Gazi Üniv. Müh. Mim. Fak. Der.
Cilt 25, No 4, 881-894, 2010

J. Fac. Eng. Arch. Gazi Univ.
Vol 25, No 4, 881-894, 2010

KONTEYNER YÜKLEME PROBLEMLERİ İÇİN KARINCA KOLONİSİ OPTİMİZASYONU YAKLAŞIMI

Türkay DERELİ[*1], Gülesin Sena DAŞ[*2]
[*1] Gaziantep Üniversitesi, Endüstri Mühendisliği Bölümü, 27310, Gaziantep, Türkiye
[*2] TÜBİTAK, Araştırma Destek Programları Başkanlığı, 06100, Ankara, Türkiye
dereli@gantep.edu.tr, sena.das@tubitak.gov.tr

(Geliş/Received: 04.03.2010; Kabul/Accepted: 17.03.2010)

ÖZET

Tedarik zincirlerinin uluslararası bir boyut kazandığı günümüzde, konteyner taşımacılığının ve ilgili taşıma maliyetlerinin düşürülmesinin önemi giderek artmaktadır. Bahsi geçen konuda maliyeti düşürmenin yollarından birisi de hiç kuşkusuz mevcut konteyner hacimlerinden daha iyi faydalanmaktır. Bu nedenle, NP-zor konteyner yükleme problemleri birçok araştırmacının ilgisini çekmektedir. Bu çalışmada, konteyner yükleme problemleri için karınca kolonisi optimizasyonu yaklaşımını temel alan iki yeni algoritma önerilmiştir. Parametreleri faktöriyel tasarım ile belirlenen bu algoritmaların performansları literatürde verilen standart problemler için test edilmiş ve sonuçlar literatürdeki diğer çalışmalar ile mukayese edilerek irdelenmiştir.

Anahtar Kelimeler: Konteyner yükleme, karınca kolonisi, optimizasyon

ANT COLONY OPTIMIZATION APPROACH FOR CONTAINER LOADING PROBLEMS

ABSTRACT

The importance of reducing the cost of container shipping as well as related transportation cost is gradually increasing with the internationalization of supply chains. There is no doubt that one of the potential ways of reducing these costs is the better utilization from the container volumes. Therefore, NP-hard container loading problems attracts the attention of many researchers. In this study, two algorithms for the container loading problems based on ant colony optimization are suggested. Having determined the parameters with the factorial design, the performance of the proposed algorithms is tested with the standard test cases in the literature and the results are discussed comparatively with reference to the other works in the literature.

Keywords: Container loading, ant colony, optimization

1. GİRİŞ (INTRODUCTION)

Tedarik zincirlerinin uluslararasılaşmasıyla konteyner taşımacılığı ciddi bir artış göstermiştir. Yüklerin %90'ını konteynerlerde (konteynerler ile) taşınmaktadır ve yılda yaklaşık 250 milyon konteyner varış noktalarına ulaştırılmak üzere gemilere yüklenmektedir [1]. Konteynerlerin taşımacılıkta bu kadar önemli bir rol oynadığı günümüzde, lojistik şirketleri maliyetlerini aşağı çekebilmek için rota optimizasyonu ve yük konsolidasyonu gibi uygulamaların yanı sıra, mevcut konteyner hacimlerinden de en iyi şekilde faydalanmaya çalışmaktadır. Bu talep, son zamanlarda birçok araştırmanın *'NP-zor'* bir problem olarak bilinen konteyner yükleme (KY) problemlerine yönelmesini sağlamış ve aktif bir araştırma alanı açılmasına vesile olmuştur [2].

KY problemi, birçok farklı türü olan ve esas olarak "kesme ve paketleme" *(cutting and packing)* problemleri kümesinde bulunan bir problemdir. "Kesme ve Paketleme" problemleri geometrik-kombinatorik olarak tanımlanırlar [3]. Üç boyutlu bir problem olan konteyner yükleme problemlerinin literatürde iki boyutlu "kesme ve paketleme" problemleri kadar sıklıkla çalışılmadığı da söylenebilir. KY probleminde amaç; en (w), boy (d) ve yüksekliği (h) belirli n adet dikdörtgen prizması nesnenin (küçük kutunun), boyutları belirli (W, D, H) bir konteynerin içine yerleştirilebilmesi suretiyle herhangi bir çakışma olmaksızın mevcut konteyner

hacminden maksimum derecede faydalanmaktır, konteyneri mümkün mertebe tam doldurabilmeye çalışmaktır. Bu yönüyle, bir "hacim doruklaştırma (maksimizasyonu)" problemi olarak ele alınmaktadır. Literatürde KY problemleri için kullanılan sınıflandırılmış yöntemler ve ilgili araştırmacılar Tablo 1'de özetlenmiştir.

Tablo 1'de de özetlendiği üzere, literatürdeki mevcut çalışmalar incelendiğinde; konteyner yükleme problemlerinin çözümüne yönelik olarak bugüne değin yapılan çalışmalarda karınca kolonisi optimizasyonu çözüm yaklaşımının yaygın olarak kullanılmadığı göze çarpmaktadır. Konu ile ilişkin olarak yalnızca Liang ve arkadaşları [24] tarafından 2007 yılında yapılan bir çalışmaya rastlanmıştır. Bahsi geçen çalışmada, konteyner yükleme problemlerini çözmek için karınca kolonisi optimizasyonu ve genetik algoritmaları kullanan melez bir meta-sezgisel önerilmiştir. Karınca kolonisi optimizasyonu, çalışmada konteynere yerleştirilecek kutulardan kuleler elde etmekte kullanılmıştır. Kutuları konteynere üç boyutlu olarak yerleştirmek için kullanılan metotlardan biri olan bu yaklaşımda, önce kutular üst üste konularak kutu kuleleri elde edilmekte, daha sonra ise bu kuleler konteynere yerleştirilerek çözüm elde edilmektedir. Ancak, Liang ve arkadaşları [24] tarafından yayımlanan makalede kullanılan yöntemlerle ilgili olarak detaylı bir bilgi verilmediğini ve sonuçların yeterince karşılaştırılmadığını ve tartışılmadığını da vurgulamak gerekir.

Literatürde; konteyner yükleme problemlerinin çözümüne yönelik olarak karınca kolonisi optimizasyonunun bir çözüm yaklaşımı olarak kullanılmasında bir boşluk olduğu açık bir biçimde göze çarpmaktadır. Bu boşluğun doldurulması ya da boşluğun nedenlerinin araştırılması önem arz etmektedir. Bu noktadan hareketle, bu çalışmada konteyner yükleme problemlerinin çözümüne yönelik olarak karınca kolonisi kullanılan iki çözüm yaklaşımı önerilmiş ve bu yaklaşımlar literatürdeki diğer yöntemlerle bilinen test problemleri eşliğinde kıyaslanmıştır. Bölüm 2'de *Karınca Kolonisi Optimizasyonu (KKO)* ve bu çalışmada kullanılan *Karınca Kolonisi Sistemi (KKS)* tanıtılmıştır. Önerilen algoritmalar ve bu algoritmaların parametrelerini belirlemek için yapılan çalışmalar ise Bölüm 3'de sunulmuştur. Algoritmaların performansı Bölüm 4'te irdelenmiş olup, Bölüm 5 sonuçlar ve tartışmalara ayrılmıştır.

2. KARINCA KOLONİSİ OPTİMİZASYONU (ANT COLONY OPTIMIZATION)

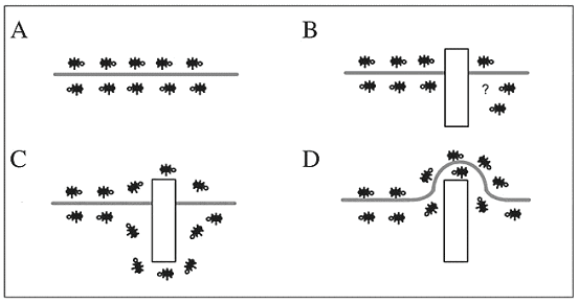
Kolektif böcek davranışlarından ilham alan yöntemler arasında pratikte en sık kullanılanlardan biri Karınca Kolonisi Optimizasyonu (KKO) metasezgiselidir [25]. Karıncaların yiyecek arama/toplama davranışından esinlenen ilgili algoritmalar ilk olarak Dorigo [26] tarafından ortaya atılmıştır. Başlangıçta kombinatorik optimizasyon problemlerine odaklanan bu yaklaşım bir meta-sezgisel olarak Karınca Kolonisi Optimizasyonu (KKO) şemsiye ismi altında anılmaktadır [27]. KKO, daha ziyade zor kombinatorik optimizasyon problemleri için kullanılan çok etmenli (multi agent) bir yöntemdir. Algoritmanın ilk uygulamaları gezgin satıcı problemi ve diğer rotalama problemleri üzerine odaklanmasına rağmen günümüzde algoritma; en kısa müşterek üst-dizi (shortest common supersequence), genelleştirilmiş atama, çoklu sırt çantası, kısıt sağlama problemlerinin de aralarında bulunduğu geniş bir kombinatoryel optimizasyon (eniyileme) problemleri yelpazesine uygulanmıştır [28]. Ayrıca, son yıllarda sürekli optimizasyon problemlerinin çözümünde de güncel ve genişletilmiş KKO yaklaşımları kullanılmaya başlanmıştır [29].

Karıncalar, koloniler halinde yaşayan, kolektif davranışlar sergileyen böceklerdir ve davranışları kolonin bir bileşeninden çok tümünün yaşamına yöneliktir [30]. Gerçek karıncalar bir yiyecek kaynağından yuvalarına herhangi bir görsel ipucu kullanmadan feromon (*pheromone*) bilgisinden faydalanarak en kısa yolu bulma kabiliyetine sahiptirler [31]. Vücutlarının salgıladığı bir kimyasal madde olan feromon ile haberleşirler. Bir yiyecek kaynağına doğru yola çıkan ilk karıncaların bıraktıkları feromon izleri, kolonin geri kalanının seçimini etkilemektedir. Eğer varılacak hedefe giden yolda feromon izi yoksa karıncalar rassal olarak

Tablo 1. Literatürde KY problemleri için kullanılan yöntem ve yaklaşımlar ile ilgili araştırmacılar (The methodologies and approaches in the literature for solving container loading problems along with corresponding researchers)

	← Kullanılan Yöntem ve Yaklaşımlar →		
	Farklı veri yapıları	*Sezgisel algoritmalar*	*Meta-sezgisel algoritmalar*
↑ **Yazarlar** ↓	Morabito ve Arenales [4] Eley [5] Lim ve ark. [6] Wang ve ark. [7]	George ve Robinson [8] Bischoff ve Marriott [9] Gehring ve ark.[10] Haessler ve Talbot [11] Ngoi ve ark. [12] Pisinger [13] Bischoff [14] Moura ve Oliveira [15] Huang ve He [16]	Gehring ve Bortfeldt [17] Bortfeldt ve Gehring [18] Faina [19] Bortfeldt ve Gehring [20] Gehring ve Bortfeldt [21] Mack ve ark. [22] Yeung ve Tang [23] Liang ve ark. [24]

hareket ederler. Aksi takdirde, karıncalar feromon izini fark ederler ve tercih ederler ki böylece yol yeniden işaretlenir ve bu izi izleyecek daha fazla karıncayı çeker [27]. Bu şekilde, karıncalar yiyecek kaynağından yuvalarına giden en kısa yolu seçebilirler. Karıncaların bu davranışı Şekil 1'de kısaca özetlenmiştir. KKO algoritmaları gerçek karıncaların davranışını taklit eden yapay karıncaları kullanmaktadır.



Şekil 1. (a) Karıncalar yiyecek kaynağı ve yuva arasına feromon izi bırakırlar (b)yerleştirilen bir engel izi böler (c) karıncalar engelin etrafından dolaşarak iki farklı yol bulurlar (d) daha kısa olan yol boyunca yeni bir feromon izi oluşur ((a) Ants in a pheromone trail between nest and food (b) an obstacle interrupts the trail (c) ants find two paths to go around the obstacle (d) a new pheromone trail is formed along the shorter path) [32].

Konu ile ilişkili olarak ilk önerilen algoritma olan karınca sistemi (ant system) [33] algoritmasından sonra, KKO algoritmaları günümüze değin belirli bir evrim geçirmiş ve geliştirilmiştir. Literatürde; Karınca Kolonisi Sistemi (Ant Colony System), Maksimum-Minimum Karınca Sistemi (Max-Min Ant System), Mertebe Temelli Karınca Sistemi (Rank based Ant System) ve En İyi-En Kötü Karınca Sistemi (Best-Worst Ant System) ve benzeri gibi birçok farklı algoritma önerilmiştir [28]. Bunlardan en başarılıları Karınca Sistemi, Karınca Kolonisi Sistemi ve Maksimum-Minimum Karınca Sistemi'dir [34]. Bu çalışmada ise orijinal Karınca Sistemi algoritmasının güncellenmesi ile ortaya çıkan ve 1997 yılında Dorigo ve Gambardella [31] tarafından önerilen *Karınca Kolonisi Sistemi* (KKS) kullanılmıştır. Temel KKO algoritmasının görsel anlatımla desteklenmiş detayları ile yukarıda anılan diğer gelişmiş KKO algoritmaları hakkındaki temel bilgilere Türkçe dilinde [35, 36] numaralı kaynaklardan ulaşılabilmektedir.

Karınca Kolonisi Optimizasyonu'nda; tipik olarak, düğümler *(vertices)* kümesi V ve ayırtlar *(edges)* kümesi E ile ifade edilen G yapısal çizgesinde *(construction graph)* $G=(V,E)$; kaynak ve hedef düğüm arasındaki en kısa yol bulma problemi çözülmek istenmektedir.

Başlangıçta bir grup karınca; $k=1,...,n_k$, rassal olarak kaynak düğümüne yerleştirilir. Her iterasyonda her bir karınca hedef düğüme kadar aşamalı olarak bir rota çizer [37]. Bu süreç esnasında; her bir düğümde her bir karınca gideceği bir sonraki düğümü seçmeye çalışır. Eğer k. karıncanın bulunduğu düğüme *i* düğümü adı verilirse, karınca gideceği bir sonraki düğüm olan $j \in N_i^k$ düğümünü Denklem 1'deki olasılık bağıntısına göre belirler.

$$p_{ij}^k(t) = \begin{cases} \dfrac{\tau_{ij}^{\alpha}(t)}{\sum_{j \in N_i^k} \tau_{ij}^{\alpha}(t)} & if \;\; j \in N_i^k \\ 0 & if \;\; j \notin N_i^k \end{cases} \quad (1)$$

Burada;

$\tau_{ij} =$ (i,j) arkındaki toplam feromon yoğunluğunu

$\tau_{ij}(0) =$ (i,j) arkındaki başlangıç feromon miktarını

$N_i^k =$ k. karıncanın ziyaret edebileceği i düğümüne bağlı mümkün düğümler kümesini

$\alpha =$ feromon miktarının önemini gösteren pozitif bir tamsayı

$L^k =$ k. karıncanın tur uzunluğunu

Tüm karıncalar turlarını tamamladıklarında, her karıncanın izlediği yola feromon bırakılır. Denklem 2 ve Denklem 3'e göre k**.** karıncanın (t) anında izlediği turun (i,j) ayırtına (*edge*), yine aynı karıncanın oluşturduğu tur uzunluğuna $L^k(t)$ bağlantılı olarak feromon bıraktığı görülmektedir.

$$\Delta\tau_{ij}^k(t) \;\; \alpha \;\; 1/L^k(t) \quad (2)$$

$$\tau_{ij}(t+1) = \tau_{ij}(t) + \sum_{k=1}^{n_k} \Delta\tau^k{}_{ij}(t) \quad (3)$$

Algoritmanın her iterasyonunda; her bir ayırttaki feromon buharlaşarak, karıncaları çözüm uzayını daha fazla keşfetmeye teşvik etmekte ve çözümün erken yakınsamasını engellemeye zorlamaktadır (Bkz. Denklem 4).

$$\tau_{ij}(t) \leftarrow (1-\rho).\tau_{ij}(t) \;\; where \;\; p \in [0,1] \quad (4)$$

Denklem 4'te p sabiti her bir ayırt (*edge*) için feromonun buharlaşma oranını gösterir. Bu parametre, karıncaların önceki kararlarının unutmalarını sağlar.

Bu çalışmada kullanılan Karınca Kolonisi Sistemi (KKS) algoritmasının yukarıda bahsedilen temel yaklaşımdan bazı farklılıkları bulunmaktadır. Bunlar; *(i)* farklı bir geçiş kuralı kullanılmaktadır, *(ii)* farklı bir feromon güncelleme kuralı tanımlanmıştır, *(iii)* lokal feromon güncellenmesi tanıtılmıştır ve *(iv)*

belirli düğümlere yönlendirme için aday listeleri kullanılmaktadır [37].

KKS algoritması sanki-rassal-orantılı (*pseudo-random-proportional rule*) adı verilen bir geçiş kuralını kullanmaktadır. Bu kurala göre *i* düğümünde bulunan k. karıncanın ziyaret edeceği bir sonraki düğüm olan *j* düğümü, Denklem 5'te gösterildiği şekilde seçilir;

$$j = \begin{cases} \arg\max_{u \in N_i^k(t)} \{\tau_{iu}^{\alpha}(t)\eta_{iu}^{\beta}(t)\} & \text{if } r \le r_0 \\ J & \text{if } r > r_0 \end{cases} \quad (5)$$

Denklem 5'te, τ_{iu} *i* düğümü ile *u* düğümü arasındaki feromon miktarını, η_{iu}^{β} *i* düğümü ile *u* düğümü arasındaki seçilebilirlik parametresini, β seçilebilirlik parametresinin önemini gösteren alt-parametreyi (1 ile 10 arasında), *r* rassal bir sayıyı (0 ve 1 arasında) ve r_0 ise kullanıcı tarafından tanımlanan bir parametreyi göstermektedir. Buna göre, eğer *r* rassal sayısı r_0'dan büyükse ($r > r_0$) ise en iyi ayırt Denklem 6'ya göre seçilir;

$$p_{iJ}^{k}(t) = \frac{\tau_{iJ}^{\alpha}(t)\eta_{iJ}^{\beta}(t)}{\sum_{u \in N_i^k} \tau_{iu}^{\alpha}(t)\eta_{iu}^{\beta}(t)} \quad (6)$$

KKS algoritmasında $\alpha = 1$ olarak kullanılmaktadır. Bu algoritmada yalnızca global en iyiyi bulan karıncanın ek feromon bırakmasına izin verilir. Global en iyiyi bulan karınca en kısa tur uzunluğuna sahip karıncadır. Global güncelleme tüm karıncalar turlarını tamamladıktan sonra uygulanır. Global güncellemede ayırtlardaki feromon güncellemesi Denklem 7 ve Denklem 8 kullanılarak yapılmaktadır;

$$\tau_{ij}(t+1) = (1 - p_1)\,\tau_{ij} + p_1\,\Delta\tau_{ij}(t) \quad (7)$$

$$\Delta\tau_{ij}(t) = \begin{cases} 1/f(x^{+}(t)) & if\ (i,j) \in x^{+}(t) \\ 0 & otherwise \end{cases} \quad (8)$$

Burada; p_1, feromon buharlaşma oranını (0 ve 1 arasında), $x^{+}(t)$, en kısa yolu, $\Delta\tau_{ij}$ ise global en iyiyi bulan karıncanın tur uzunluğunun tersini ifade etmektedir.

Diğer bir feromon güncelleme kuralı ise lokal feromon güncellemedir. Bu güncelleme kuralının global güncelleme kuralından temel farkı, uygulama zamanıdır. Lokal güncelleme, daha önce ziyaret edilen ayırtların cazip hale getirilmesini amaçlar ve feromon güncellemesi her düğümün ziyaretinden sonra Denklem 9'a göre yapılır:

$$\tau_{ij}(t) = (1 - p_2)\,\tau_{ij} + p_2\ \tau_0 \quad (9)$$

p_2, 0 ve 1 arasında tanımlanan lokal feromon buharlaşma parametresini, τ_0, küçük bir pozitif sabiti ifade etmektedir.

Bu kısımda; Karınca Kolonisi Optimizasyonu (KKO) ve konteyner yükleme problemleri için bu çalışmada kullanılan Karınca Kolonisi Sistemi (KKS) formülasyonu ana hatlarıyla verilmiştir. Önerilen çözüm yaklaşımları, aşağıda Bölüm 3'te detaylı olarak tanıtılmaktadır.

3. KONTEYNER YÜKLEME PROBLEMİ İLE KARINCA KOLONİSİ SİSTEMİ ALGORİTMASININ EŞLEŞTİRİLMESİ VE ÇÖZÜM YAKLAŞIMLARI (MAPPING "CONTAINER LOADING PROBLEM" WITH "ANT COLONY SYSTEM" ALGORITHM AND SOLUTION APPROACHES)

Konteyner yükleme problemlerinin Karınca Kolonisi Sistemi algoritması ile çözülebilmesi için, problem ilk olarak bir çizge olarak tasarlanmıştır. Bir başka deyişle, *n* adet nesne/kutu içeren bir konteyner yükleme problemi *n* düğümü olan bir çizge olarak kabul edilmiştir. Her bir düğüme sanal olarak bir kutu yerleştirilmiştir. Benzer şekilde; popülâsyondaki her karıncanın bir konteyneri olduğu düşünülmüştür. Her bir karıncanın turu esnasında ziyaret ettiği her düğümde, o düğümde bulunan kutuyu kendi konteynerine yerleştirdiği kabul edilmiştir. Her karınca kendi turunu tamamladığında, elde ettiği çözümün kalitesi sezgisel bir doldurma algoritması ile hesaplanmıştır. Bu algoritma ile kutular konteynere üç boyutlu olarak – *kutular arasında çakışma olmadan ve konteyner boyutları dâhilinde* – yerleştirilmekte ve konteyner doluluk oranı hesaplanmaktadır.

Her bir karıncanın ziyaret edeceği bir sonraki düğümü seçmesi gerektiğinde lokal (yerel) güncelleme uygulanmıştır. Global güncellemeyi en iyi çözüme sahip karıncanın yapmasına izin verilmiştir. Geliştirilen algoritmalar, 'durdurma ölçütü' olarak belirlenen maksimum iterasyon sayısı kadar çalıştırılmıştır. Bu parametrenin değeri ve algoritmaya özgü diğer bazı parametrelerin değeri deney tasarımı ile belirlenmiştir.

Karınca kolonisi sistemi algoritmalarında problem tipine uygun bir feromon tanımı yapmak gerekir. Bu çalışmadan önerilen algoritmalardaki feromon tanımı Levine ve Ducatelle [38] tarafından önerilene benzer şekilde yapılmıştır. Bahsi geçen çalışmada $\tau(i,j)$, boyutu *i* ve *j* olarak ifade edilen tek boyutlu (sadece ağırlık ile ifade edilen) iki nesnenin, kapasitesi kısıtlı olan aynı bidon içine paketlenmesinin istenirliğini göstermektedir. Bu çalışmada ise, $\tau(i,j)$ nesne (kutu) *j*'nin nesne (kutu) *i*'den sonra konteynere

yerleştirilmesinin istenirliğini göstermektedir. Karınca kolonisi sistemi algoritmalarında yapılması gereken bir başka önemli seçim ise seçilebilirlik parametresinin belirlenmesidir. Bu algoritmada seçilebilirlik parametresi $n\ (j) = w_j$ şeklinde bir nesnenin (kutunun) enine eşit olarak belirlenmiştir. Yukarıda işleyişi anlatılan algoritma KKS–1 olarak adlandırılmış olup kod taslağı *(pseudo code)* aşağıda Tablo 2'de sunulmaktadır.

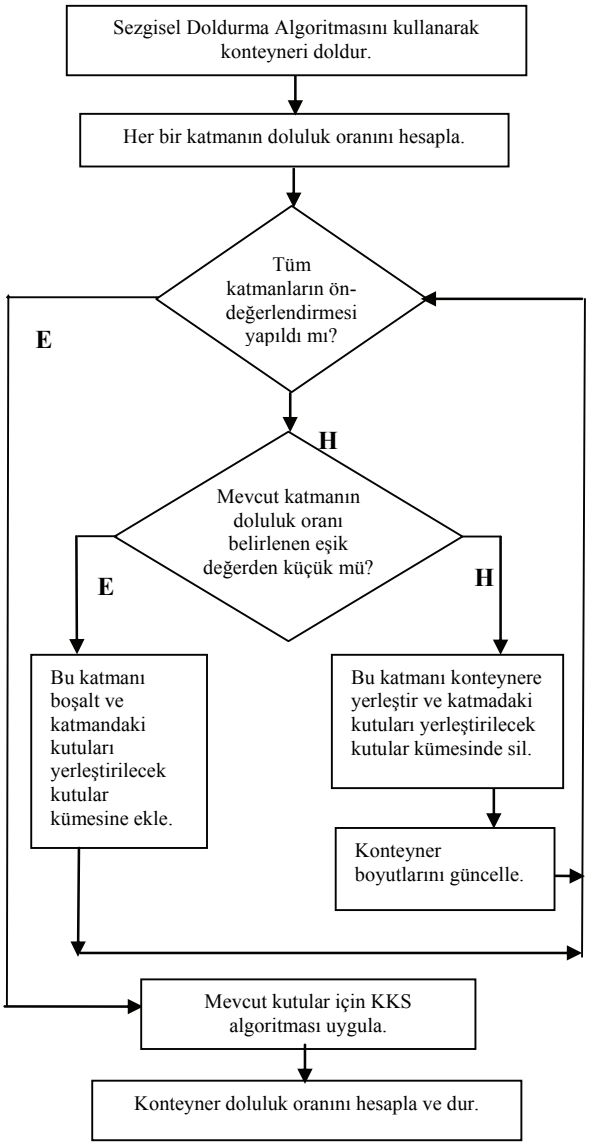
Tablo 2. KKS–1 algoritmasının temel adımları (Main steps of the KKS–1 algorithm)

```
for her koloni
 for her karınca
        rotayı tamamla
        rotayı sezgisel doldurma algoritmasını
 kullanarak değerlendir
        feromon güncelle (lokal)
 end
 feromon güncelle (global)
end
```

Ayrıca yine bu çalışmada; KKS–1 algoritmasından ayrı olarak, çizgedeki 'düğümün sayısının', bir başka deyişle 'kutu sayısının' görece olarak azaltılmaya çalışıldığı bir alternatif yaklaşım (algoritma) daha geliştirilmiş ve KKS–2 algoritması olarak adlandırılmıştır.

Önerilen her iki algoritmada da, karıncaların elde ettikleri tur aslında kutuların hangi sıralamayla konteynere yerleştirileceğini göstermektedir. KKS–2 algoritmasında kutular, kullanılan *sezgisel yerleştirme (doldurma) algoritması* gereği katmanlar halinde konteynere yerleştirilmekte ve 'çözüm kalitesi' konteynerdeki kutuların toplam hacminin konteyner hacmine bölünmesiyle bulunmaktadır. Başka bir deyişle; elde edilen çözümün kalitesi, konteynere yerleştirilen her bir katmanın ne kadar iyi doldurulduğuna bağlıdır. Buradan hareketle, KKS–2 olarak adlandırılan algoritmada, katmanların doluluk oranlarını artırmayı hedefleyen bir yaklaşım uygulandığı söylenebilir.

KKS–2 olarak adlandırılan algoritmada; konteyner ilk olarak aşağıda Bölüm 3.1'de açıklanacak olan *sezgisel doldurma algoritması* yardımıyla katmanlar şeklinde doldurulmaktadır. Önerilen KKS–1 algoritmasında ise böyle bir aşama bulunmamaktadır. Elde edilen çözümdeki katmanların doluluk oranı bir ön-değerlendirmeye tabi tutulmuştur. Bu değerlendirme kapsamında her bir katmanın doluluk oranı önceden belirlenmiş bir eşik değere göre değerlendirilir. Bu eşik değer *deney tasarımı* ile belirlenmiştir. Eğer değerlendirilen bir katmanın doluluk oranı bu değerin üstünde ise bu katman korunmakta ve bu katmanlarda bulunan kutular *'yerleştirilmiş'* olarak kabul edilmekte, aksi takdirde bu katman çözümden çıkarılmakta ve bu katmanda bulunan kutular *'yeniden konteynere yerleştirilecek kutular'* kümesine eklenmektedir. Bu şekilde, problemdeki kutu sayısı görece azaltılmaktadır. Bu aşamadan sonra, doluluk oranı iyi olan katmanlar konteynere yerleştirilmekte ve konteynerdeki mevcut katmanlar göz önünde bulundurularak henüz yerleştirilmeyen kutuların yerleştirilebilmesi için hedef konteynerin (arta kalan) boyutları güncellenmektedir. Bir sonraki aşamada ise karınca kolonisi yaklaşımı küçültülmüş probleme uygulanmaktadır. KKS–2 algoritmasının akış şeması Şekil 2'de sunulmuştur. Her iki algoritma için de kullanılan *sezgisel doldurma algoritması* ise Bölüm 3.1'de detayları ile tanıtılmaktadır.



Şekil 2. KKS–2 algoritmasının akış şeması (Flowchart of the KKS–2 algorithm)

3.1. Sezgisel Doldurma Algoritması (Heuristic Filling Algorithm)

Önerilen sezgisel doldurma algoritmasında "duvar-örme" (wall-building) yaklaşımı temel alınmıştır. İlk kez George ve Robinson [8] tarafından önerilen duvar örme ve katman yaklaşımı literatürde en sık kullanılan ve daha sonraki araştırmacılar tarafından yüksek

etkinliği ve yüksek kalitesi nedeniyle en çok güncellenen doldurma yaklaşımlardan birisidir [39]. Bu çalışmada kullanılan sezgisel doldurma algoritması ile konteyner *katman–katman* özyinelemeli (recursive) olarak doldurulmaktadır. Her bir katman doldurulmadan önce, katmanın boyutları sonraki altbölümde anlatılacağı şekilde belirlenmiştir.

Boyutları belirlenen bir katman doldurulduktan sonra mevcut katman kapatılmış ve yeni bir katman için boyutlar belirlenerek açılan bu yeni katman doldurulmuştur. Doldurma işlemine; konteynerde yeni bir katman yerleştirilecek hacim kalmayana kadar veya bir başka deyişle yerleştirilecek kutular kümesi boşalana kadar devam edilmiştir. Bu işlem sonunda konteynerde kutuların yerleştiği yalıtık (izole) katmanlar elde edilmiştir. Bu işlem esnasında herhangi bir kutunun bir kısmının başka bir katmana taşmasına izin verilmemiştir.

Katman Boyutlarının Belirlenmesi (Determination of Layer Dimensions)

Doldurma işlemine başlamadan önce, katmanın boyutlarını hesaplamak büyük önem taşımaktadır, çünkü iyi bir sonuç elde etmek için katmanının genişliği dikkatli bir şekilde seçilmelidir [13].

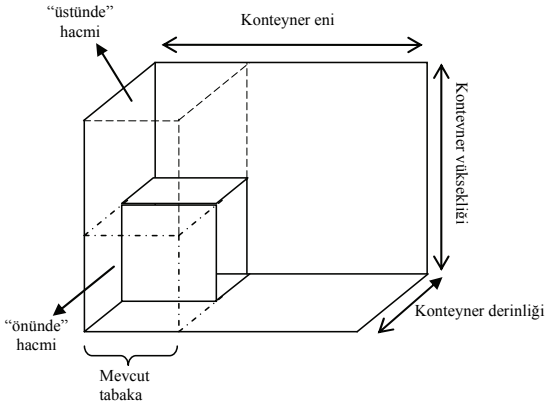
Bu çalışmada, her bir katmanın eni olan w_L Katmanı Belirleyen Kutu'nun - KBK (Layer Determining Box) enine eşit olarak belirlenmiştir. KBK'nın belirlenmesi için, ilk önce 'yerleştirilecek kutular' kümesindeki kutular enlerine göre büyükten küçüğe doğru sıralanmıştır. Sonuç olarak, en büyük *w (width, genişlik)* boyutuna sahip (en büyük eni olan) kutuya öncelik verilmiştir. Eğer sıralamadan sonra aynı en boyutuna sahip birden fazla kutu ile karşılaşılırsa, bu takdirde bu kutulardan en küçük *d (depth, derinlik)* boyutuna sahip kutuya öncelik verilmiştir. Bu şekilde, en yüksek önceliğe sahip kutu KBK olarak belirlenmiştir. Belirlenen bir KBK'ya göre bir katmanın boyutları Denklem 10'da gösterilmiştir. KBK'nın belirlenmesinden sonra, eni KBK'nın enine ve diğer boyutları konteynerin boyutlarına eşit olan katman doldurulmuştur.

$$\begin{aligned} w_L &= \max\ (w_i), \qquad i = 1, ..., n \\ h_L &= H \\ d_L &= D \end{aligned} \tag{10}$$

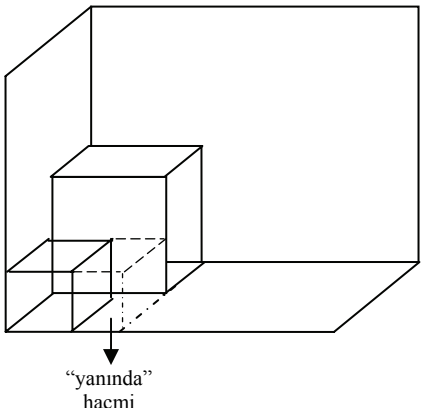
Katmanın Doldurulması (Filling the Layer)

Katman boyutlarının belirlenmesi sonrasında doldurma işlemi başlatılmaktadır. Katmanlar özyinelemeli (recursive) bir şekilde doldurulmaktadır. Özyinelemeli algoritmaları kullanmanın temel avantajı; uygulanmalarının kolay ve basit olması yanında, gerektirdiği yapıların belirli bir problemi çözmek için gerekli olan girdi sayısını azaltması, bir başka deyişle daha az bir girdi sayısıyla çözüme ulaşabilmesidir.

KBK olarak adlandırılan ilk kutu, boyutları belirlenen katmana yerleştirildiğinde; yerleştirilen kutunun etrafında, Şekil 3 ve Şekil 4'de gösterildiği üzere *"yanında", "önünde"* ve *"üstünde"* olarak adlandırılan üç adet yeni boş hacim üretilir [2]. Mevcut durumda; KBK'nın eninin katmanın enine eşit olması nedeniyle oluşan iki yeni hacim; yani, *"önünde"* ve *"üstünde"* hacimleri Şekil 3'de gösterilmiştir. Bu katmana başka bir kutu daha yerleştirildiğinde katmanda oluşan *"yanında"* hacmi ise Şekil 4'te gösterilmiştir. Bu çalışmada önerilen doldurma algoritması ile katmanlarda oluşan boş hacimler belirli bir sıra izlenerek doldurulmuştur. Bu kapsamda, öncelik *"önünde"* hacmine verilmiş, daha sonra *"yanında"* ve *"üstünde"* boş hacimlerine kutular yerleştirilmiştir.



Şekil 3. *"Üstünde"* ve *"önünde"* hacimleri (Empty-spaces: *"in front of"* and *"above"*) [2]



Şekil 4. *"Yanında"* hacmi (Empty-space: *"beside")* [2]

Bir kutunun Şekil 3'de gösterildiği gibi belirlenen katmana yerleştirildiği kabul edilirse, katmana yerleştirilecek bir sonraki kutu, yerleştirilecek kutular kümesinde en yüksek önceliğe sahip olan kutudur. Bu

kutuyu yerleştirmek için önce *"önünde"* hacmi kontrol edilir. Eğer bu kutu, bu hacme yerleştirilebiliyorsa, kutu bu hacme yerleştirilir ve yerleştirilecek kutular kümesinden silinir. Eğer yerleştirilemiyorsa, daha önce yerleştirilmiş olan kutunun yanında mevcut bir *"yanında"* hacmi mevcut olmadığından, bu yeni kutuyu yerleştirmek için *"üstünde"* hacmi kontrol edilir. Eğer bu kutu, *"üstünde"* alanına da yerleştirilemiyorsa, yerleştirilecek kutular kümesinde en yüksek önceliğe sahip olan ikinci kutunun mevcut hacimlere yerleşip yerleşemeyeceği sorgulanır. Bu yerleştirme süreci özyinelemeli bir yapıda olmak üzere; yerleştirilecek kutular kümesinde her kutu ve tüm boş hacimler için katmanda boş hacim kalmayıncaya kadar veya yerleştirilecek kutular kümesindeki tüm kutular yerleştirilene kadar tekrarlanır. Önerilen sezgisel doldurma algoritmasının temel aşamaları Tablo 3'de gösterilmiştir.

Tablo 3. Sezgisel doldurma algoritmasının temel aşamaları (The main steps of heuristic filling algorithm)

Aşama 1. Problem verisini al ve problem ait yerleştirilecek kutular kümesindeki her kutu için belirlenen sıralama ölçütüne göre öncelikleri belirle.

Aşama 2. Konteynerde yeni bir tabaka için yeterli hacim var mı? Eğer varsa Aşama 3'e git, aksi takdirde Aşama 10'a git.

Aşama 3. Katman Belirleyen Kutuyu seç

Aşama 4. Yerleştirilecek kutular kümesindeki en öncelikli kutuyu seç.

Aşama 5. Seçilen kutuyu mevcut tabakaya yerleştirmek mümkün mü? Eğer kutu yerleştirilebiliyorsa Aşama 6'ya git, aksi takdirde Aşama 4'e git.

Aşama 6. Seçilen kutuyu yerleştir ve bu kutuyu yerleştirilecek kutular kümesinden çıkar.

Aşama 7. Mevcut tabakadaki boş hacimleri güncelle.

Aşama 8. Mevcut tabakada yeni bir kutu için yeterli hacim var mı? Eğer varsa Aşama 9'a git, aksi takdirde Aşama 2'ye git.

Aşama 9. Yerleştirilecek kutular kümesinde yerleştirilecek kutu var mı? Eğer varsa Aşama 4'e git, aksi takdirde Aşama 10'a git.

Aşama 10. Konteyner doluluk oranını hesapla ve dur.

3.2. Önerilen KKS Algoritmalarının Kontrol Parametrelerinin Belirlenmesi (Determination of The Control Parameters of The Proposed ANT Colony System Algorithms)

KKO algoritmalarının performansını etkileyen bazı kontrol parametreleri mevcuttur. Bu parametreler Tablo 4'de gösterilmiştir.

İki ve daha fazla faktörün çalışılması gerektiği problemler için faktöriyel tasarım etkin bir araç olarak kabul edilmektedir [40]. Bu çalışmada önerilen KKS–1 ve KKS–2 algoritmalarının parametrelerinin belirlenmesi için de faktöriyel tasarım kullanılmıştır.

Tablo 4. Genel KKO parametreleri (General ACO parameters) [37]

Parametre
Karınca sayısı
İterasyon sayısı
Başlangıç feromon miktarı
Feromon buharlaşma oranı
Feromon miktarının önemi
Seçilebilirlik parametresinin önemi

KKS–1 algoritmasının kontrol parametreleri olan; seçilebilirlik parametresinin önemini gösteren alt-parametreyi ifade eden *beta (β) karınca sayısı (m) ve iterasyon sayısının (iter)* çözüm kalitesine etkilerinin incelenmesi ve en uygun parametre kombinasyonun (birleşiminin) bulunabilmesi için faktöriyel tasarım uygulanmıştır. KKS–1 algoritmasının kontrol parametreleri ile tasarlanan deney Tablo 5'te sunulmuştur. Bu kontrol parametrelerinin etkilerinin test edilmesi için; literatürde konteyner yükleme algoritmaların performanslarının test edilmesi için Bischoff ve Ratcliff (BR) [41] tarafından önerilen test problemlerinden (BR7 kümesinden) 10 problem için 3×3×2 (= 18) farklı deney için 3 farklı rassal sayı ile toplamda 54 kere çözülmüştür. Analiz için Minitab istatistik yazılımı kullanılmıştır.

Tablo 5. Faktöriyel tasarımdaki faktörlerin seviyeleri (Levels of factors for the factorial design)

Faktörler	Seviyeler		
beta, β	1	2	5
Karınca sayısı, m	2	4	6
İterasyon sayısı, $iter$	1000	5000	-

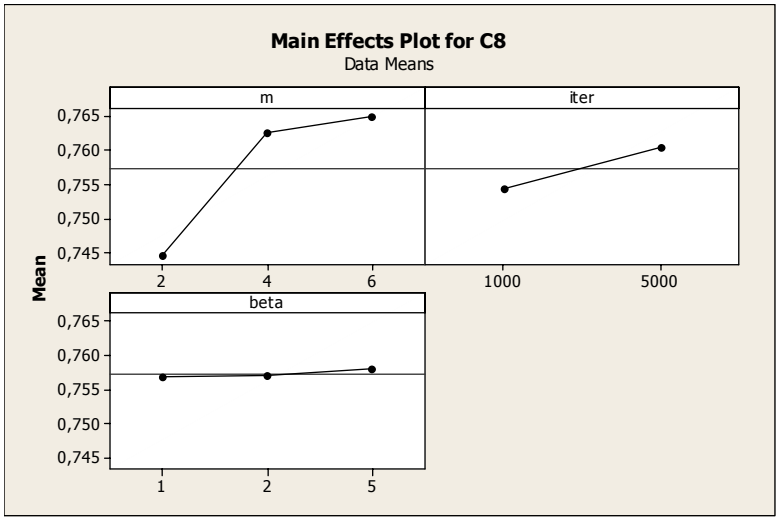
Minitab yazılımı kullanılarak elde edilen deney tasarımının sonuçlarına göre (sonuçlar Tablo 6'da verilmiştir) sadece *iterasyon sayısı (iter)* ve *karınca sayısı (m)* faktörlerinin önerilen KKS–1 algoritmanın performansını anlamlı bir şekilde etkilediğini görülmektedir. *Karınca sayısı (m)* faktörünün en etkili parametre olduğu anlaşılmaktadır. Anlamlı iki-yönlü etkileşim ise sadece beta *(β)* ve *(m)* parametreleri arasında belirlenmiştir. Bu etkileşimler dışında, parametreler arasında anlamlı iki-yönlü ve üç-yönlü etkileşim tespit edilmemiştir. Analiz sonucunda elde edilen grafikler Şekil 5 ve Şekil 6'da gösterilmiştir. Şekil 5'te de görüldüğü gibi, faktörlerin en yüksek seviyeleri için yani β değeri 5, m değeri 6 ve *iter* değeri 5000 için en iyi sonuçlar elde edilmektedir.

Tablo 6. KKS–1 algoritması için faktöriyel tasarım (Factorial design for the KKS–1 algorithm)

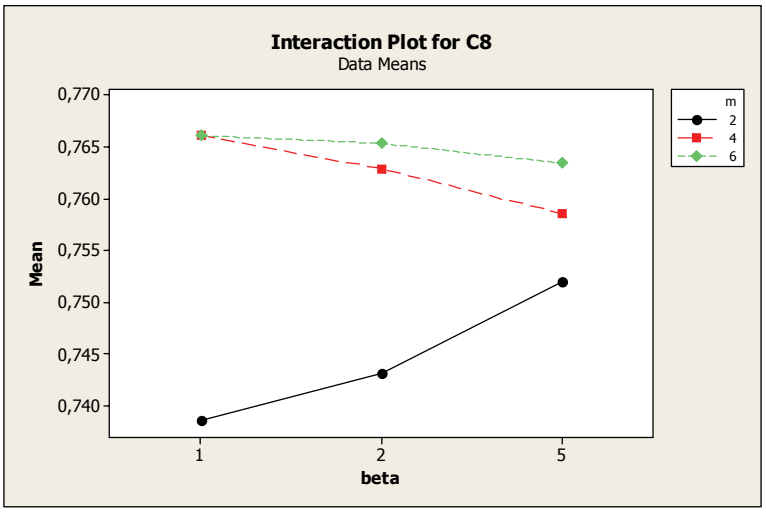
Analysis of Variance for C8, using Adjusted SS for Tests

Source	DF	Seq SS	Adj SS	Adj MS	F	P
m	2	0,0044598	0,0044598	0,0022299	135,61	0,000
iter	1	0,0005143	0,0005143	0,0005143	31,28	0,000
beta	2	0,0000115	0,0000115	0,0000057	0,35	0,708
m*iter	2	0,0000189	0,0000189	0,0000094	0,57	0,568
m*beta	4	0,0007430	0,0007430	0,0001858	11,30	0,000
iter*beta	2	0,0000226	0,0000226	0,0000113	0,69	0,509
m*iter*beta	4	0,0000476	0,0000476	0,0000119	0,72	0,582
Error	36	0,0005920	0,0005920	0,0000164		
Total	53	0,0064097				

S = 0,00405505 R-Sq = 90,76% R-Sq(adj) = 86,40%



Şekil 5. KKS–1 algoritması için parametrelerin etkilerini gösteren grafik (Graph showing the effects of the parameters for the KKS–1 algorithm)



Şekil 6. KKS–1 algoritması için parametrelerin etkileşimini gösteren grafik (Graph showing the interaction of the parameters for the KKS–1 algorithm)

Tablo 7. Faktöriyel tasarımdaki faktörlerin seviyeleri (Levels of factors for the factorial design)

Faktörler	Seviyeler		
beta, β	1	2	5
Karınca sayısı, m	2	4	6
İterasyon sayısı, $iter$	1000	5000	-
Eşik Değer, $level$	0,8	0,85	-

Benzer şekilde bu çalışmada önerilen KKS–2 algoritması için de faktöriyel tasarım uygulanmıştır. KKS–1 algoritması için incelenen kontrol parametrelerine ek olarak KKS–2 için eşik değer *(level)* parametresi de analize dâhil edilmiştir. Dört kontrol parametresi ile tasarlanan deney Tablo 7'da sunulmuştur. Yine BR7 test probleminden 10 problem

Tablo 8. KKS–1 algoritması için faktöriyel tasarım (Factorial design for the KKS–1 algorithm)

Analysis of Variance for C9, using Adjusted SS for Tests

Source	DF	Seq SS	Adj SS	Adj MS	F	P
beta	2	0,0001590	0,0001590	0,0000795	5,00	0,009
m	2	0,0040857	0,0040857	0,0020428	128,48	0,000
iter	1	0,0002789	0,0002789	0,0002789	17,54	0,000
level	1	0,0010490	0,0010490	0,0010490	65,97	0,000
beta*m	4	0,0001746	0,0001746	0,0000436	2,74	0,035
beta*iter	2	0,0000063	0,0000063	0,0000031	0,20	0,821
beta*level	2	0,0000302	0,0000302	0,0000151	0,95	0,392
m*iter	2	0,0002503	0,0002503	0,0001252	7,87	0,001
m*level	2	0,0001409	0,0001409	0,0000705	4,43	0,015
iter*level	1	0,0000068	0,0000068	0,0000068	0,43	0,514
beta*m*iter	4	0,0000200	0,0000200	0,0000050	0,31	0,867
beta*m*level	4	0,0000992	0,0000992	0,0000248	1,56	0,194
beta*iter*level	2	0,0000041	0,0000041	0,0000021	0,13	0,878
m*iter*level	2	0,0000237	0,0000237	0,0000119	0,75	0,478
beta*m*iter*level	4	0,0000490	0,0000490	0,0000123	0,77	0,548
Error	72	0,0011448	0,0011448	0,0000159		
Total	107	0,0075226				

S = 0,00398752 R-Sq = 84,78% R-Sq(adj) = 77,38%

3×3×2×2 (= 36) farklı deney için 3 farklı rassal sayı için 108 defa çözülmüştür.

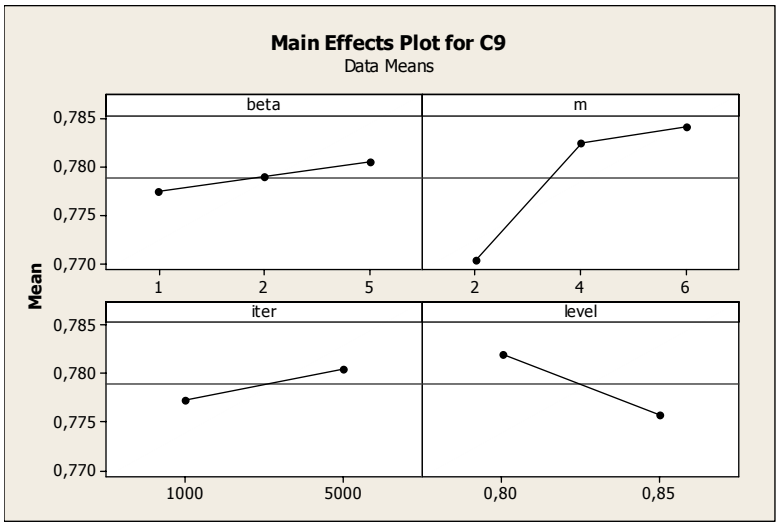
Minitab yazılımı kullanılarak elde edilen deney tasarımının sonuçları; (sonuçlar Tablo 8'de gösterilmiştir) β, m, *iter* ve *level* parametrelerinin tümünün algoritmanın performansını anlamlı şekilde etkilediğini göstermektedir. En anlamlı faktörler ise sırasıyla; karınca sayısı (*m)* ve eşik değer (*level)* parametreleridir. Bununla birlikte, m parametresinin değeri arttıkça algoritmanın performansı artarken, *level* parametresinin değeri arttıkça algoritmanın performansı düşmektedir. Bir başka deyişle, algoritmanın performansı kullanılan karınca sayısı arttıkça artarken, kullanılan eşik değer parametresinin değeri arttıkça azalmaktadır.

Anlamlı iki-yönlü etkileşim ise β ve m, m ve *iter* ve m ve *level* parametreleri arasında belirlenmiştir. Parametreler arasında üç-yönlü ve dört-yönlü anlamlı etkileşim tespit edilmemiştir. Analize ait grafikler Şekil 7 ve Şekil 8'de gösterilmiştir. Şekil 7'de de görüldüğü gibi KKS–2 algoritması için en iyi sonuç; β değeri 5, m değeri 6, *iter* değeri 5000 ve *level* değeri 0,8 olarak kullanıldığında elde edilmektedir.

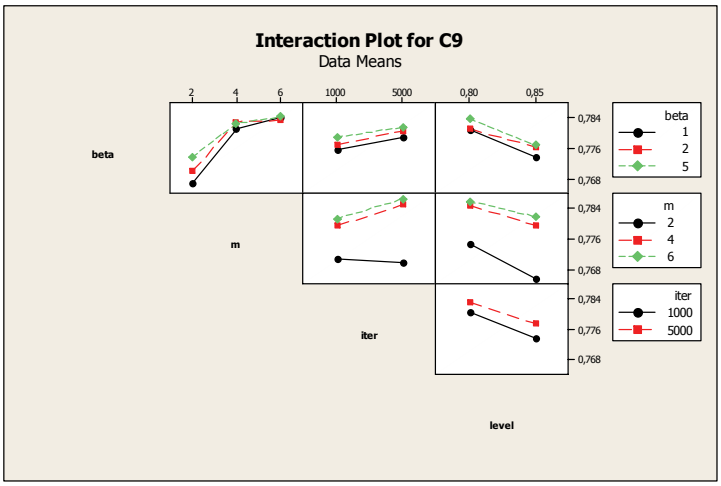
Deney tasarımı ile belirlenen değerlerin yanı sıra T_0 değeri 0.001, α değeri 1, p_1 değeri 0,9 ve p_2 değeri 0,9 olarak alınmıştır. Bu çalışmada önerilen KKS–1 ve KKS–2 algoritmaları yukarıda belirlenen parametreler kullanılarak çalıştırılmış ve performansları Bölüm 4'te sunulmuştur.

4. ÖNERİLEN ALGORİTMALARIN PERFORMANSLARININ DEĞERLENDİRİLMESİ (ASSESSMENT OF THE PERFORMANCE OF THE PROPOSED ALGORITHMS)

Bu çalışmada önerilen KKO algoritmalarının (KKS–1 ve KKS–2) performansları, literatürde sıklıkla kullanılan Loh ve Nee (LN) [42] ile Bischoff ve



Şekil 7. KKS–2 algoritması için parametrelerin etkilerini gösteren grafik (Graph showing the effects of the parameters for the KKS–2 algorithm)



Şekil 8. KKS–2 algoritması için parametrelerin etkileşimini gösteren grafik (Graph showing the interaction of the parameters for the KKS–2 algorithm)

Tablo 9. LN problemleri için literatürde sezgisel yaklaşımlarla elde edilen karşılaştırmalı sonuçlar (Comparative results obtained through heuristic approaches in the literature for the LN problems)

Problem	Loh ve Nee [42]	Ngoi ve ark. [12]	Bischoff ve ark. [43]	Bischoff ve Ratcliff [41]	Eley [5]	Lim ve ark. [6]
LN01	78,1	62,5	62,5	62,5	62,5	62,5
LN02	76,8	80,7	89,7	90,0	90,8	80,4
LN03	69,5	53,4	53,4	53,4	53,4	53,4
LN04	59,2	55,0	55,0	55,0	55,0	55,0
LN05	85,8	77,2	77,2	77,2	77,2	76,7
LN06	88,6	88,7	89,5	83,1	87,9	84,8
LN07	78,2	81,8	83,9	78,7	84,7	77,0
LN08	67,6	59,4	59,4	59,4	59,4	59,4
LN09	84,2	61,9	61,9	61,9	61,9	61,9
LN10	70,1	67,3	67,3	67,3	67,3	67,3
LN11	63,8	62,2	62,2	62,2	62,2	62,2
LN12	79,3	78,5	76,5	78,5	78,5	69,5
LN13	77,0	84,1	82,3	78,1	85,6	73,3
LN14	69,1	62,8	62,8	62,8	62,8	62,8
LN15	65,6	59,5	59,5	59,5	59,5	59,5
Ortalama	74,2	69,0	69,5	68,6	69,9	67,0

Ratcliff (BR) [41] test problemleriyle sınanmıştır. LN problemleri 15 adet problemden oluşmaktadır ve her bir problemde boyutları farklı bir konteynerin doldurulması istenmektedir. BR problemlerinde ise 7 farklı problem seti ve her sette 100 adet problem mevcuttur. LN problemlerinden farklı olarak tüm BR problemlerinde kullanılan konteynerin boyutu aynıdır. Belirtilen test problemleri daha önce geliştirilen pek çok sezgisel ve meta-sezgisel algoritma ile de çözdürülmüştür. Bu çalışmada önerilen algoritmalarla (KKS–1 ve KKS–2) elde edilen sonuçlar ile aynı problemler için önceki çalışmalarda elde edilen sonuçlar, Tablo 9 ve Tablo 10'da karşılaştırmalı sunulmuştur. LN problemlerini değerlendirirken, Loh ve Nee [42] tarafından önerilen çözüm tekniğinde diğer yaklaşımlardan farklı olarak doluluk oranı konteyner boyutlarına göre değil, paketleme yapılan kutuları kapsayan en küçük dikdörtgen hacim dikkate alınarak hesaplandığı dikkate alınmalı ve karşılaştırma yapılırken bu durum göz önünde bulundurulmalıdır.

Tablo 10'da da görüldüğü üzere, KKS–2, KKS–1 algoritmasından daha iyi performans göstermiştir. KKS–2 algoritması, LN02, LN06, LN07, LN12 ve LN13 numaralı problemler dışında kalan bütün problemler için en iyi çözümü bulabilmiştir. KKS–2 algoritmasının performansı literatürde bulunan mevcut sezgisel yaklaşımlarla karşılaştırıldığında, en iyi performansa sahip algoritma ile KKS–2 algoritması arasındaki performans farkının sadece %1,72 düzeyinde olduğu görülmektedir. Meta-sezgisel yaklaşımlarla karşılaştırıldığında ise; en iyi performansa sahip algoritma ile KKS–2 algoritması arasındaki ortalama performans farkının %3,1 düzeyinde olduğu görülmektedir.

Tablo 10. LN problemleri için literatürde meta-sezgisel yaklaşımlarla elde edilen karşılaştırmalı sonuçlar
(Comparative results obtained through metaheuristic approaches in the literature for the LN problems)

Problem	Gehring ve Bortfeldt [17]	Bortfeldt ve Gehring [18]	Bortfeldt ve Gehring [20]	Bortfeldt ve ark.[14]	Moura ve Oliveira [15]	Liang ve ark. [24]	KKS–1	KKS–2
LN01	62,5	62,5	62,5	-	-	62,5	62,5	62,5
LN02	90,7	96,7	89,8	-	-	89,7	84,3	80,8
LN03	53,4	53,4	53,4	-	-	53,4	53,4	53,4
LN04	55,0	55,0	55,0	-	-	55,0	55,0	55,0
LN05	77,2	77,2	77,2	-	-	77,2	77,2	77,2
LN06	91,1	96,3	92,4	-	-	91,4	82,5	85,2
LN07	82,7	84,7	84,7	-	-	84,6	82,9	84,0
LN08	59,4	59,4	59,4	-	-	59,4	59,4	59,4
LN09	61,9	61,9	61,9	-	-	61,9	61,9	61,9
LN10	67,3	67,3	67,3	-	-	67,3	67,3	67,3
LN11	62,2	62,2	62,2	-	-	62,2	62,2	62,2
LN12	78,5	78,5	78,5	-	-	78,5	74,8	77,3
LN13	85,6	85,6	85,6	-	-	85,6	81,6	81,6
LN14	62,8	62,8	62,8	-	-	62,8	62,8	62,8
LN15	59,5	59,5	59,5	-	-	59,5	59,5	59,5
Ortalama	70,0	70,9	70,1	70,9	70,3	70	68,5	68, 7

Tablo 11. BR problemleri için literatürde sezgisel yaklaşımlarla elde edilen karşılaştırmalı sonuçlar
(Comparative results obtained through heuristic approaches in the literature for the BR problems)

Problem (Kutu tipi)	Bischoff ve ark. [43]	Bischoff ve Ratcliff [41]	Eley [5]	Bischoff [44]	Lim ve ark. [6]
BR1 (3)	81,76	83,79	-	89,39	87,40
BR2 (5)	81,70	84,44	-	90,26	88,70
BR3 (8)	82,98	83,94	-	91,08	89,30
BR4 (10)	82,60	83,71	-	90,90	89,70
BR5 (12)	82,76	83,80	-	91,05	89,70
BR6 (15)	81,50	82,44	-	90,70	89,70
BR7 (20)	80,51	82,01	-	90,44	89,40
Ortalama	81,97	83,50	88,75	90,55	89,13

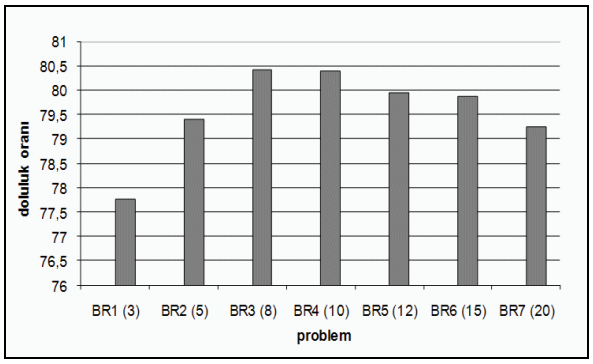
Tablo 12. BR problemleri için literatürde meta-sezgisel yaklaşımlarla elde edilen karşılaştırmalı sonuçlar
(Comparative results obtained through metaheuristic approaches in the literature for the BR problems)

Problem (Kutu tipi)	Gehring ve Bortfeldt [17]	Bortfeldt ve Gehring [18]	Bortfeldt ve Gehring [20]	Gehring ve Bortfeldt [21]	Bortfeldt ve ark.[14]	Moura ve Oliveira [15]	Liang ve ark. [24]	KKS–2
BR1 (3)	85,80	92,63	87,81	88,10	93,52	89,07	88,7	77,75
BR2 (5)	87,26	92,70	89,40	89,56	93,77	90,43	90,7	79,41
BR3 (8)	88,10	92,31	90,48	90,77	93,58	90,86	91,6	80,41
BR4 (10)	88,04	91,62	90,63	91,03	93,05	90,42	91,8	80,40
BR5 (12)	87,86	90,86	90,73	91,23	92,34	89,57	91,7	79,94
BR6 (15)	87,85	90,04	90,72	91,28	91,72	89,71	91,3	79,87
BR7 (20)	87,68	88,63	90,65	91,04	90,55	88,05	90,8	79,23
Ortalama	**87,50**	**91,26**	**90,10**	**90,43**	**92,70**	**89,73**	**90,94**	**79,57**

LN problemlerinin çözümü ile ilgili olarak; bu çalışmada önerilen KKS–2 algoritmasının performansının KKS–1 algoritmasından daha iyi olması sebebiyle, BR test problemleri sadece KKS–2 algoritması kullanılarak çözdürülmüştür. Önerilen KKS–2 algoritması ile elde edilen sonuçlar ile önceki çalışmalarda elde edilen BR test problemlerinin sonuçları, Tablo 11 ve Tablo 12'de sunulmuştur.

BR test problemleri KKS–2 algoritması ile çözüldüğünde ortalama %79.57'lik bir doluluk oranı elde edilmiştir. Bu problem seti için, KKS–2 algoritmasının performansı diğer yaklaşımlar karşılaştırıldığında, en iyi performansa sahip meta sezgisel ile %14,16 ve en iyi performansa sahip sezgisel yaklaşım ile %12,1'lik bir performans farkı tespit edilmiştir.

Tablo 11 ve Tablo 12'den de görülebileceği üzere, BR problemlerinde her problem setinde kutu tipi sayısı değişiktir. Önerilen KKS–2 algoritmasının performansı kutu tipi sayısına bağlı olarak değişmiştir (bkz Şekil 9). Kutu tipi sayısı arttıkça önce algoritmanın performansı keskin bir şekilde artmış, daha sonra ise bir miktar azalmıştır. Bu durum diğer çözüm yaklaşımlarında da gözlemlenebilmektedir. En iyi doluluk oranı ise 8 farklı kutu tipi olan BR3 problemleri için elde edilmiştir.



Şekil 9. BR problemlerinde KKS–2 algoritmasının performans değişimi (Performance change of the KKS–2 algorithm for BR problems)

5. TARTIŞMA, SONUÇ VE GELECEKTEKİ ÇALIŞMALAR (DISCUSSION, CONCLUSION AND FUTURE WORKS)

Bu çalışmada; elde tek bir konteyner olduğu varsayılarak, konteyner yükleme problemlerinin çözümü için, literatürde gezgin satıcı problemi, çoklu sırt çantası problemi vb gibi birçok kombinatorik eniyileme problemine başarıyla uygulanmış olan *karınca kolonisi optimizasyonu* yaklaşımını kullanan iki çözüm algoritması (KKS–1 ve KKS–2) önerilmiştir. KKS–2 algoritması, KKS–1 algoritmasına bir alternatif olarak, çözüm kümesinde bulunan düğüm sayısı azaltılarak (her bir düğüm konteyner içine yerleştirilecek olan bir kutuyu temsil etmektedir) türetilmiş ve elde edilen çözümlerin amaç fonksiyonu değerini elde etmek için özyinelemeli bir yapıda çalışan sezgisel doldurma algoritması kullanılmıştır. Bu yönüyle, bu çalışmada sunulan algoritmaların da bir bakıma hibrid (melez) algoritmalar olduğu söylenebilir. Her iki algoritmanın parametreleri de deney tasarımı kullanılarak belirlenmiştir. Önerilen algoritmaların performansı, Loh ve Nee (LN) [42] ile Bischoff ve Ratcliff (BR) [41] test problemleri ile birlikte sınanmış ve elde edilen sonuçlar önceki çalışmalarda önerilen yöntemler ile üretilen sonuçları ile karşılaştırılmıştır. Elde edilen sonuçlar konteyner yükleme problemlerinin çözümünde önerilen karınca kolonisi yaklaşımının kullanılabilirliğini ortaya koymuş ve bazı sorunların tartışılmasına zemin hazırlamıştır.

Beklendiği üzere; problemin boyutunu küçülten yaklaşım olan KKS–2 algoritması ile KKS–1 algoritmasından daha iyi sonuçlar elde edilmiştir. Ancak, ikisi arasındaki performans farkının oldukça sınırlı olması, özellikle önerilen sezgisel doldurma algoritmasının yapısında ve kullanılan alt yaklaşımlarında bazı iyileştirilmeler yapılması gerektiğini ortaya koymaktadır.

Literatür ile karşılaştırılmalı olarak verilen Tablo 9, Tablo 10, Tablo 11 ve Tablo 12'de hesaplama zamanları ile ilgili detaylı bilgiler verilememiştir. Bunun nedeni önceki çalışmalarda kimi zaman bu bilgilerin verilmemiş olması ya da çalışmalarda kullanılan bilgisayar sistemlerinin farklı olmasından kaynaklanmaktadır. Değerler normalize edilemediğinden (standartlaştırılmadığından) hesaplama zamanları detaylı bir biçimde karşılaştırılamamıştır. Ancak, genel olarak irdelendiğinde, özyinelemeli yapısına rağmen, bu çalışmada önerilen algoritmaların hesaplama zamanları açısından diğerleri ile yakınsak ve rekabet edebilir bir düzeyde olduğu söylenebilir.

Bu çalışmada önerilen algoritmaların performanslarının, Loh ve Nee (LN) test problemlerinin çözümü için ortaya konan önceki çalışmaların performanslarına oldukça yakın olduğu müşahede edilmiştir. Bununla birlikte, önerilen algoritmaların performanslarının, daha karmaşık olarak bilinen Bischoff ve Ratcliff (BR) test problemlerinin çözümü için ortaya konan önceki çalışmaların performanslarının ise altında kaldığı gözlemlenmiştir. Bunun nedenleri için pek çok şey söylenebilir; problemin çizge üzerinde tanımlanması ve eşleştirilmesindeki bazı ayrıntılar, kullanılan komşuluk arama yapısı, sezgisel doldurma algoritmasında kullanılan yaklaşım ve benzeri gibi. Aşağıda bu konular kısaca irdelenmekte ve devam eden/gelecekteki çalışmalarda yapılabilecekler tartışılmaktadır.

Bu çalışmada sunulan karınca kolonisi optimizasyonu algoritmalarının tasarımında, kutuların konteynere yerleştirilirken hangi sırayla yerleştirilecekleri üzerinde durulmuş, fakat kutuların farklı eksenlerde dönebilme/döndürülebilme özelliklerinden faydalanılmamıştır. Bu özellikten de yararlanacak şekilde farklı komşuluk yapıları tasarlanabilir, kullanılabilir ve bu yeni yapıların algoritmaların performansları üzerindeki etkileri araştırılabilir.

Önerilen karınca kolonisi optimizasyonu algoritmalarının performanslarının iyileştirilmesi amacıyla yapılması gerekenlerden birisi de, bu çalışmada kullanılan sezgisel doldurma algoritmasının yapısının değiştirilmesi ve geliştirilmesi olabilir. Bu çalışmada kullanılan sezgisel doldurma algoritması 'duvar örme' yaklaşımını temel almaktadır. Kullanılan bu yaklaşımın daha probleme-özgü hale getirilmesinin önerilen karınca kolonisi algoritmalarının performansını artırabileceği

düşünülmektedir. Bu amaçla denenebilecek bir diğer yaklaşım ise 'duvar örme' yerine 'kule oluşturma', 'kübik düzenleme' ve benzeri doldurma yaklaşımların kullanılması olabilir.

Bilindiği üzere; birçok evrimsel algoritmada olduğu gibi, karınca kolonisi optimizasyonu algoritmalarının global araştırma yetenekleri çok yüksek düzeyde olmasına rağmen, lokal arama özellikleri görece zayıftır. Lokal arama süreci genellikle elde edilen bir çözümün komşuluğunda daha iyi çözümler için keşfetmek için yapılır [45]. Gelecek çalışmalar kapsamında, sunulan karınca kolonisi optimizasyonu algoritmalarına eklenebilecek bir lokal arama yordamının, algoritmanın performansı üzerine etkileri incelenebilir.

Bu çalışma; konusu 'konteyner yükleme problemleri için topluluk zekâsı kullanımı ve bir karar destek sistemi geliştirilmesi' şeklinde belirlenen ve son aşamasına gelmiş bulunan doktora tezi çalışmasından bir kesit sunmaktadır. Bu makalede, tarif edilen problemin çözümü için kullanılabilecek alternatif yaklaşımlardan biri olan karınca kolonisi yaklaşımı üzerine odaklanılmıştır. Bu nedenle; probleme bütünsel bakış, türev ve entegre problemler, karar destek sistemi geliştirilmesi vb gibi konulara bu makalede değinilmemiştir. Örneğin, gerçek hayatta karşılaşılan konteyner yükleme problemleri göz önünde bulundurulduğunda, üretilen çözüm önerilerinin görselleştirilmesi ve karar vericilerin kullanımına bir kullanıcı ara-yüzü ile sunulması da önemli bir husustur. Yazarların bu amaçla geliştirdikleri karar destek sistemi Kaynak [46]'da sunulmuştur.

TEŞEKKÜR (ACKNOWLEDGEMENTS)

İkinci yazar doktora çalışması sırasındaki burs desteği sebebiyle TÜBİTAK'a teşekkür etmektedir.

KAYNAKLAR (REFERENCES)

1. van de Voort, M., O'Brien, K.A., Rahman, A., Valeri, L. Seacurity: Improving the Security of the Global Sea-Container Shipping System, **Rand**., 2003.
2. Dereli, T., Das, G.S. "A hybrid simulated annealing algorithm for solving multi-objective container loading problems", **Applied Artificial Intelligence**, 24, 463-486, 2010.
3. Dereli, T., Daş, G.S. "A hybrid simulated annealing algorithm for two-dimensional strip packing problems". **Adaptive and Natural Computing Algorithms**, Part 1, 4431, 508-516, 2007.
4. Morabito, R.N., Arenales, M.N. "An and-or graph approach to the container loading problem." **International Transactions in Operational Research**, 1, 59-73, 1994.
5. Eley, M. "Solving container loading problems by block arrangement." **European Journal of Operational Research**, 141, 393-409, 2002.
6. Lim, Rodrigues, B., Yang, Y. "3-D Container packing heuristic." **Applied Intelligence**, 22, 125-134, 2005.
7. Wang, Z., Li, K.W., Levy, J.K. "A heuristic for the container loading problem: A tertiary-tree-based dynamic space decomposition approach." **European Journal of Operational Research**, 191, 86–99, 2008.
8. George, J.A., Robinson, D.F. "A heuristic for packing boxes into a container." **Computers and Operations Research**, 7, 147-156, 1980.
9. Bischoff E.E., Marriott M.D. "A comparative evaluation of heuristics for container loading", **European Journal of Operational Research**, 44, 267-276, 1990.
10. Gehring, H., Menschner, K., Meyer, M.A. "Computer-based heuristic for packing pooled shipment containers", **European Journal of Operational Research**, 44, 277-288, 1990.
11. Haessler, R.W., Talbot, F.B. "Load planning for shipments of low density products", **European Journal of Operational Research**, 44, 289-299, 1990.
12. Ngoi, B.K.A., Tay, M.L., Chua, E.S., "Applying spatial representation techniques to the container packing problem", **International Journal of Production Research**, 32/1, 111-123, 1994.
13. Pisinger, D. "Heuristics for the container loading problem", **European Journal of Operational Research**, 141, 382-392, 2002.
14. Bortfeldt, A., Gehring, H., Mack, D. "A parallel tabu search algorithm for solving the container loading problem". **Parallel Computing**, 29, 641-662, 2003.
15. Moura, A., Oliveira J. "A grasp approach to the container-loading problem". **IEEE Intelligent Systems,** 50-57, 2005.
16. Huang, W., He, K. "A caving degree approach for the single container loading problem". **European Journal of Operational Research**, 196, 93–101, 2009.
17. Gehring, H., Bortfeldt, A. "A genetic algorithm for solving the container loading problem". **International Transactions in Operational Research**, 4, 401-418, 1997.
18. Bortfeldt, A., Gehring, H. "Ein Tabu Search - Verfahren für Containerbeladeprobleme mit schwach heterogenem Kistenvorrat", **OR Spektrum**, 20, 237-250, 1998.
19. Faina, L. "A global optimization algorithm for the three-dimensional packing problem". **European Journal of Operational Research**, 126, 340-354, 2000.
20. Bortfeldt, A, Gehring, H. "A hybrid genetic algorithm for the container loading problem". **European Journal of Operational Research**, 131, 143-161, 2001.

21. Gehring, H., Bortfeldt, A. "A parallel genetic algorithm for solving the container loading problem". **International Transactions on Operational Research**, 9/4, 497–511, 2002.

22. Mack, D., Bortfeldt, A., Gehring, H., "A parallel hybrid local search algorithm for the container loading problem". **International Transactions in Operational Research**, 11, 511-533, 2004.

23. Yeung, L.H.W, Tang, W.K.S. "A hybrid genetic approach for container loading in logistics industry". **IEEE Transactions on Industrial Engineering**, 52, 617-627, 2005.

24. Liang S.C., Lee, C.Y., Huang S.W. "Hybrid Meta-Heuristic for the Container Loading Problem". **Communications of the IIMA**, 7/4, 73-84, 2007.

25. Dereli, T., Seçkiner, S.U., Daş, G.S., Gökçen, H., Aydın, M.E. "An exploration of the literature on the use of 'swarm intelligence-based techniques' for public service problems". **European Journal of Industrial Engineering**, 3, 379-423, 2009.

26. Dorigo, M. **"Optimization, Learning and Natural Algorithms",** PhD Thesis, Politecnico di Milano, Italy, 1992.

27. Zhao, P., Zhao, P., Zhang X. "A new ant colony optimization for the knapsack problem". 7th **International Conference on Computer-Aided Industrial Design and Conceptual Design CAIDCD '06**, 2006.

28. Cordon, O., Herrera, F., Stützle, T. "A Review on the Ant Colony Optimization Metaheuristic: Basis, Models and Trends", **Mathware & Soft Computing**, 9, 2002.

29. Socha, K, Dorigo, M. "Ant colony optimization for continuous domains". **European Journal of Operational Research**, 185, 1155–1173, 2008.

30. Dorigo, M., Di Caro, G., Gambardella, L. M. "Ant algorithms for discrete optimization". **Artificial Life**, 5/2, 137-172, 1999.

31. Dorigo, M., Gambardella, L.M. "Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem". **IEEE Transactions on Evolutionary Computation**, 1, 1, 53-66, 1997.

32. Perretto, M., Lopes, H.S. "Reconstruction of phylogenetic trees using the ant colony optimization paradigm". **Genetics and Molecular Research**, 4/3, 581-589, 2005.

33. Dorigo, M., Maniezzo, V., Colorni, A. "Ant system: optimization by a colony of cooperating agents", **IEEE Transactions on Systems, Man, and Cybernetics-Part B**, 26 (1), 29-41, 1996.

34. http://www.scholarpedia.org/article/Ant_colony_ optimization

35. Keskintürk, T., Söyler, H. "Global ant colony optimization (Global karınca kolonisi optimizasyonu)", **Journal of the Faculty of Engineering and Architecture of Gazi University**. (**Gazi Üniversitesi Mühendislik – Mimarlık Fakültesi Dergisi**), 21 (4), 689-698, 2006.

36. Alaykıran, K, Engin, O. "Karınca kolonileri meta-sezgiseli ve gezgin satıcı problemleri üzerinde bir uygulaması (Ant colony metaheuristic and an application on traveling salesman problem)", **Journal of Faculty of the Engineering and Architecture of Gazi University**. (**Gazi Üniversitesi Mühendislik – Mimarlık Fakültesi Dergisi**), 20 (1), 69-76, 2005.

37. Engelbrecht, A.P. **Fundamentals of Computational Swarm Intelligence**, Wiley, 2005.

38. Levine, J., Ducatelle, F. "Ant colony optimisation for bin packing and cutting stock problems". **Journal of Operational Research Society**, 55, 705-716, 2004.

39. He, K., Huang, W. "Solving the single container loading problem by a fast heuristic method". **Optimization Methods and Software**, 1-15, 2009.

40. Montgomery, D.C. **Design and analysis of experiments**. John Wiley & Sons, New York, 1991.

41. Bischoff, E.E., Ratcliff, M.S.W. "Issues in the development of approaches to container loading". **Omega – International Journal of Management Science**, 23/4, 337-390, 1995.

42. Loh, H. T., Nee, A. Y. C. "A packing algorithm for hexahedral boxes". **Proceedings of the Industrial Automation Conference**, Singapore, 2, 115-126, 1992.

43. Bischoff, E.E., Janetz, F., Ratcliff, M.S.W. "Loading pallets with non-identical items". **European Journal of Operational Research**, 84, 681-692, 1995.

44. Bischoff, E.E. "Dealing with load bearing strength considerations in container loading problems". **Technical Report**, European Business Management School. University of Wales, Swansea, 2003.

45. Luo, D., Wu, S., Li, M., Yang, Z. "Ant Colony Optimization with Local Search Applied to the Flexible Job Shop Scheduling Problems". **Proceedings of ICCCAS 200**8 - **IEEE**, 1015 – 1020, 2008.

46. Dereli, T., Das, G.S., "Development of a decision support system for solving container loading problems", **TRANSPORT, Research Journal of Vilnius Gediminas Technical University and Lithuanian Academy of Sciences**, ISSN 1648-4142**,** 25 (2), 138-147**,** 2010.