Yapısal Optimizasyon Teknikleri

Bilal TAYFUR

Özet. Bu ders, yapısal sistemlerin optimizasyonu konusunda kapsamlı bir giriş sunmaktadır. Öğrenciler, optimizasyon teorisinin temellerinden başlayarak, klasik ve modern optimizasyon yöntemlerinin genel farkını öğrenecek, yapısal sistemlerin tasarımında ve analizinde bu yöntemleri uygulama becerisi kazanacaklardır.

Anlatılan konuların sağ tarafında verilen karekodlar üzerinden derse ait GitHub reposunda ilgili örneğin kodlarına erişilebilir. Bu repo her yıl güncellenmektedir. Dolayısıyla güncel ders notlarına ve yeni eklenen uygulamalara da repo üzerinden erişlebilir ve arzu edilirse katkı sağlanabilir.

Bu ders kapsamında, aşağıdaki kaynak kitaplar konuları daha derinlemesine incelemek isteyen öğrencilere tavsiye edilmektedir:

- Cottle, R. W., & Thapa, M. N. (2017). **Linear and Nonlinear Optimization**. Springer.
- Rao, S. S. (2019). **Engineering Optimization: Theory and Practice, 5th Edition**. John Wiley & Sons.
- Arora, J. S. (2016). **Introduction to Optimum Design, 4th Edition**. Academic Press.
- Yang, X. S. (2021). **Nature-Inspired Optimization Algorithms, 2nd Edition**. Elsevier.
- Bendsøe, M. P., & Sigmund, O. (2003). **Topology Optimization: Theory, Methods, and Applications**. Springer.

İçerik

1 Optimizasyon Teorisine Giriş

Optimizasyon teorisi, bir sistemin performansını belirli kısıtlar altında en iyi duruma getirmeyi amaçlayan matematiksel ve metodolojik yaklaşımların bütünüdür. Bu ders, yapısal sistemlerin optimizasyonuna odaklanarak, temel kavramları ve uygulama yöntemlerini ele alacaktır.

1.1 Optimizasyonun Tanımı ve Mühendislikteki Önemi

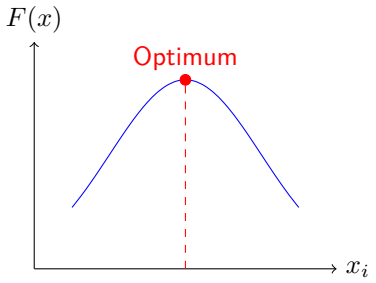
Optimizasyon, bir sistemin performansını belirli kısıtlar altında en iyi duruma getirme sürecidir. [1] Bu süreç, mühendislik tasarımlarında maliyeti düşürürken performansı artırmayı hedefler.

[1] Günlük hayatta sıkça karşılaştığımız "en iyi" kararı verme süreçleri aslında birer optimizasyon problemidir. Örneğin, işe giderken en kısa yolu seçmek, market alışverişinde en uygun fiyatlı ürünleri tercih etmek gibi.

1.2 Yapısal Optimizasyonun Temel Bileşenleri

Yapısal optimizasyon, üç temel bileşen üzerine kurulur: amaç fonksiyonu, tasarım değişkenleri ve kısıtlar. Bu bileşenler, optimizasyon probleminin matematiksel formülasyonunu oluşturur.

- **Amaç Fonksiyonu:** Minimize veya maksimize edilmek istenen hedef (örn. ağırlık, maliyet, rijitlik)
- **Tasarım Değişkenleri:** Optimize edilecek parametreler (örn. kesit boyutları, malzeme özellikleri)
- **Kısıtlar:** Tasarımın sağlaması gereken koşullar (örn. gerilme limitleri, deplasman sınırları)



Şekil 1: Tek değişkenli bir optimizasyon probleminde optimum noktanın gösterimi

1.3 Optimizasyonun Tarihçesi ve Gelişimi

Optimizasyon teorisinin temelleri, matematiksel analiz yöntemlerinin gelişimiyle paralel olarak ilerlemiştir. Modern optimizasyon yöntemleri, bilgisayar teknolojisinin gelişimiyle birlikte yeni boyutlar kazanmıştır. [2]

1.3.1 Önemli Tarihsel Gelişmeler

- 1940'lar: Doğrusal programlama ve Simplex metodunun geliştirilmesi
- 1950'ler: Dinamik programlama ve konveks optimizasyon teorisi
- 1960'lar: Sonlu elemanlar yönteminin optimizasyona uygulanması
- 1970'ler: Sayısal optimizasyon algoritmalarının geliştirilmesi
- 1980'ler: Metasezgisel algoritmaların ortaya çıkışı
- 1990'lar: Topoloji optimizasyonunun yaygınlaşması
- 2000'ler: Çok amaçlı optimizasyon ve yapay zeka tekniklerinin entegrasyonu

[2] İlk yapısal optimizasyon çalışmaları, Michell'in 1904'te yayınladığı kafes sistemlerin minimum ağırlık tasarımı ile başlamıştır. Bu çalışma, modern topoloji optimizasyonunun temelini oluşturur.

1.4 Optimizasyon Problemlerinin Genel Yapısı

Her optimizasyon problemi, bir amaç fonksiyonunun minimizasyonu veya maksimizasyonu şeklinde ifade edilir. Problem formülasyonu, tasarım değişkenlerini ve kısıtları içerir. [3]

[3] Optimizasyon probleminin matematiksel formülasyonu, problemi sistematik bir şekilde çözebilmemiz için gerekli olan yapıyı sağlar. Bu formülasyon, farklı mühendislik problemlerini ortak bir çerçevede ele almamıza olanak tanır.

$$
\begin{aligned}
\text{minimize} \quad & f(\mathbf{x}) \\
\text{subject to} \quad & g_i(\mathbf{x}) \leq 0, \qquad i = 1, \ldots, m \\
& h_j(\mathbf{x}) = 0, \qquad j = 1, \ldots, p \\
& x_k^L \leq x_k \leq x_k^U, \quad k = 1, \ldots, n
\end{aligned}
\tag{1}
$$

Yapısal Optimizasyon Örneği

Bir çelik kirişin optimum tasarımı için:

- **Amaç:** Minimum ağırlık
- **Değişkenler:** Kesit yüksekliği ve genişliği
- **Kısıtlar:**
  - Maksimum gerilme $\leq$ Akma gerilmesi
  - Maksimum sehim $\leq$ İzin verilen sehim
  - Minimum kesit boyutları

1.5 Optimizasyonun Mühendislikteki Temel Uygulama Alanları

Optimizasyon, mühendisliğin çeşitli alanlarında yaygın olarak kullanılır. İnşaat, makine, havacılık ve uzay mühendisliği başlıca uygulama alanlarıdır.

- **İnşaat Mühendisliği:**
  - Çelik yapıların kesit optimizasyonu
  - Betonarme elemanların donatı optimizasyonu
  - Köprü tasarımında form optimizasyonu
- **Makine Mühendisliği:**
  - Mekanik parçaların şekil optimizasyonu
  - Termal sistemlerin performans optimizasyonu
  - Titreşim kontrolü ve sönümleme
- **Havacılık ve Uzay Mühendisliği:**
  - Kanat ve gövde tasarımı
  - Kompozit malzeme optimizasyonu
  - Yapısal ağırlık minimizasyonu

1.6 Deterministik ve Stokastik Optimizasyon Yaklaşımları

Optimizasyon yöntemleri, problem çözme yaklaşımlarına göre deterministik [4] ve stokastik [5] olmak üzere iki ana kategoriye ayrılır.

1.6.1 Deterministik Yaklaşımlar

- Her çalıştırmada aynı sonucu verir
- Gradyan tabanlı yöntemler bu kategoridedir
- Lokal optimuma takılma riski vardır
- Başlangıç noktasına bağımlıdır

[4] Deterministik bir yöntemin, her çalıştırmada aynı sonucu vermesi gerektiği anlamına gelir. Yani stabil ve öngörülebilirdirler.

[5] Stokastik ise bir yöntemin, her çalıştırmada farklı sonuçlar üretebilmesi gerektiği anlamına gelir. Yani rastgele ve öngörülemez (daha doğrusu öngörüsü kısıtlı) bir yöntemdir.

1.6.2 Stokastik Yaklaşımlar

- Rastgelelik içerir
- Her çalıştırmada farklı sonuçlar üretebilir
- Global optimumu bulma olasılığı daha yüksektir
- Genetik algoritmalar, tavlama benzetimi gibi yöntemler bu kategoridedir

Bu yaklaşımlar, farklı problem tiplerine uygun çözüm stratejileri sunar. [6]

[6] Deterministik ve stokastik yaklaşımların seçimi, problemin yapısına ve çözüm gereksinimlerine bağlıdır. Örneğin, çok modlu bir problemde stokastik yöntemler daha avantajlı olabilir.

1.7 Doğrusal ve Doğrusal Olmayan Optimizasyon

Optimizasyon problemleri, amaç fonksiyonu ve kısıtların yapısına göre doğrusal ve doğrusal olmayan problemler olarak sınıflandırılır. Bu sınıflandırma, kullanılacak çözüm yöntemlerini belirler.

1.7.1 Doğrusal Optimizasyon

- Amaç fonksiyonu ve kısıtlar doğrusaldır
- Çözüm uzayı konvekstir
- Simplex metodu gibi etkin çözüm yöntemleri vardır
- Global optimum garanti edilir

Doğrusal Optimizasyon Örneği

Bir üretim planlaması problemi:

$$\begin{aligned} \text{maximize} \quad & 3x_1 + 2x_2 \\ \text{subject to} \quad & 2x_1 + x_2 \leq 100 \\ & x_1 + x_2 \leq 80 \\ & x_1, x_2 \geq 0 \end{aligned}$$

1.7.2 Doğrusal Olmayan Optimizasyon

- Amaç fonksiyonu ve/veya kısıtlar doğrusal değildir
- Çözüm uzayı karmaşıktır
- Lokal optimumlar içerebilir
- Yapısal problemlerin çoğu bu kategoridedir [7]

[7] Yapısal mühendislikte karşılaşılan problemlerin büyük çoğunluğu doğrusal olmayan karakterdedir. Örneğin, geometrik nonlineerite, malzeme nonlineeritesi gibi etkiler problemi doğrusal olmayan hale getirir.

Doğrusal Olmayan Optimizasyon Örneği

Bir yapısal tasarım optimizasyonu problemi:

$$\begin{aligned} \text{minimize} \quad & f(x) = x_1^2 + 2x_2^2 - 0.3x_1x_2 \\ \text{subject to} \quad & g_1(x) = x_1^2 + x_2^2 - 25 \leq 0 \\ & g_2(x) = x_1 - 2x_2 + 5 \leq 0 \\ & -10 \leq x_1, x_2 \leq 10 \end{aligned}$$

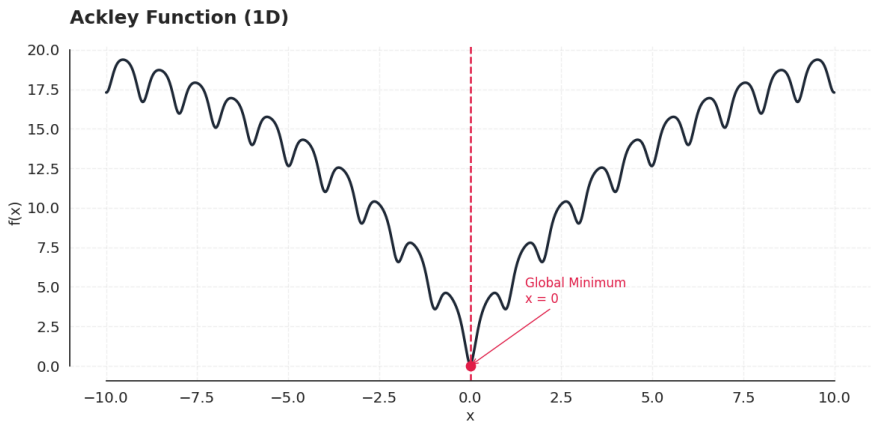
1.8 Çözüm Yöntemlerinin Genel Sınıflandırılması

Optimizasyon yöntemleri, problem tipine ve çözüm stratejisine göre analitik, sayısal ve metasezgisel yöntemler olarak sınıflandırılır. Her yöntem grubu, belirli problem tipleri için avantajlar sunar.

- **Analitik Yöntemler**
  - Diferansiyel hesap
  - Varyasyonel yöntemler
  - Lagrange çarpanları
- **Sayısal Yöntemler**
  - Gradyan tabanlı yöntemler
  - Doğrusal programlama
  - Nonlineer programlama
- **Metasezgisel Yöntemler**
  - Genetik algoritmalar
  - Parçacık sürü optimizasyonu
  - Tavlama benzetimi

1.9 Optimizasyon Problemlerinde Global ve Lokal Optimumlar

Çok modlu optimizasyon problemlerinde, birden fazla lokal optimum noktası bulunabilir. Bu durum, özellikle doğrusal olmayan problemlerde sıklıkla karşımıza çıkar.



Şekil 2: Tek boyutlu ve çok modlu optimizasyon problemi

Yukarıdaki şekilde görüldüğü gibi, çok modlu bir fonksiyonda birden fazla tepe (maksimum) ve çukur (minimum) noktası bulunabilir. Optimizasyon algoritmaları, başlangıç noktasına bağlı olarak lokal bir optimuma takılabilir ve global optimumu bulamayabilir. Bu nedenle, özellikle karmaşık mühendislik problemlerinde global optimumu bulmak için metasezgisel yöntemler tercih edilebilir.

2 Temel Optimizasyon Kavramları

Optimizasyon problemlerinin matematiksel olarak formüle edilmesi ve çözülmesi için gerekli olan temel kavramlar, bu bölümde detaylı olarak incelenecektir. Bu kavramlar, daha karmaşık optimizasyon problemlerinin anlaşılması için temel oluşturur.

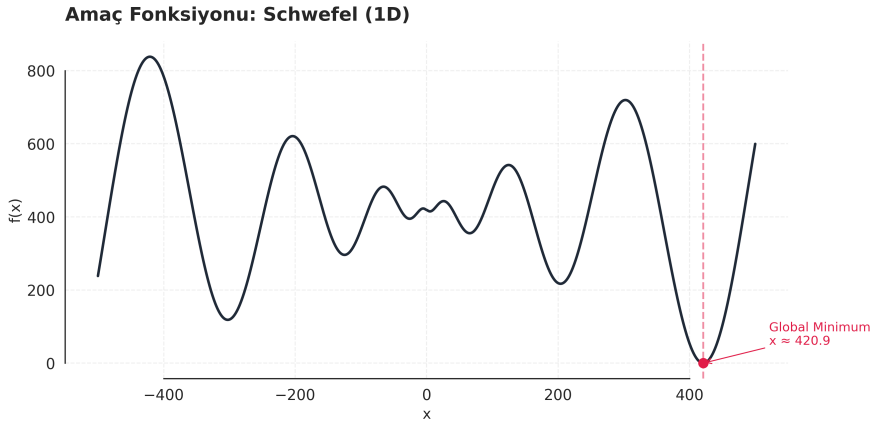
2.1 Amaç Fonksiyonu ve Kısıt Fonksiyonları

Optimizasyon problemlerinin matematiksel modellemesinde iki temel bileşen vardır: amaç fonksiyonu ve kısıt fonksiyonları.

2.1.1 Amaç Fonksiyonu

Amaç fonksiyonu, optimize edilmek istenen performans kriterini matematiksel olarak ifade eder. Bu fonksiyon:

- Minimize edilebilir (örn. maliyet, ağırlık, enerji tüketimi)
- Maksimize edilebilir (örn. verim, dayanım, rijitlik)



Şekil 3: Amaç Fonksiyonu

[8]

2.1.2 Kısıt Fonksiyonları

Kısıt fonksiyonları, tasarımın sağlaması gereken şartları matematiksel olarak ifade eder:

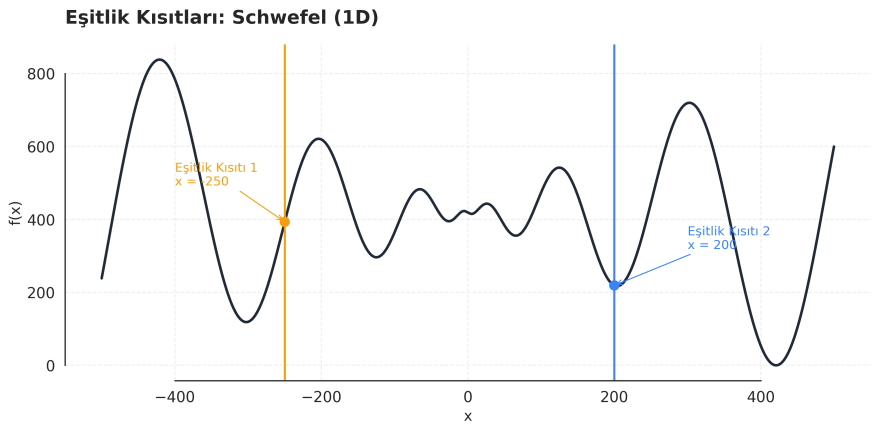
- Eşitlik kısıtları: $h_j(x) = 0$
- Eşitsizlik kısıtları: $g_i(x) \leq 0$
- Sınır kısıtları: $x_L \leq x \leq x_U$ [9]

Eşitlik kısıtları: $h_j(x) = 0$ Eşitlik kısıtları, tasarım değişkenlerinin tam olarak belirli bir değer alması gereken durumları ifade eder. Bu tür kısıtlar, optimizasyon problemi içinde bazı parametrelerin kesin olarak sağlanması gereken şartları matematiksel olarak tanımlar. Örneğin, bir yapının toplam ağırlığının belirli bir değere eşit olması veya bir kimyasal reaksiyonda kütle dengesinin sağlanması gibi durumlar eşitlik kısıtlarıyla ifade edilebilir. Eşitlik kısıtları genellikle optimizasyon uzayını daha dar bir alanda tutarak, mümkün çözümlerin sayısını önemli ölçüde azaltır.

[8] Her maksimizasyon problemi, amaç fonksiyonunun negatifi alınarak bir minimizasyon problemine dönüştürülebilir:

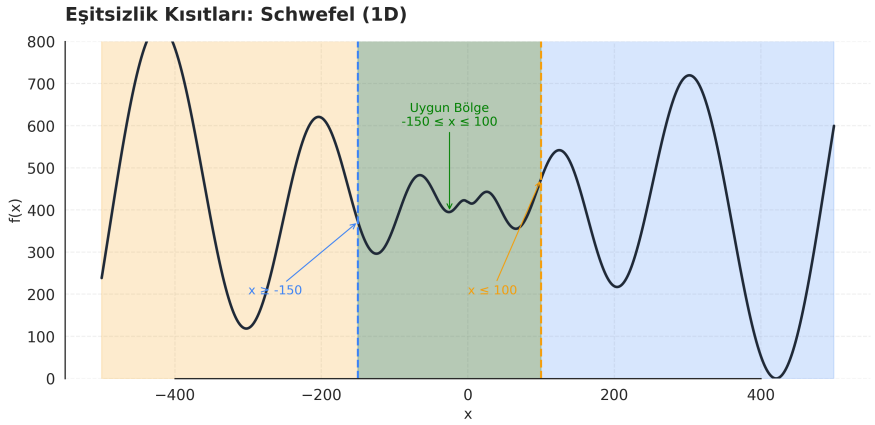
$$\max f(x) = -\min(-f(x))$$

[9] İstisnai durumlar haricinde yapısal optimizasyon problemleri genellikle eşitsizlik kısıtlarıyla ifade edilir. Probleme bağlı olarak sınır kısıtları da kullanılabilir. Ancak hiperstatik bir yapının optimizasyona konu olması halinde bu kısıtlayıcı, global optimumun ıskalanmasına da sebep olabilir.
Aslında yapısal optimizasyon problemlerini optimizasyon açısından incelenmeye değer kılan noktalardan birisi de bu hiperstatiklik durumunun birçok yapısal optimizasyon probleminin öngörülmezliğini önemli ölçüde artırmasıdır.



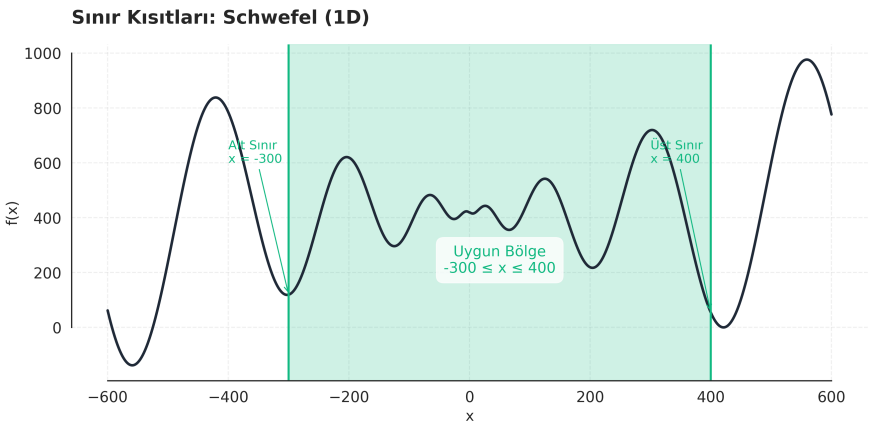
Şekil 4: Eşitlik kısıtları

Eşitsizlik kısıtları: $g_i(x) \leq 0$ Eşitsizlik kısıtları, tasarım değişkenlerinin belirli bir sınırı aşmaması gereken durumları ifade eder. Bu tür kısıtlar, bir yapının dayanabileceği maksimum gerilme değeri, bir sistemin maksimum enerji tüketimi veya bir malzemenin minimum güvenlik faktörü gibi sınırlamaları tanımlamak için kullanılır. Eşitsizlik kısıtları, tasarımın fiziksel olarak gerçeklenebilir ve güvenli olmasını sağlamak için kritik öneme sahiptir. Ayrıca eşitsizlik kısıtları, uygulanabilir çözüm alanını belirleyerek, optimizasyon algoritmasının yalnızca geçerli tasarım alanında arama yapmasını sağlar.

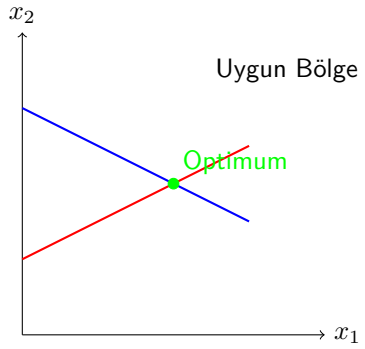


Şekil 5: Eşitsizlik kısıtları

Sınır kısıtları: $x_L \leq x \leq x_U$ Sınır kısıtları, her bir tasarım değişkeninin alabileceği minimum ve maksimum değerleri belirleyen kısıtlardır. Bu kısıtlar, tasarım değişkenlerinin fiziksel sınırlarını, üretilebilirlik koşullarını veya standartlarca belirlenmiş aralıkları yansıtır. Örneğin, bir kirişin kalınlığının üretim kısıtları nedeniyle belirli bir değerden küçük olamayacağı veya kurulum gereksinimleri nedeniyle belirli bir maksimum değeri aşamayacağı durumlar sınır kısıtlarıyla ifade edilir. Sınır kısıtları, optimizasyon algoritmasının arama uzayını daraltarak, hesaplama verimliliğini artırır ve fiziksel olarak anlamsız çözümlerin elenmesini sağlar.



Şekil 6: Sınır Kısıtları



Şekil 7: İki kısıtın kesişimi ile oluşan uygun çözüm bölgesi

2.2 Karar Değişkenleri ve Çözüm Uzayı

2.2.1 Karar Değişkenleri

Karar değişkenleri, optimizasyon sürecinde değerleri belirlenmesi gereken parametrelerdir:

- Sürekli (Continuous) değişkenler (örn. boyutlar, kalınlıklar)
- Tamsayı (Discrete) değişkenler (örn. eleman sayısı)
- İkili (Binary) değişkenler (0-1 kararları)

Yapısal Tasarımda Karar Değişkenleri

Bir çelik çerçeve optimizasyonunda:

- **Sürekli:** Kesit boyutları
- **Tamsayı:** Kat sayısı
- **İkili:** Elemanların varlığı/yokluğu

2.2.2 Çözüm Uzayı

Çözüm uzayı, tüm olası karar değişkeni kombinasyonlarının oluşturduğu uzaydır[10]:

- **Uygun Çözüm Bölgesi:** Tüm kısıtları sağlayan noktalar kümesi
- **Uygun Olmayan Bölge:** En az bir kısıtı ihlal eden noktalar

[10] Çözüm uzayının boyutu, karar değişkenlerinin sayısı ile belirlenir. Yüksek boyutlu problemlerde "lanet boyut" (curse of dimensionality) problemi ortaya çıkar.

2.3 Global ve Yerel (Lokal) Minimum/Maksimum Kavramları

2.3.1 Yerel Optimum

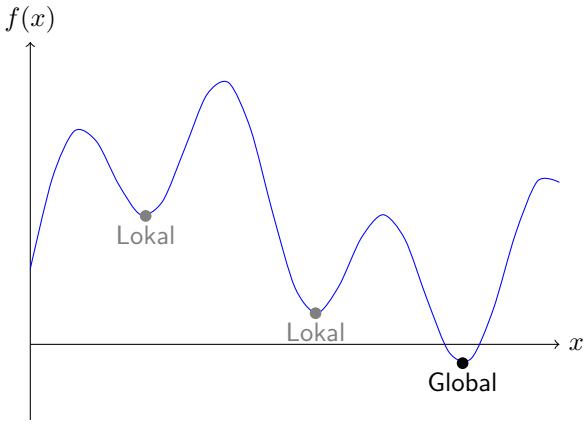
Bir nokta, belirli bir komşuluk içinde en iyi değere sahipse yerel optimumdur:

$$x^* \text{ yerel minimum } \Leftrightarrow f(x^*) \leq f(x), \forall x \in N(x^*) \tag{2}$$

2.3.2 Global Optimum

Tüm çözüm uzayında en iyi değere sahip nokta global optimumdur:

$$x^* \text{ global minimum } \Leftrightarrow f(x^*) \leq f(x), \forall x \in S \tag{3}$$



Şekil 8: Bir fonksiyonun yerel ve global minimum noktaları

2.4 Fiziksel ve Matematiksel Modelleme

2.4.1 Fiziksel Modelleme

Gerçek dünya probleminin fiziksel prensiplere dayalı olarak modellenmesi[11]:

- Kuvvet dengesi
- Enerji korunumu
- Malzeme davranışı
- Geometrik ilişkiler

[11] İyi bir matematiksel model, fiziksel gerçekliği yeterli doğrulukta yansıtmalı, ancak gereksiz karmaşıklıktan kaçınmalıdır. Bu esasen optimizasyonun başlıca konusu olmasa da yapısal optimizasyon açısından oldukça önemli bir konudur.

2.4.2 Matematiksel Modelleme

Fiziksel modelin matematiksel formülasyona dönüştürülmesi:

- Diferansiyel denklemler
- Cebirsel denklemler
- Matris formülasyonları
- Sonlu eleman modelleri

2.5 Diferansiyellenebilirlik ve Süreklilik

2.5.1 Süreklilik

Fonksiyonun sürekli olması, küçük girdi değişimlerinin çıktıda ani sıçramalara neden olmaması demektir. Matematiksel olarak:

$$\lim_{x \to x_0} f(x) = f(x_0) \tag{4}$$

Süreklilik ve Optimizasyon İlişkisi

Süreklilik, optimizasyon problemlerinde önemli bir özelliktir çünkü:

- Sürekli fonksiyonlar için kapalı ve sınırlı bir aralıkta mutlaka bir minimum ve maksimum değer vardır (Weierstrass teoremi).
- Sürekli olmayan fonksiyonların optimizasyonu daha zordur çünkü ani sıçramalar, algoritmaların doğru yönde ilerlemesini engelleyebilir.
- Gerçek mühendislik problemlerinde çoğu fiziksel davranış, sürekli fonksiyonlarla modellenir (örneğin, bir kirişin yük altında deformasyonu).

2.5.2 Diferansiyellenebilirlik

Fonksiyonun türevinin var olması, fonksiyonun davranışının her noktada bir teğet doğru ile yaklaşık olarak ifade edilebilmesi anlamına gelir. Matematiksel olarak:

$$f'(x_0) = \lim_{h \to 0} \frac{f(x_0 + h) - f(x_0)}{h} \tag{5}$$

Diferansiyellenebilirlik ve Optimizasyon Arasındaki İlişki

Diferansiyellenebilirlik, optimizasyon sürecinde kritik bir role sahiptir:

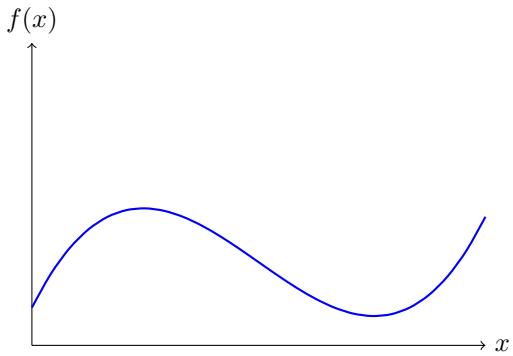
- **Gradyan Bilgisi:** Türev, fonksiyonun en hızlı artış/azalış yönünü verir, böylece optimizasyon algoritmaları nereye gideceklerini bilir.
- **Kritik Noktalar:** Fonksiyonun türevinin sıfır olduğu noktalar (kritik noktalar), potansiyel optimum noktalardır.
- **İkinci Türev:** İkinci türev, kritik noktanın minimum, maksimum veya eğer noktası olduğunu belirlemede yardımcı olur.

Gerçek Hayat Örneği: Tepeye Tırmanma

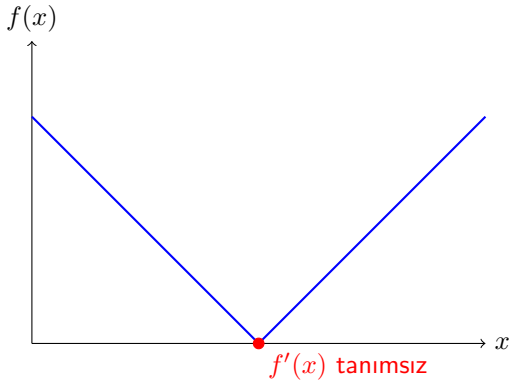
Düşünün ki bir dağa tırmanıyorsunuz ve hedefiniz zirveye ulaşmak. Eğer sisli bir günde görüş alanınız çok kısıtlıysa, nasıl ilerlemelisiniz?

- **Gradyan (Türev) Bilgisi:** Her adımda ayaklarınızla zeminin eğimini hissederek en dik yokuş yukarı yönünü (negatif gradyan) seçebilirsiniz.
- **Diferansiyellenemeyen Noktalar:** Eğer yolunuz üzerinde dik bir kayalık (türevin tanımlanamadığı nokta) varsa, doğrudan ilerleyemez, farklı bir yol bulmanız gerekir.
- **Yerel Tepe vs. Zirve:** Tırmanışınız sırasında küçük bir tepeye (yerel maksimum) ulaşabilirsiniz, ancak gerçek zirve (global maksimum) başka bir yerde olabilir.

Bu anoloji, gradyan tabanlı optimizasyon algoritmalarının çalışma prensibine çok benzerdir.



Şekil 9: Diferansiyellenebilir Fonksiyon ve Gradyan



Şekil 10: Diferansiyellenemeyen Fonksiyon (Mutlak Değer Fonksiyonu)

Diferansiyellenebilirliğe Göre Optimizasyon Yöntemleri

- **Diferansiyellenebilir Fonksiyonlar için Gradyan Tabanlı Yöntemler:**
  - **Gradyan İniş:** Fonksiyonun en hızlı düşüş yönünde ilerler.
  - **Newton Yöntemi:** İkinci türev bilgisini kullanarak daha hızlı yakınsama sağlar.
  - **Quasi-Newton:** İkinci türevi yaklaşık olarak hesaplar (BFGS, L-BFGS gibi).
- **Diferansiyellenemeyen Fonksiyonlar için Gradyan-sız Yöntemler:**
  - **Simpleks Arama:** Fonksiyon değerlerini karşılaştırarak en uygun yönü belirler (Nelder-Mead yöntemi).
  - **Genetik Algoritmalar:** Evrimsel süreçleri taklit ederek çözüm uzayını araştırır.
  - **Parçacık Sürü Optimizasyonu:** Grup davranışını taklit ederek çözüme yaklaşır.
- **Karma Problemler için Hibrit Yaklaşımlar:**
  - **Arama + İyileştirme:** Gradyan-sız bir yöntemle geniş bölgede arama, sonra bulduğu noktadan gradyan tabanlı yöntemle iyileştirme.
  - **Alt-Gradyan Yöntemleri:** Diferansiyellenemeyen noktalarda bile ilerleme sağlayan genelleştirilmiş türev yaklaşımları.

[12]

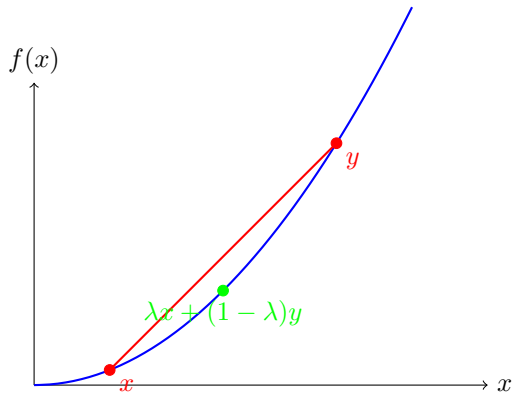
[12] Yapısal optimizasyon problemleri genellikle diferansiyellenebilir değildir. Bu nedenle, global optimumu bulmak zorlaşır ve metasezgisel yöntemler tercih edilir.

2.6 Konveks ve Konveks Olmayan Optimizasyon

2.6.1 Konveks Fonksiyonlar

Bir fonksiyonun konveks olması, fonksiyonun grafiğinin herhangi iki noktasını birleştiren doğru parçasının, bu iki nokta arasındaki fonksiyon grafiğinin üzerinde kalması demektir. Matematiksel olarak, $f : \mathbb{R}^n \to \mathbb{R}$ fonksiyonu için:

$$f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda)f(y), \quad \forall x, y \in \mathbb{R}^n, \forall \lambda \in [0,1] \tag{6}$$

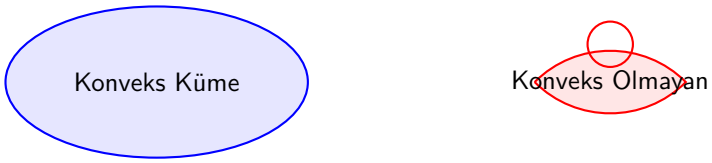


Şekil 11: Konveks Fonksiyon Örneği

2.6.2 Konveks Küme

Konveks bir küme, küme içindeki herhangi iki noktayı birleştiren doğru parçasının tamamen küme içinde kalması anlamına gelir:

$$\lambda x + (1-\lambda)y \in S, \quad \forall x, y \in S, \forall \lambda \in [0,1] \tag{7}$$



Şekil 12: Konveks ve Konveks Olmayan Küme Örnekleri

2.6.3 Konveks Optimizasyon Avantajları

Konveks optimizasyon, optimizasyon teorisinde özel bir öneme sahiptir[13]:

- **Yerel Optimum = Global Optimum:** Konveks bir fonksiyonun yerel minimumu, aynı zamanda global minimumdur. Bu, optimizasyon sürecini önemli ölçüde basitleştirir.
- **Kararlı Çözüm:** Başlangıç noktasından bağımsız olarak, uygun gradyan tabanlı algoritmalar aynı optimum noktaya yakınsar.
- **Verimli Algoritmalar:** İç nokta yöntemleri, gradyan iniş yöntemi gibi algoritmalar konveks problemlerde polinom zamanda çözüme ulaşabilir.

Konveks Optimizasyon Örneği

Karesel programlama problemi:

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & \frac{1}{2} x^T Q x + c^T x \\ \text{s.t.} \quad & Ax \leq b \end{aligned} \tag{8}$$

Eğer Q pozitif yarı-tanımlı bir matris ise, bu problem konvekstir ve verimli iç nokta yöntemleriyle çözülebilir.

[13] Konveks optimizasyon problemleri için geliştirilmiş etkili algoritmalar ve matematiksel garantiler, bu tür problemlerin çözümünü daha güvenilir hale getirir. Günümüzde sıkça duyduğumuz derin öğrenme algoritmalarının içinde yer alan optimizasyon problemleri genellikle konveks optimizasyon problemleridir. Burada kullanılan optimizasyon algoritmaları genellikle gradyan iniş yöntemleridir. Çok efektif yöntemler olmalarının yanında genellikle yapısaal optimizasyon problemlerinde verimsiz kalırlar.

2.6.4 Konveks Olmayan Optimizasyon Zorlukları

Konveks olmayan optimizasyon problemleri, yapısal optimizasyonda sıklıkla karşılaşılan zorluklardır[14]:

- **Çoklu Yerel Optimumlar:** Fonksiyon birden fazla yerel minimum içerebilir, bu nedenle global optimumu bulmak zorlaşır.
- **Başlangıç Noktası Bağımlılığı:** Gradyan tabanlı algoritmalar, başlangıç noktasına bağlı olarak farklı yerel optimumlara yakınsayabilir.
- **Hesaplama Maliyeti:** Global optimumu bulma garantisi için genellikle daha kapsamlı arama yöntemleri gereklidir, bu da hesaplama maliyetini artırır.

[14] Yapısal optimizasyon problemlerinin çoğu, birden fazla yerel optimum içerebilen konveks olmayan problemlerdir. Bu da global optimumu bulma konusunda güçlükler yaratır.

Konveks Olmayan Problemlere Yaklaşımlar

- **Çoklu Başlangıç Noktası:** Farklı başlangıç noktalarından birden fazla yerel optimizasyon çalıştırılarak en iyi sonuç seçilir.
- **Metasezgisel Algoritamalar:** Genetik algoritmalar, parçacık sürü optimizasyonu gibi yöntemler, geniş çözüm uzayını araştırarak global optimuma yaklaşmaya çalışır.
- **Konveks Yaklaşımlar:** Problemi konveks alt-problemlere ayrıştırarak veya konveks yaklaşımlar kullanarak çözüm aranabilir (Sequential Convex Programming gibi).

2.7 Kısıtsız ve Kısıtlı Optimizasyon

2.7.1 Kısıtsız Optimizasyon

Kısıtsız optimizasyon, adından da anlaşılacağı gibi herhangi bir kısıt içermeyen, sadece amaç fonksiyonunun minimize veya maksimize edilmesi gereken problemleri ifade eder. Matematiksel olarak:

$$\min_{x \in \mathbb{R}^n} f(x) \tag{9}$$

[15]

Kısıtsız optimizasyon problemlerini çözmek için kullanılan başlıca yöntemler [16]:

- **Gradyan İniş Yöntemi:** Fonksiyonun en hızlı düşüş yönünde (negatif gradyan yönünde) adım adım ilerleyerek minimuma ulaşmayı hedefler. Bu yöntem, özellikle derin öğrenme ve makine öğrenmesi uygulamalarında yaygın olarak kullanılır.
- **Newton Yöntemi:** Fonksiyonun ikinci türev (Hessian) bilgisini de kullanarak, hem yön hem de adım büyüklüğü konusunda daha akıllı kararlar verir. Kuadratik yakınsama özelliği sayesinde, doğru koşullar altında gradyan inişinden daha hızlı yakınsar.
- **Quasi-Newton Yöntemleri:** Newton yönteminin hesaplama maliyetini düşürmek için Hessian matrisini doğrudan hesaplamak yerine, yaklaşık olarak tahmin eden yöntemlerdir. BFGS ve L-BFGS en popüler Quasi-Newton algoritmaları arasındadır.
- **Eş Gradyan Yöntemi (Conjugate Gradient):** Özellikle büyük ölçekli problemlerde etkili olan ve ardışık arama yönlerinin birbirine "eş" (conjugate) olmasını sağlayan bir yöntemdir.
- **Trust Region Yöntemleri:** Her iterasyonda, fonksiyonun yerel olarak iyi modellenebileceği bir "güven bölgesi" belirleyerek bu bölge içinde optimum arayan yöntemlerdir.

[15] Kısıtsız optimizasyon, her ne kadar teorik olarak "kısıtsız" olarak adlandırılsa da, pratikte çoğu mühendislik problemi bir şekilde fiziksel veya matematiksel kısıtlar içerir. Buradaki "kısıtsız" terimi, problemin formülasyonunda açık kısıtlar olmadığı anlamına gelir.

[16] Bu beş yöntemin çalışma biçimini test eden python kodu, bağlantı üzerinden test edilebilir.

Dağa Tırmanma Analojisi

Kısıtsız optimizasyon yöntemlerini anlamak için bir dağ tırmanışı analojisi düşünelim (maksimizasyon problemi için):

- **Gradyan Tırmanışı:** Her adımda en dik yokuş yukarı yönde ilerlersiniz. Kolay uygulanır ancak dar vadilerde zigzaglar çizerek yavaş ilerleyebilir.
- **Newton Yöntemi:** Sadece zeminin eğimine (gradyan) değil, aynı zamanda arazinin şeklini (Hessian) de bakarsınız. Bu, düz alanlarda büyük adımlar atmanızı, dik yamaçlarda küçük ve dikkatli adımlar atmanızı sağlar.
- **Quasi-Newton:** Arazinin şeklini tam ölçmek yerine, önceki adımlarınızdan tahmin edersiniz. Bu, Newton kadar etkili olmasa da çok daha az çaba gerektirir.

2.7.2 Kısıtlı Optimizasyon

Kısıtlı optimizasyon, gerçek dünya problemlerinin modellenmesinde çok daha yaygındır ve amaç fonksiyonunun yanı sıra bir dizi kısıt içerir. Bu kısıtlar, çözümün sağlaması gereken şartları temsil eder. Matematiksel formülasyonu:

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f(x) \\ \text{s.t.} \quad & g_i(x) \leq 0, \quad i = 1, \ldots, m \\ & h_j(x) = 0, \quad j = 1, \ldots, p \end{aligned} \tag{10}$$

Burada $g_i(x) \leq 0$ eşitsizlik kısıtlarını, $h_j(x) = 0$ ise eşitlik kısıtlarını temsil eder.
[17]

Kısıtlı optimizasyon problemlerini çözmek için kullanılan başlıca yöntemler:

- **Lagrange Çarpanları Yöntemi:** Eşitlik kısıtlı problemler için Lagrange çarpanları adı verilen ek değişkenler tanımlayarak, kısıtlı problemi genişletilmiş bir kısıtsız probleme dönüştürür. Bu yöntem, kısıtların tam olarak sağlanması gerektiği durumlarda özellikle kullanışlıdır.
- **Karush-Kuhn-Tucker (KKT) Koşulları:** Hem eşitlik hem de eşitsizlik kısıtları için geçerli olan optimalite koşullarıdır. KKT koşulları, Lagrange çarpanları yönteminin eşitsizlik kısıtlarına genelleştirilmiş halidir.
- **Ceza Fonksiyonu Yöntemleri:** Kısıtlı problemi, kısıtların ihlaline ceza vererek kısıtsız bir probleme dönüştürür. İki ana yaklaşım vardır:
  - **Dış Ceza (Penalty) Metodu:** Kısıt ihlalleri için ceza ekler, uygun olmayan çözümlere izin verir ancak cezalandırır.
  - **İç Ceza (Barrier) Metodu:** Fizibil bölgenin sınırına yaklaşıldıkça giderek artan bir ceza ekler, böylece çözümün fizibil bölgenin içinde kalmasını sağlar.
- **Aktif Set Yöntemleri:** Her iterasyonda, aktif olduğu düşünülen kısıtları belirleyerek daha küçük bir alt problem çözer. Bu, özellikle doğrusal ve kuadratik programlama problemlerinde etkilidir.
- **İç Nokta (Interior Point) Yöntemleri:** Fizibil bölgenin içinde kalarak optimuma yaklaşır. Bariyer fonksiyonları kullanır ama kısıtları doğrudan ele alır. Büyük ölçekli doğrusal ve konveks programlama problemlerinde çok etkilidir.

[17] Yapısal optimizasyon problemleri neredeyse her zaman kısıtlı problemlerdir. Bir köprünün ağırlığını minimize etmek istediğinizde, köprünün belirli yükleri taşıyabilmesi, belirli bir güvenlik faktörüne sahip olması ve inşa edilebilir olması gibi çeşitli kısıtlar vardır. Bu kısıtlar olmadan yapılan bir optimizasyon, pratikte uygulama alanı bulamaz.

- **Ardışık Karesel Programlama (SQP):** Kısıtlı doğrusal olmayan problemleri, bir dizi karesel alt probleme dönüştürerek çözer. Yapısal optimizasyonda yaygın olarak kullanılır.

Kısıtlı Optimizasyon Analojisi: Patika Bulma

Kısıtlı optimizasyonu, hedefinize giden en iyi yolu bulmaya çalışırken belirli kurallara uymak zorunda olduğunuz bir patika bulma problemi olarak düşünebilirsiniz:

- **Hedefiniz (Amaç Fonksiyonu):** Dağın zirvesine ulaşmak.
- **Kısıtlarınız:** Sadece işaretli patikalardan yürüyebilirsiniz (eşitlik kısıtları), bazı tehlikeli bölgelerden uzak durmalısınız (eşitsizlik kısıtları).
- **Lagrange Yöntemi:** Patika haritasını ve tehlikeli bölgeleri sürekli kontrol ederek ilerlemeye benzer.
- **Ceza Metodu:** İşaretli patikadan çıkarsanız, ekstra zorluk yaşarsınız (çamura saplanma, dikenli çalılıklardan geçme gibi). Bu cezalara rağmen bazen kestirme yapmak avantajlı olabilir.
- **Bariyer Metodu:** Tehlikeli bölgelerin etrafında görünmez bir "itici güç" varmış gibi davranırsınız, yaklaştıkça geri çekilirsiniz. Böylece her zaman güvenli bölgede kalırsınız.

Kısıtların Ele Alınması

Kısıtlı bir problemi kısıtsız probleme dönüştürme yöntemleri:

- **Dış Ceza (Penalty) Yöntemi:**

$$\min f(x) + c \sum \max(0, g_i(x))^2 + c \sum (h_j(x))^2 \tag{11}$$

  Burada $c > 0$ ceza parametresidir ve genellikle iterasyonlar boyunca artırılır. Kısıt ihlalleri arttıkça ceza da artar, böylece algoritma zamanla fizibil bölgeye doğru yönlendirilir.

- **İç Ceza (Barrier) Yöntemi:**

$$\min f(x) - c \sum \ln(-g_i(x)) \tag{12}$$

  Bu yöntem, sadece $g_i(x) < 0$ (yani fizibil bölge içinde) olduğunda tanımlıdır ve kısıt sınırına yaklaştıkça $\ln(-g_i(x)) \to -\infty$ olur, bu da çözümün fizibil bölge içinde kalmasını sağlar. c parametresi genellikle iterasyonlar boyunca azaltılır.

- **Augmented Lagrangian Yöntemi:**

$$\min f(x) + \sum \lambda_j h_j(x) + \frac{c}{2} \sum (h_j(x))^2 + \sum \mu_i \max(0, g_i(x)) + \frac{c}{2} \sum \max(0, g_i(x))^2 \tag{13}$$

  Bu yöntem, Lagrange çarpanları ve ceza yöntemlerinin avantajlarını birleştirir. Lagrange çarpanları λ_j ve μ_i her iterasyonda güncellenir.

2.7.3 Kısıtlı ve Kısıtsız Optimizasyonun Karşılaştırılması

- **Problem Zorluğu:** Kısıtlı problemler genellikle daha zordur ve daha özel çözüm yöntemleri gerektirir.
- **Çözüm Uzayı:** Kısıtsız problemlerde tüm arama uzayı kullanılabilirken, kısıtlı problemlerde arama "fizibil bölge" ile sınırlıdır.
- **Gerçekçilik:** Gerçek mühendislik problemleri neredeyse her zaman kısıtlıdır, çünkü tasarımların belirli gereksinimleri karşılaması gerekir.
- **Çözüm Stratejisi:** Kısıtlı problemlerin çözümünde, genellikle önce kısıtların sağlanması, sonra amaç fonksiyonunun optimize edilmesi hedeflenir, ya da bu iki hedef dengeli bir şekilde ele alınır.

[18]

[18] Yapısal optimizasyon problemleri genellikle oldukça karmaşık kısıtlar içerir. Örneğin, bir gökdelenin tasarımında, maliyeti minimize ederken yapı dayanımı, kullanıcı konforu, deprem performansı, rüzgar yükleri gibi çok sayıda faktör kısıt olarak dikkate alınır. Bu kısıtların doğrusal olmaması ve birbiriyle etkileşimi, problemi çözmek için özel tekniklerin geliştirilmesini gerektirir.

2.8 Optimizasyon Yaklaşımlarının Sınıflandırılması

2.8.1 Problem Yapısına Göre

Optimizasyon problemleri, matematiksel yapılarına göre çeşitli kategorilere ayrılabilir, bu sınıflandırma hangi çözüm yöntemlerinin uygun olacağını belirlememize yardımcı olur:

- **Doğrusal Programlama (Linear Programming - LP)**
  - Doğrusal amaç fonksiyonu: $f(x) = c^T x$
  - Doğrusal kısıtlar: $Ax \leq b$, $Aeq \cdot x = beq$
  - Temel çözüm yöntemi: Simpleks algoritması, İç nokta yöntemleri
  - Özellikler: Tek bir global optimum, kısıtlar tarafından tanımlanan konveks bir fizibil bölge
  - Uygulama alanları: Kaynak tahsisi, üretim planlama, lojistik optimizasyonu
- **Doğrusal Olmayan Programlama (Nonlinear Programming - NLP)**
  - Doğrusal olmayan amaç fonksiyonu ve/veya kısıtlar
  - Alt kategori - Konveks Programlama: Amaç fonksiyonu konveks, kısıt kümesi konveks
  - Alt kategori - Konveks Olmayan Programlama: Amaç fonksiyonu ve/veya kısıt kümesi konveks değil
  - Çözüm yöntemleri: Gradyan tabanlı yöntemler (SQP, İç nokta, BFGS), metasezgisel yöntemler
  - Uygulama alanları: Yapısal tasarım, mekanik sistemlerin optimizasyonu, ekonomik modeller
- **Tamsayılı Programlama (Integer Programming - IP)**
  - Tüm değişkenler tamsayı olmalıdır: $x \in \mathbb{Z}^n$
  - Alt kategori - Karma Tamsayılı Programlama (MIP): Bazı değişkenler tamsayı, bazıları sürekli
  - Alt kategori - 0-1 Tamsayılı Programlama: Değişkenler yalnızca 0 veya 1 değerini alabilir
  - Çözüm yöntemleri: Dal-sınır (Branch-and-bound), kesme düzlemi (Cutting plane), dal-kesme (Branch-and-cut)

- Uygulama alanları: Çizelgeleme problemleri, kesme/paketleme problemleri, topoloji optimizasyonu

- **Stokastik Programlama**
  - Rastgele değişkenler içeren problemler
  - Belirsizliğin olasılık dağılımları ile modellendiği durumlar
  - Çözüm yöntemleri: Senaryo yaklaşımı, örnek ortalamalı yaklaşım, robust optimizasyon
  - Uygulama alanları: Risk yönetimi, portföy optimizasyonu, belirsizlik altında yapısal tasarım

- **Çok Amaçlı Programlama**
  - Birden fazla amaç fonksiyonunun eş zamanlı optimize edilmesi
  - Çözüm kavramı: Pareto-optimal çözüm kümesi
  - Çözüm yöntemleri: Ağırlıklı toplam yöntemi, epsilon-kısıt yöntemi, NSGA-II gibi evrimsel algoritmalar
  - Uygulama alanları: Mühendislik tasarımı, çevre yönetimi, ekonomik modeller

[19]

[19] Bir yapısal optimizasyon problemi, genellikle yukarıdaki kategorilerin birkaçının özelliklerini bir arada taşır. Örneğin, bir köprü tasarımında kesit boyutları sürekli değişkenler olabilirken, kullanılacak eleman tipleri tamsayı değişkenler olabilir. Ayrıca malzeme davranışı doğrusal olmayan denklemlerle ifade edilebilir, ve hem maliyet hem de deplasman gibi birden fazla kriteri optimize etmek gerekebilir. Bu durumda problem, karma tamsayılı, doğrusal olmayan, çok amaçlı bir optimizasyon problemi olur.

2.8.2 Çözüm Stratejisine Göre

Optimizasyon problemlerini çözmek için kullanılan algoritmalar, arama stratejilerine göre kategorize edilebilir:

- **Deterministik Yöntemler**
  - **Gradyan tabanlı yöntemler:** Fonksiyonun türev bilgisini kullanarak en hızlı iyileşme yönünde ilerler.
    - Avantajları: Genellikle hızlı yakınsama, yerel optimuma kesin ulaşma
    - Dezavantajları: Yerel optimumlara takılma, türev hesaplama gereksinimi
    - Örnekler: Gradyan iniş, Newton, BFGS, Conjugate gradient
  - **Doğrudan arama yöntemleri:** Türev bilgisi kullanmadan, fonksiyon değerlendirmeleri ile ilerler.
    - Avantajları: Türev gerektirmez, gürültülü fonksiyonlar için uygun
    - Dezavantajları: Genellikle daha yavaş yakınsama
    - Örnekler: Nelder-Mead simpleks, Hooke-Jeeves pattern search
  - **İç nokta yöntemleri:** Fizibil bölgenin içinde kalarak optimuma ulaşır.
    - Avantajları: Büyük ölçekli problemlerde etkili, polinom zamanlı algoritma
    - Dezavantajları: Karmaşık implementasyon, başlangıç noktası gereksinimi
    - Örnekler: Primal-dual iç nokta, bariyer yöntemleri

- **Stokastik/Metasezgisel Yöntemler**
  - **Evrimsel algoritmalar:** Doğal seleksiyon ve genetik mekanizmaları taklit eder.

    - Avantajları: Global optimumu bulma potansiyeli, türev gerektirmez, paralel işleme uygun
    - Dezavantajları: Hesaplama maliyeti yüksek, parametre ayarı hassas
    - Örnekler: Genetik algoritmalar, diferansiyel evrim, evrim stratejileri
  - **Sürü tabanlı algoritmalar:** Hayvan gruplarının kolektif davranışlarını modelleyerek çözüm arar.
    - Avantajları: Kolay implementasyon, konveks olmayan problemlerde etkili
    - Dezavantajları: Teorik garantiler zayıf, çözüm kalitesi değişken
    - Örnekler: Parçacık sürü optimizasyonu, karınca kolonisi optimizasyonu, arı algoritması
  - **Fizik tabanlı algoritmalar:** Fiziksel süreçleri taklit ederek optimizasyon yapar.
    - Avantajları: Yerel optimumlardan kaçabilme yeteneği
    - Dezavantajları: Yavaş yakınsama, parametre hassasiyeti
    - Örnekler: Tavlama benzetimi, harmony search, big bang-big crunch

- **Hibrit Yöntemler** [20]
  - **Deterministik + Stokastik hibrit:** İki yaklaşımın avantajlarını birleştirir.
    - Avantajları: Global aramayı yerel optimizasyon ile birleştirme
    - Dezavantajları: Karmaşık implementasyon, parametre ayarı zorluğu
    - Örnekler: Memetic algoritmalar, çoklu başlangıç noktalı gradyan yöntemleri
  - **Çok seviyeli yaklaşımlar:** Problemi farklı detay seviyelerinde ele alır.
    - Avantajları: Büyük ve karmaşık problemleri çözebilme yeteneği
    - Dezavantajları: Problem yapısına özgü tasarım gereksinimi
    - Örnekler: Kaba-ince (coarse-fine) grid yöntemleri, hiyerarşik optimizasyon
  - **Adaptif stratejiler:** Optimizasyon süreci boyunca algoritma parametrelerini veya stratejileri değiştirir.
    - Avantajları: Problem özelliklerine dinamik adaptasyon
    - Dezavantajları: Karmaşık kontrol mekanizmaları
    - Örnekler: Adaptif metasezgisel yöntemler, self-adaptive evrim stratejileri [21]

[20] Hibrit yöntemler, farklı yaklaşımların avantajlarını birleştirerek daha gürbüz ve etkili çözümler sunar. Örneğin, global arama için bir genetik algoritma ile başlayıp, bulunan umut verici bölgeleri yerel bir gradyan tabanlı algoritma ile iyileştirmek, hem global optimumu bulma şansını artırır hem de hassas bir çözüme ulaşmayı sağlar.

[21] Yapısal optimizasyon problemlerinde, sonlu eleman analizinin hesaplama maliyeti genellikle yüksek olduğundan, mümkün olduğunca az fonksiyon değerlendirmesi yapan algoritmalar tercih edilir. Bu nedenle, surrogate model tabanlı optimizasyon yöntemleri giderek popülerlik kazanmaktadır. Bu yöntemler, gerçek analiz yerine daha hızlı hesaplanabilen yaklaşık modeller kullanarak optimizasyon sürecini hızlandırır.

Optimizasyon Algoritması Seçimi

Bir optimizasyon problemi için hangi algoritmanın en uygun olduğu, problemin özelliklerine bağlıdır:

- **Problem boyutu:** Büyük ölçekli problemler için İç nokta yöntemleri, gradyan tabanlı yöntemler veya özel tasarlanmış metasezgisel yöntemler uygundur.
- **Türev bilgisi:** Türev hesaplaması mümkün ve ekonomikse, gradyan tabanlı yöntemler genellikle daha hızlıdır. Türev hesaplaması zor veya imkansızsa, doğrudan arama veya metasezgisel yöntemler tercih edilir.
- **Problemin doğası:** Konveks problemler için deterministik yöntemler genellikle yeterlidir. Konveks olmayan, multimodal problemler için metasezgisel veya hibrit yöntemler daha uygundur.
- **Hesaplama bütçesi:** Sınırlı hesaplama kaynakları varsa, daha verimli deterministik yöntemler tercih edilebilir. Yüksek hesaplama gücü mevcutsa, daha kapsamlı global arama yöntemleri kullanılabilir.
- **Fizibil çözüm önemi:** Eğer ara iterasyonlarda da fizibil çözümler gerekiyorsa, fizibil bölge içinde kalan yöntemler (iç nokta, bariyer yöntemleri) tercih edilebilir.

3 Klasik Optimizasyon Algoritmalarının Teorik Temelleri

Geleneksel optimizasyon yöntemleri, modern algoritmaların temel prensiplerini oluşturmakla kalmaz, aynı zamanda mühendislik problemlerinin çözümünde hala başvurulan en güvenilir araçlar arasındadır. Bu bölümde, bu yöntemlerin matematiksel altyapısına derinlemesine bir bakış sunacak ve yapısal optimizasyon problemlerine nasıl uygulandığını inceleyeceğiz.

3.1 Matematiksel Optimizasyonun Analitik Temelleri

Optimizasyon, matematik tarihi boyunca en köklü problemlerden biri olmuştur. Leibniz ve Newton'un diferansiyel hesabı geliştirmesinden önce bile matematikçiler optimizasyon problemleriyle uğraşmaktaydı. Modern anlamda matematiksel optimizasyonun analitik temelleri üç temel kavrama dayanır: stasyonerlik (durağanlık), konvekslik ve dualite.

Tarihsel Perspektif

- **Eski Yunan:** Öklidyen geometri ve maksimum-minimum problemleri
- **17. Yüzyıl:** Fermat, Leibniz ve Newton'un diferansiyel hesabı
- **18. Yüzyıl:** Lagrange ve Euler'in varyasyonel hesaplamaları
- **19. Yüzyıl:** Hamilton ve Jacobi'nin optimizasyon teorileri
- **20. Yüzyıl:** Doğrusal programlama, sayısal optimizasyon ve karmaşık algoritmaların gelişimi

3.1.1 Birinci ve İkinci Dereceden İyileştirme Koşulları

Optimizasyonun matematiksel temelini oluşturan en önemli analitik koşullar şunlardır:

Birinci Dereceden Gerek Koşul (Stasyonerlik): Eğer x^* noktası bir yerel minimum ise, $\nabla f(x^*) = 0$ olmalıdır. Bu, fonksiyonun gradyanının sıfır olduğu, yani teğet düzlemin yatay olduğu anlamına gelir.

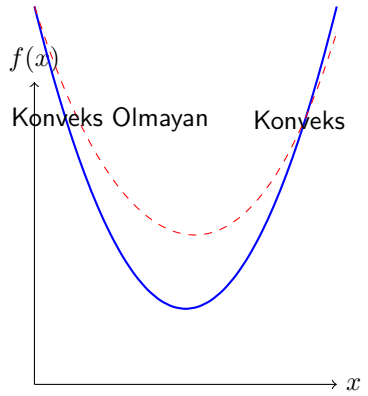
İkinci Dereceden Yeter Koşul: Eğer $\nabla f(x^*) = 0$ ve Hessian matrisi $\nabla^2 f(x^*)$ pozitif tanımlı ise, x^* bir kesin yerel minimumdur. Matematiksel olarak, tüm $d \neq 0$ için $d^T \nabla^2 f(x^*) d > 0$ koşulunun sağlanması gerekir.
[22]

[22] İkinci dereceden koşullar, kritik noktanın (gradyanın sıfır olduğu nokta) yerel minimum, yerel maksimum veya eğer noktası olduğunu belirlememize yardımcı olur. Yapısal optimizasyon problemlerinde Hessian matrisinin değerlendirilmesi, algoritmanın ilerleyişi için kritik öneme sahiptir.

3.1.2 Konveks Optimizasyon Özel Durumu

Konveks optimizasyon, klasik optimizasyon teorisinin özel bir durumudur ve dikkate değer özellikler taşır:

- Amaç fonksiyonu ve kısıt kümesi konveks ise, her yerel minimum aynı zamanda global minimumdur
- Konveks problemlerde kritik nokta koşulları, global optimumu garanti eder
- Yapısal mekanikte doğrusal elastik yapılar için potansiyel enerji minimizasyonu konveks bir problemdir
- Konveks olmayan problemler için bile, problem sıklıkla konveks alt problemlere ayrıştırılabilir



Şekil 13: Konveks ve konveks olmayan fonksiyonların karşılaştırması

3.2 Gradyan Tabanlı Optimizasyon Algoritmaları

Gradyan tabanlı algoritmalar, fonksiyonun türev bilgisini kullanarak sistematik bir şekilde optimum noktaya yaklaşan yöntemlerdir. Bu yöntemler, hesaplama verimliliği ve yakınsama özellikleri açısından yapısal optimizasyon problemlerinde büyük önem taşır.

3.2.1 Gradyan İniş Yöntemi ve Varyasyonları

Gradyan iniş yöntemi, en temel ve en eski gradyan tabanlı optimizasyon yöntemidir. Bu yöntem, fonksiyonun en hızlı azaldığı yönde ilerleyerek minimum noktaya ulaşmayı amaçlar.

$$x_{k+1} = x_k - \alpha_k \nabla f(x_k) \tag{14}$$

Burada α_k, adım boyutunu (veya öğrenme oranını) temsil eder ve algoritmanın performansını büyük ölçüde etkiler.

Gradyan İniş Varyasyonları

- **Sabit Adım Boyutu:** En basit yaklaşım, tüm iterasyonlarda sabit α kullanmaktır. Ancak hızlı yakınsama için çok düşük, stabilite için çok yüksek olabilir.
- **Line Search:** Her iterasyonda, $f(x_k - \alpha \nabla f(x_k))$ ifadesini minimize eden α_k değeri bulunur. Bu yaklaşım, Armijo, Wolfe veya Goldstein koşulları ile uygulanabilir.
- **Momentumlu Gradyan İniş:** Önceki gradyan değerlerini de dikkate alarak, yerel minimumlara takılmayı azaltır:

$$x_{k+1} = x_k - \alpha_k \nabla f(x_k) + \beta(x_k - x_{k-1})$$

- **Nesterov Hızlandırılmış Gradyan:** Momentumlu gradyan inişin geliştirilmiş halidir, gradyanı mevcut konumda değil, momentumun götüreceği noktada hesaplar.

[23]

Yapısal Optimizasyonda Gradyan Hesaplama Yapısal optimizasyon problemlerinde, gradyan hesaplama için genellikle üç yöntem kullanılır:

- **Analitik Gradyan:** Doğrudan diferansiyel hesaplama ile elde edilir. En doğru sonucu verir ancak karmaşık sistemlerde türev çıkarımı zor olabilir.
- **Sonlu Farklar:** Sayısal yaklaşımla gradyan hesaplanır:

$$\frac{\partial f}{\partial x_i} \approx \frac{f(x + he_i) - f(x)}{h}$$

  Hesaplaması kolaydır ancak h değerinin seçimi hassastır ve büyük sistemlerde hesaplama maliyeti yüksektir.
- **Adjoint Yöntemi:** Özellikle büyük ölçekli yapısal problemlerde verimlidir. Tasarım değişkeni sayısından bağımsız olarak sabit sayıda sistem çözümü gerektirir. Sonlu eleman analizlerinde sıklıkla kullanılır.

3.2.2 Newton ve Quasi-Newton Yöntemleri

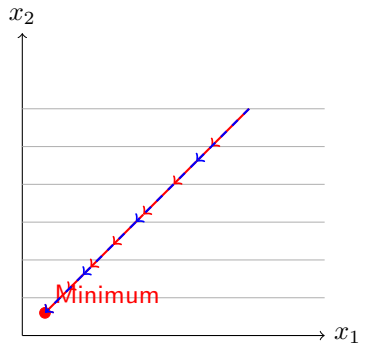
Newton yöntemi, klasik optimizasyonun en güçlü araçlarından biridir. İkinci dereceden türev bilgisini (Hessian matrisini) kullanarak, fonksiyonun yerel kuadratik yaklaşımını oluşturur ve bu yaklaşımın minimum noktasına doğrudan ilerler.

$$x_{k+1} = x_k - [\nabla^2 f(x_k)]^{-1} \nabla f(x_k) \tag{15}$$

Newton yönteminin en büyük avantajı, kuadratik yakınsama hızıdır; yani, optimum noktaya yaklaştıkça, hata her iterasyonda yaklaşık olarak karesi kadar azalır. Ancak, Hessian matrisinin hesaplanması ve tersinin alınması, özellikle yüksek boyutlu problemlerde hesaplama açısından maliyetlidir.

Modifiye Newton Yöntemleri Hesaplama maliyetini azaltmak ve Newton yönteminin kararsız olabileceği durumlarda daha iyi performans elde etmek için çeşitli modifikasyonlar geliştirilmiştir:

[23] Yapısal optimizasyon problemlerinde, gradyan iniş yöntemi genellikle büyük ölçekli problemlerin ilk aşamalarında kullanılır. Yakınsama hızı düşük olsa da, hesaplama maliyeti düşüktür ve karmaşık problemlerde iyi bir başlangıç noktası sağlayabilir.



Şekil 14: Gradyan iniş (kırmızı, düz) ve momentumlu gradyan iniş (mavi, kesikli) yöntemlerinin karşılaştırması

- **Levenberg-Marquardt Algoritması:** Hessian matrisini, daha kararlı hale getirmek için modifiye eder:

$$x_{k+1} = x_k - [\nabla^2 f(x_k) + \lambda I]^{-1} \nabla f(x_k)$$

  Burada λ, algoritmanın davranışını ayarlayan bir damping parametresidir.

- **Trust Region Newton Metodu:** Her iterasyonda, Hessian matrisinin geçerli olduğu bir "güvenilir bölge" tanımlar ve bu bölge içinde kuadratik modeli minimize eder. Özellikle yapısal optimizasyonda yaygın kullanılır.

Quasi-Newton Yöntemleri Tam Hessian matrisini hesaplama maliyetini ortadan kaldırmak için, quasi-Newton yöntemleri, Hessian matrisinin yaklaşık bir değerini iteratif olarak günceller. En popüler quasi-Newton yöntemleri şunlardır:

$$x_{k+1} = x_k - B_k^{-1} \nabla f(x_k) \tag{16}$$

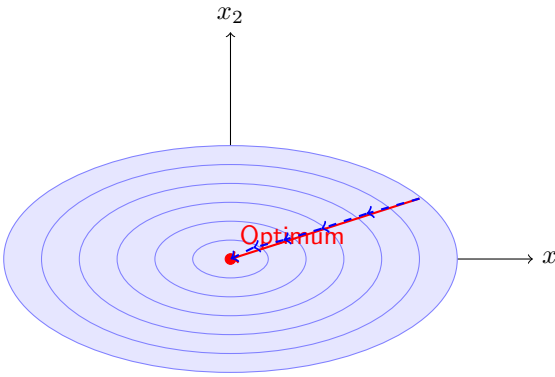
Burada B_k, Hessian matrisinin k. iterasyondaki yaklaşımıdır.

Temel Quasi-Newton Algoritmaları

- **BFGS (Broyden-Fletcher-Goldfarb-Shanno):** En yaygın kullanılan quasi-Newton yöntemidir. Hessian yaklaşımının pozitif tanımlılığını korur, bu da algoritmanın kararlılığını artırır. Yapısal optimizasyon problemlerinde sıklıkla tercih edilir.
- **DFP (Davidon-Fletcher-Powell):** BFGS'den daha eski bir yöntemdir, ancak genellikle BFGS kadar kararlı değildir.
- **SR1 (Symmetric Rank-One):** Daha az hesaplama gerektirir ancak pozitif tanımlılığı garanti etmez.
- **L-BFGS (Limited-memory BFGS):** Çok yüksek boyutlu problemler için geliştirilmiş, bellek kullanımı optimize edilmiş BFGS versiyonudur. Büyük ölçekli yapısal optimizasyon problemlerinde, özellikle topoloji optimizasyonunda tercih edilir.

Yapısal Optimizasyonda Newton ve Quasi-Newton Uygulamaları Yapısal mühendislikte, Newton ve quasi-Newton yöntemleri şu tür problemlerde sıklıkla kullanılır:

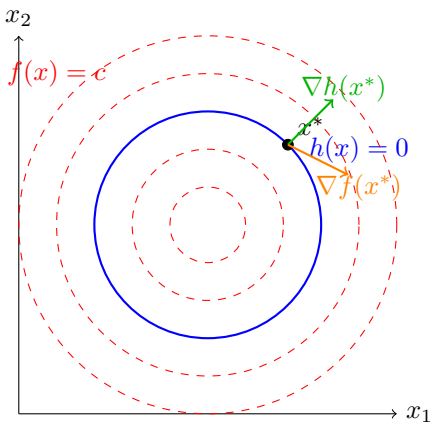
- **Şekil Optimizasyonu:** Yapının dış geometrisinin optimize edilmesinde, özellikle aerodinamik performans veya termal davranış için
- **Kesit Optimizasyonu:** Kafes ve çerçeve yapıların eleman kesitlerinin optimizasyonunda, özellikle doğrusal olmayan davranış gösteren yapılarda
- **Parametre Kalibrasyonu:** Sonlu eleman modellerinin deneysel verilerle kalibrasyonunda, malzeme parametrelerinin belirlenmesinde
- **Multi-fizik Optimizasyon:** Yapısal, termal ve akışkan davranışların birlikte optimize edildiği karmaşık mühendislik problemlerinde



Şekil 15: Newton yöntemi (kırmızı, düz) ve gradyan iniş yöntemi (mavi, kesikli) karşılaştırması. Newton yöntemi, kuadratik fonksiyonlarda tek adımda optimuma ulaşabilir.

3.3 Kısıtlı Optimizasyon ve Dualite Teorisi

Mühendislik problemleri nadiren kısıtsız olarak karşımıza çıkar. Yapısal tasarımda gerilme limitleri, deplasman sınırları, fiziksel kısıtlamalar ve kaynak sınırlamaları gibi pek çok kısıt söz konusudur. Bu bölümde, kısıtlı optimizasyon problemlerinin çözümüne yönelik klasik yaklaşımları derinlemesine inceleyeceğiz.



Şekil 16: Lagrange çarpanları yönteminin geometrik yorumu: Optimum noktada, amaç fonksiyonunun seviye eğrisi, kısıt fonksiyonunun seviye eğrisine teğettir.

3.3.1 Lagrange Çarpanları Yöntemi ve Teorik Temelleri

Lagrange çarpanları yöntemi, eşitlik kısıtlı optimizasyon problemlerinin çözümü için geliştirilen temel bir yaklaşımdır. Joseph-Louis Lagrange'ın 18. yüzyılda geliştirdiği bu yöntem, optimizasyon teorisinin ve variasyonel hesaplamanın köşe taşlarından biridir.

Eşitlik kısıtlı bir optimizasyon problemi şu şekilde ifade edilir:

$$\begin{aligned} \min \quad & f(x) \\ \text{s.t.} \quad & h_j(x) = 0, \quad j = 1, \ldots, p \end{aligned} \tag{17}$$

Lagrange fonksiyonu (veya Lagrangian) şu şekilde tanımlanır:

$$\mathcal{L}(x, \lambda) = f(x) + \sum_{j=1}^{p} \lambda_j h_j(x) \tag{18}$$

Burada λ_j değerleri, Lagrange çarpanları olarak adlandırılır ve her bir kısıtın "gölge fiyatını" veya marjinal değerini temsil eder.
[24]

Lagrange Koşulları Kısıtlı bir optimizasyon probleminin yerel minimumu için gerek koşullar şunlardır:

$$\begin{aligned} \nabla_x \mathcal{L}(x^*, \lambda^*) &= \nabla f(x^*) + \sum_{j=1}^{p} \lambda_j^* \nabla h_j(x^*) = 0 \\ \nabla_\lambda \mathcal{L}(x^*, \lambda^*) &= h_j(x^*) = 0, \quad j = 1, \ldots, p \end{aligned} \tag{19}$$

Bu koşullar, optimal noktada amaç fonksiyonunun gradyanının, kısıt fonksiyonlarının gradyanlarının doğrusal bir kombinasyonuna eşit olması gerektiğini ifade eder. Geometrik olarak, bu, $f(x)$ ve $h_j(x)$ fonksiyonlarının seviye eğrilerinin teğet olması anlamına gelir.

İkinci Dereceden Koşullar Bir kritik noktanın gerçekten minimum olup olmadığını belirlemek için, genişletilmiş Hessian matrisinin incelenmesi gerekir. Eğer bu matris, kısıtların teğet uzayında pozitif tanımlı ise, kritik nokta bir yerel minimum olarak kabul edilir.

[24] Yapısal mühendislikte Lagrange çarpanları, çoğu zaman fiziksel bir anlam taşır. Örneğin, kuvvet dengesini kısıt olarak kullandığımız bir optimizasyon probleminde, Lagrange çarpanları genellikle yer değiştirmelere karşılık gelir. Bu dualite, sonlu eleman analizinde ve yapısal optimizasyonda temel bir kavramdır.

Lagrange Yönteminin Yapısal Optimizasyondaki Uygulamaları

- **Çelik Çerçeve Yapılar:** Ağırlık minimizasyonu yaparken, düğüm noktalarında kuvvet dengesi ve eleman uygunluğu kısıtlarının sağlanması
- **Kompozit Malzeme Tasarımı:** Rijitlik maksimizasyonu yaparken, hacim kısıtı ve malzeme dengesi koşullarının sağlanması
- **Kafes Sistemler:** Ağırlık minimizasyonu yaparken, izostatik denge koşullarının sağlanması
- **Sonlu Eleman Analizi:** Enerji minimizasyonu prensibine dayalı sonlu eleman formülasyonlarında, kinematik uygunluk ve kuvvet dengesi kısıtlarının sağlanması

3.3.2 Karush-Kuhn-Tucker (KKT) Koşulları ve Eşitsizlik Kısıtları

Karush-Kuhn-Tucker (KKT) koşulları, Lagrange çarpanları yönteminin eşitsizlik kısıtlı problemlere genelleştirilmiş halidir. Bu koşullar, nonlineer programlama alanında optimum çözümlerin karakterizasyonu için temel bir çerçeve sunar.

Eşitsizlik kısıtlı bir optimizasyon problemi şu şekilde ifade edilir:

$$\begin{aligned} \min \quad & f(x) \\ \text{s.t.} \quad & g_i(x) \leq 0, \quad i = 1, \ldots, m \\ & h_j(x) = 0, \quad j = 1, \ldots, p \end{aligned} \tag{20}$$

Lagrange fonksiyonu bu durumda:

$$\mathcal{L}(x, \lambda, \mu) = f(x) + \sum_{i=1}^{m} \mu_i g_i(x) + \sum_{j=1}^{p} \lambda_j h_j(x) \tag{21}$$

KKT Koşulları Eşitsizlik kısıtlı bir problemin yerel minimumu için gerekli koşullar (KKT koşulları) şunlardır:

$$\begin{aligned} \nabla f(x^*) + \sum_{i=1}^{m} \mu_i^* \nabla g_i(x^*) + \sum_{j=1}^{p} \lambda_j^* \nabla h_j(x^*) = 0 & \\ g_i(x^*) \leq 0, \quad i = 1, \ldots, m & \\ h_j(x^*) = 0, \quad j = 1, \ldots, p & \\ \mu_i^* \geq 0, \quad i = 1, \ldots, m & \\ \mu_i^* g_i(x^*) = 0, \quad i = 1, \ldots, m & \end{aligned} \tag{22}$$

Son koşul "tamamlayıcı gevşeklik" olarak bilinir ve bir kısıtın ya aktif olması (eşitlik olarak sağlanması) ya da ilgili Lagrange çarpanının sıfır olması gerektiğini ifade eder.

KKT Koşullarının Yorumlanması

- **Stasyonerlik:** Amaç fonksiyonunun gradyanının, aktif kısıtların gradyanlarının doğrusal kombinasyonu olarak ifade edilmesi. Bu, optimum noktada ilerlemenin, en az bir aktif kısıtı ihlal etmeden mümkün olmadığını gösterir.
- **Uygunluk:** Tüm kısıtların sağlanması.
- **Çift Uygunluk:** Eşitsizlik kısıtları için Lagrange çarpanlarının (Kuhn-Tucker çarpanları) negatif olmaması. Bu, kısıtların yönünün önemli olduğunu gösterir.
- **Tamamlayıcı Gevşeklik:** Herhangi bir kısıt aktif değilse (eşitsizlik olarak sağlanıyorsa), ilgili Lagrange çarpanı sıfır olmalıdır. Bu, kısıtın "etkisiz" olduğu anlamına gelir.

KKT Koşullarının Yapısal Optimizasyondaki Önemi Yapısal optimizasyon problemleri genellikle çok sayıda eşitsizlik kısıtı içerir. Örneğin:

- Gerilme kısıtları: $\sigma_i \leq \sigma_{izin}$
- Deplasman kısıtları: $|u_i| \leq u_{izin}$
- Burkulma kısıtları: $P_i \leq P_{cr,i}$
- Geometrik kısıtlar: minimum kalınlık, genişlik vb.

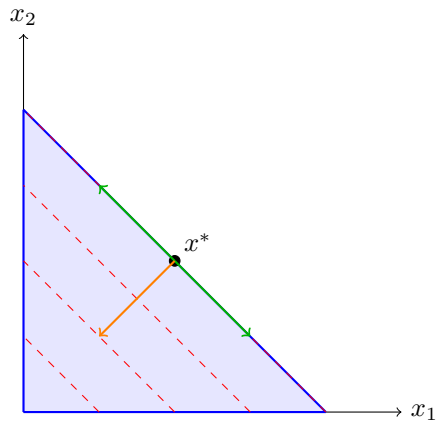
Bu tür problemlerde, KKT koşulları sadece matematiksel bir araç değil, aynı zamanda fiziksel anlamı olan bir çerçeve sunar. Örneğin, gerilme kısıtına ait Lagrange çarpanı, ilgili noktada birim gerilme artışının, optimal tasarımın ağırlığına etkisini gösterir. Bu, "tasarım duyarlılığı analizi" için temel bir kavramdır.

Dualite Teorisi ve Ekonomik Yorum Lagrange dualitesi, birincil problem ile onun dual problemi arasındaki ilişkiyi inceler. Dual problem, birincil problemin Lagrange fonksiyonunun minimizasyonu yerine maksimizasyonunu içerir:

$$\max_{\mu \geq 0, \lambda} \min_{x} \mathcal{L}(x, \lambda, \mu) \tag{23}$$

Dualite teoremi, konveks bir birincil problem için, dual problemin optimal değerinin, birincil problemin optimal değerine eşit olduğunu ifade eder (güçlü dualite). Bu teorem, pek çok optimizasyon algoritmasının temelini oluşturur.

Ekonomik açıdan, dual değişkenler (Lagrange çarpanları) "gölge fiyatları" temsil eder. Yapısal optimizasyonda bu, bir kısıtın marjinal değişiminin, optimal tasarımın değerine etkisini gösterir. Bu yorum, mühendislere hangi kısıtların tasarım üzerinde en büyük etkiye sahip olduğunu anlama imkanı verir.



Şekil 17: KKT koşullarının geometrik yorumu: Optimum noktada, amaç fonksiyonunun negatif gradyanı, aktif kısıtların gradyanlarının konveks konisinde yer alır.

Dualite ve Yapısal Analiz Arasındaki İlişki

Yapısal mekanikte, potansiyel enerji minimizasyonu (deplasman yöntemi) ve tamamlayıcı enerji minimizasyonu (kuvvet yöntemi) arasındaki ilişki, optimizasyon teorisindeki dualite kavramına mükemmel bir örnektir. Bir yapının analizi için:

- **Birincil Problem (Deplasman Yöntemi):** Uyumlu deplasman alanını bulmak için potansiyel enerjiyi minimize etmek
- **Dual Problem (Kuvvet Yöntemi):** Dengedeki kuvvet dağılımını bulmak için tamamlayıcı enerjiyi minimize etmek

Bu dualite, yapısal optimizasyon algoritmalarının tasarımında da kullanılır. Özellikle "primal-dual" yöntemler, hem birincil hem de dual problemi eşzamanlı olarak çözerek, daha verimli optimizasyon sağlar.

3.4 Doğrusal ve Kuadratik Programlama

Klasik optimizasyon yöntemlerinin önemli bir alt kümesi, doğrusal ve kuadratik programlama yöntemleridir. Bu yöntemler, belirli yapıdaki optimizasyon problemleri için özel olarak geliştirilmiş, verimli çözüm algoritmaları sunar.

3.4.1 Doğrusal Programlama ve Yapısal Uygulamaları

Doğrusal programlama (DP), hem amaç fonksiyonu hem de kısıtların doğrusal olduğu optimizasyon problemlerini inceler. Standart form:

$$\begin{aligned} \min \quad & c^T x \\ \text{s.t.} \quad & Ax = b \\ & x \geq 0 \end{aligned} \tag{24}$$

Doğrusal programlama, 1940'larda George Dantzig'in Simplex algoritmasını geliştirmesiyle büyük bir ilerleme kaydetmiştir. Günümüzde, büyük ölçekli doğrusal programlama problemleri için İç Nokta Yöntemleri de yaygın olarak kullanılmaktadır.

Simplex Yöntemi ve Geometrik Yorumu Simplex yöntemi, uygun bölgenin köşe noktalarını (uç noktaları) sistematik olarak inceleyerek optimum çözümü bulmayı amaçlar. DP'nin temel teoremi, optimal çözümün uygun bölgenin bir köşe noktasında olması gerektiğini ifade eder (eğer çözüm tekil değilse).

Bu yöntemin adımları:

- Başlangıç için bir uygun köşe noktası bulunur
- Amaç fonksiyonu değerini iyileştirecek komşu köşe noktasına geçilir
- Daha iyi bir komşu köşe bulunamayana kadar devam edilir

Yapısal Mühendislikte DP Uygulamaları

- **Plastik Limit Analizi:** Yapıların göçme yükünün belirlenmesi, plastik mafsal dağılımının optimizasyonu
- **Kafes Sistemlerin Minimum Ağırlık Tasarımı:** Eleman kesitlerinin optimizasyonunda, lineer davranış ve statik yük koşulları altında
- **Yapısal Kaynak Dağıtımı:** Sınırlı malzeme veya bütçe koşullarında, yapısal performansı maksimize edecek kaynak dağılımı
- **Ulaşım Ağı Optimizasyonu:** Köprü ve yol sistemlerinin yerleşiminin optimizasyonu, trafik akışının modellenmesi

Plastik Limit Analizi Örneği Plastik limit analizi, doğrusal programlamanın yapısal mühendislikteki en önemli uygulamalarından biridir. Statik teorem (alt sınır) formülasyonu:

$$\begin{aligned} \max \quad & \lambda \\ \text{s.t.} \quad & B^T q = \lambda F + F_0 \\ & |q_i| \leq q_i^p, \quad i = 1, \ldots, n \end{aligned} \tag{25}$$

Burada:

- λ: yük çarpanı
- q: iç kuvvetler vektörü
- B: denge matrisi
- F: değişken dış yük vektörü
- F_0: sabit dış yük vektörü
- q_i^p: eleman i için plastik limit kapasitesi

Bu formülasyon, yapının göçmeden dayanabileceği maksimum yük çarpanını belirler.
[25]

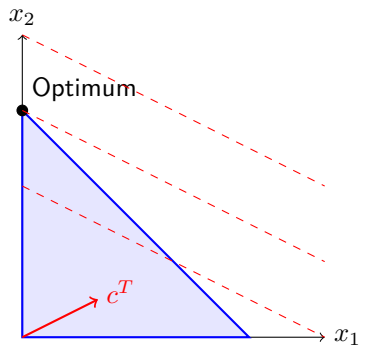
3.4.2 Kuadratik Programlama

Kuadratik programlama (KP), amaç fonksiyonunun ikinci dereceden, kısıtların ise doğrusal olduğu optimizasyon problemlerini ele alır:

$$\begin{aligned} \min \quad & \frac{1}{2} x^T Q x + c^T x \\ \text{s.t.} \quad & Ax = b \\ & x \geq 0 \end{aligned} \tag{26}$$

Burada Q, simetrik bir matristir. Eğer Q pozitif (yarı) tanımlı ise, problem konvekstir ve global optimumu bulmak göreceli olarak kolaydır.

[25] Plastik limit analizi, özellikle çelik yapıların tasarımında önemlidir. Yapının elastik sınırın ötesinde, plastik deformasyon kapasitesini kullanarak dayanabildiği maksimum yükü belirleyerek, daha ekonomik tasarımlar elde edilmesini sağlar.



Şekil 18: Doğrusal programlamada uygun bölge ve optimum nokta

Çözüm Yöntemleri Kuadratik programlama problemleri için çeşitli çözüm yöntemleri geliştirilmiştir:

- **Aktif Set Yöntemleri:** Hangi kısıtların aktif olduğuna dair tahminler yaparak, kısıtsız optimizasyon alt problemlerini çözer
- **İç Nokta Yöntemleri:** Uygun bölgenin içinden geçerek, optimum noktaya yaklaşır
- **Sıralı Kuadratik Programlama (SQP):** Nonlineer optimizasyon problemlerini, kuadratik alt problemlere bölerek çözer

Yapısal Mühendislikte KP Uygulamaları

- **Elastik Deformasyon Minimizasyonu:** Yapının rijitlik matrisini kullanarak, belirli yükler altında deformasyonu minimize etmek
- **Dinamik Davranış Optimizasyonu:** Yapının kütle ve rijitlik matrislerini kullanarak, dinamik tepkiyi optimize etmek
- **Sonlu Eleman Modeli Kalibrasyonu:** Deneysel ve sayısal veriler arasındaki farkın karelerini minimize etmek
- **Ağırlık ve Deplasman Optimizasyonu:** Hem ağırlığı hem de deplasman enerjisini içeren çok amaçlı optimizasyon

Yapısal Esneklik (Compliance) Minimizasyonu Örneği Yapısal tasarımda sıkça karşılaşılan bir kuadratik programlama problemi, belirli bir hacim kısıtı altında esnekliğin (veya deformasyon enerjisinin) minimizasyonudur:

$$\begin{aligned} \min \quad & \frac{1}{2} F^T u \\ \text{s.t.} \quad & Ku = F \\ & \sum_{e=1}^{n_e} v_e \rho_e \leq V \\ & 0 \leq \rho_e \leq 1, \quad e = 1, \ldots, n_e \end{aligned} \tag{27}$$

Burada:

- u: deplasman vektörü
- F: kuvvet vektörü
- K: global rijitlik matrisi
- ρ_e: eleman e için malzeme yoğunluğu (tasarım değişkeni)
- v_e: eleman e'nin hacmi
- V: izin verilen toplam hacim

Bu formülasyon, topoloji optimizasyonunun temelini oluşturur ve SIMP (Solid Isotropic Material with Penalization) metodunun başlangıç noktasıdır.

3.5 Modern Klasik Optimizasyon Algoritmaları

Klasik optimizasyon yöntemlerinin modern uzantıları, çeşitli yaklaşımlar kullanarak büyük ve karmaşık yapısal optimizasyon problemlerini çözmeyi amaçlar. Bu bölümde, klasik yöntemlerin daha gelişmiş versiyonlarını ele alacağız.

3.5.1 Sekant Yöntemleri ve Yapısal Uygulamaları

Sekant yöntemleri, Newton yönteminin Hessian matrisini hesaplama gereksinimini ortadan kaldırmak için geliştirilen yaklaşımlardır. Bu yöntemler, gradyan bilgisini kullanarak Hessian matrisinin yaklaşık bir tahminini oluşturur.

En yaygın sekant yöntemleri şunlardır:

- **Broyden Yöntemi:** Genel nonlineer denklem sistemlerinin çözümünde kullanılır
- **DFP (Davidon-Fletcher-Powell) Yöntemi:** Quasi-Newton yöntemlerinin ilk örneklerindendir
- **BFGS (Broyden-Fletcher-Goldfarb-Shanno) Yöntemi:** Yapısal optimizasyon dahil birçok alanda standart haline gelmiştir

BFGS yönteminde, Hessian matrisinin yaklaşık değeri şu şekilde güncellenir:

$$B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{y_k y_k^T}{y_k^T s_k} \tag{28}$$

Burada:

- $s_k = x_{k+1} - x_k$
- $y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$

BFGS yöntemi, yapısal optimizasyon problemlerinde, özellikle çok sayıda tasarım değişkeni söz konusu olduğunda yaygın kullanılır.

BFGS'in Yapısal Optimizasyondaki Avantajları

- **Bellek Verimliliği:** L-BFGS varyantı ile büyük ölçekli problemlerde bile uygulanabilir
- **Süper Lineer Yakınsama:** Optimum noktaya yaklaştıkça, yakınsama hızı artar
- **Sayısal Stabilite:** Newton yöntemine göre daha kararlıdır
- **Line Search ile Entegrasyon:** Güçlü line search stratejileriyle birleştirilebilir

3.5.2 Trust Region ve Line Search Stratejileri

Gradyan tabanlı optimizasyon yöntemlerinin performansı, adım boyutu seçimine büyük ölçüde bağlıdır. Line search ve trust region, adım boyutunu belirlemek için geliştirilen iki temel stratejidir.

Line Search Stratejileri Line search yöntemlerinde, arama yönü d_k belirlendikten sonra, uygun adım boyutu α_k bulunur:

$$x_{k+1} = x_k + \alpha_k d_k \tag{29}$$

Optimal adım boyutunu belirlemek için çeşitli kriterler kullanılır:

- **Armijo Koşulu:** Adım boyutunun, fonksiyon değerinde yeterli azalma sağlamasını garantiler
- **Wolfe Koşulları:** Armijo koşuluna ek olarak, gradyan değişiminin belirli bir oranı sağlamasını gerektirir
- **Goldstein Koşulları:** Hem fonksiyon değerinde azalma hem de adım boyutunun çok küçük olmamasını sağlar

Trust Region Yaklaşımı Trust region yöntemlerinde, modelin güvenilir olduğu bir bölge tanımlanır ve bu bölge içinde model minimize edilir:

$$\begin{aligned} \min \quad & m_k(p) = f_k + g_k^T p + \frac{1}{2} p^T B_k p \\ \text{s.t.} \quad & \|p\| \leq \Delta_k \end{aligned} \tag{30}$$

Burada:

- $m_k(p)$: k. iterasyonda fonksiyonun kuadratik modeli
- $f_k = f(x_k)$, $g_k = \nabla f(x_k)$
- B_k: Hessian matrisi veya yaklaşımı
- Δ_k: trust region yarıçapı

Her iterasyonda, model tahmini ile gerçek fonksiyon değişimi karşılaştırılarak, trust region yarıçapı güncellenir.

Yapısal Mühendislikte Trust Region Uygulamaları

Trust region yaklaşımı, yapısal mühendislikte özellikle şu durumlarda tercih edilir:

- **Kötü Koşullu Problemler:** Rijitlik matrisinin koşul sayısının yüksek olduğu durumlarda
- **Doğrusal Olmayan Davranış:** Malzeme veya geometrik nonlineerite içeren yapısal analizlerde
- **Kararsız Sistemler:** Burkulma analizi gibi kararsızlık noktalarına yakın problemlerde
- **Çoklu Fizik Analizleri:** Termal-yapısal, akışkan-yapı etkileşimi gibi karmaşık multifizik problemlerde

3.5.3 Sıralı Kısıt Programlama

Sıralı Kısıt Programlama (Sequential Constraint Programming - SCP), yapısal optimizasyon problemlerinin çözümü için geliştirilen etkili bir yaklaşımdır. Bu yöntem, nonlineer problemi, bir dizi doğrusal alt probleme dönüştürerek çözer.

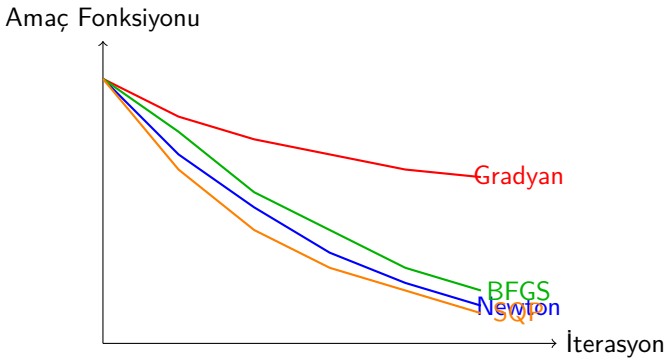
SCP'nin temel yaklaşımı:

- Nonlineer kısıtları, mevcut iterasyon noktası etrafında doğrusallaştırır
- Doğrusallaştırılmış problemi çözer
- Çözümü yeni bir iterasyon noktası olarak kullanır
- Yakınsama sağlanana kadar tekrarlar

Bu yöntem, MMA (Method of Moving Asymptotes) gibi daha gelişmiş varyantlarıyla, yapısal topoloji optimizasyonunda yaygın olarak kullanılmaktadır.

MMA'nın yapısal optimizasyondaki başlıca avantajları:

- Büyük ölçekli topoloji optimizasyonu problemlerinde verimlilik
- Doğrusal olmayan kısıtları etkin bir şekilde ele alabilme
- Salınımları azaltarak kararlı yakınsama sağlama
- Paralel hesaplamaya uygunluk



Şekil 19: Farklı optimizasyon algoritmalarının yakınsama davranışlarının karşılaştırması

3.6 Sonuç ve Modern Yöntemlere Geçiş

Klasik optimizasyon yöntemleri, yapısal mühendislikte karşılaşılan çeşitli problemlerin çözümünde hala büyük önem taşımaktadır. Bu yöntemler, modern meta-sezgisel algoritmaların ve yapay zeka tabanlı yaklaşımların temelini oluşturur.

Klasik ve Modern Yöntemlerin Karşılaştırması

- **Klasik Yöntemler:** Matematiksel sağlamlık, kesin yakınsama garantileri, verimlilik
- **Modern Yöntemler:** Çok modlu fonksiyonlar için global arama, paralel hesaplama, karmaşık kısıtların ele alınmasında esneklik

Günümüzde, klasik algoritmaların güçlü yönleri ile modern yaklaşımların esnekliğini birleştiren hibrit yöntemler büyük ilgi görmektedir. Bu hibrit yaklaşımlar, özellikle büyük ölçekli ve çok amaçlı yapısal optimizasyon problemlerinde etkili çözümler sunmaktadır.

4 Benchmark Test Fonksiyonları

Bu başlık altında incelenen test fonksiyonlarına ait python kodlarına bağlantıdan ulaşılabilir. [26]

Optimizasyon algoritmalarının performansını değerlendirmek ve karşılaştırmak için kullanılan standart test fonksiyonları[27], farklı zorluk derecelerine sahip matematiksel yapılar olarak karşımıza çıkar. Bu test fonksiyonları, bilinen global minimum noktaları, karmaşık yerel minimum yapıları ve çeşitli topolojik özellikleri ile algoritmaları sınamak için tasarlanmıştır.

4.1 Benchmark Fonksiyonlarının Optimizasyondaki Rolü

Optimizasyon algoritmalarının performansını değerlendirmek için standart test fonksiyonları kullanılır. Bu fonksiyonlar şu açılardan önem taşır:

- **Karşılaştırılabilirlik:** Farklı algoritmaların aynı problem üzerindeki başarısını objektif olarak değerlendirmeyi sağlar.
- **Güçlü ve Zayıf Yönleri Belirleme:** Algoritmaların hangi tür problemlerde iyi performans gösterdiğini, hangi tür problemlerde zorlandığını gösterir.

26

27 Test fonksiyonları, algoritmaların güçlü ve zayıf yönlerini ortaya çıkarmada önemli rol oynar. Bir algoritmanın gerçek dünya problemlerindeki başarısı, önce bu test fonksiyonları üzerinde değerlendirilir.

- **Algoritma Geliştirme:** Yeni optimizasyon algoritmalarının geliştirilmesi ve iyileştirilmesi sürecinde rehberlik eder.
- **Gerçek Dünya Uygulamalarına Hazırlık:** Algoritmaların daha karmaşık gerçek dünya problemlerine uygulanmadan önce test edilmesini sağlar.

İdeal bir benchmark fonksiyonu, gerçek dünyadaki optimizasyon problemlerinin karmaşıklığını yansıtırken, matematiksel olarak anlaşılabilir ve analiz edilebilir olmalıdır. Benchmark fonksiyonları genellikle şu özelliklere göre sınıflandırılır: doğrusallık, modalite (tepe nokta sayısı), süreklilik, türevlenebilirlik, sınırlılık ve boyutsallık.

4.1.1 Benchmark Fonksiyonlarının Ortak Özellikleri

Çoğu benchmark fonksiyonu aşağıdaki özelliklere sahiptir:

- Bilinen global minimum değeri ve konumu
- Matematiksel olarak tanımlanmış yapı
- Ayarlanabilir boyut (genellikle çok boyutlu uzaylara genişletilebilir)
- Çeşitli zorluklar (örn. yerel minimumlar, düz bölgeler, dik vadiler)
- Önerilen arama aralıkları

4.2 Çok Modlu ve Tek Modlu Test Fonksiyonları

Test fonksiyonları, içerdikleri tepe noktası (mod) sayısına göre tek modlu (unimodal) ve çok modlu (multimodal) olarak ikiye ayrılır.

4.2.1 Tek Modlu Fonksiyonlar

Tek modlu fonksiyonlar, yalnızca bir tane global minimuma sahiptir ve genellikle daha basit yapıdadır. Bu fonksiyonlar, algoritmanın yakınsama hızını ve hassasiyetini test etmek için kullanılır. Örnek olarak:

- **Sphere Fonksiyonu:** Matematiksel olarak en basit optimizasyon test fonksiyonudur ve şu şekilde tanımlanır:

$$f(\mathbf{x}) = \sum_{i=1}^{n} x_i^2 \tag{31}$$

  Global minimum $f(\mathbf{x}^*) = 0$, $\mathbf{x}^* = (0, 0, \ldots, 0)$ noktasında yer alır.

- **Booth Fonksiyonu:** İki boyutlu, çanak şeklinde bir fonksiyondur:

$$f(x, y) = (x + 2y - 7)^2 + (2x + y - 5)^2 \tag{32}$$

  Global minimum $f(1, 3) = 0$ noktasındadır.

4.2.2 Çok Modlu Fonksiyonlar

Çok modlu fonksiyonlar, birden fazla yerel minimuma sahiptir ve bu nedenle algoritmaların yerel minimumlara takılmadan global minimumu bulabilme yeteneğini test eder. Bu fonksiyonlar, özellikle metasezgisel ve evrimsel algoritmaların performansını değerlendirmek için önemlidir. Önemli örnekler arasında:

- **Ackley Fonksiyonu:** Çok sayıda yerel minimumu olan karmaşık bir test fonksiyonudur. Matematiksel olarak şöyle tanımlanır:

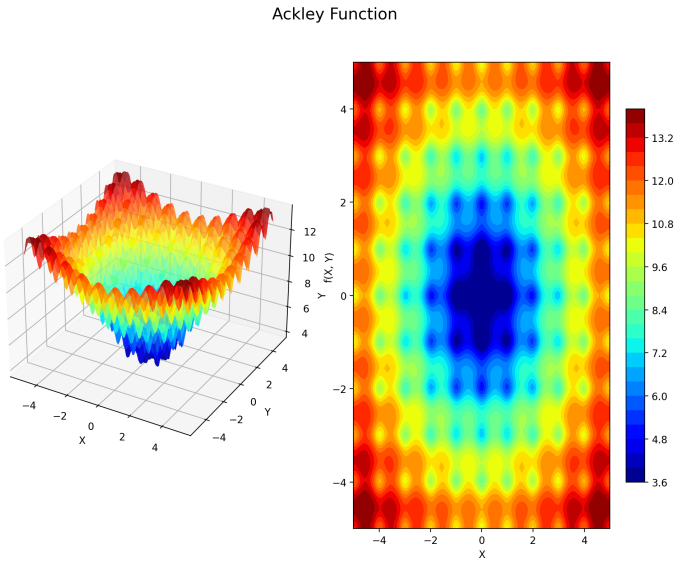
$$f(\mathbf{x}) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e \tag{33}$$

  Bu fonksiyonun global minimumu $f(\mathbf{x}^*) = 0$, $\mathbf{x}^* = (0, 0, \ldots, 0)$ noktasında bulunur. Genellikle $[-32.768, 32.768]$ aralığında tanımlanır.

- **Rastrigin Fonksiyonu:** Sinüzoidal modülasyonu ile çok sayıda yerel minimuma sahip zorlayıcı bir fonksiyondur:

$$f(\mathbf{x}) = 10n + \sum_{i=1}^{n}\left[x_i^2 - 10\cos(2\pi x_i)\right] \tag{34}$$

  Global minimum $f(\mathbf{x}^*) = 0$, $\mathbf{x}^* = (0, 0, \ldots, 0)$ noktasındadır.



Şekil 20: Ackley fonksiyonunun 3B gösterimi (soldaki) ve kontur grafiği (sağdaki). Fonksiyonun merkezde bir global minimum ve etrafında çok sayıda yerel minimum içeren yapısı dikkat çekicidir.

4.2.3 Ackley Fonksiyonu: Derinlemesine İnceleme

Ackley fonksiyonu, David Ackley tarafından 1987 yılında önerilmiş ve optimizasyon algoritmaları için standart bir test fonksiyonu haline gelmiştir. Fonksiyonun matematiksel yapısı şu özellikleri ortaya çıkarır:

- **Neredeyse düz bölgeler:** Merkezden uzaklaştıkça fonksiyon neredeyse düz bir yapıya sahiptir, bu da gradyan tabanlı yöntemlerin doğru yönü belirlemesini zorlaştırır.

- **Periyodik dalgalanmalar:** Kosinüs terimi nedeniyle, fonksiyon yüzeyi düzenli aralıklarla iniş ve çıkışlar gösterir, çok sayıda yerel minimum oluşturur.

- **Merkezdeki derin çukur:** Global minimum etrafında dik bir çukur bulunur, bu da algoritmanın optimuma yakın olduğunda hassas adımlar atmasını gerektirir.

Ackley fonksiyonu, özellikle şu tür algoritmaların test edilmesinde etkilidir:

- Yerel aramayla global arama stratejilerini dengeleyen metasezgisel algoritmalar
- Çok sayıda yerel minimumdan kaçabilme yeteneğine sahip algoritmalar
- Farklı çözüm bölgelerini aynı anda keşfedebilen popülasyon tabanlı algoritmalar

n-boyutlu Ackley fonksiyonunun optimizasyonu şu zorlukları içerir:

- Boyut arttıkça yerel minimum sayısı üstel olarak artar
- Merkeze yakın bölgelerde yüksek hassasiyet gerektirir
- Düz bölgelerde gradyan bilgisi yetersiz kalır

Ackley Fonksiyonu Optimizasyon Zorluğu

Ackley fonksiyonu, hem keşif (exploration) hem de yararlanma (exploitation) özelliklerini aynı anda test eder:

- Keşif: Geniş, neredeyse düz bölgelerde doğru yönü bulabilme
- Yararlanma: Merkezdeki dik çukurda hassas ayarlamaları yapabilme
- Denge: Yerel minimumlardan kaçarken global minimuma yakınsayabilme

4.3 Yüksek Boyutlu Optimizasyon Problemleri

4.3.1 Boyut Artışının Etkileri

- Arama uzayı üstel olarak büyür
- Hesaplama maliyeti artar
- Lokal minimum sayısı artar
- Yakınsama zorlaşır

Yüksek boyutlu optimizasyon problemleri, gerçek dünyadaki birçok mühendislik uygulamasında karşımıza çıkar. Boyut artışı, "boyutun laneti" (curse of dimensionality) olarak bilinen fenomene yol açar. Bu fenomen, boyut arttıkça arama uzayının üstel olarak büyümesi ve algoritmaların etkinliğinin dramatik biçimde azalması ile karakterize edilir.

4.3.2 Boyutun Lanetinin Etkileri

Boyut artışının optimizasyon süreci üzerindeki etkileri:

- **Arama uzayı genişliği:** n boyutlu ve her boyutta m ayrık nokta içeren bir problemde, toplam arama uzayı m^n büyüklüğündedir. Örneğin, her boyutta 10 nokta için, 2 boyutlu problemde 100 nokta, 10 boyutlu problemde 10^{10} nokta, 100 boyutlu problemde 10^{100} nokta vardır.

- **Veri seyrekliği:** Yüksek boyutlarda, veri noktaları arasındaki mesafeler artar ve veri seyrekleşir. Bu, algoritmaların doğru yönü belirlemesini zorlaştırır.
- **Örnekleme zorluğu:** Yüksek boyutlu uzayı yeterince örneklemek için gereken nokta sayısı, pratikte ulaşılamayacak kadar büyüktür.

4.3.3 Yüksek Boyutlu Test Fonksiyonları

Bazı test fonksiyonları özellikle yüksek boyutlu problemlerde algoritmaların performansını değerlendirmek için tasarlanmıştır:

- **Rosenbrock Fonksiyonu:** Dar bir vadi boyunca ilerleyen ve boyut arttıkça zorlaşan bir fonksiyondur:

$$f(\mathbf{x}) = \sum_{i=1}^{n-1} \left[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\right] \tag{35}$$

- **Schwefel Fonksiyonu:** Yüksek boyutlarda çok sayıda geniş bölgeli yerel minimuma sahiptir:

$$f(\mathbf{x}) = 418.9829n - \sum_{i=1}^{n} x_i \sin(\sqrt{|x_i|}) \tag{36}$$

4.4 Stokastik Algoritmaların Test Edilmesi

Stokastik algoritmalar, deterministik olmayan, rastgele elemanlara sahip algoritmalardır. Bu algoritmalar her çalıştırıldıklarında farklı sonuçlar üretebilirler. Metasezgisel algoritmalar, evrimsel algoritmalar ve yapay sinir ağları gibi modern optimizasyon yöntemleri genellikle stokastik karaktere sahiptir.

4.4.1 Stokastik Algoritmaların Test Prensipleri

- **Çoklu çalıştırma:** Her test fonksiyonu için algoritma birden çok kez (genellikle 30-50 bağımsız çalıştırma) çalıştırılmalıdır.
- **İstatistiksel analiz:** Sonuçların ortalaması, standart sapması, medyanı, en iyi ve en kötü değerleri raporlanmalıdır.
- **Yakınsama analizi:** Algoritmanın zaman içindeki davranışı, iterasyon/değerlendirme sayısına karşı en iyi değer grafiği ile gösterilmelidir.
- **Parametre duyarlılığı:** Algoritmanın parametre değişimlerine olan duyarlılığı test edilmelidir.

4.5 Test Prensipleri

Optimizasyon algoritmalarının adil ve kapsamlı bir şekilde değerlendirilmesi için bazı temel prensiplere uyulmalıdır:

- **Çeşitlilik:** Farklı özelliklere sahip test fonksiyonları kullanılmalıdır.
- **Adil karşılaştırma:** Karşılaştırılan tüm algoritmalar için aynı koşullar (başlangıç noktaları, fonksiyon değerlendirme sayısı, sonlandırma kriterleri) sağlanmalıdır.
- **Yeterli tekrar:** Stokastik algoritmalar için yeterli sayıda bağımsız çalıştırma yapılmalıdır.

- **Boyut değişimi:** Algoritmaların farklı problem boyutlarındaki performansı test edilmelidir.
- **Kapsamlı raporlama:** Sadece ortalama veya en iyi değerler değil, tam istatistiksel sonuçlar raporlanmalıdır.

4.6 Performans Ölçütleri

Optimizasyon algoritmalarının performansı çeşitli ölçütlerle değerlendirilebilir. Bu ölçütler, sayısal ve kalite olmak üzere iki ana kategoriye ayrılabilir.

4.6.1 Sayısal Ölçütler

- **Yakınsama hızı:** Algoritmanın istenen çözüme ne kadar hızlı ulaştığını gösterir. Genellikle fonksiyon değerlendirme sayısı veya iterasyon sayısı olarak ölçülür.
- **Hassasiyet:** Bulunan çözümün bilinen global optimuma ne kadar yakın olduğunu gösterir. Genellikle mutlak veya göreceli hata olarak ölçülür.
- **Başarı oranı:** Algoritmanın kabul edilebilir bir çözüme ulaşma yüzdesidir. Özellikle stokastik algoritmalar için önemlidir.
- **Hesaplama karmaşıklığı:** Algoritmanın çalışma süresi veya bellek kullanımı olarak ölçülebilir.

4.6.2 Kalite Ölçütleri

- **Sağlamlık (robustness):** Algoritmanın farklı problem tiplerine, başlangıç koşullarına ve parametre değişimlerine karşı duyarlılığı.
- **Genellenebilirlik:** Algoritmanın farklı problem sınıflarında gösterdiği performans.
- **Ölçeklenebilirlik:** Problem boyutu arttıkça algoritmanın performansındaki değişim.
- **Keşif-yararlanma dengesi:** Algoritmanın global arama (keşif) ve yerel iyileştirme (yararlanma) arasındaki dengeyi sağlama yeteneği.

[28]

No Free Lunch Teoremi

Hiçbir optimizasyon algoritması tüm problemlerde en iyi performansı gösteremez:

- Her algoritmanın güçlü ve zayıf yönleri vardır
- Problem yapısına uygun algoritma seçimi önemlidir
- Hibrit yaklaşımlar avantajlı olabilir

4.6.3 Benchmark Sonuçlarının Sunumu

Benchmark test sonuçlarının etkili sunumu için şu yöntemler kullanılabilir:

- **Tablo formatı:** Algoritmaların her test fonksiyonu için ortalama, standart sapma, en iyi ve en kötü değerlerini gösteren tablolar.

[28] Performans ölçütleri, farklı algoritmaların objektif olarak karşılaştırılmasını sağlar. Ancak, tek bir ölçüt yerine birden fazla ölçütün birlikte değerlendirilmesi daha sağlıklıdır.

- **Yakınsama grafikleri:** İterasyon/değerlendirme sayısına karşı en iyi değerin değişimini gösteren grafikler.
- **Kutu grafikleri (Box plots):** Sonuçların dağılımını ve medyan, çeyrekler gibi istatistiksel özetleri görsel olarak sunan grafikler.
- **Sıralama tabloları:** Algoritmaların her bir test fonksiyonu için performans sıralamasını gösteren tablolar.
- **İstatistiksel anlamlılık testleri:** Algoritmaların performans farklarının istatistiksel olarak anlamlı olup olmadığını gösteren testler (t-testi, Wilcoxon işaretli sıra testi, Friedman testi vb.).

4.7 Benchmark Fonksiyonlarının Kategorileri

Farklı özelliklere sahip benchmark fonksiyonları, algoritmaların çeşitli zorluklarla başa çıkma yeteneğini test eder. Burada, temel fonksiyon kategorileri ve örnekleri verilmiştir:

4.7.1 Çok Sayıda Yerel Minimuma Sahip Fonksiyonlar

Bu fonksiyonlar, algoritmalar için bir zorluk oluşturan birçok yerel minimuma sahiptir ve global optimizasyon yeteneğini test eder:

- Ackley
- Rastrigin
- Griewank
- Schwefel
- Levy
- Shubert

4.7.2 Çanak Şeklindeki Fonksiyonlar

Bu fonksiyonlar, dairesel veya eliptik konturlarla çevrili tek bir minimuma sahiptir ve yerel optimizasyon yeteneğini test eder:

- Sphere
- Bohachevsky
- Sum Squares
- Rotated Hyper-Ellipsoid

4.7.3 Tabak Şeklindeki Fonksiyonlar

Bu fonksiyonlar, algoritmaların doğru yönü belirleme yeteneğini zorlayabilen düz bölgelere sahiptir:

- Booth
- Matyas
- Zakharov
- McCormick

4.7.4 Vadi Şeklindeki Fonksiyonlar

Bu fonksiyonlar, birçok algoritmanın yakınsamasını yavaşlatabilen uzun, dar vadilere sahiptir:

- Rosenbrock (Muz Fonksiyonu)
- Six-Hump Camel
- Three-Hump Camel
- Dixon-Price

4.7.5 Dik Sırtlar/Düşüşler İçeren Fonksiyonlar

Bu fonksiyonlar, gradyan tabanlı yöntemleri zorlayabilecek dik sırtlara veya düşüşlere sahiptir:

- Easom
- Michalewicz
- De Jong N.5

4.8 Sonuç ve Uygulamalar

Benchmark test fonksiyonları, optimizasyon algoritmalarının geliştirilmesi, test edilmesi ve karşılaştırılması için vazgeçilmez araçlardır. Bu fonksiyonlar, algoritmaların farklı problem tiplerindeki performansını değerlendirmeye ve zayıf yönlerini belirlemeye yardımcı olur.

- **Algoritma geliştirmede:** Yeni algoritmaların güçlü ve zayıf yönlerinin belirlenmesi
- **Parametre ayarlamada:** Algoritma parametrelerinin optimizasyonu
- **Hibrit yaklaşımlarda:** Farklı algoritmaların güçlü yönlerini birleştiren hibrit yöntemlerin geliştirilmesi
- **Eğitimde:** Optimizasyon algoritmalarının davranışlarının anlaşılması
- **Gerçek dünya problemlerine hazırlıkta:** Daha karmaşık problemlere geçmeden önce algoritmaların temel yeteneklerinin değerlendirilmesi

5 Metasezgisel Optimizasyon Algoritmaları I

Doğadan esinlenen ve karmaşık optimizasyon problemlerinin çözümünde kullanılan modern algoritmalar, bu bölümde ele alınacaktır. Bu algoritmalar, klasik yöntemlerin yetersiz kaldığı durumlarda etkili çözümler sunar.

5.1 Metasezgisel Algoritmaların Temel Özellikleri

Metasezgisel algoritmalar, karmaşık optimizasyon problemlerinin çözümünde kullanılan, doğadan esinlenmiş yöntemlerdir:

- Stokastik karaktere sahip
- Problem-bağımsız yapı
- Gradyan bilgisi gerektirmez
- Global optimuma ulaşma potansiyeli

[29]

[29] Metasezgisel algoritmaların en önemli özelliği, karmaşık ve çok modlu problemlerde lokal optimumlara takılmadan global optimuma ulaşabilme potansiyelleridir.

5.2 Deterministik ve Stokastik Algoritmaların Farkları

Deterministik vs Stokastik

- **Deterministik:**
  - Aynı başlangıç $\rightarrow$ Aynı sonuç
  - Gradyan tabanlı
  - Lokal optimuma hızlı yakınsama
- **Stokastik:**
  - Rastgele arama
  - Her çalıştırmada farklı sonuç
  - Global optimum potansiyeli

5.3 Arama Yöntemi Açısından Metasezgisel Algoritmaların Sınıflandırılması

Metasezgisel algoritmalar, arama yöntemleri açısından temel olarak iki kategoriye ayrılır: popülasyon tabanlı ve tekil arama temelli algoritmalar. Popülasyon tabanlı algoritmalar birden fazla çözüm adayını eş zamanlı olarak değerlendirirken, tekil arama temelli algoritmalar tek bir çözüm üzerinde çalışır. Bu sınıflandırma, algoritmaların çalışma prensiplerini ve optimizasyon stratejilerini anlamak için önemli bir çerçeve sunar.[30]

[30] Bu bağlamda algoritmaların çalışma prensiplerini daha iyi anlayabilmek için bağlantıdaki örnek incelenebilir.

5.3.1 Popülasyon Tabanlı Algoritmalar

Popülasyon tabanlı algoritmalar, optimizasyon sürecinde birden fazla çözüm adayını eş zamanlı olarak değerlendiren ve bu çözümler arasındaki etkileşimlerden faydalanan yöntemlerdir. Bu algoritmalar, arama uzayının farklı bölgelerini aynı anda keşfederek global optimuma ulaşma olasılığını artırır. Popülasyon tabanlı yaklaşımlar, çözüm adaylarının çeşitliliğini koruyarak lokal optimumlara takılma riskini azaltır ve karmaşık, çok modlu problemlerde etkili sonuçlar verir.

Popülasyon tabanlı algoritmaların en yaygın örnekleri arasında Genetik Algoritmalar, Parçacık Sürü Optimizasyonu ve Diferansiyel Evrim bulunur. Bu algoritmalarda, popülasyondaki her birey (çözüm adayı) belirli kurallara göre evrilir ve birbirleriyle etkileşime girer. Örneğin, Genetik Algoritmalarda çaprazlama ve mutasyon operatörleri kullanılırken, Parçacık Sürü Optimizasyonunda parçacıklar kendi deneyimlerinden ve sürünün kolektif bilgisinden yararlanarak hareket eder. Bu etkileşimler, algoritmanın hem keşif (exploration) hem de sömürü (exploitation) yeteneklerini dengeli bir şekilde kullanmasını sağlar.

5.3.2 Tekil Arama Temelli

Tekil arama temelli algoritmalar, optimizasyon sürecinde tek bir çözüm adayı üzerinde çalışan ve bu çözümü adım adım iyileştiren yöntemlerdir. Bu algoritmalar, mevcut çözümün komşuluğundaki potansiyel çözümleri değerlendirerek, daha iyi bir çözüme doğru ilerler. Tekil arama yaklaşımı, genellikle daha az bellek kullanımı ve daha hızlı iterasyon süreleri gibi avantajlar sunar, ancak lokal optimumlara takılma riski taşır.

En yaygın tekil arama temelli metasezgisel algoritmalar arasında Tavlama Benzetimi (Simulated Annealing), Tabu Araması (Tabu Search) ve Değişken Komşuluk Araması (Variable Neighborhood Search) bulunur. Tavlama Benzetimi, metallerin tavlama işleminden esinlenerek, başlangıçta kötü çözümleri de belirli bir olasılıkla kabul eder ve zamanla bu olasılığı azaltır. Tabu Araması,

yakın zamanda ziyaret edilen çözümleri "tabu" olarak işaretleyerek döngüsel hareketleri engeller ve arama uzayının daha geniş bölgelerini keşfetmeyi sağlar.

Tekil Arama Temelli Algoritmaların Özellikleri

- **Avantajlar:**
  - Düşük bellek gereksinimi
  - Hızlı iterasyon süreleri
  - Basit implementasyon
  - Yerel arama yetenekleri
- **Dezavantajlar:**
  - Lokal optimumlara takılma riski
  - Geniş arama uzaylarında sınırlı keşif yeteneği
  - Başlangıç çözümüne bağımlılık

Ayrıca bu tip meta-sezgisel algoritmaların başarılı olabilmesi için genellikle bazı lokal optimumdan kaçınma mekanizmaları geliştirilir. Örneğin Tavlama benzetimi algoritması içindeki sıcaklık parametresi, mevcut en iyi çözümden uzaklaşarak lokal optimumdan kaçınmayı sağlar. Bu algoritma özelinde bu olasılık erken iterasyonlarda daha yüksekken, sonraki iterasyonlarda azalır. Benzer şekilde birçok algoritma farklı biçimlerde lokal optimumlardan kaçınma mekanizmaları barındırır.

5.4 Arama Stratejisi Açısından Metasezgisel Algoritmaların Sınıflandırılması

5.4.1 Küresel (Global) arama odaklı algoritmalar

Küresel arama odaklı algoritmalar, çözüm uzayının geniş bir bölümünü keşfetmeye odaklanan yöntemlerdir. Bu algoritmalar, arama uzayının farklı bölgelerini sistematik veya rastgele bir şekilde örnekleyerek global optimuma ulaşmayı hedefler. Küresel arama stratejileri, özellikle çok modlu ve karmaşık optimizasyon problemlerinde, lokal optimumlara takılma riskini azaltmak için önemlidir. Bu yaklaşım, arama uzayının daha geniş bir kısmını keşfederek, potansiyel olarak daha iyi çözümlerin bulunduğu bölgeleri belirlemeye yardımcı olur.

Genetik Algoritmalar ve Parçacık Sürü Optimizasyonu gibi popülasyon tabanlı yöntemler, doğal olarak küresel arama yeteneklerine sahiptir. Örneğin, Genetik Algoritmalarda yüksek mutasyon oranı ve çeşitlilik koruma stratejileri, algoritmanın keşif yeteneğini artırır. Benzer şekilde, Diferansiyel Evrim algoritmasında kontrol parametrelerinin (F ve CR) uygun değerleri, algoritmanın global arama yeteneğini güçlendirir. Bu algoritmalar, özellikle başlangıç aşamalarında, arama uzayının geniş bölgelerini keşfetme eğilimindedir ve zaman içinde daha umut verici bölgelere odaklanır. Küresel arama stratejileri, hesaplama maliyeti yüksek olsa da, özellikle önceden bilinmeyen veya karmaşık optimizasyon problemlerinde, daha kaliteli çözümlere ulaşma potansiyeli sunar.

5.4.2 Yerel arama odaklı algoritmalar

Yerel arama odaklı algoritmalar, mevcut en iyi çözümün yakın çevresini detaylı bir şekilde araştırarak daha iyi çözümlere ulaşmayı hedefleyen yöntemlerdir. Bu algoritmalar, belirli bir bölgeyi yoğun şekilde araştırarak, o bölgedeki en iyi çözümü (yerel optimum) bulmayı amaçlar. Yerel arama, genellikle daha hızlı yakınsama sağlar ve hesaplama açısından daha verimlidir. Özellikle tek modlu problemlerde

veya global optimum bölgesi hakkında önceden bilgi sahibi olunduğunda etkili sonuçlar verir.

Tepe tırmanma, benzetilmiş tavlama ve tabu araması gibi algoritmalar, temel olarak yerel arama stratejileri kullanır. Bu algoritmalarda, mevcut çözümün komşuluğundaki çözümler değerlendirilir ve belirli kriterlere göre bir sonraki adım seçilir. Örneğin, tabu aramasında, daha önce ziyaret edilen çözümlerin tekrar değerlendirilmesini önlemek için bir tabu listesi tutulur; bu, algoritmanın yerel optimumlara takılmasını engeller ve arama uzayının daha etkin bir şekilde araştırılmasını sağlar. Özellikle yapısal optimizasyon problemlerinde, gradient bilgisinin kullanılabildiği durumlarda, yerel arama stratejileri, belirli bir başlangıç noktasından itibaren hızlı bir şekilde yakınsayabilir. Ancak, bu algoritmaların başarısı, büyük ölçüde başlangıç noktasının seçimine bağlıdır ve karmaşık, çok modlu problemlerde lokal optimumlara takılma riski yüksektir.

5.4.3 Karma (Hybrid) arama

Karma arama stratejileri, küresel ve yerel arama yöntemlerinin güçlü yönlerini birleştirerek, optimizasyon sürecinin etkinliğini artırmayı hedefler. Bu yaklaşımda, genellikle algoritmanın başlangıç aşamalarında küresel arama yapılarak potansiyel çözüm bölgeleri belirlenir, daha sonra bu bölgelerde yerel arama teknikleri uygulanarak çözümler iyileştirilir. Karma stratejiler, hem geniş arama uzayının keşfedilmesini (exploration) hem de umut vadeden bölgelerin detaylı araştırılmasını (exploitation) dengeli bir şekilde sağlar.

Memetik Algoritmalar, karma arama stratejilerine iyi bir örnektir. Bu algoritmalar, Genetik Algoritmalar gibi evrimsel yöntemleri kullanarak geniş arama uzayını keşfeder, ardından her bireye (çözüm adayına) yerel arama teknikleri uygulayarak çözümleri iyileştirir. Benzer şekilde, Parçacık Sürü Optimizasyonu ve Gradyan İniş yöntemlerinin hibrit versiyonları da geliştirilmiştir. Bu hibrit yaklaşımlar, karmaşık mühendislik problemlerinde, özellikle de yapısal optimizasyon gibi alanlarda başarılı sonuçlar vermektedir. Örneğin, topoloji optimizasyonunda, önce metasezgisel bir algoritma ile genel yapı belirlenir, ardından matematik programlama teknikleri ile detaylı optimizasyon gerçekleştirilir. Karma arama stratejileri, hesaplama verimliliği ve çözüm kalitesi arasında daha iyi bir denge sağlayarak, tek başına küresel veya yerel arama stratejilerine göre daha etkili sonuçlar elde edilmesine olanak tanır.

5.5 Doğa Kaynaklı İlhamlara Göre Metasezgisel Algoritmaların Sınıflandırılması

5.5.1 Biyolojik evrim temelli

Biyolojik evrim temelli algoritmalar, Darwin'in doğal seleksiyon ve evrim teorisinden esinlenerek geliştirilmiş optimizasyon yöntemleridir. Bu algoritmalar, doğadaki canlıların nesiller boyunca çevresel koşullara adaptasyon sürecini taklit ederek optimizasyon problemlerini çözmeye çalışır. En temel evrimsel algoritma olan Genetik Algoritma, biyolojik evrimden esinlenen operatörleri kullanır: seçilim (selection), çaprazlama (crossover) ve mutasyon (mutation). Bu operatörler aracılığıyla, algoritma popülasyonu nesiller boyunca evrimleştirir ve amaç fonksiyonuna göre daha uygun bireylerin hayatta kalmasını sağlar.

Diferansiyel Evrim, Evrim Stratejileri ve Genetik Programlama gibi diğer evrimsel algoritmalar da benzer prensipleri farklı şekillerde uygular. Örneğin, Diferansiyel Evrim algoritması, popülasyon üyeleri arasındaki vektör farklarını kullanarak yeni çözümler üretir ve bu sayede arama uzayının özelliklerine uyum sağlar. Evrim Stratejileri ise özellikle sürekli optimizasyon problemlerinde, kendi kendine adapte olan mutasyon parametreleri kullanarak performansını artırır. Bu algoritmalar, özellikle çok boyutlu, süreksiz ve çok modlu optimizasyon problemlerinde etkilidir. Ayrıca, parametrelerinin kolayca ayarlanabilmesi ve farklı prob-

lem türlerine uyarlanabilmesi, bu algoritmaların yapısal optimizasyon, mekanik tasarım ve malzeme bilimi gibi mühendislik alanlarında yaygın kullanımını sağlamaktadır.

5.5.2 Sürü zekâsı temelli

Sürü zekâsı temelli algoritmalar, doğadaki sürü halinde yaşayan canlıların kolektif davranışlarından esinlenerek geliştirilmiş optimizasyon yöntemleridir. Bu algoritmalarda, basit kurallara sahip bireyler arasındaki etkileşimler sonucunda, karmaşık ve zeki davranışlar ortaya çıkar. Parçacık Sürü Optimizasyonu (PSO), kuşların ve balık sürülerinin hareketlerinden esinlenerek geliştirilmiş en popüler sürü zekâsı algoritmasıdır. PSO'da her parçacık, kendi en iyi pozisyonu ve sürünün global en iyi pozisyonu hakkındaki bilgileri kullanarak hareketini günceller. Bu kolektif zekâ, sürünün en verimli kaynakları (optimum noktaları) hızla bulmasını sağlar.

Karınca Kolonisi Optimizasyonu, karıncaların feromon izlerini kullanarak en kısa yolu bulma davranışını simüle eder ve özellikle kombinatoryal optimizasyon problemlerinde etkilidir. Yapay Arı Kolonisi algoritması ise bal arılarının nektar toplama stratejisinden esinlenerek, farklı çözüm bölgelerinin keşfedilmesini ve verimli bölgelerin daha yoğun şekilde araştırılmasını sağlar. Ateş Böceği Algoritması, ateş böceklerinin parlaklık ve çekim prensiplerini taklit ederken, Yarasa Algoritması ise yarasaların ekokonumlandırma yöntemlerini simüle eder. Sürü zekâsı temelli bu algoritmalar, genellikle uygulama kolaylığı, az sayıda kontrol parametresi ve küresel optimizasyon yetenekleri ile öne çıkar. Özellikle robot yörüngesi planlaması, enerjisi sistemleri optimizasyonu ve sensör ağlarının konumlandırılması gibi mühendislik uygulamalarında başarılı sonuçlar vermektedir.

5.5.3 Fiziksel süreçlerden esinlenen

Fiziksel süreçlerden esinlenen algoritmalar, doğadaki fiziksel olayları ve kanunları temel alarak geliştirilen optimizasyon yöntemleridir. Bu algoritmalar, termodinamik, mekanik, elektromanyetik ve kuantum fiziği gibi farklı fizik alanlarındaki prensipleri optimizasyon problemlerine uygular. Benzetilmiş Tavlama (Simulated Annealing), metallerin tavlama işleminden esinlenerek geliştirilmiş en eski fizik temelli algoritmalardan biridir. Bu algoritma, metalin yüksek sıcaklıkta ısıtılıp yavaşça soğutulması sırasında moleküler yapının düşük enerjili duruma geçişini taklit eder. Başlangıçta yüksek bir "sıcaklık" parametresi ile kötü çözümleri de kabul eden algoritma, zamanla "soğuyarak" daha seçici hale gelir ve global optimuma yakınsama şansını artırır.

Yerçekimi Arama Algoritması (Gravitational Search Algorithm), Newton'un evrensel çekim yasasını temel alır ve çözüm adayları arasındaki kütlesel etkileşimi simüle eder. Harmonik Arama (Harmony Search) algoritması, müzisyenlerin doğaçlama sırasında uyumlu tınılar oluşturma sürecinden esinlenir. Büyük Patlama-Büyük Çöküş (Big Bang-Big Crunch) algoritması, evrenin genişleme ve daralma döngülerini taklit ederken, Su Döngüsü Algoritması (Water Cycle Algorithm) ise suyun buharlaşma, yağış ve akış süreçlerini optimizasyon sürecine uyarlar. Bu fizik temelli algoritmalar, genellikle matematiksel olarak iyi tanımlanmış prensiplere dayanmaları nedeniyle, kararlı ve güvenilir sonuçlar verme eğilimindedir. Özellikle mühendislik tasarımı, sinyal işleme ve yapısal optimizasyon gibi alanlarda, karmaşık ve çok değişkenli problemlerin çözümünde yaygın olarak kullanılmaktadır.

5.5.4 Kimyasal, biyolojik veya sosyal süreçlerden esinlenen

Kimyasal, biyolojik veya sosyal süreçlerden esinlenen metasezgisel algoritmalar, doğadaki ve toplumdaki çeşitli karmaşık sistemlerin davranışlarını taklit eder.

Kimyasal Reaksiyon Optimizasyonu (Chemical Reaction Optimization), moleküllerin kinetik enerjisi, çarpışmaları ve kimyasal reaksiyonları temel alır. Bu algoritma, moleküllerin düşük enerji durumlarına ulaşma eğilimini taklit ederek, optimizasyon problemlerindeki global minimumları bulmayı hedefler. Biyolojik süreçlerden esinlenen algoritmalar arasında, bakterilerin besin arama stratejilerini simüle eden Bakteri Foraging Optimizasyonu ve bağışıklık sisteminin antijen-antikor tepkilerini taklit eden Yapay Bağışıklık Sistemi algoritmaları sayılabilir.

Sosyal süreçlerden esinlenen algoritmalar, insan topluluklarının davranışlarını ve sosyal etkileşimlerini model alır. Öğretme-Öğrenme Tabanlı Optimizasyon (Teaching-Learning-Based Optimization), bir sınıftaki öğretmen-öğrenci etkileşimini simüle ederken, İmperialist Yarışmalı Algoritma (Imperialist Competitive Algorithm), emperyalist ülkelerin kolonileri üzerindeki hakimiyet mücadelesini taklit eder. Sosyal Grup Optimizasyonu, insan gruplarının problem çözme yöntemlerini modellerken, Yapay Dil Topluluğu (Artificial Bee Colony) algoritması ise arı kolonilerinin nektar toplama stratejilerini optimizasyon sürecine uyarlar. Bu algoritmaların ortak özelliği, karmaşık sistemlerdeki emergent (ortaya çıkan) davranışları kullanarak, çok boyutlu ve çok modlu arama uzaylarında etkili çözümler üretmeleridir. Özellikle veri madenciliği, sinir ağları eğitimi, robotik ve yapay zeka uygulamalarında başarılı sonuçlar elde etmektedirler.

5.6 Keşif ve Sömürü Dengesi Açısından Metasezgisel Algoritmaların Sınıflandırılması

Metasezgisel algoritmaların başarısında, keşif (exploration) ve sömürü (exploitation) arasındaki dengenin doğru kurulması kritik öneme sahiptir. Keşif, arama uzayının henüz ziyaret edilmemiş bölgelerinin araştırılmasını ifade ederken; sömürü, önceden keşfedilmiş ve umut vadeden bölgelerin daha detaylı incelenmesini kapsar. Bu iki süreç arasındaki denge, algoritmanın performansını doğrudan etkiler. Keşif ağırlıklı algoritmalar, geniş arama uzayını daha kapsamlı tarayabilir ve global optimuma ulaşma olasılığını artırır, ancak yakınsama hızları genellikle düşüktür. Sömürü ağırlıklı algoritmalar ise belirli bölgelerde hızlı yakınsama sağlar, ancak lokal optimumlara takılma riski taşır.

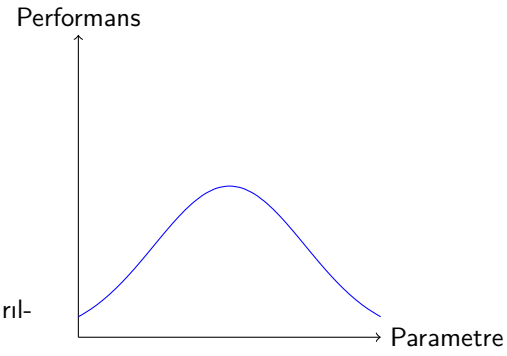
Metasezgisel algoritmaların çoğu, optimizasyon süreci boyunca keşif ve sömürü arasındaki dengeyi dinamik olarak ayarlar. Örneğin, Genetik Algoritmalarda mutasyon operatörü keşif yeteneğini artırırken, çaprazlama operatörü sömürü sürecini destekler. Parçacık Sürü Optimizasyonunda, atalet ağırlığı (inertia weight) ve hızlanma katsayıları (acceleration coefficients) bu dengeyi kontrol eder. Benzetilmiş Tavlama algoritmasında, sıcaklık parametresinin zamanla azalması, algoritmanın başlangıçta keşif ağırlıklı davranıştan sömürü ağırlıklı davranışa geçişini sağlar. Algoritmalar, problem karakteristiklerine ve optimizasyon sürecinin aşamasına göre bu iki süreci dengeleyecek şekilde tasarlanmalıdır. Yapısal optimizasyon gibi mühendislik uygulamalarında, özellikle karmaşık ve çok modlu problemlerde, keşif ve sömürü arasındaki dengenin doğru kurulması, global optimuma ulaşmada ve hesaplama verimliliğinde belirleyici rol oynar.

5.7 Algoritmaların Hiperparametre Ayarlamaları

Metasezgisel algoritmaların performansını doğrudan etkileyen ve kullanıcı tarafından belirlenen değişkenlere hiperparametreler denir. Bu parametreler, algoritmanın davranışını, yakınsama hızını ve çözüm kalitesini önemli ölçüde etkiler. Örneğin, Genetik Algoritmalarda popülasyon büyüklüğü, mutasyon oranı ve çaprazlama oranı; PSO'da atalet ağırlığı ve öğrenme faktörleri; Benzetilmiş Tavlamada başlangıç sıcaklığı ve soğutma oranı gibi değerler hiperparametrelerdir. Bu parametrelerin optimal değerlerinin belirlenmesi, algoritmanın başarısı için kritik öneme sahiptir ve genellikle problem özelliklerine göre farklılık gösterir.

5.7.1 Parametre Seçim Stratejileri

- Deneysel analiz
- Adaptif ayarlama
- Meta-optimizasyon
- İstatistiksel tasarım



Şekil 21: Parametre değeri ve performans ilişkisi

5.8 Optimizasyon Algoritmalarının Objektif Kıyaslanması

Metasezgisel optimizasyon algoritmalarının performanslarının objektif olarak karşılaştırılması, algoritma seçimi ve geliştirilmesi açısından büyük önem taşır. Bu karşılaştırma, standart test fonksiyonları, gerçek dünya problemleri ve istatistiksel analiz yöntemleri kullanılarak yapılır. Standart test fonksiyonları (Benchmark functions), farklı zorluk seviyelerinde ve karakteristiklerde (çok modlu, süreksiz, gürültülü vb.) problemler sunarak, algoritmaların çeşitli koşullar altındaki performanslarını değerlendirmeye olanak tanır. Sphere, Rastrigin, Rosenbrock ve Ackley gibi yaygın kullanılan test fonksiyonları, algoritmaların global optimizasyon yeteneklerini, yakınsama hızlarını ve keşif-sömürü dengelerini test etmek için kullanılır.

Algoritmaların objektif kıyaslanmasında, tek bir test problemi veya performans metriği yerine, çeşitli problem türlerini ve çoklu performans kriterlerini içeren kapsamlı bir değerlendirme yaklaşımı benimsenmelidir. Bu amaçla, çözüm kalitesi, yakınsama hızı, hesaplama maliyeti, sağlamlık (robustness) ve ölçeklenebilirlik gibi farklı performans metrikleri bir arada değerlendirilir. İstatistiksel anlamlılık testleri (t-testi, Wilcoxon işaretli sıra testi vb.), algoritmaların performans farklılıklarının rastgele değişkenlikten mi yoksa gerçek bir üstünlükten mi kaynaklandığını belirlemek için kullanılır. Ayrıca, algoritmaların farklı problem boyutlarındaki ve kısıt koşullarındaki davranışları da karşılaştırmanın önemli bir parçasıdır.

No Free Lunch teoremi, hiçbir optimizasyon algoritmasının tüm problem sınıflarında diğerlerinden üstün olamayacağını belirtir. Bu nedenle, algoritmaların kıyaslanması, belirli problem türleri veya uygulama alanları için hangi algoritmanın daha uygun olduğunu belirlemeye odaklanmalıdır. Yapısal optimizasyon, mekanik tasarım ve malzeme bilimi gibi mühendislik alanlarında, problem karakteristiklerine uygun algoritma seçimi, çözüm kalitesi ve hesaplama verimliliği açısından kritik öneme sahiptir. Objektif kıyaslama çalışmaları, yeni geliştirilen algoritmaların mevcut yöntemlere göre avantajlarını ve dezavantajlarını ortaya koyarak, algoritma geliştirme sürecine rehberlik eder ve uygulama alanına özgü en uygun algoritmanın seçilmesini sağlar.

5.8.1 Kıyaslama Kriterleri

- Yakınsama hızı
- Çözüm kalitesi
- Hesaplama maliyeti
- Parametre hassasiyeti

6 Metasezgisel Optimizasyon Algoritmaları II

Metasezgisel algoritmaların daha ileri uygulamaları ve hibrit yaklaşımlar bu bölümde incelenecektir. Modern optimizasyon problemlerinin çözümünde kullanılan gelişmiş teknikler ve bunların uygulamaları ele alınacaktır.

6.1 Tavlama Benzetimi (Simulated Annealing)

Tavlama Benzetimi (Simulated Annealing), metalurjideki tavlama işleminden esinlenen güçlü bir metasezgisel optimizasyon algoritmasıdır. Metallerin tavlama sürecinde, malzeme önce yüksek sıcaklıklara ısıtılır, bu sırada atomlar yüksek enerji seviyelerinde rastgele hareket eder. Ardından malzeme kontrollü bir şekilde yavaşça soğutulur, böylece atomlar düşük enerjili, kararlı bir kristal yapıya yerleşir. Bu fiziksel süreç, optimizasyon problemlerinde global optimuma ulaşmak için etkili bir strateji olarak uyarlanmıştır. Algoritma, başlangıçta yüksek "sıcaklık" parametresiyle çalışarak kötü çözümleri bile belirli bir olasılıkla kabul eder ve böylece geniş bir arama uzayını keşfeder. Sıcaklık kademeli olarak düştükçe, algoritma daha seçici hale gelir ve umut vadeden bölgelerde daha yoğun arama yapar. Bu yaklaşım, lokal optimumlara takılma riskini azaltarak, karmaşık ve çok modlu optimizasyon problemlerinde global optimuma yakınsama olasılığını artırır. [31]

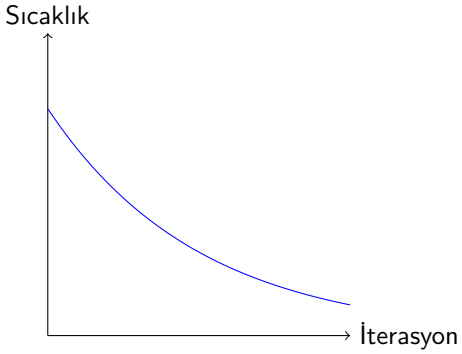
[31] Tavlama benzetimi, özellikle kombinatoryal optimizasyon problemlerinde başarılı sonuçlar verir. Sıcaklık düşüş hızı ve başlangıç sıcaklığı algoritmanın performansını etkileyen önemli parametrelerdir. Tavlama benzetimi algoritması kullanarak Ackley fonksiyonu üzerinde gerçekleştirilen örnek python kodu:

6.1.1 Algoritmanın Temeli

Metallerin tavlama işleminden esinlenmiş bir optimizasyon algoritması:

- Yüksek sıcaklıkta rastgele hareketler
- Sıcaklık düştükçe kontrollü hareketler
- Kötü çözümlerin kabul edilme olasılığı

$$P(\Delta E) = \exp\left(-\frac{\Delta E}{kT}\right) \tag{37}$$



Şekil 22: Tavlama benzetiminde sıcaklık değişimi

6.1.2 Algoritmanın Adımları

1. Başlangıç sıcaklığı ve çözümü belirle
2. Her sıcaklık seviyesinde:
   - Yeni çözüm üret
   - Enerji farkını hesapla
   - Metropolis kriterine göre kabul/ret
3. Sıcaklığı düşür
4. Durma kriteri sağlanana kadar devam et

6.2 Tabu Arama Algoritması

Tabu Arama Algoritması (Tabu Search), 1986 yılında Fred Glover tarafından geliştirilen ve insan belleğinden esinlenen güçlü bir metasezgisel optimizasyon yöntemidir. İnsan zekâsının problem çözme sürecinde geçmiş deneyimlerden faydalanma ve tekrarlayan hataları önleme yeteneğini taklit eder. Algoritma, özellikle kombinatoryal optimizasyon problemlerinde etkili sonuçlar vermektedir ve yerel arama yöntemlerinin lokal optimumlara takılma sorununu aşmak için tasarlanmıştır. Tabu Arama, adını algoritmanın temel bileşeni olan ve yakın zamanda ziyaret edilen çözümleri geçici olarak "yasaklayan" tabu listesinden alır. Bu yaklaşım, arama sürecinin aynı çözümleri tekrar tekrar ziyaret etmesini engelleyerek, arama uzayının daha geniş bölgelerinin keşfedilmesini sağlar.

Tabu Arama Algoritması, mevcut çözümün komşuluğundaki potansiyel çözümleri sistematik olarak değerlendirerek çalışır. Her iterasyonda, mevcut çözümün komşuluğundaki tüm çözümler (veya belirli bir alt kümesi) incelenir ve tabu listesinde olmayan en iyi çözüm seçilir. Seçilen çözüme karşılık gelen hareket tabu listesine eklenir ve liste belirli bir süre (tabu süresi) boyunca bu hareketi yasaklar. Ancak, tabu listesindeki bir hareket bile, eğer "aspirasyon kriteri" olarak bilinen belirli koşulları sağlarsa (örneğin, şimdiye kadar bulunan en iyi çözümden daha iyi bir çözüm üretirse) kabul edilebilir. Algoritma ayrıca, arama sürecini yönlendirmek için "yoğunlaştırma" (umut vadeden bölgelerde daha detaylı arama) ve "çeşitlendirme" (arama uzayının farklı bölgelerine yönelme) stratejilerini kullanır. Bu dengeli yaklaşım, algoritmanın hem lokal optimumları etkili bir şekilde araştırmasını hem de global optimuma yakınsama olasılığını artırmasını sağlar.

6.2.1 Temel Kavramlar

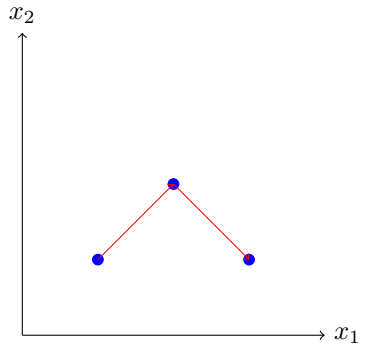
- Tabu listesi
- Komşuluk yapısı
- Aspirasyon kriteri
- Yoğunlaştırma ve çeşitlendirme

Tabu Listesinin Rolü

- Yakın zamanda ziyaret edilen çözümleri saklar
- Döngüsel hareketleri engeller
- Arama uzayının farklı bölgelerini keşfeder
- Liste uzunluğu önemli bir parametredir

6.2.2 Algoritmanın Adımları

1. Başlangıç çözümünü belirle
2. Her iterasyonda:
   - Komşu çözümleri üret
   - Tabu listesini kontrol et
   - En iyi uygun çözümü seç
   - Tabu listesini güncelle
3. Durma kriteri sağlanana kadar devam et



Şekil 23: Tabu arama algoritmasının hareketi

6.3 Genetik Algoritmalar (GA)

Genetik Algoritmalar (GA), 1975 yılında John Holland tarafından geliştirilen ve doğal evrim sürecini taklit eden bir metasezgisel optimizasyon yöntemidir. Bu algoritmalar, çözüm uzayında çalışır ve çözümlerin genetik yapılarını kullanarak arama sürecini yönetirler. GA, çözümlerin kromozomlar (genler) ile temsil edildiği ve bu kromozomların seçilen kısıtlamalar altında evrim sürecine tabi tutulduğu bir yapıya sahiptir.

Genetik Algoritmaların çalışma prensibi, doğal seleksiyon ve genetik mekanizmaları taklit eden bir süreçtir. Algoritma, potansiyel çözümleri kromozomlar olarak kodlayarak başlar ve rastgele bir başlangıç popülasyonu oluşturur. Her iterasyonda, çözümlerin uygunluk değerleri hesaplanır ve daha yüksek uygunluğa sahip bireylerin üreme şansı artar. Seçilen ebeveynler, genetik bilgilerini paylaşmak için çaprazlama işlemine tabi tutulur ve yeni nesil oluşturulur. Çeşitliliği korumak ve arama uzayının farklı bölgelerini keşfetmek için düşük bir olasılıkla mutasyon uygulanır. Bu süreç, belirli bir durma kriteri sağlanana kadar (maksimum nesil sayısı, yeterli uygunluk değeri vb.) tekrarlanır. Genetik Algoritmaların güçlü yanı, karmaşık ve çok boyutlu problemlerde bile global optimuma yakınsama yeteneği ve farklı problem türlerine kolayca uyarlanabilmesidir.

6.3.1 Temel Bileşenler

- Kromozom (çözüm) yapısı
- Popülasyon
- Seçim mekanizması
- Çaprazlama operatörü
- Mutasyon operatörü

$$P(x_i) = \frac{f(x_i)}{\sum_{j=1}^{N} f(x_j)} \tag{38}$$

[32]

[32] Genetik algoritmalar, Darwin'in evrim teorisinden esinlenmiştir. En iyinin hayatta kalması prensibi, optimizasyon sürecinde daha iyi çözümlerin üretilmesini sağlar.

6.3.2 Algoritmanın Adımları

1. Başlangıç popülasyonunu oluştur
2. Her nesilde:
    - Uygunluk (Fitness) değerlerini hesapla
    - Ebeveynleri seç
    - Çaprazlama ve mutasyon uygula
    - Yeni nesli oluştur
3. Durma kriteri sağlanana kadar devam et

Genetik Operatörler

- **Çaprazlama:**
  - Tek noktalı
  - Çok noktalı
  - Uniform
- **Mutasyon:**
  - Bit çevirme
  - Gaussian
  - Uniform

6.4 Parçacık Sürü Optimizasyonu (PSO)

Parçacık Sürü Optimizasyonu (PSO), 1995 yılında Russell Eberhart ve James Kennedy tarafından geliştirilen ve doğal evrim sürecini taklit eden bir metasezgisel optimizasyon yöntemidir. PSO, çözümlerin parçacıklar olarak temsil edildiği ve bu parçacıkların sürü içindeki diğer parçacıkların etkilerini kullanarak arama sürecini yönetirler.

PSO, parçacıkların kendi deneyimlerinden ve sürünün kolektif bilgisinden faydalanarak hareket ettiği bir süreçtir. Her parçacık, kendi en iyi çözümünü (pbest) ve sürünün en iyi çözümünü (gbest) takip eder. Parçacıkların hızı ve pozisyonu, sürüdeki diğer parçacıkların etkileriyle güncellenir ve bu sayede arama uzayında global optimuma yakınsama olasılığı artar. PSO, çok boyutlu ve karmaşık optimizasyon problemlerinde başarılı sonuçlar verir ve algoritmanın hızlı yakınsama yeteneği, az sayıda parametre ve kolay uyarlanabilirliğiyle dikkat çeker.

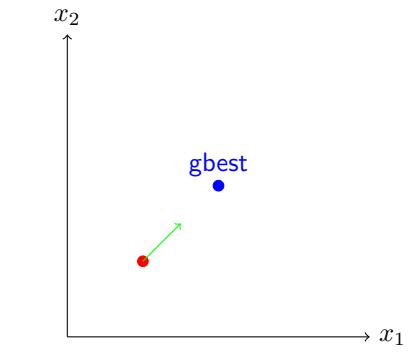
6.4.1 Temel Kavramlar

- Parçacık pozisyonu ve hızı
- Kişisel en iyi (pbest)
- Global en iyi (gbest)
- İnertia ağırlığı
- Öğrenme faktörleri

$$\begin{aligned} v_{i,j}^{t+1} &= w v_{i,j}^{t} + c_1 r_1 (pbest_{i,j} - x_{i,j}^{t}) + c_2 r_2 (gbest_j - x_{i,j}^{t}) \\ x_{i,j}^{t+1} &= x_{i,j}^{t} + v_{i,j}^{t+1} \end{aligned} \tag{39}$$

6.4.2 Algoritmanın Adımları

1. Sürüyü başlat
2. Her iterasyonda:
   - Uygunluk değerlerini hesapla
   - pbest ve gbest'i güncelle
   - Hız ve pozisyonları güncelle
3. Durma kriteri sağlanana kadar devam et



Şekil 24: PSO'da parçacık hareketi

[33]

[33] PSO, kuş sürülerinin davranışından esinlenmiştir. Parçacıklar hem kendi deneyimlerinden hem de sürünün kolektif bilgisinden faydalanarak hareket eder.

6.5 Algoritmaların Avantaj ve Dezavantajları

Karşılaştırmalı Analiz

- **Tavlama Benzetimi:**
  - + Basit implementasyon
  - + Teorik yakınsama garantisi
  - - Yavaş yakınsama
  - - Parametre ayarı hassas
- **Tabu Arama:**
  - + Döngüsel hareketlerden kaçınma
  - + Yerel aramada etkili
  - - Bellek gereksinimi
  - - Komşuluk yapısına bağımlı
- **Genetik Algoritma:**
  - + Paralel arama
  - + Geniş arama uzayı
  - - Yüksek hesaplama maliyeti
  - - Çok sayıda parametre
- **PSO:**
  - + Hızlı yakınsama
  - + Az parametre
  - - Erken yakınsama riski
  - - Lokal optimuma takılma

6.6 Karınca Kolonisi Optimizasyonu (ACO)

Karıncaların yiyecek arama davranışından esinlenmiş bir optimizasyon algoritmasıdır. Özellikle kombinatoryal optimizasyon problemlerinde etkili sonuçlar verir.

Karınca Kolonisi Optimizasyonu (ACO), karıncaların yiyecek kaynağı ile yuvaları arasında en kısa yolu bulma davranışını taklit eden bir metasezgisel optimizasyon algoritmasıdır. Karıncalar hareket ederken feromon adı verilen kimyasal bir madde bırakır ve diğer karıncalar bu feromon izlerini takip eder. Daha kısa yollar daha hızlı tamamlandığı için, bu yollarda feromon birikimi daha yoğun olur ve zamanla daha fazla karınca bu yolları tercih etmeye başlar. Algoritma, yapay karıncaların çözüm uzayında dolaşarak feromon izleri bırakması, bu izlerin zamanla buharlaşması ve karıncaların feromon yoğunluğu ile sezgisel bilgileri (genellikle mesafe gibi) birleştirerek yol seçimi yapması prensibine dayanır. Bu kolektif zeka mekanizması sayesinde, karıncalar optimale yakın çözümlere doğru yönelir ve özellikle gezgin satıcı problemi, araç rotalama ve ağ yönlendirme gibi kombinatoryal optimizasyon problemlerinde etkili sonuçlar verir.

6.6.1 Temel Prensipler

- Feromon izleri

- Olasılıklı yol seçimi
- Feromon güncelleme
- Buharlaşma mekanizması

$$p_{ij}^k = \frac{[\tau_{ij}]^\alpha [\eta_{ij}]^\beta}{\sum_{l \in N_i^k} [\tau_{il}]^\alpha [\eta_{il}]^\beta} \tag{40}$$

[34]

[34] Karınca kolonisi optimizasyonu, özellikle gezgin satıcı problemi gibi kombinatoryal optimizasyon problemlerinde başarılı sonuçlar verir.

6.6.2 Algoritmanın Adımları

1. Başlangıç feromon değerlerini ata
2. Her iterasyonda:
   - Karıncaları yerleştir
   - Çözümleri oluştur
   - Feromon izlerini güncelle
   - Buharlaşmayı uygula
3. Durma kriteri sağlanana kadar devam et

Şekil 25: Feromon izleri ve yol seçimi

6.7 Diferansiyel Evrim (DE) Algoritması

Vektör tabanlı evrimsel bir optimizasyon algoritmasıdır. Sürekli optimizasyon problemlerinde etkili sonuçlar verir.

6.7.1 Temel Operatörler

- Mutasyon vektörü
- Çaprazlama
- Seçim
- Ölçekleme faktörü (F)
- Çaprazlama oranı (CR)

$$v_i = x_{r1} + F(x_{r2} - x_{r3}) \tag{41}$$

DE Stratejileri

- DE/rand/1/bin
- DE/best/1/bin
- DE/rand/2/bin
- DE/best/2/bin
- DE/current-to-best/1/bin

6.7.2 Algoritmanın Adımları

1. Başlangıç popülasyonunu oluştur
2. Her nesilde:
   - Mutasyon vektörlerini üret
   - Çaprazlama uygula
   - Seçim yap
3. Durma kriteri sağlanana kadar devam et

[35]

[35] Diferansiyel evrim algoritması, sürekli optimizasyon problemlerinde özellikle etkilidir. Parametre ayarı görece kolaydır ve yakınsama hızı yüksektir.

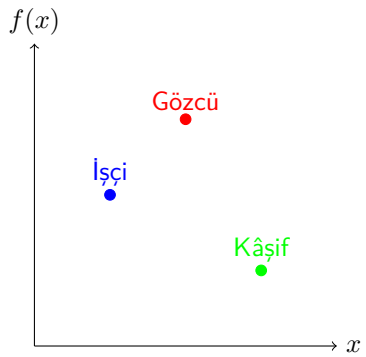
6.8 Yapay Arı Kolonisi (ABC) Optimizasyonu

Bal arılarının yiyecek arama davranışından esinlenen bir optimizasyon algoritmasıdır.

6.8.1 Arı Tipleri ve Görevleri

- İşçi arılar: Mevcut kaynakların araştırılması
- Gözcü arılar: Umut verici kaynakların değerlendirilmesi
- Kâşif arılar: Yeni kaynakların keşfi

$$v_{ij} = x_{ij} + \phi_{ij}(x_{ij} - x_{kj}) \tag{42}$$



Şekil 26: ABC'de farklı arı tiplerinin rolü

6.8.2 Algoritmanın Adımları

1. Başlangıç besin kaynaklarını belirle
2. Her döngüde:
   - İşçi arı fazı
   - Gözcü arı fazı
   - Kâşif arı fazı
3. Durma kriteri sağlanana kadar devam et

6.9 Kuantum ve Hibrit Optimizasyon Algoritmaları

Kuantum mekaniği prensiplerinden esinlenen ve farklı algoritmaların güçlü yönlerini birleştiren yaklaşımlar.

6.9.1 Kuantum-Esinli Algoritmalar

- Kuantum parçacık sürüsü
- Kuantum genetik algoritma
- Kuantum tavlama benzetimi

$$\psi(x,t) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{43}$$

Kuantum Mekanizmasının Avantajları

- Daha iyi keşif yeteneği
- Lokal optimumlardan kaçınma
- Hızlı yakınsama
- Belirsizlik prensibinden faydalanma

6.10 Diğer Metasezgisel Optimizasyon Algoritmaları

Akademik literatüre her yıl onlarca yeni metasezgisel optimizasyon algoritması eklenmektedir. Bu algoritmaların, bazıları hala oldukça özgün yaklaşımlar ortaya koyabimektedir. Ancak temel sınıflandırmalar dahilinde birçoğu daha önceki algoritmaların belli mekanizmalarını belli ölçüde taklit ederek geliştirmektedir. Bu konuya akademik perspektifin dışından bakan bir mühendis için bazı temel algoritmaları anlamak ve uygulayabilmek yeni algoritmaları sürekli olarak takip etmekten çok daha faydalıdır. Özellikle yapay zekanın gelişimi de düşünüldüğünde esas önemli olan doğru pratik sahalarında teorik bilgiyi hayata geçirmetir.

7 Ayrık Parametrelerin Optimizasyonu

Ayrık (Discrete) değişkenler içeren optimizasyon problemlerinin çözüm yöntemleri bu bölümde incelenecektir. Özellikle yapısal sistemlerde karşılaşılan ayrık parametre optimizasyonu problemleri ve çözüm stratejileri ele alınacaktır.

7.1 Ayrık ve Sürekli Optimizasyon Farkları

Optimizasyon problemlerinin çözüm uzayı yapısına göre sınıflandırılması ve temel farklılıkların incelenmesi.

7.1.1 Temel Farklılıklar

- Çözüm uzayının yapısı
- Kullanılabilecek yöntemler
- Hesaplama karmaşıklığı
- Gradyan bilgisinin kullanımı

Ayrık vs Sürekli Optimizasyon

- **Ayrık:**
  - Kesikli çözüm uzayı
  - Kombinatoryal yöntemler
  - NP-zor problemler
  - Gradyan kullanılamaz
- **Sürekli:**
  - Sürekli çözüm uzayı
  - Gradyan tabanlı yöntemler
  - Diferansiyellenebilirlik
  - Lokal bilgi kullanımı

7.2 Gezgin Satıcı Problemi (TSP)

Ayrık optimizasyon problemlerinin klasik örneği olan TSP (Travelling Salesman Problem)[36], birçok gerçek dünya probleminin modellenmesinde kullanılır. Bu problem, bir satıcının belirli şehirleri en kısa mesafede dolaşması gerektiği senaryoyu ele alır. Her şehre yalnızca bir kez uğranması ve tur sonunda başlangıç noktasına dönülmesi gerekir. TSP, lojistik, üretim planlaması, PCB devre tasarımı ve DNA dizilimi gibi alanlarda uygulanır. Problemin çözüm uzayı, şehir sayısı arttıkça faktöriyel olarak büyüdüğünden (n şehir için n! olası tur), büyük ölçekli problemler için kesin çözüm bulmak hesaplama açısından oldukça zorludur.

36

7.2.1 Problem Tanımı

- N şehir arasında en kısa turu bulma
- Her şehre bir kez uğrama
- Başlangıç noktasına dönme
- NP-zor problem sınıfı

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} x_{ij} \tag{44}$$

7.2.2 Çözüm Yaklaşımları

- Kesin yöntemler:
  - Dal-sınır
  - Tamsayılı programlama
- Sezgisel yöntemler:
  - En yakın komşu
  - 2-opt, 3-opt
  - Lin-Kernighan
- Metasezgisel yöntemler:
  - ACO

- GA
- Tabu arama

[37]

7.3 Çelik Yapıların Kesit Optimizasyonu

Ayrık optimizasyon problemlerinin, yapısal optimizasyonda en kolay karşılaşılacak örneği çelik yapıların kesit optimizasyonudur. Bu problemde, çelik yapılarda kullanılan standart kesitlerin seçimine dayalı bir optimizasyon problemi ele alınır.

7.3.1 Problem Tanımı

Çelik yapılarda kesit optimizasyonu, ayrık bir optimizasyon problemidir:

- Standart kesit tabloları
- Yapısal kısıtlar
- Minimum ağırlık hedefi
- Gruplandırma gereksinimleri

$$\begin{aligned} \min \quad & \sum_{i=1}^{n} \rho_i L_i A_i \\ \text{s.t.} \quad & \sigma_i \leq \sigma_{allow} \\ & \delta \leq \delta_{allow} \\ & A_i \in S \end{aligned} \tag{45}$$

7.3.2 Çözüm Stratejileri

- Ayrık değişkenli optimizasyon
- Metasezgisel yöntemler
- Hibrit yaklaşımlar
- Paralel hesaplama

Optimizasyon Süreci

1. Yapısal analiz
2. Kesit seçimi
3. Kısıt kontrolü
4. İteratif iyileştirme

7.4 İndislerle Problem Çözümünün Basitleştirilmesi

Çelik yapı optimizasyonu farklı biçimlerde ele alınabilir. Ancak eğer öntanımlı çelik kesitleri kullanıyorsa, kesit seçimi ayrık bir optimizasyon problemi olarak karşımıza çıkar. Bu durumda, kesit seçimi için bir veri yapısı oluşturmak ve bu veri yapısını kullanarak optimizasyon problemini çözmek gerekir. Bu noktada kesit listeleri indislenerek ele alınabilir. Ancak üzerine tartışılması gereken bir nokta şu olabilir: kesit listeleri hangi parametresi esas alınarak sıralanacak ve indislenecektir. Örneğin, yalnızca kesit alanının sıralamaya esas kabul edilmesi,

[37] TSP, ayrık optimizasyon problemlerinin prototip örneğidir. Birçok gerçek dünya problemi TSP'ye indirgenebilir. Ancak TSP için uygulanan her optimizasyon algoritması, bir başka problem için kullanılamayabilir. Dolayısıyla her optimizasyon problemi bazı yaygın problemlere benzese de, kendi özel yapısı gözönüne alınarak ele alınmalıdır.

eğilme mukavemeti açısından aynı sıralamanın oluşmasını garanti etmez. Fakat eğilme mukavemetinin esas alınması da aynı şekilde eksenel yük etkisi altında aynı sıralamanın oluşacağını garanti edemez. Dolayısıyla, problem özelinde kesit listelerinin farklı indisleme stratejisiyle ele alınması daha mantıklı olabilir. Örneğin, eksenel kuvvete maruz kalan elemanlarda kesit alanı esas alınırken, eğilme etkisin altındaki elemanlarda eğilme mukavemeti esas alınabilir. Veya daha etkili olacak farklı bir strateji geliştirebiliriz.

7.4.1 İndisleme Stratejisi

- Kesit grupları
- Eleman numaralandırma
- Düğüm noktaları
- Yükleme durumları

$$x_i = \text{ind}(A_i), \quad i = 1, \ldots, n \tag{46}$$

[38]

[38] İndisleme, ayrık optimizasyon problemlerinin çözümünü kolaylaştırır ve hesaplama verimliliğini artırır.

7.4.2 Veri Yapıları

- Kesit özellikleri tablosu
- Bağlantı matrisi
- Kısıt matrisi
- İndis dönüşüm tablosu

Veri Yapısı Örneği

```
sections = {
    1: {'A': 10.3, 'I': 171},
    2: {'A': 13.2, 'I': 375},
    ...
}
```

7.5 Optimizasyon Sonuçlarının Değerlendirilmesi

Ayrık optimizasyon problemlerinin çözüm kalitesinin ve performansının analizi.

7.5.1 Performans Ölçütleri

- Toplam ağırlık
- Maksimum gerilme oranı
- Maksimum deplasman
- Hesaplama süresi

7.5.2 Sonuçların Görselleştirilmesi

- Yakınsama grafikleri
- Gerilme dağılımları
- Deplasman şekilleri
- Kesit dağılımları

8 Sürekli Parametrelerin Optimizasyonu

Sürekli parametrelerin optimizasyonuyla, yapısal mühendislikte ve diğer mühendislik alanlarında yaygın olarak karşılaşılır. Bu bölümde, sürekli değişkenlerle ifade edilen optimizasyon problemlerinin temel özellikleri, matematiksel formülasyonu ve çözüm yöntemleri incelenecektir.

8.1 Sürekli Optimizasyonun Temel Kavramları

Sürekli optimizasyon, tasarım değişkenlerinin sürekli değerler alabildiği optimizasyon problemlerini ifade eder. Bu tür problemlerde, tasarım uzayı sonsuz sayıda noktadan oluşur ve değişkenler herhangi bir reel değer alabilir.

8.1.1 Sürekli Tasarım Değişkenleri

Sürekli tasarım değişkenleri, belirli bir aralıkta herhangi bir değer alabilen parametrelerdir. Örneğin:

- Bir kirişin kesit boyutları (genişlik, yükseklik)
- Malzeme özellikleri (elastisite modülü, yoğunluk)
- Geometrik parametreler (açılar, uzunluklar)
- Kontrol parametreleri (kuvvet büyüklükleri, sönümleme katsayıları)

8.1.2 Sürekli Optimizasyon Problemlerinin Genel Formu

Sürekli optimizasyon problemleri genellikle aşağıdaki formda ifade edilir:

$$\min_{\mathbf{x}} \quad f(\mathbf{x}) \tag{47}$$

$$\text{s.t.} \quad g_i(\mathbf{x}) \leq 0, \quad i = 1, 2, \ldots, m \tag{48}$$

$$h_j(\mathbf{x}) = 0, \quad j = 1, 2, \ldots, p \tag{49}$$

$$\mathbf{x}_L \leq \mathbf{x} \leq \mathbf{x}_U \tag{50}$$

Burada:

- $\mathbf{x} \in \mathbb{R}^n$: Tasarım değişkenleri vektörü
- $f(\mathbf{x})$: Amaç fonksiyonu
- $g_i(\mathbf{x})$: Eşitsizlik kısıtları
- $h_j(\mathbf{x})$: Eşitlik kısıtları
- $\mathbf{x}_L, \mathbf{x}_U$: Alt ve üst sınırlar

Sürekli Optimizasyon Örneği

Bir konsol kirişin ağırlık minimizasyonu problemi:

$$\min_{b,h} \quad \rho \cdot L \cdot b \cdot h \tag{51}$$

$$\text{s.t.} \quad \sigma_{max} = \frac{6PL}{bh^2} \leq \sigma_{allow} \tag{52}$$

$$\delta_{max} = \frac{PL^3}{3EI} \leq \delta_{allow} \tag{53}$$

$$b_{min} \leq b \leq b_{max} \tag{54}$$

$$h_{min} \leq h \leq h_{max} \tag{55}$$

Burada b ve h sırasıyla kirişin genişliği ve yüksekliğidir.

[39]

8.2 Sürekli Optimizasyon Problemlerinin Matematiksel Formülasyonu

Sürekli optimizasyon problemlerinin matematiksel formülasyonu, problemin doğasını ve çözüm yöntemlerini belirleyen temel bir adımdır. Bu formülasyon, amaç fonksiyonu, kısıtlar ve tasarım değişkenlerinin matematiksel olarak ifade edilmesini içerir.

8.2.1 Amaç Fonksiyonu

Amaç fonksiyonu, optimize edilmek istenen mühendislik performans ölçütünü matematiksel olarak ifade eder. Yapısal optimizasyon problemlerinde yaygın olarak kullanılan amaç fonksiyonları şunlardır:

- **Ağırlık minimizasyonu:** $f(\mathbf{x}) = \sum_{i=1}^{n} \rho_i V_i(\mathbf{x})$
- **Esneklik minimizasyonu:** $f(\mathbf{x}) = \mathbf{F}^T \mathbf{u}(\mathbf{x})$
- **Gerilme minimizasyonu:** $f(\mathbf{x}) = \max_i \sigma_i(\mathbf{x})$
- **Deplasman minimizasyonu:** $f(\mathbf{x}) = \max_i |u_i(\mathbf{x})|$
- **Frekans maksimizasyonu:** $f(\mathbf{x}) = -\omega_1(\mathbf{x})$ (ilk doğal frekans)
- **Maliyet minimizasyonu:** $f(\mathbf{x}) = \sum_{i=1}^{n} c_i x_i$

8.2.2 Kısıt Fonksiyonları

Kısıt fonksiyonları, tasarımın belirli gereksinimleri karşılamasını sağlayan matematiksel ifadelerdir. Yapısal optimizasyon problemlerinde sıklıkla kullanılan kısıtlar şunlardır:

- **Gerilme kısıtları:** $g_i(\mathbf{x}) = \sigma_i(\mathbf{x}) - \sigma_{allow} \leq 0$
- **Deplasman kısıtları:** $g_i(\mathbf{x}) = |u_i(\mathbf{x})| - u_{allow} \leq 0$
- **Burkulma kısıtları:** $g_i(\mathbf{x}) = P_{cr,i}(\mathbf{x}) - P_{applied} \leq 0$
- **Frekans kısıtları:** $g_i(\mathbf{x}) = \omega_{min} - \omega_i(\mathbf{x}) \leq 0$
- **Denge kısıtları:** Yapının statik denge koşullarını sağlaması gerektiğini ifade eder.
- **Geometrik kısıtlar:** Tasarım değişkenlerinin belirli geometrik ilişkileri sağlaması gerektiğini ifade eder.

8.2.3 Duyarlılık Analizi

Duyarlılık analizi, amaç fonksiyonu ve kısıtların tasarım değişkenlerine göre türevlerinin hesaplanmasını içerir. Bu türevler, gradyan tabanlı optimizasyon algoritmalarında arama yönünü belirlemek için kullanılır.

$$\nabla f(\mathbf{x}) = \left[\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_n} \right]^T \tag{56}$$

$$\nabla g_i(\mathbf{x}) = \left[\frac{\partial g_i}{\partial x_1}, \frac{\partial g_i}{\partial x_2}, \ldots, \frac{\partial g_i}{\partial x_n} \right]^T \tag{57}$$

Duyarlılık analizi için kullanılan yöntemler:

[39] Sürekli optimizasyon problemlerinde, amaç fonksiyonu ve kısıtlar genellikle sürekli ve türevlenebilir fonksiyonlardır, bu da gradyan tabanlı optimizasyon yöntemlerinin kullanılmasına olanak sağlar.

- **Analitik yöntemler:** Türevlerin doğrudan matematiksel ifadelerle hesaplanması
- **Sonlu farklar yöntemi:** Nümerik yaklaşımla türevlerin hesaplanması
- **Adjoint yöntem:** Karmaşık sistemlerde verimli duyarlılık hesaplaması için kullanılır
- **Otomatik türev alma:** Bilgisayar programlarının otomatik olarak türev hesaplaması

8.2.4 Sürekli Optimizasyonun Ayırt Edici Özellikleri

Sürekli optimizasyon problemleri, ayrık optimizasyon problemlerinden farklı olarak, tasarım değişkenlerinin sürekli değerler alabildiği problemlerdir. Bu tür problemlerin ayırt edici özellikleri şunlardır:

- **Sürekli tasarım uzayı:** Tasarım değişkenleri reel sayılar kümesinden değerler alabilir, bu da sonsuz sayıda olası çözüm anlamına gelir.
- **Türevlenebilirlik:** Amaç fonksiyonu ve kısıtlar genellikle türevlenebilir fonksiyonlardır, bu da gradyan tabanlı optimizasyon yöntemlerinin kullanılabilmesini sağlar.
- **Konvekslik:** Problem formülasyonunun konveks olup olmaması, global optimuma ulaşılabilirliği belirler. Konveks problemlerde, lokal optimum aynı zamanda global optimumdur.
- **Süreklilik:** Fonksiyonların sürekli olması, optimizasyon algoritmasının daha kararlı çalışmasını sağlar.
- **Diferansiyellenebilirlik:** Yüksek dereceden türevlerin varlığı, Newton benzeri yöntemlerin kullanılabilmesine olanak tanır.

Sürekli optimizasyon problemleri, yapısal mühendislikte kesit boyutları, malzeme özellikleri veya geometrik parametreler gibi değişkenlerin optimize edilmesinde yaygın olarak kullanılır. Bu problemlerin çözümünde, gradyan tabanlı yöntemler (Newton yöntemi, eşlenik gradyan yöntemi), gradyan gerektirmeyen yöntemler (Nelder-Mead simpleks yöntemi) veya meta-sezgisel algoritmalar (genetik algoritma, parçacık sürü optimizasyonu) kullanılabilir.

8.3 Sürekli Optimizasyonun Yapısal Optimizasyondaki Genel Uygulamaları

Bu noktada yaygın iki optimizasyon problemi örneklendirilmiş olsa da, parametrelere bağlı olarak hesaplanan birçok çıktı optimizasyon problemi olarak ele alınabilir ve iyileştirilebilir.

8.3.1 Öntanımlı Olmayan Boyut Optimizasyonu

Öntanımlı olmayan boyut optimizasyonu, yapısal elemanların kesit boyutlarının standart katalog değerleriyle sınırlı olmadan, sürekli değişkenler olarak ele alınmasını içerir. Bu yaklaşım, tasarımcılara daha geniş bir tasarım uzayı sunar ve potansiyel olarak daha verimli yapılar elde edilmesini sağlar. Geleneksel yaklaşımlarda, yapı elemanları için belirli standart kesitler (örneğin, I-profiller, kutu profiller) arasından seçim yapılırken, öntanımlı olmayan optimizasyonda kesit özellikleri (alan, atalet momenti, vb.) doğrudan tasarım değişkenleri olarak kullanılır.

Bu tür optimizasyon problemlerinde, genellikle minimum ağırlık veya maksimum rijitlik gibi amaçlar gözetilirken, gerilme, deplasman ve burkulma gibi

yapısal kısıtlar dikkate alınır. Optimizasyon sonucunda elde edilen kesit özellikleri, daha sonra üretilebilir kesitlere dönüştürülmek üzere yorumlanır veya özel kesitler olarak üretilir. Öntanımlı olmayan boyut optimizasyonu, özellikle havacılık, uzay ve otomotiv endüstrilerinde, malzeme kullanımını minimize ederken performansı maksimize etmek için yaygın olarak kullanılmaktadır.

8.3.2 Topolojik Optimizasyon

Topolojik optimizasyon, bir yapının en verimli malzeme dağılımını belirlemek için kullanılan ileri bir yapısal optimizasyon yöntemidir. Bu yöntem, belirli bir tasarım alanı içinde malzemenin nerede bulunması ve nerede bulunmaması gerektiğine karar vererek, yapının temel formunu ve bağlantı yapısını optimize eder. Geleneksel optimizasyon yöntemlerinden farklı olarak, sadece boyutları veya şekli değil, yapının topolojisini de değiştirebilir.

Topolojik optimizasyon süreci genellikle bir tasarım alanının sonlu elemanlara bölünmesiyle başlar ve her elemana malzeme yoğunluğunu temsil eden bir tasarım değişkeni atanır. Optimizasyon algoritması, belirli kısıtlar altında (örneğin maksimum ağırlık veya minimum esneklik) bu değişkenleri ayarlayarak en iyi malzeme dağılımını arar. Sonuç olarak, genellikle doğada bulunan yapılara benzeyen, yüksek verimli ve hafif yapılar ortaya çıkar. Bu yöntem, otomotiv ve havacılık endüstrilerinde hafif parçalar tasarlamak, medikal implantlar geliştirmek ve 3D baskı teknolojileri için optimize edilmiş yapılar oluşturmak gibi çeşitli alanlarda yaygın olarak kullanılmaktadır.

9 Yapısal Optimizasyona Giriş

Yapısal optimizasyon, her ne kadar farklı bir optimizasyon türü olarak ele alınmış olsa da özü itibariyle tüm optimizasyon problemleriyle benzer prensiperle ilerler. Ancak problemin tanımlanması, dolayısıyla da efektif çözüm algoritmalarının seçilebilmesi mühendis yargısına bağlıdır. Bu bölümde klasik optimizasyon başlıkları altında tanımlanan kavramların yapısal optimizasyon bağlamında nasıl bir bağlama dönüştüğü anlatılmaya çalışılacaktır.

9.1 Yapısal Optimizasyon Terminolojisi

9.1.1 Amaç Fonksiyonları

Yapısal optimizasyonda amaç fonksiyonu, optimize edilmek istenen mühendislik hedefini matematiksel olarak ifade eder. Yapısal tasarımda en yaygın kullanılan amaç fonksiyonları şunlardır:[40]

- **Ağırlık minimizasyonu:** Yapının toplam ağırlığını en aza indirmeyi hedefler. Özellikle havacılık ve uzay yapılarında kritik öneme sahiptir.
- **Maliyet minimizasyonu:** Yapının üretim, malzeme ve işçilik maliyetlerini en aza indirmeyi amaçlar.
- **Rijitlik maksimizasyonu:** Yapının belirli yükler altında deformasyona karşı direncini en üst düzeye çıkarmayı hedefler.
- **Dayanım maksimizasyonu:** Yapının taşıyabileceği maksimum yükü artırmayı amaçlar.
- **Enerji sönümleme:** Dinamik yükler altında yapının enerji sönümleme kapasitesini optimize eder.

[40] Çok amaçlı optimizasyon problemlerinde, birden fazla amaç fonksiyonu ağırlıklandırılarak tek bir fonksiyona dönüştürülebilir veya Pareto-optimal çözümler aranabilir.

9.1.2 Kısıtlar

Yapısal optimizasyonda kısıtlar, tasarımın uygulanabilir olması için sağlanması gereken şartları tanımlar. Bu kısıtlar, klasik optimizasyon problemlerindeki matematiksel kısıtların yapısal mühendislik bağlamındaki karşılıklarıdır:

- **Gerilme kısıtları:** Yapıdaki gerilmelerin izin verilen maksimum değerleri aşmamasını sağlar. Örneğin: $\sigma_i \leq \sigma_{izin}$
- **Deplasman kısıtları:** Yapıdaki yer değiştirmelerin belirli sınırlar içinde kalmasını sağlar. Örneğin: $\delta_i \leq \delta_{izin}$
- **Burkulma kısıtları:** Yapı elemanlarının burkulma yüklerinin, uygulanan yüklerden belirli bir güvenlik faktörü kadar büyük olmasını sağlar.
- **Titreşim kısıtları:** Yapının doğal frekanslarının belirli değerlerin üzerinde veya altında olmasını sağlar.
- **Geometrik kısıtlar:** Yapısal optimizasyon bağlamında, tasarım değişkenlerinin fiziksel olarak uygulanabilir sınırlar içinde kalmasını sağlar. Örneğin:
  - Minimum ve maksimum kesit boyutları
  - Minimum duvar kalınlıkları
  - Elemanlar arası bağlantı gereksinimleri[41]
  - Montaj ve üretim kısıtlamaları
- **Denge kısıtları:** Yapının statik denge koşullarını sağlaması gerektiğini ifade eder.
- **Uyumluluk kısıtları:** Deformasyonların sürekli ve uyumlu olması gerektiğini belirtir.

[41] Örneğin çelik bir yapının üst katlarında kullanılan kesit boyutları, alt katlarındaki kesit boyutlarından daha büyük olması istenebilir ve bu aplikasyon açısından da oldukça mantıklıdır.

Yapısal Optimizasyon Kısıtları Örneği

Bir köprü tasarımında:

$$\sigma_{max} \leq 250 \text{ MPa} \quad \text{(Gerilme kısıtı)} \tag{58}$$
$$\delta_{orta} \leq L/400 \quad \text{(Deplasman kısıtı)} \tag{59}$$
$$f_1 \geq 2.0 \text{ Hz} \quad \text{(Titreşim kısıtı)} \tag{60}$$
$$t_{min} \geq 8 \text{ mm} \quad \text{(Geometrik kısıt)} \tag{61}$$

[42]

[42] Kısıtların matematiksel formülasyonu, sonlu eleman analizinin sonuçlarına dayalı olarak ifade edilir ve genellikle doğrusal olmayan fonksiyonlar şeklindedir.

9.1.3 Tasarım Değişkenleri

Yapısal optimizasyonda tasarım değişkenleri, optimize edilecek parametreleri temsil eder. Bu değişkenler, optimizasyon algoritması tarafından değiştirilebilen ve en iyi çözümü bulmak için ayarlanabilen parametrelerdir. Yapısal mühendislikte yaygın kullanılan tasarım değişkenleri şunlardır:

- **Kesit özellikleri:**
  - Profil boyutları (genişlik, yükseklik)
  - Duvar kalınlıkları
  - Kesit alanı
  - Atalet momenti

- **Malzeme özellikleri:**
  - Elastisite modülü
  - Yoğunluk
  - Akma dayanımı
- **Geometrik parametreler:**
  - Düğüm noktalarının koordinatları
  - Eğrilik yarıçapları
  - Açılar
- **Topolojik parametreler:**
  - Malzeme varlığı/yokluğu (0-1 değişkenleri)
  - Malzeme yoğunluğu (0-1 arasında değişen sürekli değişkenler)[43]
  - Bağlantı noktalarının varlığı

Tasarım Değişkenleri Gösterimi

Tipik bir çelik çerçeve optimizasyonunda tasarım değişkenleri şu şekilde gösterilebilir:

$$\mathbf{x} = [A_1, A_2, \ldots, A_n, I_{y1}, I_{y2}, \ldots, I_{yn}, I_{z1}, I_{z2}, \ldots, I_{zn}]^T \tag{62}$$

Burada A_i kesit alanlarını, I_{yi} ve I_{zi} ise atalet momentlerini temsil eder.

[43] Optimizasyon veya regresyon benzeri birçok kodlama gerektiren hesaplama yönteminde, verilerin daha standart şekilde ele alınabilmesi için normalizasyon adlı bir yaklaşım kullanılır. Bu yaklaşım, verilerin 0-1 arasında değişen bir değer aralığına sahip olmasını sağlar. Mevcut veriler içerisindeki en küçük veri 0, en büyük veri ise 1 olarak dönüştürülür. Tüm ara değerler ise bu aralıkta oransal bir değer alır.

9.2 Yapısal Optimizasyon Kategorileri

Yapısal optimizasyon problemleri kategorik olarak (Şekil, boyut, topoloji vb.) gibi bazı temel başlıklara ayrılabilir. Ancak bir mühendis, birbiriyle çelişen çıktıları üreten parametrelerin olduğu her sorunu bir optimizasyon problemi olarak ele alabilir.

Örneğin bir yapıyı hafifletmek çoğu zaman gerilme kapasitelerinden feragat etmek anlamına gelebilir. Bu çelişen çıktılar, aynı parametrelerin sonucudur.

9.2.1 Boyutlandırma Optimizasyonu

Boyutlandırma optimizasyonu, yapının genel geometrisi sabit tutularak elemanların kesit boyutlarının optimize edilmesidir. En temel ve yaygın kullanılan yapısal optimizasyon yaklaşımıdır.

- **Tasarım değişkenleri:** Kesit alanı, kalınlık, genişlik-yükseklik gibi kesit özellikleri
- **Avantajları:**
  - Matematiksel olarak nispeten daha basit formülasyon
  - Mevcut tasarımların iyileştirilmesi için uygun
  - Endüstride yaygın kullanım
- **Uygulama alanları:** Çelik yapılar, çerçeve sistemler, kafes sistemler

9.2.2 Şekil Optimizasyonu

Şekil optimizasyonu, yapı elemanlarının şekillerinin veya düğüm noktalarının konumlarının değiştirilmesiyle gerçekleştirilir. Yapının genel topolojisi korunurken sınır geometrisi değiştirilir.

- **Tasarım değişkenleri:** Düğüm noktalarının koordinatları, eğrilik parametreleri, kontrol noktaları
- **Avantajları:**
  - Boyutlandırma optimizasyonuna göre daha fazla tasarım esnekliği
  - Gerilme yoğunlaşmalarının azaltılmasında etkili
- **Zorluklar:**
  - Geometrik değişimler sonlu eleman ağının yeniden oluşturulmasını gerektirebilir
  - Karmaşık matematiksel formülasyon
- **Uygulama alanları:** Havacılık yapıları, otomotiv parçaları, köprü konstrüksiyonları

9.2.3 Topoloji Optimizasyonu

Topoloji optimizasyonu, yapının temel yapısının veya topolojisinin değiştirilmesiyle gerçekleştirilir. Malzemenin yapı içindeki dağılımı optimize edilir ve genellikle malzemenin olması veya olmaması gereken bölgeler belirlenir.

- **Tasarım değişkenleri:** Malzeme yoğunluğu, malzeme varlığı/yokluğu
- **Avantajları:**
  - En yüksek tasarım serbestliği
  - Yenilikçi ve öngörülemeyen tasarımlar üretebilme
  - Malzeme kullanımında önemli tasarruf potansiyeli
- **Zorluklar:**
  - Matematiksel ve hesaplamalı olarak karmaşık
  - Üretilebilirlik kısıtlarının uygulanması zor olabilir
  - Sonuçların yorumlanması ve uygulanabilir tasarımlara dönüştürülmesi
- **Uygulama alanları:** Havacılık, otomotiv, medikal implantlar, 3D baskı yapıları

Örnek: Konsol Kiriş Optimizasyonu

Aynı konsol kiriş probleminin üç farklı yaklaşımla optimizasyonu:
Boyutlandırma: Kiriş kesitinin yüksekliğinin uzunluk boyunca değişimi optimize edilir.
Şekil: Kirişin alt ve üst yüzeylerinin şekli optimize edilir.
Topoloji: Kirişin iç yapısındaki malzeme dağılımı optimize edilir, sonuçta genellikle kafes benzeri bir yapı ortaya çıkar.

9.3 Yapısal Optimizasyon Formülasyonu

Bir yapısal optimizasyon problemi, matematiksel olarak şu şekilde ifade edilebilir:

$$\text{Minimize: } f(\mathbf{x}) \tag{63}$$
$$\text{Kısıtlar: } g_j(\mathbf{x}) \leq 0, \quad j = 1, 2, \ldots, m \tag{64}$$
$$h_k(\mathbf{x}) = 0, \quad k = 1, 2, \ldots, p \tag{65}$$
$$\mathbf{x}_L \leq \mathbf{x} \leq \mathbf{x}_U \tag{66}$$

Burada:

- $\mathbf{x}$: tasarım değişkenleri vektörü
- $f(\mathbf{x})$: amaç fonksiyonu (minimizasyon problemi için)
- $g_j(\mathbf{x})$: eşitsizlik kısıtları
- $h_k(\mathbf{x})$: eşitlik kısıtları
- $\mathbf{x}_L$ ve $\mathbf{x}_U$: tasarım değişkenlerinin alt ve üst sınırları

9.3.1 Sonlu Eleman Analizi ile Bağlantı

Yapısal optimizasyon problemlerinde, amaç fonksiyonu ve kısıtlar genellikle sonlu eleman analizi (FEA) sonuçlarına bağlıdır. Bu bağlantı aşağıdaki şekilde ifade edilebilir:

$$\mathbf{K}(\mathbf{x})\mathbf{u} = \mathbf{F} \tag{67}$$
$$f(\mathbf{x}) = f(\mathbf{x}, \mathbf{u}(\mathbf{x})) \tag{68}$$
$$g_j(\mathbf{x}) = g_j(\mathbf{x}, \mathbf{u}(\mathbf{x})) \tag{69}$$
$$h_k(\mathbf{x}) = h_k(\mathbf{x}, \mathbf{u}(\mathbf{x})) \tag{70}$$

Burada:

- $\mathbf{K}(\mathbf{x})$: tasarım değişkenlerine bağlı rijitlik matrisi
- $\mathbf{u}$: deplasman vektörü
- $\mathbf{F}$: dış kuvvet vektörü

Yapısal Optimizasyon Algoritması Seçimi

Yapısal optimizasyon problemlerinde algoritma seçimi şu faktörlere bağlıdır:

- Problem boyutu (tasarım değişkeni sayısı)
- Kısıt sayısı ve karmaşıklığı
- Fonksiyon değerlendirmelerinin hesaplama maliyeti
- Tasarım uzayının karakteristiği (çoklu yerel optimumların varlığı)
- Duyarlılık bilgisinin mevcudiyeti

10 Topolojik Optimizasyon

Yapısal sistemlerin en temel formunu belirlemeyi amaçlayan topolojik optimizasyon yöntemleri bu bölümde incelenecektir. Malzeme dağılımının optimizasyonu ve modern topoloji optimizasyonu teknikleri ele alınacaktır.

10.1 Topolojik Optimizasyonun Temelleri

Topolojik optimizasyon, bir yapının en verimli malzeme dağılımını belirlemek için kullanılan ileri bir yapısal optimizasyon yöntemidir. Geleneksel optimizasyon yöntemlerinden farklı olarak, topolojik optimizasyon sadece boyutları veya şekli değil, yapının temel formunu ve bağlantı yapısını da optimize eder. Bu yaklaşımda, belirli bir tasarım alanı içinde malzemenin nerede bulunması ve nerede bulunmaması gerektiğine karar verilir.

Topolojik optimizasyon süreci genellikle bir tasarım alanının sonlu elemanlara bölünmesiyle başlar. Her elemana, malzeme yoğunluğunu temsil eden 0 ile 1 arasında değişen bir tasarım değişkeni atanır. Optimizasyon algoritması, belirli kısıtlar altında (örneğin, maksimum ağırlık veya minimum esneklik) bu değişkenleri ayarlayarak en iyi malzeme dağılımını arar. Sonuç olarak, genellikle doğada bulunan yapılara benzeyen, yüksek verimli ve hafif yapılar ortaya çıkar.

Bu yöntem, otomotiv ve havacılık endüstrilerinde hafif ve dayanıklı parçalar tasarlamak, medikal implantlar geliştirmek ve 3D baskı teknolojileri için optimize edilmiş yapılar oluşturmak gibi çeşitli alanlarda yaygın olarak kullanılmaktadır. Topolojik optimizasyon, mühendislere geleneksel tasarım yaklaşımlarıyla elde edilmesi zor olan yenilikçi ve verimli çözümler sunma imkanı sağlar.

10.1.1 Temel Kavramlar

- Malzeme dağılımı [44]
- Yapısal topoloji [45]
- Homojenizasyon [46]
- Tasarım değişkenleri [47]

$$\min_{x \in [0,1]^n} \quad c(x) = F^T U(x) \tag{71}$$

[44] Tasarım alanı içinde malzemenin nasıl yerleştirildiğini gösteren, genellikle yoğunluk değişkenleriyle ifade edilen dağılım.

[45] Bir yapının temel formunu, bağlantı yapısını ve malzeme dağılımını tanımlayan geometrik düzen.

[46] Mikro yapıların makro özelliklerini belirlemek için kullanılan, kompozit malzemelerin efektif özelliklerini hesaplama yöntemi.

[47] Optimizasyon sürecinde değiştirilebilen, genellikle her sonlu elemana atanan ve malzeme varlığını temsil eden parametreler.

10.2 Sonlu Elemanlar Yöntemi ve Optimizasyon İlişkisi

Topolojik optimizasyon, sonlu elemanlar yöntemi (FEM) ile doğrudan ilişkilidir ve bu yöntem optimizasyon sürecinin temel bileşenidir. Sonlu elemanlar yöntemi, karmaşık geometrileri daha küçük ve basit elemanlara bölerek analiz etmeyi sağlar, bu da topolojik optimizasyon için gerekli olan yapısal davranışın hassas bir şekilde hesaplanmasına olanak tanır.

Optimizasyon sürecinde, her iterasyonda malzeme dağılımı değiştiğinde, yapının mekanik davranışı (gerilmeler, deplasmanlar, doğal frekanslar vb.) sonlu elemanlar analizi ile yeniden hesaplanır. Bu analiz sonuçları, optimizasyon algoritmasının bir sonraki adımda hangi bölgelerde malzeme ekleneceğine veya çıkarılacağına karar vermesini sağlar. Böylece FEM, topolojik optimizasyonun hem analiz hem de karar verme mekanizmasının ayrılmaz bir parçası haline gelir.

10.2.1 FEM Formülasyonu

- Rijitlik matrisi
- Yük vektörü
- Deplasman alanı
- Eleman tipleri

$$K(x)U = F \tag{72}$$

10.2.2 Sonlu Eleman Modelinin API ile Oluşturulması

Sonlu eleman modellerinin oluşturulması ve analizi için çeşitli yazılımlar Application Programming Interface (API) sunmaktadır. Bu API'ler, topolojik optimizasyon algoritmalarının sonlu eleman analizleriyle entegre çalışmasını sağlar. Özellikle otomatik iterasyon gerektiren optimizasyon süreçlerinde, API kullanımı manuel model oluşturma ve analiz süreçlerini ortadan kaldırarak büyük verimlilik sağlar.

SAP2000 OAPI (Open Application Programming Interface), yapısal analiz ve optimizasyon için yaygın kullanılan bir API örneğidir. Bu arayüz, Python, MATLAB veya C++ gibi programlama dilleri aracılığıyla SAP2000 yazılımının tüm özelliklerine erişim sağlar. Topolojik optimizasyon sürecinde, algoritma her iterasyonda SAP2000 OAPI kullanarak:

- Güncellenmiş malzeme özelliklerini modele uygulayabilir
- Analizi otomatik olarak çalıştırabilir
- Analiz sonuçlarını (gerilmeler, deplasmanlar, vb.) okuyabilir
- Bu sonuçlara göre yeni malzeme dağılımını hesaplayabilir

Bu tür API entegrasyonları, topolojik optimizasyon sürecinin tamamen otomatikleştirilmesini sağlayarak, karmaşık yapıların bile verimli bir şekilde optimize edilmesine olanak tanır. Ayrıca ANSYS, Abaqus ve NASTRAN gibi diğer sonlu eleman yazılımları da benzer API'ler sunmaktadır. İlerleyen konularda SAP2000 OAPI'nin kullanıldığı daha detaylı örnekler incelenecektir.

10.3 Yoğunluk Tabanlı Yöntemler

Yoğunluk tabanlı topolojik optimizasyon yöntemlerinin en yaygın kullanılanı SIMP (Solid Isotropic Material with Penalization) yöntemidir. Bu yöntem, her sonlu elemana 0 ile 1 arasında değişen bir yoğunluk değişkeni atayarak çalışır. Burada 0 malzeme yokluğunu, 1 ise tam malzeme varlığını temsil eder.

SIMP yönteminin temel prensibi, ara yoğunluk değerlerini (0 ile 1 arasındaki değerler) cezalandırarak, optimizasyon sonucunda daha belirgin bir 0-1 dağılımı elde etmektir. Bu, malzeme özelliklerinin (örneğin elastisite modülü) yoğunluk değişkeninin bir üs fonksiyonu olarak tanımlanmasıyla sağlanır. Ceza parametresi genellikle 3 veya daha yüksek bir değer olarak seçilir.

SIMP yöntemi, otomotiv parçalarının hafifletilmesi, uçak yapısal elemanlarının optimizasyonu ve medikal implantların tasarımı gibi çeşitli mühendislik uygulamalarında başarıyla kullanılmaktadır. Yöntem, matematiksel olarak iyi tanımlanmış olması ve gradyan tabanlı optimizasyon algoritmaları ile uyumlu çalışması sayesinde endüstride standart bir yaklaşım haline gelmiştir.

10.3.1 SIMP Yöntemi

Solid Isotropic Material with Penalization:

$$E(x) = E_{min} + x^p(E_0 - E_{min}) \tag{73}$$

- Yoğunluk değişkenleri: $x \in [0, 1]$
- Ceza parametresi: $p > 1$
- Minimum rijitlik: E_{min}
- Tam malzeme rijitliği: E_0

SIMP Yönteminin Avantajları

- Basit implementasyon
- Hızlı yakınsama
- Ara yoğunlukların penalizasyonu
- Endüstriyel uygulamalarda yaygın kullanım

10.4 ESO ve BESO Yöntemleri

Evolutionary Structural Optimization (ESO) ve Bi-directional Evolutionary Structural Optimization (BESO) yöntemleri, yapısal topoloji optimizasyonunda kullanılan sezgisel yaklaşımlardır. ESO yöntemi, "verimsiz malzemeyi kademeli olarak kaldır" prensibine dayanır ve düşük gerilme veya enerji yoğunluğuna sahip elemanları yapıdan çıkararak optimum tasarıma ulaşmayı hedefler. BESO ise ESO'nun geliştirilmiş bir versiyonudur ve sadece malzeme çıkarma değil, aynı zamanda gerekli bölgelere malzeme ekleme işlemini de içerir. Bu yöntemler, matematiksel olarak SIMP kadar sağlam bir temele sahip olmasa da, uygulanması kolay ve sezgisel olarak anlaşılabilir olmaları nedeniyle mühendislik uygulamalarında tercih edilmektedir.

10.5 Level-Set Yöntemi

Level-Set yöntemi, topoloji optimizasyonunda yapı sınırlarını açık bir şekilde tanımlamak için kullanılan matematiksel bir yaklaşımdır. Bu yöntemde, yapının sınırları bir seviye kümesi fonksiyonunun sıfır seviye eğrisi (veya yüzeyi) olarak temsil edilir. Optimizasyon süreci boyunca, bu seviye kümesi fonksiyonu Hamilton-Jacobi denklemleri kullanılarak güncellenir ve böylece yapı sınırları pürüzsüz bir şekilde evrilir. Level-Set yöntemi, keskin ve net sınırlar oluşturma, topoloji değişikliklerini doğal bir şekilde ele alma ve üretilebilirlik kısıtlarını kolayca dahil etme gibi avantajlara sahiptir. Özellikle akışkan-yapı etkileşimi problemleri ve çok malzemeli tasarımlarda etkili sonuçlar vermektedir.

11 Boyut ve Şekil Optimizasyonu

Yapısal sistemlerin boyut ve şekil parametrelerinin optimizasyonu, daha genel bir ifadeyle kesit optimizasyonu olarak da adlandırılabilir. Probleme bağlı olarak bunlardan biri veya ikisi aynı anda problemin parametresi haline gelebilir.

11.1 Boyut Optimizasyonunun Temelleri

Boyut optimizasyonu, yapısal sistemlerin kesit özelliklerinin (genişlik, yükseklik, kalınlık, vb.) en uygun değerlerini belirlemek için kullanılan bir optimizasyon yöntemidir. Bu yöntem, yapının topolojisini değiştirmeden, sadece elemanların boyutlarını değiştirerek optimum tasarıma ulaşmayı hedefler.

11.1.1 Problem Parametreleri

Boyut optimizasyonunda kullanılan tasarım değişkenleri genellikle şunları içerir:

- **Kesit boyutları:** Kirişlerin genişlik ve yükseklikleri, plakaların kalınlıkları
- **Kesit alanları:** Çubuk elemanların kesit alanları
- **Atalet momentleri:** Kirişlerin eğilme ve burulma rijitliklerini belirleyen parametreler

- **Malzeme özellikleri:** Elastisite modülü, yoğunluk gibi değişkenler
- **Takviye elemanları:** Güçlendirme elemanlarının boyutları ve konumları

11.1.2 Problem Çıktıları

Boyut optimizasyonu sonucunda elde edilen çıktılar şunlardır:

- **Optimum kesit boyutları:** Her yapı elemanı için en uygun boyutlar
- **Minimum ağırlık/maliyet:** Optimizasyon sonucunda elde edilen yapının toplam ağırlığı veya maliyeti
- **Yapısal performans göstergeleri:** Gerilmeler, deplasmanlar, doğal frekanslar
- **Malzeme kullanım verimliliği:** Her elemanın taşıma kapasitesinin ne kadar verimli kullanıldığı
- **Duyarlılık bilgileri:** Tasarım değişkenlerindeki değişimlerin amaç fonksiyonu üzerindeki etkileri

11.1.3 Kısıtlayıcılar ve Karar Mekanizması

Boyut optimizasyonunda, çeşitli kısıtlayıcılar problemin çözüm uzayını sınırlandırır ve karar mekanizmasını etkiler:

- **Gerilme kısıtları:** $\sigma_{max} \leq \sigma_{allow}$
  - Yapıdaki maksimum gerilmelerin izin verilen değerleri aşmaması gerekir
  - Elemanların boyutlarını artırma yönünde etki yapar
- **Deplasman kısıtları:** $\delta_{max} \leq \delta_{allow}$
  - Yapıdaki maksimum yer değiştirmelerin belirli sınırlar içinde kalmasını sağlar
  - Genellikle yapının rijitliğini artırma yönünde etki eder
- **Burkulma kısıtları:** $P_{cr} \geq P_{design}$
  - Basınç elemanlarının burkulma yüklerinin tasarım yükünden büyük olmasını sağlar
  - Narin elemanların boyutlarını artırma yönünde etki yapar
- **Frekans kısıtları:** $\omega_i \geq \omega_{min}$ veya $\omega_i \leq \omega_{max}$
  - Yapının doğal frekanslarının belirli aralıklarda olmasını sağlar
  - Dinamik yüklere maruz yapılarda önemlidir
- **Üretilebilirlik kısıtları:** $x_{min} \leq x \leq x_{max}$
  - Tasarım değişkenlerinin pratik üretim sınırları içinde kalmasını sağlar
  - Çözüm uzayını gerçekçi değerlerle sınırlar
- **Geometrik kısıtlar:** Örneğin $h/b \leq \alpha$
  - Kesit oranlarının belirli sınırlar içinde kalmasını sağlar
  - Yerel burkulma ve stabilite sorunlarını önler

Boyut Optimizasyonu Süreci

1. Başlangıç tasarımının oluşturulması
2. Yapısal analiz (FEM) ile performans değerlendirmesi
3. Duyarlılık analizi ile tasarım değişkenlerinin etkilerinin belirlenmesi
4. Optimizasyon algoritması ile tasarım değişkenlerinin güncellenmesi
5. Yakınsama sağlanana kadar 2-4 adımlarının tekrarlanması

Boyut optimizasyonu, yapısal mühendislikte çelik yapıların kesit optimizasyonu, betonarme yapıların donatı optimizasyonu, köprü ve kule tasarımı gibi birçok uygulamada yaygın olarak kullanılmaktadır. Optimizasyon sonucunda, malzeme kullanımı azaltılırken yapısal performans gereksinimleri karşılanmakta, böylece daha ekonomik ve sürdürülebilir tasarımlar elde edilmektedir.

11.2 Şekil Optimizasyonunun Temelleri

Yapı elemanlarının dış sınırlarının ve iç boşluklarının optimizasyonu veya mühendisin konuyu bakış biçimine bağlı olarak farklı şekillerde ele alınabilir.

- **Sınır Temsili:** Geometrik parametreler
- **Kontrol Noktaları:** Şekil değişimi kontrolü
- **Düzgünlük:** Geometrik süreklilik

[48]

[48] Şekil optimizasyonu, topolojiyi değiştirmeden yapının geometrisini iyileştirir.

11.3 Matematiksel Formülasyon

Boyut ve şekil optimizasyonu problemlerinin matematiksel ifadesi:

$$\begin{aligned} \text{minimize} \quad & f(\mathbf{x}, \mathbf{s}) \\ \text{subject to} \quad & g_i(\mathbf{x}, \mathbf{s}) \leq 0, \quad i = 1, \ldots, m \\ & h_j(\mathbf{x}, \mathbf{s}) = 0, \quad j = 1, \ldots, p \\ & \mathbf{x}^L \leq \mathbf{x} \leq \mathbf{x}^U \\ & \mathbf{s}^L \leq \mathbf{s} \leq \mathbf{s}^U \end{aligned} \tag{74}$$

Burada:

- $\mathbf{x}$: Boyut parametreleri
- $\mathbf{s}$: Şekil parametreleri
- f: Amaç fonksiyonu
- g_i, h_j: Kısıt fonksiyonları

11.4 Parametrik Modelleme

Tasarım değişkenlerinin matematiksel temsili:

- **CAD Parametreleri:** Geometrik boyutlar
- **Spline Eğrileri:** Sınır temsili
- **Morph Box:** Şekil deformasyonu

11.5 Duyarlılık Analizi

Bazı yapısal optimizasyon problemlerinde tasarım değişkenleri (parametreler) için duyarlılık analizi yapılabilir. Fakat yapısal optimizasyon problemleri genellikle hiperstatik yapıda olduğundan, duyarlılık analizleri beklenen sonuçları veremez ve hatta yanıltıcı olabilir. [49]

[49] Hiperstatiklik konusu, optimizasyon açısından önemlidir. Örneğin bir parametre için seçilen kesit, limit dayanıma yakın olmasına rağmen, yapının bir başka parametresindeki kesitin değişimi yük dağılımını tamamen etkileyerek, ilk kesitin gerrilmesini sınır dayanımın çok daha altına düşürebilir veya üzerine çıkarabilir.

11.6 Kısıt İşleme

Tasarım kısıtlarının ele alınması:

- **Gerilme Kısıtları:** Malzeme dayanımı
- **Deplasman Kısıtları:** Şekil değiştirme limitleri
- **Geometrik Kısıtlar:** Üretilebilirlik

11.7 Çok Amaçlı Optimizasyon

Daha sonra çok amaçlı optimizasyon konusundan çok daha ayrıntılı bir şekilde bahsedilecektir.

Çok Amaçlı Yaklaşımlar

- **Pareto Optimizasyonu:** Trade-off analizi
- **Ağırlıklı Toplam:** Tek amaç fonksiyonu
- **Hedef Programlama:** İdeal nokta yaklaşımı

11.8 Mesh Adaptasyonu

Sonlu eleman modellerinde sıklıkla kabusa dönüşebilen mesh oluşturma süreci de zaman zaman optimizasyonun konusu olabilmektedir. Fakat bu konu ayrı bir uzmanlık alanının ve dersin konusudur.

- **Mesh Kalitesi:** Eleman şekli kontrolü
- **Adaptif Mesh:** Otomatik ağ iyileştirme
- **Remeshing:** Yeniden ağ oluşturma

11.9 Üretilebilirlik ve Pratik Kısıtlar

Tasarımın pratik uygulanabilirliği:

- **Standart Kesitler:** Katalog seçimi
- **İmalat Yöntemi:** Üretim kısıtları
- **Maliyet:** Ekonomik faktörler

[50]

[50] Üretilebilirlik kısıtları, teorik optimum ile pratik çözüm arasında denge kurmayı gerektirir ve bu da bambaşka bir optimizasyon problemi olarak ele alınabilir.

11.10 Optimizasyon Sonuçlarının Değerlendirilmesi

Elde edilen sonuçların analizi ve yorumlanması:

Değerlendirme Kriterleri

- **Performans İyileştirmesi:** Başlangıç durumuna göre kazanımlar
- **Kısıt Sağlama:** Tüm tasarım kısıtlarının kontrolü
- **Üretilebilirlik:** Pratik uygulanabilirlik analizi
- **Maliyet Analizi:** Ekonomik değerlendirme

12 Çok Amaçlı Optimizasyon

Yapısal sistemlerin optimizasyonunda zaman zaman birbiriyle çelişen birden fazla amacın eş zamanlı olarak optimize edilmesi gerekebilir. Bu bölümde, çok amaçlı optimizasyon yöntemleri ve uygulamaları incelenecektir. [51]

[51] Çok amaçlı optimizasyonda, tek bir optimal çözüm yerine Pareto-optimal çözümler kümesi elde edilir. Çok amaçlı optimizasyonun daha iyi anlaşılması için bağlantıdaki örnek kod incelenebilir.

12.1 Çok Amaçlı Optimizasyonun Temelleri

Birden fazla amaç fonksiyonunun eş zamanlı optimizasyonu:

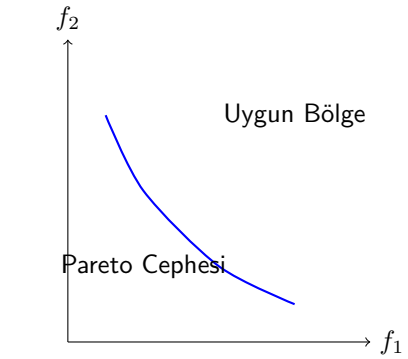
Temel Kavramlar

- **Pareto Optimallik:** Baskın çözümler
- **Trade-off:** Amaçlar arası ödünleşim
- **Karar Verme:** Çözüm seçimi
- **Ağırlıklandırma:** Amaçların önceliklendirilmesi

12.2 Matematiksel Formülasyon

Çok amaçlı optimizasyon probleminin genel yapısı:

$$\begin{aligned} \text{minimize} \quad & \mathbf{F}(\mathbf{x}) = [f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_k(\mathbf{x})]^T \\ \text{subject to} \quad & g_i(\mathbf{x}) \leq 0, \qquad i = 1, \ldots, m \\ & h_j(\mathbf{x}) = 0, \qquad j = 1, \ldots, p \\ & \mathbf{x}^L \leq \mathbf{x} \leq \mathbf{x}^U \end{aligned} \tag{75}$$



Şekil 27: Pareto cephesi örneği

12.3 Çözüm Yaklaşımları

Çok amaçlı optimizasyon problemlerinin çözümünde kullanılan temel stratejilerin incelenmesi, problemi ele alış biçimine göre farklı yaklaşımlar sunar. Yani esas olarak burada karar verici mühendisin kendisidir. Amaç fonksiyonlarının birbirlerine kıyasla üstün olup olmaması veya oransallığı bu kararda ve seçilecek stratejide etkilidir.

12.3.1 Skalerleştirme Yöntemleri

Çok amaçlı optimizasyon problemini tek amaçlı probleme dönüştüren yaklaşımlardır. Bu yöntemler, çoklu amaçları birleştirerek problemi daha kolay çözülebilir

hale getirir. Skalerleştirme yöntemleri, hızlı ve kolay uygulanabilir olmaları nedeniyle mühendislik problemlerinde yaygın olarak kullanılır. Ancak uygun ağırlıkların belirlenmesi, çözümün kalitesini doğrudan etkilediği için süreci zorlaştırabilir.

Yaygın Skalerleştirme Yöntemleri

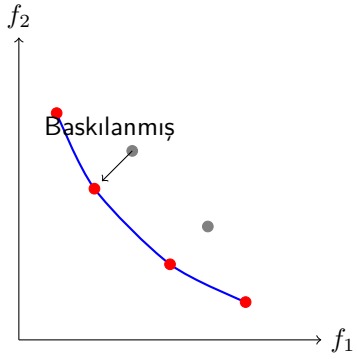
- **Ağırlıklı Toplam Yöntemi:** $F(x) = \sum_{i=1}^{k} w_i f_i(x)$
- **ε-Kısıt Yöntemi:** Bir amaç optimize edilirken diğerleri kısıt olarak tanımlanır
- **Hedef Programlama:** $\min \sum_{i=1}^{k} w_i |f_i(x) - T_i|$ (burada T_i hedef değerlerdir)

12.3.2 Pareto Tabanlı Yaklaşımlar

Pareto tabanlı yaklaşımlar, tüm Pareto-optimal çözümleri veya bunların iyi bir temsilini bulmayı hedefler. Bu yöntemler, çözüm uzayının daha geniş bir bölümünü keşfetmeyi sağlar ve karar vericiye daha fazla alternatif sunar. Pareto tabanlı yaklaşımlar, özellikle evrimsel algoritmaların kullanıldığı durumlarda daha etkilidir.

Pareto tabanlı yaklaşımların temel bileşenleri şunlardır:

- **Baskınlık İlişkisi:** Bir çözümün diğerine göre daha iyi olup olmadığının belirlenmesi
- **Çeşitlilik Mekanizmaları:** Pareto cephesi boyunca çözümlerin homojen dağılmasını sağlayan teknikler
- **Elit Stratejiler:** İyi çözümlerin korunmasını sağlayan mekanizmalar



Şekil 28: Pareto baskınlık kavramı

12.3.3 İnteraktif Yöntemler

İnteraktif yöntemler, karar vericiyi optimizasyon sürecine dahil ederek, tercihlerine göre çözüm uzayını daraltır. Bu yaklaşım, karar vericinin bilgi ve deneyimini algoritmanın çalışmasına entegre eder. İnteraktif yöntemler, özellikle karmaşık mühendislik problemlerinde, uzman bilgisinin çözüm sürecine dahil edilmesi açısından değerlidir.

İnteraktif yaklaşımların avantajları:

- Karar vericinin tercihlerini doğrudan sürece yansıtabilme
- Hesaplama kaynaklarını ilgi duyulan çözüm bölgesine yönlendirme
- Daha anlamlı ve uygulanabilir sonuçlar elde etme

12.4 Evrimsel Çok Amaçlı Optimizasyon Algoritmaları

Evrimsel algoritmaların çok amaçlı optimizasyon problemlerine uygulanması, klasik yöntemlere göre önemli avantajlar sağlar. Bu algoritmalar, tek seferde birden fazla çözüm üretebilme, karmaşık amaç fonksiyonlarını ele alabilme ve geniş çözüm uzaylarını etkili bir şekilde tarayabilme özellikleriyle öne çıkar.

12.4.1 NSGA-II (Non-dominated Sorting Genetic Algorithm)

NSGA-II, çok amaçlı evrimsel optimizasyon alanında en yaygın kullanılan algoritmalardan biridir. Baskınlık sıralama ve yoğunluk mesafesi hesaplama mekanizmalarıyla, hem Pareto-optimal çözümlere yakınsama hem de çözümler arasında

çeşitlilik sağlar. NSGA-II, $O(MN^2)$ hesaplama karmaşıklığıyla oldukça verimli çalışır ve birçok mühendislik probleminde başarıyla uygulanmıştır.

NSGA-II'nin temel bileşenleri:

- **Hızlı Baskınlık Sıralama:** Popülasyondaki çözümleri baskınlık ilişkisine göre sıralar
- **Yoğunluk Mesafesi:** Aynı baskınlık seviyesindeki çözümler arasında çeşitliliği korur
- **İkili Turnuva Seçimi:** Baskınlık seviyesi ve yoğunluk mesafesine dayalı seçim mekanizması

12.4.2 MOEA/D (Multiobjective Evolutionary Algorithm based on Decomposition)

MOEA/D, çok amaçlı optimizasyon problemini bir dizi tek amaçlı alt probleme ayrıştırarak çözen bir evrimsel algoritmadır. Her alt problem, komşu alt problemlerle bilgi paylaşımı yaparak eş zamanlı olarak optimize edilir. Bu yaklaşım, özellikle çok sayıda amaç fonksiyonu içeren problemlerde etkilidir ve hesaplama açısından verimlidir.

MOEA/D'nin avantajları:

- Komşuluk yapısı sayesinde etkili bilgi paylaşımı
- Çok sayıda amaç fonksiyonu içeren problemlere uygunluk
- Farklı ayrıştırma yöntemlerinin kullanılabilmesi (Tchebycheff, ağırlıklı toplam vb.)

12.4.3 SPEA2 (Strength Pareto Evolutionary Algorithm)

SPEA2, sabit büyüklükte bir arşiv kullanarak Pareto-optimal çözümleri saklayan ve rafine eden bir evrimsel algoritmadır. Algoritma, her çözüme bir uygunluk değeri atayarak hem baskınlık ilişkisini hem de çözüm yoğunluğunu dikkate alır. SPEA2, özellikle çeşitlilik ve yakınsama arasında iyi bir denge kurması nedeniyle tercih edilir.

SPEA2'nin önemli özellikleri:

- **Güç Değeri:** Bir çözümün kaç çözümü baskıladığını ölçer
- **Yoğunluk Tahmini:** K-en yakın komşu yöntemiyle hesaplanır
- **Harici Arşiv:** Pareto-optimal çözümleri etkin bir şekilde depolar ve günceller

12.5 Karar Verme Süreçleri

Çok amaçlı optimizasyon, bir dizi Pareto-optimal çözüm üretir ve bu çözümler arasından seçim yapılması gerekir. Karar verme süreci, optimizasyon sürecinin kritik bir parçasıdır ve çeşitli yaklaşımlarla desteklenebilir.

12.5.1 Çözüm Seçimi Kriterleri

Pareto-optimal çözümler arasından seçim yaparken kullanılabilecek çeşitli kriterler vardır:

- **Uzaklık Ölçüleri:** İdeal noktaya en yakın çözüm (örn. Öklid mesafesi, Tchebycheff mesafesi)

- **Tatmin Düzeyi:** Her amaç için belirlenen eşik değerlerini sağlayan çözümler
- **Göreceli İyileştirme:** Bir amacın diğerine göre iyileşme oranı (trade-off analizi)
- **Risk Analizi:** Belirsizlik altında çözümlerin güvenilirliği

Örnek: Ağırlıklı Tchebycheff Metriği

$$d(F, F^*) = \max_{i=1,\ldots,k} \{w_i \cdot |F_i - F_i^*|\} \tag{76}$$

Burada F^* ideal nokta, w_i amaçların ağırlıkları, F_i mevcut çözümün i. amaç değeridir.

12.5.2 Ağırlıklandırma Stratejileri

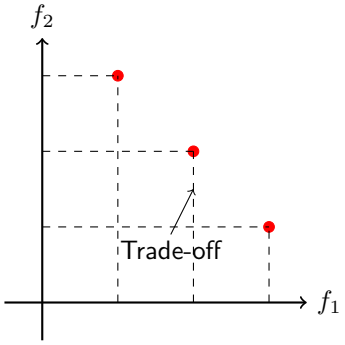
Amaçların göreceli önemini belirlemek için çeşitli ağırlıklandırma stratejileri kullanılabilir:

- **Doğrudan Atama:** Karar vericinin doğrudan ağırlık ataması
- **AHP (Analitik Hiyerarşi Süreci):** İkili karşılaştırmalar yoluyla ağırlık belirleme
- **Entropi Tabanlı Yöntemler:** Veri dağılımına göre ağırlık hesaplama
- **TOPSIS:** İdeal çözüme benzerlik ile ağırlıklandırma

12.5.3 Çözümler Arası Kıyaslama

Farklı Pareto-optimal çözümlerin karşılaştırılması, karar vericinin tercihine uygun çözümü seçmesine yardımcı olur:

- **Görselleştirme Teknikleri:** Paralel koordinat grafikleri, yıldız diyagramları, ısı haritaları
- **Duyarlılık Analizi:** Parametrelerdeki değişimlerin çözüme etkisi
- **Robust Değerlendirme:** Belirsizlik altında çözümlerin performansı
- **Yaşam Döngüsü Analizi:** Uzun vadeli performans ve maliyet değerlendirmesi



Şekil 29: Pareto çözümleri arasındaki trade-off analizi

12.6 Çok Amaçlı Optimizasyonda Performans Metrikleri

12.6.1 Hypervolume (Hiperhaciim)

Hiperhacim, çok amaçlı optimizasyon algoritmalarının performansını değerlendirmek için kullanılan en yaygın metriklerden biridir. Bu metrik, Pareto cephesinin referans noktasına göre kapladığı alanın veya hacmin ölçüsünü ifade eder. Daha yüksek hiperhacim değeri, algoritmanın daha iyi bir Pareto cephesi bulduğunu gösterir, çünkü bu durum çözüm uzayının daha geniş bir bölümünün kapsandığını ifade eder.

12.6.2 IGD (Ters Nesil Mesafesi)

Ters Nesil Mesafesi (IGD), gerçek Pareto cephesi ile algoritma tarafından bulunan çözüm kümesi arasındaki mesafenin bir ölçüsüdür. Bu metrik, bulunan çözümlerin gerçek Pareto cephesine ne kadar yakın olduğunu ve cepheyi ne kadar iyi temsil ettiğini gösterir. Daha düşük IGD değeri, algoritmanın gerçek Pareto cephesine daha yakın ve daha iyi dağılmış çözümler bulduğunu ifade eder.

12.6.3 Yayılım (Çeşitlilik)

Yayılım metriği, çözüm kümesindeki noktaların birbirlerine olan uzaklığının ölçüsüdür ve Pareto cephesi boyunca çözümlerin ne kadar homojen dağıldığını gösterir. Bu metrik, algoritmanın çözüm uzayını ne kadar iyi keşfettiğini ve çeşitli alternatifler sunabildiğini değerlendirir. Düzgün dağılmış bir Pareto cephesi için daha düşük yayılım değerleri tercih edilir.

12.6.4 Hesaplama Süresi

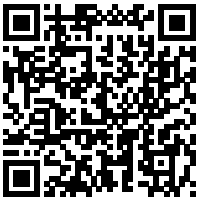
Hesaplama süresi, bir algoritmanın çözüme ulaşmak için harcadığı zamanı ölçer. Bu metrik, algoritmanın verimliliğini ve pratik uygulamalardaki kullanılabilirliğini değerlendirmek için önemlidir. Özellikle karmaşık mühendislik problemlerinde, makul bir sürede iyi sonuçlar verebilen algoritmalar tercih edilir. Hesaplama süresi, algoritmanın karmaşıklığına, problem boyutuna ve uygulama ortamına bağlı olarak değişebilir. [52]

[52] Bu parametre oldukça bağıl olması sebebiyle, her zaman güvenilir sonuçlar vermez. Günümüzde CPU Time gibi daha modern metrikler kullanılır.

13 Uygulama I

Bu bölümde beş farklı kısımdan oluşan bir konsol kirişin boyut optimizasyonu gösterilecektir.
[53]

[53]



13.1 Problem Tanımı

Bu uygulamada, 5 eşit parçadan oluşan 3 metre uzunluğundaki bir konsol kirişin ağırlığını minimize etmeyi amaçlayan bir optimizasyon problemi ele alınacaktır. Problemin detayları şu şekildedir:

- 3 metre uzunluğunda, 5 eşit parçaya bölünmüş bir konsol kiriş
- Her bir parçanın içi boş dairesel kesiti vardır ve iki tasarım parametresi bulunur:
  - $r_{dß§}$: Dış yarıçap
  - $r_{iç}$: İç yarıçap
- Malzeme: S270 çeliği (E = 210 GPa, σ_y = 270 MPa)
- Kirişin serbest ucuna dik olarak F = 500 kN'luk bir kuvvet uygulanmaktadır
- Amaç: Kirişin ağırlığını minimize etmek

13.1.1 Kısıtlamalar

1. Kirişin serbest ucunda maksimum 2 cm'lik bir deplasman izin verilmektedir
2. Her bir parça için dış yarıçap, iç yarıçaptan büyük olmalıdır
3. Bir önceki parçanın iç yarıçapı, bir sonraki parçanın dış yarıçapından küçük olmalıdır (kaynak yapılabilirlik koşulu)
4. S270 çeliğinin akma gerilmesi (270 MPa) aşılmamalıdır

13.2 Yapısal Analiz

Konsol kirişin deplasman ve gerilme analizi için sonlu elemanlar yöntemi kullanılmıştır:

- Her kiriş parçası bir Euler-Bernoulli kiriş elemanı olarak modellenmiştir
- Her düğüm noktasında 2 serbestlik derecesi vardır (deplasman ve dönme)
- Deplasmanları hesaplamak için global rijitlik matrisi oluşturulmuştur
- Gerilmeler, eğilme momenti ve kesit özellikleri kullanılarak hesaplanmıştır

13.2.1 Rijitlik Matrisi Oluşturma

Kiriş elemanı için rijitlik matrisi şu şekilde oluşturulur:

$$k_e = \begin{bmatrix} \frac{12EI}{l^3} & \frac{6EI}{l^2} & -\frac{12EI}{l^3} & \frac{6EI}{l^2} \\ \frac{6EI}{l^2} & \frac{4EI}{l} & -\frac{6EI}{l^2} & \frac{2EI}{l} \\ -\frac{12EI}{l^3} & -\frac{6EI}{l^2} & \frac{12EI}{l^3} & -\frac{6EI}{l^2} \\ \frac{6EI}{l^2} & \frac{2EI}{l} & -\frac{6EI}{l^2} & \frac{4EI}{l} \end{bmatrix} \tag{77}$$

Burada:

- E: Young modülü
- I: Atalet momenti
- l: Eleman uzunluğu

İçi boş dairesel kesit için atalet momenti:

$$I = \frac{\pi}{4}(r_{dış}^4 - r_{iç}^4) \tag{78}$$

13.2.2 Sınır Koşulları ve Çözüm

Konsol kirişin sol ucu sabitlenmiştir, bu nedenle ilk düğüm noktasındaki serbestlik dereceleri sıfırdır. Sağ uca uygulanan kuvvet, global kuvvet vektörüne eklenir. Deplasman vektörü, indirgenmiş rijitlik matrisi ve kuvvet vektörü kullanılarak çözülür:

$$\mathbf{K}_{\mathbf{indirgenmiş}} \cdot \mathbf{u} = \mathbf{F} \tag{79}$$

13.3 Optimizasyon Yaklaşımı

Optimizasyon için Tavlama Benzetimi (Simulated Annealing) algoritması kullanılmıştır:

- Yerel optimumlardan kaçınmak için rastgele arama stratejisi
- Çözüm uzayının etkili bir şekilde keşfi için adaptif adım boyutu
- Daha iyi çözümler bulmak için süreç sıcaklığının yavaş soğutulması
- Fiziksel olarak uygulanabilir çözümleri sağlamak için kısıtlamaların etkin kontrolü

13.3.1 Amaç Fonksiyonu

Amaç fonksiyonu, kirişin toplam ağırlığıdır:

$$W = \rho \sum_{i=1}^{5} A_i \cdot l_i \tag{80}$$

Burada:

- ρ: Malzeme yoğunluğu
- A_i: Her bir parçanın kesit alanı ($A_i = \pi(r_{dß§,i}^2 - r_{iç,i}^2)$)
- l_i: Her bir parçanın uzunluğu

13.3.2 Kısıtlama Fonksiyonları

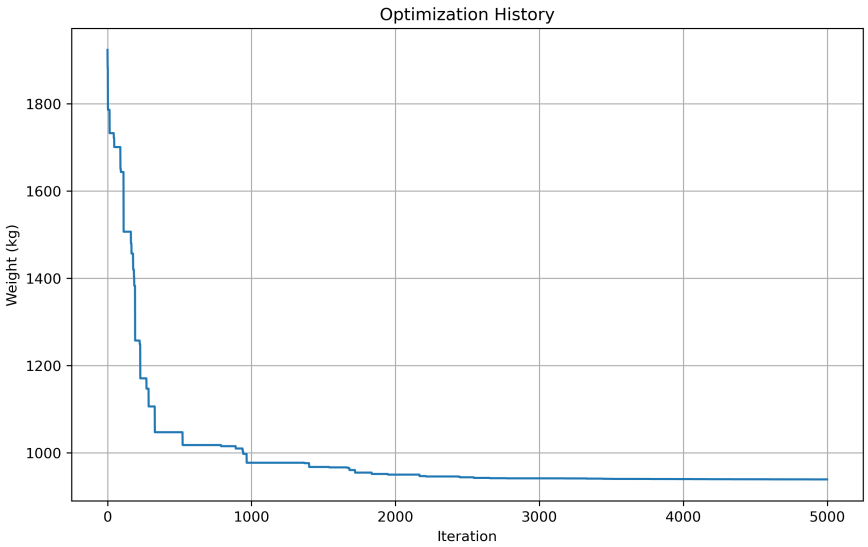
Optimizasyon sürecinde dört kısıtlama fonksiyonu kullanılmıştır:

1. Deplasman kısıtlaması: $u_{max} \leq 0.02$ m
2. Yarıçap kısıtlaması: $r_{dß§,i} > r_{iç,i}$ (her parça için)
3. Kaynak yapılabilirlik kısıtlaması: $r_{iç,i} < r_{dß§,i+1}$ (bitişik parçalar için)
4. Gerilme kısıtlaması: $\sigma_{max} \leq \sigma_{yield}$

13.4 Optimizasyon Sonuçları

Optimizasyon, başlangıç tasarımına kıyasla daha hafif bir kiriş tasarımı ile sonuçlanmıştır:

- Başlangıç tasarımının ağırlığı: ∼1924 kg
- Optimize edilmiş tasarımın ağırlığı: ∼939 kg (%51 azalma)



Şekil 30: Ağırlık İzleme Grafiği

13.4.1 Optimize Edilmiş Yarıçaplar (cm)

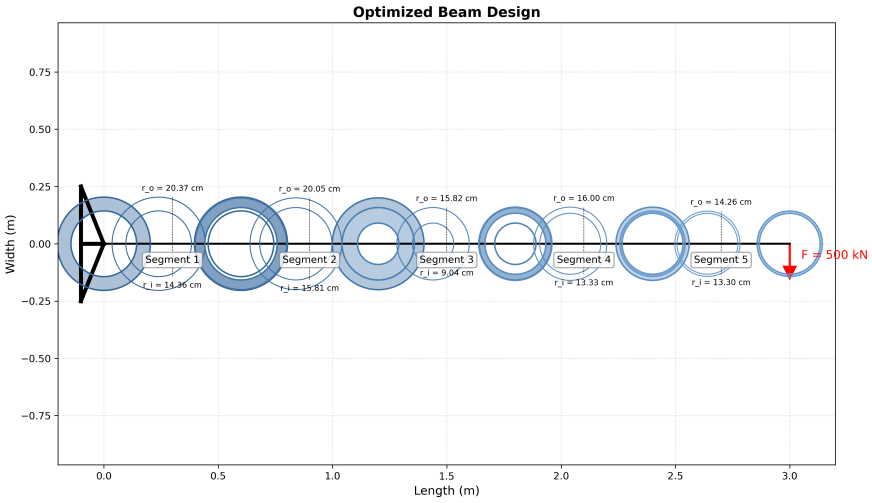
Parça	Dış Yarıçap ($r_{dß§}$)	İç Yarıçap ($r_{iç}$)
1	20.37	14.36
2	20.05	15.81
3	15.82	9.04
4	16.00	13.33
5	14.26	13.30

Optimize edilmiş tasarımda:

- Uç noktadaki deplasman 0.12 cm'dir (izin verilen maksimum 2 cm)
- Gerilme kısıtlaması aktiftir (0.00 MPa marj)
- Kesit boyutları kaynak yapılabilirlik koşulunu sağlamaktadır

13.4.2 Optimize Edilmiş Kiriş Tasarımı

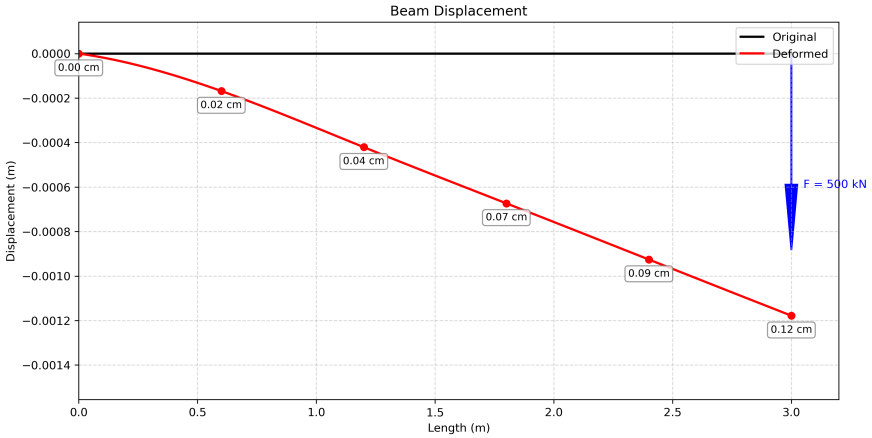
Optimize edilmiş kirişin geometrisi, destek noktasından (sol taraf) serbest uca doğru kesit boyutlarının azaldığını göstermektedir. Bu, eğilme momentinin destekte maksimum olması ve serbest uca doğru azalması ile uyumludur.



Şekil 31: Optimize Edilmiş Kiriş Tasarımı

13.4.3 Deformasyon Şekli

Yük altındaki konsol kirişin deformasyon şekli, serbest uçta maksimum 2 cm'lik bir deplasmana sahiptir. Her düğüm noktasındaki deplasman değerleri de gösterilmiştir.

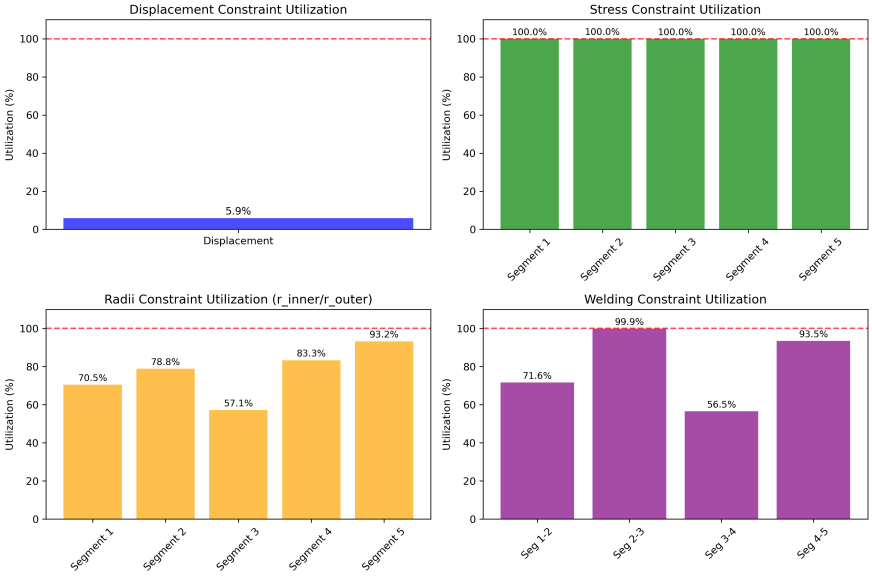


Şekil 32: Deformasyon Şekli

13.4.4 Kısıtlama Kullanım Oranları

Optimize edilmiş tasarımda her bir kısıtlamanın ne kadar kullanıldığını gösteren grafikler oluşturulmuştur:

- **Deplasman Kısıtlaması**: Maksimum deplasman sınırı tamamen kullanılmıştır (%100)
- **Gerilme Kısıtlaması**: Her segment için akma gerilmesinin kullanım oranı
- **Yarıçap Oranı**: İç yarıçapın dış yarıçapa oranı
- **Kaynak Yapılabilirlik**: Bitişik segmentler arasındaki kaynak koşulunun kullanım oranı



Şekil 33: Sınırlayıcı Kapasite Kullanım Oranları

Grafiklerden görüldüğü üzere, optimal tasarımda deplasman kısıtlaması aktiftir (tamamen kullanılmıştır). Bu, optimize edilmiş tasarımın ağırlık minimizasyonu açısından limite ulaştığını göstermektedir.

13.5 Sonuç ve Değerlendirme

Bu uygulamada, Benzetimli Tavlama optimizasyon algoritması kullanılarak, kısıtlamalar altında minimum ağırlık için bir konsol kirişin optimal tasarımı elde edilmiştir. Sonuçlar, başlangıç tasarımından %51 daha hafif bir yapının elde edildiğini göstermektedir.

Optimize edilmiş tasarımda, özellikle gerilme kısıtlamasının tamamen kullanıldığı (aktif olduğu) gözlemlenmektedir. Bu, ağırlık minimizasyonu problemlerinde teorik olarak beklenen bir sonuçtur, çünkü tipik olarak en az bir kısıtlamanın aktif olması beklenir.

Destek noktasından serbest uca doğru kiriş geometrisinin kademeli olarak azalması da yapısal açıdan beklenen bir sonuçtur. Eğilme momenti destek noktasında maksimum olduğundan, bu bölgede daha büyük kesitler oluşmuştur.

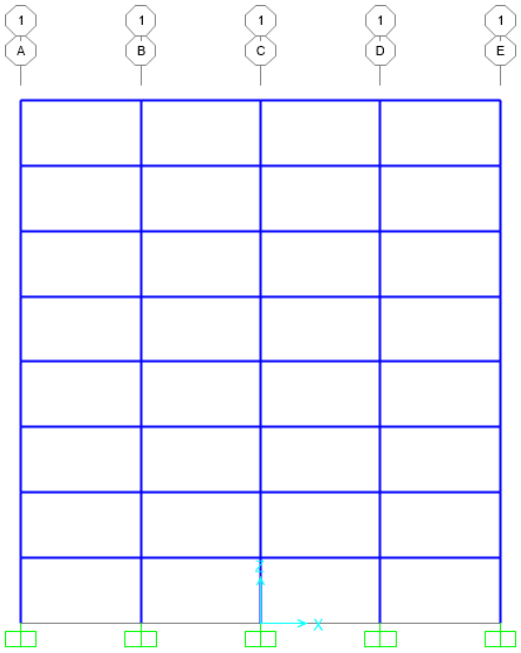
14 Uygulama II

Bu bölümde SAP2000 OAPI kullanılarak 4 açıklıklı, 8 katlı basit bir çelik çerçevenin kesit optimizasyonu gerçekleştirilecektir. Optimizasyon işlemi için tavlama benzetimi algoritması kullanılacak ve LRFD yöntemine göre tasarım yapılacaktır.
54

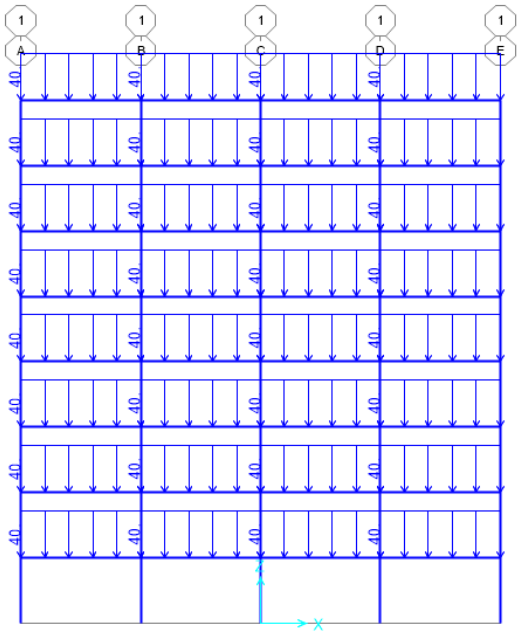
54

14.1 Çelik çerçeve modeli

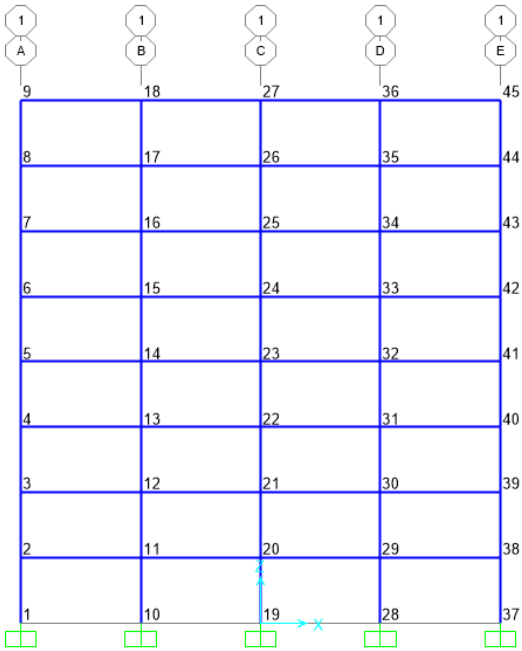
Model, 4 açıklıklı (5m açıklık mesafesi) ve 8 katlı (3m kat yüksekliği) bir çelik çerçeve yapıdan oluşmaktadır. Her katta açıklık üzerinde 40 kN/m düzgün yayılı yük bulunmaktadır. Tüm mesnetler ankastre olarak modellenmiştir. Yapıda Grade 36 çeliği kullanılmaktadır. Kirişlerin serbest burkulma boyu, kiriş uzunluğunun 1/5'i olarak kabul edilmiştir. SAP2000 Steel Design aracı kullanılarak tasarım ve optimizasyon işlemleri gerçekleştirilecektir.



Şekil 34: Çelik çerçeve modeli



Şekil 35: Yükleme Durumu



Şekil 36: Düğüm noktalarının numaralandırılması

14.2 Kesit gruplandırması

Yapıda kolon ve kirişler iki katta bir değişecek şekilde gruplandırılmıştır. Bu durumda toplam 4 kolon grubu ve 4 kiriş grubu bulunmaktadır. Her grup için farklı kesit seçimleri yapılacak ve optimizasyon sürecinde bu grupların kesitleri optimize edilecektir.

14.2.1 Seçilebilir parametrelerin belirlenmesi

Optimizasyon sürecinde kullanılacak kesit listesi, AISC kataloğundan seçilen W profillerinden oluşmaktadır. Kesit listesi W12'den W33'ye kadar farklı boyutlardaki profilleri içermektedir. Her grup için bu listeden uygun kesit seçimi yapılacak ve optimizasyon algoritması tarafından en uygun kesitler belirlenecektir.

14.3 Optimizasyon

14.3.1 Amaç Fonksiyonu

Optimizasyon sürecinde amaç fonksiyonu, yapının toplam ağırlığını minimize etmek olarak belirlenmiştir. Bu sayede hem ekonomik bir çözüm elde edilecek hem de yapının performansı optimize edilecektir. Yapı ağırlığının hesaplamak için model içinde "weight" adlı bir "load case" oluşturulmuştur. Bu "load case" başka bir kuvvet barındırmadığından z eksenindeki mesnet reaksiyonları toplamı yapı ağırlığını verir. Bununla birlikte birçok yapısal optimizasyon probleminde olduğu gibi bu problemde de amaç fonksiyonu aşağıdaki gibi ifade edilebilir:

$$f(x) = \sum_{i=1}^{n} \rho_i \cdot A_i \cdot L_i \tag{81}$$

Burada ρ_i kesit ağırlığını, A_i kesit alanını ve L_i kesit uzunluğunu ifade etmektedir. [55]

[55] W kesitlerin kullanıldığı daha karmaşık problemlerde kesit isimlendirmesinden faydalanılabilir. Örneğin W12×35 kesitinin $x35$ kısmı birim uzunluk başına ağırlığı ifade eder. Ancak bu değerlerin SI birimlerine çevrilmesi gerekmektedir.

14.3.2 Sınırlayıcılar

Optimizasyon sürecinde aşağıdaki sınırlayıcılar SAP2000 Çeelik Tasarım aracı kullanılarak gerçekleştirilir. VerifyPassed() metodu, yönetmelik şartlarını sağlamayan çerçeve elemanı sayısını döndürür. Arka planda ise burkulma boyundan genel bileşik denklemlere kadar tüm kontroller gerçekleştirilir. Ancak birçok optimizasyon çalışmasında -özellikle hız amacıyla OAPI ve benzeri araçların kullanılmadığı- aşağıdaki bileşik etki denklemleri kullanılarak kontroller sağlanır:

$$\frac{P_u}{\phi_c P_n} \geq 0.2; \quad c_1 = \frac{P_u}{\phi_c P_n} + \frac{8}{9}\left(\frac{M_{ux}}{\phi_b M_{nx}} + \frac{M_{uy}}{\phi_b M_{ny}}\right) \tag{82}$$

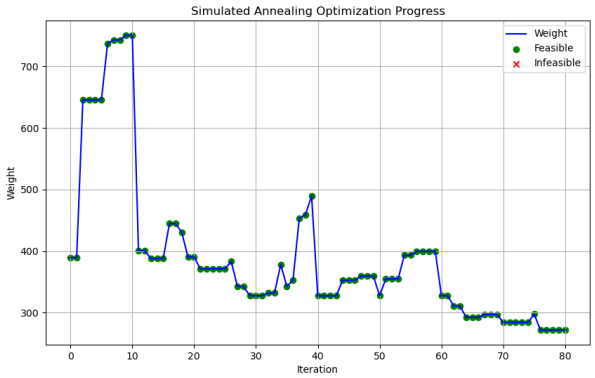
$$\frac{P_u}{\phi_c P_n} < 0.2; \quad c_2 = \frac{P_u}{\phi_c P_n} + \left(\frac{M_{ux}}{\phi_b M_{nx}} + \frac{M_{uy}}{\phi_b M_{ny}}\right) \tag{83}$$

$$\forall i : c_i \leq 1.0 \tag{84}$$

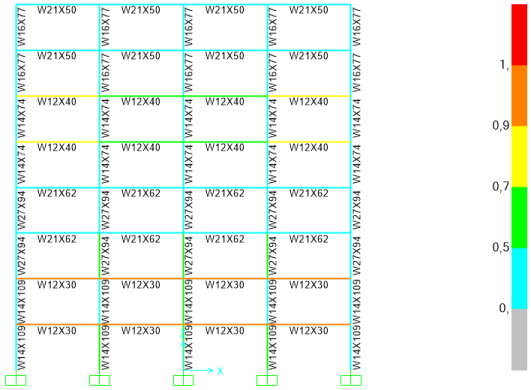
14.4 Optimizasyon Sonuçları

Tavlama benzetimi algoritması daha önce bahsedildiği gibi bir stokastik yöntemdir. Bu nedenle her çalıştırmada farklı sonuçların elde edilmesi doğaldır. Fakat araştırma amacıyla yapılan bir çalışmada algoritmanın tutarlığını sağlamak için genellikle çok sayıda çalıştırma yapılarak algoritmanın genel başarımı daha iyi anlaşılabilir. Bu örnekte yalnızca bir çalıştırma yapılacak ve sonuçları incelenecektir. Ancak yeterli analiz süresi tanındığında, örnek kapsamındaki kod da kolayca bu bağlamda değiştirilebilir.

Tablo 1: Kiriş ve kolon kesitleri



Şekil 37: Yapı ağırlığının optimizasyon süresince değişimi



Şekil 38: Optimum tasarımın Talep/Kapasite Oranı gösterimi