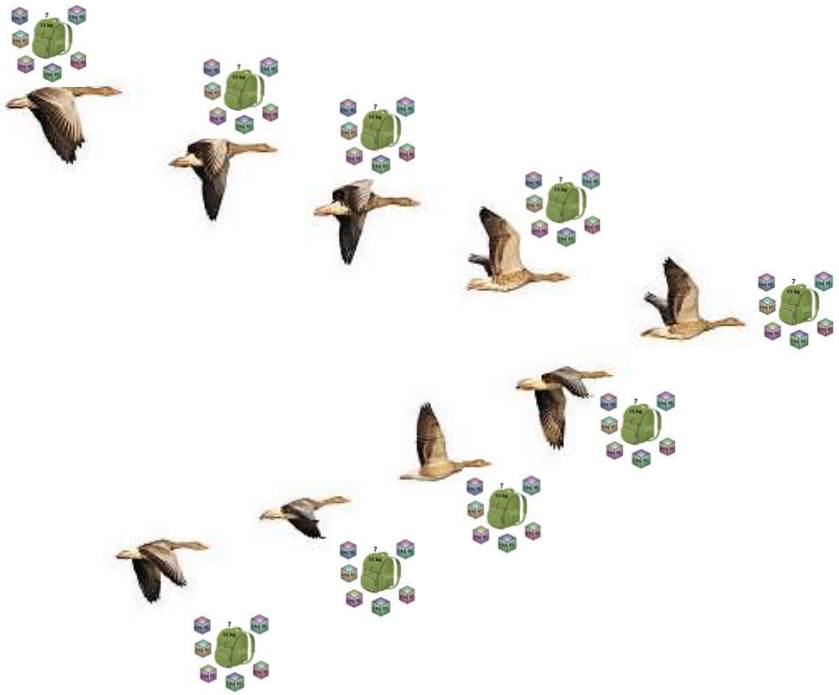


T. C.
Gazi Üniversitesi
Bilişim Enstitüsü
Bilgisayar Bilimleri

OPTİMİZASYONDA METASEZGİSEL YÖNTEMLER

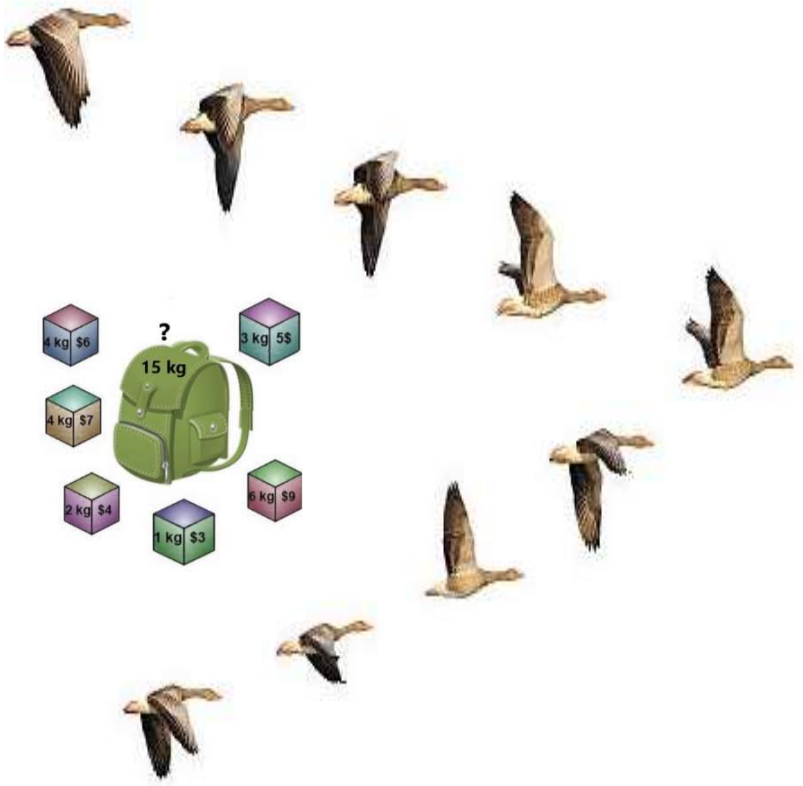
MBO Algoritmasının 0-1 Sırt Çantası Optimizasyon Problemine Uygulanması

İshak Kutlu

Mayıs 2023

0-1 Sırt Çantası (knapsack) Problemi ve Göçmen Kuşlar Optimizasyon (migrating birds optimization, MBO) Algoritması

1. Ülker, E. ve Tongur, V. (2017). Migrating Birds Optimization (MBO) Algorithm to Solve Knapsack Problem. *Procedia Computer Science*, 111(2017), 71-76.
2. Tongur, V. (2018). Ayrık Optimizasyon Problemlerinin Çözümünde Göçmen Kuşlar Optimizasyon (MBO) Algoritmasının İyileştirilmesi. Doktora Tezi, T.C. Selçuk Üniversitesi Fen Bilimleri Enstitüsü.



0-1 Sırt Çantası (knapsack) Probleminin Matematiksel Formu

$$f(x)_{max} = \sum_{i=1}^{d} p_i x_i$$

$$\sum_{i=1}^{d} w_i x_i \leq C$$

$$x_i \in \{0,1\},\ i = 1, 2, \ldots, d$$

d: Problemin boyutu/nesne sayısı

p_i: Her boyutun/nesnenin fayda değeri

w_i: Her nesnenin ağırlığı/maliyeti

C: Sırt çantasının maksimum kapasitesi

f(x): Amaç fonksiyonu

Çözüm Temsili: Doğrusal, ikili (binary) perm.

Problem Tipi: Ayrık, kombinatoryel maks.

Alfabe: {0,1}

Arama Uzayı: 2^d

0-1 Sırt Çantası (knapsack) Problemi İçin Basit Bir Örnek

w_i = {2, 5, 4, 3, 1}, p_i = {5, 10, 15, 10, 20}, d = 5, C = 10

x_i = [1, 1, 0, 0, 1]

$$f(x)_{max} = \sum_{i=1}^{d} p_i x_i = 5x1 + 10x1 + 15x0 + 10x0 + 20x1 = 35$$

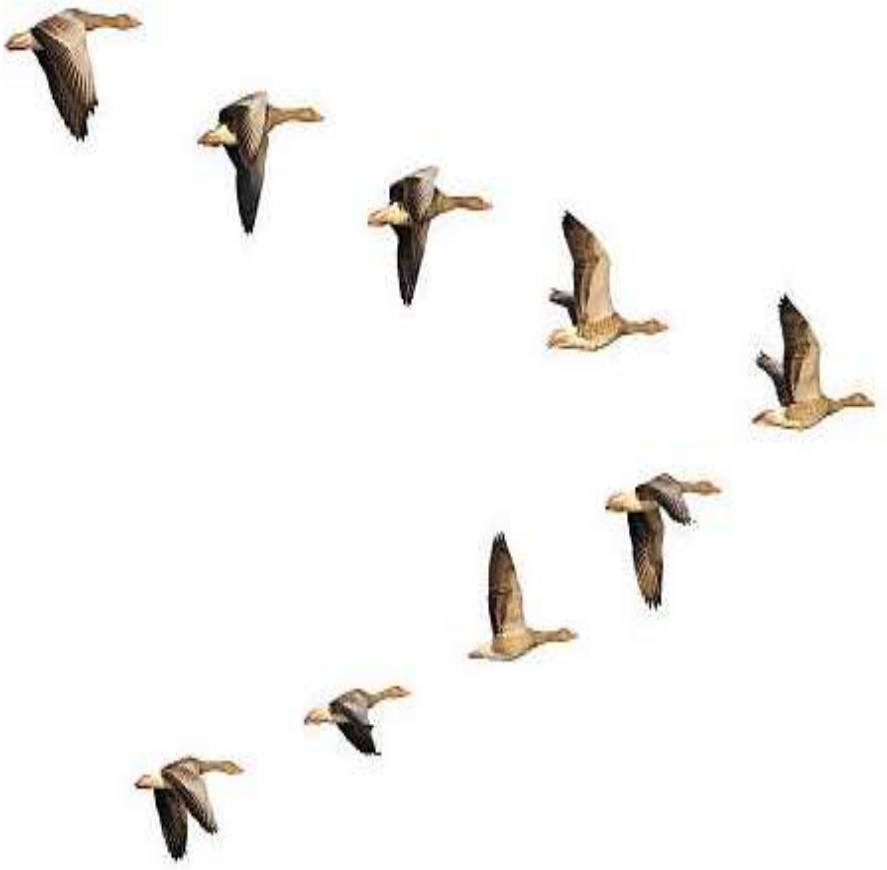
$$\sum_{i=1}^{d} w_i x_i \leq C = 2x1 + 5x1 + 4x0 + 3x0 + 1x1 = 8$$

Sırt Çantası (knapsack) Problemleri

1. 0-1 (zero-one) sırt çantası problemi
2. Sınırlandırılmış (bounded) sırt çantası problemi
3. Kesirli (fractional) sırt çantası problemi
4. …

MBO Algoritmasına Hızlı Bir Bakış

1. 2012 yılında Duman ve arkadaşları tarafından göçmen “V” diziliminden sağladıkları enerji tasarrufundan esinlenerek geliştirilmiştir.
2. MBO, ayrık problemlerden biri olan karesel atama problemi (quadratic assignment problem, QAP) için tasarlanmıştır.
3. Kombinatoryal optimizasyon problemi 0-1 sırt çantası çözümünde kullanılabilir.



MBO Algoritmasının Temel Yapısı ve İşleyişi-1

1. Popülasyondaki birey sayısı kadar rassal bir başlangıç çözümüyle başlatılır.
2. Komşuluk yapısı ve fayda aktarım mekanizmasıyla her adımda çözümler geliştirilmeye çalışılır.
3. Komşuluk, her kuşun/bireyin mevcut çözümden (ona yakın) yeni çözümler üretmesidir.
4. **Fayda aktarımıyla** her kuş/birey kendisi için kullanmadığı komşu çözümlerinden en iyilerini sürüdeki diğer kuşlarla paylaşır.
5. Sürüdeki her kuş etkileşim içindedir; çözüme daha hızlı ulaşılabilir.

MBO Algoritmasının Temel Yapısı ve İşleyişi-2

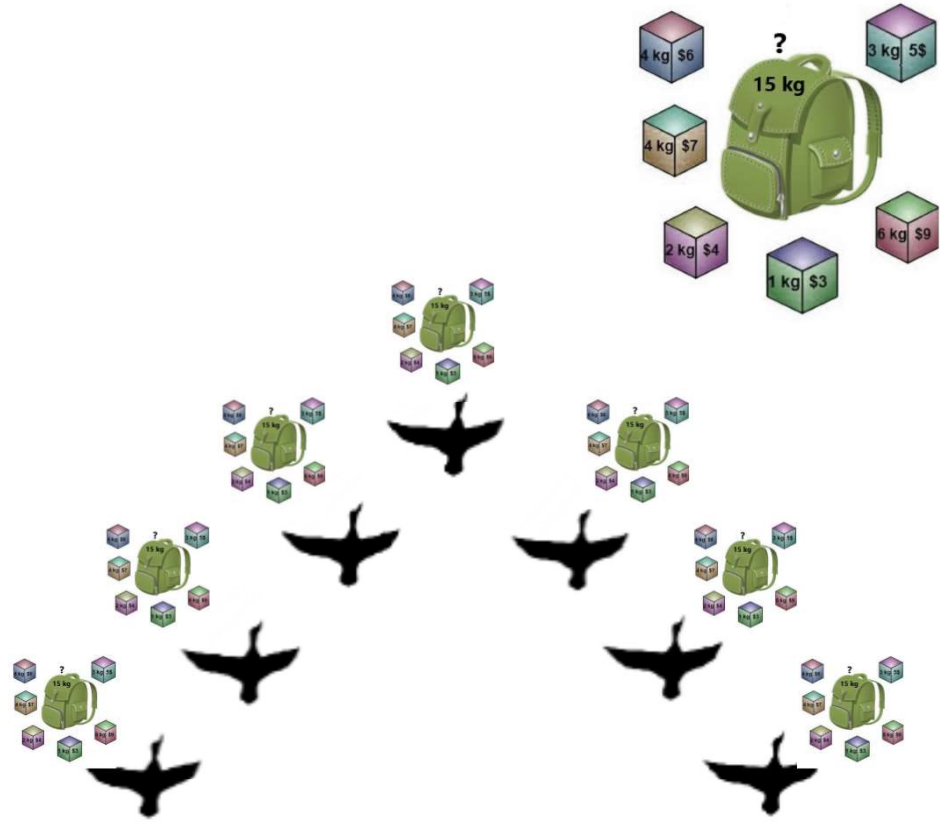
6. Her iterasyonda komşu çözüm üretme ve paylaşma, liderin kanat çırpma sayısı kadardır.
7. Lider kanat çırpma sayısına ulaştığında, ilk olarak sürünün **sol** kanadının en sonuna gönderilir. Sol kanattan lideri takip eden kuş lider olur. Kanat çırpma parametresi sıfırlanır.
8. Yeni lider kanat çırpma sayısına ulaştığında, artık sürünün **sağ** kanadının en sonuna gönderilir. Sağ kanattan lideri takip eden kuş lider olur. Kanat çırpma parametresi tekrar sıfırlanır.
9. İterasyon, sol ve sağ kanatlar üzerinde algoritma sonlandırılıncaya kadar devam eder.

MBO Algoritmasının Parametreleri

Parametre	Açıklama
p	Popülasyonun başlangıç çözüm sayısı
k	Liderin komşu çözüm sayısı
n	Lider dışındaki her bireyin komşu çözüm sayısı
x	Komşu çözümlerin paylaşım sayısı
m	Kanat çırpma sayısı
d	Durdurma kriteri olarak problemin boyutu

Popülasyon Büyüklüğü

1. Popülasyondaki birey/kuş sayısıdır. Sabit bir parametredir.
2. Popülasyon büyüklüğü en az üç olmak üzere 3, 5, 7, 9 gibi tek sayılardan oluşur. Aksi durumda ‘V’ formasyonu oluşmaz.
3. Biri lider, kalanlar ise yarı yarıya sol ve sağ kanadı oluşturur.
4. Her bireyin ikili (binary) bir permütasyon çözümü var.
5. Her birey arama uzayında bir çözümü temsil eder.
6. Permütasyonun uzunluğu, problemin boyutuna eşittir.
7. Her bireyin başlangıç çözümü, kapasiteyi aşmayacak şekilde, rassal olarak belirlenir.



Komşuluk ve Paylaşım Yapısı-1

1. Mevcut çözüme yakın farklı yeni çözümlere komşu çözümler denir.
2. Komşu çözüm üretme yöntemleri takas (swap), araya ekleme (insertion) ya da hibrit olabilir.
3. Lider (kuş) en az üç komşu çözüm üretir.
4. En iyisi kendisine ayrılır.
5. Kalanlar ise sol ve sağ kanattan kendisine takip eden bireylerle paylaşılır.
6. Lider dışındaki bireylerin ürettiği komşu çözüm sayısı, paylaşımdan aldıkları çözüm sayısı kadar az olur.
7. Böylece (lider dahil) tüm bireylerin komşu çözüm sayısı eşit olur.

Komşuluk ve Paylaşım Yapısı-2

$s \in N^+; k = 2s+1; k = \{3, 5, 7, ...\}$

$x \in N^+; 1 \leq x \leq \left(\frac{k-1}{2}\right); x = \{1, 2, 3, ..., (k-1)/2\}$

$n = k - x$

k: Liderin ürettiği komşu çözüm sayısı (sabit bir parametre)

x: Paylaşılan komşu çözüm sayısı (sabit bir parametre)

n: Lider dışında kalan bireylerin ürettiği komşu çözüm sayısı (sabit bir parametre)

k ifadesi aynı zamanda her bireyin sahip olduğu (paylaşımdan gelen + üretilen) komşu çözüm sayısını ifade eder (n+x=k).

Komşuluk ve Paylaşım Metodu-1

$w_i = \{2, 5, 4, 3, 1\}$, $p_i = \{5, 10, 15, 10, 20\}$, $d = 5$, $C = 10$

0	1	0	1	0		0	1	1	1	0		0	1	1	0	0
Adım 1						Adım 2						Adım 3				

ağırlıkların toplamı (1x5+1x3=8) ağırlıkların toplamı (1x5+1x4+1x3=12) ağırlıkların toplamı (1x5+1x4=9)

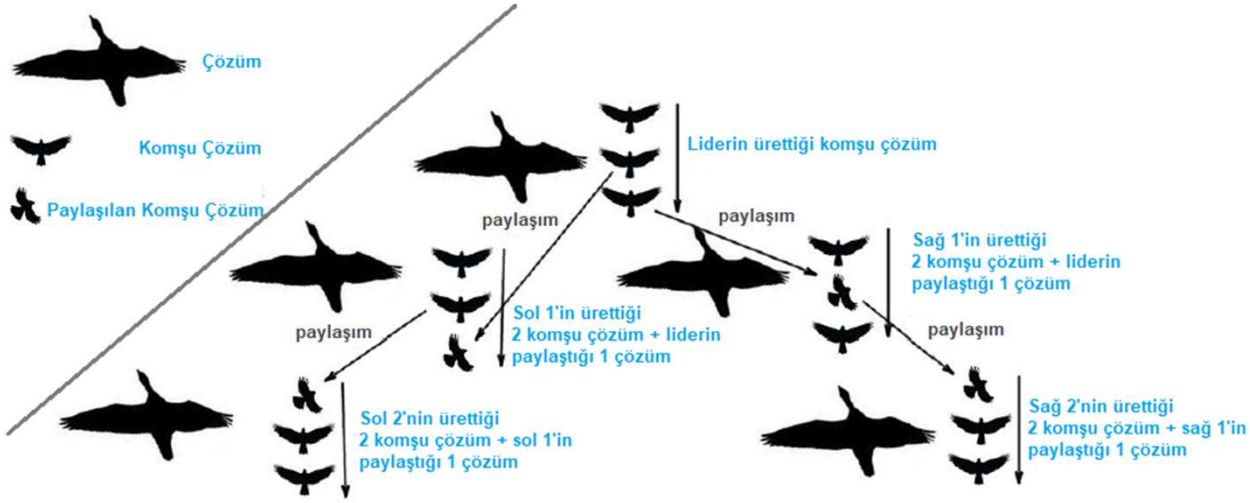
1. Her bireyin komşu çözüm üretme yöntemi aynıdır.
2. Değeri 0 olan bir alan rassal şekilde seçilir ve 1 değeri atanır.
3. Kapasite aşılırsa, bu defa değeri 1 olan bir alan rassal olarak seçilir ve 0 olarak değiştirilir.
4. Üretilen komşu çözüm amaç (objective) fonksiyonuna göre en iyiden kötüye doğru sıralanır.
5. En iyi komşu çözüm, mevcut çözümü iyileştirmek için ayrılır.
6. Kalan x'er adet komşu çözüm, lideri soldan ve sağdan takip eden iki bireyle sol-sağ-sol-sağ şeklinde paylaşılır. Diğer bireyler ise komşu çözümlerini sadece kendi tarafları ile paylaşır.

Komşuluk ve Paylaşım Metodu-2

Başlangıç popülasyonu p=5,

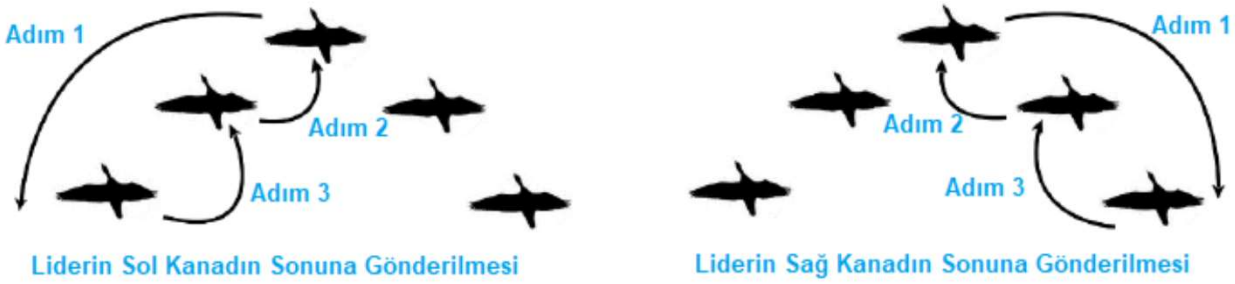
Komşu çözüm sayısı k=3,

Paylaşılan çözüm sayısı x=1



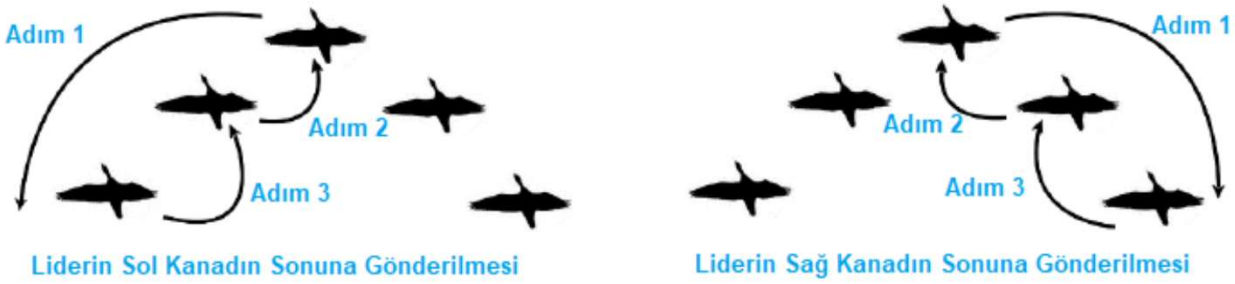
1. k=3 iken, liderin her iki taraftan kendisini takip eden bireylerle paylaştığı komşu çözüm sayısı x=1 (birer adet) olacaktır.
2. Liderin kendisine ayırdığı en iyi çözüm dışında kalan 2 komşu çözümün x=1'i, sol taraftan lideri takip eden bireyle; kalan x=1'i ise sağ taraftan lideri takip eden diğer bireyle paylaşılır.
3. Sürünün sol ve sağ tarafın en sonunda bulunan bireyler, kendi komşu çözümlerini diğer bireylerle paylaşamaz.

Kanat Çırpma Parametresi ve Sol Kanatta Lider Değişimi



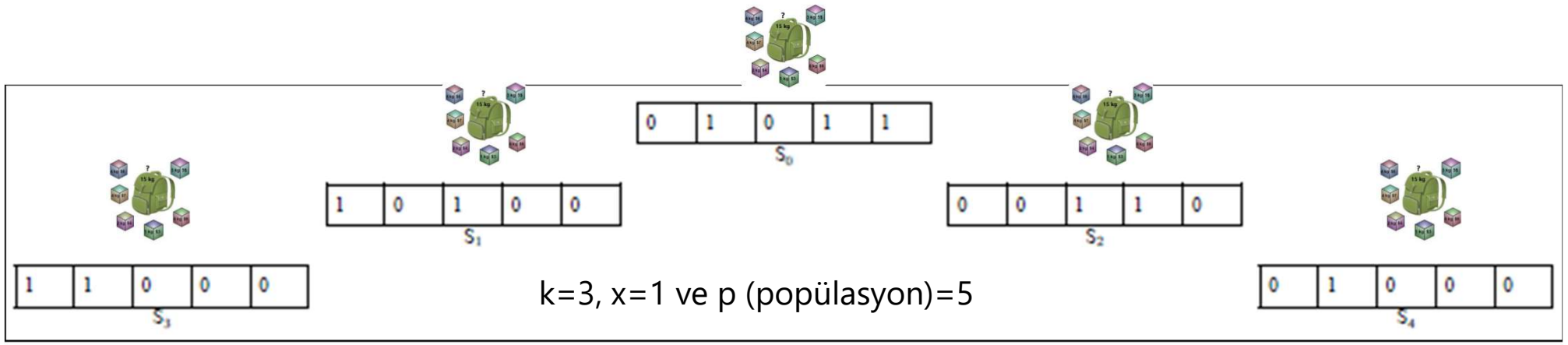
1. Kanat çırpma parametresi, sürünün liderinin (ya da diziliminin) ne sıklıkta değiştirileceğini ifade eder. 15 ya da 30 kanat çırpmada bir sürünün liderinin, dolayısıyla diziliminin değişmesi gibi.
2. Sürünün lider değişimi, önce sol tarafta gerçekleştirilir.
3. Lider sol kanadın en arkasına gönderilir.
4. Lideri soldan takip eden birey yeni lider olur.
5. Zincirleme bir şekilde aradaki boşluklar doldurulur.
6. Kanat çırpma parametresi sıfırlanır.

Kanat Çırpma Parametresi ve Sağ Kanatta Lider Değişimi



1. Sürünün lider değişimi, sonraki aşamada sağ tarafta gerçekleştirilir.
2. Lider sağ kanadın en arkasına gönderilir.
3. Lideri sağdan takip eden birey yeni lider olur.
4. Zincirleme bir şekilde aradaki boşluklar doldurulur.
5. Kanat çırpma parametresi sıfırlanır.
6. Lider değişimi, sol-sağ-sol sırasıyla, yinelemeli bir şekilde algoritmanın çalışması sonlandırılıncaya kadar devam ettirilir.
7. Algoritmanın durdurma kriteri problemin boyutuna ya da belirli bir kalitedeki çözümün sağlanmasın göre belirlenebilir.

MBO ve Örnek Bir Sırt Çantası Problemi



$w_i = \{2,5,7,3,1\}$, $p_i = \{10,30,70,50,1\}$, $d = 5$, $C = 10$

MBO ve Örnek Bir Sırt Çantası Problemi (devam-1)

1. N ifadeleri, her bireyin sahip olduğu komşu çözümleri temsil eder.
2. N'ler kendi içlerinde en iyiden kötüye doğru sıralanmıştır.
3. Her birey sahip olduğu komşu çözümlerden en iyisini (yani birincisi sıradakini) kendi çözümünü geliştirmek için kullanır.
4. Diğerini (yani ikincisini) ise kendisinden sonra gelen birey ile paylaşır.
5. Temsili örnekte x=1 olduğu için her bireyin en kötü (yani üçüncü) komşu çözümü atılır.

Neighbor solutions of each bird for $k=3$ and $x=1$.

Bird	Permutation	Fitness (Sum of Profit)	Neighbor	Permutation	Fitness (Sum of Profit)
			N_{01}	00110	80
S_0	01011	81	N_{02}	00101	71
			N_{03}	11001	41
			N_{11}	10101	81
S_1	10100	80	N_{12}	00110	80
			N_{13} (Shared=N_{02})	00101	71
			N_{21}	10010	60
S_2	00110	120	N_{22}	00011	51
			N_{23} (Shared=N_{03})	11001	41
			N_{31} (Shared=N_{12})	00110	80
S_3	11000	40	N_{32}	00010	50
			N_{33}	11001	41
			N_{41}	01010	80
S_4	01000	30	N_{42} (Shared=N_{22})	00011	51
			N_{43}	01001	31

MBO ve Örnek Bir Sırt Çantası Problemi (devam-2)

1. N_{01}, N_{02} ve N_{03} ifadeleri S_0 ile temsil edilen liderin ürettiği komşu çözümlerdir.
2. N_{11} ve N_{12} ifadeleri ise **sol kanatta** yer alan S_1 bireyinin ürettiği komşu çözümlerdir.
3. N_{13} ise liderin (yani S_0'ın) S_1 bireyi ile paylaştığı komşu çözümdür.
4. Benzer şekilde N_{21} ve N_{22} ifadeleri **sağ kanatta** yer alan S_2 bireyinin ürettiği komşu çözümleri, N_{23} ise liderin S_2 bireyi ile paylaştığı komşu çözümü temsil eder.

Neighbor solutions of each bird for *k=3* and *x=1*.

Bird	Permutation	Fitness (Sum of Profit)	Neighbor	Permutation	Fitness (Sum of Profit)
			N_{01}	00110	80
S_0	01011	81	N_{02}	00101	71
			N_{03}	11001	41
			N_{11}	10101	81
S_1	10100	80	N_{12}	00110	80
			N_{13} (Shared=N_{02})	00101	71
			N_{21}	10010	60
S_2	00110	120	N_{22}	00011	51
			N_{23} (Shared=N_{03})	11001	41
			N_{31} (Shared=N_{12})	00110	80
S_3	11000	40	N_{32}	00010	50
			N_{33}	11001	41
			N_{41}	01010	80
S_4	01000	30	N_{42} (Shared=N_{22})	00011	51
			N_{43}	01001	31

MBO ve Örnek Bir Sırt Çantası Problemi (devam-3)

1. N_{32} ve N_{33} ifadeleri **sol kanatta** yer alan S_3 bireyinin ürettiği komşu çözümleridir.
2. N_{31} ise yine sol tarafta yer alan S_1 bireyinin S_3 bireyi ile paylaştığı komşu çözümdür.
3. Benzer şekilde N_{41} ve N_{43} ifadeleri **sağ kanatta** yer alan S_4 bireyinin ürettiği komşu çözümleri, N_{42} ise yine sağ kanatta yer alan S_2 bireyinin S_4 bireyi ile paylaştığı komşu çözümü temsil eder.

Neighbor solutions of each bird for *k=3* and *x=1*.

Bird	Permutation	Fitness (Sum of Profit)	Neighbor	Permutation	Fitness (Sum of Profit)
			N_{01}	00110	80
S_0	01011	81	N_{02}	00101	71
			N_{03}	11001	41
			N_{11}	10101	81
S_1	10100	80	N_{12}	00110	80
			N_{13} (Shared=N_{02})	00101	71
			N_{21}	10010	60
S_2	00110	120	N_{22}	00011	51
			N_{23} (Shared=N_{03})	11001	41
			N_{31} (Shared=N_{12})	00110	80
S_3	11000	40	N_{32}	00010	50
			N_{33}	11001	41
			N_{41}	01010	80
S_4	01000	30	N_{42} (Shared=N_{22})	00011	51
			N_{43}	01001	31

MBO ve Örnek Bir Sırt Çantası Problemi (devam-4)

$f(.)$ amaç fonksiyonu olmak üzere,

$f(S_i) < f(N_{i1})$ ise

$S_i = N_{i1}$ olarak güncellenir.

Tablonun Fitness kolonunda yer alan bireylerin yeni çözümleri:

S_0 = 81, S_1 = 81, S_2 = 120, S_3 = 80 ve S_4 = 80

Neighbor solutions of each bird for *k=3* and *x=1*.

Bird	Permutation	Fitness (Sum of Profit)	Neighbor	Permutation	Fitness (Sum of Profit)
			N_{01}	00110	80
S_0	01011	81	N_{02}	00101	71
			N_{03}	11001	41
			N_{11}	10101	81
S_1	10100	80	N_{12}	00110	80
			N_{13} (Shared=N_{02})	00101	71
			N_{21}	10010	60
S_2	00110	120	N_{22}	00011	51
			N_{23} (Shared=N_{03})	11001	41
			N_{31} (Shared=N_{12})	00110	80
S_3	11000	40	N_{32}	00010	50
			N_{33}	11001	41
			N_{41}	01010	80
S_4	01000	30	N_{42} (Shared=N_{22})	00011	51
			N_{43}	01001	31

Lider ve Diğer Bireyler İçin Algoritmanın Özeti

Lider İçin Algoritmanın Özeti

1. Üretilen komşu çözümlerin uygunluğu (fitness) en iyiden kötüye doğru sıralanır.
2. En iyi (birinci sıradaki) komşu çözüm liderin kendi çözümünü geliştirmede kullanılır.
3. İkinci sıradaki komşu çözüm, sürünün sol tarafında yer alan lideri takip eden birey ile paylaşılır.
4. Üçüncü sıradaki komşu çözüm, sürünün sağ tarafında yer alan lideri takip eden birey ile paylaşılır.

Diğer Bireyler İçin Algoritmanın Özeti

1. Üretilen ve önündeki bireyden alınan komşu çözümlerin uygunluğu (fitness) en iyiden kötüye doğru sıralanır.
2. En iyi (birinci sıradaki) komşu çözüm bireyin kendi çözümünü geliştirmede kullanılır.
3. İkinci sıradaki komşu çözüm, (kendi kanadında) bireyi takip eden diğer birey ile paylaşılır.
4. Üçüncü sıradaki komşu çözüm atılır.

Makalede Uygulanan MBO'nun Parametreleri

Parameter	Value
Bird *(p)*	71
Neighbor (*k*)	5
Flap (*m*)	30
Shared neighbor (*x*)	1
Stop criteria	problem dimension

Deneysel araştırmada kullanılan 10 adet 0-1 sırt çantası problemine aşağıdaki bağlantıdan ulaşılabilir.

http://artemisa.unicauca.edu.co/~johnyortega/instances_01_KP/

Makalenin Deneysel Sonuçları

Algoritma, her problem için 30 kez çalıştırılmış.
Ortalama, en kötü ve en iyi sonuçlar ile tepki süreleri sunulmuş.
8 no.lu problem hariç tüm problemlerin optimal çözümleri bulunmuş.
6 no.lu problemin çözümü optimalin bir miktar üzerinde gerçekleşmiş.

Problem	Optimum Value [9]	Best	Average	Worst	Time(*s*)
P_1	295	**295**	294.6666	293	0.002
P_2	1024	**1024**	1011.1	991	0.003
P_3	35	**35**	35	35	0.0004
P_4	23	**23**	23	23	0.0004
P_5	481.0694	**481.0694**	431.4180	399.8016	0.002
P_6	50	**52**	52	52	0.001
P_7	107	**107**	107	107	0.0008
P_8	9767	9765	9765	9765	0.013
P_9	130	**130**	130	130	0.0005
P_{10}	1025	**1025**	1011.2	990	0.003

Problemlerin Boyutları ve Permütasyon Çözümleri

Problem	Dimension	Solution Permutation	Best Result
P_1	10	0, 1, 1, 1, 0, 0, 0, 1, 1, 1	295
P_2	20	1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1	1024
P_3	4	1, 1, 0, 1	35
P_4	4	0, 1, 0, 1	23
P_5	15	0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1	481.0694
P_6	10	0, 0, 1, 1, 0, 1, 1, 1, 1, 1	52
P_7	7	1, 0, 0, 1, 0, 0, 0	107
P_8	23	1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0	9765
P_9	5	1, 1, 1, 1, 0	130
P_{10}	20	1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1	1025

Yapılan Deneysel Çalışmaya İlişkin Teknik Bilgiler

IDE: Spyder
Sürüm: Python 3.9

Çalışmanın yapıldığı bilgisayarın teknik özellikleri:

1. CPU: AMD Ryzen 5 3500X 6-Core Processor, 3600 Mhz, 6 Core(s), 6 Logical Processor(s)
2. GPU: NVIDIA GeForce RTX 3060, VRAM 12 GB
3. RAM: 16 GB
4. İşletim Sistemi: Windows 10

Python ile Yapılan Deneysel Çalışmanın Sonuçları-1

Algoritmanın Python ile kodlanması aşamasında, sözde kodlarda yer almayan, başlangıç çözümleri optimize edilmiştir.

Başlangıç çözümleri optimize edildikten sonra algoritmanın bir miktar daha verimli çalıştığı, optimal çözümlere ulaşma kararlığının arttığı gözlenmiştir.

```
Migrating Bird Optimization (MBO)

P 1   |  profit:  295   |  [0, 1, 1, 1, 0, 0, 0, 1, 1, 1]
P 2   |  profit:  1024  |  [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1]
P 3   |  profit:  35    |  [1, 1, 0, 1]
P 4   |  profit:  23    |  [0, 1, 0, 1]
P 5   |  profit:  481   |  [0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1]
P 6   |  profit:  52    |  [0, 0, 1, 0, 1, 1, 1, 1, 1, 1]
P 7   |  profit:  107   |  [1, 0, 0, 1, 0, 0, 0]
P 8   |  profit:  9767  |  [1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0]
P 9   |  profit:  130   |  [1, 1, 1, 1, 0]
P 10  |  profit:  1025  |  [1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1]
```

Problem	Optimum Value [9]	Best	Average	Worst	Time(s)
P_1	295	**295**	294.6666	293	0.002
P_2	1024	**1024**	1011.1	991	0.003
P_3	35	**35**	35	35	0.0004
P_4	23	**23**	23	23	0.0004
P_5	481.0694	**481.0694**	431.4180	399.8016	0.002
P_6	50	**52**	52	52	0.001
P_7	107	**107**	107	107	0.0008
P_8	9767	9765	9765	9765	0.013
P_9	130	**130**	130	130	0.0005
P_{10}	1025	**1025**	1011.2	990	0.003

Python ile Yapılan Deneysel Çalışmanın Sonuçları-2

Repair ve optimization yöntemiyle 0-1 sırt çantası optimizasyon probleminin çözümü.

Sonuçlar incelendiğinde, 2, 4, 8 ve 10 no.lu problemlerin çözümü optimum çözümlere ulaşamamıştır.

MBO ile kıyaslandığında, MBO'nun daha etkin sonuçlar ürettiği görülmektedir.

```
ZeroOneKnapSack

P 1   |  profit:  295   |  [0, 1, 1, 1, 0, 0, 0, 1, 1, 1]
P 2   |  profit:  1018  |  [1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1]
P 3   |  profit:  35    |  [1, 1, 0, 1]
P 4   |  profit:  22    |  [0, 1, 1, 0]
P 5   |  profit:  481   |  [0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1]
P 6   |  profit:  52    |  [0, 0, 1, 0, 1, 1, 1, 1, 1, 1]
P 7   |  profit:  107   |  [1, 0, 0, 1, 0, 0, 0]
P 8   |  profit:  9748  |  [1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0]
P 9   |  profit:  130   |  [1, 1, 1, 1, 0]
P 10  |  profit:  1019  |  [1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1]
```

Problem	Optimum Value [9]	Best	Average	Worst	Time(s)
P_1	295	**295**	294.6666	293	0.002
P_2	1024	**1024**	1011.1	991	0.003
P_3	35	**35**	35	35	0.0004
P_4	23	**23**	23	23	0.0004
P_5	481.0694	**481.0694**	431.4180	399.8016	0.002
P_6	50	**52**	52	52	0.001
P_7	107	**107**	107	107	0.0008
P_8	9767	9765	9765	9765	0.013
P_9	130	**130**	130	130	0.0005
P_{10}	1025	**1025**	1011.2	990	0.003

Python ile Yapılan Deneysel Çalışmanın Sonuçları-3

Sınırlandırılmış sırt çantası (bounded knapsack) probleminde, kapasiteye ilave olarak seçilecek nesne adedi de bir kısıt olarak bulunur. Ancak seçilecek nesne adedi, problemin boyutuna eşit olacak şekilde belirlendiğinde, problem 0-1 sırt çantası problemine dönüşür.

Sınırlandırılmış sırt çantası yöntemi 5 no.lu problemin çözümü dışında diğer tüm çözümleri optimal seviyede bulmuştur. MBO ile kıyaslandığında, MBO algoritmasının çözüm üretme performansının bir miktar önde olduğu ifade edilebilir.

Kesirli sırt çantası yöntemi, nesnelerin parçalanabildiği esasına dayandığından, MBO algoritmasından, hatta optimal çözümlerden daha iyi sonuçlar üretmiştir. Ancak MBO ayrık problemler, kesirli sırt çantası yöntemi ise sürekli problemlerin çözümünde kullanıldığından ikisinin kıyaslanması doğru bir yaklaşım olmayabilir.

```
BoundedKnapSack

P 1   |  profit:  295
P 2   |  profit:  1024
P 3   |  profit:  35
P 4   |  profit:  23
P 5   |  profit:  477
P 6   |  profit:  52
P 7   |  profit:  107
P 8   |  profit:  9767
P 9   |  profit:  130
P 10  |  profit:  1025
```

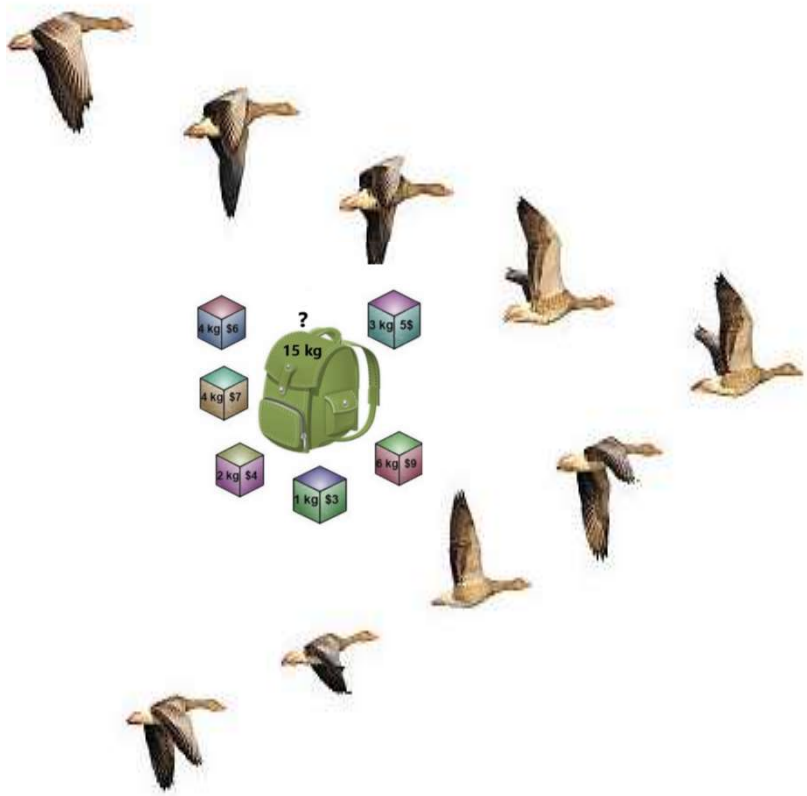
```
FractionalKnapSack

P 1   |  profit:  312
P 2   |  profit:  1035
P 3   |  profit:  37
P 4   |  profit:  26
P 5   |  profit:  488
P 6   |  profit:  54
P 7   |  profit:  107
P 8   |  profit:  10000
P 9   |  profit:  137
P 10  |  profit:  1036
```

Problem	Optimum Value [9]	Best	Average	Worst	Time(*s*)
P_1	295	**295**	294.6666	293	0.002
P_2	1024	**1024**	1011.1	991	0.003
P_3	35	**35**	35	35	0.0004
P_4	23	**23**	23	23	0.0004
P_5	481.0694	**481.0694**	431.4180	399.8016	0.002
P_6	50	**52**	52	52	0.001
P_7	107	**107**	107	107	0.0008
P_8	9767	9765	9765	9765	0.013
P_9	130	**130**	130	130	0.0005
P_{10}	1025	**1025**	1011.2	990	0.003

Sonuç

Sonuç olarak karesel atama problemleri için geliştirilen MBO algoritmasının, sırt çantası problemlerinin çözümünde, kıyaslanan diğer algoritmalardan daha başarılı olduğu görülmektedir.

Bu sonuçlara göre MBO algoritması, 0-1 sırt çantası problemlerinin çözümü için başvurulabilecek önemli bir yöntem olarak öne çıkmaktadır.



TEŞEKKÜRLER