

## Karides Sürüsü Algoritmasının Görüntü Sıkıştırma Kullanılması

Fatma HARMAN<sup>1</sup> İlker KILIÇ<sup>2</sup>

<sup>1,2</sup>Elektrik Elektronik Mühendisliği, Celal Bayar Üniversitesi, Manisa, TURKEY

<sup>1</sup>[fatma.harman@cbu.edu.tr](mailto:fatma.harman@cbu.edu.tr)<sup>2</sup>[ilker.kilic@cbu.edu.tr](mailto:ilker.kilic@cbu.edu.tr)

### Özet

Teknolojinin günden güne gelişmesiyle görüntü sıkıştırmanın hem önemi artmış hem de görüntü sıkıştırma yeni çözümler geliştirilmiştir. Bu anlamda, birçok problemin çözümünde kullanılan sezgi üstü algoritmaların bu alanda kullanımı yaygınlaşmıştır. Bu çalışmada, sezgi üstü algoritmaların genel yapısı ve görüntü sıkıştırma üzerindeki etkisi incelenmiştir. Sezgi üstü algoritmalarından olan Karides Sürüsü Algoritması, Genetik Operatörlü Karides Sürüsü Algoritması ve Parçacık Sürüsü Algoritması standart görüntüye uygulanmıştır. Fakat Karides Sürüsü Algoritmasının genellikle yerel minimum noktasına takıldığı görülmüştür. Bu nedenle, Karides Sürüsü Algoritması Genetik Operatörlerle birleştirilmiş ve çaprazlama ve mutasyon oranları sırasıyla %10 ve %4 olarak belirlenmiştir. Böylece, algoritma yerel minimumdan kurtarılmış ve sıkıştırma ve ortalama karesel hata oranlarında Genetik Operatörsüz Karides Sürüsü Algoritmasına göre daha iyi sonuçlar elde edilmiştir. Yapılan analizlere ek olarak probleme Parçacık Sürüsü Algoritması uygulanmıştır. Algoritmaların çözüm performansı karşılaştırıldığında ise Genetik Operatörlü Karides Sürüsü Algoritmasının Karides Sürüsü Algoritması ile Parçacık Sürüsü Algoritmasına göre aynı sıkıştırma oranında daha etkili ve daha düşük ortalama karesel hataya sahip olduğu görülmüştür.

### 1. Giriş

Teknolojinin gelişmesiyle birlikte bilgiyi hızlı bir şekilde elde etme ve bilgiden yararlanmaya olan gereksinim artmıştır. Bu gereksinim bilginin verimli bir şekilde saklanabilmesi, ulaşıp iletilmesi durumunu beraberinde getirmektedir[1]. Veri sıkıştırma, bant genişliğinin daralması, bellek kapasite değerinin azalmasını sağladığı için veri transferlerinde önemli bir noktaya sahiptir. Görüntü sıkıştırma ise bilginin fark edilebilir bir kaybı olmadan görüntünün iletimi ve depolanması için gerekli olan bit sayısının azalmasıyla ilgili bir tekniktir. Kayıplı ve kayıpsız olmak üzere iki tip görüntü sıkıştırma tekniği vardır. Kayıplı teknikte bilginin kaybı vardır ve belirli bir hata oranında orijinal görüntü elde edilir. Kayıpsız sıkıştırma ise bilginin kaybı yoktur ve orijinal görüntü düşük sıkıştırma oranında tekrar oluşturulur [2].

Günümüzde araştırmacılar, görüntü sıkıştırma görüntü bloklarının nasıl seçildiği ve optimize edildiği, sıkıştırma ve

açma hızının dengelenmesi, sıkıştırma oranı ve sıkıştırma sonrası görüntünün kalitesinin geliştirilmesi üzerine yoğunlaşmaktadır[3].

Son zamanlarda sezgi üstü algoritmaları bilginin minimum kayıpla sıkıştırılması üzerinde kullanılmaktadır. Sezgi üstü algoritmaları, evrimsel, doğadan esinlenen, sürü zekâsı, fiziksel ve diğer doğadan esinlenen algoritmalar olarak sınıflandırılır[4]. Karides Sürüsü Algoritması doğadan esinlenen bir algoritma iken Parçacık Sürüsü Algoritması sürü zekâsı algoritmalarındandır [5].

### 2. Karides Sürüsü Algoritması

Karides sürüsü, beslenme yeteneği, avcılardan korunma, çöğalma ve çevresel şartlara uyum sağlamada önemli bir işlergeye sahiptir [6,7].

Antarktik karidesleri en çok araştırılan deniz hayvanlarından biridir. Bu karideslerin en önemli yeteneklerinden biri büyük sürüler oluşturabilmesidir [7,8,9]. Ancak karides sürüsü popülasyonunun dağılmasına sebep olan birçok belirsizlikler vardır[7,10]. Karides sürüsünün gözlemlenen formunu anlamak için önerilen kavramsal modeller vardır ve bu modellerde elde edilen sonuçlar karides sürülerinin temel birimlerden oluştuğunu gösterir [7,11].

Avcılar, karides sürüsüne her saldırdığında, karides bireylerinin hepsi karides yoğunluğunu azaltma yönünde hareket ederler. Bu saldırıdan sonra karides sürüsünün oluşumu 2 temel amaç içerir. Bunlardan birincisi karides yoğunluğunu artırma, diğeri ise yeme ulaşmadır. Sürü yoğunluğunun artışı ve yiyeceğe göre tüm karides bireyi en iyi olası çözümün olduğu noktaya göre hareket eder [7].

Karides Sürüsü Algoritması ise bu Antarktik karideslerin beslenme davranışlarından esinlenerek ortaya sürülen bir doğadan esinlenen algoritmadır. (Gandomi ve Alavi, 2012).

Karides bireylerinin zamana bağlı durumları 3 temel eylemle gerçekleşir.

- Diğer karideslerin neden olduğu hareket

- Beslenme hareketi
- Fiziksel yayılma

$n$  boyutlu karar uzayında Lagrange modeli,

$$\frac{dX_i}{dt} = Ni + Fi + Di \quad (1)$$

Burada  $Ni$  diğer karideslerin neden olduğu hareketi,  $Fi$  beslenme hareketi,  $Di$  ise  $i$ . karidesin random yayılmasıdır.

### 2.1. Diğer Karideslerin Neden Olduğu Hareket

Her bir karides bireyi için bu hareket şu şekilde ifade edilir.

$$N_i^{\text{yeni}} = N^{\text{maks}} a_i + w_n N_i^{\text{eski}} \quad (2)$$

$$a_i = a_i^{\text{lokal}} + a_i^{\text{hedef}} \quad (3)$$

Burada  $N^{\text{maks}}$  maksimum neden olunan hız, 0.01 (m/s) olarak alınmıştır.  $a_i$  neden olunan hareketin doğrultusu,  $w_n$  [0,1] aralığında neden olunan hareketin atalet ağırlığı,  $N_i^{\text{eski}}$  son neden olunan hareket,  $a_i^{\text{lokal}}$  komşu karideslerin sağladığı lokal etkiler,  $a_i^{\text{hedef}}$  en iyi karides tarafından sağlanan hedef yönün etkisidir. Karidesin tekilliğinden uzaklaşmak için küçük pozitif değer  $\varepsilon$  paydaya eklenmiştir. Karidesin hareketine komşu karidesin etkisi şu şekilde formülize edilir.

$$a_i^{\text{lokal}} = \sum_{j=1}^{NN} K_{i,j} X_{i,j} \quad (4)$$

$$X_{i,j} = \frac{x_j - x_i}{\|x_j - x_i\| + \varepsilon} \quad (5)$$

$$K_{i,j} = \frac{K_i - K_j}{K_{\text{worst}} - K_{\text{best}}} \quad (6)$$

Burada  $K_{\text{worst}}$  ve  $K_{\text{best}}$  karides bireylerinin şimdiye kadarki en iyi ve en kötü uygunluk değeridir.  $K_i$   $i$ . karides bireyinin amaç fonksiyon değerini gösterirken,  $K_j$   $j$ . komşu bireyin amaç fonksiyonu değeridir. Her karides bireyinin fonksiyonu  $X$  iken,  $NN$  toplam komşu sayısını göstermektedir. Komşu seçimi ise hissedilen uzaklık (sensing distance) temel alınarak yapılmaktadır [12].

$$ds_i = \frac{1}{5N} \sum_{j=1}^N \|X_i - X_j\| \quad (7)$$

Burada,  $ds_i$  karides bireyinin hissedilen uzaklığını gösterirken,  $N$  toplam karides sayısını gösterir. Denkleme

göre eğer iki karides bireyi arasındaki uzaklık  $ds$  den küçükse, karideslerin komşu olduğu çıkartılmaktadır. En iyi amaç fonksiyonuna sahip karidesin  $i$ . karides üzerindeki etkisi şu şekildedir.

$$a_i^{\text{hedef}} = C^{\text{best}} \cdot K_{i,\text{best}} \cdot X_{i,\text{best}} \quad (8)$$

$C^{\text{best}}$  etki katsayısıdır.

### 2.2. Beslenme Hareketi

Bu konuyla ilgili iki temel adım önemlidir. Bunlardan biri beslenmek için yemin yeri, diğeri ise önceki tecrübelerdir.  $i$ . karides bireyi için beslenme hareketi şu şekildedir.

$$F_i = V_f \beta_i + w_f F_i^{\text{eski}} \quad (9)$$

$$\beta_i = \beta_i^{\text{yiyecek}} + \beta_i^{\text{best}} \quad (10)$$

$V_f$  beslenme aşamasındaki yiyeceği yemi arama hızıdır ve Price'in çalışmasına dayanarak 0.02 (m/s) olarak alınmıştır.  $w_f$  beslenme hareketinin [0,1] aralığında atalet ağırlığı,  $F_i^{\text{eski}}$  son yiyeceği olan yemi arama hareketi,  $\beta_i^{\text{yem}}$  yiyeceği yemin çekiciliğini ve  $\beta_i^{\text{best}}$   $i$ . karidesin şu zamana kadar ki en iyi amaç fonksiyonunun etkisidir. Her iterasyonda beslenme merkezi şu şekilde güncellenir.

$$X_i^{\text{yiyecek}} = \frac{\sum_{i=1}^N (1/K_i) X_i}{\sum_{i=1}^N 1/K_i} \quad (11)$$

$i$ . karides bireyi için yiyeceği yemin çekiciliği ise

$$\beta_i^{\text{yiyecek}} = C^{\text{yiyecek}} K_{i,\text{yiyecek}} X_{i,\text{yiyecek}} \quad (12)$$

$$C^{\text{yiyecek}} = 2 \left(1 - \frac{I}{I_{\text{max}}}\right) \quad (13)$$

$i$ . karides bireyine ait en iyi amaç fonksiyonu modeli şu şekildedir.

$$\beta_i^{\text{best}} = K_{i,\text{best}} \cdot X_{i,\text{best}} \quad (14)$$

$K_{i,\text{best}}$  önceki gidilmiş en iyi pozisyon değeridir.

### 2.3 Fiziksel Yayılma

Random bir süreç olan fiziksel yayılım şu şekilde modellenmiştir.

$$D_i = D_{maks} \left(1 - \frac{I}{I_{maks}}\right) \delta \quad (15)$$

$D_{maks}$  [0.002, 0.01] (m/s) aralığında maksimum yayılım hızı,  $\delta$  ise [-1,1] aralığında random yönlü vektördür.

### 2.4. Karides Sürüsünün Hareketini Tamamlama Süreci

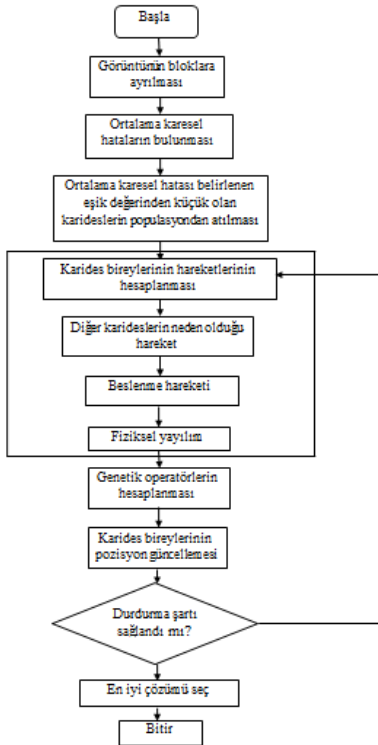
Karides bireyinin  $t$  ve  $t + \Delta t$  zaman aralığında pozisyon vektörü eşitliği şu şekildedir.

$$X_i(t + \Delta t) = X_i(t) + \Delta t \frac{dX_i}{dt} \quad (16)$$

$$\Delta t = C_i \sum_{j=1}^{NV} UB_j - LB_j \quad (17)$$

$NV$  (number of variable) toplam değişken sayısını,  $LB_j$  ve  $UB_j$   $j$ . değişkene ait alt ve üst limitleri,  $C_i$  değeri ise sabit bir değer olup  $\Delta t$  değeri 0.001 olarak alınmıştır.

### 2.5. Akış Diyagramı



Şekil 1. Karides Sürüsü Algoritması Akış Diyagramı

Sürekli girdi uzayını ayrıntılı olarak yansıtırken ortaya çıkan her bir çıkış vektörüne kod kelimesi, bu çıkışlardan oluşan

vektörlerin kümesine ise kod kitabı denir. Belirtilen sayı değeri kadar kod kitabı için kod kelimeleri oluşturulur ve görüntünün her bir bloğu ile kod kelimeleri arasındaki mesafe Öklid uzaklığı ile hesaplanır [1].

$$\text{Öklid uzaklığı: } d(x,y) = |x-y|^2 \quad (18)$$

Popülasyondaki her bir karides kod kitabı ile temsil edilir ve oluşturulan her bir karides sırasıyla zamana bağlı temel adımları gerçekleştirir. Her bir adımda konumunu değiştiren karideslerin belirli adımlarda yerel minimum noktasına takıldığı gözlemlenmiştir. Bu noktalardan kurtarmak için algoritmaya genetik operatörler eklenmiştir. Her iterasyonda karides özellikleri %10 çaprazlama, %4 mutasyon ile güncellenir. Böylece algoritma yerel minimumdan kurtarılır ve görüntü yeniden oluşturulur.

### 3. Parçacık Sürüsü Algoritması

Birçok bilim adamı kuş veya balık sürüsünde organizmaların hareketini çeşitli şekilde yorumlayarak bilgisayar simülasyonunu oluşturdu. Reynolds, Heppner ve Grenander başta olmak üzere kuş sürüsünün simülasyonunu ileri sürdüler [13].

Kuş veya balık sürülerinin sosyal davranışlarından esinlenen bu algoritma, popülasyon temelli sürü zekâsı algoritması olarak geliştirilmiştir (Eberhart ve Kennedy,1995)[14].

Parçacık Sürüsü Algoritması, genetik algoritmalar gibi evrimsel hesaplama tekniği ile benzerlikleri bulunmaktadır. Algoritma random çözümler içeren bir popülasyonla başlatılıp en iyi çözüm için jenerasyonlarını sürekli güncelleyerek arama yapar. Genetiğin aksine çaprazlama ve mutasyon gibi operatörler yoktur. Parçacık Sürüsü Algoritmasında parçacık (kuş) denilen potansiyel çözümler, mevcut olan en iyi çözümü takip ederek çözüm uzayında dolaşırlar (uçarlar)[14].

Parçacık Sürüsü Algoritmasında parçacık kuş ya da balık sürüsünün hareketine benzer hareketler yapar ve her iterasyonda yiyeceğe ne kadar uzaklıkta olduğunu bilip o yiyeceğe en yakın kuşu takip etme mantığına dayanır. Parçacık çözüm uzayındaki kuşlar ve tüm parçacıkların optimize edecek uygunluk değeri ile uçuşları yönlendiren hız bilgileri vardır. Parçacıklar çözüm uzayında mevcut en iyi parçacığı takip eder [14].

Parçacık Sürüsü Algoritması random olarak üretilen parçacıklarla başlayıp, sürekli güncellenerek en uygun değer

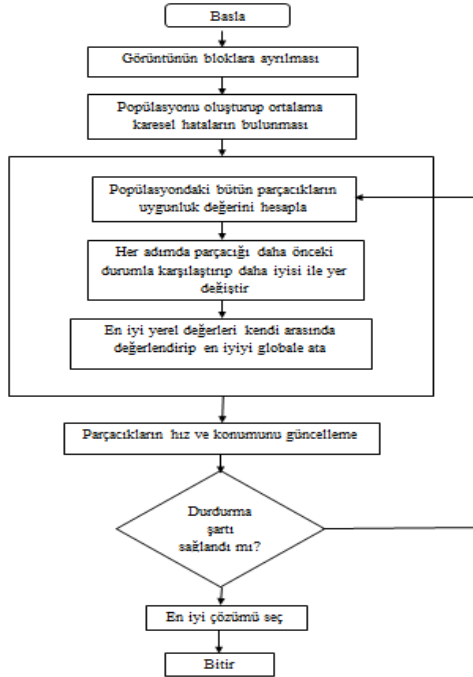
çözüm uzayında araştırılır. Her iterasyonda parçacık iki en iyi değerle güncellenir. Bunlardan birincisi bir parçacığın o ana kadarki en iyi uygunluk değeridir ki bu değer hafızada tutulup *pbest* olarak adlandırılır, ikincisi ise popülasyondaki herhangi bir parçacık tarafından o ana kadar elde edilmiş en iyi uygunluk değeri olup *gbest* olarak adlandırılır.

$$V_i^{k+1} = V_i^k + c_1 \cdot \text{rand}_1^k \cdot (pbest_i^k - x_i^k) + c_2 \cdot \text{rand}_2^k \cdot (gbest^k - x_i^k) \quad (19)$$

$$x_i^{k+1} = x_i^k + V_i^{k+1} \quad (20)$$

*rand* 0 ile 1 arasında üretilen rastgele bir değeri, *i* parçacık numarası, *k* iterasyon sayısıdır. *c*<sub>1</sub> ve *c*<sub>2</sub> öğrenme faktörü olup *pbest* ve *gbest* *i* *c*<sub>1</sub> kendi *c*<sub>2</sub> diğer parçacıklara göre yönlendirir [15].

Parçacık Sürüsü Algoritması Şekil 2 de gösterilmiştir.



Şekil 2. Parçacık Sürüsü Algoritması Akış Diyagramı

Popülasyondaki bütün parçacıkların uygunluk değeri hesaplanır. Algoritmadaki her bir adımda parçacık sürüsünün hız ve konumu güncellenip görüntü yeniden oluşturulur.

Yeniden oluşturulan görüntüde hata tanımı olarak genellikle MSE ve PSNR incelenir.

$$MSE = \sum_{i=1}^N \frac{(X_i - x_i)^2}{N} \quad (21)$$

*X<sub>i</sub>* = Orijinal piksel,

*x<sub>i</sub>* = Tekrar elde edilen piksel,

N = Toplam piksel sayısı

$$PSNR = 10 \log_{10} \left( \frac{(\text{max.gri seviyesi})^2}{MSE} \right) \quad (22)$$

#### 4. Sonuçlar

Lena görüntüsü için 4x4 lük kod kelimeleri ile kuantalanmış görüntününün 100 iterasyon sonucunda ortalama karesel hata değerleri, PSNR değerleri, sıkıştırma oranları Tablo 1.1 görülmektedir. Bu sonuçlara göre Genetik Operatörlü Karides Sürüsü Algoritmasının, Parçacık Sürüsü Algoritmasına ve Karides Sürüsü Algoritmasına göre aynı sıkıştırma oranında daha etkili ve daha düşük ortalama karesel hataya sahip olduğu görülmüştür.

Tablo 1.1 Performans Sonuçları

Lena	KSA	KSA Mutasyon	KSA Çaprazlama	KSA ve Genetik	PSA
Orjinal Boyut	256x256 piksel	256x256 piksel	256x256 piksel	256x256 piksel	256x256 piksel
Kod Kitabı Büyüklüğü	128 vektör	128 vektör	128 vektör	128 vektör	128 vektör
Sıkıştırma Oranı	39:1	39:1	39:1	39:1	39:1
MSE	334.6887	329.205	327.9911	324.9711	326.9881
PSNR	22.8843	22.9561	22.9721	23.01235	22.9854



(a)

(b)

Şekil 3. (a) Orijinal Resim Lena (b) KSA Genetik Operatörsüz



(a)

(b)

Şekil 4. (a) KSA Mutasyon Operatörüyle (b) KSA Çaprazlama Operatörüyle



(a)

(b)

Şekil 5. (a) KSA Genetik Operatörlerle Boyut (b) PSA

Sezgi Üstü Algoritmalarıyla 100 iterasyon sonucu Lena resmine ait test sonuçları Şekil 3, Şekil 4, Şekil 5 'te görülmektedir. Aynı kod kitabı büyüklüğünde ve sıkıştırma oranında Sezgi Üstü Algoritmaların değişimine bağlı olarak görüntüdeki değişim gözlemlenir.

## 5. Kaynaklar

[1] Adıgüzel, V. Okatan, A. Atlı, Av. "Vektör Kuantalama Yöntemi ve Farklı İmgelerde Sonuçları", *emo. org.tr*

[2] Prakash, S.R. Shetty, V.S. "Review On Optimization Techniques Used For Image Compression", *International Journal of Research in Engineering and Technology*, 2015, February, 562-567

[3] Li, M. Ou, S. Zhang, H. "The New Progress In Research Approach of Fractal Image Compression" *Journal of Engineering Graphics*, 2004, 4(3): 143-152

[4] Nanda, S. J., Panda, G." A Survey on Nature Inspired Metaheuristic Algorithms for Partitional Clustering" *Swarm and Evolutionary Computation*, 2014, 1-18

[5] Malik, K. ,Tayal A. "Comparison of Nature Inspired Metaheuristic Algorithms" *International Journal of Electronic and Electrical Engineering*, 2014, 799-802

[6] Singh, G. P, Singh, A. "Comparative Study of Krill Herd, Firefly and Cuckoo Search Algorithms for Unimodal and Multimodal Optimization" *I.J. Intelligent Systems and Applications*, 2014, 03, 35-49

[7] Gandomi, A.H, Alavi A.H, "Krill herd: A New Bio-inspired Optimization Algorithm", *Communications in Nonlinear Science and Numerical Simulation*, 2012, 4831-4845

[8] Hardy, AC., Gunther ER. "The Plankton of the South Georgia Whaling Grounds and Adjacent Waters", *Fishes of Antarctica*, 1998, 61

[9] Watkins, J.L. "Variations In the Size of Antarctic Krill, Euphausia Superba Dana in Small Swarms" *Marine Ecology-Progress Series*, 1986, 67-83

[10] Marr JWS. *The Natural History and Geography of The Antarctic Krill (Euphausia Superba Dana)*. 1962, 32: 33-464.

[11] Nicol S. "Living krill, Zooplankton and Experimental Investigations". *Proceedings of the International Workshop on Understanding Living Krill for Improved Management and Stock Assessment Marine and Freshwater Behaviour and Physiology*, 2003, 191-205.

[12] Gölcük, İ. Baykasoğlu, A, Madenoğlu, S" Kril Sürüsü Algoritması İle Atölye Çizelgeleme" *DEU Mühendislik Fakültesi Mühendislik Bilimleri Dergisi*, Cilt 16 Sayı 48 61-75, EYLÜL 2014

[13] Kennedy, J. Eberhart, R "Particle Swarm Optimization", *Neural Network*, 1995, 1942-1948

[14] Feng, H.M. Chen, C.Y. , Ye, F." Evolutionary Fuzzy Particle Swarm Optimization Vector Quantization Learning Scheme in Image Compression" *Expert Systems with Application*, 2007, 213-222

[15] Neri, F., Mininno, E., Iacca, G. "Compact Particle Swarm Optimization", *Information Sciences*, 2013, 96-121.