T.C.

SELÇUK ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

GEZGİN SATICI PROBLEMLERİNİN ÇÖZÜMÜ İÇİN KARINCA KOLONİSİ OPTİMİZASYONU TABANLI HİBRİT BİR ALGORİTMA GELİŞTİRİLMESİ

Batuhan Saygın ARSLAN

YÜKSEK LİSANS TEZİ

Bilgisayar Mühendisliği Ana Bilim Dalı

Haziran - 2018
KONYA

TEZ KABUL VE ONAYI

Batuhan Saygın ARSLAN tarafından hazırlanan "Gezgin Satıcı Problemlerinin Çözümü İçin Karınca Kolonisi Optimizasyonu Tabanlı Hibrit Bir Algoritma Geliştirilmesi" adlı tez çalışması 25/06/2018 tarihinde aşağıdaki jüri tarafından oy birliği ile Selçuk Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Ana Bilim Dalı'nda YÜKSEK LİSANS TEZİ olarak kabul edilmiştir.

Jüri Üyeleri	**İmza**
Başkan Doç. Dr. Halife KODAZ	
Danışman Doç. Dr. Mustafa Servet KIRAN	
Üye Dr. Öğr. Üyesi Onur İNAN	

Yukarıdaki sonucu onaylarım.

Prof. Dr. Mustafa YILMAZ
FBE 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.

Batuhan Saygın ARSLAN

25/06/2018

ÖZET

YÜKSEK LİSANS TEZİ

GEZGİN SATICI PROBLEMLERİNİN ÇÖZÜMÜ İÇİN KARINCA KOLONİSİ OPTİMİZASYONU TABANLI HİBRİT BİR ALGORİTMA GELİŞTİRİLMESİ

Batuhan Saygın ARSLAN

Selçuk Üniversitesi Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

Danışman: Doç. Dr. Mustafa Servet KIRAN

2018, 55 Sayfa

Jüri
Doç. Dr. Mustafa Servet KIRAN
Doç. Dr. Halife KODAZ
Dr. Öğr. Üyesi Onur İNAN

Optimizasyon, bilimsel veya endüstriyel çalışmalarda elde edilecek çıktıyı en iyi sonuca ulaştırmak için kullanılır. Sonuç çıktısının minimize veya maksimize edilmesiyle optimizasyonun hedeflediği en iyi sonuca ulaşılır. Optimizasyon problemleri, aldığı parametrelere göre ayrık ve sürekli olarak sınıflandırılmaktadır. Gezgin satıcı problemi, satıcının belirli bir noktadan başlayıp uğranması istenen hedeflere sadece tek sefer uğrayarak en kısa yolu kullanıp başladığı noktaya geri dönüşünü içeren ayrık bir optimizasyon problemidir. Gezgin satıcı problemi çözümünde kullanılan birçok çözüm metodu vardır. Bu çözümlerden metasezgisel çözümler, problemin çözüm süresi bakımından diğer metotlara göre daha avantajlıdır ve başarılı sonuçlar elde ederler. Karınca kolonisi optimizasyonu metasezgisel bir çözüm yöntemidir ve gezgin satıcı problemi çözümünde sıkça kullanılır. Bu tez çalışmasında, karınca kolonisi optimizasyonunda kullanılan feromon, ayak izi olarak değiştirilmiş ve algoritma, komşuluk operatörleri ile hibritlenmiştir. Komşuluk operatörleri olarak rastgele yerleştirme, altdizileri rastgele yerleştirme, altdizilerin rastgele yerleştirilmesini terse çevirme yöntemleri daha iyi sonuçların elde edilmesi amacıyla kullanılmıştır. Yapılan bu değişiklikler ile yeni ve daha etkili bir çözüm türü elde edilmesi amaçlanmıştır. Bu çalışmada önerilen hibrit yaklaşım çok bilinen gezgin satıcı problemlerine, Türkiye'nin illerine ve Konya'nın ilçelerine uygulanmıştır. Sonuçlar standart karınca kolonisi optimizasyonu ile kıyaslanmıştır. Süre ve tur uzunluğu göz önünde bulundurulduğunda, oluşturan bu hibrit yaklaşımla, karınca kolonisi algoritmasına göre daha iyi sonuçlar elde edilmiştir.

Anahtar Kelimeler: Ayrık Optimizasyon, Karınca Kolonisi Optimizasyonu, Komşuluk Operatörleri, Gezgin Satıcı Problemi

ABSTRACT

MS THESIS

DEVELOPING A HYBRID ALGORITHM BASED ON ANT COLONY OPTIMIZATION TO SOLVE TRAVELLING SALESMAN PROBLEMS

Batuhan Saygın ARSLAN

THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCE OF SELÇUK UNIVERSITY
THE DEGREE OF MASTER OF SCIENCE
IN COMPUTER ENGINEERING

Advisor: Assoc. Prof. Dr. Mustafa Servet KIRAN

2018, 55 Pages

Jury
Assoc. Prof. Dr. Mustafa Servet KIRAN
Assoc. Prof. Dr. Halife KODAZ
Asst. Prof. Dr. Onur İNAN

Optimization is used to achieve the best result in scientific or industrial studies. By minimizing or maximizing the result output, the best result of optimization is achieved. Optimization problems are classified as discrete or continuous optimization problems according to the parameters. The traveling salesman problem is a discrete optimization problem that involves the seller who starts from a specific point, visits to the desired destination only once and returns to the starting point using the shortest route. There are many methods to solve travelling salesman problem. In these solutions, metaheuristic methods are more advantageous than the other methods in terms of solution time and achieving successful results. Ant colony optimization is a metaheuristic solution and often used to solve the travelling salesman problem. In this thesis study, the pheromone that used in ant colony optimization is modified as footprint and hybridized with neighborhood operators. Random insertion, random insertion of subsequences and reverse random insertion of subsequences operators as neighborhood operators are used to improve solutions. With these changes, it is aimed to get a new and more effective solution. In this study, the proposed hybrid approach applied to best-known travelling salesman problems, provinces of Turkey and districts of Konya. The results are compared with standard ant colony optimization. When the duration and lap length were evaulated, proposed hybrid approach acquires best results than ant colony optimization.

Keywords: Discrete Optimization, Ant Colony Optimization, Neighborhood Operators, Travelling Salesman Problem

ÖNSÖZ

Bu çalışma esnasında desteğini ve değerli fikirlerini esirgemeyen danışmanım Sayın Doç. Dr. Mustafa Servet Kıran'a, tezimin düzeltilmesinde değerli bilgilerini benimle paylaşan jüri üyelerim Sayın Doç. Dr. Halife Kodaz ve Sayın Dr. Öğr. Üyesi Onur İnan'a, çalışmalarım boyunca her zaman yanımda olan sevgili eşim Esra Ün Arslan'a ve manevi desteklerini hep yanımda hissettiğim ailemin tüm bireylerine teşekkürlerimi sunarım.

Batuhan Saygın ARSLAN
KONYA - 2018

İÇİNDEKİLER

SİMGELER VE KISALTMALAR

Kısaltmalar

ABC: Artificial bee colony
ACO: Ant colony optimization
ACS: Ant colony system
AS: Ant system
EAS: Elitist ant system
GSP: Gezgin satıcı problemi
KKA: Karınca kolonisi algoritması
KKO: Karınca kolonisi optimizasyonu
MMAS: Max-min ant system
TSP: Travelling salesman problem

1. GİRİŞ

1.1. Tezin Amacı ve Önemi

Optimizasyon, verilen kısıtlar ve parametreler altında, probleme uygun olarak belirlenen çözüm yolları yardımıyla, istenen faktörlerin maksimize, istenmeyen faktörlerin ise minimize edilerek en etkili şekilde ve en kısa sürede istenen bir çıkış bulunması işlemidir. Problemin kısa sürede ve etkili şekilde sonuç üretmesi, optimizasyon probleminin çözümünün etkili olduğunu gösterir. Farklı problemler için farklı çözümler mevcuttur. Bu çözümlerden hangisinin kullanılacağı, problemlerden yola çıkılarak belirlenmekte, hatta birden fazla çözüm yolu hibritlenerek kullanıma sunulmaktadır.

Optimizasyon yöntemleri, kullanılan parametreler yönünden sürekli ve ayrık olmak üzere ikiye ayrılır. Sürekli olanlar sonsuz değer alırken, ayrık olanlar sınırlı değerler alır. Örneğin, yapılacak işler bir liste halinde verilmiş ise, bu işlerin yapılması birbirinden bağımsız olacağından bu problemin ayrık parametreli olduğu düşünülebilir. Ayrık parametreli optimizasyon, kombinasyonel bir optimizasyon olarak da adlandırılabilir.

Ayrık zamanlı optimizasyon problemi olarak bilinen “Gezgin Satıcı Problemi” (Traveling Salesman Problem) veya kısaca “GSP” (TSP), aralarındaki uzaklıklar bilinen belirli sayıda noktanın (şehir, parça veya düğüm gibi) her birinden yalnızca bir kez geçerek başlangıç noktasına dönen en kısa veya en az maliyetli turun bulunmasını hedefleyen bir problemdir.

Gezgin satıcı problemi ilk olarak 1930’larda matematikçi Karl Menger tarafından tanımlanmıştır (Held ve ark., 1984). Problem tanımı basit olmasına rağmen oluşabilecek çok sayıda olasılık yüzünden çözümü zordur. Problemde değişken sayısının artışına bağlı olarak çözüm uzayı genişler, problemin çözüm zamanı üssel olarak artar. Bu yüzden problemin çözümünde standart çözüm yöntemleri yetersiz kalmaktadır. Tüm çözüm uzayını taramak yerine mantıksal çıkarımlar ile çözüm uzayında kısmi taramalar yapan sezgisel ve metasezgisel yöntemler problemin çözümünü garanti etmezler, fakat standart yöntemlere göre daha iyi sonuçlar verirler.

Karınca kolonileri, görsel ipucu kullanmadan sistematik olarak iki yoldan daha kısa olanı seçme yeteneğine sahiptir. Çevredeki değişikliklere de uyum sağlayabilirler; yeni oluşan bir engel nedeniyle kullandıkları yol kapanırsa, en kısa olan bir başka yol

bulmaya çalışırlar (Beckers ve ark., 1992). Karıncalar yürürken belli miktarda feromon biriktirirler ve genelde feromon açısından zengin bir yönü seçmeyi tercih ederler. Aslında, beklenmedik bir engele rağmen kısa yolu bulma yetenekleri buna bağlıdır.

Karıncalar kullandıkları olağan yolun beklenmeyen bir engelle kesilmesiyle feromonu takip edemezler. Böyle bir durumda karıncaların yarısının uzun yolu diğer yarısının kısa yolu seçmesi beklenir. Şans eseri, kısa yolu seçen karıncalar feromon izini, uzun yolu seçenlere göre daha çabuk oluşturur. Tüm karıncalar yaklaşık aynı hızda hareket edip, yaklaşık olarak aynı oranda feromon izi bırakır; ilginç olan uzun yolun kısa yoldan çok daha uzun sürmesi ile feromon izinin kısa tarafta daha hızlı birikmesini sağladığı gerçeğidir (Dorigo ve Gambardella, 1997).

Bu tezin amacı ise literatürde ayrık zamanlı optimizasyon problemleri olarak geçen problemlerin çözümüne yeni hibrit bir yaklaşımla çözüm sunmaktır. Kullanılacak olan algoritma, ayrık zamanlı optimizasyon problemi olarak bilinen gezgin satıcı problemi üstünde denenecektir. Bu problemin literatürde bilinen metasezgisel olan çözümlerinin yerelde takılı kaldığı, gerçek minimum mesafeyi bulamadığı bilinmektedir. Ayrıca metasezgisel çözümler dışındaki klasik çözümlerin uzun çalışma süresi gerektirdiği de bilinmektedir.

Bu problemin daha önceki çözümlerinden de esinlenilerek, eksik olan kısımları giderilecek şekilde, bir kısmı değiştirilmiş ve geliştirilmiş olan karınca kolonisi algoritması yardımıyla bir çözüm sunulması hedeflenmektedir. Algoritmanın son adımında, diğer çözümlerin dezavantajı olan yerel sonuçta takılma probleminin çözülmesi için yerel olarak iyileştirme sağlayacak olan başka metotlar da kullanılacaktır. Böylece bu ve buna benzer ayrık optimizasyon problemlerinin çözümlerinin daha stabil, hızlı ve yerel sonuca takılmayacak bir şekilde yapılmasına katkı sağlanması hedeflenmektedir. Tanıtılan hibrit çözüm, genel olarak bilinen ve kabul gören gezgin satıcı problemleri üzerinde uygulanacaktır. Ayrıca Türkiye'nin illeri ve Konya'nın ilçeleri için de bir uygulama yapılacaktır.

Gezgin satıcı probleminin çözümünü yapmak ve optimize etmek için karınca kolonisi optimizasyonu kullanılan yöntemlerden biridir. Günümüzde kullanılan iş gücü ve zamanın azaltılması ile daha verimli işler ortaya çıkarılabilmektedir. Bunu sağlayan optimizasyon yöntemleri ve bu konuda yapılan çalışmalar önem arz etmektedir. Bu tezde de bahsedilen önemli konu içinde yer alan ayrık optimizasyon problemlerine yeni hibrit bir çözüm sunarak bir iyileştirme sağlanması hedef olarak benimsenmiştir. Bu kapsamda, literatürde gezgin satıcı problemi olarak bilinen ayrık optimizasyon problemi

ele alınacak ve bu problemin daha stabil ve hızlı bir şekilde çözülmesi sağlanarak, bu çözümün gerçek hayata uyarlanması ve uygulanması üzerinde çalışılacaktır. Yapılacak olan tez ile ortaya çıkarılacak çözümün, sadece gezgin satıcı problemi için değil, başka ayrık optimizasyon problemleri için de çözüm olarak kullanılması sağlanabilecektir.

Tezde sunulan yeni çözüm ile diğer metasezgisel optimizasyon çözüm yöntemlerinin dezavantajları ortadan kaldırılarak, yerel sonuca takılı kalınması yani yanlış çözüm bulunması konusunda iyileştirmeler yapılacaktır. Bu sayede ayrık optimizasyon problemleri üzerinde kullanılabilecek ve daha doğru sonuçlar elde edebilecek bir hibrit yaklaşım ortaya çıkarılıp, kullanılması sağlanacaktır.

1.2. Tez Organizasyonu

Bu tez çalışması giriş, kaynak araştırması, materyal ve yöntem, uygulama sonuçları, sonuç ve öneriler olmak üzere beş bölümden oluşmaktadır.

Birinci bölüm, giriş kısmını oluşturmaktadır. Bu bölümde çalışmanın konusu ile genel bilgiler verilmiş, tezin amacına ve önemine değinilmiştir.

İkinci bölümde, ayrık optimizasyon problemleri, çözüm yöntemleri ve yerel iyileşme sağlayan yaklaşımlardan bahsedilmiştir.

Üçüncü bölümde, bu tez çalışmasında kullanılan materyal ve metotlar hakkında bilgi verilmiştir. Standart karınca kolonisi optimizasyonu ile önerilen algoritmanın farklarına değinilmiştir.

Dördüncü bölümde, önerilen hibrit yaklaşım ile çok bilinen bazı problemler, Türkiye'nin illeri ve Konya'nın ilçeleri baz alınarak gezgin satıcı problemi üzerinde uygulama çalışması yapılmıştır.

Beşinci bölümde ise elde edilen sonuçlar ve karşılaştırma tabloları verilmiştir.

Tez çalışmasının sonunda ise kaynak bilgisi yer almaktadır.

2. KAYNAK ARAŞTIRMASI

2.1. Ayrık Optimizasyon Problemleri

Optimizasyon, elde edilecek olan sonucu minimize veya maksimize etme prensibiyle ortaya çıkmıştır. Optimizasyon işleminde ilk olarak sonucu direkt etkileyecek olan karar parametrelerinin belirlenmesi esastır. Sonrasında da bu parametrelere bağlı olarak enküçükleme(maliyet) veya enbüyükleme(kâr) fonksiyonu belirlenir. Bu işlem yapılırken belirli sınırlamalar mevcuttur. Maliyet fonksiyonu, olabildiğince küçük değerler üretmeye çalışırken, kâr fonksiyonu olabildiğince büyük değerler üretmeye çalışır. Maliyet fonksiyonu matematiksel olarak Denklem 2.1, Denklem 2.2, Denklem 2.3 ve Denklem 2.4'te gösterilmiştir.

$$x = (x_1, x_2, \dots, x_i, \dots, x_n) \tag{2.1}$$

$$f(x) = (x_1, x_2, \dots, x_n) \tag{2.2}$$

$$h_j(x) = h_j(x_1, x_2, \dots, x_n) = 0, \qquad 1 \leq j \leq p \tag{2.3}$$

$$g_i(x) = g_i(x_1, x_2, \dots, x_n) \leq 0, \qquad 1 \leq i \leq m \tag{2.4}$$

Denklemlerdeki x, n değişkenli bir vektördür. x_i, i. parametrenin değerini göstermektedir. f(x), maliyet fonksiyonudur. $h_j(x)$, maliyet fonksiyonunun sahip olabileceği p tane eşitlik sınırlamasını göstermektedir. $g_i(x)$, maliyet fonksiyonunun sahip olabileceği m tane eşitsizlik sınırlamasını göstermektedir. Problemlerin bazılarında birden fazla maliyet fonksiyonu olabilir. Bu durumda bulunacak birden fazla optimizasyon sonucu ortaya çıkar. Böyle problemlere, çok amaçlı optimizasyon problemleri adı verilir (Karaboga, 2004).

Optimizasyon problemleri, kullandığı parametrelere göre ayrık veya sürekli optimizasyon olarak ikiye ayrılmaktadır. Ayrık optimizasyon problemleri belirli bir aralıktaki sınırlı değerleri alırken, sürekli optimizasyon problemleri belirli bir aralığı olmayan sürekli değerler almaktadır. Ayrık optimizasyon problemi türündeki birçok problem NP-hard problemdir. P-problem polinomal zamanda çözülebilen problemlerin, NP-hard problem ise polinomal zamanda bir çözümü olduğu ispatlanamayan karar

problemlerin karmaşıklık sınıfıdır. Bu tür problemler sezgisel yöntemlerle çözülebildiği gibi, metasezgisel yöntemlerle de çözülebilmektedir. Metasezgisel çözümler, sezgisel çözümlere göre zor problemleri daha uygun sürede çözümleyebilmektedir.

Ayrık optimizasyon problemleri gerçek dünya problemlerine yakın problemleri içerir. Bunlara örnek olarak, müşterilere uygun minimum ücret planı oluşturmak, internet veri paketlerinin en iyi şekilde rotalandırılması, üretim bandında en optimal iş düzeninin bulunması, havaalanı uçuş planının belirlenmesi ve daha birçok örnek verilebilir.

Ayrık optimizasyon problemlerini çözümlerken, en iyi çözüme ulaşmanın zorluğu değerlendirilmelidir. Bunu yaparken en kötü zaman karmaşıklığı belirlenmelidir. Bazı önemli ayrık optimizasyon problemleri polinomal zamanda çözülebilmesine rağmen, bir çoğu polinomal zamanda çözülememektedir. Bu problemler için çözüm süreleri, uzay büyüklüğünün artmasına bağlı olarak katlanarak artmaktadır.

2.2. Gezgin Satıcı Problemi

GSP'nin ilk kullanıldığı yer tam olarak bilinmese de, problemin ilk örnekleri 1832 yılında Almanca "Der Handlungsreisende – Von einem alten Commis – Voyageur" kitabında Almanya ve İsviçre arasında seyahat eden satıcı ile ortaya çıkmıştır. Bu kitap sadece bir rehberdir ve bu kitapta matematiksel ifadeler yer almamaktadır. Daha sonraları ise insanlar optimal yollar bulmanın zamansal olarak tasarruf sağladığının farkına varmışlar ve kısa yollar bulmak için çalışmalar yapmaya başlamışlardır (Cook, 2012).

Literatürde bilinen şekliyle GSP, ilk olarak 1930'larda ortaya çıkmaya başlamıştır. 1972'de Richard M. Karp, Hamilton çevrim problemlerinin NP-hard bir problem olduğunu göstermiş ve optimal turlar bulmanın zorluklarını ortaya çıkarmıştır (Cook, 2012).

GSP, belirli düğüm sayısı ve yol ağırlıklarına sahip bir Hamilton çevrimindeki minimum çıktıyı bulmaya çalışan genelleştirilmiş bir problemdir. GSP, NP-hard bir problem türüdür, yani büyük uzaya sahip türleri polinomal zamanda çözümlenemez.

Problemin tanımının basit olması ve bu konuda bilinen problemler ve çözümlerinin mevcut olmasından dolayı bu tezde gezgin satıcı problemi ele alınmıştır.

Böylece kıyaslama ve doğruluk konusunda sıkıntı yaşanmaması öngörülmüştür. Problemi kısaca tanımlayacak olursak, GSP şu şekilde tanımlanabilir:

- Ana şehirden yola çıkan bir gezgin satıcı var.
- Bu satıcı n şehirden en kısa yolu gidecek şekilde ve her şehirden sadece bir kere geçerek, ana şehrine geri dönmek istiyor.

Tam ağırlıklı graf G = (N,A) olmak üzere, N şehirleri, A ise yolları göstermektedir. Her yol (i,j) ∈ A olmak üzere, i ve j şehirleri arasındaki uzaklık d_{ij}'dir ve i, j ∈ N' dir. Denklem 2.5'teki f(x) fonksiyonunun minimum değeri bulunmaya çalışılmaktadır.

$$f(x) = \sum_{i=1}^{n-1} d_{(i)(i+1)} + d_{(n)(1)} \tag{2.5}$$

GSP'nin değişik türleri mevcuttur. Şehirler arasındaki uzaklıkların karşılıklı olarak aynı olduğu GSP, simetrik GSP olarak adlandırılır. Şehirler arasındaki uzaklıkların karşılıklı olarak farklı olduğu GSP türü ise asimetrik GSP olarak adlandırılır (Gutin ve Punnen, 2002).

GSP'nin çözümünde iteratif yöntemler kullanılabilmektedir. Bu çözümler tüm ihtimallerin değerlendirildiği çözümler oldukları için, kesin olarak doğru sonuca ulaşabilmektedir. Fakat, şehir sayısı arttıkça, oluşacak olan ihtimal sayısı katlanarak artacağından dolayı, çözüme ulaşmak günler, aylar hatta yıllar sürebilmektedir. Günümüzde süper hızlı bilgisayarlar bile şehir sayısının çok olduğu problemleri çözmek için yıllara ihtiyaç duymaktadır.

Hızla gelişen teknoloji alanında, çözüme en hızlı şekilde ulaşmak bir gereksinim olmuştur. Az şehir sayılı problemlerde kesin çözüme ulaşmak adına iteratif yöntemler tercih edilebilirken, çok şehir sayılı problemlerde, doğru sonuca ulaşacağı kesin olmasa bile, metasezgisel çözümler tercih edilmektedir. Böylece zaman olarak daha kısa bir sürede doğru veya doğruya yakın sonuçlar ortaya çıkar.

2.3. Karesel Atama Problemi (QAP)

Lokasyonlar arasındaki mesafeler ve tesisler arasındaki akışlara göre, en iyi şekilde tesisleri mevcut lokasyonlara yerleştirme problemidir. Hedef, tüm tesisler

yerleştirildikten sonra, toplam maliyetin minimum yapılmasıdır. Yani hedef, aralarında çok akış olan tesisleri birbirine yakın, az akış olanları birbirinden uzağa yerleştirmektir. n sayıda tesis ve n sayıda lokasyon olduğu varsayılırsa, A=[a_{ij}] ve B=[b_{rs}] olmak üzere, a_{ij} i ile j lokasyonları arasındaki mesafe ve b_{rs} r ve s tesisleri arasındaki akış miktarını göstermektedir. Buna göre matematiksel olarak QAP problemi Denklem 2.6'daki $f(\emptyset)$ fonksiyonunu minimize etmeye çalışmaktadır (Stützle ve Hoos, 2000).

$$f(\emptyset) = \sum_{i=1}^{n} \sum_{j=1}^{n} b_{ij} a_{\emptyset(i)\emptyset(j)} \tag{2.6}$$

Denklem 2.6'daki $\emptyset$, tesislerin konumlara atanmasına karşılık gelen 1 ve n arasındaki permütasyondur. Yani $\emptyset$(i), $\emptyset$ içindeki i tesisinin lokasyonunu göstermektedir. $b_{ij}a_{\emptyset(i)\emptyset(j)}$ ifadesi ise, i tesisinin $\emptyset(i)$ lokasyonuna, j tesisinin $\emptyset$(j) lokasyonuna atanmasını ifade etmektedir.

QAP bir NP-hard problem türüdür ve en zor optimizasyon problemlerinden birisi olarak kabul edilir. n $\geq$ 20 olan problemler genelde optimal olarak çözülememektedir (Stützle ve Hoos, 2000).

2.4. Gezgin Satıcı Problemi Çözüm Önerileri

Ayrık niceliklerin optimal olarak düzenlenmesi, gruplanması, sıraya konulması veya seçilmesi olarak tanımlanan ayrık optimizasyon konusu üzerinde bir çok çalışma mevcuttur. Bunlardan bazıları sezgisel, bazıları ise metasezgiseldir. GSP de bu kategori içine girdiği için genel olarak kullanılan çözümler, GSP için uygundur.

2.4.1. Sayımsal çözümler

GSP, bilgisayarların tüm ihtimalleri değerlendirmesi yöntemiyle çözülebilir. Fakat verilen şehir sayısına göre, bilgisayarın hesaplayacağı çözümlerin sayısında aşırı bir artış söz konusudur. Tablo 2.1.'de n şehir sayısına bağlı olarak hesaplanması gereken ihtimal sayısı verilmiştir.

Tablo 2.1. Hesaplama iterasyon sayıları

n	**(n-1)!**
3	2
4	6
5	24
6	120
7	720
8	5040
9	40320
10	362880
20	1.216451004 E+17
30	8.841761993 E+30
40	2.039788208 E+46

Tabloda görüldüğü üzere, ihtimallerin çokluğundan dolayı, bilgisayarların kabul edilebilir bir süre içinde, fazla şehir sayısına sahip problemleri çözmesi mümkün değildir.

2.4.2. Sezgisel çözümler

Optimizasyon problemlerinin çözümünde, çeşitli çözümlerden etkili olana karar verilerek uygulanan çözümlere sezgisel çözümler denir. Bu tür algoritmalar kesin çözüme her zaman ulaşamaz, yakınsama özelliğine sahiptirler ve gerçek çözüme yakın çözümler üretmeyi sağlarlar.

Optimizasyon problemleri her zaman kesin çözüme sahip olmayabilir. Bu gibi durumlarda yakınsama yöntemiyle çözüme ulaşılır, yani sezgisel çözümler kullanılır. Sezgisel çözümlerin karar verici için basit olması da sezgisel çözüm seçimi için avantaj olarak değerlendirilebilir. Gerçek hayat optimizasyon problemleri standart matematiksel yöntemlerle çözümlenirken, gerekli olan bazı parametreler eksik veya yanlış olabilir. Böyle durumlarda çözümde yanlışlıklar olabilir. Sezgisel çözümlerin kullanılması, bu veya buna benzer yanlışlıkların oluşmasının önüne geçer (Karaboga, 2004).

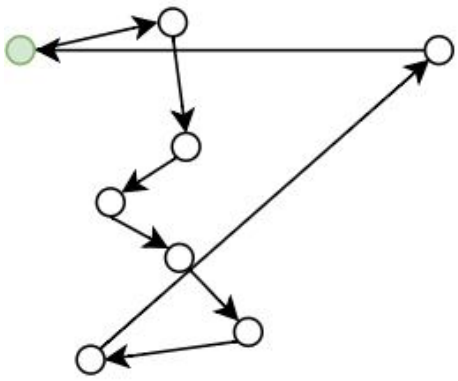
Sezgisel algoritmalar, normal algoritmalara göre daha kısa sürede doğru veya doğruya yakın çözümler sunar. Sezgisel algoritmaların uygulanması basittir ve problemlere kolay uygulanabilir. Kullanılan problemlerin değiştirilmesi durumunda,

kolay bir şekilde yeni probleme adapte olabilir. Karmaşık algoritmaların aksine, kolayca analiz edilebilirler.

Avantajlarının yanında sezgisel algoritmalar dezavantajlara da sahiptir. Algoritma, global optimum çözüm yerine, yerel optimum çözümü bulmuş ve burada takılmış olabilir. Yerel çözümün, global çözüme uzak olması veya başlangıç çözümünün yerel çözüme yakın olması, takılmanın sebeplerindendir. Böyle bir durumda başlangıç çözümünün uygun seçilmesi veya birden fazla başlangıç çözümü ile algoritmanın tekrar tekrar çalıştırılması takılmayı engelleyebilir, fakat global çözümün bulunmasını garantilemez. Birden fazla algoritmanın birlikte çalıştırılması yani hibritlenmesi de bir başka takılmama yöntemidir.

2.4.2.1. En yakın komşu algoritması

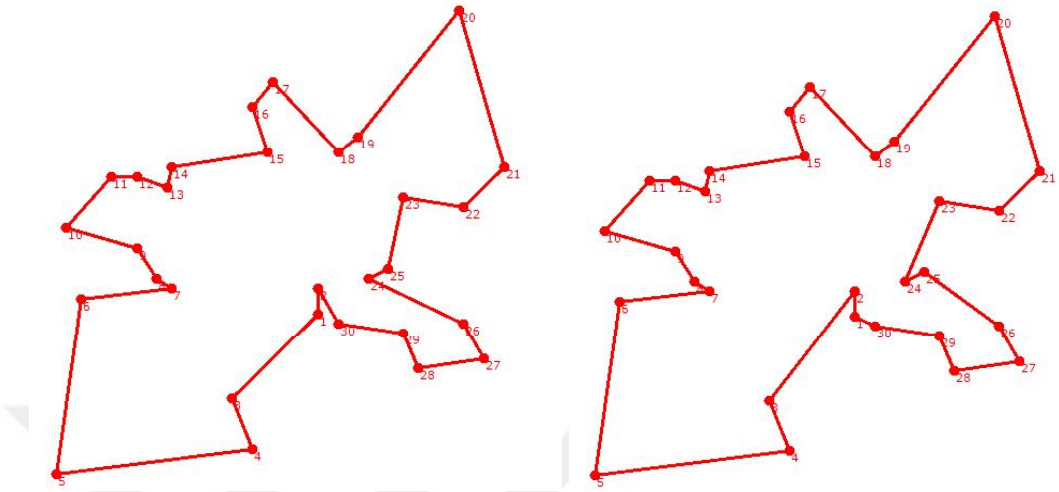
Problemin bilinen en basit çözüm algoritmalarından biridir. Bu algoritmaya göre bir başlangıç şehri seçilir ve bu şehre en yakın olan şehir ziyaret edilir ve gezilecek şehir kalmayana kadar devam edilir (Louis Bentley, 1992). Algoritma, $O(n^2)$ zaman karmaşıklığına sahiptir (Johnson ve A. McGeoch, 1997). Algoritmada değişik başlangıç şehirlerine göre, değişik sonuçlar ortaya çıkabilir. Algoritmanın daha iyi çalışması için tüm şehirlerin başlangıç olarak seçildiği iterasyonlar çalıştırılıp, tüm durumlar değerlendirilebilir. Tüm bunlara rağmen algoritmanın doğru sonuç vermesi kesin değildir. En iyi tur sonucunu vermeyen bir örnek Şekil 2.1.'de gösterilmiştir.



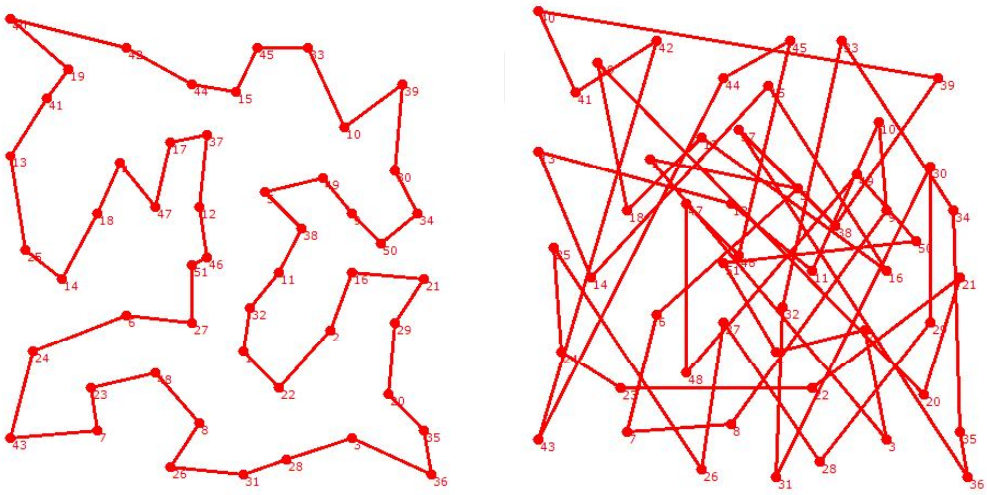
Şekil 2.1. En yakın komşu algoritmasına göre bulunan en iyi olmayan tur

Oliver30 ve Eil51 problemleri için önerilen algoritma ve en yakın komşu algoritmasıyla oluşturulan çözümler Şekil 2.2. ve Şekil 2.3.'te verilmiştir. En yakın

komşu algoritması, Oliver30 probleminde global en iyi çözüme yakın bir sonuç ortaya çıkarmasına rağmen, Eil51 probleminde çok kötü bir sonuç ortaya çıkarmıştır.



Şekil 2.2. Oliver30 için solda önerilen algoritma, sağda en yakın komşu algoritmasıyla elde edilen en iyi yol sonucu



Şekil 2.3. Eil51 için solda önerilen algoritma, sağda en yakın komşu algoritmasıyla elde edilen en iyi yol sonucu

2.4.3. Metasezgisel çözümler

Gezgin satıcı problemi için birçok metasezgisel çözüm önerildiği görülmüştür. Bu çözümlere örnek olarak, parçacık sürü algoritması (PSO), tabu arama algoritması, karınca kolonisi algoritması (KKO), yapay arı kolonisi algoritması (ABC) ve genetik algoritma (GA) gösterilebilir. Bunlar genelde sürü tabanlı teknikler olarak bilinmektedir.

Sürü tabanlı tekniklerin ortak özelliği, optimizasyon problemi çözülürken, sürü veya koloninin kendi arasında çözümler hakkında bilgi paylaşmasıdır. Örneğin, karınca sistemindeki yapay karıncalar, diğer karıncalar için feromon bırakırken, yapay arı kolonisindeki arılar dans yardımıyla birbirileriyle iletişim halindedirler. Bu nedenle, bu algoritmalarda kollektif zeka, bilginin paylaşılmasından oluşur.

2.4.3.1. Parçacık sürü optimizasyonu (PSO)

Parçaçık sürü optimizasyonu (PSO), 1995 yılında Eberhart ve Kennedy tarafından bulunmuş ve kuş ve balıkların davranışlarından esinlenilmiştir (Eberhart ve Kennedy, 1995).

PSO'daki sürü içinde tanımlanan parçacıklar, arama uzayında hareket etmektedirler. Arama uzayındaki parçacıkların pozisyonundaki değişiklikler sonucu, diğer parçacıkların da bu başarıyı taklit etme davranışlarından esinlenilmiştir. Yani her bir parçacığın yer değişimi, komşusunun deneyimi veya bilgisinden etkilenir. Bu sosyal davranışın modellenmesinin sonucu, arama uzayında başarılı bölgelere doğru geri dönmeler şeklinde olmalıdır (Engelbrecht, 2007).

Her bir iterasyonda, parçacıklar kendi hızlarını Denklem 2.7'ye göre hesaplarlar (Kang-Ping ve ark., 2003).

$$V_{id} = \omega * V_{id} + \eta_1 * rand() * (P_{id} - X_{id}) + \eta_2 * rand() * (P_{gd} - X_{id}) \qquad (2.7)$$

Denklem 2.7'deki ω hareketsizlik faktörünü, X_{id} parçacığın şuanki durumunu, P_{id} parçacığın yerel en iyi çözüm sonucunu, P_{gd} sürünün global en iyi çözüm sonucunu, η_1 ve η_2 ise lokal ve global en iyi çözüm sonucunu etkileyen ağırlığı göstermektedir. V_{id} hesaplandıktan sonra, parçacığın yeni pozisyonu Denklem 2.8'e göre hesaplanır.

$$X_{id} = X_{id} + V_{id} \qquad (2.8)$$

Genel olarak PSO ile GSP problemi çözümü için kullanılan adımlar Şekil 2.4.'te gösterilmiştir (Özsağlam, 2009).

Başlangıç {
Sürüdeki parçacıkların popülasyon, hız ve pozisyonlarını oluştur.
Durdurma kriteri olana kadar {
Her bir iterasyon tamamlanana kadar {
Bir önceki iterasyonun en iyisi ile karşılaştır, daha iyi ise yer değiştir.
} Bitir
En iyi değerleri kendi arasında karşılaştır ve en iyisini global en iyi olarak ata.
Denklem 2.7 ve Denklem 2.8'e göre hız ve popülasyon değerlerini yenile.
} Bitir
En iyi sonucu bildir.
} Bitir

Şekil 2.4. PSO ile TSP çözüm adımları

2.4.3.2. Tabu arama algoritması

Tabu arama algoritması ilk olarak F. Glover tarafından optimizasyon problemlerinin çözümü için geliştirilmiştir. Temel amacı problemin çözümünün yerel en iyi çözüme takılmasını engellemek ve global en iyi çözüme gitmesine yardım etmektir. Değerlendirme fonksiyonu yardımıyla her iterasyonda bulunan en iyi hareketin bir sonraki iterasyonda da kullanılmasını sağlar (Karaboga, 2004).

Tabu arama algoritmasında, kötü sonuç veren hareketlerin sonraki iterasyonlarda kullanılmasını engellemek için hafıza bulunmaktadır. Hafızada tutulan bu hareketlere tabu hareketler adı verilir (Dorigo ve Stützle, 2004).

Temel olarak bir tabu arama algoritması adımları Şekil 2.5.'te gösterilmiştir (Glover, 1989).

Adım 1.	Başlangıç çözümü al.
Adım 2.	Komşu çözümler bul ve bunların arasında amaç değeri en yüksek olanı en iyi olarak seç. En iyi olanı seçerken tabu listesinde olmadığından emin ol.
Adım 3.	Mevcut çözümü, yeni çözüm ile değiştir ve tabu listesini güncelle.
Adım 4.	Durdurma kriteri sağlanana kadar Adım 2'ye git ve devam et.

Şekil 2.5. Basit tabu arama adımları

2.4.3.3. Yapay arı kolonisi algoritması (ABC)

Yapay arı kolonisi, 2005 yılında Karaboğa tarafından bulunmuş ve arıların yem bulma ve dans davranışlarından esinlenilmiştir (Karaboga, 2005; Özkış ve Babalik, 2013).

Arı kolonisinde üç çeşit arı olabilir. Bunlar işçi, gözcü ve kaşif arıdır. İşçi arı, kovandan dışarı çıkarak kaynak arar ve bulduğu kaynakların uzaklığını, yönünü ve kalitesini kovandaki dans alanına gelerek, çeşitli danslar yardımıyla gözcü arılara bildirir. Bu adımdan sonra tekrar kaynak aramak için işçi arı olarak kalabileceği gibi, gözcü arı veya kaşif arıya da dönüşebilir. Gözcü arı, karar vermek için vardır. İşçi arının kendisine aktardığı bilgilere göre ne yapılacağına karar verir. Bu adımdan sonra karar vermek için bekleyebileceği gibi işçi arıya dönüşerek kaynağa gidebilir. Kaşif arı ise rastgele olarak yeni kaynakları keşfetmekten sorumludur (Akça, 2011).

Yapay arı kolonisi algoritmasında temel olarak üç önemli adım vardır.

İlk adımda, başlangıç kaynakları rastgele yerleştirilir. Tüm işçi arılar elde edilen sonuçları güncellemek için kendilerine bir önceki seçtiklerinden farklı başka bir kaynak seçerler. Seçilen kaynağın pozisyonu Denklem 2.9'a göre hesaplanır.

$$v_{ij} = x_{ij} + r(x_{ij} - x_{kj}) \tag{2.9}$$

Denklem 2.9'da belirtilen v_{ij} yeni çözümü, x_{ij} önceki çözümü, x_{kj} ise komşu çözümü göstermektedir. r, -1 ve 1 arasında rastgele bir sayıdır. $k \in \{1,2,3,\dots,SN\}, j \in \{1,2,3,\dots,D\}$ olmak üzere, SN kaynakların sayısını ve D problemin boyutunu ifade etmektedir. Yeni kaynak, eski kaynaktan daha iyi sonuca sahipse, işçi arı yeni kaynağı tercih eder, değilse eskisiyle devam eder.

İkinci adımda, her bir gözcü arı rulet tekerleği kuralıyla, işçi arılar tarafından önerilen kaynaklardan birisini seçer. Seçim olasılığını belirleyen formül Denklem 2.10'daki gibidir.

$$P_i = \frac{fit_i}{\sum fit_n} \tag{2.10}$$

Denklem 2.10'da belirtilen fit_i i'deki kaynağın uygunluk değeridir.

Son adımda, önceden belirlenen ve limit adı verilen döngü sayısı, güncellenmeyen çözümlerin güncellenmesini sağlar. Limit sayısını geçen kaynaklar terkedilir ve yeni pozisyondaki kaynaklar ile yer değiştirilir. İşçi arılar gözcü arıya dönüşür. Yeni seçilen pozisyon ise Denklem 2.11'e göre hesaplanır.

$$x_{ij} = x_{min}^{j} + rand(0,1)(x_{max}^{j} - x_{min}^{j}) \quad (2.11)$$

Denklem 2.11'de belirtilen x_{min}^{j} j boyutundaki kaynakların alt limitini, x_{max}^{j} ise üst limitini belirtmektedir (Li ve ark., 2012).

2.4.3.4. Genetik algoritma (GA)

Genetik algoritma, doğadaki canlıların hayatta kalma ve bir sonraki nesli oluşturma içgüdülerinden esinlenerek oluşturulmuştur (Holland, 1992).

Genetik algoritmada temel olarak 5 adım mevcuttur. Bunlar; kodlama, değerlendirme, çaprazlama, mutasyon ve çözümleme adımlarıdır.

Kodlama adımında, probleme uygun olarak bir kodlama türü oluşturulur. Burada önemli olan, her bir kodlanan kromozomun eşsiz yani birbirinden farklı olarak gösteriliyor olmasıdır. GSP probleminde, her bir şehir numarası farklı olduğundan, bunlar birleştirilerek bir kodlama yapılabilir. Burada önemli olan, aynı numaranın tekrarlanmasından kaçınılmasıdır.

Değerlendirme adımında, oluşturulan veya sonradan elde edilecek olan kromozomların, uygunluk değerleri hesaplanır. Bu yapılırken probleme uygun olarak seçilen bir uygunluk fonksiyonundan yararlanılır. GSP için eğer kuş uçuşu uzaklıklar kullanılacak ise, uygunluk fonksiyonu Denklem 2.12'deki gibi seçilir. Denklemde belirtilen x ve y'ler her bir şehrin pozisyonlarını göstermektedir.

$$f_{ij} = \sqrt{(x_i - x_j)^2 - (y_i - y_j)^2} \quad (2.11)$$

Çaprazlama adımında, eşleştirilecek kromozomların seçimi ve kendileri arasında gen alışverişleri yapılır. Kromozomların yarısı seçilir ve bu seçilen kromozomlardan rastgele ikisi rulet tekerleği gibi rastgele seçim yapan bir metot yardımıyla alınır. Seçilen kromozomlar arasında, belirli bir duruma göre veya rastsal şekilde gen alışverişi yapılır. GSP için burada dikkat edilecek nokta, gen alışverişi sonrasında, kromozomların içinde aynı şehirlerden bulunmamasıdır. Başka problemlerde de buna benzer kısıtlar olabilir ve bu kısıtlara uygun şekilde işlem yapılır. Çaprazlama işlemi, seçilen havuzdaki kromozomlar bitene kadar devam eder.

Mutasyon adımında, çaprazlama sonucunda elde edilen yeni kromozomların düşük olasılıkla mutasyona uğraması gerçekleştirilir. Bu adım, optimizasyon probleminin yerel bir optimal noktaya takılmasının önüne geçilmesi için yapılmaktadır. Düşük bir olasılıkla, genlerden bir veya birkaçı, mantıklı bir değişken ile değiştirilir. GSP için bu adımda, kromozom içindeki rastgele seçilen iki genin değiştirilmesi uygundur.

Çözümleme adımında, elde edilen tüm yeni kromozomların uygunluk değerleri hesaplanır. Eski kromozomlardan uygunluk değeri düşük olan varsa, yeni olanlar eskileriyle değiştirilebilir. Algoritma belirli bir adım sayısı kadar çalıştırılacaksa, adım sayısı gelene kadar çaprazlama adımından itibaren adımlar tekrarlanır. Tüm işlemler bittikten sonra ve algoritma bitiş iterasyonuna ulaştığı zaman, elde edilen en iyi uygunluk değerine sahip kromozom optimum sonuç olarak değerlendirilir. GSP için, elde edilen bu kromozom, en kısa turu vermektedir.

Genetik algoritma temel olarak Şekil 2.6.'daki adımlardan oluşur (Hussain ve ark., 2017).

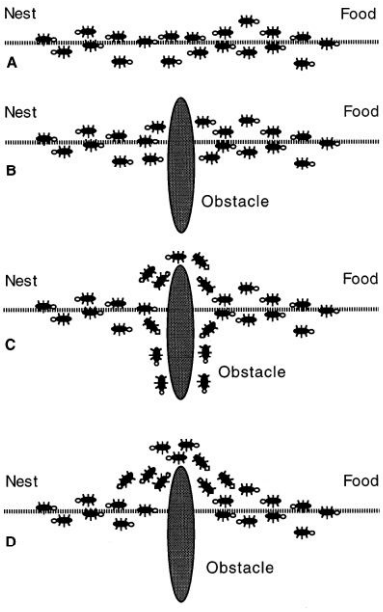
Adım 1.	Popülasyondaki P sayıda kromozomu rastgele oluştur.
Adım 2.	Her kromozomun uygunluk değerini hesapla.
Adım 3.	P/2 sayıda kromozomu seç.
Adım 4.	Seçilen kromozomlar içinden rastgele ikisini seçip çaprazla.
Adım 5.	Düşük bir olasılıkla yeni oluşan kromozoma mutasyon uygula.
Adım 6.	Tüm kromozomlar çaprazlanana kadar Adım 4 ve Adım 5'i tekrarla.
Adım 7.	Yeni popülasyon üyelerini uygunluk değeri düşük olan eskileriyle değiştir.
Adım 8.	Yeni popülasyondaki her kromozomun uygunluk değerini hesapla.
Adım 9.	Jenerasyon sayısı üst limite geldiğinde veya çözüme yaklaşıldığı anlaşıldığında, algoritmayı bitir ve sonuçları göster, diğer durumda Adım 3'e git.

Şekil 2.6. Basit genetik algoritma adımları

2.5. Karınca Kolonisi Algoritması ile Gezgin Satıcı Problemi Çözümü

Gerçek karıncalar yuvalarından yiyecek kaynağına olan en kısa yolu bulma yeteneğine sahiptirler. Ayrıca çevre koşullarının değişikliklerine karşı adapte olma yatkınlıkları da vardır. Yollarına çıkacak olan bir engel ile en kısa yol değişebilmekte ve karıncalar yeni en kısa yol arayışına girerek en kısa yollarını güncelleyebilmektedirler. Şekil 2.7.-A'da görüldüğü üzere, karıncalar düz bir hat üzerinde yiyecek kaynağından

yuvalarına hareket etmektedirler. Bu düz yolu seçmelerinin nedeni, karıncaların salgıladıkları feromon adı verilen kimyasal salgılardır. Karıncalar yürüdükleri yollarda belirli bir miktar feromon bırakırlar. Her bir karınca zengin feromon miktarının olduğu yolları tercih etme eğilimine sahiptir. Bu özellikleri, karıncaların Şekil 2.7.-B'deki gibi yollarına bir engel çıktığı zaman nasıl en kısa yolu seçtiklerinin bir göstergesidir. Karıncalar önlerine çıkan engelden dolayı, yollarına düz devam edememektedirler. Bu yüzden karıncalar, sağa veya sola dönme hareketlerinden birisini seçmek durumundadırlar. Başlangıçta hareket edecekleri yönde herhangi bir seçim unsuru bulunmadığı için, yarı olasılıkla bir yöne dönerler, yani karıncaların yarısı sağa, yarısı sola dönerler. Şekil 2.7.-C'deki gibi, karıncaların bir süre sonraki halleri bu şekilde olur. Kısa yolu seçen karıncalar, uzun yolu seçenlere göre yiyecek kaynağına daha kısa sürede gidip, yuvalarına daha kısa sürede dönmüşlerdir. Yani kısa yol üzerindeki karınca miktarı ve dolayısıyla feromon miktarı, uzun yola göre daha fazladır. Böylece Şekil 2.7.-D'deki gibi karıncaların kısa yolu seçme olasılıkları, feromon miktarının da artmasıyla artmıştır. Burada karıncaların salgıladıkları feromon miktarı ve hızları aynı kabul edilmiştir (Dorigo ve Gambardella, 1997).



Şekil 2.7. Karıncaların engele göre hareketleri (Dorigo ve Gambardella, 1997)

Karınca davranışları kısaca şu şekilde tanımlanabilir:

- Karıncalar, yuvalarından yiyecek kaynaklarına veya yiyecek kaynaklarından yuvalarına en kısa yolu bulmaya çalışmaktadırlar.

- Dış etkenler sonucu takip ettikleri mevcut yol artık en kısa yol değilse, yeni en kısa yolu çevrede bulunan birbirlerinin salgıladıkları feromon miktarları yardımıyla bulabilmektedirler.

Son yıllarda farklı optimizasyon problemlerini çözmek için parçacık tabanlı optimizasyon metotları bulunmuştur. Bu metotlarda genelde doğadan esinlenilmiştir. Örnek olarak; M. Dorigo gerçek karıncaların davranışlarından esinlenerek karınca sistemini bulmuştur. Sistemde gerçek karıncaların yuva ve yem arasındaki davranışlarından esinlenilmiştir (Dorigo ve ark., 1996).

GSP, KKA'nın geliştirilmesinde önemli bir rol oynamaktadır. Bunun için birçok sebep vardır. Bu sebepler içinde, birçok uygulamada kullanılan önemli bir NP-hard problemi olması, problemin kolay uygulanabilir olması, problemin kolay anlaşılabilir olması ve karşılaştırma yapılabilecek bir altyapıya sahip olması sayılabilir.

KKA'nın GSP çözümünde şehirden şehire dolaşan ajan adı verilen yapay karıncalar bulunmaktadır. Her bir ajan, şehirler arasındaki yollarda biriken feromonlar ve yolların uzunluklarına bağlı bir olasılık fonksiyonu yardımıyla hareket eder. Yapay karıncalar, feromon miktarını ve yol uzunluğunu baz alan bir olasılıkla seçimlerini yaparlar (Dorigo ve Gambardella, 1997).

KKA'da ilk olarak m adet yapay karınca, rastgele şehirlere yerleştirilir. Her bir adımda, karıncalar yeni bir şehre giderler ve gittikleri yoldaki feromonu arttırarak güncellerler. Bu güncelleme işlemine yerel feromon güncellemesi adı verilir. Tüm karıncalar tüm şehirleri gezdikten sonra, aralarında en kısa tura sahip olan karıncanın gezdiği yollardaki feromon miktarlarları tekrar arttırılır. Bu güncelleme işlemine global feromon güncellemesi adı verilir. Gerçek karınca davranışlarından, yapay karıncalara aktarılan üç adet fikir vardır. Bunlar;

i. Yüksek feromon miktarına sahip yolun seçilmesi,
ii. Kısa yollarda daha fazla feromon birikmesi,
iii. Karıncaların arasındaki iletişimdir.

Gerçek karınca davranışlarından farklı olarak yapay karıncaları aşağıdaki yeteneklere de sahiptir;

i. Şehirlerin kendilerine ne kadar uzaklıkta olduğunu bilmeleri,
ii. Gezdikleri şehirleri hafızalarında tutmalarıdır.

KKA, problemin türüne göre özelleştirilebilmektedir. GSP çözümü içinse temel olarak Şekil 2.8'deki adımlar izlenir.

Adım 1.	Başlangıç feromon değerleri belirlenir.
Adım 2.	Karıncalar her düğüme rastsal olarak yerleştirilir.
Adım 3.	Her karınca, sonraki şehri yerel arama olasılığına bağlı olarak(şehrin uzaklığı ve feromon miktarına göre) seçmek suretiyle turunu tamamlar.
Adım 4.	Her karınca tarafından katedilen yolların uzunluğu hesaplanır ve yerel feromon güncellemesi yapılır.
Adım 5.	En iyi çözüm hesaplanır ve global feromon yenilemesinde kullanılır.
Adım 6.	Maksimum iterasyon sayısı yada yeterlilik kriteri sağlanana kadar Adım 2'ye gidilir.

Şekil 2.8. GSP için KKA çözüm adımları

Matematiksel olarak ifade edilecek olursa, r şehrinde bulunan bir k yapay arısı, daha önce gitmediği bir s şehrine seçmek için Denklem 2.12'deki ifadeyi kullanmaktadır;

$$s = \begin{cases} arg\ \max_{u \notin M_k} \{[\tau(r,u)].[\eta(r,u)]^{ß}\} & eğer\ q \leq q_0 \\ S & diğer\ durumlar \end{cases} \quad (2.12)$$

Denklem 2.12'deki, $\tau(r,u)$, (r, u) yolundaki feromon miktarını göstermektedir. $\eta(r,u)$, r ve u şehirleri arasındaki uzaklığın tersini göstermektedir. ß, feromon miktarı ve uzaklığın etkisini gösteren parametredir. q ve q_o, 0 ile 1 arasında rastgele seçilen birer sayıdır($0 \leq q \leq 1$). M_k, karıncanın daha önce gitmiş olduğu şehirlerin listesidir. S ise fazla feromon miktarına ve yol kısalığına bağlı olan Denklem 2.12'teki olasılık fonksiyonuna göre seçilen rastgele bir değişkendir.

$$p_k(r,s) = \begin{cases} \frac{[\tau(r,s)].[\eta(r,s)]^{ß}}{\sum_{u \in M_k}[\tau(r,u)].[\eta(r,u)]^{ß}} & eğer\ s \notin M_k \\ 0 & diğer\ durumlar \end{cases} \quad (2.13)$$

Bu ifadede, $p_k(r,s)$, k karıncasının r şehrinden s şehrine gitmeyi seçme olasılığıdır.

Feromon miktarı yerel ve global olmak üzere iki şekilde değiştirilmektedir. Global değişim sırasında, tüm turlarını tamamlayan karıncalar arasından en kısa turu yapan karıncanın gittiği yollardaki feromon miktarı arttırılır. Yani en kısa yol üzerindeki feromon miktarı diğer yollara kıyasla zamanla artar. Böylece en iyi çözüm için bir kuvvetlendirme sağlanmış olur.

Yerel değişim sırasında ise karıncalar gittikleri yoldaki feromon miktarını değiştirerek, diğer tüm karıncalar tarafından güçlü tek bir yolun seçilmesinin önüne geçmektedirler. Yerel güncelleme bu koşullarda arttırıldığı gibi, buharlaşma etkisiyle azaltılmaktadır.

KKO algoritması temel halinin dışında, güçlendirilmiş başka algoritmaların da türetilmesini sağlamıştır. Bunlardan bazıları, aşağıda açıklanmıştır.

2.5.1. Min-max ant system (MMAS)

Standart KKA yönteminden 3 şekilde farklıdır. Bunlar, her döngüde sadece (iterasyondaki veya globaldeki) en iyi karıncanın yollardaki feromonu güncellemesi, feromon miktarlarının önceden belirlenen minimum ve maksimum değerlerin arasında olması ve başlangıçta tüm feromon miktarlarının maksimum olmasıdır. Feromon miktarının başlangıçta maksimum seçilmesinin nedeni, algoritmanın başlangıcından itibaren en iyi sonuca ulaşmayı hedeflemesidir (Stützle ve Hoos, 2000; Li ve Gong, 2003).

Her döngüde iterasyonun en iyi karıncasının mı yoksa globalin en iyi karıncasının mı feromon miktarlarını güncellemesi gerektiği değişiklik gösterebilir. Eğer sadece globaldeki karınca seçilirse, global en iyiye yaklaşım oranı artmaktadır, fakat yanlış bir optimal çözüm bulunması durumunda, bu noktaya takılma söz konusu olabilir. Eğer sadece iterasyonun en iyi karıncası seçilecek olursa, yanlış noktaya takılma olasılığı düşer, fakat bu durumda da optimal nokta iterasyondan iterasyona farklı değerler için bulunmaya başlar. Bu iki durumun dezavantajlarından kurtulmak için, MMAS'da dinamik bir şekilde iki ihtimal de değerlendirilir. Standart KKA'ya göre daha iyi sonuçlar ortaya koymaktadır (Stützle ve Hoos, 2000).

2.5.2. Elitist ant system (EAS)

Standart KKO algoritmasından farkı, feromon güncellemedeki farklılıklardır. Elitist karınca sistemi, algoritmanın başlangıcından itibaren, en iyi sonucu bulmak için global bilgileri kullanır. Feromon güncelleme adımının formülü Denklem 2.14'teki gibidir (Martinovic ve Bajer, 2012a).

$$\tau_{i,j} \leftarrow (1-\rho)\tau_{i,j} + \sum_{k=1}^{m} \Delta_{\tau_{i,j}}^{k} + e\Delta_{\tau_{i,j}}^{bs} \; ; i,j = 1,2,\dots,n \tag{2.14}$$

Deklem 2.14'teki $\Delta_{\tau_{i,j}}^{bs}$, algoritmanın başından itibaren, en iyi turu yapan elit karınca tarafından yollarda biriktirilen feromon miktarını göstermektedir ve Deklem 2.15'deki gibi hesaplanmaktadır.

$$\Delta_{\tau_{i,j}}^{bs} = \begin{cases} \frac{1}{L^{bs}}, eğer\ \{i,j\}\ kenarı\ T^{bs'}e\ aitse \\ 0, \qquad diğer\ durumda \end{cases} \tag{2.15}$$

Deklem 2.15'deki T^{bs} şu ana kadar yapılan en iyi turu, L^{bs} ise turun uzunluğunu ifade etmektedir. Karınca sistemi algoritmasından farklı olarak, global en iyi çözüm için güçlü feromon birikimi yapılmaktadır. Böylece karıncaların en iyi turu bulmaya karşı yatkınlıkları artmaktadır.

2.6. Yerel İyileştirme Sağlayan Yaklaşımlar ve Komşuluk Operatörleri

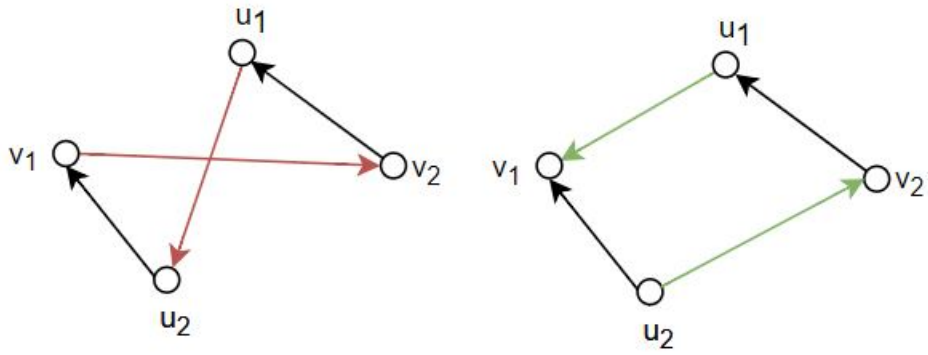
GSP'de yerel iyileştirme sağlayan bazı yaklaşımlar mevcuttur. Bu yaklaşımlar, yinelemeli olarak bazı değişim işlemleri gerçekleştirerek, iyileştirilmiş sonuçlar bulmaya çalışır (Martinovic ve Bajer, 2012b).

2.6.1. Lin-Kernighan komşuluğu

Shen Lin ve Brian Kernighan tarafından 1973 yılında bulunan bir komşuluk yöntemidir (Lin ve Kernighan, 1973). Simetrik GSP için optimal veya optimale yakın çözümler elde etmek için etkili bir yöntemdir (Helsgaun, 2000). 2-Opt ve 3-Opt gibi başka algoritmaların temelini oluşturmaktadır. Lin-Kernighan komşuluğu uyarlanabilir-dir ve her iterasyonda en kısa yolu bulmak için kaç tane yolun değiştirilmesine ihtiyaç olduğuna karar verir.

2.6.2. 2-Opt komşuluğu

2-Opt algoritması, GSP üzerinde uygulanan en temel yerel arama algoritmalarından birisidir. 2-Opt algoritması, kendisine verilen bir başlangıç turu ile çalışmaya başlar ve turdaki iki kenarı başka iki kenar ile değiştiren ardışık iyileştirmeler yaparak turu kademeli olarak iyileştirmeye çalışır. 2-Opt'da genellikle birbiri üzerinde çaprazlama yapmış olan kenarları bulmaya ve bunları düzeltmeye çalışılır. Örnekle açıklanacak olursa, 2-Opt algoritması, u_1, u_2, v_1, v_2 turundaki $\{u_1, u_2\}$ ve $\{v_1, v_2\}$ kenarlarını seçer ve bunları kendisi arasında yer değiştirir. Bu yer değiştirme sonucunda, $\{u_1, v_1\}$ ve $\{u_2, v_2\}$ kenarları elde edilir. Böylece elde edilen yeni tur, u_1, v_1, u_2, v_2 olur. Eğer elde edilen yeni tur uzunluğu, eski tur uzunluğundan kısa ise, yeni tur en iyi çözüm olarak kabul edilir. Bahsedilen durum, Şekil 2.9.'da gösterilmiştir (Englert ve ark., 2014).



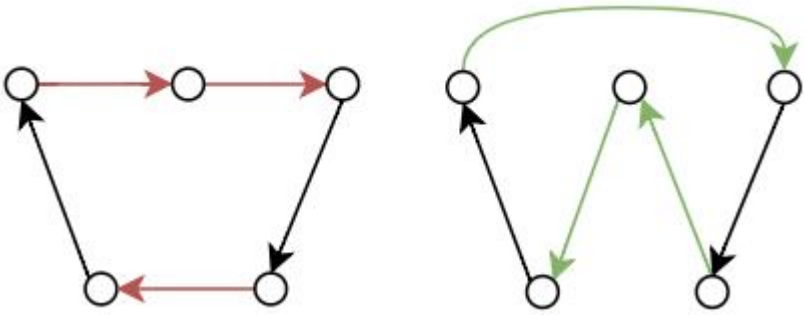
Şekil 2.9. 2-Opt operasyonu için başlangıç turu ve elde edilen yeni tur

2-Opt algoritması sonucunda, her zaman değiştirilen kenarlar ile daha iyi sonuçlar bulunmaz. Çünkü gerçekleştirilen yer değiştirme işlemi olasılığa dayanmaktadır ve bu olasılık sonucunda tur içinde bulunan daha iyi kenarlar, daha kötü kenarlar ile yer değiştirilebilir (Okada ve ark., 1998).

2.6.3. 2.5-Opt komşuluğu

2.5-Opt algoritması, 2-Opt algoritmasının güçlendirilmiş, 3-Opt algoritmasının ise kısıtlanmış bir versiyonudur. 2-5 Opt algoritması, 2-Opt yer değiştirmesi yapılırken, aynı zamanda şehir ve ondan sonra gelen şehir arasına bir başka şehir yerleştirilmesini de sağlar. Deneysel sonuçlar, 2.5-Opt algoritmasının, 2-Opt algoritmasından daha iyi, 3-

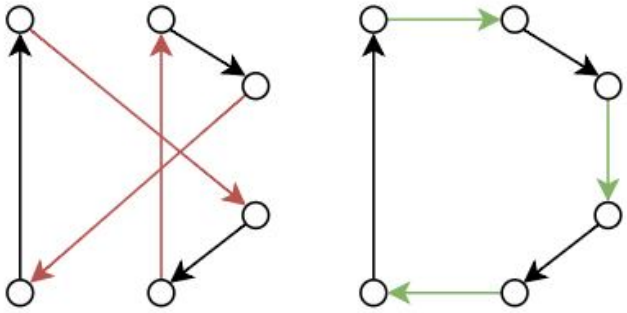
Opt algoritmasından daha kötü olduğunu göstermektedir (Dorigo ve Stützle, 2004). Şekil 2.10.'da bir örnek gösterilmektedir.



Şekil 2.10. 2.5-Opt operasyonu için başlangıç turu ve elde edilen yeni tur

2.6.4. 3-Opt komşuluğu

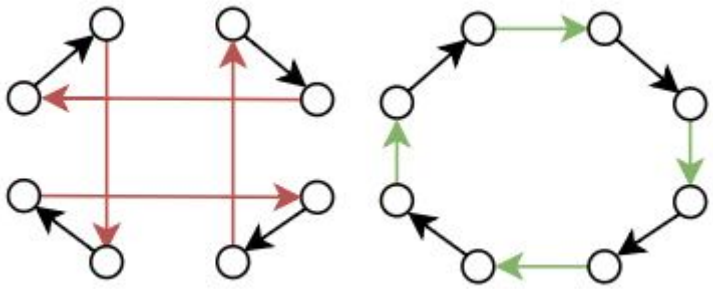
3-Opt algoritması, kendisine verilen bir başlangıç turu ile çalışmaya başlar ve turdaki üç kenarı başka üç kenar ile değiştiren ardışık iyileştirmeler yaparak turu kademeli olarak iyileştirmeye çalışır. Paralel şekilde çalıştırılarak, algoritmanın hızlandırılması da mümkündür (Johnson ve A. McGeoch, 1997). Şekil 2.11.'de bir örnek gösterilmektedir.



Şekil 2.11. 3-Opt operasyonu için başlangıç turu ve elde edilen yeni tur

2.6.5. k-Opt komşuluğu

k-Opt komşuluğu, 2-Opt ve 3-Opt'dan farklı olarak daha fazla değişime olanak sağlar. k tane kenarı başka k tane kenar ile değiştirerek turu iyileştirmeye çalışır($2 \leq k \leq n$). $O(n^k)$ zaman karmaşıklığına sahiptir. Daha iyi çözümler elde edebilmek için, algoritma içinde k sayısı dinamik bir şekilde seçilebilir. k=4 için bir örnek Şekil 2.12.'de gösterilmiştir.

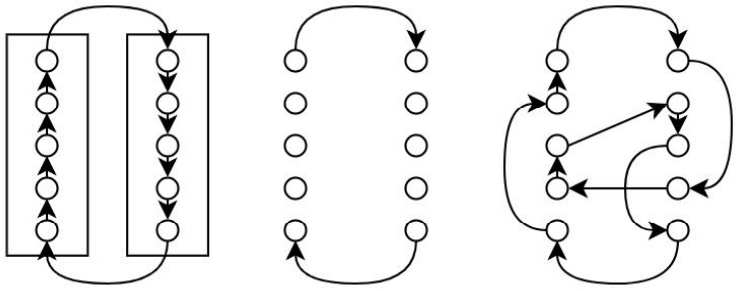


Şekil 2.12. k=4 için k-Opt operasyonuna verilen başlangıç turu ve elde edilen yeni tur

2.6.6. HyperOpt komşuluğu

Asimetrik GSP üzerinde uygulanan yerel iyileştirme sağlayan bir algoritmadır. Yerel optimumları aşmayı ve yeni kaliteli turlar oluşturmayı sağladığı belirtilmektedir. Problemi alt problemlere bölmek ve daha sonrasında da bunları optimize etme prensibiyle çalışmaktadır. Başlangıç ve bitiş yollarını silmeyecek şekilde, geri kalan yolların kendi arasında dinamik şekilde değiştirilmesiyle yeni turlar elde edilir (Burke ve ark., 2001).

Baskı devre kartlarının üretiminde ortaya çıkan sıralama probleminden ilham almıştır. Baskı devre kartlarının üretiminde komponentlerin alımı ve sonrasında kart üzerinde yerleştirilmesi esastır. Bu durumda bu problem türünde, standart GSP probleminden farklı olarak iki farklı grup(komponent alma bölümü ve karta yerleştirme bölümü) bulunmaktadır. Soldaki grup komponent alma, sağdaki grup karta yerleştirme bölümü olarak düşünülecek olursa, yapılan HyperOpt işlemi Şekil 2.13.'te gösterilmiştir.

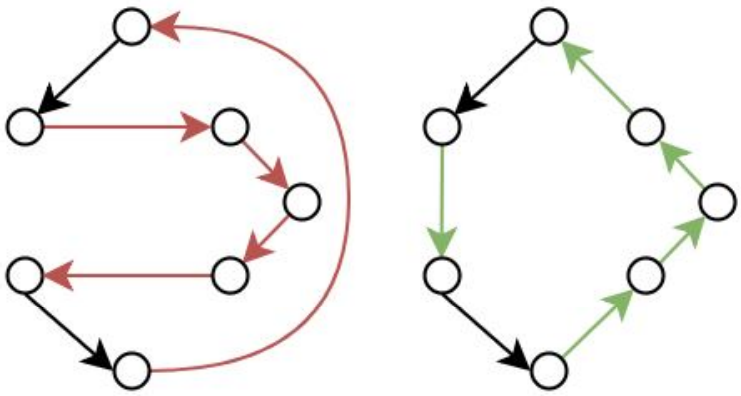


Şekil 2.13. HyperOpt operasyonuna verilen başlangıç turu ve elde edilen yeni tur

2.6.7. Or-Opt komşuluğu

1976 yılında İlhan Or tarafından sunulmuştur. Or-Opt yöntemi, geçerli çözümden komşu bir çözüme art arda ilerleyerek uygulanabilir çözümlerin bir alt kümesini araştıran yerel bir arama tekniğidir. Her bir çözüm için, bir şehirden ulaşılabilecek tüm konfigürasyonlardan oluşan bir komşuluk tanımlanır. Arama operatörü, daha iyi bir çözüm için tanımlanmış komşuluğu arar. Bulduktan sonra süreç yeni konfigürasyondan yeniden başlar. Geçerli konfigürasyona bağlı olan komşulukta daha fazla iyileştirme yapılamayana kadar tekrarlanır (Li ve Alidaee, 2002).

3-Opt algoritmasının kısıtlı bir versiyonudur ve bu yüzden bazı durumlarda hızlı çalışır, fakat 3-Opt algoritmasından daha kötü sonuçlar verir. Belirtilen kısıtlama, koparılacak olan yollardan ikisinin, turda birbirine yakın olması gerektiğidir. Ne kadar yakın olacağı ise algoritmaya parametre olarak verilir. Parametrenin 3 şehir olarak verildiği bir örnek Şekil 2.14.'te gösterilmiştir.



Şekil 2.14. 3 şehir için Or-Opt operasyonuna verilen başlangıç turu ve elde edilen yeni tur

2.6.8. Pareto yerel arama

Pareto yerel arama yöntemi herhangi bir hedef toplama işlemi veya sayısal parametre gerektirmez. Tüm turların her bir komşuluğu araştırılır ve eğer komşuluk, çözüm içinde baskın bir role sahipse, komşuluk bir listeye eklenir ve eğer başka turlarda mevcut değilse, bu turlara dahil edilmeye çalışılır. Yöntem, baskın olan komşuluklar bulunamayana kadar devam ettirilir (Lust ve Teghem, 2010).

3. MATERYAL VE YÖNTEM

Bu tez konusunda, bir ayrık optimizasyon problemi olan gezgin satıcı problemi üzerinde durulmuştur ve bu probleme uygun çözüm olarak değiştirilmiş ve geliştirilmiş bir karınca kolonisi algoritması sunulmaya çalışılmıştır. Karınca kolonisi algoritması, gezgin satıcı probleminin ana hedefi olan en kısa yolu bulmayı hedefleyen bir algoritma olduğu için bu tür problemlerde sık sık kullanılmaktadır.

Problemin tanımının basit olması ve bu konuda bilinen problemler ve çözümlerinin mevcut olmasından dolayı bu tezde bu problem ele alınmıştır. Böylece kıyaslama ve doğruluk konusunda sıkıntı yaşanmaması öngörülmüştür.

Tezde sunulacak olan çözümde temel hatlarıyla karınca kolonisi algoritması kullanılacak ve gezgin satıcı problemi çözülmeye çalışılacaktır. Farklı olarak "feromon" yerine "ayak izi" mekanizması kullanılacak ve feromonun buharlaşması etkeni ortadan kaldırılıp, değişmeyen ve kollektif olarak artan ayak izi miktarı ile çözüm bulunmaya çalışılacaktır. Ayrıca elde edilen genel çözümün iyileştirilmesi için literatürde "Komşuluk Operatörleri" olarak bilinen operatörler kullanılacak ve yerel iyileştirmeler yapılmaya çalışılacaktır.

Algoritma iki farklı adımdan oluşacaktır. Bu adımlar "Tur Oluşturma Adımı" ve "Tur Geliştirme Adımı" olarak belirlenmiştir.

3.1. Tur Oluşturma Adımı

Bu adımdaki amaç, ayak izi ve şehirler arası mesafelere göre turlar oluşturmaktır. Bu adımdaki karıncalara oluşturucu karıncalar denilmektedir. Karıncalar şehirlere rastgele şekilde yerleştirilmektedir. Her bir karıncanın başka bir şehri seçme olasılığı ise Denklem 3.1'e göre hesaplanmaktadır.

$$P_{ij} = \frac{{F_{ij}}^{a} \; x \; [(\frac{1}{D_{ij}})]^{b}}{\sum_{k=1}^{N} {F_{ij}}^{a} \; x \; [(\frac{1}{D_{ij}})]^{b}} \tag{3.1}$$

i. şehirde bulunan karıncanın, j. şehire gitme olasılığı P_{ij}'dir. F_{ij}, i. ve j. şehir arasındaki ayak izi miktarını, D_{ij} ise uzaklığı belirtmektedir. N, gezilmemiş şehir

miktarını, a ve b ise önemli parametreleri belirtmektedir. Bu tez için, a = 1, b = 5 olarak seçilmiştir. Bu fonsiyon karınca kolonisi optimizasyon tekniğinde geçiş kuralı olarak kullanılmaktadır (Dorigo ve ark., 1996). Rulet tekerleği yöntemiyle, olasılık fonksiyonunda belirlenen olasılıkların seçimi yapılmaktadır. Böylece her karıncanın bir sonraki gideceği şehir belirlenmektedir. Tüm şehirler dolaşıldıktan sonra karınca, tekrar ilk şehire geri gelmektedir. Fonksiyona göre, seçim mekanizması, hem kollektif veri olan ayak izini, hem de şehirler arasındaki uzaklık verisini kullanmaktadır.

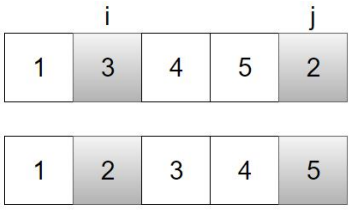
Ayak izi mekanizması, kollektif zekanın oluşmasında önemli bir faktördür. Başlangıçta her bir şehir arasına belirli bir miktar ayak izi bırakılır. Her bir karınca, turlarını bitirdikten sonra, eğer tüm karıncaların ortalama tur uzunluğundan daha iyi bir sonuca sahipse, gezdiği şehirler arasındaki ayak izi miktarını 1 arttırır, yani gezdiği şehirler arasına ayak izini bırakır.

3.2. Tur Geliştirme Adımı

Bu adımdaki karıncalara geliştirici karıncalar denilmektedir. Tur oluşturma aşamasında elde edilen en iyi tur, daha önceki iterasyonlarda elde edilen global en iyi tur ile karşılaştırılır. En iyi olan tura, geliştirici karıncalar yardımıyla geliştirme işlemleri uygulanır (Kıran ve ark., 2013). Geliştirme için komşuluk operatörleri kullanılır. Bu tezde, 3 çeşit komşuluk operatörü kullanılmıştır. 3 farklı komşuluk operatöründen hangisinin uygulanacağı rastgele bir şekilde seçilmektedir. Eğer komşuluk operatörleri sonucu elde edilen tur, eski turdan daha iyi sonuca sahipse, en iyi tur olarak kabul edilir ve kullanılır.

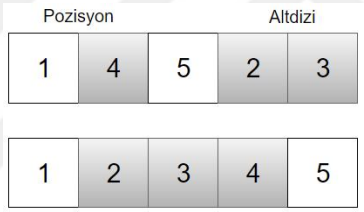
Bu tez için kullanılan komşuluk operatörleri, rastgele yerleştirme (random insertion (RI)), altdizileri rastgele yerleştirme (random insertion of subsequences (RIS)), altdizilerin rastgele yerleştirilmesini terse çevirme (reverse random insertion of subsequences (RRIS))'dir. Bu komşuluk operatörleri her bir geliştirici karınca tarafından eşit olasılıkta rastgele seçilerek çözüme uygulanmaktadır. Bunların dışında rastgele değiştirme (random swap (RS)), aldizileri rastgele değiştirme (random swap of subsequences (RSS)), altdizileri rastgele terse çevirme (random reversing of subsequences (RRS)) ve altdizilerin rastgele değiştirilmesini ters çevirme (random reversing swap of subsequences (RRSS)) komşuluk operatörleri de mevcuttur (Kıran ve ark., 2013).

Rastgele yerleştirme operatörü, rastgele seçilen şehri, rastgele seçilen pozisyona koyar ve geri kalan şehirleri öteler. Örnek olarak rastgele seçilen pozisyon i=2, ve rastgele seçilen şehir j=5 (i ≠ j) olmak üzere, Şekil 3.1.'de ilk durum üstte, sonraki durum ise altta olacak şekilde gösterilmiştir.



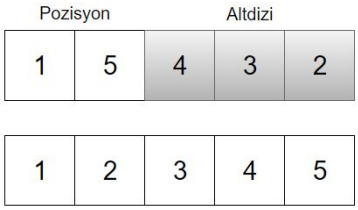
Şekil 3.1. Örnek rastgele yerleştirme işlemi

Altdizileri rastgele yerleştirme operatörü, rastgele seçilen altdiziyi, rastgele seçilen pozisyona koyar ve geri kalan şehirleri öteler. Örnek bir uygulama, Şekil 3.2.'de gösterilmiştir.



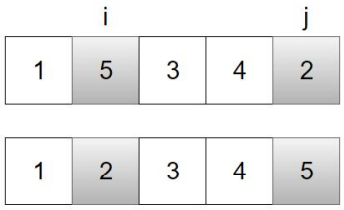
Şekil 3.2. Örnek altdizileri rastgele yerleştirme işlemi

Altdizilerin rastgele yerleştirilmesini terse çevirme operatörü, aldizileri rastgele yerleştirme operatörü gibi çalışır. Tek farkı %50 olasılıkla, yer değiştirilen altdizi, kendi içinde terse çevrilir. Örnek bir uygulama, Şekil 3.3.'te gösterilmiştir.



Şekil 3.3. Örnek altdizilerin rastgele yerleştirilmesini terse çevirme işlemi

Rastgele değiştirme operatörü, rastgele seçilen iki şehrin yerlerini değiştirir. Örnek olarak rastgele seçilen birinci pozisyon i=2 ve ikinci pozisyon j=5 (i ≠ j) olmak üzere, Şekil 3.4.'de ilk durum üstte, sonraki durum ise altta olacak şekilde gösterilmiştir.



Şekil 3.4. Örnek rastgele değiştirme işlemi

Aldizileri rastgele değiştirme operatörü, rastgele seçilen iki altdizinin yerlerini değiştirir. Örnek bir uygulama, Şekil 3.5.'te gösterilmiştir.

Şekil 3.5. Örnek altdizileri rastgele değiştirme işlemi

Altdizileri rastgele terse çevirme operatörü, rastgele seçilen bir altdiziyi, kendi içinde ters çevirir. Örnek bir uygulama Şekil 3.6.'da gösterilmiştir.

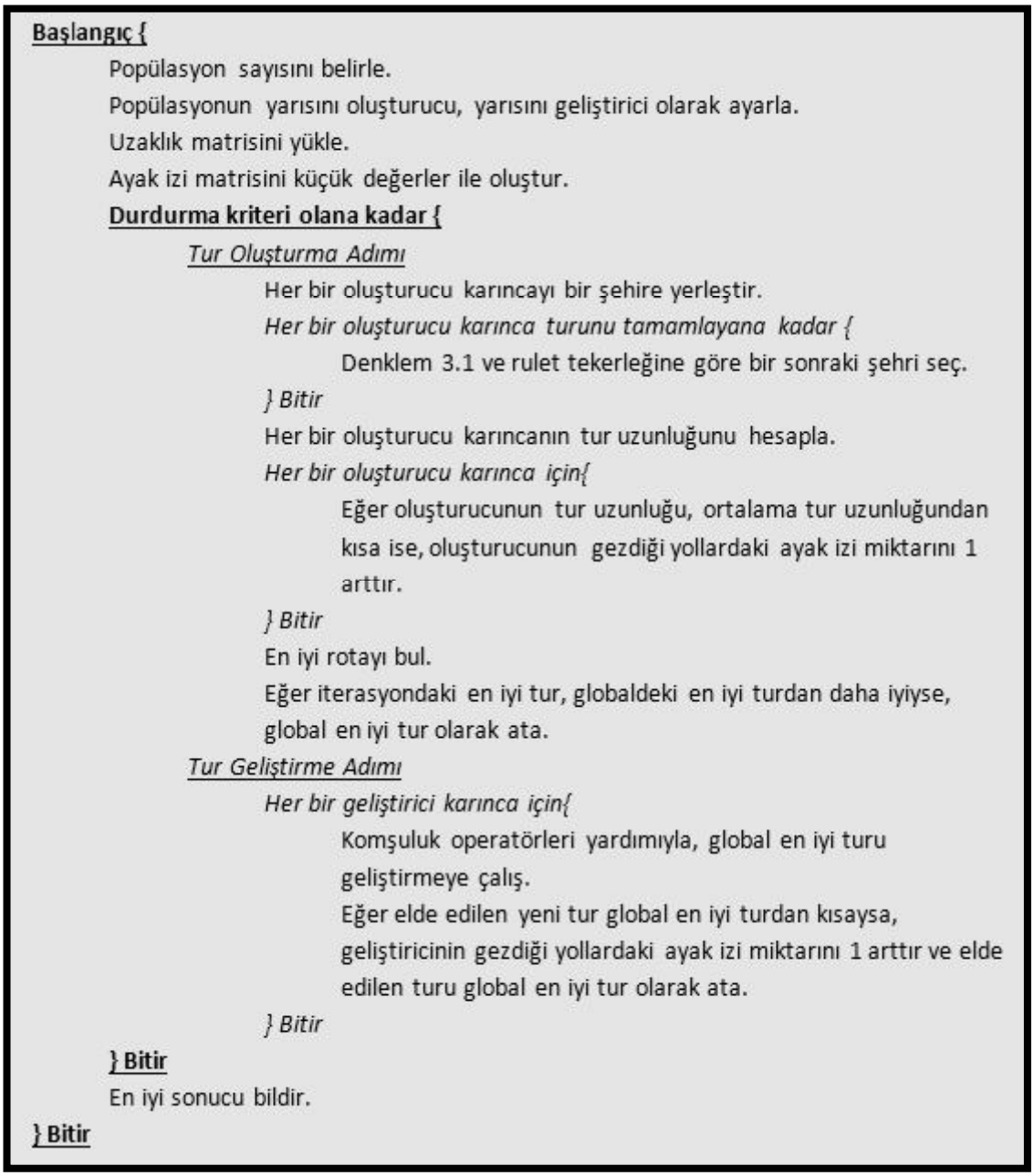
Şekil 3.6. Örnek altdizileri rastgele terse çevirme işlemi

Altdizilerin rastgele değiştirilmesini terse çevirme operatörü, aldizileri rastgele değiştirme operatörü gibi çalışır. Tek farkı %50 olasılıkla, yer değiştirilen altdizi, kendi içinde terse çevrilir. Örnek bir uygulama, Şekil 3.7.'de gösterilmiştir. Burada 1. ve 2. altdizi yer değiştirilmiş ve ayrıca 1. altdizi kendi içinde terse çevrilmiştir.

Şekil 3.7. Örnek altdizilerin rastgele değiştirilmesini terse çevirme işlemi

3.3. Algoritmanın Zaman Karmaşıklığı

Kullanılacak olan algoritmanın kısaca çalışma şekli Şekil 3.8.'de gösterilmektedir.



Şekil 3.8. Önerilen algoritma adımları

Şekil 3.8.'de görüldüğü üzere, algoritma temel olarak 3 adımdan oluştmaktadır. Bunlar, "Başlangıç Adımı", "Tur Oluşturma Adımı", "Tur Geliştirme Adımı"dır. Oluşturucu ve geliştirici karıncaların sayısını N dersek, şehir sayısı N'nin iki katı kadardır.

Başlangıç adımında beş farklı süreç vardır. Bunlar, popülasyon sayısını belirleme, oluşturucuların sayısını belirleme, geliştiricilerin sayısını belirleme, ayak izi matrisini oluşturma ve uzaklık matrisini yüklemedir. Bu adımın zaman karmaşıklığı her bir işlemin belirleme aşaması olmasından dolayı, W(Başlangıç adımı) = O(5)'dir.

Tur oluşturma adımındaki her bir oluşturucu, kendi turlarını tamamlamaktadır. Yani her bir N adet oluşturucu 2xN adet şehiri gezmektedir. Bu durumda, W(Tur oluşturma adımı) = O(2xNxN) = O(N^2) olmaktadır.

Tur geliştirme adımındaki her bir geliştirici, sadece bir kez işlem yapmaktadır. Böylece, W(Tur geliştirme adımı) = O(N) olmaktadır.

Genel olarak zaman karmaşıklığı hesaplanacak olursa, W(Algoritma) = max{W(Başlangıç adımı), W(Tur oluşturma adımı), W(Tur geliştirme adımı)}

= O(N^2) olmaktadır.

4. HİBRİT YAKLAŞIMIN ÇOK BİLİNEN PROBLEMLERE, TÜRKİYE'DEKİ İLLERE ve KONYA'DAKİ İLÇELERE UYGULANMASI

4.1. Çok Bilinen Problemler

Bu bölümde algoritma, birden fazla çok bilinen GSP üzerine uygulanmıştır. Tablo 4.1.'de üzerinde çalışılan problemlerin şehir sayıları ve optimum tur uzunlukları verilmiştir. Çalışılan bu problemler TSPLIB kütüphanesinden alınmıştır.

Tablo 4.1. Çok bilinen problemlerin bilinen optimum tur uzunlukları

Problem	**Şehir sayısı**	**Optimum tur uzunluğu**
Oliver	30	423.74
Eil	51	428.87
Berlin	52	7544.37
St	70	677.11
Pr	76	108159.44
Eil	76	545.39
Kroa	100	21285.44
Eil	101	642.31
Ch	150	6532.28
TSP	225	3859.00

Tablo 4.2.'de KKO ve önerilen algoritmanın kullandığı parametreler verilmiştir. Buradaki D parametresi, GSP'deki şehir sayısıdır. P, popülasyon büyüklüğünü göstermektedir. α ve ß değerleri, her iki algoritmada da bulunmaktadır ve karıncaların şehir seçme olasılıklarındaki önemli üstsel parametrelerdir. ρ değeri, KKO algoritmasında feromon buharlaşma oranını göstermektedir. Tezde önerilen algoritmada ise feromon yerine ayak izi kullanıldığı için ve ayak izleri yok olmadıkları için, bu parametre bulunmamaktadır.

Tablo 4.2. Kullanılan parametre değerleri

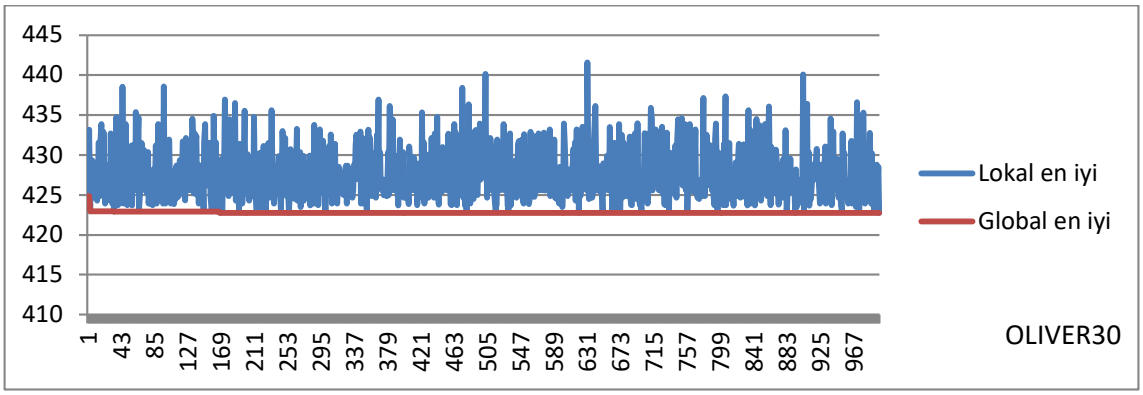
Parametre	KKO	Önerilen Algoritma
Popülasyon Büyüklüğü(P)	D	D
Maksimum Döngü Sayısı	500	500
Alfa(α)	1.0	1.0
Beta(ß)	5.0	5.0
Rho(ρ)	0.65	-

Tablo 4.3.'te ise standart KKA'nın ve tezde önerilen algoritmanın çok bilinen problemlere uygulanması ve bulunan en iyi mesafeler gösterilmiştir. Her iki algoritma da aynı sistemde, Oliver30, Eil51, Berlin52, St70, Pr76, Eil76, Kroa100, Eil101, Ch150, Tsp225 problemlerine uygulanmıştır ve 100 iterasyon olarak çalıştırılmıştır.

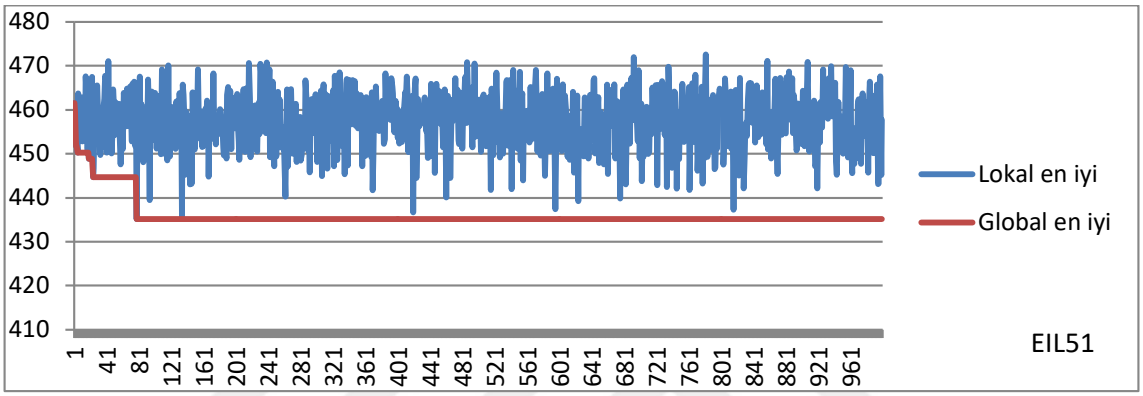
Tablo 4.3. Algoritma tur uzunluğu kıyaslama grafiği

	Karınca Kolonisi Algoritması				Önerilen Algoritma			
Problem	En iyi	En kötü	Ortalama	Standart Sapma	En iyi	En kötü	Ortalama	Standart Sapma
Oliver30	423.74	429.36	424.68	1.4053	423.74	423.95	**423.75**	0.4424
Eil51	450.59	463.55	457.86	4.0745	431.17	450.17	**439.80**	7.0247
Berlin52	7548.99	7681.75	7659.30	31.553	7544.37	7765.62	**7583.26**	134.3872
St70	696.05	725.26	709.16	8.2671	679.37	712.63	**697.36**	12.47421
Pr76	115166.66	118227.41	116321.22	885.79	110629.72	119697.34	**115207.45**	1994.596
Eil76	554.46	568.62	561.98	3.4982	548.33	571.10	**560.44**	5.616653
Kroa100	22455.89	23365.46	22880.12	235.17	21709.47	22231.07	**21990.09**	331.6127
Eil101	678.04	705.65	693.42	6.8023	660.73	690.13	**679.81**	8.563091
Ch150	6648.51	6726.27	**6702.87**	20.733	6621.79	6777.61	6710.64	54.09038
Tsp225	4112.35	4236.84	4176.08	28.339	4079.36	4219.52	**4160.11**	51.14434

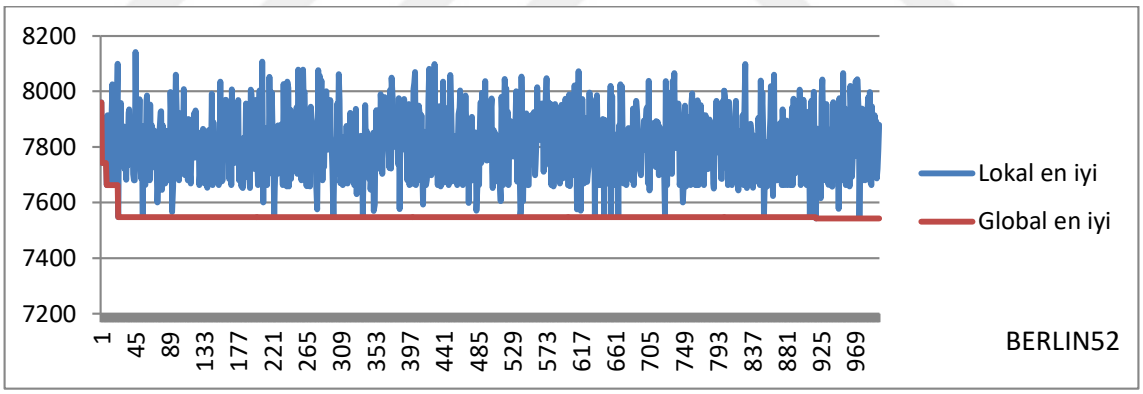
Tezde önerilen algoritmanın, Oliver30, Eil51, Berlin52, St70, Pr76, Eil76, Kroa100, Eil101 problemlerinin 1000 iterasyon sonucunda nasıl global en iyi sonuca yaklaştıkları Şekil 4.1.'den Şekil 4.8.'e kadar gösterilmiştir.



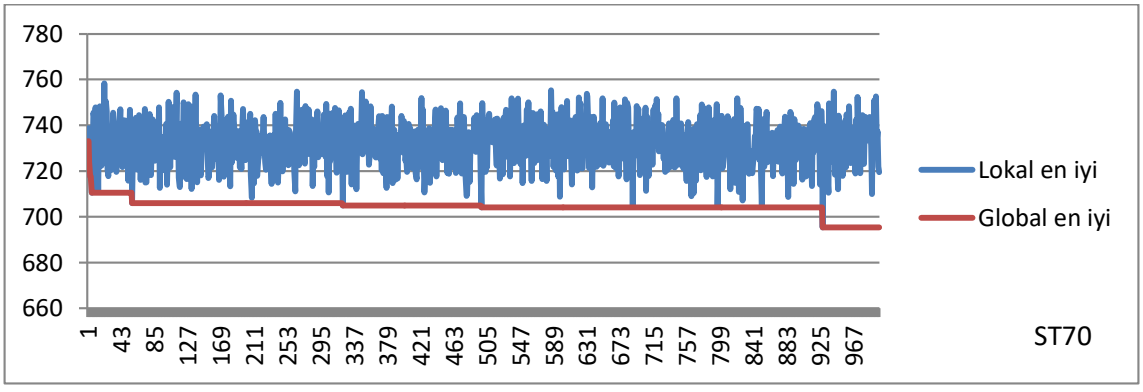
Şekil 4.1. Oliver30 için 1000 iterasyonda yerel en iyi ve global en iyi değer grafikleri



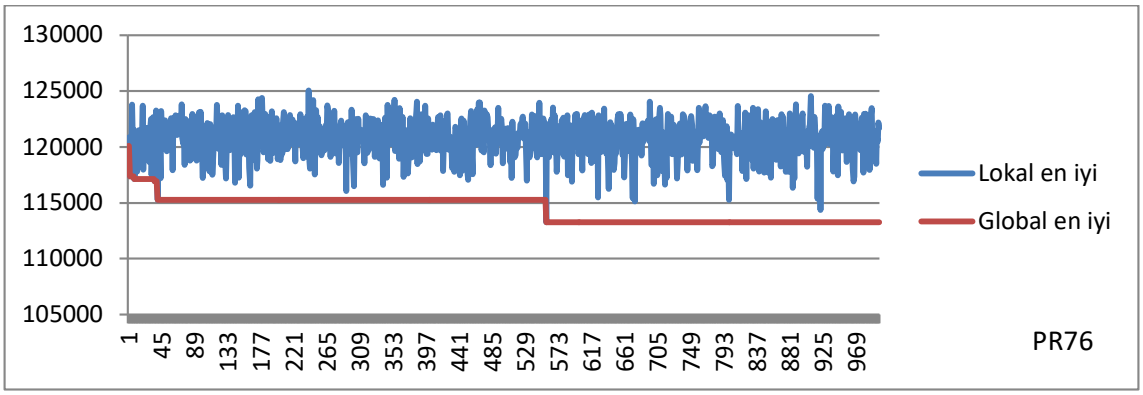
Şekil 4.2. Eil51 için 1000 iterasyonda yerel en iyi ve global en iyi değer grafikleri



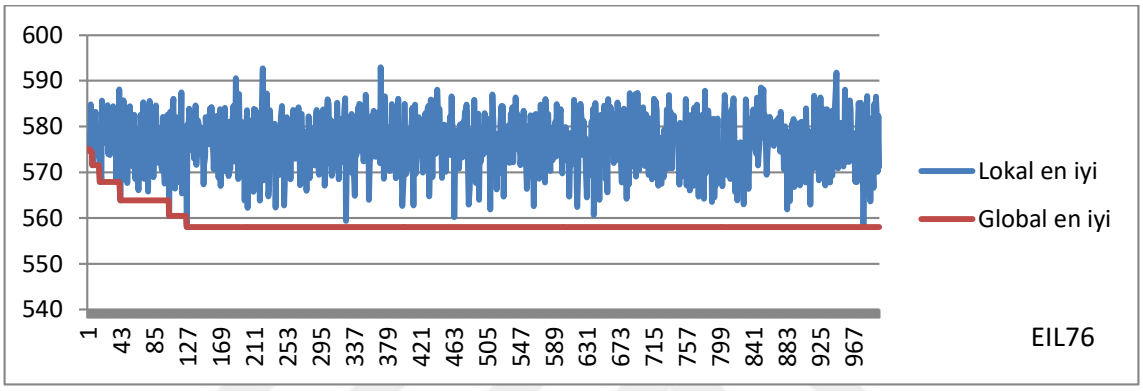
Şekil 4.3. Berlin52 için 1000 iterasyonda yerel en iyi ve global en iyi değer grafikleri



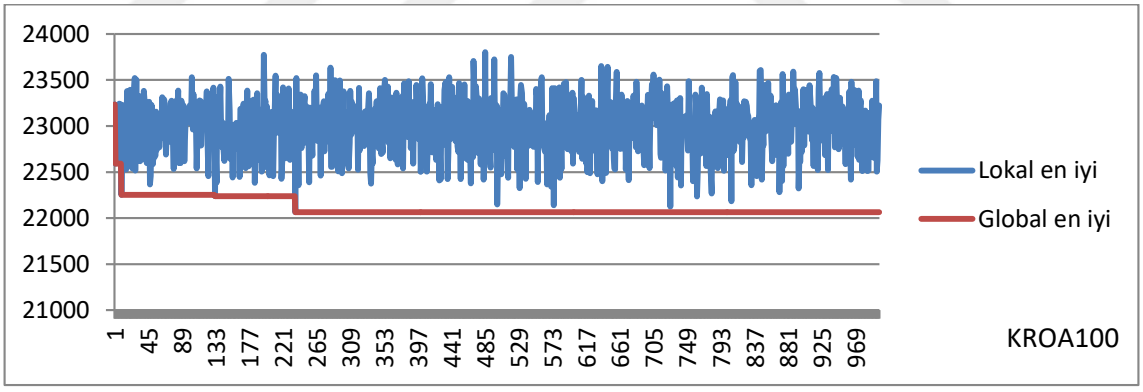
Şekil 4.4. St70 için 1000 iterasyonda yerel en iyi ve global en iyi değer grafikleri



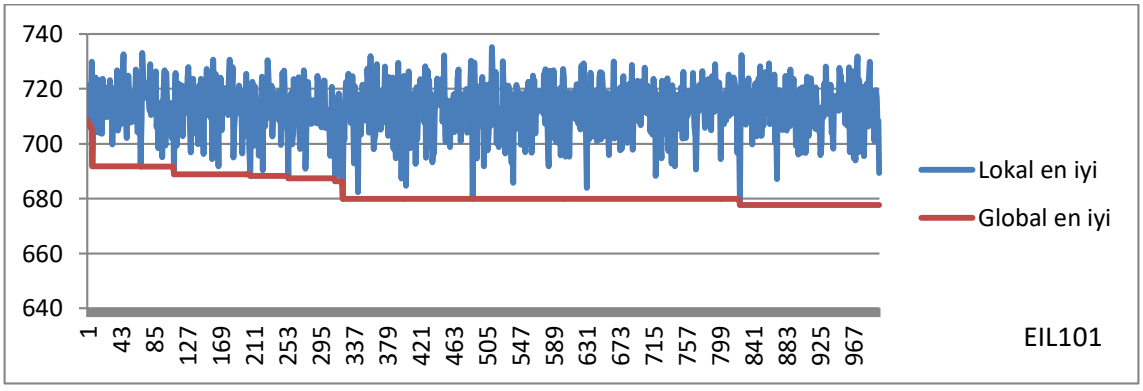
Şekil 4.5. Pr76 için 1000 iterasyonda yerel en iyi ve global en iyi değer grafikleri



Şekil 4.6. Eil76 için 1000 iterasyonda yerel en iyi ve global en iyi değer grafikleri



Şekil 4.7. Kroa100 için 1000 iterasyonda yerel en iyi ve global en iyi değer grafikleri



Şekil 4.8. Eil101 için 1000 iterasyonda yerel en iyi ve global en iyi değer grafikleri

4.2. Türkiye'nin İlleri

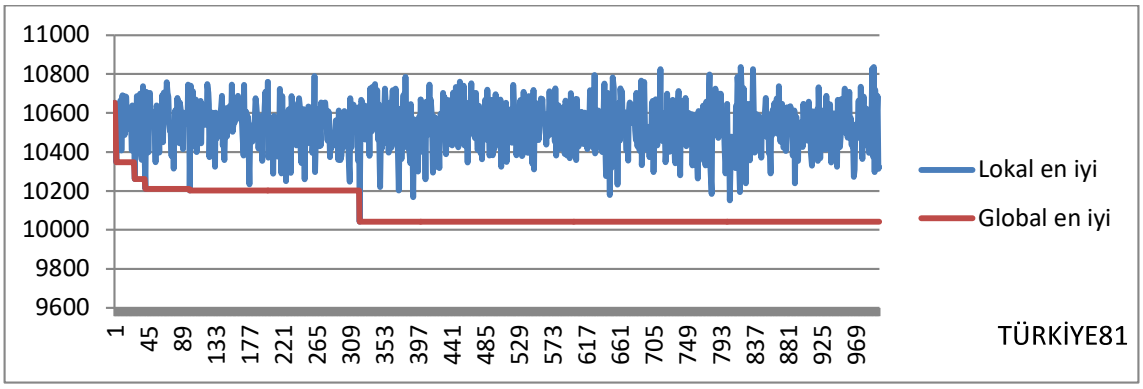
Algoritma, Türkiye'nin 81 iline uygulanmıştır. İller arasındaki mesafeler Karayolları Genel Müdürlüğü (KGM) internet sitesinden alınan 01.01.2018 tarihinde güncellenen mesafelerdir. (http://www.kgm.gov.tr/SiteCollectionDocuments/KGMdocuments/Root/Uzakliklar/ilmesafe.xls)

Türkiye'nin 81 ili için, standart GSP problemlerinden farklı bir yapı mevcuttur. GSP problemlerinde, satıcı sadece bir kez aynı şehre uğrayabiliyorken, Türkiye'nin illerinde aynı ilden birden fazla kere geçebilmektedir. Fakat, satıcı, belirlenen il merkezlerine sadece bir kere uğramakta, eğer aynı ilden birden fazla geçecekse, il merkezinden uzakta bulunan farklı yolları tercih etmektedir. Bu farklılık, problemin çözümünde bir değişikliğe sebep olmamaktadır. Genel olarak istenen şey olan, Türkiye'nin 81 ilinin en kısa tur uzunluğu ile gezilmesi problemi başarılı bir şekilde çözümlenmiştir.

Önerilen algoritmanın 1000 kere çalıştırılmasıyla Türkiye'nin 81 ilinin gezilmesi sonucu elde edilen en iyi, en kötü ve ortalama mesafe değerleri ve algoritmanın çalışma süreleri Tablo 4.4.'te verilmiştir. Bu iterasyonların en iyi değerleri ve global en iyi sonuca ulaşma eğrileri, Şekil 4.9.'da gösterilmiştir.

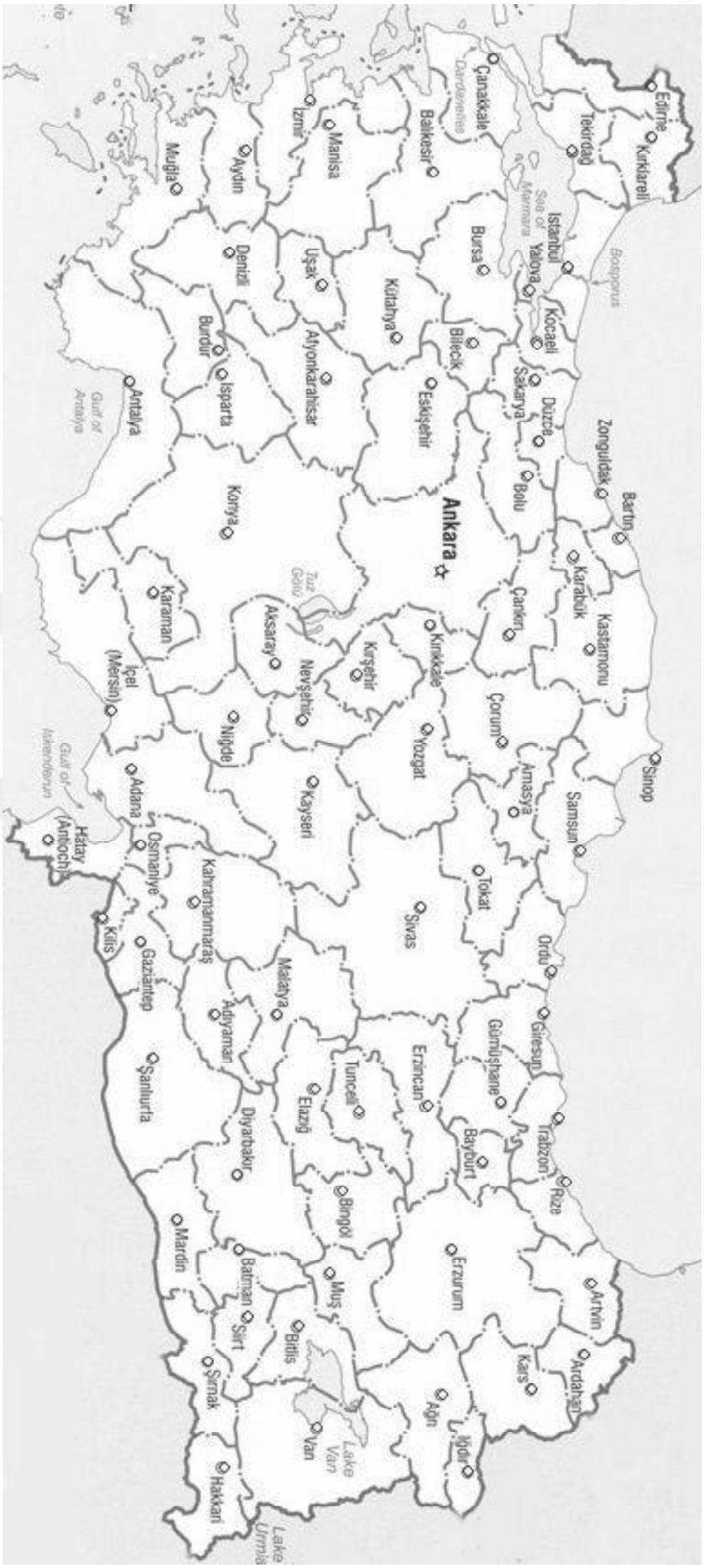
Tablo 4.4. 1000 çalıştırma sonucunda Türkiye'nin 81 ili için en iyi, en kötü ve ortalama mesafe ve süre değerleri

	Mesafe (km)	**Süre (sn)**
En iyi	10042	7,7715
En kötü	10837	9,863828
Ortalama	10314,92	8,163422
Standart Sapma	116,6166	0,249584

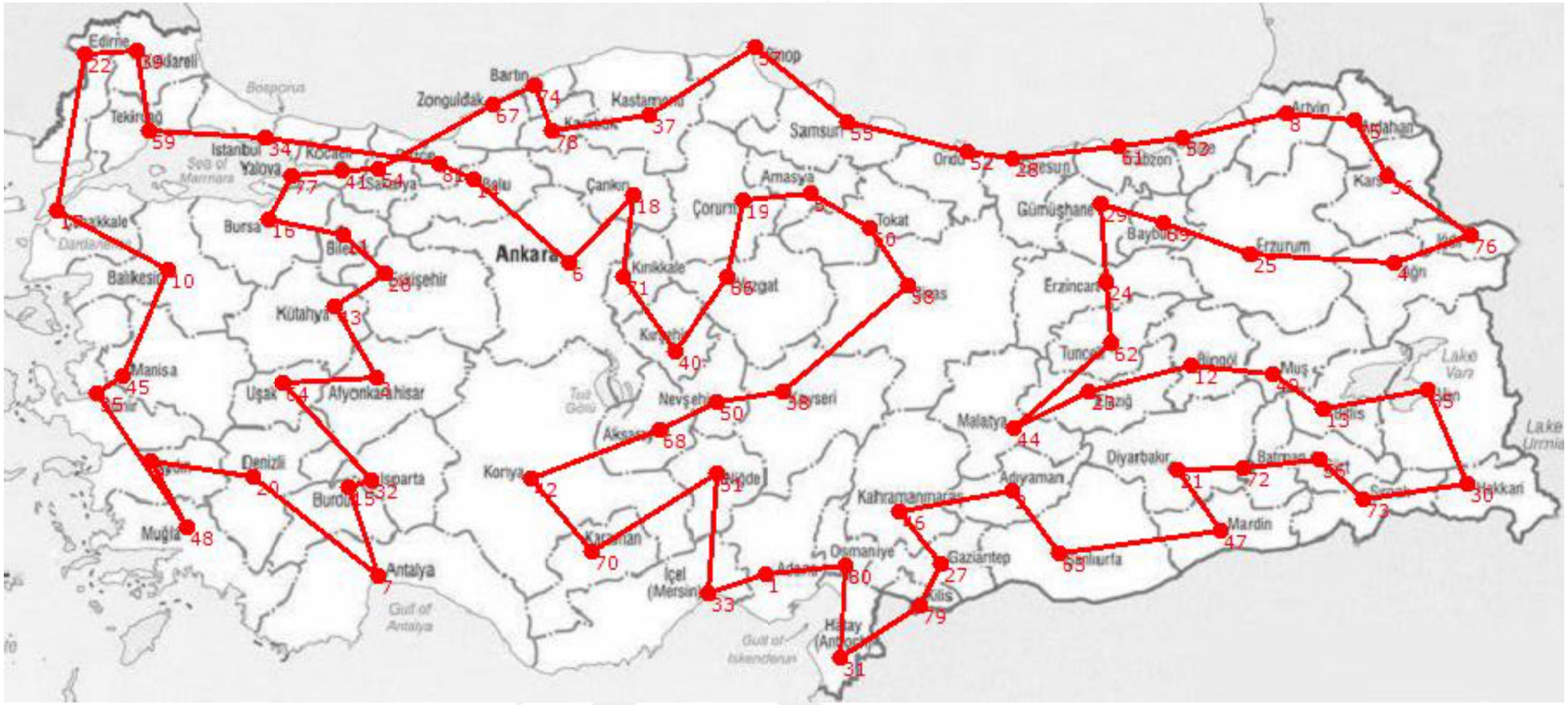


Şekil 4.9. Türkiye'nin 81 ili için 1000 iterasyonda yerel en iyi ve global en iyi değer grafikleri

Türkiye haritası ve elde edilen en iyi sonuç haritası Şekil 4.10. ve Şekil 4.11.'de verilmiştir. Şekil 4.11.'de görüldüğü üzere, İstanbul'dan Düzce'ye gidebilmek için Kocaeli'nden geçmek gerekmektedir, fakat Kocaeli merkezine uğramadan geçiş işlemi yapılmıştır. Kocaeli merkezine ise başka bir hat üzerinden gelinmiştir. Bu da göstermektedir ki, şehir merkezlerinden başka şehir merkezlerine gitmek için, aradaki şehrin merkezine uğramadan, alternatif yollar üzerinden gidildiği varsayılmıştır.



Şekil 4.10. Türkiye'nin illeri haritası(Özsağlam, 2009)



Şekil 4.11. Önerilen algoritma ile bulunan Türkiye illerinin en kısa turu

4.3. Konya'nın İlçeleri

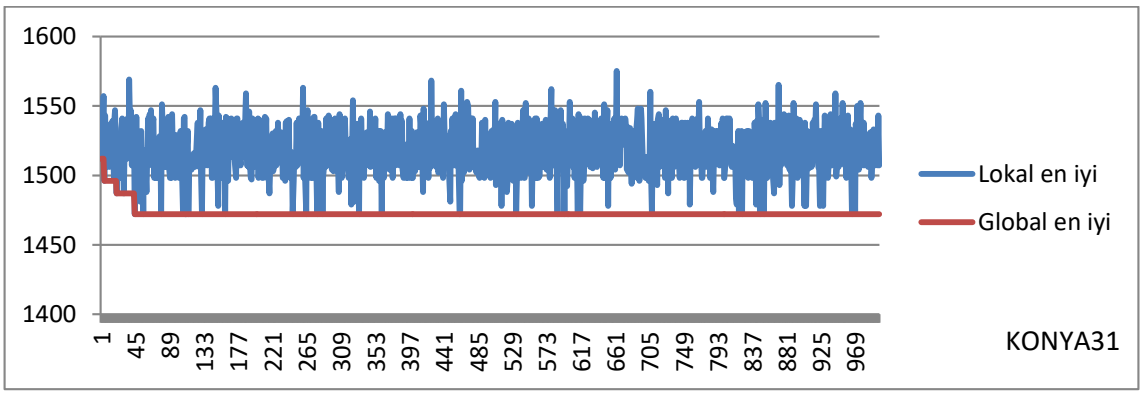
Algoritma, Konya'nın 31 ilçesine uygulanmıştır. İlçeler arasındaki mesafeler Karayolları Genel Müdürlüğü (KGM) internet sitesinden alınmıştır. (http://www.kgm.gov.tr/Sayfalar/KGM/SiteTr/Uzakliklar/ilcedenIlceyeMesafe.aspx)

Konya'nın 31 ilçesi için, standart GSP problemlerinden farklı bir yapı mevcuttur. GSP problemlerinde, satıcı sadece bir kez aynı şehre uğrayabiliyorken, Konya'nın ilçelerinde aynı ilçeden birden fazla kere geçebilmektedir. Fakat, satıcı, belirlenen ilçe merkezlerine sadece bir kere uğramakta, eğer aynı ilçeden birden fazla geçecekse, ilçe merkezinden uzakta bulunan farklı yolları tercih etmektedir. Bu farklılık, problemin çözümünde bir değişikliğe sebep olmamaktadır. Genel olarak istenen şey olan, Konya'nın 31 ilçesinin en kısa tur uzunluğu ile gezilmesi problemi başarılı bir şekilde çözümlenmiştir.

Önerilen algoritmanın 1000 kere çalıştırılmasıyla, Konya'nın 31 ilçesinin gezilmesi sonucu elde edilen en iyi, en kötü ve ortalama mesafe değerleri ve algoritmanın çalışma süreleri Tablo 4.5.'te verilmiştir. Bu iterasyonların en iyi değerleri ve global en iyi sonuca ulaşma eğrileri, Şekil 4.12.'de gösterilmiştir.

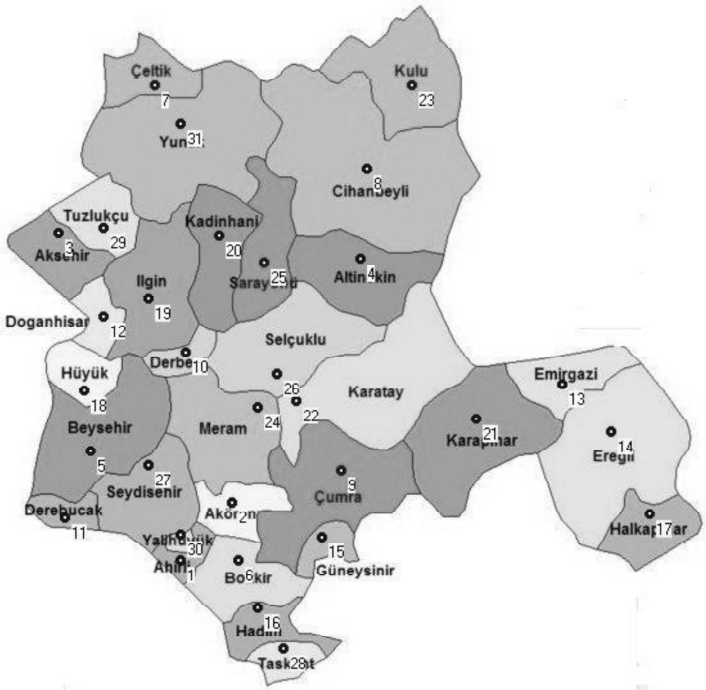
Tablo 4.5. 1000 çalıştırma sonucunda Konya'nın 31 ilçesi için en iyi, en kötü ve ortalama mesafe ve süre değerleri

	Mesafe (km)	**Süre (sn)**
En iyi	1472	0,271355
En kötü	1575	0,323154
Ortalama	1516,104	0,282071
Standart Sapma	19,05593	0,005053

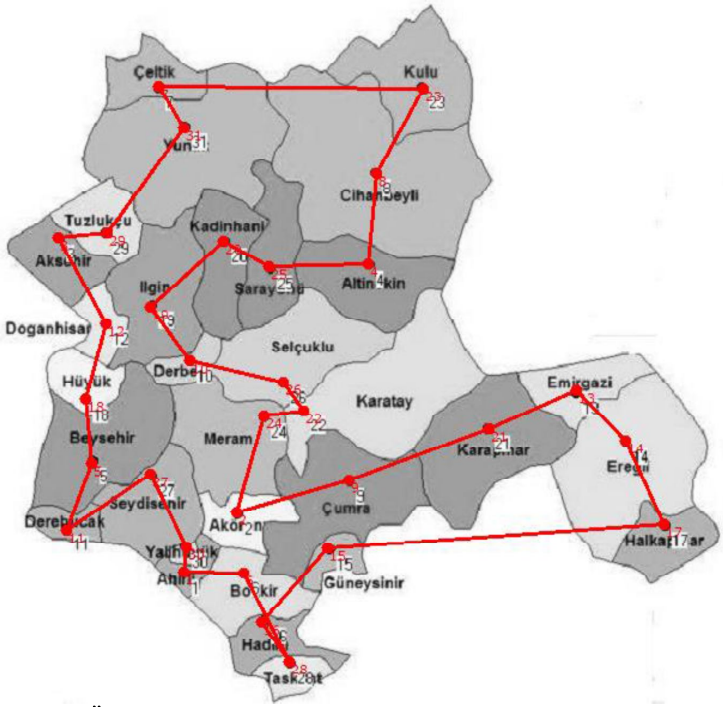


Şekil 4.12. Konya'nın 31 ilçesi için 1000 iterasyonda yerel en iyi ve global en iyi değer grafikleri

Konya haritası ve elde edilen en iyi sonuç haritası Şekil 4.13. ve Şekil 4.14.'te verilmiştir.



Şekil 4.13. Konya'nın ilçeleri haritası (Özsağlam, 2009)



Şekil 4.14. Önerilen algoritma ile bulunan Konya ilçelerinin en kısa turu

5. SONUÇ VE ÖNERİLER

KKO algoritması optimizasyon problemlerinde kısa sürede iyi sonuçlar vermesi sebebiyle sıkça kullanılır. Bu çalışmada, karınca kolonisi optimizasyonu temel alınarak değiştirilmiş ve komşuluk operatörleri ile hibritlenerek algoritmadaki buharlaşan feromon mekanizması yerine sabit kalan ayak izleri tercih edilmiştir. Yeni geliştirilen hibrit algoritma sonuçları standart KKO ile kıyaslanmıştır.

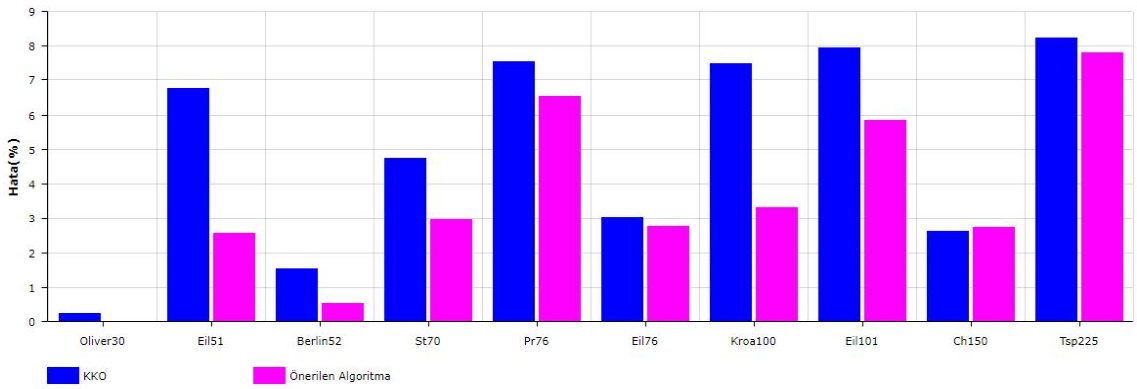
Hibrit algoritma oluştururken şu noktalara dikkat edilmiştir;

- Satıcının tüm şehirlere sadece bir kere uğrayacak şekilde turlaması
- Satıcının turlama esnasında en kısa yolu kullanması
- Problemin çözümünün mümkün olan en kısa sürede tamamlanması
- Oluşturulacak çözümün, standart çözüme göre mesafe ve süre bakımından daha iyi sonuçlar vermesi.

Yeni algoritma ile standart KKO algoritması karşılaştırılırken, bu iki metodun hata oranları kıyaslanmıştır. Hata oranı, algoritmaların çalışmalarındaki doğruluk oranlarını gösteren bir değeri ifade eder ve Denklem 5.1'deki gibi hesaplanır. Formüldeki B değeri algoritma tarafından bulunan mesafeyi, O ise optimum mesafeyi göstermektedir.

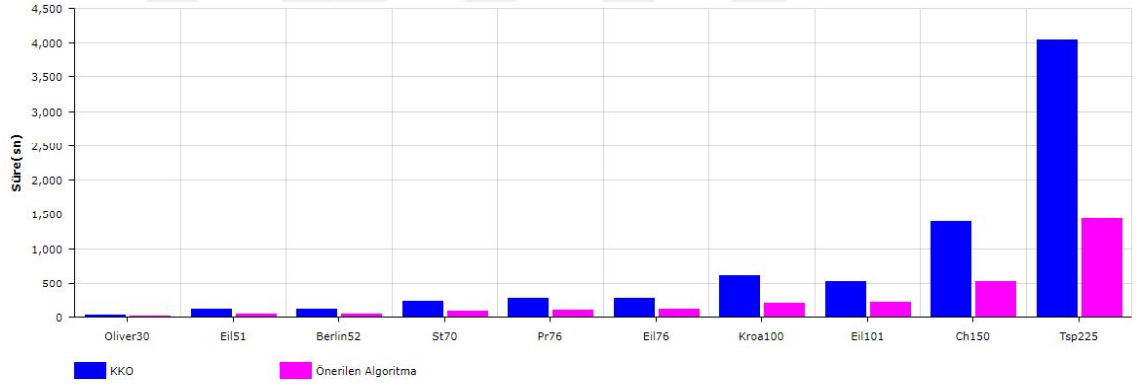
$$Hata = \frac{B - O}{O}\ x\ 100 \tag{5.1}$$

KKO ve tezde önerilen algoritmanın çok bilinen gezici problemleri çözümündeki hata oranları karşılaştırılmıştır. Çok bilinen gezgin satıcı problemlerinden sadece Ch150' de KKO daha düşük bir hata oranı göstermiş olsa da iki metodun hata oranı farkı çok düşüktür. Oliver30, Eil51, Berlin52, St70, Pr76, Eil76, Kroa100, Eil101, Tsp225 problemlerinde yeni yaklaşımla geliştirilen hibrit algoritma daha düşük hata oranlarına sahiptir. KKO ve tezde önerilen algoritmanın hata oranları kıyaslaması Şekil 5.1.'de gösterilmiştir.



Şekil 5.1. Algoritma hata oranı kıyaslama grafiği

Bu çalışmada oluşturulan hibrit algoritmanın daha kısa sürede sonuç vermesi dikkat edilen noktalardan birisidir. KKO ve tezde önerilen algoritmanın çalışma süreleri saniye olarak ölçülmüştür. Bu iki algoritmanın çalışma süreleri kıyaslaması Şekil 5.2.'de gösterilmiştir.



Şekil 5.2. Algoritma çalışma süresi kıyaslama grafiği

Hibrit algoritma çok bilinen problemlerden başka, Türkiye'nin 81 iline ve Konya'nın 31 ilçesine uygulanmıştır. Buna göre, önerilen algoritmanın 1000 kere çalıştırılması sonucunda, Türkiye'nin 81 ili için en kısa mesafe 10042 kilometre olarak bulunmuştur. Algoritma bu mesafeyi 8,46 saniyede hesaplamıştır. Konya'nın 31 ilçesi için en kısa mesafe 1472 kilometre olarak bulunmuştur. Algoritma bu mesafeyi 0,28923 saniyede hesaplamıştır.

Önerilen algoritmanın tur geliştirme adımında kullanılan komşuluk operatörleri çeşitlendirilerek daha iyi sonuçlar elde edilip edilmediği kontrol edilebilir.

KAYNAKLAR

Akça, M. R., 2011, Yapay Arı Kolonisi Algoritması Kullanılarak Gezgin Satıcı Probleminin Türkiyedeki İl ve İlçe Merkezlerine Uygulanması, *Selçuk Üniversitesi*, Konya, 90.

Beckers, R., Deneubourg, J. L. and Goss, S., 1992, Trails and U-Turns in the Selection of a Path by the Ant Lasius-Niger, *Journal of Theoretical Biology*, 159 (4), 397-415.

Burke, E. K., Cowling, P. I. and Keuthen, R., 2001, Effective local and guided variable neighbourhood search methods for the asymmetric travelling salesman problem, *Applications of Evolutionary Computing, Proceedings*, 2037, 203-212.

Cook, W. J., 2012, In Pursuit of the Traveling Salesman
Mathematics at the Limits of Computation, Princeton University Press, p. 272.

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

Dorigo, M. and Gambardella, L. M., 1997, Ant colonies for the travelling salesman problem, *Biosystems*, 43 (2), 73-81.

Dorigo, M. and Stützle, T., 2004, Ant Colony Optimization, Bradford Company, p. 305.

Eberhart, R. and Kennedy, J., 1995, A new optimizer using particle swarm theory, *Micro Machine and Human Science, 1995. MHS '95., Proceedings of the Sixth International Symposium on*, 39-43.

Engelbrecht, A. P., 2007, Computational Intelligence: An Introduction, Wiley Publishing, p. 628.

Englert, M., Roglin, H. and Vocking, B., 2014, Worst Case and Probabilistic Analysis of the 2-Opt Algorithm for the TSP, *Algorithmica*, 68 (1), 190-264.

Glover, F., 1989, Tabu Search—Part I, *ORSA Journal on Computing*, 1 (3), 190-206.

Gutin, G. and Punnen, A., 2002, The Traveling Salesman Problem and Its Variations, Kluwer Academic Publishers, p. 40.

Held, M., Hoffman, A. J., Johnson, E. L. and Wolfe, P., 1984, Aspects of the Traveling Salesman Problem, *Ibm Journal of Research and Development*, 28 (4), 476-486.

Helsgaun, K., 2000, An effective implementation of the Lin-Kernighan traveling salesman heuristic, *European Journal of Operational Research*, 126 (1), 106-130.

Holland, J. H., 1992, Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence, MIT Press, p. 228.

Hussain, A., Muhammad, Y. S., Nauman Sajid, M., Hussain, I., Mohamd Shoukry, A. ve Gani, S., 2017, Genetic Algorithm for Traveling Salesman Problem with Modified Cycle Crossover Operator, *Computational Intelligence and Neuroscience*, 2017, 7.

Johnson, D. and A. McGeoch, L., 1997, The Traveling Salesman Problem: A Case Study in Local Optimization, p. 103.

Kang-Ping, W., Lan, H., Chun-Guang, Z. and Wei, P., 2003, Particle swarm optimization for traveling salesman problem, *Proceedings of the 2003 International Conference on Machine Learning and Cybernetics (IEEE Cat. No.03EX693)*, 1583-1585 Vol.1583.

Karaboga, D., 2004, Yapay zeka optimizasyon algoritmalari : tabu arastirma, isil islem, genetik, karinca koloni, bagisiklik ve diferansiyel gelisim algoritmalari, *Istanbul*, Atlas, p. 246.

Karaboga, D., 2005, An Idea Based on Honey Bee Swarm for Numerical Optimization, Technical Report - TR06, 10.

Kıran, M. S., İşcan, H. ve Gündüz, M., 2013, The analysis of discrete artificial bee colony algorithm with neighborhood operator on traveling salesman problem, *Neural Computing and Applications*, 23 (1), 9-21.

Li, L., Cheng, Y., Tan, L. and Niu, B., 2012, A Discrete Artificial Bee Colony Algorithm for TSP Problem, Berlin, Heidelberg, 566-573.

Li, W. Q. and Alidaee, B., 2002, Dynamics of local search heuristics for the traveling salesman problem, *Ieee Transactions on Systems Man and Cybernetics Part a-Systems and Humans*, 32 (2), 173-184.

Li, Y. and Gong, S. H., 2003, Dynamic ant colony optimisation for TSP, *International Journal of Advanced Manufacturing Technology*, 22 (7-8), 528-533.

Lin, S. and Kernighan, B. W., 1973, An Effective Heuristic Algorithm for the Traveling-Salesman Problem, *Operations Research*, 21 (2), 498-516.

Louis Bentley, J., 1992, Fast Algorithms for Geometric Traveling Salesman Problems, 4, 387-411.

Lust, T. and Teghem, J., 2010, Two-phase Pareto local search for the biobjective traveling salesman problem, *Journal of Heuristics*, 16 (3), 475-510.

Martinovic, G. and Bajer, D., 2012a, Elitist Ant System with 2-opt Local Search for the Traveling Salesman Problem, 12, 25-32.

Martinovic, G. and Bajer, D., 2012b, Elitist Ant System with 2-opt Local Search for the Traveling Salesman Problem, *Advances in Electrical and Computer Engineering*, 12 (1), 25-32.

Okada, M., Taji, K. and Fukushima, M., 1998, Probabilistic analysis of 2-opt for travelling salesman problems, *International Journal of Systems Science*, 29 (3), 297-310.

Özkış, A. ve Babalik, A., 2013, Performance Analysis of ADL-ABC Algorithm on Continuous Optimization Problems.

Özsağlam, M. Y., 2009, Parçacık Sürü Optimizasyonu Algoritmasının Gezgin Satıcı Problemine Uygulanması ve Performansının İncelenmesi, *Selçuk Üniversitesi*, Konya, 131.

Stützle, T. and Hoos, H. H., 2000, MAX–MIN Ant System, *Future Generation Computer Systems*, 16 (8), 889-914.

ÖZGEÇMİŞ

KİŞİSEL BİLGİLER

Adı Soyadı : Batuhan Saygın ARSLAN
Uyruğu : T.C.
Doğum Yeri ve Tarihi : Elazığ, 1991
Telefon : 05374808833
Faks : -
e-mail : batuhanarslan@gmail.com

EĞİTİM

Derece		Adı, İlçe, İl	Bitirme Yılı
Lise	:	Selçuklu Anadolu Lisesi, KONYA	2009
Üniversite	:	TOBB Ekonomi ve Teknoloji Üniversitesi, ANKARA	2014
Yüksek Lisans	:	Selçuk Üniversitesi, KONYA	Halen
Doktora	:	-	

İŞ DENEYİMLERİ

Yıl	Kurum	Görevi
2014-2016	Treysis Elektronik ve Yazılım Sistemleri	Şirket Müdürü / Bilgisayar Mühendisi
2016-Halen	Necmettin Erbakan Üniversitesi, Bilgisayar Mühendisliği	Araştırma Görevlisi

UZMANLIK ALANI

Donanım, Yapay zeka, Optimizasyon

YABANCI DİLLER

İngilizce, İyi
Almanca, Orta

BELİRTMEK İSTEĞİNİZ DİĞER ÖZELLİKLER

YAYINLAR

Arslan B.S., Kiran M.S. - A Novel Hybrid Approach based on Ant Colony Optimization to Solve Travelling Salesman Problem - International Conference on Engineering and Natural Sciences - 2017

Arslan B.S., Kiran M.S. - QAP implementation with a new hybrid ACO algorithm - International Conference on Engineering and Natural Sciences - 2017