T.C.
MALTEPE ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

GRAF RENKLENDİRİLMESİ İLE DERS ZAMAN ÇİZELGESİ OLUŞTURULMASI

Hüseyin Fehmi Selim BAYRAKLI

Yüksek Lisans Tezi

Tez Danışmanı
Prof. Dr. Kemal KÖYMEN

İSTANBUL – 2008

T.C.
MALTEPE ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

GRAF RENKLENDİRİLMESİ İLE DERS ZAMAN ÇİZELGESİ OLUŞTURULMASI

YÜKSEK LİSANS TEZİ

Hüseyin Fehmi Selim BAYRAKLI

Tez Danışmanı
Prof. Dr. Kemal KÖYMEN

İSTANBUL – 2008

ÖZET

Yüksek Lisans Tezi, Graf Renklendirme Yöntemiyle Ders Zaman Çizelgesi Oluşturulması, T.C. Maltepe Üniversitesi, Fen Bilimleri Enstitüsü, Bilgisayar Mühendisliği Anabilim Dalı.

Bu tezde üniversitemizdeki haftalık ders programı oluşturma problemi göz önüne alınarak bu problemi bir uygulama yazılımı geliştirerek çözmek amaçlanmıştır.

Bilgisayar bilimleri literatüründe bu problem "Ders Zaman Çizelgesi Oluşturma Problemi (Course Timetabling Problem)" olarak adlandırılmış ve genetik algoritmalar, tamsayı programlama ve graf renklendirme gibi çeşitli yöntemlerle çözümler geliştirilmiş ve geliştirilmektedir.

Tezde varolan genel çözüm yöntemlerine kısaca değinildikten sonra graf renklendirme yöntemi üzerinde durulmuş ve problemimiz bu yönteme benzetilerek çözülmeye çalışılmıştır.

Bu tez 2008 yılında tamamlanmıştır ve 47 sayfadan oluşmaktadır.

Anahtar Kelimeler: Zaman çizelgesi, ders programı, graf renklendirme, ders zaman çizelgesi.

ABSTRACT

Master Thesis, Course Timetabling Based on Graph Coloring. T.C. Maltepe University, Graduate School of Natural and Applied Sciences, Department of Computer Engineering.

The aim of this thesis was to solve the course scheduling problem of our university by developing an application software based on a sequential method (e.g. graph coloring).

This problem has been named as “Course Timetabling Problem” in computer science literature. Many solutions have been developed by using a number of methods such as genetic algorithms, integer programming and graph coloring.

In this thesis general solution methods have been mentioned, and then the graph coloring method has been utilized to solve our problem.

This thesis has been completed in 2008 and consists of 47 pages.

Keywords: Timetabling, course timetabling, course scheduling, graph coloring.

TEŞEKKÜR

Bu tez konusunu seçmemde beni yönlendiren, destek ve yardımlarını esirgemeyen danışman hocam Maltepe Üniversitesi Rektörü sayın Prof. Dr. Kemal KÖYMEN'e, değerli bilgilerinden istifade ettiğim hocam sayın Prof. Dr. Şaban EREN'e, çalışmalarım sırasında beni yönlendiren ve gerekli kaynakların sağlanmasında yardımcı olan hocam sayın Yrd. Doç. Dr. Çiğdem ALABAŞ USLU'ya, fikirlerinden beni mahrum bırakmayan hocam sayın Öğr. Gör. Kadir ÇAMOĞLU'na, yardım ve desteklerini benden esirgemeyen çalışma arkadaşlarım ve hocalarım sayın Öğr. Gör. Fatih YÜCALAR, sayın Öğr. Gör. Erdal GÜVENOĞLU ve sayın Öğr. Gör. İzzet TAMER'e, çalışma arkadaşım sayın Ar. Gör. Gökhan ÇAM'a ve tez uygulamasını geliştirirken yardımını benden esirgemeyen değerli arkadaşım Şemseddin AKSOY başta olmak üzere tüm çalışma arkadaşlarıma, maddi ve manevi desteğini benden hiçbir zaman esirgemeyen çok değerli aileme ve çalışmalarım sırasında emeği geçen herkese teşekkürlerimi sunarım.

İÇİNDEKİLER

KISALTMALAR

Kısaltma	Türkçesi
MÜBİS	Maltepe Üniversitesi Bilişim Sistemi
DEPOS	Ders Programı Oluşturma Sihirbazı

ŞEKİLLER

TABLOLAR

1. GİRİŞ

Zaman çizelgesi problemleri nakliyat, iş yönetimi ve eğitim gibi alanlarda ortaya çıkmaktadır. Bu problemler kırk küsür seneden beri araştırma konusu olmuştur ve halen bu konu üzerindeki araştırmalar devam etmektedir. Çünkü problemler daha karmaşık hale gelmektedir ve yeni yöntemler bu problemlerin çözümünde daha iyi sonuçlar üretmektedir.

1.1 Zaman Çizelgesi Problemi nedir?

Zaman çizelgesi problemi çok kısıtı olan bir çeşit planlama (scheduling) problemidir. 1996'da Wren tarafından şu şekilde tanımlanmıştır:

"Kısıtları göz önünde bulundurarak mümkün olduğunca istenileni karşılayabilecek bir şekilde, verilen kaynakların belirli zaman aralıklarına atanması işlemidir." [1]

Genel bir zaman çizelgesi problemi belli bir sayıdaki faaliyetlerin (sınav, ders, toplantı vb.) kısıtlardan dolayı oluşan çakışmaları en aza indirerek limitli sayıdaki zaman aralıklarına atanmasını içerir. Farklı zaman çizelgesi problemlerinin farklı kısıtları vardır. Her bir probleme ait kısıtlar genelde iki kategoriye ayrılır: Katı (Hard) kısıtlar ve Zayıf (Soft) kısıtlar. Katı kısıtlar, şartlar ne olursa olsun bozulmadan yerine getirilmesi gerekmektedir. Katı kısıtlara örnek olarak şunu verebiliriz: "Aynı zaman diliminde iki veya daha fazla derse aynı öğrenci veya hoca giremez." [1]

Zayıf kısıtların karşılanıp karşılanmaması çok önemli değildir. Zaten gerçekte bir problemde bütün kısıtları karşılamak neredeyse imkânsızdır. Bu nedenle bu tür problemlerin genel optimum bir çözümü yoktur. Kısıtlara birkaç örnek daha verecek olursak: Aynı katılımcılara sahip iki faaliyetin ardışık olup olmadığı; Bir faaliyetin başka bir faaliyetten önce olması gerekliliği, bir faaliyet gerçekleşirken önceden belirtilmiş bazı faaliyetlerin gerçekleşmemesi gerekliliği söylenebilir.

1.2 Problemin Özellikleri ve Ayrıntılı Tanımı

Zaman çizelgesi problemleri karmaşıktır ve yapı olarak çeşitlidir. Formal tanımları için araştırmalar halen sürmektedir. Bu kısımda birçok durum için yeterli olacak bir tanım yapılacaktır. [5]

1.2.1 Genel Problem

Tanım: Zaman çizelgesi problemi dört parametreden oluşan bir problemdir. Bunlar; saatlerin sonlu kümesi, *T*; kaynakların sonlu kümesi, *R*; faaliyetlerin sonlu kümesi, *E*; kısıtların sonlu kümesi, *C*. Problem, kısıtları olabildiğince karşılayarak saat ve kaynakları oturumlara atamaktır. Bu tanımda bahsedilen parametreler aşağıda açıklanacaktır.

1) Saatler (Times)

- Bir *saat, T* saat kümesinin bir elemanıdır.
- Bir *saat boşluğu*, bir *saati* içerme ile sınırlı olan bir değişkendir.
- Saat boşluklarına bazen önceden atama yapılabilir (Daha önceden belirlenmiş sabit zamanlı faaliyet varsa).
- Bir kısıt, iki saat boşluğunun ard arda dolu olması gerektiğini belirtebilir veya bir saat boşluğu kümesi tüm haftaya dengeli bir şekilde dağıtılmış saatlerden oluşacağını belirtebilir.
- Hazırlanan bazı zaman çizelgeleri her hafta kullanılabilir (haftalık ders programı vb.). Bazıları sadece bir veya iki hafta kullanılabilir (sınav çizelgesi vb.).

Durumlar

- Bazen saat boşluklarına önceden atama yapılabilir (daha önceden belirlenmiş sabit zamanlı faaliyet varsa)
- Bir kısıt iki saat boşluğunun ard arda dolu olması gerektiğini belirtebilir veya bir saat boşluğu kümesi tüm haftaya dengeli bir şekilde dağıtılmış zamanlardan oluşacağını belirtebilir.
- Hazırlanan bazı zaman çizelgeleri her hafta kullanılabilir, bazıları sadece bir veya iki hafta kullanılabilir (ders programı çizelgesi, kongre takvimi vb.)

2) Kaynaklar (Resources)

Toplantılar hocaları, derslikleri ve öğrencileri kapsayabilir. Bütün bunlar kaynak olarak adlandırılır.

Tanımlar

- Bir kaynak, zaman çizelgesi problemindeki kaynak kümesinin bir elemanıdır.
- Kaynak boşluğu, bir kaynağı içerme ile sınırlı olan bir değişkendir.

Durumlar

- Genellikle kaynak boşluklarına önceden atama yapılır.
  Örnek: Zaman çizelgesi oluşturmanın temel kısıtı aynı vakitteki iki ayrı toplantıda aynı kaynaklar bulunamaz.

3) Toplantılar (Meetings)

Tanım

- Bir buluşma, saat ve kaynak boşlukları bütününün isimlendirilmiş halidir. Saat boşluklarına atanan saatler içerisinde kaynak boşluklarına atanan kaynaklar o toplantıda bir araya gelecektir.

Örnekler:

- Sınav zaman çizelgesi hazırlamada bir toplantı tek bir sınavı temsil eder ve bu sınav bir saat boşluğu, daha önceden ataması yapılmış öğrenciler ve bir veya daha fazla sınıf odası içerir.
- Okul zaman çizelgesinde bir toplantı, bir hafta boyu okutulacak bir dersi temsil eder ve az sayıda saat boşluğu, önceden ataması yapılmış bir öğrenci grubu, bir hoca boşluğu (genellikle önceden atanmış olur) ve bir sınıf içerir.

4) Kısıtlar

Zaman çizelgesi uygulayıcıları birçok organizasyonda farklı bir çok kısıtı ortaya çıkarmışlardır. Geniş kapsamlı bir kısıt listesini burada vermemiz mümkün değildir. Sonuçlara karşı kısıtları değerlendirirken kabul edilebilir tam sonuçlara "0" değerini ve sonuçları daha az kabuledilebilir durumlara geldikçe artan değerler atanması uygun olur.

Tanımlar

- S, bir zaman çizelgesi probleminin bütün çözümlerinin kümesi olsun. Katı bir kısıt, mutlaka mutlaka karşılanması gereken bir kısıttır. Her bir katı kısıtta ortak olan, ikili (binary) değerli bir fonksiyondur h: S → {0,1}. Her bir ω çözümü şu şekilde tanımlanır;

$$h(\omega) = \begin{cases} 1, \text{Eğer } \omega \text{ çözümü verilen kısıtı karşılamıyorsa} \\ 0, \text{Karşılıyorsa} \end{cases}$$

- Uygun olan bir çözüm, bütün katı kısıtları karşılayan S kümesindeki her bir ω'dir. Bütün durumlar için h(ω) = 0.
- S, bir zaman çizelgesi probleminin bütün çözümlerinin kümesi olsun. Bir zayıf (soft) kısıt, karşılanıp karşılanmaması çok önemli olmayan kısıttır. Her bir zayıf kısıtta ortak olan s: S → Z^+ fonksiyonudur. Yani, S kümesinde s(ω) için hangi ω çözümü küçükse o çözüm tercih edilir.

- S, bir zaman çizelgesi probleminin bütün çözümlerinin kümesi olsun. Kusur fonksiyonu (Badness function): b: S → Z^+, S kümesindeki bir ω çözümü için tek bir rakamlı toplu derece puanı veren fonksiyondur.
- Tamlık kısıtı (Completeness constraint): Her bir saat boşluğunun değer almasını gerektirir.
- Çakışmasız kısıt (No-clashes constraint): Her bir kaynağın aynı zaman içindeki iki farklı buluşmada olmamasını gerektirir.
- Müsait olma kısıtı (Availaiblity constraint): Belirli bir kaynağın sadece belirli zamanlarda mevcut olduğunu belirtir. Örneğin part-time çalışan bir hoca sadece perşembe veya cuma günleri müsait olabilir.

Örnek:

Verilen bir problem için katı kısıtlar h_1, h_2,..., h_n olsun ve zayıf ksıtlar da s_1, s_2, ..., s_m olsun. Alışılmış bir yaklaşım kusur fonksiyonunu seçmektir. Bu fonksiyon şu değerlerin ağırlıklandırılmış toplamıdır:

$$b(S) = \sum_{i=1}^{n} v_i h_i(S) + \sum_{j=1}^{m} \omega_j s_j(S)$$

v_i ve ω_i söz konusu kısıtın önemini belirten pozitif tamsayı olan ağırlık katsayısıdır. v_i, ω_i'den daha büyüktür.

Notlar

- Ders programı zaman çizelgesinde çakışmasız kısıt hocalar için katı kısıt fakat öğrenciler için zayıf kısıt olacaktır. Çünkü genelde her öğrencinin isteklerini karşılamak imkânsızdır.
- Diğer bazı kısıtlar şöyle olabilir; her bir hocanın bir gün içinde en az bir saati boş olacaktır. Her öğrencinin öğle yemeği saati olacaktır. Her bir gündeki iki ders arasındaki büyük boşluklar olabildiğince azaltılmalıdır.

Tezin geri kalanında Zaman çizelgesi oluşturma yöntemlerinden kısaca bahsedilecek ve bu yöntemlerden *Sıralı Yöntem* üzerinde durulacak. Bunun içinde ilk önce graf teorisinin ve graf renklendirme üzerinde durulacak. Daha sonra graf renklendirme ile ders programı zaman çizelgesi oluşturma problemi için genel bir örnek üzerinde durulacak. Son olarak Maltepe Üniversitesi Bilişim Sistemi'ne (MUBİS) ileride entegre edilecek, öğrenci işlerinin belirlediği kriterler doğrultusunda geliştirilen uygulama program yapısı anlatılacaktır.

2. ZAMAN ÇİZELGESİ OLUŞTURMA YÖNTEMLERİ

Zaman çizelgesi oluşturmada saat boşluklarına atanacak faaliyet sayısının çok olması ve çok kısıt olması olası çözüm kümesinin çok büyümesine neden olur. Zaman çizelgesini oluşturmak çok zor olabilir ve elle yapmaya kalkışıldığında çok fazla zaman ve iş gücü gerektirebilir. [1]

Zaman çizelgesi problemleri 40 küsür senedir bazı bilim dallarının araştırma konusu olmuştur (Yapay Zekâ, Yöneylem Araştırması). Literatürde tanımlanmış birçok çözüm yöntemi vardır. Bu yöntemler genel olarak dört ana başlık altında toplanabilir. Bunlar;

1. Sıralı (Sequential) Yöntemler
2. Sezgi Üstü (Metaheuristic) Yöntemler
3. Küme (Cluster) Yöntemler
4. Kısıt Tabanlı (Constraint Based) Yöntemler

2.1 Sıralı Yöntemler

Bu yöntemler faaliyetleri ortam sezgiselleri kullanarak sıralar ve sonra geçerli saat boşluklarına atarlar. Böylece hiç bir faaliyet birbiriyle çakışmaz. Sıralı yöntemlerde zaman çizelgesi problemleri genellikle graf olarak gösterilir. Faaliyetler düğüm, faaliyetler arası çakışma da ayrıt olarak gösterilir. Örneğin bazı öğrenciler iki faaliyete birden katılmak zorundadır. Düğümler arasında çakışmayı gösteren bir ayrıt vardır. Çakışmasız bir zaman çizelgesi oluşturulması bir graf renklendirme problemi gibi modellenebilir. Zaman çizelgesindeki her bir saat boşluğu graf renklendirme problemindeki bir renkte karşılık gelir ve grafın düğümleri de iki komşu düğümün aynı renkle renklenmeyeceği şekilde renklendirilir. [1]

Çakışmasız zaman çizelgesi oluşturmak için literatürde birçok graf renklendirme sezgiselleri bulunmaktadır. Bu sezgiseller faaliyetleri, atama zorluklarının tahminine göre sıralar. Sıkça kullanılan sezgiseller şunlardır;

- **Önce en büyük derece:** Diğer faaliyetlerle çok fazla çakışması olan faaliyetlerin atanması önce yapılır. Çakışması çok olan faaliyetler atanması en zor faaliyetlerdir yani katı kısıda sahip olan faaliyetlerdir. Bu nedenle en önce bu faaliyetlerin saat boşluklarına atanması gerekmektedir.
- **Önce en büyük ağırlıklı derece:** Eğer zaman çizelgesi oluşturmada derse katılacak öğrenci sayısı da dikkate alınıyorsa çakışmalar öğrenci sayısına göre ağırlıklandırılır.
- **Satürasyon derecesi:** Zaman çizelgesi oluşturmanın her adımında, o ana kadar oluşturulmuş zaman çizelgesindeki atama için uygun en az sayıda geçerli saate sahip olan faaliyet seçilir.
- **Renk derecesi:** Bu sezgisel, boşluklara atanmış faaliyetlerle atanmamış faaliyetler arasındaki çakışma derecelerine göre öncelik belirler.

Faaliyetler zamanlama için bir kez sıralandığında her bir faaliyet için uygun bir saat boşluğu seçmek amacıyla bir kaç yaklaşım kullanılabilir. Örneğin en erken uygun saat boşluğu seçilebilir veya hedef fonksiyonda tanımlanan en iyi uygun saat boşluğu seçilir.

2.2 Sezgi Üstü Yöntemler

Geçtiğimiz 20 yıl boyunca simulated annealing, tabu arama, genetik algoritmalar ve hibrid yaklaşımlar gibi çeşitli sezgi üstü yöntemler araştırılmış ve araştırılmaktadır. Sezgi üstü yöntemler bir veya daha fazla önceden üretilmiş çözümle başlar ve verilen parametreler doğrultusunda yeni bir sonuç üretir. Bu sonuç başlangıçtakinden daha iyi değilse, istenilene yakın bir sonuç bulununcaya kadar algoritma rekürsif bir şekilde çalışmaya devam eder. Bunun için arama stratejileri kullanılır (Yerel arama,

Tabu arama vb.). Bu arama algoritmalarının hepsi çok iyi sonuçlar üretebilir fakat maliyetleri çoktur. [1]

2.2.1 Hibrid Sezgisel Yöntemler

Faaliyetler sınav olsun ve graf renklendirmedeki sezgisellerle sıraya dizilmiş olsun. Sıralanmış sınavlar uygun saat boşluklarına değerlendirme (evaluation) fonksiyonu ile atanırlar. Değerlendirme fonksiyonu *maliyet(t), t* zaman çizelgesindeki bazı zayıf kısıtlarla hesaplanır:

$$maliyet(t) = 5000 * atanmamış(t) + 3 * aynıgün(t) + gecelik(t)$$

atanmamış(t): Saat boşluğuna atanmamış sınavların sayısını tutar.
aynıgün(t) ve gecelik(t): Gündüz ve gece birbirlerine çok yakın (komşu) zamanda sınavı olan öğrencileri içeren kısıtların sayısını tutar.

İki seçim tipi vardır:

1. **Turnuva seçimi:** Rastgele faaliyetlerin alt kümesini oluşturur ve sonra bu alt kümelerdeki birinci olarak işaretlenmiş faaliyeti seçer.
2. **Bias seçimi:** İlk *n* tane faaliyetten bir faaliyetin rastgele seçilmesidir.

Araştırmalarda, bu şekilde yapılan zaman çizelgesi oluşturma yöntemleri, sıralı yöntemlerdeki sade sezgisellerle elde edilen sonuçlardan daha iyi sonuç verdiği gözlenmiştir.

2.2.2 Tabu Arama (Tabu Search)

Tabu arama, birleşimsel (combinatorial) optimizasyon problemlerini çözmek için kullanılan sezgi üstü bir algoritmadır. Bir bitiş kriteri karşılanana kadar tekrarlı olarak bir *x* çözümünden *x'* çözümüne geçiş için lokal veya komşuluk arama prosedürü kullanır. [2]

Zaman çizelgesi oluşturmada Tabu Arama farklı yerlerde çeşitli özel gereksinimlerle birlikte uygulanmaktadır. Tabu aramanın değişik uygulamalarıyla ve bu uygulamarda kullanılan uygun bir şekilde seçilmiş parametrelerle (tabu listesi, başlangıç çözümleri, hedef fonksiyonlar v.b.) iyi sonuçlar elde edilmiştir. Tabu arama ile diğer yöntemlerin birleştirilmesi üzerinde de çalışılmıştır. 1997'de White ve Zhang kısıt tabanlı yöntemden elde edilen sonucu tabu arama yönteminde başlangıç çözümü oalrak kullanmış ve normalden daha iyi sonuç elde etmişlerdir.

2.2.3 Simulated Annealing

Simulated annealing, global optimizasyon problemi için genel olasılıksal bir algoritmadır. Büyük bir arama uzayında verilmiş bir fonksiyonun global optimumuna iyi bir yaklaşık değer bulur. Genellikle arama uzayı ayrık olduğu zaman kullanılır (Örn: Belirli bir grup şehri gezen turlar). Algoritmadaki amaç en iyi olası çözümü bulmak değil, belirli bir zamanda kabul edilebilir iyi bir sonuç bulmaktır.[2]

Ders programı zaman çizelgesi oluşturmada bu algoritma üzerinde de çalışmalar yapılmıştır. 1991'de Abramson yaptığı çalışmalarda ileride karşılaşılaşılacak problemlere değinmiştir. Bazı çalışmalar göstermiştir ki simulated annealing'in uygulanması birçok düzenlemelere ve parametrelere bağlıdır.

2.3 Küme Yöntemler

Bu yöntemlerde durum kümesi, katı kısıtları karşılayacak gruplara ayrılır ve sonra zayıf kısıtları karşılamak için gruplar zaman aralıklarına atanırlar. Durum gruplarını zaman aralıklarını atama problemini çözmek için farklı optimizasyon teknikleri kullanılmıştır. Bu yaklaşımların temel dezavantajı; durum kümeleri algoritmanın başında oluşturuluyor ve bu da zayıf bir zaman çizelgesi oluşmasına neden olmasıdır.

2.4 Kısıt Tabanlı Yöntemler

Bu yöntemlerde zaman çizelgesi problemi bir değişkenler kümesi olarak modellenir (örneğin faaliyetler). Bu değişkenlere, belirli sayıdaki kısıtları karşılamak için derslik sayısı, saat boşlukları gibi değerler atanmanlıdır. Genellikle kaynakları faaliyetlere atamak için belli sayıda kural tanımlanır. Eğer hiç bir kural çözüme uygun değilse geri dönüş yapılır ve uygun bir çözüm bulunana kadar işlem devam eder.

Kısıt tabanlı yöntemler için Kısıt Mantıklı Programlama (Constraint Logic Programming) dilleri geliştirilmiştir. Araştırmaların çoğu farklı problemlere adapte edilebilecek teknikler geliştirmek üzere yapılmıştır. Ders programı zaman çizelgesi oluşturmada yaygınca kullanılan iki deklaratif dil geliştirilmiştir; CHIP ve ECLIPS. Başka zaman çizelgesi problemleri için de farklı diller geliştirilmiştir; WPROLOG, COASTOOL, Oz ve EaCL. [1]

3. GRAFLAR

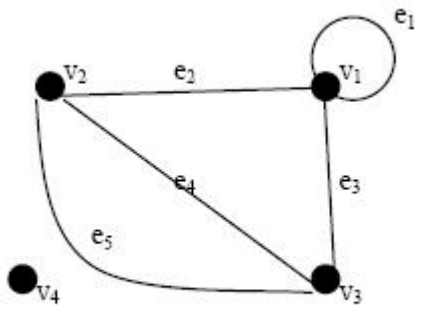
3.1 Graf Nedir?

Bir graf düğüm olarak adlandırılan noktalar ve her biri bu noktaları veya sadece noktanın kendisini birleştiren ve ayrıt olarak adlandırılan çizgiler topluluğudur. Örnek olarak şehirleri düğüm(vertice) ve onları bağlayan yolları ayrıt(edge) olarak gösteren yol haritaları verilebilir. [3]

Bir grafı tanımlamak için öncelikle düğümlerin ve ayrıtların kümesini tanımlamamız gerekir. Daha sonra hangi ayrıtların hangi düğümleri bağladığını belirtmeliyiz. Bir ayrıt her iki ucunda da bir düğüm olacak şekilde tanımlandığından graftaki tüm ayrıtların uç noktalarını bir düğüm ile ilişkilendirmek gerekir. Bu nedenle, her bir e ayrıtı için $\{v_i, v_j\}$ kümesi tanımlarız. Bunun anlamı e ayrıtının v_i ve v_j düğümlerini bağladığıdır. $v_i = v_j$ olabilir. $\{v_i, v_j\}$ kümesi $\delta(e)$ ile gösterilir ve düğümler kümesinin bir alt kümesidir.

Tanım: Bir yönsüz (undirected) graf G şunlardan oluşur: G(V,E)

i. Boş olmayan sonlu bir V düğümler kümesi
ii. Sonlu bir E ayrıtlar kümesi ve
iii. Bir δ:E→*P*(V) fonksiyonu öyle ki her bir e ayrıt' i için δ(e) V 'nin bir veya iki elemanlı bir alt kümesidir.



Şekil 1 - Yönsüz G graf gösterimi

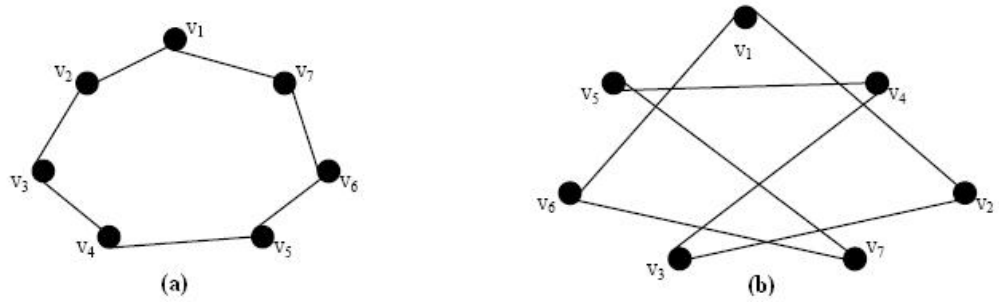
Şekil 1'deki G grafına bakalım. Açıktır ki, G grafının düğüm kümesi $\{v_1, v_2, v_3, v_4\}$ ve ayrıt kümesi $\{e_1, e_2, e_3, e_4, e_5\}$ dir. δ: $E \rightarrow P(V)$ fonksiyonu şöyle tanımlanmaktadır:

$\delta: e_1 \rightarrow \{v_1\}$
$\delta: e_2 \rightarrow \{v_1, v_2\}$
$\delta: e_3 \rightarrow \{v_1, v_3\}$
$\delta: e_4 \rightarrow \{v_2, v_3\}$
$\delta: e_5 \rightarrow \{v_2, v_3\}$

Bu basitçe, e_1'in v_1 düğümini kendisine, e_2'nin v_1 ve v_2 düğümlerini vs. bağladığını gösterir.

Yukarıda görüldüğü gibi bir ayrıt bir düğümü yine kendisine bağlayabileceği gibi (loop), bir düğüm hiçbir ayrıt ile bağlanmamış olabilir (v_4' te olduğu gibi). Ayrıca iki düğüm birden fazla ayrıt ile de (multiple edge) bağlanmış olabilir. Eğer bir graf bir düğümü yine kendisine bağlayan bir ayrıt ve aynı iki düğümü birden fazla bağlayan ayrıtlara sahip değilse basit (simple) graftır.

Dikkat edilmesi gereken bir nokta bir graf ile onu temsil eden diyagram aynı değildir. Daha önce de söylediğimiz gibi bir graf bir fonksiyon ile birlikte iki kümeden oluşur. Şekil 1 kendi başına bir graf değildir sadece bir grafın gösterimidir. Verilen bir graf birbirinden farklı şekillerde gösterilebilir. Örneğin şekil 2'deki iki diyagram çok farklı görünmelerine rağmen aynı G grafını temsil ederler. Şekil 2(a)'daki graf 7 düğümlü çember (wheel) graf olarak adlandırılır ve W_7 şeklinde gösterilir. Tüm n pozitif tamsayıları için n düğümlü ve n ayrıtlı bir W_n çember grafı vardır.



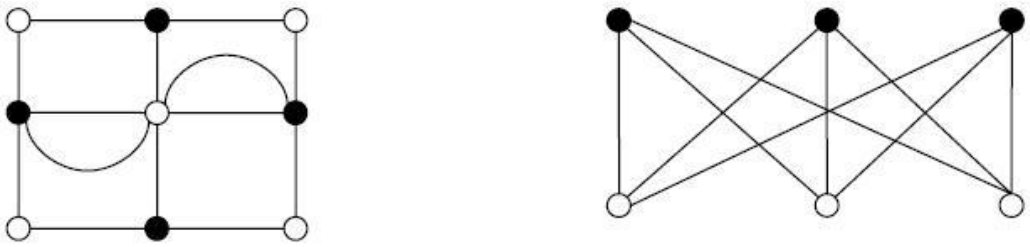
Şekil 2 - G grafı iki farklı şekilde gösterim

Tanımlar

- v ve w düğüm çiftini bağlayan bir e ayrıt' ı varsa bu iki düğüm komşudur (adjacent).
- $e_1, e_2, \ldots e_n$ ayrıtları en az bir ortak düğüme sahipse komşudur.
- Bir v düğümünün derecesi (degree) $\sigma(v)$, v düğümüne bağlı olan ayrıtların sayısıdır. (Aksi belirtilmediği sürece v' yi kendisine bağlayan ayrıt v'nin derecesini iki arttırır.) Tüm düğümleri aynı r derecesine sahip grafa r dereceli düzenli (regular) graf veya r-derece denir.
- Bir boş graf (null) veya tamamen bağlı olmayan (totally disconnected) graf ayrıt kümesi boş olan graftır.
- Bir tam (complete) graf ayrı düğüm çiftlerinin tümü bir ayrıt ile bağlı olan basit graftır ve n düğüm sayısı olmak üzere K_n şeklinde gösterilir.
- Düğüm kümesi, tüm ayrıtların V_1'in bir düğümünü V_2'nin bir düğümüne bağladığı $\{V_1, V_2\}$ şeklinde bir bölmelemeye sahip olan grafa iki parçalı (bipartite) graf denir.
- Bir tam iki parçalı (complete bipartite) graf V_1'in tüm düğümlerini V_2'nin tüm düğümlerine tek bir ayrıt ile bağlayan iki parçalı graftır.

Örnek: G, V düğüm kümesinin $\{V_1, V_2\}$ bölmelemesine sahip olduğu iki parçalı bir graf olsun. Dikkat edilirse G basit graf olmak zorunda değildir. Gereken tek şey her bir ayrıt V_1'in bir düğümü ile V_2'nin bir düğümünü bağlamalıdır. $v_1 \in V_1$ ve $v_2 \in V_2$ dersek, bunları bağlayan birden fazla ayrıt olabilir veya hiç ayrıt olmayabilir. Açıkça görülüyor ki, G'de loop yoktur.

Bir tam iki parçalı graf $|V_1|$ ve $|V_2|$ ile belirtilir. n ve m düğümlü tam iki parçalı graf $K_{n,m}$ ile gösterilir ve $|V_1|$=n ve $|V_2|$=m 'dir. Şekil 3 iki tane iki parçalı graf örneğidir. Her iki şekilde de V_1'in düğümleri içi dolu noktalar ile V_2'nin düğümleri ise içi boş noktalar ile gösterilmiştir. Şekil 3(b)'deki graf tam iki parçalı $K_{3,3}$ grafıdır.



Şekil 3 - İki parçalı graflar

G grafının farklı gözüken diyagramlar ile gösterilebileceğini belirtmiştik. Bir grafı göstermenin başka bir yolu da aşağıda tanımlayacağımız komşuluk matrisi (adjacency matrix) yardımıyla olur.

Tanım: G düğüm kümesi $\{v_1, v_2,..,v_n\}$ olan bir graf olsun. G'nin komşuluk matrisi; a_{ij}, v_i ve v_j'yi bağlayan ayrı ayrıtların sayısı olmak üzere n x n A = A(G) matrisidir.

Komşuluk matrisi v_i ve v_j'yi bağlayan ayrıtların sayısı, v_j ve v_i'yi bağlayan ayrıtların sayısıyla aynı olduğundan simetrik olmalıdır. v_i düğümünün derecesi komşuluk matrisinden kolayca belirlenebilir. v_i de bir loop yoksa bu düğümün derecesi matrisin i. sütunundaki değerlerin toplamıdır. Her bir loop dereceyi iki kere etkilediğinden i. sütundaki değerleri toplarken a_{ii} diyagonal elemanın iki katı alınır.

Örnek: Aşağıdaki A komşuluk matrisi Şekil 1'de gösterilen grafa aittir.

$$A = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 0 & 2 & 0 \\ 1 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Şekil 4 - Graf komşuluk matrisi

A'nın satır ve sütunları grafın düğümlerini temsil eder. Grafın iki özelliği matrise bakılarak hemen görülebilir. Öncelikle diyagonale bakıldığında bir tek çevrim vardır (v_1' den kendisine). İkincisi, son satır veya sütundaki 0'lar v_4'ün bir izole edilmiş düğüm yani başka hiçbir düğüme bağlı olmayan (kendisi dahil) bir düğüm olduğunu gösterir.

Tanım: Σ grafının tüm e ayrıtleri için, $V_\Sigma \subseteq V_G$, $E_\Sigma \subseteq E_G$ ve $\delta_\Sigma(e)=\delta_G(e)$ ise Σ grafı G grafının alt grafıdır (subgraph) denir ve $\Sigma \leq G$ şeklinde gösterilir. Σ grafının tüm e ayrıtları için $\delta_\Sigma(e)=\delta_G(e)$ durumu Σ alt grafının ayrıtlarının G de olduğu gibi aynı düğümleri bağlaması gerektiği anlamına gelir. Eğer G' nin diyagramından bazı düğümleri veya ayrıtları silerek Σ 'nin diyagramını elde edebiliyorsak Σ, G 'nin alt grafıdır. Tabii ki, bir düğümü siliyorsak ona bağlı olan tüm ayrıtları da silmeliyiz.

3.2 Yollar (Paths) ve Devreler (Circuits)

Eğer bir graf çeşitli şehirleri bağlayan yol ağını temsil ediyorsa aklımıza şu soru gelebilir: Her yoldan bir kere geçerek ve her şehre sadece bir kez uğrayarak aynı şehirden başlayıp aynı şehirde biten bir yolculuk yapılabilir mi?

Tanımlar

- G grafında n uzunluğunda ayrıt dizisi; i =1, 2, … n-1 için e_i ve e_{i+1} komşu olmak üzere e_1, e_2,…,e_n ayrıtlarının dizisidir. Ayrıt dizisi, $\delta(e_i)=\{v_{i-1}, v_i\}$ olmak üzere $v_0,v_1,v_2,\ldots,v_{n-1},v_n$ düğüm dizisini belirler. v_0'a ilk düğüm, v_n'e son düğüm denir. v_0 ile başlayıp v_n ile biten ayrıt dizisine tur (walk) denir. [4]
- Eğer turun başlangıç düğümü ile bitiş düğümü aynı ise ($v_0 = v_n$) bu tur kapalı turdur (closed walk).[4]
- Eğer bir turdaki ayrıtların üzerinden birden fazla geçilmiyorsa buna iz (trail) denir. Kapalı ize devre (circuit) denir.[4]
- Eğer bir turdaki düğümler birden fazla ziyaret edilmiyorsa bu tura yol (path) denir. Kapalı yola çember (cycle) denir.[4]

Bir ayrıt dizisi, grafın diyagramında kalemi kağıdın üzerinden kaldırmadan çizebileceğimiz herhangi sonlu ayrıt dizisidir. Bir yolda aynı ayrıttan birden fazla geçmeye izin verilmez. Ayrıt dizisi veya yol, bir yerden başlayıp aynı yerde bitiyorsa kapalıdır.

G, Şekil 1'de gösterilen graf olsun. Komşuluk matrisi Şekil 4'de gösterilen matris olsun. A' nın (i,j). elemanı v_i ve v_j düğümlerini bağlayan ayrıtların sayısıdır. Bunu, bu iki düğümü bağlayan 1 uzunluğunda ayrıt dizilerinin sayısı şeklinde düşünebiliriz. Bu durumda, komşuluk matrisinin karesi:

$$A^2 = \begin{bmatrix} 3 & 3 & 3 & 0 \\ 3 & 5 & 1 & 0 \\ 3 & 1 & 5 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Şekil 5 - A komşuluk matrisinin karesi

A^2'de (i,j). eleman v_i ve v_j'yi bağlayan 2 uzunluğunda ayrıt dizilerinin sayısını temsil eder. Örneğin, (2,2). eleman 5'tir ve v_2'yi kendisine bağlayan 2 uzunluğunda 5 tane ayrıt dizisi vardır: e_2, e_2; e_4, e_4; e_5, e_5; e_4, e_5; e_5, e_4.

Bunun niçin böyle olduğunu görmek kolaydır. A^2'nin (i,j). elemanı A'nın i. satırı ile j. sütununun çarpılması ile elde edilir:

$$\sum_{k=1}^{n} a_{ik} a_{kj}$$

Toplamdaki r. terim $a_{ir}a_{rj}$, v_i ve v_r'yi bağlayan ayrıtların sayısı ile v_r ve v_j'yi bağlayan ayrıtların sayısının çarpımıdır. Bir başka ifade ile v_i ve v_j'yi v_r aracalığı ile bağlayan 2 uzunluğundaki ayrıt dizilerinin sayısıdır. Tüm k değerleri için ortaya çıkanların toplanması v_i ve v_j'yi bağlayan 2 uzunluğunda ayrıt dizilerinin sayısını verir.

Teorem: G, düğüm kümesi $\{v_1,v_2,\ldots, v_n\}$ ve komşuluk matrisi A olan bir graf olsun. A^n'nin (i,j). elemanı v_i ve v_j' yi bağlayan n uzunluğunda ayrıt dizilerinin sayısıdır.

3.2.1 Bağlılık (Connectedness)

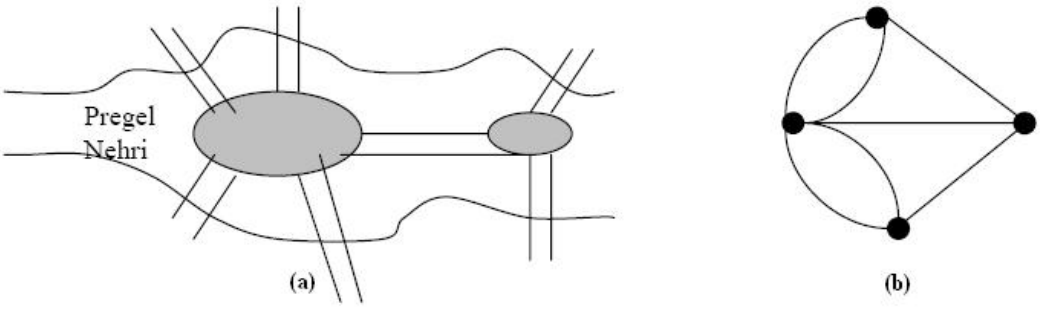
Bazı graflar tek parça halinde iken diğerleri çeşitli parçalardan oluşuyor olabilir. Bu fikri daha belirgin hale getirmek için yolları kullanabiliriz.[2]

Tanım: Bir grafın birbirinden ayrı düğümlerini bağlayan bir yol varsa o graf bağlıdır (connected). Herhangi bir graf bileşen (component) adı verilen belli sayıda bağlı alt graflara bölünebilir. Bileşenler maksimal bağlı alt graflar olarak tanımlanabilirler. Bunun anlamı G_1, G'in bağlı bir alt grafı ise ve kendisi G'nin başka herhangi bir bağlı alt grafının alt grafı değilse, G'nin bileşenidir. Bu ikinci durum maksimal terimi ile anlatmak istediğimiz şeydir yani Σ, $G_1 \leq \Sigma$ olacak şekilde bir bağlı alt grafsa $\Sigma = G_1$, böylece G'nin G_1'den daha büyük bağlı bir alt grafı yoktur.

3.2.2 Euler Yolları

Euler 1736 yılında yazdığı makale ile graf teorisinin doğmasına sebep olmuştur. Bu makale aşağıda tanımlanan Königsberg Köprü problemlemini çözebilen bir teoriyi içeriyordu. Pregel nehri Königsberg kasabasının içinden akmaktadır. Nehrin ortasında Şekil 6(a)'daki gibi nehrin kıyılarına ve birbirine köprüler ile bağlı iki ada bulunmaktadır. Königsberg kasabasının vatandaşları için problem: Kıyıların veya adaların birinden başlayıp tüm köprülerden sadece bir kez geçerek başladığımız yere yürüyebilir miyiz?

Euler öncelikle Şekil 6(b)'deki gibi Königsberg coğrafyasının gerekli özelliklerini bir graf ile gösterdi. Her bir nehir kıyısı ve adalar bir düğüm ile köprüler de ayrıtlar ile temsil edildi. Graf teorisi terimleri ile problem şu hale geldi: Grafın tüm ayrıtlarını içeren kapalı bir yol var mıdır?



Şekil 6 - Könignsberg köprüleri ve Euler grafı

Tanım: G grafındaki bir Euler Yolu, G'nin tüm ayrıtlarını içeren kapalı bir yoldur. Bir graf içinde en az bir Euler yolu barındırıyorsa bu graf Euler grafıdır. Bir yolda hiçbir ayrıtdan birden fazla geçilemeyeceğinden Euler yolu tüm ayrıtları sadece bir kez içerir fakat düğümlerden birden fazla geçilebilir. Bağlı bir G grafında Euler yolu olup olmadığını belirlemek için gereken durumu tanımlamak çok kolaydır. Bütün düğümlerin derecesi çift olmalıdır. Bunu görmek için G, bağlı ve Euler yoluna sahip bir graf olsun. G bağlı olduğundan Euler yolunun düğüm dizisi bütün düğümleri içerir. Yol ne zaman bir düğümden geçse bu derecesini iki katkı yapar. Tüm ayrıtlar yolda bir kere bulunduğundan her düğüm çift dereceye sahip olmalıdır. Königsberg'dekiler aradıkları yolu bulamamakta haklıdırlar zira böyle bir yol yoktur. Problemi temsil eden Şekil 6(b)'deki graf bağlıdır fakat gerekli koşulu sağlamaz. Aslında tüm düğümlerin derecesi tektir.

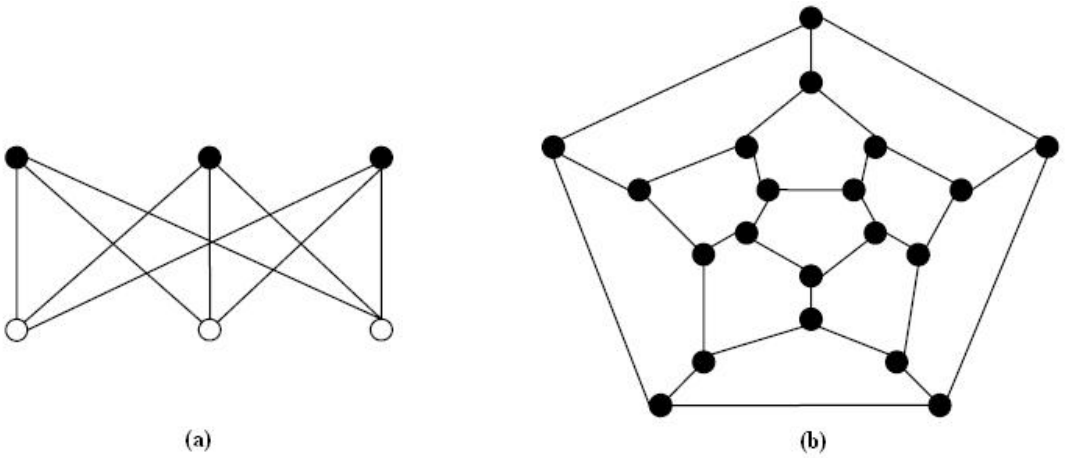
Teorem: Bağlı bir G grafı sadece ve sadece tüm düğümleri çift dereceli ise Euler grafıdır.

3.2.3 Hamilton Devreleri

Benzer bir problem de herhangi bir ayrıtdan birden fazla geçmemek kaydıyla her bir düğümü sadece bir kez ziyaret edip başladığımız yere geri dönebilir miyiz? Bu problem Hamilton tarafından irdelenmiştir ve ismi bu yollar ile birlikte anılmaktadır.

Tanım: Bir graftaki Hamilton devresi tüm düğümlerden bir kez geçen bir devredir. Bir graf içinde bir Hamilton devresi barındırıyorsa Hamilton grafıdır.

Örnek: Şekil 7'de iki tane Hamilton devresi vardır.



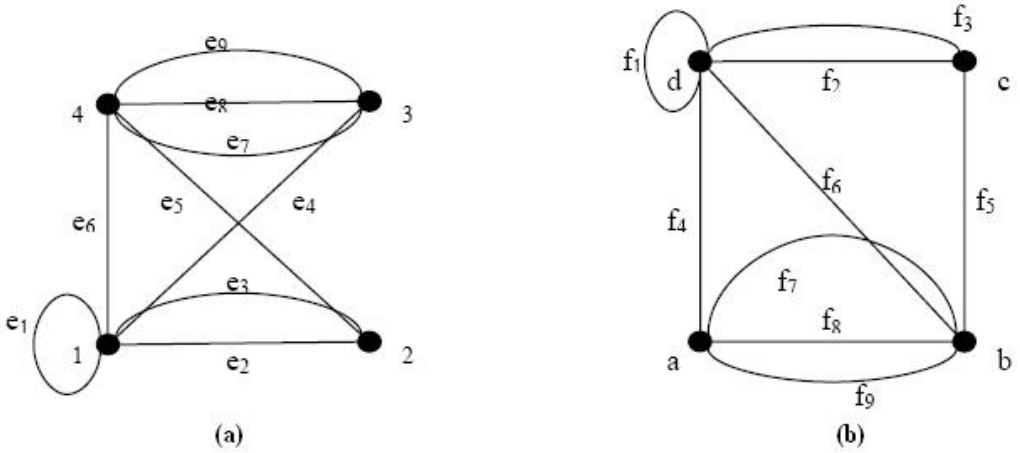
Şekil 7 - Hamilton devreleri

Euler grafları basit bir karaktere sahipken aynı durum Hamilton grafları için söz konusu değildir. Aslında bir asırdan beri üzerinde çalışıldığı halde Hamilton graflarının yapısı hakkında bir şey bilinmemektedir (Yapısı ile bir grafın Hamilton olması için gerek ve yeter koşul kastedilmiştir). Bu, graf teorisinin çözülememiş büyük problemlerinden biridir. Açık bir gerek koşul grafın bağlı olmasıdır. Ayrıca çeşitli yeter koşullar da bilinmektedir.

Teorem: Eğer, G_n (≥ 3) düğümlü basit bağlı bir graf ise ve tüm v düğümleri için derecesi $\sigma(v) \geq 1/2n$ ise G Hamiltondur. Dereceler ile ilgili koşul G'nin Hamilton olması için gerek koşul değildir o halde, bu koşulu sağlamayan bir graf da Hamilton olabilir. Şekil 7(b)'ye bakarak bunu görebiliriz. Grafın 15 düğümü vardır, her düğümün derecesi 3'tür fakat hala Hamilton grafıdır.

3.3 Graf İzomorfizmi

Aşağıdaki G ve Σ graflarını düşünelim. G'nin düğüm kümesi {1,2,3,4}, komşuluk matrisi Şekil 9'daki A ve Σ'nin düğüm kümesi {a,b,c,d}, komşuluk matrisi Şekil 9'daki B olsun.[3]



Şekil 8 - (a) G grafı, (b) Σ grafı

$$A = \begin{bmatrix} 1 & 2 & 1 & 1 \\ 2 & 0 & 0 & 1 \\ 1 & 0 & 0 & 3 \\ 1 & 1 & 3 & 0 \end{bmatrix} \quad B = \begin{bmatrix} 0 & 3 & 0 & 1 \\ 3 & 0 & 1 & 1 \\ 0 & 1 & 0 & 2 \\ 1 & 1 & 2 & 1 \end{bmatrix}$$

Şekil 9 - G ve Σ graflarının komşuluk matrisleri

Biraz dikkatli bakılırsa Şekil 8'de gösterilen grafların aynı yapıya sahip olduğu görülebilir. Σ grafındaki a,b,c,d düğümlerini 3,4,2,1 şeklinde; ve f_i ayrıtlarını i=1,…,8 için e_i ile tekrar etiketlersek Şekil 8'deki iki diyagrama aynı grafın farklı gösterimleri şeklinde bakabiliriz. Tabii ki, G ve Σ grafları birebir aynı değildir. Örneğin farklı düğüm kümelerine sahiptirler. Fakat aynı yapıya sahiptirler. G ve Σ grafları izomorfik graflar diyebiliriz. G ≅ Σ şeklinde gösterilir.

Teorem: G'den Σ'ya bir izomorfizm olsun. Bu durumda;

- G ve Σ aynı sayıda düğüme sahiptir;
- G ve Σ aynı sayıda ayrıta sahiptir;
- G ve Σ aynı sayıda bileşene sahiptir;
- Birbirine karşılık gelen düğümler aynı dereceye sahiptir;
- G basitse, Σ da öyledir;
- G Euler grafı ise Σ de Eulerdir.
- G Hamilton grafı ise Σ de Hamiltondur.

3.4 Graf Renklendirme

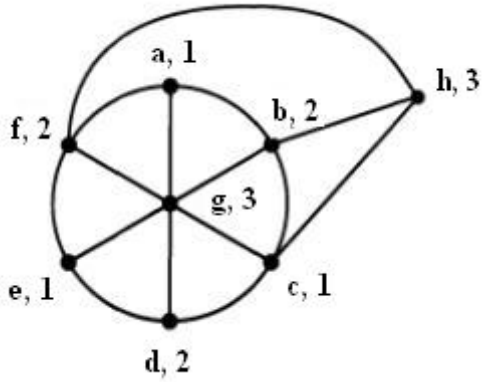
Bir kimya şirketinde X çalışanı şirket deposundaki kimyasal bileşenlerin depolanmasından sorumludur. Belli tip bileşenlerin (asit ve baz gibi) aynı yerde bulundurulmaması gerektiğinden dolayı, Y çalışanından depoyu iki ayrı depolama alanına bölmesini istiyor. Böylelikle birbirine zıt olan kimyasallar ayrık yerlerde depolanabilecek. X, Y'nin kuracağı depolama alanlarının sayısına nasıl karar verir?[4]

Eğer bu şirket n kimyasal bileşen satıyorsa $V=\{c_1,c_2,...,c_n\}$, düğüm kümesi olsun. Bütün $1 \leq i < j \leq n$ için, eğer c_i ve c_j farklı alanlarda depo edilmek zorundaysa $\{c_i,c_j\}$ ayrıtını çizeriz. Bu bize yönsüz G=(V,E) grafını oluşturur.

Eğer G=(V,E) grafı yönsüz bir grafsa G'nin düğümlerini renklendirdiğimizde uygun renklendirme yapmış oluruz. Eğer {a,b} bir ayrıtsa o zaman a ve b düğümleri farklı renklerle renklendirilirler. Yani komşu düğümler farklı renklerle renklendirilirler. G'nin uygun renklendirilmesi için gerekli olan minimum renk sayısına G'nin renksel (chromatic) sayısı denir ve $\chi(G)$ ile gösterilir.

Örnek: Şekil 10'daki G grafı için a düğümünden başlayarak her bir sonraki düğüme uygun renklendirme yapmak için renk numarası veriyoruz. b düğümündeki 2 rakamı, ikinci bir renge ihtiyacımız olduğunu belirtiyor çünkü a ve b düğünleri komşular ve

komşu düğümler aynı renkle renklendirilemezler. f de dahil olmak üzere {a,b,c,d,e,f} düğümlerinin uygun renklendirme için 2 farklı renge ihtiyacı olduğunu görüyoruz. g düğümü için üçüncü bir renge ihtiyacımız vardır. Bu renk h düğümü için de kullanılabilir çünkü g ve h düğümleri komşu değillerdir (aralarında tek ayrıt yoktur). Böylelikle bu sıralı renklendirme yöntemiyle G grafının renksel sayısının 3 olduğu tespit edilmiştir ($\chi(G) = 3$).



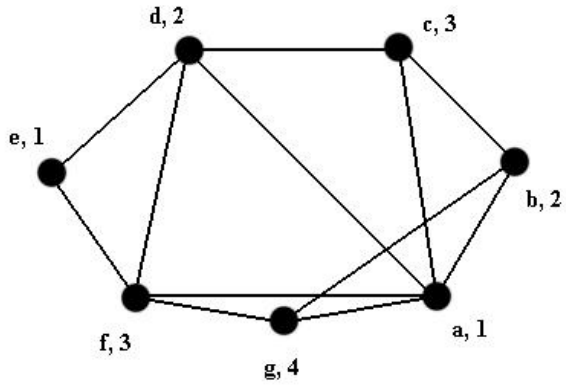
Şekil 10 - Renk numaraları atanmış G grafı

Örnek: Final sınavlarının planlamasını yapacağımızı düşünelim. Bir öğrencinin aynı gün içerisinde birden fazla sınava girmemesini istiyoruz. A = {a, b, c, d, e, f, g} kümesi dersleri göstersin. Tablo 1'de en az 1 ortak öğrencisi olan dersler * ile gösterilmiştir ve bu derslerin aynı günde sınavı olamaz. Bütün sınavları planlamak için en az kaç güne ihtiyacımız vardır?[6]

	a	**b**	**c**	**d**	**e**	**f**	**g**
a	.	*	*	*	-	*	*
b	*	.	*	-	-	-	*
c	*	*	.	*	-	-	-
d	*	-	*	.	*	*	-
e	-	-	-	*	.	*	-
f	*	-	-	*	*	.	*
g	*	*	-	-	-	*	.

Tablo 1 - Dersler arasındaki öğrenci çakışması

İlk önce problemimizi nasıl renklendirme problemi olarak modelleyeceğimizi düşünmemiz lazım. Dersleri düğüm olarak ve öğrencinin gireceği dersleri de komşu olarak gösterecek bir graf hazırlarız. Bu grafı renklendirdiğimiz zaman (Şekil 11), komşu olan düğümler farklı renkle renklendirilecekler ve mümkün olan en az renk sayısı kullanılacaktır.



Şekil 11 - Çakışmaları gösteren renklendirilmiş graf. Harflerin yanındaki rakamlar belli bir rengi temsil etmektedir.

Grafın renksel sayısı 4'tür ve olası bir çözüm şu şekildedir;

Gün	Sınav
1	a, e
2	b, d
3	c, f
4	g

Tablo 2 - Sınav yerleştirme problemine olası bir çözüm

4. DERS ZAMAN ÇİZELGESİ OLUŞTURMA

Ders zaman çizelgesi oluşturma 1997 yılında Carte ve Laporte tarafından şu şekilde tanımlanmıştır:

"Öğrencilerin, hocaların derslere veya sınıflara, derslerin dersliklere ve saat boşluklarına atandığı çok boyutlu bir atama problemidir."[2]

Ders zaman çizelgesi oluşturma probleminde belli sayıdaki dersler dersliklere ve bir hafta içindeki limitli sayıdaki saat boşluklarına atanır. Tabi ki ders zaman çizelgesi oluşturma beraberinde bir grup kısıtı da ortaya çıkarır. Bu kısıtlar katı ve zayıf olacak şekilde sınıflandırılabilir. Okulların veya üniversitelerin kendilerine özel çok veya az sayıda kısıtları olabilir. Literatürdeki ders zaman çizelgesi oluşturma üzerindeki araştırmalar şu şekilde gruplanmıştır; Sınıf-hoca zaman çizelgesi oluşturma, öğrenci planlama, ders ataması, hoca ataması ve derslik ataması. Bu tezde ders ataması üzerinde çalışılmıştır.[2]

Şimdi bu problemin graf renklendirmede nasıl yapıldığına dair bir örnek problem üzerinde durulacak.

4.1 Graf Renklendirme ile Ders Zaman Çizelgesi

4.1.1 Temel Model

Konunun geri kalanında aşağıdaki notasyon kullanılacaktır.
$C = \{C_1, C_2, \ldots, C_n\}$ kümesi H haftası boyunca verilecek olan dersleri temsil etsin. H de saat boşluklarının kümesi olsun. Her bir C_i dersinin c_i ders saatini içerdiğini varsayıyoruz. Yani $C_i = \{C_i^1, C_i^2, \ldots, C_i^{c_i}\}$. Her bir s_t öğrencisi için $\overline{S}_t$'de, s_t öğrencisi tarafından seçilmiş derslerin kümesi olsun. [5]

Tanımlar

- Ders zaman çizelgesi oluşturma, C_i dersinin içerdiği her bir c_i ders saatinin H haftası içinde c_i tane saat boşluğuna atanması işlemidir.
- Bir s_t öğrencisi tarafından seçilmiş iki farklı dersin en az birer saati aynı zamana denk geliyorsa çakışma var demektir. Yani $C_i, C_j \in \overline{S}_t$, $\overline{C}_i \cap \overline{C}_j \neq \varnothing$.

  Bu tezin uygulama kısmında bir öğrencinin seçtiği derslerin saatlerinin çakışması bir kısıt olarak göz önüne alınmamıştır.[5]

Şimdi gösterilecek örnekte dersliklerin büyüklüğü ve ders mevcudu göz önüne alınmamıştır. Verilen veriler doğrultusunda daima uygun bir zaman çizelgesi oluşturmak mümkün olmayabilir. Bazı durumlarda gereksinimleri düşürüp bazı çakışmalara göz yummamız gerekebilir. Oluşturulacak olan zaman çizelgesi çakışmaları en aza indirme işine dönüşür. Çakışmanın ölçüsü yukarıda anlatılan saatleri çakışan derslerin kaç öğrenci tarafından seçildiğidir.

4.1.2 Graf Formülasyonu

Grafımızdaki düğümler dersleri, düğümler arasındaki ayrıtlar da iki ders arasında çakışma olduğunu belirtir.[5]

Tanımlar

- **Ağırlık (ceza) puanı:** İki C_i^r ve C_j^s dersi arasındaki ağırlık puanı (ω_{ij}), bu iki dersi aynı saatte alan öğrenci sayısıdır.

$$\omega_{ij} = \left| \left\{ t \mid C_i, C_j \in \overline{S}_t \right\} \right|$$

- Bir çakışmalı graf ayrıt ağırlıklandırmalı graftır ve şu şekilde tanımlanır: Her bir C_i dersi için c_i tane düğüm vardır. C_i^1, C_i^2,…, $C_i^{c_i}$ bu dersin saatlerini belirtir. Her bir *i, j* çifti için ($i \neq j$) eğer $\omega_{ij} > 0$ ise o zaman her bir *r, s* çifti için ($r \neq s$) C_i^r ve C_j^s düğümleri arasında ω_{ij} ağırlıklı bir ayrıt oluşturur.

Notasyon: x ve y düğümlerini bağlayan ayrıt [x, y] olarak gösterilir.

- Bir G grafında karşılıklı komşu olmayan düğümler alt kümesi, bağımsız (independent) düğümler kümesi olarak adlandırılır.
- G grafındaki komşu düğümlerin farklı renkler alacak şekilde renklendirilmesine uygun düğüm renklendirmesi denir.

Durumlar

- Zaman çizelgesi problemi toplam cezayı en aza indirmek için V(G) düğüm kümesini bir parça P (partition) bulma amacıyla $k=|W|$ tane alt kümeye böler.

$$z(P)=\sum_{u=1}^{k}\left(\omega_{ij}\,\middle|\, C_i^r, C_j^s \in S_u\right)$$

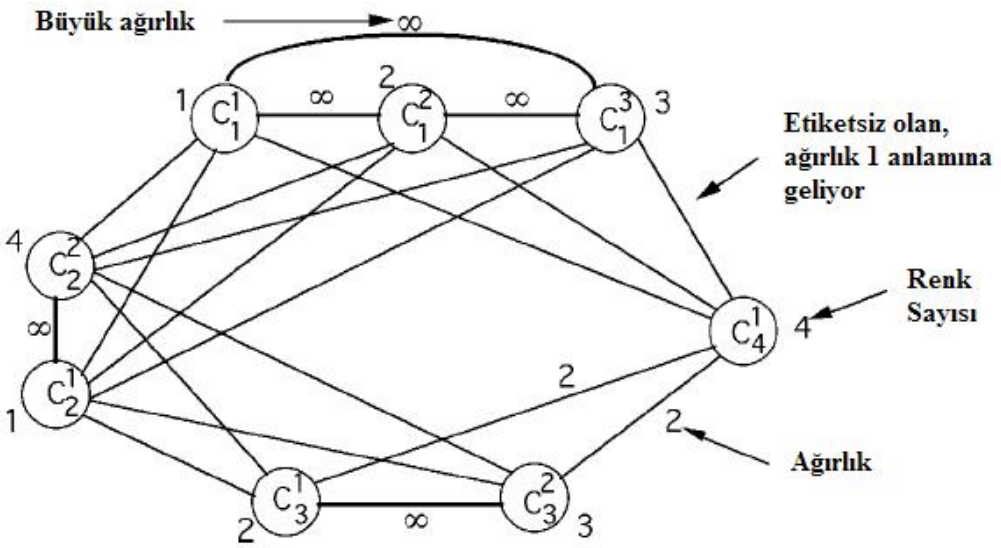
- Çakışmasız zaman çizelgeleri ve P parçalarıyla z(P) = 0 arasında bire bir örtüşme olduğunu görmek zor değildir. Böyle bir parçada $C_i^r \in S_u$ şu anlama gelir: C_i dersinin r ders saati $u \in W$ saat boşluğuna atanmıştır.
- z(P) = 0 olan bir P parçasında uygun düğüm renklendirmesi olur. Bu renklendirme parçanın bir hücresindeki her düğüme aynı renk verilerek yapılır. Böylelikle farklı hücreler farklı renkler alır.

Örnek: Şekil 12'de bir üniversite zaman çizelgesi problem örneği verilmektedir. Bütün derslerin farklı hocalar tarafından verildiği düşünülmektedir. Böyle düşünmediğimiz takdirde de aynı hocanın verdiği farklı dersler arasına büyük ağırlık koyarız. Bu işlem problemimizin yapısını değiştirmez.

C_1 = 3 saat
C_2 = 2 saat
C_3 = 2 saat
C_4 = 1 saat

$\overline{S_1} = C_1, C_2$
$\overline{S_2} = C_2, C_3$
$\overline{S_3} = C_1, C_3, C_4$
$\overline{S_4} = C_3, C_4$
$\overline{S_5} = C_2$

$\omega_{12} = 1$
$\omega_{13} = 1$
$\omega_{14} = 1$
$\omega_{23} = 1$
$\omega_{24} = 0$
$\omega_{34} = 2$



Şekil 12 - Üniversitesi Zaman Çizelgesi örneği

C_1^1, C_2^1 : 1. zaman aralığı		C_1^3, C_3^2 : 3. zaman aralığı	
C_1^2, C_3^1 : 2. zaman aralığı		C_2^2, C_4^1 : 4. zaman aralığı	

Çok şubeli dersleri planlama

m tane dersin planlanacağını düşünelim, $\{C_1, C_2,\ldots,C_m\}$. Her bir dersin tek saati olduğunu varsayalım. C_i dersinin de h_i tane şubesi olduğunu varsayalım, $h_1 \geq h_2 \geq \ldots \geq h_m$. Aşağıda anlatılacak olan 4 adımlık yöntem m tane dersin bütün şubeleri için zaman çizelgesi oluşturmaktadır. Yöntem bir örnek üzerinde gösterilerek anlatılacaktır.[5]

Adım 1: $G^* = (L^*, R^*, E^*)$ iki parçalı graf oluşturulur. $L^* = \{C_1, C_2, \ldots, C_m\}$ sol küme, $R^* = \{1, 2,\ldots, h_1\}$ sağ küme ve her bir i için, $i = 1, 2,\ldots, m$, $[C_i, j] \in E^*, j = 1, 2, \ldots, h_i$ için.

Adım 2: Her bir dersin şubeleri için uygun renk kümeleri oluşturulur.

- Eğer Δ G^*'deki maksimum düğüm derecesiyse o zaman $\Delta = \max\{m, h_1\}$

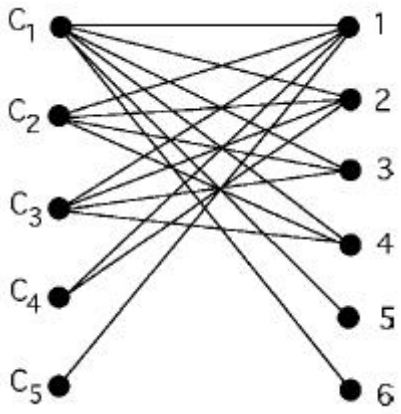
- G^* grafının uygun ayrıt Δ-renklendirmesi vardır. Bu grafın ayrıt renklendirmesi için $p(C_i)$, C_i dersine bağlı olan ayrıtlar için kullanılan renklerin kümesini temsil etsin, $i = 1, 2,\ldots, m$. $P(C_i) = h_i$

Adım 3: Öğrenci gruplarının bir $g = \{g_1, g_2,\ldots, g_n\}$ bir kümesi olsun. $G^{**} = (L^{**}, R^{**}, E^{**})$ iki parçalı graf oluşturulur. $L^{**} = \{C_1, C_2,..., C_m\}$ sol küme, $R^{**} = \{g_1, g_2,\ldots, g_n\}$ sağ küme ve her bir i, j çifti için, $i = 1, 2,\ldots, m$, $j = 1, 2,\ldots, n$, sadece ve sadece g_j öğrenci grubunun C_i dersini alırsa $[C_i, g_j] \in E^{**}$

Adım 4: Öğrenci gruplarının g kümesi $C_1, C_2,..., C_m$ derslerinin şubelerine hiçbir şubenin saat boşluğu değiştirilmeksizin atanır.

Zaman çizelgesi problemi açısından eğer C_i dersini alacak olan öğrenci grup sayısı C_i için planlanmış şubelerin sayısına eşitse, o zaman her öğrenci grubunun şube dersini alabileceği bir şube-öğrenci grubu ataması olmuş olur.

Örnek: Şekil 13'de $m = 5$ ve $(h_1, h_2,\ldots, h_3) = (6, 4, 4, 2, 1)$ ile verilmiş G^* grafı gösterilmektedir.



Şekil 13 - G* iki parçalı graf

Şekil 13'deki grafta Δ = 6 dır ve {a,b,c,d,e,f} renklerini kullanarak uygun ayrıt 6-renklendirmesi aşağıdaki matris ile gösterilmiştir. (i, j). Giriş [C_i, j] ayrıtına atanan renktir.

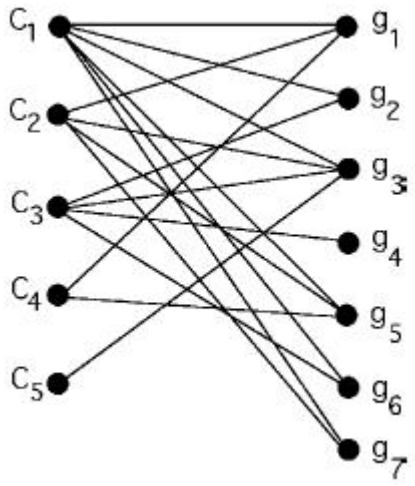
	1	2	3	4	5	6
C_1	a	b	c	d	e	f
C_2	b	e	d	a		
C_3	c	d	a	e		
C_4	d	a				
C_5	e					

Şekil 14 - Renklerin atanması

Buna karşılık düşen ders zaman çizelgesi C_i dersinin şubeleri için uygun renklere sahip $p(C_i)$ kümeleriyle verilir, i = 1, 2,…, m:

$p(C_1)$ = {a, b, c, d, e, f}
$p(C_2)$ = {a, b, d, e}
$p(C_3)$ = {a, c, d, e}
$p(C_4)$ = {a, d}
$p(C_5)$ = {e}

Şekil 15'de gösterilen G^{**} iki parçalı graf, 7 tane öğrenci grubunun spesifik gereksinimlerini göstermektedir. Örneğin, g_1 grubu C_1, C_2 ve C_4 derslerini alıyor ve grup g_6 da C_1 ve C_4 derslerini alıyor.



Şekil 15 - G iki parçalı graf**

Şekil 15'deki G^{**} iki parçalı graf için uygun ayrıt 6-renklendirmesi aşağıdaki matriste gösterilmiştir (Şekil 16).

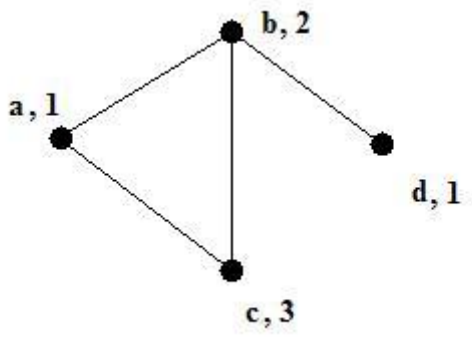
	1	2	3	4	5	6	7
C_1	f	e	d		a	b	c
C_2	d		b		e		a
C_3		d	c	a		e	
C_4	a				d		
C_5			e				

Şekil 16 - G grafı için 6-renklendirme**

5. UYGULAMA

Bu bölümde sıralı yöntem mantığı kullanarak geliştirilmiş olan Ders Programı Oluşturma Sihirbazı (DEPOS) uygulama yazılımı anlatılacaktır. Kısıt olarak 2 katı, 1 zayıf kısıt belirlenmiştir. Katı kısıtlar, günü ve saati sabit olan dersler ve bir dersin verildiği saat içerisinde başka bir veya birden fazla dersle çakışmaması. Zayıf kısıt, dışarıdan görevlendirme ile gelen hocaların, verdiği ders için gün ve saat önerisinde bulunması. Uygulamada öğrencilerin alttan veya üstten aldığı derslerle kendi sınıflarından aldığı derslerin çakışmama kısıdı göz ardı edilmiştir çünkü bölümümüzde ders programları ders seçimlerinden önce hazırlanmaktadır.

Şekil 17'de bir dersin başka derslerle çakışmamasını modelleyen renklendirilmiş bir G grafı gösterilmiştir. a dersi b ve c dersleriyle çakışmayacak ve b dersi c ve d dersleriyle çakışmayacak.



Şekil 17 - Derslerin yanlarındaki rakamlar renkleri temsil etmekte

a ve d dersleri aynı renk numarasını alabilmiştir çünkü çakışma kısıtları yoktur. Aralarında çakışma olan derslerin arasına ayrıt çizilir ve aralarında ayrıt olan düğümler (dersler) farklı renkle numaralandırılır.

5.1 Uygulama Veritabanı Modeli

Uygulamada veritabanı olarak MS Access kullanılmıştır. Veritabanımızda 6 tablo bulunmaktadır. Bunlar; Hocalar, Dersler, Hoca Öneri, Ders Çakışma, Sabit Dersler, Dağılım.

Hoca tablosunun yapısı Tablo 3'de gösterilmiştir. Hocanın adı-soyadı, hafta içindeki boş günü ve bir günde vereceği maksimum ders saati bilgileri tutulmaktadır. 'hocaID' alanı birincil anahtardır.

hocaID	Otomatik Sayı
hocaAd	Metin
bosGun	Sayı
gunlukMaxSaat	Sayı

Tablo 3 - Hoca tablo yapısı

Dersler tablosunun yapısı Tablo 4'de gösterilmiştir. Dersin kodu, adı, dersi veren hocanın ID bilgisi, dersin haftalık saat sayısı, saatlerin haftaya dağılım şekli, kaçıncı sınıf dersi olduğu ve dersin kısıt tipi bilgileri tutulmaktadır. "ID" alanı birincil anahtardır.

ID	Otomatik Sayı
dersKodu	Metin
dersAdi	Metin
hocaID	Sayı
saatSayisi	Sayı
dagilimID	Sayı
sinif	Sayı
tip	Sayı

Tablo 4 - Dersler tablo yapısı

Hoca Öneri tablosunun yapısı Tablo 5'de gösterilmiştir. Önerilen dersin kodu, öneri verilen gün ve saat bilgileri tutulmaktadır. "ID" alanı birincil anahtardır.

ID	Otomatik Sayı
dersKodu	Metin
gunOnerisi	Sayı
saatOnerisi	Sayı

Tablo 5 - Hoca Öneri tablo yapısı

Ders Çakışma tablosunun yapısı Tablo 6'da gösterilmiştir. Dersin kodu ve bu dersin saatlerinin çakışmayacağı ders veya derslerin kod bilgileri tutulmaktadır. "cakismaID" alanı birincil anahtardır.

cakismaID	Otomatik Sayı
dersKodu	Metin
cakismaDersKodu	Metin

Tablo 6 - Ders Çakışma tablo yapısı

Sabit Dersler tablosunun yapısı Tablo 7'de gösterilmiştir. Dersin kodu, dersin sabit olarak yerleşeceği saat ve gün bilgileri tutulmaktadır. "sabitDersID" alanı birincil anahtardır.

sabitDersID	Otomatik Sayı
dersKodu	Metin
gun	Sayı
saat	Sayı

Tablo 7 - Sabit Dersler tablo yapısı

Dağılım tablosunun yapısı Tablo 8'de gösterilmiştir. Haftalık dağılımın şekli ve toplam saat sayısı bilgileri tutulmaktadır. "dagilimID" alanı birincil anahtardır.

dagilimID	Otomatik Sayı
dagilim	Metin
saat	Sayı

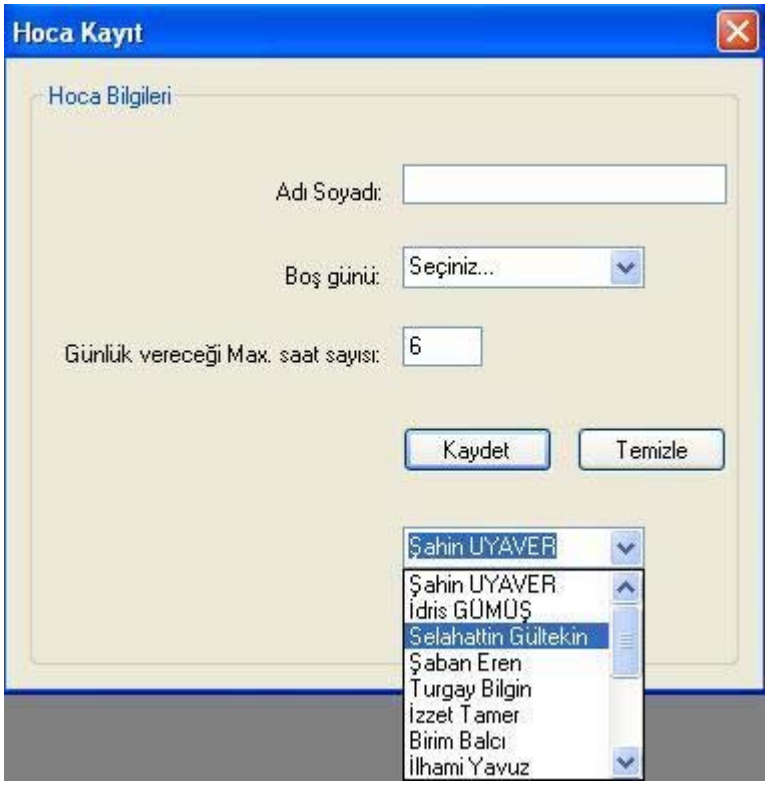
Tablo 8 - Dağılım tablo yapısı

5.2 Uygulama Modülleri

Uygulamamız 4 temel modülden oluşmaktadır. Bunlar; Hoca Kayıt, Ders Kayıt, Kısıtlar ve Programın Oluşturulması.

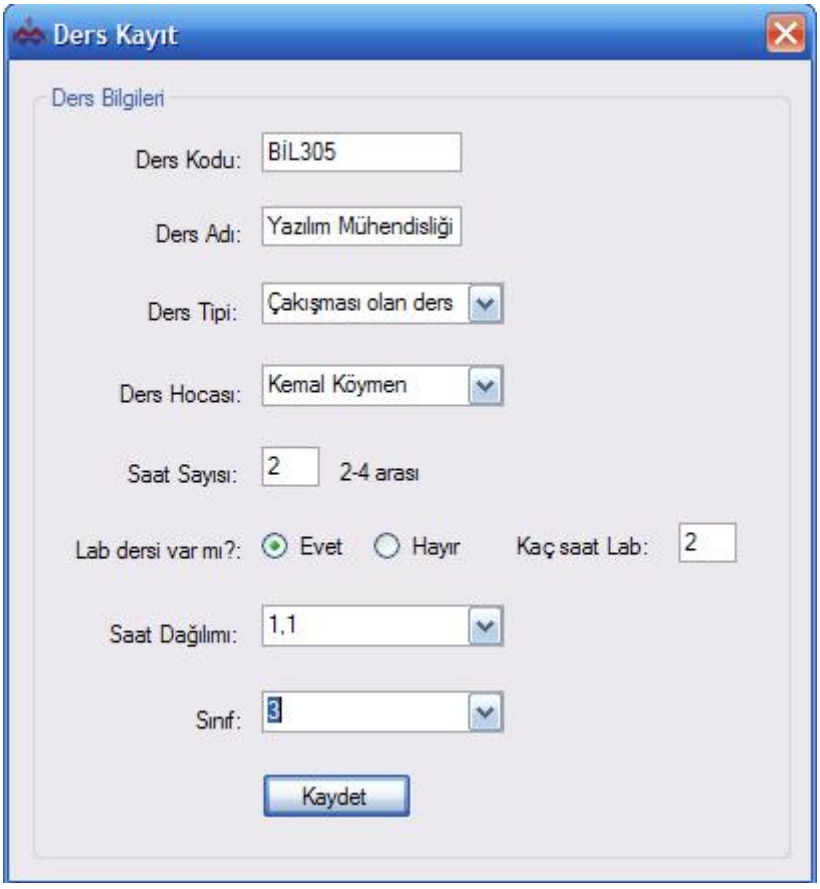
Şekil 18- Programın ana ekranı

Hoca kayıt kısmında programı oluşturulacak bölümün ders hocalarının bilgileri kaydedilebilir, değiştirilebilir veya silinebilir.



Şekil 19 - Hoca kayıt ekranı. Combo Box'da daha önceden kaydedilmiş hocalar listeleniyor.

Ders kayıt modülünde bölümdeki ders bilgileri kaydedilebilir, değiştirilebilir veya silinebilir. Ders bilgileri girilirken dersin tipine (Sabit ders mi, çakışması olan ders mi, gün ve saati hoca tarafından önerilen ders mi veya kısıtsız/normal ders mi), saat sayısına, bu saat sayısına bağlı olarak saat dağılım tipine ve kaçıncı sınıf dersi olduğuna dikkat edilmelidir. Eğer bir ders hem teori hem de laboratuvar saatlerinden oluşuyorsa, dersi eklerken ders saat sayısı, teori ve laboratuvar saat sayısının toplamı olarak girilmeli, "Lab. dersi var mı?" sorusuna evet işaretlenip sağ tarafta beliren kutucuğa laboratuvar saat sayısı girilmelidir.

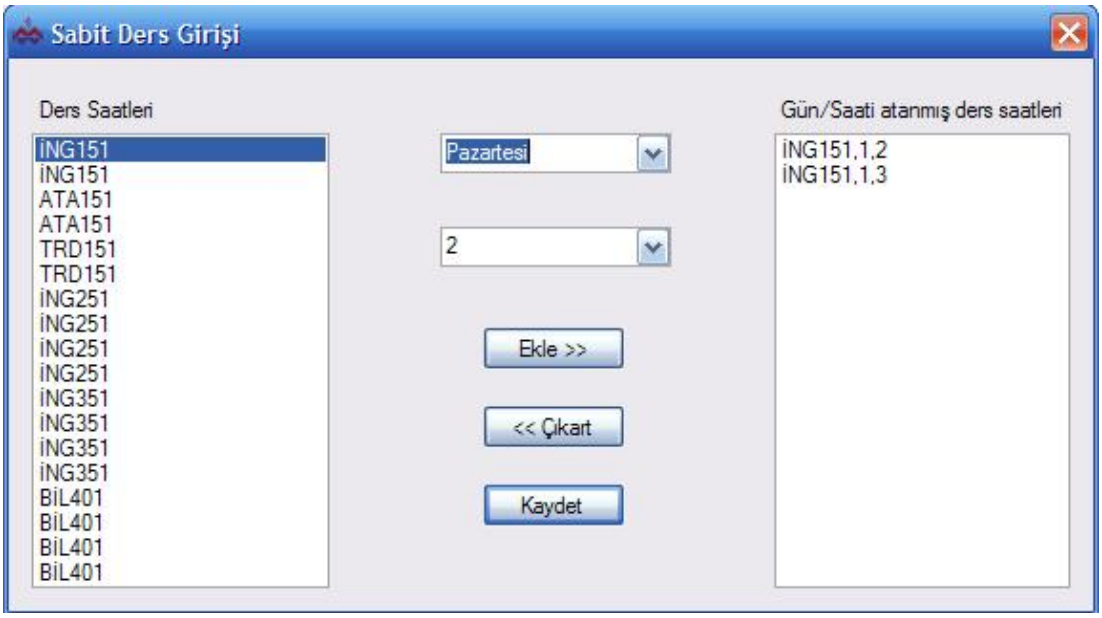


Şekil 20 - Ders giriş ekranı örnek ders girişi

Kısıtlar modülü 3 kısma ayrılmıştır. Bunlar; Sabit dersler, Hoca önerisi ve Çakışmayacak dersler. Bu kısıtlar (sezgiseller) bölümümüz tarafından belirlenmiştir.

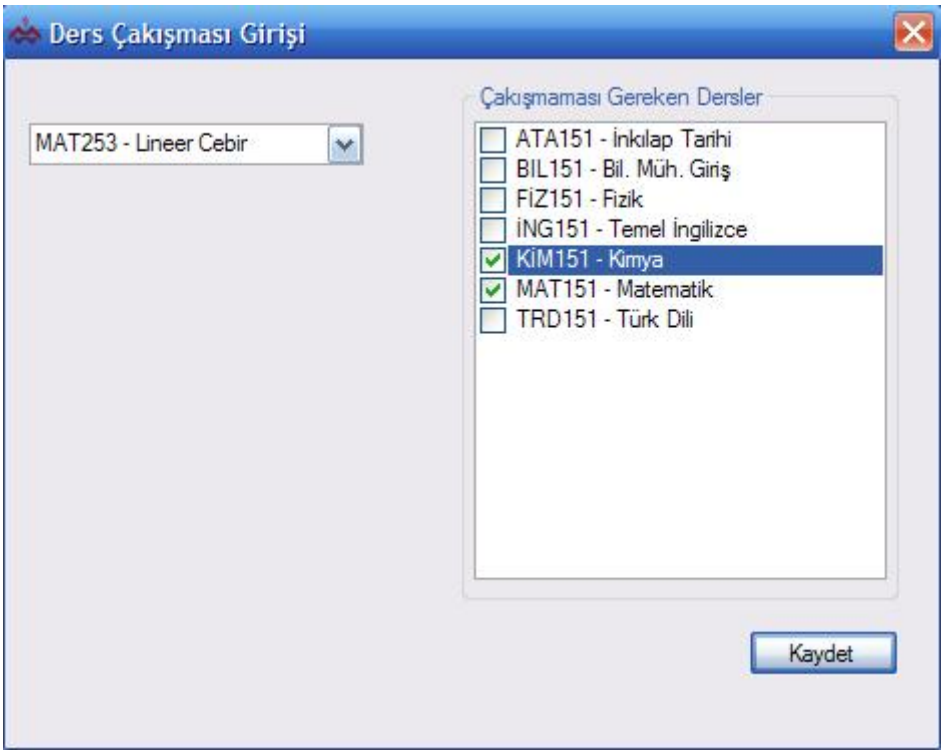
Şekil 21 – Programın kısıt seçenekleri

Graf renklendirme mantığındaki ağırlık sıralaması burada söz konusudur. En yüksek ağırlık sabit derslere aittir. Yani ders programı oluşturulurken ilk önce sabit dersler yerleştirilecektir. Sabit dersler programa girdi olarak verilirken dersin kodu ve dersin her bir saati için gün ve saat bilgileri girilir (Şekil 22).



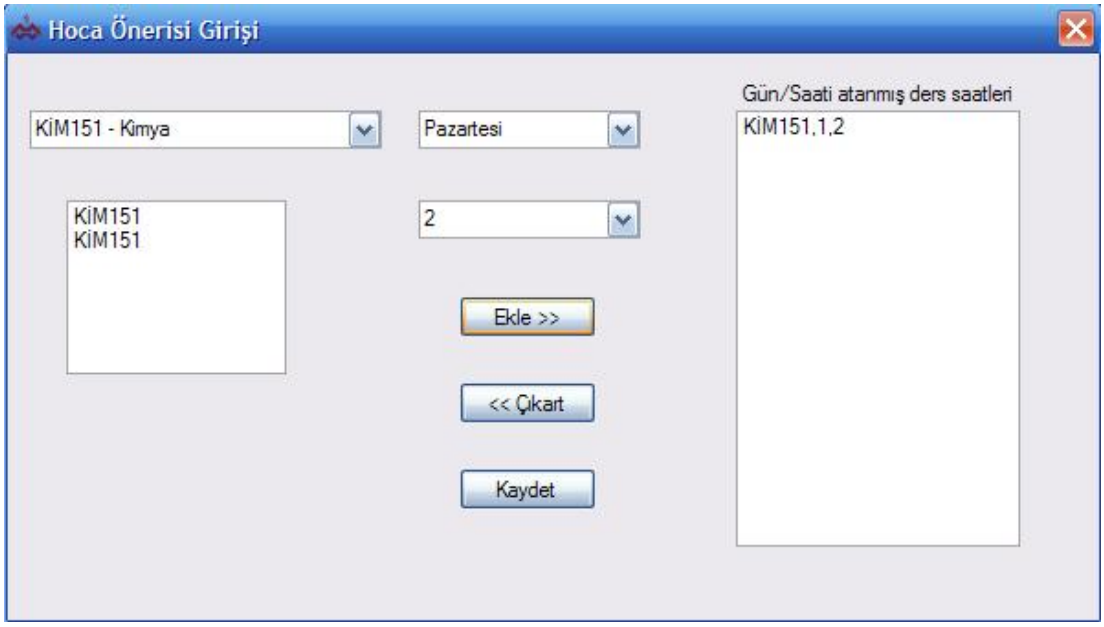
Şekil 22 - Sabit derslerin gün/saat ataması yapılıyor

İkinci derece ağırlıklı dersler, aynı saatte çakışmamaları istenen derslerdir. Sabit dersler yerleştikten sonra bu dersler uygun saat boşluklarına yerleştirileceklerdir. Programda bu kısıt için girdi yapılırken üst sınıf dersi seçilir, daha sonra sağ kısımdaki listeden alt sınıftan hangi dersle çakışmaması isteniyorsa o dersler seçilip kaydedilir (Şekil 23).



Şekil 23 - 2. sınıf Lineer cebir dersinin 1. sınıf Matematik ve Kimya dersleriyle aynı saatlerde çakışmaması isteniyor

Üçüncü derecede ağırlıklı dersler, dışarıdan görevlendirmeyle gelen hocaların dersleri için gün ve saat önerisi yapılan derslerdir. Bu tip dersler sabit ve çakışması olan derslerden sonra eğer önerilen gün ve saatlerde uygun saat boşluğu varsa yerleştirilecektir. Aksi halde gün ve saatlerde değişiklik yapılmak zorundadır. Giriş yapılan ekran, sabit ders giriş ekranına benzerdir (Şekil 24).



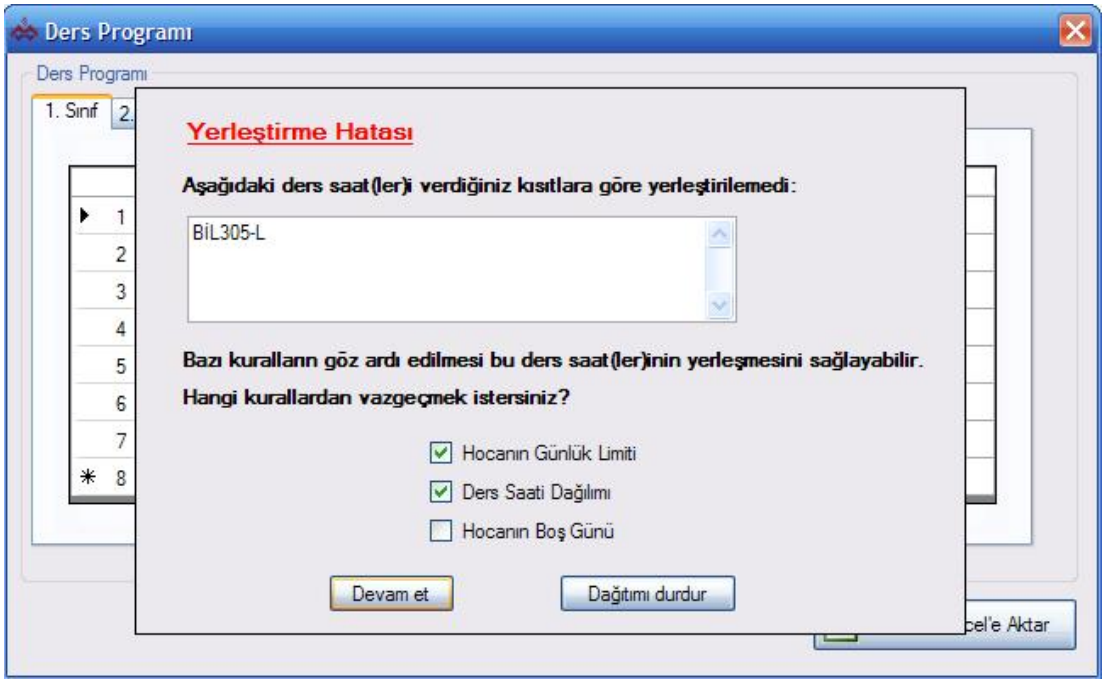
Şekil 24 - Hoca önerisi giriş ekranı

Programı oluştur modülünde ders programı hazırlama işlemi gerçekleşmektedir. Bu işlem sırasında kısıtlar doğrultusunda dersler yerleştirilirken şu durumlar göz önüne alınır;

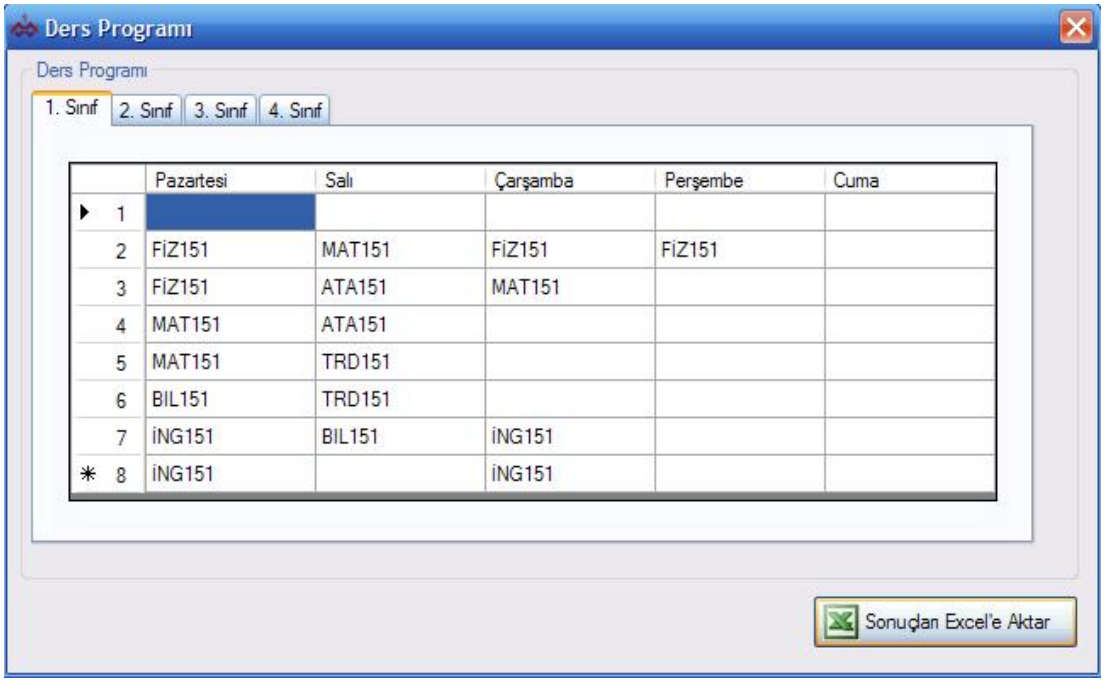
- Dersin hocasının önceden belirtilen o günde vereceği maksimum ders saati aşılmaz.
- Ders yerleşirken dersi veren hocanın o saatte başka sınıfta dersi olup olmadığı kontrol edilir.
- Yerleşecek dersin laboratuvar dersi de varsa teori ve laboratuvar saatleri aynı gün içerisinde olmaz.
- Laboratuvar ders saatleri art arda olur.
- Ders saatleri önceden belirtilen dağılım şekline göre yerleştirilmeye çalışılır.
- Ders yerleştirilirken dersi veren hocanın boş günü dikkate alınıp o günün herhangi bir saatine atama yapılmaz.

Verilen kısıtlarla bazı ders saatlerinin yukarıdaki durumlara göre ataması gerçekleşmeyebilir. Böyle bir durumda program yerleşmeyen ders saatlerini listeler ve eğer ki kullanıcı bu ders saatlerinin yerleşmesini isterse hangi kural veya kurallardan (Hocanın günlük ders saati limiti, ders saati dağılım şekli, hocanın boş

günü) vazgeçmek istediği sorulur (Şekil 25). Kullanıcı istediği kuralı iptal edip derslerin yerleşmesini sağlayabilir veya işlemi orada kesebilir. En sonunda oluşmuş program kullanıcının ekranında belirir (Şekil 26). Kullanıcı "Sonuçları Excel'e Aktar" düğmesine bastığı takdirde program, oluşmuş ders programını MS Excel ortamına aktarıp kaydedilmeye hazır hale getirir.



Şekil 25 - Yerleştirme hatası



Şekil 26 - Oluşturulmuş ders programı

6. SONUÇ

Zaman çizelgesi oluşturmanın genel açıklaması verilip, literatürdeki zaman çizelgesi oluşturma metotlarından kısaca bahsedildi. Bu metotlardan "Sıralı Yöntem" yani graf renklendirme üzerinde duruldu. Graf renklendirmeden önce kısaca graf konusuna değinildi ve daha sonra ders zaman çizelgesinin grafla oluşturulması üzerinde duruldu.

Üniversitemizin istediği doğrultuda belirlenen kısıtlar çerçevesinde "Sıralı Yöntem" mantığında ders zaman çizelgesi oluşturmak için uygulama programı geliştirilmiştir. Geliştirilen program kısıtlar doğrultusunda tatmin edici sonuçlar üretebilmektedir.

İleride Yapılacaklar

- Veriler MUBİS'in veritabanı yapısında DEPOS'un ihtiyacı olduğu şekilde tutulduğu takdirde iki program entegre edilebilecektir. Böylelikle DEPOS'un başlangıç safhasındaki hoca ve ders kayıtları MUBİS veritabanından otomatik oalrak alınabilecektir.
- Uygulama şuan bir anda 4 senelik tek bir bölüm için ders programı oluşturabilmektedir. İleride fakültedeki tüm bölümler için daha sonra da tüm üniversite için bir anda bütün ders programlarının oluşturulması sağlanacaktır.
- Uygulamanın MUBİS'e entegre edilmesi en fazla 2 hafta ve tüm fakültelerin bir anda ders programlarını oluşturmak için kodun geliştirilmesi de en fazla 3 hafta gibi bir zaman gerektirir.

KAYNAKLAR

1. Burke E. K., Petrovic S., "Recent Research Directions in Automated Timetabling", European Journal of Operational Research - EJOR, 140/2, 2002, 266-280.
2. Qu R., "Case-Based Reasoning For Course Timetabling Problems", Ph.D. Thesis submitted to the University of Nottingham, August 2002.
3. Soğukpınar İ., "Ayrık Matematik dersi ders notları, Graf Teorisi", Gebze Yüksek Teknoloji Enstitüsü, 80-90.
4. Grimaldi R. P., "Discrete and Combinatorial Mathematics: An Applied Introduction", ISBN: 9780201726343, 5, Addison-Wesley, 2004.
5. Gross J. L., Yellen J., "Handbook of Graph Theory", ISBN: 1-58488-090-2, 1, CRC Press, 2004.
6. https://cpi.utc.edu/oneweb/~Christopher-Mawata/petersen/lesson8.htm (05.07.2008)

ÖZGEÇMİŞ

Hüseyin Fehmi Selim BAYRAKLI, 1984 yılında İstanbul / Üsküdar'da doğdu. Öğrenimlerini sırasıyla Faik Reşit Unat İlkokulu ve Kartal Anadolu İmam Hatip Lisesi'nde tamamladı. 2002 yılında Maltepe Üniversitesi Mühendislik – Mimarlık Fakültesi Bilgisayar Mühendisliği Bölümünü kazandı ve 2006 yılında bölüm birincisi olarak mezun oldu. Yaz stajlarını, Radyan Bilgisayar (İstanbul, 2003 yaz, 2004 yaz) ve Amadeus Türkiye'de (İstanbul, 2005 yaz, 21 iş günü) yaptı. Ekim 2006'da, Maltepe Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Yüksek Lisans programında yüksek lisans öğrenimi yapmaya hak kazandı. Kasım 2006'da, halen çalışmalarını sürdürmekte olduğu Maltepe Üniversitesi Mühendislik Fakültesi Bilgisayar Mühendisliği Bölümü'ne Araştırma Görevlisi olarak atandı. Eylül 2007 – 2008 tarihleri arasında geliştirilen Maltepe Üniversitesi Uzaktan Eğitim Sistemi yazılım projesinde görev aldı.

EKLER

EK A: DEPOS Algoritması

Algoritmanın çalışması sırasında ilk işlem olarak günü/saati sabit olan yani 1. kısıttaki derslerin yerleştirilmesi yapılmaktadır. Bu işlem için "sabitDoldur" adındaki fonksiyon çalışmaktadır. Fonksiyonda veritabanından tipi "sabit" yani "0" olan dersler alınır ve gün/saat bilgisi doğrultusunda çizelgeye yerleştirilirler.

İkinci yapılan işlem bir veya daha fazla dersle çakışmaması gereken derslerin yerleştirilmesi yani 2 kısıttaki derslerin yerleştirilmesi yapılmaktadır. Bu işlem birçok alt fonksiyona bölünmüştür. Ana fonksiyon içerisinden sürekli bu alt fonksiyonlar çağırılmaktadır. Ana fonksiyonun adı "cakismaDoldur". Fonksiyonun başında veritabanındaki dersler sınıf sırasıyla alınır.

```
cmd.CommandText = "SELECT DISTINCT sinif FROM dersler ORDER BY
sinif"

conn.Open()

Dim drSinif As OleDbDataReader = cmd.ExecuteReader()
```

Daha sonra sınıf sayısı kadar bir döngüye girilir. Hemen ardından sabit dersler haricindeki ders tipleri sayısı uzunluğunda bir döngüye daha girilir. Bu döngü içerisinde kısıt önem sırasına göre dersler çağırılacaktır.

Kısıt türü 2 olan yani hoca önerisi olan derslere sıra geldiğinde "oneriDoldur" adındaki fonksiyon çağırılır. 3. tip derslerin yerleştirilmesi yine "cakismaDoldur" fonksiyonunda yapılır.

```
Do While drSinif.Read()

            For i = 1 To 3

                If i = 2 Then

                    oneriDoldur(drSinif(0))

                Else

                Dim cmdDersKodu As New OleDbCommand("SELECT
dersKodu FROM dersler WHERE tip = " & i & " AND sinif = " &
drSinif(0))
               cmdDersKodu.Connection = conn

               Dim drDersKodu As OleDbDataReader =
cmdDersKodu.ExecuteReader()

               Do While drDersKodu.Read()
                  For j = 0 To saatSayisiBul(drDersKodu(0)) - 1
```

Çağırılan ders sayısı uzunluğunda bir döngü daha açılır. Bu döngü açıldıktan hemen sonra yerleşecek olan dersin saat sayısı bulunup (bu işlem "saatSayisiBul" adlı fonksiyonla yapılmaktadır) bu sayı uzunluğunda bir döngüye daha girilir. Bu döngü içerisinde ders saatini yerleştirmek için, günün 1. ve 8. saatleri göz ardı edilerek bütün hafta içerisinde uygun saat boşluğu aranır (1. Ve 8. saatlerin göz ardı edilmesinin sebebi dersleri haftaya olabildiğince yaygın yerleştirmektir. İlk ana döngü bittikten sonra bir başka yerleştirme fonksiyonu olan "tekrarYerlestir" adlı fonksiyon çağırılarak, yerleşmeyen ders saatlerini, günün 1. ve 8. saatlerini göz önüne alarak yerleştirme işlemini gerçekleştirir. Diğer türlü algoritma, dersleri yerleştirmeye 1. gün ve 1. saatten başladığı için sonuçta çıkan ders programında haftanın ilk 3 gününe yığılım oluyor ve son 2 gün çok boş kalıyordu). Bu saat boşluğu aranırkenki kurallardan 5.2'de bahsetmiştik. Bu kuralları gerçekleyen fonksiyonlar şunlardır;

- **bosGunMu(hocaID,gun):** Hoca ve gün bilgisini alarak o günün hocanın boş günü olup olmadığını kontrol eder ve Doğru/Yanlış şeklinde geri bilgi gönderir.
- **hocaBul(dersKodu):** Ders kodu bilgisini alarak o dersin hangi hocaya ait olduğu bilgisini geri gönderir.

- **hocaGunlukLimit(hocaID):** Hoca bilgisini alarak o hocanın günlük maksimum vereceği ders saati bilgisini geri gönderir.

- **hocaGunlukYuk(hocaID,gun):** Hoca ve gün bilgisini alarak o hocanın o günkü ders saati limitini doldurup doldurmadığını kontrol eder ve o günkü vermiş olduğu ders saati toplamını geri gönderir.

- **ayniHocaMi(hocaID,saatBoslugu):** Hoca ve saat boşluğu bilgilerini alarak, eğer o saatte ders varsa o dersin o hocaya ait olup olmadığı bilgisini Doğru/Yanlış şeklinde geri gönderir.

- **dersKontorlu(gun, saat, sinif):** Gün, saat ve sınıf bilgisini alarak o sınıfta o gün ve saatte ders olup olmadığı bilgisini gönderir. Eğer ders varsa, var olan dersin ders kodunu gönderir. Yoksa string olarak "YOK" gönderir.

- **ayniSaatteDersKontrol(hocaID,gun,saat,sinif):** Hoca, gün, saat ve sınıf bilgilerini alarak o hocanın, o gün ve saatte başka sınıfta dersi olup olmadığı bilgisini Doğru/Yanlış şeklinde geri gönderir.

- **cakismaKontrol(dersKodu,gun,saat):** Ders kodu, gün ve saat bilgisini alarak o dersin o gün ve saatte başka dersle çakışma durumu olup olmadığı bilgisini Doğru/Yanlış şeklinde geri gönderir.

- **gundekiDersSaatToplami(dersKodu,gun,sinif):** Ders kodu, gün ve sınıf bilgisini alarak o dersin o gün kaç saatinin yerleştiğini hesaplayıp geri gönderir.

- **dagilimKontrolu(dersKodu,gun,saat,sinif):** Ders kodu, gün, saat ve sınıf bilgisini alarak o ders saatlerinin o gün ve saatte haftaya dağılım şekline uygun olarak yerleştirme yapılıp yapılamayacağı bilgisini Doğru/Yanlış olarak geri gönderir.

- **genelSayacKontrol(dersKodu,sinif,dagilim):** Ders kodu, sınıf ve dersin dağılım bilgisini alarak, derse ait kaç saatin yerleştiğine bakarak dağılımın takip edilmesini sağlar.

- **blokArama(dersKodu,gun,saat,sinif):** Ders kodu, gün, saat ve sınıf bilgisini alarak o sınıftaki dersin o gün ve saatte dağılım durumuna uygun saat boşlukları olup olmadığını kontrol edip Doğru/Yanlış olarak geri gönderir.

"cakismaDoldur" fonksiyonu, kısıtlara göre dersleri yerleştirirken bazı ders saatleri için verilen kurallara göre uygun saat boşluğu bulunamayabilir. Böyle bir durumda döngü içerisindeyken tüm hafta saat boşlukları tarandıktan sonra yerleştirilemeyen ders saatinin kodu "yerlesmeyenDersler" adında string tipinde bir diziye eklendir.

```
If gun = 4 And saat = 6 And yerlesti = False Then
ReDim Preserve yerlesmeyenDersler(yerlesmeyenDersler.Length)
yerlesmeyenDersler(yerlesmeyenDersler.Length - 1) = drDersKodu(0)
End If
```

İlk yerleştirme işlemi tamamlandıktan sonra yine 5.2 de bahsedilen, kullanıcının vereceği direktif doğrultusunda "tekrarYerlestir" fonksiyonu çağırılarak yerleşmeyen ders saat(ler)ini, iptal edilen kuralları göz ardı edip uygun saat boşluğuna atamaya çalışır.