DEU FMD 26(78) (2024) 449-457

Dokuz Eylül Üniversitesi Mühendislik Fakültesi Fen ve Mühendislik Dergisi
Dokuz Eylul University Faculty of Engineering Journal of Science and Engineering
Elektronik/Online ISSN: 2547-958X

ARAŞTIRMA MAKALESI / RESEARCH ARTICLE

Altı Serbestlik Dereceli Robot Manipülatörün Ters Kinematik Analizi Için Gko Ve Çdde Algoritmalarının Karşılaştırmalı Analizi

Comparative Analysis of Gwo and Mtde Algorithms for Inverse Kinematic Analysis of Six Dof Robot Manipulator

Gökçe Sena Hocaoğlu, Nazlıcan Çavli, Emrah Benli *

Elektrik Elektronik Mühendisliği Bölümü, Karadeniz Teknik Üniversitesi, Trabzon, TÜRKİYE
*Sorumlu Yazar / Corresponding Author **: emrahbenli@ktu.edu.tr

Öz

Robot manipülatörlerinin eklem sayıları artmasıyla veya geometrik yapılarından kaynaklı manipülatörün ters kinematik analizinin yapılması zorlaşır. Bu durumda akıllı algoritmalara başvurulur. Bu çalışmada altı serbestlik derecesine (SD) sahip Mitsubishi Melfa RV-7FL-D robot manipülatörün ters kinematik analizi Gri Kurt Optimizasyon (GKO) algoritması ve Çoklu Denemeli Diferansiyel Evrim (ÇDDE) algoritması kullanılarak yapılmıştır. İlk adım için 3 adet senaryo tasarlanarak bu senaryolara ait hedeflenen konum değerleri belirlenmiştir. Ardından MATLAB üzerinde, çalışmada kullanılan robot manipülatörünün kinematiği matematiksel olarak modellenmiştir. Manipülatörün uç-efektörünün hedeflenen konuma gelmesini sağlamak için gerekli eksen açıları akıllı algoritmalar ile bulunmuştur. Geleneksel ve Geliştirilmiş GKO algoritmasıyla elde edilen sonuçlar ÇDDE algoritmasıyla elde edilen sonuçlarla karşılaştırılmıştır. Optimizasyon sonucu alınan veriler değerlendirildiğinde ÇDDE algoritmasının çok daha hızlı eklem açı değerlerini verdiği sonucuna varılmıştır. Optimizasyon algoritmaları ile yapılan ters kinematik analizi için literatür taraması yapıldığında ÇDDE algoritmasının kullanıldığı bir çalışmaya rastlanmamıştır. Çalışma, ÇDDE algoritmasını kullanarak optimizasyonla ters kinematik analizinde literatüre katkı sağlamayı amaçlamaktadır.

Anahtar Kelimeler: *Gri kurt optimizasyonu, çoklu denemeli diferansiyel evrim, ters kinematik analizi, robot manipülatör*

Abstract

With the increase in the number of joints of robot manipulators or due to their geometric structures, inverse kinematic analysis of the manipulator becomes difficult. In this case, intelligent algorithms are used. In this study, inverse kinematic analysis of the six-degrees-of-freedom (DOF) Mitsubishi Melfa RV-7FL-D robot manipulator was performed using Grey Wolf Optimization (GWO) algorithm and Multi-trial Vector-based Differential Evolution (MTDE) algorithm. The initial step involved designing three scenarios and determining their respective targeted position values. Subsequently, the mathematical modelling of the robot manipulator used in the study was conducted on MATLAB. Intelligent algorithms were employed to determine the axis angles necessary to ensure that the manipulator's end-effector reaches the targeted position. The results obtained with the traditional and improved GWO algorithm were compared with the results obtained with the MTDE algorithm. When the data obtained as a result of the optimization were evaluated, it has been observed that the MTDE algorithm gave much faster joint angle values. When the literature was searched for the inverse kinematics analysis made with optimization algorithms, no study was found in which the MTDE algorithm was used. The study aims to contribute to the literature in inverse kinematics analysis with optimization using the MTDE algorithm.

Keywords: *Grey wolf optimization, multi-trial vector-based differential evolution, inverse kinematic analysis, robot manipulator*

EXTENDED ABSTRACT

Introduction

In this study, the inverse kinematic analysis of Mitsubishi Melfa RV-7FL-D robot manipulator is performed using meta-heuristic algorithms such as Traditional Grey Wolf, Improved Grey Wolf and Multi-trial Vector-based Differential Evolution algorithms. Three different position values of the end effector were given to the system and the joint angles were calculated using metaheuristic optimization algorithms. The results were then checked by replacing the obtained angle values in the forward kinematic equation and the error value was calculated.

Materials and Methods

In this study, the inverse kinematic analysis of the Mitsubishi Melfa RV-7FL-D robot manipulator was performed. The inverse kinematics problem is difficult and complex to solve by numerical methods. Therefore, traditional grey wolf optimization, improved grey wolf optimization and multi-trial vector-based differential evolution optimization algorithms are used for the inverse kinematic analysis. The position of the end effector is given and joint angles suitable for the position are obtained using optimization algorithms.

DOI: https://doi.org/10.21205/deufmd.2024267811

Geliş Tarihi / Received: 13.10.2023
Kabul Tarihi / Accepted: 27.01.2024

Results and Discussion

This study performs the inverse kinematic analysis of the robot manipulator using traditional grey wolf optimization, improved grey wolf optimization and multi-trial vector-based differential evolution optimization algorithms. When the results are compared, it is observed that the traditional GWO algorithm for low iteration number and the MTDE algorithm for high iteration number achieve the result with less error. In both cases, the MTDE algorithm reached the result faster.

Conclusion

Inverse kinematics analysis is a challenging problem. In this study, inverse kinematic analysis of a robot manipulator was successfully performed with very small errors using metaheuristic algorithms

1. Giriş

Manipülatörler, robotik alanında önemli bir yere sahiptir ve manipülatörler üzerine yapılan araştırmalar gün geçtikçe artmakta, bu alandaki teknoloji gelişmektedir. Manipülatörlerin ters kinematik analizi, manipülatör tasarımında ve kontrolünde önemli bir yere sahiptir. Ters kinematik analizi, manipülatörün hareket analizinin ve yörünge planlamasının yapılmasında önemlidir. Manipülatörün ters kinematik analizi yapılarak konumu ve duruş pozu bilinen robotun sahip olduğu tüm eklem açıları bulunur. Ters kinematik çözümü analitik ve sayısal olmak üzere iki şekilde yapılabilir. Analitik çözüm yöntemi de kendi içinde cebirsel yöntem ve geometrik yöntem olmak üzere ikiye ayrılır [1]. Geometrik yapısı gereği veya montaj sırasında meydana gelen hatalardan dolayı bazı robotların ters kinematik analizini yapmak analitik çözüm yöntemleri ile zorlu ve karmaşıktır. Bu noktada akıllı algoritmalar kullanılabilir. Son yıllarda robotların herhangi bir pozisyonda ters kinematik analizini yapmak için yaygın olarak akıllı algoritmalardan yararlanılmaktadır [2].

Alkayyali ve Tutunji yaptıkları çalışma ile 6 serbest dereceli ekleme sahip KUKA robot manipülatörün kinematik analinizini Parçacık Sürü Optimizasyon (PSO) algoritmasını kullanarak gerçekleştirmişlerdir. PSO algoritmasının atalet ağırlığını ve daralma katsayısını yeniden formüle etmişlerdir. Sonuç olarak küçük bir hata yüzdesiyle robot manipülatörün duruş pozuna göre 6 ekleminin açılarını hesaplamışlardır [3].

Yiyang ve arkadaşları 6 serbest dereceli ekleme sahip Comau NJ-220 robot manipülatörün kinematik analizini Geliştirilmiş Parçacık Sürü Optimizasyon (PSO) tabanlı algoritmayı kullanarak gerçekleştirmişlerdir. Sabit değer olarak alınan atalet ağırlığını benzerlik faktörüne bağlı doğrusal olmayacak şekilde yeniden formüle etmişlerdir. Buna ek olarak göç operatörünü algoritmaya dahil etmişler ve orijinal PSO'ya göre daha iyi sonuç aldıklarını görmüşlerdir [2].

Dereli ve Köker çalışmalarında 7 serbest dereceli ekleme sahip robot manipülatörü için Ateş Böceği algoritmasını kullanarak ters kinematik hesaplaması sunmuşlardır. Öncelikle D-H parametreleri kullanılarak konum denklemleri elde edilmiştir. Ardından manipülatörü, hesaplanan konuma getirilebilmesi için gereken açı değerlerine Ateş Böceği Optimizasyon algoritması kullanılarak ulaşılmıştır. Sonuç olarak Ateş Böceği algoritmasının diğer optimizasyon algoritmaları ile karşılaştırıldığında 10 kat ile 10000 kat arasında daha iyi çözümler ürettiği ve daha kısa sürede çözüme ulaştığı görülmüştür [4].

Çift yönlü Parçacık Sürüsü Optimizasyonu (PSO) ile mobil robotun ters kinematiğinin hesaplanması üzerine yapılan bir çalışmada [5] çarpışma algılama ve engellerden kaçınma kısıtları üzerinde durulmuştur. Ters kinematik hesabı yapılırken manipülatör ayırma tekniği kullanılmıştır. 4 farklı topoloji denenmiş en iyi sonuç manipülatörün ortadan ayrıldığı topolojide elde edilmiştir.

7 serbest dereceli ekleme sahip robotun ters kinematik çözümü üzerine yapılan çalışmada [6] geleneksel PSO ve atalet ağırlığının değiştirildiği iki farklı tür PSO algoritması kullanılmıştır. Çalışma sonucunda atalet ağırlığı global en iyi sonuç için düzenlenen PSO algoritmasının diğer iki algoritmaya göre daha az hata ile hedeflenen pozisyona ulaştığı görülmüştür.

Marić ve arkadaşları çalışmalarında Riemannian optimizasyonunu kullanarak mesafe tabanlı ters kinematik problemi çözümü sunmuştur. Önerilen yöntem çeşitli görev kısıtları altında büyük bir robot grubuna uygulanmış ve geleneksel yöntemlere göre daha yüksek başarı oranına ulaştığı görülmüştür. Ayrıca önerilen yöntem çoklu çalışma alanı kısıtı içeren problemlerle karşılaştırıldığında daha iyi performans göstermiştir [7].

Dereli 7 serbest dereceli ekleme sahip robotun ters kinematik çözümünü Geliştirilmiş Gri Kurt Optimizasyon algoritması kullanarak çözmüştür. Önerilen GKO algoritmasında yakınsama katsayısı değiştirilerek sonuçların yerel optimum değerlere takılmadan daha hızlı bir şekilde en iyi sonuca yakınsaması amaçlanmıştır. Geleneksel GKO ve diğer optimizasyon algoritmaları ile karşılaştırıldığında önerilen optimizasyon algoritmasının çok daha iyi sonuç verdiği gözlenmiştir [8].

Amiri ve Ramli Genetik Algoritma (GA) ve Parçacık Sürü Optimizasyonunun (PSO) kombinasyonu olan Genetik Sürü Optimizasyonunu (GSO) kullanarak 5 serbest dereceli ekleme sahip bir robotun ters kinematik analizini yapmıştır. Bu çalışmada her bir eklem ayrı ayrı Orantılı-İntegral-Türev (PID) denetleyicilerle kontrol edilmiş ve PID parametreleri GSO algoritması kullanılarak hesaplanmıştır. Çalışma sonucunda GSO algoritmasının GA ve PSO algoritmalarına göre yaklaşık olarak %20 daha iyi sonuç verdiği gözlenmiştir [9].

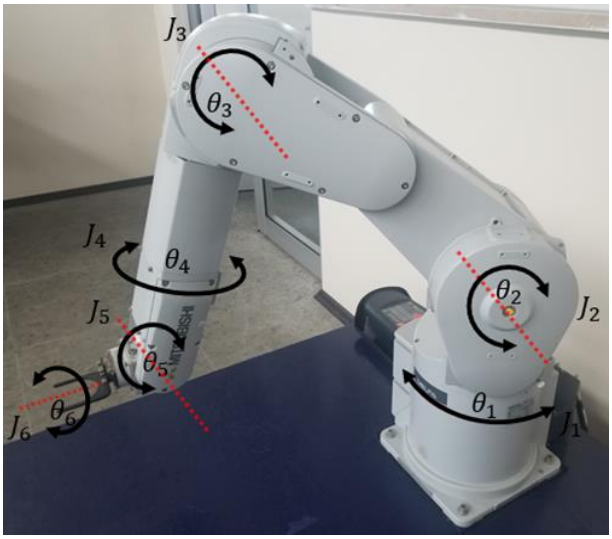
Literatüre bakıldığında ters kinematik problemin çözümü için meta-sezgisel optimizasyon algoritmalarının sıkça kullanıldığı görülmektedir. Fakat yine literatür taraması yapıldığında, sezgisel algoritmaların ters kinematik analizi sistemleri için karşılaştırmalı analiz çalışmalarına yer verildiği pek görülmemiştir. Bu çalışmada kullanılacak olan Geleneksel ve Geliştirilmiş GKO algoritmaları ile de yapılmış çalışmalar [8] bulunmaktadır. Fakat literatürde ÇDDE algoritmasının ters kinematik probleminin çözümünde kullanılmadığı görülmüştür. Bu çalışmada 6 serbest dereceli ekleme sahip Mitsubishi Melfa RV-7FL-D robot manipülatörün ters kinematik analizi için Geleneksel GKO, Geliştirilmiş GKO ve ÇDDE algoritmaları kullanılacaktır. Bölüm 2'de ters kinematik analiz ve Mitsubishi Melfa RV-7FL-D manipülatörün D-H parametreleri verilmiştir. Bölüm 3'te GKO algoritması ve Geliştirilmiş GKO algoritması, Bölüm 4'te ise ÇDDE algoritmaları açıklanmıştır. Simülasyon çalışması ve elde edilen sonuçlar Bölüm 5'te ele alınmış ve karşılaştırılmıştır. Son olarak Bölüm 6'da çalışmanın sonuçları ele alınmıştır.

2. Materyal ve Metot

2.1. Ters kinematik analizi

Kinematik analizi yapabilmek için öncelikle robot manipülatörüne ait Denavit-Hartenberg (D-H) parametrelerinin belirlenmesi gerekmektedir. Robot manipülatörün kaç adet

eklemi varsa her biri için D-H parametreleri çıkarılır ve tablo haline getirilir. Şekil 1.'de robot manipülatörünün görseli verilmiştir. Eklemler (J) ve eklem açıları (θ) şekil üzerinde gösterilmiştir. Şekil 1. üzerinde verilen kırmızı çizgiler, robot eklemlerinin koordinatlarından (x, y, z) biri olan z eksenini, bir başka deyişle eklemlerin dönüş eksenlerini temsil etmektedir. Tablo 1.'de ise Mitsubishi Melfa RV-7FL-D serisinin robot manipülatörünün kinematik analizi için oluşturulmuş D-H tablosu verilmiştir [10].



Şekil 1. Mitsubishi Melfa RV-7FL-D robot manipülatörün görseli ve eklem açıları.

Figure 1. Visualisation of Mitsubishi Melfa RV-7FL-D robot manipulator and joint angles.

Eksenin z ekseni boyunca dönüş açısı θ, ardışık iki eklemin z ekseni boyunca x eksenleri arasındaki mesafe d, x ekseni boyunca ardışık iki z ekseni arasındaki mesafe a, x ekseni boyunca dönüş açısı α ile ifade edilir. Burada S ile adlandırılan sütun, eklemlerin en büyük ve en küçük sınır açılarını ifade etmektedir.

Tablo 1. Mitsubishi Melfa RV-7FL-D robot manipülatörü serisinin D-H parametreleri.

Table 1. D-H parameters of Mitsubishi Melfa RV-7FL-D robot manipulator series.

N	θ	d(mm)	a(mm)	α(°)	S(°)
1	θ_1	d_1=400	0	a_1=90	±240
2	θ_2	d_2=435	a_2=-50	0	-110~130
3	θ_3	d_3=470	0	a_3=90	0~162
4	θ_4	d_4=85	0	a_4=90	-200~200
5	θ_5	0	0	a_5=90	-120~120
6	θ_6	0	0	0	-360~360

Bir ekleme ait dönüşüm matrisini elde etmek için Eş. 1'de verilen bitişik koordinat sistemleri arasındaki dönüşüm matrisi kullanılır.

$$T = \begin{bmatrix} cos\theta_i & -sin\theta_i & 0 & a_i \\ sin\theta_i cosa_i & cos\theta_i cosa_i & -sina_i & -d_i sina_i \\ sin\theta_i cosa_i & cos\theta_i cosa_i & cosa_i & d_i cosa_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (1)$$

Her bir eklemin dönüşüm matrisinin peşpeşe çarpılmasıyla uç-efektörün konumu belirlenebilir. Eş. 2'de eklemlerin dönüşüm matrislerinin çarpımları sonucu elde edilen homojen dönüşüm matrisi verilmiştir.

$${}^0_6T = \prod_{i=1}^{6} {}^{i-1}_{i}T = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2)$$

Eş. 2'de verilen a, o, n, uç-efektörün 3 birim vektörünü ifade eder ve sırasıyla x, y ve z eksenlerinin yön vektörleridir. Uç-efektörün pozisyon vektörü, p ile gösterilmiştir. Robot manipülatörün duruş pozu Eş.3'te gösterilmiştir.

$$P(\theta_1,\theta_2,\theta_3,\theta_4,\theta_5,\theta_6) = \begin{bmatrix} R_{3\times3} & P \\ 0 & 1 \end{bmatrix} \quad (3)$$

R_{3x3}, uç-efektörün koordinat sistemini gösteren 3×3'lük matristir. P ise uç-efektörün duruş matrisidir.

2.2. Gri kurt optimizasyon algoritması

Gri Kurt Optimizasyon algoritması kurt sürüsünün doğadaki hiyerarşik yapısından esinlenerek oluşturulmuştur. Gri kurtlar sürü şeklinde avlanır. Av bulunduğunda; Alfa, beta, delta ve omega kurtlarının avını kovalamasına ve en sonunda avını kuşatma davranışlarını tanımlar.

Sürüdeki en güçlü kurt olma özelliğini gösteren alfa kurt göç ve avlanmada lider konumundadır. Alfa kurttan sonra gelen ikinci güçlü kurt beta olarak adlandırılır. Hiyerarşi sıralamasında beta kurdu delta kurt takip eder. Alfa, beta, delta kurtlarının dışında kalan sürünün diğer kurtları omega kurt olarak adlandırılır. Gri kurt optimizasyon algoritmasının matematiksel modeli aşağıda verilmiştir [11].

Kurt ile av arasındaki mesafe Eş. 4 - Eş. 6'daki gibi hesaplanır.

$$\vec{D}_a = \left|\vec{C}_1 . \vec{X}_a - \vec{X}(t)\right| \quad (4)$$

$$\vec{D}_\beta = \left|\vec{C}_2 . \vec{X}_\beta - \vec{X}(t)\right| \quad (5)$$

$$\vec{D}_\delta = \left|\vec{C}_3 . \vec{X}_\delta - \vec{X}(t)\right| \quad (6)$$

Alfa, beta ve delta kurtlarının rastgele bozulmalarını temsil eden ve Eş. 7 ile hesaplanan vektörler sırasıyla $\vec{C}_1, \vec{C}_2, \vec{C}_3$ vektörleridir. $\vec{D}_a$, $\vec{D}_\beta$, $\vec{D}_\delta$ sırasıyla alfa, beta ve delta kurtlarının av ile aralarındaki mesafeyi, $\vec{X}_a$, $\vec{X}_\beta$, $\vec{X}_\delta$ kurtların pozisyon vektörünü temsil eder. $\vec{X}(t)$ gri kurdun t. iterasyondaki geçerli konumunu ifade eder. $\vec{X}_i(i = \alpha, \beta, \delta)$ avın konum vektörleridir.

$$\vec{C}_i = 2\vec{r}_2 \quad (7)$$

$\vec{r}_i$, [0,1]' de rastgele bir vektördür.

Alfa, beta ve delta kurtlarının pozisyonları aşağıda verilen Eş. 8 – Eş. 10'daki gibi hesaplanır.

$$\vec{X}_1 = \vec{X}_a - \vec{A}_1 . \vec{D}_a \quad (8)$$

$$\vec{X}_2 = \vec{X}_\beta - \vec{A}_2 . \vec{D}_\beta \quad (9)$$

$$\vec{X}_3 = \vec{X}_\delta - \vec{A}_3 . \vec{D}_\delta \quad (10)$$

$\vec{X}_i$(i=1,2,3) alfa, beta ve delta kurtlarının konum vektörleridir. $\vec{A}_i$(i=1,2,3) alfa, beta ve delta kurtlarının konum vektörlerinin

katsayılarıdır. $\vec{D}_i$(i=α,β,δ) alfa, beta ve delta kurtlarının av ile arasındaki mesafeyi belirtir.

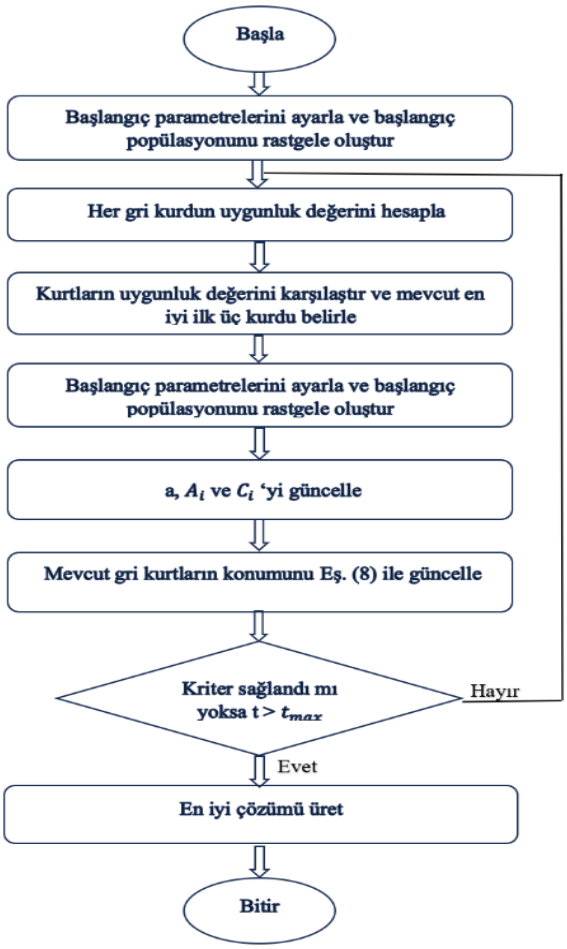
$$\vec{A} = 2a\vec{r_1} - a \quad (11)$$

$\vec{r}$ [0,1] arasında rastgele seçilmiş bir vektördür. a yakınsama faktörüdür ve iterasyon sayısının artmasıyla birlikte 2'den 0'a doğrusal olarak düşer. a faktörü Geleneksel Gri Kurt Optimizasyon algoritmasında şu şekilde tanımlanabilir:

$$a = 2 - 2\left(\frac{t}{max}\right) \quad (12)$$

t iterasyon sayısıdır, max ise en büyük iterasyon sayısıdır. $|\vec{A}| > 1$ olduğunda kurtlar küresel aramaya karşılık iyi bir av bulabilmek için arama alanlarını genişletirler. $|\vec{A}| < 1$ olduğunda kurtlar yerel aramaya karşılık gelen alanlarını daraltırlar.

Gri Kurt algoritmasının sözde kodu Şekil 2.'de verilmiştir.



Şekil 2. Gri Kurt Optimizasyon Algoritması sözde kodu [12]

Figure 2. Grey Wolf Optimisation Algorithm pseudo code [12]

Geleneksel Gri Kurt Optimizasyon algoritmasında yakınsama katsayısı doğrusal olarak alınmaktadır. Geliştirilmiş Gri Kurt Optimizasyon algoritmasıyla yakınsama katsayısı, doğrusal olmayan yakınsama formülü ile değiştirilmiştir. Algoritmada belirlenen ajan sayısı yakınsama formülüne dahil edilmiştir. Doğrusal olmayan yakınsama katsayısı formülü Eş. 13'te verilmiştir [8].

$$a = 2 \times \left(e^{\frac{-t*s}{max}}\right) \quad (13)$$

Burada s algoritmada seçilen ajan sayısıdır, t geçerli iterasyon sayısıdır, max ise algoritmada verilen en büyük iterasyon sayısıdır.

2.3. Çoklu denemeli diferansiyel evrim algoritması

Çoklu Denemeli Diferansiyel Evrim (ÇDDE) algoritmasında amaç farklı optimizasyon problemleri için farklı arama stratejileri kullanacak bir algoritma geliştirmektir. ÇDDE algoritması iki temel adımdan oluşmaktadır. Bunlar sırasıyla başlatma adımı ve çoklu deneme vektörü yaklaşımına dayanan hareket adımlarıdır. Başlatma adımında x_1'den x_N'e kadar N tane birey Eş.14 kullanılarak alt sınır (l) ve üst sınır (u) arama uzayına rastgele dağıtılır.

$$x_{ij} = l_j + (u_j - l_j) \times rand(0.1) \quad (14)$$

Burada x_{ij}, i. bireyin j. boyuttaki konumunu, l_j ve u_j ise j. boyuttaki alt ve üst sınırı ifade etmektedir. f(x_i) uygunluk fonksiyonu ile her bir bireyin uygunluk fonksiyonu hesaplanarak en küçük uygunluk değerine sahip birey g_{best} olarak atanır [13].

Çoklu Deneme Vektörü yaklaşımına dayalı hareket adımında problemler için farklı deneme vektörü üreticileri (DVÜ/TVP) kullanılmaktadır. Böylelikle çeşitli problemler için anlamlı bir arama stratejisi geliştirmeyi amaçlamaktadır. Bu amaçla üç farklı DVÜ kullanılır. Bunlar sırasıyla; temsili tabanlı deneme vektörü üreticisi (T-DVÜ/R-TVP), yerel rastgele tabanlı deneme vektörü üreticisi (Y-DVÜ/L-TVP) ve küresel en iyi geçmişe dayalı deneme vektörü üreticisidir (K-DVÜ/G-TVP). Bunlardan T-DVÜ yerel optimuma yakalanmayı önleyecek ve çeşitliliği koruyacak, Y-DVÜ hızlı bir yakınsamanın yanında keşif ve kullanım arasındaki dengeyi sağlayacak, K-DVÜ ise yerel optimumdan yararlanma ve kaçış sağlayacaktır. Hareket adımı dört alt adımdan oluşmaktadır. Bunlar sırasıyla kazanana dayalı dağıtım, çoklu deneme vektörü üretme, değerlendirme ve ömür boyu arşivleme adımlarıdır. Kazanana dayalı dağıtım adımı Tanım 1 ve Tanım 2'ye dayanmaktadır [13].

Tanım 1: Her biri n iterasyon içeren k tane WinIter verilmiş ve her WinIter için Win-TVP ile ifade edilen bir tane kazanan DVÜ bulunur. İlk iterasyon için T-DVÜ Win-TVP olarak kabul edilir. Daha sonra her iterasyon için en yüksek iyileştirme oranına sahip DVÜ yeni Win-TVP olarak güncellenir. $IR_{X\text{-}TVP}$ ile ifade edilen iyileştirilmiş X-TVP oranı Eş. 15'de verilmiştir. X-TVP; T, Y ya da K-DVÜ'yü ifade etmektedir.

$$IR_{X-TVP} = IF_{X-TVP} + FE_{X-TVP} \quad (15)$$

Burada IF_{X-TVP} ve FE_{X-TVP} sırasıyla X-TVP tarafından geliştirilmiş inceliklerin sayısını ve bir iterasyondaki fonksiyon değerlendirmelerinin sayısını ifade eder. Win-TVP belirlendikten sonra Tanım 2'deki dağıtım politikası kullanılarak deneme vektörleri T-DVÜ, Y-DVÜ ve K-DVÜ arasında dağıtılır.

Tanım 2: Verilen $N_{R\text{-}TVP}$, $N_{L\text{-}TVP}$ ve $N_{G\text{-}TVP}$ sırasıyla $X_{R\text{-}TVP}$, $X_{L\text{-}TVP}$ ve $X_{G\text{-}TVP}$ alt popülasyonların boyutlarını ifade etmektedir. Ardından, N birey bu alt popülasyonlara rastgele dağıtılarak büyüklükleri Win-TVP'ye göre Eş. 16 ve Eş. 17'de belirtilen ödül ve ceza kuralları ile belirlenir.

Ödül kuralı: Eğer T-DVÜ ya da Y-DVÜ Win-TVP ise,

$$N_{Win-TVP} = 0.6 \times N \text{ ve } N_{Loser-TVPs} = 0.2 \times N \quad (16)$$

Ceza kuralı: Eğer K-DVÜ Win-TVP ise,

$N_{G-TVP} = 0.2 \times N$ ve $N_{Loser-TVPs} = 0.4 \times N$ (17)

Nihai deneme vektörlerini oluşturmak için, hem T-DVÜ hem de K-DVÜ ürettikleri deneme vektörlerini karşılık gelen bireyler üzerinden çaprazlar. Bu çaprazlama her iterasyonda oluşturulan M dönüşüm matrisi ve onun tersi M^- ile yapılır. Ayrıca, T-DVÜ ve Y-DVÜ popülasyondaki bireylerin birliği olan unionPopulation ($X_{U\text{-}pop}$) adlı ek bir popülasyon ile çeşitlilik ve bilgi paylaşımını sürdürmek için ömür boyu arşiv kullanır. Her x_i için $F_i = randc_i(\mu_f, \sigma)$ olacak şekilde Cauchy dağılımı ile F ölçek faktörü hesaplanır. Burada μ_f Geliştirilmiş ölçek faktörlerinin ortalama değerini ve σ sabit değeri 0.2 olan varyansı ifade etmektedir. F_i değeri (0,1] arasında olmalıdır. 1'den büyük ise 1 kabul edilir, 0'a eşit veya küçük ise tekrar hesaplanır. μ_f değeri başlangıçta 0.5 olarak alınır. Popülasyonda herhangi bir gelişmiş birey varsa μ_f Eş. 18'de verilen ağırlıklı Lehmer ortalaması ile hesaplanır. Gelişmiş bir birey yoksa ölçek faktörleri değişmeden kalır.

$$\mu_f = \frac{\sum_{f_i \in S_f} \omega_{f_i} \times f_i^2}{\sum_{f_i \in S_f} \omega_{f_i} \times f_i} \quad (18)$$

Burada S_f geçerli iterasyondaki $f(u_i) < f(x_i)$ olan tüm Fs kümesini ve ω_{f_i} ağırlığı Eş. 19'da verildiği gibi hesaplanmaktadır.

$$\Delta f_i = f(x_i) - f(u_i), \omega_{f_i} = \frac{\Delta f_i}{\sum_{f_i \in S_f} \Delta f_i} \quad (19)$$

T-DVÜ, yerel optimum yakalanmayı önlemek ve çeşitliliği sağlamak için tasarlandığından unionPopulation'dan rastgele bir birey olan x_{u_pop} ile x_{i_best} ve x_{i_worst} bireylerinin konumlarını kullanarak x_i bireyini alt populasyonundan taşır. x_{i_best} ve x_{i_worst} sırasıyla $X_{R\text{-}TVP}$ alt populasyonunun uygunluk değerlerinin artan ve azalan sıralı gösteriminin i. öğeleridir. T-DVÜ'daki u_i deneme vektörü Eş. 20'de M ve M^- matrisleri kullanılarak üretilir. v_i ise Eş. 21'deki gibi elde edilir [13].

$$u_i = M \times x_i + \underline{M} \times v_i \quad (20)$$

$$v_i = x_i + F_i \times \left(x_{i_{best}} - x_i\right) + F_i \times \left(x_{i_{worst}} - x_i\right) + a_1 \times \left(x_{u\_pop} - x_i\right) \quad (21)$$

Burada a_1 arama sürecinin erken safhalarında daha yüksek, ileri safhalarında daha düşük değerler alan doğrusal olarak azaltılmış bir katsayıdır ve Eş. 22'deki gibi hesaplanır.

$$a_1 = 2 - iter \times (2/MaxIter) \quad (22)$$

Y-DVÜ, sadece kendi alt popülasyonundaki bireylerden değil, ayrıca unionPopulationdaki bireylerden de öğrenerek keşif ve kullanım arasındaki dengeden ve hızlı bir yakınsamadan yararlanır. $X_{L\text{-}TVP}$'den rastgele seçilen x_{r1} ve x_{r2} bireylerinin konumları arasındaki fark ve x_i'nin rastgele seçilmiş bir x_{u_pop}'den farkı Eş. 23'de gösterildiği gibi x_i'nin konumunu değiştirir.

$$v_i = x_i + F_i \times (x_{r1} - x_{r2}) + a_2 \times (x_{u_{pop}} - x_i) \quad (23)$$

Burada a_2 Eş. 24'teki gibi hesaplanan doğrusal olmayan bir şekilde azalan katsayıyı ifade eder.

$$a_2 = initial - (initial - final) \times \left(\left(\frac{MaxIter - iter}{MaxIter}\right)\right)^{Mu} \quad (24)$$

Burada initial ve final a_2 kontrol parametresinin başlangıç ve sonuç değerlerini, Mu ise bağımlı değerin boyutunu ifade eder. T-DVÜ'den farklı olarak burada nihai deneme vektörü u_i bir evrime değil bireysel öğrenmeye dayalıdır. Bu sebeple bir geçiş kullanmaz ve u_i=v_i olduğu söylenebilir.

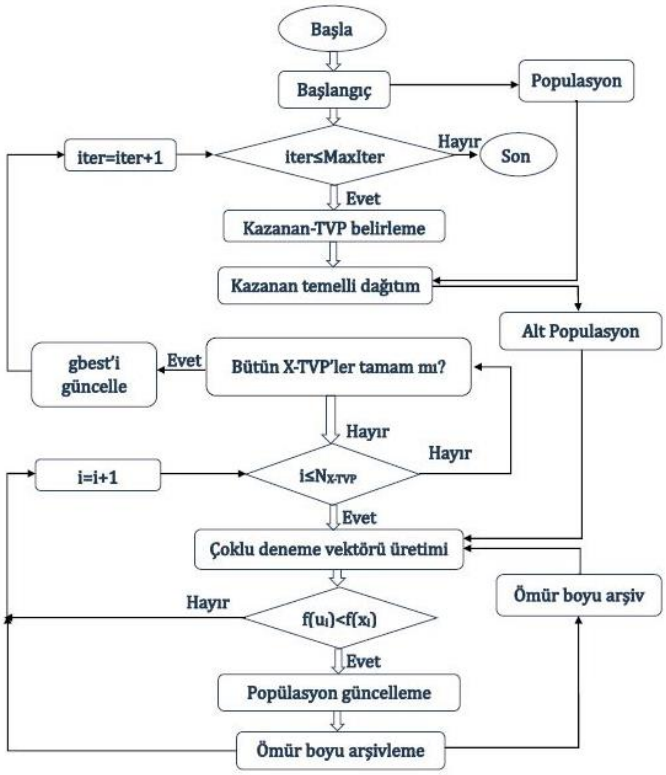
K-DVÜ küresel en iyi değer geçmişini kullanarak yerel optimuma yakalanma sorununun üstesinden gelmeye çalışır. İlk iterasyon için küresel en iyi değer (g_{best}) ile başlatılır. Ardından her iterasyonda g_{best} değerleri küresel en iyi değer geçmişine eklenir. Ayrıca K-DVÜ için g_{best} geçmişinden $N_{G\text{-}TVP}$ satırları ve D sütunları olan $M_{gb\text{-}h}$ matrisi oluşturulur. Bu yapı sayesinde farklı en iyi bireyler kullanılarak çeşitlilik kaybı önlenebilir. Her iterasyonda $M_{gb\text{-}h}$ matrisi g_{best} geçmişini N/m kez kopyalayarak oluşturulur. Burada m g_{best} geçmiş üyelerinin sayısını ifade etmektedir. Ardından Eş. 25 ve Eş. 26 kullanılarak sırasıyla u_i deneme vektörü ve v_i üretilir [13].

$$u_i = M \times x_i + \underline{M} \times v_i \quad (25)$$

$$v_i = x_{i_{gb-h}} + a_2 \times (x_{r1} - x_{r2}) \quad (26)$$

Burada $x_{i_{gb-h}}$ $M_{gb\text{-}h}$ matrisinin i. satırıdır. x_{r1} ve x_{r2} ise $X_{G\text{-}TVP}$'den rastgele seçilmiş iki bireydir.

Değerlendirme adımı, $U_{X\text{-}TVP}$'den gelen her u_i'nin kalitesi, $X_{X\text{-}TVP}$'den karşılık gelen x_i bireyleriyle karşılaştırılır. $f(u_i) \geq f(x_i)$ ise x_i değişmeden kalır, eğer $f(u_i) < f(x_i)$ ise x_i'nin konumu u_i ile güncellenir. Ayrıca, her bir DVÜ tarafından iyileştirilen birey sayısı 1 arttırılarak denklem (18)'de kullanmak üzere F_i değeri depolanır.



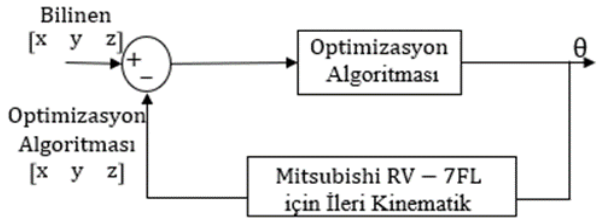
Şekil 3. Çoklu Denemeli Diferansiyel Evrim Algoritması sözde kodu [13].
Figure 3. Pseudo code of the Multiple Trial Differential Evolution Algorithm [13].

Ömür boyu arşivleme adımında x_i ömür boyu arşive daha düşük bir çözüm olarak eklenir. Arşiv başlangıçta boştur ve N tane bireyin konum ve ömür bilgilerinin tutabilir. Her iterasyon sonunda arşivlenen bireylerin ömrü 1 artar ve N tane en genç birey arşivlenir. Son olarak her iterasyonda iterasyon sayısı (iter)

1 arttırılır ve tanımlanmış olan iterasyon değerine (MaxIter) ulaşana kadar evrimsel arama yinelenir [13]. ÇDDE algoritmasının sözde kodu Şekil 3'te verilmiştir.

3. Bulgular ve Tartışma

Çalışmada robot manipülatörün duruş pozu için konum değerleri seçilmiştir. Seçilen konum değerleri ile optimizasyon algoritmasının bulduğu eklem açı değerlerine göre ileri kinematik hesaplaması yapılmıştır. Hesaplama sonucu elde edilen duruş pozu ile hedeflenen duruş pozu arasındaki hata verilen iterasyonlar tamamlanıncaya kadar hesaplanarak amaç fonksiyonu en aza indirgemek istenmiştir. Şekil 4'te sistemin blok diyagramı verilmiştir.



Şekil 4. Sistem blok diyagramı

Figure 4. System block diagram

Ters kinematik analizi için kullanılan 3 algoritmada da amaç fonksiyonu olarak Eş. 27'de verilen matematiksel ifade kullanılmıştır. İterasyon sonucu elde edilen F değeri, hedeflenen konum ile optimizasyon algoritmaları sonucunda elde edilen açı değerlerinin yerine konmasıyla alınan konum bilgisi arasındaki mutlak hatayı ifade etmektedir.

$$F = \sqrt{(p_x - \underline{p}_x)^2 + (p_y - \underline{p}_y)^2 + (p_z - \underline{p}_z)^2} \quad (27)$$

p_x, p_y, p_z, optimizasyon sonucu verilen koordinat değerleridir. $\underline{p}_x, \underline{p}_y, \underline{p}_z$ ise hedef koordinat değerleri.

Kullanılan optimizasyon algoritmaları ile birlikte amaç fonksiyonunu minimize etmek istenmektedir. Robot manipülatörünün hedef duruş pozu ve optimizasyon sonucu alınan eklem açı değerleri ile elde edilen duruş pozu arasındaki öklid uzaklığının en az olması amaçlanmaktadır.

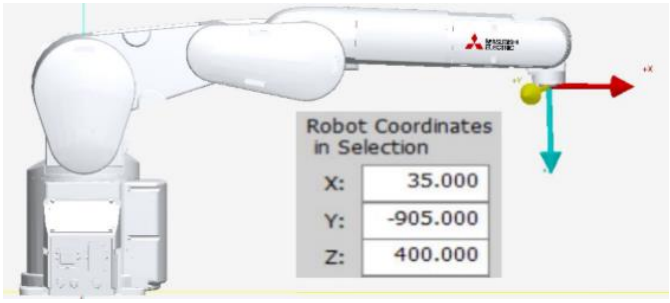
Bu çalışmada üç farklı senaryo için farklı konum değerleri seçilmiş, Geleneksel GKO, Geliştirilmiş GKO ve ÇDDE optimizasyon algoritmaları ile koşturulmuştur. Algoritmalarda kullanılacak iterasyon sayısının belirlenebilmesi için yapılan doygunluk çalışmasında her iki algoritma belli iterasyon değerleriyle koşturulmuş ve GKO algoritmasının 1000 iterasyon sonucu, ÇDDE algoritmasının ise 600000 iterasyon sonucunda doygunluğa ulaştığı görülmüştür. Her iki algoritmanın doygunluk iterasyon sayısı sonuçlarının rahat şekilde karşılaştırılabilmesi için, 3 senaryo da 300 ajan sayısı ile önce 1000 iterasyon daha sonra 600000 iterasyon ile kodlar 10'ar kere çalıştırılmış ve sonuçlar kaydedilmiştir. Bu senaryolar için kullanılan konum değerleri Tablo 2'de verilmiştir. Her bir senaryo farklı bir robot konfigürasyonunu ifade etmektedir. Çalışma için Mitsubishi RV-7FL-D robot manipülatörün fiziksel özellikleri ve mekanik sınırları ile ileri kinematiğin matematiksel ifadesi kodda tanımlanmıştır. Manipülatörün 6. ekleminin d-h parametresinde a, α ve d değerleri 0 olduğu için manipülatörün duruş pozu θ_6'dan bağımsızdır. Kod, intel CORE i5 işlemcili ve 8 gb ram'lik bilgisayarda Matlab R2022b sürümünde çalıştırılmıştır.

Tablo 2. Senaryolar için kullanılan konum değerleri.

Table 2. Location values used for the scenarios.

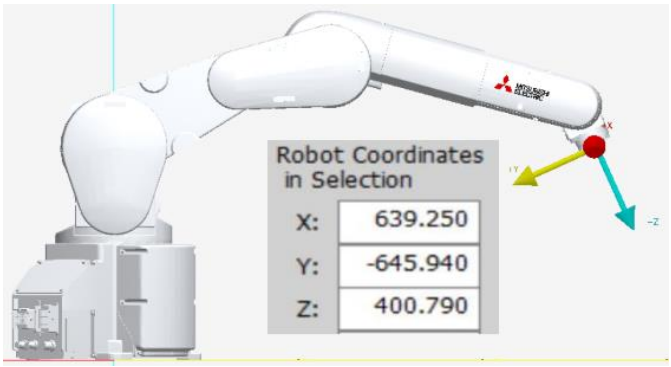
Senaryo	x (mm)	y (mm)	z (mm)
Durum 1	35	-905	400
Durum 2	639.25	-645.94	400.79
Durum 3	-50	-905	315

Şekil 5., 6. Ve 7.'de Tablo 2.'deki konum değerleri için robot kol duruş pozlarına ait simülasyon görselleri verilmiştir. Simülasyon görselleri RT Toolbox 3 simülasyon programı kullanılarak elde edilmiştir.



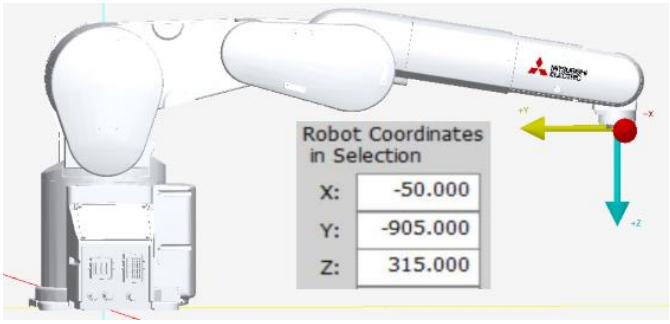
Şekil 5. Durum 1 için robot kol duruş pozu

Figure 5. Robot arm pose for case 1



Şekil 6. Durum 2 için robot kol duruş pozu

Figure 6. Robot arm pose for case 2



Şekil 7. Durum 3 için robot kol duruş pozu

Figure 7. Robot arm pose for case 3

Uygulanan üç senaryo, kullanılan optimizasyon algoritmaları ile 10'ar kere koştuktan sonra elde edilen en iyi sonuçlar 1000 iterasyon için Tablo 3.,4. ve 5.'te, 600000 iterasyon için Tablo 6., 7. Ve 8.'de verilmiştir.

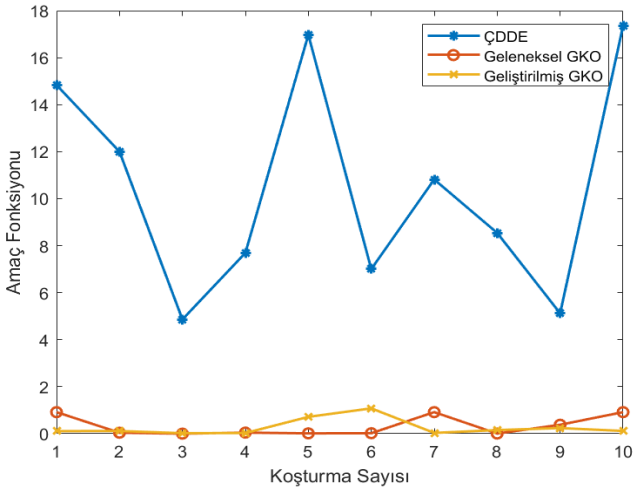
Farklı konum değerleri için oluşturulmuş senaryolar 1000 iterasyonla koşturulduğunda üç algoritma için alınan değerler tablo haline getirilmiş ve amaç fonksiyonu ile iterasyonu tamamlama süreleri bakımından karşılaştırılmıştır. Tablo 3.,4. ve 5.'te verilen değerlere bakıldığında düşük iterasyonda Geleneksel GKO algoritmasının diğer algoritmalara göre daha küçük bir hata verdiği görülmüştür. Algoritma hızları karşılaştırıldığında ise

ÇDDE algoritmasının diğer algoritmalara göre çok daha hızlı iterasyon sayısını tamamladığı sonucuna varılmıştır. Yine 1000 iterasyon için Şekil 8., 9. ve 10. incelendiğinde Geleneksel GKO algoritmasının amaç fonksiyon değerinin diğer iki algoritmaya göre daha düşük olduğu gözlemlenmiştir.

Tablo 3. Durum 1 için elde edilen benzetim sonuçları.

Table 3. Simulation results for case 1.

	Geleneksel GKO	Geliştirilmiş GKO	ÇDDE
θ_1	-0.000	-0.034	-1.270
θ_2	-1.573	-13.040	69.626
θ_3	90.647	95.413	50.209
θ_4	2.127	-108.054	107.185
θ_5	60.021	8.772	101.883
θ_6	-351.786	-19.308	-198.600
Süre(sn)	7.335	13.010	0.062
F	0.0005234	0.020942	4.8606



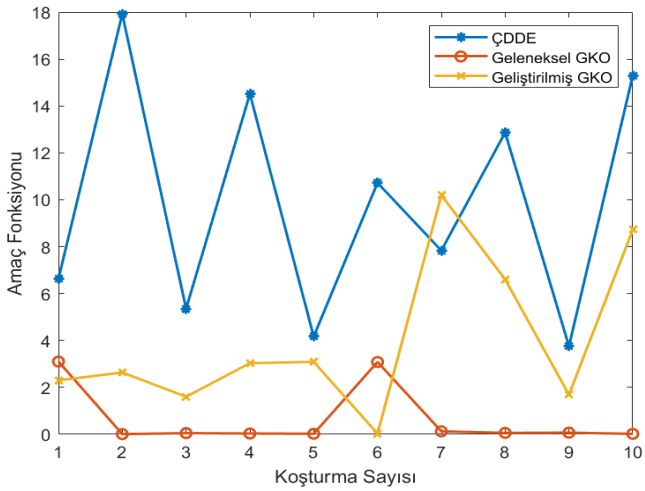
Şekil 8. Durum 1 için amaç fonksiyonunun karşılaştırılması.

Figure 8. Comparison of the objective function for case 1.

Tablo 4. Durum 2 için elde edilen benzetim sonuçları.

Table 4. Simulation results for case 2.

	Geleneksel GKO	Geliştirilmiş GKO	ÇDDE
θ_1	49.925	39.494	47.744
θ_2	74.607	-104.047	125.486
θ_3	160.190	159.899	115.788
θ_4	51.792	-9.511	187.684
θ_5	-16.766	115.859	5.659
θ_6	8.247	-41.369	351.257
Süre(sn)	6.651	9.417	0.065
F	0.0096858	0.028033	3.7567



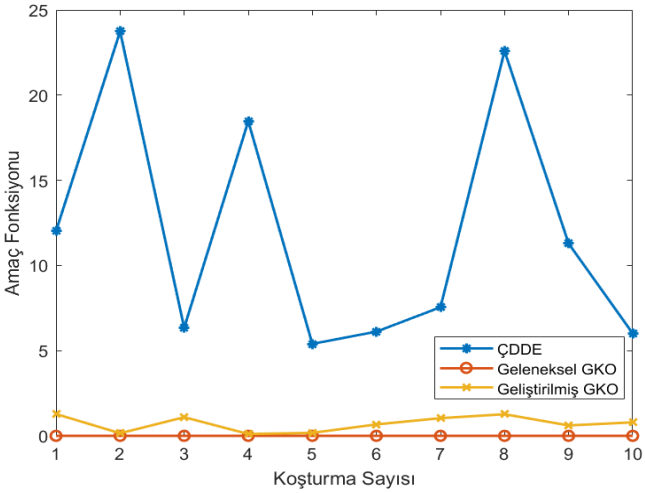
Şekil 9. Durum 2 için amaç fonksiyonunun karşılaştırılması.

Figure 9. Comparison of the objective function for case 2.

Tablo 5. Durum 3 için elde edilen benzetim sonuçları.

Table 5. Simulation results for case 3.

	Geleneksel GKO	Geliştirilmiş GKO	ÇDDE
θ_1	-3.552e-14	-0.110	-5.132
θ_2	-5.455e-15	0.080	48.779
θ_3	3.854e-13	1.098	5.293
θ_4	-0.662	-27.317	-62.995
θ_5	-4.380	93.551	-112.584
θ_6	35.804	124.724	-248.472
Süre(sn)	9.540	16.274	0.043
F	0	0.10833	5.3988



Şekil 10. Durum 3 için amaç fonksiyonunun karşılaştırılması.

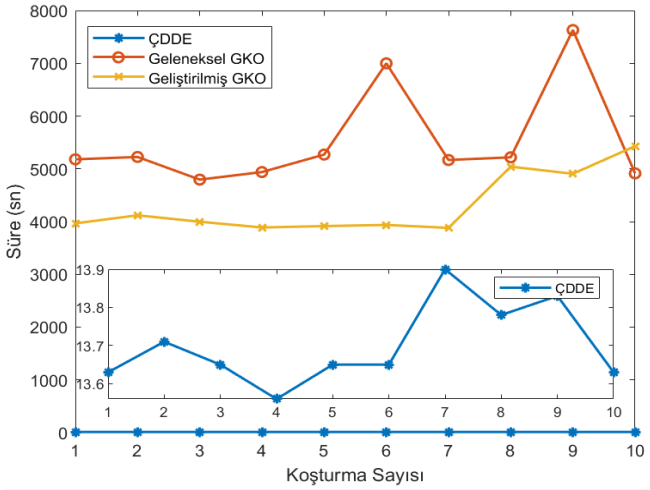
Figure 10. Comparison of the objective function for case 3.

İterasyon değeri 600000 yapılarak algoritmaların her biri uygulanan üç senaryoya göre tekrar koşturulmuş ve yeni değerlere göre tablolar tekrar oluşturulmuştur.

Tablo 6. Durum 1 için elde edilen benzetim sonuçları.

Table 6. Simulation results for case 1.

	Geleneksel GKO	Geliştirilmiş GKO	ÇDDE
θ_1	-4.188e-07	1.974e-05	-1.840e-13
θ_2	0.044	0.245	3.048e-05
θ_3	89.981	89.898	89.999
θ_4	16.216	-40.808	-111.123
θ_5	7.208	4.464	40.848
θ_6	-215.0742	20.866	-33.632
Süre(sn)	5167.561	3997.204	13.566
F	4.071e-07	0.000501	0



Şekil 11. Durum 1 için benzetim süreleri karşılaştırılması.

Figure 11. Comparison of simulation times for case 1.

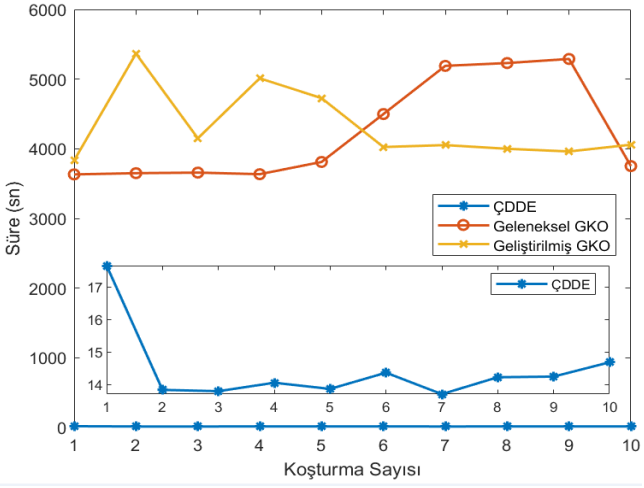
Durum 1 için [35, -905, 400] konum değerleri seçilmiştir. Geleneksel GKO, Geliştirilmiş GKO ve ÇDDE algoritmaları için elde edilen en iyi sonuçlar Tablo 6.'da verilmiştir. Sonuçlara bakıldığında en hızlı sonuca ulaşan algoritmanın ÇDDE olduğu görülmüştür. Amaç fonksiyonlarına bakıldığında ÇDDE algoritmasının diğer algoritmalara göre daha hızlı bir şekilde 0 hataya ulaştığı gözlemlenmiştir.

Şekil 11.'de Durum 1 için tekrarlanan benzetim çalışmalarının süreleri verilmiştir. Şekle bakıldığında ÇDDE algoritmasının diğer algoritmalara göre çok daha hızlı olduğu görülmüştür.

Tablo 7. Durum 2 için elde edilen benzetim sonuçları.

Table 7. Simulation results for case 2.

	Geleneksel GKO	Geliştirilmiş GKO	ÇDDE
θ_1	39.474	39.474	49.928
θ_2	105.452	-104.359	-75.641
θ_3	19.735	160.264	19.735
θ_4	-37.033	12.298	138.765
θ_5	-0.485	29.04	8.730
θ_6	7.208	5.141	33.997
Süre(sn)	3632.120	3832.507	13.703
F	1.685e-05	0	0



Şekil 12. Durum 2 için benzetim süreleri karşılaştırılması.

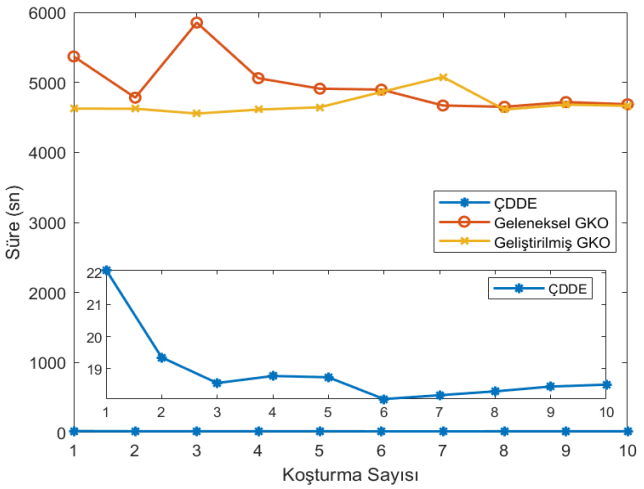
Figure 12. Comparison of simulation times for case 2.

Durum 2 için [639.25, -645.94, 400.79] konum değerleri seçilmiştir. Geleneksel GKO, Geliştirilmiş GKO ve ÇDDE algoritmaları için elde edilen en iyi sonuçlar Tablo 7.'de verilmiştir. Sonuçlara bakıldığında Geliştirilmiş GKO ile ÇDDE algoritmalarının amaç fonksiyonlarının sıfırlandığı görülmüştür. Şekil 12.'de de görülebileceği gibi ÇDDE algoritmasının daha hızlı bir şekilde sonuca ulaştığı gözlenmiştir.

Tablo 8. Durum 3 için elde edilen benzetim sonuçları.

Table 8. Simulation results obtained for case 3.

	Geleneksel GKO	Geliştirilmiş GKO	ÇDDE
θ_1	-1.136e-15	1.090e-15	-6.132e-14
θ_2	1.120e-14	-8.729e-15	-3.052e-14
θ_3	0	0	6.854e-13
θ_4	-28.409	-170.772	-4.852
θ_5	-71.222	120	26.650
θ_6	360	6.435	-165.478
Süre(sn)	4651.523	4555.162	18.050
F	0	0	0



Şekil 13. Durum 3 için benzetim süreleri karşılaştırılması.

Figure 13. Comparison of simulation times for case 3.

Durum 3 için [-50, -905, 315] konum değerleri seçilmiştir. Geleneksel GKO, Geliştirilmiş GKO ve ÇDDE algoritmaları için elde edilen en iyi sonuçlar Tablo 8.'de verilmiştir. Sonuçlara bakıldığında tüm algoritmalar için amaç fonksiyonlarının sıfırlandığı görülmüştür. Fakat ÇDDE algoritmasının diğer algoritmalara göre daha hızlı bir şekilde iterasyonlarını tamamladığı gözlenmiştir. Şekil 13. incelendiğinde yine ÇDDE algoritmasının daha hızlı olduğu görülmüştür.

4. Sonuçlar

Bu çalışmada Mitsubishi Melfa RV-7FL-D robot manipülatörün ters kinematik analizi yapılmıştır. Belirtilen konumlar için Geleneksel GKO, Geliştirilmiş GKO ve ÇDDE algoritmaları kullanılarak farklı iterasyon değerlerine göre eklem açıları hesaplanmıştır. Alınan sonuçlar neticesinde ÇDDE optimizasyon algoritmasının Geleneksel GKO ve Geliştirilmiş GKO algoritmasına göre iki iterasyon değerinde de çok daha hızlı iterasyonlarını tamamladığı görülmüştür. Düşük iterasyon değeri için ÇDDE algoritması diğer algoritmalara göre daha kötü sonuç vermiştir fakat çok daha hızlı olması avantajları arasındadır. Benzer şekilde yüksek iterasyon değerlerinde ÇDDE algoritmasının amaç fonksiyonu değerlerinin tüm konumlar için 0'a ulaştığı gözlemlenmiştir. İterasyon sayısı düşük tutulduğunda Geleneksel GKO algoritması ile ÇDDE ve Geliştirilmiş GKO'ya göre daha iyi amaç fonksiyon değeri alınmıştır. İterasyon sayısını arttırıldığında ise ÇDDE ve Geliştirilmiş GKO algoritmalarının Geleneksel GKO'ya göre daha iyi sonuç verdiği, hızlı bir şekilde amaç foksiyonunun minimuma indirdiği görülmüştür.

Kısaca söylemek gerekirse, ters kinematik analizinde Newton-Raphson gibi sayısal ve analitik yöntemler sıklıkla kullanılırken eklem sayısının artması ve işlemlerin daha karmaşık hale gelmesi, çözüme ulaşma sürecini zorlaştırması ve zaman alması sebebiyle akıllı algoritmalara başvurulmuştur. Optimizasyon algoritmaları farklı sistemlerde farklı sonuçlar verebilmektedir. Bir optimizasyon algoritması bir sistemde çok iyi sonuç verebilirken başka bir sistemde iyi sonuç vermeyebilir. Ters kinematik analizi sisteminde kullanılan GKO ve ÇDDE algoritması karşılaştırılmasında ÇDDE'nin sonuç verme süresinin daha kısa olması ve mutlak hatayı sıfırlaması ile GKO algoritmasına üstün geldiği görülmüştür.

Literatür taraması sonucunda elde edilen bilgilere göre optimizasyon algoritmalarının gerçek zamanlı robot manipülasyonlarında kullanıldığı görülmüştür. Gelecek çalışmada bu çalışmada elde edilen verilerin gerçek zamanlı robot konfigürasyonu üzerinde denenmesi ve analizinin yapılması hedeflenmektedir.

Etik kurul onayı ve çıkar çatışması beyanı

Hazırlanan makalede etik kurul izni alınmasına gerek yoktur.

Hazırlanan makalede herhangi bir kişi/kurum ile çıkar çatışması bulunmamaktadır.

Kaynaklar

[1] Zhang Q., Zhang Y., Gao L. and Cao H. 2021. Analysis of the effect of objective function on the performance of the algorithm for the inverse kinematic of manipulator, 2021 14th International Symposium on Computational Intelligence and Design (ISCID), 11-12 December, Hangzhou, China, 91-95.

[2] Yiyang L., Xi J., Hongfei B., Zhining W. and Liangliang S. 2021. A General Robot Inverse Kinematics Solution Method Based on Improved PSO Algorithm, IEEE Access, vol. 9, pp. 32341-32350. DOI: 10.1109/ACCESS.2021.3059714.

[3] Alkayyali M. and Tutunji T. A. 2019. PSO-based Algorithm for Inverse Kinematics Solution of Robotic Arm Manipulators, 2019 20th International Conference on Research and Education in Mechatronics (REM), 23-24 May, Wels, Austria, 1-6.

[4] Dereli S. and Köker R. 2020. Calculation Of The Inverse Kinematics Solution Of The 7-DOF Redundant Robot Manipulator By The Firefly Algorithm And Statistical Analysis Of The Results In Terms Of Speed And Accuracy, Inverse Probl. Sci. Eng., vol. 28(5), pp. 601-613. DOI: 10.1080/17415977.2019.1602124

[5] Ram R.V., Pathak P.M. and Junco S.J. 2019. Inverse kinematics of mobile manipulator using bidirectional particle swarm optimization by manipulator decoupling, Mech. Mach. Theory, vol. 131, pp. 385-405. DOI: 10.1016/j.mechmachtheory.2018.09.022

[6] Dereli S. and Köker R. 2018. IW-PSO Approach To The Inverse Kinematics Problem Solution Of A 7-DOF Serial Robot Manipulator, Sigma J. Eng. Nat. Sci., vol. 36(1), pp. 77-85.

[7] Marić F., Giamou M., Hall A. W., Khoubyarian S., Petrović I. and Kelly J. 2022. Riemannian Optimization for Distance-Geometric Inverse Kinematics. IEEE Transactions on Robotics, vol. 38(3), pp. 1703-1722. DOI: 10.1109/TRO.2021.3123841.

[8] Dereli S. 2021. A new modified grey wolf optimization algorithm proposal for a fundamental engineering problem in robotics, Neural Computing and Applications., vol. 33, pp. 14119–14131. DOI: 10.1007/s00521-021-06050-2

[9] Amiri M.S. and Ramli R. 2021. Intelligent Trajectory Tracking Behavior of a Multi-Joint Robotic Arm via Genetic–Swarm Optimization for the Inverse Kinematic Solution, Sensors, vol. 21(9), pp. 3171. DOI: 10.3390/s21093171

[10] Özen F., Tukel D. and Dimirovski G. 2017. Synchronized dancing of an industrial manipulator and humans with arbitrary music, Acta Polytech. Hung., vol. 14(2), pp. 151-169. DOI: 10.12700/APH.14.2.2017.2.8

[11] K. Zhao, Y. Liu and K. Hu, 2022. Optimal Pattern Synthesis of Array Antennas Using Improved Grey Wolf Algorithm, 2022 IEEE 12th International Conference on Electronics Information and Emergency Communication (ICEIEC), 15-17 July, Beijing, China, 172-175.

[12] Gai W., Qu C., Liu J. and Zhang J. 2018. An improved grey wolf algorithm for global optimization, 2018 Chinese Control And Decision Conference (CCDC), pp. 2494-2498, Shenyang, China, DOI: 10.1109/CCDC.2018.8407544.

[13] Nadimi-Shahraki M. H., Taghian S., Mirjalili S. and Faris H. 2020. MTDE: An effective multi-trial vector-based differential evolution algorithm and its applications for engineering design problems, Appl. Soft Comput., vol. 97(A), pp. 106761. DOI: 10.1016/j.asoc.2020.106761