PAMUKKALE ÜNİVERSİTESİ FEN BİLİMLERİ ENSTİTÜSÜ

EVRİMSEL HESAPLAMA TEKNİĞİ KULLANILARAK OTOMATİK SINAV PROGRAMI OLUŞTURMA

YÜKSEK LİSANS TEZİ

Ceyda BAYSAL

Anabilim Dalı : Bilgisayar Mühendisliği

Programı: Bilgisayar Mühendisliği

Ocak 2011

YÜKSEK LİSANS TEZ ONAY FORMU

Pamukkale Üniversitesi Fen Bilimleri Enstitüsü 081281005 nolu öğrecisi Ceyda Baysal tarafından hazırlanan "**EVRİMSEL HESAPLAMA TEKNİĞİ KULLANILARAK OTOMATİK SINAV PROGRAMI OLUŞTURMA**" başlıklı tez tarafımızdan okunmuş, kapsamı ve niteliği açısından bir Yüksek Lisans tezi olarak kabul edilmiştir.

Tez Danışmanı : (Jüri Başkanı) **Doç. Dr. Abdullah T. TOLA (PAÜ)**

Jüri Üyesi : **Doç. Dr. Sezai TOKAT (PAÜ)**

Jüri Üyesi : **Yrd. Doç. Dr. Gürhan GÜNDÜZ (PAÜ)**

Pamukkale Üniversitesi Fen Bilimleri Enstitüsü Yönetim Kurulu'nun 02.07.2014 tarih ve ...07/13.... sayılı kararıyla onaylanmıştır.

Prof. Dr. Nuri KOLSUZ
Müdür

Bu tezin tasarımı, hazırlanması, yürütülmesi, araştırmalarının yapılması ve bulgularının analizlerinde bilimsel etiğe ve akademik kurallara özenle riayet edildiğini; bu çalışmanın doğrudan birincil ürünü olmayan bulguların, verilerin ve materyallerin bilimsel etiğe uygun olarak kaynak gösterildiğini ve alıntı yapılan çalışmalara atfedildiğine beyan ederim.

İmza : 

Öğrenci Adı Soyadı : Ceyda BAYSAL

TEŞEKKÜR

Bu çalışmanın gerçeklenmesinde katkıda bulunan Tez Danışmanım ve Pamukkale Üniversitesi Bilgi İşlem Dairesi Başkanı Doç. Dr. Abdullah T. TOLA'ya, PAÜ Bilgisayar Mühendisliği Bölümü Öğretim Üyelerinden Yrd. Doç. Dr. A. Kadir YALDIR ve Yrd. Doç. Dr. Gürhan GÜNDÜZ'e teşekkür ederim.

Ayrıca bu çalışmamı sonuçlandırmam sırasında verdikleri desteklerle bana yardımcı olan aileme ve Çağdaş Sevinç'e teşekkürlerimi bir borç bilirim.

İÇİNDEKİLER

Sayfa

KISALTMALAR

GA **:** Genetik Algoritma

GP **:** Genetik Programlama

EH **:** Evrimsel Hesaplama

EA **:** Evrimsel Algoritma

EP **:** Evrimsel Programlama

f **:** Uygunluk fonksiyonu

OLAP **:** Online Analytical Processing (Çevrimiçi Analitik İşleme)

OLTP **:** Online Transaction Processing (Çevrimiçi Hareket İşleme)

MSSQL **:** Microsoft Structured Query Language (Microsoft Yapılandırılmış Sorgu Dili)

XML **:** Extensible Markup Language (Genişletilebilir İşaretleme Dili)

AJAX **:** Asynchronous JavaScript and XML (Eşzamansız JavaScript ve XML)

LINQ : Language Integrated Query (Dil ile Bütünleşik Sorgu)

TABLO LİSTESİ

Tablolar

ŞEKİL LİSTESİ

Şekiller

ÖZET

EVRİMSEL HESAPLAMA TEKNİĞİ KULLANILARAK SINAV TAKVİMİ OTOMASYONU UYGULAMASI

BAYSAL, Ceyda
Yüksek Lisans Tezi, Bilgisayar Mühendisliği ABD
Tez Yöneticisi: Doç. Dr. Abdullah T. TOLA

Ocak 2011

Bu çalışma ile belirli dönemlerde manuel olarak hazırlanan sınav takvimi uygulaması otomasyon kapsamına alınmıştır. Uygulamada evrimsel hesaplama yöntemi kullanılmış, gerekli kontroller üniversite yönetiminin isteği doğrultusunda yapılmıştır. Uygulama iki bölümden oluşmaktadır. Birinci bölümde veriler ilgili birimlerden web ortamında toplanmakta, ikinci bölümde ise geliştirilen masaüstü uygulama çalıştırılarak istenen sonuçlar alınmaktadır.

Uygulama ile birlikte sınav takvimi hazırlamak için harcanan zamandan kazanılacak, ayrıca uygulama, kullanımı sırasında istenen veriler dışındaki verileri doğrudan gerçek zamanlı olarak üniversite veritabanından almakta olduğu için; alttan dersi kalan ya da üstten ders alan öğrenci takibi gibi sınav çakışması sorunları da tamamen ortadan kalkacaktır.

Çalışmada sisteme eklenen tablolar detaylarıyla açıklanmakta, ayrıca eldeki verilerin genetik algoritma yöntemine uygulanma aşamaları verilmektedir. Uygulama aşamasında kullanılan kısıtlar da programın önemli bir parçasını oluşturmaktadır. Bunun dışında programın ara yüzleri örnekleriyle birlikte yer almaktadır. Sonuç olarak, oluşturulmuş sınav programı takviminden bir örnek çıktıya da yer verilmiştir.

Anahtar Kelimeler: Sınav Takvimi Otomasyonu, Genetik Algoritma

SUMMARY

DEVELOPING EXAMINATION SCHEDULING AUTOMATION BY USING EVOLUTIONARY COMPUTING TECHNIQUES

BAYSAL, Ceyda
M. Sc. Thesis in Computer Engineering
Supervisor: Assistant Prof. Dr. Abdullah T. TOLA

January 2011

In this study, manually (paperwork) and periodically prepared exam scheduling application has been taken into a computer automation system by developing a software solution. In the software developed by the authors, Evolutionary Algorithm method has been applied and university administration's specific requests have been taken into consideration while developing the software. The developed software has two parts: First part is about collecting data through the web application, and the second part is the application project, which calculates the final scheduling results.

By utilizing the software, a considerable amount of time lost by manually preparing exam schedules will be saved. In addition, with the real-time connection to the campus database system, numerous problems will be vanished, i.e. students/classes exam scheduling conflicts, etc.

In the study, tables added to system have been explained in detail. Additionally, by applying the genetic algorithm methods to the various parts of data have been examined along with the constraints used in the application, which are essential parts of the software. User interfaces have been designed with their sample instances. Finally, exam-scheduling table has been created and an example output of the schedule has been generated in the study.

Keyword: Developing Exam Scheduling Automation, Genetic Algorithm

1. GİRİŞ

Sınav takvimleri, çoğu üniversitede akademik veya idari personel tarafından manuel olarak hazırlanmakta, bunun sonucu olarak da çok fazla hatalar, bu hatalara bağlı olarak defalarca düzeltmeler yapılmaktadır. Bu işlem, en az vakit alacak şekilde tasarlanıp, hatasız sonuçlar döndürecek uygulamalar sayesinde otomasyon sistemleri ile yapılmak istenmektedir.

1.1 Tezin Amacı

Bu tezde, manuel sınav programı hazırlamakla kaybedilen zamanı en aza indirmek temel amaçtır. Bunun yanında olası çakışmaları yerleştirme sırasında minimize ederek, sonradan sınav tarihleri üzerinde oynamaları engellemek de her açıdan kazanç sağlayacaktır. Başka bir açıdan da bakacak olursak takvimin otomasyon sistemi ile hazırlanması sebebiyle insan ilişkilerinde gereksiz memnuniyetsizlik bildirimlerinin de önüne geçilecektir. Otomasyon sistemi kapsamında geliştirilen uygulama kendi içinde zaten kabul edilemez hataları ortadan kaldırmaktadır. Geriye kalan ve lükse giren istekler uygulama tarafından karşılanmayacaktır. Karşılanmayacak bu istekler için öğretim elemanının zorunluluk dışı istemediği sınav zamanını seçebilmesi, sınavlarını bir güne toplamak istemesi gibi özel istekleri örneklendirebiliriz. Tez çalışması bu özellikler doğrultusunda oluşturulup çalışır duruma getirilmiştir.

1.2 Literatür Özeti

Genetik Algoritma (GA), yapay zekânın gittikçe genişleyen bir kolu olan Evrimsel Hesaplama (EH) tekniğinin bir parçasını oluşturmaktadır. GA, doğada en iyinin yaşaması kuralından esinlenerek oluşturulan, bir veri öbeğinden özel bir veriyi bulmak için kullanılan bir arama yöntemidir. GA, geleneksel yöntemlerle çözümü zor veya imkânsız olan problemlerin çözümünde kullanılmaktadır. Herhangi bir problemin GA ile çözümü, problemi sanal olarak evrimden geçirmek suretiyle yapılmaktadır [1]. EH ilk olarak 1960'larda I. Rechenberg tarafından "Evrim

Stratejileri (Evolutionsstrategie)" isimli eserinde tanıtılmıştır [2]. Onun fikri daha sonra başka araştırmacıların da ilgisini çekmiş ve geliştirilmiştir. John Holland evrim sürecinin bir bilgisayar kullanılarak, bilgisayara anlayamadığı çözüm yöntemlerinin öğretilebileceğini düşündü. EH böylece John Holland tarafından bu düşüncenin bir sonucu olarak bulundu. Onun öğrencileri ve arkadaşları tarafından geliştirildi ve bu sayede Holland'ın kitabı "Doğal ve Yapay Sistemlerde Adaptasyon (Adaption in Natural and Artificial Systems)" 1975 yılında yayınlandı. Takip eden yıllarda GA yaygınlaştı, 1992'de ise ilk uygulama dili sayılan LISP ile GP ortaya çıktı [3].

Robot uygulamaları, başarılı çözüm teknikleri gerektiren birçok problem ortaya çıkarır. EH de bu tip problemlerin bir kısmında başarılı olan bir teknikler grubudur. Bu terim, arama, optimizasyon ve öğrenme gibi problemlere biyolojik popülasyon genetiği kurallarına dayanarak yaklaşan hesaplama tekniklerini içerir. EH formülize edilişine göre GA, Evrimsel Programlama (EP), Evrim Stratejileri ve GP gibi değişik isimlerle anılırlar. Geleneksel arama metotları, probleme bir çözüm adayı önerir ve onu değiştirerek daha iyi çözümler elde etmeye çalışır. Aksine EH'de ise, bir çözüm adayları popülasyonu oluşturulur ve bu popülasyon zamanla evrimleşir. Bir adayın çözüme ne kadar yakın olduğu, uygulamaya bağlı bir fonksiyondur. Bir çözüm adayı bir parametreler topluluğunu, bir kuralı, bir kurallar grubunu veya ağaç yapısında bir bilgisayar programını temsil edebilir. Hepsinde de, algoritma her adayın ne kadar güçlü olduğunu hesaplar ve buna göre bir sonraki neslin ebeveynleri olacak ya da yok olacak bireyleri belirler. Daha sonra, makul bir yeni nesil oluşturmak için ebeveynlere genetik arama işlemcilerini (yeniden yapılanma ve mutasyon) uygular. Bu döngü her defasında daha güçlü bireyler oluşturarak tekrarlanır. Belirli bir probleme Evrimsel Algoritma (EA) uygulayabilmek için, kullanıcı, aday çözümler için bir temsil şekli, bir uygunluk hesaplama yöntemi ve genetik arama işlemcilerini belirlemelidir. Çoğu zaman, bunları belirlemek büyük bir çaba gerektirir. Mutasyon ve yeniden yapılanma için tanımlanan işlemcilerle kullanılan temsil şekli uyum içinde olmalıdır. EA, çözüm adayları içinde zeki bir rastgele arama yapar. Analiz zor problemlerde verimli olmasına rağmen, basit sezgisel arama veya ayrıntılı numaralama yöntemleriyle çözülebilen kolay problemlerde seçilmemelidir [4].

Bu çalışma ile üniversitelerde uygulanan sınav programlarının GA yöntemiyle çözümlenmesi amaçlanmıştır. Genel olarak bu soruna zaman çizelgeleme problemi denilebilir. Zaman çizelgeleme problemleri için geliştirilen modeller genelde

birbirinden oldukça farklıdır. Bu sebeple literatürde bulunan sonuçları karşılaştırmak zordur. Son yıllarda, birçok araştırmacı çözüm yaklaşımlarını Tavlama Benzetimi (Simulated Annealing), Tabu Arama (Tabu Search) ve GA gibi yapay zekâ yöntemlerine dayandırmaktadır. Bu çalışmalardan farklı olarak yeni bir yaklaşım olan ve probleme bağımlı olmayan üst-sezgisel(hyper-heuristics) algoritmalar da kullanılmıştır. Yeditepe Üniversitesi'nde yapılan bir çalışma zaman çizelgeleme problemlerinden ders çizelgeleme problemini üst-sezgisel algoritmalar yöntemiyle çözüme ulaştırmıştır [5].

Daha önce tanımlanan, Yeditepe Üniversitesi'nde yapılan çalışmada sert ve yumuşak kısıtların (yerine getirilmesi zorunlu olan ve olmayan) her biri için bir sezgisel yazılmış, bu sezgisellerin her biri sadece belirli bir kısıtı düzeltmeye çalışmıştır. Problemin performans değerini etkileyen belirli bir ağırlığı bulunmaktadır. Sert kısıtlar daha önemli olduğu için yumuşak kısıtlara göre ağırlık değerleri fazladır. Böylece, öncelikle sert kısıtlamaların düzeltilmesi sağlanmış, yumuşak kısıtlamalar da minimuma indirilmiştir. Testlerde bütün kısıtların bulunduğu çözüm oranı, en iyi çözüm değerleri ortalamaları ve en iyi zamanlama değerleri ortalamaları deneylerde performans ölçütü olarak kullanılarak çözüme ulaşılmıştır.

Bu tarz zaman çizelgeleme problem çözümü örneklerinden bir başkası da Süleyman Demirel Üniversitesi'nde yapılan "Hemşire Çizelgeleme Problemlerinin Genetik Algoritmayla Çözümü" adlı çalışmadır. Bu çalışma ile Süleyman Demirel Üniversitesi Araştırma ve Uygulama Hastanesi'nde çalışan hemşirelerle yapılan görüşmeler sonucu problem kısıtları belirlenmiş ve aylık hemşire nöbet çizelgeleri bu kıstaslar üzerinden oluşturulmuştur [4].

Bu gibi problemlerin çözümünde kullanılan diğer yaklaşımlar; Tavlama Benzetimi ve Tabu Arama'dır. Tavlama benzetimi zor problemlerinin çözümünde iyi performans gösteren sezgisel bir yöntemdir. Tavlama benzetimi algoritması, pek çok değişkene sahip fonksiyonların en büyük veya en küçük değerlerinin bulunması ve özellikle pek çok yerel en küçük değere sahip doğrusal olmayan fonksiyonların en küçük değerlerinin bulunması için tasarlanmıştır. Yerel arama teknikleri, mümkün çözümlerin sadece küçük bir kısmıyla, birçok problemi en iyi şekilde çözme veya en iyi çözümü verme yeteneğine sahiptir. Burada mümkün çözümlerden kasıt mevcut çözümün komşularının araştırılmasıdır. Bu esnada karşılaşılan problemlerden birisi yerel optimum çözüme takılmadır. Birçok arama tekniğinde, incelenecek komşu

seçimi belli bir olasılıkla rastgele seçildiğinden dolayı yerel optimumdan kaçma imkânı bulunabilir. Fakat incelenmiş çözümlere tekrar geri dönme olasılığı olduğundan arama yerel optimum civarında takılabilir. Bu aşırı zaman kaybına sebep olur. Bu yüzden daha önceden incelenmiş belli sayıda çözüm bir listede tutulur. Bu listede yer alan çözümler tekrar hesaplamaya katılmadığından aramanın tekrarlanması mümkün değildir. İşte böyle listelerin oluşturulduğu ve 'tabu listesi' adı verilen arama yaklaşımına tabu arama algoritması denir.

Çizelgeleme problemlerinin farklı yöntemlerle çözümü bu şekildedir. GA'nın kullanıldığı diğer çalışmalardan birine örnek olarak Başkent Üniversitesi'nde yapılan "Genetik Algoritma Kullanarak Tel Anten Tasarımı" adlı tez çalışması verilebilir. Bu çalışmada GA ile çeşitli tel anten tasarımları yapılmıştır. GA ile tel anten tasarımında tasarımcı istenen elektromanyetik özellikleri belirler, problemin sınırlarını çizer ve algoritma en uygun çözümü bulur. Tasarımlar için MATLAB® ortamında GA yazılımları geliştirilmiştir. Antenlerin performansını belirlemek için moment yöntemi ile anten analizi yapan SuperNEC adlı elektromanyetik benzetim programı kullanılmıştır. GA, anten parametrelerini SuperNEC'e gönderir. SuperNEC her bir antenin benzetimini gerçekleştirir ve anten performanslarını belirleyen benzetim sonuçlarını bir dosyaya yerleştirir. GA, bu dosya içerisinden gerekli parametre değerlerini alır ve değerlendirir. Bu çalışmada, geliştirilen GA yazılımı ile bükük tel anten, karıştırıcı tel anten ve ultra-geniş bantlı tel anten tasarımları yapılmıştır [6].

1.3 Kullanılan Teknolojiler

Çalışmada veritabanı modeli Microsoft SQL Server 2008 üzerinde oluşturulmuş, kodlama ise Microsoft Visual Studio 2008 üzerinde geliştirilmiştir. Proje ASP.NET ile yazılmış içerisinde LINQ ve AJAX gibi yazılım teknolojileri kullanılmıştır.

1.3.1 Microsoft SQL Server

MS SQL Server, iki tür veritabanını yönetmek için kullanılır. Bunlar OLTP (Çevrimiçi Hareket İşleme) ve OLAP (Çevrimiçi Analitik İşleme) veritabanlarıdır. Genel olarak farklı istemciler ağ üzerinden haberleşerek veritabanlarına erişirler. MS SQL Server ile yoğun veriler işlenebilir, saklanıp analiz edilebilir ve yeni uygulamalar geliştirilebilir. MS SQL Server OLTP ve OLAP için gerekli olan veri

saklama ürünlerini ve teknolojilerini destekler. MS SQL Server aynı zamanda ilişkisel bir veritabanı yönetim sistemidir [7].

OLTP Veritabanları: Bir OLTP veritabanı modeli içinde veriler genellikle ilişkisel tablolar içinde organize edilir. Gereksiz veri yığınlarını azaltır ve veri güncelleme hızını arttırır. MS SQL Server çok sayıda kullanıcının gerçek zamanlı olarak veri analiz edebilmesini ve güncellemesini sağlar. Örnek olarak OLTP veritabanları havayolu bilet satış bilgileri ve bankacılık işlemlerini içerir [7].

OLAP Veritabanları: OLAP teknolojisi büyük verilerin organize edilmesini ve incelenmesini sağlar. Örneğin bir analist büyük verileri hızlı ve gerçek zamanlı olarak değerlendirebilir. MS SQL Server Analiz Servisi toplu raporlama ve analizde, veri modelleme ve karar desteğe kadar geniş alanda çözümler sunar [7].

Kullanıcılar SQL Server ve Analiz Servisine doğrudan ulaşamaz; verilere erişmek için yazılmış istemci uygulamaları kullanırlar. Bu uygulamalara örnek olarak Transact_SQL verilebilir[7].

Transact-SQL: T-SQL bir sorgulama dili olup, SQL'in farklı bir versiyonudur. MS SQL Server kullanıcıları için birincil bir sorgulama ve programlama dilidir.

XML: Bu format bir sorgu ve prosedürün çalışması sonucu gelen verinin http üzerinden URL (Birörnek Kaynak Konumlayıcı) veya şablonlar kullanılarak iletilmesidir. XML'i veritabanına veri girerken, güncellerken ve silerken kullanılabilir.

MS SQL Server aşağıdaki servisleri içerir;

MSSQL SERVER service, SQLServerAgent service, Microsoft Distributed Transaction Coordinator (MS DTC) ve Microsoft Search.

MSSQL SERVER Service: MSSQL SERVER bir veritabanı motorudur. Tüm T-SQL yapılarını çalıştıran ve veritabanını kapsayan tüm dosyaları yöneten servistir. MSSQL SERVER servisi;

1. Sistem kaynaklarını birden fazla kullanıcıya paylaştırır.
2. Mantıksal hataları engeller. Mesela bir veriyi aynı anda güncellemek isteyen kişileri engeller.
3. Veri bütünlüğünü sağlar.

SQLServerAgent Service: Bu servis MS SQL Server ile birleşik olarak çalışır ve alert'leri ve multiserver işlemlerin yönetilmesini sağlar [7].

1. Alert'ler bir işlemin sonuçları hakkında bilgi verir. Örneğin "bir sorgu bitti" veya "çalışma sırasında bazı hatalarla karşılaşıldı" gibi.
2. SQLServerAgent görev oluşturma ve zamanlama aracı ile bazı işlemlerin otomatikleştirilmesini sağlar.
3. SQLServerAgent servisi bir problem olduğunda e-posta atabilir, çağrı cihazına mesaj gönderebilir veya başka bir uygulamayı çalıştırabilir. Örneğin bir veritabanı dolduğunda veya bir yedekleme işlemi bittiğinde bir kişiye e-posta atması sağlanabilir.

Microsoft Distributed Transaction Coordinator: MS DTC bir işlem ile birden fazla farklı kaynağın üzerinde işlem yapılmasını sağlar. Mesela bir işlem ile tüm sunucular üzerinde kalıcı bir güncelleme işlemi yapabilir veya yapılmış bu işlemi hepsinden geri alabilir.

Microsoft Search: Microsoft Search Windows 2000 üzerinde bir servis olarak çalışan full-text (zengin içerik arama) bir arama motorudur.

SQL Server Entegrasyonu: SQL Server, Microsoft işletim sistemi ve diğer sunucu uygulamaları ile bütünleşik çalışabilen istemci-sunucu bileşenlerine sahiptir. Farklı işletim sistemleri üzerinde bulunan Internet tarayıcıları ve diğer üçüncü parti yazılımlar SQL Server'a erişebilmektedir [7].

1.3.2 Microsoft Visual Studio

Microsoft Visual Studio, Microsoft tarafından bir takım programlama dillerini destekleyen bir tümleşik geliştirme ortamıdır. İçerisinde Visual Basic, Visual C#, Visual C++ gibi programlama dillerinin tümleşik geliştirme ortamlarını ve Microsoft tarafından özelleştirilmiş kütüphaneleri barındırmaktadır [8].

1.3.3 ASP.NET

ASP (Active Server Page), sayfa uzantısı ".asp" olan asp.dll isimli ISAPI (İnternet Server Application Programming Interface) yorumlayıcı tarafından yorumlanır ve kodla istenilen işlemlerin gerçekleştirilmesinden sonra sunucunun istemciye göndermesi mantığı ile çalışır.

ASP.NET Microsoft tarafından tasarlanan web uygulama dilidir. Programcılar ASP.NET kullanarak dinamik web siteleri, web uygulamaları ve XML web servisleri geliştirebilirler. ASP.NET, .NET platformunun bir parçasıdır ve ASP'nin devamı olarak nitelendirilmektedir. ASP.NET Ortak Dil Çalışma Zamanı üzerine inşa edilmiştir. Bu demektir ki programcılar herhangi bir Microsoft.NET dilini kullanarak ASP.NET kodu üretebilirler. aspx uzantısı ASP.NET de programlanan web sitelerinin uzantısıdır. Eğer bir web sayfasının uzantısı aspx ise bu demektir ki, bu web sitesi ASP.NET kullanarak tasarlanmıştır. aspx dosyası içerisinde <% -- dinamik kod --%> PHP, JSP ve ASP'de de olduğu gibi doğrudan sayfa üzerinde ASP.NET kodları da yazılabilir [9].

1.3.4 LINQ

Veritabanı nesnelerinin programlama ortamında sınıf gibi tipler ve metot benzeri üyeler ile ifade ediliyor olması, bu tiplere ait nesne örnekleri üzerinden sorgulamalar yapılabilmesi ihtiyacını da ortaya çıkartmıştır. Bir veritabanı nesnesinin programlama ortamındaki karşılığının nesne yönelimli bir dilde geliştirilmesi son derece kolaydır. LINQ (**L**anguage **In**tegrated **Q**uery) mimarisi de, temel anlamda programatik tarafta yazılan ifadeleri arka planda metotlar ve temsilciler yardımıyla kurduğu bir modele dönüştürmektedir. LINQ'in kullanıldığı alanlar göz önüne alındığında en popüler seçeneklerden birisi de LINQ to SQL mimarisidir. LINQ to SQL mimarisi ile varlık tipleri üzerinden sorgular çalıştırılabilir. Basit anlamda, nesneler üzerinde uygulanabilen LINQ sorguları, SQL tarafına ulaştıklarında ise bildiğimiz sorgu ifadelerine dönüşmektedir [10].

1.3.5 AJAX

AJAX (**A**synchronous **J**avaScript **A**nd **X**ML), Internet sayfalarında JavaScript ve XMLHttpRequest kullanımı ile etkileşimli uygulamalar yaratan tekniğin adıdır. En yaygın kullanım alanı, sayfayı yeniden yüklemeye gerek kalmaksızın, sayfada görünür değişiklikler yapmaktır. XMLHttpRequest kullanılarak birden fazla bağımsız işlem yapılabilir. Klasik bir istemci-sunucu uygulamasında kullanıcı tarafından yapılan her işlem sunucuya http Request olarak gönderilir, yorumlanır ve veriler işletildikten sonra sonuçlar istemciye gönderilir. Bu da çok büyük bir ağ trafiğine neden olur. AJAX kullanıldığında ise istekler HttpRequest yerine XMLHttpRequest olarak gönderilir. Burada sunucu ve istemci arasında taşınan

veriler sıkıştırılmış XML formatındadır. Bu sıkıştırılmış yapı istemci tarafında açılır böylece sunucu ve istemci arasındaki bant genişliği boş yere işgal edilmemiş olur [11].

2. bölümde GA hakkında genel bilgiler verilmiştir. 3. bölümde veritabanı oluşumu, var olan tablolar ile buna eklenen tablolar anlatılmaktadır. Bu tablolar alanlarıyla birlikte şekillerde gösterilmiştir. 4. bölümde program için veri toplama ölçütleri; programın çalışması için gerekli olan veri girişi ekranları, programın çalışma prensibi; programın EA'yı hangi yönde nasıl kullandığı, bilgilerine ulaşılabilir. Son olarak 5. Bölümde ise program ara yüz görüntüleri ile sonuç bilgi ve raporlarına yer verilecektir.

2. GENETİK ALGORİTMALAR

2.1 Genetik Algoritmalara Giriş

GA, doğada gözlemlenen evrimsel sürece benzer bir şekilde çalışan arama ve eniyileme yöntemleridir. Karmaşık çok boyutlu arama uzayında en iyinin hayatta kalması ilkesine göre bütünsel en iyi çözümü arar. GA, problemlere tek bir çözüm üretmek yerine farklı çözümlerden oluşan bir çözüm kümesi üretir. Böylelikle, arama uzayında aynı anda birçok nokta değerlendirilmekte ve sonuçta bütünsel çözüme ulaşma olasılığı yükselmektedir. Çözüm kümesindeki çözümler birbirinden tamamen bağımsızdır. Her biri çok boyutlu uzay üzerinde bir vektördür [12].

GA'nın temel çalışma prensibi Darwin'in 'Doğal Seçim' ilkesine dayanır. Darwin, "Türlerin Kökeni" adlı yapıtında iki varsayımı ortaya atmıştır [13]:

1.Tüm organizmalar, gereğinden fazla yavru meydana getirme yeteneğine sahiptirler. Bununla beraber elemine edilenler ile popülasyonda denge sağlanmaktadır.

2.Bir tür içerisindeki bireyler, kalıtsal özellikleri bakımından farklıdırlar.

GA, doğadaki canlıların geçirdiği evrim sürecini ele alır. Amaç, doğal sistemlerin uyum sağlama özelliğini dikkate alarak, yapay sistemler tasarlamaktır. GA'da tasarlanan yapay sistemde ele alınan en önemli faktör ise sağlamlıktır. Yapay sistemler, doğal sistemler kadar sağlam olabilse, mevcut sistemler faaliyetlerini daha uzun zaman sürdürecekler ve pahalı olan yeniden tasarlama ve uyarlama işlemleri ortadan kalkacaktır. GA konusundaki esas gelişim ise, John Holland'ın doktora öğrencisi David E. Goldberg tarafından 1985 yılında hazırlanan "Gaz Boru Hatlarının Genetik Algoritma Kullanılarak Denetlenmesi" konulu tez ile sağlanmıştır. Bu ilk uygulamadan sonra Goldberg'in 1989 yılında yayımladığı "Makine Öğrenmesi, Arama ve Optimizasyon İçin Genetik Algoritma" adlı kitabı, GA'ya yeni bir boyut kazandırmış ve günümüzde bile GA konusunda en kapsamlı referans olma özelliğini korumuştur [14].

GA, problemlerin çözümü için evrimsel süreci bilgisayar ortamında taklit ederler. Diğer eniyileme yöntemlerinde olduğu gibi çözüm için tek bir yapının geliştirilmesi yerine, böyle yapılardan meydana gelen bir küme oluştururlar. Problem için olası pek çok çözümü temsil eden bu küme GA terminolojisinde nüfus adını alır. Nüfuslar vektör, kromozom veya birey adı verilen sayı dizilerinden oluşur. Birey içindeki her bir elemana gen adı verilir. Nüfustaki bireyler evrimsel süreç içinde GA işlemcileri tarafından belirlenirler.

Problemin bireyler içindeki gösterimi problemden probleme değişiklik gösterir. GA'nın problemin çözümündeki başarısına karar vermedeki en önemli faktör, problemin çözümünü temsil eden bireylerin gösterimidir. Nüfus içindeki her bireyin problem için çözüm olup olmayacağına karar veren bir uygunluk fonksiyonu vardır. Uygunluk fonksiyonundan dönen değere göre yüksek değere sahip olan bireylere, nüfustaki diğer bireyler ile çoğalmaları için fırsat verilir. Bu bireyler çaprazlama işlemi sonunda çocuk adı verilen yeni bireyler üretirler. Çocuk, kendisini meydana getiren ebeveynlerin (anne, baba) özelliklerini taşır. Yeni bireyler üretilirken düşük uygunluk değerine sahip bireyler daha az seçileceğinden bu bireyler bir süre sonra nüfus dışında bırakılırlar. Yeni nüfus, bir önceki nüfusta yer alan uygunluğu yüksek bireylerin bir araya gelip çoğalmalarıyla oluşur. Aynı zamanda bu nüfus önceki nüfusun uygunluğu yüksek bireylerinin sahip olduğu özelliklerin büyük bir kısmını içerir. Böylelikle, pek çok nesil aracılığıyla iyi özellikler nüfus içerisinde yayılırlar ve genetik işlemler aracılığıyla da diğer iyi özelliklerle birleşirler. Uygunluk değeri yüksek olan ne kadar çok birey bir araya gelip, yeni bireyler oluşturursa arama uzayı içerisinde o kadar iyi bir çalışma alanı elde edilir. Probleme ait en iyi çözümün bulunabilmesi için [12];

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

Bu durumda çözüm kümesi problem için bir noktada birleşecektir. GA, diğer eniyileme yöntemleri kullanılırken büyük zorluklarla karşılaşılan, oldukça büyük arama uzayına sahip problemlerin çözümünde başarı göstermektedir. Bir problemin bütünsel en iyi çözümünü bulmak için garanti vermezler. Ancak problemlere makul bir süre içinde, kabul edilebilir, iyi çözümler bulurlar. GA'nın asıl amacı, hiçbir

çözüm tekniği bulunmayan problemlere çözüm aramaktır. Kendilerine has çözüm teknikleri olan özel problemlerin çözümü için mutlak sonucun hızı ve kesinliği açısından GA'lar kullanılmazlar. GA ancak;

1. Arama uzayının büyük ve karmaşık olduğu,
2. Mevcut bilgiyle sınırlı arama uzayında çözümün zor olduğu,
3. Problemin belirli bir matematiksel modelle ifade edilemediği,
4. Geleneksel eniyileme yöntemlerinden istenen sonucun alınmadığı alanlarda etkili ve kullanışlıdır.

GA parametre ve sistem tanılama, kontrol sistemleri, robot uygulamaları, görüntü ve ses tanıma, mühendislik tasarımları, planlama, yapay zekâ uygulamaları, uzman sistemler, fonksiyon ve kombinasyonel eniyileme problemleri ağ tasarım problemleri, yol bulma problemleri, sosyal ve ekonomik planlama problemleri için diğer eniyileme yöntemlerinin yanında başarılı sonuçlar vermektedir [12].

Diğer yöntemlerden farkı şunlardır:

1. GA, problemlerin çözümünü parametrelerin değerleriyle değil, kodlarıyla arar. Parametreler kodlanabildiği sürece çözüm üretilebilir. Bu sebeple GA ne yaptığı konusunda bilgi içermez, nasıl yaptığını bilir.
2. GA, aramaya tek bir noktadan değil, noktalar kümesinden başlar. Bu nedenle çoğunlukla yerel en iyi çözümde sıkışıp kalmazlar.
3. GA, türev yerine uygunluk fonksiyonunun değerini kullanır. Bu değerin kullanılması ayrıca yardımcı bir bilginin kullanılmasını gerektirmez.
4. GA, gerekirci kuralları değil olasılıksal kuralları kullanır.

Modern bilimde veri kümeleri arasındaki ilişkileri, tecrübelerden de faydalanarak belirlemek, üzerinde çokça çalışılan ve araştırılan bir taslaktır. Günümüzdeki araştırma konuları ve problemleri eskiye nazaran çok daha karışıktır. Bu karışıklık problemi etkileyen parametre sayısının fazlalığından ve problemin çözüm kümesinin boyutunun büyümesinden kaynaklanmaktadır. Bundan dolayı elimizdeki verilerin analizi ve sonucu bu verilerden tahmin etme yöntemlerinin önemi araştırmacılar için gittikçe artmaktadır. Faydalı iyi bir veri analiz yöntemi şu ölçütlere göre değerlendirilebilir: İyi tahmin veya sonucu kestirmeye yönelik olmalı, sistemin

içindeki her bir mekanizma analiz edilebilmeli ve sonuçlar mümkün olabilecek çözüm uzayı kümesinde olmalı. Bu tür problemlerdeki çözüm kümesinin büyüklüğü bir taraftan elde edilen çözümün değerlendirilmesinde zorluk çıkarırken diğer taraftan lineer yöntemlerin uygulanmasını imkânsız kılacaktır. Geçmişte araştırmacılar tarafından çalışılan parametreler arasındaki ilişkiler, genelde deneme yoluyla, zor olan örneklerde karmaşık veya sabit olmayan ilişkiler için yapılmıştır. Fakat parametre sayısı artınca çözümsüzlük veya elde edilen çözümü değerlendirememe problemini beraberinde getirmiştir. İstatistiksel yöntemler, araştırmacılara ilişkileri bulmada faydalı olan ilk araçlardandır. İstatistiksel yöntemlerde verinin normal toplanması, eşitlik ilişkisinin belirli bir formda olması (örneğin: lineer, polinomsal) ve değişkenlerin bağımsız olması gerekir.

Eğer problem bu ölçütleri sağlarsa, istatistiksel yöntem ilişkileri bulmada faydalı olabilir. Oysa gerçek hayatta problemler bu ölçütleri nadiren sağlarlar. Modern sonuç kestirme veya sonuç geliştirme algoritmaları bu ölçütlerle sınırlandırılamazlar. Neural Network (Yapay Sinir Ağları) veya Artificial Intelligence (Yapay Zeka) teknikleri karmaşık ilişkileri kapsamayabilir; fakat mekanizmanın önemli ilişkilerini tanımlayabilen güçlü tahmin modelleridir. Buna rağmen, diğer bir teknik GA ve GP teknikleri çok daha güçlüdürler ve karışık çözüm uzayını daha da geniş bulabilirler. Bağımsız olan veri ve parametreler ile mekanizmanın ilişkilerini bulmada başarılı örnekleri vardır. GA, biyolojik bir sistemin, çevresine adaptasyonunda kullandığı metodun örneklendirilmesidir. Bu tür çok parametreli optimum bulma problemlerine ve makine öğrenme problemlerine çözüm modeli olarak da alınabilir.

Mühendislikte, bilimde, ekonomide, finansmanda v.b. deki problemleri çözmede kullanılan arama teknikleri, hesap-temelli ve doğrudan arama teknikleri olarak sınıflandırılabilir. Eğer problemler sayısal veya analitik olarak iyi tanımlanabiliyorsa veya çözüm uzayı küçük ve tek ise, hesap temelli arama tekniği daha iyi çalışır. Buna rağmen hesap-temelli teknik mühendislik optimizasyonlarında gittikçe artan optimum bulma fonksiyonlarında oldukça zayıf kalır. Sadece fonksiyon bilgisi gerekli olan doğrudan arama tekniği, hesap-temelli teknikten daha kısa sürede işler ve daha etkilidir. Doğrudan arama tekniğinin esas problemi, ulaşılabilen bilgisayar zamanı ile optimal çözümün kesinliği arasındaki bağıntıdır [15].

2.2 Genetik Algoritma Tekniği

GA'da kullanılan temel kavramlar şu şekilde sıralanabilir [16]:

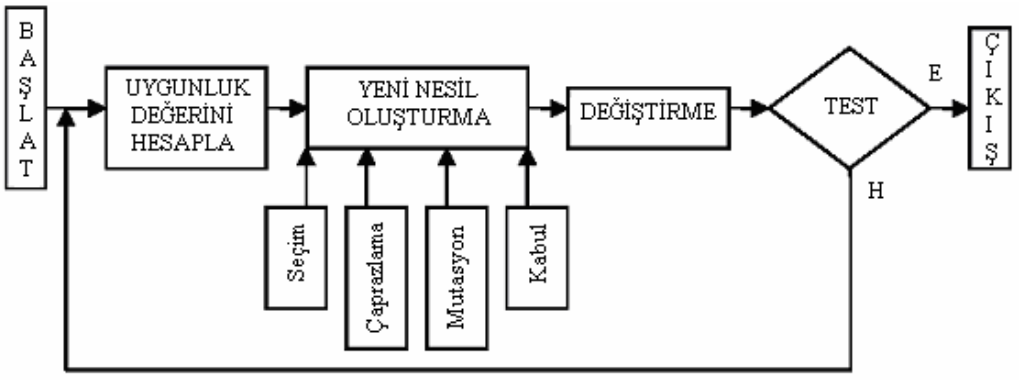
- **Gen:** Kendi başına anlamlı genetik bilgi taşıyan en küçük genetik yapıdır. GA'nın kullanıldığı programlama yapısında gen yapıları programcının tanımlamasına bağlıdır [13].
- **Kromozom:** Birden fazla genin bir araya gelerek oluşturduğu diziye denir. Kromozomlar alternatif aday çözümleri gösterirler. Kromozomlar GA yaklaşımında üzerinde durulan en önemli birim olduğu için bilgisayar ortamında iyi ifade edilmesi gerekir [13].
- **Popülasyon:** Kromozomlardan oluşan topluluğa denir. Popülasyon geçerli alternatif çözüm kümesidir. Popülasyondaki birey sayısı (kromozom) genellikle sabit tutulur. Popülasyondaki birey sayısı arttıkça çözüme ulaşma süresi azalır. Problemin özelliğine göre seçilecek olan popülasyon sayısı programcı tarafından iyi belirlenmelidir [13].
- **Allel (Allele) :** Bir özelliği temsil eden bir genin alabileceği değişik değerlere denir [13].
- **Locus:** Kromozom üzerindeki her bitin yerine verilen isimdir [13].
- **Genotip (Genotype):** Kodlanmış çözümden eski haline dönüştürülen çözümdür [13].
- **Fenotip (Fenotype):** Kodlanmış çözümdür [13].

2.3 Çalışma Prensibi

GA, doğal seçim ilkesine dayanan bir sayısal optimizasyon yöntemidir. GA, çözüm dizilerinden oluşan bir başlangıç nesliyle, çaprazlama ve mutasyon gibi doğal seçim operatörlerini kullanmaktadır [17]. GA'larda bağımsız parametrelerin kromozomlar içinde kodlanması gerekmektedir. Yığındaki her birey ikili düzende veya tamsayı olarak kodlanmaktadır. GA, oldukça genel prensiplerle Şekil 1.1'deki akış şemasında görüldüğü gibi çalışmaktadır. Öncelikle ele alınan problem için rastgele n kromozomlu bir popülasyon oluşturulur. Daha sonra popülasyondaki her bir

kromozom için f(x) uygunluk fonksiyonu hesaplanır. Yeni bir popülasyon oluşuncaya kadar aşağıdaki adımlar tekrar edilir [17]:

1. Seleksiyon (Seçim): İki ebeveyn kromozomun f(x)'e göre seçimi, burada uygunluk derecesi yüksek olanın seçilme şansı yüksektir.
2. Çaprazlama: Yeni bir birey oluşturmak için ebeveynlerin bir çaprazlama olasılığına göre çaprazlanması. Eğer çaprazlama uygulanmazsa bireyler atalarının tamamen kopyası olacaklardır.
3. Mutasyon: Kromozom üzerindeki bazı genlerin değerleri değiştirilerek nesillerin yozlaşması önlenir.
4. Ekleme: Yeni bireyin yeni topluma eklenmesi.
5. Değiştirme: Algoritmanın yeniden çalıştırılmasından oluşan yeni toplumun kullanılması.
6. Test: Eğer sonuç tatmin ediciyse algoritmanın sona erdirilmesi ve son toplumun çözüm olarak sunulması.
7. Döngü: 2. adıma geri dönülmesi [17].



Şekil 1.1 Genetik algoritmaların genel akış şeması

Yeni popülasyon kabul edildikten sonra hesaplama yeni popülasyonla tekrarlanır. Hedeflenen uygunluk değerine ulaşıldığında program durdurulur ve popülasyondaki en iyi çözüm alınır. GA'larda kromozomlarla bir başlangıç popülasyonu rastgele oluşturulur. Burada popülasyon genişliğinin belirlenmesi gerekmektedir. Büyük popülasyonlarda, çözüm uzayı iyi örneklendiği için aramanın etkinliği artmakta, fakat buna bağlı olarak da arama süresi uzamaktadır. Küçük popülasyonlarda ise, çözüm uzayını yeterli örnekleyememe ve zamansız yakınsama oluşabilmektedir [17]. GA'nın her çevriminde, yığındaki dizilerin bir değerlendirme fonksiyonu yardımıyla uygunluk değeri hesaplanır [18]. Uygunluk fonksiyonu her bir çözümün yeni nesil çözümlere katkı sağlayıp sağlamayacağına karar verir. Sonrasında insan

üremesindeki gen transferine benzer operasyonlar kullanan algoritma yeni bir çözüm adayı popülasyonu oluşturur. Genellikle GA'ların başarısı bu fonksiyonun verimli ve hassas olmasına bağlıdır [19].

2.4 Kodlama Yöntemleri

Kodlama planı, GA'ların önemli bir kısmını teşkil eder. Çünkü bu plan bilginin çerçevesini şiddetle sınırlayabilir. Öyle ki probleme özgü bilginin bir kromozomsal gösterimiyle temsili sağlanır. Kromozom genellikle, problemdeki değişkenlerin belli bir düzende sıralanmasıdır. Kromozomu oluşturmak için sıralanmış her bir değişkene "gen" adı verilir. Buna göre bir gen kendi başına anlamlı genetik bilgiyi taşıyan en küçük genetik yapıdır. Mesela; 101 bit dizisi bir noktanın x-koordinatının ikilik düzende kodlandığı gen olabilir. Aynı şekilde bir kromozom ise bir ya da daha fazla genin bir araya gelmesiyle oluşan ve problemin çözümü için gerekli tüm bilgiyi üzerinde taşıyan genetik yapı olarak tanımlanabilir. Örnek vermek gerekirse; 100011101111 *x1, y1, x2, y2* koordinatlarından oluşan iki noktanın konumu hakkında bize bilgi verecektir. Bu parametreleri kodlarken dikkat edilmesi gereken en önemli noktalardan biri ise kodlamanın nasıl yapıldığıdır. Örnek olarak kimi zaman bir parametrenin doğrusal ya da logaritmik kodlanması GA performansında önemli farka yol açar. Kodlamanın diğer önemli bir hususu ise kodlama gösteriminin nasıl yapıldığıdır. Bu da yeterince açık olmamakla birlikte GA performansını etkileyen bir noktadır. Bu konu ileride ele alınacaktır [12].

2.5 Uygunluk Teknikleri

Başlangıç topluluğu bir kez oluşturulduktan sonra evrim başlar. GA, bireylerin uygunluk ve iyiliklerine göre ayrılıp fark edilmesine gerek duyar. Uygunluk, topluluktaki bir kısım bireyin problemi nasıl çözeceği için iyi bir ölçüdür. O problem parametrelerini kodlamayla ölçer ve uygunluk fonksiyonuna giriş olarak kullanır. Büyük olasılıkla, uygun olan bu üyeler tekrar üreme, çaprazlama ve mutasyon operatörleriyle seçilirler. Bazı problemler için bireyin uygunluğu, bireyden elde edilen sonuç ile tahmin edilen sonuç arasındaki hatadan bulunabilir. Daha iyi bireylerde bu hata sıfıra yakın olur. Bu hata genellikle, girişin tekrar sunulacak kombinezonlarının ortalaması veya toplamıyla hesaplanır (değerler değişkenlerden bağımsızdır). Beklenen ve üretilen değer arasındaki ilgileşim etkeni, uygunluk

değerini hesaplamak için kullanılabilir. Objektif fonksiyonu (değerlendirme fonksiyonu) her bir kromozomun durumunu değerlendirmek için mekanizmayı sağlayan ana bir kaynaktır. Bu GA ve sistem arasında önemli bir bağlantıdır. Fonksiyon giriş olarak kodu çözülmüş şekilde kromozom alır ve kromozomun performansına bir ölçü olarak bir objektif değer üretir. Bu diğer kromozomlar için de yapıldıktan sonra bu değerler kullanılarak, uygun değerler uygunluk fonksiyonuyla hesaplanıp belli bir düzende planlanır [12].

2.6 Genetik Operatörler

Kullanılan genetik operatörler, var olan nesil (population) üzerine uygulanan işlemlerdir. Bu işlemlerin amacı, daha iyi özelliğe sahip yeni nesiller üretmek ve arama algoritmasının alanını genişletmektir. 3 tip genetik operatör vardır[12]:

1. Seleksiyon
2. Çaprazlama
3. Mutasyon

2.6.1 Seleksiyon

Yeniden üretme operatörü, hazır topluluktan uygun olan bireylerin seçilmesi ve bunların sonraki topluluğa kopyalanarak hayatta kalmalarıyla ilgilidir. Seçim modeli, tabiatın hayatta kalabilmek için uygunluk mekanizması modelidir. Yeniden üretme işleminde, bireyler onların uygunluk fonksiyonlarına göre kopya edilirler. Uygunluk fonksiyonu, mümkün olduğu kadar yükseltilmesi gereken bazı faydalı ve iyi ölçülerdir. Topluluk uzayındaki her bir bireyin uygunlukları temel alınarak ne kadar sayıda kopyasının olacağına karar verilir. En iyi bireylerden daha fazla kopya alınır, en kötü bireylerden kopya alınmaz. Bu hayatta kalmak için uygunluk stratejisinin GA ya sağladığı avantajdır [12].

2.6.2 Rulet tekerleği seçimi

GA tarafından üretilen döllerin sayısını belirlemede birkaç yol vardır. Birbirine yakın parametrelerden kaçınmak için uygun bir seçim metodu kullanılmalıdır. Tekrar üretme başlangıcında basit bir yöntem rulet tekerleğiyle seçim (roulette wheel selection)'e göre bireylerin uygunluk değerlerini bir rulet tekerleğinde hazırlar. Rastgele tekerleğin döndürülmesinden sonra, bireyin bir sonraki nesil için seçilmesi,

tekerlek üzerinde kapladığı alanla doğrudan bağlantılıdır. Bu yöntem düşük uygunluğa sahip bireylere de seçilme hakkı verir [12].

Pseçilen=Fi / å Fi

i=1

Fi: i. eleman için uygunluk değeri

N: Birey sayısı

Ebeveynler uygunluklarına göre seçilirler. Kromozomlar ne kadar iyiyse, o kadar seçilme şansları fazladır. Sonra bir bilye atılır ve kromozomu seçer. Daha fazla uygunluğu olan kromozomlar daha çok seçilecektir. Bu aşağıdaki algoritmayla simule edilebilir:

1. [Sum]. Popülasyondaki tüm kromozom uygunlukları toplamını hesapla – toplam S.
2. [Select]. (0,S) – r aralığından rastgele bir sayı üret.
3. [Loop]. Populasyon boyunca git ve uygunlukları 0'dan toplam s'e kadar topla.

Eğer toplam s, r'den büyükse dur ve olduğun yerdeki kromozomu geri gönder. Tabii ki, 1. basamak her popülasyon için bir kez uygulanır [12].

2.6.3 Rank seçimi

Yukarıdaki seçim eğer uygunluklar çok fazla değişiyorsa bazı problemlere yol açacaktır. Mesela en iyi kromozom uygunluğu tüm rulet tekerleğinin %90'ı ise diğer kromozomların seçilme şansları çok az olacaktır. Rank seçimi önce popülasyonu sıralar ve daha sonra her kromozom uygunluğu bu sıralamadan sonra alır. En kötüsü 1 uygunluğunu alacak, ikinci en kötü 2 ve en iyisi N uygunluk değerini alacak ki N de popülasyondaki kromozom sayısıdır [12].

Bundan sonra her kromozomun seçilme hakkı olacaktır. Fakat bu metot daha yavaş gibidir, çünkü en iyi kromozomlar diğerlerinden fazla değişiklik göstermez.

2.6.4 Kararlı hal seçimi (Steady-State)

Bu yerine geçme yöntemi olarak da adlandırılabilir. Bu ebeveynleri seçmek için kısmi bir metot değildir. Bu seçimin ana fikri kromozomların büyük kısmı bir sonraki nesilde hayatta kalmak zorundadır. O zaman GA şu şekilde çalışır. Yeni

çocuklar oluşturmak için her nesilde güzel iyi uygunluklu birkaç kromozom seçilir. Sonra kötü düşük uygunluklu bazı kromozomlar atılır ve yeni çocuk onun yerine yerleştirilir. Popülasyonun geri kalan kısmı yeni nesilde hayattadır. Yani kısaca bu yöntemde alt popülasyon oluşturulduktan sonra uygunluklar hesaplanır, en kötü kromozomlar yerlerini başlangıç popülasyonundaki en iyi kromozomlara terk eder [12].

2.6.5 Elitizm

Mutasyon ve çaprazlamalarla yeni nesil oluştururken en iyi kromozomu seçmek için büyük bir şansa sahip oluruz. Elitizm, en iyi kromozomu ya da birkaç en iyi kromozomları yeni nesle kopyalama metodunun adıdır. Gerisi klasik yolla yapılır. Elitizm çok hızlı bir şekilde GA'nın performansını arttırır. Çünkü en iyi bulunan çözümü kaybetmeyi önler [12].

2.6.6 Çaprazlama

Amaç, ebeveyn kromozom genlerinin yerini değiştirerek çocuk kromozomlar üretmek ve böylece var olan uygunluk değeri yüksek olan kromozomlardan, uygunluk değeri daha yüksek olan kromozomlar elde etmektir. Burada önemli olan bir konuda, çaprazlama noktasının çaprazlamadan elde edilecek çocuk kromozomların uygunluk değerleri üzerindeki etkisidir. Bu işlem yapılırken her zaman sonuçlar önceden tahmin edilemez. Bu yüzden gelişigüzel yapılan değişikliklerde sonucun mükemmelliğe doğru gitmesi için belirli ölçütler bulmak için çalışılır. Kromozomlardaki genlerin yapısı ve etkileri araştırılarak, bu genlere yapılan müdahalelerle bireye bazı iyi özellikler kazandırılabilir. Çaprazlamadan elde edilecek çocuk kromozomların uygunluk değeri bir önceki ana kromozomlardan daha yüksek olmayabilir [12].

Benzer şekilde GA, çaprazlama işlemini uygunluk değerlerine göre seçilmiş iki ebeveyn bireyden, iyi özellikte yeni bireyler elde etmek için kullanır. Çaprazlama rastgele seçilmiş iki çift katarın içindeki alt küme bilgilerin değiştirilmesi işlemidir. Kendi içindeki bilgilerini 1. pozisyondan itibaren, katarın uzunluğunun bir eksik pozisyonuna kadar, aradaki bilgi kısmen karşılıklı bireyler arasında yer değiştirilir. Eğer iki bireyin problemin çözümünde bazı etkileri var ise onların bir parçaları faydalı, iyi veya uygun nitelenebilecek bilgi taşımaktadır. Çaprazlama belki problemin çözümünde, bu faydalı bilgileri birleştirerek, daha çok etkili yeni bireyler

üretecektir [12]. Çaprazlamadan başka tersinme denilen bir üreme yöntemi daha vardır. Holland bunu tanımlayarak kromozom uzunluğu çok olan bireylerde çaprazlama yerine bunun kullanılmasını performans açısından önermiştir. Tersinme (inversion) bir kromozomu oluşturan genlerden ardışık bir grubun kendi içerisinde birbirleriyle yer değiştirerek ters dizilmeleridir. Örneğin:011110101 kromozomu (her genin bir bit olduğu varsayımı ile) 5. ve 8. gen kromozomları arasında tersindiğinde ortaya 011101011 kromozomu çıkar. Tersinme genellikle kromozom uzunluğu fazla olan popülasyonlara uygulanır [12].

2.6.7 Mutasyon

Amaç, var olan bir kromozomun genlerinin bir ya da birkaçının yerlerini değiştirerek yeni kromozom oluşturmaktır. Yeniden ve sürekli yeni nesil üretimi sonucunda belirli bir süre sonra nesildeki kromozomlar birbirlerini tekrarlama konumuna gelebilir ve bunun sonucunda farklı kromozom üretimi durabilir veya çok azalabilir. İşte bu nedenle nesildeki kromozomlarının çeşitliğini artırmak için kromozomlardan bazıları mutasyona uğratılır. Açıklandığı gibi mutasyonun birinci maksadı bir popülasyonun içindeki değişimi tanımlamaktır. Mutasyon popülasyonlarda çok önemlidir. Öyle ki, burada ilk popülasyon mümkün olan tüm alt çözümlerin küçük bir alt kümesi olabilir ve ilk popülasyondaki tüm kromozomların önemli biti sıfır olabilir. Hâlbuki o bitin problemin çözümü için 1 olması gerekebilir ve bunu da çaprazlama düzeltemeyebilir. Bu durumda o bit için mutasyon kaçınılmazdır. Genellikle önerilen mutasyon oranı 0.005/bit/jenerasyondur. Bu işlem çaprazlamadan sonra gelir. Mutasyonun yapılıp yapılmayacağını bir olasılık testi belirler. Örneğin yeni neslin ortalama uygunluğu £, eski neslin ortalama uygunluğu ise; x. mutasyon kromozomun y. bitini değiştir denilebilir. Bu değişim rast gele olabilir. İkili kodlama için rast gele seçilmiş bitlerden 0'ları 1, 1'leri 0 yaparız [12].

2.7 Uygulama Alanları

2.7.1 Genel uygulama alanları

Bir arama yöntemi olan GA, farklı bilim dallarındaki optimizasyon problemlerini çözmede kullanılmaktadır. GA'ların uygulandığı optimizasyon problemleri, fonksiyon optimizasyonu ve birleşi (combinatorial) optimizasyonu altında toplanabilir. GA araştırmalarının önemli bir bölümü fonksiyon optimizasyonu ile

ilgilidir. GA, geleneksel optimizasyon tekniklerine göre zor, süreksiz ve gürültü (noise) içeren fonksiyonları çözmede daha etkindirler Optimize edilecek amaç fonksiyonunun süreksiz olması halinde, süreksizlik noktalarında fonksiyonun türevi alınamayacağından, türev almaya dayalı optimizasyon yöntemleri kullanılamamaktadır. Oysa GA, problemlerin çözümü için türev veya diğer yardımcı bilgilere gereksinim duyan diğer yöntemlere göre önemli bir üstünlük sağlamaktadır [20].

GA'nın uygulandığı diğer bir optimizasyon problem sınıfı olan birleşi optimizasyon problemleri ise, istenen amaçlara ulaşmak üzere, sınırlı kaynakların etkin tahsis edilmesiyle ilgilidir. Bu sınırlar genel olarak, işgücü, tedarik veya bütçe ile ilgilidir. Sözü geçen "birleşi" kelimesi, yalnızca sonlu sayıda alternatif uygun çözümün mevcut olması ile ilgilidir [20].

GA'nın yaygın olarak kullanıldığı alanlardan biri de, belirli ve özel görevler için gerekli olan bilgisayar programlarını geliştirmedir. Ayrıca, diğer hesaplama gerektiren yapıların tasarımı için de kullanılmaktadır. Bunlara örnek olarak, bilgisayar çipleri tasarımı, ders programı hazırlanması ve ağların çizelgelenmesi verilebilir. GA kullanılarak dağıtılmış bilgisayar ağlarının tasarımı da gerçekleştirilmektedir. Bu problem tipinde ağ güvenilirlik parametrelerini (çap, ortalama uzaklık ve bilgisayar ağ güvenilirliği gibi) optimize etmek için birden fazla amaç fonksiyonu kullanılmaktadır. Ağ tasarımında GA'nın kullanılması, tasarım sürelerinin ve maliyetlerinin azalmasında önemli bir katkı sağlamıştır [20].

Mekanik öğrenme; ilki, gözlenmiş bir veri takımını anlamak ve yorumlamak, ikincisi de görülmemiş objelerin özelliklerini tahmin etmek olan iki temel amaç için model kurmayı amaçlar. Parametrik istatistikten ziyade çok büyük veri takımlarının yönetimi üzerinde çalışır. Kullandığı metotların çoğu dağılımdan bağımsız metotlar olarak sınıflanabilir. Uygun model seçimi için işe problem hakkındaki varsayımlarla başlamaz. Onun yerine uygun model yapısını belirlemek için doğrudan mevcut veriden hareketle bir araç kutusu yaklaşımı kullanır. Sınıflama sistemi, GA'nın mekanik öğrenme alanında bir uygulamasıdır. Basit dizi kurallarını öğrenen bir mekanik öğrenme sistemi olan sınıflama sisteminin kural ve mesaj sistemi, özel bir üretim sistemi olarak adlandırılabilir. Bu üretim sistemi, "eğer-sonra" kural yapısını kullanır. Bir üretim kuralı, "eğer" yapısından sonra belirtilen durum için, "sonra" yapısından sonra gelen faaliyetin gerçekleştirilmesini içerir. GA, sınıflama

sistemlerinde kural-bulma mekanizması olarak kullanılmaktadırlar. GA ayrıca, sinir ağlarında ve proteinin yapısal analizinde de kullanılmaktadır [20].

Bir sistemi ölçen deneysel olarak gözlenmiş değişkenler arasındaki matematiksel ilişkiyi keşfetme problemi ekonomide en önemli problemlerden biridir. Pratikte gözlenmiş veri gürültü içerebilir ve kapsanan ilişkileri kesin ve açık bir şekilde açıklayacak bir yol bilinmeyebilir. Bu tip problemler, sembolik sistem tanımlama, kara kutu, veri madenciliği ve modelleme problemleri olarak bilinir. Eğer keşfedilen model, sistemin durum değişkenlerinin gelecek değerlerini tahmin etme için kullanılacaksa problem öngörüleme problemi adını alır. Geleneksel doğrusal, ikinci dereceden ve üstel regresyon modellerinde sapma hataları minimize edilerek fonksiyonlara uygun sayısal katsayılar bulunur. Buradaki yaklaşım, model seçildikten sonra uygun sayısal katsayıların aranmasıdır. Gerçek problem ise verinin değerlendirilmesi için hangi tip modelin uygun olduğunun kararıdır. Keyfi bir matematiksel ilişkiyi açıklamada bilgisayarlar, bu ilişkiyi formüller ve denklemler aracılığı ile açıklamaktan daha esnektir. Bu nedenle, bu tip ilişki açıklamaları için sembolik regresyon kullanılabilmektedir. Sembolik regresyonlar, hem fonksiyon formunu hem de o fonksiyondaki uygun katsayıyı araştırmaktadır. Bunu bulma ise, verilen girdiler için arzu edilen çıktıları üreten özel bir hesaplama programı ile program uzayında arama yapmaya benzemektedir. GA'nın kullanıldığı GP ile bu tip problemlere tatmin edici çözümler çok daha kolay getirilebilmektedir. GA, yenilik sürecinin modellenmesi amacıyla da kullanılmaktadır. Ayrıca GA'nın, fiyat verme stratejilerinin gelişim süreçlerini ve kazanç getiren pazarların ortaya çıkış süreçlerini modelleme alanlarında da kullanımları oldukça yaygındır. GA, sosyal sistemlerin evrimsel yönlerini anlamak amacıyla kullanılmaktadır. Bunlara örnek olarak işbirliğinin evrimi, iletişimin evrimi ve karıncalardaki iz takibi davranışının evrimi verilmektedir [20].

2.7.2 İşletmedeki uygulama alanları

GA; başta üretim/işlemler olmak üzere finans ve pazarlama gibi işletmelerin fonksiyonel alanlardaki birçok farklı iş probleminin çözümü için kullanılmaktadır. GA'nın özellikle, kaynak tahsisi, iş atölyesi çizelgelemesi, makine parça gruplaması ve bilgisayar ağ tasarımı gibi çeşitli alanlarda uygulamaları mevcuttur.

GA, finansal modelleme uygulamaları için son derece uygundurlar. GA, amaç fonksiyonu odaklıdır. Finans problemlerinde genel olarak, amaç fonksiyonları tahmin etme gücüne veya bir kıyaslama sonucuna bağlı getirilerdeki gelişmeleri içerir. Kullanılan araç ve problemler arasında mükemmel bir eşleşme mevcuttur.

Tüketicilere ait verileri analiz etmek, çeşitli tüketici kalıpları çıkarmak ve bu kalıplara dayanarak pazarlama stratejileri uygulamak, pazarlamanın en önemli fonksiyonlarından biridir. Tüketicilerin profilleri çıkarılarak, belirli satın alma kalıpları yakalanabilmektedir. Ancak tüketici profilini çıkarabilmek için, çok büyük veri tabanlarını işletme amaçları doğrultusunda hızlı ve etkin biçimde kullanmak gerekmektedir. Burada kullanılan teknik, veri madenciliğidir. Veri madenciliği, çok geniş veri tabanlarından veriyi süzme tekniğidir. Pazarı ve tüketiciyi tanımada son derece önemli rol oynayan veri madenciliği, veriyi bilgiye bilgiyi de güvenli kararlara dönüştürür. Veri madenciliğinin verimlilik, karlılık, müşteri tatmini ve rekabet edebilme yeteneği gibi yaşamsal konularda işletme üzerinde çok önemli etkileri bulunmaktadır. Rekabet edebilme yeteneği karar alma kalitesine bağlıdır ve bundan dolayı işletmeler sürekli karar kalitelerini geliştirmeye çalışırlar. Veri madenciliğinde kullanılan tekniklerden birisi de GA'dır. GA tabanlı yaklaşım kullanılarak veri yığınlarından modeller elde edilmektedir [20].

GA'nın en çok uygulandığı alanların başında üretim/ işlemler gelmektedir. Montaj işlemi endüstrilerde çok önemli bir rol oynamaktadır. Nof ve arkadaşlarının 1997'de yayınlanan çalışmalara göre üretilen mamullerin montajı, toplam üretim zamanının % 50'sine, toplam birim üretim maliyetinin % 20'sine ve işçilik maliyetlerinin % 30-% 50'sine karşılık gelmektedir. Bundan dolayı montaj hattı dengeleme problemi, firmalar açısından yaşamsal öneme sahiptir [20].

GA'nın çizelgeleme problemine ilk uygulama çalışması, Davis tarafından 1985 yılında yapılmıştır. 1987'de Liepins ve arkadaşları, belirli teslim tarihleri ve işlem süreleri olan işlerin çizelgelenmesi problemini araştırmışlardır. Bu problem en basit çizelgeleme problemi olarak adlandırılmaktadır. Genel olarak GA'lar, çizelgeleme problemlerine optimala yakın çözüm bulmuşlardır. Fakat çözüm bulma süreleri diğer çözüm yöntemlerine göre oldukça hızlı olmuştur [20].

Tesis yerleşim problemleri araç/gereçleri veya diğer kaynakları belirli bir ölçüte göre optimum performans sağlayacak şekilde yerleştirme kararını içermektedir. Bu gibi

kararlar, araç/gereçlerin genellikle farklı ürünleri üretme esnasında kullanılmasından dolayı karmaşık hale gelmektedir. Her ürünün kendine özgü gereksinimleri olabilir ve tüm ürünler için toplam üretim maliyetinin optimum olması sağlanacak şekilde yerleşim tasarlanabilir. Yerleşim kararları hızlı ve doğru verilmelidir. Çünkü kararların zayıflığı üretim esnasında ortaya çıkmakta ve bu da artı maliyetlere yol açmaktadır. Örneğin, üretimde robot kullanan işletmelerin tesis yerleşimi tasarımında karmaşıklık söz konusudur. Tek bir robot bir makineden diğerine parçalar taşırken hareketsiz bir noktada sabitlenir ve yalnızca bir eksen etrafında hareket eder. Robotun hareketine göre, makineler tek-sıra, doğrusal çift-sıra, dairesel tek-sıra ve çoklu-sıra gibi dört farklı yerleşim şekliyle yerleştirilebilir. Burada, dairesel tek-sıra, doğrusal tek-sıranın özel bir durumudur [20].

Genel olarak atama problemi; n elemanın, n farklı göreve atanması problemidir. *i*. kişinin, *j*. işi yapma maliyeti cij dir. Bu durumda problem amaç fonksiyonunu minimize edecek *{Π1, ..., Πn}* atama kümesinin bulunması şeklinde tanımlanabilir. Burada problem çözümü, *{1, ..., n}* sayılarının *{Π1,...,Πn}* permütasyonu olarak gösterilmektedir [20].

Hücresel üretim kavramı, üretim sistemlerinin verimliliğini arttırmada anahtar faktörlerden biridir. Hücresel üretim, parça ailelerini belirledikten sonra, her parça ailesini ayrı bir üretim hücresinde imal ederek hücreler arası taşımaları en aza indirmeyi amaçlamaktadır. GA, hücreler arası taşımanın minimum olduğu bir hücre kuruluşu amaçlanmasında kullanılabilmektedir [20].

Bir sistemin güvenilirliği, belirli koşullar altında belirli bir zaman aralığında sistemin başarılı olarak çalışma olasılığı olarak tanımlanmaktadır. Çoğu sistem, çeşitli işlemlerde kritik bir role sahiptir ve eğer sistemde arıza olursa sonuçları oldukça ciddi olmaktadır. Bu alanda optimizasyon, etkisiz parçaların sisteme en iyi şekilde tahsis edilebilme veya yararlanabilme yolunu bulmayı içermektedir [20].

Tedarikçilerden tüketicilere, talebi karşılamak üzere, minimum maliyetle tek tipte mamul gönderilmesini içermektedir. Burada *m* tane tedarikçi ve *n* tane de tüketici mevcuttur. Tek tedarikçiden her bir tüketiciye bir birim mamul ulaştırma maliyeti bilinmektedir. Problem, tüm talebin karşılanması ve maliyet minimizasyonu şartıyla mamulün arz yerinden talep yerine optimum tahsisini sağlamaktır. Son zamanlarda,

çeşitli taşıma problemlerinin çözümü için evrimsel (evolutionary) yaklaşımlarla çözüm önerileri sunulmaktadır [20].

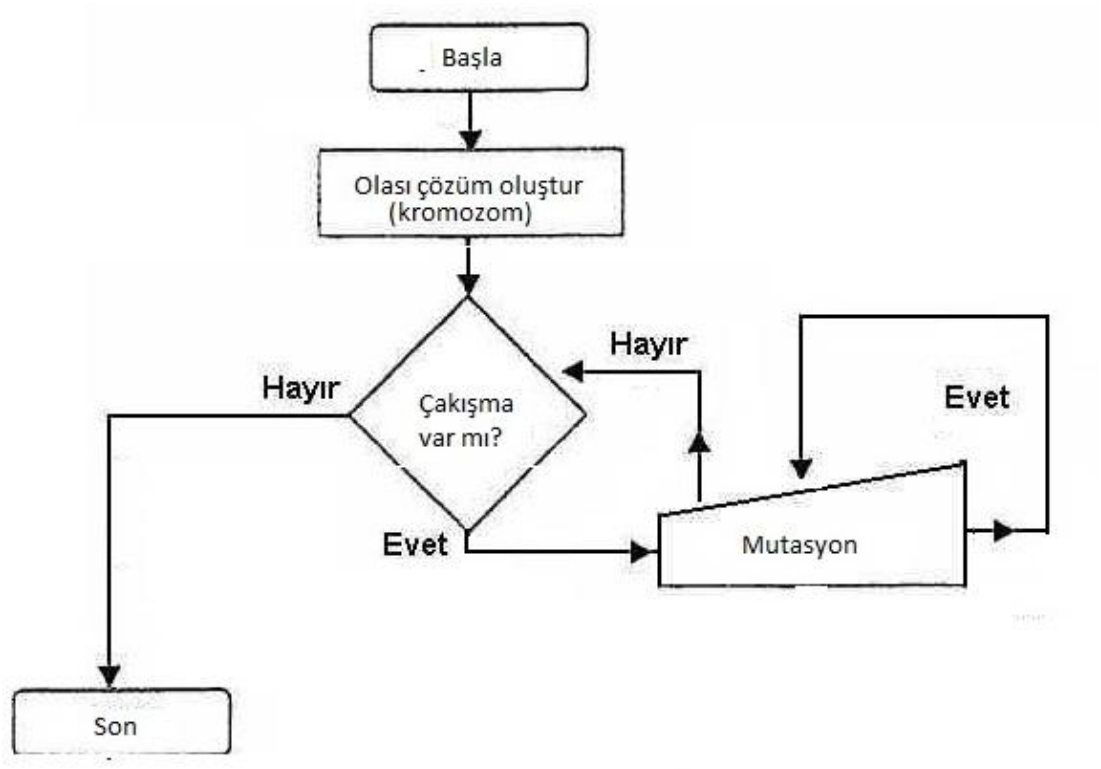
GA'nın, birleşi optimizasyon problemlerine uygulamaları ile ilgili çeşitli çalışmalar mevcuttur. En yoğun yapılan çalışmalardan biri de gezgin satıcı problemleri için yapılmaktadır. Gezgin satıcı probleminde amaç, kat edilen toplam mesafeyi minimize eden bir yolculuk planı oluşturmaktır. Birçok problem tipi gezgin satıcı problemi gibi modellenebilmektedir. Bunlara örnek olarak; devre tasarımı, posta taşıyıcılarının, okul otobüslerinin rotalarının bulunması verilebilir. Gezgin satıcı probleminin bir özelliği de değişken sayısı artıkça üstel artış gösteren zaman ihtiyacı içinde çözüme ulaştırılabilmesidir. Bu durum bir örnekle şöyle açıklanabilir; bir satış görevlisinin ziyaret etmek durumunda olduğu n tane şehir olsun. Burada tüm şehirlerarasındaki maksimum izlenecek rota sayısı *(n-1)!* dir. Tüm mümkün rotaları basitçe inceleyen ve en kısa olan rotayı bulan bir algoritma kullanılır. Fakat şehir sayısı arttıkça algoritmanın hesaplama için gereksinim duyduğu zaman daha da büyük bir oranda artmaktadır. Ziyaret edilmesi gereken 25 şehir varsa, algoritmanın inceleyeceği rota sayısı 24!'dir. Bu da yaklaşık $6{,}2 \times 10^{23}$ sayısına karşılık gelmektedir. Saniyede bir milyon rota inceleme kapasitesine sahip bir bilgisayar, bu problemi, $6{,}2 \times 10^{11}$ saniyede yani, $1{,}96 \times 10^{1^o}$ yılda çözebilmektedir. Herhangi bir problem için kullanılan algoritmanın en yaygın performans ölçütü, algoritmanın çözüme ulaşma süresidir. Gezgin satıcı gibi değişken sayısı arttıkça çözüm zamanı üstel olarak artan problemlerde bu daha da önemlidir. GA birleşi optimizasyon problemlerini klasik yöntemlere göre çok daha kısa sürede çözmektedir. Sonuçta optimale yakın ve kabul edilebilir bir çözüm bulunmaktadır [20].

Birleşi optimizasyon problemlerinin örneklerinden biri de araç rotalama problemidir. Temel araç rotalama problemi, talebi belirli olan müşterileri kapsar. Tek bir depodan araçlar ayrılmakta ve müşteri taleplerini karşılayarak tekrar depoya dönmektedir. Her aracın kapasite kısıtı vardır. Bu temel probleme ayrıca, her aracın alacağı yol da mesafe kısıtı olarak eklenebilir. Her bir müşterinin talebini yalnızca bir araç karşılamaktadır. Problem, bu kısıtlar altında minimum toplam maliyeti veren rotaları bulmaktır. Daha karmaşık bir araç rotalama problemi olan zaman pencereli rotalama probleminde ise amaç müşteri talebini belirli zaman aralıkları içersinde minimum toplam maliyetle karşılamaktır [20].

3. SİSTEM TASARIMI

3.1 Veri Akış Diyagramı

Uygulamanın veri akış diyagramı Şekil 2.1'de verilmiştir. Buna göre uygulama, olası çözüm içeren kromozomun oluşturulmasıyla başlar. Çakışmalar olduğu sürece kromozom mutasyonlara uğrar. Çakışmalar yok edildiğinde ise uygulama sona ermiş olur.



Şekil 2.1 Veritabanı Modeli

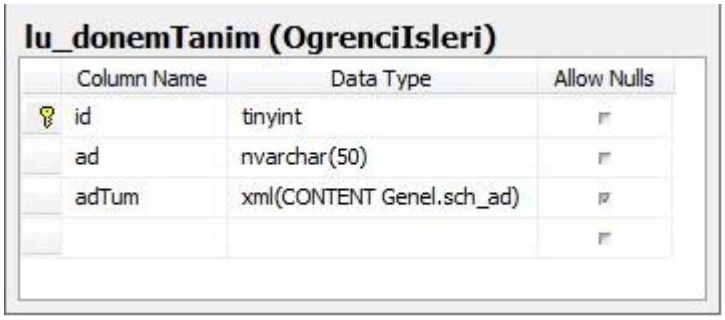
Uygulamada kullanılan tablolar Pamukkale Üniversitesi Pusula Sistemi altyapısında var olan veritabanı modeli esas alınarak oluşturulmuştur. Pusula sistem veritabanı Microsoft SQL Server 2008 üzerinde 2008 yılından beri kullanılmaktadır.

Veritabanının ana iskeleti de buradaki tablolardır. Proje, gereksinim duyulan tabloların mevcut veritabanına eklenmesiyle geliştirilmiştir.

3.2 Var Olan Tablolar

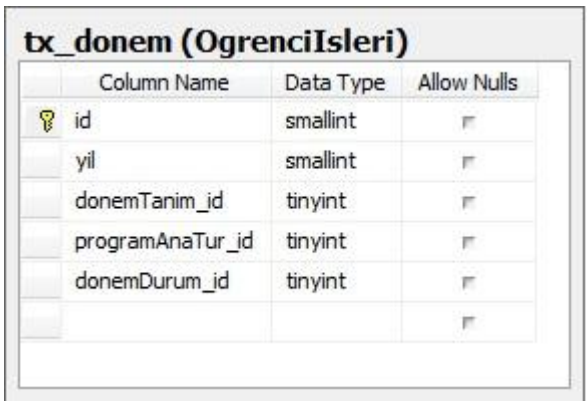
Pamukkale Üniversitesinde 2008 yılından itibaren veritabanı yenilenme süreci başlamıştır. Bu yeni sistemde var olan ve bu projede kullanılan tablolar aşağıda yer almaktadır.

Proje veritabanında öğrenim dönemleri 2010 Güz, 2010 Bahar, 2010 Yaz şeklindedir. Şekil 2.2'deki lu_donemTanim tablosunda bu dönemlerin tanım adları bulunur. (Güz, Bahar, Yaz)



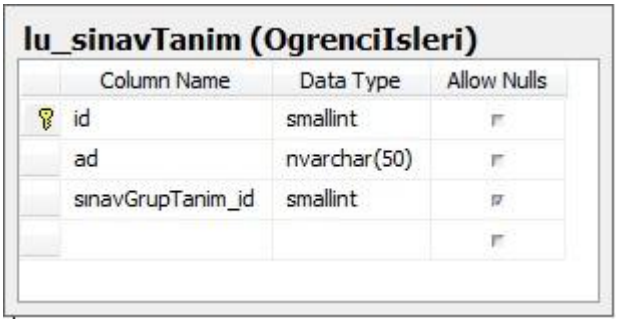
Şekil Error! Use the Home tab to apply 0 to the text that you want to appear here.**2.2 lu_donemTanim**

Şekil 2.3'deki lu_donemTanim tablosunda bağlı tx_donem tablosunda da yıl ve projeyle ilgili olmayan programAnaTur_id (lisans, yüksek lisans, vb.) bilgileri tutulmaktadır.



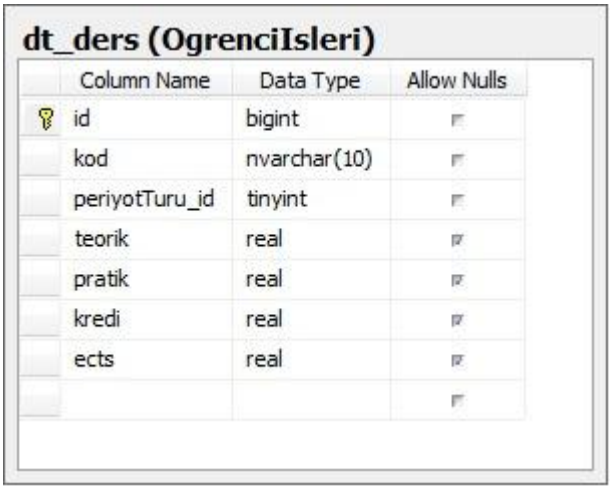
Şekil Error! Use the Home tab to apply 0 to the text that you want to appear here.**2.3 tx_donem tablosu**

Şekil 2.4'deki lu_sinavTanim tablosunda sınavların türleri yer almaktadır. Burada şimdilik sadece final ve ara sınav tanımları bulunur. Üniversite yönetmeliğinde yapılan değişikliklere göre gerekli tanımlar eklenebilir.



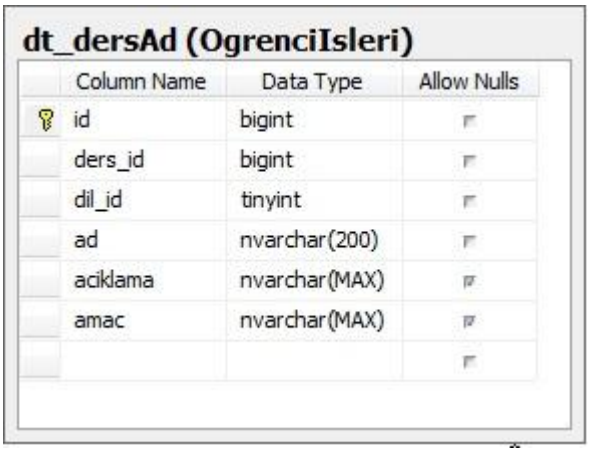
Şekil 2.4 lu_sinavTanim tablosu

dt_ders tablosu derslerin kodu ile teorik, pratik, kredi ve acts gibi bilgilerini tutan tablodur. Bu tablo Şekil 2.5'de sunulmuştur.



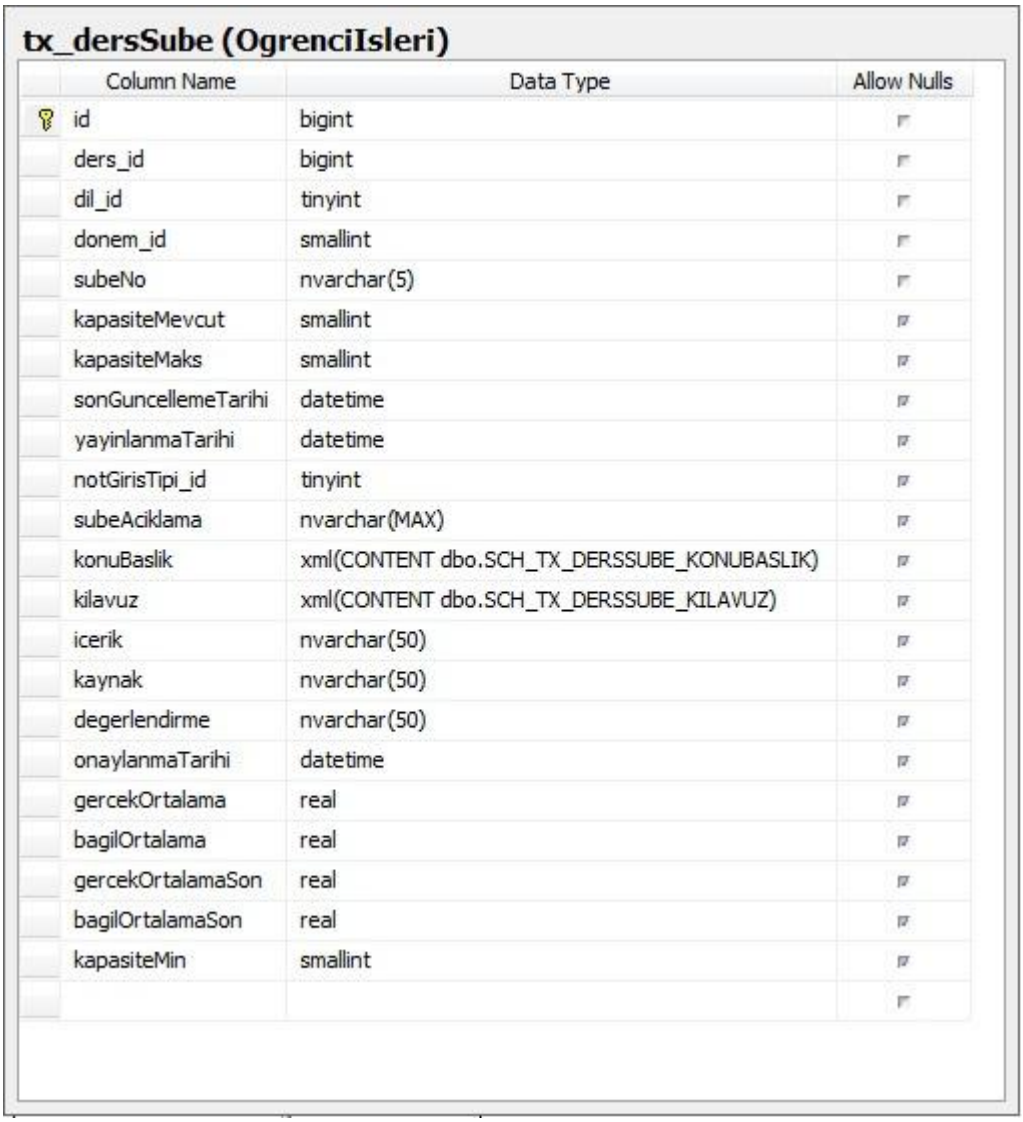
Şekil 2.5 dt_ders tablosu

Şekil 2.6'da verilen dt_dersAd tablosu ise dt_ders tablosuna bağlı olarak buradaki derslerin isimlerini şu an için 2 farklı dilde tutmaktadır.



Şekil 2.6 dt_dersAd tablosu

Şekil 2.7'de gösterilen tx_dersSube tablosu, sınav yerleştirmede ana veri olacaktır. Sınavlar, bu tablodaki id'ler üzerinden yerleştirilecektir. Tabloda dersler belirli şube numaralarıyla açıldığı dönemde yer almaktadır. Diğer alanlar program için anlamsızdır.

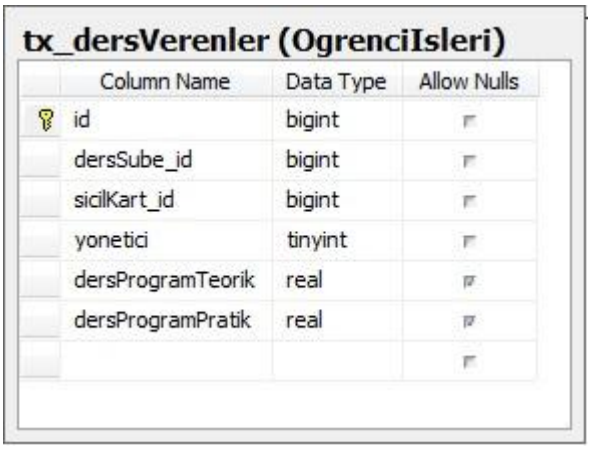


tx_dersSube (OgrenciIsleri)

Column Name	Data Type	Allow Nulls
id	bigint	
ders_id	bigint	
dil_id	tinyint	
donem_id	smallint	
subeNo	nvarchar(5)	
kapasiteMevcut	smallint	
kapasiteMaks	smallint	
sonGuncellemeTarihi	datetime	
yayinlanmaTarihi	datetime	
notGirisTipi_id	tinyint	
subeAciklama	nvarchar(MAX)	
konuBaslik	xml(CONTENT dbo.SCH_TX_DERSSUBE_KONUBASLIK)	
kilavuz	xml(CONTENT dbo.SCH_TX_DERSSUBE_KILAVUZ)	
icerik	nvarchar(50)	
kaynak	nvarchar(50)	
degerlendirme	nvarchar(50)	
onaylanmaTarihi	datetime	
gercekOrtalama	real	
bagilOrtalama	real	
gercekOrtalamaSon	real	
bagilOrtalamaSon	real	
kapasiteMin	smallint	

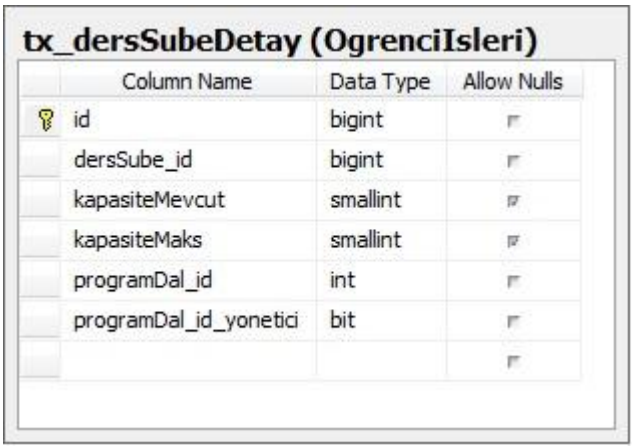
Şekil 2.7 tx_dersSube tablosu

Şekil 2.8'deki tx_dersVerenler tablosu açılan şubenin hangi öğretim elemanına bağlı olduğu bilgisini içermektedir.



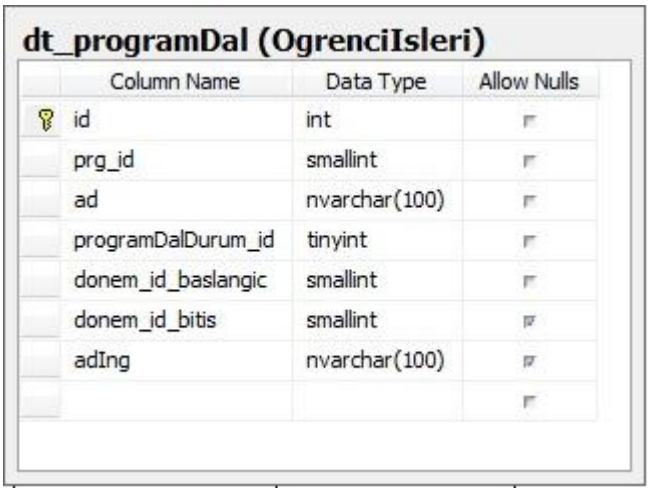
Şekil 2.8 tx_dersVerenler tablosu

tx_dersSubeDetay tablosu açılan şubenin hangi bölüme ait olduğu bilgisini tutmaktadır. Tablo şekil 2.9'da detaylandırılmıştır.



Şekil 2. Error! Use the Home tab to apply 0 to the text that you want to appear here.**9**
tx_dersSubeDetay tablosu

Şekil 2.10'daki dt_programDal tablosu tx_dersSubeDetay tablosunda yer alan programdal bilgilerini içerir. Bölüm verileri, birçok tablonun bağlanmasıyla oluşmakta olup bu proje için sadece dt_programDal tablo verisi yeterli olacaktır.

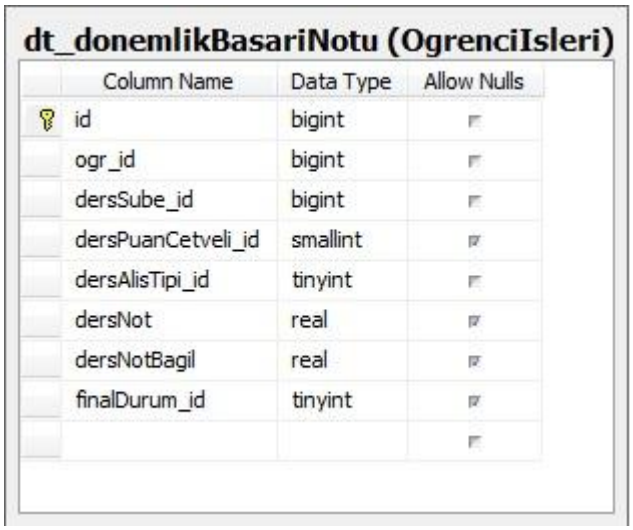


Şekil 2.3 dt_programDal tablosu

Şekil 2.11'deki dt_ogrenci tablosu, öğrenci verilerinin tutulduğu tablodur. Bu tablo, sınav yerleşimi bittikten sonra öğrencilerin sınıflara yerleştirilmesi sırasında kullanılacaktır.

Şekil 2.4 dt_ogrenci tablosu

Şekil 2.12'deki dt_donemlikBasariNotu tablosu belirlenen dönemlerde hangi öğrencinin hangi dersi aldığını gösterir. Buradan yerleştirilecek sınava kaç öğrenci gireceği bilgisi alınıp, sınıflara bu bilgilere göre yerleştirilir.



Şekil 2. Error! Use the Home tab to apply 0 to the text that you want to appear here.**5 dt_donemlikBasariNotu tablosu**

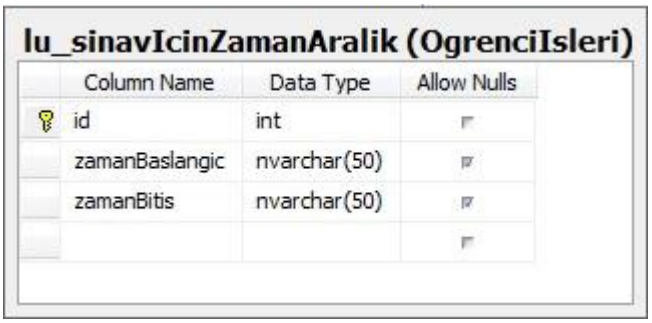
3.3 Eklenen Tablolar

Var olan bu tablolara ek olarak proje için aşağıdaki tablolar oluşturulmuştur. Bunların bazıları Pusula sistemindeki veriler tamamlanınca kullanılmayacak olan geçici tablolardır.

Şekil 3.1'deki lu_subeSinavTur tablosunda yer alan tanımlar teorik ve uygulama şeklinde 2 adettir. Bu tablo, tek bir dersin teorik ve uygulama olarak birden fazla sınavı yapılmak istendiğinde bu sınavları ayırt etmekte kullanılacaktır.

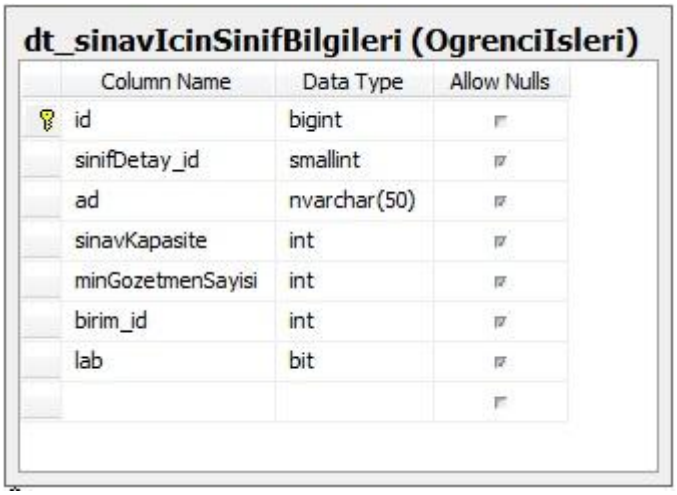
Şekil 3. 1 lu_subeSinavTur tablosu

Şekil 3.2'deki lu_sinavIcinZamanAralik tablosu 08.00'dan 19.00'a kadar birer saat farkla aralıkların tanımlı olduğu tablodur. Bu tablo, dersin sınav süresine bağlı olarak sınavın yerleşim tablosunda yer alacaktır.



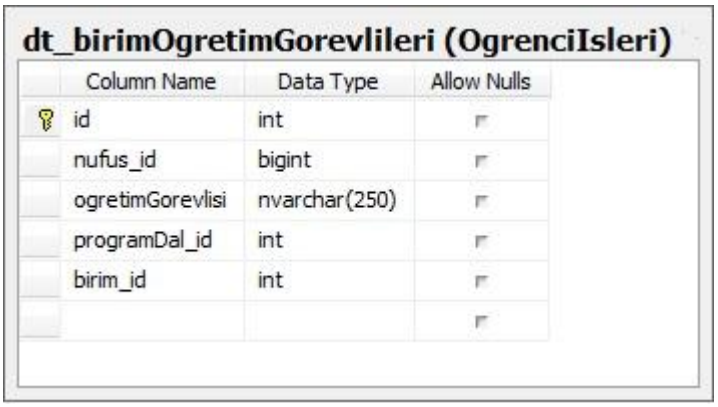
Şekil 3. 2 lu_sinavIcinZamanAralik tablosu

Şekil 3.3'deki dt_sinavIcinSinifBilgileri projede kullanılan ancak geçici bir tablodur. Burada yer alan veriler, farklı tablolardaki tablo bilgilerinin birleştirilmiş halidir. Üniversitenin taşınmazlar projesi hayata geçirildiği zaman bu tablo verileri doldurulacaktır. Ancak proje içinde bu veriler fonksiyon yardımıyla kullanıldığı için, veriler tamamlandığında yapılacak tek değişiklik fonksiyon içinde; çekilen verinin konumunu değiştirmek olacaktır. Burada sınıfın adı, sınav kapasitesi, o sınıf için gerekli minimum gözetmen sayısı, sınıfın hangi birimde yer aldığı (mühendislik, fen edebiyat, vb…), son olarak da laboratuar sınıfı olup olmadığı bilgisi tutulmaktadır.



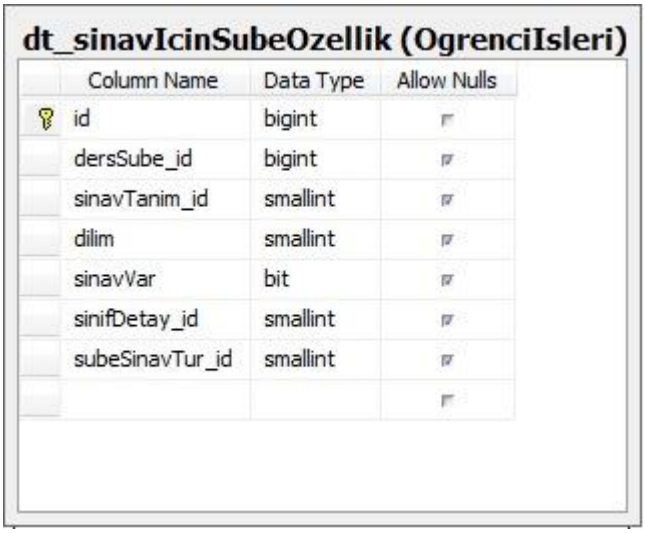
Şekil 3. 3 dt_sinavIcinSinifBilgileri tablosu

Şekil 3.4'deki dt_birimogretimGorevlileri tablosu sınavlarda gözetmen olarak yer alacak öğretim elemanlarının listesini tutar. Burada hangi öğretim elemanının hangi programa ve birime bağlı olduğu bilgisi de bulunmaktadır.



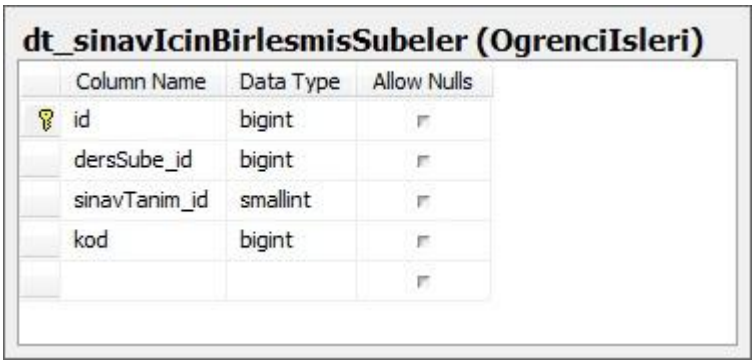
Şekil 3. 4 dt_ birimogretimGorevlileri tablosu

Şekil 3.5'deki dt_sinavIcinSubeOzellik tablosunda derslerin, hangi sınav zamanında, kaç saat süreceği, yapılıp yapılmayacağı, önceden belirli ise hangi sınıfta yapılması gerektiği ve yapılacaksa ek uygulama sınavı bilgileri yer alır.



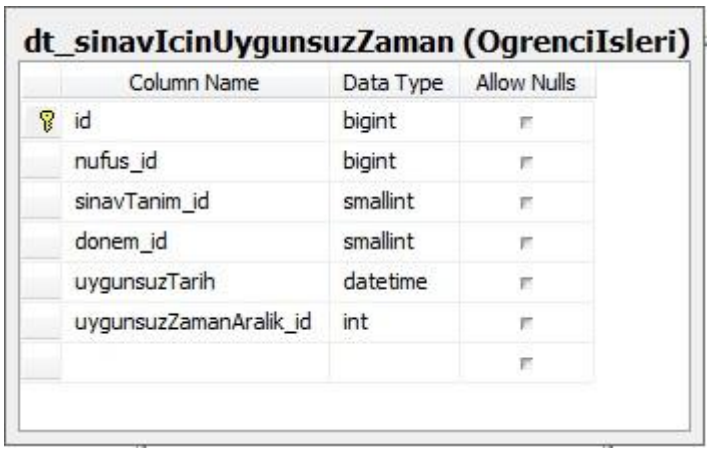
Şekil 3. 5 dt_ sinavIcinSubeOzellik tablosu

Şekil 3.6'daki dt_sinavIcinBirlesmisSubeler tablosunda sınavı aynı zamanda yapılmak istenen dersler toplanır. Böylece örnek olarak aynı öğretim elemanının birleştirilmiş dersleri aynı gün ve saate yerleştirilip burada çakışma sorununa bakılmamış olunur.



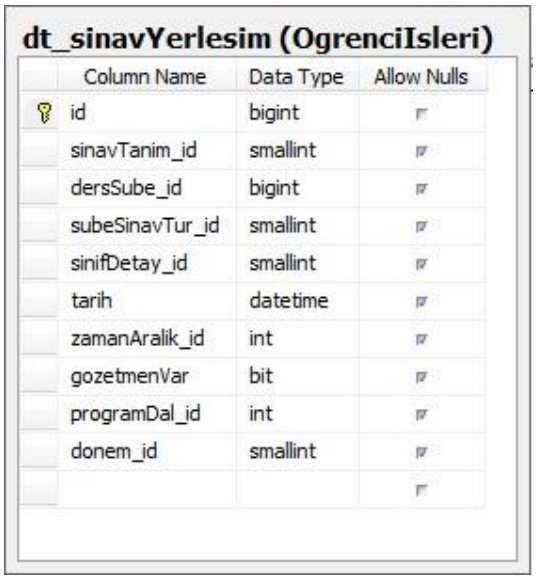
Şekil 3. 6 dt_ sinavIcinBirlesmisSubeler tablosu

Şekil 3.7'deki dt_sinavIcinUygunsuzZaman tablosunda zorunluluk halinde öğretim elemanlarının, hangi dönemde, hangi sınav zamanında, hangi gün ve saatlerde sınava giremeyeceği belirlenir. Program buna göre öğretim elemanlarının sınavlarını belirttikleri tarihlere yerleştirmez.



Şekil 3. 7 dt_ sinavIcinUygunsuzZaman tablosu

Şekil 3.8'deki dt_sinavYerlesim tablosu, program, çalışmasının ilk aşamasını bitirdikten sonra sınav yerleşimini yazacağı tablodur. Burada derslerin hangi programa bağlı olduğu(gözetmen atama sırasında kullanılacak), hangi dönemde, hangi sınav zamanında(ara sınav, final, vb..), hangi sınıfta, hangi tarih ve saatte sınavının yapılacağı bilgisi yer almaktadır.

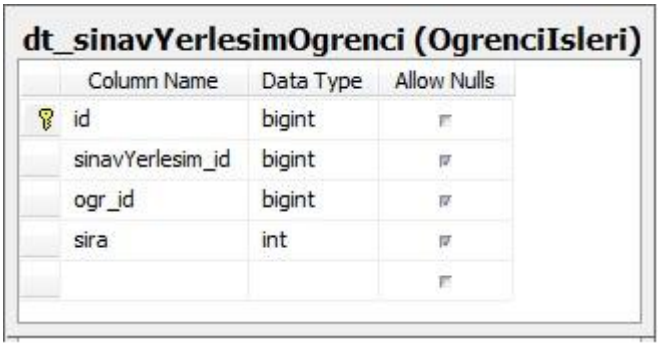


Şekil 3. 8 dt_ sinavYerlesim tablosu

Şekil 3.9'daki dt_sinavYerlesimGozetmen tablosu, sınavların yerleşimi bittikten sonra gözetmenlerin belirlenmesiyle doldurulacak tablodur. dt_sinavYerlesim ve dt_nufus tabloları verilerinin birleşimiyle oluşmaktadır. Bir sınava birden fazla gözetmen atanabileceği için ayrı bir tablo olarak tasarlanmıştır.

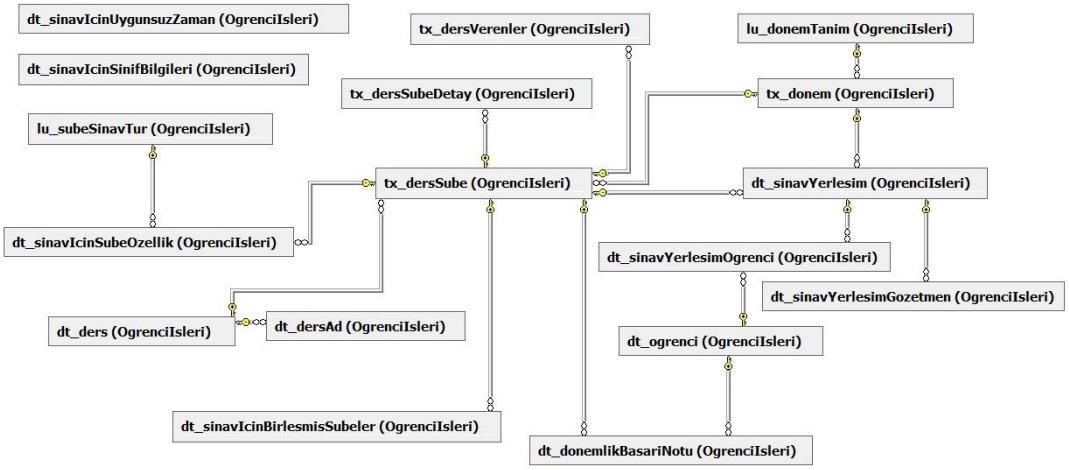
Şekil 3. 9 dt_ sinavYerlesimGozetmen tablosu

Şekil 3.10'daki dt_sınavYerlesimOgrenci tablosu da yine dt_sinavYerlesim tablosu id'sini alarak bu sınav sınıfında hangi öğrencilerin sınava gireceği bilgisini tutmaktadır.



Şekil 3. 10 dt_ sınavYerlesimOgrenci tablosu

Oluşturulan ilişkisel veritabanı modeli Şekil 3.11'de verilmiştir.



Şekil 3. 11 İlişkisel veritabanı modeli

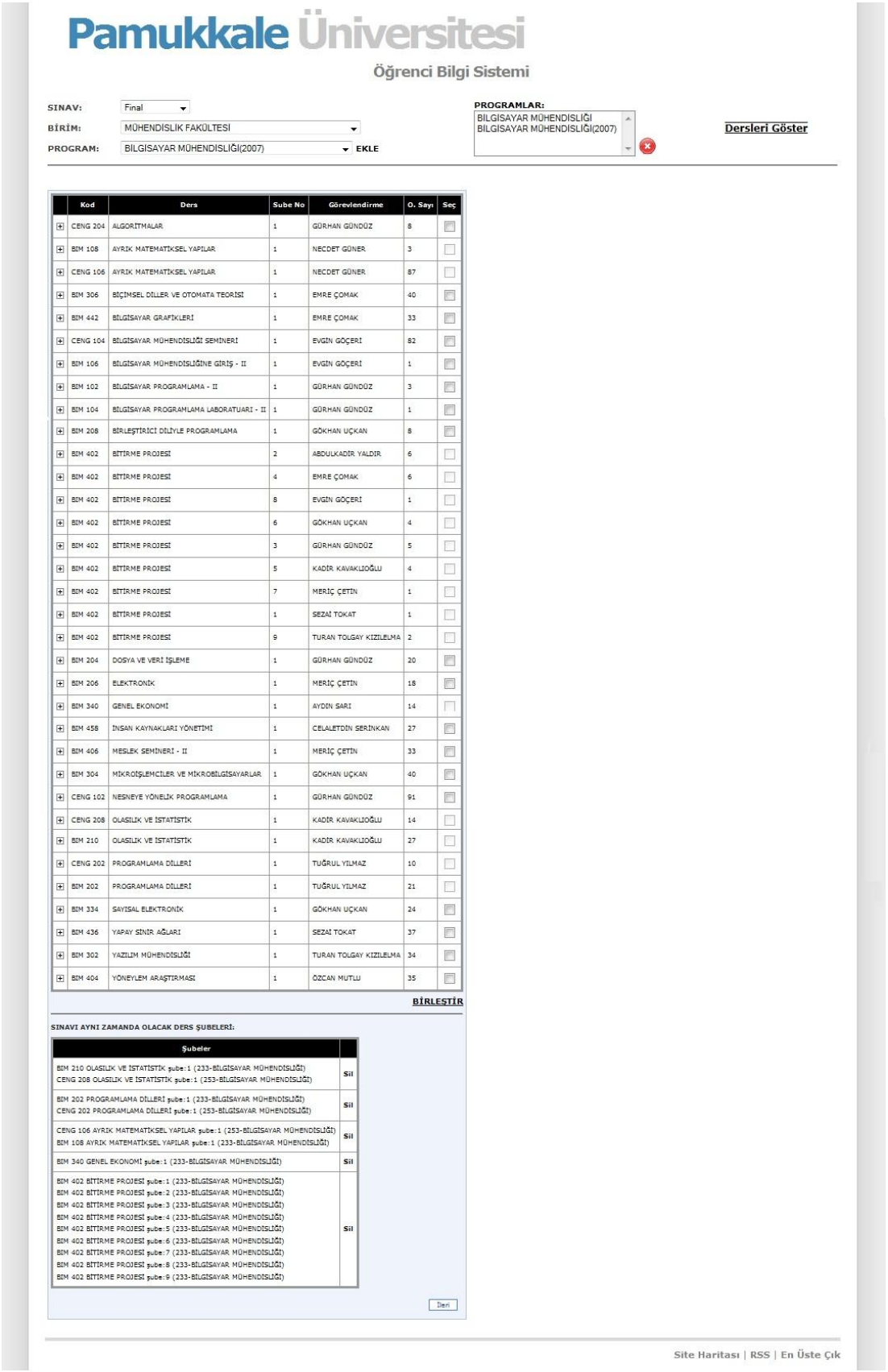
4. GELİŞTİRİLEN UYGULAMA

4.1 Veri Toplama

Sınav programı hazırlayabilmek için öncelikle sınavı yapılacak tüm dersler özellikleriyle birlikte sisteme depolanmalıdır. Programın bu modülü hangi sınavın kaç saat boyunca süreceğini ya da istenen özel bir sınıf olup olmadığı bilgilerini alır. Bu özellikler, bölüm ve fakülte sınav koordinatörleri tarafından web ortamında hazırlanmış ara yüzden girilir. Buradaki temel amaç sınavların otomatik diziliminde varsa özel durumların belirtilebilmesi, programın kalıplaşmış olmaktan kurtarılmasıdır. Kısaca program, veriler içine gömülmesindense, farklı durumlarda da kullanılabilir hale getirilmelidir.

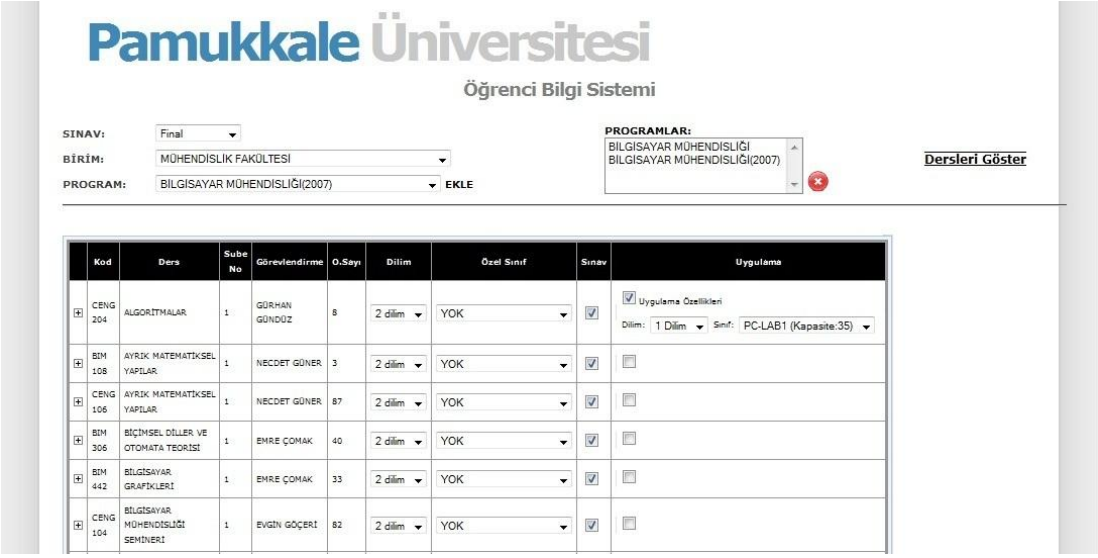
4.1.1 Sınav özellikleri

Gerekli ders bilgilerinin toplanması için fakülte ve bölüm koordinatörlerine açılacak ek bir web programı hazırlanmıştır. Bu web program ara yüzü örnekleri Şekil 4.1, 4.2 ve 4.3'de sırasıyla gösterilmektedir. Sınav programı oluşturmada derslerde öncelikli bilinmesi gereken hangi ders şubelerinin (öğrenci sayısına göre ya da farklı öğretim elemanlarının aynı dersi vermesi nedeniyle parçalanan-farklı zamanlarda işlenen-dersler gibi), hangileriyle beraber yapılmak istendiğidir. Mesela "Atatürk İlkeleri ve İnkılâp Tarihi" dersi bir fakültenin tüm bölümlerinde farklı gün ve saatlerde, farklı öğretim elemanlarıyla okutulabilmektedir. Ancak genelde tüm fakülte, sınavı tek bir günde gerçekleştirir. Ya da bir öğretim elemanı, aynı dersi normal ve ikinci öğretimden 2 sınıfa veriyor olabilir, ancak bu dersin sınavını aynı zamanda yapmak isteyebilmektedir. Bu tarz birleştirmeler yapılmadığı durumda program bu sınavları aynı öğretim görevlisine ait olduğu için çakıştırmamaya yani aynı gün ve saate koymamaya çalışacaktır. Bu durumda öğretim elemanı 2 farklı sınav hazırlamak zorunda kalacaktır. İlk olarak hazırlanacak sınav tipi, daha sonra da şubeleri görüntülenmek istenen bölümler seçilir ve buradan birleştirilmek istenen ders şubeleri seçilip kaydedilir. Bu ekranda bölüm koordinatörleri için seçilen bölümlerin kodunu taşıyan dersler gösterilir. Fakülte koordinatörü içinse, koordinatörün seçtiği fakülteye ait olan, ama bölüm kodu taşımayan, yani MAT, AIT gibi ortak alan dersleri gösterilir. Bu şekilde farklı ekranlara aynı ders düşmesi ve farklı birleştirmeler yapılması engellenmiş olur.

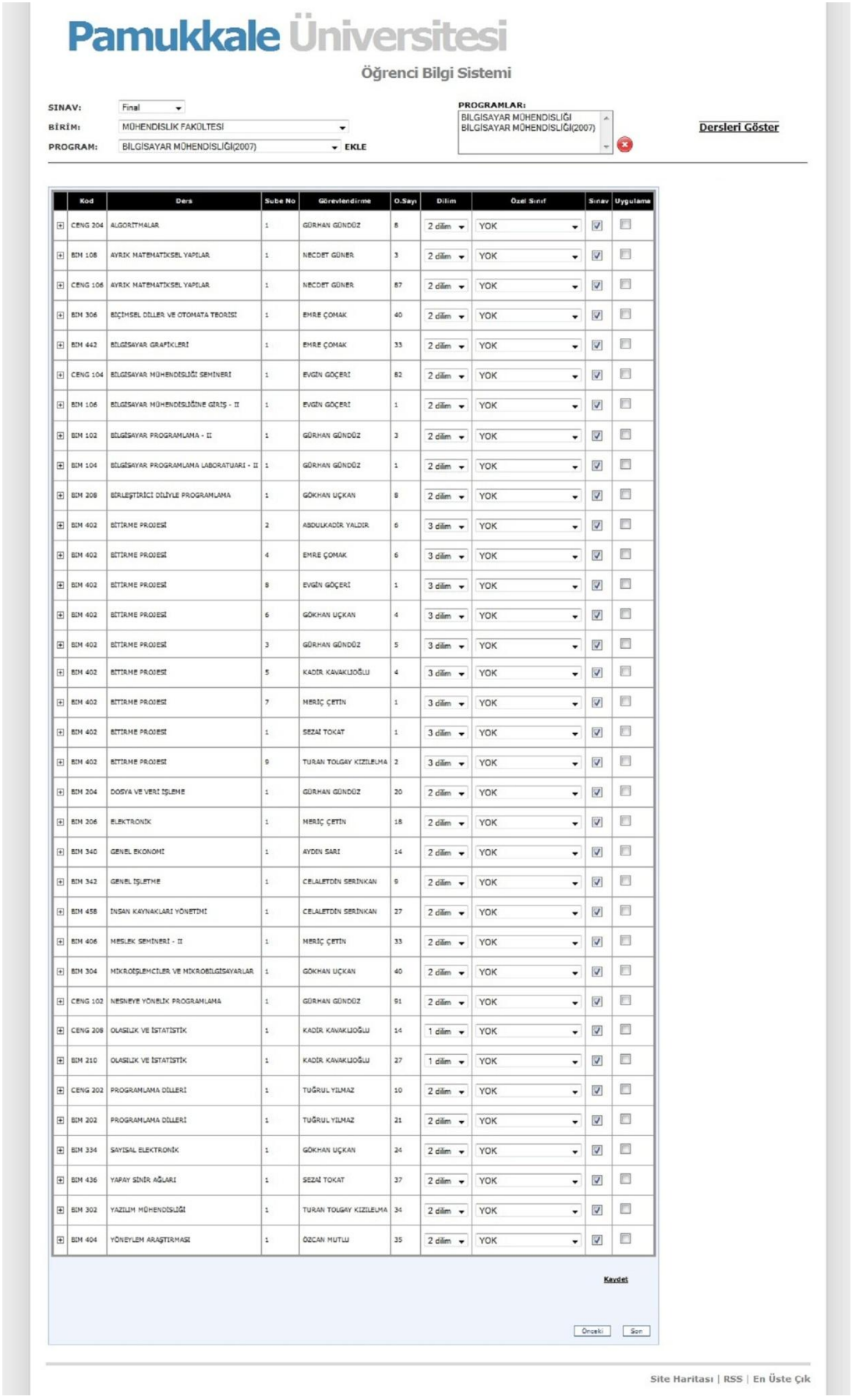


Şekil 4.1 Ders birleştirme ara yüzü

Şube birleştirme işlemi bittikten sonra sınav yapılacak derslerin listesi birleşmiş halleri ile gelir. Bu aşamada da sınavın kaç saat süreceği, isteniyorsa hangi sınıflarda yapılacağı bilgilerinin girilmesi beklenir. Dilim, sınavın süreceği saat bilgisidir. Ayrıca seçili sınav döneminde sınavı yapılmayacak dersler de bu kısımda belirlenir. Son olarak, dersin normal sınavına ek olarak uygulama sınavı da yapılmak istenirse; uygulama alanı seçilir ve aynı özellikler uygulama sınavı için de doldurulur. Bu işlemlerin yapılacağı ara yüzler Şekil 4.2 ve 4.3'de verilmiştir.



Şekil 4.2 Ders uygulama sınavı özellik giriş ara yüzü



Şekil 4.3 Ders özellik giriş ara yüzü

4.1.2 Öğretim elemanları için uygun olmayan zaman dilimleri

Ek olarak, sadece fakülte koordinatörünün göreceği ekranda öğretim elemanlarının sınav yapamayacakları ya da yapmak istemedikleri günler özel şartlar altında kaydedilir. Burada arama yapılarak istenen öğretim elemanı seçilir ve bu öğretim elemanının seçili sınav döneminde sınav yapmak istemediği gün ve saatler seçilerek kaydet butonuna basılır. Şekil 4.4'de arayüz gösterilmektedir.

Şekil 4.4 Öğretim elemanı uygun olmayan zaman giriş ara yüzü

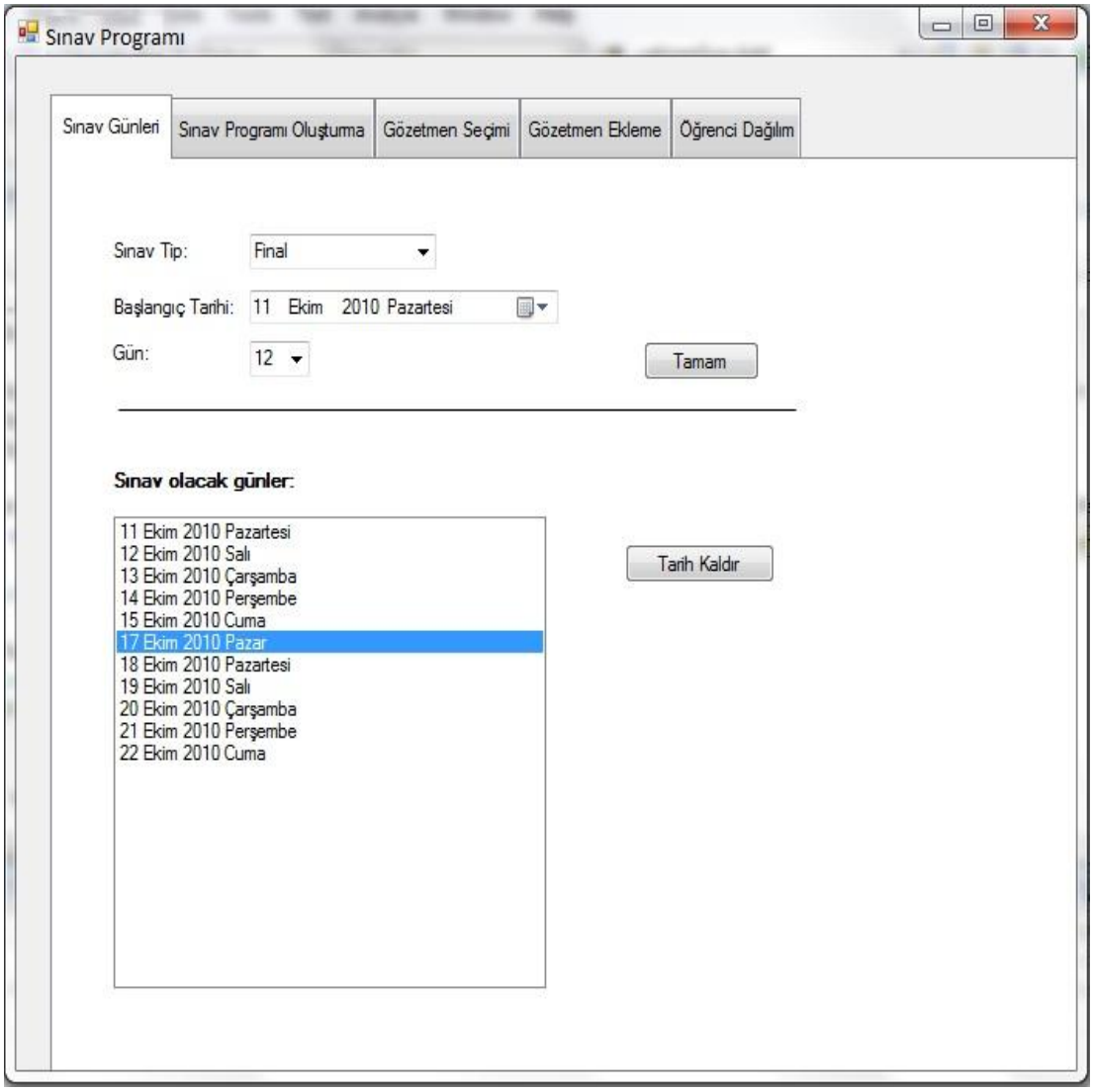
4.2 Sınav Programı

Sınav programı çalışmasının temel ekranı bir masaüstü uygulaması olarak tasarlanmıştır. Bunun en önemli sebebi program çalışma süreci uzun olması beklendiğinden, web ortamındaki "zaman aşımı süresi doldu" hatasıyla karşılaşmak istenmemesidir. Bu masaüstü uygulaması sadece fakülte koordinatörlerinin

bilgisayarlarına kurulacak olup, istenen tüm veriler girildikten sonra sınav takviminin oluşturulması tek bir kişinin sorumluluğunda olacaktır.

4.2.1 Sınav günleri

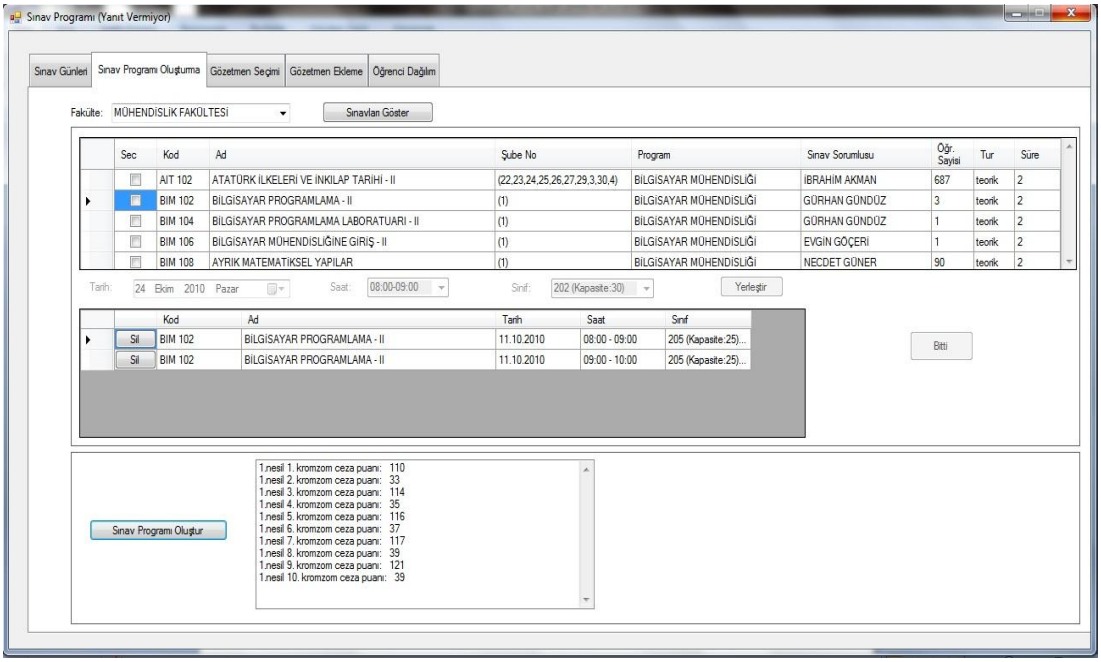
Program çalıştırılmadan önce ilk yapılacak işlem, sınav günlerinin belirlenmesidir. Sınav tipi (Ara Sınav, Final, vb…) seçiminden sonra sınav zamanının başlayacağı tarih ve bu zamanın süresi gün olarak girilir. Bu şekilde başlangıç gününden itibaren seçili gün kadar sınav günü listelenir. Eğer liste içinde sınav yapılmak istenmeyen resmi tatil günleri ya da hafta sonu tatilleri varsa çıkartılır. Sonuç olarak listede programın kullanacağı günler sabitlenmiş olur. Sınavın yapılacağı günlerin seçildiği arayüz Şekil 4.5'de sunulmuştur.



Şekil 4.5 Sınav günü belirleme ara yüzü

4.2.2 Sınav programı oluşturma

Gün seçimi yapıldıktan sonra, program çalışmaya başlatılmadan, belirli gün, saat ve sınıfta yapılması özellikle istenen bir sınav varsa burada yerleştirilebilir. Önceden yerleştirme işlemi bittiğinde sınav program oluştur butonu kullanılabilir hale gelecektir. Durum Şekil 4.6'da gösterilmektedir.



Şekil 4.6 Sınav programı oluşturma ara yüzü

4.2.3 Veri oluşturma

Programda ilk olarak sınavların yerleşeceği tablo, datatable olarak oluşturulur. Burada başta belirlenmiş günler, seçili fakültenin sınıflar ve 08.00'dan 19.00'a kadar saatler birer saatlik parçalar halinde kullanılır. Bu şekilde dt_yerlesim datatable'ı örnek olarak Tablo 4.1'deki gibi oluşturulur. "dolu" alanı o gün, saat ve sınıfta yerleşmiş sınav olup olmadığını anlamak için kullanılacaktır. Bu yerleşim alanlarından, herhangi bir sınava atanan yerin "dolu" alanı "true" değerine çevrilir. Diğer sınavlara atanacak yerler ise "dolu" alanı "false" olan satırlar arasından seçilir.

id	gün	sınıf	saat	dolu
1	11. 10. 2010	202	08.00-09.00	False
2	11. 10. 2010	202	09.00-10.00	False
3	11. 10. 2010	203	08.00-09.00	False
4	11. 10. 2010	203	09.00-10.00	False
5	12. 10. 2010	202	08.00-09.00	False
6	12. 10. 2010	202	09.00-10.00	False
7	12. 10. 2010	203	08.00-09.00	False
8	12. 10. 2010	203	09.00-10.00	False
9	12. 10. 2010	204	08.00-09.00	False
10	12. 10. 2010	204	09.00-10.00	False

Tablo 4.1 Yerleşim planı tablosu

Eğer önceden sınav yerleştirme alanında yerleştirilmiş sınav varsa; o sınavların yerleştiği gün, saat ve sınıflar bu tabloda yer almaz. Birleştirilmiş şubeler özelliklerinin (öğrencileri gibi) birleştirilmiş haliyle, oluşturulan Sube nesnesine atanır. Sube nesnesinin özellikleri sube_id, şube öğrencileri, şubenin ait olduğu öğretim elemanı, şubenin bağlı bulunduğu program, şubenin sınav süresi, varsa yapılmak istenen özel sınıf bilgisi gibi verilerdir. Sube sınıf yapısı Ek 1'de yer almaktadır. Aynı şekilde önceden yerleştirilmiş sınavlar bu listede yer almaz.

4.2.4 Kromozom oluşturma

Veriler yerleştikten sonra EH'nin ilk ayağı olan kromozom oluşturulur. Burada kromozom 2 boyutlu int bir dizidir. Bu dizi tanımı int [,]. kromozom = new int [şube sayısı, yerleşim sayısı]. şeklindedir.

Kromozom [0,0]. değeri Sube nesnesinin 0. indeksindeki şubenin atandığı yerleşim alanının datatable üzerindeki id'sini verir. Eğer Sube nesnesinin 0. indeksindeki şubenin sınavı birden fazla saate ya da öğrenci sayısı nedeniyle birden fazla sınıfa yerleşecekse yerleşim tablosunda birden fazla yere atanacak demektir. Bu durumda yerleşim tablosuna yerleşeceği diğer id kromozom[0,1]. alanına atanır. Şubenin nesnesinin 0. indeks şubesi yerleştikten sonra 1. indeksindeki şube yerleşimi kromozom[1,0]. alanına atanarak yerleşim devam eder. Program içinde bu şekilde olası çözümlerin atandığı 10 kromozom oluşturulur. Oluşturulan bu 10 kromozom ilk nesil ebeveynleri temsil eder.

4.2.5 Uygunluk fonksiyonu

Oluşturulan olası çözümlerin doğru çözümlerden birini verip vermediğini bulabilmek için bir uygunluk fonksiyonu oluşturulur. Burada uygunluk fonksiyonu, olası çözüm

içerisinde çakışmalara ve istenmeyen durumlara neden olan yerleşim sayısının ceza puanı olarak hesaplanmasıdır. Bu nedenle bizim çözümümüz uygunluk fonksiyonunun 0 olduğu durumlarda bulunmuş demektir. Ceza puanı hesaplanırken 4 ana problem dikkate alınır. Bunlardan ilki aynı gün ve aynı saate yerleştirilmiş sınavların ortak öğrencisinin olmasıdır. Öğrenci çakışması denen bu durumda aynı gün ve saatteki sınavları alan tek bir öğrenci olsa dahi bu sınavlardan biri olası çözümün ceza puanına eklenir. İkinci problem ise bir öğretim elemanın aynı gün ve saat içerisinde birden fazla sınav yapması durumudur. Bu durumda da aynı güne yerleştirilmiş sınavlardan biri öğretim elemanı çakışması olarak olası çözümün ceza puanına eklenir. Diğer bir problem de normal öğretim sınavlarının saat 17.00'dan sonra yapılmasıdır. Bu durumda sınav yapacak öğretim elemanına ek mesai ücreti ödenmesi gerektiğinden normal öğretim sınavları 17.00'dan önce bitmelidir. Yine sınav saatlerinden herhangi biri 17.00 ve sonrasına denk gelen normal öğretim sınavları olası çözümün ceza puanına eklenecektir. Son olarak kontrol edilen problem önceden kaydedilen, öğretim elemanının uygun olmadığını belirttiği saatlere yerleştirilen sınavlarının olması halidir. Bu zamanlara yerleştirilen sınavlar da ceza puanına yazılır. Sonuç olarak burada belirtilen 4 çakışma sorununa da cevap verebilen çözüm, istenen çözümlerden biri olacaktır.

4.2.6 Mutasyonlar

Ebeveynlerden yeni nesil çocuklar üretebilmek için olası çözüm örnekleri olan kromozomlara bazı mutasyonlar uygulanmıştır. Bu mutasyonlar tamamen ceza puanına katkı sağlayan sorunlar üzerinden oluşturulmakla beraber 4 adettir. Mutasyonların adları farklı olsa da mantıkları tamamen aynıdır. Her biri işlem yapacağı kromozomun sorunlu sınavına farklı bir yerleşim id'si atar.

- **Öğrenci çakışması**

Aynı sırada gidersek ilk mutasyonumuz "ogrenciCakismaMutasyonu" adını alır. Bu mutasyonda ortak öğrencisi bulunan ve aynı gün ve saate atanmış derslerden birinin yerleşim yeri (id'si) değiştirilir. Burada değişiklik yapılan sınavın yeni yerleşim yeri, önceki gün ve saatin dışındaki alandan seçilmektedir. Bu şekilde akıllı mutasyonla çözüme ulaşmayı hızlandırmış oluruz.

- **Öğretim elemanı çakışması**

İkinci çalışma durumu bir öğretim elemanının aynı günde birden fazla sınav yapması haliydi. Bu nedenle ikinci mutasyona "ogretimElemaniCakismaMutasyonu" adı verildi. Bu mutasyonda da değişiklik yapılan sınava önceki yerleşim gününden farklı günlerden biri atanmaktadır.

- **Öğretim elemanı uygunsuz saat ataması**

Diğer mutasyonumuz, sınavı 17.00'da ve sonrasına yerleştirilmiş normal öğretim derslerinin sayısını sıfırlamaya çalışmaktadır. Burada da sorunlu sınavların yerleşimleri, saat 17.00'dan önceki yerleşim yerleriyle değiştirilir. Bu mutasyona da "NormalOgretimCakısmaMutasyonu" denmiştir.

- **Normal öğretim**

Son mutasyon çeşidi de öğretim elemanlarının, istemediği zamanlarda yerleştirilen sınavlarını, değiştirmeye yöneliktir. "UygunsuzZamanMutasyonu" denen bu mutasyonda da değişecek sınav zamanı, öncekinden farklı olan yerleşim alanları arasından seçilir.

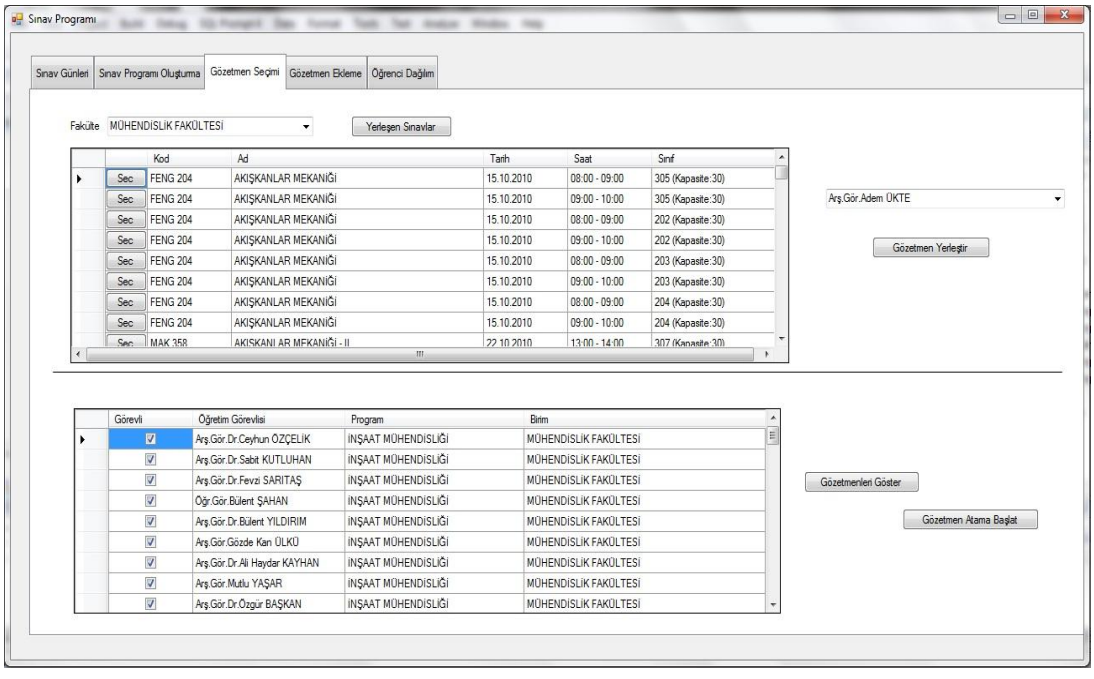
4.2.7 Yeni nesil

Ebeveyn oluşumundan sonra, ilk çocuklar bu ebeveynlerin mutasyona uğratılmasıyla oluşturulur. 10 ebeveyn kromozomunun mutasyonlarla 10 çocuk kromozomu olur. Bu kısımda çocuk kromozomlarının her biri uygunluk fonksiyonuna sokulur. Buradan istenen sonuç elde edilmezse; bir döngü oluşturulur ve burada, oluşan 10 çocuk ve 10 ebeveynin en iyi 5'er elemanı yeni nesle geçer. Bu şekilde uygunluk fonksiyonu değeri en düşük olan 10 kromozom, yeni nesli oluşturmuş olur. Bu yeni nesil döngü içindeki yeni ebeveyndir. Daha sonra bu ebeveynlerin her biri tekrar mutasyona uğratılarak yeni çocuklar oluşturulur. Çözüm bu çocuklar arasında da değil ise, yeni nesil ebeveyn ve çocukların en iyileri yani uygunluk fonksiyonu en düşük olanlarından yeniden oluşturulur. Bu döngü uygun çözüm bulunana kadar sürer. Her mutasyon sonucu kesin olmamakla birlikte bir öncekinden daha iyi sonuçlar çıkmaktadır. Ve her yeni nesil önceki nesillerin en iyileri arasından seçilmektedir. Bu nedenle çözüme en yakın olası çözüm hiçbir zaman kaybedilmeyerek hep ileriki nesillere taşınır.

Tüm çakışmaların sona erdiği yani ceza puanının 0 olduğu bir çözüm bulunduğunda döngüden çıkılarak çözüm veritabanına yazılır.

4.2.8 Gözetmen atama

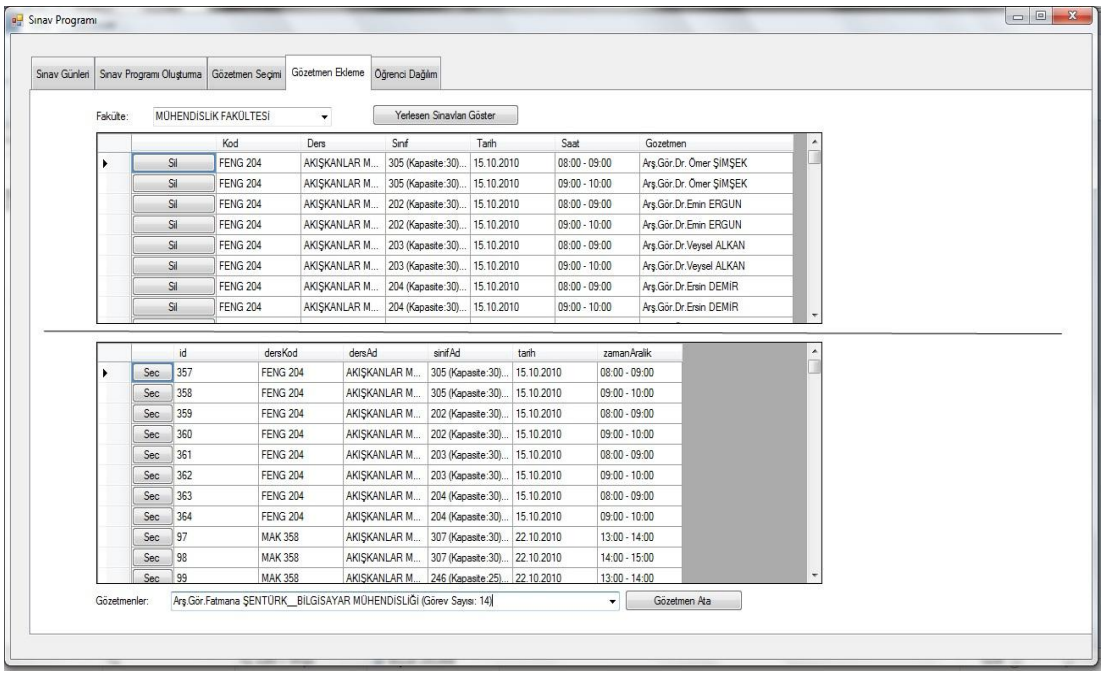
Sınavların hangi gün ve saatte hangi sınıfta yapılacağı yerleştirildikten sonra, sıra gözetmen belirleme işlemine gelmiştir. Bu üçüncü kısımda istenirse belirli gözetmenler, belirli sınav sınıflarına önceden yerleştirilebilirler. Ancak sınav saati ne kadar ise gözetmenin, sınavın sınıfına o süre boyunca yerleştirilmesine dikkat edilmelidir. Önceden gözetmen yerleştirme işlemi bittikten sonra "Gözetmen Göster" butonuyla seçili fakülteye bağlı gözetmenlik yapacak öğretim elemanlarının listesi görüntülenir. Burada tüm öğretim elemanları görevli konumunda gelir ancak o sınav döneminde raporlu olan ya da sınavlara giremeyecek olan kişilerin görevli kısmındaki tik kaldırılır. Bu şekilde gözetmen atama butonuna basıldığında önceden yerleştirilmiş sınav sayısı, listede seçili olan gözetmen sayısına bölünüp kişi başına düşen görev sayısı bulunarak program başlatılmış olur. Arka planda ilk olarak görevli listesi datatable şeklinde görev sayıları başta 0 olmak üzere atanır. Daha sonra yerleşen sınav listesi sırasıyla taranarak, bunlar görev sayısı tamamlanmamış kendi bölümümün öğretim elemanı varsa öncelik tanınacak şekilde eşleştirilir. Tüm sınavlara gözetmen ataması yapıldıktan sonra, yerleşen gözetmenlerin aynı gün ve saatte görevlendirmeleri olup olmadığı kontrol edilir. Bu çakışmalar sıfıra inene kadar bir döngü içerisinde gözetmenler birbiriyle yer değiştirir. Aynı gün ve saatte görevli gözetmen kalmadığında gözetmen atama işlemi bitmiş olur. Bu işlemler de Şekil 4.7'deki arayüzden yapılmaktadır.



Şekil 4.7 Gözetmen atama ara yüzü

4.2.9 Gözetmen değişikliği

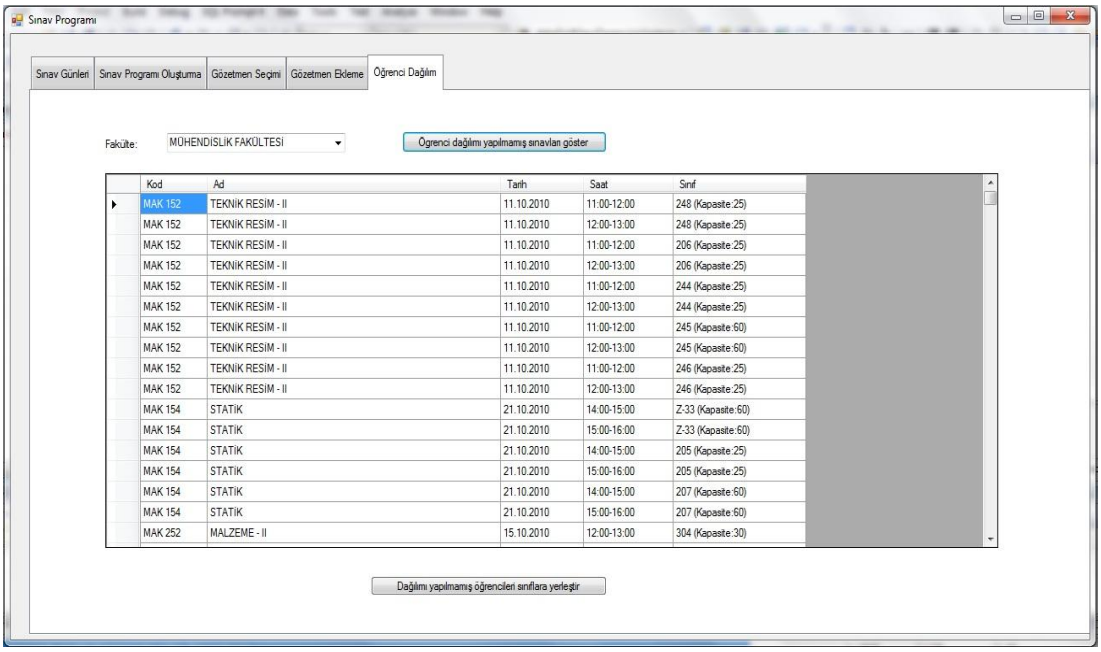
Gözetmen atamaları bittikten sonra istenmeyen durumlar karşısında gözetmenlerin görevleri birbirleriyle bu alanda değiştirilebilir. İlk listede sınavlar görevli gözetmenleriyle yer almaktadır. Buradan istenen sınav silinerek aynı sınav bir sonraki listeden seçilir ve istenen gözetmenle tekrar eşleştirilebilir. Şekil 4.8'de gösterilen bu ekranda ayrıca hiçbir gözetmen silinmeden; istenen sınıflara ek gözetmen de atanabilir.



Şekil 4.8 Gözetmen değişikliği yapma ara yüzü

4.2.10 Öğrenci dağılım

Sınavlar zamanları ve gözetmenleriyle birlikte belirlendikten sonra hangi öğrenci hangi sınıfta sınava gireceği bilgisi de bu ekrandan elde edilebilir. Burada, fakülte seçimi yapıldıktan sonra henüz öğrencileri yerleştirilmemiş sınavlar listelenir. "Dağılımı yapılmamış öğrencileri sınıflara yerleştir" butonuna basıldığında da program sınavları sırasıyla tarayarak bu sınavlara girecek öğrencileri numara sırasına göre listeler. Ve sınıfların kapasitesine göre öğrenci listesinin bir üstünden, bir altından öğrenci almak suretiyle sınıflara öğrenci yerleşimini gerçekleştirir. İşlemin yapıldığı arayüz Şekil 4.9'da gösterilmektedir.



Şekil 4.9 Öğrenci dağılım ara yüzü

Çalışma sonucu örnek bir rapor oluşturulduğunda Mühendislik Fakültesi Endüstri Mühendisliği Bölümü için aşağıdaki tablo ortaya çıkmaktadır. Bu şekilde her bölüm için ayrı raporlar gerekli parametreler girildiği takdirde alınabilir. Şekil 4.10'daki raporda hangi dersin sınavının, hangi gün ve saatte, hangi gözetmen eşliğinde yapılacağı rahatlıkla görülebilmektedir.

T.C.
MÜHENDİSLİK FAKÜLTESİ
ENDÜSTRİ MÜHENDİSLİĞİ BÖLÜMÜ
2009-2010 BAHAR YARIYILI
FİNAL SINAV TARİHLERİ

Sınav Tarihi	Sınav Saati	Ders Kodu	Dersin Adı (Öğr. Sayısı)	Sınav Yeri	Gözetmen
11.10.2010 Pazartesi	13:00 - 15:00	IENG 102	ENDÜSTRİ MÜHENDİSLİĞİ UYGULAMALARI (37)	249	ÇİĞDEM ERSAN
					ÖNER ATALAY
	15:00 - 17:00	IENG 102	ENDÜSTRİ MÜHENDİSLİĞİ UYGULAMALARI (37)	249	ÇİĞDEM ERSAN
					ÖNER ATALAY
12.10.2010 Salı	09:00 - 11:00	ENM 212	MÜHENDİSLİK MEKANİĞİ (16)	308	BANU YETKİN EKREN
					HASAN AKYER
13.10.2010 Çarşamba	09:00 - 11:00	ENM 340	VERİTABANI YÖNETİMİ (37)	249	BANU YETKİN EKREN
					OLCAY POLAT
	15:00 - 17:00	IENG 206	MÜHENDİSLER İÇİN İSTATİSTİK (2)	205	BANU YETKİN EKREN
	17:00 - 19:00	IENG 206	MÜHENDİSLER İÇİN İSTATİSTİK (2)	205	BANU YETKİN EKREN
14.10.2010 Perşembe	08:00 - 09:00	ENM 438	KARAR DESTEK SİSTEMLERİ (29)	202	BANU YETKİN EKREN
	09:00 - 11:00	ENM 102	ENDÜSTRİ MÜHENDİSLİĞİ UYGULAMALARI (37)	249	BANU YETKİN EKREN
					OLCAY POLAT
		ENM 436	DENEYSEL TASARIM (27)	202	FİGEN TURAN
	11:00 - 13:00	ENM 102	ENDÜSTRİ MÜHENDİSLİĞİ UYGULAMALARI (37)	249	BANU YETKİN EKREN
					OLCAY POLAT
		ENM 436	DENEYSEL TASARIM (27)	202	FİGEN TURAN
	15:00 - 17:00	ENM 344	DIŞ TİCARET (35)	Z-32	BANU YETKİN EKREN
					OLCAY POLAT
	17:00 - 19:00	ENM 344	DIŞ TİCARET (35)	Z-32	BANU YETKİN EKREN
					OLCAY POLAT
15.10.2010 Cuma	13:00 - 15:00	ENM 306	ÜRETİM PLANLAMA VE KONTROLÜ (48)	207	BANU YETKİN EKREN
					OLCAY POLAT
	15:00 - 17:00	ENM 306	ÜRETİM PLANLAMA VE KONTROLÜ (48)	207	BANU YETKİN EKREN
					OLCAY POLAT
18.10.2010 Pazartesi	09:00 - 11:00	ENM 210	TEMEL ÜRETİM YÖNTEMLERİ (15)	206	BANU YETKİN EKREN
	11:00 - 13:00	ENM 342	TAM ZAMANINDA ÜRETİM VE UYGULAMALARI (45)	Z-33	BANU YETKİN EKREN
					OLCAY POLAT
	13:00 - 15:00	ENM 208	MÜHENDİSLİK EKONOMİSİ (12)	206	BANU YETKİN EKREN
	15:00 - 17:00	ENM 208	MÜHENDİSLİK EKONOMİSİ (12)	206	BANU YETKİN EKREN
	17:00 - 19:00	ENM 210	TEMEL ÜRETİM YÖNTEMLERİ (15)	206	BANU YETKİN EKREN
19.10.2010 Salı	11:00 - 13:00	ENM 204	ERGONOMİ (40)	250	BANU YETKİN EKREN
					OLCAY POLAT
	15:00 - 17:00	ENM 202	YÖNEYLEM ARAŞTIRMASI - I (11)	244	BANU YETKİN EKREN
	17:00 - 19:00	ENM 202	YÖNEYLEM ARAŞTIRMASI - I (11)	244	BANU YETKİN EKREN
20.10.2010 Çarşamba	08:00 - 09:00	CEKO 206	ENDÜSTRİYEL PSİKOLOJİ (2)	206	OLCAY POLAT
	09:00 - 10:00	SBR 161	YÜZME (9)	206	OLCAY POLAT
	09:00 - 11:00	ENM 106	PROGRAMLAMAYA GİRİŞ (1)	244	BANU YETKİN EKREN
		ENM 442	FİNANSAL YÖNETİM (18)	206	OLCAY POLAT
	11:00 - 13:00	ENM 106	PROGRAMLAMAYA GİRİŞ (1)	244	BANU YETKİN EKREN
		ENM 442	FİNANSAL YÖNETİM (18)	206	OLCAY POLAT
	13:00 - 15:00	ENM 444	E - TİCARET (23)	206	OLCAY POLAT
	15:00 - 17:00	ENM 444	E - TİCARET (23)	206	OLCAY POLAT
	17:00 - 19:00	SBR 161	YÜZME (9)	206	OLCAY POLAT
21.10.2010 Perşembe	08:00 - 09:00	CENG 192	GÖRSEL PROGRAMLAMA (19)	244	RAMAZAN HAKAN ÖZCAN
			GÖRSEL PROGRAMLAMA (25)	208	RECEP YURTSEVEN
			GÖRSEL PROGRAMLAMA (30)	305	ARZUM ULUKÖY
		ENM 206	İSTATİSTİK (12)	Z-32	OLCAY POLAT
					SEMİH COŞKUN
	09:00 - 11:00	ENM 404	İŞ GÜVENLİĞİ VE İŞ HUKUKU (30)	202	OLCAY POLAT
	11:00 - 13:00	ENM 404	İŞ GÜVENLİĞİ VE İŞ HUKUKU (30)	202	OLCAY POLAT
22.10.2010 Cuma	08:00 - 09:00	ENM 302	İŞ ETÜDÜ (46)	Z-33	SEMİH COŞKUN
					TURAN TOLGAY KIZILELMA
	09:00 - 11:00	ENM 402	TAHMİN TEKNİKLERİ (35)	249	ENGİN TAN
					TURAN TOLGAY KIZILELMA
	11:00 - 13:00	ENM 402	TAHMİN TEKNİKLERİ (35)	249	ENGİN TAN
					TURAN TOLGAY KIZILELMA

SINAV GÖZETMENLERİNİN ORTAK ZORUNLU DERSLERİN SINAV SAATİNDEN EN AZ 10 DAKİKA ÖNCE MİSAFİR ÖĞRETİM ÜYESİ VE SINAV KOORDİNASYON ODASINDA TOPLANMALARI GEREKMEKTEDİR.

Şekil 4.10 Yerleşen program örnek çıktısı

5. SONUÇ VE ÖNERİLER

Çalışma denemeleri, pilot uygulama alanı olan Pamukkale Üniversitesi Mühendislik Fakültesi'nde 2009–2010 öğretim yılı Bahar dönemi final programı için yapılmıştır. 2010–2011 öğretim yılı Güz dönemi ilk ara sınavı için uygulanması beklenmektedir. Program eldeki verilerle tek bir fakülte üzerinde denenmiş olsa da, yapısı itibariyle gerekli veriler tamamlandığında tüm üniversite için çalıştırılabilecek duruma getirilmiştir.

Projenin uygulanması için üniversite bünyesinde her bölüm için birer bölüm koordinatörü ile her fakülte için de bir fakülte koordinatörü görevlendirilmesi gerekmektedir. Bölüm koordinatörleri sınav döneminden önce sınav birleştirme ve sınav özellikleri girme işlemlerini bitirmelidirler. Aynı şekilde fakülte koordinatörleri de fakültenin ortak alan derslerinin gerekli işlemlerini yapmalıdırlar. Bu işlemler bitiminde fakülte koordinatörleri kendi fakülteleri için bu kademeli sınav programı oluşturma programını çalıştırarak, kendi fakülte sınav takvimlerine ulaşabilirler.

Çalışma, Pamukkale Üniversitesi Öğrenci Bilgi Sistemi veritabanına yer alan tabloların ve ders şubelerine erişilebilinecek bir view'in eklenmesi ile diğer üniversitelerin öğrenci bilgi sistemlerine de entegre edilebilir.

Bu çalışmada, problemin analizi sırasında programı kullanacak kişilerle görüşülüp bu kişilerin farklı isteklerini gerçekleşebilecek bir bütünlük içine sokmak karşılaşılan ilk zorluk olmuştur. Bu dönem tamamlandıktan sonra problemin GA'ya ne şekilde uygulanacağı da zaman alan problemlerden biri olmuştur. Ölçütlerin belirlenip uygunluk fonksiyonunun oluşturulması sıkıntı olmasa da kromozom yapısının oluşturulması düşündürücü bir aşamaydı.

Çalışma sınavları, bu sınavlarda görev alacak gözetmenleri, sınavların yapılacağı sınıflar ile hangi öğrencilerin hangi sınıflarda, kaçıncı sırada sınava gireceği gibi verileri oluşturmaktadır. Sınavlar çakışma olmayacak şekilde rastgele yerleştirilmektedir.

Uygulamaya ek olarak sınavlara zorluk dereceleri verilip gün içindeki sınav saatleri bu derecelere göre düzenlenebilirdi. Ayrıca ileride üniversitedeki sınıf profilleri çıkarıldıktan

sonra ÖSYM sistemine benzer şekilde hangi sıraya hangi öğrencinin oturacağı önceden belirlenebilecek şekilde düzenlenebilir.

Bu tez çalışması, GA yöntemini, daha önce kullanılmayan farklı veri bütünleriyle, farklı olan problem üzerinde çözüme ulaştırmayı öğretmiştir. Böylece GA ile yapılan sıradan zaman çizelgeleme problemlerine, Pamukkale Üniversitesi veritabanı verilerine göre şekillenmiş yeni bir çözüm tekniği eklenmiş oldu.

KAYNAKLAR

[1]. Elmas, Ç., 2003. Bulanık Mantık Denetleyiciler, Birinci Baskı, s. 24, Seçkin Yayınevi, ss. 225,Ankara.

[2]. Rechenberg, I., 1973. Evolution Strategy: Optimization of Technical Systems According to the Principles of Biological Evolution (in German), Frommannn-Holzboog, Stuttgart.

[3]. Holland, J., 1975. Adaptation in Natural and Artificial Systems, University of Michigan Pres, Ann Arbor, Mich.

[4]. http://robot.cmpe.boun.ed.tr/593/evrim.pdf

[5]. Murat Kalender, 2007, Ders Çizelgeleme Programı

[6]. http://fbe.baskent.edu.tr/dokuman/tezler/tez_SemanurKirici.pdf

[7]. http://www.aspnedir.com/Article/DisplayArticle.aspx?ID=655

[8]. http://tr.wikipedia.org/wiki/Microsoft_Visual_Studio

[9]. http://www.netogretim.com/dokumangoster.aspx?id=68

[10]. http://www.csharpnedir.com/articles/read/?id=818

[11]. http://tr.wikipedia.org/wiki/AJAX_%28programlama%29

[12]. http://tr.wikipedia.org/wiki/Genetik_algoritmalar

[13]. Demirsoy, A., 1998, Kalıtım ve Evrim, Meteksan Yayınları, No:14, 902s. Ankara.

[14]. Körez, M.T., 2005, Sıralı Akış Tipi Çizelgeleme Problemlerinde GA
Uygulaması, Yıldız Teknik Üniversitesi Fen Bilimleri Enstitüsü, Yüksek Lisans Tezi

[15]. Ufuk Okuyucu, Genetik Algoritmalar,
http://www.bilmuh.gyte.edu.tr/BIL523/presentations/ufuk/GenetikAlgoritmalarRapor.doc

[16]. Cengiz, Y., 2004, Optimum Performanslı Mikrodalga Kuvvetlendirici Tasarımı.
Yıldız Teknik Üniversitesi Fen Bilimleri Enstitüsü, Doktora Tezi, 224s, İstanbul.

[17]. Ceylan, H., ve Haldenbilen, S., 2005, Şehirler Arası Ulaşım Talebinin Genetik Algoritma ile Modellenmesi, İMO Teknik Dergi, Yazı 238, 3599-3618.

[18].Dengiz, B., ve Altıparmak, F., 1998, Genetik Algoritmalar, Gazi Üniversitesi Fen Bilimleri Enstitüsü Dergisi, 11: (3), 523-541.

[19]. Luger, G. F., 2002, Artificial Intelligence, Structures and Strategies for Complex Problem Solving, Fourth Edition, at page 471, Harlow, England: Addison-Wesley.

[20]. http://www.sayisalyontemler.com/?q=node/83

EKLER

EK A. 1: **Sube** class yapısı

```
class Subeler
    {
        List<Subeler> dizi = new List<Subeler>();
        Yerlesim[]. yer = new Yerlesim[100].;
        public Subeler this[int index].
        {
              get
            {
                if (dizi.Count <= index)
                {
                    Subeler y = new Subeler();
                    dizi.Insert(index, y);
                }
                return dizi[index].;
            }
            set { dizi[index]. = value; }
         }
        internal Yerlesim[]. Yer
        {
            get { return yer; }
            set { yer = value; }
        }
        private long sube_id;
        public long Sube_id
        {
            get { return sube_id; }
            set { sube_id = value; }
        }
        private string dersKod;
        public string DersKod
        {
            get { return dersKod; }
            set { dersKod = value; }
        }
        private string dersAd;
        public string DersAd
        {
            get { return dersAd; }
            set { dersAd = value; }
        }
        private int programDal_id;
        public int ProgramDal_id
        {
            get { return programDal_id; }
            set { programDal_id = value; }
        }
        private string programDal_ad;
        public string ProgramDal_ad
```

```
    {
        get { return programDal_ad; }
        set { programDal_ad = value; }
    }
    private int birim_id;
    public int Birim_id
    {
        get { return birim_id; }
        set { birim_id = value; }
    }
    private string birim_ad;
    public string Birim_ad
    {
        get { return birim_ad; }
        set { birim_ad = value; }
    }
    private int ogrSayisi;
    public int OgrSayisi
    {
        get { return ogrSayisi; }
        set { ogrSayisi = value; }
    }
    private long dersVerenNufus_id;
    public long DersVerenNufus_id
    {
        get { return dersVerenNufus_id; }
        set { dersVerenNufus_id = value; }
    }
    private ArrayList dersAlanOgr_idleri=new ArrayList();
    public ArrayList DersAlanOgr_idleri
    {
        get { return dersAlanOgr_idleri; }
        set { dersAlanOgr_idleri = value; }
    }
    private short derece;
    public short Derece
    {
        get { return derece; }
        set { derece = value; }
    }
    private short dilim;
    public short Dilim
    {
        get { return dilim; }
        set { dilim = value; }
    }
    private short? sinifDetay_id;
    public short? SinifDetay_id
    {
        get { return sinifDetay_id; }
        set { sinifDetay_id = value; }
    }
    private short sinavTur_id;
    public short SinavTur_id
    {
        get { return sinavTur_id; }
        set { sinavTur_id = value; }
    }
}
```

ÖZGEÇMİŞ

Ad Soyad: Ceyda BAYSAL

Doğum Yeri ve Tarihi: 08. 04. 1986 İzmir

Adres: İnönü Cad. 52/56 Sok. No:11/5 Esenyalı / İZMİR

Lisans Eğitimi: Pamukkale Üniversitesi Bilgisayar Mühendisliği Bölümü