Gazi Üniversitesi Fen Bilimleri Dergisi
Part:C, Tasarım Ve Teknoloji
GU J Sci Part:C
4(4):225-233 (2016)

Göç Eden Kuşlar Algoritmasinda Kaos Fonksiyonlarinin Kullanilmasi

Dindar ÖZ [1,♠]

[1]*Yaşar Üniversitesi, Selçuk Yaşar Kampüsü, Üniversite Caddesi No:37-39Ağaçlı Yol, Bornova, İzmir*

Başvuru: 15/06/2016 Düzeltme: 04/11/2016 Kabul: 07/11/2016

ÖZ

Olasılıksal eniyileme algoritmaları çalışmalarının birçok aşamasında rastlantısal veri kullanmaktadırlar ve performansları büyük oranda bu rastlantısal verinin dağılımına göre değişiklik göstermektedir. Bu noktadan hareketle farklı rastlantısal veri kaynaklarının eniyileme algoritmalarının performansına etkisi son zamanlardaki birçok çalışmanın odak noktası olmuştur. Kaotik eşlem fonksiyonları matematiksel özellikleri sonucu rastlantısal veri kaynağı olarak kullanılmaya oldukça elverişlidir. Bu çalışmada kaotik eşlem fonksiyonlarının popülasyon tabanlı evrimsel bir algoritma olan göç eden kuşlar algoritmasına etkisi bilgisayar mimarisinin güncel problemlerinden biri olan görev dağıtım problemi üzerinde deneysel olarak incelenmiştir. Deneyler neticesinde bir kısım kaotik eşlem fonksiyonlarının ele alınan problem için uygun olmadığı gözlense de, klasik rastlantısal veri üretme algoritmaları ile başa baş performans sergileyen kaotik eşlem fonksiyonlarının da bulunduğu görülmüştür.

Anahtar Kelimeler: *Olasılıksal Eniyileme, Kaos Teorisi , Görev Dağıtım Problemi*

Use Of Chaos Functions in Migrating Birds Optimization Algorithm

ABSTRACT

Stochastic optimization algorithms use randomly generated data heavily in various steps. The form of this randomly generated data affects the performance of stochastic optimization algorithms significantly. Therefore, the effect of different random data sources on the performance of optimization algorithms is a common focus of many recent studies. Thanks to their mathematical properties, chaotic map functions are very convenient for being used as random data sources. In this work, an empirical analysis is provided in order to show the effect of chaotic map functions on the performance of migrating birds optimization algorithm. This empirical analysis is based on the task allocation problem which is a recent computer architecture problem. According to our experimental results, it is observed that some chaotic map functions perform inefficiently. On the other hand, it is also observed that there are some particular chaotic map functions that can compete with the classical random data generators.

Keywords: *Stochastic Optimization, Chaos Theory, Task Allocation Problem*

♠*İletişim yazarı, e-mail: dindar.oz@yasar.edu.tr*

1 . GİRİŞ (INTRODUCTION)

Evrimsel algoritmaların yakınsama performansının algoritmanın birçok aşamasında yeralan ve rastlantısal veri ihtiyacı için kullanılan rastlantısal veri dizilerine göre ciddi oranda değişiklik gösterdiği görülmüştür [1]. Henüz analitik olarak evrimsel algoritmaların performansını artırmayı garanti eden bir rastlantısal veri dizisi gösterilememiş olsa da, birçok kaotik eşlem fonksiyonunun rastlantısal veri kaynağı olarak kullanımının sanki-rastlantısal (pseudorandom) dizilere göre daha başarılı sonuçlar verdiği deneysel olarak gösterilmiştir. Kaos, başlangıç koşullarına fazlaca duyarlı, doğrusal olmayan sistemlerde sınırsız periyodik bir hareket içeren ve dinamik ve kararsız bir davranışı ifade eder [2]. Rastlantısal bir görüntü sergilemesine karşın, deterministik koşullar altında deterministik doğrusal olmayan bir sistemdir. Literatürdeki birçok kaotik eşlem fonksiyonu kesinlik (certainty), döngelik (ergodicity), ve rastlantısallık (stochasticity) özelliği gösterir.

Son yıllardaki birçok çalışmada kaotik diziler rastlantısal veri kaynağı olarak önerilmiş ve başarılı sonuçlar elde edilmiştir. Kaotik dizilerin tercih edilmesi bu dizilerin tahmin edilemezliği ve geniş sprektrumlu bir dağılım göstermesi özellikleri ile teorik olarak da desteklenmektedir.Kaotik fonksiyonların klasik rastlantısal veri üreteçleri yerine kullanılmasının neden avantaj sağladığı bu çalışmada ve önceki çalışmalarda da teorik olarak açıklanmamakta yalnızca deneysel sonuçlar sunulmaktadır. Güvenli veri iletimi, iletişim, kriptografi [3,4], tıp [5],Termal Mühendislik [6] kaos tabanlı fonksiyonların kullanıldığı çalışma alanlarından birkaçıdır. Bunlara ilaveten evrimsel algoritmaların performansının iyileştirilmesinde de kaotik eşlem fonksiyonlarından yararlanılmaya başlanmıştır [1, 7, 8,9,10,11]. Bu çalışmanın literatüre önemli bir katkısı, göç eden kuşlar eniyileme algoritmasına kaos eşlem fonksiyonları ile oluşturulmuş bir kaotik dizinin, algoritmanın ihtiyaç duyduğu rastlantısal veri kaynağı olarak entegre edilmesi ve bu iyileştirmenin algoritmanın yakınsama performansına etkisinin bilgisayar mimarisinde güncel ve yaygın olarak çalışılan problemlerden biri olan çok amaçlı görev dağıtım problemi üzerinde deneysel olarak incelenmesidir. Kullanılan kaos fonksiyonları ve bu fonksiyonların algoritmanın hangi aşamalarında ne şekilde kullanıldığı detaylı olarak 3. Bölüm'de anlatılmaktadır.

Bir sonraki bölümde çok amaçlı görev dağıtım problemi literatür taraması ve matematiksel modeli ile biçimsel bir şekilde sunulmaktadır. Bu bölümü göç eden kuşlar eniyileme algoritmasını ve bu algoritmaya uygulanan kaos eşlem fonksiyonları entegrasyonunu açıklayan 3. Bölüm takip etmektedir. Deneysel çalışmanın ve bulguların yeraldığı 4. Bölüm'den sonra makale sonuçların ve ileriki çalışma olanaklarının tartışıldığı 5. Bölüm ile sonlanmaktadır.

2 . ÇOK AMAÇLI GÖREV DAĞITIM PROBLEMİ (MULTIOBJECTIVE TASK ALLOCATION PROBLEM)

Dağıtık sistemler birbirlerine yüksek hızlarla bağlı işlemcilerin bir arada çalışması ile yoğun hesaplama gerektiren paralel uygulamalarda yüksek performans sağlarlar. Bu paralel uygulamalar genellikle görev tabanlı olmakta olup bu görevlerin işlemcilere dağıtılma stratejisi sistemin performansı ve güvenilirliği üzerinde önemli etkiye sahiptir ve bu alanda sıklıkla çalışılan problemlerden biri olan görev dağıtım problemini (TAP-task allocation problem) oluşturmaktadır[12, 13, 14, 15, 16, 17, 18, 19, 20].Görev dağıtım problemi literatürde 1977'deki Stone'un çalışmasıyla [21] başlamak üzere çok sayıda araştırmanın konusu olmuştur. Bu alandaki bazı çalışmalar [22, 23] toplam yürütme ve haberleşme zamanını düşürerek performans eniyilemesini amaçlamıştır. Diğer bir gruptaki araştırmalar [12, 15] ise arıza olasılığını düşürerek sistem güvenirliğini arttırmaya çalışmışlardır. Bunlardan başka [14, 17, 18, 20] gibi bildirilerde iki kriterin de (performans ve güvenirlik) birlikte ele alınmış ve çok amaçlı görev dağıtım problemi için çözüm yöntemleri sunulmuştur.

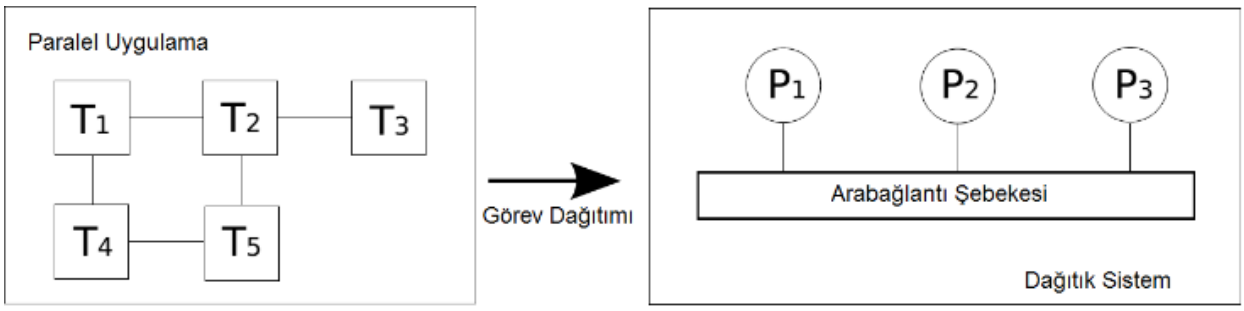
Evrimsel üstsezgisel algoritmalar görev dağıtım problemi için önerilen çözüm yöntemleri arasında önemli bir sınıfı oluşturmaktadır. Bu amaçla Attiya ve arkadaşları [15] benzetimli tavlama (simulated annealing) yöntemini kullandılar. Bu yöntemde probleme özel bir enerji fonksiyonu tanımladılar. Çok amaçlı görev dağıtım problemi için Yin ve arkadaşları hibrit bir parçacık sürüsü eniyileme yöntemi (particle swarm optimization) önerdiler [17]. Ele almış oldukları problem modelinde amaç fonksiyonu olarak hem performans hem de güvenirliği hesaba katan bir fonksiyon kullandılar. Duman ve arkadaşları kendi tasarlamış oldukları göç eden kuşlar eniyileme algoritmasını karesel atama problemi için uyarladılar ve başarılı sonuçlar elde ettiler [16].

Bu çalışmada sistem performansını ve sistem güvenirliğini arttırmayı hedefleyen çok amaçlı bir görev dağıtım problemi ele alınmıştır ve bu problem için popülasyon tabanlı evrimsel bir algoritma olan göç eden kuşlar eniyileme algoritması (migrating birds optimization - MBO) kullanılmıştır [16]. Önerilen yöntemden farklı olarak bu çalışmada, evrimsel eniyileme algoritmalarının temel bileşenlerinden biri olan rastlantısal veri kaynağı olarak kaotik eşlem fonksiyonları kullanılmıştır. Kaotik eşlem fonksiyonlarının çeşitli üstsezgisel eniyileme algoritmalarına etkisi güncel birçok çalışmada incelenmiş ve kaydadeğer sonuçlar alınmıştır. Söz konusu fonksiyonların MBO algoritmasına etkisi ilk kez bu çalışmada ele alınmaktadır.

Problem literatürdeki diğer çalışmalarda [15, 17] yer alan modele uygun olarak formüle edilmiştir.

2.1 . Sistem Modeli (System Model)

Birbirine bir arabağlantı şebekesi ile bağlı N işlemciden (P_1, P_2, ... P_N) oluşan heterojen bir dağıtık hesaplama sistemi olduğunu ve bu sistemde K görevden (T_1, T_2, ...T_K) oluşan paralel bir uygulamanın çalıştırılacağını düşünelim. Bu sistemi kabaca özetleyen bir diyagram Şekil 1'de görüntülenmektedir.



Şekil 1 Heterojen Dağıtık Hesaplama Sistemi
(Heterogenous Distributed Computing System)

Her işlemci (P_n) şu özelliklere sahiptir:

- C_n: Hesaplama iş yükü kapasitesi
- M_n: Hafıza kapasitesi
- λ_n: İşlemci arıza oranı

Her işlemci çifti arasında bir iletişim bağlantısı mevcuttur ve P_n işlemcisi ile P_m işlemcisi arasındaki iletişim bağı şu özelliklere sahiptir:

- DTR_{nm}: Veri transfer hızı
- μ_{nm}: İletişim bağlantısı arıza oranı

Paralel uygulamayı oluşturan görevlerden herbiri (T_k) şu özelliklere sahiptir:

- c_k: Hesaplama gereksinimi
- m_k: Hafıza gereksinimi

Her bir görev ikilisi (T_k, T_l) birbiri ile veri alışverişi yapabilir ve bu iletişim için şu özellik tanımlıdır:

- D_{kl}: Aktarılacak veri miktarı

Bu sistem heterojen bir sistem olduğu için bir görevin tamamlanma zamanı üzerinde çalıştığı işlemciye göre farklılık göstermektedir. T_k görevinin P_n işlemcisinde tamamlanma zamanı ET_{kn} ile gösterilmektedir. Problemin amacı sistem maliyetini en aza indirecek ve sistem güvenirliğini eniyileyecek görev dağıtımını tespit etmektir ve bu dağıtım iki boyutlu X bit matrisi ile ifade edilebilir. Bu matrisin büyüklüğü KxN olur ve X_{kn}= 1 durumu T_k görevinin P_n işlemcisine atandığını ve X_{kn}= 0 durumu bunun tersini ifade eder.

2.2 . Maliyet Fonksiyonu (Cost Function)

Ele alınan problemin amaçları performansı ve güvenirliği arttırmak olduğundan dolayı hedeflenen sistem için hesaplanacak maliyet değeri bu iki amacı karşılayacak şekilde performans ve güvenirlik maliyetleri olarak iki bileşenden oluşmaktadır.

2.2.1 . Performans Maliyeti (Performance Cost)

Sistemin performans maliyeti hesaplama maliyeti ve iletişim maliyeti şeklinde iki alt kısımdan oluşur. Bu anlamda X matrisi ile temsil edilen bir görev dağıtımı için hesaplama maliyeti şu şekilde ifade edilebilir:

$$C_{exec}(X) = \sum_{n=1}^{N}\sum_{k=1}^{K} X_{kn} ET_{kn} \qquad (1)$$

Benzer şekilde, iki işlemci (P_n, P_m) arasındaki toplam iletişim zamanını

$\sum_{k=1}^{K-1}\sum_{l=k+1}^{K} X_{kn}X_{lm}(D_{kl}, DTR_{nm})$şeklinde hesaplayarak bütün işlemciler için toplam iletişim maliyetini şu şekilde hesaplayabiliriz:

$$C_{comm}(X) = \sum_{n=1}^{N-1}\sum_{m>n}^{N}\sum_{k=1}^{K-1}\sum_{l>k}^{K} X_{kn}X_{lm}(D_{kl}, DTR_{nm}) \qquad (2)$$

Bu durumda sistemin toplam performans maliyeti aşağıda ifade edildiği gibi hesaplama ve iletişim maliyetlerinin toplamından oluşmaktadır:

$$C(X) = C_{exec}(X) + C_{exec}(X) = \sum_{n=1}^{N}\sum_{k=1}^{K} X_{kn} ET_{kn} + \sum_{n=1}^{N-1}\sum_{m>n}^{N}\sum_{k=1}^{K-1}\sum_{l>k}^{K} X_{kn}X_{lm}(D_{kl}, DTR_{nm}) \qquad (3)$$

2.2.2 . Güvenirlik Maliyeti (Reliability Cost)

Dağıtık sistemin güvenirliği sistemin uygulamayı oluşturan bütün görevleri başarılı bir şekilde tamamlayabilme olasılığı ile ifade edilmektedir. Başka bir deyişle, bütün görevlerin çalıştırılması esnasında herhangi bir işlemcinin yada iletişim ağının arıza yapmama olasılığı olarak hesaplanabilir. Bu anlamda bir işlemcinin (P_n) t zaman aralığınca güvenirliği $e^{-\lambda nt}$ şeklinde hesaplanır ve verilen bir X görev dağıtımı için P_n işlemcisinin güvenirliği $e^{-\lambda n \sum_{k=1}^{K} X_{kn} ET_{kn}}$ formülü ile hesaplanır. Bu durumda sistemdeki bütün işlemcilerin güvenirliği şu şekilde hesaplanabilmektedir:

$$R_{exec}(X) = \prod_{n=1}^{N} e^{-\lambda n \sum_{k=1}^{K} X_{kn} ET_{kn}} \qquad (4)$$

Yine aynı yaklaşımla P_n ve P_m işlemcileri arasındaki iletişim bağlantısının güvenirliği $e^{-\mu nm \sum_{k=1}^{K-1}\sum_{l=k+1}^{K} X_{km}X_{ln}(D_{kl}/DTR_{nm})}$formülü ile hesaplanır. Bu durumda arabağlantı şebekesindeki bütün iletişim bağlantılarının güvenirliği şu şekilde hesaplanabilir:

$$R_{comm}(X) = \prod_{n=1}^{N-1}\prod_{m>n}^{N} e^{-\mu nm \sum_{k=1}^{K-1}\sum_{l=k+1}^{K} X_{km}X_{ln}(D_{kl}/DTR_{nm})} \qquad (5)$$

Sistemin toplam güvenirliği iletişim ve hesaplama güvenirliklerinin çarpımı ile bulunabilir:

$$R(X) = R_{exec}(X) \times R_{comm}(X)$$
$$= \prod_{n=1}^{N} e^{-\lambda n \sum_{k=1}^{K} X_{kn} ET_{kn}} \times \prod_{n=1}^{N-1} \prod_{m>n}^{N} e^{-\mu nm \sum_{k=1}^{K-1} \sum_{l=k+1}^{K} X_{km} X_{ln} (D_{kl}/DTR_{nm})} \quad (6)$$

Sistem güvenirliği sitem güvenirlik maliyeti cinsinden $R(X) = e^{-RC(X)}$ şeklinde ifade edişlebileceğinden dolayı sistemin güvenirlik maliyeti şöyle tanımlanabilir:

$$RC(X) = \sum_{n=1}^{N} \sum_{k=1}^{K} \lambda n X_{kn} ET_{kn} + \sum_{n=1}^{N-1} \sum_{m>n}^{N} \sum_{k=1}^{K-1} \sum_{l>k}^{K} \mu nm X_{kn} X_{lm} (D_{kl}, DTR_{nm}) \quad (7)$$

Bu çalışmada sistemin amaç fonksiyonu olarak Denklem (1) ve Denklem (7) gösterilen performans ve güvenirlik maliyetleri birlikte kullanılmıştır. Bu durumda ele alınan çok amaçlı görev dağıtım probleminin matematiksel formülasyonu şu şekilde yapılabilir:,

$$\min Z(x) = C(X) + RC(X) \quad (8)$$

$Kısıtlamalar$

$$\sum_{n=1}^{N} X_{kn} = 1 \; \forall k = 1, 2, .. K \quad (9)$$

$$\sum_{k=1}^{K} m_k X_{kn} \leq M_n \; \forall n = 1, 2, .. N \quad (10)$$

$$\sum_{k=1}^{K} c_k X_{kn} \leq C_n \; \forall n = 1, 2, .. N \quad (11)$$

$$X_{kn} \in \{0, 1\} \; \forall k, n \quad (12)$$

Denklem (8) birleştirilmiş maliyet fonksiyonunu ifade etmektedir ve bu çalışmada bu değerin en aza indirilmesi hedeflenmektedir. Denklem(9)'da ifade edilen kısıtlama her bir görevin bir ve yalnız bir işlemciye atanmasını garanti eder. Denklem (10) ve (11) işlemcilerin hafıza ve hesaplama kapasitelerine ilişkin kısıtlamalardır. Son olarak Denklem (12) *X* matrisindeki elemanların ikili sayı sisteminden bir değer almasını garanti etmektedir.

Bu problem modeli bir 0-1 karesel programlama problemi modelidir ve NP-Zor problem sınıfındadır. Bundan dolayı geniş ölçekli sistemler için problemin optimal çözümünü bulmaya çalışmak uygulanabilir bir yöntem değildir.

3 . ÖNERİLEN YÖNTEM (PROPOSED METHOD)

Bu çalışmada görev dağıtımı probleminin çözümü için göç eden kuşlar eniyileme algoritması yöntemi kullanılmıştır. Daha önceki çalışmalardan farklı olarak algoritmanın rastlantısal veri ihtiyacı duyduğu çeşitli aşamalarında rastlantısal veri kaynağı olarak kaos eşlem fonksiyonları tarafından üretilen diziler kullanılmış ve bu dizilerin algoritma performansına etkisi incelenmiştir. Bir sonraki bölümde göç eden kuşlar eniyileme algoritmasının (MBO) kısa bir özeti yeralmaktadır.

3.1 . MBO Algoritması (MBO Algorithm)

Göç eden kuşlar eniyileme algoritması Duman ve arkadaşları tarafından [16], tabiattaki göçmen kuş sürülerinin davranışlarından esinlenerek önerilmiş bir üstsezgisel algoritmadır ve karesel atama problemlerinde başarılı olduğu güncel çalışmalar ile gösterilmiştir [16, 19]. Popülasyon tabanlı evrimsel bir yöntem olan algoritmada, her çözüm adayı birbirini V düzeninde takip etmekte olan bir sürü içerisindeki bir kuş olarak düşünülmekte ve bu anlamda her kuş (çözüm adayı) önündeki kuşlardan istifade etmektedir. Algoritma 1'de gösterilmekte olan MBO algoritması rastlantısal olarak üretilmiş çözüm adayları ile çalışmaya başlar. Bu çözüm adayları bir lider ve onu takip eden ve çevresini keşfederek lider çözüm adayını geliştirmeye çalışan takipçi adaylar olarak organize olurlar. Algoritma takipçi çözüm adaylarının lideri ve komşu takipçileri kullanarak kendi durumlarını geliştirmeye çalışması ile yinelemeli olarak devam eder. Her bir yinelemenin sonunda lider çözüm adayı grubun sonuna geçer ve yeni bir lider belirlenir. MBO algoritmasının aşamaları şu şekilde sıralanmaktadır:

- **İlklendirme:** ilk çözüm adaylarının üretildiği ve algoritma parametrelerinin belirlendiği aşamadır. Bu parametreler sürüdeki kuş sayısı (*n*), bakılacak komşu çözüm adayı sayısı (*k*), bir sonraki çözüm adayına aktarılacak komşu çözüm adayı sayısı (*x*) ve algoritmanın yineleme sayısı (*m*) olarak sıralanmaktadır. Duman ve arkadaşları derledikleri deneyler neticesinde bu parametreler için en başarılı değerleri şu şekilde tespit etmişlerdir:
n = 51, k = 3, x = 1, m = 10

- **Lideri Geliştirme:** Lider çözüm adayı kendini geliştirmek için *k* adet komşu çözüm adayı üretir ve bu adayı hedef fonksiyonuna göre değerini hesaplar. Üretilen adaylardan biri liderden daha iyi ise lider kendini daha iyi olan çözüm adayı olarak günceller. Geriye kalan komşu çözüm adaylarından en iyi *x* tanesi sürünün diğer bireylerine..paylaştırılır.

Algoritma 1 MBO algoritması[16].

```
1: n adet çözüm adayı üret
2: while sonlanmadi do
3:     for tur = (1 to m) do
4:         Lider için k komşu çözüm adayı üret
5:         if bir komşu aday liderden daha iyi then
6:             Lideri güncelle
7:         end if
8:         x adet lider komşusu adayı ilk takipçiye aktar
```

```
9:          for sürüdeki her bir takipçi do
10:             k-x adet komşu üret.
11:             x adet komşuyu bir önceki takipçiden al
12:             if bir komşu daha iyi then
13:                 takipçiyi güncelle
14:             end if
15:         end for
16:     end for
17:     Lideri sürü sonuna at. Yeni lider belirle
18: end while
```

- **Takipçileri Geliştirme:** Takipçi çözüm adayları sıra ile şu şekilde bir işlem sırası takip eder: Aday kendini geliştirmek için *k-x* adet komşu çözüm adayı üretir. Üretilen bu çözüm adaylarına bir önceki geliştirme işleminden gelen *x* çözüm adayı eklenir. Bu *k* adayın hedef fonksiyonuna göre en iyisi takipçi çözüm adayından daha iyiyse takipçi aday kendini en iyi aday olarak günceller. Geriye kalan komşu adaylardan en iyi *x* tanesi bir sonraki takipçi çözüm adayına aktarılır.

- **Lider Değişimi:** Liderin ve takipçilerin kendilerini geliştirdikleri aşamaların her birine bir tur denir ve *m* tur yinelendikten sonra lider çözüm adayı sürünün en arkasına geçer ve bir sonraki takipçi yeni lider olur. Liderin arkasına geçeceği kanat iki kanadında aynı oranda paylaşımda bulunması amacı ile sıra ile bir sağ bir sol olacak şekilde belirlenir.

3.2 . Çözüm Adayı Veri Yapısı (Solution Representation)

Diğer çalışmalarda da olduğu gibi [15, 17, 18] çözüm adayı veri yapısı olarak *K* (problemdeki görev sayısı) uzunluğunda bir tamsayı vektörü, $P=(p_1, p_2, \ldots p_k)^T$, kullanılmıştır. Bu vektörün her bir elemanı o elemanın temsil ettiği görevin hangi işlemciye atandığını ifade etmektedir. Örneğin 4 görevin ve 3 işlemcinin bulunduğu bir sistemde (1,2,1,3) vektörü 1.ve 3. görevlerin 1. işlemciye, 2. görevin 2. işlemciye ve 4. görevin 3. işlemciye atandığı çözümü temsil etmektedir. Bu veri yapısının kullanımı ile Denklem (9) ve (12)'de ifade edilen problem kısıtlamalarına uyulması garanti edilmiş olmaktadır. Yine bu sayede çözüm adayları hafızada oldukça küçük bir alan tutmakta ve çözüm adayından çözüme geçme ve çözümden çözüm adayına geçme (encode/decode) işlemleri çok hızlı gerçekleşebilmektedir.

3.3 . Komşu Çözüm Adayı Üretme Fonksiyonu (Neighboring Function)

Bu çalışmada yine diğer çalışmalarda kullanılan [15, 17, 18] basit bir komşu çözüm adayı üretme fonksiyonu kullanılmıştır. Bu fonksiyonda rastlantısal olarak bir görev seçilir ve bu görevin atanmış olduğu işlemci yine rastlantısal olarak değiştirilir.

3.4 . Hedef Fonksiyonunun Hesaplanması (Calculation of the Objective Function)

Her bir çözüm adayının elverişlilik(fitness) değerlerinin hesaplanması için Denklem (8) kullanılabilir. Ancak sadece Denklem (8)'in kullanımı algoritmanın işlemciler için tanımlı olan hesaplama ve hafıza kapasitesi kısıtlamalarına uygun çözüm üretmesi için yeterli değildir. Bu kısıtlamaları çiğneyen çözüm adaylarının hedef fonksiyonuna göre düşük elverişlilikte çıkması için bu çözüm adaylarına bir ceza değeri uygulanmaktadır. Bu ceza değeri kısıtlamaların çiğnenme miktarına göre değişmekte olup şu şekilde hesaplanmaktadır:

$$P(X) = \sum_{n=1}^{N} (\max(0, \sum_{i|\, p[i]=n} m_i - M_n) + \max(0, \sum_{i|\, p[i]=n} c_i - C_n)) \quad (13)$$

Bu ceza uygulamasının neticesinde bir çözüm adayının hedef fonksiyonu değeri şu şekilde hesaplanır:

$$F(X) = Z(X) + \omega P(x) \quad (14)$$

Denklem (14)'te geçen ω katsayısı, bir başka çalışmada olduğu gibi [13], ceza değerini maliyet değerlerine uygun bir şekilde ölçeklendirmek amacı ile kullanılmaktadır.

3.5 . Kaos Fonksiyonları Entegrasyonu (Chaos Functions Integration)

Göç eden kuşlar algoritması başlıca iki farklı noktada rastlantısal veriye ihtiyaç duymaktadır. Bunlardan ilki başlangıç çözüm adaylarının oluşturulması aşaması olup görevlerin işlemcilere atanması rastlantısal olarak gerçekleştirilmektedir. İkinci nokta ise Bölüm 3.3'te açıklanan komşu çözüm adayı üretme fonksiyonudur. Bu iki aşamada da rastlantısal veri sağlayıcı olarak yaygın olarak kullanılan bir rastlantısal veri üretme fonksiyonunun yanı sıra kaotik eşlem fonksiyonları da kullanılmış ve bu fonksiyonların algoritmanın yakınsama performansına etkisi incelenmiştir. Bu amaçla kaotik eşlem fonksiyonları ile beslenen rastlantısal sayı üreteçleri gerçeklenerek algoritmanın içerisinde kullanılması sağlanmıştır. Gerçeklenen rastlantısal sayı üreteçlerinin çalışma şekliAlgoritma 2'deki gösterilmektedir.Üreteçler rastlantısal olarak üretilen *seed* değerleri ile ilklendirilmekte ve MBO algoritmasında rastlantısal veri ihtiyacı duyulan noktalarda Algoritma2'de yer alan *next()* yordamı kullanılmaktadır.

Algoritma 2 Rastlantısal sayı üreteci.

1: **function**init(seed)
2: X_0 = seed mod 1
3: $Y_0 = X_0$sdsd*// İki parametreli bir kaos fonksiyonu ise*
4: **end-function**
5: **function** next()
6: */* X ve Y kullanılan kaos eşlem fonksiyonu ile hesaplanır */*
7: $X_{n+1} = Fx(X_n, Y_n)$
8: $Y_{n+1} = Fy(X_n, Y_n)$*// İki parametreli bir kaos fonksiyonu ise*
9: **return** X_{n+1} mod 1
10: **end-function**

Bu çalışmada gerçeklenen ve performansı incelenen kaotik eşlem fonksiyonları şunlardır:

3.5.1 . Lozi Eşlem Fonksiyonu (Lozi Map Function)

Henon eşlem fonksiyonuna benzer bir davranış gösteren iki boyutlu parçalı doğrusal bir eşlem fonksiyonudur [24]. Denklem (15) ve Denklem (16)'da görüldüğü gibi tanımının oldukça basit olması gerçeklemesini kolay kılmaktadır.

$$X_{n+1} = 1 - a|X_n| + bY_n \quad (15)$$

$$Y_{n+1} = X_n \quad (16)$$

Lozi *a* ve *b* değerleri için 1.7 ve 0.5 değerlerini önermiştir [24].

3.5.2 . Tent Eşlem Fonksiyonu (Tent Map Function)

Bir çadır şeklini andırmasından dolayı bu isimi almıştır. Aşağıdaki eşitlik ile üretilmektedir [24] .

$$X_{n+1} = \begin{cases} X_n/0.7, & X_n < 0.7 \\ 10/(3X_n(1 - X_n)), & X_n \geq 0.7 \end{cases} \quad (17)$$

3.5.3 . Arnold'un Kedisi Eşlem Fonksiyonu (Arnold's Cat Map Function)

Vladimir Arnold'un 1960'larda bir kedinin görüntüsünü kullanarak oluşturduğu ve kaotik olduğunu gösterdiği fonksiyondur [25]. Fonksiyon Denklem (18) ve Denklem (19)'da gösterilmektedir.

$$X_{n+1} = X_n + Y_n (mod\ 1) \quad (18)$$

$$Y_{n+1} = X_n + 2Y_n (mod\ 1) \quad (19)$$

3.5.4 . Burgers Eşlem Fonksiyonu (Burgers Map Function)

Burgers'ın [26] hidrodinamik akışlardaki çatallanmalar üzerine yaptığı çalışmalar sırasında oluşturduğu kaotik eşlem fonksiyonudur. Denklem (20) ve Denklem (21)'deki gibidir.

$$X_{n+1} = aX_n + Y_n^2 (mod\ 1) \quad (20)$$

$$Y_{n+1} = bY_n + X_n Y_n (mod\ 1) \quad (21)$$

Denklemlerde geçen *a* değeri için 0.75 ve *b* değeri için 1.75 değerleri, X_0için -0.1 ve Y_0için 0.1 değerleri kullanılmıştır [11].

3.5.5 . Sinai Eşlem Fonksiyonu (Sinai Map Function)

Sinai eşlem fonksiyonu aşağıdaki denklem ile temsil edilmektedir (9). Bu denklem *a=1* durumunda (1,0) aralığında kaotik bir seri üretmektedir.

$$X_{n+1} = X_n + Y_n + a\cos(2\pi Y_n) (mod\ 1) \quad (22)$$

$$Y_{n+1} = X_n + 2Y_n (mod\ 1) \quad (23)$$

4 . DENEY ÇALIŞMALARI (EXPERIMENTAL WORK)

4.1 . Deneylerin Organizasyonu (Experimental Setup)

Bu çalışmada çeşitli kaos eşlem fonksiyonlarının MBO algoritmasının rastlantısal veri kaynağı olarak kullanılmasının algoritma performansına etkisi çok amaçlı görev dağıtım problemi üzerinde incelenmiştir. Çok amaçlı görev dağıtım problemi için kabul görmüş bir karşılaştırma veri kümesi (benchmark) bulunmadığı için deneyde kullanılan problem kümesi rastgele olarak üretilmiştir. Problem üretme yöntemi olarak bu alandaki diğer bir çalışmada kullanılan yöntem kullanılmıştır[13]. Üretilen problemlerin parametreleri Tablo 1'de verilmiştir.

Tablo 1: Problem parametreleri.(Problem paramaters)

Parametre Adı	Değeri
Görev sayısı(*K*)	20, 30 ,40
İşlemci sayısı(*N*)	8
Görev etkileşim yoğunluğu(*D*)	0.3, 0.5, 0.8
İletişim/Hesaplama Oranı(*CCR*)	0.5, 1.0, 2.0

Tablo 1'de *D* ile gösterilen görev etkileşim yoğunluğu veri alışverişi yapan görev ikilisi sayısının tüm görev ikilileri sayısına oranını ifade etmektedir. Yine aynı tablodaki *CCR* ile gösterilen iletişim/hesaplama oranı ise görevler arası ortalama iletişim miktarının görevlerin ortalama hesaplama miktarına oranını ifade etmektedir.

Tablo 1'de yer alan her parametre grubu için (3x1x3x3 =27 grup) 10 farklı problem oluşturularak toplamda 270 farklı problem den oluşan bir küme elde edilmiştir. Bunun yanı sıra MBO algoritması 1 rastgele sayı üreteci ve Bölüm 3.5'te listelenen 5 farklı kaotik eşlemfonksiyonu ile toplamda 6 farklı rastlantısal veri kaynağı ile incelenmiştir. Rastgele sayı üreteci olarak akademik çalışmalarda yaygın olarak kullanılmakta olan

Mersenne-Twister üreteci kullanılmıştır. MBO algoritması her bir rastlantısal veri kaynağı ile her problem için 10 kez yinelemeli olarak çalıştırılmış ve bu yinelemeler üzerinden ortalama alınarak algoritma performansı hesaplanmıştır. Toplamda 6x270x10 = 16200 adet deney tamamlanmıştır. Deneyler Intel I7 2.60 GHz işlemci ve 8Gb hafıza bulunan bir makine üzerinde çalıştırılmıştır.

Görev dağıtım problemi bir karesel atama problemi olduğu için deneylerde çalışan MBO algoritması parametreleri olarak Duman ve arkadaşlarının karesel atama problemleri için belirlemiş olduğu ideal parametre değerleri olan*n = 51, k = 3, x = 1, m = 10*değerlerikullanılmıştır. Ayrıca problem modelinde yer alan ölçeklendirme katsayısı (ω) yapılan deneyler

Tablo 2. Farklı rastlantısal veri kaynakları ile MBO algoritması performans karşılaştırması
(The comparison of MBO algorithm performance with different random data sources)

Problem Senaryosu			Algoritma Performansı					
K	D	CCR	MersenneTwister	Lozi	Burgers	Arnold	Tent	Sinai
20	0.3	0.5	**57,53**	69,39	70,11	57,67	112,03	58,34
		1.0	**64,27**	76,66	81,22	64,30	116,46	67,62
		2.0	**85,11**	103,34	107,98	86,67	146,58	89,10
	0.5	0.5	70,65	84,83	89,36	**70,29**	126,06	70,93
		1.0	86,72	97,53	102,66	**85,88**	136,31	88,35
		2.0	121,62	136,52	141,28	**121,04**	185,47	128,23
	0.8	0.5	87,64	99,51	102,09	**87,32**	135,56	84,41
		1.0	119,21	133,86	135,49	**118,20**	177,43	115,52
		2.0	**163,04**	175,12	183,06	163,20	230,28	175,66
30	0.3	0.5	**84,54**	101,28	105,11	84,77	153,12	83,71
		1.0	112,00	127,68	132,87	**111,16**	182,24	108,90
		2.0	150,59	173,19	176,31	**149,91**	237,78	155,56
	0.5	0.5	**113,07**	129,47	132,77	113,49	170,83	108,62
		1.0	**140,02**	159,35	161,65	140,19	205,61	138,38
		2.0	**193,58**	209,43	218,88	195,82	280,34	201,13
	0.8	0.5	**144,87**	162,84	167,68	145,12	207,33	136,45
		1.0	217,74	233,34	243,68	**214,77**	274,42	202,99
		2.0	**263,06**	281,27	286,29	265,07	368,81	291,14
40	0.3	0.5	130,04	153,03	154,79	**129,36**	221,20	126,51
		1.0	167,50	191,92	200,37	167,84	263,03	161,54
		2.0	223,48	252,66	256,13	**222,55**	342,22	234,60
	0.5	0.5	166,50	190,05	197,46	**166,06**	246,38	157,31
		1.0	**237,08**	269,90	278,53	239,06	342,60	237,92
		2.0	316,36	346,52	356,36	**316,16**	423,90	327,38
	0.8	0.5	**237,87**	265,21	271,06	238,21	330,52	228,03
		1.0	322,72	346,65	360,12	**323,03**	438,20	294,56
		2.0	504,28	532,64	551,13	**502,23**	643,03	526,63
Genel			168,43	187,73	193,65	**168,37**	247,48	170,35

neticesinde 100 olarak belirlenmiştir. Algoritmanın sonlandırma kriteri olarak 30000 döngü sayısı kullanılmıştır. Bu sayının üretilen problemlerde algoritmanın bir çözüme yakınsaması için yeterli olduğu gözlemlenmiştir.

4.2 . Deney Sonuçları (Experimental Results)

Tablo 2'de farklı rastlantısal veri kaynaklarının MBO algoritmasının çoklu görev dağıtım problemi üzerindeki performansına etkileri görüntülenmektedir. Performansı ölçütü olarak algoritmanın ulaşmış olduğu en iyi çözümün elverişlilik değeri (fitness) kullanılmıştır. Her satır farklı bir deney konfigürasyonunu temsil etmekte ve o konfigürasyondaki en başarılı algoritma performansı kalın olarak gösterilmektedir. Bu sonuçlara göre Lozi, Burgers, Tent ve Sinai kaotik eşlem fonksiyonları görev dağıtım probleminde algoritma performansını olumsuz olarak etkilemektedirler. Ancak Arnold kaotik eşlem fonksiyonu yaygın bir rastlantısal veri üreteci olan Mersenne-Twister ile başa baş bir performans sergilemekte ve deneylerin genelinde algoritma performansını iyileştirdiği görülmektedir.Bu çalışmada ele alınan kaos fonksiyonları performansları ile Mersenne-Twister üreteci performansının ikili t-testi sonuçları (p-değeri) Tablo-3'te görüntülenmektedir. Bu sonuçlar deneylerde en başarılı sonuç veren Arnold kaos fonksiyonu ile Mersenne-Twister sonuçları arasında istatistiksel olarak kayda değer bir fark bulunmadığını (>0.05) göstermektedir.

Tablo 3. Mersenne-Twister ve kaos fonksiyonları performansı t-testi sonuçları (T-test results for Mersenne-Twister and chaos functions)

Karşılaştırılan İkili	p-değeri
Mersenne Twister-Lozi	1,256E-108
Mersenne Twister-Burgers	1,382E-116
Mersenne Twister-Arnold	0,386
Mersenne Twister-Tent	1,138E-103
Mersenne Twister-Sinai	0,423

5 . SONUÇ (CONCLUSION)

Kaotik eşlem fonksiyonları kesinlik, döngelik ve rastlantısallık gibi sahip oldukları matematiksel özelliklerinden dolayı rastlantısal veri kaynağı olarak kullanılmak için oldukça iyi bir alternatiftir. Bu çalışmada bu fonksiyonların güncel olasılıksal eniyileme algoritmalarından biri olan göç eden kuşlar algoritmasında(MBO) rastlantısal veri kaynağı olarak kullanılmasının etkisi deneysel olarak incelenmiştir. Algoritma performansı dağıtık sistemlerde sıklıkla karşılaşılan NP-Zor bir problem olan görev dağıtım problemi üzerinde ele alınmış, 5 farklı eşlem fonksiyonu yaygın bir klasik rastlantısal veri üreteci ile karşılaştırılmıştır. Deneyler bazı kaos fonksiyonlarının klasik üreteçler ile rekabet edebilir düzeyde iyi sonuçlar verdiğini göstermiş ve sonraki çalışmalarda bu fonksiyonların göç eden kuşlar algoritmasına etkisinin diğer problem kümeleri içinde incelenmesi gerektiğini göstermiştir.

KAYNAKLAR (REFERENCES)

[1] Copponetto R., Fazzino S. ve Xibillia M. G., "Chaotic Sequences to Improve the Performance of Evolutionary Algorithms", IEEE Transactions On Evolutionary Computation, Cilt 7, No 3, 289-304, 2003.

[2] Schuster H. G., "Deterministic Chaos", Wiley, New York, 1988.

[3] Shabika A., Hooshmandasl M. ve Meybodi M., "Cryptanalysis of Multiplicative Coupled Cryptosystems Based on the Chebyshev Polynomials", Int J Bifurc Chaos, Cilt:26, No 7,1650112, 2016.

[4] Farajallah M., El Assad S. ve Deforges O., "Fast and Secure Chaos-Based Cryptosystem for Images", Int J Bifurc Chaos, Cilt:26, No 2, 1650021, 2016.

[5] Tang B., Xiao Y., Tang S. ve Cheke R., "A Feedback Control Model of Comprehensive Therapy for Treating Immunogenic Tumours", Int J Bifurc Chaos, Cilt:26, No 3, 1650039, 2016.

[6] Mariani V., Duck A., Guerra F., Coelho L., Rao R., "A chaotic quantum-behavedparticle swarm approach applied to optimization of heat exchangers", Appl. Therm. Eng., Cilt 42, 119-128, 2012.

[7] Senkerik R., Pluhacek M., Davendra D., Zelinka I., Kominkova Z., "Chaos driven evolutionary algorithm: a new approach for evolutionaryoptimization", Int. J. Mathematics and Computers in Simulation, Cilt 7, No 3, 363-368, 2013

[8] Liu D. ve Cao Y.A.,"Chaotic genetic algorithm for fuzzy grid job scheduling" IEEE International Conferenceon Computational Intelligence and Security, Guangzhou,Cilt 1, 320–323, Kasım 2006.

[9] Alataş B., Akın E. ve Özer B., "Chaos embedded particle swarm optimization algorithms", Chaos, Solitons & Fractals; Cilt 40, 1715–1734, 2009.

[10] Pluhacek M., Senkerik R. ve Davendra D.,"Chaos particle swarm optimization with Eensemble of chaotic systems", Swarm and Evolutionary Computation, Cilt 25, 29-35, 2015.

[11] Metlicka M. ve Davendra D. "Chaos driven discrete artificial bee algorithm for location and assignment optimisation problems", Swarm and Evolutionary Computation, Cilt 25, 15-28, 2015.

[12] Shatz, S., Wang, J.P. ve Goto, M.,"Task allocation for maximizing reliability of distributed computer systems" IEEE Transactions on Computers, Cilt 41, 1156–1168, 1992.

[13] Kang Q., He H. ve Deng R., "Bi-objective task assignment in heterogeneous distributed systems using honeybee mating optimization", Applied Mathematics and Computation, Cilt 219, 2589–2600, 2012.

[14] Salman A., Ahmad I. ve Al-Madani, S.,"Particle swarm optimization for task assignment problem", Microprocessors and Microsystems, Cilt 26, 363 - 371, 2002.

[15] Attiya G. ve Hamam Y.,"Task allocation for maximizing reliability of distributed systems: A simulated annealing approach", Journal of Parallel and Distributed Computing, Cilt 66, 1259–1266, 2006.

[16] Duman E., Uysal M. ve Alkaya A. F.,"Migrating birds optimization: Anew metaheuristic approach and its performance on quadratic assignment problem", Information Sciences, Cilt 217, 65–77, 2012.

[17] Yin P. Y., Yu S.-S., Wang P. ve Wang, Y. T,"Multi-objective task allocationin distributed computing systems by hybrid particle swarm optimization", AppliedMathematics and Computation, Cilt 184, 407–420, 2007.

[18] Lin S.W., Ying K.C. ve Huang C.-Y.,"Multiprocessor task schedulingin multistage hybrid flowshops: A hybrid artificial bee colony algorithm with bidirectional planning", Computers and Operations Research, Cilt 40, 1186–1195, 2013.

[19] Pan Q.K. ve Dong Y.,"An improved migrating birds optimisation for a hybridflowshop scheduling with total flowtime minimisation",Information Sciences, Cilt 277,643–655, 2014.

[20] Pendharkar P. C.,"An ant colony optimization heuristic for constrained taskallocation problem", Journal of Computational Science, Cilt 7, 37–47, 2015.

[21] Stone H. S., "Multiprocessor scheduling with the aid of network flow algorithms", IEEE Transactions Software Engineering, Cilt 3, 85–93, 1977.

[22] Hansen J. V. ve Giauque W. C.,"Task allocation in distributed processing systems", Operations Research Letters, Cilt 5, 137-143, 1986

[23] Chen, W.-H., & Lin, C.-S.,"A hybrid heuristic to solve a task allocation problem", Computers & Operations Research, Cilt 27, 287-303, 2000.

[24] Peitgen H., Jurgens H., Saupe D.,"Chaos and fractals New Frontiers of Science", Springer-Verlag, Berlin, 1992.

[25] Peterson G. "Arnold's cat map", Math 45 - Linear algebra,1997.

[26] Burgers J., "Mathematical examples illustrating relations occurring in the theory of turbulent fluid motion", Selected Papers of J. M. Burgers, Springer, Netherlands, 281–334, 1995.