T.C.
KONYA TEKNİK ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

ÇOKLU GEZGİN SATICI PROBLEMİNİN SEZGİSEL ALGORİTMALAR İLE ÇÖZÜMÜ

Sevda DAYIOĞLU GÜLCÜ

YÜKSEK LİSANS TEZİ

Bilgisayar Mühendisliği Anabilim Dalı

Haziran-2019
KONYA

TEZ KABUL VE ONAYI

Sevda DAYIOĞLU GÜLCÜ tarafından hazırlanan "Çoklu Gezgin Satıcı Probleminin Sezgisel Algoritmalar ile Çözümü" adlı tez çalışması 11/06/2019 tarihinde aşağıdaki jüri tarafından oy birliği ile Konya Teknik Üniversitesi Lisansüstü Eğitim Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı'nda YÜKSEK LİSANS TEZİ olarak kabul edilmiştir.

Jüri Üyeleri	**İmza**
Başkan Doç. Dr. Halife KODAZ	.....
Danışman Doç. Dr. Humar KAHRAMANLI ÖRNEK	.....
Üye Dr. Öğr. Üyesi Ömer Kaan BAYKAN	.....

Yukarıdaki sonucu onaylarım.

Prof. Dr. Yakup KARA
Enstitü Müdürü

TEZ BİLDİRİMİ

Bu tezdeki bütün bilgilerin etik davranış ve akademik kurallar çerçevesinde elde edildiğini ve tez yazım kurallarına uygun olarak hazırlanan bu çalışmada bana ait olmayan her türlü ifade ve bilginin kaynağına eksiksiz atıf yapıldığını bildiririm.

DECLARATION PAGE

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Sevda DAYIOĞLU GÜLCÜ

Tarih: 11.06.2019

ÖZET

YÜKSEK LİSANS TEZİ

ÇOKLU GEZGİN SATICI PROBLEMİNİN SEZGİSEL ALGORİTMALAR İLE ÇÖZÜMÜ

Sevda DAYIOĞLU GÜLCÜ

Konya Teknik Üniversitesi
Lisansüstü Eğitim Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

Danışman: Doç. Dr. Humar KAHRAMANLI ÖRNEK

2019, 68 Sayfa

Jüri
Doç. Dr. Humar KAHRAMANLI ÖRNEK
Doç. Dr. Halife KODAZ
Dr. Öğr. Üyesi Ömer Kaan BAYKAN

Günümüzde, biyolojik yapıların oluşturduğu sistemler önem kazanmakta ve araştırmacıların ilgisini çekmektedir. Doğadaki bazı sosyal sistemler, sınırlı yetenekli basit bireyler tarafından oluşturulmalarına rağmen kolektif zekâ davranışları sergilemektedirler. Bu bireylerin kendi içerisindeki organizasyonları ve dolaylı iletişimleri, mühendislikte ve günlük yaşantıda karşılaşılan optimizasyon problemlerinin çözümünde kullanılmaktadır. Ayrıca, yapay zekâ sistemlerinin gelişmesine de katkı sunmaktadırlar. Parçacık Sürü Optimizasyonu (PSO) algoritması kuşların sosyal davranışlarına dayalı bir metasezgisel algoritmadır. Bu tez çalışmasında, Çoklu Gezgin Satıcı Problemini (ÇGSP) çözmek için PSO tabanlı 2 algoritma, APSO ve HAPSO, geliştirilmiştir. ÇGSP'de amaç, başlangıç ve bitiş noktası depo olan *m* adet satıcı için her bir ara şehir bir kere ziyaret edilmek şartıyla gezilen bütün şehirlerin toplam tur maliyetini minimize etmektir. APSO algoritması ÇGSP'yi çözmek için PSO, 2-opt algoritmalarını ve path-relink, takas operatörlerini kullanmaktadır. Diğer taraftan, HAPSO algoritması ÇGSP'yi çözmek için Açgözlü Rasgeleleştirilmiş Adaptif Arama Prosedürü (GRASP) algoritması, PSO, 2-opt algoritmalarını ve path-relink, takas operatörlerini kullanmaktadır. Deneylerde, 9 Gezgin Satıcı Problemi(GSP) örneği kullanılmıştır ve bu örnekler üzerinde HAPSO algoritması ve APSO algoritması tur grafikleri, çalışma süreleri ve kutu grafikleri açısından 2, 3, 4, 5, 6, 7, 8 ve 9 satıcı için detaylı olarak karşılaştırılmıştır. Tur grafikleri incelendiğinde HAPSO algoritması ile oluşturulan turların uzunluklarının daha kısa olduğu görülmüştür. Algoritmaları kıyaslamak için kullanılan bir diğer yöntem ise kutu grafiğidir. Kutu grafiği sayesinde algoritmaların sonuçları ile ilgili istatistiksel bilgiler görselleştirilmektedir ve böylece bu bilgilerin yorumlanması kolaylaşmaktadır. Kutu grafikleri incelendiğinde, HAPSO algoritmasının daha istikrarlı ve daha gürbüz olduğu görülmektedir. Ayrıca, 5 GSP örneği üzerinde APSO ve HAPSO algoritmaları literatürdeki Genetik Algoritma ve Karınca Koloni Optimizasyonu algoritmaları ile karşılaştırılmıştır. Sonuçlara göre, HAPSO algoritması birçok örnekte diğer algoritmalardan daha iyi performans sergilemiştir. Ayrıca, HAPSO algoritması APSO algoritmasından daha kararlı sonuçlar üretmektedir ve HAPSO algoritmasının performansı tüm ÇGSP örneklerinde daha iyidir. Bu nedenle, HAPSO algoritması APSO algoritmasından daha gürbüzdür.

Anahtar Kelimeler: 2-opt algoritması, Çoklu gezgin satıcı problemi, Grasp algoritması, Parçacık sürü optimizasyonu.

ABSTRACT

MS THESIS

SOLVING THE MULTIPLE TRAVELING SALESMAN PROBLEM USING HEURISTIC ALGORITHMS

Sevda DAYIOĞLU GÜLCÜ

Konya Technical University
Institute of Graduate Studies
Department of Computer Engineering

Advisor: Assoc. Prof. Dr. Humar KAHRAMANLI ÖRNEK

2019, 68 Pages

Jury
Assoc. Prof. Dr. Humar KAHRAMANLI ÖRNEK
Assoc. Prof. Dr. Halife KODAZ
Asst. Prof. Dr. Ömer Kaan BAYKAN

Nowadays, systems formed by biological structures have gained importance and attracted the attention of researchers. Some social systems in nature exhibit collective intelligence, although they are created by simple individuals with limited abilities. The organizations of these individuals and their indirect communication among these individuals are used to solve the optimization problems encountered in engineering and daily life. They also contribute to the development of the artificial intelligence systems. The Particle Swarm Optimization (PSO) algorithm which is a meta-heuristic algorithm based on the social behavior of birds. In this article, 2 algorithms based on PSO, called APSO and HAPSO, were proposed to solve the Multiple Travelling Salesman Problem (MTSP). The aim in the MTSP is to find the tours for all *m* salesmen, who all start and end at the depot, such that each intermediate node is visited exactly once and the total cost of visiting all nodes is minimized. The APSO algorithm is based on the PSO and 2-opt algorithms, the path-relink and swap operators. On the other hand, the HAPSO algorithm is based on the GRASP, PSO and 2-opt algorithms, the path-relink and swap operators. In the experiments, the 9TSP instances were used and the HAPSO and APSO algorithms were compared in detail on these samples for the 2, 3, 4, 5, 6, 7, 8 and 9 salesmen in terms of the tour charts, the computational times and the boxplots. When the tour graphs are examined, the lengths of the tours created by HAPSO algorithm are shorter. Another method used to compare algorithms is the boxplot. Through the box graph, the statistical information about the results of the algorithms are visualized and thus the interpretation of this information is easy. When the boxplots are examined, it is seen that HAPSO algorithm is more stable and more robust. In addition, on the 5 TSP instances the HAPSO and APSO algorithms were compared with the Genetic Algorithm and Ant Colony Optimization algorithms in the literature. According to the results, the HAPSO algorithm has the better performance than the other algorithms on the most instances. In addition, the HAPSO algorithm produces more stable results than the APSO algorithm and the performance of the HAPSO algorithm is better in all the MTSP instances. Therefore, the HAPSO algorithm is more robust than the APSO algorithm.

Keywords: 2-opt algorithm, Grasp algorithm, Multiple travelling salesman problem, Particle swarm optimization.

ÖNSÖZ

Bu çalışmada, parçacık sürü optimizasyon algoritması, 2-opt sezgisel algoritması, Açgözlü Rasgeleleştirilmiş Adaptif Arama Prosedürü (GRASP) algoritması ve gezgin satıcı problemi ile ilgili teknik bilgiler verilmiştir. Parçacık sürü algoritması, 2-opt sezgisel algoritması ve GRASP algoritması ile çoklu gezgin satıcı problemi çözümlenmiştir.

Çalışmalarım sırasında; yönlendirme ve desteklerinden dolayı çok değerli danışmanım Sayın Doç. Dr. Humar KAHRAMANLI ÖRNEK'e, tez çalışmasının son halini alması aşamasında kıymetli fikirleri ile katkıda bulunan jüri üyeleri Sayın Doç. Dr. Halife KODAZ'a ve Sayın Dr. Öğr. Üyesi Ömer Kaan BAYKAN'a, bugüne kadar tecrübelerinden faydalandığım bana emeği geçmiş bütün hocalarıma, maddi ve manevi desteğini esirgemeyen çok sevdiğim eşime ve aileme en içten teşekkürlerimi sunarım.

Sevda DAYIOĞLU GÜLCÜ
KONYA-2019

İÇİNDEKİLER

ÖZET ..... **iv**

ABSTRACT ..... **v**

ÖNSÖZ ..... **vi**

İÇİNDEKİLER ..... **vii**

SİMGELER VE KISALTMALAR ..... **viii**

ŞEKİLLER LİSTESİ ..... **ix**

ÇİZELGELER LİSTESİ ..... **x**

1. GİRİŞ **1**

1.1. Tezin Amacı 1
1.2. Tezin Önemi 2

2. KAYNAK ARAŞTIRMASI **4**

3. MATERYAL VE YÖNTEM **8**

3.1. Gezgin Satıcı Problemi 8
3.2. Çoklu Gezgin Satıcı Problemi 9
3.3. Parçacık Sürü Optimizasyonu Algoritması (PSO) 10
3.4. Açgözlü Rasgeleleştirilmiş Adaptif Arama Prosedürü (GRASP) Algoritması 12
3.5. 2-opt Algoritması 15
3.6. Geliştirilen Yöntemler 15
3.6.1. ÇGSP'yi çözmek için geliştirilen PSO tabanlı meta sezgisel algoritma (APSO) 15
3.6.2. ÇGSP'yi çözmek için geliştirilen PSO ve GRASP tabanlı hibrit bir meta sezgisel algoritma (HAPSO) 20

4. ARAŞTIRMA SONUÇLARI VE TARTIŞMA **24**

5. SONUÇLAR VE ÖNERİLER **38**

5.1 Sonuçlar 38
5.2 Öneriler 40

KAYNAKLAR **42**

EKLER **47**

ÖZGEÇMİŞ **68**

SİMGELER VE KISALTMALAR

2-opt	:	2-opt sezgisel algoritması
APSO	:	Ayrıklaştırılmış Parçacık Sürü Optimizasyonu
ASD	:	Akıllı su damlaları algoritması
ÇGSP	:	Çoklu Gezgin Satıcı Problemi
GA	:	Genetik algoritma
gBest	:	Sürünün en iyi parçacığı
GRASP	:	Açgözlü Rasgeleleştirilmiş Adaptif Arama Prosedürü algoritması
GSP	:	Gezgin Satıcı Problemi
HAPSO	:	PSO ve GRASP algoritmaları tabanlı hibrit meta sezgisel algoritma
KKO	:	Karınca koloni optimizasyonu
pBest	:	Parçacığın geçmişteki en iyi konumu
PSO	:	Parçacık Sürü Optimizasyonu
KAL	:	Kısıtlı aday listesi
TSPLIB	:	GSP kütüphanesi
YAK	:	Yapay arı koloni algoritması

ŞEKİLLER LİSTESİ

Şekil 3.1. GSP örneği (berlin52) 8
Şekil 3.2. Tekli depo ÇGSP örneği (4 satıcı) 9
Şekil 3.3. PSO algoritmasının akış diyagramı 11
Şekil 3.4. 2-opt algoritmasının örnek uygulaması (Johnson ve McGeoch, 1997) 15
Şekil 3.5. APSO algoritmasının akış diyagramı (pb: popülasyon boyutu, i: parçacık indisi) 16
Şekil 3.6. HAPSO algoritmasının akış diyagramı (pb: popülasyon boyutu, i: parçacık indisi) 21
Şekil 4.1. Düzgün sekizgen 24
Şekil 4.2. 2 satıcı ve depo *i* şehri için sekizgenin en iyi çözümü 25
Şekil 4.3. 2 satıcı ve depo *a* şehri için sekizgenin en iyi çözümleri 26
Şekil 4.4. att48, berlin52, bier127, eil51 ve eil76 problemleri üzerinde 2 satıcı için HAPSO ve APSO ile bulunan en iyi turlar 33
Şekil 4.5. pr152, pr226, pr76 ve rat99 problemleri üzerinde 2 satıcı için HAPSO ve APSO ile bulunan en iyi turlar 34
Şekil 4.6. 2 satıcı için ÇGSP örneklerinin kutu grafiği 37
Şekil EK-1.1. 3 satıcı için HAPSO ve APSO ile bulunan en iyi turlar 47
Şekil EK-1.2. 3 satıcı için HAPSO ve APSO ile bulunan en iyi turlar (Devamı) 48
Şekil EK-1.3. 4 satıcı için HAPSO ve APSO ile bulunan en iyi turlar 49
Şekil EK-1.4. 4 satıcı için HAPSO ve APSO ile bulunan en iyi turlar (Devamı) 50
Şekil EK-1.5. 5 satıcı için HAPSO ve APSO ile bulunan en iyi turlar 51
Şekil EK-1.6. 5 satıcı için HAPSO ve APSO ile bulunan en iyi turlar (Devamı) 52
Şekil EK-1.7. 6 satıcı için HAPSO ve APSO ile bulunan en iyi turlar 53
Şekil EK-1.8. 6 satıcı için HAPSO ve APSO ile bulunan en iyi turlar (Devamı) 54
Şekil EK-1.9. 7 satıcı için HAPSO ve APSO ile bulunan en iyi turlar 55
Şekil EK-1.10. 7 satıcı için HAPSO ve APSO ile bulunan en iyi turlar (Devamı) 56
Şekil EK-1.11. 8 satıcı için HAPSO ve APSO ile bulunan en iyi turlar 57
Şekil EK-1.12. 8 satıcı için HAPSO ve APSO ile bulunan en iyi turlar (Devamı) 58
Şekil EK-1.13. 9 satıcı için HAPSO ve APSO ile bulunan en iyi turlar 59
Şekil EK-1.14. 9 satıcı için HAPSO ve APSO ile bulunan en iyi turlar (Devamı) 60
Şekil EK-2.1. 3 satıcı için ÇGSP örneklerinin kutu grafiği 61
Şekil EK-2.2. 4 satıcı için ÇGSP örneklerinin kutu grafiği 62
Şekil EK-2.3. 5 satıcı için ÇGSP örneklerinin kutu grafiği 63
Şekil EK-2.4. 6 satıcı için ÇGSP örneklerinin kutu grafiği 64
Şekil EK-2.5. 7 satıcı için ÇGSP örneklerinin kutu grafiği 65
Şekil EK-2.6. 8 satıcı için ÇGSP örneklerinin kutu grafiği 66
Şekil EK-2.7. 9 satıcı için ÇGSP örneklerinin kutu grafiği 67

ÇİZELGELER LİSTESİ

Çizelge 3.1. Deneylerde kullanılan GSP veri seti 10
Çizelge 3.2. PSO algoritmasının sözde kodu 12
Çizelge 3.3. GRASP algoritmasının sözde kodu 13
Çizelge 3.4. Yapım aşaması safhasının sözde kodu 14
Çizelge 3.5. Lokal arama safhasının sözde kodu 14
Çizelge 3.6. Örnek bir parçacık alttur bilgisi (5 gezgin satıcı) 17
Çizelge 3.7. Geliştirilen APSO algoritmasının sözde kodu 19
Çizelge 3.8. Path-relink operatörü 20
Çizelge 3.9. Takas operatörü 20
Çizelge 3.10. Geliştirilen HAPSO algoritmanın sözde kodu 23
Çizelge 4.1. Düzgün sekizgen için uzaklık matrisi 25
Çizelge 4.2. 2 satıcı için takas, path-relink operatörlerinin ve 2-opt algoritmasının etkisini 26
Çizelge 4.3. 3 satıcı için takas, path-relink operatörlerinin ve 2-opt algoritmasının etkisini 27
Çizelge 4.4. 4 satıcı için takas, path-relink operatörlerinin ve 2-opt algoritmasının etkisini 27
Çizelge 4.5. Süre açısından (sn.) takas, path-relink operatörlerinin ve 2-opt algoritmasının karşılaştırılması 27
Çizelge 4.6. 2 satıcı için APSO, HAPSO, GA ve KKO algoritmaların karşılaştırılması 28
Çizelge 4.7. 3 satıcı için APSO, HAPSO, GA ve KKO algoritmaların karşılaştırılması 29
Çizelge 4.8. 4 satıcı için APSO, HAPSO, GA ve KKO algoritmaların karşılaştırılması 29
Çizelge 4.9. APSO ve HAPSO algoritmaların karşılaştırılması (2 satıcı için) 30
Çizelge 4.10. APSO ve HAPSO algoritmaların karşılaştırılması (3 satıcı için) 30
Çizelge 4.11. APSO ve HAPSO algoritmaların karşılaştırılması (4 satıcı için) 30
Çizelge 4.12. APSO ve HAPSO algoritmaların karşılaştırılması (5 satıcı için) 31
Çizelge 4.13. APSO ve HAPSO algoritmaların karşılaştırılması (6 satıcı için) 31
Çizelge 4.14. APSO ve HAPSO algoritmaların karşılaştırılması (7 satıcı için) 31
Çizelge 4.15. APSO ve HAPSO algoritmaların karşılaştırılması (8 satıcı için) 32
Çizelge 4.16. APSO ve HAPSO algoritmaların karşılaştırılması (9 satıcı için) 32
Çizelge 4.17. Süre açısından (sn.) HAPSO ve APSO algoritmaların karşılaştırılması .. 35

1. GİRİŞ

Günümüzde, biyolojik yapılardan esinlenerek oluşturulan sistemler önem kazanmakta ve araştırmacıların ilgisini çekmektedir. Doğadaki bazı sosyal sistemler, sınırlı yetenekli basit bireyler tarafından oluşturulmalarına rağmen kolektif zekâ davranışları sergilemektedirler. Problemlere uyarlanan zeki çözümler, bu bireylerin kendi içerisindeki organizasyonları ve dolaylı iletişimlerinden ortaya çıkar. Bu sistemler, dağıtık yapay zekâ sistemlerinin geliştirilmesinde önemli tekniklerin kaynağı olmaktadır (Nabiyev, 2012).

Gezgin Satıcı Problemi (GSP) en çok çalışılan optimizasyon problemlerinden birisidir. GSP, aralarındaki uzaklıkları bilinen noktalardan bir kez geçilmek şartı ile tüm noktaların en az maliyetle dolaşılıp, başlangıç noktasına tekrar dönülmesini anlatmaktadır (Dorigo ve Stützle, 2004). GSP'nin kolay formüle edilmesi, zor çözülmesi ve çok sayıda uygulama alanı olması bu problem üzerinde pek çok çalışma yapılmasına neden olmaktadır. Literatürde gezgin satıcı probleminin çok sayıda çeşidi ve genellemeleri vardır (Gutin ve Punnen, 2006).

Bu çalışmadaki bölümlendirme şu şekilde düzenlenmiştir: Giriş bölümünde tezin amacı ve önemi hakkında bilgiler verilmiştir. İkinci bölümde GSP'nin çözüm yöntemleri ve ÇGSP'nin çözüm yöntemleri ile ilgili önceki çalışmalar verilmiştir. Üçüncü bölümde öncelikle gezgin satıcı ve çoklu gezgin satıcı probleminin tanımı verilmiştir. Daha sonra tezde kullanılan algoritmalar anlatılmış ve son olarak ÇGSP'nin çözümü için Parçacık Sürü Optimizasyonu (PSO), 2-opt, GRASP tabanlı 2 algoritma önerilmiştir. Dördüncü bölümde önerilen algoritmaların deneysel sonuçları verilmiştir ve önerilen algoritmaların başarıları karşılaştırılmıştır. Son olarak Beşinci bölümde tez sonuçlandırılmıştır ve gelecek çalışmalar hakkında bilgiler verilmiştir.

1.1. Tezin Amacı

GSP, üzerinde en çok çalışılan optimizasyon problemlerinden biridir. Optimizasyon; en iyileme anlamına gelmektedir ve matematikteki tanımı, bir fonksiyonu minimize ya da maksimize etmek amacı ile gerçek ya da tamsayı değerlerini tanımlı bir aralıkta seçip fonksiyona yerleştirerek sistematik olarak bir problemi incelemek ya da çözmek işlemlerini ifade eder. Optimizasyon problemleri genellikle NP-zor, karmaşık ve zaman alıcıdır.

GSP, bir gezgin satıcının bulunduğu şehirden başlayıp her şehre sadece bir kez uğradıktan sonra başladığı şehre geri dönen en kısa turu bulmayı amaçlamaktadır. Literatürde gezgin satıcı probleminin çok sayıda çeşidi ve genellemeleri vardır. Bunlardan biri Çoklu Gezgin Satıcı Problemidir (ÇGSP).

ÇGSP; *n* adet şehri, *m* adet tura bölmektedir ve bu *m* adet tur *m* adet satıcıya atanmaktadır. ÇGSP, GSP'den daha kompleks bir problemdir. Çözümü için öncelikle her bir satıcıya hangi şehirlerin atanacağının ve satıcıların turları üzerindeki şehirlerin optimal sıralamasının belirlenmesi gerekmektedir. ÇGSP'nin en yaygın uygulaması planlama alanındadır. Bir üretim hattı üzerindeki işlerin planlanması işlemi genellikle GSP olarak modellenir. Üretimin, işlere tahsis edilecek birden çok paralel hat üzerine genişlediği durumlarda ise problem ÇGSP olarak modellenmektedir.

Optimizasyon problemlerinin çözümü için iki yöntem kullanılır: kesin yöntemler ve meta sezgisel yöntemler. Kesin yöntemler en iyi çözümü bulurlar. Fakat çok büyük problemlerinin kesin yöntemler ile çözümü çok uzun zaman aldığı için bu yöntemler pek kullanılamazlar. Diğer taraftan meta sezgisel teknikler, problemin en iyi çözümü veya en iyi çözüme yakın uygun bir çözümü kabul edilebilir bir sürede bulmaktadırlar (Talbi, 2009).

Optimizasyon problemlerinin çözümü için çeşitli meta sezgisel teknikler geliştirilmiştir ve yeni teknikler de önerilmektedir. Bunlardan birkaçı şunlardır: PSO algoritması (Kennedy ve Eberhart, 1995), genetik algoritma (GA) (Holland, 1992), yapay arı koloni (YAK) algoritması (Karaboga ve Basturk, 2007), akıllı su damlaları (ASD) algoritması (Shah-Hosseini, 2009) ve karınca koloni optimizasyonu (KKO) algoritmasıdır (Dorigo ve Gambardella, 1997).

Bu tezde, PSO, GRASP ve 2-opt sezgisel algoritmaları ile ÇGSP'nin çözümü amaçlanmaktadır.

1.2. Tezin Önemi

Günümüzde, biyolojik yapılardan esinlenerek oluşturulan sistemler önem kazanmakta ve araştırmacıların ilgisini çekmektedir. Doğadaki bazı sosyal sistemler, sınırlı yetenekli basit bireyler tarafından oluşturulmalarına rağmen kolektif zekâ davranışları sergilemektedirler. Problemlere uyarlanan zeki çözümler, bu bireylerin kendi içerisindeki organizasyonları ve dolaylı iletişimlerinden ortaya çıkar. Bu sistemler,

dağıtık yapay zekâ sistemlerinin geliştirilmesinde önemli tekniklerin kaynağı olmaktadır (Nabiyev, 2012). Bu sistemlere örnek olarak GA, PSO ve YAK algoritmaları verilebilir.

PSO algoritması, bir hedefe ulaşmak için çabalayan kuşların sosyal davranışlarına dayalı bir metasezgisel algoritma olarak Kennedy ve Eberhart (1995) tarafından geliştirilmiştir. PSO algoritması, sürü zekâsının en yaygın örneklerinden biridir. PSO algoritması kuş sürülerinin davranışlarını simüle etmektedir. Gerçek hayattaki kuş, PSO algoritmasında parçacık olarak adlandırılmaktadır ve her bir parçacık problem uzayında bir çözümü temsil etmektedir.

ÇGSP çok sayıda günlük hayat problemini modellemek için kullanılabilecek karmaşık bir problemdir ve GSP'nin bir türevidir. GSP'de *n* adet şehir için alt-tur olmaksızın, her şehir yalnızca bir defa ziyaret edilmekte ve en iyiye yakın turun bulunması hedeflenmektedir.

Günlük hayattaki bazı problemler bir ÇGSP olarak karşımıza çıkmaktadır. Gazete, medya ve matbaa alanında reklamları yerleştirme (Carter ve Ragsdale, 2002) ve baskı çizelgeleme problemi (Gorenstein, 1970) bir ÇGSP olarak görülmüş ve çözülmüştür. Okul servisi rotalama problemi (Baykasoğlu ve Özbel, 2016), doğal afet yönetiminde kurtarma birimlerinin planlanması (Wex ve ark., 2014), fotoğraf stüdyosundaki personellerin çizelgelemesi (Zhang ve ark., 1999) ve evde bakım hizmeti alanında sunulan hizmetin çizelgelemesi (Mankowska ve ark., 2014) bir ÇGSP olarak görülmüş ve çözülmüştür. Görüldüğü gibi, farklı birçok alanda ÇGSP karşımıza çıkmaktadır ve bundan dolayı ÇGSP'yi çözmek çok önemlidir.

Bu tez çalışmasında PSO, GRASP ve 2-opt sezgisel algoritmaları ile ÇGSP'nin çözümü hedeflenmektedir.

Literatürde ÇGSP'yi çözen algoritmalar bulunmaktadır. Fakat tez çalışmasında geliştirilen algoritmalar literatürdeki çalışmalara göre daha iyi sonuçlar vermektedir. Ayrıca, geliştirilen algoritmalarda gezgin satıcılar dengeli tur oluşturmaktadır. Buna ek olarak GSP çoklu gezgin satıcı ile çözüldüğünde toplam süre kısalmaktadır.

2. KAYNAK ARAŞTIRMASI

Literatürde, GSP üzerine detaylı yapılmış çok sayıda çalışma bulunmaktadır. Fakat ÇGSP konusunda detaylı çalışma az sayıda bulunmaktadır. Bu bölümde, ÇGSP konusunda yapılmış kaynak araştırmaları verilmektedir.

Carter (2003) tarafından yapılan doktora tezi çalışmasında, ÇGSP'nin çözümü için GA kullanılarak yeni bir yöntem önerilmektedir. ÇGSP'nin GA ile çözümünde, literatürde 2 farklı kromozom gösterimi bulunmaktadır: İki kromozom tekniği ve tek kromozom tekniği. İki kromozom tekniğinde 2 tane kromozom kullanılmaktadır. Kromozomlardan bir tanesinde şehirler bulunmaktadır. Diğerinde ise bu şehirlerin hangi satıcı tarafından ziyaret edildiği bilgisi tutulmaktadır. Tek kromozom tekniğinde ise 1 tane kromozom bulunmaktadır. Her bir satıcının gezdiği şehirlerin listesi verilmektedir. Bir diğer satıcının listesine geçildiğini belirtmek için, listelerin arasına farklı bir gen (örneğin -1) konulmaktadır. Carter'in çalışmasında tek kromozom tekniği kullanılmıştır. Fakat Carter yeni bir kromozom gösterimi önermiştir. GA'da kullanılan kromozom iki parçadan oluşmaktadır. Kromozomun ilk parçasında satıcıların gezdiği şehirler bulunmaktadır. İkinci parçanın uzunluğu satıcı sayısına eşittir ve her bir gende satıcıların gezdiği şehir sayıları bulunmaktadır. Böylelikle, hangi satıcının hangi şehirleri ziyaret ettiği belli olmaktadır. Önerilen iki-parçalı kromozom yöntemi ile mevcut iki kromozom tekniği ve tek kromozom tekniği karşılaştırılmıştır. Önerilen yöntemin daha avantajlı olduğu deneysel çalışmlarla gösterilmiştir. İki-parçalı kromozom yönteminin daha küçük çözüm uzayı sayesinde daha iyi sonuçlar üretmeye izin verdiği belirtilmektedir. Ayrıca önerilen yöntem; üretim planlama problemi, kalite kontrol planlaması problemi gibi endüstriyel problemlerin çözümünde de kullanılmıştır ve başarılı sonuçlar elde ettiği gözlemlenmiştir.

Junjie ve Dingwei (2006) yayınladıkları çalışma, ÇGSP'nin karınca kolonisi optimizasyonu ile çözülmesine ilişkin modern bir örnektir. Yazarlar karınca kolonisi optimizasyonunu problemin çözümüne uygularken yetenek kısıtlamasından faydalanmışlardır. Önerilen yöntemi, TSPLIB'deki birkaç test problemi üzerinde denemiş ve sonuçları GA ile karşılaştırmışlardır. Deneysel sonuçlar önerilen algoritmanın oldukça başarılı olduğunu göstermektedir.

Dang ve ark. (2007) yayınladıkları çalışmada, ÇGSP'yi çözmek için Darwin evrimini simüle eden bir GA önermektedirler. Bu çalışmada, ÇGSP'yi standart GSP'ye dönüştürerek çözen yeni bir yöntem sunmaktadırlar.

Bektaş (2006), ÇGSP'nin tanımlanması, türevleri, kullanım alanları ve çözüm yöntemleri hakkında derleme niteliğinde bir çalışma yapmıştır. Çalışmada, ÇGSP tek ve çok depo türleri açısından ele alınmaktadır. Ayrıca, ÇGSP'nin gerçek yaşamdaki uygulamalarına örnekler verilmektedir. Problemlerin çözümü için geliştirilen kesin algoritmalar ve sezgisel yöntemler hakkında bilgi verilmektedir. Bektaş (2012) bir diğer çalışmasında çoklu depo çoklu gezgin satıcı problemini çözen yeni modelleri ve kesin algoritmaları incelemiştir.

Mitrović-Minić ve Krishnamurti (2006) yayınladıkları çalışmada ÇGSP'yi zaman pencereleri kavramı ile birlikte incelemektedirler ve araçlar arasında üstünlük çizgelerini temel alan bağlantılardan bahsetmektedir. Önerilen yöntemi, deneylerde zaman pencereli araç rotalama problemini zaman ÇGSP'ne çevirerek çözmektedirler.

Malik ve ark. (2007) yöneylem araştırma çerçevesinden bakarak genelleştirilmiş çok istasyonlu Çoklu GSP için, minimum yayılan ağaç (minimum spanning tree) veri yapısını kullanan yeni bir algoritma önermektedirler.

Oberlin ve ark. (2009) çoklu depolu ve çoklu gezgin satıcı problemini tekli ve asimetrik gezgin satıcı problemine dönüştürmektedirler. Dönüştürme işleminin başarısını test etmek için sezgisel Lin-Kernighan sezgisel (LKH) yöntemini test problemlerine uygulamışlardır. Sonuçlar, dönüştürme işleminin başarılı olduğunu ve yüksek kaliteli sonuçların elde edildiğini göstermektedir.

Ponraj ve Amalanathan (2014) çalışmasında yol kapasitesine dayalı ÇGSP'yi çözmek için yeni bir yöntem önermektedirler. Bu çalışmada, yol kapasite kısıtına bağlı olarak satıcıların tur zamanlarını minimize etmeyi amaçlamışlardır. Ayrıca algoritmanın performansını artırmak için önerilen yöntemi 85 işlemcide paralel olarak çalıştırmaktadırlar. Böylece algoritmanın hesaplama zaman karmaşıklığını minimize etmektedirler. Elde edilen sonuçları mevcut yöntemin sonuçlarıyla ve algoritmanın seri çalıştırılmasından elde edilen sonuçlarla karşılaştırmaktadırlar ve daha iyi sonuç elde ettiklerini göstermektedirler.

Son zamanlarda, optimizasyon problemlerini çözmek için sezgisel yöntemlerin hibrit olarak kullanılması sıkça karşılaşılan bir durumdur. Yu ve ark. (2012) ÇGSP'ni çözmek için GA, k-ortalamalar algoritması, Lin-kernighan ve dallandır ve kes algoritmasını hibrit olarak kullanmaktadır. GA ile satıcılar arasındaki şehirler takas edilmektedir ve çıkan sonuçlar k-ortalamalar algoritması ile kümelenmektedir. Alt seviyede ise Lin-kernighan ve dallandır ve kes algoritmaları her bir satıcıya ait alt-problemleri çözmek için uygulanmaktadır. Böylelikle bu yöntem, GA sayesinde global

optimizasyon kabiliyetine ve dallandır ve kes algoritması ile lokal optimizasyon kabiliyetine sahip olmaktadır. Deneylerde veri seti olarak TSPLIB kullanılmaktadır. Elde edilen sonuçlar, başka bir hibrit yöntemle karşılaştırılmaktadır ve önerilen yöntemin daha başarılı olduğu gözlemlenmektedir.

Yuan ve ark. (2013) GA kullanarak ÇGSP'nin çözümü için yeni bir çaprazlama operatörü olan 2-parçalı kromozom çaprazlamayı (TCX) önermişlerdir. TCX yöntemini farklı üç çaprazlama yöntemiyle karşılaştırmışlardır. Yuan ve arkadaşları yaptıkları testlerde deney veri seti olarak TSPLIB kütüphanesindeki test problemlerini kullanmaktadırlar. Deney sonuçlarında ise, önerdikleri algoritma ile birçok problemde diğer üç çaprazlama operatörünün elde ettiği sonuçlardan daha iyi sonuçlar elde etmektedirler.

Sofge ve ark. (2002) ÇGSP'nin çözümü için "Neighborhood Attractor Schema" adında yeni bir lokal arama algoritması önermektedirler. Veri seti olarak TSPLIB kullanılmaktadır ve önerilen yöntem GA, PSO ve Monte-Carlo algoritmaları ile karşılaştırılmaktadır. Elde edilen sonuçların başarılı olduğu ve yeni çalışmalar yapmak için cesaret verici olduğu söylenmektedir.

Hou ve Liu (2012) ÇGSP'nin genel özelliklerini analiz etmektedir ve ÇGSP'yi basit bir grafiğe çevirerek çözmektedir. Ayrıca 2-opt lokal arama algoritması kullanarak elde edilen turları iyileştirmektedir.

Király ve Abonyi (2015) ÇGSP'yi çözmek için GA tabanlı çoklu-kromozom tekniği sunmaktadır. Geliştirilen yöntem Google Maps ile entegre edilerek gerçek bir lojistik probleminde (Macaristan'ın en büyük enerji sağlayıcılarından birinde mobil mekanik tedarikinde) uygulanmaktadır. Geliştirilen yöntemde satıcı sayısı alt ve üst sınır ile sınırlandırılmaktadır. Ayrıca mutasyon operatörü olarak kaydırma, lokal arama, çaprazlama, takas ve ters çevirme işlemlerinden biri kullanılmaktadır. Algoritmanın tasarımına bağlı olarak bu operatörler ardışık olarak da kullanılabilmektedir.

Necula ve ark. (2015a) çalışmalarında ÇGSP'yi çözen KKO tabanlı beş algoritma önermektedirler ve bu algoritmaların performans karşılaştırmalarını yapmaktadırlar.

Necula ve ark. (2015b) ÇGSP'yi çoklu optimizasyon problemi olarak ele almaktadır ve KKO algoritması ile problemi çözmektedirler. Hem en uzun tur maliyetini düşürmeyi hem de dengeli alt turlar oluşturmayı hedeflemektedirler. Bunun için pareto yöntemini uygulamaktadırlar.

Necula ve ark. (2018) tekli-depo ÇGSP'yi çözmek için KKO algoritmasını uygulamışlardır. Alt turları dengeli hale getirmeyi amaçlamaktadırlar. Bunun için de en

uzun alt tur uzunluğunu minimize etmektedirler. Böylece satıcılar arasındaki iş yükü dengeli hale gelmektedir. Öncelikle K-ortalamalar kümeleme algoritması ile başlangıç problemi bir kaç şehir grubuna bölünmektedir. Daha sonra her bir alt grup KKO algoritması ile çözülmektedir.

Zhou ve ark. (2018) ÇGSP'yi çözmek için rulet seçimi ve elit seçimi yöntemi ile GA'yı iyileştirmişler ve iki tane GA tabanlı yöntem önermişlerdir. Ayrıca dört yeni mutasyon operatörü önermektedirler. İlk algoritmada çaprazlama operatörü kullanılmamaktadır. Bundan dolayı yüksek bir hesaplama verimliliği elde edilmektedir. Fakat bunun yanında, orijinal popülasyonun çeşitliliğine de ihtiyaç duymaz. Çaprazlama operatörü kullanılmadığından dolayı birkaç tane mutasyon operatörü kullanılmaktadır. İkinci algoritma ise birinci algoritmanın iyileştirilmiş halidir. Birinci algoritmada rulet seçimi ve bazı mutasyon operatörleri lokal optimumlara takılmaya engel olamadığı için seçim yöntemi ve mutasyon operatörleri iyileştirilerek daha iyi sonuçlar elde edilmektedir.

Yousefikhoshbakht ve ark. (2013) tekli-depo ÇGSP'yi çözmek için ekleme, takas ve 2-opt algoritmalarıyla birleştirilmiş KKO algoritmasının modifiye edilmiş yeni bir versiyonunu önermektedirler. Geliştirdikleri yöntemin lokal optimum noktalarından kaçmada başarılı olduğunu belirtmektedirler. Geleneksel KKO algoritmalarından farklı olarak, önerilen algoritma mevcut en iyi çözüm ve şimdiye kadar bulunan en iyi çözüm için sadece global güncelleme kullanmaktadır. Ayrıca, önerilen algoritmanın etkinliğini artırmak için yeni bir durum geçiş kuralı ve etkin bir aday listesi kullanılır. Böylece algoritma global aramada daha başarılı olmaktadır.

Shabanpour ve ark. (2017) ÇGSP'yi çözmek için GA ve kümeleme tekniğini birleştirerek hibrit bir algoritma önermektedirler. Algoritma iki aşamadan oluşmaktadır. İlk aşamada kümeleme algoritması ile şehirler gruplanmakta ve satıcılara atanmaktadır. Böylelikle, kümeleme yöntemiyle ÇGSP GSP'ye dönüştürülmekte ve problem arama uzayı daraltılmaktadır. İkinci aşamada ise GA ile satıcıların turları oluşturulmakta ve bu turlar iyileştirilmeye çalışılmaktadır. Ayrıca, kromozom yapısı olarak ikili-parça kromozom kullanılmaktadır.

Literatürde ÇGSP'yi çözen algoritmalar bulunmaktadır. Fakat tez çalışmasında geliştirilen algoritmalar literatürdeki çalışmalara göre daha iyi sonuçlar vermektedir. Ayrıca, geliştirilen algoritmalarda gezgin satıcılar dengeli tur oluşturmaktadır.

3. MATERYAL VE YÖNTEM

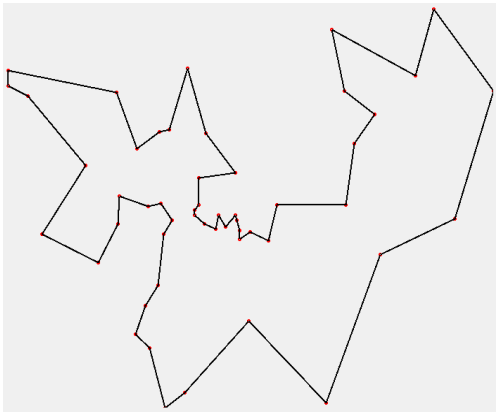
Bu bölümde, tez çalışmasında kullanılan materyaller, algoritmalar ve ÇGSP'nin çözümü için geliştirilen 2 meta sezgisel algoritma detaylı olarak anlatılmaktadır.

3.1. Gezgin Satıcı Problemi

GSP, bir optimizasyon problemidir. İlk olarak 1930'da formüle edilmiştir. GSP'nin amacı, bir gezgin satıcının bulunduğu şehirden başlayıp her şehre sadece bir kez uğradıktan sonra başladığı şehre geri dönen en kısa turu bulmaktır (Dorigo ve Gambardella, 1997). Örnek bir GSP ve çözümü Şekil 3.1'de gösterilmektedir. GSP, bir graf üzerine yerleştirilmiş noktalar ve aralarındaki maliyetler göz önüne alınarak her düğüme yalnız bir kere uğramak şartıyla en uygun maliyetle grafdaki tüm düğümlerin dolaşılması olarak da tanımlanabilir (Kalaycı, 2006). Nokta sayısı arttıkça grafdaki en uygun maliyetli turu belirlemek karmaşıklaşmaktadır. Bundan dolayı, GSP NP-zor problemi olarak sınıflandırılmaktadır (Donald, 2011).

GSP matematiksel olarak şu şekilde ifade edilebilir: Şehirler kümesi $\{c_1, c_2, \ldots, c_n\}$ ve farklı iki şehir arasındaki uzaklık $d(c_i, c_j)$ olsun. Amaç tur uzunluğunu mimimize eden şehirlerin sırasını (π) bulmaktır. GSP formülü Denklem (3.1)'de gösterilmektedir.

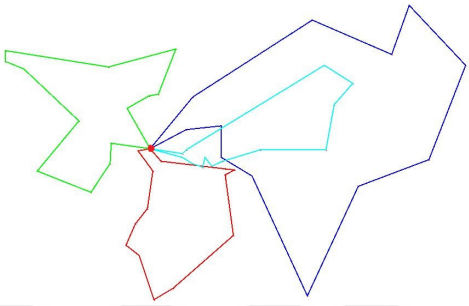
$$\min \sum_{i=1}^{n-1} d(c_{\pi(i)}, c_{\pi(i+1)}) + d(c_{\pi(n)}, c_{\pi(1)}) \tag{3.1}$$



Şekil 3.1. GSP örneği (berlin52)

3.2. Çoklu Gezgin Satıcı Problemi

ÇGSP genel olarak şu şekilde tanımlanabilir: *n* adet şehir ve *m* adet satıcı verilmiş olsun. *m* adet satıcı *tekli depo*da konumlandırılmaktadır. Ziyaret edilecek olan diğer şehirler ara şehirler olarak adlandırılmaktadır. ÇGSP'de amaç, tura depodan başlayan ve turu depoda bitiren *m* adet satıcı için her bir ara şehir bir kere ziyaret edilmek şartıyla gezilen bütün şehirlerin toplam tur maliyetini minimize etmektir (Bektaş, 2006). Şekil 3.2'de örnek bir ÇGSP gösterilmektedir. Bu örnekte 4 tane satıcı bulunmaktadır. Her bir satıcı tura depo şehirden başlamaktadır ve turunu depo şehirde tamamlamaktadır.



Şekil 3.2. Tekli depo ÇGSP örneği (4 satıcı)

Tez kapsamında tekli depo ÇGSP üzerinde çalışılmıştır. Fakat literatürde ÇGSP'nin çeşitli türevleri de bulunmaktadır (Bektaş, 2006):

Çoklu depo: Çoklu depo ÇGSP'de çok depo bulunmaktadır ve her bir depoda bir satıcı bulunmaktadır. Satıcı turunu tamamladıktan sonra başladığı depoya dönmektedir.

Zaman penceresi: Belirli şehirler belirli zaman periyodunda ziyaret edilmek zorundadır.

Satıcı sayısı: ÇGSP'nin bu versiyonunda, satıcı sayısı önceden tam belli olabileceği gibi belirli aralıkla da sınırlandırılabilir.

Literatürde standart bir ÇGSP veri seti bulunmamaktadır. Bundan dolayı, ÇGSP'yi çözen algoritmaların performansını sınamaktaki en büyük eksiklik budur. Bu nedenle, deneylerde sıklıkla GSP veri seti kullanılmaktadır. Bu çalışmada da TSPLIB (Reinelt 1991, 1995) kütüphanesinde bulunan simetrik GSP veri seti kullanılmıştır. Deneylerde kullanılan GSP veri seti Çizelge 3.1'de sunulmaktadır. Çizelgeden de görüldüğü gibi problemin adındaki sayı problemde bulunan şehir sayısını temsil etmektedir. Çizelgedeki optimum sonucu, GSP'nin tek bir gezgin satıcı tarafından çözüldüğünde elde edilen en iyi sonuçtur. Örneğin eil51 problemi tek bir gezgin satıcı

tarafından çözüldüğünde gezgin satıcının en iyi tur uzunluğu 426 olmaktadır. GSP problemi tez kapsamında tekli depo ÇGSP olarak kabul edildiğinden dolayı, birden fazla gezgin satıcı bu problemi çözecek ve doğal olarak toplam tur uzunluğu fazla olacaktır. Örneğin berlin52 probleminin optimum sonucu 7542'dir ve Şekil 3.1'de gösterilmektedir. Şekil 3.2'de gösterilen ÇGSP, berlin52 probleminin 4 gezgin satıcı ile çözümüdür. Şekil 3.2'de görüldüğü gibi her bir satıcı depo şehirden tura başlamakta ve tekrar depo şehrine gelerek turunu tamamlamaktadır. Bundan dolayı toplam tur uzunluğu ÇGSP'nin çözümünde daha fazla olmaktadır.

Çizelge 3.1. Deneylerde kullanılan GSP veri seti

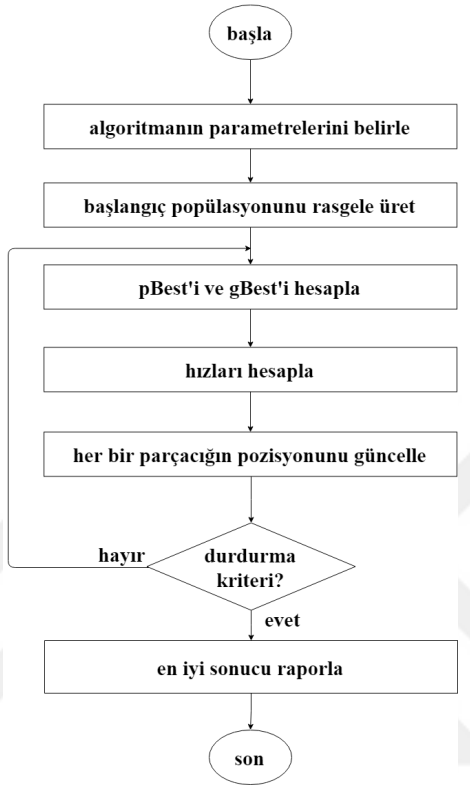
#	Problem adı	Şehir sayısı	Optimum sonucu
1	att48	48	10628
2	eil51	51	426
3	berlin52	52	7542
4	eil76	76	538
5	pr76	76	108159
6	rat99	99	1211
7	bier127	127	118282
8	pr152	152	73682
9	pr226	226	80369

3.3. Parçacık Sürü Optimizasyonu Algoritması (PSO)

PSO algoritması, bir hedefe ulaşmak için çabalayan kuşların sosyal davranışlarına dayalı bir metasezgisel algoritma olarak Kennedy ve Eberhart (1995) tarafından geliştirilmiştir (Bozorg-Haddad ve ark., 2017). PSO algoritması başarılı performans sergilemesinden dolayı endüstri mühendisliği (Nouiri ve ark., 2018), inşaat mühendisliği (AminShokravi ve ark., 2018), elektrik mühendisliği (Lee ve ark., 2018), jeoloji mühendisliği (Hajihassani ve ark., 2018) gibi farklı alanlarda kullanılmıştır.

Sürü zekâsı, dağıtık bir kontrol ve kendi kendini örgütleme kullanarak koordine eden birçok bireyden oluşan doğal ve yapay sistemler ile ilgilenen bir disiplindir. Özellikle sürü zekâsı, bireylerin birbirleriyle ve çevreleriyle olan yerel etkileşimlerinden kaynaklanan kolektif davranışlara odaklanır. PSO algoritması, sürü zekâsının en yaygın örneklerinden biridir. PSO algoritması kuş sürülerinin davranışlarını simüle etmektedir. Gerçek hayattaki kuş, PSO algoritmasında parçacık olarak adlandırılmaktadır ve her bir parçacık problem uzayında bir çözümü temsil etmektedir. Her bir parçacığın hız bilgisi bulunmaktadır ve bu hız bilgisi parçacığın bir sonraki konumunu etkilemektedir. Ayrıca,

parçacığın yeni konumunu etkileyen diğer önemli etkenler parçacığın geçmişteki en iyi konumu (*pBest*) ve sürünün en iyi parçacığıdır (*gBest*). Şekil 3.3, PSO algoritmasının akış diyagramını, Çizelge 3.2 ise algoritmanın sözde kodunu göstermektedir.



Şekil 3.3. PSO algoritmasının akış diyagramı

Algoritmanın başlangıcında, algoritmanın parametreleri belirlenmektedir ve başlangıç popülasyonu rasgele üretilmektedir. Daha sonra her bir parçacığın konumu, uygunluk fonksiyonu ile değerlendirilir. *pBest* ve *gBest* bilgileri hesaplanır. Her bir parçacığın hız bilgisi Denklem (3.2) kullanılarak hesaplanır. Denklem (3.3) ile her bir parçacığın konum bilgisi güncellenir. Bu işlemler durdurma kriteri sağlanıncaya kadar devam eder ve *gBest* bilgisi çıktı olarak raporlanır. Durdurma kriteri genellikle maksimum iterasyon sayısıdır.

$$V_i^d = V_i^d + c_1 * rand1_i^d * (pbest_i^d - X_i^d) + c_2 * rand2_i^d * (gbest^d - X_i^d) \quad (3.2)$$

$$X_i^d = X_i^d + V_i^d \quad (3.3)$$

Burada V_i^d bilgisi i parçacığın d boyutundaki hız bilgisini göstermektedir. X_i^d ise i parçacığın d boyutundaki konum bilgisini göstermektedir. c_1 ve c_2 parametreleri ise

ivmelenme katsayılarıdır ve *pBest* ve *gBest*'e bağlı olarak parçacığın optimuma doğru olan hareketlerini kontrol altında tutmaktadır. *rand1* ve *rand2* 0 ile 1 arasında üretilen rasgele sayılardır.

Çizelge 3.2. PSO algoritmasının sözde kodu

Algoritma 1: PSO Algoritması

```
int N                                   /* parçacık sayısı */
int T                                   /* maksimum iterasyon sayısı */
double c1, c2                   /* ivmelenme katsayıları */
N, T, c1, c2 parametrelerine ilk değer ata
Başlangıç popülasyonunu üret ve parçacıkların uygunluk değerlerini hesapla
Parçacıkların hızlarını rasgele olarak belirle
for i ← 1, N do
        pBest_i = parçacık i
end for
gBest = popülasyondaki en iyi çözüm
int t=0;                                /* iterasyon sayacına ilk değer ata*/
while (t < T)
        t++;                            /* iterasyon sayacını artır */
        for i ← 1, N do
                if uygunluk(parçacık_i) pBest_i'den daha iyiyse then
                        pBest_i = parçacık i
                end if
                if pBest_i gBest'den daha iyiyse then
                        gBest = pBest_i
                end if
        end for
        for i ← 1, N do
                Denklem (3.2)'i kullanarak hızı güncelle
                Denklem (3.3)'i kullanarak konumu güncelle
                Uygunluğu hesapla
        end for
end while
gBest çözümü raporla
```

3.4. Açgözlü Rasgeleleştirilmiş Adaptif Arama Prosedürü (GRASP) Algoritması

Feo ve Resende (1989) tarafından geliştirilen GRASP (Greedy Randomized Adaptive Search Procedure) algoritması genellikle kombinasyonlu optimizasyon problemlerine uygulanan bir metasezgisel algoritmadır. GRASP algoritması tek-çözümlü bir metasezgisel yöntemidir. Tek-çözümlü metasezgisel yönteminde, mevcut bir tane çözüm bulunmaktadır ve bu çözüm komşuluklar sayesinde arama uzayında yeni çözüme geçmektedir (Talbi, 2009). GRASP algoritmasında her bir iterasyon 2 safhadan oluşmaktadır: yapım safhası ve lokal arama safhası. Yapım safhası uygun bir çözüm

oluşturur. Daha sonra, üretilen çözümü iyileştirmek için lokal arama safhasına geçilir. Algoritmanın sonunda en iyi çözüm sonuç olarak raporlanır (Gendreau ve Potvin, 2010).

Minimizasyon problemi için, algoritmanın sözde kodu Çizelge 3.3, Çizelge 3.4 ve Çizelge 3.5'de sunulmaktadır. Kullanıcının belirlediği maksimum iterasyon sayısı T ve eşik değeri α algoritmaya parametre olarak gönderilmektedir. Eşik değeri α, 0 ile 1 arasında değer almaktadır. Algoritma, maksimum iterasyon sayısına ulaşıncaya kadar her iterasyonda sırasıyla *Açgözlü_Randomize_Yapım* metodunu ve *Lokal_Arama* metodunu çağırmaktadır. *Açgözlü_Randomize_Yapım* metodunun bulduğu *çözüm*, *Lokal_Arama* metodu tarafından iyileştirilmeye çalışılmaktadır. *Lokal_Arama* metodu tarafından elde edilen *çözüm*, bu aşamaya kadar bulunan *enİyiÇözüm* değerinden daha iyiyse *çözüm* değeri *enİyiÇözüm* değerine atanarak *enİyiÇözüm* değeri güncellenmektedir.

Çizelge 3.3. GRASP algoritmasının sözde kodu

Algoritma 2: GRASP Algoritması

```
1:  int T;                                  /* maksimum iterasyon sayısı */
2:  double α;                               /* eşik değeri */
3:  T, α parametrelerine ilk değer ata;
4:  for i ← 1, T do
5:          çözüm = Açgözlü_Randomize_Yapım(α);
6:          çözüm = Lokal_Arama(çözüm);
7:          if çözüm enİyiÇözüm'den daha iyiyse then
8:                  enİyiÇözüm = çözüm;
9:          end if
10: end for
11: enİyiÇözüm değerini raporla;
```

GRASP algoritmasının ilk safhası olan yapım safhası bir yapıcı sezgiseldir (constructive heuristic). Yapıcı sezgiseller, başlangıçta boş olan bir çözüme eklemeler yaparak parçalı çözüm üretirler. Bu işlem çözüm tam oluncaya kadar devam etmektedir (Alba, 2005). Bu safhanın sözde kodu Çizelge 3.4'de sunulmaktadır. Algoritmadaki kısaltma ifadesinin açıklaması şu şekildedir: e problem elemanı, E problem elemanlarından oluşan sonlu küme, C aday kümedir. $c(e)$ maliyeti temsil etmektedir. Kısıtlı aday listesi (KAL) maliyeti düşük olan elemanları içermektedir. Örneğin GSP için, e iki şehri birbirine bağlayan yolu temsil etmektedir, E ise şehirleri birbirine bağlayan tüm yollardır. C bir şehirden gidilebilecek bütün şehirleri temsil etmektedir. $c(e)$ iki şehri birbirine bağlayan yolun uzunluğudur. c_{min} gidilebilecek en yakın şehrin maliyetini ve c_{max} ise gidilebilecek en uzak şehrin maliyetini göstermektedir. Kısıtlı aday listesi (KAL) gidilebilecek en yakın şehirlerin listesidir (Feo ve Resende, 1995).

Algoritmanın yapım safhasının çalışması şu şekildedir: Başlangıçta *çözüm* boş kümedir. Aday küme *C* oluşturulur ve aday kümedeki her bir elemanın $c(e)$ maliyeti hesaplanır. Aşağıdaki adımlar *C* aday kümede eleman olduğu müddetçe devam eder ve yapım safhasının sonunda elde edilen *çözüm* geri döndürülür:

- Aday kümedeki elemanlardan en az maliyeti c_{min} ve en fazla maliyeti c_{max} hesaplanır.
- c_{min}, c_{max} ve α eşik değeri kullanılarak kısıtlı aday listesi oluşturulur.
- Kısıtlı aday listesinden rasgele bir eleman *e* seçilir ve *çözüm*e eklenir.
- Aday küme *C* güncellenir ve
- $c(e)$ maliyetleri tekrar hesaplanır.

Çizelge 3.4. Yapım aşaması safhasının sözde kodu

Algoritma 3: Açgözlü_Randomize_Yapım(α)

```
çözüm = Ø;
aday kümeyi oluştur: C ← E;
her bir e ∈ C için c(e) maliyetleri hesapla;
while (C ≠ Ø)
        c_min ← min{ c(e) | e ∈ C};
        c_max ← max{ c(e) | e ∈ C};
        KAL ← { e ∈ C | c(e) ≤ c_min + α (c_max-c_min)};
        KAL listeden rasgele bir eleman e seç;
        çözüm ← çözüm ∪ e;
        C aday kümeyi güncelle;
        her bir e ∈ C için c(e) maliyetleri tekrar hesapla;
end for
return çözüm;
```

Algoritmanın lokal arama safhasının sözde kodu Çizelge 3.5'de sunulmaktadır. Burada $f(.)$, amaç fonksiyonunu temsil etmektedir. Tez çalışmasında, yerel arama algoritması olarak 2-opt algoritması kullanılmaktadır. Mevcut *çözüm* üzerinde yerel değişiklikler yapılarak *yeniÇözüm* elde edilir. *yeniÇözüm* var olan *çözüm*den iyiyse algoritma *yeniÇözüm* ile iyileştirme işlemlerine devam etmektedir. Lokal arama safhasının sonunda elde edilen iyileştirilmiş *çözüm* geri döndürülür.

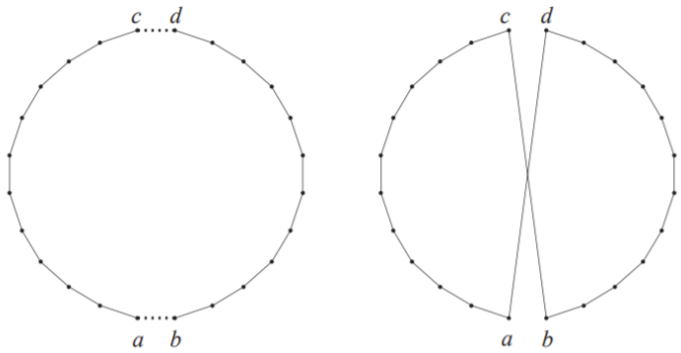
Çizelge 3.5. Lokal arama safhasının sözde kodu

Algoritma 4: Lokal_Arama(çözüm)

```
while (çözüm yerel olarak optimal değil)
        f(yeniÇözüm) < f(çözüm) olan yeniÇözüm bul;
        çözüm = yeniÇözüm;
end while
return çözüm
```

3.5. 2-opt Algoritması

2-opt algoritması Croes (1958) tarafından geliştirilen bir lokal arama algoritmasıdır. 2-opt algoritması bir GSP turundaki iki kenarı siler. Bu şekilde tur iki parçaya ayrılmış olur. Daha sonra bu iki parçayı bir GSP turu olacak şekilde farklı olasılıkları deneyerek tekrar birleştirir. Yeni tur eski turdan daha iyiyse yeni tur çözüm olur. 2-Opt algoritması iyileştirme bulamayıncaya kadar turu silmeye ve yeniden oluşturmaya devam eder (Johnson ve McGeoch, 1997; Gutin ve Punnen, 2006). 2-Opt lokal arama algoritmasının çalışma örneği Şekil 3.4'de görülmektedir. Şekilde görüldüğü gibi, tur 2 parçaya ayrılmıştır ve bu 2 parça olası bir çözüm için farklı tur oluşturacak şekilde tekrar birleştirilmiştir.



Şekil 3.4. 2-opt algoritmasının örnek uygulaması (Johnson ve McGeoch, 1997)

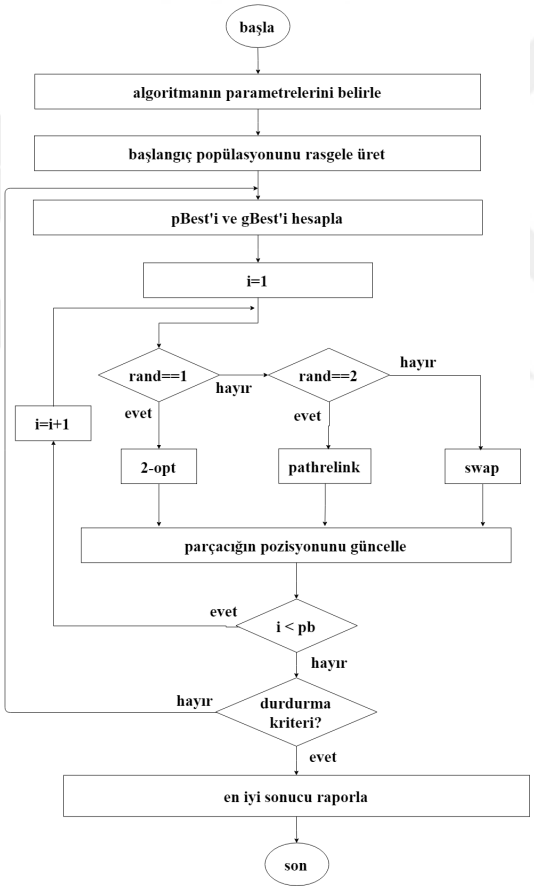
3.6. Geliştirilen Yöntemler

Bu bölümde, ÇGSP'yi çözmek için geliştirilen yöntemler anlatılmaktadır. Evrimsel algoritmalar GSP'yi genellikle şehirlerin permütasyonu ile çözmektedirler. Bu çalışmada da ÇGSP çözümü permütasyon olarak temsil edilmektedirler.

3.6.1. ÇGSP'yi çözmek için geliştirilen PSO tabanlı meta sezgisel algoritma (APSO)

Bu bölümde ÇGSP'yi çözmek için geliştirilen PSO tabanlı meta sezgisel algoritma Ayrıklaştırılmış Parçacık Sürü Optimizasyonu (APSO) detaylı olarak

anlatılmaktadır. Geliştirilen yöntem, PSO ve 2-opt algoritmalarını, path-relink ve takas (swap) operatörlerini kullanarak ÇGSP'yi çözmektedir. Bu çalışmada, kesikli optimizasyon problemi olan ÇGSP'nin çözümü için PSO tabanlı bir algoritma geliştirildiği için geliştirilen algoritma Ayrıklaştırılmış Parçacık Sürü Optimizasyonu (APSO) algoritması olarak isimlendirilmektedir. APSO algoritmasının başlangıç popülasyonu rasgele oluşturulur. Bu parçacıkların her biri ÇGSP'nin çözümünü temsil etmektedir. APSO algoritması ÇGSP'yi çözmek için arama uzayında yeni çözümler bulmaktadır. 2-opt algoritması, path-relink ve takas operatörleri ise APSO algoritmasının bulduğu çözümleri iyileştirmektedir. Geliştirilen APSO algoritmasının akış diyagramı Şekil 3.5'de gösterilmektedir.



Şekil 3.5. APSO algoritmasının akış diyagramı (pb: popülasyon boyutu, i: parçacık indisi)

Denklem (3.2)'de görüldüğü gibi, PSO'da bir parçacığın hareketini etkileyen üç ana etken bulunmaktadır: parçacığın bir önceki hız bilgisi, parçacığın şu ana kadarki en iyi konumu (*pBest*) ve tüm sürüdeki parçacıkların şu ana kadarki en iyi konumu (*gBest*). PSO algoritması sürekli optimizasyon problemleri için geliştirildiği için, algoritmanın saf hali kesikli optimizasyon problemlerine doğrudan uygulanamamaktadır. Bu çalışmada,

PSO algoritmasını ÇGSP'nin çözümünde kullanabilmek için algoritmada bazı değişiklikler yapılmıştır. Bu değişiklikler, bir parçacığın yeni konumunu 2-opt lokal arama algoritması, path-relink operatörü ve takas operatörü ile belirlemesidir.

Geliştirilen algoritmada gezgin satıcı sayısı *m*, kullanıcı tarafından belirlenmektedir. Ayrıca, dengeli bir tur oluşturabilmek için satıcı başına düşen gezilecek şehir sayısının bir üst limiti *K* bulunmaktadır. Literatürde üst limit *K*'yı belirlemek için farklı öneriler bulunulmaktadır (Necula ve ark., 2015a). Bu çalışmada ise, üst limit *K* değeri, Denklem (3.5) ile hesaplanmaktadır. TSPLIB kütüphanesindeki çözülmek istenen GSP örneği algoritmaya aktarıldığında *K* değeri hesaplanmaktadır. Örneğin berlin52 problemi 5 gezgin satıcı ile çözülmek istendiğinde *K* değeri 12 olmaktadır. Yani berlin52 problemi için bir satıcı 12 şehirden fazla şehir gezememektedir.

$$K = \lfloor ((d-1)+m)/m \rfloor + 1 \tag{3.5}$$

Burada *K* üst limiti, *d* problemin boyutunu yani şehir sayısını ve *m* gezgin satıcı sayısını temsil etmektedir.

Geliştirilen algoritmada parçacık, ÇGSP'nin bir çözümünü temsil etmektedir. Her bir parcacık, *m* adet alttur bilgisini tutmaktadır. Bir alttur sadece bir gezgin satıcı tarafından oluşturulmaktadır. Her bir şehrin id bilgisi bulunmakta ve bu bilgi GSP örneği dosyasındaki şehirlerin sırasıdır. Dosyadaki ilk şehrin id değeri 0'dır ve bu şehir depo şehirdir. Çizelge 3.6'da örnek bir parçacık alttur bilgisi gösterimi sunulmaktadır. Her bir alttur, depo şehir (id değeri 0) ile başlamaktadır. Satıcı tarafından oluşturulan altturda son ziyaret edilen şehirden sonra tekrar depo şehrine dönülmektedir. Örneğin Çizelge 3.6'da 1 indisli satıcı 41. şehirden sonra turu tamamlamak için tekrar 0. şehre dönmektedir.

Çizelge 3.6. Örnek bir parçacık alttur bilgisi (5 gezgin satıcı)

Satıcı indisi	Alttur	Gezilen şehir sayısı
1	[0, 48, 35, 33, 45, 24, 11, 27, 28, 1, 6, 41]	12
2	[0, 25, 46, 13, 51, 12, 26, 14, 23, 47, 36, 34]	12
3	[0, 31, 42, 32, 10, 50, 3, 5, 4, 37, 39, 38]	12
4	[0, 21, 30, 17, 16, 2, 40, 7, 8, 9, 18, 44]	12
5	[0, 43, 15, 49, 19, 22, 29, 20]	8

Geliştirilen APSO algoritmasının akış diyagramı Şekil 3.5'de ve sözde kodu Çizelge 3.7 gösterilmektedir. Geliştirilen algoritmanın çalışması şu şekildedir: Öncelikle algoritmanın parametreleri belirlenmektedir. Bunlar parçacık sayısı, maksimum iterasyon

sayısı, satıcı sayısı, satıcı başına düşen gezilecek şehir sayısının üst limit değeri *K*. Daha sonra PSO algoritmasının başlangıç popülasyonu rasgele üretilmektedir. Burada dikkat edilmesi gereken husus, bir şehrin sadece bir satıcı tarafından ziyaret edilmesi gerekliliğidir. Yani, depo şehri hariç herhangi bir şehir sadece bir altturda bulunmak zorundadır. Böylece her bir parçacık ÇGSP için *m* adet altturdan oluşan bir çözüme sahip olmaktadır. Daha sonra iterasyon döngüsü başlamakta ve maksimum iterasyona ulaşıncaya kadar aşağıdaki adımlar tekrar etmektedir. Öncelikle *pBest* ve *gBest* bilgisi hesaplanmaktadır. Daha sonra her bir parçacık için şu adımlar uygulanmaktadır: rasgele bir *rand* sayısı üretilmektedir. $rand \in \{1, 2, 3\}$. *rand* değeri 1 ise parçacıktaki altturlara 2-opt algoritması uygulanmaktadır. *rand* değeri 2 ise *pBest* bilgisi kullanılarak parçacıktaki altturlara path-relink operatörü uygulanmaktadır. *rand* değeri 3 ise parçacıktaki altturlara takas operatörü uygulanmaktadır. Algoritmanın sonunda *gBest* bilgisi çıktı olarak raporlanır. Algoritmada kullanılan path-relink operatörü ve takas operatörü aşağıda anlatılmaktadır.

Çizelge 3.7. Geliştirilen APSO algoritmasının sözde kodu

Algoritma 5: APSO Algoritması

```
1:  int N                                   /* parçacık sayısı */
2:  int T                                   /* maksimum iterasyon sayısı */
3:  int K                                   /* gezilecek şehir üst limit sayısı */
4:  N, T, K parametrelerine ilk değer ata
5:  Başlangıç popülasyonunu rasgele üret ve parçacıkların uygunluk değerlerini hesapla
6:  for i ← 1, N do
7:      pBest_i = parçacık i
8:  end for
9:  gBest = popülasyondaki en iyi çözüm
10: int t=0;                                /* iterasyon sayacına ilk değer ata*/
11: while (t < T)
12:     t++;                                /* iterasyon sayacını artır */
13:     for i ← 1, N do
14:         if uygunluk(parçacık_i) pBest_i'den daha iyiyse then
15:             pBest_i = parçacık i
16:         end if
17:         if pBest_i gBest'den daha iyiyse then
18:             gBest = pBest_i
19:         end if
20:     end for
21:     for i ← 1, N do
22:         rand değerini üret
23:         if rand==1 then
24:             i. parçacığa 2-opt algoritmasını uygula
25:         else if rand==2 then
26:             i. parçacığa pathrelink operatörünü uygula
27:         else if rand==3 then
28:             i. parçacığa takas operatörünü uygula
29:         end if
30:         Uygunluğu hesapla
31:     end for
32: end while
33: gBest çözümü raporla
```

Path-relink operatörü, aramayı elit sonuçlara doğru yönlendirmek için bir yoğunlaştırma stratejisi olarak Glover (1963) tarafından geliştirilmiştir (Resende ve ark., 2010). Path-relink operatörü, iki çözüm arasında yeni çözümleri üreterek bir zincir oluşturmaktadır. Bu çalışmada kaynak çözüm parçacığın oluşturduğu turdur. Hedef çözüm ise parçacığın *pBest* bilgisidir. Kaynak çözüm ile hedef çözüm arasında ara çözümler üretilmekte ve bu ara çözümlerin kalitesi değerlendirilmektedir. Çizelge 3.8'de path-relink operatörünün kullanımına bir örnek verilmektedir. Çizelgede kaynak; parçacığa ait altturu ve hedef ise *pBest* bilgisindeki altturu temsil etmektedir. Kaynak çözüm ile hedef çözüm arasında 4 tane çözüm üretilmektedir. Bu çözümlerin uygunluk değerleri hesaplanmaktadır. Bu çözümler *pBest* veya *gBest* bilgisinden daha iyiyse *pBest* ve *gBest* bilgisi güncellenmektedir.

Çizelge 3.8. Path-relink operatörü

Adım	Alttur
kaynak	[0, 48, 35, **33**, 45, 24, 11, 27, 28, 1, 6, 41]
1	[0, 48, **33**, 35, 45, 24, 11, 27, 28, 1, 6, 41]
2	[0, **33**, 48, 35, 45, 24, 11, 27, 28, 1, 6, 41]
3	[0, 33, 48, 35, **24**, 45, 11, 27, 28, 1, 6, 41]
4	[0, 33, 48, **24**, 35, 45, 11, 27, 28, 1, 6, 41]
hedef	[0, 33, **24**, 48, 35, 45, 11, 27, 28, 1, 6, 41]

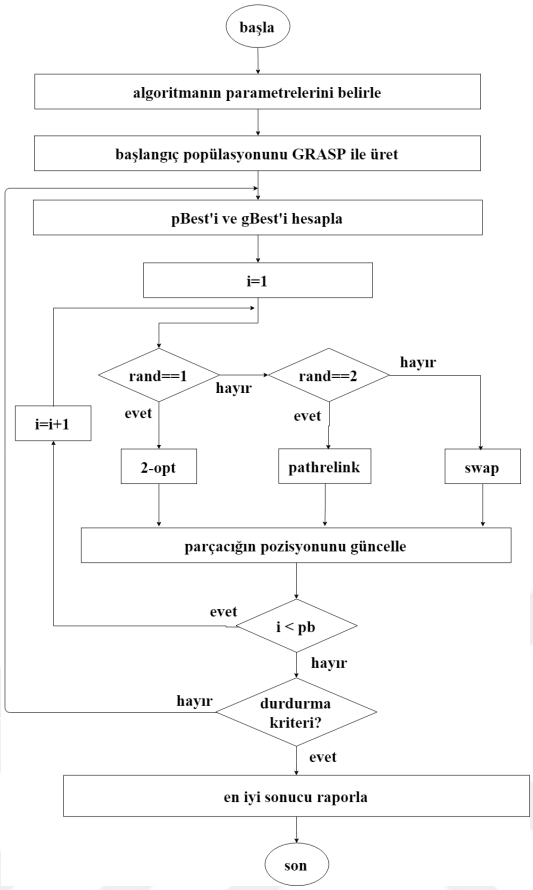
Takas operatörü, bir parçacıkta rasgele seçilen 2 altturun rasgele seçilen birer şehirlerinin yer değiştirmesi işlemidir. Çizelge 3.9'da takas operatörünün kullanımına bir örnek verilmektedir. Öncelikle rasgele iki tane satıcı belirlenmektedir. Çizelge 3.9'da bu satıcılar 1 ve 2'dir. Daha sonra bu satıcılardan rasgele birer tane şehir seçilmektedir ve bu şehirler takas edilmektedir. Örnekte 45. ve 36. şehirler seçilmektedir ve takas edilmektedir.

Çizelge 3.9. Takas operatörü

Satıcı indisi	Alttur
1	[0, 48, 35, 33, **45**, 24, 11, 27, 28, 1, 6, 41]
2	[0, 25, 46, 13, 51, 12, 26, 14, 23, 47, **36**, 34]
	Takas işlemi ⇩
1	[0, 48, 35, 33, **36**, 24, 11, 27, 28, 1, 6, 41]
2	[0, 25, 46, 13, 51, 12, 26, 14, 23, 47, **45**, 34]

3.6.2. ÇGSP'yi çözmek için geliştirilen PSO ve GRASP tabanlı hibrit bir meta sezgisel algoritma (HAPSO)

Bu bölümde ÇGSP'yi çözmek için geliştirilen PSO ve GRASP algoritmaları tabanlı hibrit bir meta sezgisel algoritma (HAPSO) anlatılmaktadır. Geliştirilen yöntem GRASP, PSO ve 2-opt algoritmalarını ve path-relink ve takas operatörlerini kullanarak ÇGSP'yi çözmektedir. GRASP algoritması PSO algoritmasının başlangıç popülasyonunu oluşturmak için kullanılmaktadır. PSO algoritması ÇGSP'yi çözmek için arama uzayında yeni çözümler bulmaktadır. 2-opt algoritması ise PSO algoritmasının bulduğu çözümleri iyileştirmektedir. Geliştirilen yöntemin akış diyagramı Şekil 3.6'da gösterilmektedir. HAPSO algoritması APSO algoritmasının geliştirilmiş şeklidir. APSO algoritmasında algoritmanın başlangıç popülasyonu rasgele üretilmektedir. Fakat HAPSO algoritmasında başlangıç popülasyonu GRASP algoritması ile üretilmektedir.



Şekil 3.6. HAPSO algoritmasının akış diyagramı (pb: popülasyon boyutu, i: parçacık indisi)

HAPSO algoritmasında gezgin satıcı sayısı *m*, kullanıcı tarafından belirlenmektedir. Ayrıca, dengeli bir tur oluşturabilmek için satıcı başına düşen gezilecek şehir sayısının bir üst limiti *K* bulunmaktadır. Bu üst limit *K* Denklem (3.5) ile hesaplanmaktadır. TSPLIB kütüphanesindeki çözülmek istenen GSP örneği algoritmaya aktarıldığında *K* değeri hesaplanmaktadır. Örneğin berlin52 problemi 5 gezgin satıcı ile çözülmek istendiğinde *K* değeri 12 olmaktadır. Yani berlin52 problemi için bir satıcı 12 şehirden fazla şehir gezememektedir.

Geliştirilen algoritmada parçacık, ÇGSP'nin bir çözümünü temsil etmektedir. Her bir parcacık, *m* adet alttur bilgisini tutmaktadır. Bir alttur sadece bir gezgin satıcı tarafından oluşturulmaktadır. Her bir şehrin id bilgisi bulunmakta ve bu bilgi GSP örneği dosyasındaki şehirlerin sırasıdır. Dosyadaki ilk şehrin id değeri 0'dır ve bu şehir depo şehirdir. Çizelge 3.6'da örnek bir parçacık alttur bilgisi gösterimi sunulmaktadır. Her bir alttur, depo şehir (id değeri 0) ile başlamaktadır. Satıcı tarafından oluşturulan altturda son ziyaret edilen şehirden sonra tekrar depo şehrine dönülmektedir. Örneğin Çizelge 3.6'da 1 indisli satıcı 41. şehirden sonra turu tamamlamak için tekrar 0. şehre dönmektedir.

GRASP algoritmasındaki eşik değeri α, açgözlülük derecesini belirlemektedir ve 0 ile 1 arasında değer almaktadır. 0'a yaklaştıkça açgözlülük artmaktadır. 1'e yaklaştıkça da rasgelelik artmaktadır. α değerinin [0.4, 0.6] aralığında daha iyi sonuçlar verdiği belirtilmektedir (Feo ve Resende, 1995). Bu çalışmada eşik değeri α, 0.4 olarak alınmaktadır. Ayrıca, GRASP algoritmasında lokal arama algoritması olarak 2-opt sezgisel algoritması kullanılmaktadır.

Geliştirilen algoritmanın akış diyagramı Şekil 3.6'da ve sözde kodu Çizelge 3.10'da gösterilmektedir. Geliştirilen algoritmanın çalışması şu şekildedir: Öncelikle algoritmanın parametreleri belirlenmektedir. Bunlar parçacık sayısı, maksimum iterasyon sayısı, satıcı sayısı, satıcı başına düşen gezilecek şehir sayısının üst limit değeri K ve GRASP algoritmasındaki eşik değeri α'dır. Daha sonra PSO algoritmasının başlangıç popülasyonu GRASP algoritması ile üretilmektedir. Böylece her bir parçacık ÇGSP için m adet altturdan oluşan bir çözüme sahip olmaktadır. Daha sonra iterasyon döngüsü başlamakta ve maksimum iterasyona ulaşıncaya kadar aşağıdaki adımlar tekrar etmektedir. Öncelikle *pBest* ve *gBest* bilgisi hesaplanmaktadır. Daha sonra her bir parçacık için şu adımlar uygulanmaktadır: rasgele bir *rand* sayısı üretilmektedir. $rand \in \{1, 2, 3\}$. *rand* değeri 1 ise parçacıktaki altturlara 2-opt algoritması uygulanmaktadır. *rand* değeri 2 ise *pBest* bilgisi kullanılarak parçacıktaki altturlara pathrelink operatörü uygulanmaktadır. *rand* değeri 3 ise parçacıktaki altturlara takas operatörü uygulanmaktadır. Algoritmanın sonunda *gBest* bilgisi çıktı olarak raporlanır. Algoritmada kullanılan path-relink operatörü ve takas operatörü bir önceki bölümde anlatılmaktadır.

Çizelge 3.10. Geliştirilen HAPSO algoritmanın sözde kodu

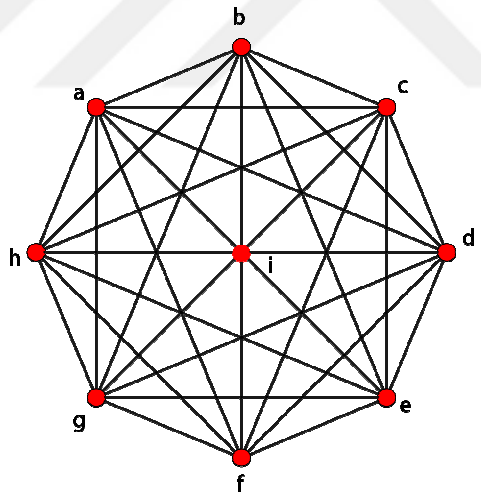
Algoritma 6: HAPSO Algoritması

```
1:  int N                                   /* parçacık sayısı */
2:  int T                                   /* maksimum iterasyon sayısı */
3:  int K                                   /* gezilecek şehir üst limit sayısı */
4:  double α                                /* eşik değeri */
5:  N, T, K, α parametrelerine ilk değer ata
6:  Başlangıç popülasyonunu GRASP ile üret ve parçacıkların uygunluk değerlerini hesapla
7:  for i ← 1, N do
8:      pBest_i = parçacık i
9:  end for
10: gBest = popülasyondaki en iyi çözüm
11: int t=0;                                /* iterasyon sayacına ilk değer ata*/
12: while (t < T)
13:     t++;                                /* iterasyon sayacını artır */
14:     for i ← 1, N do
15:         if uygunluk(parçacık_i) pBest_i'den daha iyiyse then
16:             pBest_i = parçacık i
17:         end if
18:         if pBest_i gBest'den daha iyiyse then
19:             gBest = pBest_i
20:         end if
21:     end for
22:     for i ← 1, N do
23:         rand değerini üret
24:         if rand==1 then
25:             i. parçacığa 2-opt algoritmasını uygula
26:         else if rand==2 then
27:             i. parçacığa pathrelink operatörünü uygula
28:         else if rand==3 then
29:             i. parçacığa takas operatörünü uygula
30:         end if
31:         Uygunluğu hesapla
32:     end for
33: end while
34: gBest çözümü raporla
```

4. ARAŞTIRMA SONUÇLARI VE TARTIŞMA

Bu bölümde, geliştirilen APSO ve HAPSO algoritmalarının deneysel sonuçları sunulmaktadır. Algoritmaların geliştirilmesinde ve deneylerde kullanılan bilgisayarın teknik özellikleri şu şekildedir: Windows 10 işletim sistemi, intel i5 3 GHz, 4 GB hafıza, java SE 8.

ÇGSP'de depo şehrinin seçimine bağlı olarak toplam tur uzunluğu değişmektedir. Depo şehrinin seçimiyle ilgili olarak literatürde belirli bir ölçüt bulunmamaktadır. Bundan dolayı literatürdeki çalışmaların deneysel sonuçlarının karşılaştırılması uygun ve adil olmamaktadır. Bu çalışmada depo şehri, tur uzunluğunu kısaltmak için özel olarak belirlenmemiştir. Bunun yerine, depo şehri okunan TSPLIB dosyalarındaki ilk şehir olarak belirlenmiştir. ÇGSP'de depo şehrine göre toplam tur uzunluğunun değiştiğini bir örnek üzerinde gösterelim. Şekil 4.1'de bir düzgün sekizgen gözükmektedir. Şekilde toplam 9 adet şehir bulunmaktadır. Sekizgenin kenar uzunluğunu 10 birim olarak kabul edersek şehirler arasındaki uzaklık mesafeleri Çizelge 4.1'de gösterilmektedir.

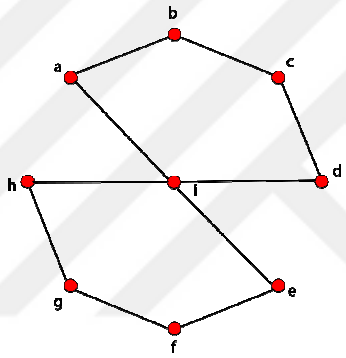


Şekil 4.1. Düzgün sekizgen

Çizelge 4.1. Düzgün sekizgen için uzaklık matrisi

	a	b	c	d	e	f	g	h	i
a	0	10	18.46	24.14	26.1	24.14	18.46	10	13.05
b	10	0	10	18.46	24.14	26.1	24.14	18.46	13.05
c	18.46	10	0	10	18.46	24.14	26.1	24.14	13.05
d	24.14	18.46	10	0	10	18.46	24.14	26.1	13.05
e	26.1	24.14	18.46	10	0	10	18.46	24.14	13.05
f	24.14	26.1	24.14	18.46	10	0	10	18.46	13.05
g	18.46	24.14	26.1	24.14	18.46	10	0	10	13.05
h	10	18.46	24.14	26.1	24.14	18.46	10	0	13.05
i	13.05	13.05	13.05	13.05	13.05	13.05	13.05	10	0

Şekil 4.1'deki düzgün sekizgeni bir ÇGSP olarak kabul edelim. 2 gezgin satıcı ile bu problemi çözelim. Depo şehrini *i* olarak kabul ettiğimizde, problemin en iyi çözümü Şekil 4.2'de gösterilmektedir ve toplam tur uzunluğu 112.2 birimdir.

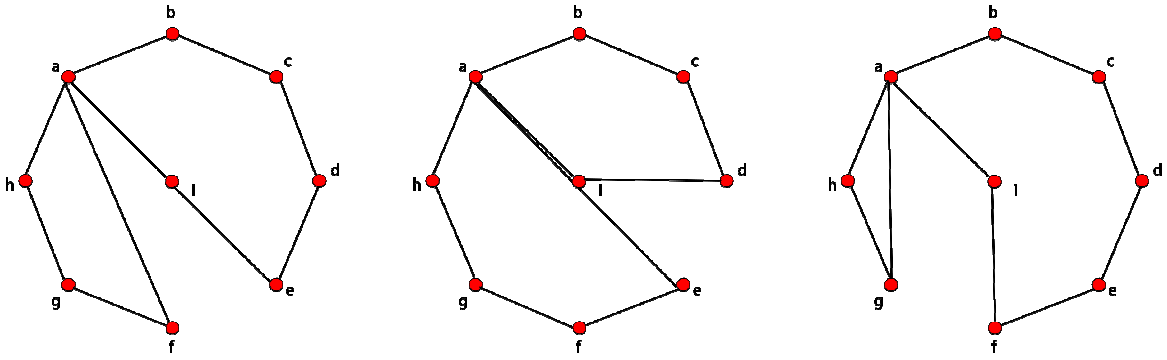


tur uzunluğu: 112.2

Şekil 4.2. 2 satıcı ve depo *i* şehri için sekizgenin en iyi çözümü

Depo şehrini *a* olarak kabul ettiğimizde ise, problemin olası en iyi çözümleri Şekil 4.3'de gösterilmektedir. En iyi toplam tur uzunluğu ise 114.56 birim olarak hesaplanmaktadır (Şekil 4.3.c). Fakat bu çözümde ise satıcıların turları arasında bir dengesizlik bulunmaktadır. Birinci satıcı 7 şehri ziyaret ederken diğer satıcı 3 şehri ziyaret etmektedir. Ayrıca alt tur uzunlukları da sırasıyla 76.1 ve 38.46 birimdir. Şekil 4.3.a ve Şekil 4.3.b'deki çözümlerde satıcıların turları arasında daha dengeli bir yapı bulunmaktadır. Şekil 4.3.a'daki çözümün toplam tur uzunluğu 120.24 birim olarak hesaplanmaktadır. Birinci satıcı 6 şehri ziyaret ederken diğer satıcı 4 şehri ziyaret etmektedir. Ayrıca alt tur uzunlukları da sırasıyla 66.1 ve 54.14 birimdir. Şekil 4.3.b'deki çözümün toplam tur uzunluğu 122.2 birim olarak hesaplanmaktadır. Birinci satıcı 5 şehri ziyaret ederken diğer satıcı da 5 şehri ziyaret etmektedir. Ayrıca alt tur uzunlukları da sırasıyla 56.1 ve 66.1 birimdir.

Özet olarak, Şekil 4.2 ve Şekil 4.3'de de görüldüğü gibi depo şehrinin toplam tur uzunluğu üzerinde etkisi bulunmaktadır.



tur uzunluğu: 120.24 a) tur uzunluğu: 122.2 b) tur uzunluğu: 114.56 c)

Şekil 4.3. 2 satıcı ve depo *a* şehri için sekizgenin en iyi çözümleri

APSO algoritmasındaki 2-opt algoritmasının, takas ve path-relink operatörlerinin etkisini incelemek için, algoritmanın çalışması boyunca bu yöntemlerden sadece bir tanesi kullanılarak APSO algoritması çalıştırılmıştır. Örneğin, sadece path-relink operatörün etkisini incelemek için APSO algoritmasındaki *rand* değeri algoritmanın çalışması boyunca 2'ye eşitlenmiştir. Karşılaştırma sonuçları Çizelge 4.2, Çizelge 4.3 ve Çizelge 4.4'de verilmiştir. Karşılaştırmada 5 farklı simetrik GSP örneği kullanılmıştır: att48, berlin52, bier127, pr76 ve rat99. Algoritmaların parametreleri şu şekilde ayarlanmıştır: parçacık sayısı 50, maksimum iterasyon sayısı 2000, gezgin satıcı sayısı sırasıyla 2, 3 ve 4 olarak ayarlanmıştır. Ayrıca, her bir deney bağımsız olarak 20 kez çalıştırılmıştır. Çizelge 4.2, Çizelge 4.3 ve Çizelge 4.4'de 20 kez çalıştırma sonucunda elde edilen değerlerin ortalaması ve standart sapması sunulmaktadır. Çizelgelerden görüldüğü gibi, APSO algoritması diğer yöntemlerden daha iyi sonuçlar elde etmiştir.

Çizelge 4.2. 2 satıcı için takas, path-relink operatörlerinin ve 2-opt algoritmasının etkisini

Örnek	APSO	Sadece takas operatörü	Sadece path-relink operatörü	Sadece 2-opt algoritması
att48	44121.4 ± 569.7	64837 ± 2935.8	51055 ± 1628.8	47198 ± 522.4
berlin52	9872.9 ± 244.4	14317.1 ± 612	11327 ± 289.9	10295.7 ± 373.4
bier127	161693.8 ± 2851.1	348738.5 ± 7501.2	194093.1 ± 4264.4	173372.4 ± 4024.8
pr76	148538.8 ± 2377.9	270853.6 ± 8026.8	178823.1 ± 4787	155493.1 ± 3866.9
rat99	1698.6 ± 22.4	3731.2 ± 98.9	2116.4 ± 46.5	1987.4 ± 39.8

Çizelge 4.3. 3 satıcı için takas, path-relink operatörlerinin ve 2-opt algoritmasının etkisini

Örnek	APSO	Sadece takas operatörü	Sadece path-relink operatörü	Sadece 2-opt algoritması
att48	54393.1 ± 1375.4	71052.1 ± 3530.5	64800 ± 1615	58988.9 ± 855.4
berlin52	11583.1 ± 319.4	14946.9 ± 725.3	13774.6 ± 423.2	12709.1 ± 352.4
bier127	193300.1 ± 3270.8	336899 ± 10012.4	228338.8 ± 4601.8	200035.9 ± 3188.6
pr76	179152.1 ± 2963	283189.8 ± 9757.4	220951.5 ± 6932.2	193438.4 ± 3914.2
rat99	2048.9 ± 20.1	3787.2 ± 135.7	2519.3 ± 62.6	2407.6 ± 47.4

Çizelge 4.4. 4 satıcı için takas, path-relink operatörlerinin ve 2-opt algoritmasının etkisini

Örnek	APSO	Sadece takas operatörü	Sadece path-relink operatörü	Sadece 2-opt algoritması
att48	64488.3 ± 2368.8	75379.7 ± 2837.9	77748.3 ± 1635.7	70382.5 ± 1934.4
berlin52	13204.8 ± 213.4	15998.1 ± 650.2	15754 ± 421.6	14840.3 ± 252.1
bier127	214992.2 ± 3495	325998.6 ± 5077.2	257287.9 ± 7775.1	230256 ± 3919.4
pr76	207663.5 ± 4143.9	291256 ± 12129.4	251020.7 ± 6726.7	228033.5 ± 4116.9
rat99	2386.7 ± 21.8	3786.3 ± 132.9	2891 ± 41.6	2582 ± 28.4

Çizelge 4.5. Süre açısından (sn.) takas, path-relink operatörlerinin ve 2-opt algoritmasının karşılaştırılması

Örnek	2 satıcı için			
	APSO	Sadece takas operatörü	Sadece path-relink operatörü	Sadece 2-opt algoritması
att48	693	0.1	0.5	2059
berlin52	977	0.1	0.6	2947
bier127	14689	0.1	2.8	44167
pr76	3139	0.1	1.1	9445
rat99	6478	0.1	1.7	19341

Örnek	3 satıcı için			
	APSO	Sadece takas operatörü	Sadece path-relink operatörü	Sadece 2-opt algoritması
att48	319	0.1	0.4	904
berlin52	402	0.1	0.4	1125
bier127	6481	0.2	2.0	18641
pr76	1364	0.1	0.8	4296
rat99	3212	0.1	1.3	9708

Örnek	4 satıcı için			
	APSO	Sadece takas operatörü	Sadece path-relink operatörü	Sadece 2-opt algoritması
att48	238	0.1	0.3	803
berlin52	292	0.1	0.4	895
bier127	4067	0.2	1.6	11712
pr76	815	0.1	0.7	2470
rat99	1765	0.2	1.0	5079

Ayrıca, çalışma süresi açısından da karşılaştırma yapılmıştır ve sonuçlar Çizelge 4.5'de sunulmaktadır. Sadece 2-opt algoritması kullanıldığında algoritmanın çalışması çok uzun sürmektedir. Sadece takas operatörü veya sadece path-relink operatörü kullanıldığında ise algoritma problemi çalışması çok kısa sürede çözmektedir. APSO algoritması bu 3 yöntemi içinde barındırdığı için ve bu yöntemler rasgele seçilerek çalıştırıldığı için APSO algoritmasının çalışma süresi, sadece takas operatörü ve sadece path-relink operatörünün kullanıldığı yöntemin çalışma süresinden uzun olmaktadır. Fakat sadece 2-opt algoritması kullanılan yöntemle karşılaştırıldığında ise APSO algoritmasının çalışma süresinin daha kısa sürdüğü görülmektedir.

Geliştirilen APSO ve HAPSO algoritmaları, literatürdeki güncel çalışmalardan (Latah, 2016) KKO ve GA algoritmaları ile karşılaştırılmıştır ve sonuçlar Çizelge 4.6, Çizelge 4.7 ve Çizelge 4.8'de verilmiştir. Karşılaştırmada 5 farklı simetrik GSP örneği kullanılmıştır: att48, berlin52, bier127, pr76 ve rat99. Geliştirilen APSO ve HAPSO algoritmaların parametreleri şu şekilde ayarlanmıştır: parçacık sayısı 50, maksimum iterasyon sayısı 2000, gezgin satıcı sayısı sırasıyla 2, 3 ve 4 olarak ayarlanmıştır. Satıcı başına düşen gezilecek şehir sayısının üst limit K değeri Denklem (3.5) ile hesaplanmaktadır. Ayrıca, her bir deney bağımsız olarak 20 kez çalıştırılmıştır.

Çizelge 4.6, geliştirilen APSO ve HAPSO yöntemlerin 2 satıcı için 5 ÇGSP örneğinde elde ettiği en iyi, en kötü ve ortalama sonuçları ile birlikte GA ve KKO algoritmaların ortalama sonuçlarını sunmaktadır. Çizelgede görüldüğü gibi HAPSO algoritması APSO, GA ve KKO algoritmalarından daha iyi sonuçlar elde etmiştir. Ayrıca, APSO algoritması ikinci en iyi sonuçları elde etmiştir.

Çizelge 4.6. 2 satıcı için APSO, HAPSO, GA ve KKO algoritmaların karşılaştırılması

Örnek	APSO			HAPSO			GA (Latah, 2016)	KKO (Latah, 2016)
	En iyi	En kötü	Ortalama	En iyi	En kötü	Ortalama		
att48	43424	44929	44121.4	36891	38461	**37690.9**	50725.81	71151.36
berlin52	9482	10244	9872.9	7994	8377	**8268.4**	11066.69	16354.02
bier127	155651	165175	161693.8	125480	128649	**127089.9**	282343.86	425016.29
pr76	143560	151283	148538.8	120490	124412	**123341.7**	184176.1	309514.32
rat99	1662	1724	1698.6	1427	1463	**1442.3**	2487.64	3980.43

Çizelge 4.7, geliştirilen APSO ve HAPSO yöntemlerin 3 satıcı için 5 ÇGSP örneğinde elde ettiği en iyi, en kötü ve ortalama sonuçları ile birlikte GA ve KKO algoritmaların ortalama sonuçlarını sunmaktadır. Çizelgede görüldüğü gibi geliştirilen yöntem GA ve KKO algoritmalarından daha iyi sonuçlar elde etmiştir.

Çizelge 4.7. 3 satıcı için APSO, HAPSO, GA ve KKO algoritmaların karşılaştırılması

Örnek	APSO			HAPSO			GA (Latah, 2016)	KKO (Latah, 2016)
	En iyi	En kötü	Ortalama	En iyi	En kötü	Ortalama		
att48	51510	55974	54393.1	40637	45965	**43845.9**	49709.78	51032.81
berlin52	11149	12089	11583.1	8876	9467	**9180.5**	10898.75	11726.73
bier127	187575	196370	193300.1	129621	143525	**137325.9**	257228.63	349770.9
pr76	174485	183105	179152.1	138096	148349	**143102.8**	170857.76	265550.49
rat99	2018	2087	2048.9	1680	1772	**1734.1**	1970.48	3328.02

Çizelge 4.8, geliştirilen yöntemlerin 4 satıcı için 5 ÇGSP örneğinde elde ettiği en iyi, en kötü ve ortalama sonuçları ile birlikte GA ve KKO algoritmaların ortalama sonuçlarını sunmaktadır. Çizelgede görüldüğü gibi att48 ve berlin52 örneğinde KKO algoritması daha iyi sonuçlar elde etmiştir. Geliştirilen HAPSO yöntemi bier127 ve pr76 örneğinde APSO, GA ve KKO algoritmalarından daha iyi sonuçlar elde etmiştir. rat99 örneğinde GA algoritması daha iyi sonuç elde etmiştir. Ayrıca, HAPSO algoritması berlin52 ve rat99 problemlerinde ise ikinci en iyi sonucu elde etmiştir. Hatta çizelgede görüldüğü gibi rat99 probleminde GA algoritmasının elde ettiği sonuca yakın sonuç elde etmiştir.

Çizelge 4.8. 4 satıcı için APSO, HAPSO, GA ve KKO algoritmaların karşılaştırılması

Örnek	APSO			HAPSO			GA (Latah, 2016)	KKO (Latah, 2016)
	En iyi	En kötü	Ortalama	En iyi	En kötü	Ortalama		
att48	58445	67170	64488.3	50014	55736	52716.6	47083.53	**42169.88**
berlin52	12952	13602	13204.8	9893	10908	10365	11736.74	**8820.24**
bier127	207814	219357	214992.2	141496	150427	**147184.6**	233708.3	294273.77
pr76	201022	214005	207663.5	159964	175331	**168679.15**	168717.69	207333.11
rat99	2353	2428	2386.7	1963	2097	2033.8	**1945.36**	2814.74

APSO ve HAPSO algoritmaları 2, 3, 4, 5, 6, 7, 8 ve 9 satıcı için ÇGSP üzerinde detaylı olarak karşılaştırılmıştır. Elde edilen sonuçlar Çizelge 4.9 ile Çizelge 4.16 arasında sunulmaktadır. Ayrıca, 2 satıcı için bulunan en iyi turların grafiği Şekil 4.4 ve Şekil 4.5'de gösterilmektedir. Diğer satıcılar için bulunan en iyi turların grafikleri ise Şekil EK-1.1 ile Şekil EK-1.14 arasında gösterilmektedir.

Çizelge 4.9. APSO ve HAPSO algoritmaların karşılaştırılması (2 satıcı için)

Örnek	En iyi tur uzunluğu		En kötü tur uzunluğu		Ortalama tur uzunluğu	
	HAPSO	APSO	HAPSO	APSO	HAPSO	APSO
att48	36891	43424	38461	44929	37690.9	44121.4
berlin52	7994	9482	8377	10244	8268.4	9872.9
bier127	125480	155651	128649	165175	127089.9	161693.8
eil51	450	516	469	549	454.5	534.2
eil76	569	670	603	705	585.2	691.4
pr152	113429	124649	118762	129074	116726.8	127367.4
pr226	102304	126101	115924	134277	107925.8	130594.1
pr76	120490	143560	124412	151283	123341.7	148538.8
rat99	1427	1662	1463	1724	1442.3	1698.6

Çizelge 4.10. APSO ve HAPSO algoritmaların karşılaştırılması (3 satıcı için)

Örnek	En iyi tur uzunluğu		En kötü tur uzunluğu		Ortalama tur uzunluğu	
	HAPSO	APSO	HAPSO	APSO	HAPSO	APSO
att48	40637	51510	45965	55974	43845.9	54393.1
berlin52	8876	11149	9467	12089	9180.5	11583.1
bier127	129621	187575	143525	196370	137325.9	193300.1
eil51	484	609	516	636	498.0	624.3
eil76	622	794	662	816	639.8	806.8
pr152	146788	171315	161482	175972	152675.4	173516.6
pr226	132705	173688	137109	179088	134496.9	176881.8
pr76	138096	174485	148349	183105	143102.9	179152.1
rat99	1680	2018	1772	2087	1734.2	2048.9

Çizelge 4.11. APSO ve HAPSO algoritmaların karşılaştırılması (4 satıcı için)

Örnek	En iyi tur uzunluğu		En kötü tur uzunluğu		Ortalama tur uzunluğu	
	HAPSO	APSO	HAPSO	APSO	HAPSO	APSO
att48	50014	58445	55736	67170	52716.6	64488.3
berlin52	9893	12952	10908	13602	10365.0	13204.8
bier127	141496	207814	150427	219357	147184.6	214992.2
eil51	536	677	573	727	556.1	707.2
eil76	658	906	702	933	679.1	919.3
pr152	156660	208638	188036	218834	176101.9	214838.8
pr226	165707	219097	171052	222670	167282.0	221334.2
pr76	159964	201022	175331	214005	168679.2	207663.5
rat99	1963	2353	2097	2428	2033.8	2386.7

Çizelge 4.12. APSO ve HAPSO algoritmaların karşılaştırılması (5 satıcı için)

Örnek	En iyi tur uzunluğu		En kötü tur uzunluğu		Ortalama tur uzunluğu	
	HAPSO	APSO	HAPSO	APSO	HAPSO	APSO
att48	55139	69097	60358	76188	57488.4	72483.0
berlin52	10545	14065	11897	15380	11313.7	14677.8
bier127	146239	229117	158821	240486	154349.7	234666.3
eil51	582	764	626	819	602.4	797.1
eil76	716	977	775	1030	744.8	1006.0
pr152	181092	241667	220718	254621	210967.3	250263.9
pr226	190749	260605	200069	265496	195667.5	262905.0
pr76	182113	225898	203399	237539	193858.3	231572.6
rat99	2264	2681	2395	2813	2346.8	2727.7

Çizelge 4.13. APSO ve HAPSO algoritmaların karşılaştırılması (6 satıcı için)

Örnek	En iyi tur uzunluğu		En kötü tur uzunluğu		Ortalama tur uzunluğu	
	HAPSO	APSO	HAPSO	APSO	HAPSO	APSO
att48	60547	72229	68347	84944	63808.6	79938.4
berlin52	11230	15278	13205	16352	12414.3	15797.6
bier127	154088	241215	171153	252894	164977.0	248979.6
eil51	631	844	692	898	664.1	870.9
eil76	809	1097	850	1132	827.5	1114.6
pr152	206333	269630	250498	288312	236613.5	281791.3
pr226	228418	296928	237867	302612	234034.2	300607.4
pr76	200021	257231	221283	269830	213813.2	262043.1
rat99	2573	3078	2768	3152	2700.1	3120.7

Çizelge 4.14. APSO ve HAPSO algoritmaların karşılaştırılması (7 satıcı için)

Örnek	En iyi tur uzunluğu		En kötü tur uzunluğu		Ortalama tur uzunluğu	
	HAPSO	APSO	HAPSO	APSO	HAPSO	APSO
att48	67519	79742	75753	91033	71824.6	86794.4
berlin52	13167	16374	14169	17728	13621.1	17027.4
bier127	164377	258592	182233	270495	172861.8	264475.5
eil51	682	919	756	992	721.7	946.8
eil76	884	1174	941	1244	914.1	1204.2
pr152	241573	307510	273502	319506	261893.5	314404.2
pr226	259963	338112	269078	341520	265264.1	339947.2
pr76	233313	280153	247344	291971	241919.6	287922.0
rat99	2936	3373	3137	3509	3063.8	3451.9

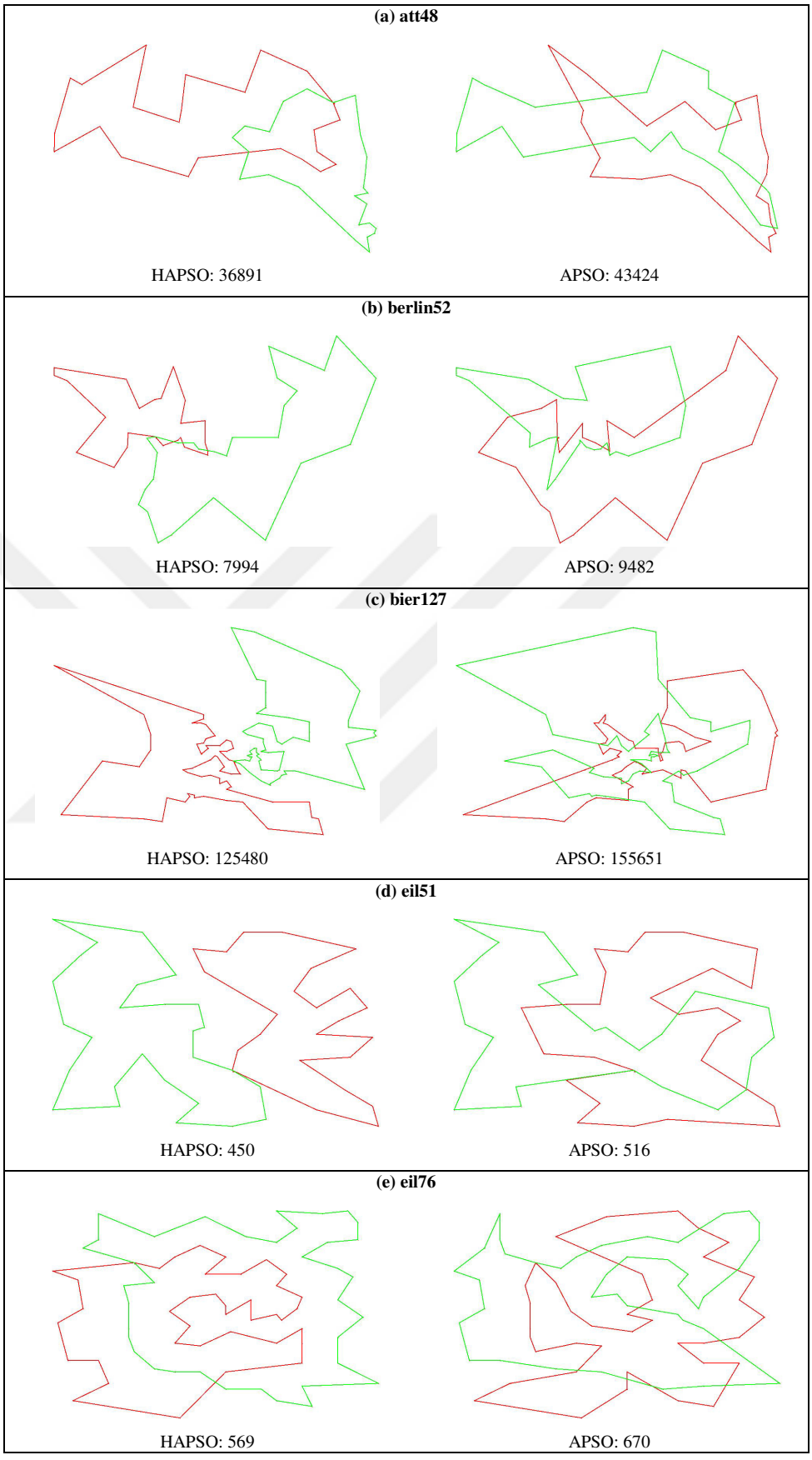
Çizelge 4.15. APSO ve HAPSO algoritmaların karşılaştırılması (8 satıcı için)

Örnek	En iyi tur uzunluğu		En kötü tur uzunluğu		Ortalama tur uzunluğu	
	HAPSO	APSO	HAPSO	APSO	HAPSO	APSO
att48	74746	87789	82250	96682	77616.8	92289.3
berlin52	13643	16765	15145	18506	14403.6	17921.4
bier127	171826	270267	187635	288717	180159.1	276965.3
eil51	740	955	808	1047	778.7	1007.2
eil76	962	1259	1024	1330	991.4	1304.8
pr152	278932	334154	294558	349763	286266.6	341335.3
pr226	290206	371567	297839	379298	293523.6	375553.6
pr76	254719	299817	279275	319296	267279.4	311342.9
rat99	3257	3772	3468	3860	3385.2	3822.8

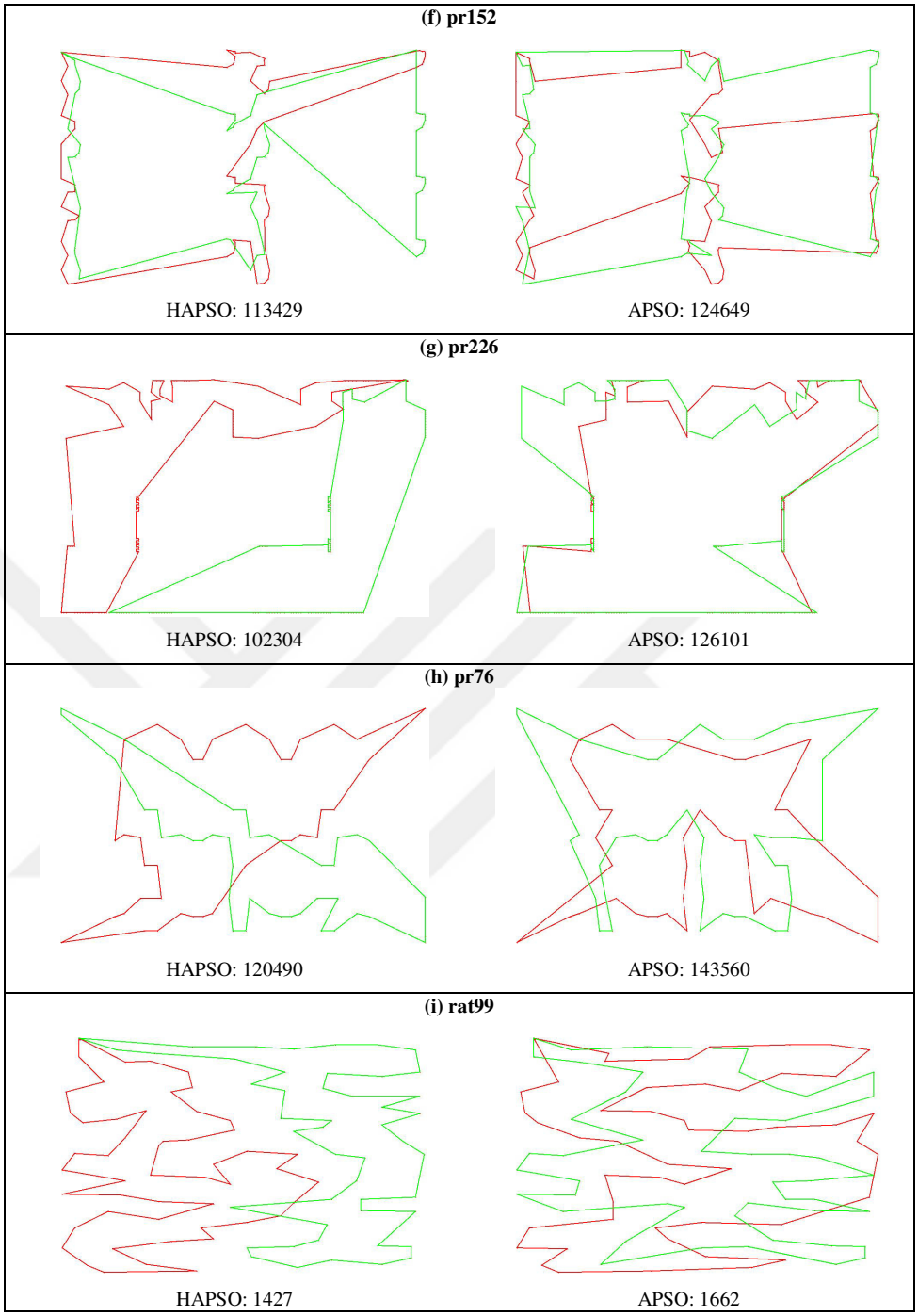
Çizelge 4.16. APSO ve HAPSO algoritmaların karşılaştırılması (9 satıcı için)

Örnek	En iyi tur uzunluğu		En kötü tur uzunluğu		Ortalama tur uzunluğu	
	HAPSO	APSO	HAPSO	APSO	HAPSO	APSO
berlin52	14573	17679	16170	19543	15471.0	18692.6
bier127	186848	283781	201255	303074	193204.5	292402.4
eil51	814	1004	874	1088	840.3	1056.9
eil76	1021	1352	1095	1411	1068.8	1387.4
pr152	291007	352351	319194	379269	309318.8	365944.6
pr226	321917	407526	336727	417083	328694.7	412054.8
pr76	278845	333939	301064	345129	291429.7	339039.0
rat99	3493	4087	3736	4227	3614.6	4172.1

HAPSO ve APSO algoritmalarının 2 satıcı için bulduğu en iyi turların grafiği Şekil 4.4 ve Şekil 4.5'de gösterilmektedir. Diğer satıcılar (3, 4, 5, 6, 7, 8, 9) için bulunan en iyi turların grafikleri ise Şekil EK-1.1 ile Şekil EK-1.14 arasında gösterilmektedir. Grafiklerin daha anlaşılabilir olması için her bir satıcının oluşturduğu tur farklı bir renk ile gösterilmiştir. Her bir satıcı aynı depo şehrinden tura başlamaktadır ve turunu tamamladıktan sonra tekrar depo şehrine geri dönmektedir.



Şekil 4.4. att48, berlin52, bier127, eil51 ve eil76 problemleri üzerinde 2 satıcı için HAPSO ve APSO ile bulunan en iyi turlar



Şekil 4.5. pr152, pr226, pr76 ve rat99 problemleri üzerinde 2 satıcı için HAPSO ve APSO ile bulunan en iyi turlar

APSO ve HAPSO algoritmaları çalışma süreleri (saniye açısından) karşılaştırılmış ve elde edilen sonuçlar Çizelge 4.17'de sunulmaktadır. Çizelgeden de görüldüğü gibi APSO ve HAPSO algoritmalarının çalışma süreleri birbirine yakın çıkmaktadır.

Çizelge 4.17. Süre açısından (sn.) HAPSO ve APSO algoritmaların karşılaştırılması

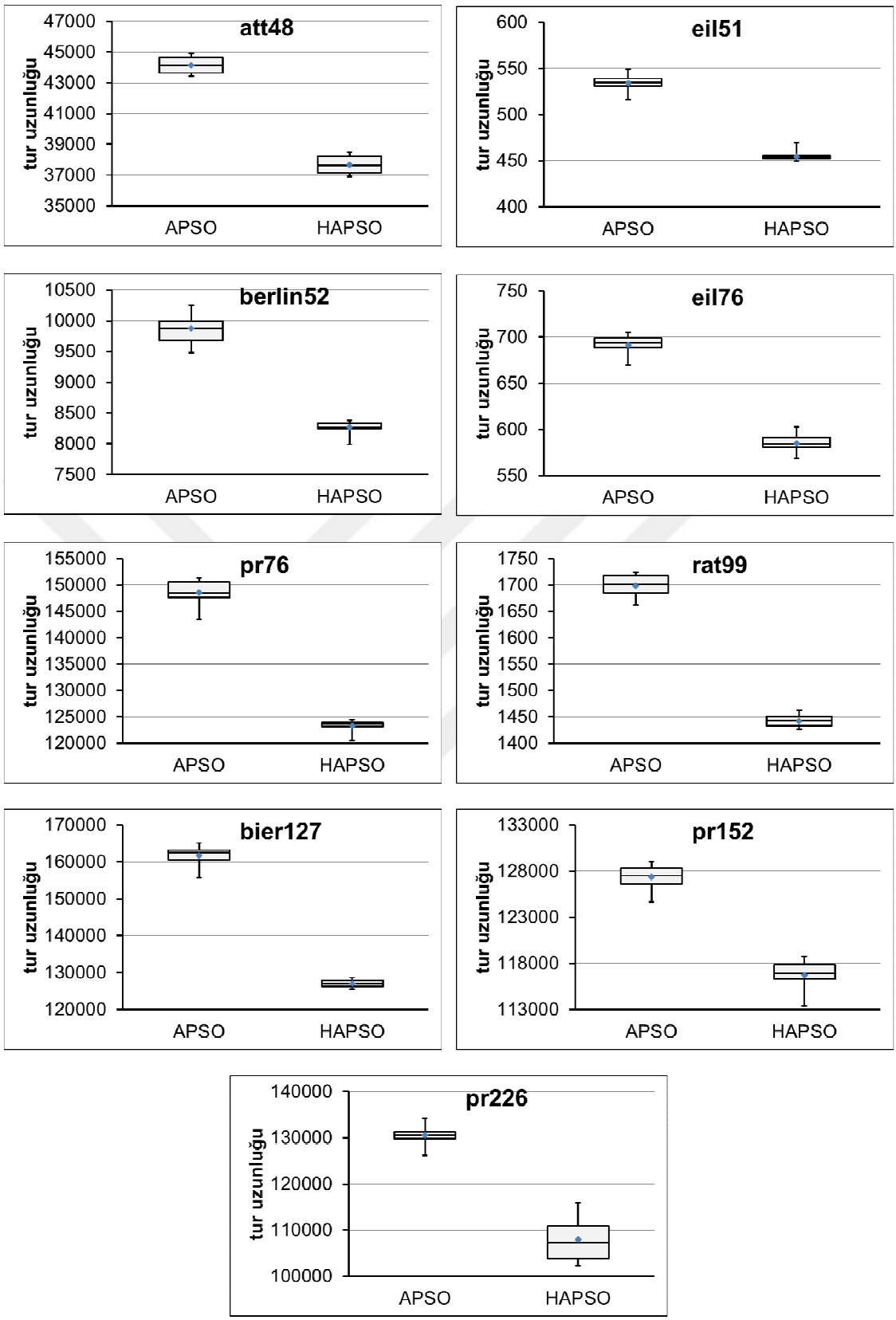
Örnek	2 satıcı için		3 satıcı için		4 satıcı için	
	HAPSO	APSO	HAPSO	APSO	HAPSO	APSO
att48	711	693	320	319	231	238
berlin52	945	977	404	402	284	292
bier127	14830	14689	6435	6481	4056	4067
eil51	876	893	377	378	269	274
eil76	3126	3174	1223	1364	812	823
pr152	27376	26073	10658	10386	6922	6956
pr226	89204	86205	39234	35194	23599	23931
pr76	3129	3139	1221	1364	813	815
rat99	6580	6478	3016	3212	1722	1765

Örnek	5 satıcı için		6 satıcı için		7 satıcı için	
	HAPSO	APSO	HAPSO	APSO	HAPSO	APSO
att48	164	169	120	123	93	96
berlin52	199	206	148	154	121	124
bier127	2375	2419	1657	1700	1267	1291
eil51	192	198	142	146	117	120
eil76	562	569	405	420	313	321
pr152	4425	4529	2846	2935	2067	2135
pr226	15221	15685	10020	10252	7476	7587
pr76	555	568	402	412	312	318
rat99	1133	1155	826	845	645	660

Örnek	8 satıcı için		9 satıcı için	
	HAPSO	APSO	HAPSO	APSO
att48	77	79		
berlin52	97	100	78	77
bier127	1000	1014	822	817
eil51	93	96	73	74
eil76	263	267	205	205
pr152	1645	1650	1329	1342
pr226	5710	5843	4803	4822
pr76	260	268	206	206
rat99	520	530	405	406

Algoritmaları kıyaslamak için kullanılan bir diğer yöntem ise kutu grafiğidir. Kutu grafiği sayesinde algoritmaların sonuçları ile ilgili istatistiksel bilgiler görselleştirilmektedir ve böylece bu bilgilerin yorumlanması kolaylaşmaktadır (Acılar, 2013). Kutu grafiği; elde edilen sonuçların kalitesi hakkında birçok bilgiyi bir bakışta sunmak için kullanılan bir grafik türüdür. Bu yöntem karşılaştırılacak verilerin kalitesini ve dağılımını belirlemek için kullanılır (Scheck, 2008). Kutu grafiği; sonuçların en küçük değerini, birinci çeyrek değerini, ortanca değerini, ortalama değerini, üçüncü çeyrek değerini ve en büyük değerini sunmaktadır. Bundan dolayı, grafik karşılaştırılacak algoritmaların sonuçlarının kapsamını en küçük değerden en büyük değere kadar göstermektedir. Böylece, kutu grafiği algoritmanın istikrarı ve gürbüzlüğü hakkında bilgi vermektedir (Gülcü, 2017). Şekil 4.6 APSO ve HAPSO algoritmalarının 2 satıcı için ÇGSP örnekleri üzerinde elde ettiği sonuçların kutu grafiğini sunmaktadır. Diğer satıcılar

için kutu grafikleri Şekil EK-2.1 ile Şekil EK-2.7 arasında sunulmaktadır. Kutu grafikleri incelendiğinde ÇGSP örneklerinin hepsinde HAPSO algoritması APSO algoritmasından daha istikrarlı sonuçlar üretmektedir ve performansı daha iyidir. Bundan dolayı HAPSO algoritması APSO algoritmasından daha gürbüzdür.



Şekil 4.6. 2 satıcı için ÇGSP örneklerinin kutu grafiği

5. SONUÇLAR VE ÖNERİLER

5.1 Sonuçlar

Günümüzde, biyolojik yapılardan esinlenerek oluşturulan sistemler önem kazanmakta ve araştırmacıların ilgisini çekmektedir. Doğadaki bazı sosyal sistemler, sınırlı yetenekli basit bireyler tarafından oluşturulmalarına rağmen kolektif zekâ davranışları sergilemektedirler. Bu bireylerin kendi içerisindeki organizasyonları ve dolaylı iletişimleri, mühendislikte ve günlük yaşantıda karşılaşılan optimizasyon problemlerin çözümünde kullanılmaktadır. Ayrıca, yapay zekâ sistemlerinin gelişmesine de katkı sunmaktadırlar.

GSP, üzerinde en çok çalışılan optimizasyon problemlerinden biridir. GSP'nin tanımı şu şekilde verilebilir: Aralarındaki uzaklıkları bilinen şehirlerden bir kez geçilmek şartı ile tüm noktaların en az maliyetle dolaşılıp, başlangıç şehrine tekrar dönülmesidir. Literatürde gezgin satıcı probleminin çok sayıda çeşidi ve genellemeleri vardır. Bunlardan biri ÇGSP'dir.

Optimizasyon problemlerinin çözümü için iki yöntem kullanılır: kesin yöntemler ve meta sezgisel yöntemler. Kesin yöntemler en iyi çözümü bulurlar. Fakat çok büyük problemlerin kesin yöntemler ile çözümü çok uzun zaman aldığı için bu yöntemler pek kullanılamazlar. Diğer taraftan meta sezgisel teknikler, problemin en iyi çözümü veya en iyi çözüme yakın uygun bir çözümü kabul edilebilir bir sürede bulmaktadırlar.

Optimizasyon problemlerinin çözümü için çeşitli meta sezgisel teknikler geliştirilmiştir ve yeni teknikler de önerilmektedir. Bu tezde, PSO, GRASP ve 2-opt sezgisel algoritmaları ile ÇGSP'nin çözümü amaçlanmıştır. Bunun için 2 tane algoritma, APSO ve HAPSO, geliştirilmiştir.

APSO algoritması ÇGSP'yi çözmek için PSO, 2-opt algoritmalarını ve path-relink, takas operatörlerini kullanmaktadır. APSO algoritmasının başlangıç popülasyonu rasgele üretilmektedir. Her bir parçacık ÇGSP'nin bir çözümünü temsil etmektedir ve her bir satıcı için bir alttur oluşturmaktadır. 2-opt algoritması, path-relink ve takas operatörleri ise çözümleri iyileştirmektedirler. 2 opt algoritması, parçacığın bir altturundaki iki kenarı siler. Bu durumda, alttur iki parçaya ayrılır. Daha sonra, bu iki parça yeni bir alttur oluşturmak için farklı olasılıklar denenerek yeniden birleştirilir. Path-relink operatörü ise, iki çözüm (kaynak ve hedef) arasında bir çözüm zinciri oluşturur. Bu çalışmada kaynak çözüm parçacığın oluşturduğu turdur. Hedef çözüm, parçacığın *pBest*

bilgisidir. Ara çözümler, kaynak çözüm ile hedef çözüm arasında üretilir ve bu ara çözümlerin kalitesi değerlendirilir. Takas operatörü, bir parçacık içinde rastgele seçilen 2 altturun rastgele seçilmiş birer şehrini yerdeğiştirme işlemidir.

HAPSO algoritması ÇGSP'yi çözmek için GRASP, PSO, 2-opt algoritmalarını ve path-relink, takas operatörlerini kullanmaktadır. HAPSO algoritması APSO algoritmasının gelişmiş bir versiyonudur. APSO algoritmasında başlangıç popülasyonu rasgele üretilmektedir. Fakat HAPSO algoritmasında başlangıç popülasyonu GRASP algoritması ile üretilmektedir.

PSO algoritması sürekli optimizasyon problemlerinin çözümü için geliştirildiğinden dolayı, PSO algoritmasının saf hali kesikli optimizasyon problemlerinin çözümünde doğrudan kullanılamaz. Bu tez çalışmasının literatüre katkısı, PSO tabanlı olan APSO ve HAPSO algoritmalarının kesikli bir optimizasyon problemi olan ÇGSP'yi çözme yeteneğine sahip olmalarıdır.

ÇGSP'de amaç, tura depodan başlayan ve turu depoda bitiren *m* adet satıcı için her bir ara şehir bir kere ziyaret edilmek şartıyla gezilen bütün şehirlerin toplam tur maliyetini minimize etmektir. Literatürde standart bir ÇGSP veri seti bulunmamaktadır. ÇGSP veri setinin bulunmaması ÇGSP'yi çözen algoritmaların performansını test etmede büyük bir eksikliktir. Bundan dolayı, bu tez çalışmasında, standart bir GSP kütüphanesi olan TSPLIB kütüphanesindeki simetrik GSP veri seti kullanılmıştır.

Deneysel çalışmalarda, APSO ve HAPSO algoritmaları, literatürdeki güncel çalışmalardan KKO ve GA algoritmaları ile karşılaştırılmıştır. Karşılaştırmada 5 farklı simetrik GSP örneğini kullandık: att48, berlin52, bier127, pr76 ve rat99. Gezgin satıcı sayısı sırasıyla 2, 3 ve 4 olarak ayarlanmıştır. Ayrıca, her bir deney bağımsız olarak 20 kez çalıştırılmıştır.

2 satıcı için, HAPSO algoritması APSO, GA ve KKO algoritmalarından daha iyi sonuçlar elde etmiştir. Ayrıca, APSO algoritması ikinci en iyi sonuçları elde etmiştir.

3 satıcı için, HAPSO algoritması APSO, GA ve KKO algoritmalarından daha iyi sonuçlar elde etmiştir.

4 satıcı için, HAPSO algoritması bier127 ve pr76 örneğinde APSO, GA ve KKO algoritmalarından daha iyi sonuçlar elde etmiştir. Ayrıca, berlin52 ve rat99 problemlerinde ise ikinci en iyi sonucu elde etmiştir. att48 ve berlin52 örneğinde KKO algoritması, rat99 örneğinde ise GA algoritması daha iyi sonuçlar elde etmiştir.

Deney sonuçlarına göre, HAPSO algoritması birçok problemde diğer algoritmalardan daha iyi sonuçlar elde etmiştir. Bu karşılaştırma analizine göre, önerilen

HAPSO meta-sezgisel algoritması literatürdeki GA ve KKO meta-sezgisel algoritmalarına göre rekabetçi bir avantaj göstermiştir.

HAPSO algoritması ve APSO algoritması tur grafikleri, çalışma süreleri ve kutu grafikleri açısından 9 GSP örneğinde 2, 3, 4, 5, 6, 7, 8 ve 9 satıcı için detaylı olarak karşılaştırılmıştır. Bu 9 GSP örneği şunlardır: att48, berlin52, bier127, eil51, eil76, pr152, pr226, pr76 ve rat99. Tur grafikleri incelendiğinde HAPSO algoritması ile oluşturulan turların uzunlukları daha kısadır. Bunun sebebi, gezgin satıcılar daha makul turlar oluşturmaktadır. Çalışma süreleri açısından incelendiğinde, iki algoritmanın da çalışma süreleri birbirine yakındır.

Algoritmaları kıyaslamak için kullanılan bir diğer yöntem ise kutu grafiğidir. Kutu grafiği sayesinde algoritmaların sonuçları ile ilgili istatistiksel bilgiler görselleştirilmektedir ve böylece bu bilgilerin yorumlanması kolaylaşmaktadır. Kutu grafiği; elde edilen sonuçların kalitesi hakkında birçok bilgiyi bir bakışta sunmak için harikadır. Bu yöntem karşılaştırılacak verilerin kalitesini ve dağılımını belirlemek için kullanılır. Ayrıca, kutu grafiği algoritmanın istikrarı ve gürbüzlüğü hakkında bilgi vermektedir. Kutu grafikleri incelendiğinde, HAPSO algoritmasının daha istikrarlı ve daha gürbüz olduğu görülmektedir.

Genel olarak HAPSO ve APSO algoritmalarının deney sonuçlarını incelediğimizde, bu 2 algoritma da ÇGSP'nin çözümünde başarılıdır. Fakat HAPSO algoritmasının başlangıç çözümü GRASP algoritması ile üretildiği için, HAPSO algoritması APSO algoritmasından daha avantajlı duruma gelmektedir. Bunun sonucu olarak HAPSO algoritması daha iyi sonuçlar üretmektedir.

Sonuç olarak, kesikli optimizasyon problem olan ÇGSP'nin çözümü için PSO, GRASP ve 2-opt sezgisel algoritmaları ile 2 yeni algoritma geliştirilmiştir.

5.2 Öneriler

Geliştirilen APSO ve HAPSO algoritmalarının performansını etkileyen bazı faktörler bulunmaktadır. Bunlar, kullanılan parçacık sayısı, maksimum iterasyon sayısı ve GRASP algoritmasında kullanılan eşik değeri vb. parametre değerleridir. Gelecek çalışma olarak bu parametre değerlerinin en uygun değerlerini bulmak için çalışma yapılabilir.

APSO ve HAPSO algoritmalarında lokal arama algoritması olarak 2-opt algoritması kullanılmıştır. Literatürde birkaç tane daha lokal arama algoritması

bulunmaktadır. 2-opt algoritması yerine başka bir lokal arama algoritması kullanılabilir ve performansı incelenebilir.

Geliştirilen APSO ve HAPSO algoritmaları asimetrik GSP örnekleri ve genelleştirilmiş GSP örnekleri gibi farklı problem türlerine de uygulanabilir.

Paralel metasezgisel günümüzde popüler bir konudur. Bu konuda yeni araştırmalar, çalışmalar yapılmaktadır. APSO ve HAPSO algoritmalarının paralel versiyonları geliştirilebilir.

KAYNAKLAR

Acılar A. M., 2013. The fuzzy system designing using artificial immune system algorithms, Ph.D. Thesis, *Selçuk University, the graduate school of natural and applied science*, Konya, Turkey.

Alba E., 2005. Parallel metaheuristics: a new class of algorithms, *John Wiley & Sons*, Hoboken, New Jersey.

AminShokravi A., Eskandar H., Derakhsh A. M., Rad H. N., Ghanadi A., 2018. The potential application of particle swarm optimization algorithm for forecasting the air-overpressure induced by mine blasting, *Engineering with Computers*, 34 (2), 277-285.

Baykasoğlu A., Özbel B. K., 2016, Multiple traveling salesman game for cost allocation: a case problem for school bus services, *14. International Logistics and Supply Chain Congress, LM-SCM 2016*, İzmir, Turkey, 64.

Bektaş T., 2006, The multiple traveling salesman problem: an overview of formulations and solution procedures, *Omega*, 34 (3), 209-219.

Bektaş T., 2012, Formulations and Benders decomposition algorithms for multidepot salesmen problems with load balancing, *European Journal of Operational Research*, 216 (1), 83-93.

Bozorg-Haddad O., Solgi M., Loáiciga H. A., 2017, Meta-heuristic and evolutionary algorithms for engineering optimization, *John Wiley & Sons*.

Carter A. E., 2003, Design and application of genetic algorithms for the multiple traveling salesperson assignment problem, Ph.D. Thesis, *Virginia Polytechnic Institute and State University*, Blacksburg, Virginia.

Carter A. E., Ragsdale C. T., 2002, Scheduling pre-printed newspaper advertising inserts using genetic algorithms, *Omega*, 30 (6), 415-421.

Croes G. A., 1958, A method for solving traveling-salesman problems, *Operations research*, 6 (6), 791-812.

Dang J-w., Wang Y-p., Zhao S-x., 2007, Study on a novel genetic algorithm for the combinatorial optimization problem, *International Conference on Control, Automation and Systems*, *ICCAS'07*, Seoul, South Korea, 2538-2541.

Donald D., 2011, Traveling salesman problem, theory and applications, *InTech*, Rijeka.

Dorigo M., Gambardella L. M., 1997, Ant colony system: a cooperative learning approach to the traveling salesman problem, *IEEE Transactions on Evolutionary Computation*, 1 (1), 53-66.

Dorigo M., Stützle T., 2004, Ant Colony Optimization, *Bradford Book*.

Feo T. A., Resende M. G., 1989, A probabilistic heuristic for a computationally difficult set covering problem, *Operations research letters*, 8 (2), 67-71.

Feo T. A., Resende M. G., 1995, Greedy randomized adaptive search procedures, *Journal of Global Optimization*, 6, 109-133.

Gendreau M., Potvin J-Y., 2010, Handbook of metaheuristics, *Springer*.

Glover F., 1963, Parametric combinations of local job shop rule, *ONR Research Memorandum*, Pittsburgh, PA.

Gorenstein S., 1970, Printing press scheduling for multi-edition periodicals, *Management Science*, 16 (6), 373-383.

Gutin G., Punnen A. P., 2006, The traveling salesman problem and its variations, *Springer Science & Business Media*.

Gülcü Ş,, 2017, Parallelization of the particle swarm and ant colony optimization algorithms by using the greedy information swap strategy, Ph.D. Thesis, *Selçuk University, the graduate school of natural and applied science*, Konya.

Hajihassani M., Armaghani D. J., Kalatehjari R., 2018, Applications of particle swarm optimization in geotechnical engineering: a comprehensive review, *Geotechnical and Geological Engineering*, 36 (2), 705-722.

Holland J. H., 1992, Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence, *MIT press*.

Hou M., Liu D., 2012, A novel method for solving the multiple traveling salesmen problem with multiple depots, *Chinese science bulletin*, 57 (15), 1886-1892.

Johnson D. S., McGeoch L. A., 1997, The traveling salesman problem: A case study in local optimization, *Local search in combinatorial optimization*, 1, 215-310.

Junjie P., Dingwei W., 2006, An ant colony optimization algorithm for multiple travelling salesman problem, *1. International Conference on Innovative Computing, Information and Control, ICICIC'06*, 210-213.

Kalaycı T., 2006, Yapay zeka teknikleri kullanan üç boyutlu grafik yazılımları için extensible 3d (x3d) ile bir altyapı oluşturulması ve gerçekleştirimi, Master Thesis, *Ege Üniversitesi Fen Bilimler Enstitüsü*, İzmir, Turkey.

Karaboga D., Basturk B., 2007, A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm, *Journal of global optimization*, 39 (3), 459-471.

Kennedy J., Eberhart R., 1995, Particle Swarm Optimization, *IEEE International Conference on Neural Networks*, Perth, WA, Australia, 1942-1948.

Király A., Abonyi J., 2015, Redesign of the supply of mobile mechanics based on a novel genetic optimization algorithm using Google Maps API, *Engineering Applications of Artificial Intelligence*, 38, 122-130.

Latah M., 2016, Solving multiple TSP problem by K-means and crossover based modified ACO algorithm, *International Journal of Engineering Research & Technology*, 5 (2), 430-434.

Lee J. H., Song J-Y., Kim D-W., Kim J-W., Kim Y-J., Jung S-Y., 2018, Particle swarm optimization algorithm with intelligent particle number control for optimal design of electric machines, *IEEE Transactions on Industrial Electronics*, 65 (2), 1791-1798.

Malik W., Rathinam S., Darbha S., 2007, An approximation algorithm for a symmetric generalized multiple depot, multiple travelling salesman problem, *Operations Research Letters*, 35 (6), 747-753.

Mankowska D. S., Meisel F., Bierwirth C., 2014, The home health care routing and scheduling problem with interdependent services, *Health care management science*, 17 (1), 15-30.

Mitrović-Minić S., Krishnamurti R., 2006, The multiple TSP with time windows: vehicle bounds based on precedence graphs, *Operations Research Letters*, 34 (1), 111-120.

Nabiyev V. V., 2012, Yapay zeka: insan-bilgisayar etkileşimi, *Seçkin Yayıncılık*, Ankara.

Necula R., Breaban M., Raschip M., 2015a, Performance evaluation of ant colony systems for the single-depot multiple traveling salesman problem, *International Conference on Hybrid Artificial Intelligence Systems*, Bilbao, Spain, 257-268.

Necula R., Breaban M., Raschip M., 2015b, Tackling the bi-criteria facet of multiple traveling salesman problem with ant colony systems, *IEEE 27th International Conference on Tools with Artificial Intelligence (ICTAI)*, Vietri sul Mare, Italy, 873-880.

Necula R., Raschip M., Breaban M., 2018, Balancing the Subtours for Multiple TSP Approached with ACS: Clustering-Based Approaches Vs, MinMax Formulation, *EVOLVE-A Bridge between Probability, Set Oriented Numerics, and Evolutionary Computation VI*, 210-223.

Nouiri M., Bekrar A., Jemai A., Niar S., Ammari A. C., 2018, An effective and distributed particle swarm optimization algorithm for flexible job-shop scheduling problem, *Journal of Intelligent Manufacturing*, 29 (3), 603-615.

Oberlin P., Rathinam S., Darbha S., 2009, A transformation for a multiple depot, multiple traveling salesman problem, *American Control Conference, ACC'09*, 2636-2641.

Ponraj R., Amalanathan G., 2014, Optimizing multiple travelling salesman problem considering the road capacity, *Journal of computer science*, 10 (4), 680.

Reinelt G., 1991, TSPLIB—A traveling salesman problem library, *ORSA journal on computing*, 3 (4), 376-384.

TSP library, 1995, http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/, [Ziyaret tarihi: 20.12.2016].

Resende M. G. C., Ribeiro C. C., Glover F., Martí R., 2010, Scatter Search and Path-Relinking: Fundamentals, Advances, and Applications, Handbook of Metaheuristics, *Springer US*, Boston, 87-107.

Scheck R., 2008, Create Dynamic Charts in Microsoft Office Excel 2007 and Beyond, *Microsoft Press*, Washington.

Shabanpour M., Yadollahi M., Hasani M. M., 2017, A New Method to Solve the Multi Traveling Salesman Problem with the Combination of Genetic Algorithm and Clustering, *International Journal of Computer Science and Network Security*, 17 (5), 221.

Shah-Hosseini H., 2009, The intelligent water drops algorithm: a nature-inspired swarm-based optimization algorithm, *International Journal of Bio-Inspired Computation*, 1 (1), 71-79.

Sofge D., Schultz A., De Jong K., 2002, Evolutionary computational approaches to solving the multiple traveling salesman problem using a neighborhood attractor schema, *Workshops on Applications of Evolutionary Computation*, Kinsale, Ireland, 153-162.

Talbi E-G., 2009, Metaheuristics: from design to implementation, *John Wiley & Sons*.

Wex F., Schryen G., Feuerriegel S., Neumann D., 2014, Emergency response in natural disaster management: Allocation and scheduling of rescue units, *European Journal of Operational Research*, 235 (3), 697-708.

Yousefikhoshbakht M., Didehvar F., Rahmati F., 2013, Modification of the ant colony optimization for solving the multiple traveling salesman problem, *Romanian Journal of Information Science and Technology*, 16 (1), 65-80.

Yu Q., Wang D., Lin D., Li Y., Wu C., 2012, A novel two-level hybrid algorithm for multiple traveling salesman problems, *International Conference in Swarm Intelligence*, Shenzhen, China, 497-503.

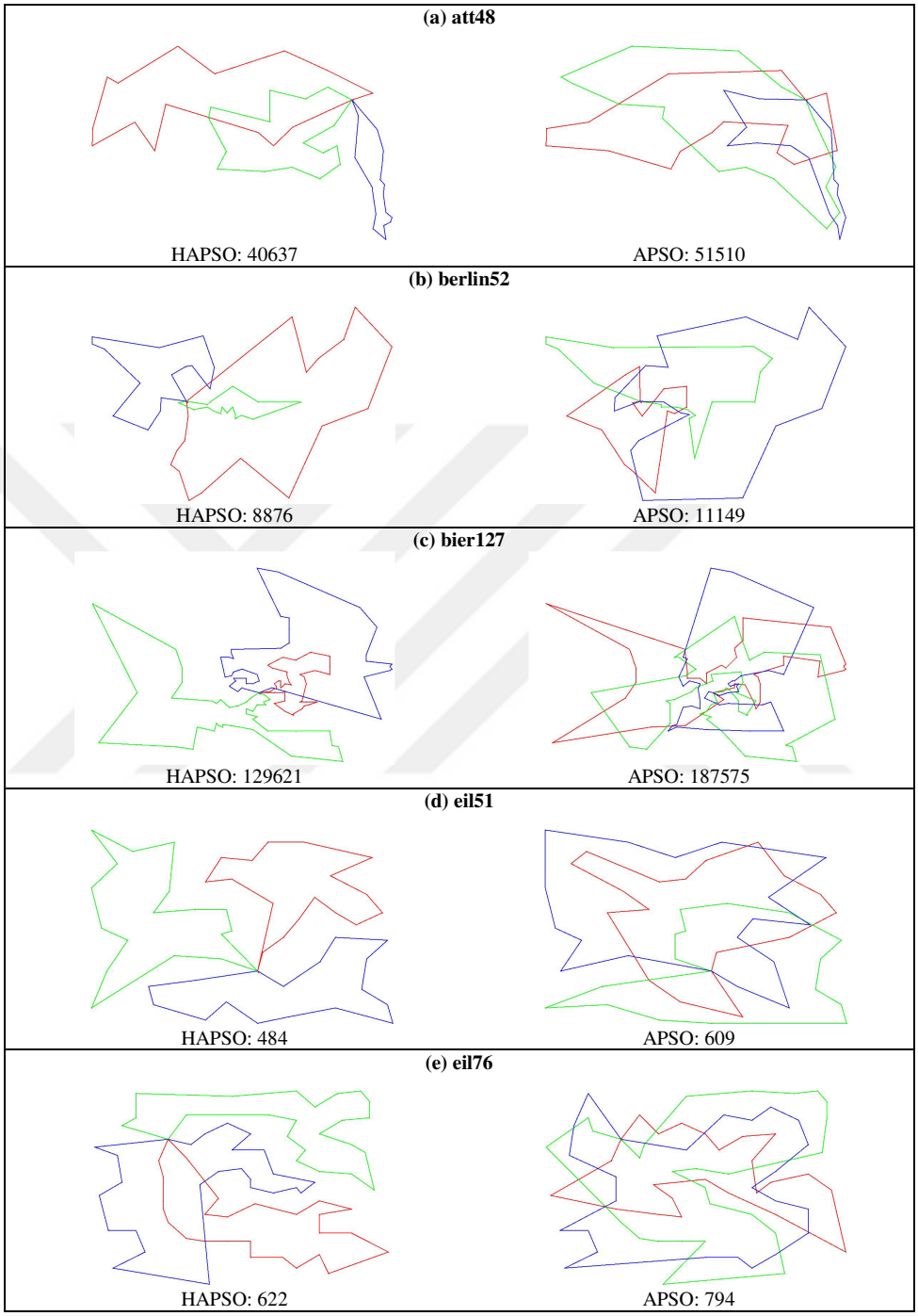
Yuan S., Skinner B., Huang S., Liu D., 2013, A new crossover approach for solving the multiple travelling salesmen problem using genetic algorithms, *European Journal of Operational Research*, 228 (1), 72-82.

Zhang T., Gruver W., Smith M. H., 1999, Team scheduling by genetic search, *Proceedings of the Second International Conference on Intelligent Processing and Manufacturing of Materials, IPMM'99*, Honolulu, USA, 839-844.

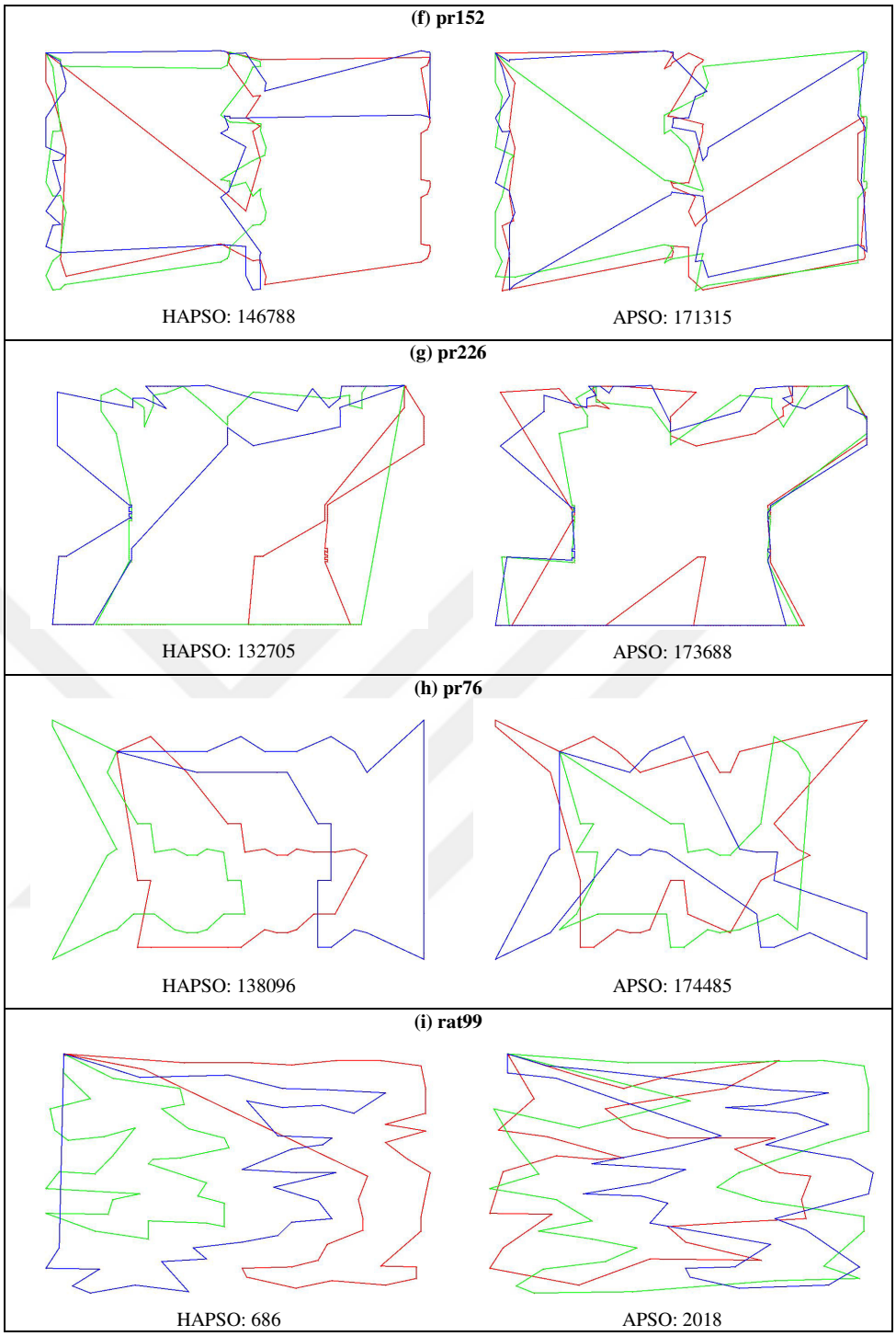
Zhou H., Song M., Pedrycz W., 2018, A comparative study of improved GA and PSO in solving multiple traveling salesmen problem, *Applied Soft Computing*, 64, 564-580.

EKLER

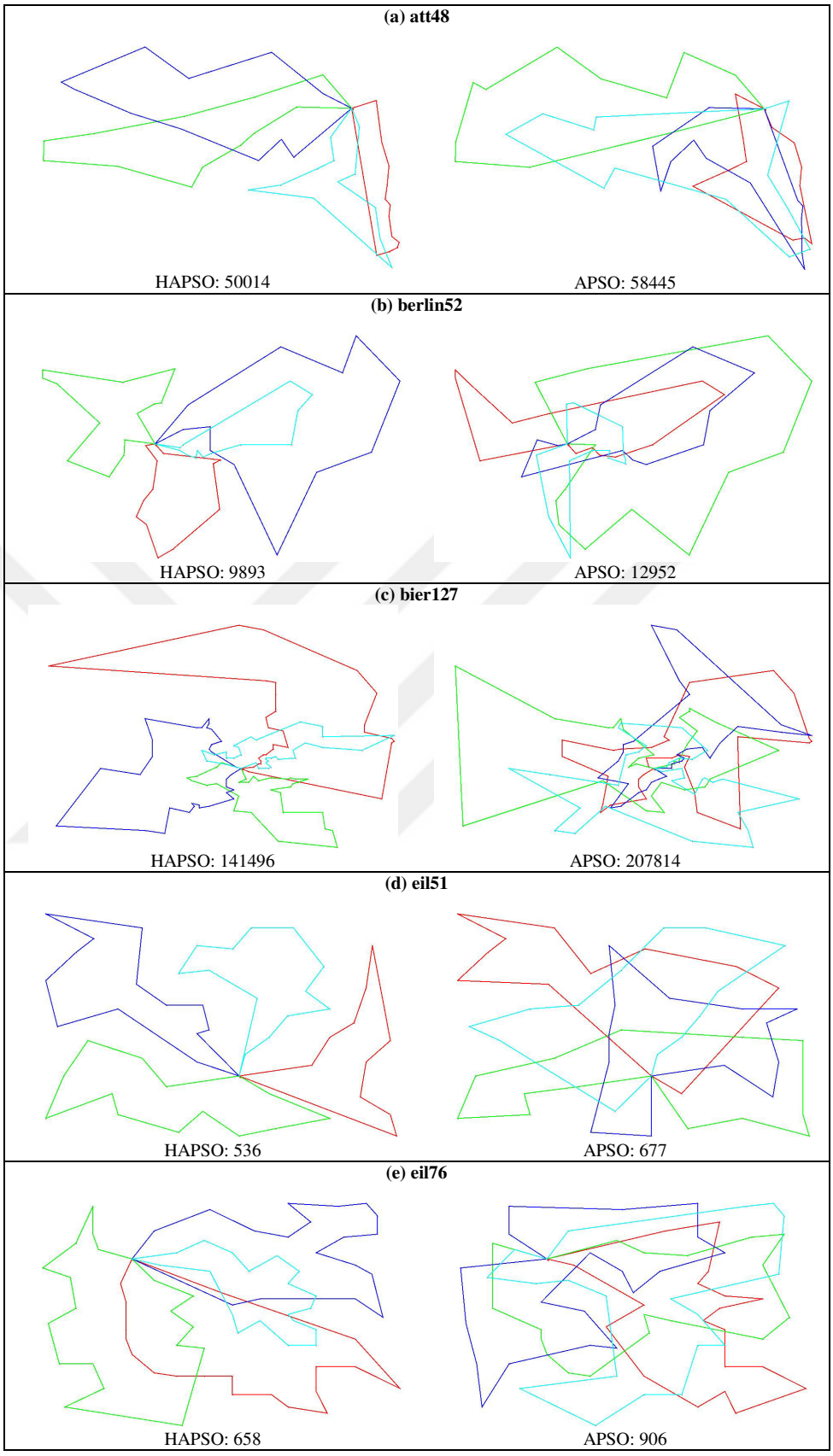
EK-1 HAPSO ve APSO ile bulunan en iyi turlara ait şekil ve çizelgeler.



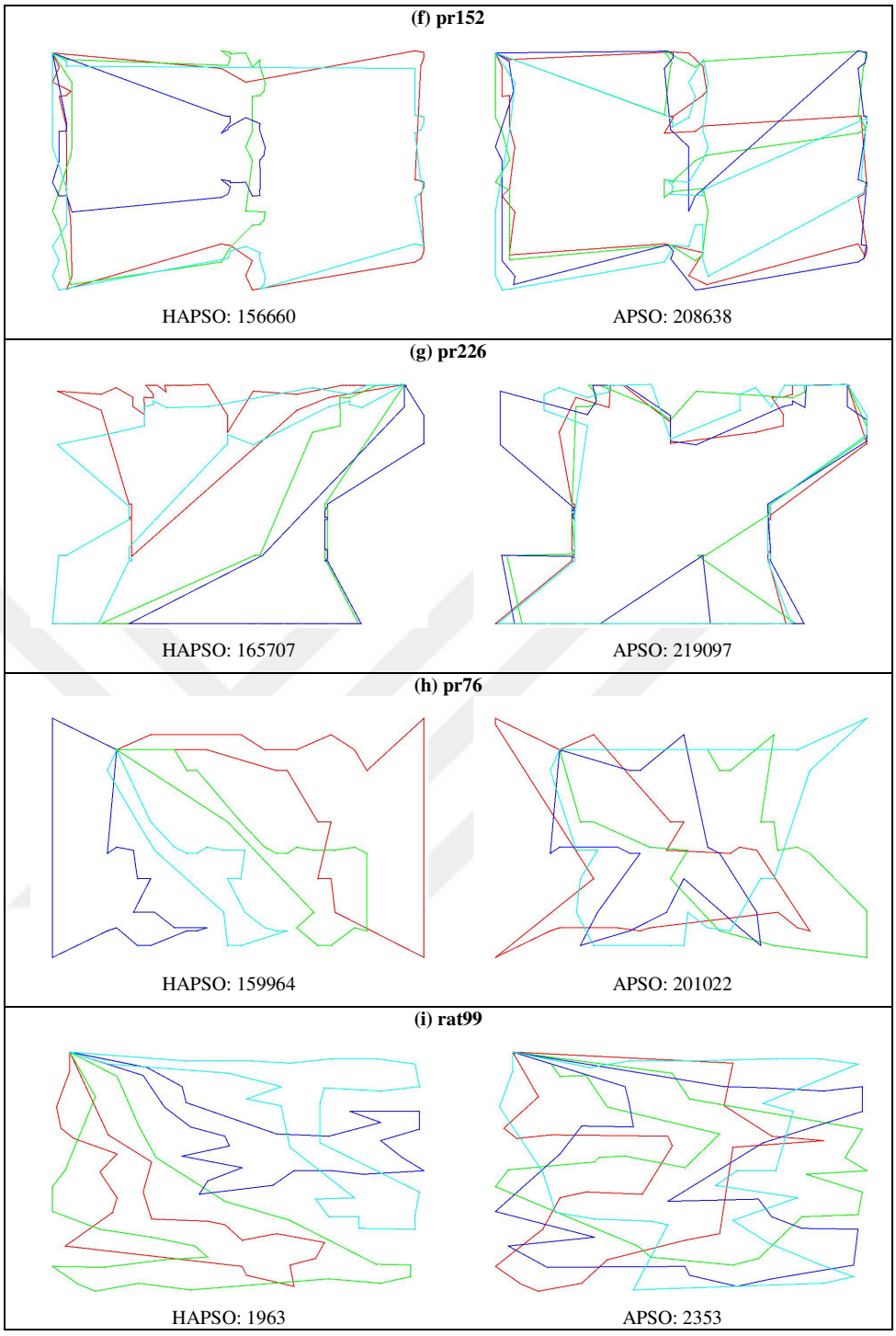
Şekil EK-1.1. 3 satıcı için HAPSO ve APSO ile bulunan en iyi turlar



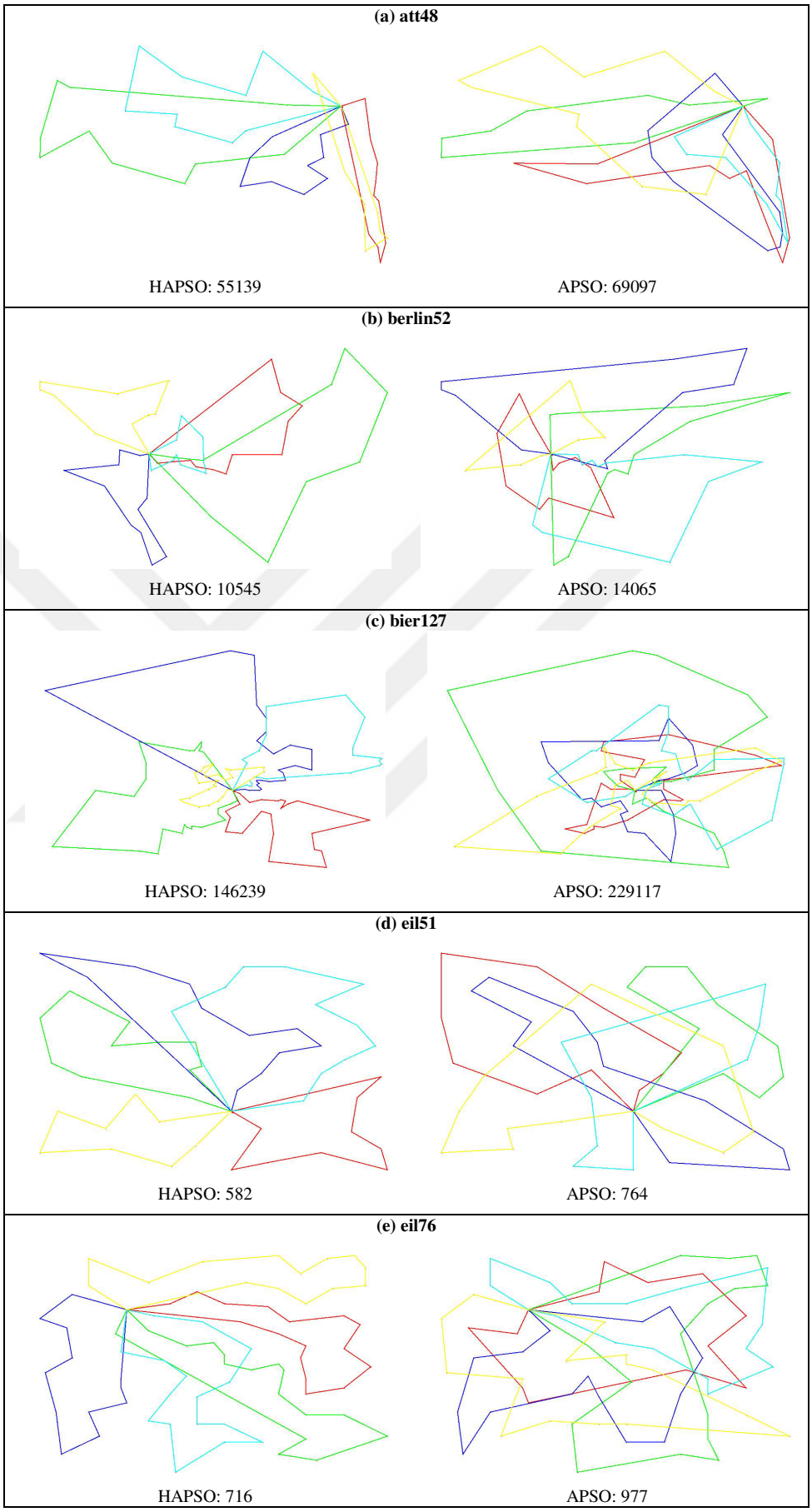
Şekil EK-1.2. 3 satıcı için HAPSO ve APSO ile bulunan en iyi turlar (Devamı)



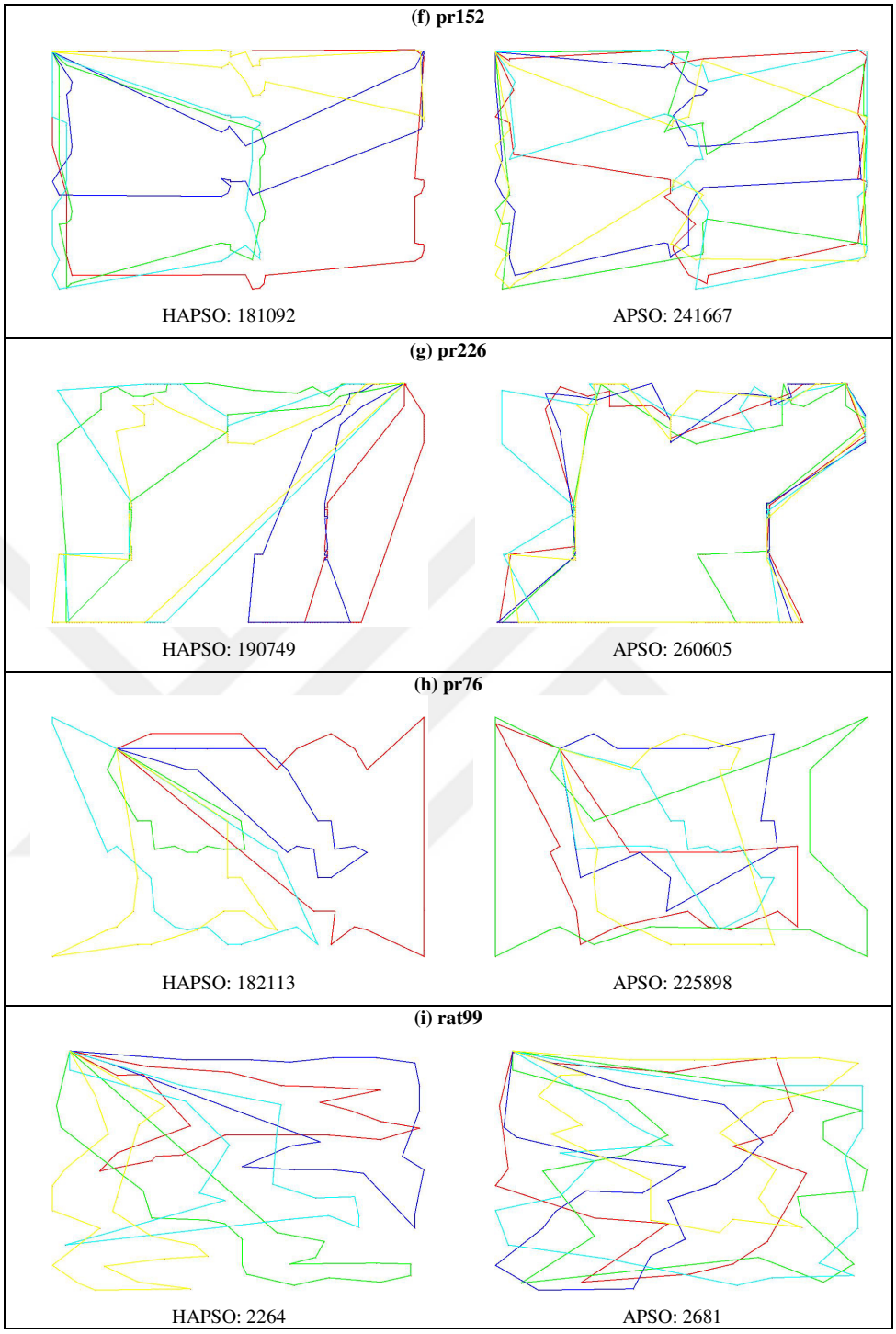
Şekil EK-1.3. 4 satıcı için HAPSO ve APSO ile bulunan en iyi turlar



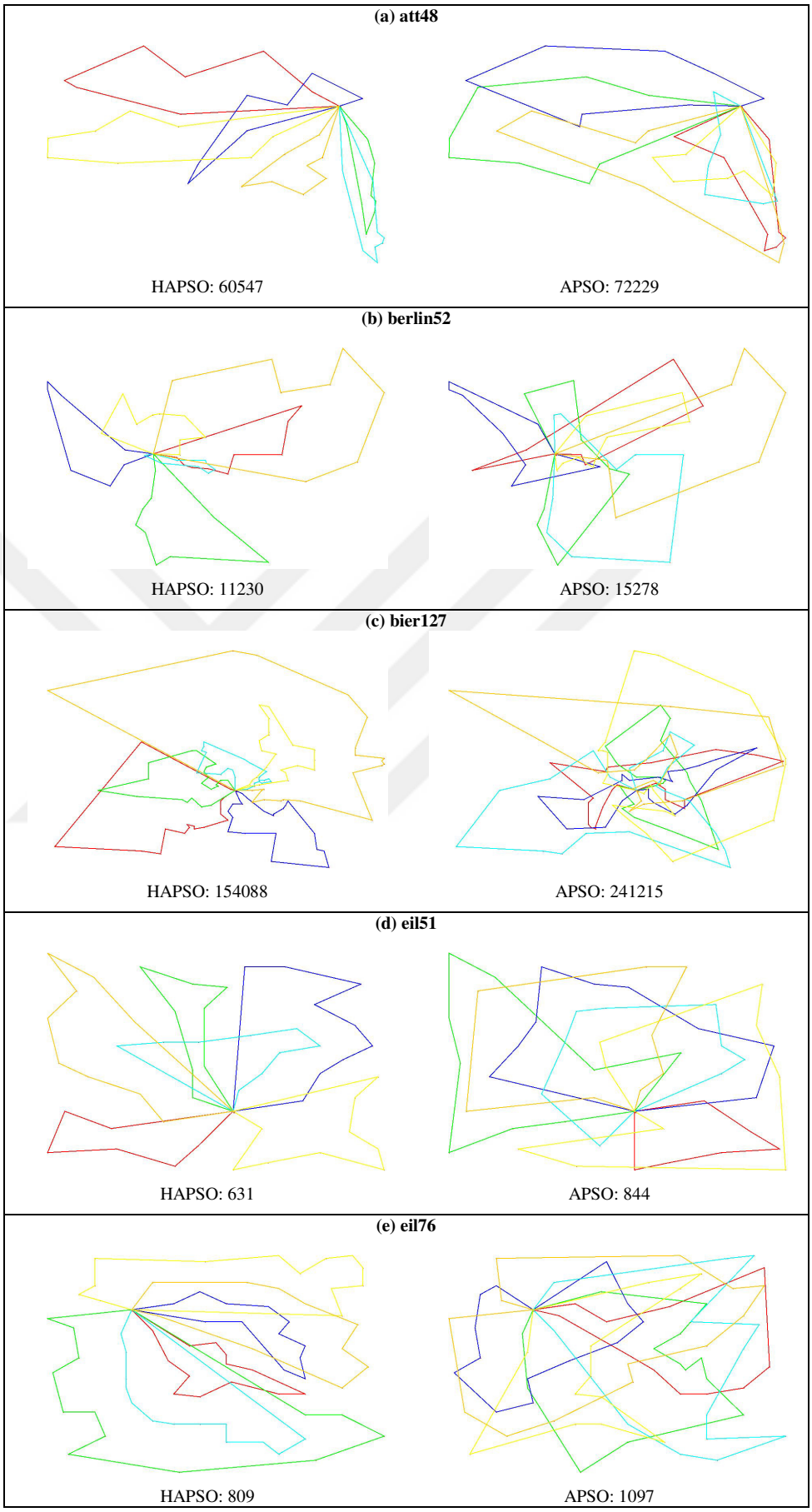
Şekil EK-1.4. 4 satıcı için HAPSO ve APSO ile bulunan en iyi turlar (Devamı)



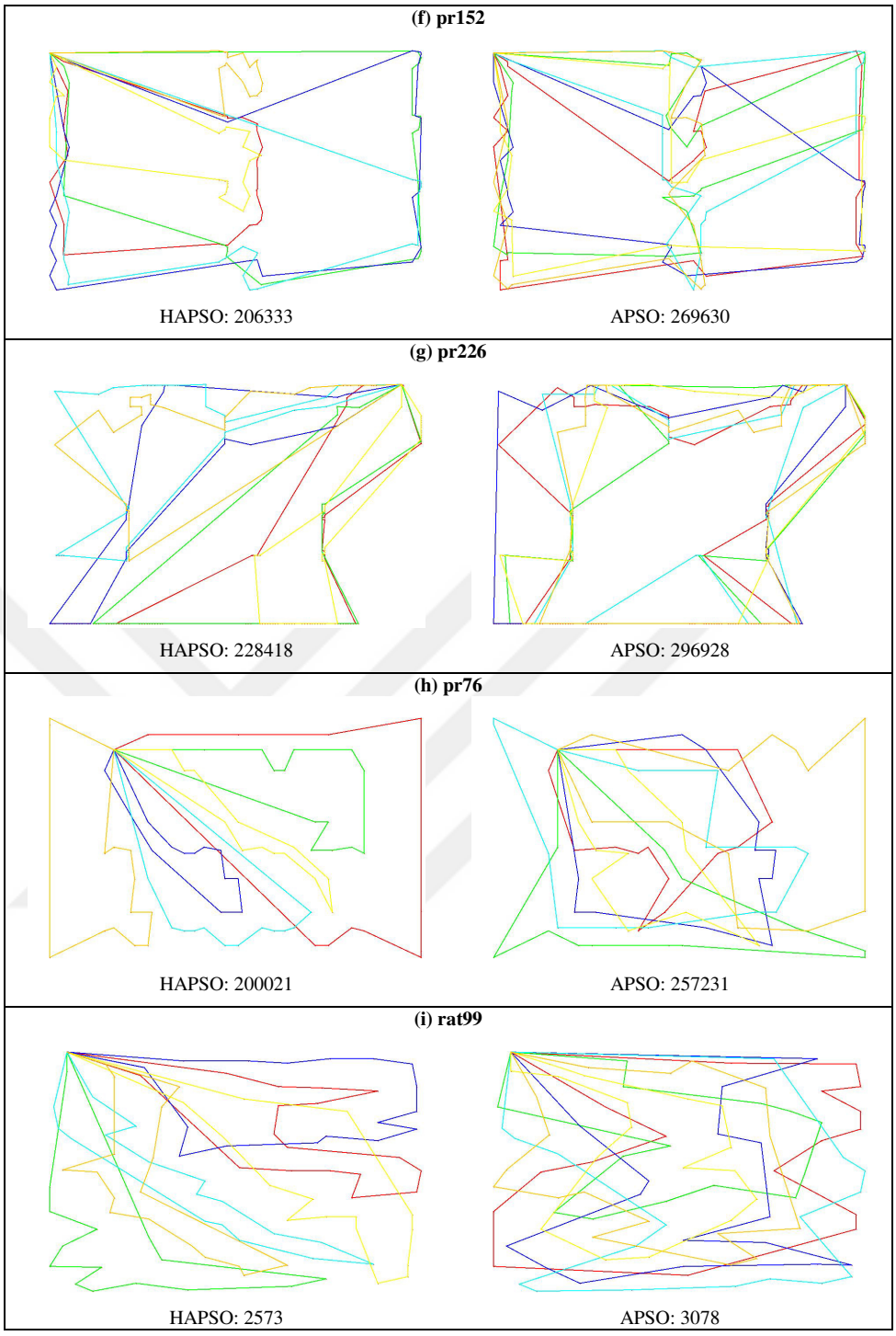
Şekil EK-1.5. 5 satıcı için HAPSO ve APSO ile bulunan en iyi turlar



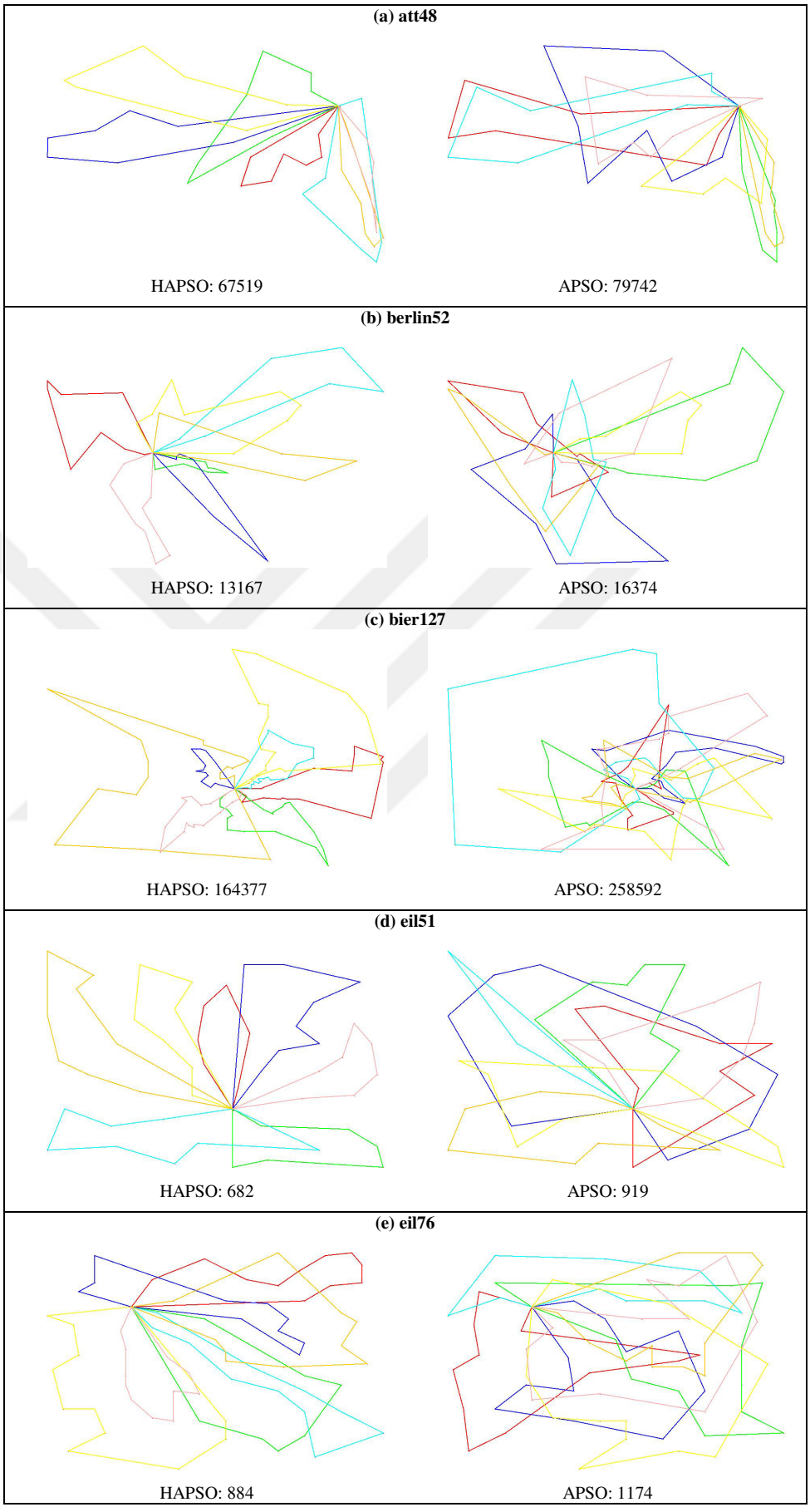
Şekil EK-1.6. 5 satıcı için HAPSO ve APSO ile bulunan en iyi turlar (Devamı)



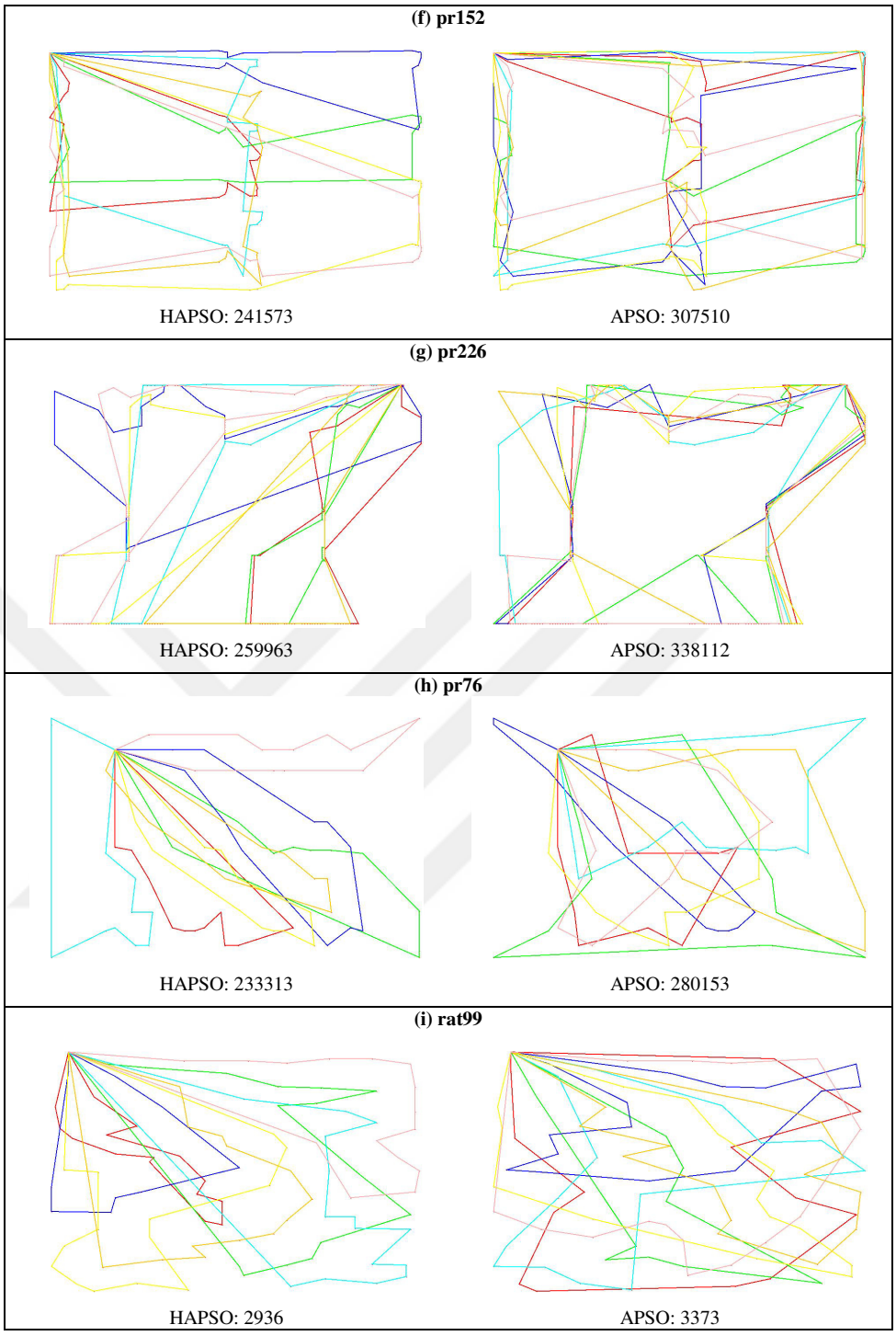
Şekil EK-1.7. 6 satıcı için HAPSO ve APSO ile bulunan en iyi turlar



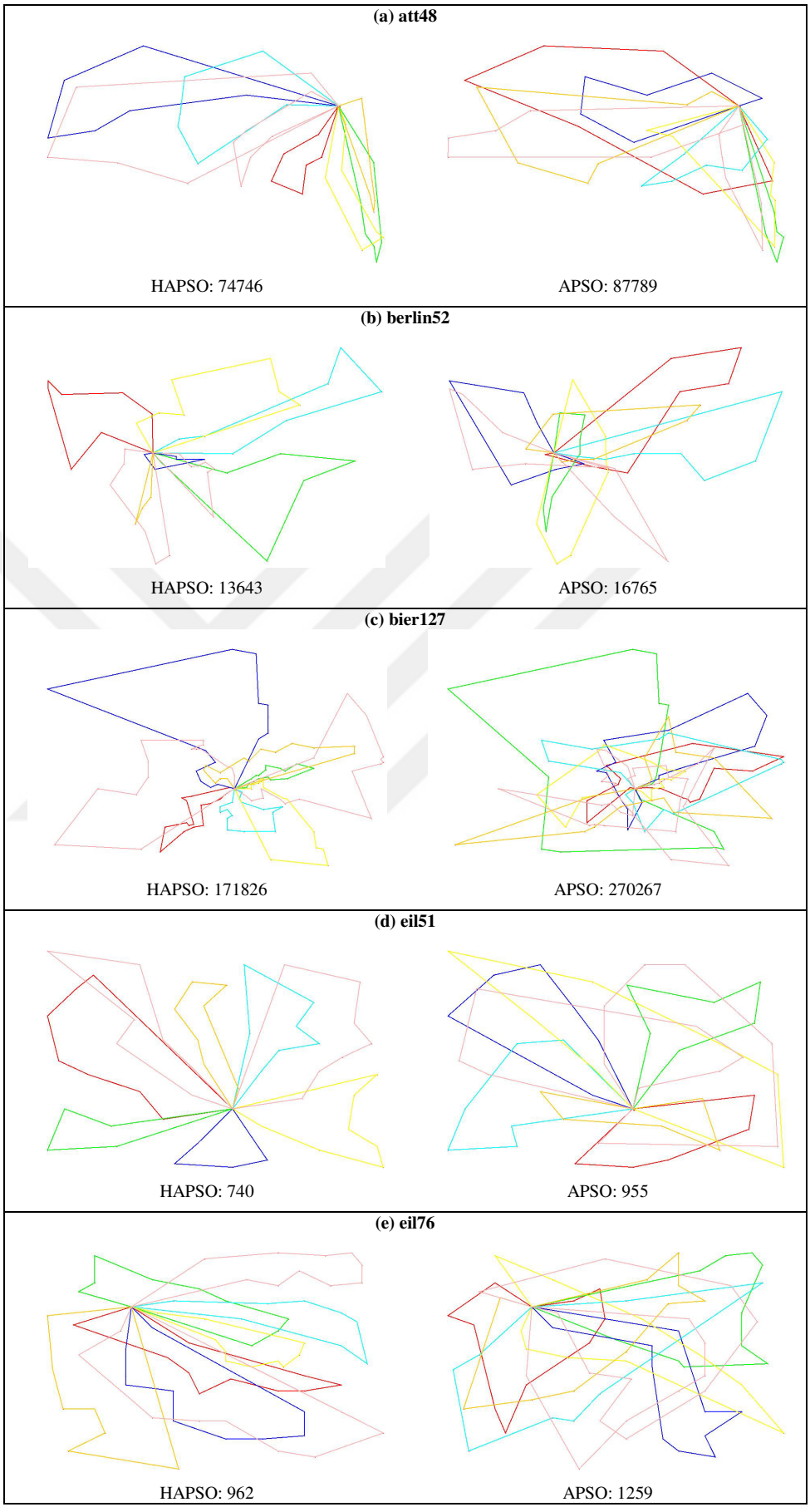
Şekil EK-1.8. 6 satıcı için HAPSO ve APSO ile bulunan en iyi turlar (Devamı)



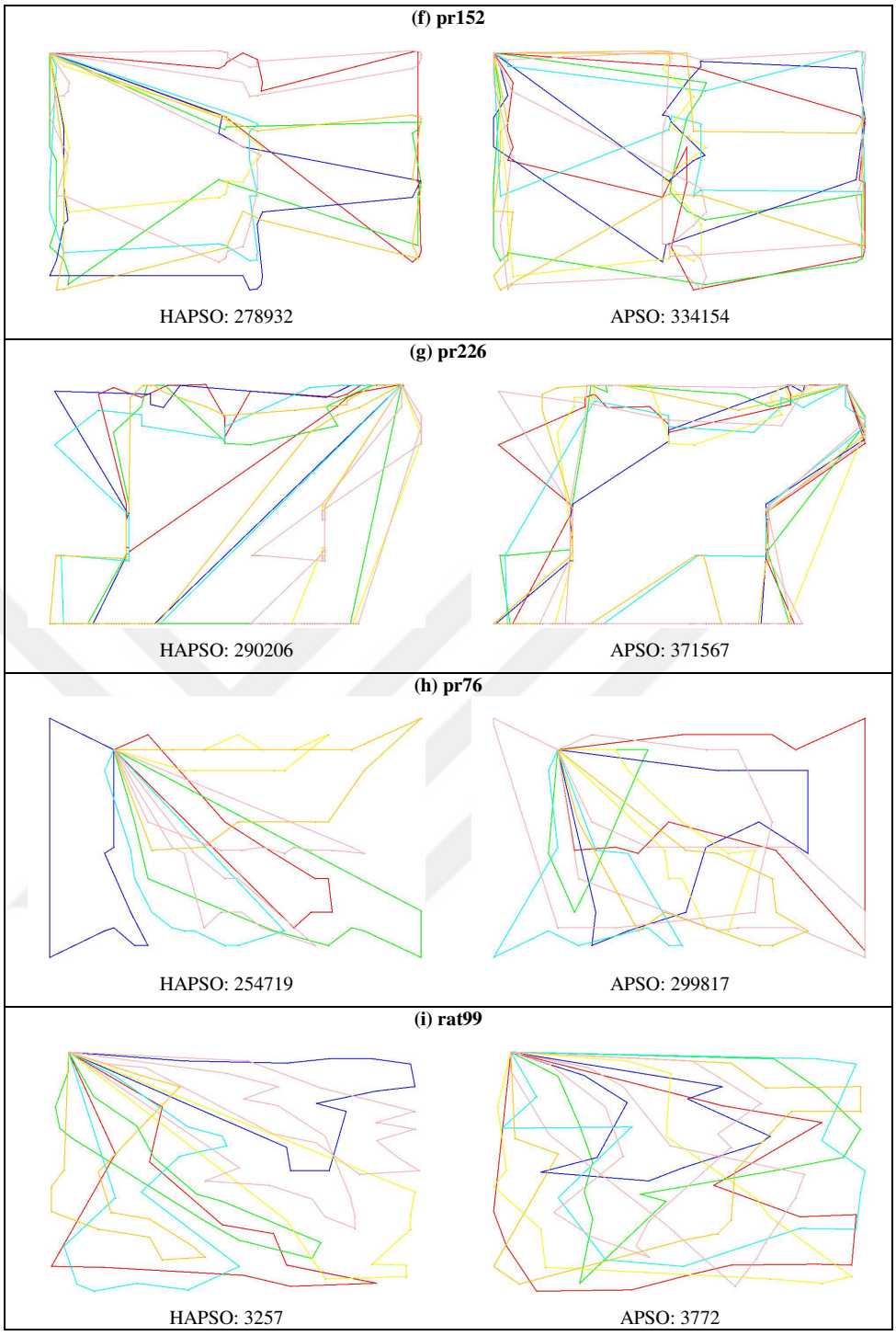
Şekil EK-1.9. 7 satıcı için HAPSO ve APSO ile bulunan en iyi turlar



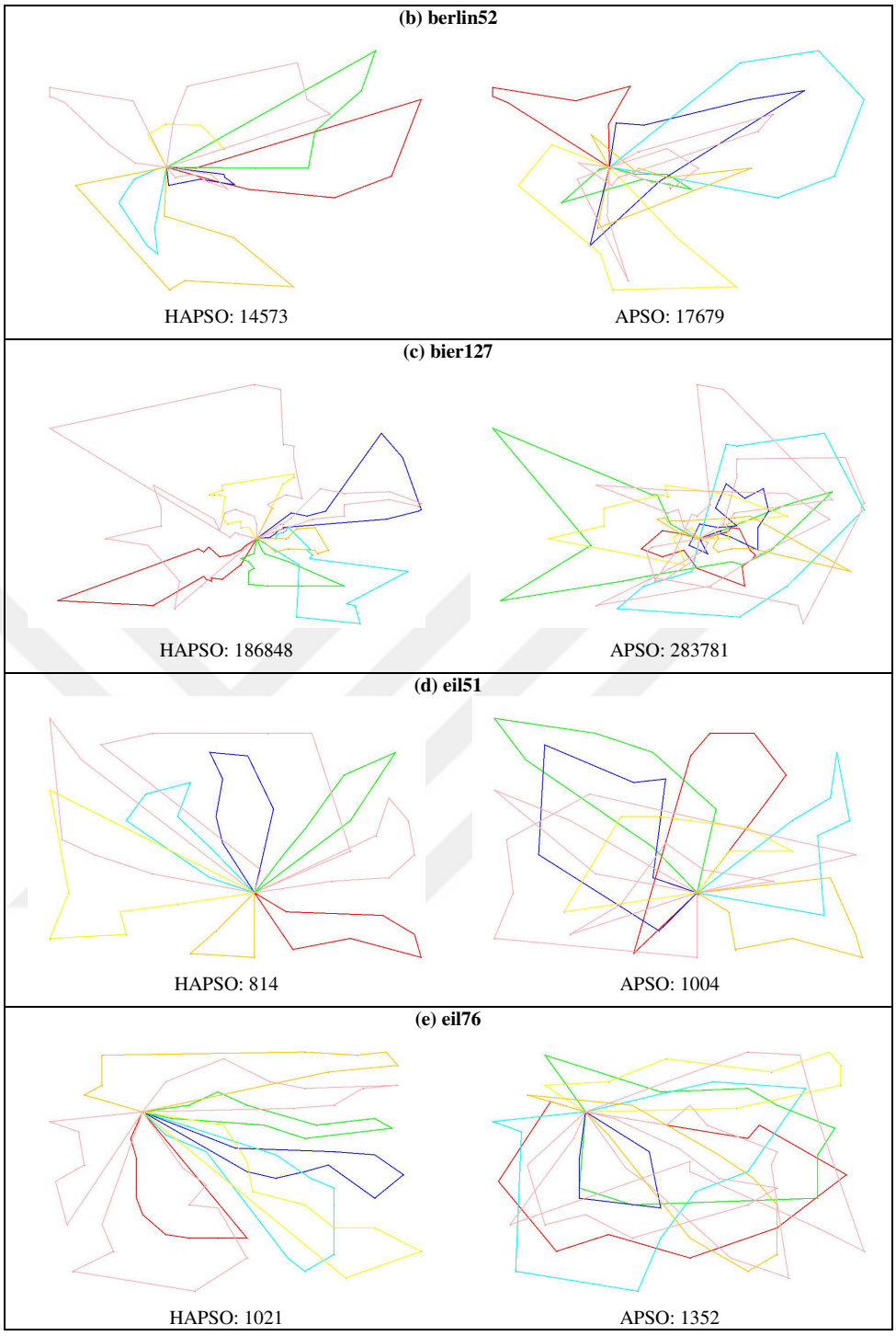
Şekil EK-1.10. 7 satıcı için HAPSO ve APSO ile bulunan en iyi turlar (Devamı)



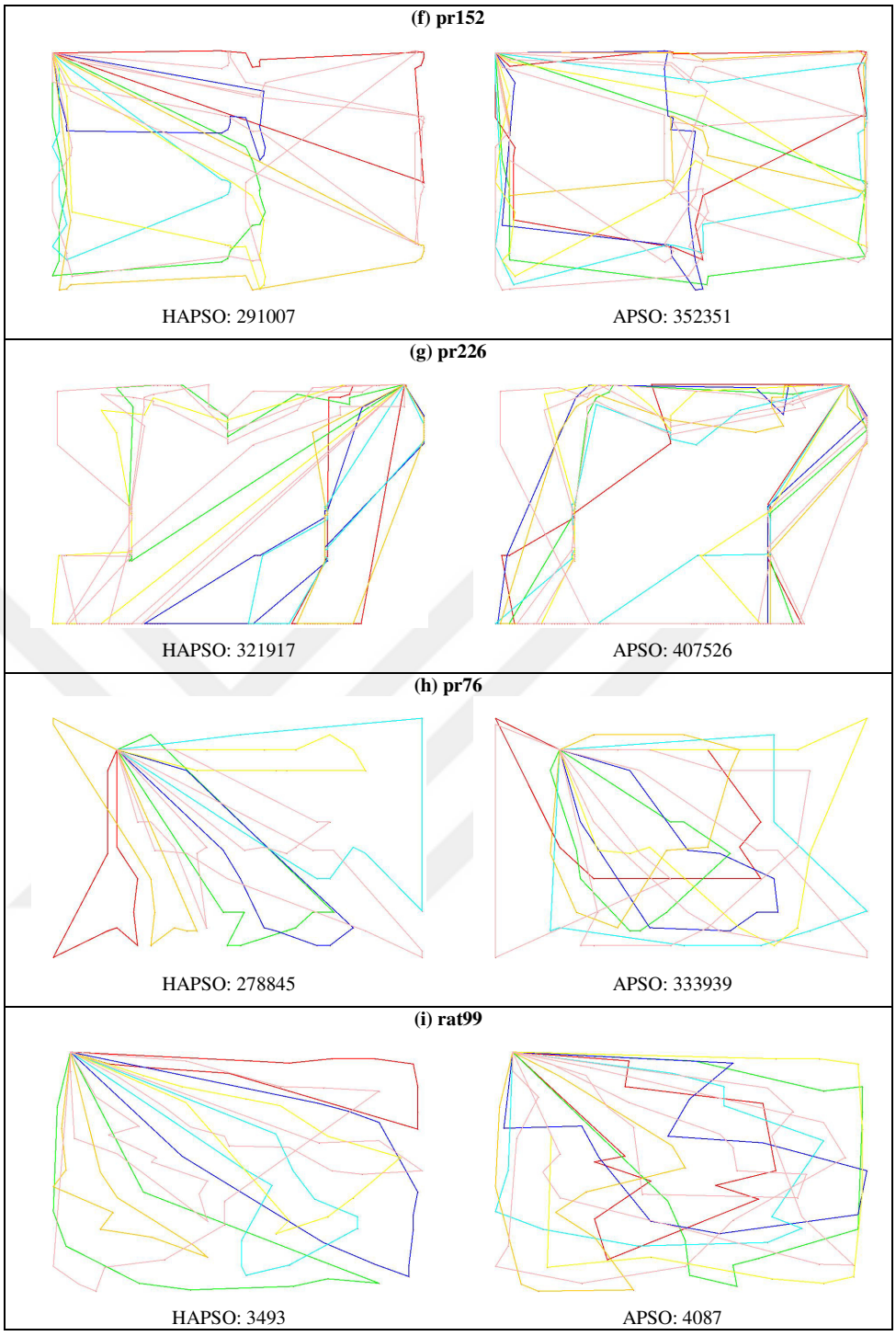
Şekil EK-1.11. 8 satıcı için HAPSO ve APSO ile bulunan en iyi turlar



Şekil EK-1.12. 8 satıcı için HAPSO ve APSO ile bulunan en iyi turlar (Devamı)

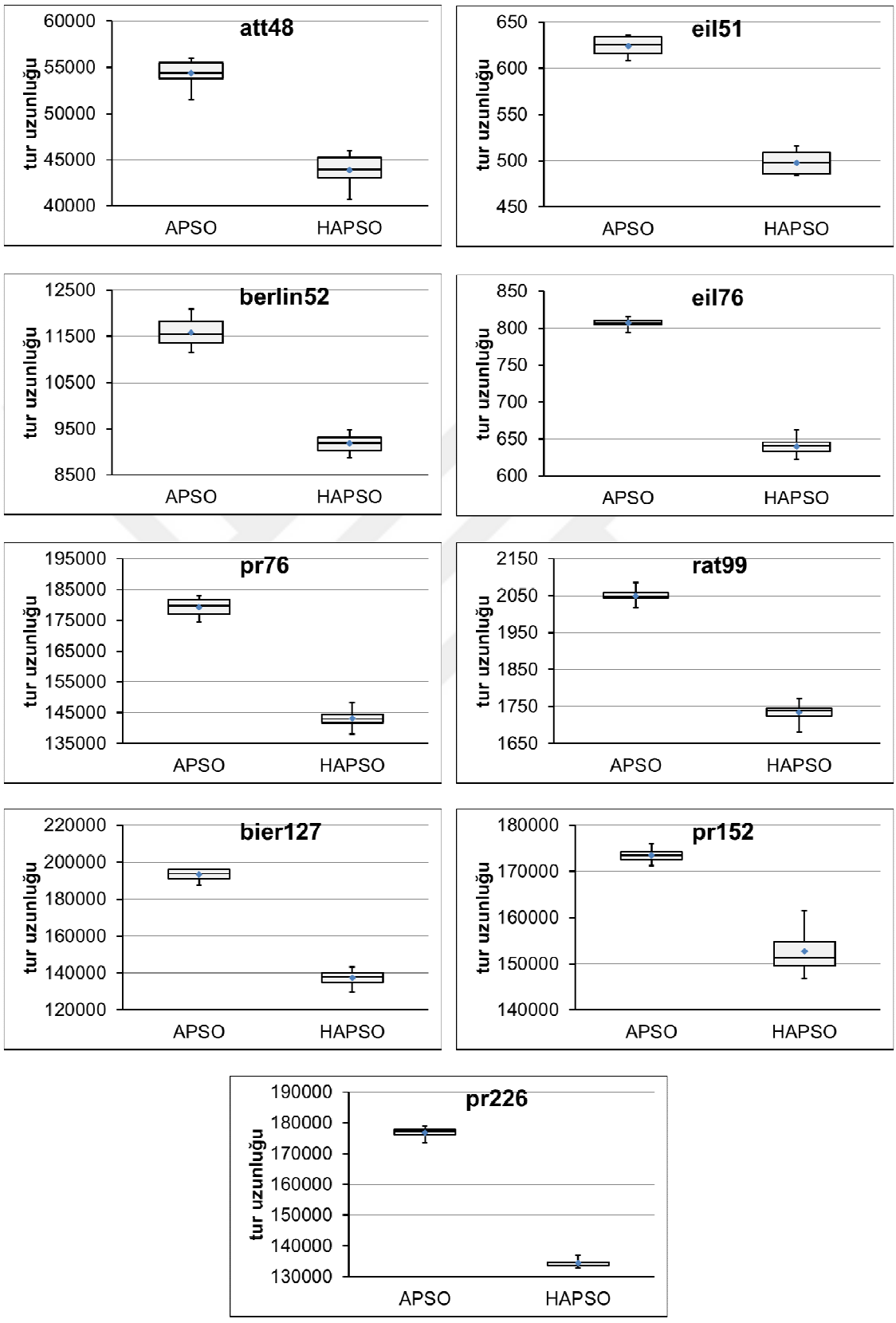


Şekil EK-1.13. 9 satıcı için HAPSO ve APSO ile bulunan en iyi turlar

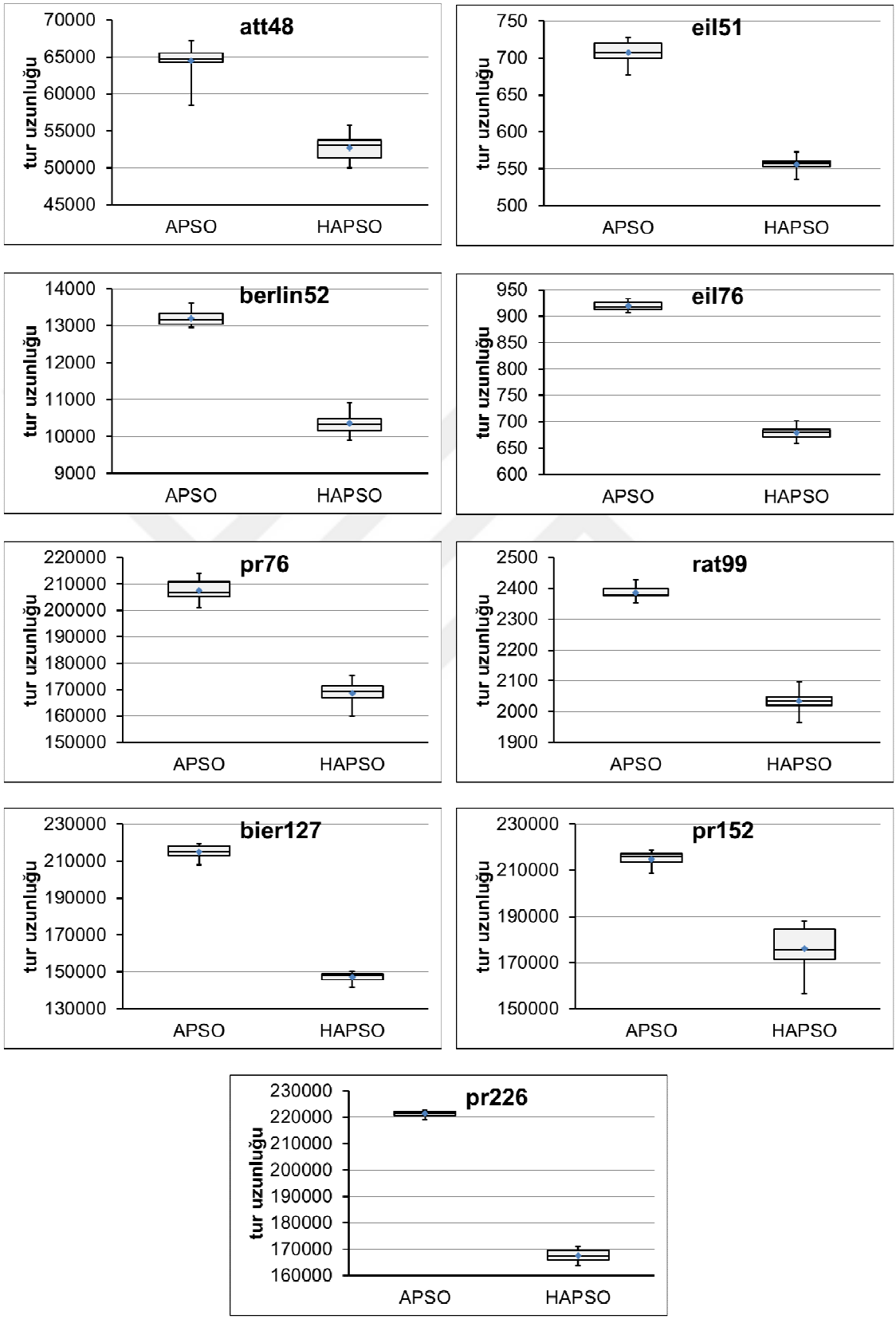


Şekil EK-1.14. 9 satıcı için HAPSO ve APSO ile bulunan en iyi turlar (Devamı)

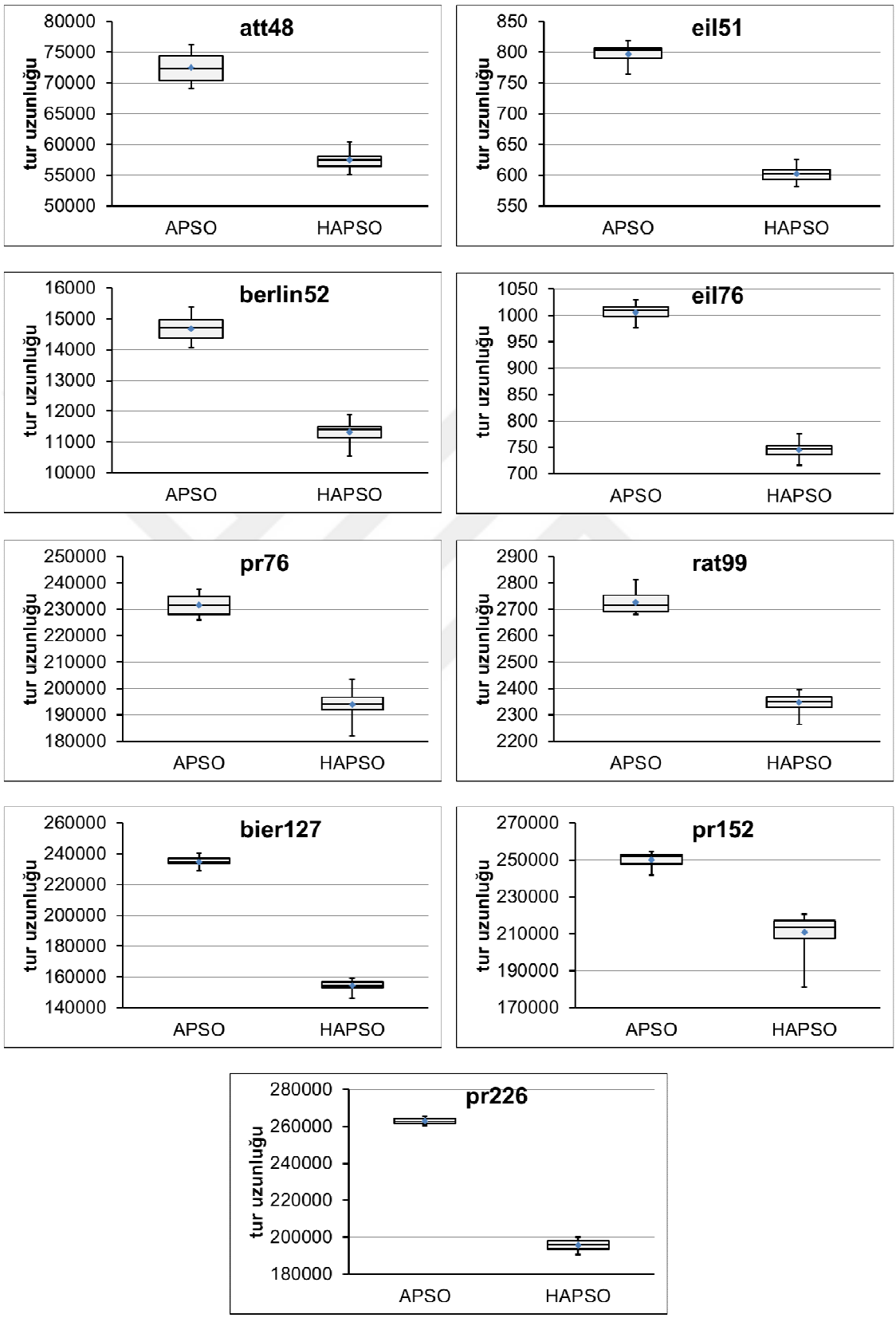
EK-2 APSO ve HAPSO algoritmalarının ÇGSP örneklerindeki kutu grafiği.



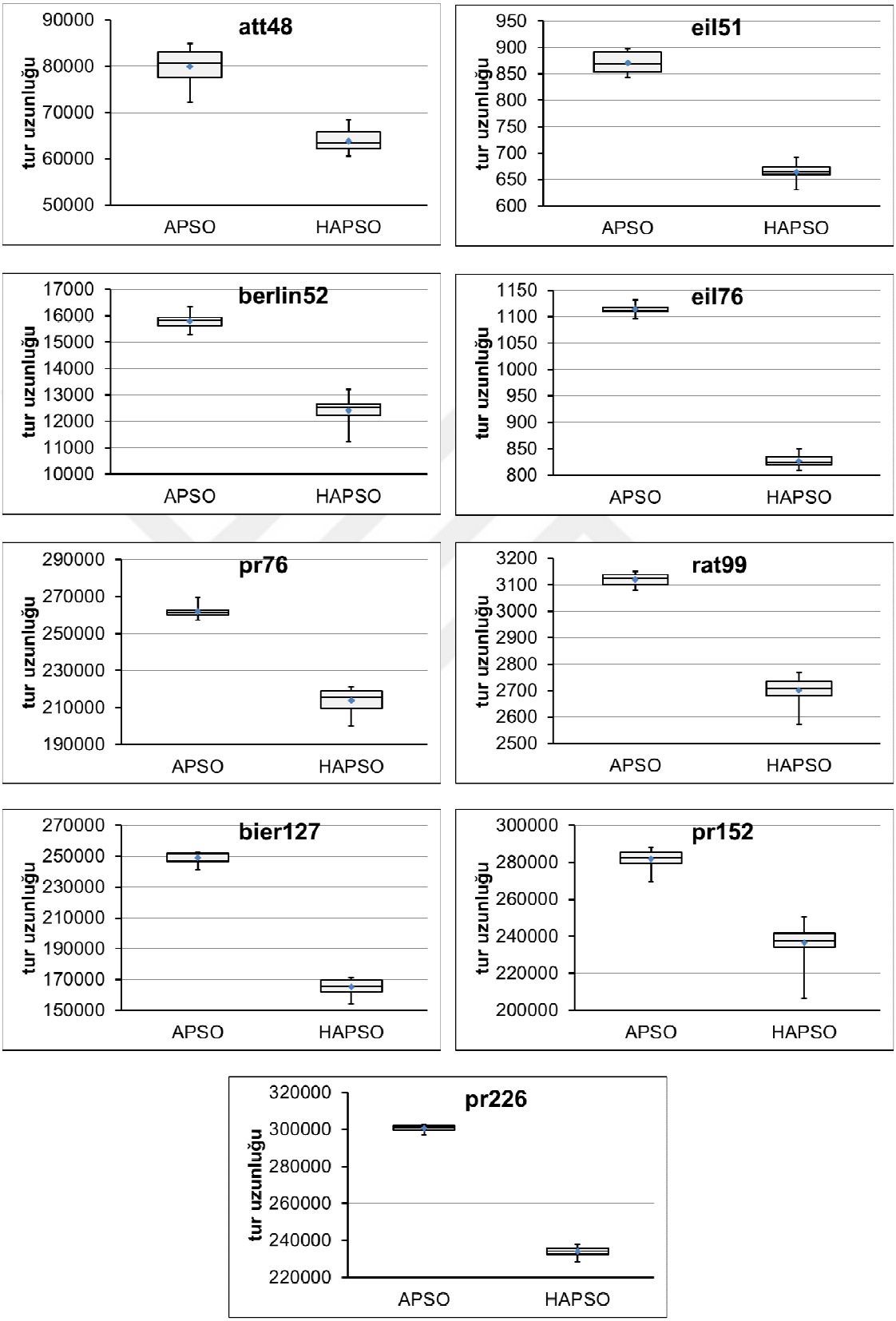
Şekil EK-2.1. 3 satıcı için ÇGSP örneklerinin kutu grafiği



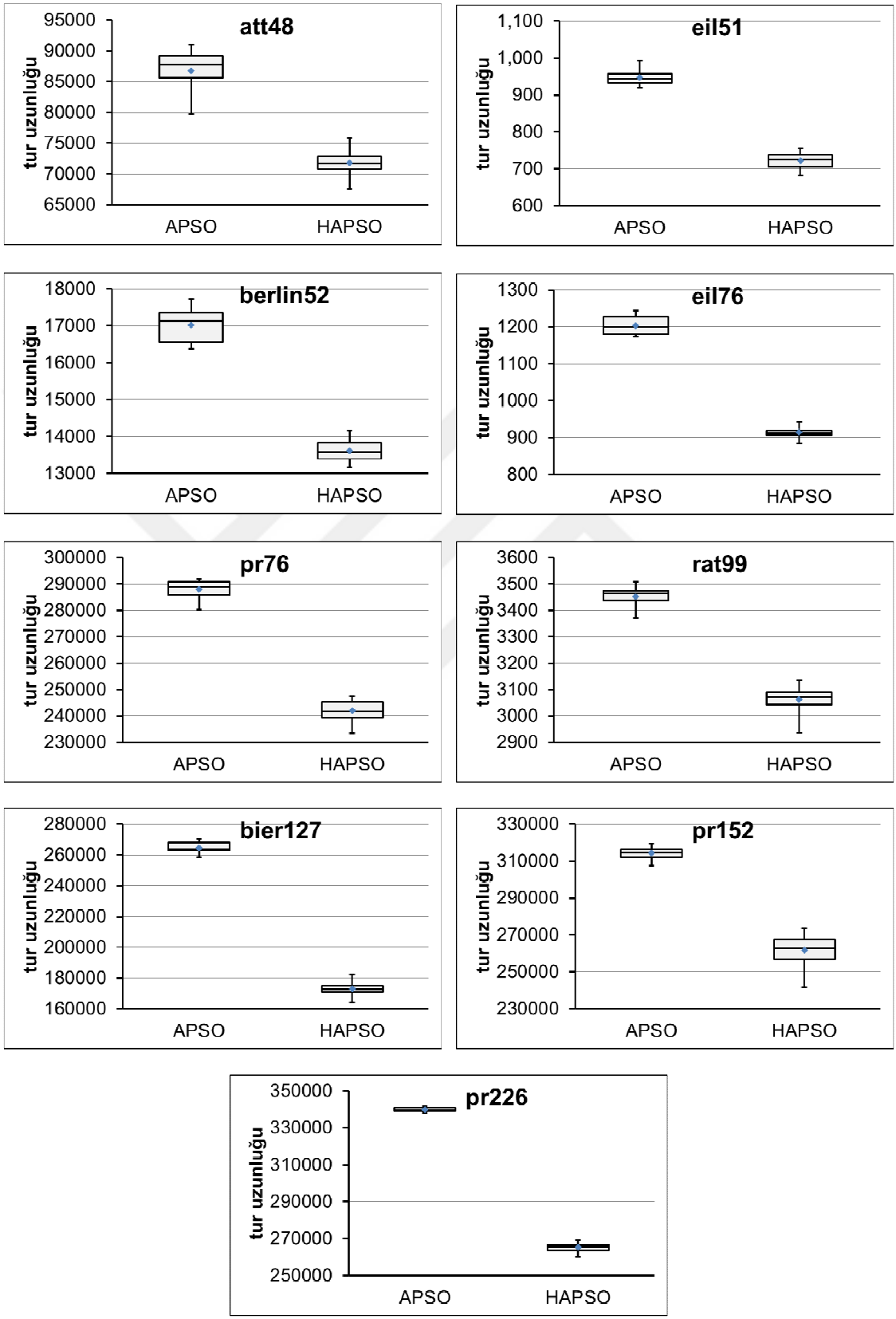
Şekil EK-2.2. 4 satıcı için ÇGSP örneklerinin kutu grafiği



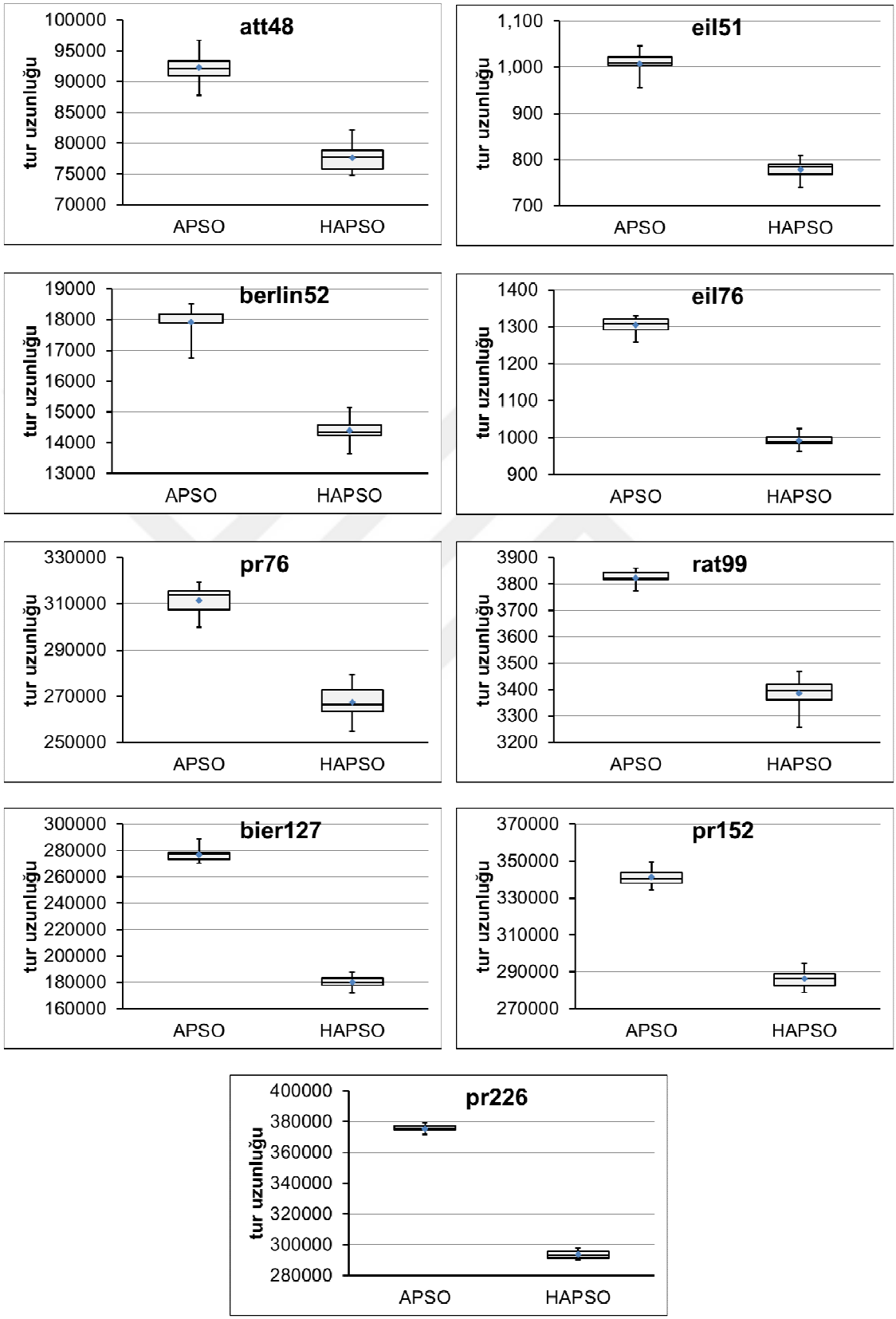
Şekil EK-2.3. 5 satıcı için ÇGSP örneklerinin kutu grafiği



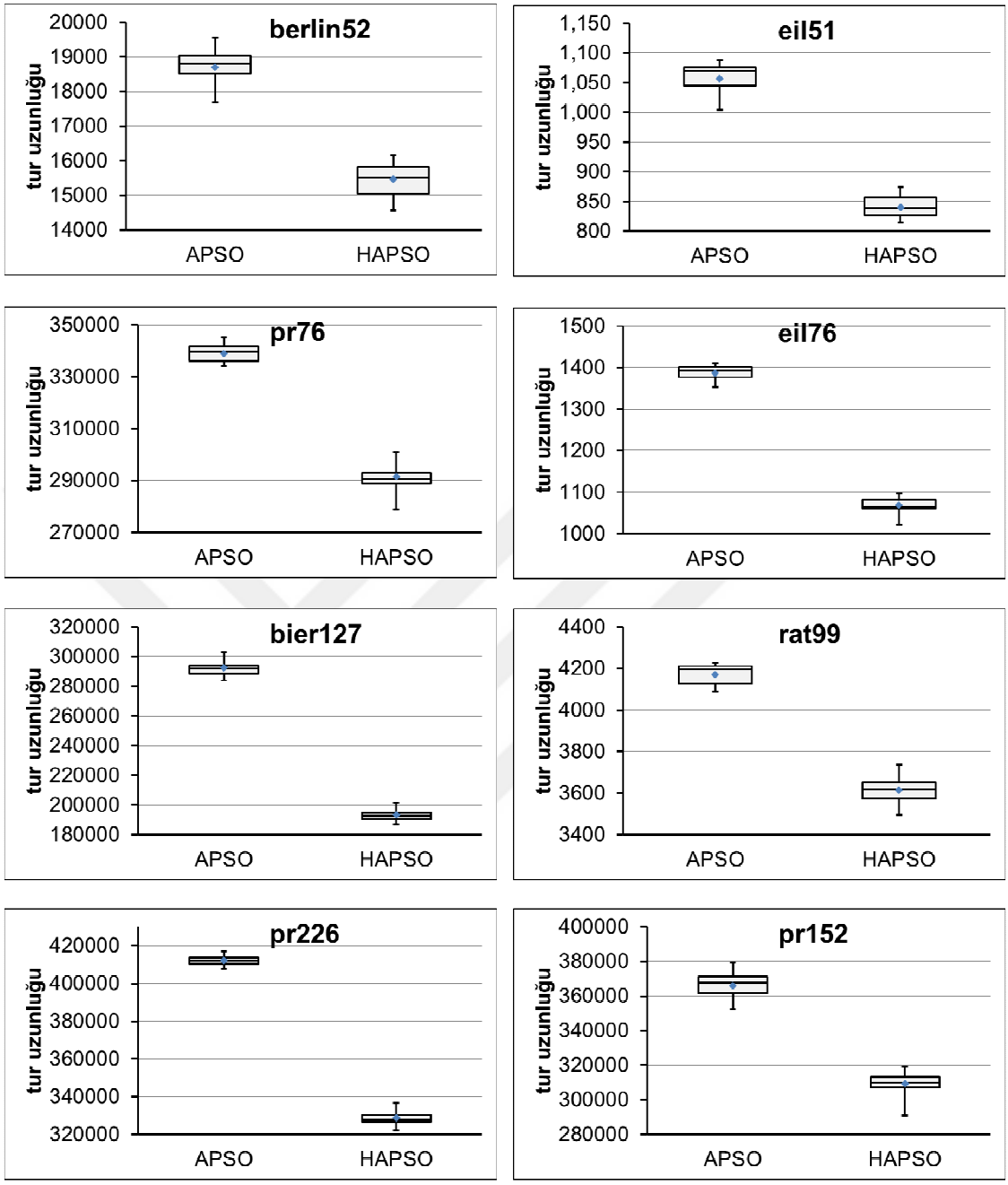
Şekil EK-2.4. 6 satıcı için ÇGSP örneklerinin kutu grafiği



Şekil EK-2.5. 7 satıcı için ÇGSP örneklerinin kutu grafiği



Şekil EK-2.6. 8 satıcı için ÇGSP örneklerinin kutu grafiği



Şekil EK-2.7. 9 satıcı için ÇGSP örneklerinin kutu grafiği

ÖZGEÇMİŞ

KİŞİSEL BİLGİLER

Adı Soyadı	**:**	Sevda DAYIOĞLU GÜLCÜ
Uyruğu	**:**	T.C.
Doğum Yeri ve Tarihi	**:**	Meram 10.05.1989
Telefon	**:**	
Faks	**:**	
e-mail	**:**	

EĞİTİM

Derece		**Adı, İlçe, İl**	**Bitirme Yılı**
Lise	:	Zeki Özdemir Süper Lisesi	2007
Üniversite	:	Selçuk Üniversitesi, Teknik Eğitim Fakültesi, Bilgisayar Sistemleri Öğretmenliği Bölümü	2012
Üniversite	:	Selçuk Üniversitesi, Mühendislik Fakültesi, Bilgisayar Mühendisliği Bölümü	2013
Yüksek Lisans	:	Konya Teknik Üniversitesi, Lisansüstü Eğitim Enstitüsü, Bilgisayar Mühendisliği Anabilim Dalı	

İŞ DENEYİMLERİ

Yıl	**Kurum**	**Görevi**
2014 – devam ediyor	Milli Eğitim Bakanlığı	Öğretmen

UZMANLIK ALANI: Optimizasyon algoritmaları

YABANCI DİLLER: İngilizce

YAYINLAR

Gülcü S. D., Örnek H. K., 2019, Solution of Multiple Travelling Salesman Problem using Particle Swarm Optimization based Algorithms, *International Journal of Intelligent Systems and Applications in Engineering*, 7(2), pp. 72-82. (Yüksek lisans tezinden)