrenci grupları tarafından alınan derslerin daha önceden planlanması gibi kısıtları en büyüklemeye çalışmak da olabilmektedir Botsalı (2000).

Sınav çizelgeleme problemleri de diğer çizelgeleme problemleri gibi katı ve esnek kısıtlar altında çözülmektedir. Literatüre bakıldığında Merlot vd. (2003) yaptıkları çalışmada, kısıtların yaygın biçimlerini aşağıdaki gibi sıralamışlardır:

- Her öğrenci bir oturumda sadece bir sınava katılabilir,
- Tüm sınavlarda aynı oturumda ve aynı sınıfta sınava girecek öğrencilerin toplam sayısı, sınıfın kapasitesinden fazla olamaz,
- Tüm sınavlarda aynı oturumlarda sınava girecek öğrencilerin toplam sayısı, oturumun kapasitesinden fazla olamaz,
- Bir oturumda gerçekleştirilecek sınavların sayısı belirlenen sayıdan az olmalıdır,
- Bazı sınavların bazı özel oturumlara atamaları önce yapılabilir ya da bu sınavlar sadece belirli oturumlarda geçekleştirilebilir,
- Bazı sınıflar sadece bazı oturumlar için uygundur,
- Bazı sınav çiftleri birbirlerine göre planlanabilir (birbirinden önce ya da sonra atanma gibi),
- Bazı sınavlar bazı özel sınıflarda gerçekleştirilebilir,

- Bazı öğrencilerin sınav çizelgeleme tablolarında kısıtlamalar yapılabilir (hiçbir öğrenci ardışık üç sınav oturumunda iki sınava katılamaz),
- Çok fazla öğrencisi bulunan sınavlar, sınav periyodundaki ilk periyotlara atanmalıdır.

Üniversitelerde çizelgeleme problemleri de ders ve sınav çizelgeleme olarak iki aşamada incelenmektedir. Örneğin ders çizelgeleme problemlerinde bir dersin sadece bir sınıfa atanması gerekirken, sınav çizelgeleme problemlerinde bazı sınavların bir sınıfa atanması gerekebilir ya da sınavın birden fazla sınıfa bölünmesi gerekebilir. Ders çizelgeleme problemlerinde öğrencilerin art arda iki veya daha fazla ders alması istenirken, sınav çizelgeleme problemlerinde genellikle ardışık periyotlarda sınava girecek öğrencilerin sayısının azaltılması amaçlanır Burke ve Petrovic (2002).

2.3 Çözüm Yaklaşımları

Eğitimsel zaman çizelgeleme problemlerinin çözümünde kullanılmak için yıllar içerisinde çeşitli yöntemler geliştirilmiştir. Her kurumun kısıtlarının ve amaçlarının farklı olması ve de kendine özgü yapılarının olması, bilim insanlarını yeni çözüm yolları geliştirmeye yönlendirmiştir.

Genellikle bir çözüm yaklaşımı için en iyisi demek neredeyse imkânsızdır. Bir kurumdan diğerine değişen kısıtlar sebebiyle, bir problem için çok iyi sonuç veren çözüm yaklaşımının, diğer problem için güçsüz kalması söz konusu olabilmektedir. Bazen de birden fazla çözüm yaklaşımının kombinasyonu en iyi sonucu verebilmektedir Botsali (2000).

Literatüre bakıldığında, eğitimsel zaman çizelgeleme problemlerinin ilk olarak elle çözüldüğü görülmektedir. Elle çözümün fazlaca yetersiz gelmesiyle zaman içinde çeşitli yöntemler üzerinde çalışılmaya başlanmıştır.

Bu bölümde eğitimsel zaman çizelgeleme problemlerinin çözüm yaklaşımlarından olan, matematiksel programlama yaklaşımları, grafik renklendirme yaklaşımları, çok amaçlı modelleme yöntemleri, kısıt programlama temelli yaklaşımlar, sezgiseller ve metasezgiseller ele alınacaktır. Literatürdeki problem çözüm yaklaşımlarında, bu yöntemler arasından en fazla matematiksel programlama temelli yaklaşımların ve metasezgisellerin kullanıldığı görülmektedir.

2.3.1 Matematiksel programlama temelli yaklaşımlar

Matematiksel programlama temelli yaklaşımlar, literatürde en yaygın kullanılan yöntemlerden bir tanesidir. Bu yaklaşımda matematiksel programlamanın uzantıları olan, doğrusal programlama, doğrusal olmayan programlama, tam sayılı programlama, hedef programlama, 0-1 tam sayılı programla gibi yöntemler kullanılır.

Literatüre bakıldığında matematiksel programlama temelli yaklaşımların 1970'lerle birlikte ilk önemli çalışmalarının yapıldığı görülmektedir. Akkoyunlu (1973) çalışmasında derslik boyutunu ortadan kaldırmış ve üniversitedeki sadece bir bölüm için doğrusal programlama modeli geliştirmiştir. Farklı modellerin kurulmasının ardından, Ferland ve Roy (1985) yaptıkları çalışmada problemi iki aşamaya bölerek incelemişlerdir. Dersleri ilk önce zaman periyotlarına ve sonrasında sınıflara atayacak şekilde ayırarak, 0-1 tam sayılı bir matematiksel model geliştirmişlerdir.

İlk öncü çalışmalardan bu yana matematiksel programlama temelli çözüm yaklaşımları üzerine çalışmalar devam etmekte ve her geçen yıl yeni özellikler eklenerek süreç geliştirilmektedir. İsmayilova vd. (2007) yaptıkları çalışmada yönetimin ve gözetmenlerin tercihlerini göz önünde bulundurarak ve AHP(Analitik Hiyerarşi Prosesi) ve AAS (Analitik Ağ Süreci)yöntemlerini de kullanıp çok amaçlı 0-1 tam sayılı bir model geliştirmişlerdir.

Yakın zamanda matematiksel programlama çözüm tekniği ile yapılan çalışmalara örnek olarak Van Den Broek ve Hurkens (2012) gösterilebilir. Bu çalışmayla, üniversite ders çizelgeleme problemi için doğrusal tam sayılı bir matematiksel model geliştirilmiş ve devamında sezgisel yönteme başvurularak problemin çözümü araştırılmıştır.

Problemdeki kısıtlar ve değişkenler fazlalaştıkça, kurulan matematiksel modeller de daha karmaşık bir hale gelmekte ve çözüme ulaşılması zorlaşmaktadır. Zaman çizelgeleme problemlerinin karmaşık bir yapısı olması sebebiyle de, çoğu kez bir sonuca ulaşılamamaktadır. Bu sebeple araştırmacılar her zaman alternatif çözüm yolları arama yönelimi sürdürmektedirler.

2.3.2 Grafik renklendirme temelli yaklaşımlar

Ağ renklendirme temelli yaklaşımlar literatürde yer alma tarihi eskilere dayanan problem çözme yaklaşımlarındandır. Bu yaklaşımda dersler ve sınavlar düğümler ile zaman periyotları da renkler ile gösterilmektedir. Eğer iki ders ya da sınav aynı öğrenci

grubu tarafından alınıyorsa ya da bu sınava ya da derse aynı öğreticinin katılması gerekiyorsa ve de buna benzer aralarında bağlayıcı kısıtlar varsa dersi/sınavı temsil eden düğümler birbirlerine bir çizgiyle bağlanır. Böylelikle zaman ataması için renklendirme yapılırken bu iki düğüm hiçbir zaman aynı renkte olmaması gerektiği gözetilir.

Dandashi ve Al-Mouhamed (2010) yaptıkları çalışmada sınıf çizelgeleme problemini ağ renklendirme çözüm yaklaşımıyla ele almışlardır.

Üzerinde kolaylıkla değişiklik yapılamaması ve büyük problemler için çok kullanışlı ve etkili olmaması bu yöntemin negatif özellikleri arasında gösterilebilir.

2.3.3 Kısıt programlama temelli yaklaşımlar

Kısıt programlama temelli yaklaşımlar, problemde yer alan tüm kısıtların sağlanmasıyla bir zaman çizelgesi oluşturan yöntemlerdir. Eğitimsel zaman çizelgeleme problemlerinin çok sayıda kısıt içeren karmaşık yapıları göz önünde bulundurulduğunda, çözüm yaklaşımlarında bu yöntemin başvurulan yöntemlerden biri olması muhtemeldir.

Literatüre bakıldığında Zang ve Lau (2005) üniversite zaman çizelgeleme probleminin çözümünü kısıt programlama yaklaşımına göre ele almışlardır. Zang ve Lau (2005) ve kısıt programlama yöntemi üzerinde çalışan çoğu araştırmacılar, problem çözümlerinde, ILOG çözücüsü ve ILOG çizelgeleme araçlarını kullanmışlardır.

Yapılan çalışmalarla, kısıt programlama temelli yaklaşımların, tek başlarına kullanıldıklarında problem çözümünde yetersiz kaldıkları ve başka bir yöntemle desteklenip melez yapıda kullanılmaları gerektiği görülmüştür.

2.3.4 Çok amaçlı modelleme yöntemleri

Eğitimsel zaman çizelgeleme problemleri sadece bir amaç fonksiyonundan oluşabileceği gibi, birden fazla amaç fonksiyonun gerçekleştirilmesi de istenebilir. Lee ve Sehniederjans (1983) yaptıkları çalışmayla, toplam maliyetin en küçüklenmesi ve öğreticilerin, okulların ve de yönetimin tercihlerinin en iyilenmesi amaçlarını belirleyerek, 22 okula öğretmenlerin atanması problemini ele almış ve belirlenen amaçların dengelemesine yönelik bir hedef programlama modeli geliştirmişlerdir.

Bu tarz problemlerin ele alındığı çoğu çalışmada hedef programlama tekniği kullanılmaktadır. Hedef programlama yönteminde amaçlardan sapmanın en küçüklenmesi istenmektedir. Öztürk (2010) çalışmasında çok amaçlı yaklaşımlarda,

sağlanmayan esnek kısıtların her birinin en küçüklenmesiyle amaç fonksiyonlarının oluşturulabileceğini ifade etmiştir.

2.3.5 Sezgiseller

Sezgisel algoritmalar, eğitimsel zaman çizelgeleme problemlerinin çözümünde çok kullanılan problem çözme tekniklerinden bir tanesidir. Sezgisel algoritmalar sonucun doğruluğunun doğru olup olmadığı ile ilgilenmezler ve en iyi çözümü de her zaman garanti etmeyebilirler ancak iyiye yakın çözüm yollarına ulaşırlar. Genellikle büyük problem boyutlarında eğer matematiksel yöntemle çözüm bulunamıyorsa sezgisel algoritma geliştirme yoluna başvurulur.

Literatüre bakıldığında da sezgisel algoritmaların üniversite, ders, sınav atamalarının çözüm yöntemleri araştırmaları sürecinde ilk kullanılan yöntemlerden biri olduğu görülmektedir. Burke ve Newall (2002) yaptıkları çalışmada, sınav çizelgeleme problemlerine sezgisel algoritma yöntemi ile yaklaşmışlardır. Kahara ve Kendalla (2010) yaptıkları çalışmayla, gerçek hayat problemi olan bir üniversitenin sınav çizelgeleme problemi için bir sezgisel algoritma geliştirerek problemin çözümünü araştırmışlardır.

2.3.6 Metasezgiseller

Metasezgisel yöntem, arama uzayında farklı kavramları akıllı bir şekilde birleştirerek ve de alt seviye sezgisellere rehberlik ederek oldukça kaliteli sonuçlar bulabilen iteratif bir süreçtir.

Gerçek hayatta karşılaşılan eğitimsel zaman çizelgeleme problemlerinin kısıt ve değişken sayılarının çok fazla olması ve bu yüzden karmaşık yapıya sahip olmaları, araştırmacıları problemlerin kesin çözümlerini bulmak için metasezgisel yöntemlere doğru yönlendirmiştir. Literatürde, özellikle 1990'lı yıllardan itibaren metasezgisel yöntemlere başvurarak çok fazla çalışma yapıldığı görülmektedir.

Metasezgisel yöntemler tek çözümlü ve popülasyon tabanlı algoritmalar olarak sınıflandırılmakta ve aşağıdaki bileşenlerden oluşmaktadır:

Tek çözümlü:

- Tavlama benzetimi
- Tabu arama
- Değişken komşuluk arama

- Yerel arama

Popülasyon tabanlı algoritmalar:

- Genetik algoritma
- Karınca kolonisi optimizasyonu (KKO)
- Parçacık sürü optimizasyonu
- Memetik algoritma
- Harmoni arama

Tavlama Benzetimi

Tavlama benzetimi, uygulama alanı oldukça geniş olan ve optimizasyon problemleri için kullanılan bir çözüm tekniğidir. Adını gerçek hayatta bir katının yavaş yavaş soğutulması olarak adlandırdığımız fiziksel bir işlem olan tavlama işleminden almaktadır.

Literatüre bakıldığında 1990'lı yıllardan itibaren bu alanda daha fazla çalışmanın yapıldığı görülmektedir. Kirkpatrick vd. (1983) yılında bir optimizasyon problemi için, Cherny (1985) ise bir gezgin satıcı problemi için bu yöntemi kullanarak, bu yöntemin kullanım alanının gelişmesine katkı sağlamışlardır.

Bu yöntemi kullanarak son zamanlarda yapılan çalışmalara örnek olarak Basir vd. (2013) gösterilebilir. Basir vd. (2013) yaptıkları çalışmayla, gerçek hayat eğitimsel zaman çizelgeleme problemi olan, bir üniversitenin bir programına ait çizelgeleme problemi çözümü için tavlama benzetimi yaklaşımını kullanmışlardır.

Tabu Arama

Tabu arama, kombinatoryal optimizasyon problemlerinin çözümünde kullanılan bir metasezgisel yöntemdir. Çoğunlukla başka yöntemlerle birlikte kullanılır ve bunun da amacı birlikte kullanıldığı yöntemin en iyiye takılmasını engellemektir. Bu yöntem başlangıç çözümü, mevcut çözümler arasından seçebilir ve her adımda iteratif olarak ilerleyerek yeni bir çözüm arar.

Tabu arama yöntemi konusunda ilk çalışmalar Glover (1989) ve Glover (1990)' a aittir. Literatürde eğitimsel zaman çizelgeleme problemlerinin çözümünde de tabu araştırması yönteminin kullanıldığı görülmektedir. Aladağ vd. (2009)'da yaptıkları

çalışmada, bir üniversitenin bir bölümü için ders çizelgeleme problemini, tabu arama yöntemini kullanarak incelemişlerdir.

Değişken Komşuluk Arama

Literatürde eğitimsel zaman çizelgeleme problemlerinin çözümü için değişken komşuluk arama yöntemiyle ilgili çok fazla çalışma yer almadığı görülmektedir.

Abdullah vd. (2005) yaptıkları çalışmada, bir üniversitenin ders çizelgeleme probleminin çözümü için değişken komşuluk arama yöntemini kullandıkları görülmüştür.

Yerel Arama

Yerel arama algoritması, çözüm uzayı adayları arasında optimum çözümü bulana kadar yerel değişiklikleri de uygulayarak bir adaydan diğerine geçmesi olarak tanımlanabilir.

Tepe tırmanma (Hill Climbing) algoritması yerel arama yöntemlerinden birisidir. Algoritma rassal bir başlangıç çözümden başlayarak komşu noktalara gider ve her noktadaki amaç fonksiyonunu hesaplar. Komşuluktaki değer daha iyiyse değer güncellenir, daha iyi değilse de çözüm değiştirilmez.

Eğitimsel zaman çizelgeleme alanında yerel arama algoritmasının kullanımı için Schaerf ve Gaspero (2001)'de yaptıkları çalışma örnek gösterilebilir. Bu çalışmayla, sınav çizelgeleme problemini yerel arama yöntemi ve de aynı zamanda başka yöntemler kullanarak ayrı ayrı çalışıp sonuçları incelemişlerdir. Jat ve Yang (2011) yaptıkları çalışmada bir üniversite zaman çizelgeleme problemi çözümü için yerel arama tekniklerini kullanarak genetik algoritma araştırması gerçekleştirmişlerdir.

Genetik Algoritma

Genetik algoritma mantığının temeli, Holland (1975) tarafından yapılan çalışmalarla birlikte atılmıştır. Holland (1975) yaptığı çalışmada, doğadan ve evrimden etkilenmiş, bunun sonucunda da sadece bir yapının öğrenme yeteneğinin geliştirilmesi yerine, çiftleşmek üremek vb. gibi evrimsel süreçleri mekanik yapılara da uygulayıp daha başarılı nesiller elde edilebileceğini çalışmasında göstermiştir. Böylelikle ortaya çıkan yöntem de genetik algoritma adını almıştır.

Genetik algoritmalar, eğitimsel zaman çizelgeleme problemlerinin çözümünde en çok kullanılan metasezgisel yaklaşımlardan biridir. Literatür incelendiğinde, eğitimsel zaman çizelgeleme alanında genetik algoritma yaklaşımlarıyla çok fazla sayıda çalışmanın olduğu ve iyi sonuçlar alındığı görülmektedir. Bu çalışmalara örnek olarak Yu ve Sung (2002) yaptıkları çalışmada bir üniversitede haftalık ders programının hazırlanması için, bir genetik algoritma geliştirerek çözüm aramışlar ve sonuçların başarılı olduğunu görmüşlerdir. Alsmadi vd. (2011) yaptıkları çalışmada bir üniversite ders çizelgeleme probleminin çözümü için genetik algoritma geliştirmişlerdir. Çalışmanın sonucunu diğer yöntemlerin sonuçlarıyla da karşılaştırıp, daha az kullanılmayan sınıf olduğunu ve gözetmenlerin aşırı görevlendirme yükünün de azaldığını tespit etmişlerdir.

Karınca Kolonisi Optimizasyonu

Karınca kolonisi optimizasyonu da doğadan ve canlılardan esinlenerek geliştirilmiş bir metasezgisel yöntemdir. Karıncalar besin arayışı için toprak üzerinde ilerlerken, feromon adlı bir kimyasal bırakırlar. Sonrasında da karınca toplulukları gidecekleri yolları bu maddeye göre belirlerler. Bir rota üzerinden ne kadar çok sayıda karınca geçerse, o bölge üzerindeki feromon kimyasalının miktarı da artar. Böylelikle bir rotadan geçen karınca sayısı, o yolun seçilme ihtimalini doğrudan artırmaktadır.

Literatürde eğitimsel zaman çizelgeleme problemlerinin çözümü için karınca kolonisi optimizasyonu ile yapılan çalışmalarda verimli sonuçların elde edildiği görülmüştür. Mayer vd. (2008) yaptıkları çalışmada ders kayıtları tamamlanan bir üniversite için, dersleri zaman periyotlarına ve dersliklere atayacak ders atama problemi çözümü için karınca kolonisi tabanlı bir sezgisel algoritma geliştirmişlerdir.

Parçacık Sürü Optimizasyonu

Parçacık sürü optimizasyonu, doğa ve canlılardan esinlenilerek geliştirilmiş diğer bir metasezgisel yöntemdir. Bu yöntem Kenedy ve Eberhart (1995) tarafından sürü halindeki böcek ve balıklar gözlemlenerek geliştirilmiştir.

Eğitimsel zaman çizelgeleme problemleri alanında da literatürde çalışmalar yer almaktadır. Shiau (2011), yaptığı çalışmada bir üniversite ders çizelgeleme probleminin çözümü için parçacık sürü optimizasyonu yöntemini kullanmıştır.

Memetik Algoritma

Memetik algoritmalar, genetik algoritma ve yerel arama tekniklerinin bir arada kullanıldığı melez bir metasezgisel yöntemdir.

Literatürde, eğitimsel zaman çizelgeleme problemlerinin çözümünde bu yöntemin kullanıldığı çalışmalar görülmektedir. Alkan ve Özcan (2003) bir üniversite zaman çizelgeleme probleminin çözümünü hedefleyerek yaptıkları çalışmada memetik algoritma yöntemini kullanmışlardır. Joudaki vd. (2010) yaptıkları çalışma da üniversite ders çizelgeleme problemi çözümü için memetik algoritması kullanılarak çözüm aranan çalışmalardan bir tanesidir.

Harmoni Arama

Harmoni arama yöntemi, orkestradaki müzisyenlerin nota çalarak en iyi melodiyi elde etmek için oluşturdukları armoniyi taklit eden, canlılardan esinlenmiş bir diğer metasezgisel yöntemdir. Greem vd. (2001) tarafından temelleri atılmıştır.

Eğitimsel zaman çizelgeleme problemlerinde bu yöntemin kullanımı için Wahid ve Hussin (2013) ortaya koydukları çalışma, üniversite ders zaman çizelgeleme problemi çözüm yaklaşımına örnek olarak gösterilebilir.

2.3.7 Yeni Yöntemler

Literatürün incelenmesiyle birlikte görüldüğü üzere, eğitimsel zaman çizelgeleme problemi çözümleri için çok sayıda ve birbirinden farklı metotlar geliştirilerek çalışmalar yapılmış ve günümüzde de bu çalışmalara devam edilmektedir.

Bir yöntemin çözümün herhangi bir açıdan yetersiz kaldığının görülmesi, bu konuda yapılan çalışanları hep yeni yollar keşfetmek için ileriye taşımıştır. Bazen de mevcut yöntemler birlikte kullanılarak çözümler denenmiş ve sonuçlar karşılaştırılmıştır.

Yukarıda bahsedilen çözüm yöntemlerine ek olarak yeni yöntemler üzerinde çalışılmaya devam edilmektedir. Bu yöntemlerin arasında melez algoritmalar, bulanık yöntemler, yapay sinir ağları, karar destek sistemleri, kümeleme algoritmaları yer almaktadır.

Melez algoritmalar yöntemiyle yapılan çalışmaya örnek olarak Asham vd. (2011) yaptıkları çalışma örnek gösterilebilir. Bu çalışma ile bir ders çizelgeleme problemi için

grafik renklendirme ve genetik algoritma yöntemleri birlikte kullanılarak melez algoritma yöntemine başvurulup, çözüm araştırılmıştır.

Chaudhur ve Kajal (2010) tarafından gerçekleştirilen çalışma, üniversite ders çizelgeleme problemleri için bulanık yöntemin kullanıldığı bir çalışmaya örnek teşkil etmektedir. Bulanık yöntemler oldukça yeni bir yaklaşım olduğundan literatürde çok fazla sayıda bu yöntem kullanılarak çalışılmış örnek yer almamaktadır.

Carrasco ve Pato (2004) ortaya koydukları çalışma, eğitimsel zaman çizelgeleme problemlerinin çözümünde yapay sinir ağları yöntemi ile çözüm arayan ait örnek bir çalışmadır. Dasgupta ve Khazanchi (2005)'nin çalışmaları ders çizelgeme problemi için bir karar destek sistemi yöntemi yaklaşımı örneği iken, Shatnawi vd. (2010) yaptıkları çalışma da yine eğitimsel zaman çizelgeleme problemleri için kümeleme algoritmaları yaklaşımına örnek gösterilebilecek bir çalışmadır.

3. ÇOK OTURUMLU SINAVLARDA KİTAPÇIK OPTİMİZASYONU

Türkiye'de çok oturumlu sınavlar birçok kurum tarafından uygulanmaktadır. Yükseköğretime geçiş sınavları, kamu personelinin yerleştirilmesi için yapılan sınavlar, açık ve uzaktan eğitim veren kurumların gerçekleştirdiği sınavlar birden fazla oturumda gerçekleştirilen, çok oturumlu sınavlara örnek gösterilebilecek sınavlardır.

Bu tez kapsamında da çok oturumlu sınavlardan olan Anadolu Üniversitesi Açıköğretim Fakültesi programlarına ait sınavlar ele alınmıştır.

3.1 Anadolu Üniversitesi Açıköğretim Fakültesi

Bir ülkenin eğitim seviyesi, o ülkenin gelişmişlik düzeyini gösteren en önemli unsurlardan bir tanesidir. Bu nedenle her yaştan, her eğitim seviyesinden ve de toplumun her kesiminden insanların istedikleri zaman ve istedikleri şekilde kolayca eğitime ulaşabilmeleri, bir ülkenin eğitim seviyesini artıran önemli bir özelliktir. Eğitime bu bahsettiğimiz şekillerde ulaşmak ise açık ve uzaktan eğitim sistemiyle sağlanmaktadır. Açık ve uzaktan eğitim sistemi öğrenciyi, öğreticiyi ve öğrenme ortamını bir araya getiren çağdaş teknolojik bir sistemdir.

Anadolu Üniversitesi Açıköğretim Fakültesi, 20 Temmuz 1982'de çıkartılan 41 sayılı Kanun Hükmünde Kararname ile açık ve uzaktan eğitim alanında Türkiye'ye hizmet vermek için açılmıştır ve Türkiye'de bu alandaki ilk fakültedir. Yıllar içinde Açıköğretim Fakültesi kendini sürekli geliştirmiş, hızla büyümüş ve bu iyileşme de kısa sürede büyük öğrenci kitlesine ulaşmasına sebep olmuştur. Şu anda 2 milyona yakın öğrenci sayısıyla dünyanın önde gelen Açıköğretim fakültelerinden bir tanesi haline gelmiştir. Ülke genelinde büyük hizmetler veren fakülte, son yıllarda uluslararası alanlara da genişleyerek dünyanın birçok yerine eğitim hizmeti sunar hale gelmiştir.

1993 yılında ise Açıköğretim Sistemi, 496 sayılı Kanun Hükmünde Kararname ile güncel eğitim gereksinimlerine göre yeniden yapılandırılmış ve İktisat ve İşletme programları da dört yıllık İşletme ve İktisat Fakültelerine dönüştürülmüştür.

3.2 Anadolu Üniversitesi Açıköğretim Sistemi Sınavları

Anadolu Üniversitesi Açıköğretim sistemi aşağıdaki fakültelerden oluşmaktadır:

- Açıköğretim fakültesi-lisans,
- Açıköğretim fakültesi-ön lisans,
- İktisat fakültesi,
- İşletme fakültesi

Her fakültenin altında da adayların alacakları eğitimi seçip kaydolabilecekleri çeşitli bölümler yer almaktadır.

Açıköğretim sisteminin altında Türkiye programları dışında, yeni açılan programlarla birlikte çok sayıda yurt dışı programı bulunmaktadır, yurt dışı programları aşağıdaki gibidir:

- Kuzey Amerika Programları,
- Suudi Arabistan Programları,
- Batı Avrupa Programları,
- Azerbaycan Programları,
- Bulgaristan Programları,
- Kosova Programları,
- Makedonya Programları,
- Arnavutluk Programları,
- Bosna Hersek Programları

Her bir yurt dışı programı da Türkiye programlarında olduğu gibi, Açıköğretim Fakültesi lisans, Açıköğretim Fakültesi ön lisans, iktisat fakültesi ve işletme fakültesine ait bölümleri bünyesinde barındırmakta ve öğrenciler eğitim almak istedikleri alanda tercihlerini gerçekleştirmektedir.

Her yıl ilgili eğitim dönemi içerisinde Türkiye ve yurt dışı programları için sınavlar gerçekleştirilmektedir. Bu sınavların organizasyonu Bilgisayar Araştırma ve Uygulama Merkezi, Test Araştırma Birimi ve Basımevinin işbirliği ile yapılmaktadır.

Sınavlar güz ve bahar eğitim dönemi olmak üzere iki dönemde gerçekleşmekte ve her dönem için bir ara ve bir dönem sonu sınavı olmak üzere her dönemde iki sınav yapılmaktadır. Ayrıca bahar dönem sonu sınavları da tamamlandıktan sonra, mezun

durumunda olan ve üç dersten kalmış öğrenciler için 2017 yılından itibaren üç ders sınavı yapılmaktadır. Bu sınavın değerlendirilmesi sonucunda öğrencilerin mezuniyet durumları değişebilmektedir. Böylelikle her Açıköğretim Fakültesi programı için, bir eğitim öğretim yılında toplamda 5 sınav gerçekleştirilmektedir.

Sınav organizasyonu, belli adımlardan oluşan büyük ve özverili bir süreçtir. Öğrencilerin sınav binalarına atanması, görevlilerin sınav bina ve salonlarına atanması, sınav kitaplarının ve sınav evraklarının hazırlanması, kontrolü ve basılması, sınav sonuçların değerlendirilmesi ve açıklaması vb. gibi farklı birçok adımdan meydana gelmektedir. Bu adımlardan bir tanesi de oturum düzeninin hazırlanmasıdır ve bir nevi sınav organizasyonunu başlatan bir adımdır. Oturum düzeni veri tabanına eklenip kontroller gerçekleştirildikten sonra, öğrencilerin ders ve bina atamaları da tamamlanarak kalan tüm sınav organizasyonu adımları sürece uygun şekilde tamamlanmaktadır.

3.3 Oturum Düzeninin Hazırlanması

Oturum düzeninin hazırlanması, genel anlamıyla ifade edilecek olunursa, belirli kısıtlar altında bir kitabın içinde hangi derslerin yer alacağına ve o kitabın hangi oturumda olacağına karar veren çalışmalar bütünüdür.

Türkiye ve yurt dışı programları için sınavların oturum düzenleri Test Araştırma birimi tarafından hazırlanmaktadır. Sonrasında hazırlanan oturum düzeni Bilgisayar Araştırma ve Uygulama Merkezindeki sınav organizasyonu birimine gönderilir. Hazırlanan oturum düzeni sınav organizasyonu birimi tarafından veri tabanına eklenir ve böylelikle sınav organizasyonu süreci başlatılır. Bazı yeni açılan programlar ya da yeni sınav türleri için oturum düzenleri, Bilgisayar Araştırma ve Uygulama Merkezinde sınav organizasyonu birimi tarafından da hazırlanabilmektedir (Açıköğretim Programları 3 ders sınavı).

Oturum düzeni belirli kısıtlar göz önünde bulundurularak hazırlanır. Bu kısıtlar kesinlikle gerçekleştirilmesi gereken ve literatürde katı kısıt olarak tabir edilen kısıtlardır. Literatürde bahsedilen bu tarz problemlerdeki diğer kısıt türü ise esnek kısıtlardır. Bu kısıtlar ise gerçekleştirilmesi zorunlu olmayan ancak gerçekleştirildiği takdirde modele fayda sağlayan kısıtlardır. Açıköğretim sistemi için hazırlanan oturum düzenlerinde katı kısıtlar mevcuttur, esnek kısıtlar mevcut sistemde yer almamaktadır. Sistemin kısıtları aşağıdaki gibidir:

- Her ders sadece bir oturuma atanabilir ve tüm derslerin mutlaka bir oturuma atanması gerekmektedir,
- Oluşturulan bir kitap türü sadece bir oturuma atanabilir,
- Bir ders sadece bir oturumda yer alması koşuluyla, birden fazla kitap türünün içinde yer alabilir,
- Bir öğrenci bir oturumda en fazla beş dersten sınava girebilir,
- Bir öğrenci bir oturumda sadece bir kitap alabilir,
- Güz dönemlerinde öğrencilere 1., 3., 5., ve 7. yarıyıl dersleri, bahar dönemlerinde ise 2., 4., 6., ve 8. yarıyıl dersleri atanmalıdır,
- Bir kitap türü en fazla beş ders içerebilir. (Bu kural Açıköğretim sistemi Türkiye programları sınavları için geçerli bir kuraldır, yurt dışı programlarında bir kitapta daha fazla sayıda ders yer alabilmektedir ancak bu durumda da öğrencinin bir oturumda en fazla beş dersten sınava girmesi kuralı devam etmektedir.)

Türkiye ve yurt dışı programlarının altına yeni bölümler açıldığı zaman ise, Test Araştırma Birimi, Açıköğretim Fakültesi Dekanlığının gönderdiği yeni bölüme ait ders bilgilerini yukarıdaki kısıtlar ve işleyişi göz önünde bulundurarak sınav oturum düzeninin içine yerleştirir ve mevcut oturum düzeninde gerekli güncellemeleri yerine getirir.

2017-2018 Öğretim yılı bahar dönemi ara sınavı lojistik ve iktisat programlarına ait örnek oturum düzenleri Şekil 3.1 ve Şekil 3.2'de gösterilmektedir.

LOJİSTİK PROGRAMI (79)

BAHAR DÖNEMİ

2. YARIYIL

1.OTURUM (1279) CT 09.30

DERS KODU	DERS ADI	SORU SAYISI
LOJ102U	Çağdaş Lojistik Uygulamaları	20
LOJ104U	Lojistik Yönetimi	20
BİL102U	Temel Bilgi Teknolojileri-II	20

2.OTURUM (2279) PZ 09.30

DERS KODU	DERS ADI	SORU SAYISI
İKT102U	Proje Yönetimi	20
MAT106U	Matematik-II	20
SOS114U	Davranış Bilimleri-II	20
İŞL106U	İşletme Fonksiyonları	20
İNG102U	İngilizce-II	20
ALM102U	Almanca-II	20
FRA102U	Fransızca-II	20

4. YARIYIL

3.OTURUM (3407) CT 14.00

DERS KODU	DERS ADI	SORU SAYISI
PZL208U	Müşteri İlişkileri Yönetimi	20
HUK208U	Ticaret Hukuku	20
TÜR202U	Türk Dili-II	20

4.OTURUM (4479) PZ 14.00

DERS KODU	DERS ADI	SORU SAYISI
LOJ202U	Lojistikte Teknoloji Kullanımı	20
LOJ204U	Lojistik Maliyetleri ve Raporlama-II	20
İST206U	Yöneylem Araştırması-II	20
PZL212U	Tedarik Zinciri Yönetimi	20
TAR202U	Atatürk İlkeleri ve İnkılap Tarihi-II	20

Şekil 3.1 *Lojistik Programı için Örnek Oturum Düzeni*

İKTİSAT PROGRAMI (11)

BAHAR DÖNEMİ

2. YARIYIL

1.OTURUM (1200) CT 09.30

DERS KODU	DERS ADI	SORU SAYISI
MUH104U	Genel Muhasebe-II	20
İKT104U	İktisada Giriş-II	20
HUK110U	Borçlar Hukuku	20
BİL102U	Temel Bilgi Teknolojileri-II	20

2.OTURUM (2211) PZ 09.30

DERS KODU	DERS ADI	SORU SAYISI
İKT102U	Proje Yönetimi	20
İŞL104U	Yönetim ve Organizasyon	20
MAT106U	Matematik-II	20
İNG302U	İngilizce-II	20
ALM302U	Almanca-II	20
FRA302U	Fransızca-II	20

4. YARIYIL

3.OTURUM (3400) CT 14.00

DERS KODU	DERS ADI	SORU SAYISI
HUK208U	Ticaret Hukuku	20
İST202U	İstatistik-II	20
TÜR202U	Türk Dili-II	20

4.OTURUM (4411) PZ 14.00

DERS KODU	DERS ADI	SORU SAYISI
İKT202U	Doğal Kaynaklar ve Çevre Ekonomisi	20
İKT204U	Makro İktisat	20
İKT206U	Uluslararası Ekonomik Kuruluşlar	20
HUK210U	İdare Hukuku	20
TAR202U	Atatürk İlkeleri ve İnkılap Tarihi-II	20

6. YARIYIL

1.OTURUM (1611) CT 09.30

DERS KODU	DERS ADI	SORU SAYISI
İKT104U	İktisada Giriş-II	20
İKT302U	Bilgi Ekonomisi	20
İKT308U	Uluslararası İktisat Politikası	20
MUH104U	Genel Muhasebe-II	20
HUK110U	Borçlar Hukuku	20

2.OTURUM (2611) PZ 09.30

DERS KODU	DERS ADI	SORU SAYISI
İKT304U	Hizmetler Ekonomisi	20
İKT306U	Para Politikası	20
HUK108U	İş ve Sosyal Güvenlik Hukuku	20
ÇEK308U	Çalışma Ekonomisi	20
İNG302U	İngilizce-II	20
ALM302U	Almanca-II	20
FRA302U	Fransızca-II	20

8. YARIYIL

3.OTURUM (3811) CT 14.00

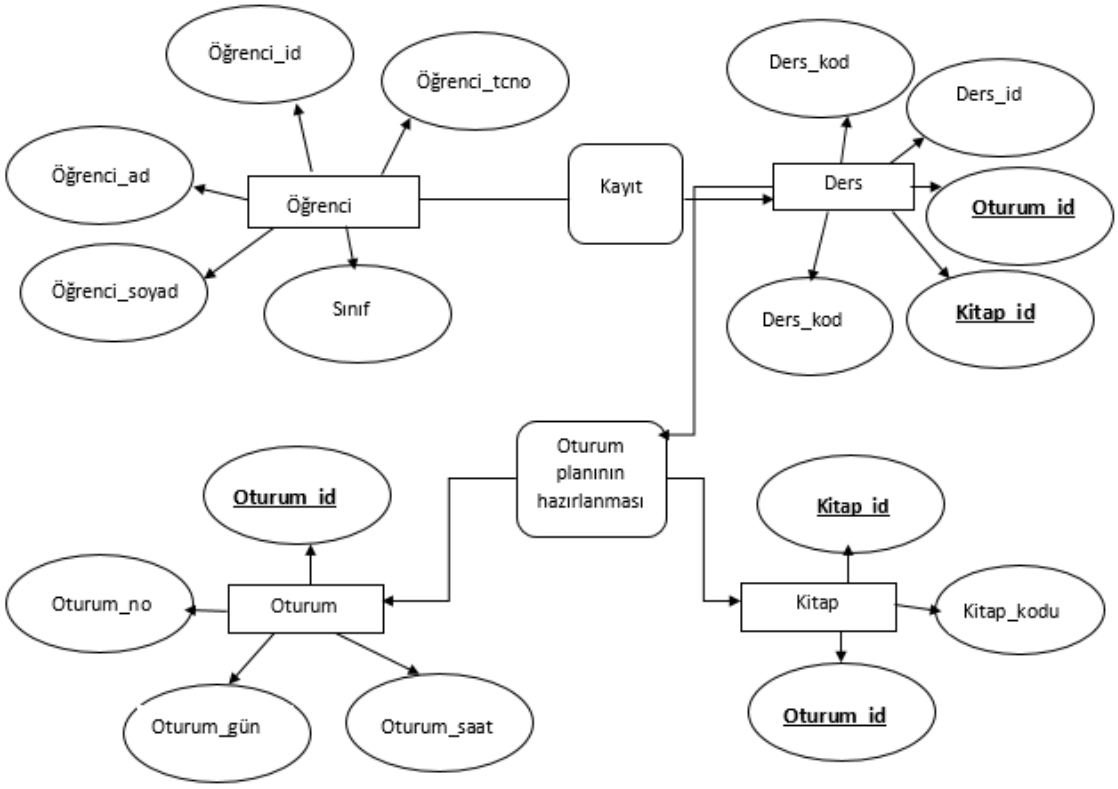
DERS KODU	DERS ADI	SORU SAYISI
İKT402U	Türkiye Ekonomisi	20
İLT184U	Etkili İletişim Teknikleri	20
İKT408U	Ekonominin Güncel Sorunları	20
TÜR202U	Türk Dili-II	20

4.OTURUM (4811) PZ 14.00

DERS KODU	DERS ADI	SORU SAYISI
İKT406U	Avrupa Birliği ve Türkiye İlişkileri	20
TAR202U	Atatürk İlkeleri ve İnkılap Tarihi-II	20
İKT412U	İktisat Tarihi	20
MLY402U	Maliye Politikası	20
İKT410U	İktisadi Büyüme	16

Şekil 3.2 *İktisat Programı için Örnek Oturum Düzeni*

Mevcut sisteme ait, oturum düzeni hazırlandıktan ve veritabanına girildikten sonra sistemin varlıklarını, bu varlıkların özelliklerini ve de sistemdeki varlıkların birbirleriyle olan ilişkilerini gösteren örnek bir varlık ilişki diyagramı (ER diyagramı) da Şekil 3.3'te gösterilmektedir:



Şekil 3.3 *Oturum Düzeninin İçeriye İşlenmesiyle Elde Edilen Sistemin Varlık İlişki (ER) Diyagramı*

3.4 Problemin Tanımı

Mevcut sistemde oturum düzeninin hazırlanması, katı kısıtların gerçekleştirilmesini sağlayarak manuel olarak gerçekleştirilen bir süreçtir. Bu işlemler Açıköğretim Fakültesi sistemi Türkiye programları sınavları için uzun yıllardır süregelen işlemler olması sebebiyle, oldukça sistematik ve oturmuş bir yapıya sahiptir. Ancak yeni açılan yurt dışı programları veya mevcut sisteme yeni sınavların (üç ders sınavı) eklenmesi durumunda, bir anlamda sistemin ihtiyaçlarının güncellendiği durumlarda, manuel çözümler yetersiz kalabilmektedir. Çünkü bu tarz durumlar, sistemin bazı katı kısıtlarının değişmesine neden olmaktadır.

Örneğin Açıköğretim sistemi Türkiye programları dönem sınavlarında, bir öğrenci bir oturumda en fazla beş ders alabilir kısıtı, üç ders sınavları için, bir öğrenci bir oturumda en fazla üç ders alabilir şekline dönüşmektedir.

Bu çalışmanın ele alınmasındaki sebeplerden bir tanesi yeni eklenebilecek sınav türlerindeki veya yeni açılan programların sınavları için ortaya çıkabilecek, mevcutta gerek duyulmayan ihtiyaçlar doğrultusunda oturum düzenini kolayca oluşturabilecek ve manuel işlemlere en az seviyede yer verecek yeni bir sistem oluşturmaktır. Başka bir neden ise mevcut kemik yapının bu sistemle vereceği sonuçları inceleyerek, mevcut yöntemin değiştirilip değiştirilemeyeceğine karar vermektir.

4. ÇOK OTURUMLU SINAVLARDA KİTAPÇIK OPTİMİZASYONU İÇİN GELİŞTİRİLEN MATEMATİKSEL MODEL

4.1 Matematiksel Model

Oturum düzeni hazırlama sürecinin modellenmesi için öncelikle sürecin baştan sona tüm adımları incelenmiştir. Bu incelemeyle birlikte, sistemin hangi bileşenlerden oluştuğu, sistemin eksiksiz şekilde istenen sonucu vermesi için hangi kısıtların sağlanması gerektiği, parametrelerin neler olacağı ve de sistemin hangi amaç veya amaçlar doğrultusunda modellenmesi gerektiğine karar verilmiştir.

Mevcut sistemde Açıköğretim sistemi sınavları için hazırlanan oturum düzenlerinde bölüm bilgisi de göz önünde bulundurulur. Her bölümün kendi içinde kitap-oturum, ders-oturum, ders-kitap yerleştirmesi yapılır. Türkiye ve yurt dışı programlarının altında çok fazla sayıda bölüm olması, bazı derslerin ortak dersler olarak her bölümün altında yer alması gerekliliği ve de bölüm derslerinin kendi bölümleri içinde kitaplara yerleştirilmesi gerektiğinden süreç adımları fazlalaşmakta, oturum bazında kitaplar, dersler ve öğrenciler dengeli bir dağılıma sahip olamamaktadır.

Tez kapsamında, sistemin matematiksel modeli kurulurken, bölüm bilgisi dikkate alınmadan alternatif bir model geliştirilmiştir. Girdi olarak sadece öğrenciler ve üzerlerindeki dersler alınmış ve bölüm bilgisi dışarıda bırakılmıştır. Açıköğretim sistemi Türkiye programına ait sınavlarda öğrenci sayısı milyondan fazla olduğu için, yurt dışı programı olan ve yeni açılan bir program olması sebebiyle görece az öğrenciye sahip olan Kuzey Amerika Programına ait 2017-2018 eğitim dönemi güz dönemi dönem sonu sınavı örneklem sınav olarak seçilmiştir.

Sistemin analizi yapıldıktan sonra, sistemin dört bileşenden oluştuğuna karar verilmiştir. Bunlar öğrenciler, dersler, sınav kitapları ve sınav oturumlarıdır. Bölüm bilgisi sistem bileşenlerinden birisi değildir. Öneri sistemimizin kısıtları ise aşağıdaki gibidir:

- Her ders sadece bir oturuma atanabilir ve tüm derslerin mutlaka bir oturuma atanarak yerleştirilmesi gerekmektedir,
- Oluşturulan bir kitap türü sadece bir oturuma atanabilir,

- Bir ders sadece bir oturumda yer alması koşuluyla, birden fazla kitap türünün içinde yer alabilir,
- Bir öğrenci bir oturumda en fazla beş dersten sınava girebilir,
- Bir öğrenci bir oturumda sadece bir kitap alabilir,
- Bir kitap türü en fazla 10 ders içerebilir.

Modelin amaç fonksiyonu ise, öğrencilerin üzerlerindeki tüm derslerin yukarıdaki kısıtlar göz önünde bulundurularak dağıtıldığında oluşacak toplam kitap türünün en küçüklenmesidir.

Çok oturumlu sınavlarda kitapçık optimizasyonu için kurulan matematiksel model izleyen adımlarda verilmiştir:

İndis Kümeleri:

- Ders indisleri kümesi $I=\{i \mid i=1,\ldots,185\}$
- Öğrenci indisleri kümesi $S=\{s \mid s=1,\ldots,149\}$
- Oturum indisleri kümesi $O=\{o \mid o=1,...,4\}$
- Kitap indisleri kümesi $K=\{k \mid k=1,\ldots,L\}$

Parametreler:

M: skaler

$\mathbf{P_{si}}$**:** *s.* öğrenci, *i.* dersi alıyor ise 1, diğer durumda 0 olan öğrenci ders matrisi elemanları

$\mathbf{NC_s}$**:** *s.* öğrencinin kayıtlı olduğu ders sayısı

Karar değişkenleri:

- $x_{ik} = \begin{cases} 1, i.\text{ders},\ k.\text{kitap türüne atanırsa} \\ 0, \text{diğer durum} \end{cases}$
- $z_{io} = \begin{cases} 1, i.\text{ders } o.\text{oturuma atanırsa} \\ 0, \text{diğer durum} \end{cases}$
- $y_{ko} = \begin{cases} 1, k.\text{kitap türü } o.\text{oturuma atanırsa} \\ 0, \text{diğer durum} \end{cases}$

- $q_k = \begin{cases} 1, k.\text{kitap türü kullanıldıysa} \\ 0, \text{diğer durum} \end{cases}$

$$1 \leq \sum_i x_{ik} \leq 10 \qquad \forall k \qquad (4.1)$$

$$\sum_o z_{io} = 1 \qquad \forall i \qquad (4.2)$$

$$\sum_o y_{ko} = 1 \qquad \forall k \qquad (4.3)$$

$$\sum_{k=1}^{l} x_{ik} \geq 1 \qquad \forall i \qquad (4.4)$$

$$\sum_i P_{si} * z_{io} \leq 5 \qquad \forall (s,o) \qquad (4.5)$$

$$q_k \leq \sum_o y_{ko} \leq M * q_k \qquad \forall k \qquad (4.6)$$

$$q_k \leq \sum_i x_{ik} \leq M * q_k \qquad \forall k \qquad (4.7)$$

$$\sum_i \sum_o P_{si} * z_{io} = NCs \qquad \forall s \qquad (4.8)$$

$$-(1 - x_{ik}) \leq z_{io} - y_{ko} \leq (1 - x_{ik}) \qquad \forall (i,o,k) \qquad (4.9)$$

$$x_{ik} \in \{0,1\} \qquad \forall (i,k) \qquad (4.10)$$

$$z_{io} \in \{0,1\} \qquad \forall (i,o) \qquad (4.11)$$

$$y_{ko} \in \{0,1\} \qquad \forall (k,o) \qquad (4.12)$$

$$q_k \in \{0,1\} \qquad \forall k \qquad (4.13)$$

Kısıtları altında

$$enk\ a = \sum_k q_k \qquad (4.14)$$

Kısıt kümesi (1) ile bir ders bir kitap türüne atandığında, her kitap türü için derslerin toplam sayısının on değerini geçmesi engellenmekte ve o kitap türünde en az bir ders olması sağlanmaktadır.

Açıköğretim Fakültesi sınavlarında kullanılan cevap kâğıdı şablonuna göre, bir oturumda bir öğrenci en fazla beş ders alabilir, daha az olması muhtemel ve modelde izin verilen bir durumdur, (1) numaralı kısıt kümesi ile bir kitapta yer alması gereken derslerin

alt ve üst sınırları çizilmektedir. Bu kısıt kümesindeki alt ve üst limitler, bir öğrencinin bir oturumda aldığı ders sayısı beşi geçmediği sürece değiştirilebilir.

Kısıt kümesi (2) ile bir dersin sadece bir oturuma atanması ve aynı zamanda tüm derslerin mutlaka bir oturuma atanması sağlanmaktadır. Sınav soruları aynı olacağından, bir ders farklı kitaplarda yer alsa bile, bir dersin sadece bir oturumda yer alması zorunludur.

Bir kitap türü de sadece bir oturuma atanabilir. Bu da (3) numaralı kısıt kümesi ile sağlanmaktadır.

(4) numaralı kısıt kümesi ile tüm derslerin mutlaka bir kitap türüne atanması sağlanmaktadır. Ders kümesinde açıkta ders kalmayacak şekilde, tüm derslerin mutlaka bir kitap türüne yerleştirilmesi zorunludur.

Kısıt kümesi (5) ile bir öğrencinin bir oturumda beşten fazla ders alması engellenir. Yukarıda bahsettiğimiz gibi cevap kâğıdı şablonu dolayısıyla bir öğrencinin bir oturumda alabileceği en fazla ders sayısı beştir.

Herhangi bir (k) kitap türü, oturum (o)'ya atanırsa, kısıt kümesi (6)'deki eşitsizliğin sağ tarafı q_k'nın 1 değerini almasını zorlar. (k) kitap türünün, (o) oturumuna atanmaması durumunda ise yine kısıt kümesi (6)'deki eşitsizliğin sol tarafı sıfır değerini alır. Böylece kısıt kümesi (6), q_k'nın 0 değerini almasını zorlamaktadır.

Herhangi bir (i) dersi, (k) kitap türüne atanırsa, kısıt kümesi (7)'deki eşitsizliğin sağ tarafı q_k'nın 1 değerini almasını zorlar. (i) dersinin, (k) kitap türüne atanmaması durumunda ise yine kısıt kümesi (7)'deki eşitsizliğin sol tarafları sıfır değerini alır. Böylece kısıt kümesi (7), q_k'nın 0 değerini almasını zorlamaktadır.

Kısıt kümesi (8) ile bir (i) dersi (o) oturumuna atanırsa, her öğrenci için, öğrenci ders matrisi olan P_{si} oturum ve dersler üzerinden toplanarak, bulunan değerin her öğrencinin ders sayısını veren NC_s parametresine eşit olup olmadığının kontrolü yapılır.

Kısıt kümesi (9) çift yönlü çalışan bir kısıt kümesidir. Bir (i) dersi (k) kitap türüne atandığında, yani x_{ik} değişkeni bir değerini aldığında, bir (i) dersinin (o) oturumuna atanması karar değişkeni z_{io} ve bir (k) kitap türünün (o) oturumuna atanması karar değişkeni olan y_{ko} karar değişkenlerinin de bir değerini almasını zorlar. Böylelikle bir (i)

dersi (k) kitap türüne atandığında, (i)'nin oturumu (o) ve (k)'nın oturumu (o) aynı olmaya zorlanır.

(14) numaralı denklem amaç fonksiyonudur. Amaç fonksiyonunda, (k) kitap türünün kullanılması karar değişkeni olan q_k'nın toplamının en küçüklenmesi amaçlanmaktadır. Modelde tek amaç fonksiyonu yer almaktadır.

4.2 Sayısal Sonuçlar

Örneklem sınav olarak seçilmiş olan Kuzey Amerika programı 2017-2018 eğitim yılı güz dönemi dönem sonu sınavı problemi, bir optimizasyon yazılımı olan GAMS (The General Algebraic Modeling System) programı ile kodlanarak çalıştırılmıştır. Program sürümü olarak 23.3.3 versiyonu kullanılmıştır. Problem 0-1 tam sayılı doğrusal bir yapıda modellenmiş ve çözücü olarak tam sürümü mevcut olan CPLEX çözücüsü kullanılmıştır.

Bu sınava ait örneklem boyutları 149 öğrenci ve 185 adet distinct ders şeklindedir. Gams yazılımının modeldeki öğrenci ve derslerinin bilgilerini okuması bir 0-1 öğrenci-ders matrisi hazırlanarak sağlanmıştır. Problem boyutu büyük geldiğinden çözüm alınamamış ve problem ilk sıradaki 30 öğrenci ve bu öğrencilere ait 88 ders alınarak daha küçük boyutlu bir problem haline getirilmiştir. Bu veriler de örneklem 2 olarak tanımlanmıştır. Örneklem 2 verisinin GAMS ile çalıştırılması sonucu, sonuç dosyaları incelenmiş ve elde edilen oturum düzeni Ek1'de gösterilmiştir.

Elde edilen oturum düzeninde 88 dersin her birinin sadece bir oturumda yer aldığı, her bir kitabın sadece bir oturumda yer aldığı, 88 dersin her birinin yerleştirilmesinin yapıldığı ve kitap ve derslerin atandıkları oturumların aynı olduğu görülmektedir.

Ancak matematiksel modelleme yönteminin, çözümü oldukça küçük boyutlu bir örneklem için vermesi ve hem Açıköğretim sistemi sınavların öğrenci sayılarının fazlalığı hem de sistemin kendi başına oldukça karmaşık bir yapısının olması sebebiyle aynı kısıtlar ve amaç fonksiyonu kullanılarak, bir sezgisel algoritma geliştirilmesine karar verilmiştir. Probleme ait matematiksel modelin boyut analizi Tablo 4.1'de verilmiştir. Bu tabloyla birlikte modelin ne kadar karmaşık bir yapıya sahip olduğu daha iyi görülmektedir.

Tablo 4.1 *Boyut Analizi*

Kısıt No	**İndisler**	**Toplam kısıt sayısı**
1	k	k
2	i	i
3	k	k
4	i	i
5	s, o	$s \times o$
6	k	k
7	k	k
8	s	s
9	i,o,k	$i \times o \times k$
Değişken	**İndisler**	**Toplam değişken sayısı**
x_{ik}	i, k	$i \times k$
z_{io}	i, o	$i \times o$
y_{ko}	k, o	$k \times o$
q_k	k	k
NC_s	-	1
Toplam kısıt sayısı	2i+4k+s+so+iok	
Toplam değişken sayısı	k+ik+io+ko+1	

Ele alınan örneklem verileri, Açıköğretim sistemi sınavları içinde oldukça az veriye sahip olan bir programın, sadece küçük bir parçası alınarak elde edilmiştir. Sadece bu verilerle dahi modelin kısıt ve değişken sayıları çok büyük değerlerdir, boyut analizinde bulunan değerlere göre, 30 öğrencilik örneklem 2'nin kısıt ve değişken değerleri aşağıdaki gibidir:

Toplam kısıt sayısı= 2*88 + 4*12 + 30 + 30*4 + 88*4*12 = **4598**

Toplam değişken sayısı = 12 + 88*12 + 88*4 + 12*4 + 1 = **1469**

5. ÇOK OTURUMLU SINAVLARDA KİTAPÇIK OPTİMİZASYONU İÇİN GELİŞTİRİLEN SEZGİSEL YÖNTEM

Çok oturumlu sınavlarda kitapçık optimizasyonu problemi, diğer eğitimsel zaman çizelgeleme problemlerinde olduğu gibi, çok fazla değişken ve kısıt kümesine sahip olduğu için matematiksel modelleme yöntemiyle en iyiye ulaşılması zor problemdir.

Bölüm 4'de ele alınan, çok oturumlu sınavlarda kitapçık optimizasyonu problemi için geliştirilen matematiksel modelin çözümü incelenmiş ve boyut analizinde de görüldüğü gibi çok fazla kısıt ve karar değişkeni kümesine sahip olduğu için en iyi çözüme matematiksel modelleme yöntemiyle ulaşılamayacağına karar verilmiştir. Bu noktada problem için sezgisel yöntem ile bir algoritma geliştirilmesi ihtiyacı duyulmuş ve matematiksel model ile aynı kısıt ve amaç fonksiyonuna sahip bir sezgisel algoritma geliştirilmiştir.

Bölüm 4'de bahsedildiği gibi mevcut sistemde, Açıköğretim sistemi sınavları için hazırlanan oturum düzenleri bölüm bilgisini de göz önünde bulundurarak hazırlanmaktadır. Her bölümün kendi içinde kitap-oturum, ders-oturum, ders-kitap yerleştirmesi yapılır. Matematiksel model geliştirilirken bölüm bilgisi dikkate alınmayıp girdi olarak sadece öğrenciler ve üzerlerindeki dersler baz alınmış ve bölüm bilgisi dışarıda bırakılmıştır. Sezgisel algoritma geliştirilirken de yine aynı şekilde bölüm bilgisi göz ardı edilip sadece öğrenciler ve üzerlerindeki dersler dikkate alınarak bir sezgisel algoritma geliştirilmiştir. Problem verisi olarak da matematiksel modellemede küçük bir örneklemi kullanılan yurt dışı programı Kuzey Amerika Programına ait 2017-2018 eğitim dönemi güz dönemi dönem sonu sınavına ait verilerin bütünü kullanılmıştır.

Geliştirilen sezgisel algoritma, ilk önce öğrenci derslerinin oturumlara yerleştirilmesini daha sonra da her bir oturumdaki derslerin oturum bazında kitaplara yerleştirilmesini gerçekleştirir. Bu işlemler iki ardışık süreç olarak düşünülmüş, algoritma adımları ve süreç akış şemaları da buna göre tasarlanmıştır.

Algoritma bu işlemlere hangi öğrenciden başlayacağına ise; öğrencilerin aldıkları ders sayılarına göre büyükten küçüğe gruplama yaparak ve devamında en büyük ders sayısına sahip gruptan başlayarak ilerleyecektir. Aynı grup içindeki öğrencilerden hangi öğrenciyi seçeceğine ise derslerin ağırlıklarına göre karar vermektedir. Algoritmanın parametreleri ve adımları izleyen alt bölümlerde verilmiştir.

5.1 Algoritmanın Parametreleri

Üst sınır kitap - ders sayısı: Bu parametre, bir kitapta yer almasına izin verilen en fazla ders sayısını gösteren parametredir. Algoritma, dersleri yerleştireceği uygun kitabı ararken her bir kitap için bu değeri kontrol eder. Parametrenin değeri algoritma için 10 olarak belirlenmiştir.

Dersin oturumu: Bu parametre, dersin hangi oturumda yer aldığını gösteren parametredir. Oturum düzeninde bir ders sadece bir oturumda yer alabilir. Bir ders bir oturuma atandığı zaman, o dersin oturum bilgisi ders gösterim kodu – oturum bilgisi sütunlarından oluşacak bir liste şeklinde kaydedilir ve algoritma tekrar bu ders ile karşılaşırsa, bu ders için tekrar oturum ataması gerçekleştirmez ve bu dersle ilk karşılaşması olmadığı için bu dersi geçer. Oluşturulacak örnek dersin oturumu parametresi listesi gösterimi aşağıdaki gibidir:

Tablo 5.1 *Örnek Dersin Oturumu Parametresi Listesi*

Ders no	Ders gösterim	Oturum
1	BİL101U	2
2	MAT105U	4
3	İKT101U	3
-	-	-
185	YBS309U	1

Üst sınır öğrenci - oturum - ders sayısı: Bu parametre, oturum bazında her bir öğrencinin aldığı ders sayısını göstermektedir. Bir öğrenci bir oturumda en fazla beş ders alabilir. Algoritma için bu parametrenin değeri 5 olarak belirlenmiştir. Parametre farklı çözümler denenmek istendiğinde beşi geçmeyecek şekilde farklı değerler de alabilir.

Üst sınır öğrenci - oturum - kitap sayısı: Bu parametre, oturum bazında öğrencinin aldığı kitap sayısını göstermektedir. Bir öğrenci bir oturumda sadece bir kitap alabilir ya da o oturumda kitap almayabilir. Öğrencinin bir oturumda ders veya dersleri varsa, hepsinin sadece bir kitapta yer alması gerekmektedir. Algoritma için bu parametrenin değeri 1'dir.

Kitaptaki ders sayısı: Bu parametre, her bir kitapta toplam kaç ders olduğunu gösteren parametredir. Oturumdaki derslerin kitaplara yerleştirilmeleri yapılırken, bu parametreye bakılarak ders yerleştirme işlemi yapılır. Kitaplara her ders yerleştirilmesi yapıldığında bu parametre değerleri güncellenir. Örnek bir test case ve kitaptaki ders sayısı parametresi listesi gösterimi aşağıdaki gibidir:

Tablo 5.2 *1. Oturumdaki Kitapları Gösteren Bir Test Case*

1. oturum kitapları			
Kitap 1	Kitap 2	Kitap 3	Kitap 4
BİL101U	TÜR201U	SOS105U	MEİ101U
HUK101U	SOS113U	ÇEK101U	MEİ103U
İKT103U	TAR201U		MEİ303U
TÜR201U	PSİ103U		MUH303U
SOS113U	İLT107U		PZL207U
TAR201U	MUH103U		
	MAT103U		
	SOS101U		

Tablo 5.3 *Örnek Kitaptaki Ders Sayısı Parametresi Listesi*

Kitaplar	Ders Sayısı
Kitap 1	6
Kitap 2	8
Kitap 3	2
Kitap 4	5

Öğrenci ders katsayısı: Bu parametre öğrencinin oturum bazında aldığı ders sayısını gösteren parametredir. Bu parametreye ikinci süreç olarak tanımladığımız derslerin kitaplara yerleştirilmesi adımında bakılır. İlgili oturumda dersleri olan öğrenciler, katsayısı en büyük öğrenciden en küçük öğrenciye ilerleyecek şekilde sıralanır ve katsayısı en büyük öğrenciden başlayarak derslerin kitaplara yerleştirilme işlemi yapılır.

5.2 Algoritma Adımları

Algoritma birbirini izleyen iki ardışık süreç olarak tasarlanmıştır, bunlardan ilki derslerin oturumlara yerleştirilmesi işlemidir. Diğeri ise, oturum ataması yapılan derslerin, yine o oturumlarda yaratılacak kitaplara yerleştirilme işlemidir. Bu süreçler ve algoritma adımları şunlardır.

5.2.1 Derslerin oturumlara yerleştirilme süreci adımları:

Algoritmanın ilk ayağı olarak tanımladığımız bu sürece ait algoritma adımları ve süreç akış şeması aşağıda verilmiştir.

Adım 1: Öğrencilerin ders sayılarının bulunması

Her bir öğrencinin kaç ders aldığı hesaplanır.

Adım 2: Ders ağırlıklarının bulunması

Her bir dersin kaç öğrenci tarafından alındığı bulunur. Bu değer dersin ağırlığı olarak belirlenmiştir. Bir ders ne kadar çok öğrenci tarafından alınıyorsa, dersin değeri o kadar yüksektir. Örnek bir gösterim aşağıdaki tabloda verilmiştir. Bu tablodan BİL101U dersinin 59 öğrenci tarafından alındığı ve BİL101U dersinin ağırlığının 59 olduğu görülmektedir.

Tablo 5.4 *Örnek Ders Ağırlıkları Listesi*

Ders gösterim kodu	Ders ağırlığı
BİL101U	59
HUK101U	37
İKT103U	33
TÜR201U	30
SOS113U	27

Adım 3: Öğrencilerin ders ağırlıklarının bulunması

Her bir öğrenci için, aldığı ders ya da derslerin 2. adımda bulunan ağırlık değerleri toplanarak, öğrenci ders ağırlığı değeri elde edilir. Örnek bir gösterim aşağıdaki tabloda verilmiştir.

Tablo 5.5 *Örnek Öğrenci Ders Ağırlığı Listesi*

Öğrenciler	Öğrenci dersleri	Ders ağırlığı	Öğrenci ders ağırlığı
Tc1	BİL101U	59	186
Tc1	HUK101U	37	
Tc1	İKT103U	33	
Tc1	TÜR201U	30	
Tc1	SOS113U	27	
Tc2	ULİ305U	18	33
Tc2	İŞL401U	15	

Adım 4: Öğrenci derslerinin oturumlara yerleştirilmesi

En fazla alınan ders sayısı grubundan ve bu grup içinde de en yüksek öğrenci ders ağırlığına sahip öğrenci seçilerek dersler yerleştirilmeye başlanır. Dersler en az ders sayısına sahip ve en küçük numaralı oturumdan başlayacak şekilde, her bir öğrencinin dersi bitene kadar oturumlara birer birer yerleştirilir. Oturumlarının ders sayılarının eşit olması durumunda yerleştirmeye en küçük numaraları oturumdan başlanır.

Adım 5: Dersin oturumunun olup olmadığının kontrolü

Öğrencinin dersleri oturumlara yerleştirilirken, ilk önce dersin oturumu parametresine bakılır. Eğer dersin oturumu varsa o ders için oturum ataması yapılmaz, öğrencinin oturum ataması yapılmayan diğer dersine geçilir. Öğrencinin tüm derslerinin oturumu var ise, sıradaki öğrenciye geçilir.

Adım 6: Öğrencilerin oturum bazında ders sayısı kontrolü

Bir öğrencinin bir dersi bir oturuma atandığında, tüm öğrencilerin dersleri gezilerek o oturumda, üst sınır öğrenci - oturum - ders sayısı parametre değeri olan 5'i geçip geçmediği kontrol edilir. Eğer 5 dersi geçen öğrenci tespit edilirse, o ders sıradaki oturuma atanır. Adım 6 yeni oturum için de tüm öğrenciler üzerinden tekrarlanır. Parametre değerini aşan bir öğrenci söz konusu değilse, öğrencinin sıradaki dersine, öğrencinin tüm dersleri yerleştirildiyse sıradaki öğrenciye geçilir. Parametre değerini aşan bir durum söz konusu ise ders sorunsuz bir şekilde yerleştirilene kadar Adım 6 tekrarlanır.

Adım 7: Dersin Oturumu Listesinin Güncellenmesi

Adım 6'da bir öğrencinin bir dersinin oturum ataması yapıldıktan sonra, o ders ve oturumu bilgileri, algoritma parametresi olan dersin oturumu listesine eklenir. Böylelikle bu ders tekrar algoritmanın karşısına geldiğinde, bu listede yer aldığı için algoritma bu ders için işlem yapmaması gerektiğini anlar.

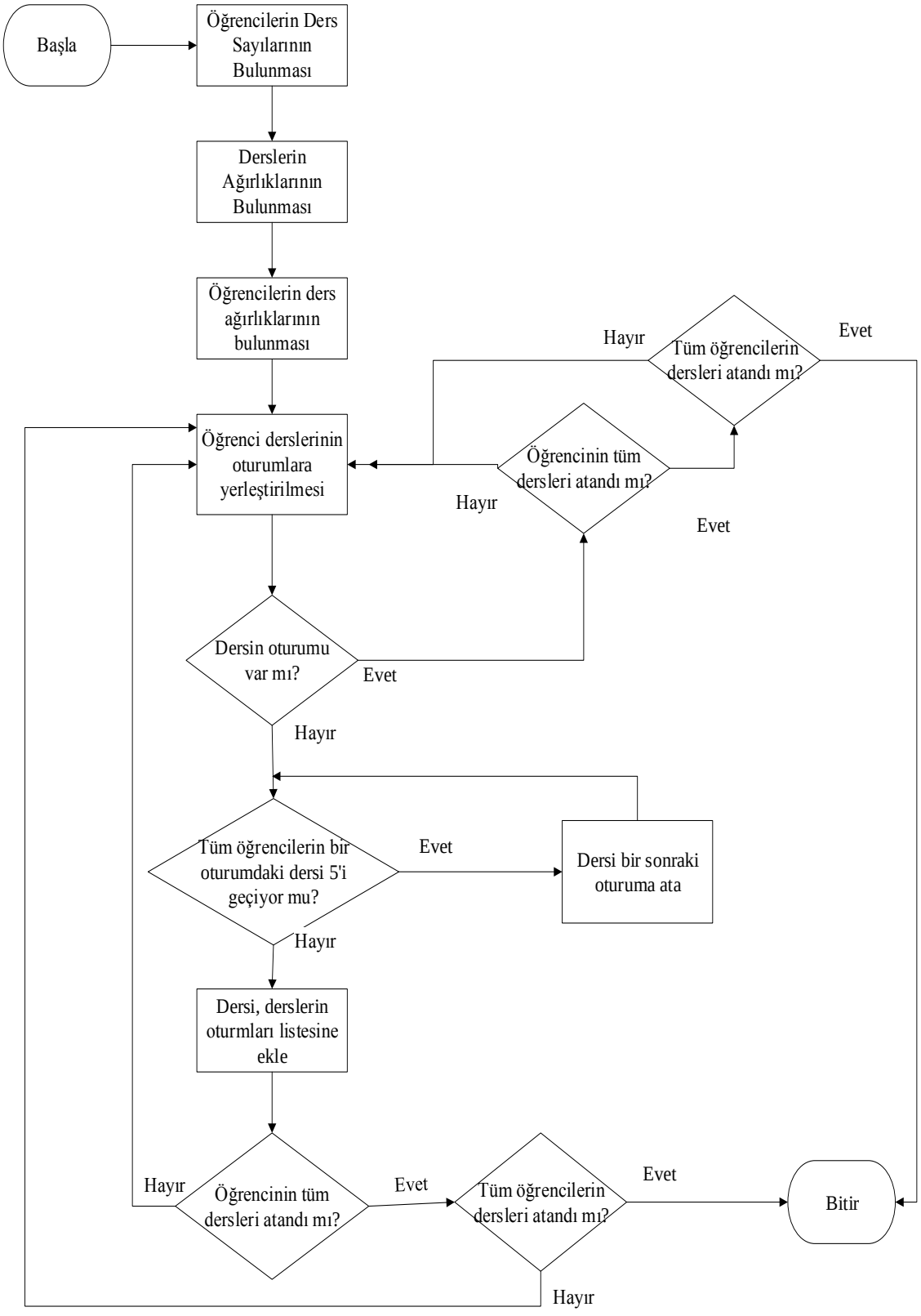
Adım 8: Öğrencinin bütün derslerinin atandığının kontrolü

Öğrencinin tüm derslerinin oturumlara atanıp atanmadığı kontrol edilir. Eğer henüz bir oturuma atanmamış dersi varsa, Adım 4 aynı öğrenci için tekrarlanır. Eğer öğrencinin tüm dersleri bir oturuma yerleştirildiyse Adım 9'a geçilir.

Adım 9: Bütün öğrencilerin derslerinin atandığının kontrolü

Sınavda yer alan tüm öğrencilerin derslerinin oturumlara atanıp atanmadığının kontrolü yapılır. Eğer sistemde oturum ataması yapılmamış dersi olan öğrenci varsa 4. Adım tekrarlanır. Eğer sistemde oturum ataması yapılmamış dersi olan öğrenci kalmadıysa, derslerin oturumlara yerleştirilme süreci için algoritma sonlanır.

Derslerin oturumlara yerleştirilmesi sürecine ait süreç akış şeması Şekil 5.1' de gösterilmiştir.



Şekil 5.1 *Derslerin Oturumlara Yerleştirilme Sürecine ait Süreç Akış Şeması*

5.2.2 Oturumlardaki derslerin kitaplara yerleştirilme süreci adımları:

Adım 10: Oturumun belirlenmesi

Anadolu Üniversitesi tarafından gerçekleştirilen sınavlar 1,2,3,4 gibi farklı sayıda oturumlara bölünerek gerçekleştirilebilir. Örneklem problemimiz 3 oturum olacak şekilde düzenlenmiştir. Algoritma, oturum atamaları gerçekleştirilen dersleri kitaplara atamak için bir oturumu seçmelidir. Bunun için de en küçük oturum numarasından başlar ve o oturumdaki tüm dersleri kitaplara yerleştirmeyi tamamladığında küçükten büyüğe olacak şekilde bir sonraki oturuma geçer.

Adım 11: Oturumdaki dersleri alan öğrencilerin bulunması

İşlem yapılan oturumdaki dersleri alan öğrenciler bulunur.

Adım 12: Öğrenci ders katsayılarının hesaplanması

Oturumdaki dersleri alan öğrenciler bulunduktan sonra, her bir öğrencinin o oturumda ne kadar ders aldığı bulunur. Bu değer öğrencinin ders katsayısı olarak belirlenmiştir. Örnek bir gösterim aşağıdaki tabloda verilmiştir:

Tablo 5.6 *Örnek Öğrenci Ders Katsayıları Listesi*

1. Oturum Derslerini Alan Öğrenciler		
Öğrenci	Alınan ders sayısı	Öğrenci ders katsayısı
Tc1	5	5
Tc2	1	1
Tc3	3	3

Adım 13: Öğrencinin derslerinin kitaplara yerleştirilmesi

Bir öğrencinin oturumdaki dersleri kitaplara yerleştirilirken, öğrenci seçimi Adım 12'de hesaplanan öğrenci ders katsayısına göre yapılır. Oturum içinde öğrenci ders katsayısı en fazla olan öğrenci seçilir ve en fazladan en aza doğru ilerlenir. Öğrenci ders katsayılarında eşitlik olması durumunda öğrenci ders katsayı ağırlıkları listesi devreye girer. Her bir öğrencinin, her oturumda aldığı ders ağırlıkları toplanarak öğrenci ders katsayı ağırlığı bulunur. Aynı katsayıya sahip öğrencilerde öğrenci ders katsayısı ağırlığı en yüksek öğrenciden başlanarak en düşük öğrenciye ilerleyecek şekilde derslerin

kitaplara yerleştirilmesi yapılır. İlgili katsayı grubundaki öğrencilerin ders yerleştirmeleri tamamlandığında, bir sonraki katsayı grubuna geçilir. Oturumdaki tüm öğrencilerin dersleri kitaplara yerleştirildiğinde de sıradaki oturuma geçilir. Öğrenci ders katsayısı 5 olan 2 öğrenciye ait örnek bir gösterim aşağıdaki tabloda verilmiştir. Algoritma burada ders katsayıları eşit olduğu için, öğrenci ders katsayı ağırlığı yüksek olan öğrenciyi seçecektir.

Tablo 5.7 *Örnek Öğrenci Ders Katsayı Ağırlığı Listesi*

Öğrenci	Ders gösterim kodu	Dersin ağırlığı	Öğrenci	Ders gösterim kodu	Dersin ağırlığı
Tc 1	İKT103U	33	Tc 2	PSİ103U	24
Tc 1	TÜR201U	30	Tc 2	İLT107U	23
Tc 1	SOS113U	27	Tc 2	MUH103U	21
Tc 1	TAR201U	26	Tc 2	MAT103U	19
Tc 1	PSİ103U	24	Tc 2	SOS101U	19
Öğrenci ders katsayı ağırlığı		**140**	Öğrenci ders katsayı ağırlığı		**106**

Adım 14: Sistemde kitap var mı kontrolü ve yeni kitap yaratılması

Öğrenci seçiminden sonra, algoritma daha önce sistemde yaratılmış kitap bulunup bulunmadığını kontrol eder. Eğer işleme alınan ilk öğrenciyse ve sistemde hiç kitap bulunmuyorsa, yeni kitap yaratılır ve öğrencinin tüm dersleri o kitaba eklenir.

Tüm öğrencilerin dersleri yerleştirildiyse, süreç bir sonraki oturuma geçer. Sistemde hala dersleri kitaplara yerleştirilmeyen öğrenciler varsa Adım 13 tekrarlanır.

Adım 15: Birebir eşleşen kitap kontrolü

Sistemde daha önceden yaratılan kitap varsa, öğrencinin tüm dersleri birlikte olacak şekilde, yaratılan kitaplardan birinin içinde öğrenci derslerinin mevcut olup olmadığı kontrol edilir. Eğer öğrencinin tüm derslerini içeren bir kitap sistemde mevcutsa, algoritma o öğrencinin derslerini bir kitaba yerleştirmeye gerek duymaz. Tüm öğrencilerin dersleri yerleştirildiyse, süreç bir sonraki oturuma geçer. Sistemde hala dersleri kitaplara yerleştirilmeyen öğrenciler varsa Adım 13 tekrarlanır.

Adım 16: En az bir ders kesişimi içeren kitap kontrolü

Adım 15'teki kontrolde eğer öğrencinin tüm dersleri birlikte, sistemde var olan kitaplardan bir tanesinin içinde yer almıyorsa, en az bir dersi kesişen kitap var mı kontrolü yapılır. Kesişim sayısı sadece bir olabileceği gibi birden fazla da olabilir, önemli olan bir kesişim yakalayabilmektir. Kesişim içeren kitap bulunursa Adım 17'ye, kitap bulunmazsa Adım 18'e geçilir.

Adım 17: Kesişim ders içeren kitap listesinin bulunması

Sistemde öğrencinin dersleriyle en az bir dersin kesiştiği kitaplar bulunur. Burada bir ya da birden fazla kitap söz konusu olabilir, bulunan kitaplar listelenir.

Adım 18: Sistemde uygun kitap var mı kontrolü

Adım 16'te eğer en az bir dersin kesiştiği bir kitap bulunmazsa, sistemde var olan kitapların üst sınır kitap ders sayısı parametresine göre uygunlukları kontrol edilir. Uygun bulunan kitap listeleri arasından, en az ders sayısına sahip olan kitaba dersler yerleştirilir. Tüm öğrencilerin dersleri yerleştirildiyse, süreç bir sonraki oturuma geçer. Sistemde hala dersleri kitaplara yerleştirilmeyen öğrenciler varsa Adım 13 tekrarlanır.

Eğer bu kontrolde uygun kitap bulunmazsa Adım 14 uygulanarak yeni kitap yaratılarak tüm dersler o kitaba eklenir. Tüm öğrencilerin dersleri yerleştirildiyse, süreç bir sonraki oturuma geçer. Sistemde hala dersleri kitaplara yerleştirilmeyen öğrenciler varsa Adım 13 tekrarlanır.

Adım 19: Kesişen derslerin kitabına öğrencinin diğer derslerinin yerleştirilmesi

Adım 17'de listelenen kesişim içeren kitaplar arasından, en fazla ders kesişimine sahip ilk kitaba öğrencinin kesişim dışında kalan dersleri yerleştirilir.

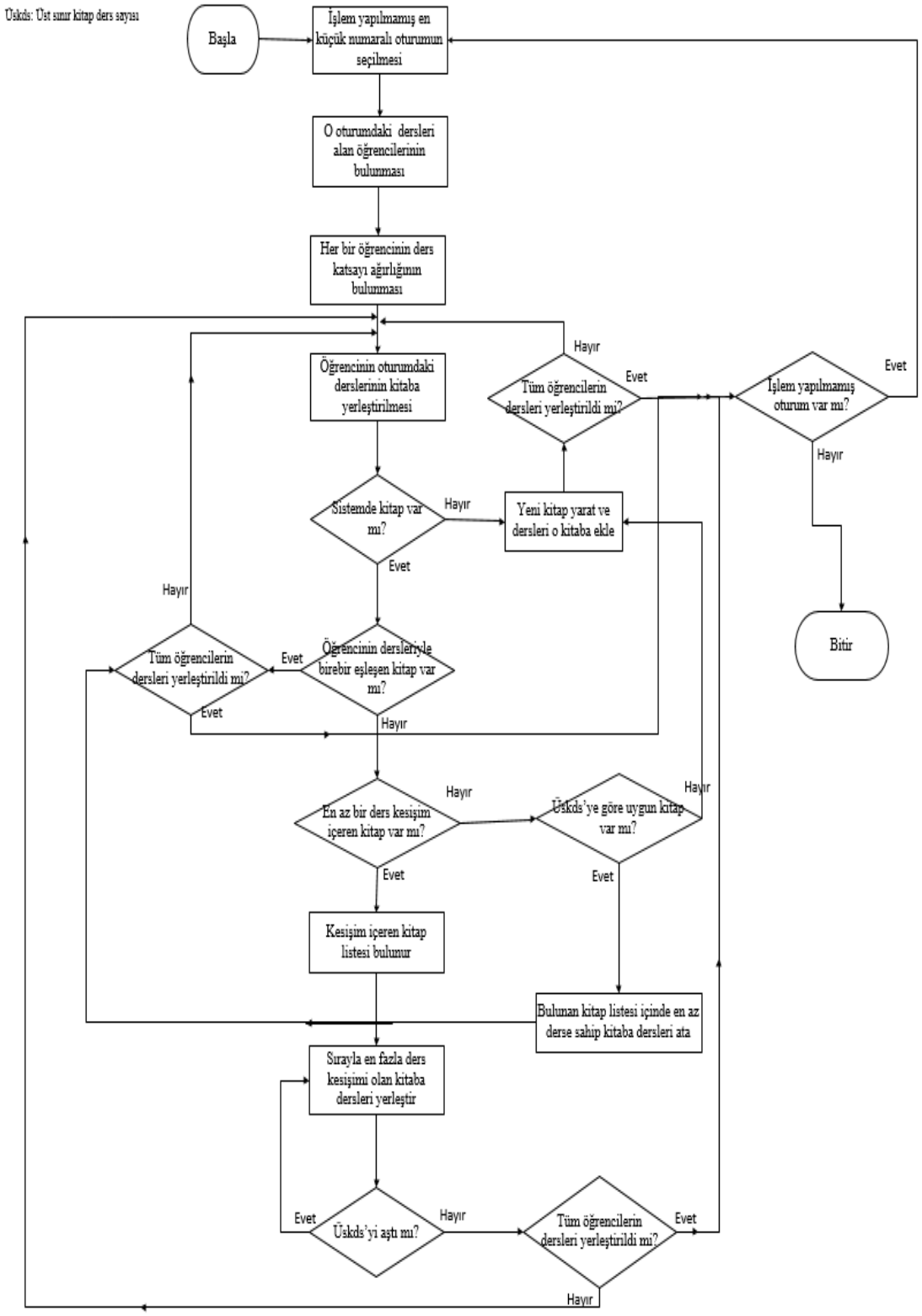
Adım 20: Kitap üst sınırı aştı mı kontrolü

Adım 19'da en fazla ders kesişiminin bulunduğu kitaba öğrencinin kalan dersleri yerleştirildikten sonra, kitabın uygunluğu üst sınır kitap ders sayısı parametresine göre kontrol edilir. Eğer kitaptaki ders sayısı bu parametreyi geçiyorsa Adım 19 tekrarlanır ve dersler bir sonraki uygun kitaba yerleştirilir.

Eğer derslerin atandığı kitap üst sınır kitap ders parametresine göre uygunsa, sistem tüm öğrencilerin derslerinin kitaplara atanıp atanmadığını kontrol eder. Sistemde dersleri kitaplara yerleştirilmeyen öğrenciler varsa adım 13 tekrarlanır. Eğer tüm öğrenciler için derslerin kitaplara yerleştirilme süreci tamamlandıysa, algoritma bir sonraki oturuma geçer.

Tüm bu algoritma adımları sırayla sistemdeki tüm oturumlar için tekrarlanır. Sistemdeki en son sıradaki oturum için de tüm adımlar tamamlandığında derslerin kitaplara yerleştirilme süreci algoritması sonlanır. Böylelikle çok oturumlu sınavlarda kitapçık optimizasyonu problemi için algoritma adımları tamamlanmış olur.

Derslerin kitaplara yerleştirilmesi sürecine ait süreç akış şeması Şekil 5.2'de gösterilmiştir.



Şekil 5.2 *Derslerin Kitaplara Yerleştirilme Sürecine ait Süreç Akış Şeması*

5.3 Sezgisel Algoritma Sonuçları

Geliştirilen sezgisel algoritma Java programlama dili ile kodlanmıştır. Örneklem verisi olan, Açıköğretim sistemi yurt dışı programına ait Kuzey Amerika Programı sınav verileri kullanılarak kod çalıştırılmıştır. Program çıktısı oturum düzeni Ek-3'de gösterilmiştir. Programın çıktı verdiği oturum düzeninin sistemin tüm kısıtlarına uygunluğu hazırlanan testler ile kontrol edilmiştir.

Gerçek sınavın ders atama verileriyle, algoritmanın çıktı verdiği oturum düzeni kullanılarak yapılan ders atama verileri karşılaştırılmıştır. Bulunan sonuçlar aşağıdaki gibidir:

Tablo 5.8 *Ders Ataması Sonuç Karşılaştırması*

Gerçek sınav ders ataması		**Algoritma ders ataması**	
Distinct kitap türü	52	Distinct kitap türü	29
Toplam basılan kitap	315	Toplam basılan kitap	424
Toplam basılan sayfa	5386	Toplam basılan sayfa	8240

Oluşturulan distinct kitap sayısı yeni algoritma sonucunda daha az değere ulaşmıştır. Bununla beraber öğrencilerin derslerinin oturumlara dağıtılması işleminin dengeli bir şekilde gerçekleştirilmesi sonucunda, öğrenciler daha fazla oturumda sınava girmekte ve oturumlarda birbirine yakın değerlerde dersten sınava girmektedir. Öğrencilerin sınavda aldıkları ders sayısı çeşitliliği bu sınav için 1 ile 12 arasında değişmektedir. Oturumda en fazla 5 ders alınabilir kısıtıyla birlikte, bu ders sayısındaki çeşitlilik oranı derslerin oturumlara dengeli dağıtılması durumunu beraberinde getirmiştir.

Öğrencinin daha fazla sınava katılması, toplam basılan kitap sayısındaki artışı açıklamaktadır. Gerçek sınav ve algoritma verileriyle gerçekleştirilen ders atamaları farklı kriterler için de karşılaştırılmış ve edinilen sonuçlarla, geliştirilen algoritmanın avantaj ve dezavantajları araştırılmıştır. Tespit edilen avantajlar, dezavantajlar ve karşılaştırma yapılan veri grubu aşağıda verilmiştir.

Avantajlar:

1. Bölüm bilgisi kullanılmadığı için, oturum düzeni hazırlamaya olan yaklaşım daha kolay bir hale getirilmiştir. Bölüm bazında çalışma yapılmasına duyulan ihtiyaç ortadan kaldırılmış, sınavdaki öğrencilerin kimlik numaraları ve ders gösterim kodlarını içeren bir liste oturum düzeni oluşturmak için yeterli hale gelmiştir.

2. Mevcut sistemde oturum düzenleri manuel hazırlanmakta iken, geliştirilen algoritmanın kodlanması sonucu, program çıktısı ile oturum düzeni elde edilebilir bir sistem geliştirilmiştir. Böylelikle oturum düzeni hazırlamak için harcanan süre oldukça kısalmıştır.

3. Sınavın oturumu kümesi parametreye bağlanmıştır. Böylelikle kod çalıştırılarak, sınav öğrenci-ders verilerinin 1,2,3 veya 4 oturum değerleri için uygunlukları denenip sınav oturumuna karar verilebilir. Öğrencilerin oturum bazında en fazla alınabilen ders sayısı olan 5 engeline takılmadan, sınavların hangi oturum sayı veya sayılarında gerçekleştirilebileceği kontrol edilebilir bir durum haline gelmiştir. Daha az oturumda gerçekleştirilen sınavlar görevli ve baskı maliyetleri açısından iyileştirme sağlayan bir durumdur.

Örneklem sınav olan Kuzey Amerika programı sınavı, öğrenci verisi az olduğu için diğer Açıköğretim sistemi sınavlarının farklı olarak 3 oturumda gerçekleştirilmiştir. Algoritma da oturum sayısı 3 seçilerek çalıştırılmıştır. Ancak kod 4 oturum seçilerek de çalıştırılmış, sınavın 3 ve 4 oturum gerçekleşmesi durumunda sistemin nasıl bir sonuç vereceği aşağıdaki tablodaki gibi bulunmuştur. Sınavın 4 Oturum gerçekleştirilmesi durumunda basılan kitap ve sayfa sayısının arttığı görülmektedir.

Tablo 5.9 *Algoritma Ders Atamasının 3 ve 4 Oturumlar için Karşılaştırılması*

Algoritma ders ataması 3 oturum		**Algoritma ders ataması 4 oturum**	
Distinct kitap türü	29	Distinct kitap türü	26
Toplam basılan kitap	424	Toplam basılan kitap	545
Toplam basılan sayfa	8240	Toplam basılan sayfa	10672

4. Kitapta en fazla yer alabilecek ders sayısı parametreye bağlanmıştır. Böylelikle öğrencilerin oturum bazında en fazla alabildikleri ders sayısı olan 5 engeline takılmadan, kitapta yer alabilecek en fazla ders sayısı sınav bazında belirlenebilir bir durum haline getirilmiştir. Böylelikle kitaplarda aynı ya da birbirine yakın değer sayılarda ders yer alabilecektir.

5. Dersler, kitaplar ve öğrenciler oturumlara dengeli dağılmaktadır. Böylelikle her oturumda birbirine yaklaşık değerde ders, kitap ve öğrenci bulunmaktadır. Bu durum, bazı sınavlar için bir oturumda fazla yığılma olurken, bir oturumda belki sadece birkaç kitap için sınav yapılma durumunu ortadan kaldırır hale gelmiştir. Bu kriter için gerçek sınav ve algoritma ders atama değerlerinin karşılaştırılması aşağıdaki tablolardaki gibidir.

Tablo 5.10 *Oturum Bazlı Öğrenci ve Kitap Sayısı Karşılaştırması*

Gerçek sınav oturum düzeni		**Algoritma oturum düzeni**	
Oturum bazlı öğrenci ve kitap sayısı		Oturum bazlı öğrenci ve kitap sayısı	
1	138	1	142
2	141	2	142
3	35	3	140
Toplam	314	Toplam	424

Tablo 5.11 *Oturum Bazlı Ders Sayısı Karşılaştırması*

Gerçek sınav oturum düzeni		**Algoritma oturum düzeni**	
Oturum bazlı ders sayısı		Oturum bazlı ders sayısı	
1	90	1	62
2	64	2	62
3	31	3	61
Toplam	185	Toplam	185

Dezavantajlar:

1. 4 ve daha az dersi olan adayların derslerinin oturumlara bölünmesi adaylar için istenmeyen bir durum olabilir. Toplamda 3 dersi veya 2 dersi olan öğrencinin sadece 1 oturumda sınava girip, sınavlarını tamamlamaları arzu edilen bir durumdur. Ancak

oturumda alınabilecek en fazla ders sayısının 5 olması ve derslerin oturumlara eşit sayıda bölünmesi böyle bir sonucu da beraberinde getirmektedir.

6. SONUÇ VE ÖNERİLER

Tez kapsamında Anadolu Üniversitesi Açıköğretim sistemi sınavlarında, öğrencilerin derslerinin ders kitaplarına ve ders oturumlarına yerleştirilme işlemi olan oturum düzeni hazırlama süreci ele alınmıştır. Oluşturulacak kitap türünün en küçüklenmesi hedeflenerek, sistemin kısıtları altında oturum düzeni oluşturulması çalışılmıştır. Örneklem sınav olarak, gerçek sınav olan Açıköğretim sistemi yurt dışı programlarından Kuzey Amerika Programı 2017-2018 eğitim dönemi güz dönemi dönem sonu sınavı seçilmiştir. Yeni açılan bir program olması sebebiyle az öğrenci sayısına sahip bu programın örneklem sınavında 149 öğrenci ve 185 ders bulunmaktadır. Mevcut sistemde oturum düzeninin nasıl hazırlandığı araştırılarak, sistemin tüm kısıtları tespit edilmiştir.

İlk çözüm yaklaşımı olarak, problemin kısıtları, karar değişkenleri ve amaç fonksiyonu belirlenerek 0-1 tam sayılı doğrusal bir matematiksel model geliştirilmiştir. Geliştirilen matematiksel model Gams programı ile kodlanmış ve örneklem sınav verisiyle çalıştırılmıştır. Ancak hem kısıt ve değişken sayılarının fazla olması hem de modelin karmaşıklığından dolayı matematiksel model ile çözüm sağlanamamış, problem örneklem 2 olarak adlandırdığımız daha küçük bir boyutlu probleme indirgenmiştir. Örneklem 2'nin boyutları 30 öğrenci ve 88 dersten oluşmaktadır. Örneklem 2 için matematiksel modelleme yöntemi çözüm vermiş ancak oldukça küçük bir veri grubu için sonuç döndürdüğünden, çok büyük öğrenci sayılarına sahip Açıköğretim sistemi programları sınavları için oturum düzeni oluşturmada yetersiz kaldığı tespit edilmiştir.

Matematiksel modelleme çözüm yönteminin, çok oturumlu sınavlarda kitapçık optimizasyonu problemi çözümünde yetersiz kalmasıyla, yeni çözüm yaklaşımı olarak sezgisel algoritma yöntemine başvurulmuştur. Algoritma adımları iki ardışık süreçten oluşacak şekilde tasarlanmıştır. İlk süreç öğrenci derslerinin sistem kısıtları altında oturumlara yerleştirilmesi, ikinci süreç ise, oturum atamaları yapılan derslerin yine sistem kısıtları altında kitaplara yerleştirilmesi şekilde oluşturulmuştur. Algoritma sonucu oturum düzeni, gerçek sınavın oturum düzeni verileriyle karşılaştırılmış ve yeni sistemin oldukça fazla avantaja sahip olduğu görülmüştür.

Gelecek çalışmalarda, farklı sezgisel yöntemler denenerek sistemin performansı ölçülebilir. Derslerin sınav oturumlarına ve sınav kitaplarına yerleştirilme işlemi seçenekleri çok fazla kombinasyon kümesine sahip olduğu için farklı sezgisel

yaklaşımların sonuçlarının sistemin performansını nasıl etkilediği araştırılabilir. Literatürde yer alan eğitimsel zaman çizelgeleme problemi çözüm yaklaşımlarından olan metasezgisel ve hipersezgisel algoritma yaklaşımlarıyla da problemin çözümü araştırılabilir.

KAYNAKÇA

Abdullah, S., Burke, E.K. ve McCollum, B. (2005). "An investigation of variable neighbourhood search for university course timetabling". 2nd Multidisciplinary International Conference on Scheduling and Applications, 2, 413-427, New York, USA.

Akkoyunlu, E.A. (1973). "A linear algorithm for computing the optimum university timetable". The Computer Journal, 16(4), 347-350.

Aladag, C.H., Hocaoglu, G.A. ve Basaran, M. (2009). "The effect of neighborhood structures on tabu search algorithm in solving course timetabling problem". Expert Systems with Application, 36(10), 12349-12356.

Alkan, A. ve Özcan, E. (2003). "Memetic algorithms for timetabling". Proceedings of 2003 IEEE Congress on Evolutionary Computation, 1796-1802.

Alsmadi, O.M.K., Abo- Hammour, Z.S., Abu-Al-Nadi, D.I. ve Algsoon, A. (2011). "A novel genetic algorithm technique for solving university course timetabling problems". International Workshop on Systems, Signal Processing and their Applications, Algeria.

Asham, G.M., Soliman, M.M. ve Ramadan, A.R. (2011). "Trans genetic coloring approach for timetabling problem". Artificial Intelligence Techniques Novel Approaches & Practical Applications (IJCA), 1, 17-25.

Basir, N., Ismail, W. ve Norwawi, N.M. (2013). "A Simulated annealing for tahmidi course timetabling". Procedia Technology, 11, 437-445.

Botsalı, A.R. (2000). A Timetabling Problem: Constraint and Mathematical Programming Approaches. MSc Thesis, Bilkent University, Ankara, Türkiye.

Burke, Ed., McCollum, B., Meisels, A., Petrovic, S. ve Qu, R. (2005). "A graph based hyper-heuristic for educational timetabling problems". European Journal of Operation Research, 176(1), 177-199.

Burke, E.K. ve Petrovic, S. (2002). "Recent research directions in automated timetabling". European Journal of Operational Research, 140(2), 266-280.

Burke, E.K. ve Newall, J.P. (2002). "Enhancing Timetable Solutions with Local Search Methods", PATAT 2002, Proceedings of the 4th international conference on the Practice and Theory of Automated Timetabling, 336-347, Gent, Belçika.

Burke, E. K., Elliman, D.G., Ford, P.H. ve Weare, R. F. (1996). "Examination timetabling in British universities: a survey"., In: E. K. Burke and P. Ross (eds). Practice and Theory of Automated Timetabling: Selected papers from the first international conference, Volume 1153 of Lecture notes in computer science,76-90, Berlin.

Carrasco, M.P. ve Pato, M.V. (2004). "A Comparison of discrete and continuous neural network approaches to solve the class/teacher timetabling problem". European Journal of Operational Research, 153(1), 65-79.

Carter, M.V. ve Laporte, G. (1997). "Recent developments in practical course scheduling". 2nd Conference on Practise and Theory of Automated Timetabling II, 20-22, Kanada.

Cherny, V. (1985). "Thermodynamical approach to traveling salesman problem: an efficient simulation algorithm". Journal of Optimization Theory and Applications, 45(1), 41-51.

Chaudhuri, A. ve Kajal, D. (2010). "Fuzzy genetic heuristic for university course timetable problem". International Journal Advance Soft Computation Applications, 2(1), 100-123.

Dandashi, A. ve Al – Mouhamed, M. (2010). "Graph coloring for class scheduling". 8th ACS/IEEE International Conference on Computer Systems and Applications, Tunus.

Dasgupta, P. ve Khazanchi, D. (2005). "Adaptive decision support for academic course scheduling using intelligent software agents". International Journal of Technology in Teaching and Learning, 1(2), 63-78.

Ferland, J.A. ve Roy, S. (1985). "Timetabling problem for university as assignment of activities to resources". Computers and Operation Research, 12, 2, 207-218.

Geem, Z.W., Kim, J.H. ve Loganathan, G.V. (2001). "A new heuristic optimization algorithm: harmony search". Simulation, 76(2), 60-68.

Glover, F. (1989). "Tabu search part 1". ORSA Journal on Computing, 1(3), 190-206.

Glover, F. (1990). "Tabu search part 2". ORSA Journal on Computing, 2(1), 4-32.

Holland, J.H. (1975). "Adaptation in Natural and Artificial Systems". Ann Arbor, USA, University of Michigan Press.

Ismayilova, N.A., Sagir, M. ve Gasimov, R.N. (2007). "A multiobjective faculty-course-time slot assignment problem with preferences". Mathematical and Computer Modelling, 46(7-8), 1017-1029.

Jat, S.N. ve Yang, S. (2011). "Genetic algorithms with guided and local search strategies for university course timetabling". IEEE Transactions on Systems, Man, and Cybernetics- Part C: Applications and Reviews, 4(1), 93-106.

Kahara, M. ve Kendalla, G. (2010). "The examination timetabling problem at Universiti Malaysia Pahang: Comparison of a constructive heuristic with an existing software solution". European Journal of Operational Research, 207(2), 557-565.

Kennedy, J. ve Eberhart, R.C. (1995). "Particle Swarm Applied Mathematics". Proc. of the IEEE Int. Conference on Neural Networks, 4, 1942-1948.

Kirkpatrick, S., Gelatt, C.D. ve Vecchi, M.P. (1983). "Optimization by simulated annealing". Science, 220(4598), 671-680.

Lai, L., Hsueh, N., Huang, L. ve Chen, T. (2006). "An Artificial Intelligence Approach to CourseTimetabling". Proceedings of the 18th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'06), 389-396, USA.

Lee, S. ve Sehniederjans, M. (1983). "Multi creteria assignment problem: A goal programming approach". Interfaces, 13(4), 75-81.

Mayer, A., Nothegger, C., Chwatal, A. ve Raidl, G. (2008). "Solving the post enrolment course timetabling problem by ant colony optimization". 7th International

Conference on the Practice and Theory of Automated Timetabling, Montreal, Kanada.

Merlot, L.T.G., Boland, N., Hughes, B.D. ve Stuckey, P.J. (2003). "A Hybrid Algorithm for the Examination Timetabling Problem". Practice and Theory of Automated Timetabling: Selected Papers from the 4^{th} International Conference, Gent, Belgium, Springer Lecture Notes in Computer Science, 2740, 207-231.

Öztürk, Z.K. (2010). Eğitimsel Zaman Çizelgeleme Problemleri için Çözüm Yaklaşımları ve Web Tabanlı Bir Karar Destek Sistemi Önerisi. Doktora Tezi, Anadolu Üniversitesi, Türkiye.

Schaerf, A. (1999). "A survey of automated timetabling". Artificial Intelligent Rewiew, 13, 87-127.

Schaerf, A. ve Gaspero, L. (2001). "Local search tecniques for educational timetabling problems". 6^{th} International Symposium on Operational Research (SOR-01), Preddvor, Slovenya.

Shatnawi S., Al-Rababah, K. ve Bani-Ismail, B. (2010). "Applying a novel clustering technique based on fp-tree to university timetabling problem: A case study". International Conference on Computer Engineering and Systems, Cairo, Egypt.

Shiau, D.F. (2011). "A hybrid particle swarm optimization for a university course scheduling problem with flexible preferences". Expert Systems Applications, 38(1), 235-248.

Van Den Broek, J. ve Hurkens, C. (2012). "An IP-based heuristic for the post enrolment course timetabling problem of the ITC2007". Annals of Operational Research, 194(1), 439-454.

Wahid, J. ve Hussin, N.M. (2013.) "Harmony great deluge for solving cirriculum based course timetabling problem". 3^{rd} International Conference on System Engineering and Technology, Malezya.

Yu, E. ve Sung, K.S. (2002). “A genetic algorithm for a university weekly courses timetabling problem”. International Transactions in Operational Research, 9, 703-717.

Zhang, L. ve Lau, S. (2005). “Constructing university timetable using constraint satisfaction programming approach”. International Conference on Computational Intelligence for Modeling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce(CIMCA-IAWTIC’05), 28-30, Washington DC, USA.

EK-1: MATEMATİKSEL MODEL İLE ELDE EDİLEN OTURUM DÜZENİ

1. oturum kitap ve dersleri			
Kitap 6	Kitap 9	Kitap 11	
EDB403U	BIL101U	EDB401U	
FOT103U	ISL105U	ILH2005	
ILH1005	ISL107U	ILT107U	
ILH2002	ISL293U	ISL201U	
MUH201U	SOS203U	MAT103U	
SOS113U	TAR113U	TDE403U	
SOS321U	TUR201U	ULI201U	
2. oturum kitap ve dersleri			
Kitap 2	Kitap 3	Kitap 4	
ARA1001	EDB405U	IKT103U	
EDB407U	ILH1003	ILH1002	
FIN201U	LOJ101U	ING101U	
FOT101U	PSI201U	MLY203U	
ILH1001	TAR119U	SOS305U	
MUH101U	ULI407U	SOS309U	
TRZ205U			
3. oturum kitap ve dersleri			
Kitap 1	Kitap 5	Kitap 8	Kitap 12
ING201U	CEK101U	EDB103U	IKT203U
IST203U	EDB105U	MUH103U	ILH1004
ISL301U	FOT105U	SOS101U	ILH2001
MUH301U	ILH2003	SOS205U	ILH2004
SOS315U	SOS103U	SOS313U	ING301U
SOS317U	TAR201U	TDE401U	MAT105U
ULI305U	TDE101U		SOS319U
4. oturum kitap ve dersleri			
Kitap 7	Kitap 10	Kitap 13	
EDB101U	HUK101U	ARA2001	
EDB107U	HUK209U	ISL403U	
EST101U	ISL401U	MLY201U	
FEL207U	ISL405U	PSI103U	
IKT201U	SIY201U	PZL193U	
IST201U	SOS105U	SOS201U	
TDE103U	SOS311U		

EK-2: ALGORİTMA İLE ELDE EDİLEN OTURUM DÜZENİ

1. oturum kitap ve dersleri									
10001	10002	10003	10004	10005	10006	10007	10008	10009	100010
MAT103U	MAT103U	FEL207U	İŞL205U	FEL207U	TİC205U	MLY201U	MAİ201U	MLY201U	İKT303U
BİL101U	ARA2001	MLY201U	İKT311U	BİL101U	BİL101U	İKT311U	İŞL205U	TİC205U	İŞL203U
İNG101U	BİL101U	ÇEK403U	ULİ407U	SOS103U	İNG101U	EDB203U	EDB305U	KOİ409U	İNÖ407U
FOT105U	SOS113U	SOS101U	YBS203U	SOS101U	SOS113U	ULİ405U	EDB201U	İKT311U	SYT203U
SOS113U	SOS101U	İŞL405U	SOS305U	EDB101U	SOS101U	EDB201U	İLT203U	EDB401U	LBV209U
SOS101U	İLH1002	SOS205U	İNG301U	SOS205U	LOJ101U	ULİ407U	MEİ303U	ÇEK403U	SOS315U
İNG301U	MUH101U	İNG301U	SOS313U	SOS305U	MUH201U	İNG301U	YBS203U	MLY403U	SOS313U
HUK209U	İLH2004	HUK301U	MAT105U	SOS203U	HUK209U	İKT307U	YBS303U	KOİ401U	İNG201U
SOS207U	İLH1004	FİN201U	ULİ401U	İNG301U	İKT307U	İKT403U	TDE305U	EDB407U	LBV203U
EST101U	MAT105U	PZL103U	SOS307U	MAT105U	PZL103U	ULİ401U			
2. oturum kitap ve dersleri									
20001	20002	20003	20004	20005	20006	20007	20008	20009	
TAR119U	İŞL105U	ULİ305U	İKT201U	İLT213U	PZL193U	EDB205U	EDB403U	LBV205U	
ULİ305U	LOJ207U	PSİ201U	SOS301U	HUK101U	İKT401U	LOJ207U	ULİ305U	MUH303U	
HUK101U	İŞL301U	MLY203U	MLY203U	HİT201U	İŞL105U	EDB105U	İŞL209U	İNÖ307U	
MLY203U	İŞL401U	İLH2002	TÜR201U	SOS111U	ÇEK401U	EDB103U	YBS307U	KOİ405U	
MUH103U	HUK101U	SİY201U	SOS319U	İLT107U	PZL207U	EDB301U	FOT101U	LBV211U	
SİY301U	İLT107U	İLH1003	İST201U	MUH103U	HUK101U	MUH103U	İLT107U	EMY205U	
SİY201U	MUH103U	İLH2005	KYT401U	İLT207U	HUK401U	FİN401U	PSİ103U		
FRA301U	TÜR201U	TÜR201U	SOS309U	MEİ101U	TÜR201U	TDE203U	YBS205U		
TÜR201U	İŞL201U	SOS319U	SOS317U	PZL303U	SOS319U	TDE301U	TDE401U		
ULİ301U	TRZ205U	ARA1001	İKT305U	TÜR201U	ÇMH201U				
3. oturum kitap ve dersleri									
30001	30002	30003	30004	30005	30006	30007	30008	30009	
İST205U	EDB107U	İKT203U	İKT103U	İKT103U	İŞL403U	EDB107U	KYT201U	LBV207U	
İKT103U	İŞL303U	PZL305U	İKT101U	TİC203U	İKT203U	TDE101U	ULİ403U	KOİ407U	
TİC203U	YBS309U	İKT103U	TAR201U	TAR201U	MLY301U	EDB303U	İKT103U	İNÖ405U	
İLH1005	KYT201U	İKT101U	HUK221U	MEİ105U	İŞL305U	SOS201U	HUK211U	FOT103U	
SOS201U	TAR201U	HİT203U	MEİ105U	İŞL107U	İŞL293U	SOS311U	ÇEK405U	SOS321U	
TAR201U	TDE205U	TAR201U	TAR113U	MEİ103U	MUH301U	SOS303U	HUK403U	TDE403U	
İLH1001	TDE103U	İLT301U	İŞL107U	İLT303U	İKT309U	TDE303U	SOS105U	EDB405U	
İLH2003	İST203U	İŞL107U	İLH2003	FİN205U	İKT405U	İST203U	İŞL293U	LBV201U	
SOS105U	TDE201U	ULİ201U	ULİ201U	SYT201U	ULİ303U	TDE103U	ÇEK101U		
İST203U	YBS201U	İLT209U	İLH2001	MUH301U	SİY303U	TDE201U	SİY303U		

EK-3: GAMS KODU

```
$title oturum

option reslim = 10000;

option iterlim = 1000000000;

SETS

o oturumlar /1*4/

k kitaplar /1*13/

i dersler /1*88/

s öğrenciler/1*30 /;

VARIABLES

x(i,k)      i. ders k. kitap türüne atanırsa

z(i,o)      i. ders o. oturuma atanırsa

y(k,o)      k. kitap türü o. oturuma atanırsa

q(k)        k. kitap türü kullanıldıysa

a           amaç;

BINARY VARIABLES

x(i,k)

z(i,o)

y(k,o)

q(k);

TABLE P(s,i) öğrenci ders matrisi

$ondelim

$include “filename.csv”

$offdelim;

PARAMETER

NCs(s) öğrencilerin aldıkları ders sayıları /

$ondelim

$include “filename.txt”

$offdelim /;
```

Scalar M;

EQUATIONS

Amaç amac fonksiyonu

Kisit1 bir kitap türünde en fazla 10 ders yer alabilir.

Kisit12 bir kitap türünde en az bir ders yer almalıdır.

Kisit2 bir ders sadece bir oturuma atanabilir.

Kisit3 bir kitap türü sadece bir oturuma atanabilir.

Kisit4 bir ders bir kitap türüne mutlaka yerleştirilmelidir, aynı oturumda olmak koşuluyla bir ders birden fazla kitap türünde yer alabilir.

Kisit5 bir öğrenci bir oturumda en fazla 5 ders alabilir.

Kisit6 kitap türünün oturuma atanması ve kitap türünün oluşturulması ilişki kısıtı.

Kisit61 kitap türünün oturuma atanması ve kitap türünün oluşturulması ilişki kısıtı.

Kisit7 dersin kitap türüne atanması ve kitap türünün oluşturulması ilişki kısıtı.

Kisit71 dersin kitap türüne atanması ve kitap türünün oluşturulması ilişki kısıtı.

Kisit8 öğrencinin tüm derslerinin oturumlara yerleştirilmelidir.

Kisit9 bir ders bir kitap türüne atandığında, dersin ve kitap türünün oturumu aynı olmalıdır.

Kisit91 bir ders bir kitap türüne atandığında, dersin ve kitap türünün oturumu aynı olmalıdır ;

amac.. a =e= sum(k,q(k));

kisit1(k).. sum(i, x(i,k)) =l=10;

kisit12(k).. sum(i, x(i,k)) =g=1;

kisit2(i).. sum(o, z(i,o)) =e= 1;

kisit3(k).. sum(o, y(k,o)) =e= 1;

kisit4(i).. sum(k, x(i,k)) =g= 1;

kisit5(s,o)..sum(i, P(s,i)*z(i,o))=l= 5;

kisit6(k)..sum((o), y(k,o)) =l= M* q(k);

kisit61(k)..sum((o), y(k,o)) =g= q(k);

kisit7(k)..sum((i), x(i,k)) =g= q(k) ;

kisit71(k)..sum((i), x(i,k)) =l= M* q(k);

kisit8(s)..NCs(s) =e= sum ((i,o),P(s,i)* z(i,o)) ;

Kisit9(i,o,k)..(z(i,o)-y(k,o))=l= 1-(x(i,k));

```
Kisit91(i,o,k)..z(i,o)-y(k,o)=g= -(1-(x(i,k)));

MODEL oturum /all/;

SOLVE oturum using mip minimizing a;
```

EK-4: JAVA KODU

```
package config;

import com.lexicalscope.jewel.cli.Option;

import java.io.File;

public interface Parameters {


    /**
     * Islem yapilacak dosya
     *
     * @return file
     */
    @Option(shortName = "f")
    File getFile();


    /**
     * Oturum Sayisi
     *
     * @return Integer
     */
    @Option(shortName = "sc")
    Integer getSessionCount();

    /**
     * Adayin bir oturumda alabileceği ust ders siniri
     *
     * @return Integer
     */
    @Option(shortName = "cl")
    Integer getCourseLimit();


    /**
     * Bir kitapcikta olabilecek maksimum ders sayisi
     *
     * @return Integer
     */
    @Option(shortName = "bll")
    Integer getBookletCourseLimit();

}
```

```java
package model;

import java.util.LinkedHashSet;
import java.util.Set;

public class BookletModel {
    private SessionModel session;
    private Set<CourseModel> courses;
    private String bookletName;

    public BookletModel(SessionModel session,
LinkedHashSet<CourseModel> courses) {
        this.session = session;
        this.courses = courses;
    }

    public SessionModel getSession() {
        return session;
    }

    public void setSession(SessionModel session) {
        this.session = session;
    }

    public Set<CourseModel> getCourses() {
        return courses;
    }

    public void setCourses(LinkedHashSet<CourseModel> courses) {
        this.courses = courses;
    }

    public String getBookletName() {
        return bookletName;
    }

    public void setBookletName(String bookletName) {
        this.bookletName = bookletName;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;

        BookletModel that = (BookletModel) o;

        if (session != null ? !session.equals(that.session) :
that.session != null) return false;
        return courses != null ? courses.equals(that.courses) :
that.courses == null;
    }
    @Override
    public int hashCode() {
        int result = session != null ? session.hashCode() : 0;
        result = 31 * result + (courses != null ? courses.hashCode() :
0);
        return result;
    }
}
```

```java
package model;

import java.util.HashSet;
import java.util.Set;

public class CourseModel {

    private String name;
    private Long weight;
    private Set<StudentModel> students;

    public CourseModel(String courseName) {
        this.name = courseName;
        this.weight = 0L;
        this.students = new HashSet<>();
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public Long getWeight() {
        return weight;
    }

    public boolean addStudent(StudentModel studentModel) {
        boolean added = this.students.add(studentModel);
        calculateWeight();
        return added;
    }

    public boolean removeStudent(StudentModel studentModel) {
        boolean removed = this.students.remove(studentModel);
        calculateWeight();
        return removed;
    }

    private void calculateWeight() {
        this.weight = (long) students.size();
    }
}
```

```java
package model;

import java.util.LinkedHashSet;

public class SessionModel {

    private Integer sessionId;
    private LinkedHashSet<CourseModel> courses;

public SessionModel(Integer sessionId) {
```

```
        this.sessionId = sessionId;
        this.courses = new LinkedHashSet<>();
    }

    public Integer getSessionId() {
        return sessionId;
    }

    public void setSessionId(Integer sessionId) {
        this.sessionId = sessionId;
    }

    public LinkedHashSet<CourseModel> getCourses() {
        return courses;
    }

    public boolean addCourse(CourseModel courseModel) {
        return this.courses.add(courseModel);
    }

    public boolean removeCourse(CourseModel courseModel) {
        return this.courses.remove(courseModel);
    }

    public boolean courseContains(CourseModel courseModel) {
        return this.courses.contains(courseModel);
    }
}
```

```
package model;

public class StudentCourse {

    private String tcNo;
    private String course;

    public StudentCourse(String tcNo, String course) {
        this.tcNo = tcNo;
        this.course = course;
    }

    public String getTcNo() {
        return tcNo;
    }

    public void setTcNo(String tcNo) {
        this.tcNo = tcNo;
    }

    public String getCourse() {
        return course;
    }

    public void setCourse(String course) {
        this.course = course;
    }
}
```

```java
package model;

import org.apache.commons.collections4.SetUtils;

import java.util.LinkedHashSet;

public class StudentModel {

    private String tcNo;
    private LinkedHashSet<CourseModel> courses;
    private Integer sumOfCourseWeight;

    public StudentModel(String tcNo) {
        this.tcNo = tcNo;
        this.courses = new LinkedHashSet<>();
    }

    public StudentModel(StudentModel studentModel,
LinkedHashSet<CourseModel> courses) {
        this.tcNo = studentModel.getTcNo();
        this.courses = courses;
        getSumOfCourseWeight();
    }

    public String getTcNo() {
        return tcNo;
    }

    public LinkedHashSet<CourseModel> getCourses() {
        return courses;
    }

    public void setCourses(LinkedHashSet<CourseModel> course) {
        this.courses = course;
    }

    public Integer getCourseSize() {
        return this.courses.size();
    }


    /**
     * Ogrencinin aldigi tum derslerin agirliklarini toplar
sumOfCourseWeight degiskenine atar ve dondurur.
     *
     * @return Integer
     */
    public Integer getSumOfCourseWeight() {
        this.sumOfCourseWeight =
Math.toIntExact(courses.stream().mapToLong(CourseModel::getWeight).sum
());
        return this.sumOfCourseWeight;
    }

    /**
     * Bir adayin bir oturumda alabilecegi maksimum ders limiti olan
5'in geçilip geçilmediğinin kontrolunu yapar.
     *
     * @param sessionModel kontrol edilecek oturum
```

```
     * @param limit bir ogrencinin bir oturumda alabilecegi maksimum
ders limiti
     * @return Boolen
     */
    public boolean courseLimitIsExceeded(SessionModel sessionModel,
int limit) {
        return SetUtils.intersection(sessionModel.getCourses(),
this.getCourses()).size() > limit;
    }
}
```

```
package service;

import model.BookletModel;


public class BookletMatchingModel {

    private Integer matchingCount;

    private BookletModel bookletModel;

    public BookletMatchingModel(long matchingCount, BookletModel
bookletModel) {
        this.matchingCount = Math.toIntExact(matchingCount);
        this.bookletModel = bookletModel;
    }

    public Integer getMatchingCount() {
        return matchingCount;
    }

    public void setMatchingCount(Integer matchingCount) {
        this.matchingCount = matchingCount;
    }

    public BookletModel getBookletModel() {
        return bookletModel;
    }

    public void setBookletModel(BookletModel bookletModel) {
        this.bookletModel = bookletModel;
    }
}
```

```
package service;

import config.Parameters;
import model.BookletModel;
import model.CourseModel;
import model.SessionModel;
import model.StudentModel;
```

```
import java.util.LinkedHashSet;
import java.util.List;
import java.util.Optional;
import java.util.Set;
import java.util.concurrent.atomic.AtomicInteger;

import static java.util.Comparator.comparing;
import static java.util.stream.Collectors.*;


public class BookletService {

    private Parameters parameters;
    private List<SessionModel> sessions;
    private List<StudentModel> student;
    private LinkedHashSet<BookletModel> booklets = new
LinkedHashSet<>();

    public BookletService(Parameters parameters, List<StudentModel>
student, List<SessionModel> sessions) {
        this.parameters = parameters;
        this.sessions = sessions;
        this.student = student;
    }


    /**
     * Kitapciklari olustur
     */
    public void createBooklets() {
        for (SessionModel session : sessions) {
            List<String> courseNameOfSessions =
session.getCourses().stream().map(CourseModel::getName).collect(toList
());

            List<StudentModel> studentWithCoursesInSession =
getStudentWithCourseInSession(courseNameOfSessions);

            for (StudentModel studentModel :
studentWithCoursesInSession) {

                BookletModel booklet = new BookletModel(session,
studentModel.getCourses());
                boolean sameBookletIsCreated = booklets.stream()
                        .filter(b ->
b.getSession().getSessionId().equals(session.getSessionId()))
                        .anyMatch(bm -> bm.equals(booklet));

                //Adayin tum derslerinin oldugu ayni kitapcik varmi ?
                if (!sameBookletIsCreated) {

                    List<BookletMatchingModel> bookletListCanUsed =
getBookletCanUsed(session, studentModel.getCourses());

                    // En az 1 der eslesen kitapcik varmi ?
                    Optional<BookletMatchingModel> bookletCanUsed =
bookletListCanUsed.stream()
                            .filter(kitapEslesmeModel ->
kitapEslesmeModel.getMatchingCount() > 0)
```

```
                            .findFirst();

                    if (bookletCanUsed.isPresent()) {
                        addCourseToBooklet(session,
bookletCanUsed.get().getBookletModel(), studentModel.getCourses());
                    } else {
                        bookletCanUsed =
bookletListCanUsed.stream().findFirst();

                        // En az 1 ders eslesen kitapcik yoksa
sigabildigi baska kitap varmi ?
                        if (bookletCanUsed.isPresent()) {
                            addCourseToBooklet(session,
bookletCanUsed.get().getBookletModel(), studentModel.getCourses());
                        } else {
                            booklets.add(booklet);
                        }
                    }
                }
            }
        }
        sortAndSetNameAllBooklet();
    }


    public Set<BookletModel> getBooklets() {
        return booklets;
    }

    /**
     * Olusturulan kitapciklara isim ataması yapar ve kitap içinde yer
alan derslerin sayilarilarina gore sıralama yapar.
     */
    private void sortAndSetNameAllBooklet() {
        booklets = booklets.stream().sorted(comparing(bookletModel ->
((BookletModel)
bookletModel).getCourses().size()).reversed()).collect(toCollection(Li
nkedHashSet::new));
        booklets.stream()
                .collect(groupingBy(o ->
o.getSession().getSessionId()))
                .forEach((integer, bookletModels) -> {
                    AtomicInteger i = new AtomicInteger(1);
                    bookletModels.forEach(bookletModel -> {

bookletModel.setBookletName(String.valueOf(bookletModel.getSession().g
etSessionId() * 1000 + i.get()));
                        i.getAndIncrement();
                    });
                });
    }

    /**
     * Verilen dersi kitapciga ekler
     *
     * @param sessionModel oturum
     * @param bookletCanUsed kullanilabilir kitapcik
     * @param courseList Kitapciga eklenmek istenen dersler
     */
```

```
    private void addCourseToBooklet(SessionModel sessionModel,
BookletModel bookletCanUsed, Set<CourseModel> courseList) {

        Set<BookletModel> bookletInSession = booklets.stream()
                .filter(booklet ->
booklet.getSession().getSessionId().equals(sessionModel.getSessionId()
)).collect(toSet());
        for (BookletModel booklet : bookletInSession) {

            if (booklet.equals(bookletCanUsed)) {
                Set<CourseModel> eklenecekDersler =
courseList.stream()
                        .filter(s ->
!bookletCanUsed.getCourses().contains(s))
                        .collect(toSet());
                booklet.getCourses().addAll(eklenecekDersler);
            }
        }
    }

    /**
     * Bu oturumda ders alan tum ogrencileri ve bu oturumdaki
derslerini filtreler
     *
     * @param courseNameOfSessions oturumdaki tum derslerin isimleri
     * @return List<StudentModel>
     */
    public List<StudentModel>
getStudentWithCourseInSession(List<String> courseNameOfSessions) {
        return student.stream()
                .filter(studentModel ->
                        studentModel
                                .getCourses()
                                .stream()
                                .anyMatch(courseModel ->

courseNameOfSessions.contains(courseModel.getName()))
                ).map(studentModel -> {
                    LinkedHashSet<CourseModel> courseOfSession =
studentModel
                            .getCourses()
                            .stream()
                            .filter(courseModel ->
courseNameOfSessions.contains(courseModel.getName()))

.collect(toCollection(LinkedHashSet::new));
                    return new StudentModel(studentModel,
courseOfSession);
                })
                .sorted(comparing(StudentModel::getCourseSize)
                        .reversed()
                        .thenComparing(

comparing(StudentModel::getSumOfCourseWeight).reversed()
                        )
                ).collect(toList());
    }


    /**
```

```
     * Ilgili oturumdaki kitapciklarin eslesme sayilarini veren model
listesi dondurur
     *
     * @param sessionModel islem yapilacak oturum
     * @param courseModel  Yerlestirilecek dersler
     * @return List<BookletMatchingModel> eslesme model listesi
     */
    public List<BookletMatchingModel> getBookletCanUsed(SessionModel
sessionModel, LinkedHashSet<CourseModel> courseModel) {
        return booklets.stream()
                .filter(bookletModel ->
bookletModel.getSession().getSessionId().equals(sessionModel.getSessio
nId()))
                .map(oturumKitap -> new
BookletMatchingModel(courseModel.stream().filter(s ->
oturumKitap.getCourses().contains(s)).count(), oturumKitap))
                .filter(bookletMatchingModel -> {
                    long kitabaYerlestirilecekDersSayisi =
courseModel.size() - bookletMatchingModel.getMatchingCount();
                    long kitapdakiToplamDersSayisi =
bookletMatchingModel.getBookletModel().getCourses().size() +
kitabaYerlestirilecekDersSayisi;
                    return kitapdakiToplamDersSayisi <=
parameters.getBookletCourseLimit();
                })

.sorted(comparing(BookletMatchingModel::getMatchingCount)
                        .reversed()
                        .thenComparing(kitapEslesmeModel ->
kitapEslesmeModel.getBookletModel().getCourses().size()))
                .collect(toList());
    }

}
```

```
package service;

import model.StudentCourse;
import org.apache.poi.ss.usermodel.Cell;
import org.apache.poi.ss.usermodel.Row;
import org.apache.poi.ss.usermodel.Sheet;
import org.apache.poi.ss.usermodel.Workbook;
import org.apache.poi.xssf.usermodel.XSSFWorkbook;

import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;
import java.util.logging.Logger;

public class DataImportService {

    private final static Logger logger =
Logger.getLogger(DataImportService.class.getName());
```

```
    private List<StudentCourse> studentCourseList =new ArrayList<>();
    private File file;


    public DataImportService(File file) {
        this.file = file;
    }

    public List<StudentCourse> getAllStudentCourse() throws
IOException {

        FileInputStream excelFile = new FileInputStream(file);
        Workbook workbook = new XSSFWorkbook(excelFile);
        Sheet datatypeSheet = workbook.getSheetAt(0);
        Iterator<Row> iterator = datatypeSheet.iterator();
        iterator.next();
        while (iterator.hasNext()) {

            Row currentRow = iterator.next();
            Iterator<Cell> cellIterator = currentRow.iterator();

            String tcNo = cellIterator.next().getStringCellValue();
            String course = cellIterator.next().getStringCellValue();

            studentCourseList.add(new StudentCourse(tcNo, course));

            logger.info("Added : " + tcNo + "\t" + course);
        }

        return studentCourseList;

    }
}
```

```
package service;

import config.Parameters;
import model.CourseModel;
import model.SessionModel;
import model.StudentCourse;
import model.StudentModel;

import java.util.ArrayList;
import java.util.LinkedHashSet;
import java.util.List;
import java.util.stream.Collectors;
import java.util.stream.IntStream;

import static java.util.Comparator.comparing;
import static java.util.stream.Collectors.toList;
import static java.util.stream.Collectors.toSet;

public class SessionService {
```

```
    private Parameters parameters;

    private List<CourseModel> courses = new ArrayList<>();
    private List<StudentModel> students = new ArrayList<>();
    private List<SessionModel> sessions = new ArrayList<>();

    private List<StudentCourse> studentCourses;

    public SessionService(List<StudentCourse> studentCourses,
Parameters parameters) {
        this.studentCourses = studentCourses;
        this.parameters = parameters;
    }


    public void start() throws Exception {
        initialize();
        extractAndSortStudentAndCourse(studentCourses);
        assignCourseToSession();
    }

    private void initialize() {
        this.sessions = IntStream.rangeClosed(1,
parameters.getSessionCount())
                .mapToObj(SessionModel::new)
                .collect(toList());
    }

    /**
     * Dersleri oturumlara atar.
     *
     * @throws Exception
     */
    private void assignCourseToSession() throws Exception {
        for (StudentModel student : students) {

            for (CourseModel courseModel : student.getCourses()) {

                List<Integer> excludeSession = new ArrayList<>();
                Boolean courseIsDone = false;

                while (!courseIsDone) {

                    if (excludeSession.size() == sessions.size()) {
                        throw new Exception(courseModel.getName() + "
Kodlu Ders hicbir oturuma atanamiyor !");
                    }

                    SessionModel oturum = sessions
                            .stream()
                            .filter(o ->
!excludeSession.contains(o.getSessionId()))
                            .min(comparing(integerSetEntry ->
integerSetEntry.getCourses().size())).get();

                    courseIsDone = sessions.stream()
                            .flatMap(sessionModel ->
sessionModel.getCourses().stream().map(CourseModel::getName))
                            .collect(toSet())
                            .contains(courseModel.getName());
```

```
                    if (!courseIsDone) {
                        //Ders herhangi bir oturumda yok

                        if (!addCourseToSession(oturum.getSessionId(),
courseModel)) {
                            throw new Exception(courseModel.getName()
+ " Kodlu Ders " + oturum.getSessionId() + " Nolu oturuma
eklenemedi");
                        }

                        courseIsDone = true;
                        if (sessionCourseLimitIsExceededCheck()) {
                            // Oturuma atanamiyorsa haric oturumlara
eklenir

                            if
(!removeCourseToSession(oturum.getSessionId(), courseModel)) {
                                throw new
Exception(courseModel.getName() + " Kodlu Ders " +
oturum.getSessionId() + " Nolu oturumdan Silinemedi");
                            }

                            excludeSession.add(oturum.getSessionId());
                            courseIsDone = false;
                        }
                    }
                }
            }
        }
    }


    /**
     * Bir oturuma ders ekler.
     *
     * @param sessionId   oturum no
     * @param courseModel ders
     * @return Boolen
     */
    private boolean addCourseToSession(Integer sessionId, CourseModel
courseModel) {
        for (SessionModel session : sessions) {
            if (session.getSessionId().equals(sessionId)) {
                return session.addCourse(courseModel);
            }
        }
        return false;
    }

    /**
     * Bir oturumdan ders siler.
     *
     * @param sessionId   oturum no
     * @param courseModel ders
     * @return Boolen
     */
    private boolean removeCourseToSession(Integer sessionId,
CourseModel courseModel) {
        for (SessionModel session : sessions) {
```

```
            if (session.getSessionId().equals(sessionId)) {
                return session.removeCourse(courseModel);
            }
        }
        return false;
    }

    /**
     * Tum oturumlardaki dersleri aday verileriyle karsilastirip, bir
oturumda 5'den fazla ders alan aday varmi kontrolunu yapar.
     *
     * @return Boolen
     */
    private boolean sessionCourseLimitIsExceededCheck() {
        for (SessionModel session : sessions) {
            for (StudentModel student : students) {
                boolean courseLimitIsExceeded =
student.courseLimitIsExceeded(session, parameters.getCourseLimit());
                if (courseLimitIsExceeded) {
                    return true;
                }
            }
        }
        return false;
    }


    /**
     * Ogrenci ve ders verilerini birbirlerine baglar ve öğrencileri
aldiklari ders sayilari ve agirlik toplamlarina gore siralar.
     * Ders agirliklarina gore dersleri kendi icinde siralar.
     *
     * @param studentCourses Ham ogrenci ders verisi
     */
    private void extractAndSortStudentAndCourse(List<StudentCourse>
studentCourses) {
        for (StudentCourse studentCourse : studentCourses) {
            CourseModel course =
ensureCreatedCourse(studentCourse.getCourse());
            StudentModel student =
ensureCreatedStudent(studentCourse.getTcNo());

            student.getCourses().add(course);
            course.addStudent(student);
        }
        students = students
                .stream()
                .peek(studentModel -> {
                    LinkedHashSet<CourseModel> sortedCourse =
studentModel
                            .getCourses()
                            .stream()
                            .sorted(comparing(courseModel ->
                                    ((CourseModel)
courseModel).getWeight())
                                    .reversed()

).collect(Collectors.toCollection(LinkedHashSet::new));

                    studentModel.setCourses(sortedCourse);
```

```
                }).sorted(comparing(StudentModel::getCourseSize)
                        .reversed()
                        .thenComparing(

comparing(StudentModel::getSumOfCourseWeight)
                                                .reversed())
                ).collect(toList());
    }


    /**
     * Ders ogrenci eslestirmesi sirasinda, dersin olusturuldugundan
emin olur.
     * @param courseName Ders adi
     * @return CourseModel
     */
    private CourseModel ensureCreatedCourse(String courseName) {

        return courses.stream()
                .filter(courseModel ->
courseModel.getName().equals(courseName))
                .findFirst()
                .orElseGet(() -> {
                    CourseModel newDers = new CourseModel(courseName);
                    courses.add(newDers);
                    return newDers;
                });
    }

    /**
     * Ders ogrenci eslestirmesi sirasinda, ogrencinin
olusturuldugundan emin olur.
     * @param tcNo ogrenci T.C. nosu
     * @return StudentModel
     */
    private StudentModel ensureCreatedStudent(String tcNo) {
        return students.stream()
                .filter(studentModel ->
studentModel.getTcNo().equals(tcNo))
                .findFirst()
                .orElseGet(() -> {
                    StudentModel newAday = new StudentModel(tcNo);
                    students.add(newAday);
                    return newAday;
                });
    }

    public List<CourseModel> getCourses() {
        return courses;
    }

    public List<StudentModel> getStudents() {
        return students;
    }

    public List<SessionModel> getSessions() {
        return sessions;
    }
}
```

```
import com.lexicalscope.jewel.cli.CliFactory;
import config.Parameters;
import model.BookletModel;
import model.CourseModel;
import model.SessionModel;
import model.StudentCourse;
import service.BookletService;
import service.DataImportService;
import service.SessionService;

import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
import java.util.stream.Collectors;

import static java.util.Comparator.comparing;

public class Main {

    public static void main(String[] args) throws Exception {

        File out = new File("./output.txt");
        Parameters parameters =
CliFactory.parseArguments(Parameters.class, args);

        DataImportService dataImportService = new
DataImportService(parameters.getFile());
        List<StudentCourse> allStudentCourse =
dataImportService.getAllStudentCourse();

        SessionService oturumService = new
SessionService(allStudentCourse, parameters);

        oturumService.start();
        List<SessionModel> sessions = oturumService.getSessions();

        writeToFile("-------- OTURUM DERSLERI --------", out);
        writeToFile("OTURUM NO \t DERSLER", out);
        for (SessionModel session : sessions) {
            String result = session.getSessionId() + "\t" +
session.getCourses().stream().map(CourseModel::getName).collect(Collec
tors.joining("\t"));
            writeToFile(result, out);
        }

        writeToFile("-------- OTURUM AGIRLIK TOPLAMI --------", out);
        writeToFile("OTURUM NO \t TOPLAM", out);
        for (SessionModel session : sessions) {
            writeToFile(session.getSessionId() + "\t" +
session.getCourses().stream().mapToLong(CourseModel::getWeight).sum(),
out);
        }


        BookletService bookletService = new BookletService(parameters,
oturumService.getStudents(), sessions);
        bookletService.createBooklets();
```

```
        List<BookletModel> booklets = new
ArrayList<>(bookletService.getBooklets());

        writeToFile("------- KITAPLAR VE DERSLER -------", out);
        writeToFile("OTURUM NO\tKITAP ADI\tDERSLER", out);

        List<BookletModel> sortedBookletModel = booklets
                .stream()
                .sorted(comparing(BookletModel::getBookletName))
                .collect(Collectors.toCollection(ArrayList::new));

        for (BookletModel bookletModel : sortedBookletModel) {
            writeToFile(bookletModel.getSession().getSessionId() +
"\t" + bookletModel.getBookletName() + "\t" +
bookletModel.getCourses().stream().map(CourseModel::getName).collect(C
ollectors.joining("\t")), out);
        }

    }

    private static void writeToFile(String text, File file) {
        System.out.println(text);
        try {
            FileWriter fw = new FileWriter(file, true);
            fw.write(text + "\n");
            fw.close();
        } catch (IOException ioe) {
            System.err.println("IOException: " + ioe.getMessage());
        }
    }
}
```