Gazi Üniv. Müh. Mim. Fak. Der.
Cilt 20, No 4, 443-454, 2005

J. Fac. Eng. Arch. Gazi Univ.
Vol 20, No 4, 443-454, 2005

KONKAV MALİYETLİ ULAŞTIRMA PROBLEMİ İÇİN GENETİK ALGORİTMA TABANLI SEZGİSEL BİR YAKLAŞIM

İsmail KARAOĞLAN ve Fulya ALTIPARMAK
Endüstri Mühendisliği Bölümü, Mühendislik Mimarlık Fakültesi, Gazi Üniversitesi, Maltepe, Ankara
ikaraoglan@gazi.edu.tr, fulyaal@gazi.edu.tr

(Geliş/Received: 19.11.2004; Kabul/Accepted: 19.08.2005)

ÖZET

Konkav Maliyetli Ulaştırma Problemi (KMUP), gerçek hayatta sık karşılaşılan problemlerden birisidir. Doğrusal maliyetli problemlerin aksine, KMUP'de taşınacak miktar arttıkça birim taşıma maliyeti azalmaktadır. Bu tür problemlerde doğrusal olmayan maliyet fonksiyonundan dolayı klasik optimizasyon yöntemleri ile en iyi çözüme ulaşmak mümkün olmayabilir. Son yıllarda, genetik algoritmalar, tavlama benzetimi ve tabu arama gibi genel amaçlı sezgisel yöntemlerin bu tür zor problemlerin çözümünde başarıyla kullanıldığı görülmektedir. Bu çalışmada, KMUP için genetik algoritmalara dayalı bir karma sezgisel algoritma (karma GA) geliştirilmiştir. Algoritmanın etkinliği, tedarikçi ve müşteri sayısının 4 ile 40 arasında değiştiği ve rassal olarak üretilen 12 problem üzerinde incelenmiştir. Geliştirilen karma GA, literatürde bu problem için geliştirilmiş olan tavlama benzetimi, eşik kabulü ve doğrusal eşik kabulü yöntemine dayalı sezgisel algoritmalar ile karşılaştırılmıştır. Karşılaştırma sonucunda, geliştirilen karma GA ile dört problem için çözüm kalitesinde %0.3 ile %5 arasında iyileşmenin olduğu görülmüştür.

Anahtar Kelimeler: Ulaştırma problemi, genetik algoritmalar, yayılan ağaç, tavlama benzetimi, eşik kabulü.

A HEURISTIC APPROACH BASED ON GENETIC ALGORITHM FOR CONCAVE COST TRANSPORTATION PROBLEM

ABSTRACT

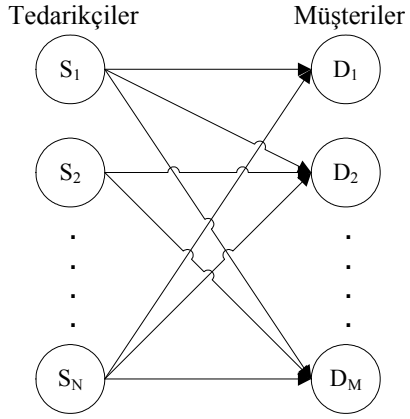
Concave Cost Transportation Problem (CCTP) is one of the widely employed problems in real applications. In this problem, the unit cost is decreasing in the amount that is transported. Since these types of problems are nonlinear, it is difficult to solve them to optimality. In literature, metaheuristic techniques such as genetic algorithm, simulated annealing and tabu search have been successfully applied for solving these problems. In this study, we develop a new hybrid algorithm based on genetic algorithms for the problem. The efficiency and effectiveness of the algorithm are investigated using twelve randomly generated test problems in which the number of suppliers and customers changes between 4 and 40. Comparisons with other heuristics based on simulated annealing, threshold acceptance and linear threshold acceptance shows that developed hybrid GA improves solution quality by 0.3% and 5% for four problems.

Keywords: Transportation problem, genetic algorithms, spanning tree, simulated annealing, threshold acceptance.

1. GİRİŞ

İlk defa Hitchcock [1] tarafından incelenen Ulaştırma Problemi (UP), teorik ve ekonomik öneminden dolayı üzerinde çok çalışılan problemlerden birisidir. UP'de amaç, değişik arz noktalarından (*tedarikçi*) değişik talep noktalarına (*müşteri*) toplam maliyeti en küçükleyecek şekilde ürünün nasıl taşınacağının tespit edilmesidir. Ancak, UP sadece ürün taşımacılığını ilgilendiren bir problem değildir. Üretim planlama, personel atama, işlerin makinelere dağıtımı, tesis yerleşimi gibi problemlerde değişik teknikler ile UP'ye dönüştürülmekte ve çözülebilmektedir.

Şekil 1'de görüldüğü gibi genel ulaştırma problemi, N tedarikçilerin kümesi, M müşterilerin kümesi, A bütün yönlü hatların kümesi olmak üzere iki parçalı (*bipartite*) şebeke, $G=(N,A,M)$, olarak tanımlanır.



Şekil 1. Ulaştırma şebekesi

Genel ulaştırma probleminin tamsayılı programlama modeli aşağıda tanımlanmaktadır.

$$Min\ f(x) = \sum_i \sum_j f(x_{ij}) \quad (1)$$

Kısıtlar

$$\sum_j x_{ij} = S_i \quad \forall i \in N \quad (2)$$

$$\sum_i x_{ij} = D_j \quad \forall j \in M \quad (3)$$

$$x_{ij} \geq 0 \text{ ve tamsayı} \quad (4)$$

Modelde, S_i; *i.* tedarikçinin kapasitesi, D_j; *j.* müşterinin talebi, x_{ij}; *i.* tedarikçiden *j.* müşteriye taşınan ürün miktarı ve $f(x_{ij})$; taşınan x_{ij} birimin taşıma maliyeti fonksiyonudur. Amaç fonksiyonu (1) toplam ulaştırma maliyetini en küçüklemektedir. (2) nolu kısıt tedarikçilerden yapılacak olan toplam taşımanın tedarikçi kapasitesine eşit olmasını sağlamaktadır. (3) nolu kısıt benzer şekilde, her müşteriye yapılacak olan toplam taşımanın müşteri talebine eşit olmasını sağlamaktadır. İşaret kısıtı (4) ise karar değişkenlerinin pozitif ve tamsayı olmasını garantilemektedir. UP'nin genel halinde, amaç fonksiyonu katsayıları doğrusaldır ve bu problemin çözümünde şebeke simpleks (network simplex) gibi doğrusal programlama metotlarından faydalanılmaktadır [2, 3]. UP'nin çözümü için toplam talebin toplam kapasiteye eşit olması yani denge şartının sağlanması gerekir. Toplam talep toplam kapasiteden büyük ise aylak tedarikçi, tam tersi durumda ise aylak müşteri eklenmek sureti ile problemde denge şartı sağlanır. Aylak tedarikçinin kapasitesi artık talebe (= toplam talep–toplam kapasite), aylak müşterinin talebi ise artık kapasiteye (= toplam kapasite-toplam talep) eşittir.

Literatürde, UP'nin gerçek hayatta karşılaşılan durumları ve özellikleri dikkate alınarak sınıflandırıldığı görülmektedir. Bu sınıflandırma içerisinde sabit maliyetli (*fixed charge*), konkav maliyetli, tek tedarik noktalı, çok amaçlı, bazı ürün gruplarının aynı depoya taşınamadığı ve çok ürünlü gibi farklı UP yer almaktadır. Genel UP'de, birim taşıma maliyetlerinin doğrusal olduğu kabul edilmektedir. Ancak, pratikte birim taşıma maliyetlerinin taşıma miktarı ile ters orantılı olması, yani taşınacak miktar arttıkça birim taşıma maliyetinin azalması durumu ile karşılaşılmaktadır. Konkav Maliyetli UP (KMUP), bu durumu modellemektedir. KMUP'de, amaç fonksiyonu doğrusal değildir. Doğrusal olmayan amaç fonksiyonlarına örnek olarak üstel, polinomyal, konkav, parçalı konkav vb. verilebilir [4]. Bu çalışmada, amaç fonksiyonunun $f(x) = c_{ij}\sqrt{x_{ij}}$ olduğu bir KMUP ele alınmıştır. Gerçek hayat problemlerinde sık karşılaşılan bu problem, NP-zor problemler sınıfında yer almaktadır [5]. Literatürde bu problemin çözümü için doğrusal yaklaşıklama (linear approximation), Lagrangian gevşetme, dal-sınır ve dinamik programlamaya dayalı çözüm yöntemlerinin yanı sıra tavlama benzetimi, eşik kabulü gibi genel amaçlı sezgisel yöntemlerin kullanıldığı görülmektedir [4-9]. Bu çalışmada ise KMUP için genetik algoritmalara (GA) dayalı bir karma sezgisel algoritma geliştirilmiştir.

GA stokastik bir arama yöntemidir. Darwin'in "en iyi olan yaşar" prensibine dayalı olarak biyolojik sistemlerin gelişim sürecini simüle eden GA, ilk defa Holland [10] tarafından önerilmiştir. Bir GA, mümkün çözümlerin kodlandığı dizilerin bir seti (*yığın*) ile biyolojik özelliği taklit eden genetik operatörlerin (*çaprazlama ve mutasyon, vb.*) bir setinden oluşur. Rassal ya da belirlenen koşullara göre elde edilen başlangıç yığınına genetik operatörlerin uygulanması ile biyolojik gelişim süreci taklit edilmek sureti ile probleme çözüm aranır. GA, sezgisel bir yöntem olduğu için dikkate alınan probleme en iyi çözümü bulamayabilir, ancak makul zaman aralığında en iyi çözüme yakın çözümleri elde edebilir [32]. Başlangıçta doğrusal olmayan (*non-linear*) eniyileme problemlerine uygulanan GA, yaklaşık son 20 yıldır çizelgeleme [11-15], şebeke tasarımı ve güvenilirliği [16-18], gezgin satıcı [19], kargo yükleme [20] ve araç rotalama [21,22] gibi değişik mühendislik problemlerine de başarıyla uygulanmıştır.

Bu çalışmada KMUP için geliştirilen karma GA'da, yayılan ağaç tabanlı kodlama kullanılmış ve genetik operatörler ile elde edilen her yeni çözüme yerel arama algoritması uygulanmıştır. Amaç, aramanın iyi bölgelere yönlendirilmesi ile en iyi çözüme ya da en iyiye yakın çözümlere daha hızlı ulaşmayı sağlamaktır. Algoritmanın etkinliği, literatürde bu problem için geliştirilen algoritmalar ile değişik boyuttaki test problemleri üzerinde karşılaştırmalı olarak incelenmiştir.

Çalışmanın ikinci bölümünde, UP çözümü için GA'ın kullanıldığı çalışmaları inceleyen literatür araştırması yer almaktadır. Üçüncü bölümde GA'nın genel yapısı ve işleyişi, dördüncü bölümde ise KMUP için bu çalışmada geliştirilen karma GA yaklaşımı incelenmektedir. Karşılaştırmalarda kullanılan algoritmalar

hakkında detaylı bilgi beşinci bölümde yer almaktadır. Çalışmada kullanılan test problemleri, GA parametreleri, deneysel ortam ve karşılaştırmalı test sonuçları altıncı bölümde verilmektedir. Son bölümde de sonuçlar değerlendirilerek, ileri çalışmalar için önerilerde bulunulmaktadır.

2. LİTERATÜR ARAŞTIRMASI

UP için GA'yı kullanan ilk çalışmalar 1991 yılında Michalewicz ve Vignaux [23,24] tarafından gerçekleştirilmiştir. Bu çalışmalarda, çözüm kodlaması için "vektör tabanlı" ve "matris tabanlı" olmak üzere iki alternatif yaklaşım sunularak, matris tabanlı kodlama için özel çaprazlama ve mutasyon operatörleri geliştirilmiştir. Bilindiği gibi, bir çözümün kodlanmasında kullanılan yaklaşım GA'nın performansında büyük etkiye sahiptir. Bu nedenle, literatürde UP için vektör ve matris tabanlı kodlamaya alternatif kodlama yapısının araştırıldığı çalışmalar söz konusudur. Bu çalışmalardan birkaçı Gen ve Li [25-27] tarafından yapılmış olup, çözümün kodlanmasında Prüfer Sayının kullanılmasını önermişler ve bu yaklaşımı "yayılan ağaç tabanlı" kodlama olarak isimlendirmişlerdir. Çalışmada, klasik genetik operatörlerin kullanıldığı bir GA geliştirilmiş ve algoritmanın etkinliği matris tabanlı kodlamanın kullanıldığı GA ile çözüm kalitesi ve çözüm zamanı açısından karşılaştırmalı incelenmiştir. Gen vd. [26], çok amaçlı UP için yayılan ağaç tabanlı kodlamanın kullanıldığı bir GA geliştirmişlerdir. Çalışmada, ulaştırma maliyeti ve ulaştırma zamanı olmak üzere iki amaç dikkate alınmıştır. Sabit maliyetli UP için Gottlieb ve Paulmann [28], matris tabanlı kodlamanın kullanıldığı bir GA geliştirirken, Gen ve Li [25] aynı problem için yayılan ağaç tabanlı kodlamanın kullanıldığı bir GA önermişlerdir. Bazı ürün gruplarının aynı depoya taşınamadığı UP için Syarif ve Gen [29] yine yayılan ağaç tabanlı kodlamanın kullanıldığı bir GA geliştirmişlerdir. Literatürdeki diğer çalışmalarda ise UP'nin çözümü için GA'da kullanılan kodlama tiplerinin etkinliği araştırılmıştır. Örneğin Gottlieb ve Eckert [30] çalışmalarında vektör tabanlı GA ile yayılan ağaç tabanlı GA'nın karşılaştırmasını yaparak, vektör tabanlı GA'nın daha etkin olduğunu göstermişlerdir. Eckert ve Gottlieb [31] ise yukarıda verilen kodlama yapılarına alternatif olarak yeni bir kodlama sistemi önermişlerdir. UP'nin çözümünde yayılan ağacın direkt gösterimine dayalı olan bu kodlama için özel çaprazlama ve mutasyon operatörlerinin kullanıldığı çalışmada, önerilen GA'nın etkinliği yayılan ağaç tabanlı ve vektör tabanlı GA ile karşılaştırmalı incelenmiştir.

3. GENETİK ALGORİTMALAR

GA, en iyileme problemlerinin çözümünde biyolojik sistemlerin gelişim sürecini simüle eden bir stokastik arama algoritmasıdır. Sezgisel bir yöntem olduğundan dolayı eniyi (optimum) çözümü garanti etmeyen GA, bilinen yöntemlerle çözülemeyen veya çözüm zamanı üstel artan problemlerde en iyiye yakın çözümleri bulurlar. GA, parametrenin kendisi yerine parametreleri temsil eden dizileri kullanması, noktadan noktaya arama yerine noktaların bir yığınını araştırması ve stokastik olması nedeniyle bilinen optimizasyon metotlarından ayrılmaktadır. Bu nedenle GA, arama uzayının büyük ve doğrusal olmadığı (*non-linear*) ve matematiksel programlama, birerleme yöntemleri gibi deterministik en iyileme yöntemlerin başarısız olduğu problemlerin çözümünde kullanılmaktadırlar. GA, çözümlerin bir yığınını kullanarak aynı anda birçok bölgede aramayı gerçekleştirdiği için, yerel optimuma yakalanma olasılığı diğer metotlara göre daha azdır [32]. Herhangi bir problemin çözümünde kullanılan basit bir GA' nın genel yapısı Şekil 2'de verilmektedir.

```
begin
  t = 0;
  P_t başlangıç yığınını oluştur;
  P_t' yi değerlendir;
  while not {bitiş koşulu} do
    begin
      t = t +1;
      P_t-1' den P_t' yi seç;
      P_t' yi değişime uğrat;
      P_t' yi değerlendir;
    end
  end
```

Şekil 2. Genetik algoritmaların genel yapısı

Genel yapısından görüldüğü gibi basit bir GA'nın ilk aşamasında, tüm mümkün çözümlerin alt kümesinden oluşan bir başlangıç yığını elde edilir. Yığının her elemanı (*bireyi*) bir dizi olarak kodlanır. Her dizi biyolojik olarak bir kromozoma eşdeğerdir. GA'nın herhangi bir adımındaki yığın, nesil (*generation*) olarak adlandırılır. Yığındaki her dizi bir uygunluk değerine (*fitness value*) sahiptir. Uygunluk değeri, hangi bireyin bir sonraki yığına taşınacağını belirler. Bir dizinin uygunluk değeri, problemin amaç fonksiyonu değerine eşittir. Bir dizinin gücü uygunluk değerine bağlı olup iyi bir dizi, problemin yapısına göre maksimizasyon problemi ise yüksek, minimizasyon problemi ise düşük uygunluk değerine sahiptir [33]. GA'ın her iterasyonunda yeniden üretim işlemi, genetik operatörler ve uygunluk değerinin hesaplanması olmak üzere üç ana adım vardır. Mevcut yığından gelecek yığına taşınacak dizilerin seçilmesi işlemi "Yeniden Üretim İşlemi" olarak adlandırılır. Bu işlem ile özel genetik yapıların bir sonraki yığına taşınması sağlanır. Seçim işlemi ile oluşturulan yığındaki dizilerin bir kısmına uygulanan genetik operatörler, çaprazlama ve mutasyon operatörleridir. Çaprazlama operatörü, farklı diziler arasında bilgi değişimini sağlayarak yeni çözümleri elde ederken mutasyon operatörü, mevcut dizilerin bir kısmında rassal değişimi sağlayarak çözüm uzayında yeni noktaları elde etmektedir. "Uygunluk değeri", yeni yığına taşınacak dizilerin belirlenmesinde kulla-

nılan bir araçtır. Bu nedenle, algoritmanın her iterasyonunda yığındaki dizilerin uygunluk değeri (*amaç fonksiyonu değeri*) hesaplanır. Bu açıklamalar doğrultusunda, herhangi bir problemin çözümünde kullanılan bir GA'nın bileşenleri aşağıda verilmektedir [34]:

- Yığını oluşturacak bireylerin (*mümkün çözümlerin*) dizi olarak gösterimi
- Başlangıç yığınının oluşturulması
- Dizilerin uygunluğunun belirlenmesi için değerlendirme fonksiyonu
- Yeniden üretim için bir seçim mekanizması
- Yeni çözümlerin elde edilmesi için genetik operatörler
- Kontrol parametreleri (*çaprazlama ve mutasyon operatörlerinin olasılıkları ve yığın genişliği*)

4. ÖNERİLEN GA TABANLI SEZGİSEL YAKLAŞIM

4.1. Kodlama

GA'nın performansına etki eden faktörlerden birisi problemin çözümünü göstermek için seçilen kodlama tipidir. En sık kullanılan gösterim tipi ikili düzende kodlama olmasına rağmen gezgin satıcı, çizelgeleme, araç rotalama, tesis yerleşimi gibi bazı kombinatoryal en iyileme problemleri için permütasyon kodlama, rassal anahtarlar, vb. farklı kodlama yapılarının kullanıldığı görülmektedir [34]. UP için de literatürde, dört farklı kodlama yapısı vardır. Bunlar;

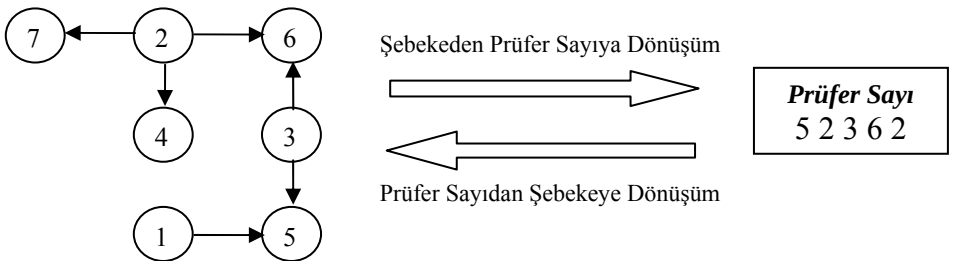
- Vektör tabanlı kodlama,
- Matris tabanlı kodlama,
- Direkt kodlama,
- Yayılan ağaç tabanlı kodlama

Matris ve vektör tabanlı kodlamada kromozomlar $|N|*|M|$ ($|N|$*: tedarikçi sayısı, $|M|$: müşteri sayısı*) boyutunda matris ve vektör, direkt kodlamada ise $2(|N| + |M| - 1)$ boyutunda bir vektör olarak tanımlanmaktadır [23,24,31]. Ancak bu kodlamalarda en büyük dezavantaj, bir çözümü göstermek için gerekli olan hafıza ihtiyacının büyük olmasıdır. Ayrıca bu tür kodlamalarda özel çaprazlama ve mutasyon operatörlerine ihtiyaç vardır. Bu dezavantajları ortadan kaldırmak amacıyla UP için Prüfer sayılara dayalı yayılan ağaç tabanlı kodlama Gen ve Li [25,27] tarafından önerilmiştir.

Prüfer [35] tarafından yönsüz şebekelerdeki yayılan ağaçları tanımlamak için geliştirilen Prüfer sayı, bir yayılan ağacı *n-2* adet *(n:düğüm sayısı)* tam sayı ile tanımlamaktadır. Tam bağlı bir şebekede n^{n-2} adet yayılan ağaç vardır ve tüm yayılan ağaçlar Prüfer sayı ile gösterilebilmektedir. Literatürde, GA ile yönsüz yayılan ağaç problemlerinin çözümünde Prüfer sayının kullanıldığı çeşitli çalışmalar vardır [36-39]. Bir UP'de, temel çözüm $|N| + |M| - 1$ hat içerir. Bu hatlar depolardan arz noktalarına giden hatlardır ve yönlü yayılan ağaçtır. Yönlü yayılan ağacı kodlamak için Gen ve Li [25], Prüfer sayının kullanılmasını öneren ilk araştırmacılardır. Bu kodlama ile mümkün bütün çözümler (*bütün yönlü yayılan ağaçlar*) kodlanabilmekte ve Prüfer sayı dizisi $|N| + |M| - 2$ adet tam sayıdan oluşmaktadır. Dolayısıyla matris ve vektör tabanlı kodlamaya göre yayılan ağaç tabanlı kodlamada daha az hafızaya gereksinim vardır. Prüfer sayının bir başka avantajı ise, Prüfer sayıdan şebekeye, şebekeden Prüfer sayıya dönüşümün mümkün olmasıdır. Ayrıca, bu tür kodlamada klasik çaprazlama (tek noktalı, çok noktalı vb.) ve mutasyon (yer değiştirme vb.) operatörleri de kulanılabilmektedir. Bu avantajlarının yanı sıra, Prüfer sayılardaki en büyük problem ise yerellik problemidir. Yani, mutasyon operatöründen sonra elde edilen kromozomun tanımladığı yayılan ağacın mevcut kromozomun yayılan ağacından çok farklı olmasıdır. Yerellik probleminin en önemli etkisi, mevcut çözüm etrafındaki çözümlerden uzaklaşmaktır [30]. Bu problemin etkisini enazlamak için bu çalışmada karma GA yaklaşımı sunulmuştur. Amaç, mutasyon operatörü sonucu kaybedilen özelliklerin kullanılan yerel arama algoritması ile yeniden elde etmek ve aramayı iyi bölgelere yönlendirmektir.

Şekil 3'de bir Prüfer sayı ve bu sayının temsil ettiği yönlü yayılan ağaç görülmektedir. Bu ağaç, 3 tedarikçiye ve 4 müşteriye sahip ulaştırma şebekesinden elde edilmiştir. P(T); orijinal Prüfer sayı ve $P(\overline{T})$; Prüfer sayı içinde olmayan sayıların kümesi olmak üzere Prüfer sayıdan şebekeye ve şebekeden Prüfer sayıya dönüşüm algoritmaları Şekil 4 ve Şekil 5'de verilmektedir.

Rassal olarak üretilen ya da genetik operatörlerden sonra elde edilen Prüfer sayı bazen UP için bir yönlü yayılan ağacı göstermeyebilir. Böyle bir durumda kodlama üzerinde bir onarım işlemi gerçekleştirilir. Gen ve Li [25] bu amaçla bir onarım algoritması geliştirmişlerdir. Bilindiği gibi, yönlü yayılan ağaçta



Şekil 3. Prüfer Sayıdan şebekeye, şebekeden Prüfer Sayıya dönüşümler

Adım1: i: $P(\overline{T})$ kümesindeki en küçük numaralı düğüm,
j: P(T) kümesinin en solundaki düğüm.
Adım2: Eğer i ve j düğümleri aynı küme içinde değilse (her ikisi de tedarikçi ya da müşteri) (i,j) hattını ulaştırma ağacına ekle. Değilse, P(T) kümesinden i ile aynı kümede olmayan ilk düğümü (k) seç ve (i,k) hattını ulaştırma ağacına ekle.
Adım3: j (veya k) düğümünü P(T), i düğümünü de $P(\overline{T})$ kümesinden sil. Eğer j (veya k) düğümü P(T) kümesinde bir daha bulunmuyorsa j (veya k) düğümünü $P(\overline{T})$ kümesine ekle.
Adım4: Ulaştırma ağacına eklenen hatta mümkün olan en büyük taşıma miktarını ata:
$x_{ij} = \min\{S_i, D_j\}$ *veya* $x_{ik} = \min\{S_i, D_k\}; i \in S:\ j,k \in D$
Adım5: Mevcut tedarik miktarları ve müşteri taleplerini güncelleştir.
$S_i = S_i - x_{ij}$ ve $D_j = D_j - x_{ij}$ veya $D_k = D_k - x_{ik}$
Adım6: P(T) kümesinde eleman varsa Adım 1'e, değilse Adım **7**'ye git.
Adım7: P(T) kümesinde eleman yoksa, $P(\overline{T})$ kümesinde kesinlikle iki düğüm kalmıştır. Bu iki düğümü ulaştırma ağacına ekle ve mevcut miktarları güncelleştir. Mevcut durumda ulaştırma şebekesinde $|N|+|M|-1$ adet hat atama-sı yapılmış olacaktır.
Adım8: Tedarikçi kapasiteleri ve müşteri taleplerinde atanmamış miktar kalmamışsa dur. Değilse, tedarikçiler kümesinde atanmamış kapasitesi bulunan bir düğüm (r), müşteriler kümesinde de karşılanmamış talebi bulunan bir düğüm (t) vardır. *(r,t)* hattını ulaştırma şebekesine ekle ve atanabilecek miktar atamasını $(x_{rt} = S_r = D_t)$ gerçekleştir. Atanmış miktarı "0" birim olan ve *(r,t)* ataması ile döngü yaratan hattı ulaştırma ağacından sil ve dur.

Şekil 4. Prüfer Sayıdan şebekeye dönüşüm algoritması

Adım1: i: Ulaştırma şebekesindeki en küçük numaralı uç düğüm
j: i d üğümünün öncül düğümü
Adım2: j düğümünü Prüfer sayının sonuna ekle.
Adım3: Bir daha hesaplamaya katılmaması için (i,j) hattını ulaştırma şebekesinden çıkar.
Adım4: Ulaştırma şebekesinde sadece iki düğüm kalmışsa dur, değilse Adım 1'e git.

Şekil 5. Şebekeden Prüfer Sayıya dönüşüm algoritması

depolardan çıkan hat sayısı talep noktalarına giren hat sayısına eşit olması gerekir. Geliştirilen onarım algoritmasının temeli yönlü yayılan ağacın bu özelliğine dayalıdır. Bir Prüfer sayının temsil ettiği yönlü yayılan ağacın bu özelliğe sahip olup olmadığı (5) nolu eşitlik ile kolayca kontrol edilebilir.

$$\sum_{i=1}^{|N|} (L_i + 1) = \sum_{i=|N|+1}^{|N|+|M|} (L_i + 1) \qquad (5)$$

L_i: Prüfer Sayı içindeki i düğümünün sayısı

Eğer bir Prüfer sayıda (5) nolu eşitlik sağlanmamış ise, Prüfer sayı yönlü yayılan ağacı temsil etmemektedir. Bu durumda Şekil 6'daki onarım algoritması kullanılarak yönlü yayılan ağacı temsil eden bir Prüfer sayı elde edilir.

Adım1: Değişkenlere başlangıç değerlerini ata ($t_s=0$, $t_d=0$).
Adım2: Prüfer Sayının t_s ve t_d değerlerini hesapla.
$$t_s = \sum_{i=1}^{|N|} (L_i + 1)\ ;\ t_d = \sum_{i=|N|+1}^{|N|+|M|} (L_i + 1)$$
Adım3: $t_s = t_d$ *ise* dur, d*eğilse* Adım 4'e git.
Adım4: [1, $|N| + |M| - 2$] arasında düzgün dağılımdan bir sayı üret, $r \sim U[1, |N| + |M| - 2]$
Adım5: $t_s < t_d$ *ise prüfer(r)=U*[1,|N|], *değilse prüfer(r)=U*[|N|+1,|N|+|M|], Adım 2'ye git.

Şekil 6. Onarım algoritması

4.2. Uygunluk Değerinin Hesaplanması ve Seçim Mekanizması

Uygunluk değeri yeni yığına taşınacak dizilerin belirlenmesinde kullanılan bir araçtır. Bu çalışmada uygunluk değeri olarak, Prüfer sayının temsil ettiği ulaştırma ağacının (*yönlü yayılan ağaç*) amaç fonksiyonu değeri dikkate alınmıştır.

Seçim mekanizması, mevcut yığından bir sonraki nesle hangi aday çözümlerin (*kromozom*) aktarılacağını belirler. Literatürde çok çeşitli seçim mekanizmaları önerilmiştir [32,33]. Bunlardan en yaygın kullanılanları; $(\mu+\lambda)$, rulet çemberi, turnuva seçim mekanizmalarıdır. Bu çalışmada $(\mu+\lambda)$ seçim mekanizması kullanılmıştır. Bu mekanizmada, yığındaki kromozomlar amaç fonksiyonu değerlerine göre sıralanmakta, tekrar eden kromozomlar silinmekte ve yığın genişliği kadar en iyi kromozom bir sonraki yığına aktarılmaktadır. Seçim esnasında yığın genişliğinden daha az sayıda kromozom bir sonraki yığına aktarılmış ise, yığın genişliğine ulaşıncaya kadar rassal olarak üretilen yeni kromozomlar yığına eklenmektedir.

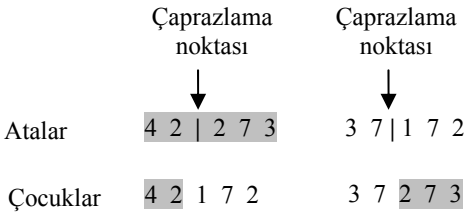
4.3. Genetik Operatörler

GA'nın diğer önemli bileşenlerinden biriside genetik operatörlerdir. Bu operatörler çözüm uzayında bir noktadan diğer bir noktaya geçişi sağlamaktadır. GA'da kullanılan başlıca genetik operatörler çaprazlama ve mutasyon operatörleridir. Bu bölümde, Prüfer

sayı için bu çalışmada kullanılan genetik operatörler açıklanmaktadır.

4.3.1. Çaprazlama

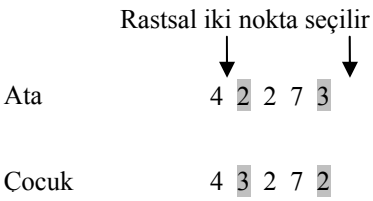
Çaprazlama, yığındaki iki bireyin özelliklerinin harmanlanarak bir sonraki yığına aktarılması işlemidir. Bu işlem, arama uzayında yeni bölgelere ulaşmayı sağlar. Bu çalışmada klasik çaprazlama operatörü olan tek noktalı çaprazlama operatörü kullanılmıştır. Bu operatörde mevcut iki kromozomun, rassal olarak belirlenen çaprazlama noktasından sonraki parçaları değiştirilir. Dolayısıyla elde edilen yeni kromozomlar, çaprazlama için kullanılan iki kromozomun bazı özelliklerini taşımaktadır. Şekil 7, tek noktalı çaprazlama işlemini göstermektedir.



Şekil 7. Tek noktalı çaprazlama

4.3.2. Mutasyon

Mutasyon operatörü, mevcut çözümün civarında aramayı gerçekleştirir ve mevcut yığında çeşitliliğin artmasını sağlar. Bu çalışmada kullanılan mutasyon operatörü olarak değiştirme (*swap*) operatörü kullanılmıştır. Bu operatörde kromozomdan rassal olarak seçilen iki birim yer değiştirmek sureti ile yeni kromozom elde edilmektedir. Şekil 8'de mutasyon operatörünün uygulaması verilmiştir.



Şekil 8. Mutasyon operatörü

4.4. GA ile Yerel Arama Algoritmasının Melezlenmesi

GA'da önemli bir başka konu, eniyi çözüme yakınsamadır. GA, yeniden üretim mekanizması ile çözüm uzayının değişik noktalarına sıçramalar yapılabilmektedir. Ancak, uzayın bu noktasına yakın daha iyi çözümlerin olup olmadığının (yerel eniyi) kontrolü yapılmamaktadır. Bu noktalara yakınsama sağlamak için değişik yöntemler geliştirilmiştir. Bu çalışmada da, algoritmanın etkinliğini arttırmak ve Prüfer sayının en büyük dezavantajı olan yerellik probleminin etkisini enazlamak için GA'ya bir yerel arama algoritması adapte edilmiş ve bu algoritma GA ile elde edilen her yeni kromozoma uygulanmıştır.

Adım1: k=1, Δ=*0, max_search:* Her aşamada incelenecek hat sayısı.
Adım2: Seçilmemiş hatlar arasından rassal olarak bir hat seç ve ulaştırma ağacına ekle.
Adım3: Oluşan çevrim içerisinden çıkacak olan hattı belirle.
Adım4: Hatlar üzerinde taşınacak miktarları düzenle ve iyileşme miktarını (Δ_k) belirle.
Adım5: $\Delta_k > \Delta$ ise yeni çözümü en iyi çözüm olarak kabul et. $k=k+1$ ve $\Delta=\Delta_k$.
Adım6: $k <$ *max_search* ise Adım 2*'* ye git, değilse dur.

Şekil 9. Yerel arama algoritması

Yerel arama algoritması, klasik UP için kullanılan "sıçrama taşı" yöntemine eşdeğerdir. Sıçrama taşı yönteminde mevcut çözümden bir sonraki yerel eniyi çözüme dual değişkenler kullanılarak, doğrusallık varsayımı altında geçilmektedir. Ancak, KMUP de amaç fonksiyonu katsayıları doğrusal değildir. Bu nedenle, yöntem üzerinde bir düzenleme yapılmıştır. Bu düzenlemeye göre, çözümde bulunmayan hatlar arasından rassal olarak hatların bir kümesi seçilmiş ve küme içinde bulunan hatlar içinde mevcut çözümde en çok iyileştirmeyi yapan hat çözüme dahil edilmiştir. Yerel arama algoritmasının adımları Şekil 9'da verilmektedir.

5. KARŞILAŞTIRMADA KULLANILAN ALGORİTMALAR

Bu çalışmada geliştirilen karma GA'nın etkinliği Yan ve Luo [4] tarafından KMUP için geliştirilen algoritmalar ile incelenmiştir. Yan ve Luo [4], genel amaçlı sezgisel algoritmalar sınıfında yer alan tavlama benzetimi ve eşik kabulü algoritmalarına dayalı üç algoritma geliştirmiş ve algoritmaların performansını değişik boyutlardaki test problemleri ile incelemişlerdir. Bu bölümde, bu algoritmaların genel özellikleri enküçükleme problemine göre incelenmektedir.

Tavlama Benzetimi (TB): TB, katı cisimlerin tavlama işlemlerinden esinlenilerek geliştirilmiş stokastik bir arama yöntemidir. Bu yöntemde, yerel en iyi çözümlerden kurtularak genel eniyi çözüme ulaşabilmek için kötü çözümler belirli olasılık ile kabul edilmektedir. TB, bir başlangıç çözümünden (S) başlayarak komşu arama mekanizmaları ile yeni çözüme (S') ulaşmaktadır. Amaç fonksiyonundaki değişim değeri ($\Delta = f(S') - f(S)$) hesaplanarak iyi bir çözüm elde edildi ($\Delta < 0$) ise bu çözüm mevcut çözüm olarak dikkate alınırken, kötü bir çözüm ($\Delta > 0$) $e^{-\Delta/T}$ olasılığı ile mevcut çözüm olarak kabul edilmektedir. Burada, T sıcaklık parametresidir. Belirli sayıdaki iterasyondan sonra T sıcaklığı, [0,1] aralığında bir katsayı ile çarpılarak düşürülmektedir. Dolayısıyla, algoritmanın ilerleyen iterasyonlarında kötü çözümlerin kabul olasılığı düşerek arama iyi çözümler etrafında yoğunlaşmaktadır. T sıcaklığı belirli bir seviyenin altına düştüğünde algoritma sonlandırılmaktadır. Bir problemin çözümü

için TB kullanılırken, başlangıç sıcaklığı (T_0), son sıcaklık (T_f), sıcaklık düşürme fonksiyonu, komşu arama yöntemi ve her sıcaklıkta aranacak komşu sayısı gibi bir takım kontrol parametrelerinin belirlenmesi gerekmektedir. Yan ve Luo'nun [4] KMUP için geliştirdiği TB algoritmasında T_0=10, T_f = 1 olarak alınmış ve ardışık 50 yeni çözüm mevcut sıcaklığa göre kabul edilmez ise sıcaklık 0.9 oranında düşürülmüştür. Mevcut çözümden komşu arama yöntemine göre $|N|+|M|$ komşu üretilerek, amaç fonksiyonunu eniyileyen çözüm yeni çözüm olarak dikkate alınmıştır. Komşu arama yöntemi olarak Bölüm 4.4.'de açıklanan yerel arama algoritması kullanılmıştır.

Eşik Kabulü (EK): EK algoritması temelde TB algoritmasına benzemektedir. TB'de olduğu gibi mevcut çözümden (S) yeni çözüm (S') bir komşu arama mekanizması ile elde edilmekte ve amaç fonksiyonundaki değişim değeri ($\Delta = f(S') - f(S)$) hesaplanmaktadır. $\Delta < 0$ ise, yeni çözüm mevcut çözüm olarak kabul edilmektedir. $\Delta > 0$ durumunda yeni çözümün kabul edilmesi için TB'deki olasılıklı yaklaşım yerine eşik değerinden (T_h) yararlanılmaktadır. Bu durumda $\Delta > 0$ ve $\Delta < T_h$ ise yeni çözüm mevcut çözüm olarak dikkate alınmakta ve bu çözümden aramaya devam edilmektedir. Her iterasyonda eşik değeri TB'de olduğu gibi bir fonksiyon yardımı ile düşürülmektedir. Arama işlemi, belirli sayıda eşik değeri için arama yapıldığında yada eşik değeri belirlenen bir değerin altına düştüğünde sonlanmaktadır. EK, bir problemin çözümü için kullanıldığında, başlangıç eşik değeri, eşik değerini düşürme fonksiyonu, kullanılacak toplam eşik sayısı veya son eşik değeri, komşu arama yöntemi ve her eşik değerinde aranacak komşu sayısı gibi parametrelerin belirlenmesi gerekmektedir. Yan ve Luo [4] EK'ya dayalı olarak geliştirdikleri algoritmalarında mevcut çözümden $2|N|+2|M|$ komşu elde edilmekte ve amaç fonksiyonunu eniyileyen komşu yeni çözüm olarak kabul edilmektedir. Komşu arama yöntemi olarak TB' de kullanılan yaklaşım kullanılmıştır. Algoritmada, başlangıç eşik değeri 0.39 olarak dikkate alınmış ve ardışık 50 yeni çözüm mevcut eşik değerine göre kabul edilmez ise eşik değeri 0.9 oranında düşürülmüştür. Sonlandırma kriteri olarak 60 eşik değeri dikkate alınmıştır.

Doğrusal Eşik Kabulü (DEK): Diğer iki yöntemde olduğu gibi DEK yönteminde de mevcut çözümden (S), yeni bir çözüm (S') komşu arama yöntemine göre elde edilerek amaç fonksiyonundaki değişim değeri ($\Delta = f(S') - f(S)$) hesaplanmaktadır. $\Delta < 0$ ise, yeni çözüm mevcut çözüm olarak kabul edilirken, $\Delta > 0$ ise yeni çözüm (6) nolu eşitlikteki olasılık değerine göre kabul edilmektedir. Eşitlikteki p_0, kabul olasılığı alt sınır değerini vermektedir.

$$p = 1 + \left(\frac{p_0 - 1}{T_h f(S)} \right) * \Delta \qquad (6)$$

Yan ve Luo [4], DEK yöntemine dayalı olarak geliştirdikleri algoritmalarında p_0 değerini 0.25 olarak almışlardır. DEK'da kullanılan diğer kontrol parametreleri EK algoritmasında kullanılanlara eşdeğerdir.

Yan ve Luo'nun [4] geliştirdiği her üç algoritma da aynı başlangıç çözümü ile aramaya başlamaktadır. Başlangıç çözümünün elde edilmesinde tedarikçiler ve müşteriler arasındaki hatlar maliyetlerine göre küçükten büyüğe sıralanmaktadır. En küçük maliyetli hattan başlamak sureti ile ulaştırma şebekesi elde edilmektedir.

6. DENEYSEL ÇALIŞMA

Bu bölümde karşılaştırmalarda kullanılacak test problemlerinin oluşturulması ve karma GA'daki parametrelerin belirlenmesi açıklanarak bu sezgisel yöntemin etkinliği, Yan ve Luo [4] tarafından geliştirilen TB, EK ve DEK kullanılarak üretilen test problemler üzerinde karşılaştırmalı incelenmiştir. Ayrıca, GA'nın yerel arama yöntemi ile melezlenmesinin algoritmanın performansı üzerindeki etkisi, test problemlerinin çözümünde klasik GA ve yerel arama yöntemi ayrı ayrı kullanılmak sureti ile araştırılmıştır. Karşılaştırmalarda performans ölçüsü olarak çözüm kalitesi ve çözüm zamanı dikkate alınmıştır. Tüm sezgisel yöntemler C++ programlama dilinde kodlanmış ve denemeler Pentium 4, 2.8 GHz hızında 512 MB hafızaya sahip bilgisayarda yapılmıştır.

6.1. Test Problemlerinin Oluşturulması ve GA Parametrelerinin Belirlenmesi

Geliştirilen karma GA'nın etkinliğinin araştırılmasında, küçük ve orta boyutlu olmak üzere iki tür test problemi kullanılmıştır. Her iki tür test probleminde 6 problem olmak üzere toplam 12 problem dikkate alınmıştır. Eniyi çözümü bilinen küçük boyutlu test problemlerinde tedarikçi ve müşteri sayısı 4 ile 6 arasında, orta boyutlu problemlerde ise tedarikçi ve müşteri sayısı 10 ile 40 arasında değişmektedir. Küçük boyutlu problemlerde eniyi çözüm birerleme yöntemi (BY) ile elde edilmiştir. Tedarikçi kapasiteleri ile müşteri talepleri toplamının eşit olduğu kabul edilerek, her problem boyutu için toplam kapasite ve toplam talep $|N|*|M|*100$ olarak belirlenmiştir. Her problem boyutunda, toplam talep (*kapasite*) müşteriler (*tedarikçiler*) arasında rassal olarak dağıtılmış ve i. tedarikçiden j. müşteriye taşıma maliyeti 3 ile 8 arasında düzgün dağılımdan üretilmiştir. Tablo 1 ve 2'de, küçük ve orta boyutlu test problemleri ile her problemdeki toplam kapasite (toplam talep) verilmektedir.

Karma GA'da kullanılan yerel arama algoritmasında her çözüm için denenecek hat sayısı $|N|+|M|$ olarak alınmıştır. Kontrol parametreleri olan çaprazlama oranı (p_c), mutasyon oranı (p_m) ve yığın genişliği (YG) ise yapılan ön denemeler sonucunda belirlenmiştir. p_c için 0.2, 0.4, 0.5 ve 0.8, p_m için 0.1,

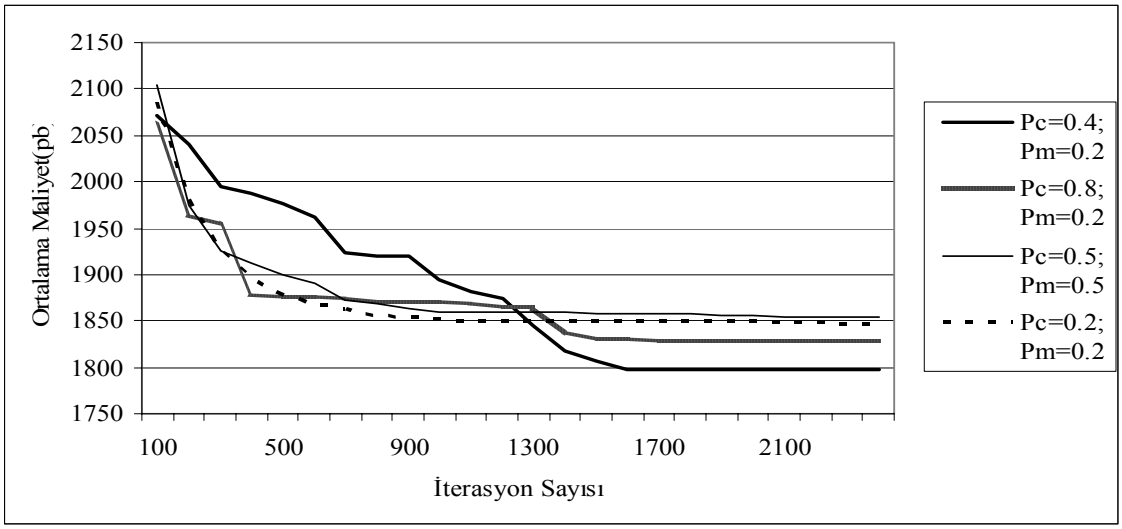
Tablo 1. Küçük boyutlu test problemleri için toplam kapasite (talep)

Problem (|*N*| x |*M*|)	Toplam Kapasite
4 x 4	1600
4 x 5	2000
4 x 6	2400
5 x 4	2000
5 x 5	2500
6 x 4	2400

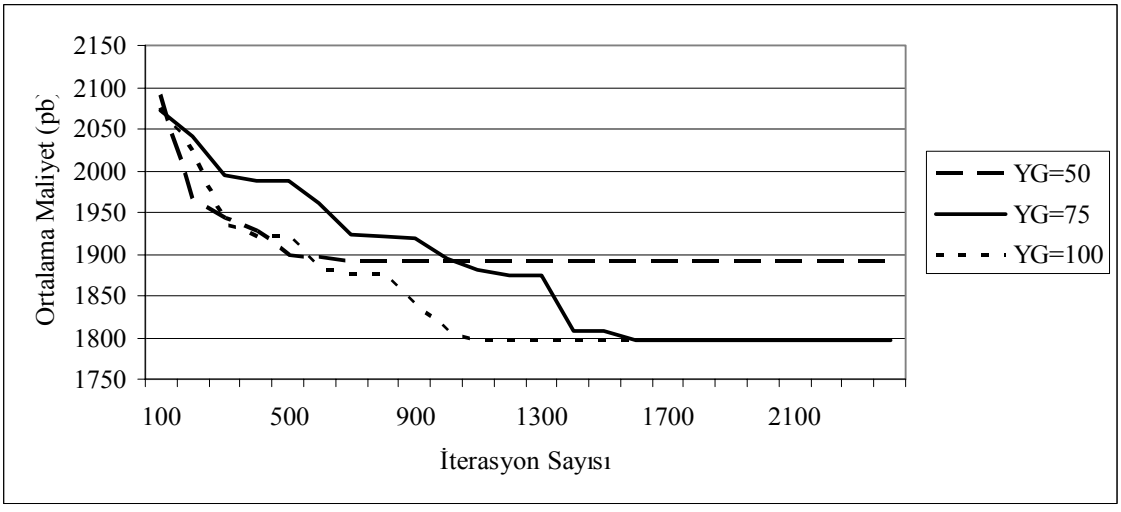
Tablo 2. Orta boyutlu test problemleri için toplam kapasite (talep)

Problem (|*N*| x |*M*|)	Toplam Kapasite
10 x 10	10000
12 x 12	14400
20 x 20	40000
25 x 25	62500
30 x 30	90000
40 x 40	160000

0.2 ve 0.3, YG için 50, 75 ve 100 değerleri alınmıştır. İlgili parametrelerin en iyi kombinasyonun belirlenmesi için çeşitli problemlerde algoritmanın iyi çözüme yakınsaması incelenmiştir. Seçilen problemler için 10 deneme yapılarak, her 100 iterasyonda bu iterasyona kadar bulunan iyi sonuç tutulmuştur. Dikkate alınan bazı kombinasyonlarda 12x12 KMUP için karma GA'nın yakınsama grafikleri Şekil 10 ve 11'de verilmektedir. Dört farklı p_c ve p_m kombinasyonu için Şekil 10'da verilen yakınsama grafiğinde YG 75 olarak alınmıştır. Şekil 10 incelendiğinde, p_c =0.4 ve p_m=0.2 kombinasyonu için algoritmanın diğer kombinasyonlara göre daha yavaş bir şekilde daha iyi çözüme yakınsadığı görülmektedir. Bu özellik söz konusu kombinasyon için algoritmanın çözüm uzayını daha iyi arama kabiliyetine sahip olduğunu göstermektedir. Bu nedenle farklı YG değerleri için algoritmanın yakınsamasının araştırıldığı Şekil 11'de çaprazlama ve mutasyon oranları için seçilen kombinasyon dikkate alınmıştır. Şekil 11 incelendiğinde, YG'nin 75 ve 100 değerleri için algoritmanın aynı çözüme yakınsadığı görülmektedir. Sabit iterasyon sayısı için yığın genişliği arttıkça algoritmanın çözüm süresi artacağından dolayı YG 75 olarak alınmıştır.



Şekil 10. Karma GA'nın farklı çaprazlama ve mutasyon oranları için yakınsama grafiği



Şekil 11. Karma GA'nın farklı yığın genişlikleri için yakınsama grafiği

Tablo 3. Küçük boyutlu problemler için algoritmaların karşılaştırılması

		Problem					
		4x4	4x5	4x6	5x4	5x5	6x4
BY	En Iyi Maliyet	455	507	508	516	622	582
	OÇS(sn)	0.66	10.09	105.03	7.48	197.09	104.09
EK	En Iyi Maliyet	455	507	508	516	622	582
	OÇS(sn)	0.03	0.05	0.06	0.06	0.06	0.07
TB	En Iyi Maliyet	455	507	508	516	622	582
	OÇS(sn)	0.06	0.07	0.09	0.08	0.08	0.09
DEK	En Iyi Maliyet	455	507	508	516	622	582
	OÇS(sn)	0.06	0.06	0.08	0.07	0.07	0.07
Karma GA	En Iyi Maliyet	455	507	508	516	622	582
	OÇS(sn)	0.05	0.06	0.09	0.07	0.07	0.08

Çalışmada kullanılan tüm algoritmalar tüm test problemleri için 10 kez denenmiştir. Algoritmaların karşılaştırılmasında performans ölçütleri olarak 10 deneme sonucunda bulunan en iyi çözümün maliyeti, ortalama maliyet ve ortalama çözüm zamanı (OÇS) dikkate alınmıştır. Küçük boyutlu problemlerde karma GA, en iyi çözüme ulaştığında ya da $(|N|+|M|)*100$ çözüm aradıktan sonra sonlanırken orta boyutlu problemler için en iyi çözüm bilinmediği için durdurma koşulu olarak aranan çözüm sayısı dikkate alınmıştır. Bu değer problem boyutuna bağlı olup $(|N|+|M|)*100$ ile hesaplanmaktadır. TB, EK ve DEK için durdurma koşulları, orta boyutlu problemlerde Bölüm 5'de verildiği gibi dikkate alınmıştır. Küçük boyutlu problemlerde ise en iyi çözüme ulaşıldığında veya ilgili durdurma koşulu sağlandığında algoritmalar sonlandırılmaktadır.

6.2. Deney Sonuçların Değerlendirilmesi

Tablo 3'de, küçük boyutlu problemler için BY ile elde edilen en iyi çözümün maliyeti ve çözüm zamanı ile birlikte TB, EK, DEK ve karma GA ile 10 deneme sonucunda bulunan en iyi çözümün maliyeti ve her bir algoritmanın OÇS verilmektedir. Tüm algoritmalar 10 denemenin tümünde en iyi çözüme ulaştığı için sadece en iyi çözüm değerleri dikkate alınmıştır. Algoritmaların OÇS satırı incelendiğinde, BY'ye göre çok kısa zamanda en iyi çözüme ulaştığı görülmektedir. Tüm algoritmalar için OÇS değerleri birbirine çok yakın olmasına rağmen, EK'nin en iyi çözüme biraz daha hızlı yakınsadığı görülmektedir.

Orta boyutlu problemler için TB, EK, DEK ve karma GA ile 10 deneme sonucunda bulunan en iyi maliyet, ortalama maliyet, en iyi maliyetten sapma yüzdesi (hata) ve OÇS Tablo 4'de verilmektedir. Problemlerin en iyi çözümleri bilinmediğinden her problem boyutunda her algoritmanın hata oranı, ilgili problem için algoritmalar ile bulunan en iyi çözümün maliyeti dikkate alınarak hesaplanmıştır. Tabloda her problem için bulunan en iyi çözümün maliyeti koyu renk ile belirtilmiştir. Her problem boyutunda algoritmalar

Tablo 4. Orta boyutlu problemler için algoritmaların karşılaştırılması

		Problem					
		10x10	12x12	20x20	25x25	30x30	40x40
TB	En Iyi Maliyet	1344.64	1797.54	3556.46	4594.42	6421.9	9203.19
	Ortalama Maliyet	1344.64	1817.39	3675.30	4652.99	6479.69	9311.72
	Hata(%)	0.00	0.00	2.24	1.5	8.77	0.34
	OÇS(sn)	0.31	0.47	1.6	2.65	4.23	9.08
EK	En Iyi Maliyet	1344.64	1797.54	3585.5	4670.25	6386.73	9276.16
	Ortalama Maliyet	1349.67	1836.8	3649.6	4680.31	6471.37	9340.64
	Hata(%)	0.00	0.00	3.07	3.17	8.18	1.14
	OÇS(sn)	1.56	2.33	7.47	12.90	20.16	41.62
DEK	En Iyi Maliyet	1344.64	1797.54	3663.89	4616.09	6224.40	9311.89
	Ortalama Maliyet	1349.67	1824.81	3682.97	4684.23	6413.81	9375.38
	Hata(%)	0.00	0.00	5.32	1.97	5.43	1.53
	OÇS(sn)	1.66	2.53	7.82	13.18	21.06	43.27
Karma GA	En Iyi Maliyet	1344.64	1797.54	3478.66	4526.70	5904.00	9172.00
	Ortalama Maliyet	1344.64	1797.54	3508.79	4536.68	5970.34	9231.02
	Hata(%)	0.00	0.00	0.00	0.00	0.00	0.00
	OÇS(sn)	20.65	29.96	176.89	456.82	1106.94	3491.02

karşılaştırmalı incelendiğinde, ilk iki problem için karma GA diğer yöntemler kadar iyi çözüm elde ederken diğer dört problem için TB, EK ve DEK'den daha iyi çözümleri elde ettiği görülmektedir. Ayrıca her algoritmanın ortalama maliyet satırı incelendiğinde, karma GA ilk iki problemin 10 denemesinde de aynı çözüme ulaşırken, TB'nin bu performansı sadece birinci problem için gösterebildiği görülmektedir. Ayrıca diğer problemler için karma GA'nın ortalama çözüm kalitesi açısından TB, EK ve DEK'dan daha iyi performansa sahip olduğu görülmektedir. Diğer algoritmalar arasından TB, beşinci problem haricinde çözüm kalitesi ve çözüm zamanı açısından daha iyi bir performansa sahiptir. Karma GA ile TB karşılaştırıldığında, karma GA ile çözüm kalitesinde %0.3 ile %5 arasında iyileşmenin olduğu görülmektedir. Ortalama çözüm süreleri açısından inceleme yapıldığında, karma GA'nın diğer algoritmalara göre daha uzun sürelere sahip olduğu görülmektedir. Bilindiği gibi karma GA'da bir çözüm (yönlü yayılan ağaç) Prüfer sayılar ile tanımlanmıştır. Dolayısıyla her iterasyonda her çözümün maliyetinin hesaplanması için öncelikle Prüfer sayıdan yayılan ağaca dönüşümün gerçekleştirilmesi gerekmektedir. Bu işlem ve her iterasyondaki çözümlere yerel arama algoritmasının kullanılması karma GA'nın çözüm zamanına ek bir yük getirmektedir. GA'nın yerel arama algoritması ile melezlenmesinin algoritmanın performansı üzerindeki etkisini incelemek için orta boyutlu problemler klasik GA (kGA) ve yerel arama yöntemi (YAY) kullanılarak yeniden çözülmüştür. Sonuçlar Tablo 5'de verilmektedir. kGA'nın karma GA'dan farkı, genetik operatörler sonucunda elde edilen çözümlere yerel arama algoritmasının uygulanmamasıdır. YAY'da ise bir başlangıç çözümünden başlanmakta ve bir sonraki yerel çözüm komşu arama mekanizması kullanılarak elde edilen $|N|+|M|$ adet yeni çözüm arasından maliyeti en azlayan çözüm olarak seçilmektedir. Durdurma koşulu sağlanıncaya kadar bu işleme devam edilerek elde edilen en iyi çözüm problemin çözümü olarak dikkate alınmaktadır. Bu algoritmada da durdurma koşulu olarak aranan çözüm sayısı kullanılmıştır. Tablo 5 incelendiğinde, kGA'nın en iyi ve ortalama maliyet açısından YAY'a göre daha iyi sonuçlar ürettiği görülmektedir. Bu durum, genetik operatörler ile aramanın etkin bir şekilde gerçekleştiğinin bir göstergesidir. Karma GA ise aynı performans ölçütleri için kGA ve YAY'dan daha iyi bir performans sergilemektedir. Bu sonuç, literatürde karma yaklaşımlar için elde edilen sonuçlara paraleldir. GA, arama uzayının farklı bölgelerini iyi çözümler üzerinde uygulanan genetik operatörler ile araştırırken yerel arama algoritması, yığındaki çözümleri probleme özgü bilgi kullanarak iyileştirmektedir. kGA ve karma GA'nın ortalama çözüm süreleri birbirine çok yakın olması, yerel arama algoritmasının karma GA'nın çözüm süresi üzerinde çok büyük etkisinin olmadığını göstermektedir. Bu durum, YAY'ın ortalama çözüm süreleri incelendiğinde de görülmektedir. kGA ve karma GA'da, Prüfer sayıdan şebekeye ve şebekeden Prüfer sayıya dönüşüm işlemi çözüm zamanında büyük bir etkiye sahiptir. Dolayısıyla, YAY'a göre her iki algoritmada da çözüm zamanı daha büyüktür. YAY çok kısa sürede çözüme ulaşmasına rağmen, çözüm kalitesi açısından iyi bir performans sergileyememektedir.

7. SONUÇ ve ÖNERİLER

Bu çalışmada, KMUP için yayılan ağaç tabanlı kodlamanın kullanıldığı bir karma GA geliştirilmiştir. Geliştirilen karma GA'da, genetik operatörler ile elde edilen her yeni çözüme yerel arama algoritması uygulanmıştır. Amaç, aramanın iyi bölgelere yönlendirilmesi ile en iyi çözüme ya da en iyiye yakın çözümlere ulaşmayı sağlamaktır. Geliştirilen karma GA'nın etkinliğinin araştırılmasında, küçük ve orta boyutlu olmak üzere iki tür test problemi kullanılmıştır. Her iki tür test probleminde 6 problem olmak üzere toplam 12 problem dikkate alınmıştır. Küçük boyutlu problemler için en iyi çözüm BY yöntemi ile elde edilmiştir. Her iki boyuttaki problemler için karma GA'nın performansı literatürde bu problemin çözümü için önerilen TB, EK ve DEK ile karşılaştırmalı ola-

Tablo 5. Orta boyutlu problemler için YAY, kGA ve karma GA'nın karşılaştırılması

		Problem					
		10x10	12x12	20x20	25x25	30x30	40x40
YAY	En Iyi Maliyet	1671.43	2338.56	5256.26	7362.60	9950.96	15545.11
	Ortalama Maliyet	1708.42	2373.97	5274.46	7481.17	10037.05	15685.04
	Hata(%)	24.30	30.10	51.10	62.65	68.55	69.48
	OÇS(sn)	0.17	0.26	3.66	10.82	25.84	144.71
kGA	En Iyi Maliyet	1423.42	2088.92	4795.02	6853.48	9604.07	14731.88
	Ortalama Maliyet	1456.06	2144.65	4870.86	7028.41	9713.59	15038.12
	Hata(%)	5.86	16.21	37.84	51.40	62.67	60.62
	OÇS(sn)	20.11	29.51	171.26	448.88	1079.23	3355.64
Karma GA	En Iyi Maliyet	1344.64	1797.54	3478.66	4526.70	5904.00	9172.00
	Ortalama Maliyet	1344.64	1797.54	3508.79	4536.68	5970.34	9231.02
	Hata(%)	0.00	0.00	0.00	0.00	0.00	0.00
	OÇS(sn)	20.65	29.96	176.89	456.82	1106.94	3491.02

rak incelenmiştir. Yapılan karşılaştırmada, karma GA'nın en az diğer yöntemler kadar iyi çözüme ya da daha iyi çözümlere ulaştığı görülmüştür. Diğer algoritmalar arasından TB, çözüm kalitesi ve çözüm zamanı açısından daha iyi bir performansa sahiptir. Karma GA ile TB karşılaştırıldığında, karma GA ile çözüm kalitesinde %0.3 ile %5 arasında iyileşmenin olduğu görülmektedir. Ayrıca, GA'nın bir yerel arama algoritması ile melezleştirilmesinin algoritmanın performansı üzerinde etkisini araştırmak için orta boyuttaki problemler klasik GA ve yerel arama algoritması ile yeniden çözülmüştür. Yerel arama algoritmasının çözüm süresi klasik GA ve karma GA'dan çok daha küçük olmasına rağmen çözüm kalitesi açısından bu iki algoritma kadar iyi performans sergileyemediği görülmüştür. Karma GA ise hem yerel arama algoritmasından hem de klasik GA'dan çözüm kalitesi açısından çok daha iyi bir performansa sahiptir. Problem büyüklüğüne bağlı olarak karma GA'da çözüm süresi yaklaşık 0.05 saniye ile 1 saat arasında değişmektedir. Karma GA'nın her iterasyonunda yığındaki çözümlerin maliyetini hesaplamak için Prüfer sayıdan yayılan ağaca dönüşüm işlemi ve her çözüme yerel arama algoritmasının uygulanması algoritmanın çözüm süresini arttırmaktadır. Ancak, ulaştırma şebekesinin uzun dönem için kullanılacağı düşünüldüğünde, günlük bazda elde edilen en az %0.3'lük bir getirinin uzun dönemdeki etkisi çok daha fazla olacaktır. Dolayısıyla, bu tür bir getiriyi elde etmek için karma GA ile katlanılması gereken süre çok uzun bir süre değildir.

Bu çalışmanın devamında yapılabilecek çalışmalar aşağıdaki gibi özetlenebilir: Geliştirilen karma GA'da kodlama için Prüfer sayılar kullanılmıştır. Literatürde, UP'nin çözümünde GA'nın kullanılabilmesi için farklı kodlama tipleri geliştirilmiştir. UP için farklı kodlama tiplerinin etkisi araştırılabilir. Bu çalışmada, küçük ve orta boyutlu problemler için karma GA'nın performansı incelenmiştir. Daha büyük boyutlu problemlerde karma GA'nın performansının araştırılması bir diğer inceleme konusudur. Son olarak, bu çalışmada dikkate alınan problemler için birim taşıma maliyeti 3 ile 8 arasında düzgün dağılımdan elde edilmiştir. Dolayısıyla, farklı maliyet aralıklarının karma GA'nın çözüm kalitesi ve çözüm zamanı üzerindeki etkisi incelenebilir.

KAYNAKLAR

1. Hitchcock, F.L., "The Distribution of a Product from Several Sources to Numerous Locations", **Journal of Mathematical Physic**, Cilt 20, 224-230, 1941.
2. Winston, W.L., **Operations Research: Applications and Algorithm, Third Edition**, Duxbury Press, California, 1994.
3. Ahuja, R.K., Magnanti, T.L., Orlin,J.B., **Network flows**, Prentice Hall, Upper Saddle River, New Jersey, 1993.
4. Yan, S., Luo, S.C., "Probabilistic Local Search Algorithms for Concave Cost Transportation Network Problems", **European Journal of Operational Research,** Cilt 117, 511-521, 1999.
5. Larsson, T., Migdalas, A., Ronnqvist, M., "A Lagrangian Heuristic for the Capacitated Concave Minimum Cost Network Flow Problem", **European Journal of Operational Research**, Cilt 78, 116-129, 1994.
6. Erickson, R.E., Monma, C.L., Veinott, A.F., "Send-And-Split Method for Minimum Concave Cost Network Fows", **Teknik Rapor**, No 33, Department of Operations Research, Stanford University, Stanford, 1986.
7. Florian, M., Klein, M., "Deterministic Production Planning with Concave Cost and Capacity Constraints", **Management Science**, Cilt 8, 12-20, 1971.
8. Gallo, G., Sodini, C., "Adjacent Extreme Flows and Application to Min Concave Cost Flow Problems", **Networks**, Cilt 9, 95-121, 1979.
9. Gallo, G., Sandi, C., Sodini, C., "An Algorithm for the Min Concave Cost Flow Problem", **European Journal of Operational Research**, Cilt 4, 248-255, 1980.
10. Holland, J.H., **Adaptation in Natural and Artificial Systems**, Ann Arbor, University of Michigan Press, 1975.
11. Chen, C.L., Vempati, V.S., Aljaber, N., "An Application of Genetic Algorithms for flow shop problems", **European Journal of Operation Research**, Cilt 80, 389-396, 1995.
12. Reeves, C.R., "A Genetic Algorithms for Flowshop Sequencing", **Computers and Operations Research**, Cilt 22, Sayı 1, 5-13, 1995.
13. Murata, T., Ishibuchi, H., Tanaka, H., "Multi-Objective Genetic Algorithms and Its Applications to Flow Shop Scheduling", **Computers and Industrial Engineering**, Cilt 30, Sayı 4, 957-968, 1996.
14. Murata, T., Ishibuchi, H., Tanaka, H., "Genetic Algorithms for Flow Shop Scheduling Problems", **Computers and Industrial Enginering,** Cilt 30, Sayı 4, 1061-1071, 1996.
15. Chen, C.L., Neppalli, R.V., Aljaber, N**., "**Genetic Algorithms Applied to the Continuous Flow Shop Problem", **Computers and Industrial Engineering,** Cilt 30, Sayı 4, 919-929, 1996.
16. Dengiz, B., Altıparmak, F., Smith, A.E., "Local Search Genetic Algorithm for Optimal Design of Reliable Networks", **IEEE Transactions on Evolutionary Computation**, Cilt 1, Sayı 3, 179-188, 1997.
17. Dengiz, B., Altıparmak, F., Smith, A.E.,"Efficient Optimization of All Terminal Reliable Networks, Using an Evolutionary Approaches, **IEEE Transactions on Reliability,** Cilt 46, Sayı 1, 18-26, 1997.
18. Altıparmak, F., Dengiz, B., Smith, A.E., "Reliability Optimization of Computer Communica-Tion

Networks Using Genetic Algorithms", **IEEE International Conference on Systems, Man, and Cybernetics,** Cilt 5, 4676-4681, 1998.
19. Katayama, K., Sakamoto, H., Narihisa, H., "The Efficiency of Hybrid Mutation Genetic Algorithm for the Traveling Salesman Problem", **Mathematical and Computer Modelling**, Cilt 31, 197-203, 2000.
20. Borthfeldt, A., Gehring, H., "A Hybrid Genetic Algorithm for the Container Loading Problem", **European Journal of Operational Research,** Cilt 131, 143-161, 2001.
21. Berger, J., Barkaoui, M., "A Parallel Hybrid Genetic Algorithm for the Vehicle Routing Problem with Time Windows**", Computers & Operations Research**, Cilt 31, Sayı 12, 2037-2053, 2004.
22. Choi, I.C., Kim, S.I., Kim, H.S., "A Genetic Algorithm with A Mixed Region Search For the Asymmetric Traveling Salesman Problem**", Computers & Operations Research**, Cilt 30, Sayı 5, 773-786, 2003.
23. Michalewicz, Z., Vignaux, G. A., Hobbs, M., "A non-Standard Genetic Algorithm for the Nonlinear Transportation Problem", **ORSA Journal of Computing**, Cilt 3, Sayı 4, 307-316, 1991.
24. Vignaux, G. A., Michalewicz, Z., "A Genetic Algorithm for the Linear Transportation Problem", **IEEE Transactions on System, Man and Cybernetics**, Cilt 21, Sayı 2, 1991.
25. Gen, M., Li, Y., "Spanning Tree Based Genetic Algorithm for Bicriteria Fixed Charge Transportation Problem", **Proceedings of the Congress on Evolutionary Computation**, Washington DC, 2265-2271, 1999.
26. Gen, M., Ida, K., Li, Y, "Bicriteria Transportation Problem by Hybrid Genetic Algorithm", **Computers and Industrial Engineering**, Cilt 35, Sayı 1-2, 363-366, 1998.
27. Li, Y.Z., M. Gen, "Spanning Tree-Based Genetic Algorithm For Bicriteria Transportation Problem With Fuzzy Coefficients", **Australian Journal of Intelligent Information Processing Systems**, Cilt 4, Sayı 3, 220-229, 1998.
28. Gottlieb, J., Paulmann, L., "Genetic Algorithm for the Fixed Charge Transportation Problems", **Proceedings of the IEEE International Conference on Evolutionary Computations**, Anchorage, 330-335, 1998.
29. Syarif, A., Gen, M., "Solving Exclusionary Side Constrained Transportation Problem by Using Hybrid Spanning Tree-Based Genetic Algorithm", **Journal of Intelligent Manufacturing**, Cilt 14, 389-399, 2003.
30. Gottlieb, J., Eckert, C., "A Comparison of Two Representations for the Fixed Charge Transportation Problem", **Lecture Notes in Computer Science,** Cilt 1917, 345-354, 2000.
31. Eckert, C., Gottlieb, J., "Direct Representation and Variation Operators Fort He Fixed Charge Transportation Problem", **Lecture Notes in Computer Science,** Cilt 2439, 77-87, 2002.
32. Goldberg, D.E., **Genetic Algorithms: in Search, Optimization and Machine Learning**, Addison-Wesley, 1989.
33. Michalewicz, Z., **Genetic Algorithms + Data Structures = Evolution Programs,** Springer-Verlag, Berlin, Heidelberg, 1992.
34. Gen, M., Cheng, R., **Genetic Algorithms and Engineering Optimization**, John Wiley & Sons, New York, 2000.
35. Prüfer, H., "Neuer Beweis eines Satzes ueber Permutationen", **Archiv für Mathematik und Physik**, Cilt 27, 742-744, 1918.
36. Zhou, G., Gen, M., "A Note on Genetic Algorithms for the Degree-Constrained Spanning Tree Problem", **Networks,** Cilt 30, 91-95, 1997.
37. Zhou, G., Gen, M., "Approach to the Degree-Constrained Minimum Spanning Tree Problem Using Genetic Algorithm", **Engineering Design and Automation**, Cilt 3, 157-165, 1997.
38. Krishnamoorthy, M., Ernst, A.T., "Comparison of Algorithms for the Degree Constrained Minimum Spanning Tree", **Journal Of Heuristics**, Cilt 7, 587-611, 2001.
39. Lo, C.C., Chang, W.H., "A Multiobjective Hybrid Genetic Algorithm for the Capacitated Multipoint Network Design Problem", **IEEE Transactions on System, Man and Cybernetics Part B,** Cilt 30, Sayı 3, 461-470, 2000.