



T.C.

SELÇUK ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

BİR AĞIR VASITANIN ÇOK AKSLI DİREKSİYON MEKANİZMASININ OPTİMİZASYONU

Abdullah ERDEMİR

YÜKSEK LİSANS TEZİ

Makine Mühendisliği Anabilim Dalı

Haziran-2015
KONYA
Her Hakkı Saklıdır

TEZ BİLDİRİMİ

Bu tezdeki bütün bilgilerin etik davranış ve akademik kurallar çerçevesinde elde edildiğini ve tez yazım kurallarına uygun olarak hazırlanan bu çalışmada bana ait olmayan her türlü ifade ve bilginin kaynağına eksiksiz atıf yapıldığını bildiririm.

DECLARATION PAGE

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Abdullah ERDEMİR

Tarih: 30 / 06 / 2015

ÖZET

YÜKSEK LİSANS TEZİ

BİR AĞIR VASITANIN ÇOK AKSLI DİREKSİYON MEKANİZMASININ OPTİMİZASYONU

Abdullah ERDEMİR

Selçuk Üniversitesi Fen Bilimleri Enstitüsü
Makine Mühendisliği Anabilim Dalı

Danışman: Doç. Dr. Mete KALYONCU

2015, 68 Sayfa

Jüri
Prof. Dr. Fatih Mehmet BOTSALI
Doç. Dr. Mete KALYONCU
Yrd. Doç. Dr. Ümit ÖNEN

Bu çalışmada, bir ağır vasıtanın çok akslı direksiyon mekanizmasının optimizasyonu yapılmıştır. Optimizasyon çalışmalarında Ackerman dümenleme hatasından türetilen amaç fonksiyonunu minimize etmek için Arı Algoritması ve Hooke-Jeeves Algoritması eşzamanlı olarak kullanılmıştır. Ackerman hatası, çok akslı direksiyon mekanizmanın matematiksel modeli yerine CAD programı tarafından sağlanmış katı modeli kullanılarak hesaplanmıştır. Direksiyon mekanizmasının matematiksel modelinin elde edilmesindeki zorluğa karşılık katı model kullanarak dümenleme açılarının hesaplanması daha kolay ve basittir. Ackerman hatası hesaplamalarında Solidworks CAD programı ortamındaki katı model kullanılmıştır. Katı model kullanılarak, optimizasyon işlemi sürecinde her bir iterasyonda çok akslı direksiyon mekanizmasının güncel konfigürasyonu kullanıcı tarafından görülebilmektedir. Ackerman hatasının optimizasyonunda Arı Algoritması ve Hooke-Jeeves algoritmasını eşzamanlı çalıştırabilmek için VisualBasic.NET dilinde bir yazılım geliştirilmiştir. Solidworks ile geliştirilen yazılım arasındaki bilgi aktarımı için Solidworks API (Uygulama Programlama Arayüzü) kullanılmıştır. Geliştirilen teknik beş akslı bir ağır vasıtanın direksiyon mekanizmasının optimizasyonu için kullanılmıştır. Dümenleme açısının optimizasyonunda bu tekniğin kullanımından elde edilen nümerik sonuçlar tartışılmış ve yorumlanmıştır. Ayrıca Hooke-Jeeves algoritmasının optimizasyon işlemine katkısı da tartışılmıştır.

Anahtar Kelimeler: Arı Algoritması, çok akslı direksiyon mekanizması, çok değişkenli optimizasyon, Hooke-Jeeves Algoritması, Solidworks API.

ABSTRACT

MS THESIS

A MULTI-AXLE STEERED HEAVY VEHICLE STEERING MECHANISM OPTIMIZATION

Abdullah ERDEMİR

THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCE OF SELÇUK UNIVERSITY
THE DEGREE OF MASTER OF SCIENCE
IN MECHANICAL ENGINEERING

Advisor: Assoc. Prof. Dr. Mete KALYONCU

2015, 68 Pages

Jury
Prof. Dr. Fatih Mehmet BOTSALI
Assoc. Prof. Dr. Mete KALYONCU
Assist. Prof. Dr. Ümit ÖNEN

In this study, multi axle steering mechanism of a heavy vehicle is optimized. The Bees Algorithm and the Hooke-Jeeves algorithm are used simultaneously in order to minimize the objective function which is derived from Ackerman steering error in the optimization processes. The Ackerman error is calculated using a solid model obtained by a CAD program instead of using a mathematical model of the multi axle steering mechanism. Calculation of steering angles is easier and simpler by using the solid model compared with using the mathematical model of the steering mechanism. The solid model in Solidworks CAD program environment is used in calculation of the Ackerman error. By using the solid model, the current configuration of the multi-axle steering mechanism can be seen by the user in each iteration of the optimization process. A software in VisualBasic.Net language is developed in order to implement The Bees Algorithm and the Hooke-Jeeves algorithm simultaneously in optimization of the Ackerman error. Solidworks API (Application Programming Interface) is used in data transfer between the Solidworks and the developed software. Developed technique is used in optimization of the steering mechanism of a five axle heavy vehicle. In steering angle optimization, obtained numerical results which are gained from this technique are discussed and comments are presented. Contribution of Hooke-Jeeves algorithm to the effectiveness of optimization process is also discussed.

Keywords: the Bees Algorithm, Hooke-Jeeves algorithm, mechanism synthesis, multi axle steering, multivariable optimization, steering optimization, Solidworks API.

ÖNSÖZ

Yüksek yük kapasiteli vasıtalar, yere olan basıncını azaltmak için çok akslara ihtiyaç duyarlar. Çok aksın kullanılması da Ackerman dümenleme hatasını azaltmada bir zorluk oluşturur. Ackerman hatasının fazla olması tekerlerin aşınmasına, vasıtanın istenen yörüngede hareket yapamamasına ve manevra esnasında titreşimlere sebep olur. Bu yüzden Ackerman dümenleme hatasının minimize etmek büyük önem arz etmektedir. Bunu minimize etmek için global optimizasyon algoritması olan Arı Algoritması ve yerel arama algoritması olan Hooke-Jeeves algoritması kullanılmıştır.

Tez çalışması süresince, engin azmi ve hoşgörüsü ile her zaman desteğini yakından hissettiğim, karşılaştığım güçlük ve sorunlarda tecrübesini esirgemeyen danışman hocam Sayın Doç. Dr. Mete KALYONCU 'ya, yapılan çalışmaların uygulanması fırsatını veren MPG Makina Prodüksiyon Grubu Makine İmalat San. ve Tic. A.Ş. Genel Müdürü Sayın Veysel ALVER 'e, hayatım boyunca maddi ve manevi olarak sürekli beni destekleyen, ilgilerini her an hissettiğim çok sevdiğim annem, babam, kardeşlerim, eşim ve oğluma içtenlik ve samimiyetle teşekkür ederim.

Abdullah ERDEMİR
KONYA-2015

İÇİNDEKİLER

ÖZET ..... **iv**

ABSTRACT ..... **v**

ÖNSÖZ ..... **vi**

İÇİNDEKİLER ..... **vii**

SİMGELER VE KISALTMALAR ..... **ix**

1. GİRİŞ **1**

1.1. Tezin Amacı 1
1.2. Tezin Önemi 1

2. KAYNAK ARAŞTIRMASI **3**

3. ÇOK AKSLI DİREKSİYON MEKANİZMASINA SAHİP BİR AĞIR VASITA **22**

4. OPTİMİZASYON ALGORİTMASI **28**

4.1. Arı Algoritması 28
4.2. Hooke-Jeeves Algoritması 32
4.3. Arı Algoritması ile Hooke-Jeeves Algoritmasının Kombinasyonu 33

5. BİR AĞIR VASITAYA AİT ÇOK AKSLI DİREKSİYON MEKANİZMASININ OPTİMİZASYON PARAMETRELERİ **35**

5.1. Problem Tanımı 35
5.2. Tasarım Parametreleri 35
5.3. Cezalandırma Fonksiyonları 38
5.4. Hata Fonksiyonu Tanımı ve Ağırlıklar 39

6. OPTİMİZASYON ALGORİTMASI İLE CAD PROGRAMINI KONTROL EDEN YAZILIM **41**

7. SAYISAL UYGULAMALAR **47**

7.1. Denklem 5.4 – Denklem 5.5'e Göre Hata Fonksiyonunun Optimizasyonu 47
7.2. Denklem 5.6 – Denklem 5.7'e Göre Hata Fonksiyonunun Optimizasyonu 49
7.3. 3, 4 ve 5. Aksın Ackerman Hatası Değişimi 52

8. SONUÇ VE ÖNERİLER **53**

9. KAYNAKLAR **56**

ÖZGEÇMİŞ.....60

EK-1.....61

SİMGELER VE KISALTMALAR

Simgeler

a_i	:	i. rot uzunluğu
e	:	Elit arı sayısı
e_A	:	Türetilmiş Ackerman Hatası
e_1	:	2. aks sol tekerinin Ackerman hatası
e_2	:	2. aks sağ tekerinin Ackerman hatası
F_i	:	i. rot boyunca iletilen kuvvet
g_i	:	i. cezalandırma fonksiyonu
k_i	:	i. cezalandırma ağırlık katsayısı
L_i	:	i. kuvvet kolu uzunluğu
m	:	En iyi arıların sayısı
m_k	:	Kısıt şartı sayısı
n	:	Kâşif arı sayısı
p	:	Uzaydaki takip edilmesi istenen nokta sayısı
P_i	:	Mekanizma üzerindeki i. mafsal noktası
r_i	:	i. link uzunluğu
x_k	:	k. x değişkeni
x_{id}	:	Yörünge noktasının x konumu
x_{ig}	:	Mekanizmanın ulaştığı x konumu
δ	:	Dümenleme açısı
θ_i	:	i. kuvvet kolu açısı
α_i	:	Birinci uzvun yörüngede olması istenilen açısı

Kısaltmalar

ABC	:	Yapay Arı Kolonisi
BA	:	Arı Algoritması
BE	:	Arı Sistemi
BCO	:	Arı Koloni Optimizasyonu
BAHJ	:	Arı Algoritması ile Hooke-Jeeves Algoritmasının Kombinasyonu
CAD	:	Bilgisayar Destekli Tasarım
CAE	:	Bilgisayar Destekli Mühendislik
DH	:	Denavit-Hartenberg
EA	:	Evrimsel Algoritmalar
GA	:	Genetik Algoritması
HJ	:	Hooke-Jeeves Algoritması
PSO	:	Parçacık Sürü Optimizasyonu

1. GİRİŞ

1.1. Tezin Amacı

Bu çalışmada; beş akslı bir ağır vasıtanın direksiyon mekanizmasının optimizasyonu yapılmıştır. Vasıtanın dönüşü esnasında Ackerman dümenleme hatası; tekerlerin aşınmasına, istenen yörüngenin takip edilememesine ve titreşimlerin oluşmasına sebep olmaktadır.

Bu tezin amacı; çok akslı vasıtalarda Ackerman dümenleme hatasını tüm dümenleme açılarında kabul edilebilir bir değerin altına indirmektir. Bu tezde, bu değer 1° olarak belirlenmiştir.

Ayrıca bir optimizasyon işleminde, hata fonksiyonunun etkin olabilmesi için ağırlık katsayılarının ve sınır şartlarının etkisini de araştırılmıştır.

Bu tezin önemli bir parçası da, bir global optimizasyon yöntemi olan Arı Algoritmasına (BA), bir yerel arama yöntemi olan Hooke-Jeeves algoritmasının (HJ) global optimizasyon yöntemine nasıl faydalı olabileceğinin araştırılmasıdır.

Çok uzuvlu mekanizmanın matematiksel modelinin elde edilmesindeki zorluktan kurtulup imalat firmalarında hali hazırda var olan CAD yazılımı ortamındaki katı modeli kullanarak kinematik analizlerin ve optimizasyon çalışmalarının yapılabilmesidir.

1.2. Tezin Önemi

Bu tezde; literatürdeki ismi ‘Steering Mechanism Optimization’ ve ‘Linkage Optimization’ olan, tekerlek doğrultularının dümenleme esnasında sürekli kesişmesinin istendiği bir mekanizmanın optimizasyonu konusu işlenmektedir. Direksiyon mekanizmalarının optimizasyonu çok akslı vasıtaların sağlıklı çalışabilmesi için gerekli ve zorunludur.

Yüksek yük taşıma kapasiteli vasıtalara olan ihtiyacın artması ile birlikte direksiyon mekanizmaları da geliştirilmiştir. Direksiyon mekanizmasının optimum şekilde tertip edilmesi vasıtaların manevra kabiliyetini artırmaktadır. Uygun olmayan direksiyon mekanizmaları ise tekerleklerin kısa sürede aşınmasına sebep olmaktadır. Bu durum ise vasıta üzerinde titreşimlere sebep olmaktadır. Bu aynı zamanda manevra yapılması istenilen yörüngenin dışına da çıkılmasına sebep olabilmektedir.

Vasıtaların bir virajda dönme hareketinden başka hareket kabiliyetlerine de sahip olması beklenebilmektedir. Örneğin yengeç yürüyüşü adı verilen bir metot ile vasıta az bir hareket ile çapraz yönde konumlanabilmektedir. Bir çekici tırın arkasındaki uzun bir treylerin dar sokaklardan dönebilmesi gerektiğinde çok farklı dümenleme taktiklerinin uygulanması gerekmektedir. Bu sebeplerden dolayı direksiyon mekanizmalarının iyileştirilmesi, mekanizma sentezi alanında da gözde bir konu olagelmiştir.

Çok akslı ağır vasıtaların dümenleme mekanizması iki boyutta ifade edilemeyecek bir geometriye sahiptir. Bu mekanizmalar üç boyutlu uzayda küresel mafsallardan ve birden fazla eksen etrafında değişen oryantasyonlara sahip gövdelerden oluşur. Bu durum ise direksiyon mekanizmasının matematiksel modelinin elde edilmesini büyük ölçüde zorlaştırmaktadır. Bu yüzden bilgisayar ortamında çizilmiş olan modellerin matematiksel modeli ile elde edilebilecek kinematik bilgilerinin, bizzat bilgisayardaki çizim programından alınması, işi büyük ölçüde kolaylaştırmaktadır. Ayrıca imalat firmaları imalatları gerçekleştirebilmek için hali hazırda CAD ortamında katı modellere sahiptirler.

Bunun yanı sıra uzayın farklı bölgelerinde sınır şartlarının değişmesi ile beraber matematiksel modellerde bu değişikliğe uyum sağlayacak şekilde oluşturulmalı ve kontrol edilmelidir. Ancak bir bilgisayar destekli tasarım (CAD) programı bu zorlu ve zahmetli işi ortadan kaldırmaktadır. Bu sebeple bir CAD programı, model ile ilgili kinematik bilgilerinin matematiksel model olarak kullanılması hem hızlı çözüm vermektedir hem de matematiksel modelin yanlış elde edilme riskini de ortadan kaldıran bir çözüm olmaktadır.

Çok akslı vasıtalar, vinçlerde kullanıldığı gibi savunma sanayinde de büyük askeri vasıtalarda kullanılmaktadır. Bu vasıtaların farklı hızlardaki manevra kabiliyetinin artırılması rakipler karşısında bir avantaj oluşturduğu için direksiyon mekanizmasının optimizasyonu bu yüzden de çok önemli bir yer tutmaktadır.

2. KAYNAK ARAŞTIRMASI

Mekanizmalar bir enerjiyi istenilen bir noktaya aktarmak için bir araya gelmiş mekanik elementler topluluğudur. 1970 yılından beri bilgisayar kaynaklarının gelişmesi ile beraber çok gövdeli karmaşık sistemlerin mekanizma sentezi problemleri çok gözde bir konu olmuştur. Mekanizmaların karmaşıklaşması ile beraber analitik çözüm elde edilmesi de zorlaşmıştır.

Çok akslı vasıtalar genellikle büyük boyutlu ve yüksek yük kapasiteli vasıtalardır. Yer ile vasıta arasındaki basıncı indirgeme gereksiniminden dolayı çok aksların kullanılmasına ihtiyaç duyulmuştur. Ackerman geometrisine göre teker eksenlerinin dönüş radyüs merkezinde kesişmesi gerekmektedir (Heißing ve Ersoy, 2010). Çok aksların kullanımı sonucunda da bu tip vasıtalarda, Ackerman hatası daha artmakta tekerlerin aşınmasına sebep olmaktadır. Bu, aynı zamanda vasıtanın istenmeyen yörüngede manevra yapmasına ve manevra esnasında da titreşimlere sebep olmaktadır. Çok değişkenli mekanizmalarda, istenilen yörüngenin takip edilebilmesi için, hata fonksiyonunu minimize eden değişken değerlerini tespit edebilmek önemlidir. Bunu yapabilmek için, istenilen tolerans aralığında hatayı minimize eden değişken dizisi bulununcaya kadar değişken dizileri üretilmeli ve bunun getirdiği hata hesaplanmalıdır. Bir optimizasyon işleminde p adet değişken var ise, bu aslında $p+1$ boyutlu bir uzaydaki grafiğin minimum değerini bulma işidir. Ve $p+1$ boyutlu uzaydaki minimum değişken konfigürasyonunu öngörebilmek neredeyse imkânsızdır. Bu yüzden çok değişkenli optimizasyon tekniklerinin kullanılması gerekmektedir.

Mekanizma sentezi üç konu başlığından oluşmaktadır: Bunlar grafik, analitik ve nümerik metotlarıdır (Hartenberg ve Denavit, 1964, Alhajj, 2011). Eğer mekanizmanın, az sayıdaki noktalardan geçmesi isteniyorsa grafik metodu hızlı çözüm bulmaktadır. Analitik metotta ise cebrik ifadelerle mekanizmanın matematiksel modeli çözülerek hassas noktalardan geçecek şekilde sonuca varılmaktadır, fakat bu hassas noktalar birbirleri ile uyumlu değilse veya sayıca fazla ise analitik metot sonucunda karmaşık sayılar elde edilebilmektedir (Vujić ve Radojković, 2000). Bu da sonucun fiziksel olarak mümkün olmadığı ve üretilemeyeceği anlamına gelmektedir. Nümerik teknikler genellikle çeşitli optimizasyon metotları ile beraber kullanılmaktadırlar. Sürü tabanlı optimizasyon algoritmaları bu alanda yaygın bir şekilde kullanılmaktadır. Sürü tabanlı optimizasyon algoritmaları şunlardır.

1. Evrimsel Algoritmalar
2. Parçacık Sürü Optimizasyonu
3. Karınca Koloni Optimizasyonu
4. Arılardan esinlenen algoritmalar (Arı Sistemi (BE), Arı Koloni Optimizasyonu (BCO), Yapay Arı Kolonisi (ABC) ve Arı Algoritması (BA))

En çok kullanılan optimizasyon teknikleri Evrimsel Algoritmalar (EA), Genetik Algoritması (GA) ve Parçacık Sürü Optimizasyonu (PSO)'dur (Yuce ve ark., 2013). Nüfus temelli optimizasyon teknikleri çoğunlukla global minimuma erişebilmektedir. Ayrıca nüfus temelli optimizasyonlarda bir başlangıç şartına da ihtiyaç yoktur, sadece değişken aralığına ihtiyaç vardır.

Boyutsal sentezde iki yaklaşım vardır: Bunlar hassas noktalar ve en uygun sentez. Hassas nokta sentezi mekanizmanın belli bir noktasının, belli sayıdaki noktalardan geçmesini hedeflemektedir (Hartenberg ve Denavit, 1964). Hedef noktaların sayısı arttıkça problem çok nonlineer bir hal almakta ve çözmesi çok zorlaşmaktadır. Analitik yöntemlerle mekanizmanın aşırı fazla noktadan geçmesi istendiğinde mekanizma boyutları karmaşık sayı şeklinde çıkabilmektedir, bu da hedeflenen noktalardan geçmesi istenilen mekanizmanın gerçekte mümkün olmadığını göstermektedir. Görüldüğü üzere hassas nokta sentezi belli bir sayıdaki noktaya kadar çözüm üretebilmektedir. Diğer bir taraftan, en uygun sentez ise rastgele üretilmiş mekanizma boyutlarını kullanarak en iyi sonucu veren değişken dizisini bulma işidir ve boyutsal sentezde çokça kullanılmaktadır.

Hata fonksiyonuna kısıt şartları dâhil edilebilmektedir. Örneğin; uzuv boyutunun negatif olmaması veya bir uzvun boyu, bir sayıdan büyük veya küçük olması durumu örnek verilebilir. Bunu başarabilmek için hata fonksiyonunun içine cezalandırma fonksiyonları eklenebilmektedir.

Optimizasyon birçok alternatif arasından en iyi çözümü bulma işlemidir. Optimizasyon, nümerik metotlara dayanır ve genellikle nonlineer problemler için kullanılmaktadırlar. Optimizasyonda yerel arama metotları ve global arama metotları vardır. Optimizasyon tekniklerinin çoğu kısıt şartlarının kullanımı üzerine kuruludurlar. Mekanizma sentezinin optimizasyonu çalışması yeni değildir. Bununla ilgili çok sayıda çalışmalar yapılmıştır. Daha gerçekçi problemler çok boyutlu olanlarıdır. Optimizasyon sırasında kısıt şartlarına uymanın tanımlanması da bir başka problem olmaktadır. Bilim adamları bunun için birçok algoritma geliştirmişlerdir. Örneğin; cezalandırma metotları, uyarlanabilir kısıtlar, birlikte evrim, çok amaçlı optimizasyon, kültür algoritmaları,

Pareto metodu ve genişletilmiş Lagrangian metodu gibi. Cezalandırma metodu ilk kısıt işleme metodudur ve halen de ilgi görmektedir (Zhang ve ark., 2011).

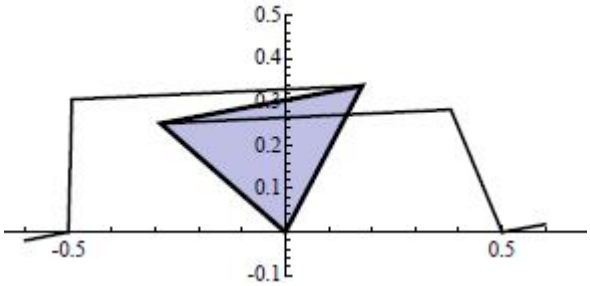
Kumar (2013), Parçacık Sürü Optimizasyonunu (PSO) ve çeşitlerini 5 uzuvlu bir mekanizmada karşılaştırmıştır. Bu çalışmada, 11 ila 25 arasındaki noktalardan oluşan yörüngenin sentezi çalışması yapılmıştır. Düzlemsel mekanizmalarda hata fonksiyonu Denklem 2.1'deki gibi tanımlanmıştır. Ortiz ve ark. (2013), dört çubuk mekanizması optimizasyonunda otomatik adaptif kontrol parametreleri ile diferansiyel gelişim algoritmasını kullanmışlar ve hata fonksiyonlarını yine Denklem 2.1'deki gibi tanımlamışlardır.

$$F_{hata} = \sum_{i=1}^{p} \left[(x_{id} - x_{ig})^2 + (y_{id} - y_{ig})^2 \right] \tag{2.1}$$

Bu denklemde x_{id}, yörüngesi takip edilmek istenen noktanın x konumu ve x_{ig} ise mekanizma üzerindeki işaretli noktanın gerçekte olduğu konum, bu aynı şekilde y için de geçerlidir. p ise yörünge üzerindeki nokta sayısıdır. Bu hata fonksiyonu istenen ile gerçekleşen nokta arasındaki mesafeyi vermektedir.

Penunuri ve ark. (2011), Diferansiyel Gelişim algoritması ile dört çubuk mekanizmasında yörünge sentezi yapmışlardır. Denklem 2.1'e ek olarak dört çubuk mekanizmasındaki birinci uzvun açısını da hata fonksiyonuna dâhil ederek Denklem 2.2 elde edilmiştir ve istenen açıda istenen koordinata konumlanmayı amaçlamışlardır. Bu tekniği düzlemsel bir Ackerman geometrisinin optimizasyonu için uyguladıklarında Şekil 2.1'deki gibi geometrisi elde edilmiştir. Bu geometriye göre dümenleme hatası 0.867° elde edilmiştir.

$$F_{hata} = \sum_{i=1}^{p} \left[(x_{id} - x_{ig})^2 + (y_{id} - y_{ig})^2 + \left(\alpha_{id} - \alpha_{ig}\right) \right] \tag{2.2}$$



Şekil 2.1 Diferansiyel Gelişim Algoritması ile optimize edilen bir Ackerman geometrisi

Dört çubuk mekanizması basit ve klasik bir mekanizma olup, dört ana çubuktan oluşur ve uç kısımları birbirlerine bağlıdır. Optimizasyon esnasında bir değişken dizisinin denenmeden önce mekanizmanın geçerli olup olmadığı test edilmelidir. Örneğin dört çubuk mekanizmasının geçerli olup olmadığı Grashof şartı ile kontrol edilir (Alhajj, 2011, Portilla-Flores ve ark. 2013). r_1, r_2, r_3 ve r_4 uzuv boyutları olmak üzere dört çubuk mekanizmasında aşağıdaki şart doğru ise mekanizma geçerlidir, bu şart doğru değil ise bu mekanizma geçersizdir ve yeni bir değişken dizisi üretmek gerektiğine karar verilmektedir.

$$r_4 < r_3 < r_2 < r_1 \text{ olmak üzere } r_1 + r_4 < r_2 + r_3 \text{ 'tür}$$

Cezalandırma metodu da hata fonksiyonuna Denklem 2.3'teki gibi eklenmektedir.

$$F_{hata} = \sum_{i=1}^{p} \left[(x_{id} - x_{ig})^2 + (y_{id} - y_{ig})^2 \right] + \sum_{j=1}^{m_k} k_j g_j \tag{2.3}$$

Burada g cezalandırma şartı, k ise bu cezalandırma şartı gerçekleşirse verilecek cezalandırma katsayısıdır. m_k ise kısıt şartlarının sayısıdır. Örnek verecek olursak;

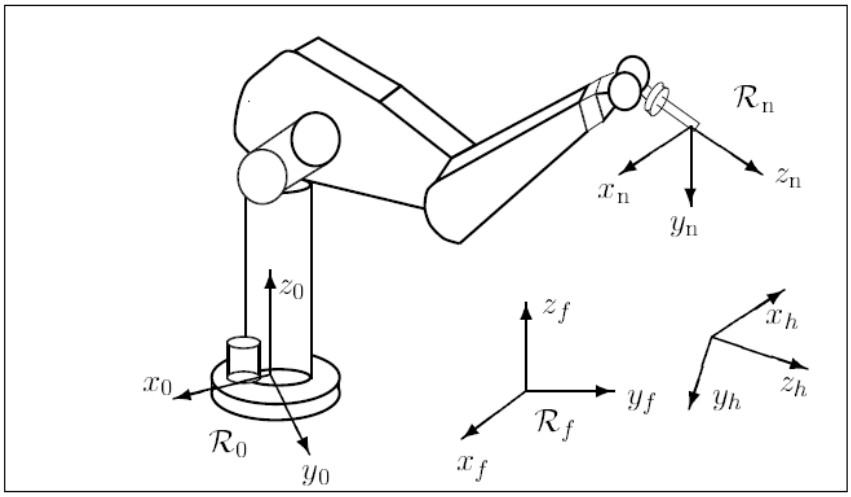
$$g_1 = r_4 < 100$$
$$k_1 = 46$$

r_4 uzvunun boyu 100'den küçük olunca g_1 'in değeri 1 olur, k_1 ve g_1 çarpımı 46 olmaktadır. Yani bulunan hataya 46 birim daha eklenmiş olmaktadır. Böylece r_4 uzvunun boyunun 100'ün altına inmesi durumu cezalandırmış olmaktadır. Eğer r_4,

100'ün altına inmemiş olsaydı, g_1 değeri 0 olur ve g_1 ile k_1 çarpılınca da 0 olurdu. Böylece ceza uygulanmamış olurdu.

Direksiyon mekanizmamızın modeli CAD programında olduğu için mekanizmanın geçerli olup olmadığını CAD programına kontrol ettirilecek ve cezalandırma gerekip gerekmediğine karar verilecektir.

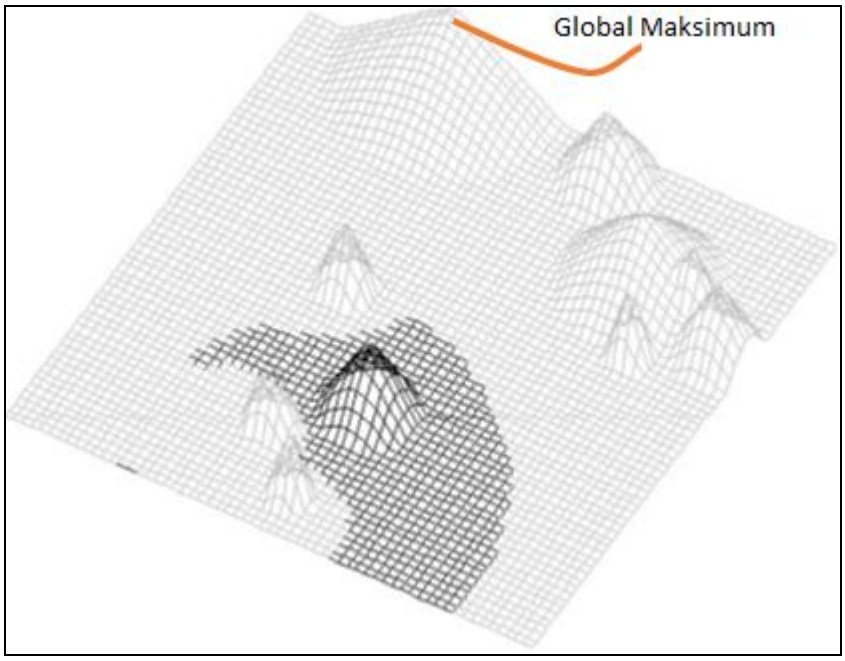
Ouezdou ve Regnier (1997), Denavit-Hartenberg parametreleri belirli robotik mekanizma sentezini, ters kinematik yaklaşım ile dağıtılmış çözüm metodu tabanlı bir algoritma ile çözmeye çalışmıştır. Denavit-Hartenberg (DH) parametreleri belli olduğundan sentez problemi bir ters kinematik problemine dönüşmektedir. DH parametrelerinin temel yapısı oluşturulduktan sonra DH tablosunda bütün değerlerini belirli aralıklar içinde kalmak koşuluyla değişken yaparak nümerik metotlarla çözmek mümkün olmaktadır. Şekil 2.2'de, DH parametre tablosu oluşturulmuş açık uçlu bir mekanizmanın optimizasyonu için yapılmış model görülmektedir.



Şekil 2.2. Denavit-Hartenberg parametre tablosu oluşturulmuş açık uçlu mekanizma sentezi

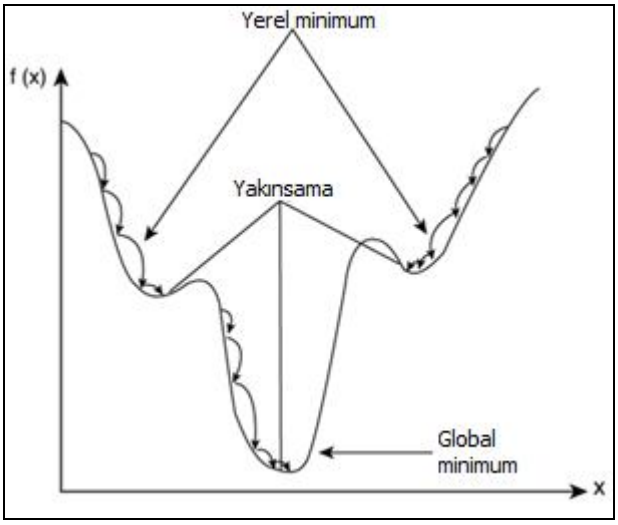
Bir fonksiyonun minimum veya maksimum değerini bulmak için geliştirilmiş birçok optimizasyon algoritması bulunmaktadır. Bu algoritmalardan bazıları başlangıç dizisinin komşuluklarında araştırma yapmaktadır. Örneğin Hooke ve Jeeves (1961) tarafından önerilen Hooke-Jeeves algoritması gibi yerel algoritmalar, yerel minimum veya maksimumu bulmada iyidirler, fakat global minimum veya global maksimumu bulamayabilmektedirler. Hooke-Jeeves doğrusal arama algoritması, bir başlangıç şartına ve komşuluk araştırması için adım boyutuna ihtiyaç duymaktadır. Her bir iterasyonda, her bir dizi değişkeninin adım komşuluğunda hatayı kontrol etmektedir. Hata azalması yön değiştirdikçe adım yarıya düşmekte ve yeni komşuluk hataları hesaplanmaktadır.

Adım büyüklüğü minimum adıma ulaşınca iterasyonlar son bulmakta ve yerel minimum elde edilmektedir.



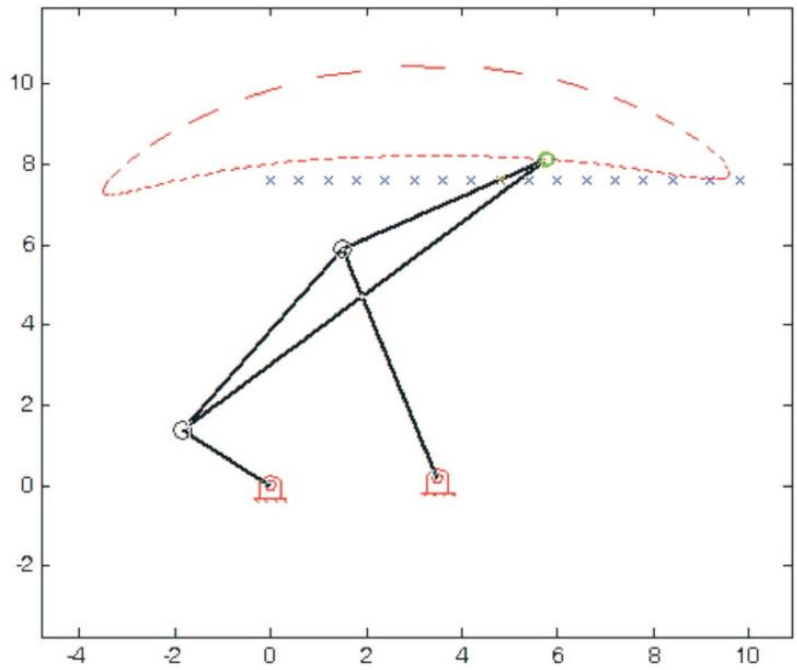
Şekil 2.3. Yerel arama algoritması Hooke-Jeeves ile arama yapılmış bir uzay

Şekil 2.3'te görüldüğü gibi Hooke-Jeeves yerel maksimumu bulabilmiş fakat global maksimumu bulamamıştır. Moser (2009), Boyutsal Yerel Arama (DLS) algoritması ile Hooke-Jeeves'i karşılaştırmış ve Hooke-Jeeves'in daha düşük hatalar elde ettiği görülmüştür. Bu algoritmalar başlangıç şartına göre yakınsadığı için sonuçlar başlangıç şartına göre değişebilmektedir. Örneğin bu tip algoritmalar Şekil 2.4'te görüldüğü üzere 3 minimumdan herhangi birinde yakınsayabilmektedir.



Şekil 2.4. Yerel arama algoritmalarında yakınsama

Bulatović ve Đorđević (2004), Şekil 2.5'teki mekanizmanın belirtilen noktalardan geçmesi için Hooke-Jeeves algoritması kullanmışlardır. Bu noktalar başlangıçtaki yörüngeye yakın olduğu için yerel bölge içinde Hooke-Jeeves ile arama yapılabilmektedir.



Şekil 2.5. Hedef noktalar, başlangıçtaki yörüngeye yakın olduğundan Hooke-Jeeves'in tercih edildiği bir mekanizma sentezi

Şekil 2.5'teki optimizasyon Hooke-Jeeves ile mümkündür. Fakat daha uzaklardaki ve daha zor noktalar istenseydi, Hooke-Jeeves bunu başaramayabilirdi. Böyle durumlarda global minimumu bulmak için rastgele değerler alabilen algoritmalar kullanılmalıdır. Sürü algoritmalarından olan Genetik Algoritması, Arı Algoritması ve Karınca Algoritması gibi algoritmalar global minimumu bulmada kullanılan algoritmalardan bazılarıdır.

Bulatović ve Đorđević (2012), benzer şekilde altı uzuvlu bir mekanizmanın optimizasyonunda bu defa diferansiyel gelişim algoritması kullanmışlar ve çok uzuvlu mekanizmalarda Hooke-Jeeves gibi algoritmalardan daha iyi sonuç verdiği görülmüştür. Diferansiyel gelişim algoritması sürekli olarak, hatanın değişken dizisindeki her bir uzuv boyutunun önceki boyuta göre değişimi üzerinden sonraki dizi tahminini yapmaya odaklanmaktadır. Bu yüzden diferansiyel gelişim algoritması da global minimumu bulmayı garantileyememektedir.

Farklı başlangıç şartlarının farklı yerel minimumları elde etmesi mekanizma sentezinde bir başka problemlerdendir. Fernandes ve ark. (2013), farklı başlangıç şartlarını paralel bir şekilde Hooke-Jeeves'e uygulatarak farklı yerel minimum çözümler de elde etmişlerdir.

Bazı algoritmalar dizi sınırları içinde kalacak şekilde rastgele diziler üretirler. Bu tip algoritmalar yerel minimum veya maksimumları bulmakla birlikte global minimum veya maksimumu da bulması mümkündür.

Pham ve Kalyoncu (2009), tek uzuvlu esnek robot kolunun bulanık mantık kontrolcü parametrelerinin optimizasyonunda Arı Algoritması kullanmışlardır. Hata fonksiyonunda ise step pulse sonucundaki yükselme zamanı, bekleme zamanı, tepe zamanı, en yüksek hedef aşımı, kalıcı hata ve yer değiştirme sonuçlarını farklı ağırlıklarla çarparak bir hata fonksiyonu elde edilmiş ve bunun minimize edilmesi için arı algoritmasından yararlanmışlardır.

Arı Algoritması ile Parçacık Sürü Optimizasyonu, Evrimsel Algoritması ve Yapay Arı Koloni algoritmalarının karşılaştırmalarında birçoğunda en iyi sonucu Arı Algoritması, bunun ardından da Yapay Arı Kolonisi gelmiştir (Pham ve Castellani, 2009, Aghazadeh ve Meybodi, 2011).

Özbakir ve ark. (2010), Arı Algoritması ile tabu arama metotlarını hız bakımından karşılaştırmıştır ve Arı Algoritmasının diğerlerinden daha kısa sürede optimizasyonu tamamladığı görülmüştür.

Arı algoritmasının, Parçacık Sürü Optimizasyonu ve Karınca Koloni Optimizasyonu algoritmalarından üç önemli farklılığı vardır. Birincisi Arı Algoritması hata fonksiyonunun yakınsamasındaki hız değişikliği ile ilgilenmemektedir. Fakat diğerleri yakınsamadaki hızın değişikliğine bağlı olarak yeni diziler üretir. İkinci farklılık ise araştırma araması (araştırmacı arılar) ile sömürücü araştırma (işe alınan avcı toplayıcı) yöntemlerinin kullanılmasıdır. Bu davranış farklılığı Arı Algoritmasının parametrelerini hafifletmedir. Yani daha az parametre ile optimizasyona başlayabilmektedir (Pham ve Castellani, 2009).

Pham ve ark. (2011), Arı Algoritması ile deterministtik tek yönlü metot (deterministic simplex method), Genetik Algoritması ve Karınca Koloni Algoritmasını karşılaştırmış ve sekiz adet amaç fonksiyonunda Karınca Koloni Algoritmasından 25 defa daha hızlı ve Genetik Algoritmasından 10 defa daha hızlı olduğunu gözlemlemişlerdir.

Pham ve ark. (2007), Ayrık Parçacık Sürü Optimizasyonu, Tabu Arama, Genetik Algoritması, Tabu Arama ile Genetik Algoritmasının Kombinasyonu, Genetik Algoritması ile Tabu Arama Kombinasyonu ve Arı Algoritması karşılaştırmıştır ve Arı Algoritmasının diğerlerinden daha stabil ve kararlı olduğunu vurgulamıştır.

Genetik Algoritması ile Arı Algoritması arasındaki en temel farklılık, Genetik Algoritmasının değişken dizilerinin hatalarını hesapladıktan sonra en iyi değişken dizilerini seçmektedir ve bunlar arasında çaprazlama yaparak mutasyona uğratarak yeni bir değişken dizisi üretmektedir. Fakat Arı Algoritması en iyi değişken dizisi etrafında sadece komşuluk araştırması yaparak bu komşuluktaki en iyi sonuç ile devam etmektedir. Ayrıca Genetik Algoritması araştırma uzayına rastgele değil mutasyon sonucu oluşan dizilerle yayılmamaktadır, fakat Arı Algoritması araştırma uzayına rastgele yayılmaktadır.

Luo ve ark. (2008), Newton-Raphson metodu ile mekanizma sentezi yapmışlardır. Newton-Raphson metodunun da yerel minimum bulmada etkili olduğu görülmüştür.

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}, \qquad k = 0,1,2,... \tag{2.4}$$

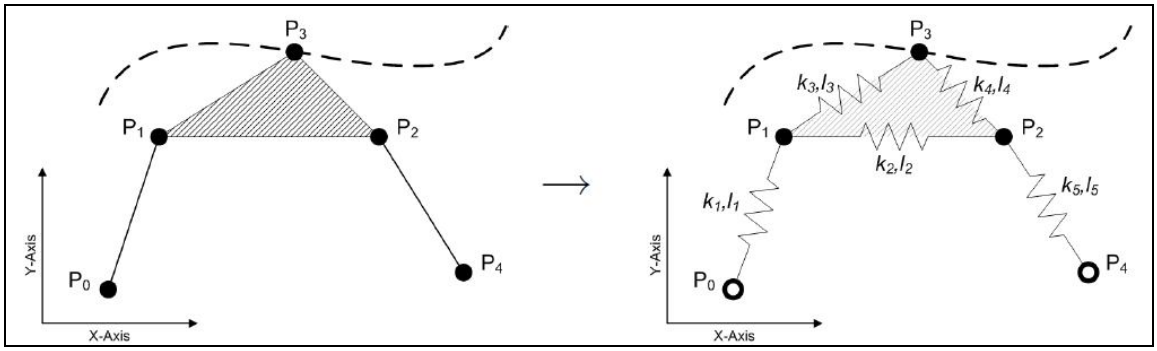
Newton-Raphson metodu Denklem 2.4'teki gibi iterasyonlar yapmaktadır. Hata fonksiyonu $f(x_k)$ ve hata fonksiyonunun türevi $f'(x_k)$ olmak üzere; x_k 'dan x_{k+1} 'i hesaplamaktadır. Bu yöntem, amaç fonksiyonunun türevine de ihtiyaç duymaktadır.

Birçok optimizasyon tekniği tam bir sonuç bulmayı garantileyememektedir. Sadece hata fonksiyonunun sonucunu azaltmaya çalışmaktadırlar. Bazı optimizasyon problemleri yerel minimumda kalabilirken, bazıları da global minimumu bulabilmektedir. Başlangıç şartlarının farklılığı da sonucu etkileyebildiği için farklı başlangıç şartları ve farklı optimizasyon tekniklerini beraber kullanmak bizi doğru sonuca daha fazla yaklaştırabilmektedir.

Sheu ve ark. (2008), dört çubuk mekanizması kinematik sentezi çalışmasında, Genetik Algoritması ve devirli koordinat düşme metodunu aynı başlangıç şartları ile kullanmışlardır. Devirli koordinat düşme metodunun sonuca ulaşamadığı fakat Genetik Algoritmasının sonuca ulaşabildiği gözlemlenmiştir.

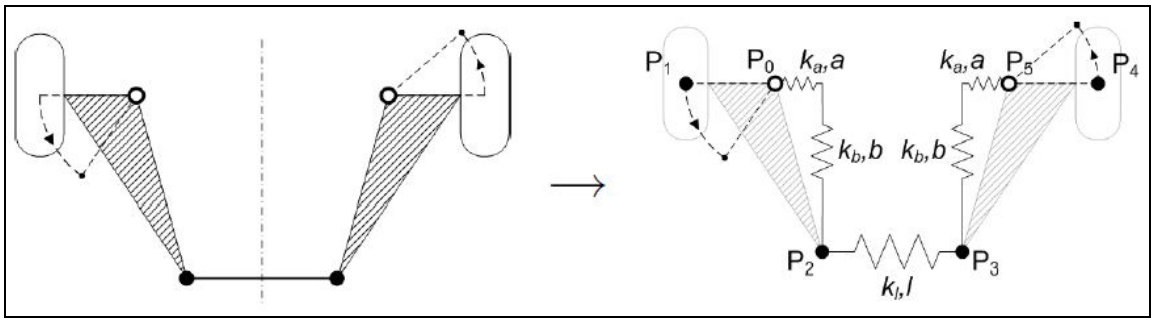
Rios-Coelho ve ark. (2010), bir Metropolis algoritmasını Hooke-Jeeves algoritması ile kombinasyon oluşturmuş ve bu kombinasyonun optimizasyon işlemini hızlandırdığı görülmüştür. Yine benzer bir şekilde Long ve Wu (2014), Genetik Algoritması ile Hooke-Jeeves algoritmasını birleştirmişlerdir. Bu iki algoritma birlikte çalıştığında daha iyi sonuçların elde edildiği görülmüştür. Yerel arama algoritmasının global optimizasyon algoritmasına yerel bölgelerde yardım ettiği görülmüştür.

Collard ve ark. (2010), mekanizma sentezi çalışmalarında Uzayabilir Uzuv yaklaşımı kullanmışlardır. Bu düzlemsel mekanizmalarda yeni bir yaklaşımdır. Bu metot ile fonksiyon ve yörünge sentezi yapılabilmektedir. Bu metot ile yayların gerilme enerjisini minimuma indirerek optimum rijit dizaynı elde etmek amaçlanmıştır. Optimizasyon algoritması olarak standart Genetik Algoritması yerine kullanıcının yönlendirmesi ile birden fazla başlangıç şartının uygulandığı Genetik Algoritması kullanılmıştır.



Şekil 2.6. Uzayabilir uzuv yaklaşımı ile mekanizma sentezi

Şekil 2.6'da; soldaki mekanizma, sağda uzayabilir uzuv yaklaşımına dönüştürülmüştür. Şekildeki P_0 ve P_4 sabit iken P_3 yörüngeyi takip etmeye çalışırken çubuk-yaylar esnemekte veya sıkışmaktadır. Bu optimizasyonda amaç, yayların gerilme enerjilerini minimize etmektir. Şekil 2.7'de düzlemsel Ackerman rijit geometrisi, uzayabilir uzuv yaklaşımına göre düzenlenmiştir. Genetik Algoritmasının bu hali ile kullanılması sayesinde düzlemsel Ackerman geometrisinin Ackerman hatası 0.85° 'ye kadar düştüğü görülmüştür.



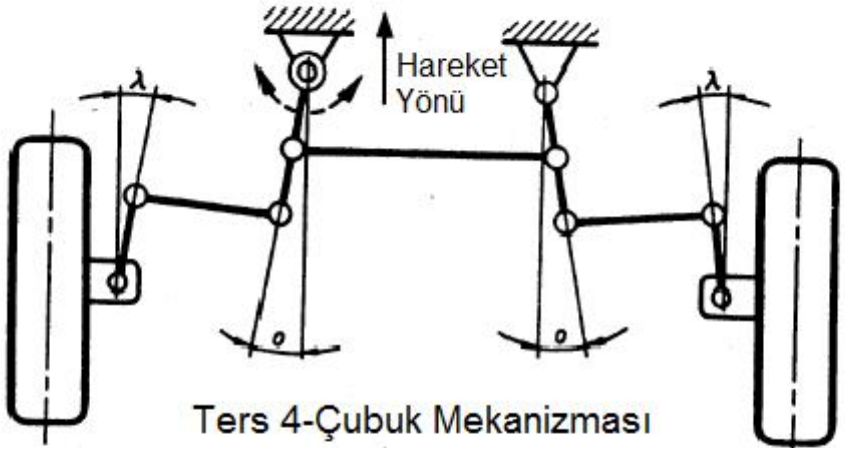
Şekil 2.7. Uzayabilir uzuv yaklaşımının Ackerman dümenleme sistemine uyarlanması

Ansarey ve ark. (2005), kremayer ve pinyonlu direksiyon mekanizmasını Genetik Algoritması ile optimize etmişlerdir. ADAMS modeli ile optimizasyon sonucunu karşılaştırmışlardır.

Wang ve ark. (2014), gerçekte tekerleklerdeki esnemeleri de düşünerek direksiyon mekanizmasının matematiksel modelini sadeleştirmişlerdir. İç tekerleğin mutlak dümenleme açı hatasını ve binom ağırlık fonksiyonunu temel alan bir amaç fonksiyonu oluşturmuşlardır. Bu amaç fonksiyonunu MATLAB'ın içindeki Genetik Algoritması bileşeni ile optimize etmişler ve çözümü ekonomik tasarruflu bir vasıta üzerinde test etmişlerdir.

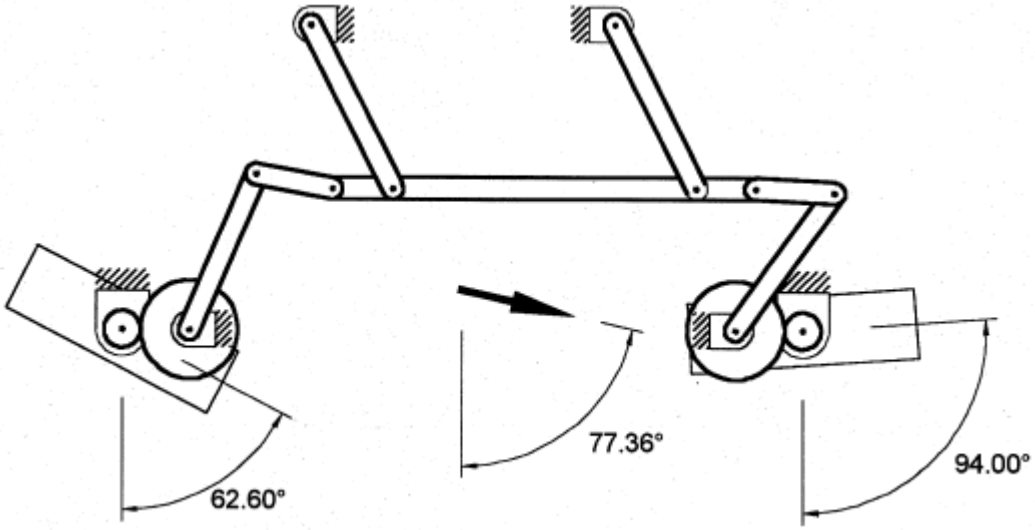
Lohith ve ark. (2013), dört tekerlekli bir vasıtada dört tekerden dümenleme sistemi tasarımı yapmışlardır. Bunun sonucunda dönüş radyüsünü düşürmüşlerdir.

Burha (2010), çalışmalarında tasarladıkları 6x6 vasıtanın aks mekanizması olarak ters dört çubuk mekanizması kullanmışlardır. Ackerman hatasını 2°-2.5° 'den 1°-1.5° 'ye indirmişlerdir. Ayrıca aks tasarımında 1.5°'ye kadar Ackerman hatasının kabul edilebilir olduğu belirtilmiştir. Şekil 2.8'de tasarladıkları ters dört çubuk mekanizması görülmektedir.

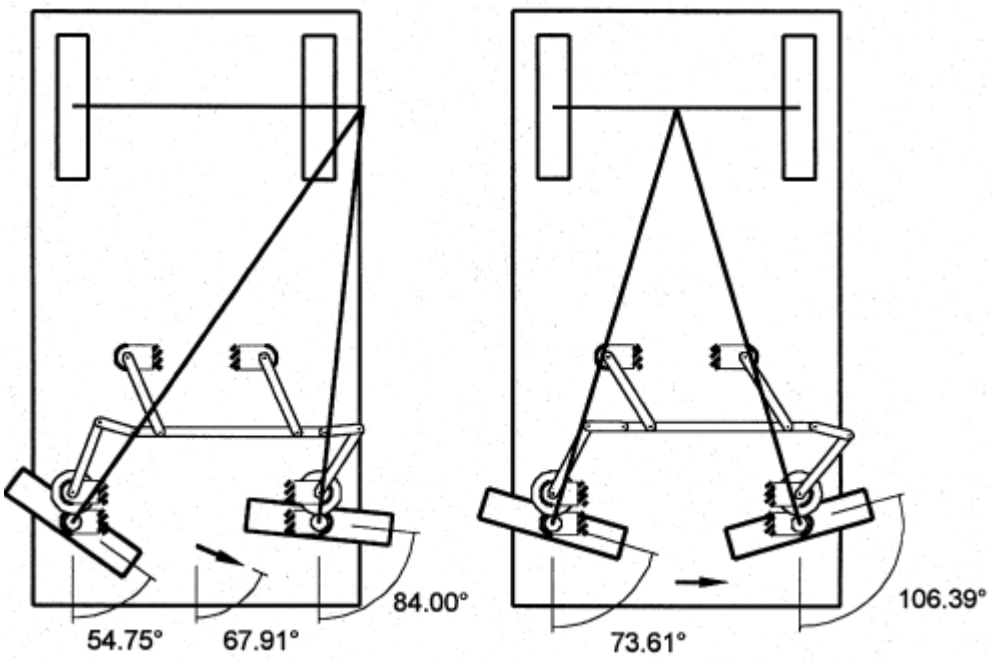


Şekil 2.8 Ters dört çubuk mekanizması kullanılarak yapılmış bir Ackerman geometrisi

Chicurel (1999), Şekil 2.9'da görülen dar alanlara park problemini çözebilmek için yüksek manevra kabiliyetli yeni tip dümenleme çubuk mekanizmaları tasarlamıştır. 'da ise 90°'den daha fazla dönebilen bu yeni tip tasarım, geleneksel aks mekanizmalarından çok daha fazla çeşitlilikte dönüş radyüs merkezi elde edebildiği görülmektedir.



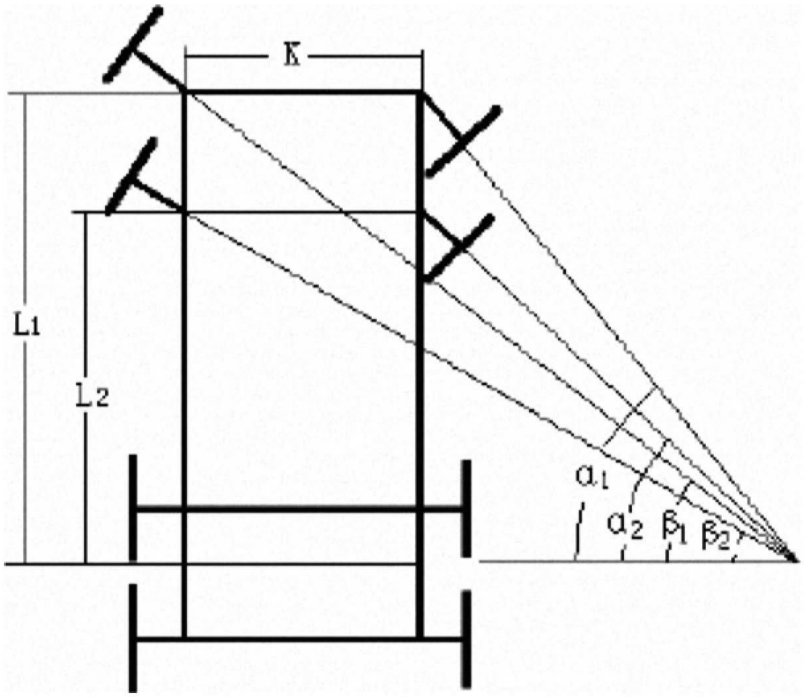
Şekil 2.9 90° 'den daha fazla dönebilen Ackerman geometrisi



Şekil 2.10 Çeşitli dönüş radyüs merkezlerinin elde edilmesi

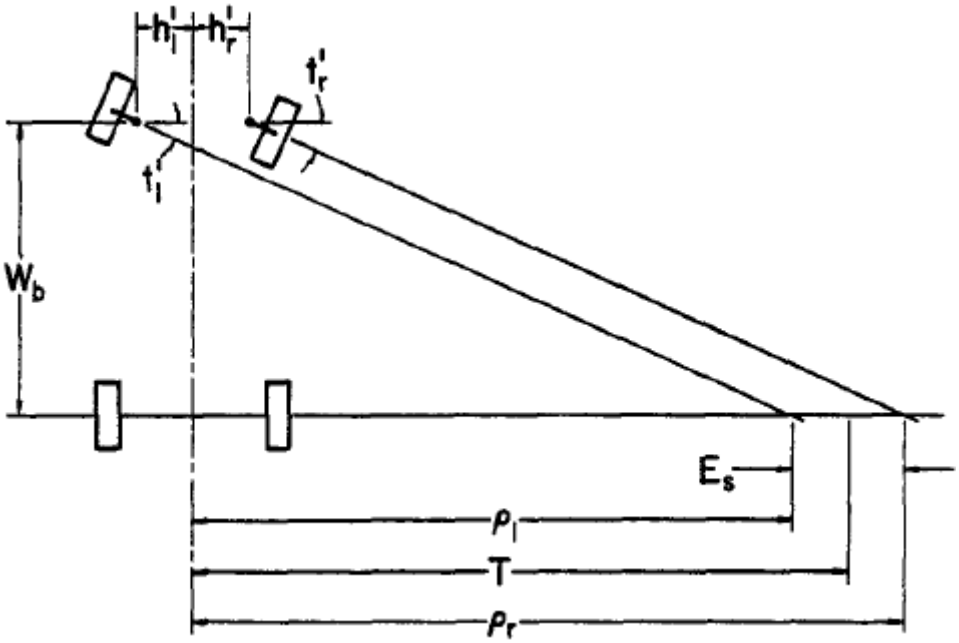
Chunyan ve ark. (2010), Şekil 2.11'de görüldüğü gibi düzlemsel mekanizma kabulü ile iki aksın dümenlendiği vasıtalarda, dümenlenen her bir tekerin olması

gereken açıyı matematiksel olarak elde etmişlerdir. Fakat düzlemsel kabulü, bizim çalışmalarımızda gerçekte üç boyutlu olan aks mekanizmaları için 0.90° hatalı hesaplamaya sebep olmaktadır.

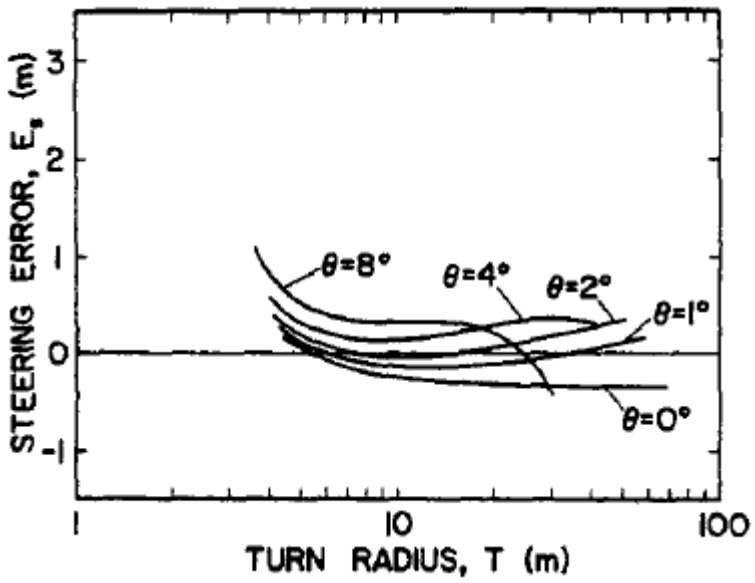


Şekil 2.11 Düzlemsel olarak iki aksın dümenlenmesindeki parametreler

Felzien ve Cronin (1985), dönüş radyüsüne bağlı olarak ön aksın tekerlerinin arka aks eksenini kestiği noktalar arasındaki mesafeyi metre biriminde dümenleme hatası olarak almışlardır. Şekil 2.12'de E_s dümenleme hatası ve T dönüş radyüsü olarak gösterilmiştir. Bu hatanın optimizasyonu çalışmasında cezalandırma fonksiyonu da kullanmışlardır. Optimizasyon sonucunda dönüş radyüsüne bağlı olarak metre biriminde dümenleme hatası grafiği Şekil 2.13'te görülmektedir.



Şekil 2.12 Mesafe olarak dümenleme hatası



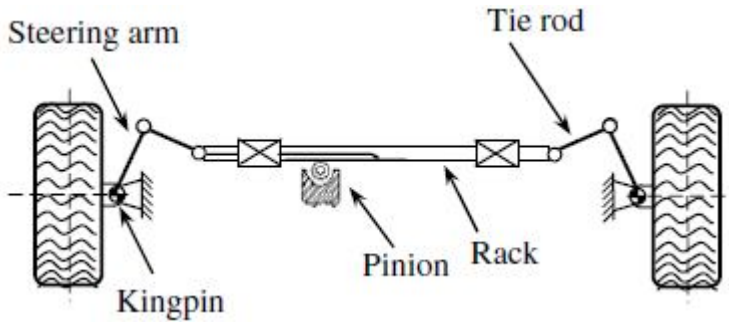
Şekil 2.13 Dönüş radyüsüne göre dümenleme hatası

Pingyuan ve Cheng (2009), Genetik Algoritması ile yerel gradient temelli algoritmalarla kombinasyonunu yaparak, bir aks mekanizmasını optimize etmişlerdir. Bu kombinasyon ile Genetik Algoritmasına yerel bölgelerde yardımcı olunmuştur. Aks mekanizmasındaki Ackerman koluna bağlı olan küçük rodların boyu ve bunların tekere bağlama açısı optimize edilmiştir. Hata fonksiyonu olarak da her bir dümenleme konumunda Ackerman hatasının karelerinin toplamı alınmıştır.

Hamidi ve ark. (2012), Yapay Bağışıklık Sistemi (Artificial Immune System) Algoritması ile aks mekanizmasının Ackerman hatasının optimizasyonunu yapmışlardır.

Geometrik kısıtlar içinde kalacak şekilde optimizasyonu yaptıklarında 7°-8° derece civarına optimize etmişlerdir. Geometrik kısıtlar ihmal edildiğinde bu hatayı 1° 'ye optimize etmişlerdir. Yine benzer şekilde Javash ve ark. (2013), Hamidi ve ark. (2012)'nın yaptığı çalışmayı Genetik Algoritması ve Yapay Bağışıklık Sistemi yapmışlardır. Geometrik kısıtlar ihmal edildiğinde Ackerman dümenleme hatasını 1° 'ye indirmişlerdir.

Hanzaki ve ark. (2009), Şekil 2.14'te görülen side take-off tipi Ackerman geometrisine sahip aks mekanizmasını optimize etmişlerdir. Optimizasyon algoritması olarak Richardson bilinene dayanan tahmin algoritmasını (Richardson's extrapolation algorithm) kullanmışlardır. Optimizasyon sonucunda 3.9° olan Ackerman dümenleme hatasını 0.75° 'ye düşürmüşlerdir.



Şekil 2.14 Side take-off tipi Ackerman geometrisi

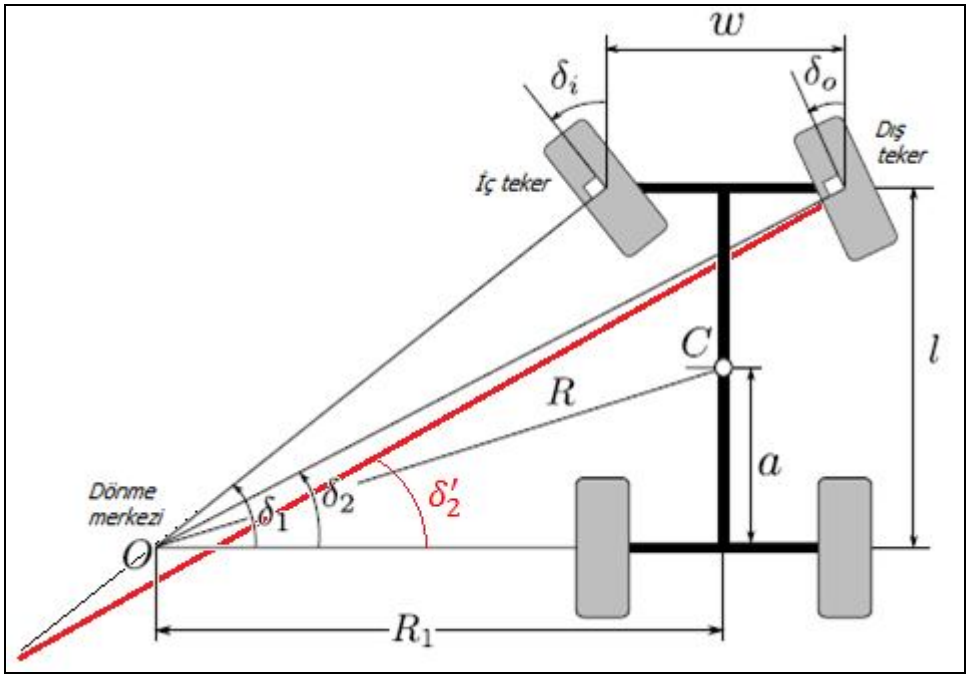
Li ve Yang (2010), çok akslı bir vasıtada her bir tekeri mekanik bağımsız olarak dümenlemişlerdir. Dümenleme açılarını bir PID kontrolcü ile gerçekleştirmişlerdir. Kontrol çalışması sonucunda 2° ile 3° arasında bir dümenleme hatası oluşmuştur.

Simionescu ve Beale (2012), dört çubuk mekanizması fonksiyon sentezinde, bir yörüngede hata fonksiyonu olarak maksimum hatanın karesinin alınmasını önermiştir ve bunu bir Ackerman dümenleme mekanizmasında uygulamışlardır. Optimizasyon algoritması olarak Brent algoritmasını kullanmışlardır. Aks arası mesafenin tekerler arası mesafeye oranına ve olarak maksimum dümenleme açısına bağlı olarak 0° ile 7° arasında değişen bir dümenleme hatasına ait geometri seçimi için tavsiye niteliğinde grafikler hazırlamışlardır.

Zhou ve Yang (2009), Matlab'ın Genetik Algoritması eklentisi ile yaptıkları iki akslı bir vasıtanın ön aks mekanizmasının optimizasyonunda Ackerman hatasını 1.9° 'den 0.4° 'ye indirmişlerdir.

Bu tez çalışmasında beş akslı ağır vasıtanın ilk iki aksı arasındaki mekanizma optimize edilmiştir. Kaynak araştırmasında ise aksın kendi içindeki mekanizmaları optimize etmeye yönelik çalışmalara rastlanılmıştır (Erdemir ve Kalyoncu, 2015).

Ackerman dümenleme hatası, hedeflenen tekerlek doğrultusu ile gerçekleşen tekerlek doğrultusu arasındaki açıdır (Zhou ve Yang, 2009). Genellikle bu hatanın 1° 'nin altında olması istenmektedir.



Şekil 2.15. Düzlemsel Ackerman geometrisi

$$\cot\delta_2 - \cot\delta_1 = \frac{\omega}{l} \tag{2.5}$$

$$\delta_1 = arc\cot\left(\cot\alpha - \frac{\omega}{l}\right) \tag{2.6}$$

Düzlemsel olarak Ackerman geometrisi Şekil 2.15'de görülmektedir. Ackerman geometrisi için kullanılan matematiksel bağıntılar Denklem 2.5 ve Denklem 2.6'daki gibidir. δ'_2 gerçekteki dümenleme açısı; ve δ_2 Ackerman'a göre gereken dümenleme açısı olmak üzere, bu durumda Ackerman hata fonksiyonu Denklem 2.7'deki gibidir.

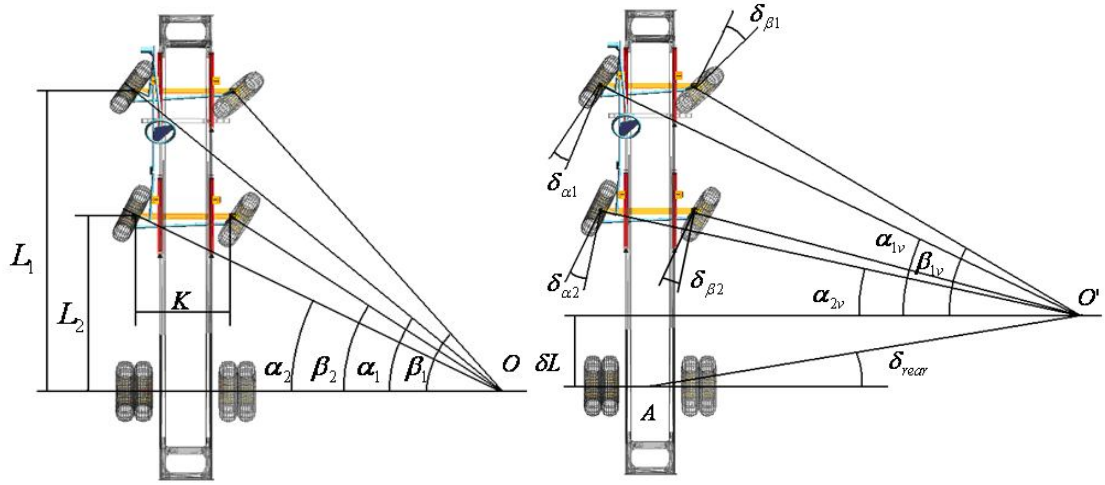
$$f = \left|\delta'_2 - \delta_2\right| \tag{2.7}$$

Ardayfio ve Qiao (1987), yedi mafsallı bir Ackerman geometrisinin üç boyutlu bir matematiksel modellenmesini çıkarmışlardır. Çalışmalarında yedi adet parametre kullanmışlardır ve bu her bir parametrenin ön aks tekerlerinin kesişim noktasının arka aks eksenine dik uzaklığına nasıl etki ettiğinin grafiklerini çıkarmışlardır.

Üç akslı bir vasıtada ikinci aksın üçüncü aksa uzaklığının birinci aksın üçüncü aksa uzaklığına aks oranı denir. Van Eeden (2007), çalışmasında üç akslı bir vasıta tasarımı yapmıştır. Üç akslı bir vasıtada aks oranının dönüş esnasında tekerlerdeki kayma açılarına etkisi incelenmiştir. Aks oranının 0.5 olması durumunda bu kayma açılarının minimum olduğu gözlemlenmiştir.

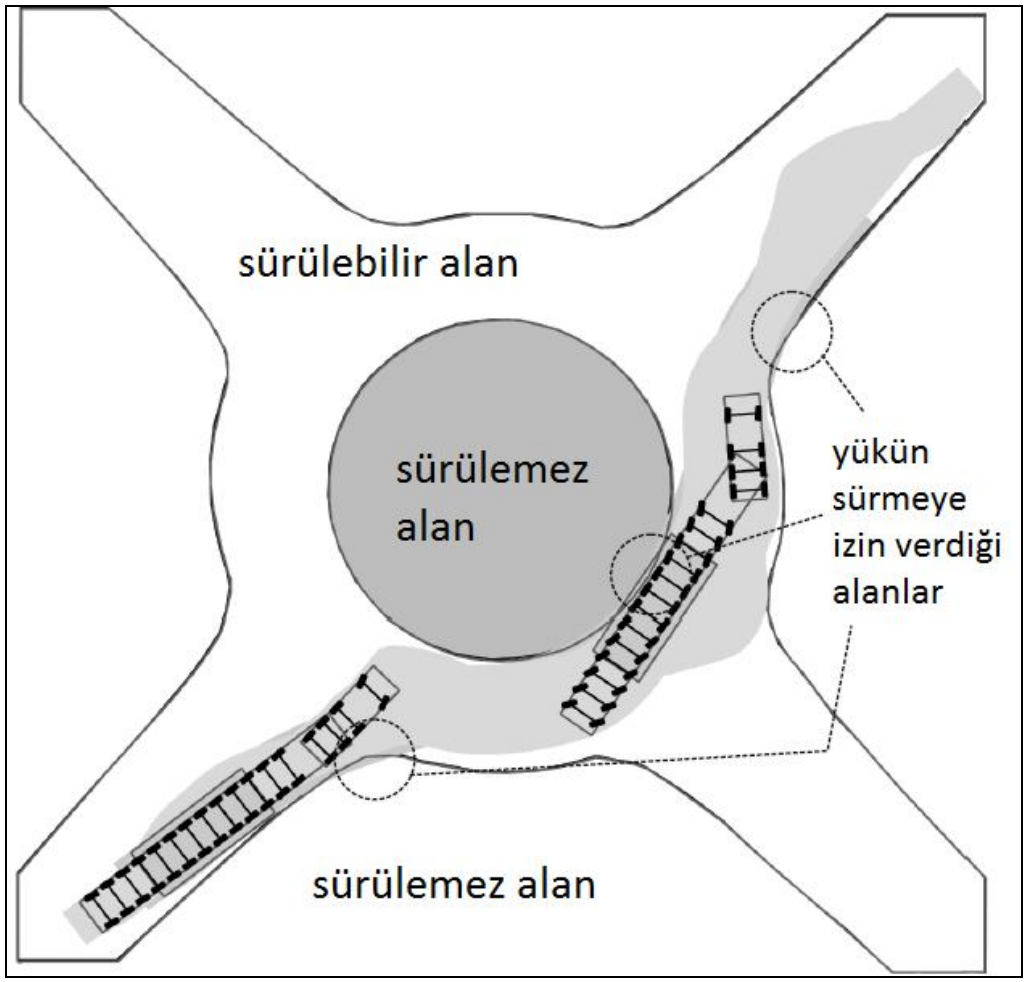
Watanabe ve ark. (2007), dönüş esnasında dönüş radyüs merkezine doğru vasıtanın tekerlerine etki eden sürtünme kuvvetlerinin etkisi ve aksların dümenleme açılarının etkisi incelenmiş ve dümenleme için gereken enerji incelenmiştir. Yetersiz dümenleme (understeer) durumunda sürtünme kuvvetleri oldukça fazlaca olmaktadır. Orta aksın dümenleme üzerinde etkisinin az olduğu görülmüştür. Vasıtanın arka tekerlerinden de dümenlenmesinin, daha az enerji ile dönmeyi sağladığı görülmüştür.

Bir vasıtanın düşük hızlarda giderken teker eksenlerinin kesişim bölgesi etrafında vasıta manevra yapmaktadır (Jujnovich ve Cebon, 2013, Wang ve ark., 2008). Fakat düşük olmayan hızlarda tekerlere ilerleme yönünde olmayan kuvvetler geldiği için tekerlerde kayma olmaktadır ve bu da manevra merkezini değiştirmektedir. Qin ve Gang (2012), bu değişimin nelere bağlı olduğunu incelemişlerdir. Düşük hızlarda tekerler kaymıyor iken Şekil 2.16'da vasıta soldaki şekildeki gibi merkez etrafında dönmektedir. Düşük olmayan hızlarda dönen vasıtanın tekerlerde oluşturduğu kayma sebebiyle vasıtanın farklı bir merkez etrafında dönmesi ve bunun tekerler üzerinde oluşturduğu eğilme momenti ve burulma momentinin yaprak yay süspansiyonuna verdiği zararı ve tekerleklerin bu hızlarda dönme merkezine olan açı hatasını azaltmayı hedeflemişlerdir.



Şekil 2.16. Solda düşük hızlarda manevra merkezi, sağda yüksek hızlarda manevra merkezi

Çok akslı vasıtalarda dümenleme sistemleri dar yollarda manevra esnasında da büyük önem arz etmektedir. Beyersdorfer ve Wagner (2013), çok uzun vasıtalarda birçok aksın farklı yönlere doğru dümenlenerek dar yerlerden dahi güvenle geçilmesinin mümkün olabildiği bir çalışma yapmışlardır.



Şekil 2.17. Birçok aksın farklı yönlerde dümenlenmesi ile dar bölgelerde manevra yapılabilmesi

Şekil 2.17'de görüldüğü gibi birçok aks farklı yönlerde hareket ederek dar yollarda manevra mümkün hale gelmiştir.

Shufeng (2010), altı tekerli bir vasıtada altı tekerden dümenleme ile dört tekerden dümenleme sistemlerini dönüş esnasındaki savrulma açısından karşılaştırmış ve dört tekerden dümenlemenin daha az savrulma getirdiğini gözlemlemiştir. Aynı şekilde beş akslı ağır vasıtalarda da yüksek hızlarda arka akslar kilitlenmektedir.

Chang ve Joo (2006), bir vasıtanın dinamik etkiler altında süspansiyon konumlarının CAD programları ile optimize edilmesi çalışması yapmışlardır. Bu çalışmalarında ProEngineer ve Solidworks API kullanarak CAD modelinden yararlanmışlardır.

Bu tez çalışmamızda, dümenleme mekanizması Solidworks CAD programında modellenmiş olup, bu mekanizma VisualBasic.NET kullanılarak Arı Algoritması ile optimize edilecektir. Hata fonksiyonu, Solidworks API aracılığıyla CAD modelinden direkt olarak hesaplanacaktır. Mekanizma optimizasyonu işleminde Arı Algoritması kullanılarak global minimum ve komşu yerel minimum alternatif çözümleri de bulunabilecektir. Diğer bir ifadeyle, Ackerman hatasını minimize eden birden fazla CAD konfigürasyonu bulunmuş olacaktır.

3. ÇOK AKSLI DİREKSİYON MEKANİZMASINA SAHİP BİR AĞIR VASITA

Bu çalışmada elde edilen sonuçların uygulaması gerçek bir ağır vasıta üzerinde yapılmıştır. Direksiyon mekanizması optimize edilen çok akslı ağır vasıta, MPG Makina Prodüksiyon Grubu Makine İmalat San. ve Tic. A.Ş. firmasının üretmiş olduğu beş akslı kendinden yürüyüşlü T0120 adlı bir teleskobik vinçtir. Çok akslı ağır vasıtanın fotoğrafı Şekil 3.1'de verilmiştir.

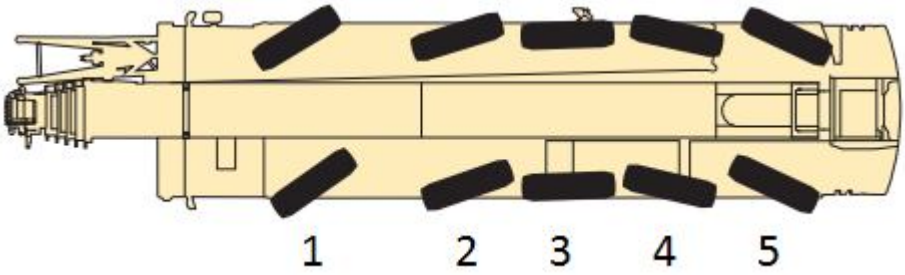


Şekil 3.1. Uygulamada kullanılan çok akslı ağır vasıta

Türkiye'nin ilk yerli üretim beş akslı kendinden yürüyüşlü arazi vasıtasıdır. T0120 teleskobik vinci 123 ton yük kaldırabilmekte ve 60 m'ye erişebilmektedir.

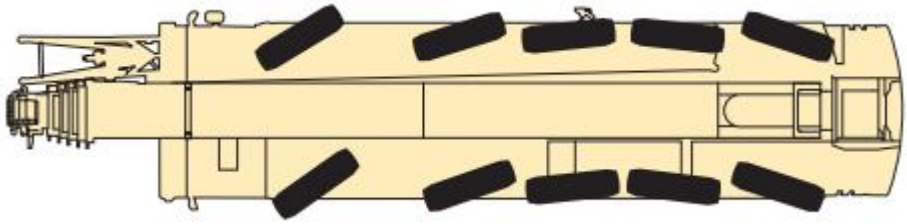
T0120 vasıtası "10 x 6 x 10" yürüyüş özelliğine sahiptir. İlk geçen 10 sayısı teker sayısını, 6 sayısı çekiş yapabilen teker sayısını ve sondaki 10 ise dümenlenebilen teker sayısını göstermektedir. 2, 4 ve 5 no'lu akslardan çekiş yapabilmektedir. 12 adet ileri ve 2 adet geri vitese sahiptir. En yüksek hız olarak 80 km/h hıza sınırlandırılmıştır.

Bu çalışmaya konu olan çok akslı ağır vasıtalar direksiyon mekanizmaları sayesinde beş farklı yürüyüş modunda hareket edebilmektedirler (Liebherr, 2014). Bunlar "tüm teker dümenleme" (all-wheel-steering), "yol dümenleme" (road steering), "yengeç dümenleme" (crab-steering), "azaltılmış menteşeleme" (reduced swing out) ve "bağımsız arka aks dümenleme" (independent rear axle steering)'dır.



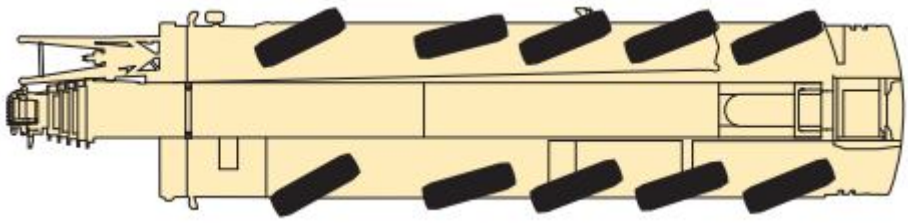
Şekil 3.2. Tüm teker dümenleme modu

Şekil 3.2'te tüm teker dümenleme modu gösterilmiştir. Numaralandırmalar aksların numarasını ifade etmektedir. Tüm teker dümenleme modunda hıza bağımlı olarak dümenleme açıları değişmektedir. Bu yürüyüş modunda, tüm teker eksenleri tek bir noktada kesişerek minimum dönüş radyüsünü elde etmektedir.



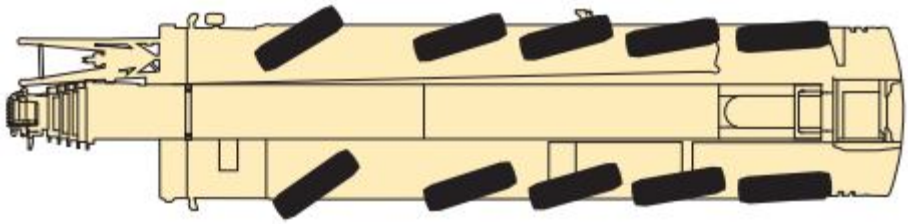
Şekil 3.3. Yol dümenleme modu

Şekil 3.3'te yol dümenleme modu gösterilmiştir. Hız 30 km/h üzerine çıktığında tüm teker dümenleme modu, yol dümenleme moduna dönüşerek, dinamik etkilerden dolayı dönüş radyüs merkezini sabit tutabilmek için 4 ve 5. akslar daha az dümenlenmektedir. Hız 60 km/h üzerinde 3, 4 ve 5. akslar 0° 'ye konumlanmakta ve kilitlenmektedir.



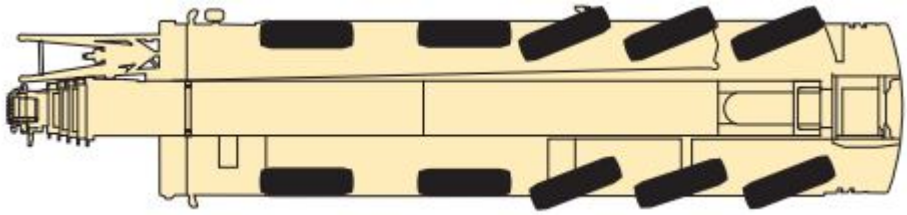
Şekil 3.4. Yengeç dümenleme modu

Şekil 3.4'te yengeç dümenleme modu gösterilmiştir. Yengeç dümenleme modunda vasıtanın tüm tekerleri bir yöne bakarak çapraz gidebilmeye olanak sağlamaktadır.



Şekil 3.5. Azaltılmış menteşeleme modu

Şekil 3.5'te azaltılmış menteşeleme modu gösterilmiştir. Bu modda vasıtanın ön akslarındaki dümenleme miktarı fazla arka akslarındaki daha az olmakta ve bu sayede vasıtanın ön kısmı geniş bir alanı süpürürken arka kısmı daha az hareket etmektedir.

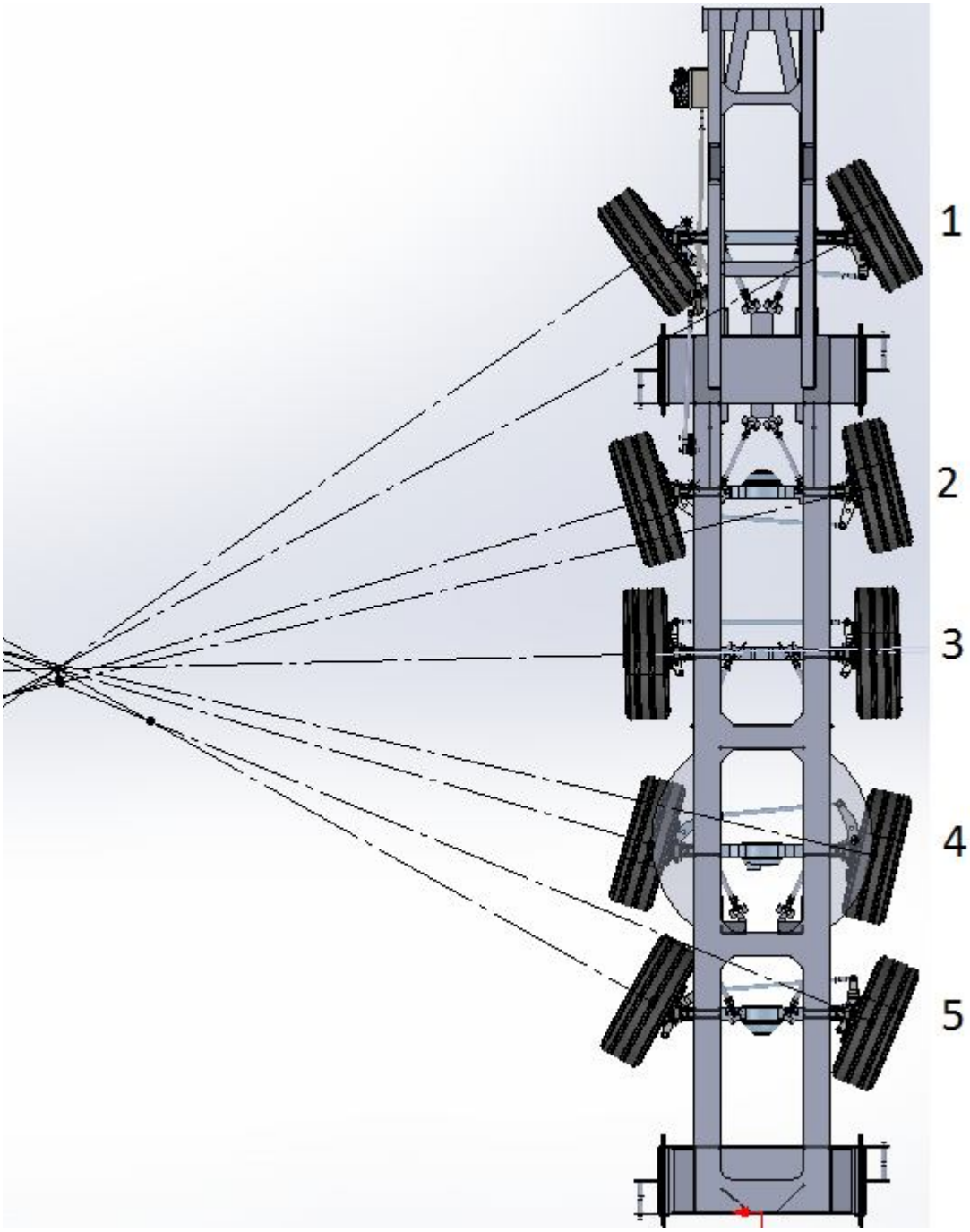


Şekil 3.6. Bağımsız arka aks dümenleme modu

Şekil 3.6'da bağımsız arka aks dümenleme modu gösterilmiştir. Bu modda vasıtanın ön akslarındaki dümenleme miktarı çok az arka akslarındaki çok daha fazla olmakta ve bu sayede vasıtanın arka kısmı geniş bir alanı süpürürken ön kısmı daha az hareket etmektedir. Böylelikle vasıta arka akslardan dümenlendiğinden dolayı dönme kabiliyeti oldukça artmaktadır.

Bu çok akslı ağır vasıtaların modlarının genel açıklamaları ışığı altında vasıtasının direksiyon mekanizmasına bakıldığında ise benzer özellikler farklı parametre değerlerine karşılık aynıdır. Bu tür vasıtalarda genel olarak aynı türden hareketler ve dümenleme modları mevcuttur.

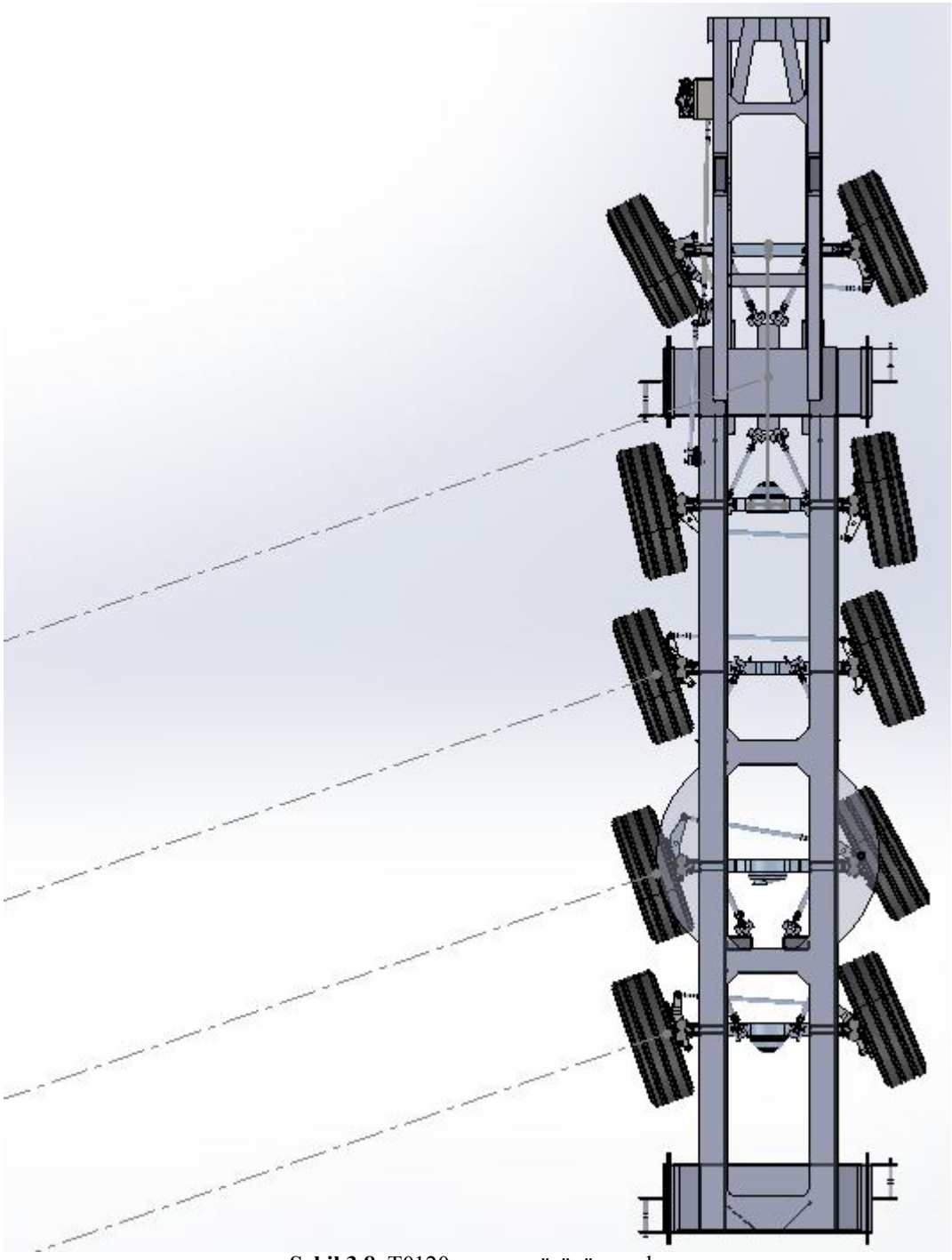
Bu çalışmada uygulama olarak kullanılan T0120 vasıtasının düşük hızlardaki tüm teker dümenleme modu ise Şekil 3.7'de görülmektedir. Şekildeki numaralandırmalar aksların numaralarını ifade etmektedir.



Şekil 3.7. T0120 tüm teker dümenleme modu

Tüm teker dümenleme modunda bütün tekerlerin eksenleri dümenleme açısına göre tek bir noktada kesişmektedir. Bu kesişen nokta aynı zamanda dönüş esnasındaki radyüs merkezidir.

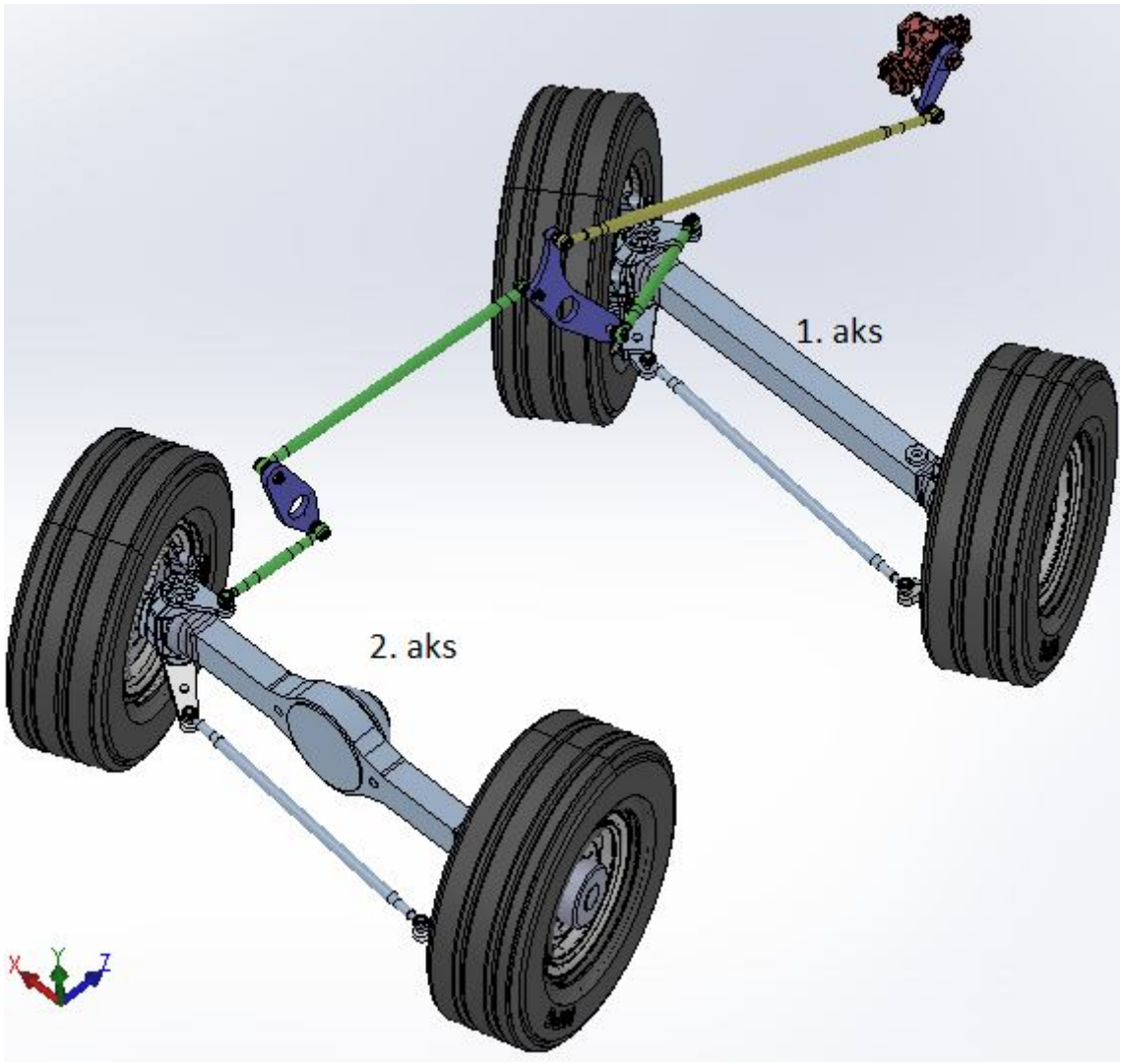
Bir diğer yürüyüş modu olan yengeç yürüyüşü Şekil 3.8'de görülmektedir.



Şekil 3.8. T0120 yengeç yürüyüş modu

Yengeç yürüyüş modu ile vasıta çapraz yönde ilerleyebilmektedir. Bu yürüyüş modu kısıtlı alanlarda vasıtanın konumlanabilme kabiliyetini büyük oranda artırmaktadır.

T0120 vasıtasında optimize edilecek olan direksiyon mekanizması Şekil 3.9'da gösterilmiştir.



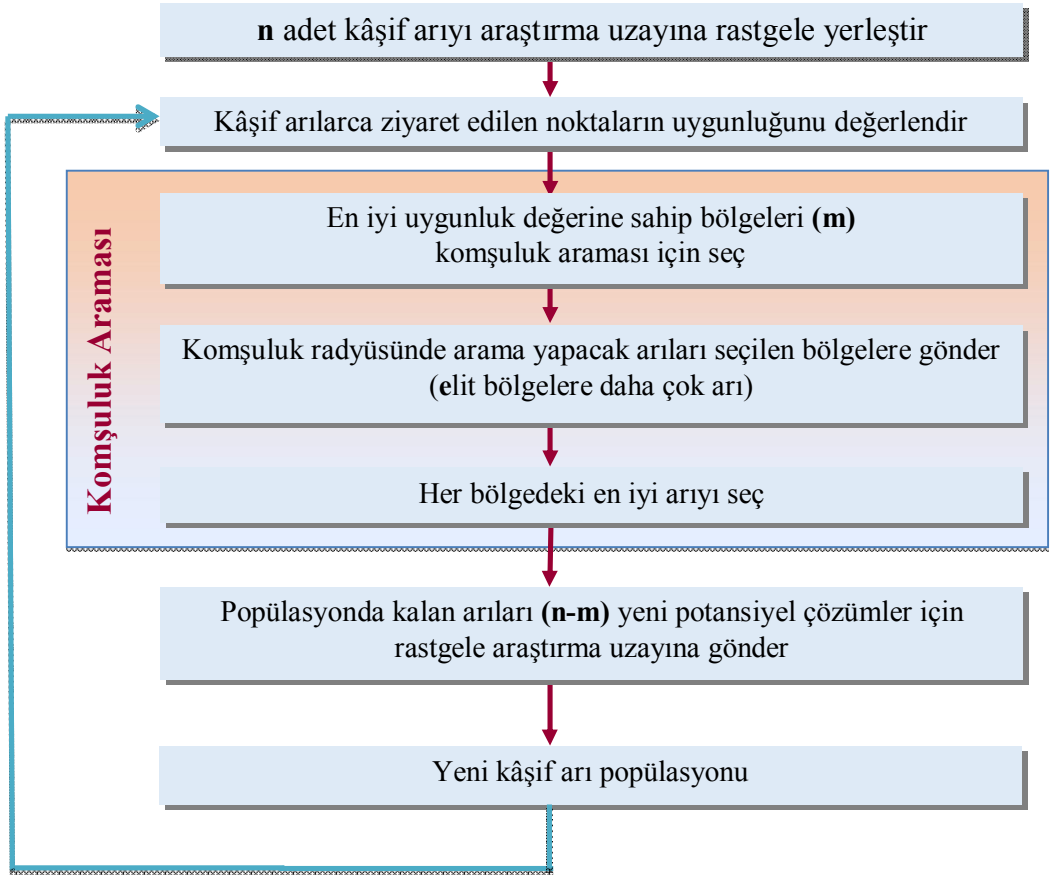
Şekil 3.9. 1. ve 2. aks arasındaki direksiyon mekanizması

4. OPTİMİZASYON ALGORİTMASI

Optimizasyonda global minimumu bulabilmek için bir global optimizasyon algoritması kullanılması gerekir. Bu nedenle Arı algoritması kullanılmıştır. Kullanılan bu global arama algoritmasının her bir yerel bölgede aramasını hızlandırabilmek için de yerel arama algoritmalarından faydalanılmıştır. Bunun için Hooke-Jeeves algoritması yerel arama algoritması olarak kullanılmıştır.

4.1. Arı Algoritması

Arı algoritması çok değişkenli bir global arama algoritmasıdır. Arı algoritması akış diyagramı Şekil 4.1'de görüldüğü gibidir.



Şekil 4.1. Arı Algoritması akış diyagramı

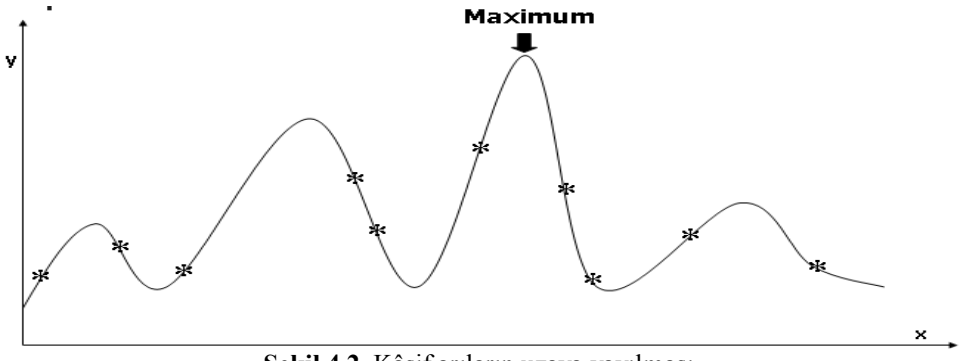
Arı Algoritması arama uzayına rastgele yerleşmiş *n* adet kâşif arısı ile başlamaktadır. Bu arıların her birinin hatası hesaplanmaktadır. Bu arıların oluşturduğu hata dizisi küçükten büyüğe doğru sıralanmaktadır. *m* adet en iyi arı, komşuluk

araştırması için seçilmektedir. Bu en iyi arılar etrafında komşuluk radyüsü içinde yeni diziler üretilmekte ve bunların hata sonuçları hesaplanmaktadır. Fakat bu *m* adet arı içindeki *e* adet en iyi olanı kıdemli sonuçlardır ve bu sonuçların etrafına daha fazla arı gönderilmektedir. Bu *m* arıdan her birinin etrafına gönderilen araştırmacı arılar da kendi aralarında tekrar küçükten büyüğe doğru sıralanmakta ve en küçük hataya sahip olanı en iyi arının yerini almaktadır. Kalan *(n-m)* arı yerine, sınır şartları içindeki uzaya rastgele yayılmış arılar üretilmektedir. Ve tüm arıların hatası tekrar en küçükten en büyüğe doğru sıralanmakta ve döngü durma şartı ile karşılaşana kadar devam etmektedir. Arı algoritmasının en basit hali ile ifadesi aşağıdaki gididir:

1. Kısıt şartları bağlı kalarak uzayda rastgele üretilmiş bir nüfus oluştur.
2. Nüfusun hatasını hesapla ve en küçükten en büyüğe doğru sırala.
3. Durdurma şartı gerçekleşmediği sürece döngüye devam et.
4. En iyi nüfusa ait bölgeleri komşuluk araştırması için seç.
5. En iyi nüfus bölgelerinin komşuluklarına arılar gönder (kıdemli *e* adet bölgeye daha fazla arı gönder) ve her bir arının hatasını hesapla.
6. Her bir komşuluk grubunun hatasını en küçükten en büyüğe doğru sırala.
7. Kalan arıları kısıt şartları içinde uzaya tekrar rastgele dağıt ve hatalarını hesapla
8. Döngüye tekrar başla (3. maddeye git)

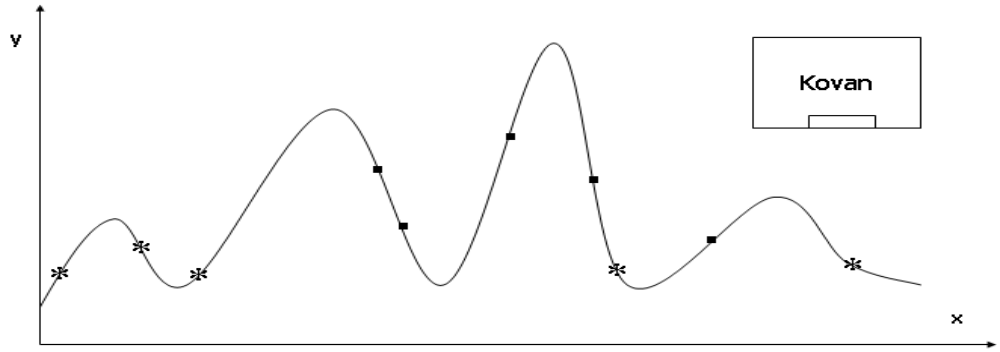
Örnek olarak Arı Algoritmasının tek değişkenli bir fonksiyonun optimizasyonu şu şekildedir. Bu iki boyutlu bir optimizasyon demektir.

1. adım: Algoritma 10 adet kâşif arının araştırma uzayına rastgele yerleştirilmesi ile başlamaktadır (Şekil 4.2).



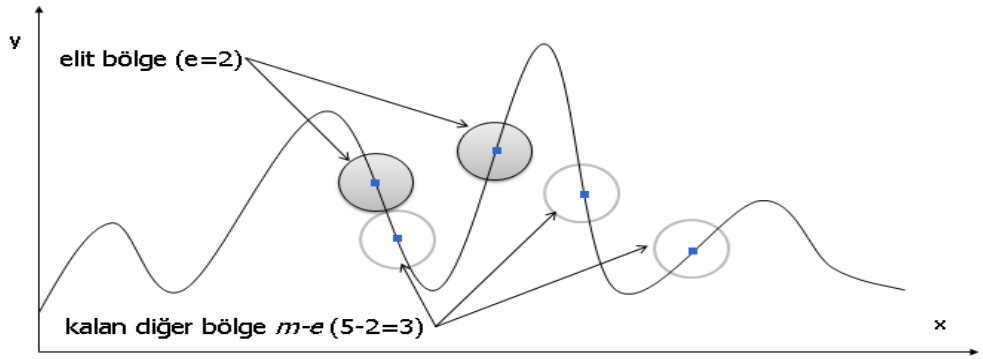
Şekil 4.2. Kâşif arıların uzaya yayılması

2. ve 3. adım: Kâşif arılar tarafından ziyaret edilen noktaların birbirlerine göre uygunlukları değerlendirilerek diğerlerine göre daha uygun sahip bölgeler seçilmektedir (Şekil 4.3).



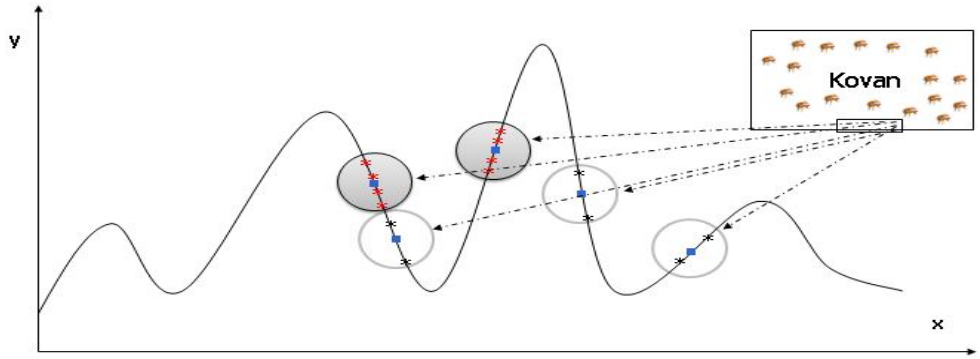
Şekil 4.3. En iyi arıların seçilmesi

4. adım: En iyi uygunluk değerine sahip elit bölgeler ve geriye kalan bölgeler seçilmektedir. Bu bölgelerin komşuluk araması boyutu belirlenmektedir (Şekil 4.4).

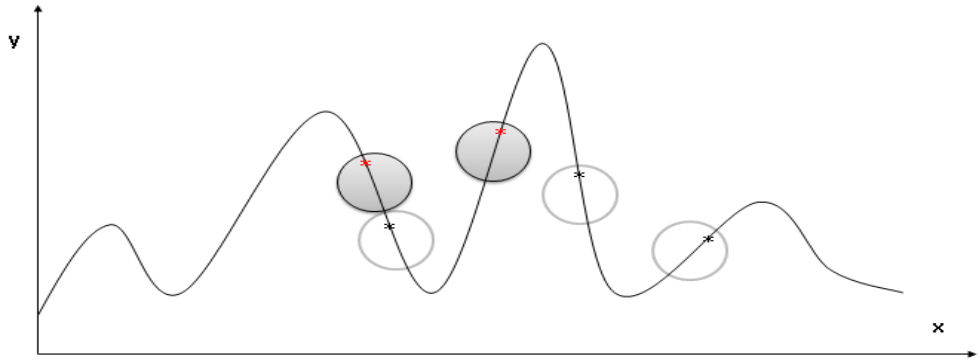


Şekil 4.4. En iyi sonuca sahip arıların seçilmesi

5. ve 6. adım: Seçilen bölgelerde komşuluk araştırması (bölge içinde daha uygun noktaların araştırılması) için, daha iyi çözümleri temsil eden en iyi bölgeye seçilen diğer bölgelere göre daha fazla araştırmacı arı, diğer bölgelere daha az araştırmacı arı gönderilerek, detaylı arama yapılmaktadır (Şekil 4.5). Yeni nüfus için her bölgedeki en uygun değere sahip arı seçilmektedir. Bölgelerdeki diğer arılar araştırma uzayından ayrılmaktadır (Şekil 4.6).

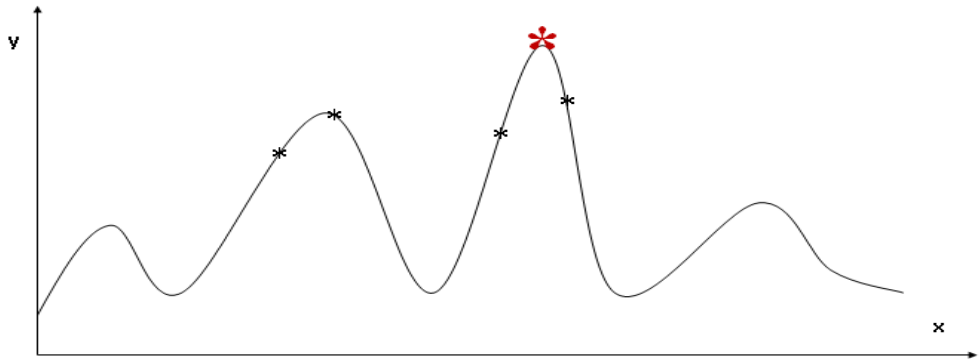


Şekil 4.5. Komşuluklara araştırmacı arıların gönderilmesi ve kendi için en iyilerin seçilmesi



Şekil 4.6. En iyi arılar tekrar belirlenir

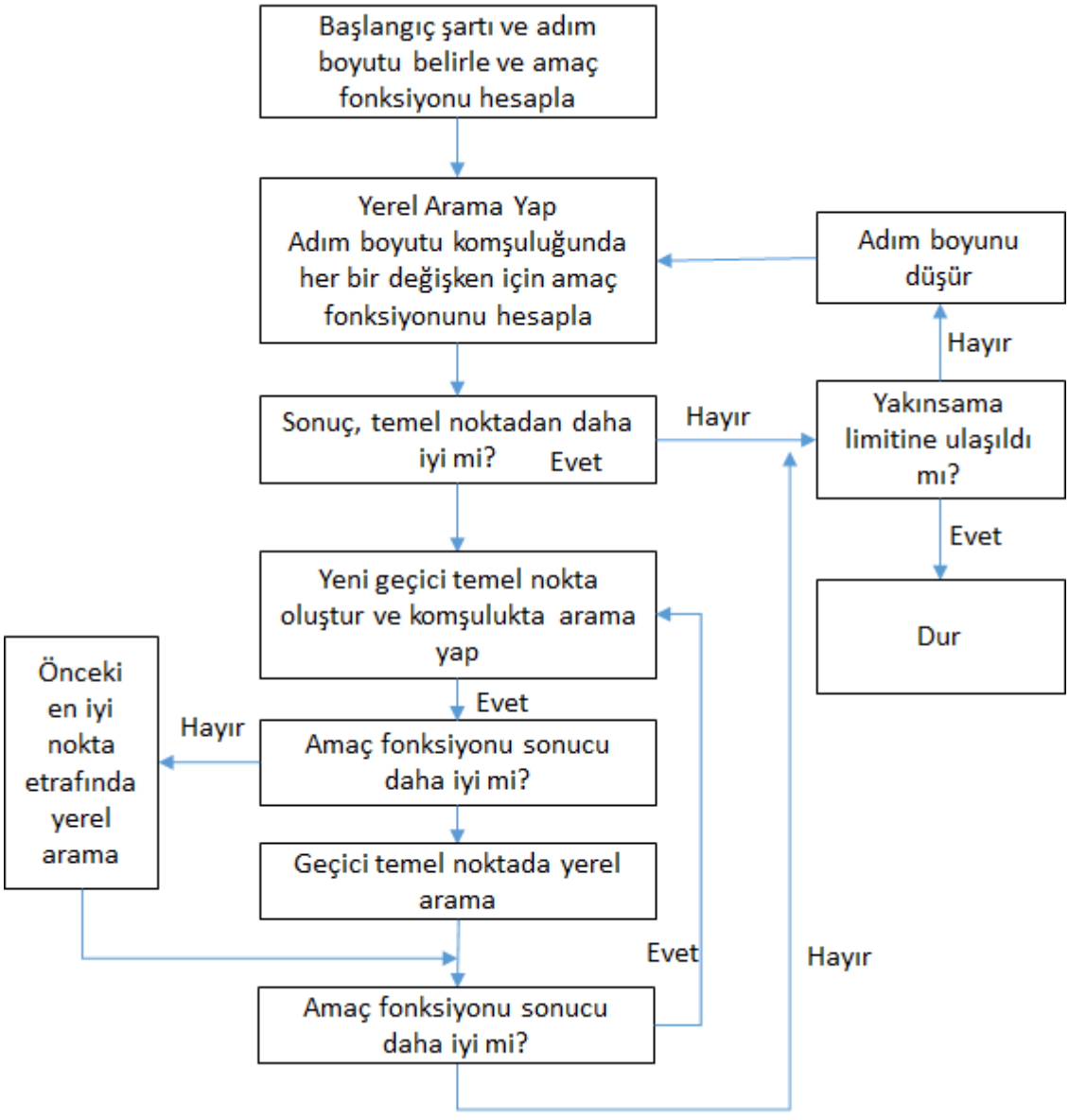
8.adım: Popülasyondaki diğer arılar yeni potansiyel çözümler elde etmek için tekrar rastgele olarak araştırma uzayına yayılmaktadırlar. Her bir iterasyonun sonunda yeni nüfus, seçilen her bir bölgenin en iyileri ve rastgele arama yapan kâşif arıları olmak üzere iki parçadan oluşacaktır. Optimizasyon durdurma kriteri sağlanana kadar devam ettirilmektedir (Şekil 4.7).



Şekil 4.7. Arı Algoritması optimizasyonunun tamamlanması

4.2. Hooke-Jeeves Algoritması

Hooke-Jeeves algoritması, Hooke ve Jeeves (1961) tarafından önerilmiş olan doğrudan arama metodunu temel alan çok değişkenli bir yerel arama algoritmasıdır. Bu algoritma, Newton-Raphson metodunun ihtiyaç duyduğu gibi amaç fonksiyonun türevine ihtiyaç duymamaktadır. Hooke-Jeeves algoritmasının akış şeması Şekil 4.8'deki gibidir.



Şekil 4.8. Hooke-Jeeves Algoritması akış şeması

Algoritma aşağıdaki gibi işlemektedir.

1. Bir başlangıç şartı belirlenir ve bu başlangıç şartına ait değişken dizisine ait amaç fonksiyonu hesaplanır.

2. Her bir değişken için + ve – yönlerdeki komşuluk adımında yeni değişken dizisi üretilir ve amaç fonksiyonu hesaplanır.
3. Eğer adım komşuluğundaki dizinin amaç fonksiyonunda düşüş yok ise, adım küçültülür ve önceki en iyi nokta etrafında arama tekrarlanır.
4. Eğer adım komşuluğundaki dizinin amaç fonksiyonunda düşüş var ise, burada yeni bir geçici temel nokta oluşturularak komşuluk araştırması her iki yönde düşüş olmayana kadar devam ettirilir.
5. Eğer geçici temel nokta daha düşük fonksiyon sonucu vermiyor ise önceki en iyi nokta etrafında bir arama yapılır.
6. Yakınsama kriteri gerçekleşene kadar devam et.

4.3. Arı Algoritması ile Hooke-Jeeves Algoritmasının Kombinasyonu

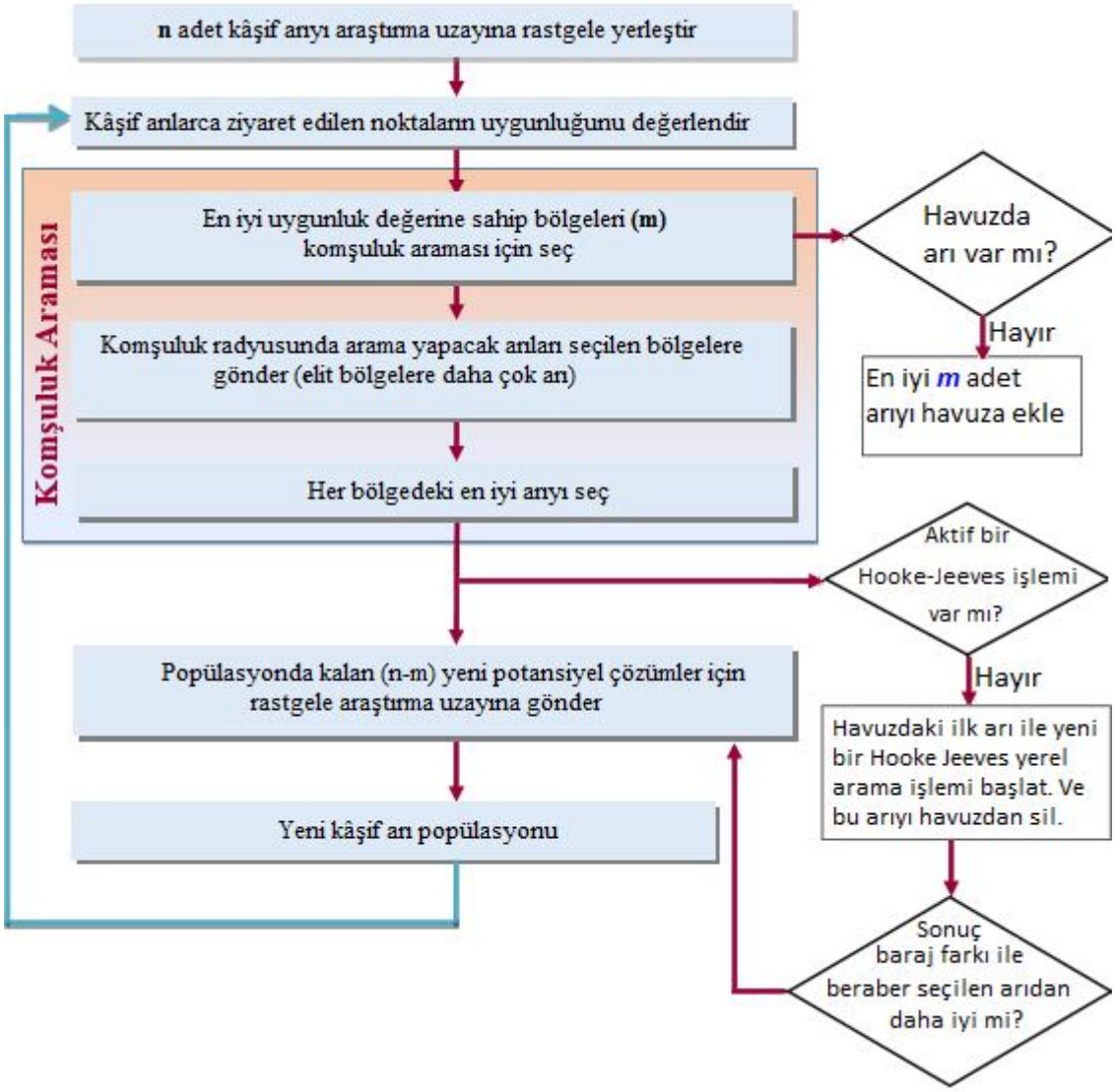
Kaynak araştırması bölümünde de bahsedildiği gibi, global arama algoritmalarından Genetik algoritması ile yerel arama algoritması olan Hooke-Jeeves algoritmasının kombinasyonunda performansın arttığı görülmüştür. Aynı şekilde Hooke-Jeeves algoritması Metropolis algoritmasında da performansı artırdığı görülmüştür. Daha önce global arama algoritmalarından Arı Algoritması ile bir yerel arama algoritmasının kombinasyonu çalışması yapılmamıştır. Bu çalışma ile ilk defa Arı Algoritmasının Hooke-Jeeves Algoritması ile kombinasyonu yapılmıştır ve Arı algoritmasının optimizasyon süresini kısalttığı görülmüştür.

Arı algoritması *m* adet en iyi arıları seçtiğinde, bu en iyi arıların bir listesi bir havuza eklenmektedir. Eş zamanlı olarak Hooke-Jeeves algoritması bu havuzdan bir arıyı seçmekte ve sonra onu havuzdan onu silmektedir. Daha sonra Hooke-Jeeves, bu arı etrafında bir yerel arama yapmaktadır. Bu yapılırken Arı Algoritması, Hooke-Jeeves'i beklememekte, o da kendi içinde çalışmaya devam eder. Arı Algoritması 6. adımda rastgele yeni arılar oluşturmaktadır. Eğer bu 6. adımda Hooke-Jeeves aramasını tamamlamış ise, Hooke-Jeeves'in sonucuna bakmaktadır. Eğer bu sonuç Hooke-Jeeves'in havuzdan seçtiği arının hatasından bir miktar (baraj değeri kadar) daha küçük ise Hooke-Jeeves'in elde ettiği sonuç, Arı algoritmasının 6. adımında ürettiği rastgele arılardan birisi ile değiştirilmektedir. Bu sonucun bir baraj değerinden daha küçük olması gerekmektedir. Örneğin Hooke-Jeeves'in başladığı hata değeri 40 olsun, eğer Hooke-Jeeves bu hatayı 39.9'a düşürmüş ise ve barajımız da 10 ise hata 40-10 = 30'un altına inmediği için bu sonuç iyi kabul edilmemektedir. Baraj değerinin çok büyük

belirlenmesi Hooke-Jeeves'in Arı algoritmasına hiç yardım edememesine sebep olmaktadır. Baraj değeri çok küçük olursa, bu durumda Arı algoritmasının en iyi değerlerinin hepsi aynı olmakta ve global arama özelliği kaybolmaktadır.

Baraj değerinin optimizasyon başlarında yüksek, optimizasyonun ilerleyen zamanlarında hatanın düşme hızının azaldığı durumlara ise baraj değerinin düşük olmasının daha uygun olduğu da görülmüştür.

Arı Algoritması ile Hooke-Jeeves algoritmasının kombinasyonunu gösteren şema Şekil 4.9'daki gibidir.



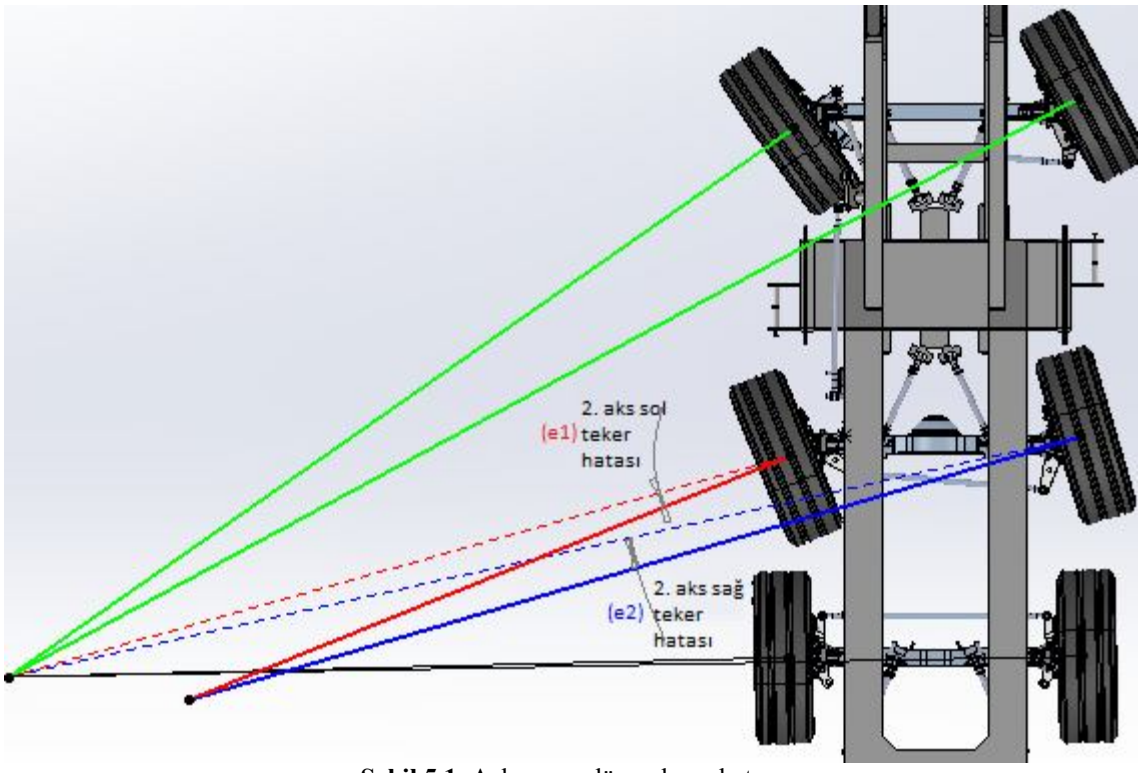
Şekil 4.9. Arı Algoritması ile Hooke-Jeeves Algoritmasının kombinasyonu şeması

Şekil 4.9'da şematik olarak gösterilen kombinasyonun kaynak kodu ekler bölümünde verilmiştir.

5. BİR AĞIR VASITAYA AİT ÇOK AKSLI DİREKSİYON MEKANİZMASININ OPTİMİZASYON PARAMETRELERİ

5.1. Problem Tanımı

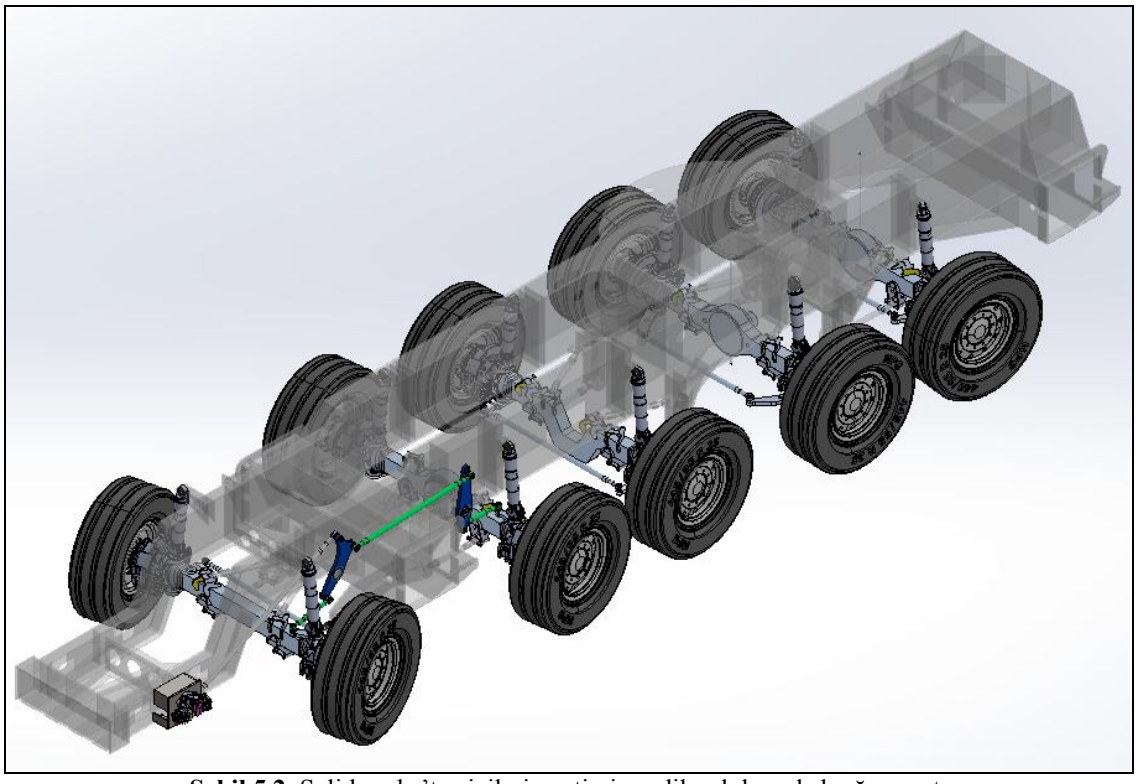
Ackerman dümenleme hatası, hedeflenen tekerlek doğrultusu ile gerçekleşen tekerlek doğrultusu arasındaki açıdır. Genellikle bu hatanın 1° 'nin altında olması istenir. Şekil 3.1'de verilen ağır vasıtaya ait direksiyon mekanizmasının Ackerman dümenleme hatası açıları Şekil 5.1'de gösterilmiştir.



Şekil 5.1. Ackerman dümenleme hatası

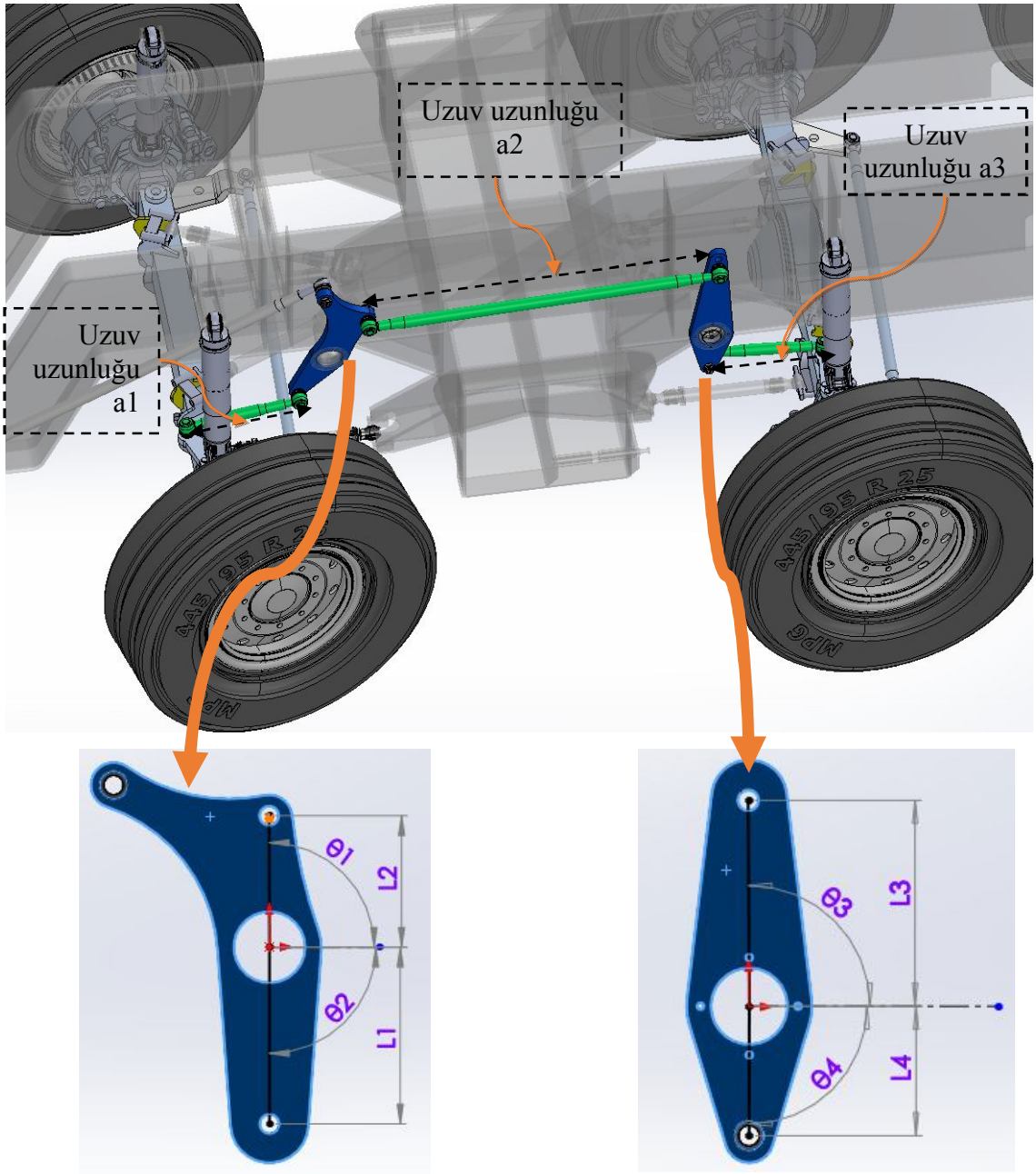
5.2. Tasarım Parametreleri

Şekil 3.1'de verilen ağır vasıtaya ait katı model Şekil 5.2'de gösterilmiştir.



Şekil 5.2. Solidworks'te çizilmiş optimize edilecek beş akslı ağır vasıta

Çok akslı ağır vasıtalar, tüm teker dümenleme ve yengeç dümenleme hareket tiplerine ihtiyaç duymaktadır. Bu iki hareket tipini birden sağlayabilmesi için vasıtanın üçüncü, dördüncü ve beşinci akslarının birinci ve ikinci akslardan mekanik olarak bağımsız olması gerekmektedir. Bu durumda sadece birinci ve ikinci akslar mekanik olarak bağlantılıdırlar. Fakat üçüncü, dördüncü ve beşinci akslar elektro hidrolik eyleyiciler tarafından tahrik edilmektedirler. Bu sentez çalışmasında mekanik olarak bağımlı olan birinci ve ikinci aks arasındaki direksiyon mekanizması optimize edilecektir.



Şekil 5.3. Değişkenler

Şekil 5.3'te görüldüğü üzere CAD parametreleri şunlardır: a_1, a_2, a_3, L_1, L_2, L_3, L_4, θ_1, θ_2, θ_3 ve θ_4. Bu parametrelerin başlangıç şartları da Çizelge 5.1'deki gibidir.

Çizelge 5.1. Optimizasyon değişkenlerinin başlangıç şartları

Parametreler	Başlangıç Şartları
a_1 [mm]	787.80
L_1 [mm]	250.60
θ_1 [°]	89.00
L_2 [mm]	294.38
θ_2 [°]	89.62

a_2 [mm]	1457.36
L_3 [mm]	415.31
θ_3 [°]	90.00
L_4 [mm]	200.00
θ_4 [°]	89.78
a_3 [mm]	487.19

5.3. Cezalandırma Fonksiyonları

Optimizasyon sırasında bazı şartların gerçekleştiği durumlarda hata fonksiyonuna ekleme yapabilmek için ceza fonksiyonları kullanılmaktadır.

Örneğin, a_1 değişkenine ait sınır şartları şu şekildedir.

$$623 \le a_1 \le 823$$

Örnek olarak bu sınır şartlarının uyulmaması durumunda ceza katsayısı 1000 olarak alınsın. Bu durumda cezalandırma fonksiyonu:

$$g_1 = (a_1 < 623) * 1{,}000$$

Yukarıdaki ifadede parantez içindeki değer doğru ise 1 değerini üretmekte, bu durumda g_1 fonksiyonunun değeri 1,000 olmaktadır. Eğer parantez içindeki değer doğru değilse 0 değerini üretmekte ve bu durumda ise g_1 değeri 0 olmaktadır.

Tüm cezalandırma fonksiyonları Çizelge 5.2'deki gibidir.

Çizelge 5.2 Cezalandırma fonksiyonları

$g_1 = (a_1<623) * 1{,}000$	$g_{12} = (a_1>823) * 1{,}000$
$g_2 = (L_1<170) * 1{,}000$	$g_{13} = (L_1>370) * 1{,}000$
$g_3 = (\theta_1<84°) * 1{,}000$	$g_{14} = (\theta_1>96°) * 1{,}000$
$g_4 = (L_2<100) * 1{,}000$	$g_{15} = (L_2>300) * 1{,}000$
$g_5 = (\theta_2<84°) * 1{,}000$	$g_{16} = (\theta_2>96°) * 1{,}000$
$g_6 = (a_2<1290) * 1{,}000$	$g_{17} = (a_2>1490) * 1{,}000$
$g_7 = (L_3<200) * 1{,}000$	$g_{18} = (L_3>450) * 1{,}000$
$g_8 = (\theta_3<84°) * 1{,}000$	$g_{19} = (\theta_3>96°) * 1{,}000$
$g_9 = (L_4<91) * 1{,}000$	$g_{20} = (L_4>291) * 1{,}000$
$g_{10} = (\theta_4<84°) * 1{,}000$	$g_{21} = (\theta_4>96°) * 1{,}000$
$g_{11} = (a_3<390) * 1{,}000$	$g_{22} = (a_3>590) * 1{,}000$

Yukarıda değişkenlerin sınır şartları cezalandırma fonksiyonu olarak tanımlanmıştır. Cezalandırma fonksiyonlarına sadece sınır şartları değil başka şartlar da girilebilmektedir. Örneğin Şekil 5.3'te görülen a_2 uzvunun burkulmaya karşı kontrolünü

de yapacak olunsaydı. Kritik burkulma formülü ile a_2 uzvuna gelen kuvveti kıyaslamak için Denklem 5.1 ve Denklem 5.2'deki gibi tanımlamanın yapılması gerekirdi.

$$F_{kritik} = \frac{\pi^2 EI}{L^2} = \frac{\pi^2 EI}{{a_2}^2} \tag{5.1}$$

$$g_{23} = \left(F_{a_2} \geq F_{kritik}\right) * 1{,}000 \tag{5.2}$$

Eğer a_2 uzvundaki kuvvet (F_{a_2}) , kritik burkulma kuvvetinden (F_{kritik}) büyük veya eşit olursa g_{23} değeri 1,000 olacaktır diğer durumda ise 0 olacaktır. g_{23} fonksiyonu, cezalandırma fonksiyonlarına örnek olması açısından verilmiştir, bu optimizasyonda sadece Ackerman geometrisi kontrol edileceği için g_{23} fonksiyonu kullanılmayacaktır.

5.4. Hata Fonksiyonu Tanımı ve Ağırlıklar

Ackerman dümenleme hatası geometrisi Şekil 5.1'deki gibi tanımlanmıştır. Bu mekanizmada dümenleme hatasının 1° 'nin altında olması istenmektedir. Yaklaşık her 5° 'de bir hata fonksiyonu hesaplanacak ve elde edilen hataların karelerinin toplamının karekökü alınarak tek bir hata elde edilecektir. Denklem 5.3'te e_1 ikinci aksın sol tekerine ait Ackerman dümenleme hatası olmak üzere ve e_2 de ikinci aksın sağ tekerine ait Ackerman dümenleme hatası olmak üzere hata fonksiyonumuzun şu şekildedir.

$$e_a = \sum_{i=1}^{p} \sqrt{\left({e_1}^2 + {e_2}^2\right)_i} + \sum_{j=1}^{m} g_j \tag{5.3}$$

Bazı hata bölgelerinin önemli olduğu durumlarda, ağırlık çarpanlarının kullanılması gerekmektedir. Örneğin, Ackerman dümenleme hatasının hareket esnasında her durumda 1° 'nin altında olması istendiği için, burada 1° 'nin üzerindeki hatalar önem arz etmektedir. Eğer hata fonksiyonu Denklem 5.4 ve Denklem 5.5'teki gibi değiştirilirse, bu hata fonksiyonu 1° 'nin üzerindeki hataları azaltmaya daha fazla odaklanacaktır.

$$e_1 = \begin{cases} 1+10(e_1-1), & e_1 > 1 \\ e_1 & e_1 \le 1 \end{cases} \tag{5.4}$$

$$e_2 = \begin{cases} 1+10(e_2-1), & e_2 > 1 \\ e_2 & e_2 \le 1 \end{cases} \tag{5.5}$$

Denklem 5.4 ve Denklem 5.5'teki e_1 ve e_2 fonksiyonlarına göre; eğer hata 1° ve 1° 'nin altında ise hatayı aynen almakta, eğer hata 1° 'nin üstünde ise 1° 'in üzerindeki değerin 10 katı ile ağırlıklı toplayıp almaktadır. Bu, hata fonksiyonun 1° 'nin üzerine odaklanmasına yardımcı olmaktadır. Fakat bu yine de yeterli değildir. 0° 'ye yakın hatalar ile 1° 'nin çok az üzerindeki hataların gerçekleşmesi durumunda bu hata, bazı durumlarda 1° 'nin çok az üzerinde optimizasyonu sonlandırmaktadır. Bu yüzden 1° 'nin üzerindeki az bir hataya dahi hata fonksiyonunu odaklayabilmek için 1° 'nin biraz altından itibaren hata fonksiyonuna ağırlık vermek gerekmektedir. Bu bir nevi tampon bölge oluşturmak gibidir.

$$e_1 = \begin{cases} 1+10(e_1-0.80), & e_1 > 1 \\ e_1 & e_1 \le 1 \end{cases} \tag{5.6}$$

$$e_2 = \begin{cases} 1+10(e_2-0.80), & e_2 > 1 \\ e_2 & e_2 \le 1 \end{cases} \tag{5.7}$$

Denklem 5.6 ve Denklem 5.7'de, eğer hata 1° ve 1° 'nin altında ise hata aynen alınmakta fakat hata 1° 'nin üstünde ise 1+(1° 'nin üzerindeki hata + 0.20°)'nin 10 katı ile ağırlıklı olarak toplayıp almaktadır. Bu, hata fonksiyonunun 1° 'nin üzerine 0.20° ek ile odaklanmasını sağlamaktadır. Bu sayede hatanın 1° 'nin biraz üstüne çıksa dahi bunu da 1° 'nin altına indirmeye çalışmaktadır.

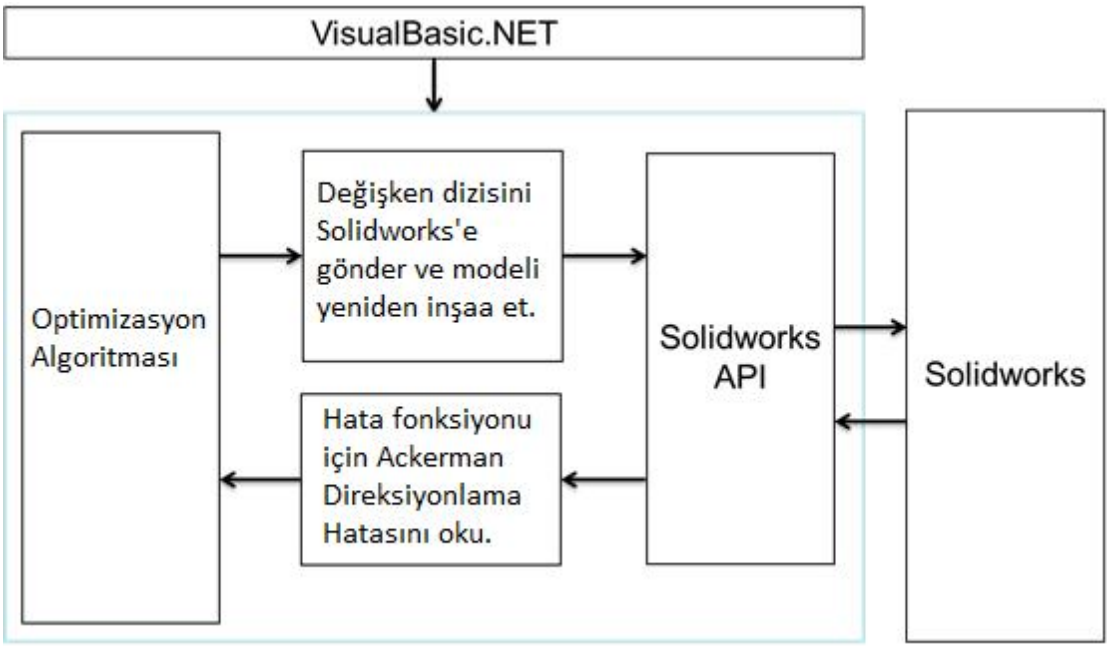
Hem Denklem 5.4 – Denklem 5.5 ile hem de Denklem 5.6 – Denklem 5.7 kullanılarak Denklem 5.3'teki hata fonksiyonu hesaplanıp kıyaslanacaktır.

Birinci aksın sol tekeri 35° sol ile 28.5° sağ aralığında çalışmaktadır. Toplam açı menzili 35 + 28.5 = 63.5° 'dir. 35° 'den -28.5° 'ye kadar 13 adım oluşturulmuştur. Her bir adım 63.5 / 12 = 5.29° 'dir. Hata fonksiyonundaki (Denklem 5.3) p değeri 13'tür. m_k ise cezalandırma fonksiyonlarının sayısıdır ve bu problemde 22 tane cezalandırma fonksiyonu olduğu için değeri 22'dir.

6. OPTİMİZASYON ALGORİTMASI İLE CAD PROGRAMINI KONTROL EDEN YAZILIM

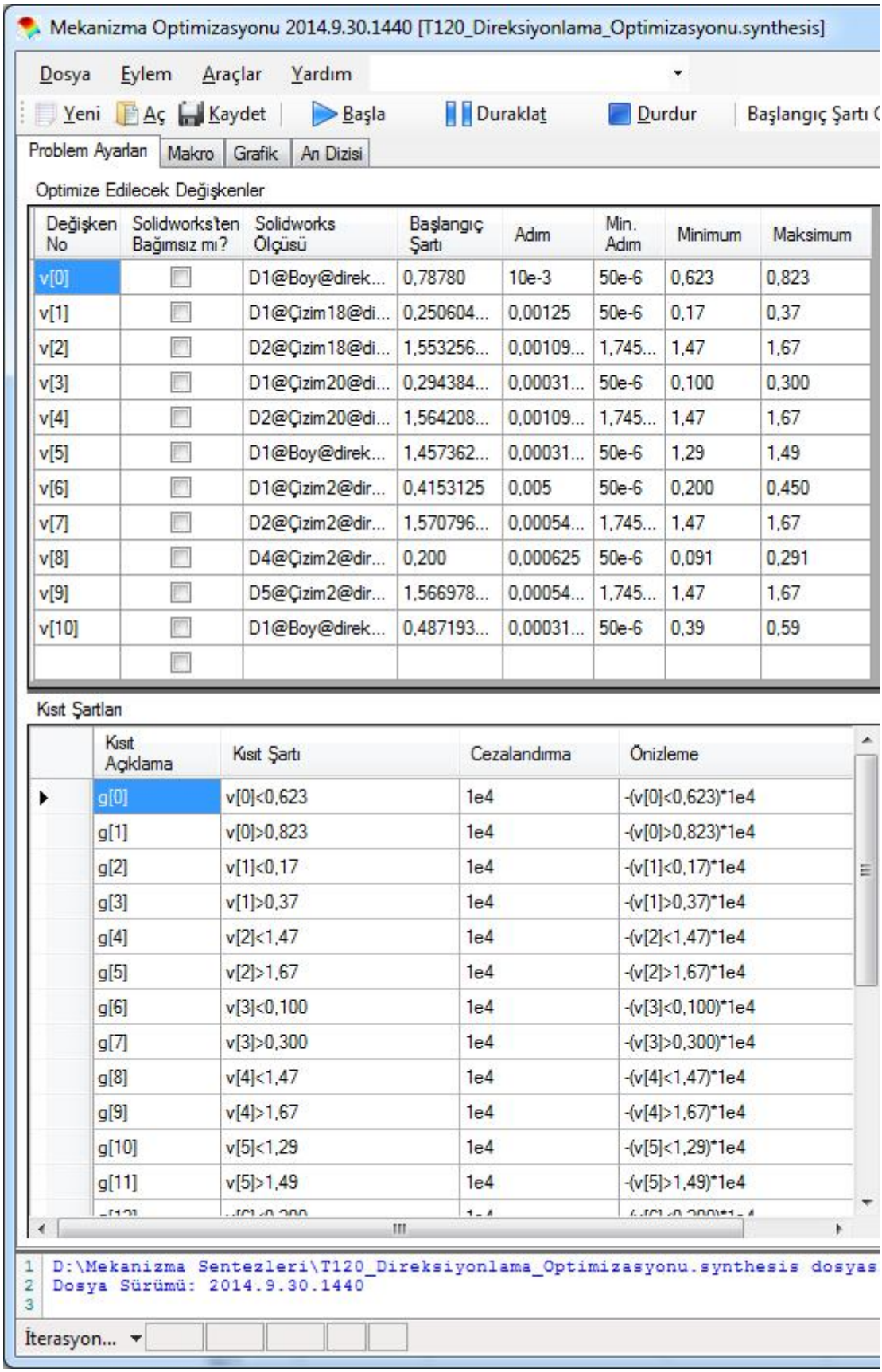
Arı algoritması ve Hooke-Jeeves algoritmasını kullanan bu yazılım VisualBasic.NET dilinde geliştirilmiş olup Solidworks CAD programı ile olan bağlantıyı Solidworks API (Uygulama Programlama Arayüzü) ile gerçekleştirmiştir. Program başlatıldığında Solidworks'te optimize edilecek CAD modeli kullanıcı tarafından açılmaktadır. Programa, optimize edilecek CAD modeli tanıtılmaktadır. Bu modeldeki hangi değişkenleri optimize edeceği tanıtılmaktadır. Her bir değişkenin Arı algoritmasında alabileceği aralık girilmektedir. Yine her bir değişkenin Hooke-Jeeves tarafından kullanılacak olan başlangıç adımı ve minimum adımı da girilmektedir. Kısıt şartları ve diğer cezalandırma fonksiyonları programa girilmektedir. Hata fonksiyonunu hesaplayabilmek için gerekli algoritma yine programdaki makro bölümüne VisualBasic dilinde girilmektedir. Daha sonra optimizasyonun Arı Algoritması ya da Hooke-Jeeves algoritmalarından hangisini kullanacağı seçilmektedir. Arı Algoritması kullanılacaksa; kâşif arı sayısı, en iyi arıların sayısı, en iyi arıların etrafına gönderilecek araştırmacı arı sayısı, en iyi arıların etrafında aranılacak komşuluk radyüsü girilmektedir. Eğer Arı Algoritmasına Hooke-Jeeves algoritmasının yardım etmesi isteniyorsa Arı Algoritması - Hooke Jeeves kombinasyonu aktifleştirilir. Bu durumda Arı algoritmasının kullandığı Solidworks modelini, Hooke-Jeeves algoritması kullanamamaktadır. Bu yüzden ikinci bir Solidworks uygulaması açılmakta ve aynı model bu Solidworks uygulamasında da açılmaktadır. Dördüncü bölümde Arı algoritması ile Hooke-Jeeves Algoritmasının kombinasyonu konusunda bahsedilen baraj değeri girilmektedir. Optimizasyona başla emri verilince, bu program Arı algoritmasını kullanarak optimizasyona başlamaktadır. Eğer Hooke-Jeeves algoritması ile kombinasyon aktif ise, bu program kendi kopyasından bir tane daha açmaktadır ve bu ikinci program da Hooke-Jeeves algoritmasını kullanarak ikinci Solidworks uygulamasındaki modeli kullanarak 1. uygulamadaki Arı Algoritmasına yardımcı olmaktadır.

Optimizasyon sırasında, program değişken dizileri üretmekte ve Solidwoks API aracılığıyla bu değişken dizilerini Solidworks'e göndermektedir. Dümenleme mekanizması soldan sağa dümenleme esnasında kontrol edilmektedir. Bu kontrollerde eğer mekanik bir hata yok ise Ackerman hataları okunmakta ve program bu Ackerman hatalarını kullanarak hata fonksiyonunu hesaplamakta ve tek bir hata elde etmektedir. Programın Solidworks ile arasındaki bağlantı Şekil 6.1'de verilmiştir.



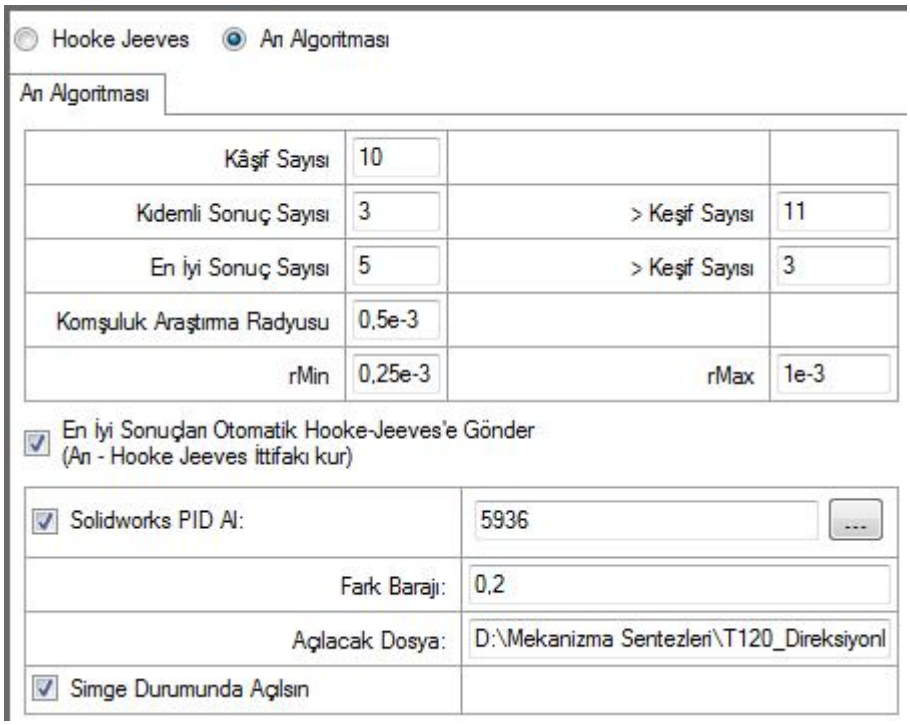
Şekil 6.1. Geliştirilen uygulama ile Solidworks arasındaki bağlantı

Geliştirilen program Şekil 6.2'deki gibi bir arayüze sahiptir. Optimize edilecek değişkenler bu ekranda girilmektedir. Sınır şartları da bu ekranda girilmektedir.



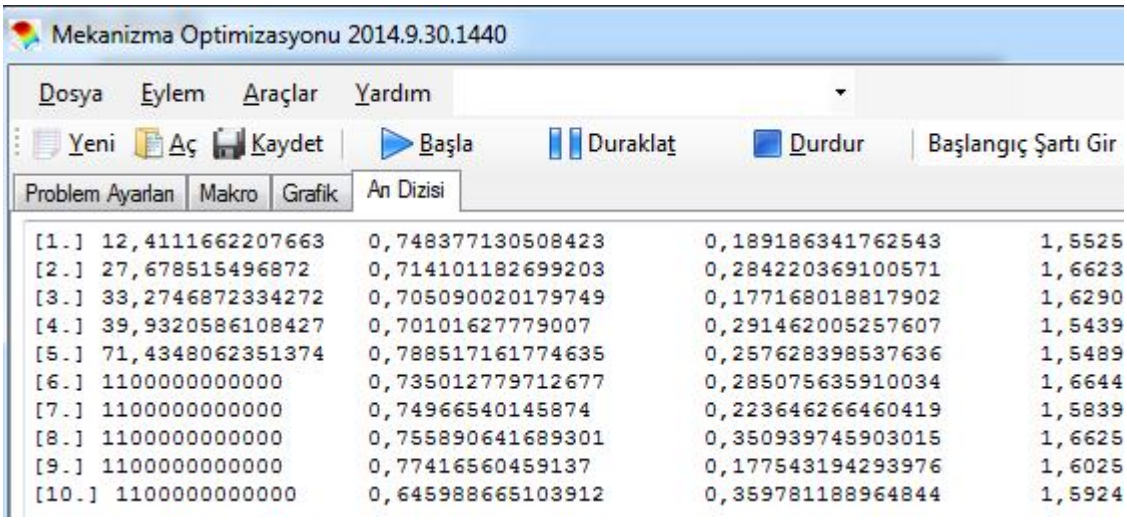
Şekil 6.2. Programın değişkenler ve başlangıç şartlarının girildiği arayüzü

Şekil 6.3'te hangi algoritmanın kullanılacağı seçilebilmektedir. Eğer Arı Algoritması seçilmişse, kullanıcı Hooke-Jeeves ile kombinasyon kurmasını isteyebilmektedir.

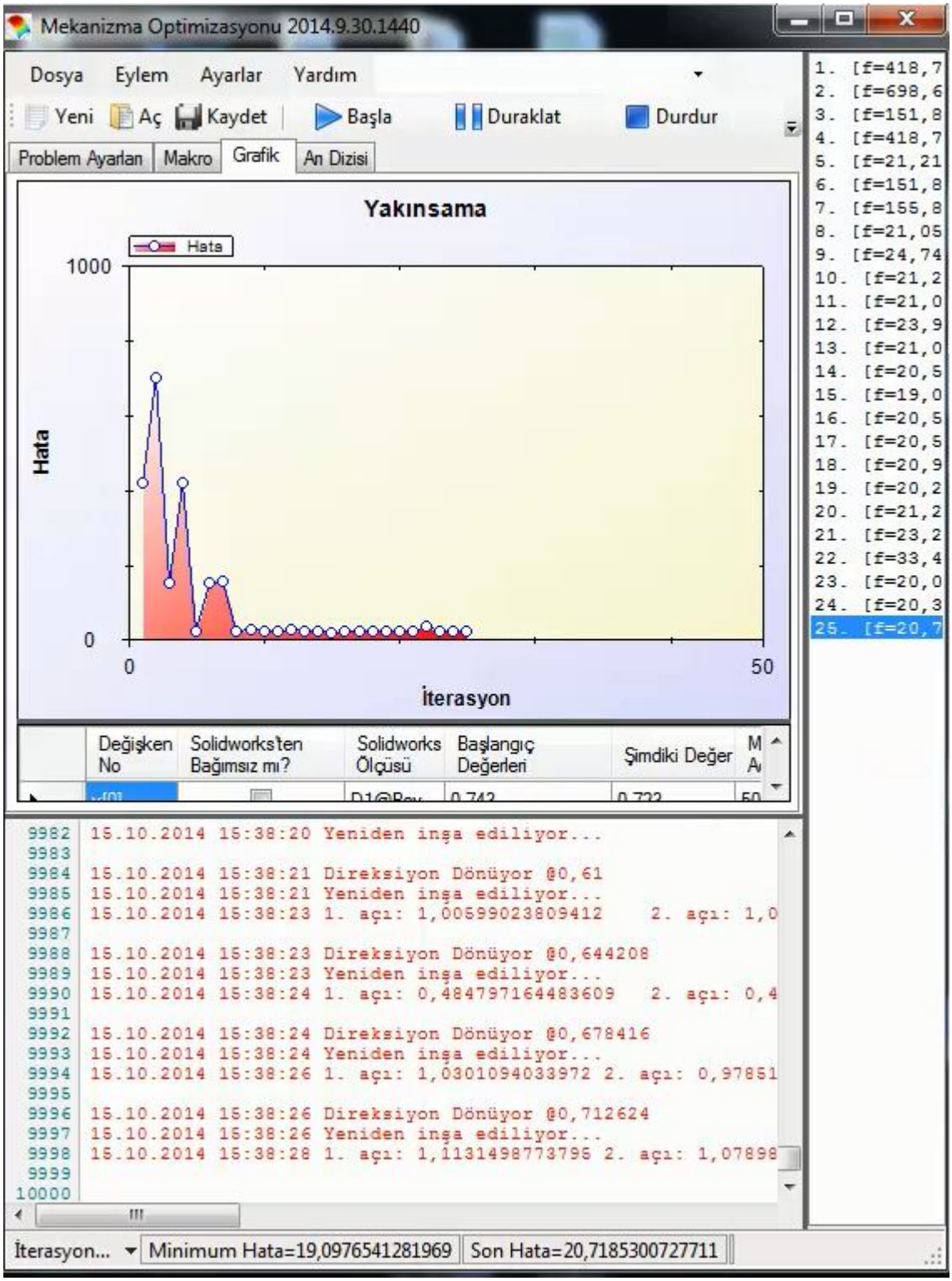


Şekil 6.3. Programın kullanacağı optimizasyon algoritmasının seçildiği ekran görüntüsü

İterasyonlar devam ederken, kullanıcı Arı Algoritmasının değişken dizilerini Şekil 6.4'teki gibi görebilmektedir.

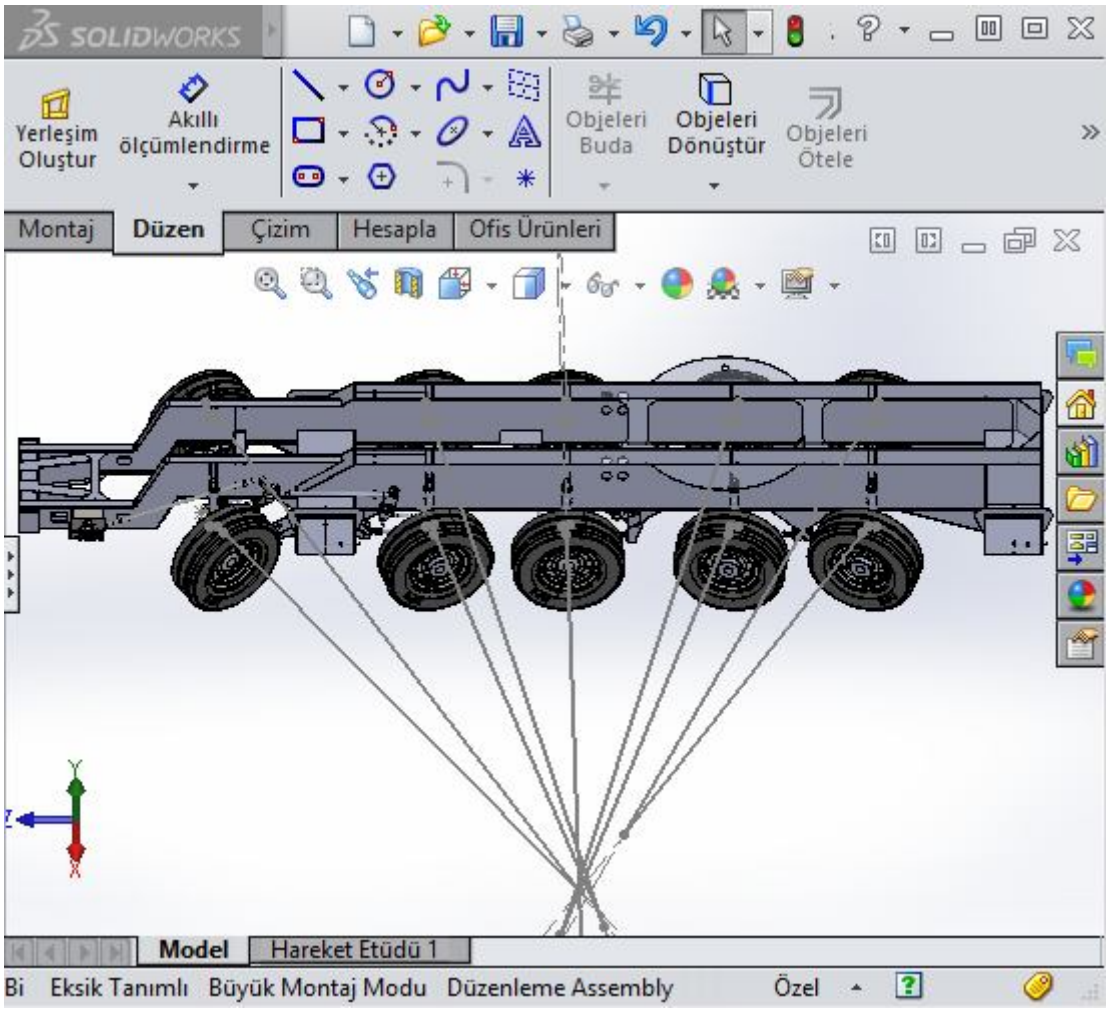


Şekil 6.4. Arı Algoritmasının değişken dizileri anlık olarak görülebilmektedir.



Şekil 6.5. Optimizasyon devam ederken uygulamanın hata takip ekranından bir görüntü

Şekil 6.5'te geliştirilen yazılımın hata takip ekranı görülmektedir. Bu ekrandan amaç fonksiyonunun yakınsayıp yakınsamadığı takip edilebilmektedir.



Şekil 6.6. Optimizasyon esnasında Solidworks'te güncel konfigürasyonun görüntüsü

Şekil 6.6'da optimizasyon anında güncel konfigürasyon kullanıcı tarafından görülebilmektedir.

7. SAYISAL UYGULAMALAR

MPG Makina Prodüksiyon Grubu Makine İmalat San. ve Tic. A.Ş. firmasının T0120 vasıtasının direksiyon mekanizmasının Ackerman hatasının azaltılması gereksiniminden dolayı, hali hazırda çizilmiş olan T0120 katı modeli, geliştirilen optimizasyon programına dümenleme sınırlarını içerisinde Ackerman hatasını okuyabilecek şekilde tanıtılmıştır. Geliştirilen yazılıma optimizasyon başlatılmış ve hata fonksiyonunun optimizasyonu sonuçları paylaşılmıştır. En uygun sonuca ait değişken dizisine karşılık gelen katı model üretilmiş ve Ackerman hatasının düştüğü fiziksel olarak da gözlenmiştir.

Bu tez çalışmasında yapılan sayısal uygulama için kullanılan bilgisayar ve yazılımların özellikleri aşağıdaki gibidir:

CPU: Intel Xeon CPU E3-1270- v3 @ 3.50 GHz,
RAM: 16 GB memory,
İşletim Sistemi: Microsoft Windows 7 Professional,
Programlama Arayüzü: Microsoft Visual Studio 2008,
CAD Programı: Solidwork 2014 SP4.

Denklem 5.4 – Denklem 5.5 ve Denklem 5.6 – Denklem 5.7'ye göre elde edilen Denklem 5.3'teki iki ayrı hata fonksiyonu ayrı ayrı optimize edilmiştir. Denklem 5.4 – Denklem 5.5'e göre hata fonksiyonu eğer hata 1° 'nin üstüne çıkarsa 1° 'nin üzerini 10 ile çarpıp ağırlık vererek 1 ile topluyordu. Denklem 5.6 – Denklem 5.7'ye göre hata fonksiyonu ise eğer hata 1° 'nin üstüne çıkarsa 1° 'nin üzerindeki hatayı 0.20 (tampon bölge) ile toplayıp daha sonra 10 ile çarpıp ağırlık vererek sonra da 1 ile topluyordu. Denklem 5.4 – Denklem 5.5 ve Denklem 5.6 – Denklem 5.7'deki hatayı optimize edip kıyaslanacaktır. Her bir hata fonksiyonu hem Arı Algoritması (BA) ve hem de Arı Algoritması ile Hooke-Jeeves Algoritması Kombinasyonu (BAHJ) ile optimize edilecektir.

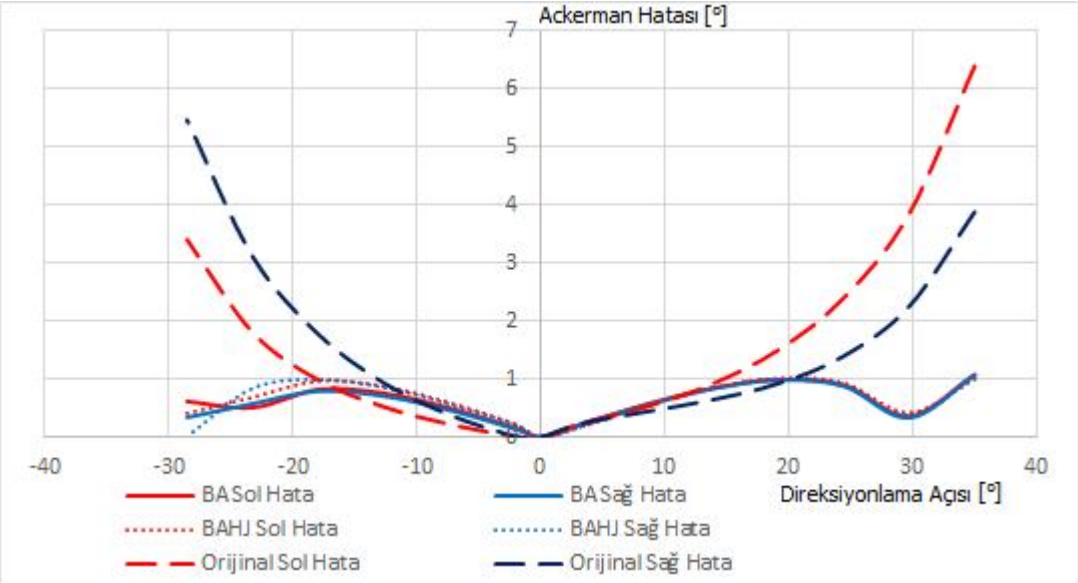
7.1. Denklem 5.4 – Denklem 5.5'e Göre Hata Fonksiyonunun Optimizasyonu

Çizelge 7.1'de Denklem 5.4 – Denklem 5.5'e göre elde edilmiş optimizasyon sonuçları görülmektedir.

Çizelge 7.1. Denklem 5.4 – Denklem 5.5'e göre optimizasyon sonuçları

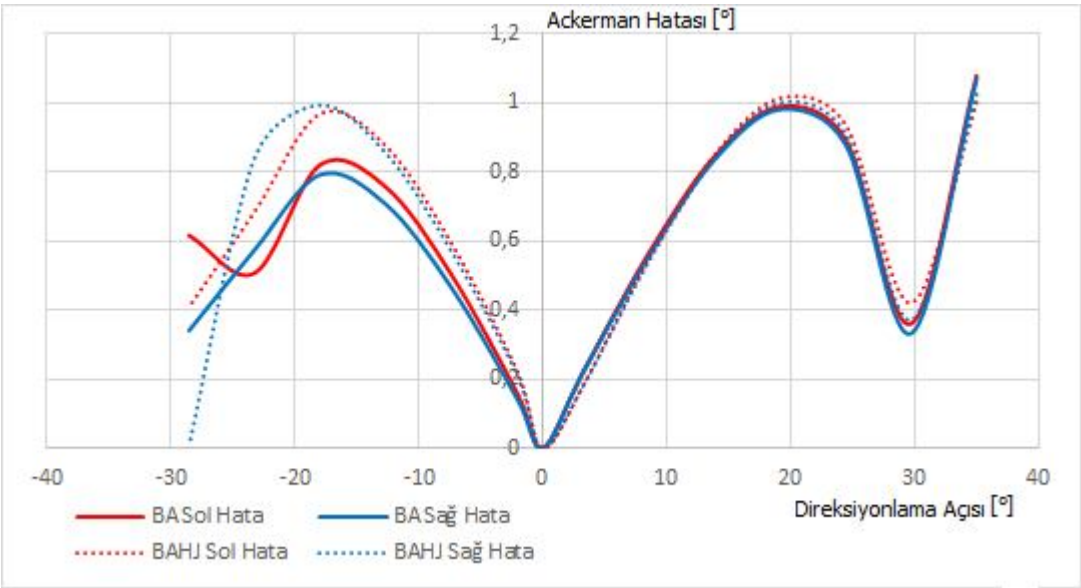
	Orijinal	**BA**	**BAHJ**
Hata	204.86	12.12	12.41
Sol Teker Hatası (Ort.) [°]	1.75	**0.64**	0.67
Sağ Teker Hatası (Ort.) [°]	1.65	**0.61**	0.64
Maksimum Sol Teker Hatası [°]	6.39	1.08	**1.01**
Maksimum Sağ Teker Hatası [°]	5.46	1.07	**1.02**
Optimizasyon Süresi [dk]	-	443	**109**
***a1* [mm]**	787.80	787.97	787.49
***L1* [mm]**	250.60	264.78	268.03
***θ1* [°]**	89.00	89.04	90.65
***L2* [mm]**	294.38	275.76	272.95
***θ2* [°]**	89.62	89.41	90.58
***a2* [mm]**	1457.36	1457.00	1457.58
***L3* [mm]**	415.31	421.78	445.95
***θ3* [°]**	90.00	90.12	89.76
***L4* [mm]**	200	181.46	196.09
***θ4* [°]**	89.78	89.70	89.84
***a3* [mm]**	487.19	493.51	486.69

Ackerman hatasının orijinal hali ve Denklem 5.4 – Denklem 5.5'e göre Arı Algoritmasının (BA) ve Arı Algoritması ile Hooke-Jeeves Algoritmasının Kombinasyonunun (BAHJ) elde ettiği sonuç Şekil 7.1'deki gibidir.



Şekil 7.1. Denklem 5.4 – Denklem 5.5'e göre orijinal hali, BA ve BAHJ sonuçları

Sadece BA ve BAHJ algoritmalarının sonuçları Şekil 7.2'deki gibidir.



Şekil 7.2. Denklem 5.4 – Denklem 5.5'e göre BA ve BAHJ algoritmalarının elde ettiği sonuç

Şekil 7.2'deki grafikte ve Çizelge 7.1'deki değerlerde görüldüğü gibi BA ve BAHJ algoritmaları Denklem 5.4 – Denklem 5.5'e göre Ackerman Hatasını 1° 'nin tamamen altına indirememiştir. Çünkü 1° 'nin çok az üzerindeki Ackerman hatasının 1° 'nin üzerinde kalan kısmının 10 ile çarpılması aslında çok büyük bir hata üretememektedir. Örneğin 1.05° Ackerman hatası Denklem 5.4 – Denklem 5.5'e göre;

$$e_1 = 1 + (1.05 - 1) * 10$$
$$e_1 = 1.5$$

Yani 1.05° 'lik Ackerman hatası 1.5° 'ye yükselmektedir. Artış ise sadece 0.45° 'dir. Aynı hesaplama Denklem 5.6 – Denklem 5.7'ye göre yapıldığında.

$$e_1 = 1 + (1.05 - 0.80) * 10$$
$$e_1 = 3.5$$

Yani 1.05° 'lik Ackerman hatası 3.5° 'ye yükselmektedir. Artış bu sefer 2.45° olmuştur. Bu 0.20° 'lik tampon bölge sayesinde olmuştur. Ve 1° 'nin üzerindeki Ackerman hatalarına büyük bir etki ile doğrusal hata artışı sağlanmış olmaktadır.

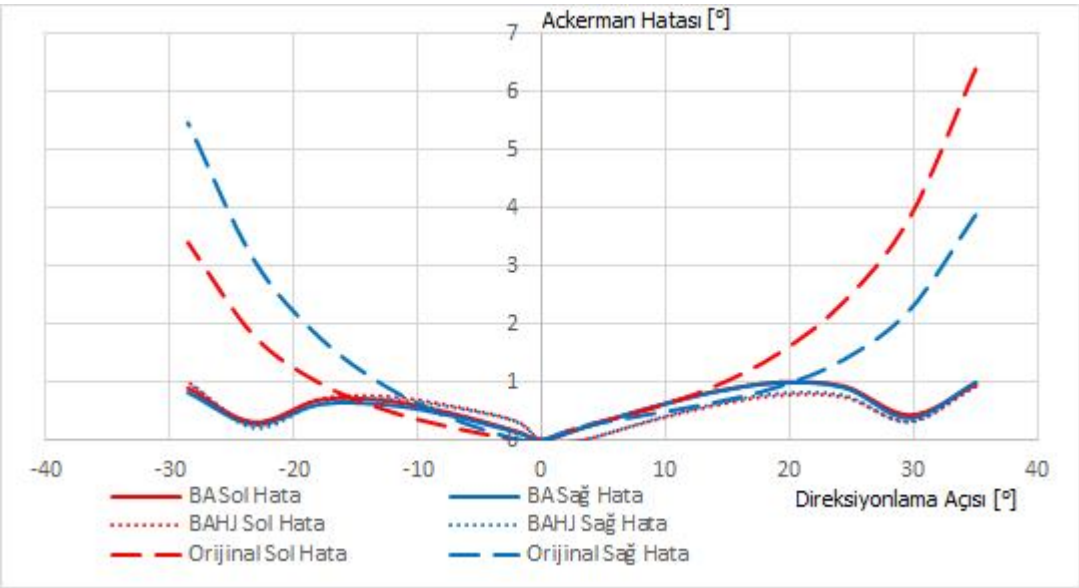
7.2. Denklem 5.6 – Denklem 5.7'e Göre Hata Fonksiyonunun Optimizasyonu

Çizelge 7.2'de Denklem 5.6 – Denklem 5.7'ye göre elde edilmiş optimizasyon sonuçları görülmektedir.

Çizelge 7.2. Denklem 5.6 – Denklem 5.7'ye göre optimizasyon sonuçları

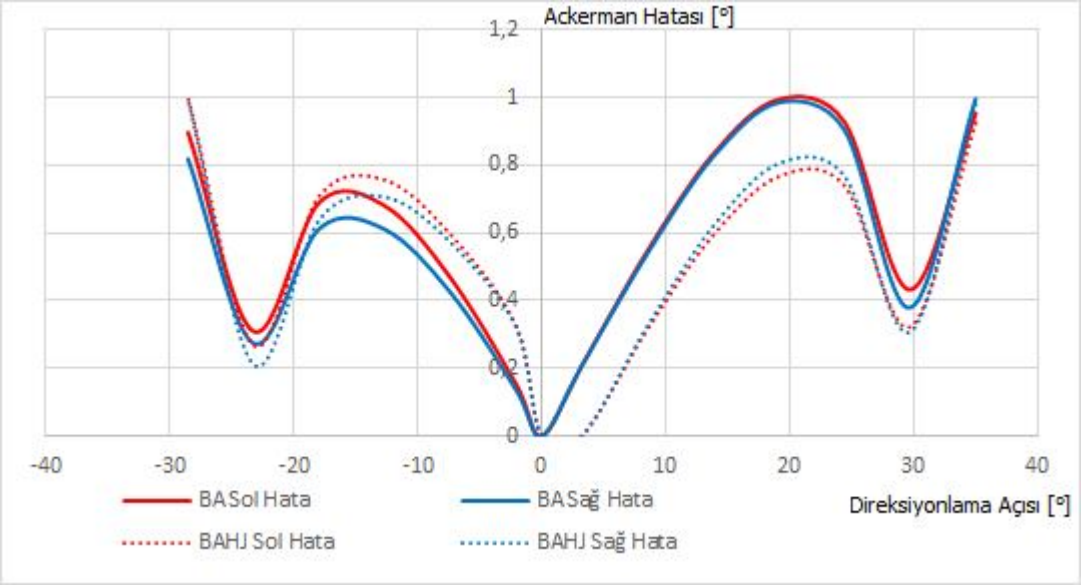
	Orijinal	**BA**	**BAHJ**
Hata	222.14	11.18	10.30
Sol Teker Hatası (Ort.) [°]	1.75	0.62	**0.56**
Sağ Teker Hatası (Ort.) [°]	1.65	0.59	**0.56**
Maksimum Sol Teker Hatası [°]	6.39	**0.99**	1.00
Maksimum Sağ Teker Hatası [°]	5.46	1.00	**0.99**
Optimizasyon Süresi [dk]	-	362	**245**
***a1* [mm]**	787.80	781.33	791.68
***L1* [mm]**	250.60	352.56	310.64
***θ1* [°]**	89.00	88.83	92.57
***L2* [mm]**	294.38	230.61	237.22
***θ2* [°]**	89.62	93.56	84.40
***a2* [mm]**	1457.36	1466.52	1332.20
***L3* [mm]**	415.31	274.60	404.28
***θ3* [°]**	90.00	94.92	92.90
***L4* [mm]**	200	189.08	234.09
***θ4* [°]**	89.78	92.10	91.59
***a3* [mm]**	487.19	529.43	445.55

Ackerman hatasının orijinal hali ve Denklem 5.6 – Denklem 5.7'ye göre Arı Algoritmasının (BA) ve Arı Algoritması ile Hooke-Jeeves Algoritmasının Kombinasyonunun (BAHJ) elde ettiği sonuç Şekil 7.3'teki gibidir.



Şekil 7.3. Denklem 5.6 – Denklem 5.7'ye göre orijinal hali, BA ve BAHJ sonuçları

Sadece BA ve BAHJ algoritmalarının sonuçları Şekil 7.4'teki gibidir.

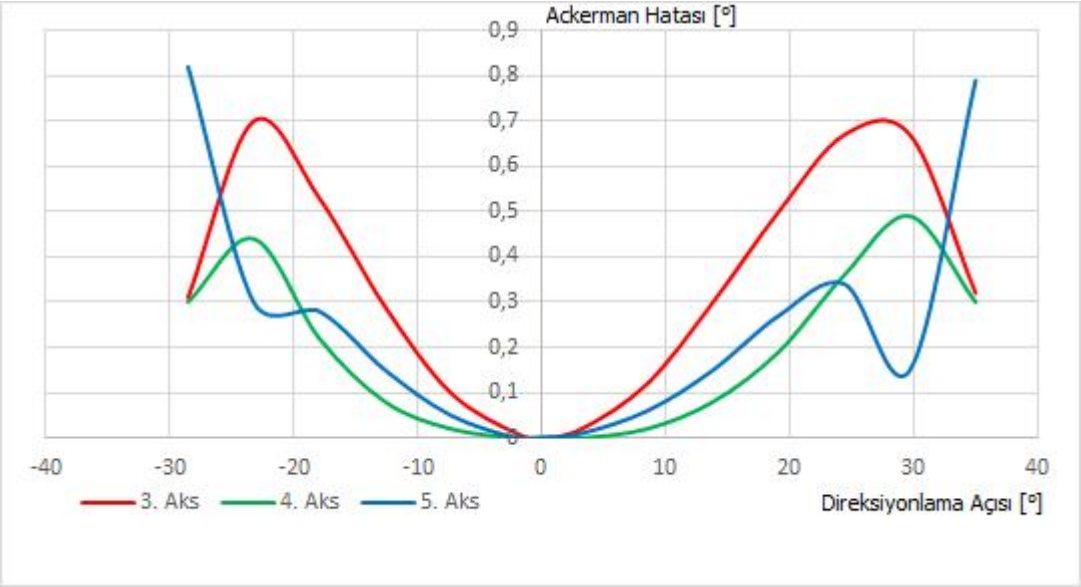


Şekil 7.4. Denklem 5.6 – Denklem 5.7'ye göre BA ve BAHJ algoritmalarının elde ettiği sonuç

Şekil 7.4'teki grafikte ve Çizelge 7.2'deki değerlerde görüldüğü gibi BA ve BAHJ algoritmaları Denklem 5.6 – Denklem 5.7'ye göre Ackerman Hatasını 1° 'ye ve 1° 'nin altına indirebilmiştir.

7.3. 3, 4 ve 5. Aksın Ackerman Hatası Değişimi

Bu çalışmada; 1. aks ile 2. aks arasındaki mekanizma optimize edilirken; 3., 4. ve 5. aksların Ackerman hatalarını etkileyecek bir durum yoktur. Çünkü bu akslar elektro-hidrolik eyleyiciler tarafından tahrik edildiği için bağımsız hareket edebiliyordu. Bu aksların bağımsız olması da yengeç yürüyüş moduna olan ihtiyaçtan dolayıdır. Sonuç olarak bu üç aksın Ackerman hatası optimizasyondan etkilenmemiştir. Bu üç aksın Ackerman hatası da Şekil 7.5'teki grafikte görülmektedir.



Şekil 7.5. Son 3 aksın dümenleme açısına bağlı Ackerman hatası değişimi

Eğer Şekil 7.5'te Ackerman hatasının 1° 'nin üzerine çıktığı bir durum gözlenseydi, bu durumda 3 – 4 – 5 akslarının ilk iki aksa olan mesafelerini optimizasyon parametresi yapılıp, bu mesafeler de optimize edilebilirdi.

8. SONUÇ VE ÖNERİLER

Bu çalışmada, bir ağır vasıtanın çok akslı direksiyon mekanizmasının optimizasyonu yapılmıştır. Optimizasyon çalışmalarında Ackerman dümenleme hatasından türetilen amaç fonksiyonunu minimize etmek için Arı Algoritması ve Hooke-Jeeves Algoritması eşzamanlı olarak kullanılmıştır. Ackerman hatası, çok akslı direksiyon mekanizmanın matematiksel modeli yerine CAD programı tarafından sağlanmış katı modeli kullanılarak hesaplanmıştır. Direksiyon mekanizmasının matematiksel modelinin elde edilmesindeki zorluğa karşılık katı model kullanarak dümenleme açılarının hesaplanması daha kolay ve basittir. Ackerman hatası hesaplamalarında Solidworks CAD programı ortamındaki katı model kullanılmıştır. Ackerman hatasının optimizasyonunda Arı Algoritması ve Hooke-Jeeves algoritmasını eşzamanlı çalıştırabilmek için VisualBasic.NET dilinde bir yazılım geliştirilmiştir. Solidworks ile geliştirilen yazılım arasındaki bilgi aktarımı için Solidworks API (Uygulama Programlama Arayüzü) kullanılmıştır. Geliştirilen teknik beş akslı bir ağır vasıtanın direksiyon mekanizmasının optimizasyonu için kullanılmıştır. Zhou ve Yang (2009), çalışmalarında Genetik Algoritması ile aksın kendi içindeki mekanizmasını 1.9° ’den 0.4°’ye indirmişlerdir. Collard ve ark. (2010), farklı başlangıç şartları ile kullandığı Genetik Algoritmasında optimizasyonunu yaptıkları düzlemsel Ackerman geometrisini Ackerman hatasını 0.85° ’ye düşürmüşlerdir. Javash ve ark. (2013), düzlemsel Ackerman geometrisini Genetik Algoritması ile optimize etmişler ve Ackerman hatasını 1° ’ye indirmişlerdir. Bu tez çalışmasında Ackerman geometrisini oluşturan aksın kendi içindeki mekanizma değil, akslar arasındaki mekanizma optimize edilmiştir. Ayrıca CAD modelini kullandığımız için düzlemsel bir kabul de yapılmamıştır. Bu tezde ağır vasıtanın akslar arası direksiyon mekanizmasının dümenleme sırasında Ackerman hatasını azaltmaya yönelik yapılan optimizasyon çalışmasında Arı Algoritmasına Hooke-Jeeves algoritması kombinasyu yapılarak Ackerman hatası hareketin her durumunda 1° ’ye ve 1° ’nin altına indirilmiştir. Ackerman hatasının bu seviyelere kadar düşmesi sayesinde tekerlekteki aşınmanın büyük ölçüde önüne geçilebilmiştir. Ackerman hatasının düşürülmesi ile daha az titreşimli bir manevra ve daha az yakıt harcaması da gerçekleştirilmiştir. Ayrıca manevra sırasında vasıtanın şasesi üzerindeki gerilmeler de azalmıştır.

Bu optimizasyonda 11 adet değişken optimize edilmiştir. Bu 12 boyutlu bir uzaya tekabül etmektedir. Böyle çok boyutlu bir uzayda dahi önerilen optimizasyon tekniğinin istenen sonuca ulaşabildiği görülmüştür.

İlave bir çabaya gerek kalmaksızın hali hazırda imalatçıların elinde bulunan bir CAD modeli kullanılarak, matematiksel model oluşturma zorluğu ve zahmeti çekilmeden doğrudan optimizasyon yapılabilmiştir. Bu ise optimizasyon işlemine geçişte hazırlık süresini büyük ölçüde kısaltmıştır.

Bu çalışmada global optimizasyon algoritmasına yerel bölgelerde yardımcı olabilmek için yerel arama algoritmalarının kombinasyonu da sağlanmıştır. Arı Algoritması ile Hooke-Jeeves algoritmasının kombinasyonunun optimizasyon sonuçları yaklaşık aynı olmuştur ve kombinasyon optimizasyon hızını da %47'ye kadar artırdığı gözlemlenmiştir. İki adet hata fonksiyonu türetilmiştir, her ikisinin de optimizasyon sürelerinin farklı oranlarda kısaldığı görülmüştür. Bunun sebebi farklı hata fonksiyonlarında araştırma uzayı değişmektedir ve farklı yakınsamalar elde edilmektedir. Aynı hata fonksiyonu kullanılsa dahi Arı Algoritması rastgele değişken dizileri ürettiği Hooke-Jeeves Algoritmasına gönderilen başlangıç şartları da değişmektedir. Farklı başlangıç şartlarında farklı yerel minimum noktalar bulunmaktadır. Bu yüzden de her problemde farklı süre düşüşleri olmaktadır. Sonuç olarak Hooke-Jeeves Algoritmasının kombinasyonu Arı Algoritmasının hızını artırmaktadır.

Optimizasyonda iki farklı hata fonksiyonu denenmiştir. Hata fonksiyonunda 0.20° 'lik bir tampon bölge hata fonksiyonunun doğrusallığını bozmadan hatanın artmasını sağlamış, bu da optimizasyonun 1° 'ye daha fazla odaklanmasını sağlamıştır.

Bu çalışmadan sonra yapılabilecek çalışmalar hakkında şunlar söylenebilir.

- CAD programındaki model ile optimizasyon, CAE programları ile ilişkilendirilebilir. Örneğin, CAD programındaki optimizasyon, sonlu elemanlar yöntemi kullanan bir program ile statik analizi yapılarak, optimizasyonun önerdiği çözümün mukavemetli olup olmadığı kontrol edilebilir.
- Son üç aksın Ackerman hataları eğer 1° 'nin üzerinde olursa, son 3 aksın 1. aksa olan mesafelerinin optimizasyonu yapılabilir.
- Çok akslı vasıtaların yüksek hızlarda dümenlenmesinde direksiyon mekanizmasının optimizasyonu veya elektro-hidrolik eyleyicili olan son 3

aksın yüksek hızlarda hangi dümenleme açılarında olması gerektiği yine optimizasyon programı ile CAE programları ilişkilendirilerek yapılabilir.

- Aks ile vasıta şasesi arasında yer alan süspansiyonlar, ±125 mm yaylanma genliğine izin verebilmektedir. Vasıtanın engebeli arazide seyri esnasında süspansiyon konumu 0 mm'den uzaklaşabilmekte ve -125 mm ile +125 mm'ye yaklaşmaktadır. Bu durumlarda hedeflenen Ackerman hatası üzerine bir standart olmadığı için bu çalışma, süspansiyon hareketi 0 mm'de kalacak şekilde, yani düz yolda olduğu kabul edilerek yapılmıştır. Eğer süspansiyon konumu ±125 mm olduğu konumlarda hedeflenen bir Ackerman hatası istenirse, optimizasyondaki amaç fonksiyonuna bu konumlar için de uygun bir ağırlık katsayısı belirlenerek hem düz yolda hem de engebeli yolda Ackerman hatasını azaltmaya yönelik bir optimizasyon çalışması yapılabilir.

9. KAYNAKLAR

Aghazadeh, F. and Meybodi, M. R., 2011, Learning bees algorithm for optimization, *International Conference on Information and Intelligent Computing*, Vol. 18.

Alhajj, A. S. M., (2011), Path synthesis of four-bar mechanism using harmony search optimization, Yüksek lisans tezi, *National Institute of Technology Rourkela*, Rourkela-India.

Ansarey M., S., Shariatpanahi, M., and Salimi, S., 2005, Optimization of Vehicle Steering Linkage With Respect to Handling Criteria Using Genetic Algorithm Methods, *SAE Technical Paper 2005-01-3499*, doi:10.4271/2005-01-3499.

Ardayfio, D. D., and Qiao, D., 1987, Analytical design of seven joint spatial steering mechanisms, *Mechanism and machine theory*, 22(4), 315-319.

Beyersdorfer, S., and Wagner, S., 2013, Novel model based path planning for multi-axle steered heavy load vehicles, *Proc. 16th International IEEE Conference on Intelligent Transportation Systems*, 424-429.

Bulatović, R. R., and Đorđević, S. R., 2004, Optimal synthesis of a four-bar linkage by method of controlled deviation, *Theoretical and applied mechanics*, 31(3-4), 265-280.

Bulatović, R. R., and Đorđević, S. R., 2012, Optimal synthesis of a path generator six-bar linkage, *Journal of mechanical science and technology*, 26(12), 4027-4040.

Burha, M., 2010, İki akstan dümenlenen üç akslı özel maksatlı bir taşıtın direksiyon mekanizmasının kinematik tasarımı, Yüksek lisans tezi, *İstanbul Teknik Üniversitesi Fen Bilimleri Enstitüsü*, İstanbul-Türkiye, 59-61.

Chang, K. H., and Joo, S. H., 2006, Design parameterization and tool integration for CAD-based mechanism optimization, *Advances in Engineering Software*, 37(12), 779-796.

Chicurel, E., 1999, A 180 steering interval mechanism, *Mechanism and machine theory*, 34(3), 421-436.

Chunyan, M., Jingnuo, Y., Yuyan, Y., and Kai, W., 2010, Design for dual-front axle steering angle of the heavy truck, *Educational and Network Technology (ICENT), 2010 International Conference,* 185-187.

Collard, J. F., Duysinx, P., and Fisette, P., 2010, Optimal synthesis of planar mechanisms via an extensible-link approach, *Structural and Multidisciplinary Optimization*, 42(3), 403-415.

Erdemir, A. ve Kalyoncu, M., 2015, Bir Ağır Vasıtanın Çok Akslı Direksiyon Mekanizmasının Arı Algoritması Kullanılarak Optimizasyonu, *Uluslararası katılımlı 17. Makina Teorisi Sempozyumu-UMTS 2015*, İzmir-Türkiye, 421-426.

Felzien, M. L., and Cronin, D. L., 1985, Steering error optimization of the Macpherson strut automotive front suspension, *Mechanism and machine theory*, 20(1), 17-26.

Fernandes, F. P., Costa, M. F. P., Fernandes, E. M. D. G., and Rocha, A. M. A., 2013, Multistart Hooke and Jeeves filter method for mixed variable optimization, *FEDER and FCT (Foundation for Science and Technology)*.

Hamidi, Y. E., Javash, M. S., Ettefagh, M. M., and Doust, F. A., 2012, Optimization of four-bar steering mechanism through Artificial Immune System (AIS) Algorithm, *Control, Automation and Systems (ICCAS), 2012 12th International Conference*, 913-917

Hanzaki, A. R., Rao, P. V. M., and Saha, S. K., 2009, Kinematic and sensitivity analysis and optimization of planar rack-and-pinion steering linkages, *Mechanism and Machine theory*, 44(1), 42-56.

Hartenberg, R. S. and Denavit, J., 1964, Kinematic synthesis of linkages, *McGraw-Hill*, *Department of Mechanical Engineering and Astronautical Sciences Northwestern University*, *Evanston-Illinois-USA*, 148-368.

Heißing, B., and Ersoy, M., 2010, Chassis handbook: fundamentals, driving dynamics, components, mechatronics, perspectives, *Springer Science & Business Media*, 26-88

Hooke, R., and Jeeves, T. A., 1961, "Direct Search" Solution of Numerical and Statistical Problems, *Journal of the ACM (JACM)*, 8(2), 212-229.

Javash, M. S., Ettefagh, M. M., and Hamidi, Y. E., 2013, Optimum design method for four-bar function generator using AIS and genetic algorithms, *Innovations in Intelligent Systems and Applications (INISTA), 2013 IEEE International Symposium*, 1-6.

Jujnovich, B. A., and Cebon, D., 2013, Path-following steering control for articulated vehicles, *Journal of Dynamic Systems, Measurement, and Control*, 135(3), 031006.

Kumar G. V. P., 2013, Comparative Evaluation of Particle Swarm Optimization Algorithms for the Optimal Dimensional Synthesis of Planar Four-bar Mechanism, *Proceedings of the 1st International and 16th National Conference on Machines and Mechanisms (iNaCoMM2013), IIT Roorkee, India*.

Li, Y., and Yang, L., 2010, Motion Synthesis and Coordinated Control in the Multi-Axle-Driving-Vehicle. *INTECH Open Access Publisher*, *Motion Control, Federico Casolo (Ed.),* ISBN: 978-953-7619-55-8

Liebherr, 2014, Mobile Crane LTM 1100-5.2 [online], http://www.liebherr.ca/AT/en-GB/products_ca-at.wfw/id-7932-0/measure-metric/tab-5515_1477 [Ziyaret Tarihi: 29 Haziran 2015].

Lohith, K., Shankapal, S. R., and Gowda, M. M., 2013, Development of four wheel Steering system for a car, *SAS Tech Journal*, 12, 90-97.

Long, Q., and Wu, C., 2014, A hybrid method combining genetic algorithm and Hooke-Jeeves method for constrained global optimization, *Journal of Industrial and Management Optimization*, 10(4), 1279-1296.

Luo, Y., Fan, X., Li, D., and Wu, X., 2008, Hyper-chaotic mapping newton Iterative method to mechanism synthesis. *Strojniški vestnik*, 54(5), 372-378.

Moser, I., 2009, Hooke-jeeves revisited, *Evolutionary Computation, CEC'09, IEEE Congress*, 2670-2676.

Ortiz, A., Cabrera, J. A., Nadal, F., and Bonilla, A., 2013, Dimensional synthesis of mechanisms using Differential Evolution with auto-adaptive control parameters, *Mechanism and Machine Theory*, 64, 210-229.

Ouezdou, F. B., and Regnier, S., 1997, General method for kinematic synthesis of manipulators with task specifications, *Robotica*, 15(06), 653-661.

Özbakir, L., Baykasoğlu, A., and Tapkan, P., 2010, Bees algorithm for generalized assignment problem, *Applied Mathematics and Computation*, 215(11), 3782-3795.

Penunuri, F., Peón-Escalante, R., Villanueva, C., and Pech-Oy, D., 2011, Synthesis of mechanisms for single and hybrid tasks using differential evolution, *Mechanism and Machine theory*, 46(10), 1335-1349.

Pham, D. T., and Castellani, M., 2009, The Bees Algorithm: modelling foraging behaviour to solve continuous optimization problems, *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, 223(12), 2919-2938.

Pham, D. T., Ghanbarzadeh, A., Koc, E., Otri, S., Rahim, S., and Zaidi, M., 2011, The Bees Algorithm–A Novel Tool for Complex Optimisation, *Intelligent Production Machines and Systems-2nd IPROMS Virtual International Conference*, 454.

Pham, D. T., and Kalyoncu, M., 2009, Optimisation of a fuzzy logic controller for a flexible single-link robot arm using the Bees Algorithm, *Industrial Informatics, INDIN 2009, 7th IEEE International Conference*, 475-480.

Pham, D. T., Koc, E., Lee, J. Y., and Phrueksanant, J., 2007, Using the bees algorithm to schedule jobs for a machine, *Proceedings Eighth International Conference on Laser Metrology, CMM and Machine Tool Performance, LAMDAMAP, Euspen, UK, Cardiff*, 430-439.

Pingyuan, X., and Cheng, W., 2009, Hybrid genetic algorithm optimization on steering trapezoid linkage of vehicle, *Computer Science and Education, ICCSE'09, 4th International Conference*, 49-52.

Portilla-Flores E. A., Morales-Cruz C., Suárez-Santillán R. A., Villarreal-Cervantes M. G., Niño Suárez, P. A., Yáñez, M. B. C., 2013, Optimal Synthesis of The Continuously Variable Transmission Input Mechanism Using An Evolutionary Approach, *International Journal of Innovative Computing, Information and Control*, 9(5), ISSN 1349-4198

Qin, G., Sun, Y., Zhang, Y., and Chen, L., 2012, Analysis and Optimization of the Double-Axle Steering Mechanism with Dynamic Loads, *The Open Mechanical Engineering Journal*, 5, 26-39.

Rios-Coelho, A. C., Sacco, W. F., and Henderson, N., 2010, A Metropolis algorithm combined with Hooke–Jeeves local search method applied to global optimization, *Applied Mathematics and Computation*, 217(2), 843-853.

Sheu, J. B., Hu, S. L., and Lee, J. J., 2008, Kinematic synthesis of a four-link mechanism with rolling contacts for motion and function generation, *Mathematical and Computer Modelling*, 48(5), 805-817.

Shufeng, W., 2010, Practice Design of Three-Axle Vehicle with Multi-axle Dynamic Steering, *Information and Computing (ICIC), 2010 Third International Conference*, 4, 177-180.

Simionescu, P. A., and Beale, D., 2002, Optimum synthesis of the four-bar function generator in its symmetric embodiment: the Ackermann steering linkage, *Mechanism and Machine Theory*, 37(12), 1487-1504.

Van Eeden, C. J., 2007, The steering relationship between the first and second axles of a 6x6 off-road military vehicle, Yüksek Lisans Tezi, *Department of Mechanical and Aeronautical Engineering, University of Pretoria*, 26-53.

Vujić, D., and Radojković, S., 2000, Synthesis Procedure of Planar Bar Linkages in Infinitesimally Close Position, *Facta Univeritatis Series: Mechanical Engineering*, 1(7), 835-848.

Wang, J. H., Yang, J. W., and Zhu, W. H., 2014, Optimal Design of Steering Geometry Using Genetic Algorithm Based on Binomial Weight Function, *Advanced Materials Research* , 850, 363-367.

Wang, S., Zhang, J., and Li, H., 2008, Steering performance simulation of three-axle vehicle with multi-axle dynamic steering, *Vehicle Power and Propulsion Conference, VPPC'08*, 1-5.

Watanabe, K., Yamakawa, J., Tanaka, M., and Sasaki, T., 2007, Turning characteristics of multi-axle vehicles, *Journal of Terramechanics*, 44(1), 81-87.

Yuce, B., Packianather, M. S., Mastrocinque, E., Pham, D. T., and Lambiase, A., 2013, Honey bees inspired optimization method: the Bees Algorithm. *Insects*, 4(4), 646-662.

Zhang, W., Shi, X., and Wang, L., 2011, Lamarckism and mechanism synthesis: approaching constrained optimization with ideas from biology, *Initiative Foundation of Scientific Research in CAUC*, arXiv:1109.6717.

Zhou, B., Li, D., and Yang, F., 2009, Optimization design of steering linkage in independent suspension based on genetic algorithm, *Computer-Aided Industrial Design & Conceptual Design, CAID & CD 2009, IEEE 10th International Conference*, 45-48.

ÖZGEÇMİŞ

KİŞİSEL BİLGİLER

Adı Soyadı : Abdullah ERDEMİR
Uyruğu : T.C.
Doğum Yeri ve Tarihi : KONYA - 1987
Telefon : 0538 313 41 21
Faks : -
E-mail : abdullaherdemir@gmail.com

EĞİTİM

Derece		Adı, İlçe, İl	Bitirme Yılı
Lise	:	Meram Konya Lisesi (Y.D.A.L.) Meram/KONYA	2006
Üniversite	:	Selçuk Ü. Müh. Mim. Fak. Makine Müh. Böl. Selçuklu/KONYA	2010
Yüksek Lisans	:	Selçuk Ü. Fen Bilimleri Enst. Makine Müh. A.B.D. Selçuklu/KONYA	
Doktora	:	-	

İŞ DENEYİMLERİ

Yıl	Kurum	Görevi
2010-2010	Sağbil Paketleme Makinaları Ltd. Şti.	Makina Mühendisi
2011-2011	Endüstriyel Elektrik Ltd. Şti.	Makina Mühendisi
2012-2013	Akınsoft Yazılım Ltd. Şti.	Mekanik Sorumlusu
2013-	MPG A.Ş.	Makina Mühendisi

UZMANLIK ALANI

Robotik Kinematik ve Dinamik, Otomatik Kontrol, CAD ve CAE Optimizasyon, Mekanizma Sentezi, Mekatronik Sistemler, VB.NET, Solidworks API, CAE Titreşim ve Şok Analizleri, Yorulma Ömrü

YABANCI DİLLER

İngilizce

BELİRTMEK İSTEĞİNİZ DİĞER ÖZELLİKLER

YAYINLAR

Erdemir, A. ve Kalyoncu, M., 2015, Bir Ağır Vasıtanın Çok Akslı Direksiyon Mekanizmasının Arı Algoritması Kullanılarak Optimizasyonu, *Uluslararası katılımlı 17. Makina Teorisi Sempozyumu-UMTS 2015*, İzmir-Türkiye, 421-426. (Yüksek Lisans tezinden yapılmıştır)

EK-1

```
'Hooke-Jeeves Algoritması
Public Function Hooke_Jeeves_Algorithm(ByVal nVars As Integer, ByRef
vars() As Double, ByRef StepSize() As Double, ByRef MinStepSize() As
Double) As Double
      Dim I As Integer
      Dim New_vars(nVars - 1) As Double
      Dim F As Double, XX As Double
      Dim BestF, LastBestF As Double
      Dim StopN As Boolean, Moved(nVars) As Boolean
      New_vars = CType(vars.Clone, Double())
      BestF = Error_Function_Calculate_1(New_vars)
      Do
            vars = CType(New_vars.Clone, Double())
            For I = 0 To nVars - 1
                  Moved(I) = False
                  Do
                        XX = New_vars(I)
                        New_vars(I) = XX + StepSize(I)
                        F = Error_Function_Calculate_1(New_vars)
                        If F < BestF Then
                              BestF = F
                              Moved(I) = True
                        Else
                              New_vars(I) = XX - StepSize(I)
                              F = Error_Function_Calculate_1(New_vars)
                              If F < BestF Then
                                    BestF = F
                                    Moved(I) = True
                              Else
                                    New_vars(I) = XX
                                    Exit Do
                              End If
                        End If
                  Loop
            Next I
            BestF = Error_Function_Calculate_1(New_vars)
            If BestF = 0 Then Exit Do
            LastBestF = BestF
            StopN = True
```

```
            For I = 0 To nVars - 1
                  If StepSize(I) >= MinStepSize(I) Then
                        StopN = False
                        Exit For
                  End If
            Next I
      Loop Until StopN
      vars = DirectCast(New_vars.Clone, Double())
      Return BestF
End Function

'Bu fonksiyon rastgele üretmek içindir.
Private Function RandomVariables(ByVal nSB As Integer, ByVal nVar As
Integer, ByVal vMax() As Double, ByVal vMin() As Double) As Double(,)
      Dim vTemp(nSB - 1, nVar - 1 + 1) As Double
      Randomize()
      For i = 0 To nVar - 1
            For j = 0 To nSB - 1
                  vTemp(j, i + 1) = vMin(i) + (vMax(i) - vMin(i)) *
Rnd()
            Next j
      Next i
      Return vTemp
End Function

'Komşuluk araştırması için kullanılan Bee Dance Fonksiyonu
Private Function Bee_Dance(ByVal rPfNS As Double, ByVal v() As Double,
ByVal nVar As Integer, ByVal Vmin() As Double, ByVal Vmax() As Double)
As Double()
      Dim TempV(UBound(v)) As Double
      For i = 0 To nVar - 1
            TempV(i) = (v(i) - rPfNS) + 2 * rPfNS * Rnd()
      Next i
      For i = 0 To nVar - 1
            If TempV(i) > Vmax(i) Then
                  TempV(i) = Vmax(i)
            End If
            If TempV(i) < Vmin(i) Then
                  TempV(i) = Vmin(i)
            End If
      Next i
```

```
        Return TempV
End Function

'Arı Algoritması
Public Function The_Bees_Algorithm(ByVal nSB As Integer, ByVal nBSP As
Integer, ByVal nESP As Integer, ByVal nRBaBsP As Integer, ByVal
nRBaEsP As Integer, ByVal rPfNS As Double, ByVal nVar As Integer,
ByVal BaslangicSartli As Boolean, ByVal v() As Double, ByVal vMin() As
Double, ByVal vMax() As Double) As Double
        'nSB 'kaşif arı sayısı 40-100
        'nItr 'iterasyon sayısı 1000-5000
        'nBSP 'en iyi arı sayısı 10-50
        'nESP 'en iyi elit arı sayısı 10-50
        'nRBaBsP 'en iyi arı etrafındaki araştırmacı arıların sayısı 10-
50
        'nRBaEsP 'en iyi elit arıların etrafındaki araştırmacı arıların
sayısı 10-50
        'rPfNS 'Komşuluk araştırma radyusu
        'nVar 'Parametre Sayısı
        Dim U(nSB - 1, nVar - 1 + 1) As Double, bestU(,) As Double,
TempU(,) As Double, ToplamHata(nSB - 1) As Double
        Dim TempV(nVar - 1) As Double, strTemp1() As String, strTemp2()
As String
        U = RandomVariables(nSB, nVar, vMax, vMin)
        For j = 0 To nSB - 1 ' nSB adet arının hatasını hesapla
                For q = 0 To nVar - 1
                        TempV(q) = U(j, q + 1)
                Next q
                U(j, 0) = Error_Function_Calculate_1(TempV)
        Next j
        bestU = DirectCast(U.Clone, Double(,))
        Array.Sort(BestU)
      U = Nothing
        Dim k As Long = 0
        While StopCondition = False
                k += 1
                ReDim TempU(nSB - 1, nVar - 1 + 1)
                For j = 0 To nESP - 1
                        ReDim U(nRBaEsP - 1, nVar - 1 + 1)
                        For i = 0 To nRBaEsP - 1
                                If i = 0 Then
```

```
                              For q = 0 To nVar
                                    U(0, q) = bestU(j, q)
                              Next q
                        Else
                              For a = 0 To nVar - 1
                                    TempV(a) = U(0, a + 1)
                              Next a
                              TempV = Bee_Dance(rPfNS, TempV, nVar,
vMin, vMax, U(0, 0))
                              For a = 0 To nVar - 1
                                    U(i, a + 1) = TempV(a)
                              Next a
                              U(i, 0) =
Error_Function_Calculate_1(TempV)
                        End If
                  Next i
                  Array.Sort(U)
                  For q = 0 To nVar
                        TempU(j, q) = U(0, q)
                  Next q
            Next j
            For j = nESP To nBSP - 1
                  ReDim U(nRBaBsP - 1, nVar - 1 + 1)
                  For i = 0 To nRBaBsP - 1
                        If i = 0 Then
                              For q = 0 To nVar
                                    U(0, q) = bestU(j, q)
                              Next q
                        Else
                              For a = 0 To nVar - 1
                                    TempV(a) = U(0, a + 1)
                              Next a
                              TempV = Bee_Dance(rPfNS, TempV, nVar,
vMin, vMax, U(0, 0))
                              For a = 0 To nVar - 1
                                    U(i, a + 1) = TempV(a)
                              Next a
                              U(i, 0) =
Error_Function_Calculate_1(TempV)
                        End If
                  Next i
```

```
                    Array.Sort(U)
                    For q = 0 To nVar
                        TempU(j, q) = U(0, q)
                    Next q
                Next j
                U = RandomVariables(nSB - nBSP, nVar, vMax, vMin)
                If chkTheBeesAndHookeCombination.Checked Then 'Algoritması
ile Hooke-Jeeves Algoritması Kombinasyonu aktif mi?
                    If GoreviniTamamlamis_Hooke_Jeeves_Var_mi Then
'Herhangi Hooke-Jeeves sonucu var mı?
                        For q = 0 To nVar
                            U(0, q) = Hooke_Jeeves_Result(0, q)
                        Next q
                    End If
                End If
                For i = 0 To nSB - nBSP - 1
                    For q = 0 To nVar - 1
                        TempV(q) = U(i, q + 1)
                    Next q
                    U(i, 0) = Error_Function_Calculate_1(TempV)
                Next i
                For i = nBSP To nSB - 1
                    For q = 0 To nVar
                        TempU(i, q) = U(i - nBSP, q)
                    Next q
                Next i
                bestU = DirectCast(TempU.Clone, Double(,))
                Array.Sort(bestU)
                'Arı Algoritması ve Hooke-Jeeves Algoritması
kombinasyonunun başlangıcı          If
chkTheBeesAndHookeCombination.Checked Then 'Algoritması ile Hooke-
Jeeves Algoritması Kombinasyonu aktif mi?
                    If Havuz.Count = 0 Then 'Havuz hiç arı yok mu?
                        For i = 0 To nBSP - 1 'Tüm en iyi arıları
havuza al.
                            strHooke = ""
                            For j = 1 To nVar
                                strHooke &= bestU(i, j) & ";"
                            Next j
                            Havuz.Add(strHooke)
                        Next i
```

```
                End If
                If Havuz.Count > 0 Then 'Havuzda arı var mı?
                    If IsThereAnyActiveHooke_Jeeves = False Then
'Aktif bir Hooke-Jeeves algoritması çalışyor mu?
                        Start_New_Hooke_Jeeves_Process(Havuz(1))
'Havuzdaki ilk arı ile bir Hooke-Jeeves işlemi başlat
                        Havuz.Remove(1) 'Havuzdaki ilk arıyı sil
                    End If
                End If
            End If
            'Arı Algoritması ile Hooke-Jeeves algoritması
kombinasyonunun sonu
        End While
        Return bestU(0, 0)
End Function

'Hata hesaplama fonksiyonu adım 1
Private Function Error_Function_Calculate_1(ByVal vars() As Double) As
Double
        Dim i As Integer, strExp As String, dblPenalty As Double
        Dim swApp As Sldworks, swModel As ModelDoc2
        swApp = CType(GetObject(, "Application.Sldworks"), Sldworks)
        swModel = CType(swApp.ActiveDoc(), ModelDoc2)
        For i = 0 To dgvKisitSartlari.RowCount - 2
            If i > 0 Then strExp &= "+"
            strExp = strExp & CStr(dgvKisitSartlari.Item(3, i).Value)
        Next i
        dblPenalty = 0
        For i = 0 To UBound(vars)
            If i <> 0 Then strExp = strExp & "+"
            strExp = strExp.Replace("v[" & i & "]", CStr(vars(i)))
            dgvDegiskenTakip.Item(4, i).Value = vars(i)
        Next i
        dblPenalty = Eval(strExp)
        For i = 0 To UBound(vars)
            If dgvDegiskenler.Item(1, i).Value = False Then
                swModel.Parameter(dgvDegiskenler.Item(1, i).Value) =
vars(i)
                swModel.EditRebuild3()
                swModel.ForceRebuild3(True)
            End If
```

```
        Next i
        dblHata = Calculate_Error(swApp) + dblPenalty
        Return dblHata
End Function

'Solidworks Modelinde Hata Var mı? Kaç tane?
Private Function SolidworksHataSay(ByVal swModel1 As ModelDoc2) As
Integer
    Dim vFeatures As Object = Nothing, vErrorCodes As Object =
Nothing, vWarnings As Object = Nothing
    Dim index As Integer
    swModel1.Extension.GetWhatsWrong(vFeatures, vErrorCodes,
vWarnings)
    If vWarnings Is Nothing Then
        Return -1
    Else
        Return UBound(vWarnings)
    End If
End Function

'Hata hesaplama fonksiyonunun 2. ve son adımı
Public Function Calculate_Error(ByRef swApp As SldWorks) As Object
    Dim MaksHata As Double = 1000000.0
    Dim dblTemp As Double, dblHataA As Double
    Dim swModel As ModelDoc2
    swModel = swApp.ActiveDoc
    dblHataA = 0
    For k = 35 To -28.5 Step (-28.5 - 35) / 12
        swDiSerbestlik = swModel.Parameter("D1@Sol1")
        If swSetValueReturnStatus_e.swSetValue_Successful <>
swDiSerbestlik.SetSystemValue3(k * Math.PI / 180,
swSetValueInConfiguration_e.swSetValue_InThisConfiguration, Nothing)
Then
            Return MaksHata
        Else
            GoTo HatayiHesapla
        End If
        If SolidworksHataSay(swModel) >= 0 Then
            Return MaksHata
        End If
        swModel.EditRebuild3()
```

```
        swDimensionHedef(0) =
swModel.Parameter("ACI_SOL2HATA@RADYUSLAR")
        dblTemp2(0) =
swDimensionHedef(0).GetSystemValue3(swSetValueInConfiguration_e.swSetV
alue_InThisConfiguration, Nothing)(0)
        dblTemp2(0) *= 180 / Math.PI
        If dblTemp2(0) < 90 Then
        ElseIf dblTemp2(0) < 270 Then
            dblTemp2(0) = Math.Abs(dblTemp2(0) - 180)
        Else
            dblTemp2(0) = Math.Abs(dblTemp2(0) - 360)
        End If
        swDimensionHedef(0) =
swModel.Parameter("ACI_SAG2HATA@RADYUSLAR")
        dblTemp2(1) =
swDimensionHedef(0).GetSystemValue3(swSetValueInConfiguration_e.swSetV
alue_InThisConfiguration, Nothing)(0)
        dblTemp2(1) *= 180 / Math.PI
        If dblTemp2(1) < 90 Then
        ElseIf dblTemp2(1) < 270 Then
            dblTemp2(1) = Math.Abs(dblTemp2(1) - 180)
        Else
            dblTemp2(1) = Math.Abs(dblTemp2(1) - 360)
        End If
        If dblTemp2(0) > 1 Then
            dblTemp2(0) = 1 + (dblTemp2(0) - 0.80) * 10
        End If
        If dblTemp2(1) > 1 Then
            dblTemp2(1) = 1 + (dblTemp2(1) - 0.80) * 10
        End If
        dblTemp = dblTemp2(0) ^ 2 + dblTemp2(1) ^ 2
        dblHataA += dblTemp
    Next k
    Return dblHataA
End Function
```