YILDIZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

SİSTEMATİK KONSTRÜKSİYONDA GENETİK ALGORİTMALARIN KULLANILMASI

Mak. Müh. Coşkun FİLİZ

FBE Makine Mühendisliği anabilim Dalı Konstrüksiyon Programında Hazırlanan

YÜKSEK LİSANS TEZİ

Tez Danışmanı: Prof. Dr. Atilla BOZACI

Doç. Dr. Ferhat Dikmen

Y. Doç. Dr. Vedat Temiz

İSTANBUL,2005

İÇİNDEKİLER

SİMGE LİSTESİ

P_m	Mutasyon oranı
P_c	Çaprazlama oranı
G	Yığın aralığı (Jenerasyon boşluğu)
W	Ölçeklendirme fonksiyonu

KISALTMA LİSTESİ

2B	İki boyutlu
3B	Üç boyutlu
CAD	Computer Aided Design
CAE	Computer Aided Engineering
CAM	Computer Aided Manufacturing
CAO	Computer Aided Optimization
EA	Evrimsel Algoritmalar
GA	Genetik Algoritma
TA	Tabu Arama
TB	Tavlama Benzetimi

ŞEKİL LİSTESİ

ÇİZELGE LİSTESİ

ÖNSÖZ

Bu çalışmanın, bu konuları araştıran kişilere yardımcı olması dileğiyle.

Çalışmalarımda bana yol gösteren danışmanım Prof. Dr. Atilla BOZACI'ya, fikirleri ile yardımların esirgemeyen çalışma arkadaşlarıma, manevi desteği ile yanımda olan sevgili eşim Özge FİLİZ'e teşekkürü bir borç bilirim.

ÖZET

Endüstrideki gelişmeler ve rekabet ortamı, mühendisliğin birçok alanını barındıran çok daha karmaşık ürünler ortaya çıkarmak zorunluluğunu doğurmuştur. Aynı zamanda, ürünler pazara en kısa sürede çıkmalı, bunun için tasarım süreci verimli kullanılmalı. Bunun yanında kalite için imalatçılar sürekli baskı altında kalır. Bütün bunların yanında karmaşık bir ürünü en kısa sürede ve kaliteli bir şekilde piyasaya çıkartırken en rekabetçi fiyat ile sunmanız gerekmektedir. Dünyanın ve ülkemizin içinde bulunduğu durum göz önüne alınırsa piyasalardaki rekabetin her geçen gün artığı ve kar pay sürekli düştüğü görülebilir. Bu durumda rekabeti sürdürebilmek için maliyetleri en alt seviyede tutmak, kaliteye önem vermek ve ürünü kısa sürede piyasaya sürmek gerekmektedir.

Bu üç temel olguyu sağlayabilmek tasarım ve üretim süreçlerini dikkatlice planlamak ve kontrol altında tutmakla mümkün olacaktır. Tasarım ve üretim teknolojilerindeki gelişmeleri yakından takip etmek eskisinden daha önemli hale gelmiştir. Teknoloji olarak yerimizde saymak gerilemek ile aynı anlama gelmektedir.

Bu çalışmada yukarıda belirtilen olgulardan tasarım sürecinin planlanması ve ürünlerin tasarım sürecinde isteklere en uygun (optimum) olarak ortaya çıkması amacıyla izlenmesi gerek yollara değinilmiştir. Tasarım sürecinin planlanmasında bu sürenin en verimli şekilde kullanılması amaçlandığı gibi tasarım optimizasyonunda da en uygun tasarımın bulunması gerekmektedir. Çalışmanın içeriğinde tasarım olgusunun tanımlanması yapılmış, tasarım sürecinde kullanılabilecek metotlardan biri olan Sistematik Tasarım yöntemi işlenmiştir. Daha sonra, sırasıyla yapısal ve tasarım optimizasyonu hakkında genel bir bilgi verilmiştir. Son yıllarda optimizasyonda kullanılmaya başlanan sezgisel (stochastic) yöntemler hakkında genel bilgi verildikten sonra bunlardan genetik algoritma daha ayrıntılı olarak incelenmiştir. Ve son olarak ta MATLAB programının genetik algoritma aracını kullanılarak, genetik algoritmanın mekanik tasarım optimizasyonunda kullanılması ile ilgili örnek çözümler yapılmıştır.

Anahtar kelimeler: Sistematik tasarım, yapısal optimizasyon, tasarım optimizasyonu, genetik algoritma.

ABSTRACT

Changes in the Industrial area and the competition enviroment has caused to many differentiation in the products. The design process must be used more efficently to fullfill the market demands. To sustain competitive advantages and the low production costs in the market design engineers are under the great pressure to improve their product and to cut the cost in order to come up with a profitable products. Design engineer's first job is to improve the quality of the product while gaining market share and increase profit margins with their products.

To improve these three (Market Share, Contribution Margin, Improving Quality Of the Product) criteria is only possible with the real planning process during the Design and the Production .The advances in the design & production process play a key role in the industry therefore design engineer must be aware of the those changes in order to develop products.

In this paper we mentioned the Planning of the Design Process in order to find the optimum solutions. In this process we have to plan not only the Design Process efficiently, but also we have to consider design optimitization as well. This paper included the definition of the Design Process and one of the method to follow ' Systematic Design Method'.Later on in this paper you can find the general studies about Structural Optimization and Design Optimization.This paper mentioned one of the new method in the optimization 'stochastic method'.Main focus of the this study is the Genetic Algorithm and revised thorougly in this paper.The MATLAB software has been used to prepare the examples of the Genetic Algorithm problem applied to Mechanical Design Optimization solutions.

Keywords: Systematic design, structural optimization, design optimization.

1. GİRİŞ

Endüstrideki gelişmeler ve rekabet ortamı, mühendisliğin birçok alanını barındıran çok daha karmaşık ürünler ortaya çıkarmak zorunluluğunu doğurmuştur. Aynı zamanda, ürünler pazara en kısa sürede çıkmalı, bunun için tasarım süreci verimli kullanılmalı. Bunun yanında kalite için imalatçılar sürekli baskı altında kalır. Bütün bunların yanında karmaşık bir ürünü en kısa sürede ve kaliteli bir şekilde piyasaya çıkartırken en rekabetçi fiyat ile sunmanız gerekmektedir. Dünyanın ve ülkemizin içinde bulunduğu durum göz önüne alınırsa piyasalardaki rekabetin her geçen gün artığı ve kar pay sürekli düştüğü görülebilir. Bu durumda rekabeti sürdürebilmek için maliyetleri en alt seviyede tutmak, kaliteye önem vermek ve ürünü kısa sürede piyasaya sürmek gerekmektedir. Bu üç temel olguyu sağlayabilmek tasarım ve üretim süreçlerini dikkatlice planlamak ve kontrol altında tutmakla mümkün olacaktır.

Bu sebeple ilk olarak tanımlaması yapılmıştır. Tasarım düşüncesinin ve sistemli bir mühendislik tasarımı yapmanın bize kazandıracakları yöntemler hakkında bilgi verilmiştir.

Her tasarımcı bu süreçte birçok problem ile karşılaşır ve bunlara çözüm bulması gerekir. İşte bu noktada en iyi olan çözümü en kısa sürede bulmak önemlidir. En iyi kavramı mühendis için en güzel olan, en pahalı/ucuz olan veya sadece en hafif ola değildir, bunların birini genelde birden fazlasını sağlaması gerekmektedir. Burada en iyi kavramı optimum olan anlamına dönüşmektedir.

Mühendislik problemlerinde optimum tasarımı bulmak çoğunlukla karmaşık, zahmetli ve pahalı bir süreç olmaktadır. Bu sebeple her zaman optimum çözüme hızlı, ucuz ve kolay ulaşmak için çalışmalar hiç durmadan devam etmektedir. Klasik yöntemlerin yetersiz kaldığı durumlarda sezgisel yöntemler ortaya çıkmıştır. Bunların arasında en çok bilinenleri olan evrimsel algoritmalar, tabu arama ve tavlama benzetimi sayılabilir. Evrimsel algoritmalardan olan genetik algoritma optimizasyonda sıklıkla kullanılmaktadır. Literatürde birçok örneği bulunan bu çalışmaların bir kaçı aşağıda verilmiştir.

- Çok amaçlı montaj planlama problemi (Chen, 2002)
- Kombine topoloji ve şekil optimizasyonu (Mancuso, 2002)
- Uçak motoru montaj optimizasyonu (Iuspa, 2003)
- Topoloji optimizasyonu (Kane, 1996)
- Modüler ürün tasarımı (Kreng, 2004)

2. MÜHENDİSLİK TASARIMI

2.1 Temel Tanımlar

2.1.1 Tasarım

Tasarım insan ihtiyacına cevap verebilmek için bir proje geliştirmektir. Buna yönelik başlangıçta ihtiyaçların iyi tanımlanması gerekir. İyi tanımlanmış isteklere örnek vermek gerekirse;

- Çevreye zarar vermeden yeterli miktarda temiz, güvenli ve ekonomik bir enerji kaynağı nasıl elde ederiz.
- Okulumuzun internet alt yapısı yeterli değildir. Yatırım maliyeti düşük, kısa sürede kurulabilecek ve mevcut eksikleri ile gelecekteki isteklere cevap verebilecek alt yapı kurulmalıdır.

Diğer taraftan problemin özellikleri iyi tanımlanmamış olabilir. Bu durumlarda problemin çözümü zorlaşır bazen de imkansızlaşabilir, bu sebepten dolayı problemin iyi tanımlanması çözüm için önemlidir.

- Birçok insan trafik kazarlında hayatın kaybediyor.
- Büyük şehirlerdeki trafik çok yoğun.

Bu ikin tip problemler ihtiyacı ve durumu iyi tanımlamamaktadır.

Tasarımlar çok çeşitli sınıflara ayrılabilir. Bunlara birkaç örnek vermek gerekirse,

Makine tasarımı	Elbise tasarımı
Gemi tasarımı	Otomobil tasarımı
Isıtma sistemi tasarımı	Bina tasarımı
Köprü tasarımı	Bilgisayar tasarımı

Bu şekilde sonsuz çeşit tasarım sınıflandırması yapılabilir.

Matematiksel veya fizik problemleri gibi tasarım probleminin genelde bir tek çözümü yoktur. Tasarım problemi için bir tek cevap olmadığı gibi bu gün için iyi olan bir çözüm gelecekte yeterli olmaya bilir.bu sebepten dolayı firmalar tasarımlarını günün istekleri doğrultusunda sürekli geliştirmek zorundadırlar. Günlük hayatta dahi herkes bir tasarım ve planlama problemi ile karşılaşmaktadır. Bir ailenin tatilini planlaması, günlük işlerin planlaması gibi bir çok örnek verilebilir.

Mühendislik tasarım problemleri bir varsayım problemi değildir. Tasarımın gerçek bir amacı vardır ve bir ihtiyaca yönelik yapılır. Burada amaç bir fonksiyona veya fiziksel bir gerçekliğe sahip bir şeyin ortaya çıkarılmasıdır.

2.1.2 Konstrüksiyon

Toplumun herhangi bir gereksinimine, bilinen temel ve mühendislik bilimlerinden yararlanarak, günün koşullarına en uygun çözümün sistematik olarak aranması, bulunması, gerçekleştirilmesi ve böylece toplumun söz konusu gereksiniminin karşılanması işlemidir. Söz konusu toplumsal ihtiyacın çözümünün plan veya proje şeklinde formüle edilmesi tasarım işlemidir. Konstrüksiyon, tasarım aşamasından sonra, ürünün maddesel gerçekleştirilmesi aşamasını da içermektedir. Bu çalışmaların tamamı belli bir yönteme, usul, tarz, düzen veya sisteme uygun olarak (sistematik, metodik) yapılmalıdır.

Yol, köprü, gemi, bina, ısıtma sistemi, torna tezgahı, elbise, kaldırma makinaları v.s. toplumun belli gereksinimlerine çözüm getirmek üzere gerçekleştirilmiş konstrüksiyon örnekleridir. (Bozacı, 2002)

2.1.3 Teknik Ödev

Yukarıdaki örneklerde olduğu gibi yanıtlanması gereken toplum gereksinimlerine (konstrüksiyon problemlerine) teknik ödev adını vereceğiz. Teknik ödevlerden bazıları, iyi tanımlanabilen, ne olduğu veya ne olmadığı açıkça ortaya konabilen türden teknik ödevlerdir. Örneğin; bir dişli çark, bir gemi, bir elektrik motoru gibi. Ancak bazı durumlarda konstrüktörün ödevinin ne olduğu açıkça belli olmayabilir. Örneğin trafik kazalarının önlenmesinin bir konstrüksiyon ödevi olarak verildiğini düşünelim. Ödevin ne olduğu açıkça belli değildir. Kasalara yol koşullarının kötülüğünü, araçların konstrüktif açıdan yetersiz oluşlarını yoksa insanların yeterli düzeyde eğitilmiş olmamalarımı yol açmaktadır? Bu örnekte ödevin ne olduğunu ortaya koymak için ciddi ön araştırmalar gerekmektedir.

Bir uçak yapımı bir dişli çark yapımından çok daha karmaşık bir problemdir. Karmaşık problemler bir takım alt problemlere bölünerek ayrı ayrı ele alınabilir. Bazı ödevlerin gerçekleştirilmesi için ekip çalışmalarına ve bunun bilimsel organizasyonuna gerek gösterirler. (Bozacı, 2002)

2.1.4 Teknik Tasarım (Konstrüksiyon) Türleri

2.1.4.1 Yeni Konstrüksiyon

Bir konstrüksiyon problemine şimdiye kadar bilinen çözümlerin dışında yeni bir çözüm getirme veya yeni bir teknik ödev tanımlayıp bu ödeve çözüm arama işlemidir, Güneş enerjisiyle çalışan bir otomobilin veya hem havada nemde su altında yol alabilen bir taşıt aracının konstrüksiyonu gibi.

2.1.4.2 Uygu Konstrüksiyonları

Bilinen bir çözümün yeni isteklere ve yeni koşullara uyum sağlayacak şekilde değiştirilmesidir. Uygu konstrüksiyonların da bilinen çözümün bazı guruplarında yeni konstrüksiyonlar gerekli olabilir. Örneğin ekvatorda çalışan bir otomobilin kutuplarda çalışabilecek bir şekle sokulması işlemi gibi. Burada aracın motorunun ekvatorda ve kutupta çalışabilecek şekilde kısmen yeniden tasarlanması gerekebilir.

2.1.4.3 Çeşitleme (Varyant) Konstrüksiyonu

Bilinen bir konstrüksiyon örnek alınarak boyutlarda, düzenlemelerde değişiklikler yaparak aynı çözümün benzerlerinin yaratılmasıdır. Fonksiyon (işlev) ve çözüm prensibi değişmediği için malzeme, teknoloji ve mukavemet açısından yeni problemler çıkmıyor. 1973 verilerine göre D.Almanya'da gerçekleştirilen konstrüksiyonların %55'i uygu, 525'i yeni %20'si çeşitleme konstrüksiyonu olmuştur. (Bozacı, 2002)

2.2 Makine Tasarımında Klasik (Geleneksel) Çalışma Düzeni

Geleneksel çalışma düzeni de sistematik konstrüksiyon tekniğinde izlenen yol özde bir birine benzemektedir. Her ikisi de problemi tanıma, anlama ve bilgilenme adımı ile başlamaktadır.

1- Problemi belirleme, tanımlama

2- Bilgilenme

a) Benzer çözümleri inceleme

b) Teorik araştırmalar

3- Kabaca soyutlandırma (ön şekillendirme)

a) Kaba mukavemet hesapları

b) Kroki şeklinde çizimler

4- Ayrıntılı boyutlandırma (son şekillendirme)

a) Malzeme seçimi

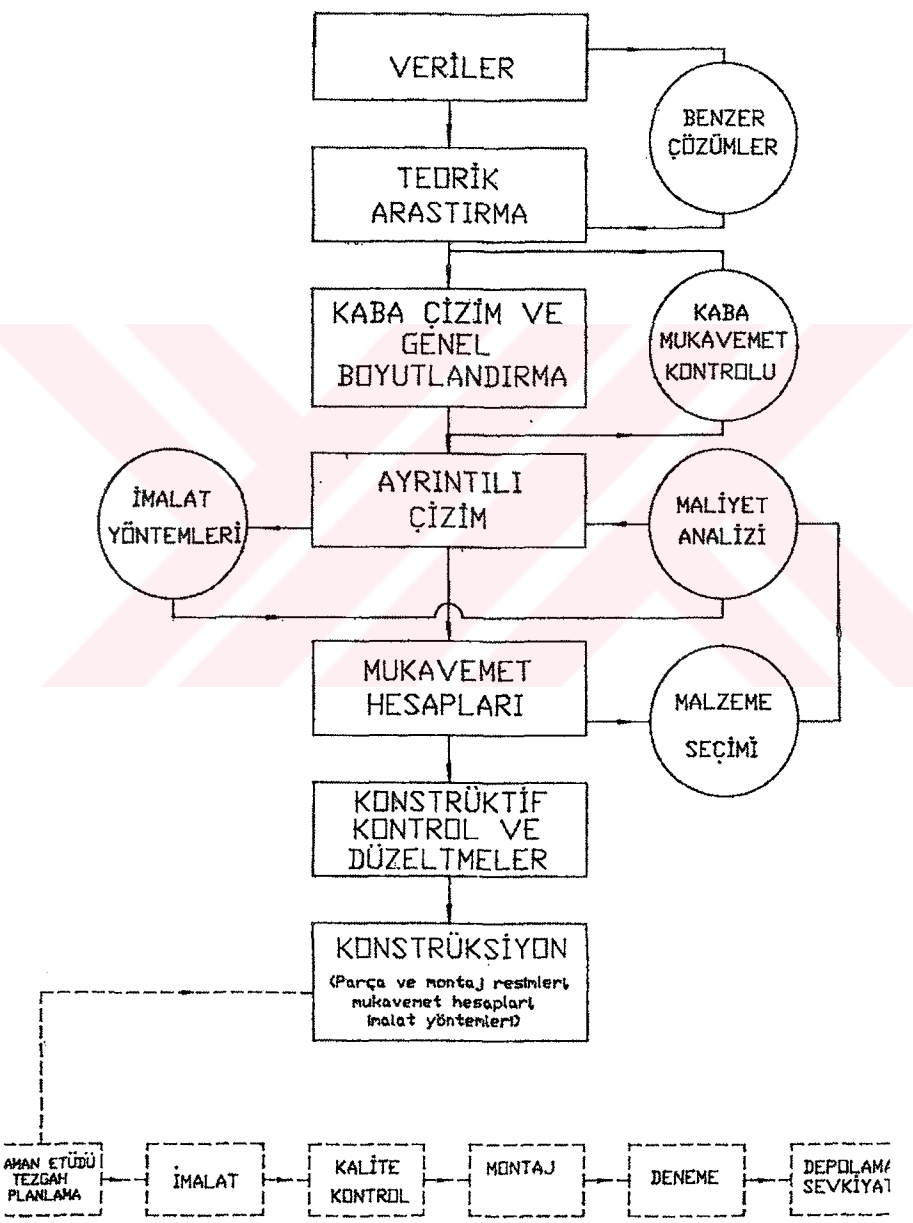
b) İmalat yöntemlerinin seçimi

c) Maliyet analizi

d) Hassas mukavemet hesapları

5- Konstrüktif kontrol ve düzeltmeler

6- İmalata yönelik parça ve montaj resimleri(Konstrüksiyon)



Şekil 2.1 Makine konstrüksiyonunda klasik çalışma düzeni (Prof. Bengisu) (Bozacı, 2002)*

2.3 Sistematik (Metodik) Konstrüksiyon (Tasarım)

2.3.1 Sistematik Konstrüksiyon Bilimi

Genel olarak bir konstrüktör temel bilimlerden, mühendislik bilimlerinden, yararlanarak konstrüksiyon ödevini plan ve proje şeklinde gerçekleştirir. Bu proje mevcut teknolojiden yararlanılarak ürün şekline dönüştürülür ve topluma sunulur.

Bir bakıma tasarım karar verme çalışmasıdır. Karar verebilmek için geniş bir bilgi, zengin bir deneyim, yaratıcı düşünce ve tasarım gücü gerekir. Yaratma bilgi sınırının üstüne başarılı bir adım atabilme şeklinde tanımlanabilir. Bilinenlere önceden bilinmeyen bir şeyi tanımlayıp ilave etmektir. Buna bulgu da diyebiliriz.

Mesleki bilgisi dışında düzenli düşünmeye, metotlu iş görmeye alışmamış mühendis bu günün çok yüksek ve karmaşık teknik problemlerine teknik ve ekonomik açıdan en iyi çözümleri bulmakta zorluk çekebilir. Sistematik konstrüksiyon mühendise düzenli düşünme ve metotlu iş görmeyi öğretir.

2.3.2 Sistematik Konstrüksiyonda Adımlar

Teknik yapıtın cinsinden bağımsız olarak teknik ödevin belirlenmesinden sistemin maddesel gerçekleşme aşamasına kadar geçen sürede yapılan işlem basamakları bütün mühendislik dallarındaki yaratıcı davranışların hepsinde var olabilecek gelişme adımlarının ara kademeleridir.

Konstrüksiyonlar dış görünüşleri ve fonksiyonları bakımından. Farklı olsalar da bunların şekillendirilmesinde uygulanacak düşünce sistemi benzerdir

Bir teknik yapıtın veya bir yöntemin geliştirilmesinde teknik ödev ve onun uygun çözümü arasında vazgeçilmez bazı aşamalardan geçilmesi gerekir.

Genel Olarak:

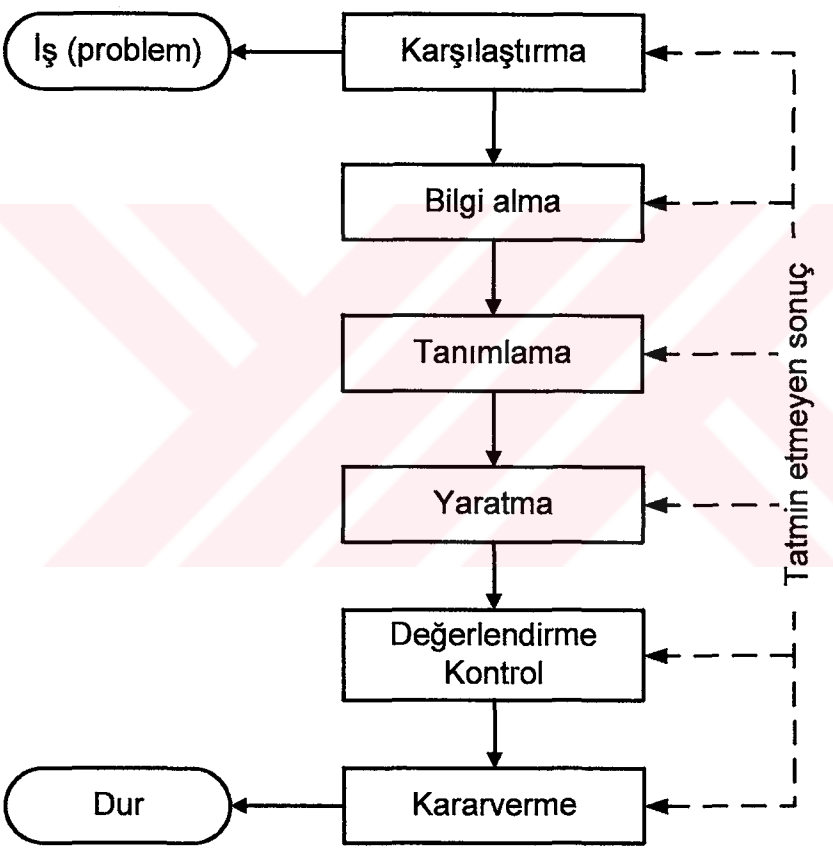
- Ödevde bütün çözümlerin esası mevcuttur.
- Her çözüm fiziksel etkileri bilinen elemanların birleşmesinden meydana gelir.
- Her çözümde noksanlar vardır. Noksanların minimuma indirilmesi mümkündür.
- Noksanları azaltılmış çözümler arasında en az noksanı olan çözüm en uygun çözümdür.

Konstrüksiyonda ödevin belirlenmesinden maddesel gerçekleştirmeye kadar işlem basamakları ve gelişme adımları konusunda değişik görüşler ortaya atılmıştır. Geleneksel çalışma düzeni ile metoda bağlı konstrüksiyon tekniğinde izlenen yol temelde bir birine

benzemektedir. Mühendis bir tasarım problemine sistematik bir yaklaşımla çözüm ararken fazla dağılmadan amacına hızla ulaşabilmesi için genelde aşağıdaki adımları izler.

2.3.2.1 Genel Problem Çözümü

En genel anlamda problem çözme yöntemi adım adım analiz ve sentez çalışmalarını içerir. Niteselden nicesele geliştirilir. Her yeni adım bir öncekinden daha somuta doğru atılır. Tasarım bilgilerin dönüştürülmesi olarak da görülebilir. Her yeni adım bir öncekinin sonuçlarını iyileştirir. Gereken gelişmeye ulaşıncaya kadar bu iterasyon işlemi tekrarlanır durur. Bilgilerin işlenmesi bir sonraki adım için veri üretirken bir önceki adımı da daha iyi aydınlatmış olur.



Şekil 2.2 Genel problem çözmede izlenen yol (Bozacı, 2002)

Önce problem hazırda mevcut bilgilerin ışığında karşılaştırmalı olarak değerlendirilir. Bilinenlerle tartılır. Tasarımcının bilgi deneyim ve ilgi alanına bağlı olarak bu aşamanın süreç içindeki önemi değişir.

İkinci olarak problem hakkında kısıtlar, muhtemel çözüm prensipleri ve benzer problemlerin bilinen çözümleri hakkında bilgi edinilir. Bu çok önemli ve gereklidir.

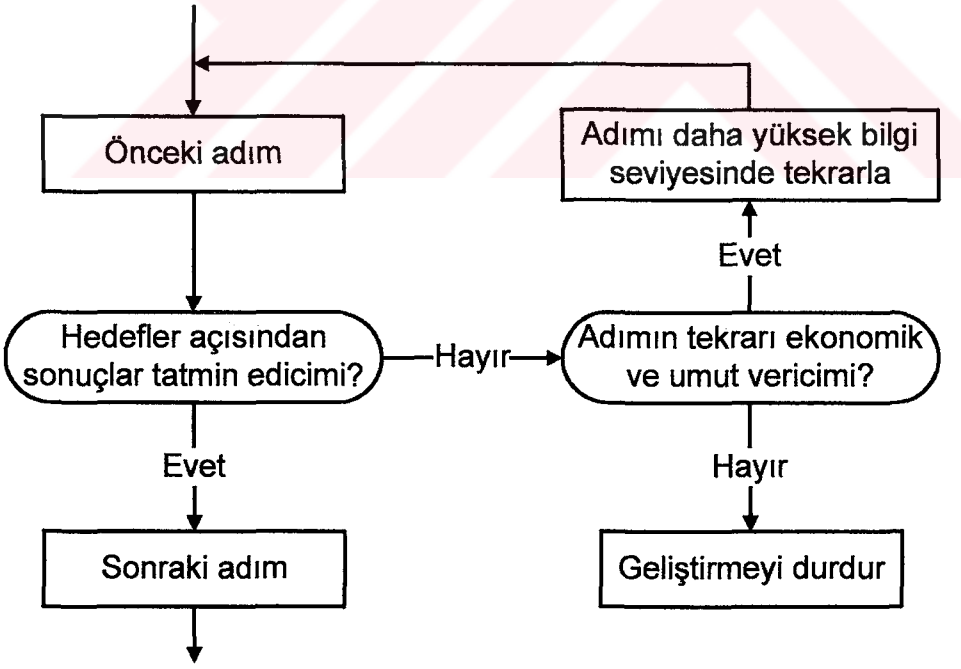
Daha sonra problemin soyut planda tanımının yapılması gelir. Hedefler ve belli başlı kısıtlamalar belirlenir. Bu tanımlar çözüme giden yolları açar.

Sonraki adım çözüme ait seçeneklerin yaratılmasıdır. Çözümler çeşitli olanaklarla geliştirilirken değişik seçenekler sistematik olarak bir araya getirilir.

Uygun gözüken seçeneklerin çokluğu durumunda en iyi seçeneğin belirlenmesi için bir değerlendirme yapılır. Zaten her adımda karar vermeden önce bir değerlendirme yapmak gerekir. Değerlendirme çalışması aynı zamanda tasarım süreci boyunca kontrol hizmetide vermiş olur.

- Eğer bir önceki adımın sonuçları amaca uygunsa bir sonraki adıma geçilir.
- Eğer sonuçlar uygun düşmezse bir sonraki adım atılmaz.
- Eğer bir önceki adımın (veya adımların) tekrarı ekonomik ve iyi sonuçlar alınacağı tahmin ediliyorsa "daha yüksek bilgi seviyesinde" adım tekrarlanmalıdır.
- Aksi halde geliştirme durdurulmalıdır.

Bu çalışma problemin (teknik ödevin) kısmen veya tamamen geliştirilmesine (eğer sonuçlar amaca uygun değilse) yardımcı olur.



Şekil 2.3 Genel karar verme süreci (Bozacı, 2002)

2.3.2.2 Tasarım Sürecinde İş Akışı

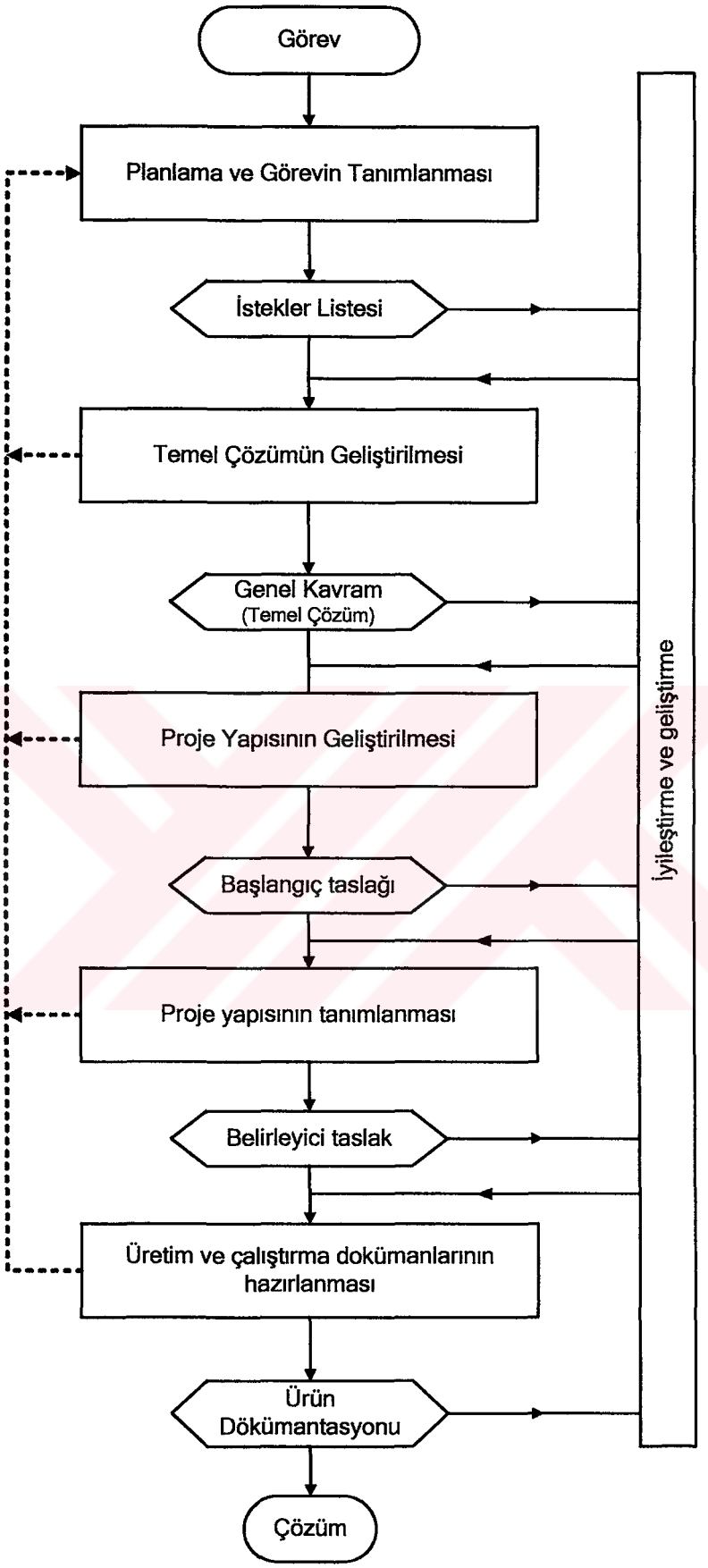
Pahl ve Beizt'e göre tasarım sürecinin ana aşamaları.

1. Yapılacak işin (teknik ödevin) iyi aydınlatılması, açıklanması: Yapılacak işin aydınlatılması aşaması iş hakkında bilgi toplamayı, karşılanması gereken istek ve kısıtların ortaya konmasına dönük çalışmaları kapsar. İşin ne olup ne olmadığını bilmeye yöneliktir. Bu çalışma aşaması bir "istekler listesi" ile özetlenir.

2. Düşünsel tasarım süreci: Bu aşamada uygun çözüm adayları aranır. Çözüm prensipleri zihinsel olarak bir araya getirilerek çözüm seçenekleri oluşturulur. Çalışmalar fikir düzeyindedir. İstekleri karşılamayan çözüm seçenekleri elenir. Kalanlar teknik ve ekonomik açıdan değerlendirmeye tabi tutulur. En iyi çözüm fikri seçilmiş olur.

3. Nesnel (cisimleştirmeye yönelik) tasarım: Tasarımcı bu aşamada seçilmiş düşünceden hareketle belirleyici şemalar çizilir. Teknik ve ekonomik koşulları da gözeterek bir teknik ürün veya yöntem geliştirmeye çalışır. Seçenekler hakkında daha somut bilgiler edinmek, karşılaştırmalar yapabilmek ve değerlendirebilmek için bu çizimlerin defalarca düzeltilerek tekrarlanması gerekebilir. Bu aşamanın girdisi bir tasarım fikri iken çıktısı biçimlendirmeye yönelik bir çizimdir. Bu çizim bir genel düzenleme bir şema veya bir şekillendirme resmi olabilir. Burada cisimleştirme soyut bir düşünceden somut bir öneri geliştirme anlamında kullanılmıştır.

4. Ayrıntılı tasarım çalışmaları: Detaylı tasarım aşaması ise imalat resimlerinin hazırlanması çalışmalarıdır. Şekil, boyut, yüzey kalitesi belirlenmiş kontrol edil miştir. Malzeme seçilmiştir. Bu konuda çok çeşitli ve iyi hazırlanmış kaynaklar bol miktarda mevcut olup genellikle bilinen çalışmaları kapsadığından burada ele alınmayacaktır. Dizayn sürecinin ana aşamalarını kesin sınırlarla ayırmak mümkün değildir. Her aşamada model ve prototiplerden yararlanılabilir.



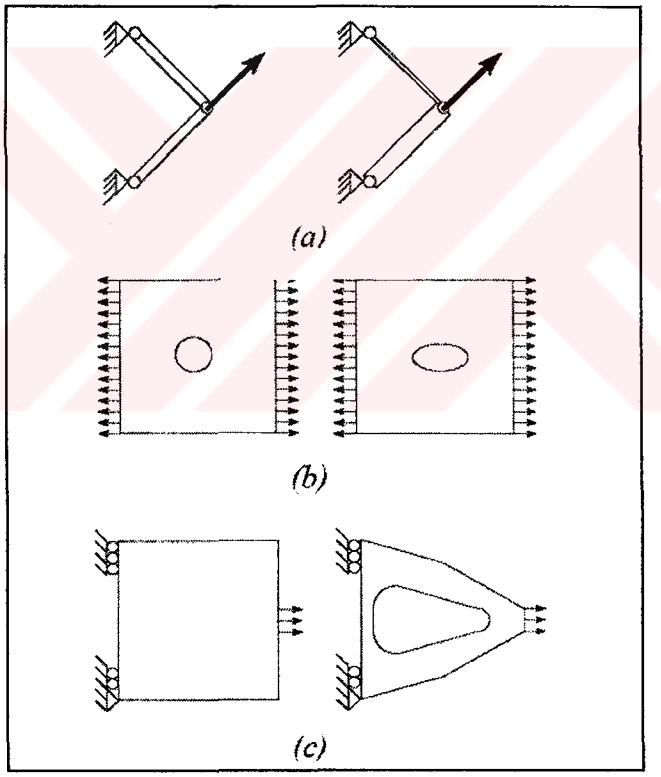
Şekil 2.4 Planlama ve tasarın sürecinin adımları (Pahl, 1996)

3. YAPISAL OPTİMİZASYON

3.1 Yapısal Optimizasyona Giriş

Makine parçalarının tasarımları 1960'lardan bu yana nümerik yöntemler ile oldukça geliştirildi. Örneğin sonlu elemanlar programları şuanda uçak, makine, gemi ve inşaat mühendisliği alanlarında sıklıkla kullanılıyor. Aynı zamanda, çeşitli matematiksel programlama problemleri için verimli ve hızlı algoritmalar ortaya çıkmıştır. Yapısal optimizasyonu yapısal optimizasyonu doğuran her iki akım, tatmin edici şekilde istekleri (kısıtlar) karşılamak, en azından bir kritere (amaca) uyan optimum çözümü, geometriyi, malzemeyi ve/veya topoloji parametreleri (değişkenleri) değiştirerek yaratmayı amaçlamaktadır (Pardalos, 2000).

Yapısal optimizasyon değişkenlerin karmaşıklığına göre üç guruba ayrılabilir:



Şekil 3.1 Tasarım (a), şekil (b), ve topoloji (c) optimizasyonu (Duysinx, 1996)

- Tasarım veya boyut optimizasyonunda, değişkenler sadece kesit alanını veya kalınlıkları (geometri ve topoloji sabit kalarak);
- Şekil optimizasyonunda, değişkenler doğrudan yapının geometrisini değiştiren parametrelerdir (topoloji sabit kalarak);

- Son olarak ta topolojik* optimizasyonda ise değişkenler yapın geometrisini ve topolojisini değiştirmek için kullanılır.

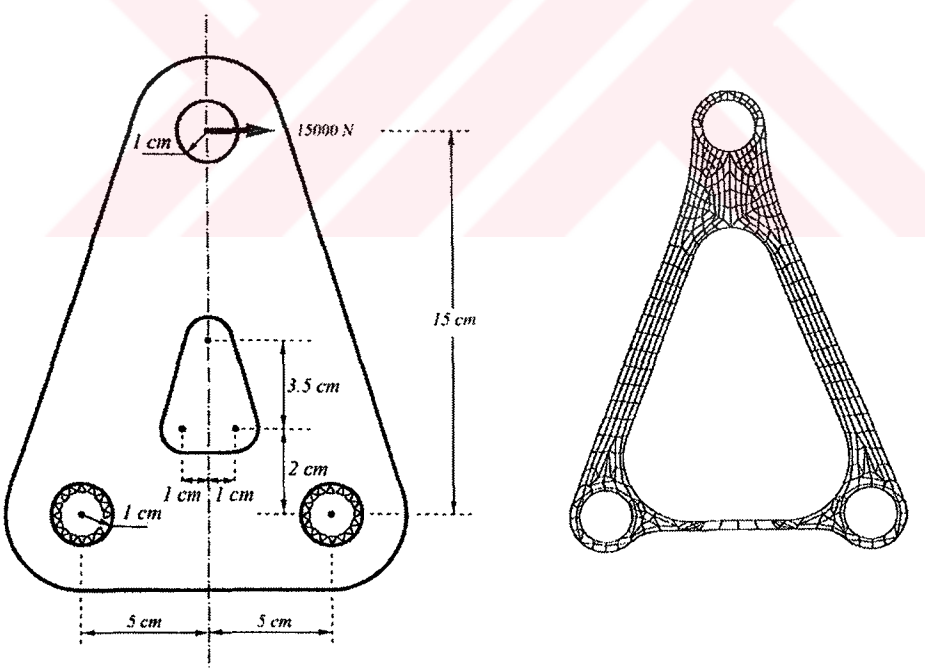
Bu kategoriler aşağıda kısaca değinilmiştir.

3.1.1 Tasarım (Boyut) Optimizasyonu

Aynı zamanda "Yapıların otomatik boyutlandırılması" (Duysinx, 1996) olarak ta bilinen bu ilk yaklaşımda değişkenler sadece kesit alanları ve kalınlıklardır (geometri ve topoloji sabit kalarak). Örneğin kafes yapılarda, kısıtlamaları yerine getirebilen (örn. gerilme ve yer değiştirmenin maksimum değeri aşmaması), en hafif yapı amaçlandığında çubukların kesit alanları tasarım değişkeni olarak rol oynar.

3.1.2 Şekil Optimizasyonu

Şekil optimizasyonunda, değişkenler yapının şeklini tanımlayan geometrik parametrelerdir. Literatürde var olan birçok çalışmada, parametreler özel noktaların (örn. iki vektörün kesiştiği nokta) koordinatlarıdır. İki boyutta, bu noktalar yapının konturlarını eğriler ile tanımlarlar. Örneğin; Lagrange, Bezier veya B-spline.



Şekil 3.2 Bir desteğin şekil optimizasyonu: başlangıç geometrisinin tanımlanması (sol) ve beş iterasyondan sonra bulunan çözüm (sağ) (Zhang, 1992)

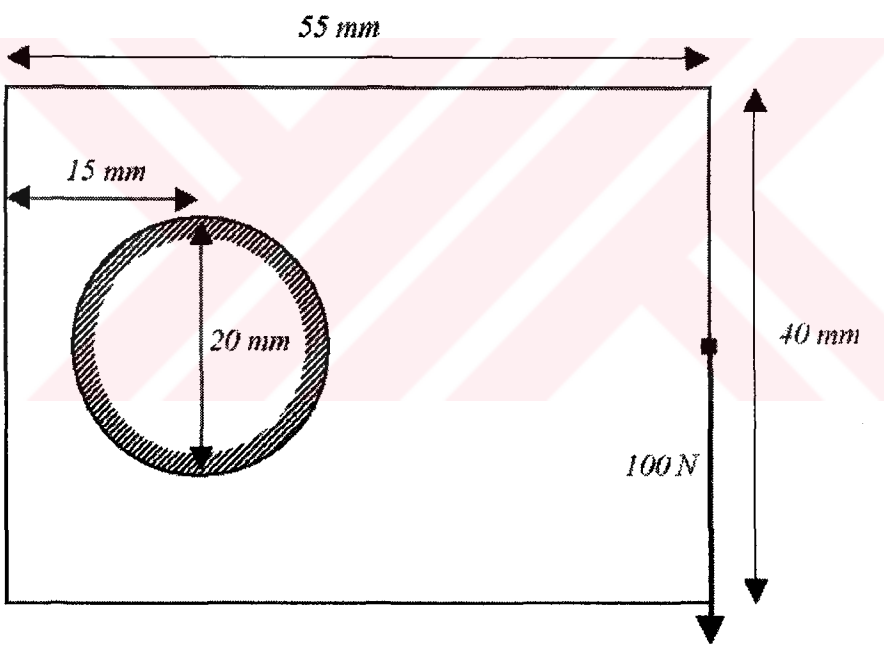
---

* Topoloji: Geometrik cisimlerin nitelikleriyle ilgili özelliklerini ve bağıl konumlarını, biçim ve büyüklüklerinden ayrı olarak alıp inceleyen geometri dalı. [Türk Dil Kurumu]

Geometri, tasarım değişkenlerini göz önünde bulundurarak, doğrudan uzunluklar, çaplar, açılar v.s. aracılığı ile de modellenebilir. Bu teknik Zhang tarafından destek parçası çalışmasında gösterilmiştir (Zhang, 1992). Yükleme hali ve sınır koşulları detayları Şekil 3.2 (sol)'de gösterilmiştir. 10 bağımsız değişken kullanılmış, destek geometrisi uzunluklar ve daire yayları ile oluşturulmuştur. Yapısal analiz sonlu elemanlar modelliyle gerçekleştirilmiş, ve optimum çözüm Zhang tarafından Şekil 3.2 'da gösterilmiştir. İlk modele göre kütle %69,9 azaltılmıştır.

3.1.3 Topoloji Optimizasyonu

Topoloji optimizasyonunda, amaç daha az yüklemeye maruz kalacak şekilde başlangıçtaki dolu hacimden adım adım malzeme çıkartarak optimum şekli hesaplamaktır. Tabiî ki sonuçta ortaya çıkan yapı daha önceden belirlenen kısıtları (örn. maksimum Von Mises gerilmesi) sağlamalıdır.

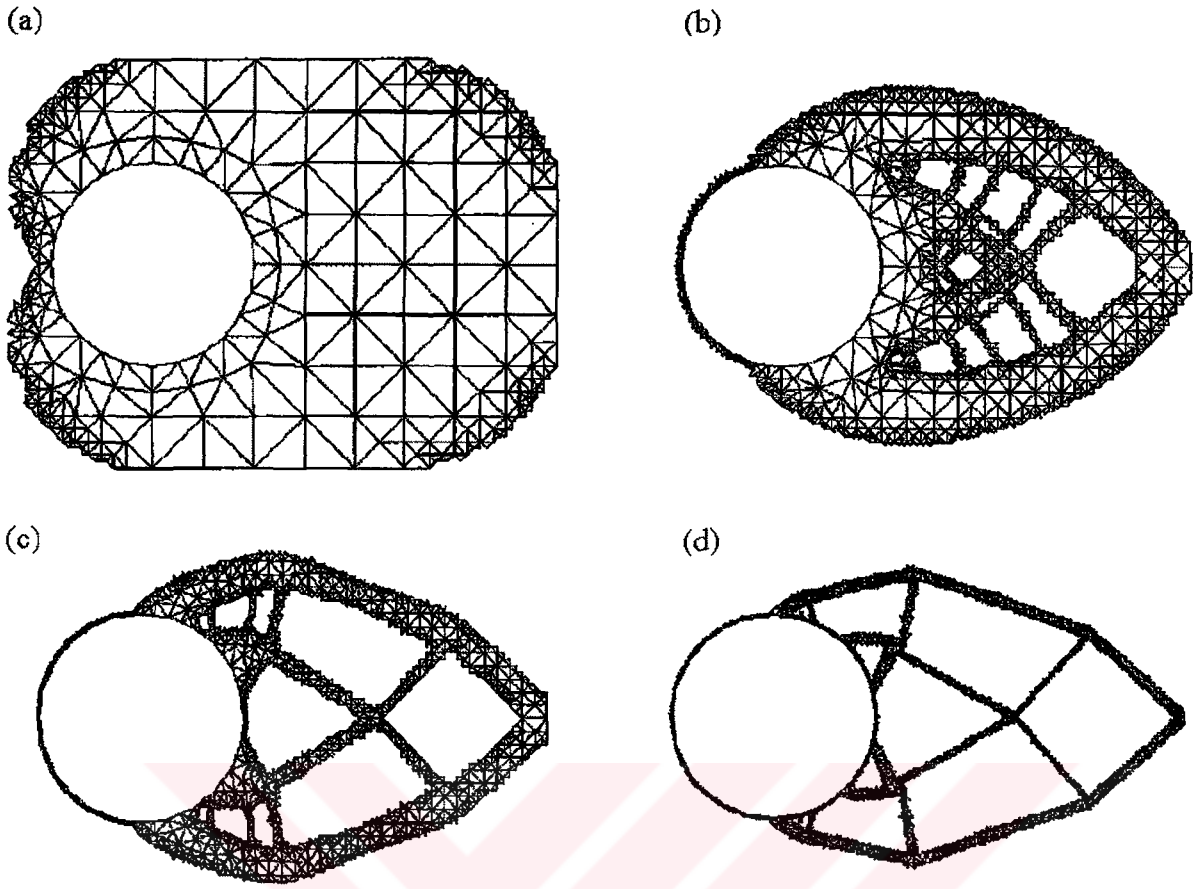


Şekil 3.3 Michell kiriş probleminin tanımlanması (Reynolds, 1999)

Hedef: En düşük ağırlık.

Sınır şartları: içteki dairesel delik sabit.

Kısıt: Von Mises gerilmesi.



Şekil 3.4 Kiriş problemine topoloji optimizasyonu uygulanması (Reynolds, 1999)

Topoloji optimizasyonunun değerlendirmede klasik olarak Şekil 3.3'te gösterilen Michell kiriş problemidir. Reynold ve diğerleri problemi tersine adaptasyon* tekniği ile çözmüştür. Bu teknik şu şekilde çalışmaktadır: bir defa başlangıç geometrisinin sonlu eleman modeli oluşturulur, sonra metot düşük gerilim bölgelerindeki eleman altbölümlerini iyileştirerek ilerler. Daha sonra, düşük gerilim bölgelerindeki elemanlar çıkartılır ve süreç tekrarlanır. Yapının 6, 42, 75 ve 120 iterasyondan sonraki halleri Şekil 3.4'de gösterilmiştir. 120. iterasyonda başlangıçtaki tüm alanın % 8,8'i kalmıştır.

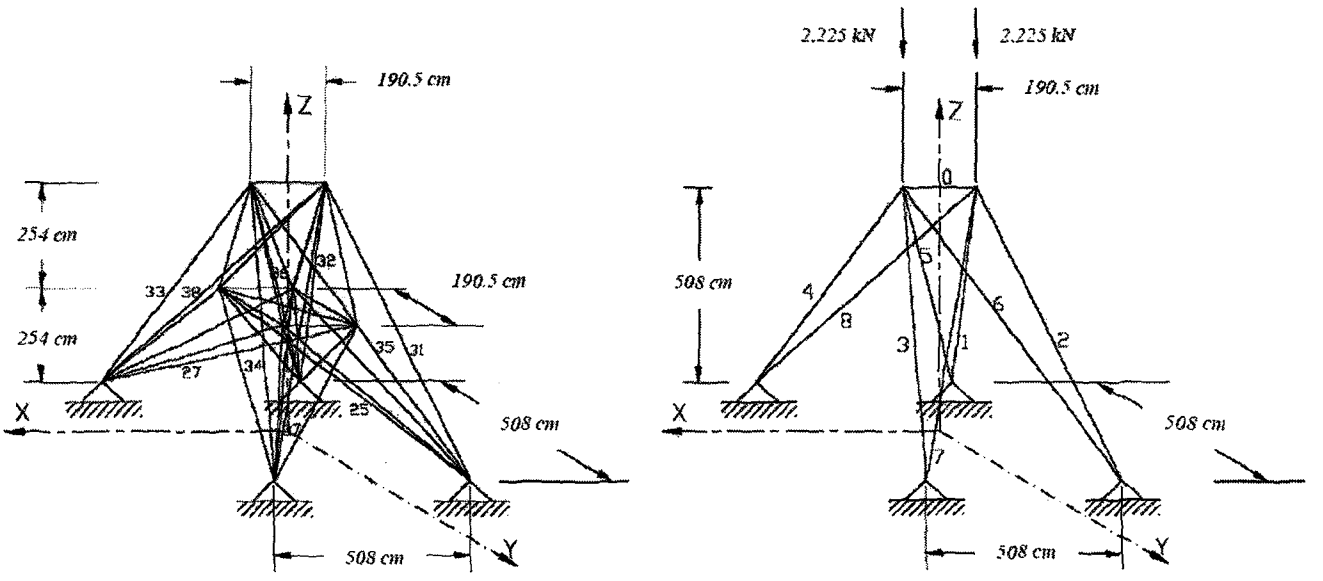
Topoloji optimizasyonu aynı zamanda kafes sistemlerinde de başarılı olmuştur. Örneğin, Deb ve Gulati 2B** ve 3B kafes sistemlerinde optimum kesit alanını ve topolojisini genetik algoritma kullanarak bulan bir metot tanımlamışlardır (Deb, 2001). Burada amaç gerilme ve yer değiştirmelerin izin verilen değerlerde kalarak kütlenin en aza indirilmesidir. Şekil 3.5'de

---

* Reverse Adaptivity.

** İki boyutlu (2D).

gösterilen 3B örnek genetik algoritma ile 9 elemanlı hale getirilmiştir (başlangıçta 39 eleman vardı).



Şekil 3.5 Kafes sisteminin GA ile topoloji optimizasyonuna örnek (Deb, 2001)

Şekil ve topoloji optimizasyonundaki en son gelişmeler sonlu elemanlar yöntemi ile sınır elemanlar yönteminin birleştirilmesidir (Macerle, 2003).

3.1.4 Diğer Yaklaşımlar

Yapısal optimizasyon, hesaplanabilir mekaniğin diğer alanları ile de ilişkilendirilebilir (örn. hata hesaplaması). Gerçektende her bir iterasyonda yapının geometrisi optimisazyon algoritması tarafında değiştirilip yeniden hesaplanmaktadır. Buda modelin güvenirliliği sorusunu ortaya çıkarır (örn. başlangıçtaki sonlu elemanlar modelinin geometri değişimler ile düzenin bozulmasında). Bazı uygulamalarda bu durum optimizasyon sırasında ele alınır. Örneğin Lacroix ve Bouillard, optimizasyon sürecinde hassasiyeti artırmak için sonlu elemanlar ve elemansız Galarkin yöntemin kombine etmiştir (Lacroix, 2003).

Yapısal optimizasyonda model doğrulamaya nazaran diğer önemli bir durumda hesaplama zamanıdır. Nümerik yöntemler çok zaman aldığında tahmin metotları zaman açısından daha karlı olmaktadır (Vandle, 2001).

3.1.5 Ön Tasarım Optimizasyonu

Optimizasyon geleneksel olarak bir bakıma katı üç alt bölüme (tasarım, şekil ve topoloji) ayrıla bilir. Gerçektende tasarımın ilk evrelerinde, başlangıçtaki taslağımızın

optimizasyonunda, sadece kesit alanları ve et kalınlıkları gibi değişkenler gerekmemektedir, aynı zamanda geometrik ve/veya topolojik parametrelere de ihtiyacımız vardır (parçanın üzerindeki deliklerin sayısı, hafifletme için yapacağımız boşaltmalar gibi).

Bu bağlamda ,tasarım optimizasyonu parametrik yapıların optimum ölçülerini bulmakla eş anlamlı hale gelir (Osyczka, 2002). Mühendisliğin çeşitli alanlarındaki tasarım optimizasyonu örnekleri aşağıda verilmiştir.

- Yaylar (Osyczka, 2002)
- Elektromanyetik sistemler (Winter, 1995)
- Basınçlı kaplar (Coello, 2002)
- Kaynaklı kirişler (Coello, 2002)
- Güçlendirilmiş beton kirişler (Shih, 2000)
- Yarı-rijit bağlantılı çelik kafesler (Kameshki, 2001)

4. TASARIM OPTİMİZASYONU

4.1 Tasarım Optimizasyonunun Mühendislikteki Yeri

Bir sorunu çözmek durumunda kalan her mühendis kendini bir optimizasyon probleminin içinde bulur. Mühendislik problemlerinin genelde birden çok çözümü mevcut olduğundan, bunların içinden en iyi olanı seçilmek zorundadır. Bu sebeple optimizasyon çözümlerin arasından en iyi olana seçmek olarak ta tanımlanabilir. Tabiî ki burada en iyiden kastedilen sadece en ucuz, en güzel veya en güçlü olan gibi tanımlar ile sınırlanamaz. Mühendislikte en iyinin anlamı sorunu isteklere ve sınırlamalara göre değerlendirip uygun çözümü bulmaktır.

Mühendislik tasarımlarındaki optimizasyon ile ilgili çalışmalar uzun sürerdi devam etmektedir. İnsanların istekleri arttıkça ve ekonomik sınırlamalar olduğu sürece mühendislerin önündeki en iyi çözümü bulma problemi daha da zorlaşarak devam edecektir.

Bilgisayarların ve programların gelişimi ile imkanlar artmış ve problemlerin çözümü daha hızlanmış oldu. Son yıllarda sıklıkla karşılaştığımız CAD/M/E* yazılımları mühendislik problemlerini nümerik çözümlerine dayalı hızlı ve çok yaklaşık sonuçlar verebilen araçlar olarak karşımıza çıkmaktadır ve gün geçtikçe kullanımı artmaktadır. Bilgisayarların performansları arttıkça gerçek sistemler daha ayrıntılı modellenebilmekte, bu modeller ile yapılan çözümlerin yaklaşıklığı daha iyi olmaktadır. Bu gelişmelerin olmasıyla bilgisayar kullanılarak optimizasyon problemlerini çözmek daha az zaman almakta ve sonuçların değerlendirilmesi yapılması gereken iyileştirmeler daha fazla zaman kalmaktadır. Bu sebeple bilgisayar yardımı ile optimizasyon (CAO**) yazılımları geliştirmiştir (Arora, 1989).

Optimum şekillendirme sırasında dikkat edilmesi gereken şartlar:

1. Olmazsa olmaz şartlar (sınır şartları): İşlev ve mukavemet bakımından minimum sağlanması gereken şartlar.
2. faydaların artırılması için gerekli şartlar: İşlevleri artırıcı veya iyleştirici, kullanma sürelerini uzatıcı, verimi artıran ve kaliteyi yükselten şartlar (Kutay, 1997).

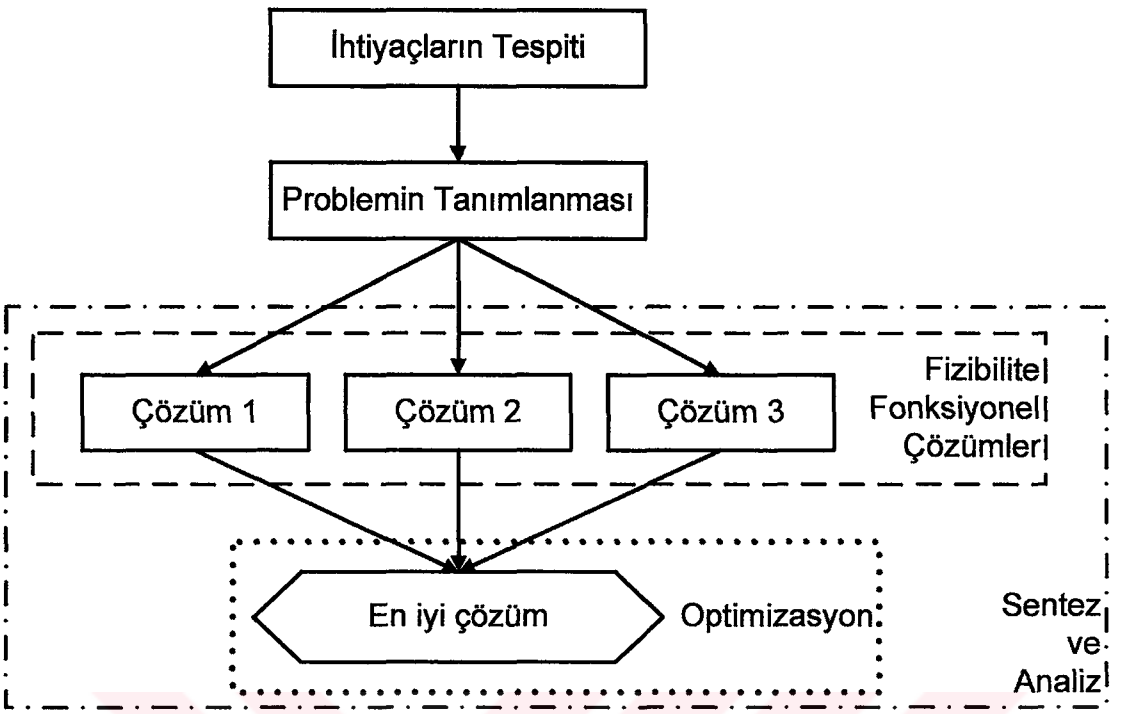
Yukarıda belirtilen şartlardan birincisi mutlaka yerine getirilmesi gerekirken, ikincisi için ise

---

* CAD/M/E: Computer Aided Design / Manufacturing / Engineering (Bilgisayar destekli tasarım / üretim / mühendislik.

** CAO: Computer Aided Optimization.

optimum bir değer tespit edilmelidir.



Şekil 4.1 Konstrüksiyonda Optimizasyon (Kutay, 1997)

Tasarım bir optimizasyon problemi olarak formüle edilirse, probleme ait tüm sınırlamaları sağlamak şartı ile optimizasyon yapılabilir. Sonuç olarak içinde parametreleri ve sınırlamaları barındıran her problem bir optimum tasarım problemi olarak ele alınabilir. Bu teknikler genel olması sebebi ile bir çok alana uygulanabilir.

4.2 Temel Tanımlar

Bu kısımda optimizasyon ile ilgili önemli kavramlar açıklanmaya çalışılacaktır.

4.2.1 Optimizasyon Ölçütü (Optimization Criterion)

Optimizasyon, bit problemin mümkün olanlar arasından en iyi belirlenmesi olarak tanımlana bilir. Ancak bu tanım bazı soru işaretleri de ortaya çıkarmaktadır. Öncelikle tanımda kullanılan "en iyi" ifadesinin tek başına bir anlamı olmayacağı görülebilir. Neye göre yada hangi ölçüte göre en iyiden bahsediyoruz? Burada "en iyi" sözcüğünün bir anlam ifade etmesi için bu sorunun yanıtlanması gereklidir. Optimizasyon problemi ancak tam olarak tanımlanmış bir optimizasyon ölçütü ile oluşur. Örnek olarak, bir mühendisin tasarladığı parçanın hafif olmasını istediğini düşünelim. Bu durumda optimizasyon ölçütü ürünün hafifliği olacaktır ve "en iyi" denildiğinde "en hafif" anlaşılacaktır (Arora, 1989. Turhan,).

Bir optimizasyon probleminde, optimizasyon ölçütünü tam ve açık bir biçimde tanımlanmış olması önem taşır. Optimizasyon ölçütü şartlara göre öznel yargılar sonucu belirlenir. Örneğin aynı mühendislik ürünü için "ucuzluk (maliyet)" da optimizasyon ölçütü olarak seçile bilir "dayanıklılık (mukavemet)" da. Bu sebeple en iyi çözümün aranması sırasında ölçüt, boyutlar veya hacim olabileceği gibi, maliyet, ağırlık, kalite, verim, mukavemet v.b. gibi ölçütlerde olabilir. Bunlardan herhangi biri veya birden fazlası dikkate alınarak tasarım optimizasyonu yapılabilir. Örneğin, ağırlık, hacim, maliyet minimum, kalite, verim veya mukavemetin maksimum olması için optimizasyon yapılabilir. Farklı yaklaşımların sonucu olarak seçilen bu ölçütlere göre en iyi ürünün birbirinden farklı olacağı, hatta birine göre en iyi olanın diğerine göre en kötü olabilme ihtimali vardır. Bu sebeple herhangi bir optimizasyon probleminin çözümünün yalnızca kullanılan optimizasyon ölçütüne göre en iyi olduğu unutulmamalıdır (Arora, 1989. Turhan,).

4.2.2 Performans Göstergesi

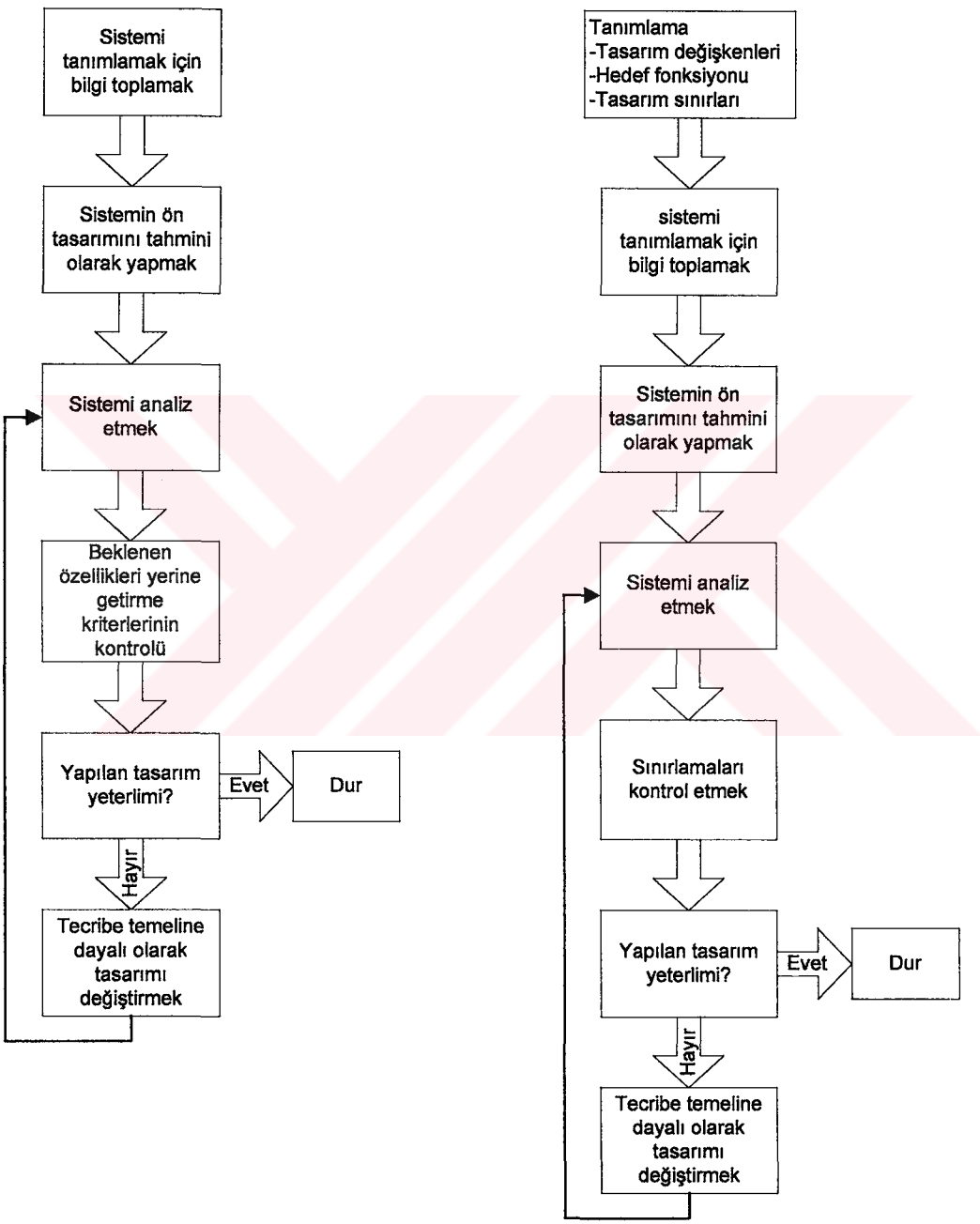
Bu ifade daha sonra karşımıza çıkacak olan "hedef fonksiyonu" ifadesine karşılık gelmektedir. Optimizasyon ölçütünün belirlenmesi ile, aranan "en iyi" tanımlanmış olur. Ancak ayır edilmesi için yeterli değildir. En iyinin ayır edilmesi için farklı ürünlerin seçilen optimizasyon ölçütüne göre birbirleri ile karşılaştırılabilmesi gerekir. Bu karşılaştırmanın bilimsel çerçevede yapılması için ölçülebilirliği ön şarttır. Bu nedenle her ürüne konulan optimizasyon ölçütüne göre ne kadar iyi olduğunu belirten bir skaler büyüklük ile eşleştirmek gerekir. Bu skalere "performans göstergesi" veya "performans ölçütü" denir (Arora, 1989. Turhan,).

Optimizasyon ölçütünün belirlenmesi, ilgili performans göstergenin de kendiliğinden belirli hale geleceği sanılsa da her zaman bu geçerli olmaz. Örneğin, bir mühendislik ürününde optimum tasarım için konulabilecek üç optimizasyon ölçütü hafiflik, dayanıklılık ve kullanışlılık olsun. Optimizasyon ölçütü olarak hafiflik seçildiğinde performans göstergesi, "P = ürünün ağırlığı" olarak ortaya çıkar. Ölçüt dayanıklılık olduğunda ise buna karşılık gelebilecek bir skalerin belirlenmesi zorlaşır. Eğer ölçüt kullanışlılık olursa buna karşılık gelen performans göstergesinin belirlenmesi imkansıza yakındır.

Performans göstergesinin bulunamaması durumundan optimizasyonu bilimsel çerçevede gerçekleştirme olanağı kalmaz. Bir bilim olarak optimizasyonun başlangıç noktasını performans göstergesi oluşturur. Optimum çözüm, performans göstergesinin ekstramumları olarak ortaya çıkar (Arora, 1989. Turhan,).

4.3 Bilgisayarın Optimum Tasarımdaki Rolü

Mühendislik sistemlerinde bilgisayar kullanılarak çok daha doğru ve hızlı analizler yapılabilir. Bu sayede sistemin davranışı daha doğru olarak anlaşılabilir. Tasarım süreci geleneksel veya optimum olabilir. Her iki durumda tekrarlanan hesaplar içerir. Ayrıca her tasarım bir çevrim gerektirir. Bu durumda bilgisayar, tasarım işlemindeki adımları kolaylaştır ve tekrarlı hesaplara hızlı cevap verirler (Arora, 1989).



Şekil 4.2 Geleneksel tasarım işlemi (sol), optimum tasarım işlemi (sağ) (Arora, 1989)

Problemde birden çok girdi ve çıktı olduğunda, sonuçların görselleştirilmesi gerektiğinde

grafiksel veya sistemin görüntüleri üzerinde anlatılabilir. Parçalardaki gerilme dağılımı araçların dinamik testleri gibi gibi pahalı ve zaman alıcı işler önce bilgisayarda denenerek daha tasarım sürecinde uygun olmayan çözümler elenebilir.

5. GENETİK ALGORİTMA

5.1 Genetik Algoritma Genel Bakış

Genetik Algoritma (GA) adından da anlaşılabileceği gibi doğadaki evrim mekanizmasını örnek alarak oluşturulmuş çözüm tekniğidir (Louis, 1993). GA'lar ilk defa Charles Darwin tarafından ortaya atılan, "doğal sistemde güçlülerin hayatta kalması" (en iyinin hayatta kalması) diye özetlenebilecek doğal seleksiyon ve doğal genetik mekanizmasına (genetiğin evrim tezine dayandırılan) dayanan araştırma algoritmalarıdır (Goldberg, 1989).

Genetik Algoritmalar, bazı doğal olayları modelleyen sezgisel (stochastic) algoritmalardır. Bu algoritmalar biyolojik evrimin işleyiş biçimini taklit eder. Doğal popülasyonlar (topluluklar), Charles Darwin'in, Türlerin Kökeni (The Origin of Species) adlı kitabında belirttiği gibi doğal seçme (natural selection) ve en iyi uyum yapanın yaşaması (survival of the fittest) prensibine göre kuşaklar boyunca gelişmektedir. Doğada, bireyler arasında yiyecek, su ve barınak gibi kıt kaynaklar veya eşler için yapılan mücadeleler, yüksek uyumlu (highly adapted) ya da uygun (fit) bireylerin daha zayıf olanlara üstünlüğü ile sonuçlanır. İyi uyum yapan bireyler yaşar ve daha fazla yavru sahibi olur. Düşük uyumlu bireyler az sayıda yavru sahibi olacak, belki de hiç yavru sahibi olamayacaktır. Bu, uygun bireylerin genlerinin sonraki her kuşakta (generation) daha fazla bireye dağılması demektir. Farklı atalardan gelen iyi özelliklerin birleşimi ile bazen her bir ataya göre daha büyük uygunluğa sahip süper uygun yavrular türeyebilir. Böylece türler, bulunduğu çevre için giderek daha uygun hale gelirler.

Örnek olarak tavşan topluluğunu (popülasyonu) düşünelim. Bazı tavşanlar diğerlerine göre daha hızlı ve daha açıkgözdür. Böyle tavşanların tilkiler tarafından yenilme olasılığı daha düşüktür. Dolayısıyla çoğu yaşamını sürdürür. Kuşkusuz, yavaş ve durgun olan tavşanların bazıları da şanslı oldukları için yaşayacaklardır. Bu yaşayan tavşan popülasyonu üremeye başlayacaktır. Popülasyonun çoğunluğu, tilkilerden kurtulan daha hızlı ve açıkgöz tavşanlar olduğundan, yavru tavşanlar önceki popülasyonlardakilerden ortalama olarak daha hızlı ve açıkgöz olacaktır. GA'lar bu süreci taklit eden sezgisel iteratif yöntemlerdir (Yeniay, 1999).

Genetik Algoritmaların mühendislik, tasarım, sinir ağı tasarımı, ekonomi ve finans, yapay hayat (artificial life), haberleşme ağları, tahmin (hava, deprem, at yarışı vb.), suç şüphelilerinin teşhisi, müzik besteleme, kombinatoryal optimizasyon (kutu paketleme, sıralama ve çizelgeleme, steiner ağacı, ulaştırma vb) gibi birçok alanda kullanıldığı yüzlerce çalışma mevcuttur.

5.1.1 Genetik Algoritmanın Tarihçesi

İlk defa 1960'larda John Holland tarafından ortaya atılan; kendisi, öğrencileri ve meslektaşları tarafından geliştirilen (Louis, 1993) GA'lar, mühendislik dünyasında birçok alanda genişçe işlenmiş, deneyleri yapılmış ve uygulanmıştır (Mitchell ve Forrest, 1993). Ayrıca 1975'de GA'nın teorik çatısının verildiği John Holland'ın "Doğal ve Yapay Sistemlerde Adaptasyon" (Adaptation in Natural and Artificial Systems) isimli kitabı yayınlandı (Louis, 1993). Holland'ın GA'sı Basit Genetik Algoritma (B.GA) olarak adlandırılmaktadır (Yeniay, 1999). Holland'ın asıl amacı, adaptasyon olgusunu tabiatta meydana geldiği şekliyle resmi olarak incelemek ve özel problemleri çözmek için algoritmalar düzenlemek yerine, doğal adaptasyon mekanizmalarının bilgisayar sistemlerine aktarılabileceği yollar geliştirmekti (Tanrıseven, 2000).

1992'de, John Koza Genetik Algoritma'yı, verilen bazı görevleri gerçekleştirecek programlar geliştirmek için kullandı. Koza, bu yönteme genetik programlama ("genetic programming" (GP)) adını vermiştir (Louis, 1993).

5.1.2 Genetik Algoritmaların Uygulama Alanları

GA'lar Bilgisayar Bilimi, Mühendislik, Yöneylem Araştırması, Sosyal Birimler, Tıp, Matematik vb alanlarda karşılaşılan çeşitli problemlerin çözümünde kullanılmaktadır. GA'lar global optimal çözümü bulmayı garanti etmezler, ancak kabul edilebilir hızla, kabul edilebilir ölçüde iyi çözümler bulunmasında genel olarak başarılıdırlar. Belirli problemlerin çözümünde

özel teknikler varsa, bunların hem sonucun doğruluğu, hem de hız açısından GA'lara göre daha iyi işlemesi olasıdır. GA'ların esas alanı, bu tür tekniklerin olmadığı alanlardır. Mevcut tekniklerin iyi işlediği yerlerde bile, bu teknikleri GA'lar ile birleştirerek ilerlemeler sağlanmıştır (Cantoni, 2000).

GA uygulamalarında başarının anahtarı, GA'yı etkili bir şekilde kullanmak ve zindelik, güçlülük değerlendirmesini anlamlıca yapabilmektir. GA'ların cazibesi, güçlü araştırma algoritmalarında kolay ve zarif, olmasının yansıra çok boyutlu problemlerden iyi sonuçlar çıkarabilme gücünden de kaynaklanmaktadır (Mitchell ve Forrest, 1993; Bodnovich, 1995).

GA'lar problem çözmede alternatif metotlar sağlamakla kalmaz, aynı zamanda problemlerin çoğunda diğer geleneksel yöntemleri de tutarlıca gerçekleştirir. Mesela optimal parametreler bulmada yer alan gerçek dünya problemleri geleneksel metotlar için zor olabilir; fakat GA'lar için idealdir. GA'lar, optimizasyondaki etkili performanslarından dolayı yanlış bir kanıyla

işlem optimizeri olarak görülmektedirler. Aslında Genetik Algoritma'lara çok fazla bakış açısı vardır. Birçok kullanıcı, GA'ya problem çözücü olarak bakmaktadır. Fakat bu kısıtlayıcı bir bakış açısıdır.

GA'lar,

- Araştırma alanı geniş, karmaşık ve anlaşılması zayıfsa,
- Konudaki bilgi azsa ya da eldeki uzman bilgi, araştırma alanını daraltmada zorlanıyorsa,
- Matematiksel analiz elde edilemiyorsa,
- Geleneksel araştırma metotları başarısız olmuşsa faydalı ve etkilidirler.

GA yaklaşımının avantajı, zorluklarla ve hedeflerle başa çıkmadaki rahatlığıdır.

GA'lar, problem çözme ve modellemede (taslak oluşturmada) kullanılmaktadır. GA'lar, bilimsel mühendislik problemlerinde, iş hayatında, ekonomi ve finansta, yapay hayatta (artificial life), haberleşme ağlarında ve eğlencede uygulanmaktadır. Bunlar:

Optimizasyon; GA'lar, içinde sayısal optimizasyon ve seyyar satıcı problemi gibi bütünleştirici optimizasyonların, devre dizaynlarının iş ve dükkan programlamalarının, video ve ses kalitesi optimizasyonlarının yer aldığı bir çok optimizasyon işinde kullanılmaktadır (Goldstein, 1991; Louis, 1993).

Otomatik Programlama: GA'lar, özellikli işler için bilgisayar programı geliştirerek, hücresel otomasyon ve ağ gibi bilgisayarla ilgili yapıları geliştirmekte kullanılmaktadır.

Bilgisayar ve robot öğrenimi: GA'lar, içerisinde sınıflama, tahmin ve protein yapısı tahmininin yer aldığı birçok bilgisayarlı öğrenim uygulamalarında kullanılmaktadır. Aynı zamanda ağ tasarlamak, sembolik üretim sistemlerine yönelik öğrenme kurallarını geliştirmek, robot tasarlamak ve kontrol etmekte kullanılmaktadır.

Ekonomik Modeller: GA'lar, açık artırmayla satış stratejilerini, yenilik süreçlerini, ekonomik piyasaların önceliklerini tasarlamakta kullanılmaktadır.

Bağışıklık Sistemi Modelleri: GA'lar, içerisinde bireyin yaşamındaki somatik mutasyon da olan doğal bağışıklık sisteminin çeşitli yönlerini tasarlamada (düzenlemede) kullanılmaktadır.

Ekolojik Modeller: GA'lar, biyolojik silahlanma yarışı, ortak yaşam ve ekolojilerdeki kaynak akışı gibi ekolojik fenomenlerin tasarlanmasında kullanılmaktadır.

Popülasyon Genetiği Modelleri: GA'lar, "Bir gen hangi şartlar altında tekrar birleşim (rekombinasyon) için evrimsel olarak yaşayabilir?" gibi popülasyon genetiği sorularına cevap bulmada model oluşturmaktadır.

Evrim ile öğrenim arası etkileşimde: GA'lar, bireyin öğrenmesiyle türlerin evriminin birbirini nasıl etkilediğini incelemede kullanılmaktadır.

Sosyal Sistem Modellerinde: GA'lar, işbirliğinin gelişmesini (evrimleşmesi), iletişimin gelişmesini ve karıncalarda ki birbirini takip etme davranışları gibi sosyal sistemlerin evrimle ilgili yönlerini inceler.

5.1.3 Genetik Algoritmaların Kullanılma Nedenleri

- GA, doğadaki evrim mekanizmasını örnek alan bir arama metodudur ve bir veri grubundan özel bir veriyi bulmak için kullanılır (Goldstein, 1991; Valenzuela, 1995). GA'lar özellikle araştırmacının kesin, konu uzmanı olmadığı zamanlarda çok yardımcıdır. Çünkü GA'lar, kendi alanlarını araştırma ve o alandan bilgi edindirmede yeteneklidirler.
- GA'lar değerlendirme için yeni ve daha iyi sonuçlar üretmenin yanı sıra varolan potansiyel sonuçlan değerlendirmek için de tasarlandıklarından dolayı başka alternatifler için büyük yardım sağlarlar (Mitchell ve Forrest, 1993).
- GA'lar klasik yöntemlerin çok uzun zamanda yapacakları işlemleri kısa bir zamanda çok net olmasa da yeterli bir doğrulukla yapabilir.
- GA, doğadaki evrimin yöntemlerini kullanan bir arama türüdür. Bu yöntem sayesinde, klasik yöntemlerle çözülmesi çok zor kimi zaman, imkansız olan problemler (NP-tam, NP-complete) çözülebilmektedir (Goldberg, 1989; Lawrence, 1990).
- GA'lar, çeşitli mühendislik problemlerini çözmek için kullanılmışlar ve optimizasyon problemleri için oldukça etkili olmuşlardır.
- GA, gerçek optimal değeri bulmayı garanti etmez, fakat uzun nesiller sonunda optimal değere çok yakın çözümler bulunmasını sağlar (Goldstein, 1991; Valenzuela, 1995).

Problem çözmedeki kullanılışlığı ve zarifliği, GA'yı diğer sıralı araştırma (graded search), rasgele araştırma (random search) vb geleneksel metotlar arasından tercih edilir hale getirmiştir.

Çok amaçlı problemleri çözerken GA, hedefler açısından birçok doyurucu çözümler sunar ve karar verecek kişinin en iyisini seçmesine müsaade eder. GA, çok amaçlı problemlerde karar vermenin taslak aşamasında yardımcı olur (Mitchell ve Forrest, 1993).

5.2 Genetik Algoritmanın Gelişimi

Çözüm uzayının çok büyük olduğu gerçek hayat problemleri için eniyi çözümün bulunması geliştirilen özel algoritmalarla bile çok uzun zaman almaktadır. Bu nedenle bu tür problemler için en iyiye yakın çözüm veren sezgisel tekniklerin geliştirilmesi önem kazanmaktadır. Sezgisel teknikler, makul zamanda iyi bir çözüme ulaşmak için problemdeki bilgiyi kullanırlar. Ancak, klasik eniyileme tekniklerinin aksine bu yaklaşımlar global eniyiyi bulmayı garanti etmezler. Sezgisel teknikler, "çözüm kurucu" ve "çözüm iyileştirici" olmak üzere iki ayrı sınıfta incelenmektedirler. Çözüm kurucular çeşitli kuralları kullanarak problem için bir çözüm elde ederken çözüm iyileştiriciler, elde edilen bir başlangıç çözümünü bitirme koşulu sağlanıncaya kadar adım adım iyileştirmeye çalışırlar. Bilinen çözüm iyileştirici sezgiseller bir problem için global en iyiyi bulmada çok başarılı değildirler. Son yıllarda çözüm iyileştirici sezgisel teknikler sınıfında bulunan ve global en iyiyi bulmada başarılı olan yeni teknikler geliştirilmiştir. Bu tekniklerden yaygın olarak kullanılanları, Tabu Arama, Genetik Algoritmalar, Sinir Ağları ve Tavlama Benzetimi'dir (Dengiz ve Altıparmak, 1998).

Çizelge 5.1 Sezgisel teknikler

SEZGİSEL TEKNİKLER	
1. Çözüm Kurucu	2.Çözüm İyileştirici
	Çözüm iyileştirici olarak geliştirilen yeni teknikler. • Tabu Arama (TA) • Genetik Algoritma (GA) • Sinir Ağları (S.A.) • Tavlama Benzetimi (T.B.)

5.2.1 Genetik Algoritmanın Tanımı

GA'ların oldukça başarılı olduğu alanlardan biri, optimizasyondur. Bir optimizasyon problemi için Genetik Algoritma'lar, doğada geçerli olan en iyinin yaşaması kuralına dayanarak sürekli iyileşen çözümler üretir. Bunun için "iyi" nin ne olduğunu belirleyen bir uygunluk (fitness) fonksiyonu ve yeni çözümler üretmek için yeniden kopyalama (recombination), değiştirme (mutation) gibi operatörleri kullanır. Genetik Algoritmaların bir diğer önemli özelliği de bir grup çözümle uğraşmasıdır. Bu sayede çok sayıda çözümün içinden iyileri seçilip kötüleri elenebilir.

GA ilk önce aday (rasgele) çözümlerin oluşturulmasıyla başlatılır. Bu çözüm kümesine

"popülasyon" denir (Goldberg, 1989; Biegel ve Davern, 1990; Lawrence, 1990). Aday çözümler, aynı sayıda elemandan oluşan bir bit dizisi formunda oluşturulur. Bir dizi olarak gösterilen her çözüm, birey ya da "kromozom" olarak bilinir. Böylece her bir kromozom n uzunluğunda bir bit dizisidir. Kromozom üzerindeki yerlerde bulunan değişkenlere gen, genin olası değerlerine o genin allelleri adı verilir. Bir genin kromozomda bulunduğu yere lokus denir. Biyolojiden bilinen bir örnek verilirse, göz rengi bir gen tarafından belirlenmektedir. Farklı göz renkleri, genin allelleridir. Göz renginin kromozom içerisindeki yeri, genin lokusudur.

Bir optimizasyon probleminde, kromozomlardan diziler, vektörler ya da çözümler olarak söz edilir. Değişkenlere genler, genin olası değerlerine alleller ve değişkenin pozisyonuna lokus denilir. Basit örneklerde değişkenin (genin) lokusu genellikle önemsizdir, ancak daha karmaşık problemlerde önemli olmaktadır (Yeniay, 1999).

Kromozomun her biri probleme bir çözüm sunar. Her çözüme, popülasyondaki diğer çözümler ile karşılaştırıldığında ne kadar iyi bir çözüm olduğunu gösteren bir uyum değeri (fitness value) atanır. Bu değer, doğada bir organizmanın yaşayabilmek için mücadele ederken ne kadar başanlı olduğunun belirlenmesine karşılık gelir. Bir bireyin uyum değeri ne kadar büyükse, sonraki kuşakta yaşama ve üreme (çözümün seçilme) şansı o kadar fazla olur. Daha uygun bireylere, popülasyondaki diğer bireyler ile eşlenerek üreme (reproduction) fırsatı verilir. Bu ise, her atanın bazı özelliklerini taşıyan yeni bireyleri meydana getirir. Popülasyondaki düşük uyumlu bireylerin, üreme için seçilme şansları azdır. Bu nedenle yok olurlar. Mevcut kuşağın en iyi bireylerini seçip, yeni bireyler elde etmek için bunları eşleyerek, olası çözümlerden oluşan yeni bir popülasyon yaratılır. Yeni kuşak, önceki kuşağın iyi bireylerinin sahip olduğu özellikleri, kuşaklar boyu popülasyona dağılır. Daha uygun bireylerin eşleşmesi sağlanarak uzayının en umut verici bölümleri incelenir. Eğer, GA iyi düzenlenirse, popülasyon problemin optimal çözümüne yakınsar. Bu döngü optimizasyon kriterleri uyuşuncaya veya belirli sayıda tekrarlamalar yapılıncaya kadar, yani istenen çözüm bulununcaya kadar uygulanır (Goldberg, 1989; Biegel ve Davern, 1990; Lawrence, 1990).

5.2.2 Ana Hatlarıyla Genetik Algoritma

1. [Başlat] Problem için rasgele n kromozomdu popülasyon oluşturulur. Problem için n tane çözüm önerilir.

2. [Uygunluk] Popülasyondaki her bir x kromozomu için f(x) uygunluk (fitness) fonksiyonu hesaplanır.

3. [Yeni popülasyon] Yeni bir popülasyon oluşuncaya kadar aşağıdaki adımlar tekrar edilir:

a) [Seçim] Uygunluk (Fitness) durumuna göre popülasyon dan iki tane kromozom çaprazlanmak (crossover) amacıyla seçilir. Uygunluk (Fitness) derecesi yüksek olanın seçilme şansı yüksektir.

b) [Çaprazlama] Seçilmiş olan ebeveyn kromozomlar, çaprazlama oranına (crossover probability) göre yeni yavrular oluşturmak üzere çaprazlanırlar. Eğer çaprazlama uygulanmazsa bireyler atalarının tamamen kopyası olacaklardır.

c) [Mutasyon] Kromozom üzerindeki bazı DNA dizilerinin (string) yerleri ile oynanarak belirli mutasyon oranına göre değişiklikler yapılır.

d) [Kabul] Oluşturulan yeni bireyler yardımıyla yeniden bir popülasyon oluşturulur.

4. [Değiştirme] Oluşturulan yeni popülasyon eksileriyle yer değiştirilir.

5. [Test] Programı bitirme şartı gerçekliyorsa program durdurulur ve popülasyondaki en iyi çözüm en iyi (best) çözüm olarak alınır.

6. [Döngü] Aksi takdirde ikinci adım tekrarlanır.

Yukarıdaki algoritmadan da anlaşılacağı üzere GA oldukça genel prensiplerle çalışır.

5.2.3 Genetik Algoritmaların Çalışma Prensibi

Genetik Algoritmanın çalışmasını aşağıdaki gibi özetleyebiliriz.

Çizelge 5.2 Genetik algoritmanın işleyişi (Çetin, 2002)

Adım 1	Olası çözümlerin kodlandığı bir çözüm gurubu oluşturulur (çözüm grubu, biyolojideki benzerliği nedeniyle, toplum (population), çözümlerin kodlarlıda kromozom olarak adlandırılır).
Adım 2	Her kromozomun ne kadar iyi olduğu bulunur.
Adım 3	Bu kromozomlar eşlenerek yeniden kopyalama ve değiştirme operatörleri uygulanır. Bu sayede yeni bir toplum oluşturulur.
Adım 4	Yeni kromozomlara yer açmak için eski kromozomlar ortadan kaldırılır.
Adım 5	Tüm kromozomların uygunlukları tekrar hesaplanır.
Adım 6	Çalışmayı sonlandırma şartı sağlanmamış ise üçüncü adıma geri dönülür.
Adım 7	O ana kadar bulunan en iyi kromozomlar sonuçtur.

Yukarıdaki algoritmayı (adımları) 4 adımda açıklayalım:

1) Bir çözüm grubu oluşturulur. Bu grubun oluşturulması tamamen rasgele olabileceği gibi probleme göre özelleşmiş de olabilir. Tamamen rasgele bir çözüm grubu genetik algoritmaya tüm problem uzayını arama şansı verir; probleme göre özelleşme ise işlemi önemli oranda hızlandırabilir. Çözüm grubunun büyüklüğü de önemli bir faktördür. Büyük çözüm grupları çok işlem gerektirir. Küçük çözüm grupları, yerel maksimum değerlerine takılabilir. Başarılı bir çözümün küçük bir grupta baskın hale geçmesi çok daha kolaydır. Bu nedenle küçük çözüm grupları çeşitliliklerini çok çabuk kaybederler.

2) Eldeki çözüm grubunun içinden başarılı çözümler seçilir. Seçme işleminin temel mantığı, doğadaki gibi başarılı olan çözümlere üstel çoğalma imkanı vermekle açıklanabilir. Bilgisayar ortamında, bellek ve işlem zamanı sınırlı olduğu için, başarılı çözümleri çoğaltmak diğer bireyleri azaltmak anlamına gelir. Bu yolla, çözüm grubunun büyüklüğü sabit tutulabilir. Kullanılan seçme yöntemlerinin amacı, başarılı bireylere üstel çoğalma imkanı vermekle çeşitliliği korumayı en verimli şekilde dengelemektir. Sıklıkla kullanılan yöntemler arasında, rulet tekerleği seçimi, turnuva seçimi, sigma ölçeklendirmesi, Boltzman seçimi, derece seçimi, kararlı durum (steady state) seçimi sayılabilir. Yardımcı olarak kullanılan elitist seçim ise, en başarılı bireylerin bir sonraki çözüm grubuna değiştirilmeden aktarılmasıdır. Bazı uygulamalarda başarılı olduğu görülmüştür.

Başarılı bireyler kullanılarak yeni çözüm grubu oluşturulur. Yeni çözüm grubunu oluşturmak için kullanılan işlemciler 'crossover' ve mutasyondur. Bu işlemciler de yapılan işleme göre değişse de genel yöntemleri pek farklı değildir.

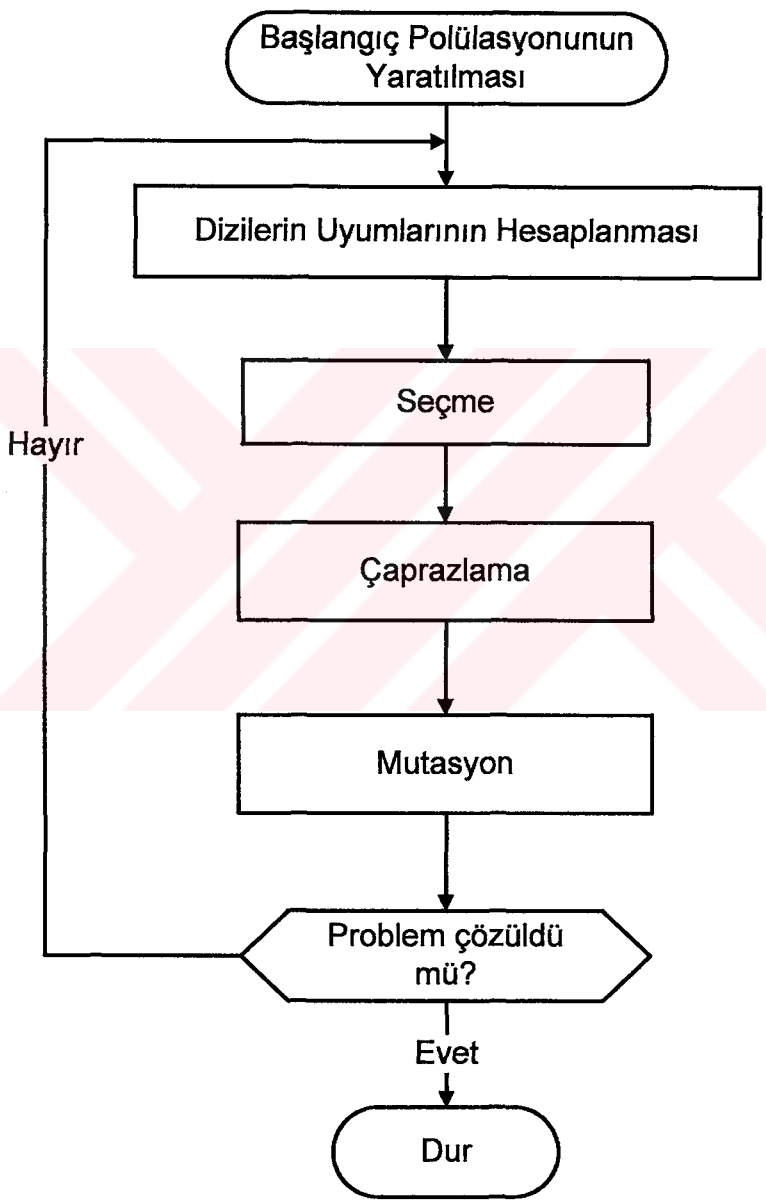
3) Çaprazlama (crossover) iki çözümün yapıtaşları kullanılarak yeni bir çözüm oluşturulması esasına dayanır. Bu işlem doğada görülen 'crossing over' olayının analoğudur. Çaprazlama işlemi genel olarak ikili dizilerin parçalarının değiş tokuşu şeklinde gerçekleştirilir. Farklı uygulamalarda, farklı kodlama yöntemleri kullanıldığı için farklı 'crossover' yöntemleri kullanılır.

Mutasyon bir çözümün, çoğunlukla rasgele olarak, değiştirilmesidir. Bu işlem çok değişik şekillerde kullanılır. Problemin yapısı bu aşamada çok önemlidir. Örneğin sıralama problemlerinde sıralamayı değiştirmek, ikilik dizi gösteriminde bitleri değiştirmek, çözüm ağaçlarında parçalan değiştirmek işlem olarak tanımlanır.

4) 2. ve 3. işlemler belirlenen bir şart (optimizasyon kriterleri uyuşuncaya kadar) sağlanana kadar tekrarlanır. Bu şart tekrar sayısının belirlenen bir sayıya ulaşması, belirlenen bir başarım değerine ulaşılması, Genetik Algoritmanın daha başardı çözümler oluşturamaması vb. olabilir (Goldberg, 1989; Biegel ve Davern, 1990; Lawrence, 1990).

5.3 Genetik Algoritmanın İşleyişi

Genetik algoritmanın işleyişi Şekil 5.1 de gösterilmiştir.



Şekil 5.1 Genetik algoritma akış diyagramı (Yeniay, 1999)

5.3.1 Başlangıç Popülasyonunun (Yığınının) Oluşturulması

GA'nın uygulamasında ilk adım, kromozomların bir başlangıç popülasyonunun oluşturulmasıdır. Başlangıç popülasyonu, çoğunlukla rasgele olarak oluşturulur. Ancak, özellikle kısıtlı optimizasyon problemlerinde, rasgelelik, uygun olmayan (infeasible) çözümlere neden olabilir. Bu durumdan kaçınmak için, genellikle incelenen probleme özgü sezgisel yöntemlerden yararlanılır. Örneğin, Grefenstette 1987'de yaptığı çalışmada, gezgin satıcı problemi için greedy sezgisellerinden; Kapsalis ve arkadaşları 1993'de yaptığı çalışmada, Steiner ağaç problemi için minimum ağaç yaklaşımından; Thiel ve Vass 1994'de yaptığı çalışmada, 0-1 sırt çantası problemi için ekleme çıkarma sezgiselinden; Chen ve arkadaşları ise 1995'de çizelgeleme problemi için Campbell-Dudek-Smith ve Dannenbrig sezgisellerinden yararlanarak başlangıç popülasyonunu oluşturmuşlardır.

GA, bir popülasyonu değerlendirir ve uygulanan bir süreç sonunda yeni bir popülasyon elde eder. Birbirini izleyen her popülasyona, kuşak denir.

GA'nın herhangi bir uygulamasında, popülasyon genişliğinin (N) belirlenmesi gerekir.Büyük popülasyonlarda, çözüm uzayı iyi örneklendiği için aramanın etkinliği artar, ancak makul bir sürede yüksek kalitede çözüme yeterince ulaşamadan ağır bir hesaplama külfetiyle karşılaşılabilir. Küçük popülasyonlar ise çözüm uzayını yeterli bir biçimde örnekleyememe riskini taşır ve zamansız yakınsama (premature convergence) görülebilir. Popülasyon genişliği için, 50-100 arasında kalan değerler yaygın olarak kullanılmaktadır. Ancak 30 diziye sahip popülasyonlar ile tatmin edici sonuçlara ulaşan birçok uygulama da vardır (Yeniay, 1999).

5.3.2 Uygunluk Fonksiyonu (Fitness Function)

Uygunluk değeri, yeni yığına taşınacak dizilerin belirlenmesinde kullanılan bir araçtır. Bu nedenle, algoritmanın her çevriminde, yığındaki dizilerin bir değerlendirme fonksiyonu yardımıyla uygunluk değerleri hesaplanır. GA'da kullanılan değerlendirme fonksiyonu problemin amaç fonksiyonudur (Dengiz ve Altıparmak, 1998).

Kromozomların ne kadar iyi olduğunu bulan fonksiyona uygunluk fonksiyonu denir. Bu fonksiyon işletilerek kromozomların uygunluklarının bulunmasına ise hesaplama (evaluation) adı verilir. Bu fonksiyon, Genetik Algoritma'nın beynini oluşturmaktadır. Genetik Algoritmada probleme özel çalışan tek kısım bu fonksiyondur. Uygunluk fonksiyonu, kromozomları problemin parametreleri haline getirerek onların bir bakıma şifresini çözmekte (decoding), sonra bu parametrelere göre hesaplama yaparak kromozomların uygunluğunu

bulmaktadır. Çoğu zaman Genetik Algoritma'nın başarısı, bu fonksiyonun verimli ve hassas olmasına bağlı olmaktadır.

GA ile çözülecek her problem için uyum fonksiyonu (fitness function) gerekir. Belirli bir kromozom verildiğinde, uyum fonksiyonu, o kromozom ile temsil edilen bireyin yararı (utility) ya da gücüyle (ability) orantılı olan bir sayısal değer verir. Uyum fonksiyonu, çevre rolünü oynamaktadır. GA, bir kromozomun uyum fonksiyonunu, o kromozomun çevreye uyum derecesini incelemede ve istenmeyen kromozomların yaşama ve üreme olasılığını azaltmada kullanır. Böylece algoritma, başarısız olan aday çözümlerin elenmesini sağlar. Bu nedenle, uyum fonksiyonunun seçimi, algoritmanın etkinliği üzerinde büyük öneme sahiptir. Çoğu problemde uyum fonksiyonuna karar vermek kolaydır (Yeniay, 1999).

5.3.3 Kodlama (Şifreleme) (Encoding)

GA'nın başarılı bir biçimde uygulanabilmesi için yapılması gereken ilk iş, bağımsız parametrelerin kromozomlar içerisinde kodlanmasıdır. Bu kromozom, belirli bir çözüm hakkındaki tüm bilgileri taşır. Kodlama mekanizması, problemin değişkenlerinin yapısına bağlıdır ve yöntemin performansında önem taşır.

Kodlama (Şifreleme) problemin tipine bağlı olarak çözüm adaylarını temsil eden yapılardır. Kodlamanın çeşitli yolları vardır:

1. İkili Kodlama (Binary Encoding)
2. Permütasyonlu Kodlama (Permutation Encoding)

5.3.3.1 İkili Kodlama (Binary Encoding)

En yaygın kullanılan kodlama yöntemidir. Sayıların ikilik sistemde gösterimine dayanır. Bu yöntemde kromozomlar sıfır ve birlerden oluşan dizilerdir. Her parametre için n-bit kullanılır, n, her parametre için farklı olabilir.

İkili düzende kodlama, çok sık kullanılan bir kodlama tipi olmasına rağmen bazı sakıncaları vardır. Örneğin, çok değişkenli fonksiyon eniyilemesi için değişkenlerin alt ve üst sınırlarına bağlı olarak elde edilen dizi çok uzun olabilir. Aynı zamanda gezgin satıcı, çizelgeleme, karesel atama gibi kombinatoryal eniyileme problemlerinde ikili düzende kodlama, araştırma uzayını tam olarak temsil edememektedir. Bu nedenle literatürde, permütasyonlu kodlama (sayısal kodlama) daha sık kullanılmaktadır.

5.3.3.2 Permütasyonlu Kodlama (Permutation Encoding)

Sıralama problemlerinde permütasyonlu kodlama kullanılır. Bu kodlamada, her kromozom, dizideki bir sırayı temsil eden sayılardan oluşur.

Örneğin, gezici satıcı probleminde (travelling salesman problem) "Verilen şehirlerarasındaki uzaklıklar bilinirken, bütün şehirleri birleştiren minimum yol uzunluğu ne olmalıdır?" sorusuna cevap aranmaktadır. Burada kromozomlar, şehirlerin hangi sırada ziyaret edilebileceğini gösteren permütasyonlar şeklinde kurulurlar (Goldstein, 1991; Valenzuela, 1995).

5.3.4 Seçme

Kodlamaya karar verilip başlangıç popülasyonu oluşturulduktan sonra, yeni popülasyona geçebilmek için seçimin nasıl yapılacağı belirlenmelidir. Seçme, bireylerin üreme için seçildiği bir işlemdir. İlke olarak, yüksek uyuma sahip bireyler daha büyük bir olasılıkla seçilmelidir. Seçim operatörünün, yüksek uyuma sahip bireylerin sonraki kuşağa kopyalanma olasılığını artırması istenir. Böylece seçim, arama uzayının başarılı olacağı umulan bölgelerinde yoğunlaşır. Sonraki kuşak için, yavru meydana getirecek bireylerin popülasyondan nasıl seçileceği ve her bir bireyin kaçar tane yavruya sahip olacağı belirlenmelidir. Sonraki kuşak için yavru meydana getirecek bireyler ile üremehavuzu (mating pool) adı verilen bir ara popülasyon oluşturulur. Daha sonra, üreme için bu havuzdan rasgele bireyler seçilir. Seçimde iki önemli konu vardır: Popülasyon çeşitliliği (population diversity) ve seçicilik baskısı (selective pressure). Bu faktörler birbirleriyle ilişkilidir. Seçicilik baskısındaki bir artış, popülasyonun çeşitliliğini azaltır. Bunun aksi de doğrudur. Yani güçlü seçicilik baskısı, GA'nın zamansız yakınsamasını destekler, zayıf seçici baskı ise aramanın etkinliğini azaltabilir. Bu nedenle bu iki faktör arasında bir denge kurulmalıdır. Örnekleme mekanizmaları bu hedefi gerçekleştirmeye çalışır.

GA literatüründe çeşitli seçim planlan önerilmiştir. Holland'ın orjinal GA'srnda popülasyon seçimi, en iyi uyum yapanın yaşaması ilkesine dayanır. GA'da uygun bir dizi çok sayıda yavru elde eder ve bunun sonucunda, sonraki kuşakta daha yüksek bir yaşama şansına sahip olur. Mevcut popülasyondan bireyleri uygunluk değerlerine göre seçen yöntemler, uyuma orantılı seçim (fitness proportionate selection) yöntemleri grubu olarak adlandırılırlar (Yeniay, 1999).

Goldberg ve Deb (1991) literatürde mevcut ve çok sık kullanılan seçim mekanizmalarını Çizelge 5.3'de görüldüğü gibi, orantılı yeniden üretim (proportionate reproduction) mekanizması, sıralı (ranking), turnuva (tournament), denge durumu (steady state, diğer adıyla

Genitor) yeniden üretim mekanizmaları olmak üzere 4 ana sınıfta toplamışlardır.

Çizelge 5.3 Yeniden üretim mekanizmaları (Çetin, 2002)

Mekanizmalar	Açıklama
Orantılı Yeniden Üretim Mekanizması	Yığındaki her bireyin seçilme olasılıkları belirlenir ve bu olasılıklar kullanılarak bir sonraki yığın oluşturulur. Bu grupta bulunan yeniden üretim mekanizmaları: rulet çemberi, sezgisel (stochastic) artan ve sezgisel (stochastic) üniversal metodudur.
Sıralı Yeniden Üretim Mekanizması	Yığındaki bireyler, uygunluk değerlerine göre küçükten büyüğe doğru sıralanır. En iyiden başlayarak bir azalan fonksiyon yardımı ile dizilere kopya sayısı atanır ve orantılı seçim mekanizmalarından birisi kullanılarak yeni yığın elde edilir.
Turnuva Yeniden Üretim Mekanizması	Yığından rassal olarak bir grup dizi (yerine koyarak / yerine koymadan) seçilir ve grup içindeki en iyi uygunluk değerine sahip dizi, yeni yığına kopyalanır. Bu işleme, yığın genişliğine ulaşıncaya kadar devam edilir. Grup genişliği en az ikidir.
Denge Durum Yeniden Üretim	Doğrusal sıralı seçim mekanizması kullanılarak Mekanizması seçilen bir ya da iki bireye genetik operatörler uygulanır. Elde edilen yeni diziler, mevcut yığındaki uygunluk değeri en küçük diziler ile yer değiştirerek yeni yığın oluştururlar.

Literatürde, orantılı, sıralı ve turnuva yeniden üretim mekanizmaları ile birlikte elitist stratejisinin de çok sık kullanıldığı görülmektedir. Elitist stratejisi ile mevcut yığındaki en iyi bir ya da birkaç dizi bir sonraki yığına taşınır. Amaç, elde edilen en iyi uygunluk değerine sahip dizinin (veya dizilerin) örnekleme hatası ya da genetik operatörler kullanımı sonucunda kaybolmasını önlemektir (Dengiz ve Altıparmak, 1998).

5.3.4.1 Orantılı Yeniden Üretim Mekanizması

1 °) Yerine Koyarak Sezgisel (stochastic) Örnekleme (Rulet Çemberi Seçimi)

Yerine koyarak sezgisel (stochastic) örnekleme, rulet çemberi (rulette wheel) seçimi olarak da adlandırılmaktadır. Bu metot ilk defa Holland tarafından ortaya çıkarıldı ve muhtemelen Genetik Algoritmada kullanılan en eski ihtimale dayanan seçim metodudur. Muhtemel seçim metotları hayatta kalma ihtimali üzerine kurulan her bir kromozom kopyasının gerçek sayısını belirler. Böylece seçim bölümü her bir kromozomun beklenen değerinin kararlaştırılması ve bu değerin yeni nesil miktarına dönüştürülmesinden oluşur. Ana düşünce, her bir kromozomun seçilme ihtimalinin uygunluk değeriyle orantılı olduğudur. f_k uygunluk (fitness)

değerindeki

her bir kromozomun seçilme olasılığı (ihtimali) P_k dır (Aksoy, 1998). Öncelikle $P_k = \frac{f_k}{\sum_{k=1}^{N} f_k}$ eşitliği yardımıyla her bir bireyin P_k seçilme olasılığı hesaplanır. Bu sayılar bir tabloda tutulur. Tablodaki olasılık değerleri birbirine eklenerek rasgele bir sayıya kadar ilerlenir. Bu rasgele sayıya ulaşıldığında ya da geçildiğinde son eklenen sayının ait olduğu birey seçilmiş olur. Bu işlem N kez tekrar edilir. Bu yönteme rulet çemberi seçimi ismi, bir daireyi, bireylerin uyum değerlerine orantılı olarak dilimleyip çevirdiğimizde olacakların benzeşimi olduğu için verilmiştir.

Rulet tekerleği seçimini adım adım aşağıdaki gibi yazabiliriz:

1- Tüm bireylerin uygunluk değerleri bir tabloya yazdır.

2- Bu değerler toplanır.

3- Tüm bireylerin uygunluk değerleri toplama bölünerek [0,1] aralığında sayılar elde edilir. Bu sayılar bireylerin seçilme olasılıklarıdır. Sayıların hepsi bir tabloda tutulur.

4- Seçilme olasılıklarını tuttuğumuz tablodaki sayılar birbirine eklenerek rastgele bir sayıya kadar ilerlenir. Bu sayıya ulaşıldığında yada geçildiğinde son eklenen sayının ait olduğu çözüm seçilmiş olur.

Rulet çemberinde sezgisel (stochastic) hatalar nedeniyle bir bireye pay edilen yavru sayısı, beklenen yavru sayısından önemli ölçüde farklı olabilir. Bu nedenle rulet çemberi seçimi büyük örnekleme hataları doğurabilmektedir. İterasyon sayısı arttıkça, örnekleme hatası da artmaktadır. Pay edilen yavru sayısı, sadece çok büyük popülasyonlar için beklenen yavru sayısına yaklaşmaktadır. Rulet çemberi seçimindeki bu örnekleme hatası nedeniyle, arama

tahmin edilenden farklı yönlerde devam etmektedir. Bu ise algoritmanın muhtemelen yerel optimuma zamansız yakınsamasına neden olmaktadır. Bu sorunu ortadan kaldırmak için çeşitli seçim yöntemleri önerilmiştir.

2°) Yerine Koymadan Sezgisel (Stochastic) Örnekleme

Yerine koymadan sezgisel (stochastic) örnekleme, De Jong'un beklenen değer modelinin öteki adıdır. Bu yöntem, rulet çemberi seçiminin sezgisel (stochastic) hatalarını azaltmaktadır. f_k, k. bireyin uyum

değerini; $\sum_{k=1}^{N} f_k$, popülasyonun toplam uyumunu ve, $\bar{f} = \frac{\sum_{k=1}^{n} f_k}{N}$ popülasyonun ortalama uyum

değerini göstermektedir. Her birey için $\frac{f_k}{\bar{f}}$ beklenen yavru sayısı hesaplanır. Bu sayı, birey çaprazlamaya her seçildiğinde 0,5, çaprazlama olmadan üremeye seçildiğinde ise 1 azaltılır.

3°) Kalanı Sezgisel (Stochastic) Örnekleme

Kalanı sezgisel (stochastic) örnekleme yöntemi, yerine koyarak ve koymadan uygulanabilir. Yöntem, beklenen birey sayılarını bilinen biçimde hesaplar ve tamsayılı kısmını bireye atar. Popülasyon genişliğine ulaşılmadıysa, beklenen değerlerin kesirli kısımlarından yararlanılır. Yerine koyarak kalanı sezgisel (stochastic) örneklemede, beklenen değerlerin kesirli kısımları, rulet çemberi seçim yönteminde bireylere pay edilen ağırlıkları hesaplamak için kullanılır. Yerine koymadan kalanı sezgisel (stochastic) örneklemede, beklenen değerlerin kesirli kısımları kullanılarak birer birer ağırlıklı para atışları (Bernoulli denemeleri) yapılır. Örneğin, beklenen kopya sayısı 1.36 olan bir birey, bir kopyayı kesin olarak, diğer kopyayı ise 0.36 olasılıkla elde eder. Popülasyon tamamlanana dek bu süreç devam eder.

4°) Sezgisel (Stochastic) Evrensel Örnekleme

James Baker, bireye atanan gerçek yavru sayısı ile beklenen yavru sayısı arasındaki farkı minimize eden sezgisel (stochastic) evrensel örnekleme adında bir yöntem önermiştir. Temel düşünce tek bir seçimle tüm N bireyi örneklemektir. Bu yöntemi gerçekleştirmek için, rulet çemberine seçim çemberi (selection whell) denilen bir ek parça eklenir. Bu parça eşit aralıklı N göstergeye sahiptir. Çember bir kez çevrilir ve durduğunda göstergelerin bulunduğu yerler bireyleri gösterir. Böylece N birey, bir adımda seçilir. Göstergeler eşit aralıklı olduğu için, bir bireyin küçük popülasyonlarda bile tüm popülasyona hakim olma tehlikesi yoktur. Bu yöntemle, her bireyin beklenen yavru sayısı kadar (daha çok değil) üremesi garanti edilir. Bu algoritma ile seçim planının belirli bir işleyişinin sonucu beklenen davranışa mümkün olduğunca yakındır, yani ortalama değişim minimumdur.

5.3.4.2 Sıralı Yeniden Üretim Mekanizması (Rank Selection)

Sıralama seçimi, ilk olarak Baker tarafından uyuma orantılı seçimin dezavantajlarını yok etmek üzere önerilmiştir. Sıralama seçiminde bireyler (kromozomlar) uyum değerlerine göre (fitness durumlarına göre) sıralanır. Her bireyin beklenen değeri, sahip olduğu sıraya bağlıdır.

Bu sıra dikkate alınarak seçme yapılır. Bu seçme işleminde tüm kromozomların seçilme ihtimali mevcuttur. Fakat bu yöntem çok farklı olmayan bireylerin seçilmesine daha çok imkan tanıyacağından istenen tarama bölgesini çabuk taramayabilir.

5.3.4.3 Turnuva (Tournament) Seçim Yöntemi

Turnuva seçim yöntemi, seçim baskısı yönünden sıralama seçimlerine benzer, ancak işlemsel olarak daha verimli olup, paralel uygulamaya daha yatkındır. Turnuva seçiminde, popülasyondan yerine koyarak ya da yerine koymadan rasgele t birey seçilir, t büyüklüğüne turnuva genişliği adı verilir. Bu gruptaki en iyi birey, yeni popülasyona kopyalanır. Bu işlem N kez yinelenir. Büyük t değeri, yöntemin seçicilik baskısını artırır. Turnuvalarda çoğunlukla t=2 alınır ve ikili turnuva olarak adlandırılır. Bu yöntemde zamansız yakınsama, durağanlaşma ve açık uyuma gereksinim yoktur.

Sezgisel (stochastic) turnuva seçimi, Wetzel tarafından önerilmiştir. Bu yöntemde seçim olasılıkları hesaplanır ve rulet çemberi yardımıyla ard arda birey çiftleri belirlenir. Bir çift belirlendikten sonra, yüksek uyuma sahip olanı alınarak yeni popülasyona dahil edilir ve diğer çift seçilir. Süreç, popülasyon tamamlanana dek sürer.

5.3.4.4 Denge Durum Yeniden Üretim Mekanizması (Seçim Yöntemi)

GA'nın her iterasyonunda mevcut popülasyondan bütünüyle yeni bir popülasyon yaratılmaktadır. Bütün popülasyonu değiştiren GA'lara kuşaksal (generational) GA'lar denir. Bazı seçim planlarında (elitist plan gibi) birbirlerini izleyen kuşaklar bir derece örtüşmekte, yani önceki kuşağın bir bölümü yeni popülasyonda korunmaktadır. Her kuşakta popülasyonun değiştirilen oranına kuşak aralığı (generation gap) denir ve G ile gösterilir.

Böylece *t.* kuşağın *N(l-G)* adet bireyi, *(t+1)*. Kuşakta aynen yaşamaya devam eder. Kuşaksal Genetik Algoritmalarda *G=l* 'dir.

Denge durumu (steady state) seçiminde, her kuşakta sadece birkaç birey değiştirilir. Bu seçimde, ata olarak iki bireyin nasıl seçileceğine ve gelecek yavrulara yer açmak için hangi iki bireyin silineceğine karar verilir. Örneğin; Whitley'in GENİTOR algoritması ataları sıralanan uyum skorlarına göre seçmekte ve yavrular en kötü iki bireyin yerini almaktadır.

Elitist Seçim Yöntemi

İlk olarak Kenneth De Jong tarafından önerilmiştir. Bu yöntem, popülasyonun en iyi bir (ya da iki) bireyini koruyup, popülasyonun geri kalan elemanlarını uyuma orantılı seçim

yöntemlerinden birini kullanarak yeni bireyler ile değiştirir. Böylece en iyi bireylerin yaşaması sağlanır. Yani her nesilde en iyi olan en az bir çözüm, olduğu gibi bir sonraki nesile aktarılır. Yöntemin avantajı, en iyi uyum değerine sahip bireyin, örnekleme hatası ya da genetik operatörlerin kullanılması ile kaybolmasını önlemektir.

Ölçeklendirme Fonksiyonları

Orantılı seçme planı, bireyin uyum değerinin popülasyonun ortalama uyum değerine oranına bağlı kalarak yavruları paylaştırmaktadır. GA'nın başlangıç kuşaklarında popülasyon genellikle düşük ortalama uyum değerine sahiptir. Yüksek uyum değerine sahip birkaç bireyin olması orantılı seçim planının bu süper bireylere çok sayıda yavru pay etmesine neden olur ve bu bireyler popülasyona hakim olur. Bu ise zamansız yakınsamaya yol açar.

GA'nın sonraki aşamalarında ise farklı bir problem ortaya çıkar. Popülasyonda hala önemli çeşitlilik olabilir, ancak popülasyonun ortalama uyumu, popülasyonun en iyi uyumuna yaklaşabilir. Eğer bu durum kendi haline bırakılırsa, ortalama elemanlar ile en iyi elemanlar sonraki kuşaklarda hemen hemen aynı sayıda birey elde ederler. Her iki durumda da uyum ölçekleme yardımcı olabilir.

Yararlı bir ölçekleme prosedürü, doğrusal ölçeklemedir. Burada ham uyum f, ölçeklendirilmiş uyumu f' ile tanımlanırsa,

$$f' = af + b \tag{5.1}$$

dönüşümü kullanılır. Her kuşakta a ve b sabitlerinin, f'_{ort} ortalama ölçeklenmiş uyumu, f_{ort} ortalama ham uyumuna eşit olacak biçimde seçimi yapılır. Böylece seçim prosedürünün kullanımı, her bir ortalama popülasyon elemanının sonraki kuşağa bir yavru verebilmesini garanti eder. Maksimum ham uyuma sahip popülasyon elemanına verilen yavru sayısını denetleyebilmek için, ölçeklendirilmiş maksimum uyumu veren,

$$f'_{\max} = c \cdot f_{ort} \tag{5.2}$$

bağıntısı kullanılır. Burada c, en iyi popülasyon elemanı için istenilen kopya sayısıdır. Küçük popülasyonlarda $c=1.2$ ile 2 başarıyla kullanılmıştır.

Bazen ölçeklenmiş uyum değerleri negatif bulunabilir. Bu durum popülasyonun çoğu elemanı yüksek uyuma, bazıları ise çok düşük uyuma sahip ise, sıklıkla karşılaşılan bir problemdir. Bu sorunu önlemek için Forrest, popülasyon değişim bilgisini kullanmayı önermiştir.

Popülasyonun standart sapma bilgisini kullandığı için, bu sürece, sigma kesim planı (sigma truncation seneme) adı verilir. Sigma kesim planı,

$$f' = f - \left(\bar{f} - c\sigma\right) \tag{5.3}$$

eşitliğini kullanır. Burada sigma, popülasyondaki uyum değerlerinin standart sapmasıdır. c sabiti, popülasyon standart sapmasının makul bir katı olarak (genellikle 1 ile 3 arasında) seçilir. Uyum değerleri $\left(f' - c\sigma\right)$'dan daha küçük olan bireyler göz ardı edilir.

Gillies, üs kuralı ölçeklemeyi (power law scaling) önermiştir. Burada ölçeklenen uyum, ham uyumun belirli bir üssü olarak alınır:

$$f' = f^k \tag{5.4}$$

k, bire yakın bir sayıdır, k parametresi, f fonksiyonunu ölçeklendirir. Bazı çalışmalarda k 'nın seçiminin probleme bağlı olması gerektiği sonucuna varılmıştır.

5.3.5 Genetik Operatörler

Seçim prosedürleri bizi herhangi bir yeni çözüme götürmez. Seçilen bireyler, üreme havuzuna alınırlar ve üreme havuzu popülasyonun büyüklüğüne ulaştığı an, seçme işlemi son bulur. Seçilen bireyler yeni kuşağın atalarıdır. Gelişmiş yeni çözümler, bu aşamada genetik operatörler yardımıyla elde edilirler. Yeni çözümler elde etmede görev alan iki genetik operatör vardır. Bunlar çaprazlama ve mutasyondur.

5.3.5.1 Çaprazlama (Crossover)

GA'ya dayalı çoğu uygulamada en önemli operatör, çaprazlamadır. Çaprazlama, farklı çözümler arasında bilgi değişimini sağlayarak arama uzayının benzer, ancak araştırılmamış bölgelerine ulaşmayı sağlayan bir arama operatörüdür. Çaprazlamanın bireylerdeki iyi Özellikleri birleştirerek daha iyi çözümler yaratması beklenir. Literatürde en çok kullanılan çaprazlama operatörleri, Tek Nokta Çaprazlama (single point crossover), İki Nokta Çaprazlama (two point crossover), Çok Nokta Çaprazlama (multi point crossover) ve Üniform Çaprazlamadır (uniform crossover).

1°) Tek Nokta Çaprazlama

GA'nın kullanıldığı en basit çaprazlama türüdür. Öncelikle tüm popülasyon rasgele eşlenerek, olası ataları içeren N/2 küme elde edilir. P_c çaprazlama olasılığı yardımıyla çaprazlama uygulanacak çözüm çiftleri belirlenir. Çoğunlukla 0,6 ile 1,0 arasında bir P_c olasılığı

kullanılmaktadır. Eğer [0,1] aralığından rasgele seçilen bir sayı, P_c den küçük ise o çifte çaprazlama uygulanır, aksi durumda bu çözümler popülasyonda değişmeden kalır. Her yeni popülasyonda yaklaşık olarak P_cN tane bireye çaprazlama uygulanır.

Çaprazlama için; "0010011010" ve "1110010001" dizilerinin çaprazlamaya seçildiğini düşünelim. Öncelikle 1 ile n–1 arasında bir tamsayı (çaprazlama noktası) rasgele seçilir. Bu nokta 6 olsun. Bu nokta, Şekil 5.2 de gösterilmiştir. Daha sonra, çaprazlama noktasından sonraki iki alt dizi atalar arasında yer değiştirilerek, yavru adı verilen iki yeni çözüm elde edilir.

Ata 1	0	0	1	0	0	1	1	0	1	0
Ata 2	1	1	1	0	0	1	0	0	0	1

Çaprazlama noktası

Yavru 1	0	0	1	0	0	1	0	0	0	1
Yavru 2	1	1	1	0	0	1	1	0	1	0

Şekil 5.2 Tek nokta çaprazlama

Birinci ve ikinci yavru, her iki atasından farklı olma eğilimindedir, ancak iki atanın bazı özellikleri korunur. Eğer her iki ata da yüksek uyuma sahipse, yavrulardan en az birinin atalardan daha iyi olma şansı yüksektir. Böyle bir durum olursa, seçim, bu yavruların üremelerini; aksi halde, yok olmalarını destekleyecektir. Çaprazlama ile tümü ya da çoğu

düşük uyuma sahip yavrular da yaratılabilir. Bu olasılığı da ortadan kaldırmak için, P_c çaprazlama olasılığından yararlanılır. Çaprazlama olmadan, hiçbir ilerleme belirtisi olmayacağından P_c genellikle yüksektir.

2°) İki Nokta Çaprazlama

Bu tip çaprazlamadaki süreç tek nokta ile aynıdır sadece rastgele olarak 2 nokta seçilir.

3°) Çok Nokta Çaprazlama

Çok nokta çaprazlama, iki nokta çaprazlamanın bir uzantısıdır. Her bir çözüm, k çaprazlama noktası ile k parçaya ayrılır. Bir atlanarak elde edilen allel blokları, çiftler arasında değiştirilerek yavrular elde edilir.

De Jong, çok nokta çaprazlama operatörlerini test edip, daha fazla alt dizi değiştirdiğinden

performansın gerilediği sonucuna varmıştır. İlerlemeyi sürdürmek için bazı değişiklikler kazandırmak önemlidir. Ancak, çok fazla değişim yapmak, bir çözümün iyi özelliklerine zarar verme olasılığını yükseltmektedir. Bazı araştırmacılar ise, çok nokta çaprazlamanın dizilerdeki bazı iyi özellikleri birleştirmeye daha uygun olduğunu düşünmektedir. Bu nedenle, mevcut çözümlerdeki iyi özelliklerin işletilmesi ve yeni özelliklerin kazanılması arasındaki denge, etkin bir G. A. araması için çok önemlidir.

4°) Üniform Çaprazlama

Üniform çaprazlamada iki ata verildiğinde, birinci yavrunun her geni, rasgele olarak yaratılan bir çaprazlama maskesi (crossover mask) uyarınca, birinci veya ikinci atanın karşılık gelen geninin kopyalanmasıyla yaratılır. Çaprazlama maskesindeki "1", o genin birinci atadan kopyalanacağını, "0" ise o genin ikinci atadan kopyalanacağını ifade eder.

Rasgele yaratılan

1001011100 çaprazlama maskesi uyarınca,

1010001110 ve 0101010011

atalarına üniform çaprazlamanın uygulanışı, Şekil 5.3'de gösterilmiştir.

1. Ata	1 0 1 0 0 0 1 1 1 0
2. Ata	0 1 0 1 0 1 0 0 1 1

Çaprazlama maskesi	1 0 0 1 0 1 1 1 0 0

1. Yavru	1 1 0 0 0 1 1 1 1 1

Şekil 5.3 Üniform çaprazlama

Birçok gerçek uygulama için, probleme özel çözüm gösterimleri ve çaprazlama operatörleri geliştirilmiştir. Bu esneklik GA'ların çekiciliklerinden biridir.

Çaprazlama ile oluşan yavrular popülasyonda olmayan bilgileri alamazlar. Örneğin, mevcut popülasyonun tüm elemanları dizinin 1 pozisyonunda 1 içeriyorsa, çaprazlama bu pozisyonda 0 olan bir bireyi hiçbir zaman yaratamaz.

Son yıllarda GA literatürü çeşitli teknikleri karşılaştırmıştır. Özellikle tek nokta ve iki nokta çaprazlama ile tek nokta ve üniform çaprazlama karşılaştırılmaktadır. Teknikleri

sınıflandırmak amacıyla pozisyon ve dağılım eğilimi (positional and distributional bases) kavramları kullanılmaktadır. Eğer bir bitin değiştirilme olasılığı, dizideki pozisyonuna bağlıysa, çaprazlama operatörü pozisyon eğilimine sahiptir. Dağılım eğilimi, çaprazlama operatörüyle değiştirilen bitlerin sayısı ile ilişkilidir. Bu sayının dağılımı üniform değilse, çaprazlama operatörü dağılım eğilimine sahiptir.

Tek nokta çaprazlama en fazla pozisyon eğilimi, en az dağılım eğilimi göstermektedir. Çünkü çaprazlama noktası, üniform dağılım kullanılarak rasgele seçilmektedir. Ancak eğilimin bu eksikliği, iyi birşey değildir. Çünkü atalar arasında bilgi değişimini sınırlamaktadır.

Üniform çaprazlama, bitlerin pozisyonlarını göstermeksizin değiştirmektedir, ancak yüksek bozucu yapısı sık sık sorun olmaktadır. Tek nokta ve iki nokta çaprazlama, popülasyon homojen olduğunda, aramaya daha az yardımcı olmaktadır.

Diğer bir konu da popülasyon büyüklüğü ile çaprazlama türü arasındaki etkileşimdir. Deneysel sonuçlar, üniform çaprazlamanın küçük popülasyonlar için, iki nokta çaprazlamanın büyük popülasyonlar için daha iyi olduğunu göstermektedir. Üniform çaprazlamanın bozuculuğu, küçük popülasyonlarda oldukça keşifsel bir arama yapmaya yardımcı olmaktadır. Büyük popülasyonlarda var olan çeşitlilik keşif gereksinimi azaltmakta ve iki nokta çaprazlamayı daha uygun kılmaktadır (Yeniay, 1999).

5.3.5.2 Mutasyon

GA'larda diğer önemli operatör, mutasyondur. Mutasyon doğada olduğu gibi GA uygulamalarında da geri planda kalan bir operatördür (Yeniay, 1999). GA'nın çalışmasında ikinci dereceden rol oynar. GA'da mutasyon operatörü, küçük bir olasılıkla bir dizi içindeki bir veya birkaç değeri rassal olarak değiştirerek, yığında yeni dizilerin (yani arama uzayında yeni çözüm noktalarının) elde edilmesini sağlar (Goldberg, 1989). Her yığına mutasyon operatörü, P_m olasılığı ile uygulanır. İkili düzende kodlamanın kullanıldığı bir dizide, mutasyon operatörü ile rassal olarak seçilen eleman değeri 1 ise 0, 0 ise 1 olarak değiştirilerek yeni bir dizi elde edilir.

Özellikle GA'ın son çevrimlerinde mutasyonun etkinliği artmaktadır. Çünkü son çevrimlerde yığın iyi çözümlere yakınsadığından diziler birbirine çok benzemektedir. Bu durum, çaprazlama operatörünün farklı yeni diziler oluşturmasını engelleyerek aramayı kısıtlar. Bu aşamada mutasyon, yığındaki değişkenliği gerçekleştirerek arama uzayında yeni çözüm noktalarının elde edilmesini sağlamaktadır (Dengiz ve Altıparmak, 1998).

1°) Ters Çevirme

Ters çevirme mutasyonu bir alt dizi oluşturmak için bir kromozomda rastgele iki pozisyon seçer. Daha sonra bu alt dizi Şekil 5.4'de olduğu gibi iki ucu arasında ters çevrilir.

İlk hal	7	4	6	7	4	2	5	2	5	1
Değiştirilmiş hal	7	4	2	4	7	6	5	2	5	1

Şekil 5.4 Ters çevirme mutasyonu

2°) Ekleme

Ekleme rastgele bir parça seçer ve onu rastgele bir yere yerleştirir.

İlk hal	7	4	6	7	4	2	5	2	5	1
Değiştirilmiş hal	7	4	2	6	7	4	2	5	5	1

Şekil 5.5 Ekleme mutasyonu

3°) Yer Değişikliği Mutasyonu

Yer değişikliği mutasyonu rastgele bir alt dizi seçer ve onu Şekil 5.6'de görüldüğü gibi rastgele bir yere yerleştirir.

İlk hal	7	4	6	7	4	2	5	2	5	1
Değiştirilmiş hal	7	4	5	2	5	6	7	4	2	1

Şekil 5.6 Yer değiştirme mutasyonu

4°) Karşılıklı Değişim Mutasyonu

Karşılıklı değişim mutasyonu, rastgele iki gen seçer ve yerlerini değiştirir (Aksoy,2001).

İlk hal	7	4	6	7	4	2	5	2	5	1
Değiştirilmiş hal	7	2	5	2	5	6	7	4	2	1

Şekil 5.7 Karşılıklı değişim mutasyonu

5.4 Genetik Algoritma Parametreleri (Kontrol Prametreleri)

GA'lann işleyişi, arama uzayından daha önce örneklenmiş olan bölgelerin işletilmesi ve bu uzaydan yeni bölgelerin araştırılmasının dengeli birleşimi olarak düşünülebilir. GA'lann performansını kontrol eden bu denge, kontrol parametrelerinin doğru seçilmesi ile belirlenmektedir (Yeniay, 1999). Bu kontrol parametreleri:

5.4.1 Yığın Genişliği (Popülasyon büyüklüğü) (N)

Bu parametre, GA'lann hem etkinliğini hem de nihai performansını etkiler. GA'lar, küçük popülasyonlarda fazla etkili değildirler. Çünkü bu popülasyonlar çoğu hiperdüzlem (hyperplane) için yetersiz sayıda örnek sunar. Geniş bir popülasyonun ise, fazla sayıda hyperplane'lerden temsilciler içermesi daha muhtemeldir. Böylece GA'lar, daha detayh bir arama gerçekleştirebilirler. Sonuç olarak da, geniş bir popülasyon, optimal olmayan çözümlere yaklaşılmasını engeller. Diğer yandan büyük bir popülasyon, herbir jenerasyon için daha fazla değerlendirmeye ihtiyaç duyacağından daha yavaş ilerleme ihtimali vardır.

5.4.2 Çaprazlama Oranı (P_c)

Çaprazlama oranı, çaprazlama operatörünün uygulandığı sıklığı kontrol eder. Herbir yeni popülasyonda $P_c \cdot N$ tane yapı, çaprazlama işlemine tabi olur. P_c çaprazlama oranı ne kadar yüksek ise, popülasyona o kadar çabuk yeni yapılar girebilir. Eğer bu oran çok fazla yüksekse, yüksek performanslı yapılar, seçimin sunacağı iyileştirmelerden daha hızlı bir şekilde atılır. Eğer oran çok fazla düşük ise, o zaman da tarama işlemi, düşük araştırma oranından dolayı durgunlaşabilir.

5.4.3 Mutasyon Oranı (P_m)

Mutasyon oranı, kromozomlara ne oranda mutasyon uygulanacağını belirler. Mutasyon, popülasyonun değişkenliğini artıran ikincil bir tarama operatörüdür. Seçim işleminden sonra, yeni popülasyondaki her bir yapının bit pozisyonları (L, her bir yapıdaki bit pozisyonu sayısı), mutasyon oranına eşit bir olasılıkla rasgele bir değişime tabi olur. Sonuç olarak da, her bir jenerasyon için yaklaşık olarak $P_m \cdot N \cdot L$ mutasyonları meydana gelir. Düşük bir mutasyon seviyesi herhangi bir bit pozisyonun bütün popülasyon içerisinde tek bir değere yaklaşmasını engellerken, yüksek seviyedeki bir mutasyon oranı ise önemli bir rasgele arama imkanı sağlar.

5.4.4 Yığın Aralığı (Jenerasyon Boşluğu) (G)

Yığın genişliği, algoritmanın yakınsaması ile doğrudan ilişkilidir. Bu parametre, her

jenerasyon boyunca değiştirilecek popülasyon oranını kontrol eder. Şöyle ki; t. kuşaktaki popülasyon yani P(t)'nin N.(1-G) tane yapısı, (t+1). Kuşaktaki popülasyonda yani P(t+1) içinde eksiksiz olarak kalması için rasgele seçilir. G=1.0 değeri, bütün popülasyonun herbir jenerasyon boyunca değiştirileceği anlamına gelir. G=0,5 değeri ise her popülasyondaki yapıların yarısının bir sonraki jenerasyona aktarılacağı anlamına gelir.

5.4.5 Seçim Stratejisi (S)

Deneyler iki seçim stratejisini karşılaştırmıştır. Eğer S=P ise, sade bir seçim stratejisi; eğer S=E (E, çözüm uzayı (cevap yüzeyi)) ise, daha seçici bir strateji kullanılmıştır. İlk olarak uygulanan seçim stratejisinden sonra daha seçici strateji, en iyi performansa sahip yapının bir sonraki jenerasyona (nesil) eksiksiz olarak geçmesini şart koşar. Böyle bir stratejinin olmadığı durumlarda örnekleme hatası, çaprazlama veya mutasyondan dolayı en iyi yapının kaybolması muhtemeldir.

5.4.6 Ölçeklendirme Fonksiyonu (Ölçekleme Penceresi) (W)

GA'nın çalışması süresince yığındaki değişkenlerin korunması önemlidir. Özellikle orantılı seçim mekanizmalarında aramanın etkin bir şekilde yürütülmesi zorlaşmaktadır. Algoritmanın ilk birkaç çevrimi sonunda elde edilen yığında uygunluk değeri yüksek birkaç dizi bulunabilmektedir. Bu dizilerin seçim olasılıkları yüksek olduğundan arama bu diziler etrafında yoğunlaşır. Bu olay algoritmanın zamansız yakınsamasına sebep olmaktadır. Diğer taraftan, algoritmanın son çevrimlerinde yığındaki dizilerin uygunluk değerleri birbirine yaklaşmaktadır. Dizilerin seçim olasılıkları da birbirine çok yakın olacağı için yeni yığınlarda iyi dizilerin korunması zorlaşmaktadır. Bu durum aramanın etkinliğini azaltmaktadır. Orantılı seçim mekanizmalarında bu iki sorunu ortadan kaldırabilmek için, uygunluk değerlerinde bir düzenlemenin yapılması gerekmektedir. Literatürde bunu gerçekleştirebilmek amacıyla ölçeklendirme fonksiyonları geliştirilmiştir. En çok bilinen ölçeklendirme fonksiyonları; doğrusal ölçeklendirme, standart sapma kadar azaltma ve üs yaklaşımıdır.

Bir GA ile nümerik bir $f(x)$ fonksiyonu maksimize edildiğinde, bir x yapısının $u(x)$ performans değerini $u(x) = f(x) - f_{min}$ olarak tanımlamak oldukça yaygındır. Burada f_{min}, belli bir araştırma uzayında f(x) in kabul edebileceği minimum değerdir. Bu dönüşüm f(x) in karakteristiğine bakılmaksızın u(x) performansının pozitif olmasını garantiler. Sıklıkla f_{min} başlangıçta mevcut değildir. Bu durumda $u(x) = f(x)-f(x_{min})$ olarak tanımlamak mantıklıdır. Burada $f(x_{min})$ şimdiye kadar değerlendirilmiş herhangi yapının minimum değeridir. $u(x)$ 'in hiçbir tanımı x'in iyi değerlerinin ayırt edilmesini etkilemez, örneğin, $f_{min} = 0$ olsun. Birkaç jenerasyon sonra,

mevcut popülasyon, yalnızca $105<f(x)<110$ olan x yapılarını içerebilir. Bu noktada, popülasyondaki hiçbir yapının ortalamadan fazla sapan bir performansı olmayacaktır. Bu da, seçim baskısını daha iyi yapılara doğru azaltır. Bir çözüm yeni bir $f'_{\min}$ parametresini, örneğin 100 değeri ile tanımlamaktır ve her bir yapıda oran bir standarda karşıdır. Örneğin $f(x_i)=110$ ve $f(x_j)=105$ $u(x_i)=f(x_i)-f'_{\min}=10$ ve $u(x_j)=f(x_j)-f'_{\min}=5$ dir; şimdi x_i nin performansı x_j nin performansının iki katı kadar iyi olarak gözükmektedir.

Deneyimlerimiz, 3 ölçekleme modunu, Ölçekleme Penceresi (W) denen bir parametreye dayanarak araştırmıştır. Eğer W=0 ise ölçekleme aşağıdaki gibi oluşturulmuştur: $f'_{\min}$ 1. iterasyonda minimum $f(x)$ olarak alınmıştır. Sonraki her bir jenerasyonda değerlendirmeleri $f'_{\min}$'den daha küçük olan böyle yapılar, seçim prosedüründen ihmal edilmiştir. $f'_{\min}$ belli bir popülasyondaki bütün yapılardan $f'_{\min}$ daha büyük değerlendirmelere her sahip olduğunda genelleştirilmiştir. Eğer $0<W<7$ ise, son W jenerasyonlarında oluşan en küçük $f(x)$ değerini,alır. $f'_{\min}$ W=7 değeri bir sonsuz pencere yani hiçbir ölçeklemenin yapılmadığını gösterir (Grefenstette, 1986, s. 122-128).

6. MATLAB® GENETİK ALGORİTMA ve DOĞRUDAN ARAMA ARACI*

Genetik Algoritma ve Doğrudan Arama Aracı, (Genetic Algorithm and Direct Search Toolbox) Optimizasyon Araçı (Optimization Toolbox) ve MATLAB® nümerik hesaplama ortamının gelişmiş fonksiyonlarının toplandığı bir uygulamadır. Genetik Algoritma ve Direk Arama Aracı, optimizasyon problemlerini çözmek için şu yöntemleri kıllanır,

- Genetik algoritma
- Doğrudan arama

Bu algoritmalar standart optimizasyon aracının kapsamı dışında kalan alanlarda optimizasyon problemlerinin farklı bir şekilde çözülmesini sağlar.

Genetik algoritma ve doğrudan arama arcının kapasitesini kendi M-dosyanızı yazarak, diğer araçlar ile birleştirerek ve MATLAB veya Simulink® kullanarak geliştirilebilir.

6.1 Optimize Etmek İstediğimiz Fonksiyon İçin M-dosyası Yazılması

Genetik Algoritma ve Direk Arama Aracını kullanmak için öncelikle optimize etmek istediğimiz fonksiyonun M-dosyasını yazmamız gerekmektedir. M-dosyası amaç fonksiyonu için bağımsız değişkenleri içermelidir. Bu bölümde M-dosyasının nasıl yazılacağı hakkında bilgi verilecektir.

6.1.1 M-dosyasının Yazılması

Aşağıdaki örnek optimize etmek istediğimiz fonksiyon için M-dosyasını nasıl yazacağımızı göstermektedir. Optimize etmek istediğimiz fonksiyonun

$$f(x_1, x_2) = x_1^2 - 2x_1x_2 + 6x_1 + x_2^2 - 6x_2 \tag{6.1}$$

olduğunu kabul edelim.

M-dosyası hesaplamak istediğimiz x_1 ve x_2 değişkenlerine karşılık gelen iki elemanlı x satır vektörü olarak kabul eder ve x in fonksiyonu olarak skaler bir büyüklü hesaplar. M-dosyası yazmak için şu adımları takip etmeliyiz.

1. MATLAB "File" menüsünden "New" seçilir.

---

* The Genetic Algorithm and Direct Search Toolbox. Bu bölümde sadece genetik algoritma aracı incelenmiştir. Bu bölümde (The MathWorks, 2005) kaynağından yararlanılmıştır.

2. M-File seçilir. Bu yeni bir M-dosyasını editörde açar.

3. M-dosyasına aşağıdaki kod satırları yazılır.

```
function z = my_fun(x)
z = x(1)^2 – 2*x(1)*x(2) + 6*x(1) + x(2)^2 – 6*x(2) ;
```

4. M-dosyasını MATLAB 'in altındaki bir klasöre kaydedin.

M-dosyasının doğru değerleri verdiğini kontrol etmek için,

```
my_fun([2 3])

ans =  -5
```

6.2 Genetik Algoritma Aracının Kullanılışı

Genetik algoritma aracı iki yolla kullanılabilir.

- Komut satırından (Command Line) genetik algoritma ga fonksiyonun çağrılması.
- Genetik Algoritma aracı için grafik ara yüzünün kullanılması.

Bu bölümde bu metotlara bir giriş yapılacaktır.

6.2.1 Komut Satırından Genetik Algoritma ga Fonksiyonun Çağrılması

Genetik algoritmayı komut satırından kullanmak için, aşağıdaki ifadeyi yazarak çağırabiliriz.

```
[x fval] = ga(@fitnessfun, nvars, options)
```

- @fitnessfun, uygunluk denklemini (fitness function) ifade eder.
- nvars, uygunluk denklemindeki bağımsız değişken sayısı.
- options, genetik algoritma ile ilgili seçenekler için kullanılır. Belirtilmez ise ga geçerli ayarları kullanır.

Sonuçlar şu şekilde verilir,

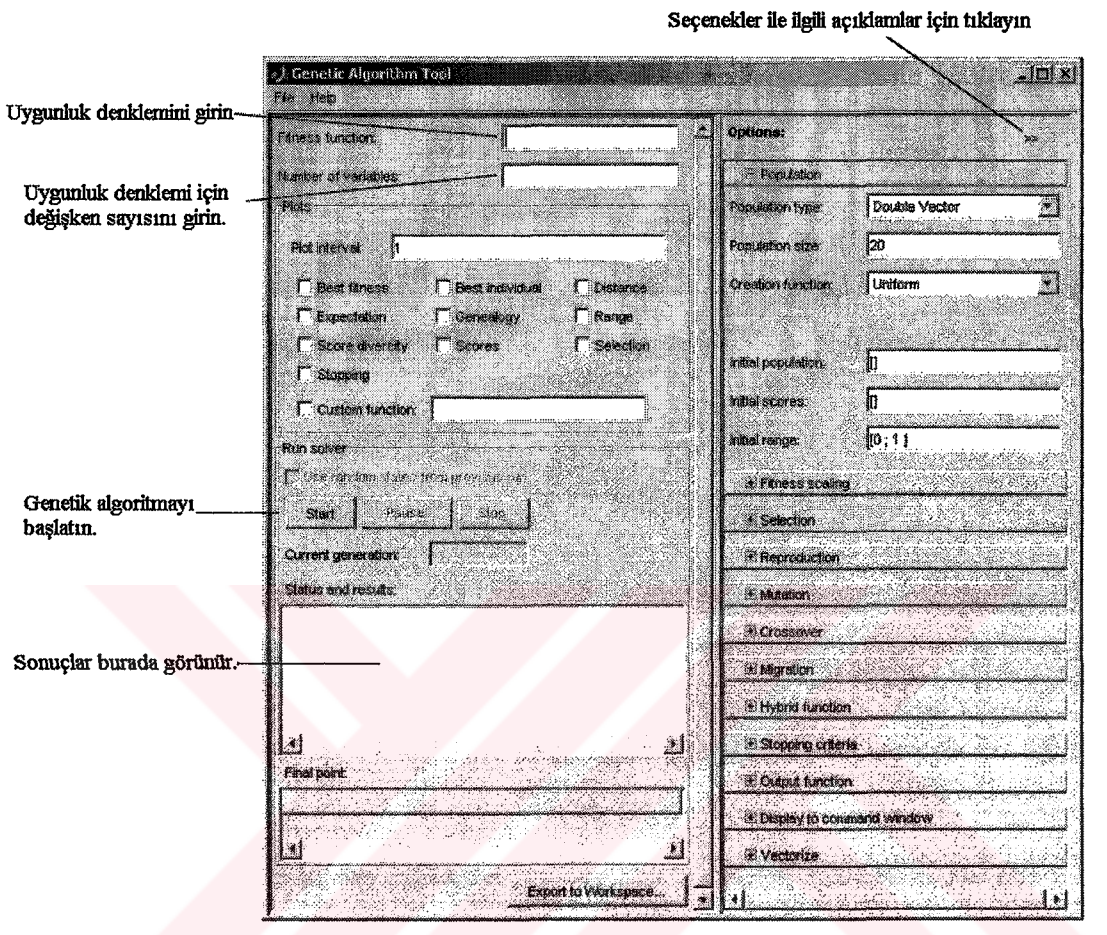
- fval — uygunluk denkleminin son değeri,
- x — son değeri oluşturan noktayı belirtir.

6.2.2 Grafik Ara Yüzünden Kullanılması

Genetik algoritma aracı, genetik algoritmayı komut satırına gerek kalmadan kullanıcı ara yüzü ile kullanmamızı sağlar. Genetik Algoritma Aracını açmak için,

gatool

komutu girilir. Bu aşağıdaki ekranı açar.



Şekil 6.1 Genetik algoritma aracı grafik ara yüzü.

Genetik algoritma aracını kullanmak için öncelikle aşağıdaki bilgiler girilmelidir:

- Fitness function — Minimize etmek istediğimiz amaç/hedef fonksiyonu (objective function). Daha önceden verilen uygunluk denklemi için M-dosyası yazma örneğinde verildiği gibi optimize etmek istediğimiz fonksiyon yazılır. Burada dikkat edilmesi gereken nokta uygunluk denkleminin yazıldığı dosya ismi ile ara yüze yazılan ismin aynı olmasıdır. Örneğin, dosya ismi "fitnessfun.m" ise buraya "@fitnessfun" yazılmalıdır.
- Number of variables — Uygunluk denklemindeki giriş vektörünün derecesi. Denklem (8.1)'i göz önüne alırsak girilmesi gereken değer iki (2) olur.

Genetik algoritmayı çalıştırmak için "Start" tuşuna basılır. Sonuçlar "Status and Results" bölümünde görünür.

Genetik algoritma ile ilgili seçenekler "Options" kısmından değiştirilebilir. "Options" bölümündeki özellikleri değiştirmek için her özelliğin başındaki + işaretine basılarak onunla ilgili seçenekler görülebilir.

6.3 Örnek: Rastrigin Fonksiyonu

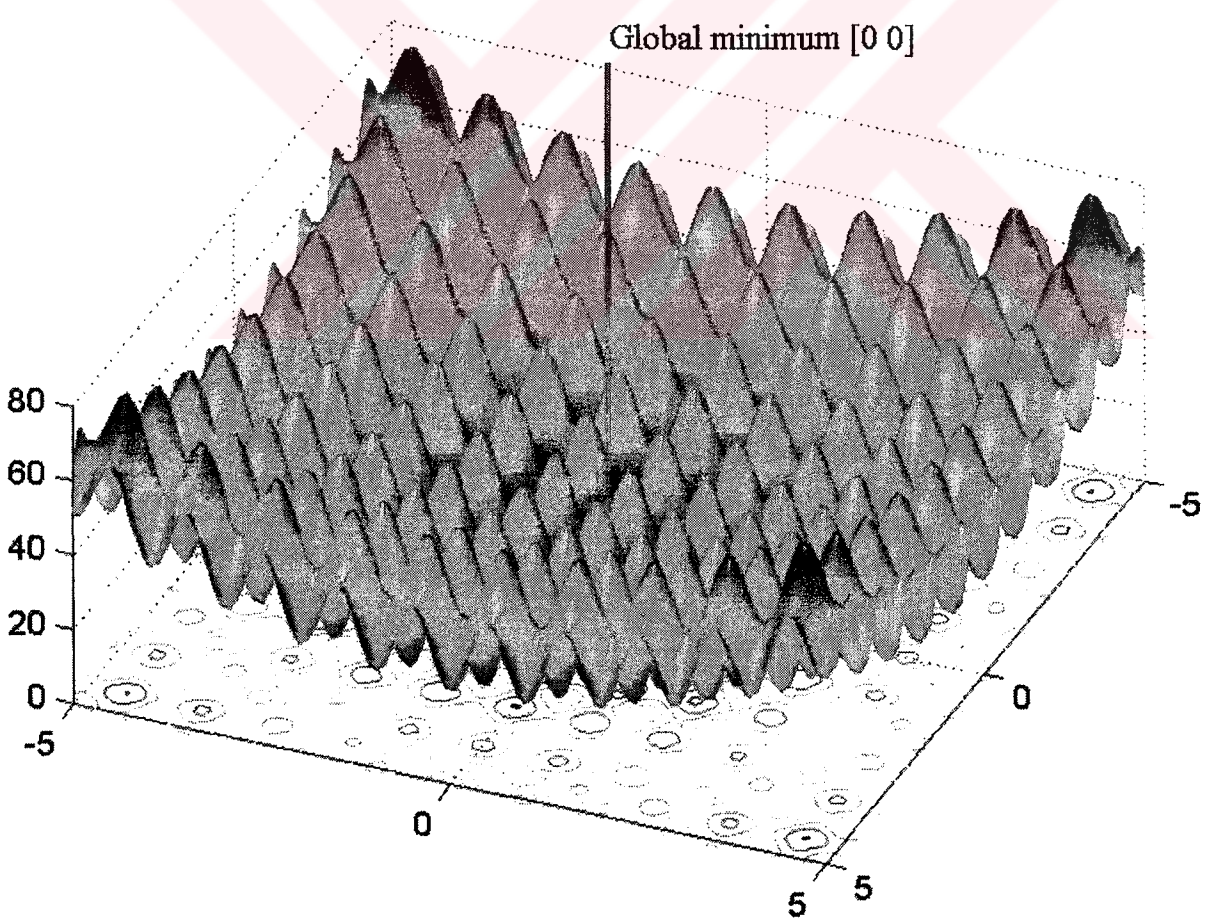
Bu bölümde genetik algoritma için sıklıkla test fonksiyonu olarak kullanılan Rastrigin fonksiyonu örnek olarak gösterilecektir.

6.3.1 Rastrigin Fonksiyonu

İki bağımsız değişken için Rastrigin fonksiyonu şu şekilde tanımlanır,

$$Ras(x) = 20 + x_1^2 + x_2^2 - 10(\cos 2\pi x_1 + \cos 2\pi x_2) \tag{6.2}$$

MATLAB araçlarında Rastrigin fonksiyonunu hesaplamak için "rastriginsfcn.m" dosyası vardır. Aşağıdaki şekilde Rastrigin fonksiyonunun çizimi gösterilmiştir.

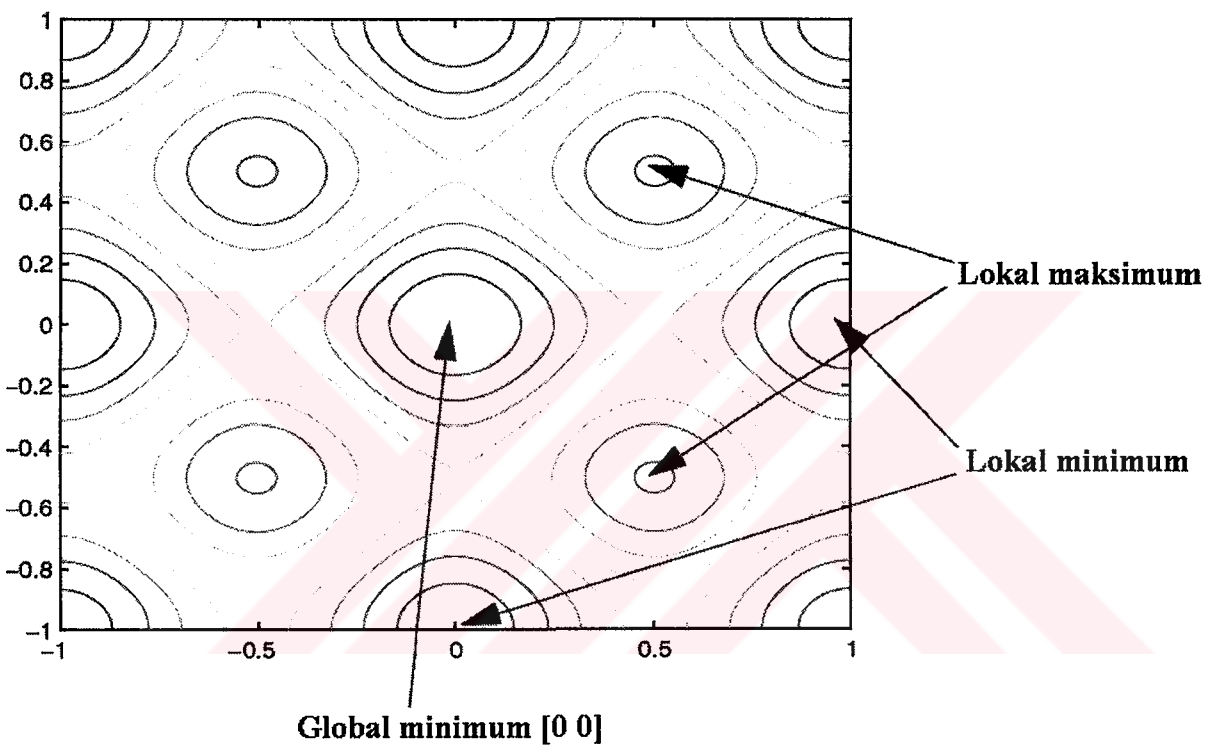


Şekil 6.2 Rastrigin fonksiyonu

Görüldüğü gibi Rastrigin fonksiyonu çizdirilen bölgede birçok yerel minimum noktası mevcuttur. Bununla beraber bu fonksiyonun sadece bir global minimum noktası [0 0] vardır. Fonksiyon x-y düzleminde [0 0] değerini aldığında fonksiyonda sıfır (0) değerini alır.

Rastrigin fonksiyonu sıklıkla genetik algoritma testi için çünkü birçok yerel minimum noktasına sahip olduğundan dolayı standart metotlar için çözmesi zor bir problemdir.

Aşağıdaki çizimde Rastrigin fonksiyonunun farklı maksimum ve minimum noktaları gösterilmiştir.



Şekil 6.3 Rastrigin fonksiyonunun farklı min. ve maks. noktaları.

6.3.2 Rastrigin Fonksiyonunun Minimumunun Bulunması

Bu bölümde Rastrigin fonksiyonunun minimum değerinin bulunması anlatılmaktadır.

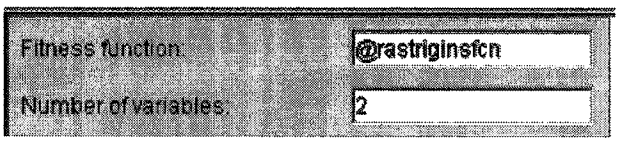
Rastrigin fonksiyonunun minimumunu bulmak için aşağıdaki adımlar izlenir.

1. Genetik algoritma aracını açmak için komut satırına "gatool" girilir.

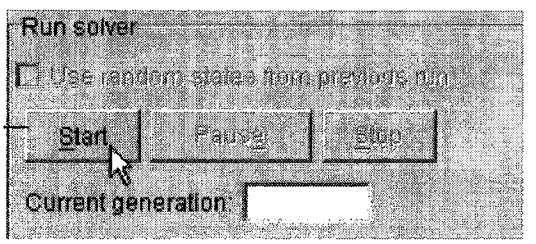
2. açılan pencereye aşağıdaki bilgiler girilir.

- "Fitness function" alanına "@rastriginfcn" girilir.
- "Number of variables" alanına, Rastrigin fonksiyonunun bağımsız değişken sayısı olan "2"

girilir.

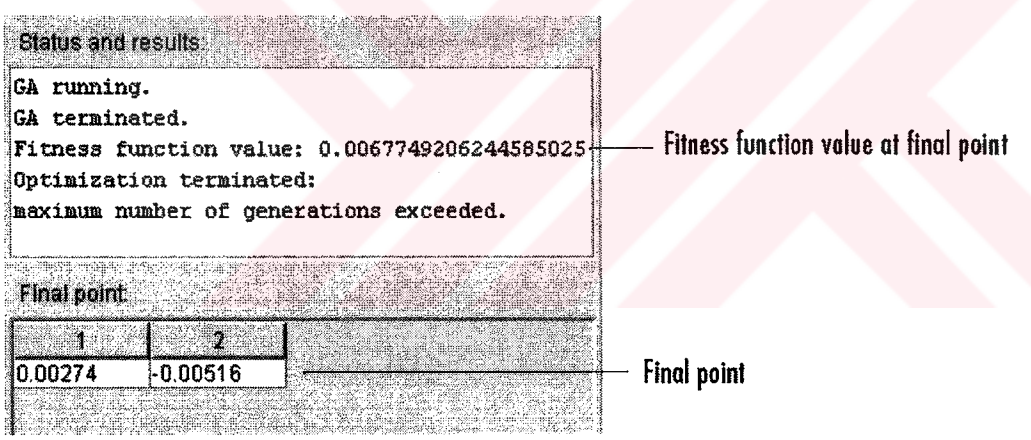


3. Çözümü başlatmak için "Start" tuşuna basılır.



Algoritma çalışırken "current generation" alanında o anki jenerasyon görünür. Algoritmayı "Pause" tuşuna basarak geçici olarak durdura biliriz.

Algoritma tamamlandığında sonuçlar "Status and results" kısmında görünür.



Bu pencerede şu sonuçlar görünür.

- Uygunluk denkleminin algoritmanın sonlandığındaki değeri.
- `Function value: 0.0067749206244585025`
- Bu değer Rastrigin fonksiyonunun minimum değerine çok yakındır.
- Algoritmanın sonlandırılma sebebi.

```
Exit: Optimization terminated:
maximum number of generations exceeded.
```

Bu örnekte maksimum jenerasyon sayısı 100 olarak belirlendiği için bu sayıya ulaşılınca algoritma durmuştur.

- Hesaplama için kullanılan son nokta.
- `[0.00274 -0.00516]`

6.3.3 Minimum Değeri Komut Satırından Bulma

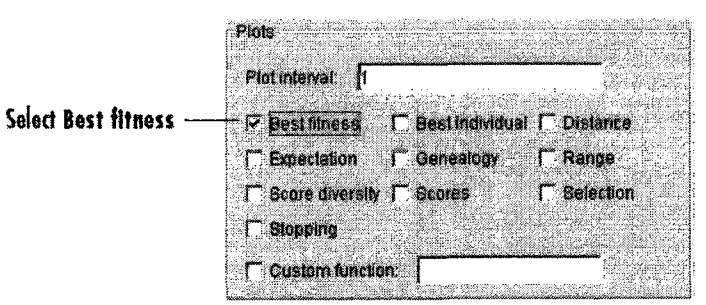
- Rastrigin fonksiyonunun minimum değerini komut satırından bulmak için,
- [x fval reason] = ga (@rastriginfcn, 2)
- girilir ve aşağıdaki sonuç elde edilir.

```
[x fval reason] = ga(@rastriginsfcn, 2)

x =

    0.0027   -0.0052


fval =

    0.0068


reason =

Optimization terminated:
maximum number of generations exceeded.
```

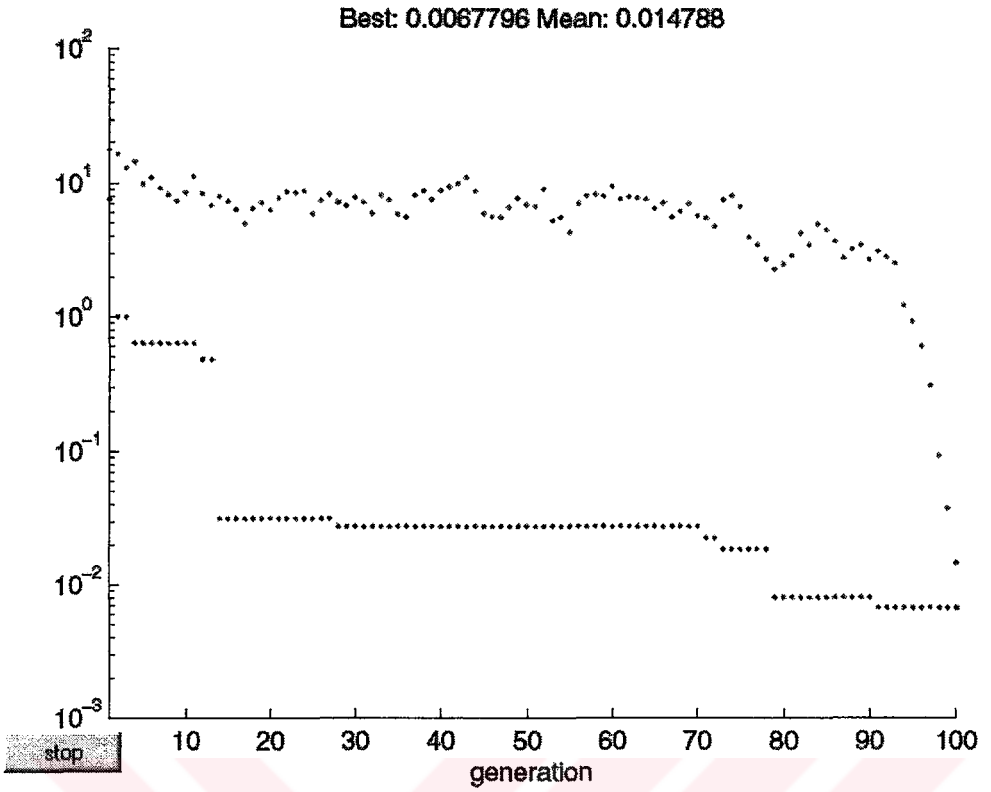
- "x", hesaplanan son nokta.
- "fval", uygunluk denkleminin son noktadaki değeri.
- "reason", algoritmanın sonlandırılma sebebi.

6.3.4 Grafiklerin Oluşturulması

"Plot" bölümü genetik algoritma çalışırken algoritma hakkındaki bilgileri grafik olarak görmemizi sağlar. Bu bilgiler algoritmanın performansını artırmak için yapacağımız ayarlarda bize yardımcı olur.



"Best fitnees" işaretlenirse Genetik Algoritma Aracı uygunluk denklemini en iyi ve ortalama değerlerini her bir jenerasyon için grafikte gösterir.

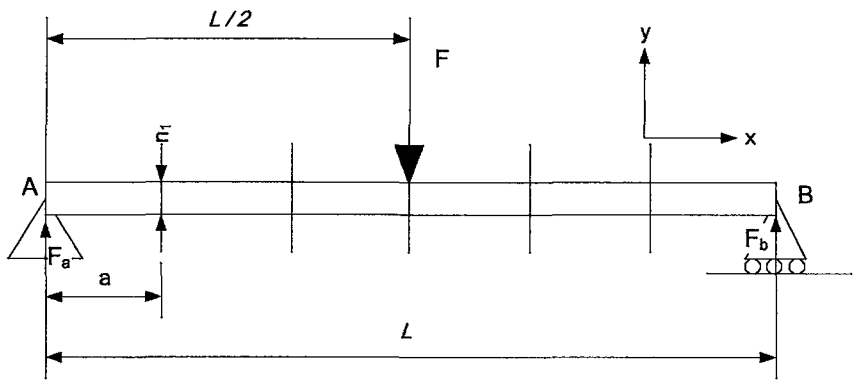


Şekil 6.4 Uygunluk denkleminin her jenerasyondaki en iyi ve ortalama değerleri

7. MATLAB® PROGRAMINDA UYGULAMA

7.1 Uygulama 1: Basit Mesnetli Bir Kirişin Optimize Edilmesi

Bu uygulamada, orta noktasından noktasal yük uygulanmış basit mesnetli bir kirişin belirlenen bir gerilme değerinde kesitlerinin eş mukavemetli olarak optimum kesit alanları bulunmaya çalışılacaktır.



Şekil 7.1 Uygulama 1: Basit mesnetli kiriş

Şekil 7.1'de görüldüğü gibi optimize edilmek istenen kesitler "A" noktasından başlayarak eşit aralıklar ile altı (6) parçaya ayrılmıştır.

$$a = L/6 \tag{7.1}$$

$L/2$ noktasından F noktasal yük uygulanmıştır. Mesnet kuvvetleri F_a ve F_b olarak ifade edilmiştir. Emniyetli gerilme 200 MPa, kiriş genişliği 0.2 m, kiriş boyu 4 m, noktasal yük 10.000 N olarak alınmıştır.

Çizelge 7.1 Problemin verileri

F	10.000 N
L	4 m
m	0.2 m
σ_e	200 MPa

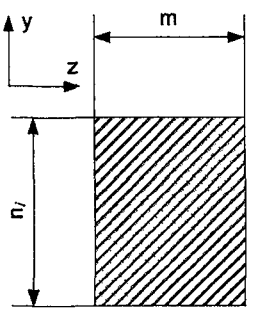
Bu problem hesaplaması kolay olmak ile beraber analitik olarak hesaplanarak sonuçları

karşılaştırmak için uygundur.

Öncelikle seçtiğimiz kesitlerin belirlen gerilmeyi sağlayacak ölçülerini bularak başlayalım.

```
% Kiriş kesitlerini eş mukavemetli olması için gerekli "n" değerini
% hesaplar
F=10000;      % (N), noktasal yük
L=4;          % (m), kirişin boyu
i=6;          % Kirişin bölüneceği parça sayısı (i-1): değişken sayısı
m=0.2;        % (m), kiriş kesitinin genişliği
sigma_e=200e6; % (Pa), emniyet gerilmesi
Lf=L/2;       % (m), kuvvetin A mesnetine olan uzaklığı
Fa=(F*(L-Lf)/L);
Fb=F-Fa;
a=L/i;
M(1)=a*Fa;
M(2)=2*a*Fa;
M(3)=3*a*Fa;
M(4)=(4*a*Fa)-(a*F);
M(5)=(5*a*Fa)-(2*a*F);
k=1:5;
n(k)=sqrt((M(k)*6)/(m*sigma_e))
```

Bu programı MATLAB'te çalıştırdığımızda bize seçtiğimiz 5 kesitte 200 MPa gerilme oluşması için gerekli "n_i" ölçülerini verecektir.



Şekil 7.2 Kirişin kesiti

```
n =

   0.0224   0.0316   0.0387   0.0316   0.0224
```

Alınan çıktılar sırasıyla n_1, n_2, n_3, n_4 ve n_5 i ifade etmektedir. Aşağıdaki program kirişin optimizasyonu için kullanılmak üzere genetik algoritma aracına uygun olarak yazılmıştır.

```
% Kiriş boyut optimizasyonu.
% Bir ucun dönel, bir ucu kayar mesnetli, ortasından noktasal yüklenmiş.
%
function sigma = beam_opt(n)
F=10000;        % (N), noktasal yük
L=4;            % (m), kirişin boyu
i=6;            % Kirişin bölüneceği parça sayısı (i-1): değişken sayısı
m=0.2;          % (m), kiriş kesitinin genişliği
sigma_e=200e6;  % (Pa), emniyet gerilmesi
%
Lf=L/2;         % (m), kuvvetin A mesnetine olan uzaklığı
Fa=(F*(L-Lf)/L);
Fb=F-Fa;
a=L/i;
M(1)=0*Fa;
M(2)=a*Fa;
M(3)=2*a*Fa;
M(4)=3*a*Fa;
M(5)=(4*a*Fa)-(a*F);
M(6)=(5*a*Fa)-(2*a*F);
M(7)=(6*a*Fa)-(3*a*F);
sigma=abs((M(2)*6/(m*n(1)^2))-sigma_e)+abs((M(3)*6/(m*n(2)^2))-sigma_e)+...
   abs((M(4)*6/(m*n(3)^2))-sigma_e)+abs((M(5)*6/(m*n(4)^2))-sigma_e)+...
   abs((M(6)*6/(m*n(5)^2))-sigma_e);
```

Daha önceki bölümde anlatıldığı gibi MATLAB genetik algoritma aracının kullanabileceği formda problem MATLAB'e aktarılır. Programın son bölümünde yazılmış olan "sigma"

kesitlerdeki gerilmelerin toplamını ifade eder ve genetik algoritmada uygunluk değeri olarak kullanılacaktır. Programın girdileri n_i değerleri, çıktı olarak uygunluk değeri (sigma) olarak yazılan bu program "beam_opt.m" adında kaydedilir. MATLAB genetik algoritma aracındaki "Fitness function" bölümüne "@beam_opt" yazılmalıdır. Değişken sayısı incelemek istediğimiz kesit sayısı olan 5 olarak girilir.

Diğer ayarlar;

- Population

Population size: 100

Initial range: [0 ; 0.5]

- Stopping criteria

Generations: inf

Fitness limit: 1e6

Stall generation: inf

Stall time limit: inf

Diğer ayarlar geçerli ayarlarında bırakılmıştır.

Yukarıdaki ayarları yaptıktan sonra "Start" a basılarak program çalıştırılır.

Program bizim belirlediğimiz uygunluk değerinin altına indiğinde kendiliğinden duracak ve sonuçları vericektir.

GA terminated.

Fitness function value: 581820.0390658677

Optimization terminated: minimum fitness limit reached.

Yukarıda belirlediğimiz uygunluk limiti $1x10^6$ idi, program yaklaşık $0.5818x10^6$ değerine geldiğinden algoritmayı durdurdu.

[0.022376, 0.031624, 0.038718, 0.031632, 0.022358]

Algoritmanın bulduğu optimum n değerleri yukarıdaki gibidir.

Bulunan bu değerleri aşağıdaki tabloda karşılaştıralım.

Çizelge 7.2 Bulunan sonuçların karşılaştırılması

Çözüm	n_1	n_2	n_3	n_4	n_5
Analitik	0.0224	0.0316	0.0387	0.0316	0.0224
Genetik algoritma	0.022376	0.031624	0.038718	0.031632	0.022358

Görüldüğü gibi sonuçlar birbirine çok yakın çıkmaktadır. Genetik algoritma ile yapılan optimizasyonda bulunan sonuçları kullanarak kesitlerdeki gerilmeleri hesaplamak için,

```
% Genetik Algoritma ile yapılan optimizasyonda bulunan kesitlere karşılık
% gelen gerilme değerlerini hesaplar.
F=10000;        % (N), noktasal yük
L=4;            % (m), kirişin boyu
i=6;            % Kirişin bölüneceği parça sayısı (i-1): değişken sayısı
m=0.2;          % (m), kiriş kesitinin genişliği
sigma_e=200e6;  % (Pa), emniyet gerilmesi
Lf=L/2;         % (m), kuvvetin A mesnetine olan uzaklığı
Fa=(F*(L-Lf)/L);
Fb=F-Fa;
a=L/i;
M(1)=a*Fa;
M(2)=2*a*Fa;
M(3)=3*a*Fa;
M(4)=(4*a*Fa)-(a*F);
M(5)=(5*a*Fa)-(2*a*F);
for k=1:5;
   sigma(k)=(M(k)*6)/(m*garesults.x(k)^2);
end
stresses=[sigma(1); sigma(2); sigma(3);sigma(4); sigma(5)]
```

bu program kullanıla bilir.

Bu programın sonuçları,

```
stresses =
  1.0e+008 *
    1.9973
    1.9998
    2.0012
    1.9988
    2.0005
```

Yukarıda genetik algoritmada bulunan sonuçlar ile hesaplanan kesit gerilmeleri verilen 200 MPa değerine çok yaklaşmıştır.

Genetik algoritma komut satırından veya MATLAB M-dosyasından da çalıştırılabilir. Bunun için aşağıdaki program kullanılmalıdır.

```
options = gaoptimset('Generations', 300,'FitnessLimit', 1e6,...
    'StallGenLimit', 100,'StallTimeLimit', inf,'PopInitRange', [0;0.5],...
    'PopulationSize', 300,'CrossoverFraction', 0.79,...
    'Plotfcns', @gaplotbestf, 'PlotInterval', 5, 'EliteCount', 25);
%
[x fval reason output] = ga(@beam_opt, 5, options)
%
rand('state', output.randstate);
randn('state', output.randnstate);
```

Bu programdan alınan çıkşlar,

```
Optimization terminated: maximum number of generations exceeded.
x =
   -0.0224    0.0316   -0.0388    0.0316    0.0224


fval =
   2.2985e+006


reason =
Optimization terminated: maximum number of generations exceeded.
```

output =

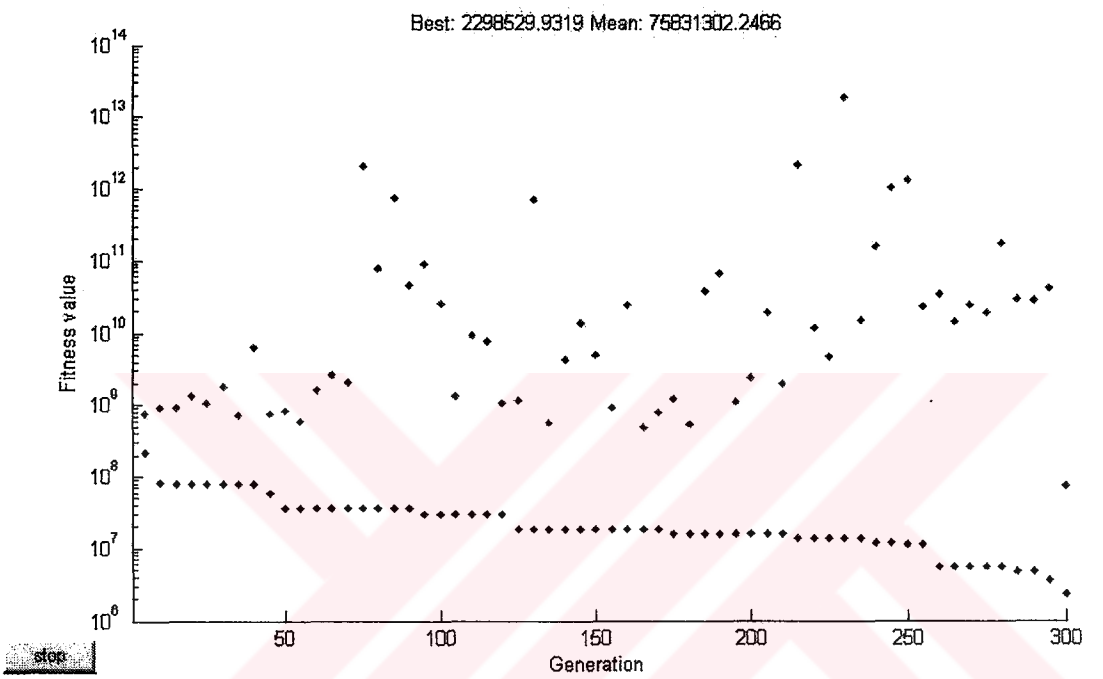
randstate: [35x1 double]

randnstate: [2x1 double]

generations: 300

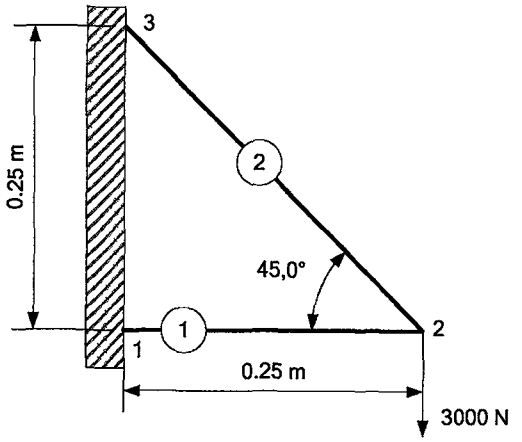
funccount: 90000

message: 'Optimization terminated: maximum number of generations exceeded.'



Şekil 7.3 Her bir nesildeki ortalama ve en iyi uyum değeri

7.2 Uygulama 2: İki Elemanlı Kafesin Optimizasyonu



Şekil 7.4 İki elemanlı kafes kiriş

Yukarıda şekli verilen problem iki elemandan oluşan basit kafes kirişin optimizasyonu yapılmaya çalışılacaktır. Burada tasarım değişkenlerimiz 1 ve 2 nolu elemanların kesit alanlarıdır. 2 nolu bağlantı noktasından 3000 N noktasal yük uygulanmıştır. 1 ve 3 nolu bağlantı noktaları x ve y düzleminde hareketi kısıtlamış sadece z ekseni etrafında dönme serbestliği verilmiştir. Gerilmeler MATLAB programında sonlu elemanlar yöntemi ile hesaplanmıştır (Kwon).

```
%To solve static 2-D truss structure
function stress_total = truss_opt(a)
nel=2;
nnel=2;
ndof=2;
nnode=3;
sdof=nnode*ndof;
gcoord(1,1)=0.0;
gcoord(1,2)=0.0;
gcoord(2,1)=.25;
gcoord(2,2)=0.0;
gcoord(3,1)=0.0;
gcoord(3,2)=.25;
elprop(1,1)=200e9; %Pa
```

```
elprop(1,2)=abs(a(1)); %m^2
elprop(2,1)=200e9; %Pa
elprop(2,2)=abs(a(2)); %m^2
nodes(1,1)=1; nodes(1,2)=2;
nodes(2,1)=2; nodes(2,2)=3;
bcdof(1)=1;
bcval(1)=0;
bcdof(2)=2;
bcval(2)=0;
bcdof(3)=5;
bcval(3)=0;
bcdof(4)=6;
bcval(4)=0;
ff=zeros(sdof,1);
kk=zeros(sdof,sdof);
index=zeros(nnel*ndof,1);
elforce=zeros(nnel*ndof,1);
eldisp=zeros(nnel*ndof,nnel*ndof);
k=zeros(nnel*ndof,nnel*ndof);
stress=zeros(nel,1);
ff(4)=-3000;
for iel=1:nel;
nd(1)=nodes(iel,1);
nd(2)=nodes(iel,2);
x1=gcoord(nd(1),1); y1=gcoord(nd(1),2);
x2=gcoord(nd(2),1); y2=gcoord(nd(2),2);
leng=sqrt((x2-x1)^2+(y2-y1)^2);
if (x2-x1)==0;
   beta=2*atan(1);
else
```

```
   beta=atan((y2-y1)/(x2-x1));
end
el=elprop(iel,1);
area=elprop(iel,2);
index=feeldof(nd,nnel,ndof);
k=fetruss2(el,leng,area,0,beta,1);
kk=feasmbl1(kk,k,index);
end
[kk,ff]=feaplyc2(kk,ff,bcdof,bcval);
disp=kk\ff;
for iel=1:nel;
nd(1)=nodes(iel,1);
nd(2)=nodes(iel,2);
x1=gcoord(nd(1),1); y1=gcoord(nd(1),2);
x2=gcoord(nd(2),1); y2=gcoord(nd(2),2);
leng=sqrt((x2-x1)^2+(y2-y1)^2);
if (x2-x1)==0;
   beta=2*atan(1);
else
   beta=atan((y2-y1)/(x2-x1));
end
el=elprop(iel,1);
area=elprop(iel,2);
index=feeldof(nd,nnel,ndof);
k=fetruss2(el,leng,area,0,beta,1);
for i=1:(nnel*ndof);
   eldisp(i)=disp(index(i));
end
elforce=k*eldisp;
stress(iel)=sqrt(elforce(1)^2+elforce(2)^2)/area;
```

```
if ((x2-x1)*elforce(3))<0;
   stress(iel)=-stress(iel);
end
end
num=1:1:sdof;
displ=[num' disp];
numm=1:1:nel;
stresses=[numm' stress];
stress_max=200e6;
%stress_total=abs(stress(2)-stress_max);
%stress_total=abs(stress(2)-stress_max)+abs(abs(stress(1))-stress_max);
stress_total=abs(stress_max-abs(stress(1)))+abs(abs(stress_max-stress(2)));
%k=3:4;
%displacement=[disp(k)];
```

Yukarıdaki programı genetik algoritma aracında çalıştırmak için aşağıdaki M-dosyası yazılır.

```
options = gaoptimset('Generations', inf,'FitnessLimit', 1e6,...
   'StallGenLimit', 100,'StallTimeLimit', inf,'PopInitRange', [0;0.0005],...
   'PopulationSize', 200,'CrossoverFraction', 0.9,...
   'Plotfcns', @gaplotbestf, 'PlotInterval', 10, 'EliteCount', 5);
[x fval reason output] = ga(@truss_opt, 2, options)
rand('state', output.randstate);
randn('state', output.randnstate);
```

İlk dört satırda genetik algoritmanın kontrol parametreleri girilmektedir.

- Population

Population size: 200

Initial range: [0 ; 0.0005]

- Stopping criteria

Generations: inf

Fitness limit: 1e6

Stall generation: 100

Stall time limit: inf

- Reproduction

Elite Count: 5

Crossover Fraction: 0.9

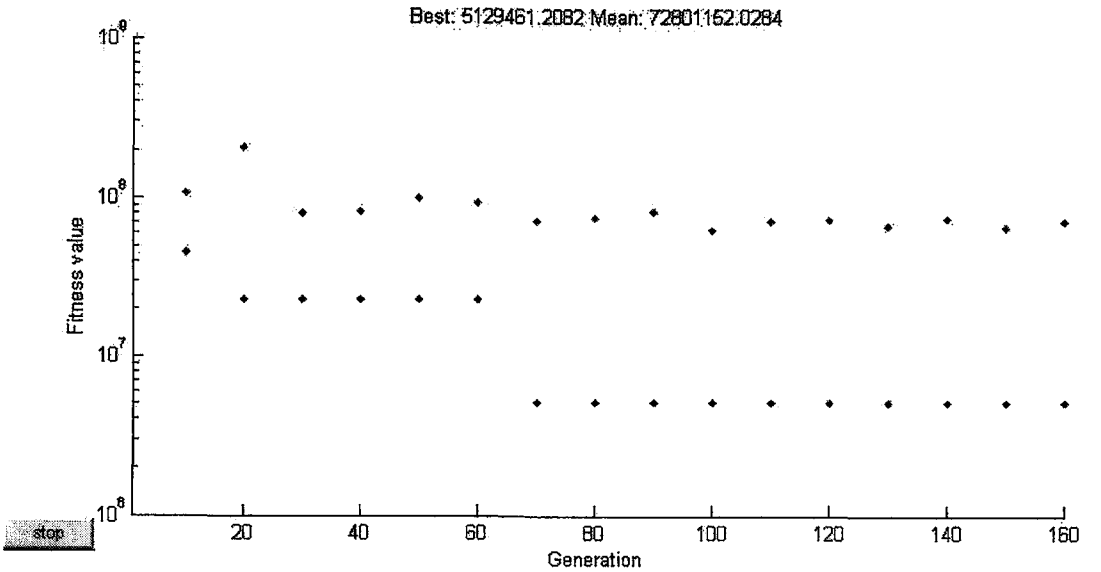
- Plot

PlotFcns: @gaplotbestf

Ayarları yapılır ve diğer ayarların geçerli değerleri kullanılarak çözüm elde edilir.

```
Optimization terminated: stall generations limit exceeded.
x =
  1.0e-004 *
    0.1479    0.2146

fval =
  5.1295e+006

reason =
Optimization terminated: stall generations limit exceeded.

output =
     randstate: [35x1 double]
    randnstate: [2x1 double]
   generations: 164
     funccount: 32800
       message: 'Optimization terminated: stall generations limit exceeded.'
```

Sonuçlar kısmında bulunan x değerleri aramkta olduğumuz sırasıyla 1 ve 2 çubukların kesit alanlarıdır. İkinci sıradaki "fval" uygunluk değeri verdiğimiz uygunluk fonksiyonunun son nesildeki en iyi değeridir. Daha sonraki bilgiler GA'nın çalışması ve neden sonlandırıldığı hakkında bilgi verir.



Şekil 7.5 Herbir nesildeki ortalama ve en iyi uyum değerleri

Genetik algoritmada çözümler yapıldıktan sonrada çıkan sonuçlar yine bir program yardımı ile elemanlardaki gerilmeler hesaplanırsa,

```
displ =
   1.0000    0.0000
   2.0000         0
   3.0000   -0.0003
   4.0000   -0.0007
   5.0000         0
   6.0000         0


stresses =
  1.0e+008 *
   0.0000   -2.0285
   0.0000    1.9772
```

Sonuçları elde edilir. İlk olarak görülen deperler nodeların metre cinsinden yerdeğitirmeleridir. 1,2 ve 5,6 nolu yer değiştirmeler 1. ve 3. node yerdeğiştirne serbestliği vermediğimiz için sıfır çıkmaktadır. 2 nolu node da sırasıyla x,y eksenlerindeki yer değiştirmeler 3 ve 4 nolu yer değiştirmelr olarak verilmiştir.

İkinci kısımda elemanlarda oluşan gerilme değerleri MPa olarak verilmektedir. İlk sıradaki 1 elemanın gerilme değeri -202,85 MPa olarak bası gerilmesi, 2. elemanın ise 197,72 MPa olarak bası gerilmesi oluştuğu görülmektedir. Optimizasyonu yaparken seçtiğimiz amaç verdiğimiz 200 MPa lık emniyetli gerilmeyi sağlayan minimum kesit alanını bulmak olduğundan görüldüğü gibi iki kolda da bu değere çok yaklaşılmıştır.

8. SONUÇLAR

Son yıllarda bir çok alanda kullanılmaya başlanan sezgisel algoritmalardan olan Genetik Algoritma yapısal optimizasyon alanında da iyi sonuçlar vermektedir. Daha önceki bölümlerde anlatıldığı gibi optimizasyon problemlerinde bir çok karmaşık değişkenler, fonksiyonlar ve geniş arama uzayları ile karşılaşılmaktadır. GAda bu tür durumları ele almak daha kolay olmaktadır. Örnek olarak türev temelli algoritmalarda problem genelde sürekli değişkenlere bağlıdır: ayrık değişkenler sadece özel uygulamalarda kullanılabilmektedir. Tasarım optimizasyonunda karşımıza çıkan anlaşılması güç diferansiyel olmayan bazende süreksiz denklemler sadece sıfırıncı derecenden algoritmalar ile çözülmesi mümkün olmaktadır, çünkü bu teknikler sadece fonksiyonun değeri ile ilgilenmekte türevlerini kullanmamaktadır. Evrimsel algoritmalar birçok yönden kolaylıklar getirmesinin yanında birçok parametreyi içinde barındırması ve bunların dikkatli kullanılması da gerekmektedir. Karmaşık ve klasik yöntemlerle çözülemeyen problemlere kısa sürede yaklaşık çözümler bulması bu yöntemlerin en büyük artılarından sayılabilir.

Sonuç olarak bu algoritmalar geliştikçe daha çok probleme daha yaklaşık ve hızlı çözümler bulunması mümkün olacaktır.

KAYNAKLAR

Aksoy, B., (2001), "Döviz Piyasalarında Teknik Analiz Temelli Bir Alım/Satım Stratejisi ve Genetik Algoritmalar İle Optimizasyonu", Yüksek Lisans Tezi, Boğaziçi Üniversitesi Fen Bilimleri Enstitüsü.

Arora, J., "Introduction to Optimumm Design", Mc Graw-Hill Book Company Inc. (1989)

Biegel, J.E. ve Davern, J.J., (1990), "Genetic Algorithms and Job Shop Scheduling", Computers and Industrial Engineering, 19(1-4): 81-91.

Bodnovich, T., (1995), "Genetic Algorithms in Business and Their Supportive Role in Decision Making", College of Business Administration Kent State University.

Bozacı, A., (2002), "Sistematik Konstrüksiyon Ders Notu", Yıldız Teknik Üniversitesi Makine Fakültesi, İstanbul, 2002.

Cantoni, M., Zio, E., ve Marseguerra, M., (2000), "Genetik Algoritmalar ve Optimal İşletme Tasarımı İçim Monte Carlo Modeli", Reliability Engineering and System Safety, 68: 29-38.

Cappello, F. ve Mancuso, A., "A Genetic Algorithm for Combined Topology and Shape Optimizations", Computer-aided Design Vol. 35 (2003)

Coello, C.A.C., "Theoretical and numerical constrant-handling techniques used with evolutionary algorthim: a survey of the state of art", Computer Method in Applied Mechanics and Engineering, 191, s. 1245-1287 (2002)

Chen, R-S., Lu, K-Y. ve Yu, S-C., "A hybrid genetic algorithm approach on multi-objective of assembly planning problem", Engineering Applications of Artificial Intelligence, 15 (2002) s. 447-457

Çetin, N., (2002), "Genetik Algoritma", Yüksek Lisans Tezi, Yıldız Teknik Üniversitesi Fen Bilimleri Enstitüsü, İstanbul, 2002.

Deb, K. ve Gulati, S., "Design of truss-structures for minimum weight using genetic algorithms", Finite Elements in Analysis and Design, vol. 37, s. 447-465 (2001)

Dengiz, B. ve Altıparmak F., (1998), "Genetik Algoritmalara Genel Bakış", Gazi Üniversitesi Mühendislik Mimarlık Fakültesi, Endüstri Mühendisliği, 9 (3).

Goldberg, D.E., (1989), "Genetic Algorithms in Search, Optimization and Machine Learning", Addison-Wesley, Reading, MA, 1989.

Grefenstette, J. J., (1986), "Optimization of Control Parameters for Genetik Algorithms", IEEE Transactions on Systems, Man and Cybernetics.

Iuspa, L., Scaramuzzino, F. and Petrenga, P., "Optimal design of an aircraft engine mount via bit-masking oriented genetic algorithms", Advances in Engineering Softwre vol. 34 (2003)

Kameshki, E.S., ve Saka, M.P., "Optimum design og nonlinear steel frames with semi-rigid connections using a genetic algorithm", Computer and Structures 79, s. 1593-1604 (2001)

Kane, C. ve Schoenauer, M., "Topological Optimum Design using Genetic Algorithm", Control and Cybernetics, Vol. 25 No. 5 (1996)

Kreng, V. B. ve Lee, T-P., "Modular Product Design with Grouping Genetic Algorithm", Computer and Industrial Engineering Vol. 46 (2004)

Kutay, T., Kocabaş, H. ve Tümkor, S., "Bilgisayar destekli Konstrüksiyon ve Uygulamaları", (1997)

Kwon, Y.W. ve Bang, H., "The Finite Element Method Using MATLAB", CRC Pres, (1996)

Lacroix, D. ve Bouillard Ph., "Improved sensitivity analysis by a coupled FE-EFG method", Computers and Structures, vol. 81, s. 2431-2439 (2003)

Mackerle, J., "Topology and shape optimization of structures using FEM and BEM: a bibliography (1999-2001), Finite Element in Analysis and Design, vol. 39, s. 243-253 (2003)

Osyczka, A., "Evolutionary algorithm for single and multicriteria design optimization", Studies in Fuzzyness and Soft Computing, Physica-Verlang, Heidenlberg, 218 s. (2002)

Pahl, G. ve Beitz, W., (1996), "Engineering Design: A Systematic Approach", Springer, 1996.

Pardalos, P.M., Edwin, H. ve Tuy, H., "Recent developments and trands in global optimization", Journel of Computational and Applied Mathematics, vol. 124, s. 209-228 (2000)

Reynolds, D., McConnachie, J., Bettess, P., Christie, W.C. ve Bull, J.W., "reverse adaptivity – A new evolutionary tool for structural optimization", Int. Journal for Numerical Method in Engineering, vol. 45 (5), s. 529-552 (1999)

Shih, C.J. ve Yang, Y.C., "Mixed discrete and integer structural optimization using generalized", Hopfield network based sequential unconstrained minimization technique with additional penalty straegy", Identification, Control and Optimization of Engineering Structures, Civil-Comp Pres, Edinburg, s. 73-78 (2000)

Tanrıseven, D., (2000), "Genetik Algoritmalar Kullanarak Peptidlerde Motif Bulma", Yüksek Lisans Tezi, Boğaziçi Üniversitesi Fen Bilimleri Enstitüsü.

The MathWorks, (2005), "Genetic Algorithm and Direct Search Toolbox User's Guide", 2005

Turhan, Ö., "Optimizasyon Yöntemleri Ders Notları", İTÜ Makina Fakültesi.

Valenzuela, C.L., (1995), "Evolutionary Divide and Conquer: a novel genetic approach to the TSP", Imperial College.

Winter, G., Périaux, J. Galan, M. ve Cuesta, P., "Genetic algorithm in engineering and computer science", John Willey and Sons Ltd, England, 480 s. (1995)

Yeniay, M.Ö., (1999), "Taguchi Deney Tasarımı Problemlerine Genetik Algoritma Yaklaşımı", Doktora Tezi, Hacettepe Üniversitesi Fen Bilimleri Enstitüsü.

ÖZGEÇMİŞ

Doğum tarihi	08.10.1978	
Doğum yeri	İstanbul	
Lise	1992-1995	Bahçelievler Lisesi
Lisans	1997-2002	Yıldız Teknik Üniversitesi Makine Fakültesi Makine Mühendisliği Bölümü
Yüksek Lisans	2002-devan ediyor.	Yıldız Teknik Üniversitesi Fen Bilimleri Enstitüsü Makine Mühendisliği Anabilim Dalı Konstrüksiyon Programı