ANKARA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

YÜKSEK LİSANS TEZİ

BÜTÇE VE ZAMAN KISITLI TURİST ROTALAMA PROBLEMİ

İbrahim YILMAZ

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

ANKARA
2022

ÖZET

Yüksek Lisans Tezi

BÜTÇE VE ZAMAN KISITLI TURİST ROTALAMA PROBLEMİ

İbrahim YILMAZ

Ankara Üniversitesi
Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

Danışman: Prof. Dr. İman ASKERBEYLİ

Son yıllarda dünya genelinde yaşanan pandemi, savaş, ambargolar gibi küresel olayların etkisiyle dünya genelinde bir ekonomik bunalım yaşanmaktadır. Bu durumun ülkemize olan etkisi, coğrafi konumu ve gelişmekte olan bir ülke olması sebebiyle daha da artmaktadır. Turizm, bu tip ekonomik sıkışıklıklarda gelirinin arttırılması en kolay alanlardan biridir. Araştırmalar şunu gösteriyor ki, turistlerin kolayca seyahat edebilmesi için rota oluşturan, bütçe ve zaman kısıtlarıyla gezmek istedikleri şehirleri belirleyebilecekleri bir uygulamanın olmadığını göstermektedir. Bunun üzerine turistler için en kısa güzergahı bulan, tatmin puanı maksimum olan ve tatil için ayrılmış bütçeyi geçmeyecek bir rota oluşturan uygulamanın yapılması, turist sayısını ve her bir turistten gelen gelirin arttırılmasını sağlayacaktır. Uygulamayı kullanan turist, arayüz üzerinden belirlemiş olduğu gezilecek şehirlerin yer aldığı en doğru rotayı bulabilecektir. Bu sayede belirlemiş olduğu şehirlere varış ve çıkış tarihleri, bulunduğu şehirde kaç gün kalması gerektiği, hangi saatteki uçak ile sonraki şehre gideceği, uçak biletinin ücreti gibi bilgileri vererek kullanıcının en uygun sonuca ulaşması sağlanmış olacaktır. Bu tezde Gezgin Satıcı Probleminden yardım alınmıştır. Lojistik, planlama ve dağıtım gibi birçok alanda kullanılmış olan gezgin satıcı problemi uzun yıllardır üzerinde çokça çalışılmış olan (NP-hard) problemlerden bir tanesidir. Bu tezde gezgin satıcı problemine çözüm aramak amacıyla sezgisel yöntemlerden bir tanesi olan genetik algoritma kullanılmıştır.

Nisan 2022, 71 Sayfa

Anahtar Kelimeler: Gezgin satıcı problemi, turist seyahat programı, takım oryantiring problemi, genetik algoritma

ABSTRACT

M.Sc. Thesis

BUDGET AND TIME CONSTARINT TOURIST ROUTING PROBLEM

İbrahim YILMAZ

Ankara University
Graduate School of Natural and Applied Sciences
Department of Computer Engineering

Supervisor: Prof. Dr. İman ASKERBEYLİ

In recent years, there has been an economic depression around the world due to the effects of global events such as pandemics, wars and embargoes. The effect of this situation on our country is increasing due to its geographical location and being a developing country. Tourism is one of the easiest areas to increase its income in such economic difficulties. Research shows that there is no application that creates routes for tourists to travel easily, and that they can determine the cities they want to visit with budget and time constraints. On top of that, the implementation of the application that finds the shortest route for the tourists, has the maximum satisfaction score, and creates a route that will not exceed the budget allocated for the holiday, will increase the number of tourists and the income from each tourist. The tourist using the application will be able to find the most accurate route with the cities to visit that he has determined through the interface. In this way, the user will be able to reach the most appropriate result by giving information such as the arrival and departure dates to the cities he has determined, how many days he should stay in the city, what time he will fly to the next city, and the cost of the plane ticket. In this thesis, help was taken from the Traveling Salesman Problem. The traveling salesman problem, which has been used in many areas such as logistics, planning and distribution, is one of the (NP-hard) problems that have been studied for many years. In this thesis, genetic algorithm, which is one of the heuristic methods, is used to find a solution to the traveling salesman problem.

April 2022, 71 pages

Key Words: Travelling salesman problem, tourist travel program, team orienteering problem, genetic algorithm

ÖNSÖZ ve TEŞEKKÜR

Çalışmalarım boyunca yardım ve katkılarıyla beni yönlendiren, kıymetli tecrübelerinden faydalandığım danışmanım Prof. Dr. İman ASKERBEYLİ' ye, öğrencilik hayatımda değerli bilgilerini esirgemeyen ve çalışmalarım süresince yanımda olan Gazi Üniversitesi Endüstri Mühendisliği Bölümü öğretim üyelerinden Doç. Dr. Talip KELLEGÖZ' e saygılarımı sunarım.

Bugüne ulaşmamı sağlayan Amasya Mehmet Varinli İlköğretim Okulu, Amasya Anadolu Lisesi, Ankara Üniversitesi Bilgisayar Mühendisliği Anabilim Dalı, Gazi Üniversitesi Yönetim Bilişim Sistemleri Anabilim Dalı hocalarıma,

Tüm imkanlarıyla yüksek lisans eğitimimi destekleyen iş yerim HAVELSAN' a

Her daim maddi ve manevi destekleriyle beni hiçbir zaman yalnız bırakmayan annem Gülay YILMAZ' a, babam Atalay YILMAZ' a, ağabeyim Mustafa YILMAZ' a, ablam Sadiye YILMAZ' a teşekkürlerimi bir borç bilirim.

Son olarak özellikle lisansüstü eğitimim boyunca yanımda olan çok değerli ailem, eşim Habibe Ezgi YILMAZ' a ve kızım Meva YILMAZ' a tüm içtenliğimle teşekkür ederim.

İbrahim YILMAZ
Ankara, Nisan 2022

İÇİNDEKİLER

SİMGELER DİZİNİ

km	Kilometre
sn	Saniye

Kısaltmalar

GA	Genetik Algoritma
YAK	Yapay Arı Kolonisi
YAKA	Yapay Arı Kolonisi Algoritması
IEEE	The Institute of Electrical and Electronics Engineers
Tez	Yüksek Lisans, Doktora veya Sanatta Yeterlik Tezi
TOP	Team Orienteering Problem
İÇN	İlgi Çekici Nokta
TA	Tabu Araştırması
STBY	Sıklık Tabanlı Bellek Yapısı
TBÇ	Tek Biçimli Çaprazlama
SKÇ	Sıralı Kromozom Çaprazlama

ŞEKİLLER DİZİNİ

ÇİZELGELER DİZİNİ

1. GİRİŞ

Teknoloji hızla gelişmektedir. Bununla birlikte insanlar her geçen gün bilgisayarlardan daha fazla faydalanmaktadırlar. Zorlu problemler, bilgisayarlar ve algoritmalar ile çok hızlı ve pratik bir biçimde sonuçlandırılabilmektedir. Bu yüzden günümüzde bilgisayarlar insan hayatının en önemli bileşenlerinden biri haline gelmiştir.

İnsanların karşısına her gün başka bir sorun çıkabilmektedir. Bu problemlerden birçoğunun çözümü için bilgisayarlara ihtiyaç duyulabilmektedir. İlgili problemler, bilgisayarların anlayabileceği bir dile çevrilip bilgisayarlara yüklenmelidir. Bu sayede bilgisayar, ilgili problemi anlayıp işlemesinin ardından sonuçları üretmeye başlamaktadır. Böylece insanlar için en verimli en iyi sonuçlar elde edilmektedir. Özetle bilgisayarları insan hayatı için en pratik halde kullanmak şöyle ifade edilebilir. Teknolojinin büyümesi ve ilerlemesi ile insan gibi davranan, düşünen, problemlere bakış açısıyla insanlar gibi sonuçlar üretebilen, kendi kendine öğrenme yetisi olan bilgisayarlar ve çok daha ötesi bugün teknolojinin içinde ve dünyamızda yer almaktadır. Bu hızla daha da fazlasının olmaması için hiçbir sebep yoktur. Bu durumda yapay zekâ ile optimizasyon terminolojisi önem taşımaktadır. Aşağıdaki tanım yapay zekâ ve optimizasyon için literatürde yapılmış tanımlardan biridir.

Optimizasyon tanımlarının geneline bakıldığında eldeki problemin bulunan çözümleri arasındaki en iyisinin belirlenmesidir. Optimizasyon pek çok çalışma alanında bulunmaktadır. Pek çok çevrede optimizasyon teknikleri kullanılmaktadır. Herhangi bir probleme optimum çözüm bulunulması için geçilecek yol optimizasyon tekniklerinden geçmektedir. Uygulama alanı çok fazla olduğundan dolayı optimizasyon ve kavramları ile ilgili çokça tanımlama vardır. Bu durumdan dolayı bu kadar tanımlamanın arasında kesin olarak bir tanım vermek oldukça zordur (Raymond, 2009).

Günlük hayatımızda pek çok durumda karar vermek zorunda kalırız. Bu kararları birçok olasılığa ve kısıta göre vermemiz gerekebilir. Özellikle çokça parametrenin olduğu durumlarda zaman zaman bizim yerimize en doğru kararın bir başkası tarafından alınmasını isteriz. Bunun sebebi, parametre sayısının artmasıyla doğru sonucu bulabilmenin zorlaşmasıdır. İşte tamda bu tip durumlarda karar destek sistemleri

insanların yardımına koşmaktadır. Tezin temel prensibi de karar vermeyi kolaylaştırmaktır. Hayatın insanlar için daha kolay hale getirilebilmesi için teknolojinin gelişimi ile birlikte ekonomik gücün öneminin çok daha belirgin olduğu günümüzde ülke ticareti ve turist sayısına katkısının olacağı düşünüldüğünden dolayı bu çalışma yapılmıştır. Özellikle ekonomiye katkıda bulunabilmek amacıyla araştırmalar yapılmış, bu motivasyon ile tezin konusu ortaya çıkmıştır.

Tez, hem yurtdışından gelen hem de yurtiçinde bir tura çıkmak isteyen turistin karşılaşacağı senaryoları ele almaktadır. Uygulamada, gezilip görüldüğünde insanda bırakacağı olumlu izlenim (Tatminlik) puanına göre sıralanmış, Türkiye' deki tüm şehirlerin listesi yer almaktadır. Bu liste içinden, ilk seçilen başlangıç noktası olmak üzere birden fazla şehir seçilerek uygulamanın ilk parametresi olan şehirler listesi oluşturulmuş olmaktadır. Tatil başlangıç ve bitiş tarihleri ile bütçe ise diğer parametrelerdir.

Yapılan araştırmalara göre zaman kısıtlı çalışmalarda Genetik Algoritmanın yaygın olarak kullanıldığı görülmektedir. Behdadnia M. (2016), zaman kısıtlı oryantiring problemi çözümünde Genetik Algoritma kullanmıştır. Tezinde Tavlama Benzetimi, Karınca Kolonisi ve Genetik Algoritma deneyleri yapmış olup farklı girdiler ile bulunan en iyi sonuçların kıyaslamasında Genetik Algoritma hepsinde en yüksek puanı almıştır. Buna dayanarak Genetik Algoritma ile Bütçe ve Zaman Kısıtlı Turist Rotalama Problemine çözüm aranmıştır.

Sonuç olarak uygulama; tatilde hangi şehirde kaç gün kalınması, bütçenin en olumlu şekilde nasıl kullanılması, tatil süresinin nasıl dolu dolu geçirilmesi gerektiği kararını insanlar yerine almaktadır. Çıkarmış olduğu çıktı, her bir turistin daha fazla turistik yere gitmesini sağlayarak ticari faktörler adına pozitif gelişmeler sunmaktadır. Bunun yanı sıra turistlerde ülkemiz adına tebessümü de arttırması hedeflenmektedir.

1.1 Problem Tanımı

Problemde, Türkiye' yi tatilinde seyahat etmek isteyen bir turistin gitmek istediği şehirleri, tatil için planladığı süresi ve ayırdığı bütçe kısıtı ile optimum şekilde planlamaktır.

Turistin tatil için ayırdığı bütçe B TL, tatile başlayabileceği tarih ve saat D_s, tatilden dönmesi gereken en son tarih ve saat D_f dir. Türkiye deki tatil yapabilecek yerlerin kümesi T={A,B,C,D,...,i}, i yerinin günlük konaklama ve diğer masrafların maliyeti C_i, sağladığı tatmin miktarı g gün için $S_i + g*d_i$ (S, sabit tatmin düzeyi, g gün sayısı, d_i değişken tatmin düzeyi) dir. T kümesinde yerler arasında ulaşım programı $(i,j,a,b,c) \in A$' dır. Burada $i \in T$ başlangıç yeri, $j \in T$ varış yeri, a seyahat başlangıç zamanı, b seyahat bitiş zamanı, c seyahat maliyeti şeklindedir.

Amaç, toplam tatmin düzeyini en fazla yapan, bütçeyi aşmayan, başlangıç ve bitiş yeri turistin seçtiği başlangıç noktası olan, başlangıç ve bitiş tarih ve saati turistin belirttiği aralıkta olan birbirini makul şekilde takip eden şehirlerin olduğu bir seyahat rotasının oluşturulmasıdır.

Programın sonuç ekranındaki çıktı;

Seyahat : Toplam tatmin ≅ 0, tutarı, süresi, başlangıç ve bitiş yeri/tarih ve saati,

Konaklama : Toplam tatmin, tutar, süre, yer,

2. LİTERATÜR TARAMASI

Bianchessi (2017), takım orienteering problemi için bir dal kesme algoritması konusunda yaptığı çalışmada, her araç için önceden tanımlanmış seyahat süresi sınırını aşmamakla birlikte, bir araç dizisi tarafından toplanan kârın toplamını maksimize etmeyi amaçlamaktadır. Polinom sayıda değişken ve kısıt içeren yeni bir iki indeksli matematiksel formülasyon geliştirmişlerdir. Bağlanabilirlik kısıtlamaları ile takviye edilen bu kompakt formülasyon, branch-and-cut algoritması ile çözülmüştür.

Ledesma ve Gonzales'in (2017) çalıştığı Takım Yöneltme Ark Yönlendirme Problemi, Takım Oryantiring Probleminin bir varyasyonudur. Bir depoda bulunan ve belli bir kapasiteye sahip araçların, bir dizi düzenli müşteriye hizmet götürmesi gerekmektedir. Her opsiyonel müşteri hizmet gördükçe bir kâr üretir. Müşteriler birden fazla hizmet görseler de kâr en çok bir kez toplanabilir. Sorun, bir zaman süresince tüm düzenli müşterilere ve isteğe bağlı müşterilerin bazılarına hizmet veren toplanan toplam kârın maksimize edilmesi sağlayan problemi branch-and-cut algoritmasıyla araç rotalarını tasarlamayı amaçlıyor.

El-Hajj, Dang, Moukrim, (2016) takım oryantiring problemini cutting planes algoritması ile çözümü başlıklı makelesinde, müşterileri seçmek ve aynı zamanda bir araç filosunun ziyaretlerini en üst düzeye çıkarmak amacıyla toplanan kazançlar ve her araç için bir seyahat süresi sınırlamasını organize etmek için uygulanmaktadır. Bu yazıda, polinom sayısı ile doğrusal bir formülasyonun etkin kullanımı incelenmektedir. İlk önce daha az araç düşünerek orijinal sorunun daha küçük ve orta düzeydeki modellerini çözebileceği düşünülmekte. Daha sonra, daha büyük modelleri çözmek ve en sonunda orijinal soruna erişmek için bilgi elde edileceği yoluyla hareket edilmektedir.

Ke, Zhai, Li, Chan, (2015) takım oryantiring problemine bir yaklaşım başlıklı makalesinde, araç yönlendirme sorununuyla başa çıkmak için Pareto mimik algoritması adı verilen yeni bir algoritma önerilmiştir. Bu problemde, her bir dizi düğüme (veya müşteriye) hizmet etmeyi planlanyan bir ödül ile ilişkili araç filosu bulunmaktadır. Bir araç bir düğüme servis edildikten sonra karşılık gelen ödül alınacaktır. Amaç zaman

sınırını aşmamak kaydıyla her güzergahın seyahat süresi boyunca alınan toplam ödülü en üst düzeye çıkarmaktır.

Keshtkaran, Ziarati, Bettinelli, Vigo, (2016) takım oryantiring problemleri için tam çözüm yöntemleri. Bu makalede TOP'a kanıtlanmış optimal çözümleri bulmak için Şube ve Fiyat yaklaşımını önerilmektedir. Bu problemde, müşterilere hizmet etmek için aynı araçlardan oluşan bir filo mevcuttur. başlangıç noktasından son varış noktasına giden yolların uzunluğu sınırı bulunmakta. Bu nedenle, tüm müşterileri ziyaret etmek mümkün olmayabilir. Her müşteri önceden tanımlanmış bir kârla ilişkilendirilir ve en fazla bir kere hizmet verilebilir. Hedef toplanan toplam karın maksimize edildiği şekilde araçlar için bir rota dizayn etmektir. Problem, azaltılmış durum uzayı gevşemesine sahip sınırlı çift yönlü dinamik programlama algoritması ile çözülür. TOP' un çözümünde Branch-and-Cut-and-Price yaklaşımı kullanılmaktadır.

Takım Oryantiring Probleminin Bulanık Puanlar ve Kısıtlarla Çözümlenmesi. Bu çalışma, puanlar ve seyahat süresi kısıtlamaları bulanık olan Team oryantiring problemini formüle eder. Sorunu çözmek için önerilen yumuşak hesaplama yöntemi, bulanık optimizasyon yöntemlerini yapıcı bir metasezgisel ile çözmektedir (Brito J., Exposito A., Moreno A., 2016).

Takım Oryantiring Problemi için Rastgele Buluşsal Yöntem başlıklı konferansta, bizimde konumuz olan turist gezisi planlamasından bahsedilmektedir. Bu yazıda, takımın yönlenme problemine benzeri görülmemiş buluşsallık uygulanmaktadır. Bulunan yöntemin yineleme yani döngü sayısını azalttığından bahsedilmektedir. Buda doğru sonucu en kısa zamanda vereceği anlamına gelmektedir (Zettam, Elbenani, 2016).

Zamana bağlı takım oryantiring problemi için sezgisel yaklaşımlar: Turistik rota planlaması için uygulama başlıklı makalede toplu taşıma araçları kullanılarak birden çok ilgi çekici nokta (İÇN) ziyaret etmekle ilgilenen turistler için rota planlama problemi ele alınmıştır. Bu sorunun temel amacı, çok sayıda parametreyi ve kısıtlamayı göz önüne alarak turist tercihleriyle eşleşen İÇN' leri bulmak ve günlük gezi için uygun zamana göre toplu taşıma araçlarını entegre etmektir. Turist Gezisi Tasarım Sorunu prototip algoritması ve Atina meydanlarının verileriyle test edilmiştir (Gavalas D., Konstantopoulos C., Mastakas K., Pantziou G., Vathis N., 2015).

Ben-Said vd. (2016) Kapasitif Takım Oryantiring Problemi için uyarlamalı bir sezgisel yöntem başlıklı makalesinde, her bir müşteriye kâr ve talep ilişkilendirerek problemi çözmek, hizmet edilecek müşterilerin toplamını, ziyaret edilen müşterilerden tahsil edilen kâr tutarını, kaynak sınırlamaları, her bir aracın maksimum uzunluk sınırı ve maksimum kapasitesi, tümü hesaplanarak en iyi sonucun elde edilmesi istenmektedir. Problem çözümünde uyarlamalı iteratif yıkıcı yapıcı sezgisel (adaptive iterative destructive constructive heuristic) yöntemi kullanılmaktadır. Bu yöntemin en önemli noktası ise rekabetçi hesaplama süresidir.

Archetti vd. (2015) Takım Yöneltme Ark Yönlendirme Sorunu için bir matematiksel çözüm başlıklı makalede seçilen müşterilere hizmet etmek için sınırlı filo aracı bulunmaktadır. Bu araçlar potansiyel müşterilerin bağlı olduğu grafikler temel alınarak seçilmektedir. Her araç maksimum yol süresi sınırlamasına tabi olmalıdır. Amaç hizmet verilen müşterilerin kârını en üst düzeye çıkarmaktır. Ark yönlendirme problemlerinde müşteriler köşe üzerinde bulunurlar. Bu işlemin basit problemlerinden biri Çin Postacı Sorunu, burada tüm kenarlar veya yaylar geçilmelidir ve Kırsal Posta Taşıyıcı Problem, burada yalnızca bazı kenarların veya yayların geçilmesi gereklidir ve minimum maliyet yolları belirlenmelidir. Problem için bir matematiksel yöntem önerilmekte ve optimal çözümün veya bir üst sınırın bilindiği bir dizi karşılaştırma örneği ile üzerinde test yapılmaktadır. İncelemenin en can alıcı yanı en uygun çözümün mevcut olduğu tüm örneklerde ortalama hata Yüzde 0,67 olmasıdır.

Wang X., Golden B., Gulczynski D., (2014) ayrılmış teslimat için takım teslimatını minimum yayın miktarlarıyla kısıtlamış olan en kötü durum analizi başlıklı makale incelemesinde, kapasitasyona tabi takım oryantiring probleminde, bir filo donanımı, kapasite ve güzergah uzunluğu kısıtlamalarını yerine getirirken toplanan toplam karı en yükseğe çıkarmak için müşterilerin bir alt grubunu ziyaret etmesi ve bunlara hizmet etmesini ele almaktadır. Problemde birden fazla araç aynı müşteriye hizmet verebilir. Her araç, müşteri talebinin tümünü değil bazılarını karşılayabilir. Aşırı durumlarda, bölünmüş teslimatların yapılmasına izin verildiğinde, toplam kar iki katına çıkabilir. Bununla birlikte, bölünmüş teslimatlar sıklıkla müşterilerin rahatsızlığına neden olur, bu nedenle bölme dağıtım takım oryantiring sorununu minimum bölünme ile birlikte

sunulmaktadır. Bu makalede, maksimum olası kâr artışının sınırlarını belirlemek için problem üzerinde en kötü durum analizi yapılmaktadır.

Archetti C., Bianchessi N., Sperenza MG. Hertz A., (2013), takım oryantiring probleminin kapasiteleştirilmesiyle bölünmüş teslimat başlıklı makalede bölünmüş teslimatlara izin verilen kapasitif takım orienteering problemi incelenmektedir. Her biri bir talep ve kâr ile bağlantılı olan bir dizi potansiyel müşteri verilir. Kapasitasyon araçlarından oluşan bir filo tarafından hizmet edilecek müşteri seti, her bir rotanın maksimum süresine ve araç kapasitesindeki kısıtlamalar ile, toplanan karın maksimize edileceği şekilde tanımlanmaktadır. Bölünmüş teslimatlara izin verildiğinde, her müşteriye birden fazla araç sunulabilir. Ayrıca bölünmüş teslimatlara izin vererek toplanan karın, her bir müşterinin en fazla bir araçla servis edilmesi gerektiği kısıtlamasında toplanan kârın yarısı kadar olabileceğini görülmektedir. Çözüm olarak branch-and-price algoritması ve bir melez sezgisel yöntem sunulmaktadır.

Salazar-Aguilar MA., Langevin A., Laporte G., (2013), çok bölgede takım oryantiring problem adlı bu yazıda, çok bölgedeki ekip yönlenme problemi tanıtılıyor. Bu problemde, biri çeşitli ilçelerde bulunan zorunlu ve isteğe bağlı görev setleri ve planlama anlayaşı ele alınmaktadır. Toplam planlama anlayışı uzunluğuna göre belirlenen mevcut zaman ilçelere dağıtılmalıdır. Her ilçedeki tüm zorunlu görevler yerine getirilmelidir; diğer görevler ise, zaman izin verdiği sürece yapılabilmektedir. İsteğe bağlı bir görev gerçekleştirildiğinde pozitif kâr veya puanlar toplanır. Bunlara ek olarak, görevler arasındaki bazı uyuşmazlık kısıtlamaları dikkate alınmaktadır. Amaç, her ilçede günlük olarak gerçekleştirilecek bir dizi görev için zaman çizelgesi hazırlarken, toplam karı en yüksek düzeye çıkarmaktır. Problemin çözümünde karışık bir tam sayılı formülasyon ve uyarlanabilir büyük mahalle arama sezgisel yöntemi kullanılmaktadır.

Tamamlanmamış hizmetle kapasite takım oryantiring problemi başlıklı makalede takım Yönelimli problemin ve hizmet verdiyse bir müşterinin tamamıyla sunulması gerektiği varsayımının rahatlatıcı etkisinin kapasite versiyonunu incelenmektedir. Tamamlanmamış Hizmetle toplanan karın, CTOP tarafından toplanan karın iki katı kadar olduğunu kanıtlıyoruz. Tamamlanmamış hizmete izin vererek kârın ortalama artışını değerlendirmek için ayrıca bir hesaplama çalışması yapılmıştır. Sonuçlar, karın

kuvvetli bir şekilde özel duruma bağlı olarak arttığını göstermektedir. Test edilen örneklerde, kâr artışı % 0 ile % 50 arasında değişmektedir (Archetti C., Bianchessi N., Speranza MG., 2013).

Dang DC., Guibadj RN., Moukrim A., (2013), takım oryantiring problemi için etkili bir Parçacık Sürü Optimizasyonundan esinlenen algoritma başlıklı makalede, Takım Yön Bulma Problemi yani belirli bir araç yönlendirme probleminin amacı, seyahat masraflarını / zaman sınırını aşmadan ziyaret edilen müşteriler tarafından kazanılan kazancı en üst düzeye çıkarmaktır. Bu makalede bir grafik aralığı modeli ve bir parçacık temelli takım oryantiring problemi için yeni ve hızlı bir değerlendirme süreci önermektedir. Problemi çözmek için Sürü Optimizasyonu Algoritması kullanılmıştır. Deneyler Algoritmanın, TOP' un standart kıyaslamasının yani mevcut çözme yöntemlerinden daha iyi performans gösterdiğini açıkça gösteriyor. Optimizasyonu Algoritması, % 0.0005'lik bir göreli hataya erişirken, literatürdeki en iyi bilinen göreceli hata % 0,0394. Kullanılan algoritma, en iyi çözümlerden birini tespit etmektedir.

3. SEZGİSEL (HEURISTIC) TEKNİKLER

Bu başlıkta literatürdeki sezgisel teknikler ile ilgili genel bilgiler verilecektir. Öncesinde sezgisel kelimesiyle alakalı bilgi vermek daha yararlı olması düşünülmektedir. Kelime anlamı olarak sezgisel kelimesi Türkçe kaynaklarda "Sezgisel" yazım olarak kullanılmaktadır. Yabancı kaynaklarda ise "Heuristic" olarak geçmektedir. Heuristic, "heuriskein" fiilinden türetilip üretilmiştir. Heuristic fiili ise keşfetmek, bulmak gibi anlamlarda kullanılmıştır. (Tunçhan, 2008) Sezgisel yöntemlerin hiçbiri tam olarak kesin çözüm bulduğunu söyleyemezler. Sezgisel yöntemler kesin bir sonuç bulamazlar. Zaten teknikler kesin sonuç bulduklarını söyleyemezler. Sezgisel teknikler limitte kesine en yakın çözümü bulabilirler. Zira sezgisel tekniklerin hiçbiri doğru sonucu tam anlamıyla bulduğunu söyleyemez.

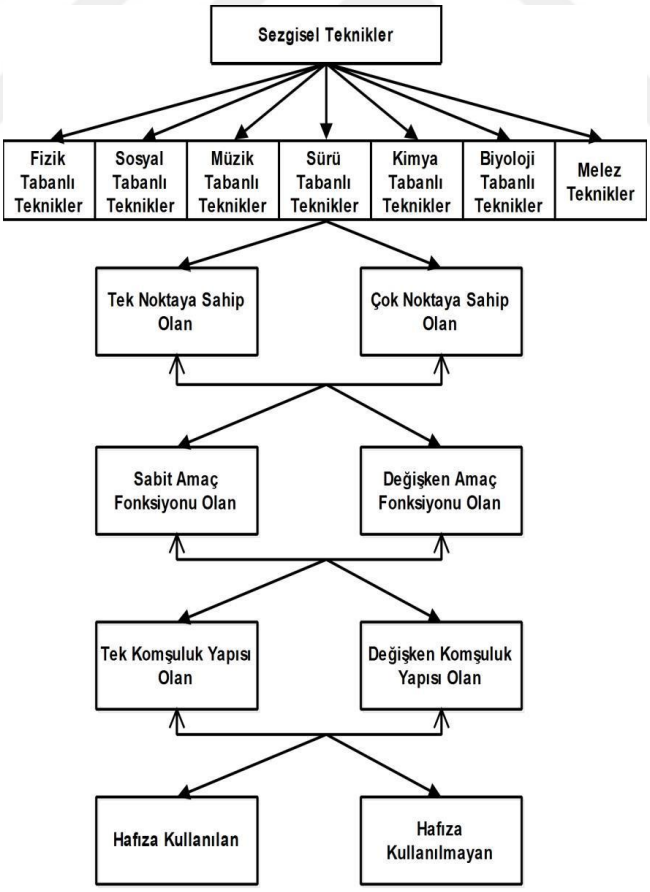
Problemlerde sezgisel yöntemler kullanıldığında bulunabilecek tüm sonuçlar bulunmamaktadır. Bunun yerine doğruya en yakın bazı çözümler elde edilmektedir. Böylece sezgisel teknik kullanılarak ulaşılmış sonucun en iyi sonuç olduğu iddia edilebilir ancak optimal olduğu söylenemez. Ulaşılabilecek tüm çözümler, problemde ki kısıtlar ve kurallara bağlı olmaktadır. Aynı zamanda çözüm uzayı içindeki bütün çözümlerin gerçekleştirilmesi çoğu zaman mümkün değildir. Bütün çözümlerin bulunması iki sonucu ortaya çıkarabilir. Bunlardan birincisi çok uzun zaman alabilecek olmasıdır. İkincisi ise sonuçlar sonsuz bir kümede yer almasından dolayı hiçbir şekilde kesin sonucun elde edilemeyecek olmasıdır.

Sezgisel optimizasyon teknikleri, fizik, biyoloji, kimya, sürü, sosyal ve müzik tabanlı olabilmektedir. Bu nedenle ayrı ayrı altı grupta değerlendirilmektedir. Bunlardan biri sürü zekâsı optimizasyon algoritmalarıdır. Örnek olarak; kedi, balık, arı, kuş verilebilir. Bu algoritmalar hareketleri ve davranışları incelenen doğadaki hayvan sürülerinin sayesinde geliştirilmiştir. (Akyol S., Alataş B., 2012) Sezgisel yöntemlerden fazla sayıda bulunmaktadır. Bunun sebebi her biri sonuçları bulabilmek için kendine özgün metotlar geliştirmiş olmasıdır. Bu sayede fazla sayıda algoritma literatüre geçmiştir. Buda çeşitliliği arttırıp optimum sonuca biraz daha yaklaşmamızı sağlamaktadır. Problem çözüm sürecinde iki optimizasyon tekniği vardır. Sezgisel tekniklerin bazılarındaki çözüm yöntemleri sürekli optimizasyonken diğerlerinde ise kesikli optimizasyon kullanılmaktadır. Bunlar problemin türüne, yapısına, çeşidine göre

değişmektedir. Kesikli optimizasyonlarda tam sayılar yer alıyorken sürekli optimizasyonlarda ise tam sayıların yanı sıra tam sayı olmayan değerlerde yer almaktadır.

Sezgisel yöntemlerde amaç fonksiyonu vardır. Bu fonksiyon problemin yapısına bağlı olarak değişebilmektedir. İki şekilde belirtmek gerekirse bunlardan birincisi sabit amaç fonksiyonudur. Diğeri ise değişken amaç fonksiyonudur. Sezgisel tekniklerde iki farklı komşuluk yapıları öne çıkmaktadır. Bunlar tek komşuluk veya değişken komşuluk yapılarıdır. Bir başka durum da algoritmalardaki hafızadır. Yönteme göre hafızalar değişebilmektedir. Bundan dolayı ilgili algoritmada hafıza olabilir ya da olmayabilir. Algoritma kullanıcı ihtiyaçları ne ise ona göre sonuçlar üretmek durumundadır. Şekil 3.1 de sezgisel teknikler ifade edilmektedir.

Sezgisel teknikler konusuna giriş yapılacaktır. Bu konuyla alakalı gerekli bilgiler verilecektir.



Şekil 3.1 Sezgisel teknikler (Akyol, 2012)

3.1 Karınca Kolonisi Algoritması

Matematiksel modelleri üzerine dayalı gerçek karınca koloni davranışlarının olduğu bir algoritmadır. Dorigo ile arkadaşları tarafından ilk çalışma yapılmıştır (Dorigo, 1991). Yapılan çalışmada elde edilen algoritmaya "karınca algoritması", sistemlerine "karınca sistemi" adını vermişlerdir.

Çevre şartları karıncalar için büyük önem taşımaktadır. Çevre şartlarına göre de yol belirlemektedir. Nitekim besin kaynağını bulmak içinde bir yol belirmek zorundadır. Yuvasından çıktığında besin kaynağının yerini tespit etmesi ve bir rota oluşturması gerekmektedir. Bu yollardan geçerken bir koku salgılanmaktadır. İlgili kokunun adı feromon dur. Bu kokuyu geçen ilk karınca yola bırakmaktadır. Yolun uzunluğuna göre kokunun yoğunluğu değişmektedir. Koku daha yoğun ise bu yolun kısa olduğu anlamına gelmektedir. Bu sayede diğer karıncalarda ilk geçen karıncanın geçtiği gibi yolda devam etmektedirler. Tercih yapılması gerekilen yol ayrımlarında ise hangi yolu tercih edeceğini öncelikle koku miktarı belirlemektedir. İkinci olarak ise rastgele bir kriter ile karar verilmektedir. Karıncaların bu hareket analizinde rastgele kriter ile seçim yapmasının nedeni ise daha iyi, daha kısa sonuçlar yani yollar bulabilmektir.

Karıncaların çalışma şekillerine değinelim. Tüm karıncalar kendi özelinde olasılığa göre hareket etmektedir. Bu sayede olasılıksal belirlemiş olduğu kurallara göre şehirler seçmektedir. Bu sayede şehirler arası turlar oluşturmaktadır. Bu kurallara göre karıncalar en kısa mesafe için kestirme yollar ile bağlanmış yoğun feromon olan yolları tercih etmektedirler. Karıncalar döngülerini tamamladığında ise feromon güncellemesi yapılmaktadır. Şehirlerden bir miktar feromon bulaştırılmaktadır. Sonrasında ise karıncalar geçmiş olduğu şehirlerdeki yolun kısalığına göre geçtiği yerlere feromon bulaştırmaktadır. Tüm karıncalar için döngüler tekrar etmektedir.

Algoritma karıncaların yapay şekilde bırakmış olduğu feromon kalıntılarına bakmaktadır. Bu kalıntıların güncelleştirilmesiyle döngüye girip kendini tekrarlamaktadır. Algoritma uygulama esnasındayken feromon kalıntıları güncellenmektedir. Karıncalar bu güncelleme ile doğru çözümü oluşturmayı hedeflemektedirler. Devamında her döngüde güncelleme yapılmaktadır. Uygulama esnasında temel olarak bakılırsa karıncaların ulaştıkları yerlerde feromon kokusu

oranının artmış olmaşı, zamanla feromon oranının bir kısmının buharlaşması, çözümler arasından optimum olanının bulunması ve yeni feromon oranlarıyla döngünün devam ettirilmesidir.

Karınca kolonisi iterasyon adımları:

1. Adım: Feremon miktarları belirlenir.
2. Adım: Karıncaların yerleri rastgele belirlenir.
3. Adım: Karıncalar gideceği şehri arama koşuluna göre seçim yapar ve döngüsü sonlanır

4. Adım: Karıncalar tarafından, geçtikleri yolların mesafesi hesaplanır. Buna göre feromonlar yenilenir.

5. Adım: Optimum çözüm hesaplaması yapılır. Böylece bu çözüm ile feromon güncellemesi yapılır.

6. Adım: En yüksek döngü sayısı veya algoritmada verilen limit sağlanan kadar 2. Adımdan devam edilir.

3.2 Yapay Arı Kolonisi Algoritması

Karaboğa yazmış olduğu makaledeki tanıma göre YAKA, bal arılarının kendine özgü bazı hareketlerinden esinlenerek oluşturulmuş bir algoritmadır. Bunlar zeki davranışları ile besin arama esnasında uyguladıkları bazı yöntemler örnek olarak verilebilir. Bu yöntem bir sürü zekası algoritmasıdır. Optimizasyon problemlerini çözmek için kullanılan bu yöntem sürü halinde hareket eden arıların yapmış olduğu hareketleri göz önüne alarak oluşturulmuştur (Karaboğa D., 2011).

Bal arıları bir düzen halinde hareket etmektedirler. İçgüdüsel bir şekilde yaşamlarını bir düzen halinde devam ettirir ve bunu bir kural olarak benimsemişlerdir. Bu düzenden çıkmadan hareket ederler. Bu düzen içinde topluluktaki her arının belirli bir görevi vardır. Bu görevi en iyi şekilde yaparlar ve kuralların dışına çıkmadan hareket ederler. Arılara verilen görevlere örnek vermek gerekirse iletişim kurma, besin bulmak, depolama ve besinin taşınması bunlardan bazılarıdır. Arıların muhteşem dengesi ve kovanlarının düzeni bilim insanlarının araştırmalarına girmiştir. Bu sayede arıların

işleyişinin modellenmesine doğru bu araştırmalar evrimleşmiştir. Arıların kovanlarında 3 sınıf bulunmaktadır. İngilizce literatürde queen, drone, worker olarak adlandırılırken Türkçe literatürde ise kraliçe arı, erkek arı ve dişi işçi arı olarak geçmektedir.

Arılar, hayatlarını devam ettirebilmek için besin bulmaları gerekmektedir. En önemli işleri budur. Arılar kovanları içinde besin biriktirirler. Ayrıca dışarıdaki besinlerin aranması, bulunması ve getirilmesi önemli süreçleridir. Kovanlarından ayrılmalarıyla besin aramaya başlamaktadırlar. Bu arama işlemi başta rastgele rotalarla yapılmaktadır. Eğer ki arılar bulduğu besin kaynağında azalma olursa diğer arıların vermiş olduğu bilgilerle yeni besin kaynağı ararlar. Bu süreçte yapılan işlemlere; aramalar sonucunda ulaşılan kaynağın kovana götürülmesi, bilgilerin paylaşılması örnek gösterilebilir. Kovanlar bölümlerden oluşur. Bunlardan biri de dans alanıdır. Dans alanında bilgiler arlılar arasında iletilmektedir. Böylece paylaşım yapılmış olur. Bu alanda her bir arı tarafından dışarıda toplanmış bilgi diğer arılara paylaşım yapılarak 1-n ilişikisi tüm arılar arasında kurulmuş olur. Yaptıkları dansın adı ise literatürde "Waggle Dance" olarak geçmektedir. Bu bilgi iletişiminin sayesinde besin kaynakları bulunur. Besin getirenler diğerlerine yol gösterebilmek için konum belirtmektedirler. Arılar besin kaynağına gidebilmek için güneşten yardım alırlar. Güneş ışınları sayesinde yönlerini bulup dans alanında öğrenmiş oldukları bilgiyle besin kaynağına gidebilmektedirler. Arıların hareketleri birçok alanda insanlara ilham olmuştur. Vücutlarındaki enerji ile maksimum mesafe gidebilmek için enerjilerini optimum kullanmak istemektedirler. Bundan dolayı üstlerindeki yüke göre alçaktan veya yüksekten uçabilmektedirler.

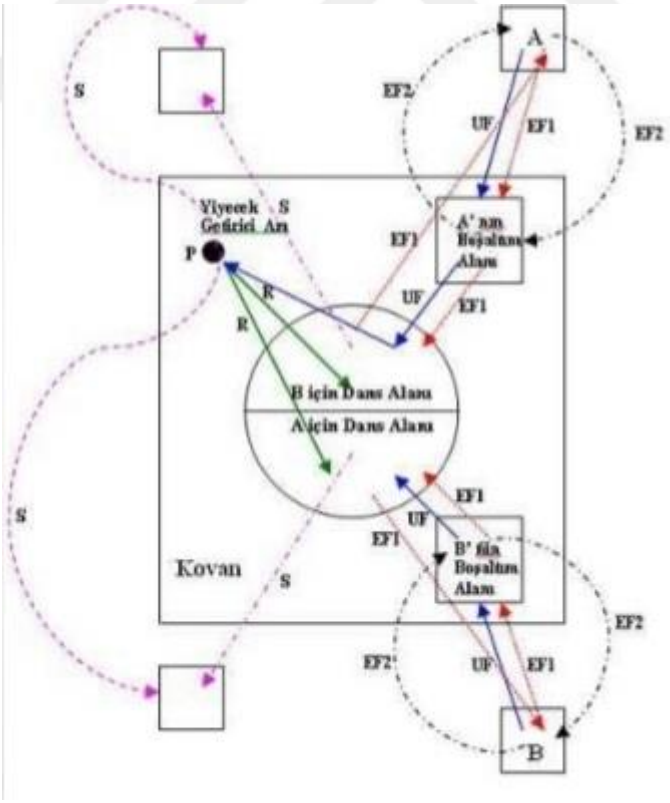
Kovan dışında bilgi iletişimini sağlayabilmek için de bir dan çeşidini kullanmaktadırlar. Bu dans ile 8 rakamı gibi hareket ederek bilgi aktarımı yapabilmektedirler. 10 kilometrelik mesafeye kadar iletişim kurabilmektedirler. Bu dansa kuyruk dansı denilmektedir.

Arıların kovan dışında besin bulmak için yapmış oldukları hareketler ile besinleri kovana götürme esnasında ki rotaları Şekil 3.2 de gösterilmiştir. Arıların hareketlerinin incelenmesinde A ile B aramalar sonucunda bulunan besinlerdir Arılar görevlerine göre hareket ederler. En başta görevi henüz verilmemiş ve besin konum bilgisini henüz almamış arı arama işlemini yapacaktır. İlgili arının şekilde iki yerde olması beklenmektedir. Birincisi kaşif arı S ile gösterilmiştir. Bu arı besin aramaktadır. İkincisi

R ile gösterilen, besin kaynaklarına gidebilmek için dans edenleri takip ederek rota ve konum bilgisi alan gözcü arıdır.

Besin kaynaklarına erişip buralarda bulunan nektarı kovana götürürler. Bu durumda başta görevi belli olmayan arılar görevlerini almış olurlar. Bu arıların besini kovana götürmeleriyle karşılarına üç seçenek çıkmaktadır:

1. EF2’ de gösterildiği gibi besin kaynağından kovana öz getirilir. Yani yapılan işe devam edilir.
2. EF1’ de gösterildiği gibi kovana gelmeden önce dans edilir. Bu dansın amacı bilgi iletişimini sağlayarak besin kaynağının konum ve büyüklüğü hakkında diğer arılara bilgi vermektir.
3. UF’ de gösterildiği gibi yeni besin kaynaklarına erişebilmek için konum ve diğer bilgileri alabilir



Şekil 3.2 Arı kolonisi örnek şekli

YAKA adımları aşağıdaki gibidir:

1. Adım: Kaynaklar rastgele oluşturulur. Kaynaklara sadık kalınır. Gözcü arı ile İşçi arı sayısı belirlenir. Limit değeri tespit edilir. Kontrol amaçlı sayaç değişkeni oluşturulur.
2. Adım: Besin kaynaklarına ait tüm besinlerin çözüm değerleri belirlenir. Amaç fonksiyonunun türüne göre çözüm değerleri hesaplanır.
3. Adım: Maksimum döngü sayısı belirlenir. İşçi arılar besin kaynaklarına gönderilir. İşçi arılar rastgele bir besine yönelir. Bu besin kaynağını işlemeye başlarlar. Bu besin kaynağı işlendikten sonra besine ait yeni besin kalitesi, yani çözüm değeri hesaplanır. Çıkan çözüm değeri önceki değerden daha iyi ise bu besin demektir. Besinle ilgili bilgiler hafızaya alınırlar. Çözüm değerinde iyileşme sağlanır ise limit değeri sıfırlanır. Aksi halde limit değeri bir arttırılır. Limit değeri için belirli bir üst değer belirlenebilir.
4. Adım: Arıların yani işçilerin safhası sona ermiştir. Daha sonra ise gözcü arılar çalışmaya başlarlar. Kaynakların uygunluk değeri belirlenir. Buna göre bir yem kaynağı belirlenir. Bu kaynak için gözcü arılar çalışırlar. Elde edilen sonuç aynı şekilde önceki sonuçtan daha iyi mi diye bakılır. Önceki sonuca göre gelişme var ise bu kaynak demektir. Bu kaynakla ilgili bilgiler hafızaya kaydedilir. Kaynakta olumlu bir durum sağlanır ise limit değeri sıfıra çekilir. Eğer ki tam tersi olursa limit değerine bir eklenir.
5. Adım: Gözcü arıların safhası da sona ermiştir. Daha sonra kâşif arı devreye girer. Yöntemin yerel en az ya da en fazla ile örtüşmesine engel olmak için kâşif arı safhası başlar. Bulunan sonucu tamamı ile bozar. Kısaca limit değerleri sıfıra eşitlenir ve yeni bir sonuç ortaya çıkması için çabalar. Bulunan sonuç ile karşılaştırmaya girer. Yapılan karşılaştırma daha önceden yapılmış ve bir sonuç üretilmiş. Karşılaştırma şu şekilde yapılır. Önceden hafızaya kaydedilmiş sonuç ile mevcut sonuç kıyaslanarak karşılaştırma gerçekleştirilir. Bunun sonucunda iyi olan kazanır ve yoluna devam eder. Yani hafızaya kaydedilir.
6. Adım: İşçi, gözcü ve kâşif arı safhaları maksimum döngü sayısı sağlanıncaya kadar çalışmaktadır. Algoritma durması için istenen kriter ve kurallar sağlandığında algoritma durdurulur.

YAKA dörde ayırılabilir. Yukarıdaki adımlardan bu 4 parça çıkarılabilir. Bunlar aşağıda belirtilmiştir.

- İşçi arıların besin kaynaklarına gönderilmesi
- Rastgele besin kaynaklarının üretilmesi
- Yem kapasitesi bitmiş olan kaynaktan ayrılmak
- Kaynağın belirli değerlere göre gözcü arı tarafından seçilmesi.

YAKA algoritmasının özelliklerini aşağıdaki gibi şöyle ifade edilmektedir. (Akay, 2007):

1. Sürü zekasına dayalıdır.
2. Kontrol parametresi azdır.
3. Arıların gerçek hayatlarında ki besin arama hareketlerinin neredeyse bire bir olarak modellenmiştir.
4. Basit ve esnek bir algoritmadır.

4. PROBLEMİN ÇÖZÜMÜ İÇİN LİTERATÜRDE KULLANILAN YÖNTEMLER

İnsanlık tarihinde, geçmişten günümüze kadar insanlar birçok zorlukla başa çıkmak için çabalamıştır. İnsanların yaşamış olduğu bu zorluklar ve problemler için birçok optimizasyon yöntemi geliştirilmiştir. Zaman ilerledikçe bilim insanları ve araştırmacılar yeni yöntemleri bulmaya ve geliştirmeye devam etmektedirler (Özdemir, 2013).

Team Orienteering ve Gezgin Satıcı Problemi yöntemleri bunlardan birkaçı. Gezgin Satıcı Problemini kısaca özetlemek gerekirse birbirleri arasındaki mesefaleri bilinen N tane düğümün hepsine minimum bir kere uğrayan ve optimum yolun bulunmasını hedeflemektedir. Optimum yol bulunurken en kısa mesefe ile minimum maliyetler göz önüne alınmaktadır. Zor yoldan arama yapılırsa permütasyonların hesaplanması ile yol uzunlukları bulunmaktadır. Bu sayede bütün permütasyonlar arasından en kısa olanı seçilerek sonuç elde edilebilmektedir. Bu çözüm yöntemi N! İle çok yüksek sürelerde çözülebilmektedir. Özellikle N büyük olduğunda çözümü imkansız kılabilmektedir. Bir başka değişle kabul edilemeyecek sürelere ulaştığından imkansız kabul edilebilmektedir. Bu tip durumların yaşanmaması için hızlı ve doğru sonuçlar üreten çözümler için algoritmalar geliştirilmiştir. Bunlardan en popüler olanı Branch-and-Cut dır. Dallan ve Kes olarak Türkçe literatüre geçmiştir. Branch-and-Cut kesin çözüm veren yöntemlerden biridir.

4.1 Tabu Araştırma Hafızası

Tabu araştırma hafızası tekniğinde daha öncesinde oluşturulmuş çözümlerin sonuçlarından yararlanıldığından dolayı TA için ussal olan bir teknik olduğu söylenebilmektedir. TA içinde tüm eylemler kaydedilmektedir. Adında anlaşıldığı gibi bir hafıza sistemi bulunmaktadır. Kaydedilenler tabu ile adlandırılmış kısıtlar için kullanılmaktadırlar. Bu kısıtlar ile daha önce üretilmiş sonuçlara tekrardan ulaşılması engellenir. Bundan dolayı bir kere ulaşılmış sonuca tekrar tekrar ulaşılmasının önüne geçilmektedir. TA' da birden fazla hafıza türü vardır. Bunlarla sorunun çözümüne göre hafızalar yapılarını şekillendirmektedir. Yaygın olarak bilinen hafıza yapılarından iki tanesine değinilecektir. Birincisi kısa dönemli veri tutan yakın geçmiş tabanlı hafızadır.

İkincisi ise uzak dönemli veri tutan sıklık tabanlı hafızadır. Bu iki hafıza yapısının aralarındaki farkın oluşmasında tabu listesinin etkisi büyüktür. Sıklık tabanlı hafızadaki kısıtlardaki hareketler, yakın geçmiş tabanlı hafızada tutulan kısıtlarda olmayabilmektedir. Kısaca şöyle ifade edilebilir. Yakın geçmiş tabanlı hafızada en son bulunan kısıtlar yani tabular yer alıyorken, sıklık tabanlı hafızada ise daha genel tabular yer alır. Aşağıdaki başlıklarda bu iki hafıza yapısından bahsedilecektir (Özdemir, 2013).

4.1.1 Yakın geçmiş tabanlı bellek yapısı

Bu hafıza tipi TA içinde en temel ve basit bellek türüdür. Bu bellek yapısının başlıca görevi isminde olduğu gibi yakın geçmişe bakıp yapılmış eylemlerin yeniden yapılmasını engellemek adına tabulaştırma yapmaktadır. Bu durumda tabuların hafızada ne kadar zaman durması gerektiği ile ilgili problemle karşılaşılmaktadır. Şu şekilde açıklanabilir, $x \varepsilon X$ ise, X sonuç kümesinde x sonucunu için eylemlerin kısıtlaştırılması ile sonrasında tabular arasından silinmesidir. Böylece x sonucunu sağlayan eylemin belirli süre ile hafızaya kalması gerekmektedir. Geçen süre iki kurala tabidir. İlki statik, ikincisi ise dinamik kurallardır. Birincisini ele aldığımızda araştırma devam ederken değişken bir süre kullanılmamaktadır. Yani statik kurallarda sabit süre kullanılmaktadır. Dinamik kurallarda tam tersi bir durum söz konusudur. Araştırma esnasında değişebilen, değişken bir süre kullanımı mevcuttur. Bu her iki kural içinde tabulaştırılan bir eylem için verilen süre sona eriğinde ilgili eylem tabulaştırılan eylem kısıtlar listesinden silinmektedir (Karaboğa, 2011).

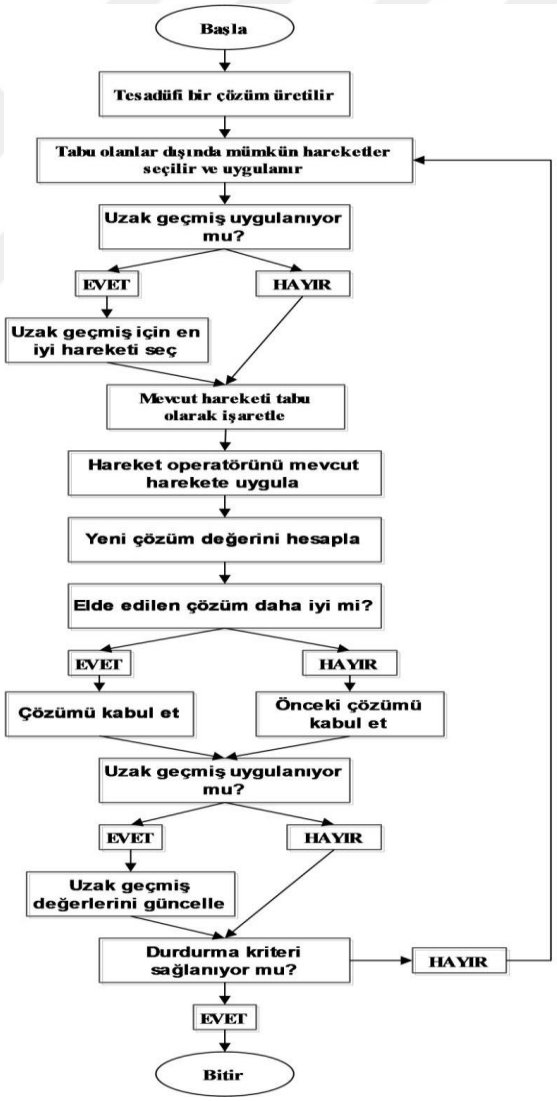
Bir süre tabu listesinde bulunan bir eylemin tabudan silinmesi için tabu yıkma kriterinin oluşturulmasına ihtiyaç duyulmaktadır. Algoritmanın çalışması esnasında tabulaştırılmış bir eylemin sonucu, o zamana kadarki tüm sonuçlar arasındaki en iyi çözüm değerinden iyiyse tabu haline getirilmiş bu eylemin tabu listesinden silinmesi gerekmektedir. Ayrıca bu eylem gerçekleştirilmelidir (Özdemir, 2013).

4.1.2 Sıklık tabanlı bellek yapısı

Tabu araştırması için iki bellek yapısından bahsedilmişti. Bunlardan ikincisi sıklık tabanlı olanıdır. Diğer hafıza yapısına göre daha genel tabu eylemlerine hitap etmekle birlikte yakın geçmiş tabanlı bellek yapısını da tamamlayıcı bir rolü bulunmaktadır. Bu

hafıza yapısı adından da anlaşılacağı gibi eylemlerin sıklığını hafızada saklamaktadır. Yapılan eylemlerin tekrarlanma sayısını değil, sonucun verimliliğine göre bilgiler hafızaya yazılmaktadır. STBY' da değinilecek dört başlık bulunmaktadır. Bunlar hareketler ile ilgilidir. Her bir hareketin toplam tekrar sayısı, en yüksek hareket tekrar sayısı, toplam hareket sayısı ve ortalama hareket tekrar sayısıdır (Tunçhan, 2008). STBY' da sürekli tekrarlanan eylemlerden başka sonuç kalitesinin yüksek olmasını sağlayan eylemler hafızada yer alır. Bu sayede en iyi sonuca yaklaşılır. Bir eylem sürekli tabu listesinde yer alıyorsa bu hareketin sonucu daha iyiye götürdüğü anlamına gelir (Özdemir, 2013).

Tabu araştırmasının aşamaları Şekil 4.1 de açıklanmaktadır.



Şekil 4.1 Tabu araştırması adımları (Özdemir, 2013)

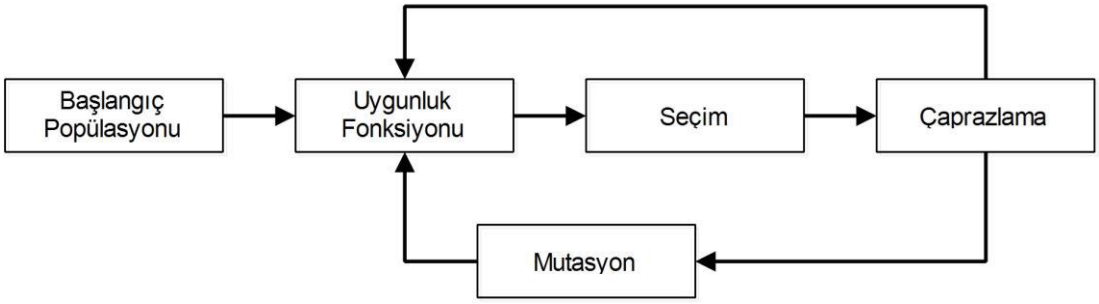
5. GENETİK ALGORİTMA

Sezgisel yöntemler genel olarak rastgele arama yaparak çözüm üretirler. Diğer algoritmalarda olduğu gibi genetik algoritma da rastgele arama yaparak sonuç bulmaktadır. Genetik algoritma 1975 yılından bu yana kullanılmaktadır. Bu süre zarfında pek çok uygulamaya dahil olmuş ve uygulamaların çalışmasına hizmet etmiştir. Dizileri kullanarak problemi kodlayıp çözüm yolları elde etmiştir. Her bir dizi problemin bir çözümüne denk gelmektedir. Elde edilen dizi çokluğu çözüm yollarını arttırır ve en iyi çözüm yoluna ulaştırır (Özdemir, 2013).

Genetik algoritma, sezgisel teknikler içerisinde yaklaşık 40 yıllık geçmişi olmasına karşın günümüzde hala popüleritesini de yitirmemiştir (Özdemir, 2013).

Genetik algoritma, doğal seleksiyona dayalı evrimsel bir algoritmadır. Güçlü olan hayatta kalır diyen evrim teorisi anlayışıyla işler. Zayıf canlılar yeni bir canlı üretemeyecek çünkü ya üreyemeyecek kadar kısa ömürlü olacak ya da üreyen canlılar hayatta kalamayacak. Güçlü canlılar hem daha uzun yaşayacak hem daha fazla üreyeceklerdir. Güçlü canlılar çoğaldıkça daha da güçlü canlılar ortaya çıkaktır. Bazen güçlü canlılardan zayıf canlılar da üreyebilir işte bunlara mutasyon denir (Özdemir, 2013).

Genetik algoritmalarda değişkenler kromozomlarla gösterilir. Her kromozom bir vektör olarak düşünülür. Vektörlerden bir kısmı alınıp popülasyon oluşturulur. Her vektörün temsil ettiği çözüm farklıdır. Temsil edilen bu çözüm kümesinin tamamı ise popülasyon olarak adlandırılmaktadır. Rastgele olarak başlangıç yani bir çözüm oluşturulur. Başlangıç çözümünden tekrarlı yöntemler kullanılarak çözüm değerleri üzerinde değişiklik yapılır. Bu sayede yeni çözüm değerleri ortaya çıkar. Kromozomlar üstünde operatörler yardımıyla çeşitlilik sağlanarak nesiller oluşturulur. Genetik algoritmanın tekrarlı yapısı ile daha iyi sonuçlar elde edilmeye çalışılmaktadır. Genetik algoritmanın tekrarlı yapısı aşağıda Şekil 5.1 de gösterilmektedir.



Şekil 5.1 Genetik algoritmanın tekrarlı yapısı

Yukarıda genetik algoritma ile alakalı genel bilgiler aktarılmıştır. Konuyu daha ayrıntılı olarak incelemek adına genetik algoritma terminolojisinde yer alan kavramlar incelenecektir (Vagifoğlu, 2010).

5.1 Genetik Algoritma Terminolojisi

Evrimsel Algoritmalar: Çözüm uzayında içerisinde rastgele arama yapma prensibine dayanarak çözüm bulmaya çalışılır. Birçok probleme uygulanabilecek kolaylıktadır çünkü az sayıda parametreye ihtiyaç duyar. Bu durum büyük bir avantaj oluşturmaktadır (Özdemir, 2013).

Kromozom: Canlılardaki kalıtımsal bilgi kromozomlar aracılığıyla taşınır. Genler birleşip kromozomları oluşturur (Özdemir, 2013).

Gen: Kromozoma ait tek bir bilgidir. Canlıların kalıtsal **özelliklerini** taşırlar. Genler karakterlerin canlıdan canlıya aktarılmasında etkilidirler (Özdemir, 2013).

Alel: Bir gen içindeki, yani ona biçilen bir değerdir. Kromozom içindeki genin farklı versiyonudur. Kalıtımda karakter oluşumunda önemli bir role sahiptir. Örnek olarak bir Saç rengi geninin anne veya babadan gelmesine rağmen farklı olmasının sebebidir (Özdemir, 2013).

Genotip: Kromozomda genler bulunur. Bu genler yan yana gelmesiyle anlamlanırlar. Topluluk haline gelip anlamlanmasıyla genotip oluşur. Örnek olarak genlerin bilgisayar diline göre 0 ve 1 ler den oluştuğunu düşünelim. Genler bir araya gelerek kromozomları oluşturduğuna göre kromozomlardaki genler ise “11100010101” şeklinde dizilir ve bu dizilimle genotip ortaya çıkmaktadır (Özdemir, 2013).

Fenotip: İsim olarak genotipe benzemektedir. Anlam olarak da genotipin anlamlanmasıyla dışarıya oluşturduğu fiziksel görünümü ve duyu organlarıyla algılanabilir özelliklerin tümüdür. Örneğin saç renginin sarı olması gibi özellikler (Özdemir, 2013).

Şema: Kromozomlarda ki genler ifade edilirken şemalara ihtiyaç duyulur. Örnek vermek gerekirse "11100101" ve "00101010" iki kromozom yapısı "**1*****" olarak gösterilebilmektedir. Bu gösterim şema ile oluşturulmuştur (Özdemir, 2013).

Ebeveyn Seçimi: Bu değer anne ve babayı seçmek için kullanılmaktadır. Bu değer hangi ebeveyn için daha yüksek ise o ebeveynin seçilme ihtimali artmış olup seçilecek ebeveyn genlerini gelecek nesillere aktarmaktadır (Özdemir, 2013).

5.2 Genetik Algoritma Operatörleri

İlgili operatörlerinden en çok kullanılan ve önem taşıyanları mutasyon ile çaprazlamadır. Bu operatörler nesillerin geçişlerinde oluşturulan popülasyonlarda aktif rol oynamaktadırlar.

İlk olarak çaprazlamaya değinilirse isminden de anlaşılacağı gibi farklı yöntemlerle ebeveynler arasında gen çeşitliliği oluşturarak bireyin kromozomlarındaki gen dizilimini oluşturmaktadır.

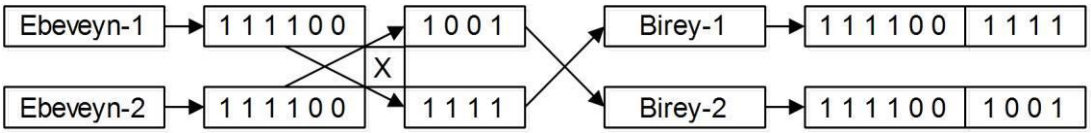
Diğeri ise mutasyondur. Bu operatör kromozomda rasgel seçilmiş bir genin bozulmasını sağlamaktadır. Bu sayede genetik çeşitlilik yapmaktadır.

Mutasyon ile çaprazlama genetik algoritmanın doğru çalışmasını ve optimum sonuçları en kısa sürede bulması için büyük önem taşımaktadır. Bu operatörler sayesinde genetik çeşitlilik artmaktadır. En iyi en doğru çözüme ulaşabilmek GA' nın amacıdır. Mutasyonun önemli bir görevi vardır. Mutasyon sayesinde yerel maksimum ya da yerel minimuma takılması engellenmektedir (Özdemir, 2013).

5.2.1 Tek noktalı çaprazlama

Kromozom içerisinden rastgele nokta seçilir. Bu noktaya göre kromozomlar yer değiştirir. Çaprazlama noktasına göre ebeveynlerin kromozomlarının farklı

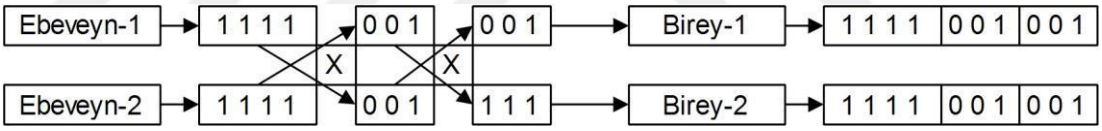
bölümlerinden çaprazlama yapılarak bireyin kromozomları oluşturulur. Yeni bireyin oluşması için kullanılan kromozomların dışında kalanlarda diğer bireyin oluşumu için kullanılmaktadır. Örnek aşağıda verilmiştir (Özdemir, 2013).



Şekil 5.2 Tek noktalı örnek (Tunçhan, 2008)

5.2.2 Çift noktalı çaprazlama

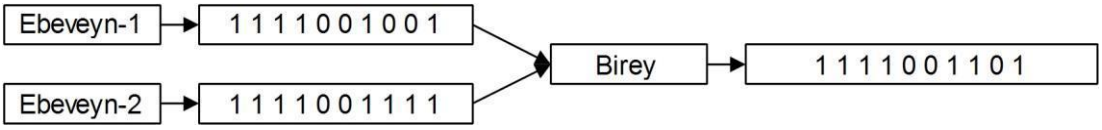
Tek noktalıda ki gibi mantık olarak aynıdır. Belirlenen noktadan ebeveynler arasında kromozomların çaprazlanması yapılmaktadır. İki yöntem arasındaki tek ayrım tek noktalı çaprazlama yönteminde tek noktadan ayrılma yapılırken çift noktalı çaprazlama yönteminde iki noktadan olmasıdır. Bir önceki örnekte ki tek noktadan çaprazlama yöntemi biraz değiştirildiğinde aşağıdaki çift noktalı çaprazlama yönteminin örneği gibi olacaktır (Özdemir, 2013).



Şekil 5.3 Çift noktalı örnek (Tunçhan, 2008)

5.2.3 Tek biçimli çaprazlama

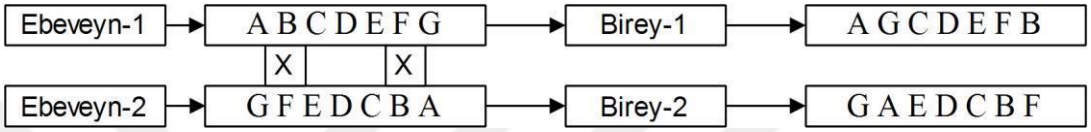
Tek biçimli çaprazlama ile gelişi güzel çaprazlama yapılmaz. Yani ilgili yöntemde çaprazlama yapılırken bir noktaya ihtiyaç duyulmaz. Ebeveynlerin kromozomları rastgele bireye aktarılır ve bireyin kromozom yapısı ortaya çıkar. Tek noktalı ve çift noktalı çaprazlamadaki örneğimizi tek biçimli çaprazlama için düzenlersek aşağıdaki gibi olmaktadır (Özdemir, 2013).



Şekil 5.4 TBÇ örneği (Tunçhan, 2008)

5.2.4 Sıralı kromozom çaprazlama

Başlıktan da anlaşıldığı üzere kromozomların çaprazlanması sıralı yapılmaktadır. Ancak bu durum her zaman gerçekleşmez. Sıralama, işin önemi ve ihtiyaçlarına yani önceliklerine göre belirlenmektedir. Aşağıda bir örnek verilmiştir. Bu örneğe göre işler A dan F ye kadar olan alfabetik harflerle belirtilmiştir. Toplamda 7 harf bulunmaktadır. Bu işin sıralı çaprazlaması yapılacak olursa "A B C D E F G" dizilimi elde edilir. Birinci A, sonuncu G olmuştur. SKÇ türü kromozom sayısının ve sıralamanın bozulmaması gerektiği durumlarda tercih edilebilmektedir.



Şekil 5.5 SKÇ örneği (Engin, Fığlalı, 2002)

5.3 Genetik Algoritma Parametreleri

Genetik algoritma parametreleri, Genetik Algoritmanın etkinliğini, başarısını yüksek oranda etkilemektedir. Başlıca genetik algoritma parametreleri; seçim, çaprazlama, mutasyon ve popülasyon büyüklüğüdür. Her bir parametrenin kendi içinde farklı özellikte çeşitleri vardır. Bunlardan hangilerinin kullanılacağı sonucu doğrudan etkilemektedir (Özdemir, 2013).

Seçim: Bu parametreden anlaşılması gereken kromozomların seçiminin nasıl yapılacağı olmalıdır. Genetik algoritmanın temel prensibi en iyinin hayatta kalmasıdır. Bundan dolayı nasıl bir seçim yapılırsa en iyi sonuca ulaşılır, en sağlıklı bireye hangi seçimle ulaşılır, ebeveynlerinden daha üstün bireylere ulaşılması için hangi seçimlerin yapılması gereklidir sorularının cevapları aranmaktadır. Seçimin çeşitli yöntemleri vardır. Bu yöntemlerden bazıları; stokastik seçim, sıralama seçimi, turnuva seçimi, sabit durum seçimi ve rulet tekerleği seçimi olarak sıralanabilir. Bunlardan en yaygın olarak kullanılanları rulet tekerleği ve turnuva seçimidir. İki yöntemin tek nokta çaprazlama ile sonuçları Çizelge 5.1 de yer almaktadır.

Parametreler:

Nesil Sayısı: 10

Birey Sayısı: 10

Mutasyon Sayısı: 1

Çaprazlama Olasılığı:0.9

Mutasyon Olasılığı: 0.05

Çizelge 5.1 Seçim yöntemleri karşılaştırılması

	Toplam Çalışma Zamanı	Optimum Sonuç Puanı	Optimum Sonucun Nesil Sayısı
Rulet Tekerleği	9.14 sn	1216	9
Turnuva	9.1 sn	1216	9

Çizelge 5.1 de ki karşılaştırmada turnuva ve rulet tekerleğinin seçim yöntemlerinin yaklaşık sonuçlara ulaştığı görülmektedir. Hem bu çalışma için kullanımı daha kolay olduğundan hem de ufakta olsa daha iyi sonuçlar ürettiği için problem çözümünde turnuva yöntemi kullanılmıştır. Farklı problemlerde farklı sonuçlar çıkabilmektedir.

Çaprazlama: Çaprazlama genetik algoritmada olan en önemli parametrelerden biridir. Çaprazlama amaç olarak çeşitliliği sağlamak ve yeni nesillerin eski nesillerde farklılaşmasını sağlamaktır. Eski nesilden çaprazlama sayesinde yeni ve farklı nesiller üretilerek çeşitlilik arttırılır. Çeşitlilik artınca doğal olarak daha geniş bir çözüm aralığı olacak ve istenilen sonuca ulaşma olasılığı da artmış olacaktır. Bu operatör ile birey hangi ebeveynden hangi kromozomu alacak bu belirlenir. Bu noktada çaprazlama oranı ne kadar artarsa o derece farklı kombinasyonda kromozomlar oluşacaktır. Çaprazlama oranı arttıkça; çözüme ulaşmak için alternatif birçok yöntem denendiğinden dolayı çözüme daha kısa hangi yoldan ulaşılacağı da ortaya çıkmış olur (Özdemir, 2013).

Seçim sonucu yeni bireylerin oluşmasını sağlamak amacıyla ebeveynler çaprazlamaya girer. Uygunluk değeri esas alınarak en iyi sonucu bulabilmek için çaprazlama yapılmaktadır. Pek çok çaprazlama yöntemi vardır. Aralarında en kolay olanı ise tek noktalı çaprazlamadır.

Mutasyon: Mutasyon parametresi, mutasyonun sayısını belirler. Mutasyonda bireyin genleri ebeveynlerinden aktarılırken değişikliğe uğrar. Mutasyon arttıkça bireyin gen yapısı da değişir. Mutasyon düşük oldukça bireyin genleri ebeveynlerinin genlerine daha çok benzer. Bu durum çözüm aralığını daraltır ve sonuca ulaşmada engel teşkil eder. Mutasyon ile yerel minimum e yerel maksimum aralığını açar. At gözlüğü gibi düşünülebilir. Mutasyon arttıkça at gözlüğü açılır ve daha fazla yer görülür. Görülen yer arttıkça da çözüme giden en iyi yol bulunur. (Özdemir, 2013)

Genetik farkındalık ve çeşitliliği sağlan ve bu çeşitliliği koruyan en önemli operatörlerden biride mutasyondur. Mutasyon ile bireyin genlerinden bir veya birkaçı değiştirilir. Böylece yeni kromozomlar elde edilir. Bu operatör tüm bireylere uygulanmaz sadece belli bir yüzdeye uygulanır. Bu nedenle de mutasyonun etkileri kromozomlarda az görülür. Mutasyona uğrayan kromozomda gen sayıları değişmez.

Çaprazlama operatörü uygulandıktan sonra oluşan bireye uygulanan mutasyon örneği Şekil 5.6 da gösterilmektedir. Aşağıdaki örnekte birinci çocuğun rastgele üçüncü ve altıncı genin seçildiğini varsayalım. Mutasyona uğrayan üçüncü ve altıncı genler yer değiştirerek çocuğun kromozomu son halini almıştır (Ç.Ü. Sosyal Bilimler Enstitüsü Dergisi, 2010).

Ç1 = 1 2 **8** 6 7 **4** 5 9 3 Mutasyona uğramış Ç1 = 1 2 **4** 6 7 **8** 5 9 3

Şekil 5.6 Mutasyon örneği

Uygunluk Fonksiyonu: Çözüm kümesine *X* denirse, ilgili çözüm kümesinde değerimiz *x* ile ifade edilmektedir. İlgili fonksiyon *f(x)* ile gösterilecektir. Genetik algoritma, tüm bireylerin kromozomlarındaki sonuçları derler ve aralarından rastgele seçim yapmaktadır. Seçmiş olduğu sonuçların değerlerini hesaplamaktadır. Uygunluk fonksiyonu çözüm değerleri arasından doğru sonuca en yakın değeri bulmaktadır (Khoo, Suganthan, 2002).

Popülasyon Büyüklüğü: Önemli konulardan biride genetik algoritma için popülasyon büyüklüğüdür. Popülasyonun boyutu algoritmanın hızı, performansı ve sonuca olan katkısı bakımından oldukça önemlidir. Popülasyon boyutu ne kadar büyük olursa sonuca ulaşmak için geçen sürede o kadar artmaktadır. Popülasyon büyüklüğü düşük ise de araştırmada yeterince kısım temsil edilmiyor demektir (Colin, Reeves, Jonathan. Rowe, 2003).

Farklı popülasyon büyüklükleri ile alınan sonuçlar incelenmiş olup aşağıdaki Çizelge 5.2 de yer almaktadır.

Parametreler:

Nesil Sayısı: 10

Mutasyon Sayısı: 1

Çaprazlama Olasılığı:0.9

Mutasyon Olasılığı: 0.05

Çizelge 5.2 Birey sayısı karşılaştırılması

Birey Sayısı	Toplam Çalışma Zamanı	Optimum Sonuç Puanı	Optimum Sonucun Nesil Sayısı
6	2.6 sn	642	2
10	9.1 sn	1216	9
20	822.0 sn	2557	10

Yapılan karşılaştırmaya göre algoritmanın çalışma performansı ve sağlamış olduğu puan dikkate alınarak bu çalışmada başlangıç popülasyonu büyüklüğü olarak 10 belirlenmiştir.

6. AÇIKLAYICI ÖRNEK

Sonuç programının oluşturulabilmesi için sadece seyahatlerin belirtilmesi yeterlidir.

Örnek:

Çizelge 6.1 Örnek seyahat programı

ID	Başlangıç Noktası	Bitiş Noktası
1	A →	B
2	D →	E
3	B →	C
4	C →	A
5	B →	D
6	C →	D
7	E →	A
…	…	…

Seyahatlerden bir tanesi başlangıç olmalıdır. Belirlenen seyahatin ilk şehri aynı zamanda turistin seyahat bitiş şehri olacaktır. Seyahat sonunda turist başladığı noktaya dönecektir.

Çizelge 6.2 Örnek seyahat rotası

1	5	2	7

Çözüm:

Rastgele değerlerle oluşturulmuş B bütçeyi geçmeyen ∑ tatmin puanı maksimum olan başlangıç şehrinden tatile çıkış tarihi D_s den sonra, başlangıç şehrine dönüş tarihi D_f den önce olan çözüm,

Çizelge 6.3 Örnek seyahat çözümü

Eylem	ID	Başlangıç	Bitiş	Süre	Tutar (TL) (C_i*d_i)	Tatmin ($S_i + g*d_i$)
Konaklama		A	-	24 saat	40x2=80	80+6x2=92
Seyahat	1	A	B	1 saat	85	0
Konaklama		B	-	48 saat	35x2=70	68+6x2=80
Seyahat	5	B	D	1 saat	82	0
Konaklama		D	-	29 saat	50x1=50	75+5x1=80
Seyahat	2	D	E	1 saat	84	0
Konaklama		E	-	82 saat	33x3=99	52+7x3=73
Seyahat	7	E	A	1 saat	95	0

Toplam Tatmin: 325

Toplam Süre: 7 gün 19 saat <= $D_f - D_s$

Toplam Harcama: 645 TL <= B

Toplam Şehir Sayısı: 4

7. ÖNERİLEN GENETİK ALGORİTMA YAKLAŞIMI

Bu problemin çözümünde Türkiye' de seçilmiş şehirlerin gerçek koordinatları ve tahmini maliyet ile kazanılan puanlar üzerinden bir turistin en çok puanı toplayarak seçilen şehirleri belirlediği tarih aralığında gezmesi amaçlanmıştır. C# dili ile yapılan uygulamada veriler program içerisinde depolanmıştır. C# kodları nesne tabanlı bir algoritma ile geliştirilmiştir. Genetik Algoritmanın uygulamasında Takım Oryantiring yaklaşımı ile Turnuva metodu ile en iyileme çalışması yapılmıştır. Hızlı bir şekilde iyi sonuçlar üreten algoritmanın kısa sürede yüksek puanlı rota çıkardığı gözlemlenmektedir.

7.1 Popülasyon

Bireylerin tutulduğu bir listedir. Problemin çözümünde popülasyon bir liste olarak tanımlanmıştır. Yapılan tanım aşağıdadır. Popülasyon listesi üzerinden erişilen fonksiyonlarla elitizm ve diğer işlemler yapılmaktadır.

7.2 Birey

Kromozomun uygunluğu, güzergahı, puanı gibi değerlerin tutulduğu bir sınıftır. Bireyler popülasyon içinde birey sınıfından bir liste içinde tutulur. Bu sayede dinamik bir şekilde birey sayısı değişebilmektedir.

Programda birey bir sınıf olarak tanımlanmıştır. Problemin çözümünde birey sınıfı popülasyonun elemanlarını oluştururken bireyin içinde ise kromozomlar ve diğer bilgiler bulunmaktadır. Programın başlangıcında sorulan birey sayısı sorusunun cevabına göre dinamik bir halde birey sayısı belirlenebilmektedir. Programda başlangıç olarak belirlediğimiz her bir nesil için 10 birey olması şeklindedir. Birey sınıfının elemanları ve tanımı aşağıdadır.

7.2.1 Birey sınıfı

Bireylerin başlıca özelliklerinin yer aldığı sınıf. Bu özellikler; kromozom, tatmin düzeyi, toplam yol, kalan para, son şehir ve güzergahtır. **(EK 1)**

7.2.2 Birey tanımı

Birey sınıfı içerisinde yer alan kromozom bir dizidir. Şehirlerin id bilgilerini tutan kromozom, çaprazlama mutasyon ve tatminlik puanı hesaplama fonksiyonlarında gerekli işlemlere tabi tutulmaktadır. Bir diğer eleman ise güzergahtır. Güzergah, tatminlik puanı hesaplama fonksiyonumuz kromozomda yer alan şehirlere sırası ile seyahatler gerçekleştirirken bir yandan da gidilen şehirleri güzergah sınıfını kullanan güzergah elemanımıza atmaktadır. Bu sayede hangi şehirde ne kadar kalınıp ne kadar para harcandığı gibi bilgileri de tutabilmekteyiz. Güzergah sınıfının elemanları ve kodu ekte yer almaktadır**. (EK 2)**

7.3 Kromozom

Programda birey sınıfı içerisinde tutulan kromozomlar, şehirlerin gen listesini oluşturmaktadır. Bu liste sayesinde belirlenen ilk şehirden son şehire doğru bir rota oluşturulmaktadır.

Kullanılan çözümde kromozom, birey sınıfının tek boyutlu bir dizisi olarak tanımlanmıştır. Kromozom şeklinde adlandırılan dizi, içerisinde şehir id lerini bulundurmaktadır. Başlangıçta kullanıcının girmiş olduğu birey sayısına göre değişkenlik gösteren kromozom dizisi, çaprazlama ve mutasyon gibi çeşitli fonksiyonlara gönderilmektedir.

Her bir iterasyonda dizinin bir genine işlem yapılmaktadır. Dizinin her bir elemanında bulunan şehir id leri genleri oluşturmaktadır. Aşağıda ilk neslin oluşturulduğu kod gösterilmektedir. If içerisinde ise kromozomlara yerid yani şehrin id alanı eklenmektedir**. (EK 3)**

Ek 3’ deki kod parçacığında BireySayisi değişkeni başlangıçta kullanıcıdan alınan değerdir. Bu değere göre her bir neslin birey sayısı belirlenmektedir. Ayrıca ilk şehir başlangıç noktası olarak seçilmektedir. Bu şehir dışında kalan diğer şehirler arasından rastgele seçim yapılarak kromozomun genlerine bir bir ekleme yapılmaktadır.

7.4 Gen

Yine bir sınıf olarak tasarlanan gen bir şehri ifade etmektedir. Gen içerisinde şehir koordinat bilgisi, başlangıç puanı, günlük maliyet ve puan gibi bilgiler bulunmaktadır. Gen yapısı Çizelge 7.1 de gösterilmiştir.

Çizelge 7.1 Gen yapısı

Adı	Enlem	Boylam	Tatmin Başlangıç	Tatmin Günlük	Günlük Harcama

Kullanılan gen yapısı aşağıda açıklanmıştır.

7.4.1 Şehir adı

Gendeki şehir bilgilerinden şehrin adı yer almaktadır.

7.4.2 Şehir enlemi

Şehrin koordinatının enlem bilgisi tutulmaktadır. Bu bilgi sayesinde turistin güzergahı boyunca kaç km yol yaptığını ölçebiliyoruz.

7.4.3 Şehir boylamı

Şehrin koordinatının boylam bilgisi tutulmaktadır. Bu bilgi sayesinde turistin güzergahı boyunca kaç km yol yaptığını ölçebiliyoruz.

7.4.4 Şehrin tatmin başlangıç puanı

Gendeki şehrin başlangıç tatmin puanı tutulmaktadır. Bu değer, program turistin güzergahı boyunca dolaştığı şehirlere gittiğinde alacağı ilk puandır. Bu puan uçaktan indiği ilk dakika itibariyle aktif hale gelir.

7.4.5 Şehrin tatmin günlük puanı

Gendeki şehrin başlangıç puanından sonra şehirde geçirdiği gün sayısı bir artana kadar tatmin günlük puanı aktif hale gelmez. İlk günden sonraki her bir gün için tatmin günlük puanı çarpılarak hesaplanır. Gendeki şehre ilk geliş tarihi 07.06.2018 saat 20.00 gidiş

tarihide 10.06.2018 saat 11.00 olduğunu varsayarsak (tatmin günlük puanı) x 3 olarak hesaba dahil edilmektedir. Yani her bir gece için 1 tatmin günlük puanı hesaplanmaktadır.

7.4.6 Şehrin günlük harcaması

Günlük harcamanın aktif hale gelebilmesi için tıpkı tatmin günlük puanının hesaplanmasında olduğu gibi ilk günden sonraki her bir gün için günlük harcama çarpılarak hesaplanmaktadır. Yine tatmin günlük harcamada verdiğimiz örnekte olduğu gibi şehre ilk geliş tarihi 07.06.2018 saat 20.00 gidiş tarihide 10.06.2018 saat 11.00 olduğunu varsayarsak (günlük harcama) x 3 olarak şehirde harcanan para bulunmaktadır. Yani her bir gece için 1 günlük harcama hesaplanmaktadır. **(EK 4)**

7.5 Çaprazlama

Birey sınıfı içerisinde bulunan tek boyutlu diziye uygulanmaktadır. Çalışmada tek noktadan ve iki noktadan çaprazlama operatörleri ayrı ayrı denenmiş olup Çizelge 7.2 deki sonuçlar elde edilmiştir. Tek noktadan çaprazlama operatörünün daha iyi sonuç vermesi üzerine bu yöntem kullanılmıştır.

Parametreler:

Nesil Sayısı: 10

Birey Sayısı: 10

Mutasyon Sayısı: 1

Çaprazlama Olasılığı:0.9

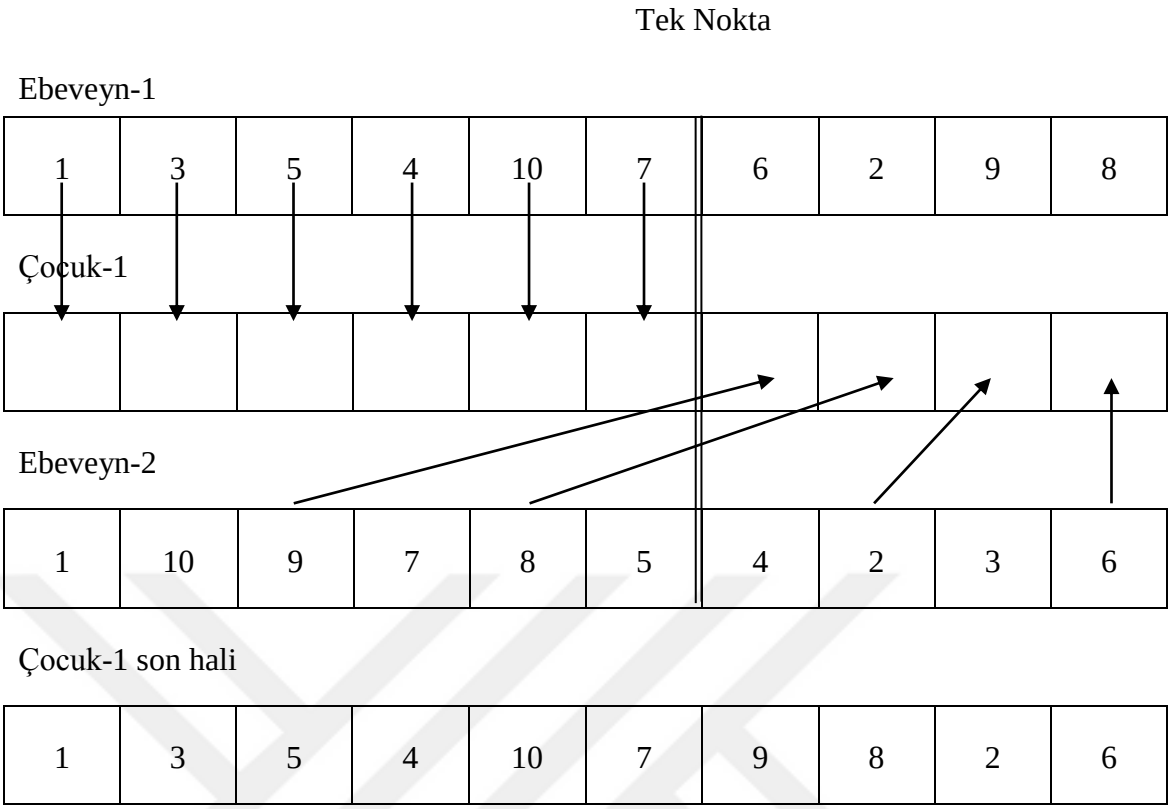
Mutasyon Olasılığı: 0.05

Çizelge 7.2 Çaprazlama operatörleri karşılaştırması

	Toplam Çalışma Zamanı	Optimum Sonuç Puanı	Optimum Sonucun Nesil Sayısı
Tek Nokta Çaprazlama	9.1 sn	1216	9
Çift Nokta Çaprazlama	10.3 sn	1216	9

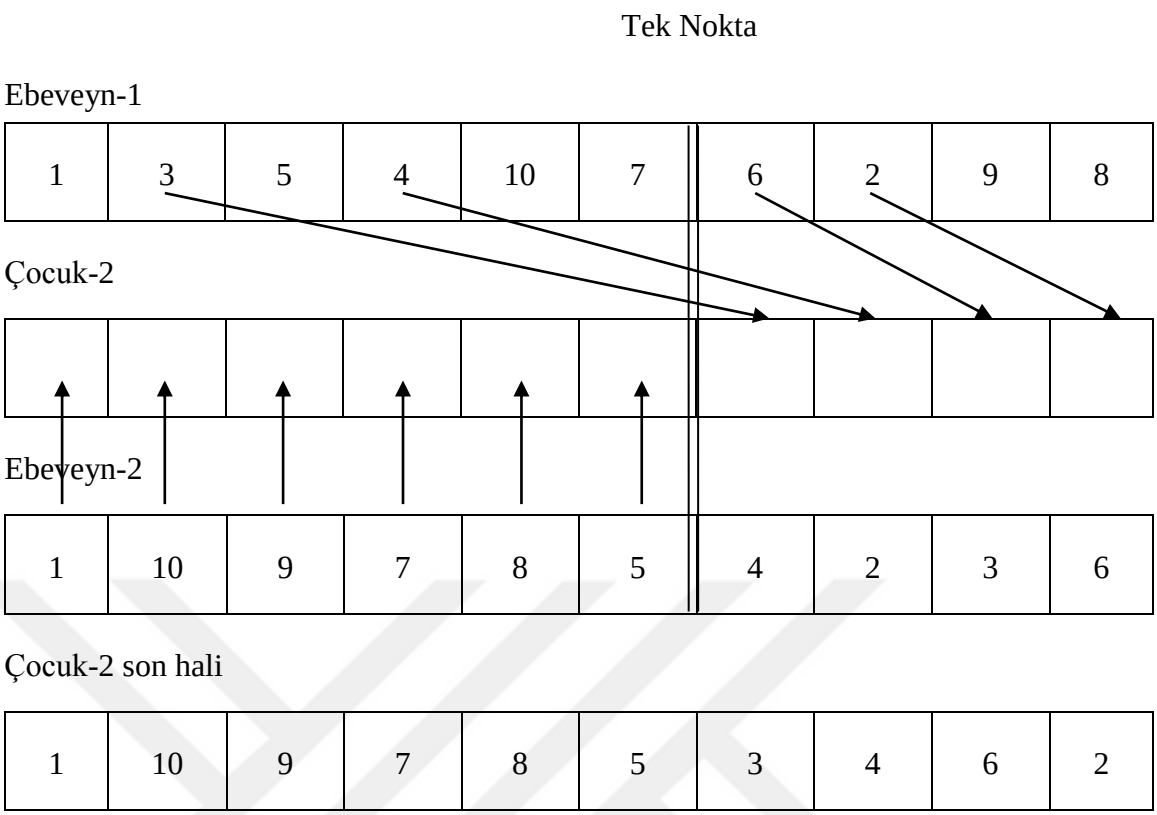
Çaprazlama oranı başlangıçta kullanıcıdan istenmektedir. Bu oran sağlandığında çaprazlama fonksiyonuna gelen iki bireyin çaprazlaması gerçekleşmektedir. Çaprazlama oranı sağlanmadığı takdirde gelen iki birey döndürülmektedir. **(EK 5)**

EK 5' de görüldüğü gibi tek nokta çaprazlama operatörü kullanılmıştır. Şekil 7.1 de çaprazlamanın nasıl yapıldığı örnek üzerinde açıklanmıştır.



Şekil 7.1 Tek nokta çaprazlama örneği - 1

Yukarıda Şekil 7.1 de görüldüğü üzere tek nokta indise kadar olan genler ebeveyn-1 den çocuk-1 e geçmiştir. Çocuk-1 de tek nokta indisten sonraki boş genler ise ebeveyn-2 nin genlerini sırayla dolaşıp çocuk-1 de olmayan genleri bulup boş olan yerlere yerleştirmektedir.



Şekil 7.2 Tek nokta çaprazlama örneği - 2

Yukarıda Şekil 7.2 de görüldüğü üzere tek nokta indise kadar olan genler ebeveyn-2 den çocuk-2 e geçmiştir. Çocuk-2 de tek nokta indisten sonraki boş genler ise ebeveyn-1 in genlerini sırayla dolaşıp çocuk-2 de olmayan genleri bulup boş olan yerlere yerleştirmektedir.

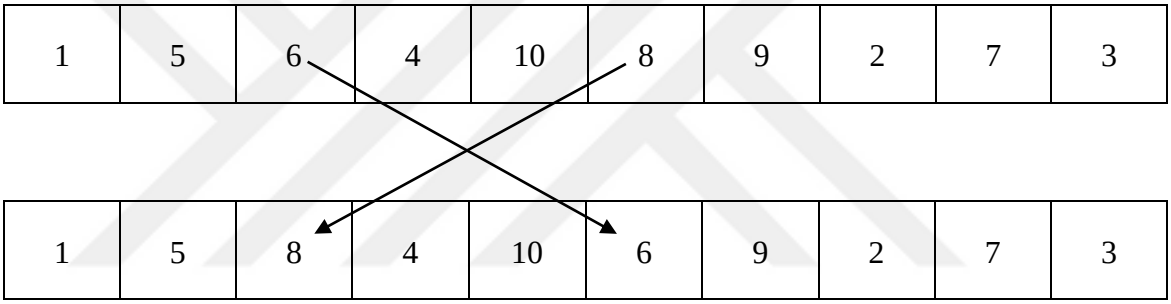
Farklı büyüklükteki kromozomların çaprazlamaya girmesi, fazla gen sayısına sahip ebeveynin tüm genlerinin kullanılamamasına yol açacağından ve farklı büyüklüğe sahip çocukların oluşacağından dolayı, çaprazlamaya girecek ebeveynlerin aynı gen sayısına sahip olması sağlanmaktadır. Bu yüzden tüm şehirlerin genlerde olması sağlanmaktadır.

7.6 Mutasyon

Birey sınıfı içerisinde bulunan tek boyutlu kromozom dizisine uygulanmaktadır. Mutasyon oranı başlangıçta kullanıcıdan istenmektedir. Bu oran sağlandığında mutasyon başlangıçta kullanıcıdan alınan mutasyon sayısı kadar gerçekleşmektedir. **(EK 6)**

Ek 6’ da mutasyon olasılığı sağlanması halinde mutasyon fonksiyonu çağırılmaktadır. Daha sonrasında popülasyona eklenmektedir. Mutasyon olasılığı sağlanmadığında ise birey popülasyona mutasyona uğramadan eklenmektedir. **(EK 7)**

Ek 7’ de görüldüğü gibi mutasyon başlangıç şehri haricindeki genlerle yapılmaktadır. 2 adet rastgele gen seçilir ve bu genler arasında şehirler değiştirilir. Behdadnia M. (2016), zaman kısıtlı oryantiring problemi çözümünde de kullandığı gibi aynı şehre 2 kez gitme durumu olduğundan dolayı 1 geni değiştirmek yerine 2 gen arasında değişim yapılarak mutasyon sağlanmaktadır. İşlemin ardından mutasyona uğrayan bireyin tatmin puanı sıfırlanmaktadır. Bunun sebebi ise yeni bir kromozom yapısı oluşmasından ötürüdür. Mutasyonun yapılması Şekil 7.3 de gösterilmiştir.



Şekil 7.3 Mutasyon örneği

Şekil 7.3 de görüldüğü gibi birey mutasyona uğramıştır. Böylece yeni bir kromozom, dolayısıyla yeni bir birey üretilmiştir.

7.7 Tatmin Puanı Hesaplama Fonksiyonu

Kromozomda yer alan şehirler gezilerek uygun güzergah ve toplam tatmin puanı hesaplanır. Aslında en önemli fonksiyon bu diyebiliriz. Kromozomdaki genlerin içinde yer alan şehirler kromozomun başlangıç indisi olan 0 dan başlayarak bütçe, uçak seferleri ve tatil bitiş tarihi göz önüne alınarak bireyin tatmin puanı hesaplanır. İterasyon kromozomun genlerini baştan sona dolaşırken bir yandan da güzergah listesine gidilen şehirleri eklemektedir. **(EK 8)**

EK 8’ de görüldüğü üzere toplam yol hesabı burada yapılmaktadır. Bu değişken sayesinde turistin gittiği şehirler arasında ki mesafeler toplanarak yaptığı toplam yol bilgisine ulaşılmaktadır.

7.8 Başlangıç Noktası Kontrol Fonksiyonu

Her bir gen için sonraki gende bulunan şehirden başlangıç noktasına dönebiliyor mu kontrolü yapılmaktadır. Bu sayede yolda kalma olasılığı ortadan kaldırılmaktadır. Eğer sonraki şehirden başlangıç noktasına belirtilen bütçe ve tatil bitiş süresi içinde dönüş gerçekleşemiyorsa mevcut iterasyon sonlanmaktadır. Bu durumda seçilen şehirlerin tamamını gezememe durumu oluşmaktadır. Bütçesi veya tatil süresi yetmeyen turist başlangıç noktasına dönmektedir. Bu durum Deney 5’ de gösterilmektedir. Eğer sonraki şehirden başlangıç noktasına bütçe ve tatil bitiş süresi içinde dönebiliyorsa sefer gerçekleştirilir ve iterasyon sonraki gene geçer. Başlangıç noktası kontrol fonksiyonu, tatmin puanı hesaplama fonksiyonu içerisinde çağırılmaktadır. **(EK 9)**

7.9 Turnuva

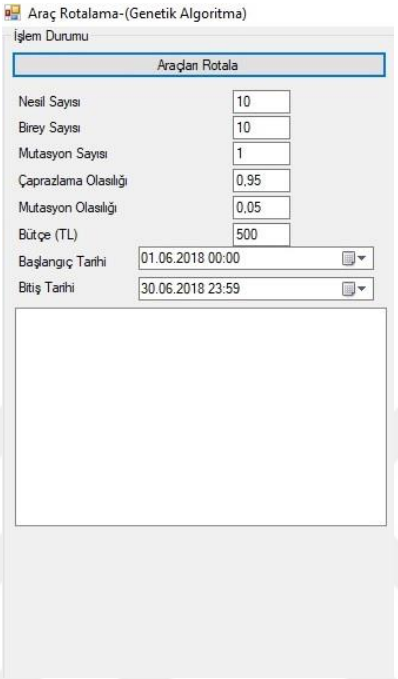
Turnuva metodu takım oryantiring problemlerinde nesil sayısı arttıkça daha kaliteli kromozomları çıkaran yöntemlerden biridir. Turnuva fonksiyonu popülasyon içerisinden rastgele seçtiği 2 bireyden tatmin puanı daha yüksek olanı yani daha iyi olan bireyi gelecek nesillere aktarmaktadır.

7.10 Mesafe Hesaplama

Mesafe hesaplama fonksiyonu, mevcut şehir ve bir sonraki şehir arasındaki mesafeyi hesaplamaktadır. Fonksiyon mevcut şehrin enlem ve boylam bilgisi ile bir sonraki şehrin enlem ve boylam bilgisini alıp gerekli matematiksel hesaplamalardan sonra iki şehir arasındaki mesafeyi dönmektedir. Bireyin toplam yolu her bir gen de toplanarak sonuç elde edilmektedir. **(EK 10)**

8. DENEYSEL ÇALIŞMA

Bu tezde Genetik Algoritma kullanan bir uygulama yapılarak turist rotalama problemi çözülmüştür. Geliştirilen bu uygulamanın ara yüzü aşağıda Şekil 8.1 de gösterilmiştir.



Şekil 8.1 Uygulama görüntüsü

Yukarıda Şekil 8.1 de görülen uygulama C# yazılım dili ile yazılmıştır. Uygulama statik olarak mevcuttaki verileri kullanarak problemi çözmektedir. Programın istemiş olduğu girdiler; birey sayısı, nesil sayısı, mutasyon olasılığı ve sayısı, bütçe, çaprazlama olasılığı, tatil başlangıç ve bitiş tarihi gibi parametreler ile uygulama çalışmaktadır. Program içerisinde yer alan algoritma sona erdiğinde bulmuş olduğu rotalar arasından en iyi 3 seçeneği kullanıcıya vermektedir. Bunlardan tatmin puanı en yüksek olan rotanın tüm şehirlerde geçirdiği süre, bindiği uçak bileti ücreti, geçirdiği yolculuk süresi, bulunduğu şehre geliş gidiş tarihleri, şehirde harcadığı para bilgileri kullanıcıya sunulmaktadır. Deneyler işlemcisi Intel I7 6700 2.6 GHZ 4 çekirdek 12GB ram e sahip bir bilgisayarda gerçekleştirilmiştir.

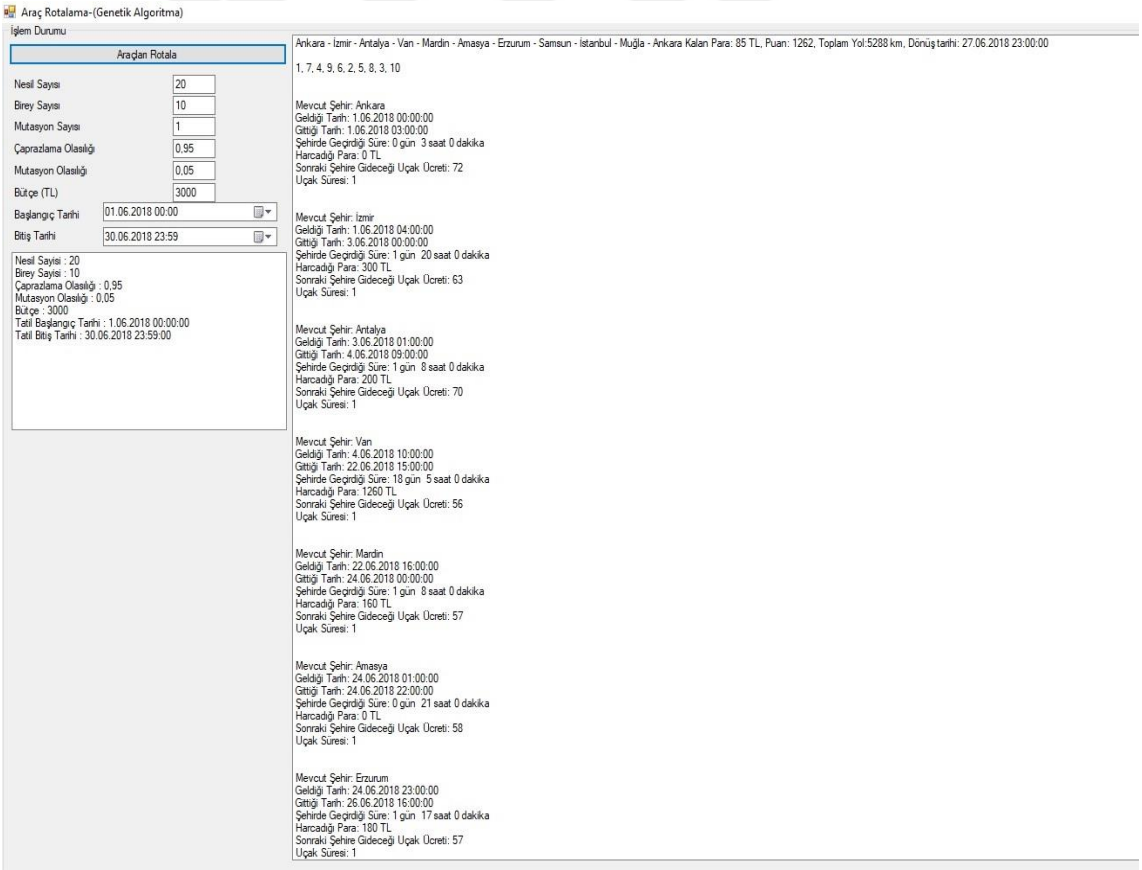
Veri setinde Amasya, Ankara, Antalya, Erzurum, İstanbul, İzmir, Samsun, Mardin, Muğla, Van şehirleri yer almaktadır. Uçak sefer saatleri saat başı olacak şekilde,

ücretleri ise 50 ile 200 arasında program başında rastgele belirlenmektedir. Uçuş süresi sabit 1 saat yapılmıştır.

Yapılan deneylerden bazılarının sonuçları aşağıda yer almaktadır.

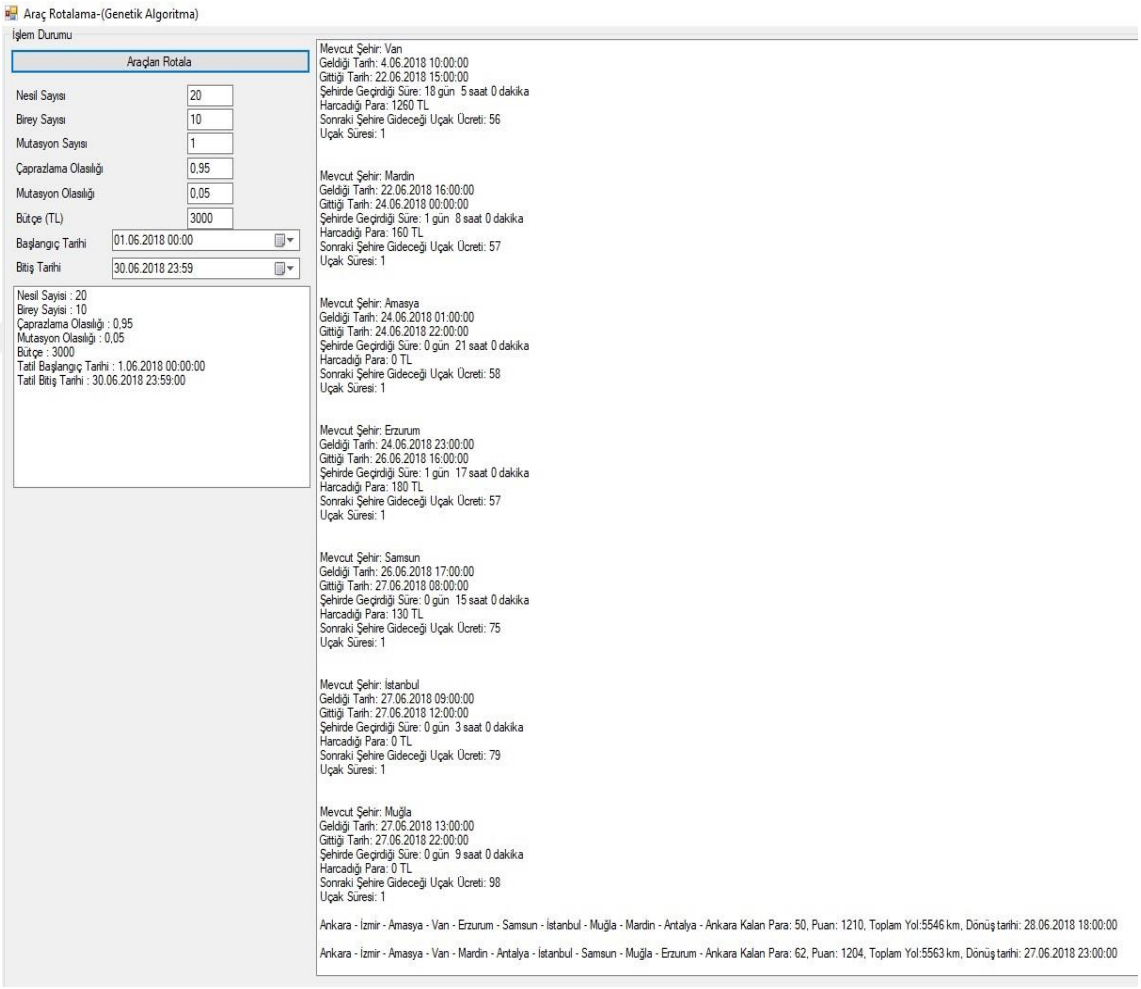
Deney 1

Deney 1 de bütçe 3000 TL, tatil için ayrılan süre 30 gün, algoritmanın kaç nesil tekrar edeceği sayı 20, her bir nesilde olan birey miktarı 10, çaprazlama olasılığı %95, mutasyon olasılığı ise %5 dir. Bu şekilde algoritma çalıştırılmış olup 10 dakika 55 sonuca ulaşmıştır. Optimum sonucun tatmin puanı 1262, tatil için ayırdığı bütçeden geriye kalan 85 TL, tatil süresince kat ettiği toplam yol uzunluğu 5288 km, tatilden dönüşü ise ilk günün ardından 27 gün sonra gerçekleşmiştir. Elde edilen sonuçlar Şekil 8.2 ve Şekil 8.3 de görülmektedir. Şekil 8.4 de ise optimum rotanın harita üzerindeki çıktısı yer almaktadır.

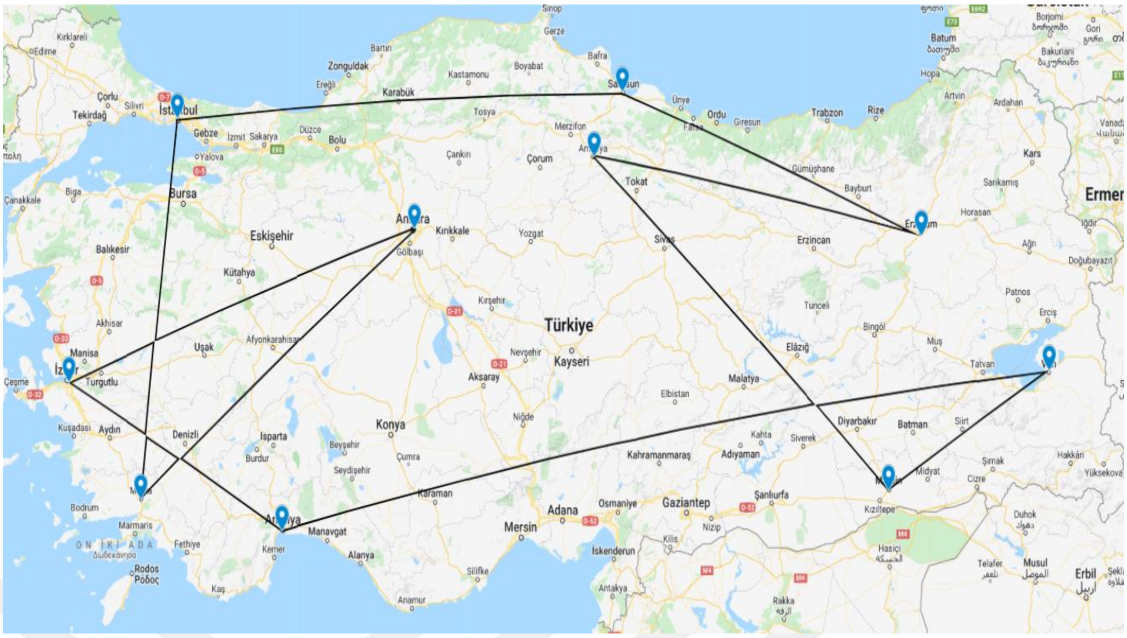


Şekil 8.2 Deney 1 - Ankara, İzmir, Antalya, Van, Mardin, Amasya, Erzurum illerinin detaylı seyahat içeriği

Şekil 8.2 de Ankara, İzmir, Antalya, Van, Mardin, Amasya, Erzurum şehirlerinde yapmış olduğu harcamalar, geçirmiş olduğu zaman, uçak ücreti bilgileri yer almaktadır. Samsun, İstanbul ve Muğla şehirleri için detay bilgileri ise Şekil 8.3 de, aşağıda yer almaktadır.



Şekil 8.3 Deney 1 - Samsun, İstanbul, Muğla illerinin detaylı seyahat içeriği



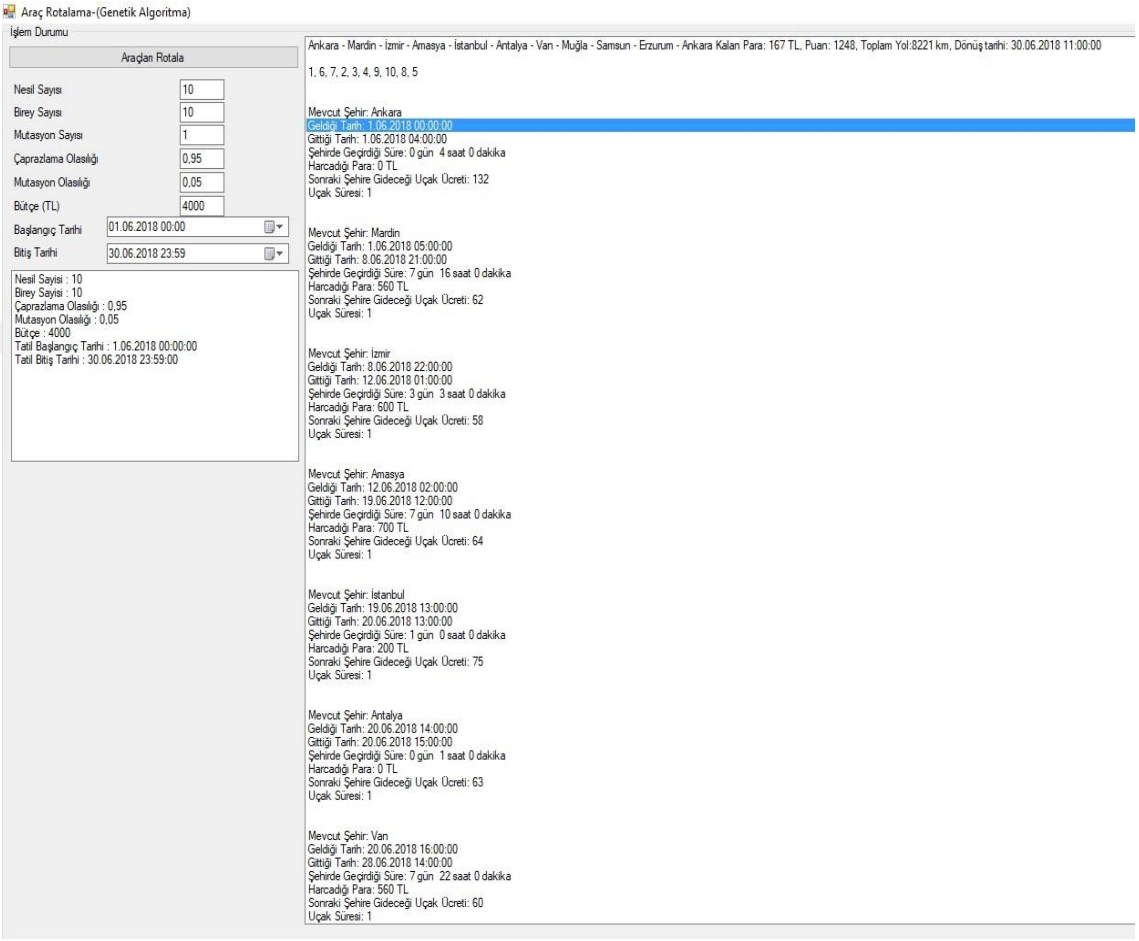
Şekil 8.4 Optimum rotanın haritada gösterimi

Uygulama çalışması bittiğinde sonuç olarak ürettiği rota incelendiğinde tüm parametrelerin ve zorunlu kısıtların çözülerek sonuca ulaştırıldığı görülmüştür. Bazı şehirlerde fazla süre geçirmesinin sebeplerine bakıldığında ise uçak biletlerinin fiyatlamasına göre uçak beklediği gözlemlenmiştir.

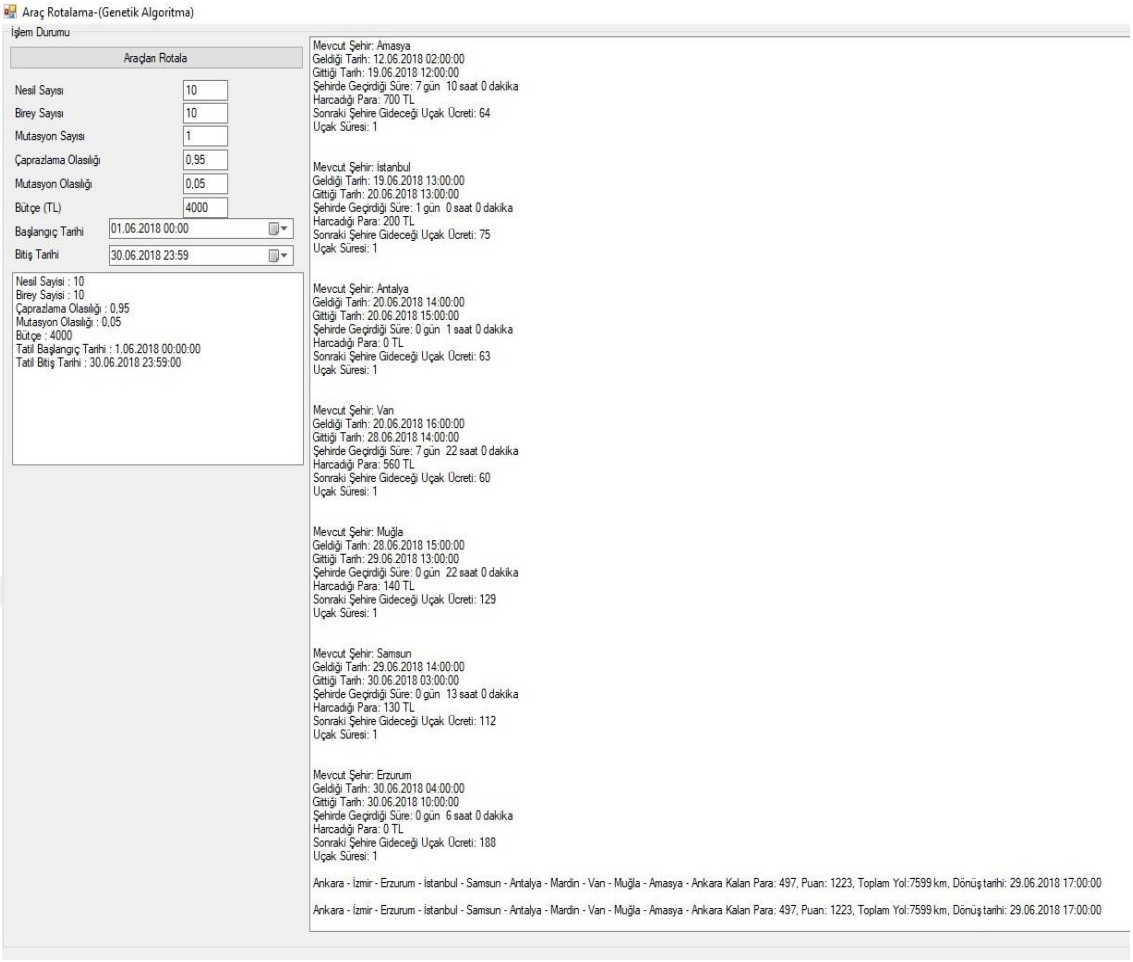
Deney 2

Deney 2 de bütçe 4000 TL, tatil için ayrılan süre 30 gün, algoritmanın kaç nesil tekrar edeceği sayı 10, her bir nesilde olan birey miktarı 10, çaprazlama olasılığı %95, mutasyon olasılığı ise %5 dir. Bu şekilde algoritma çalıştırılmış olup 25 saniyede sonuca ulaşmıştır. Optimum sonucun tatmin puanı 1248, toplam yolu 8221 km, kalan para 167 TL, dönüşü ise 30. günde olmuştur. Sonuçlar Şekil 8.5 ile Şekil 8.6 da görülmektedir.

Aşağıda Şekil 8.5 de Ankara, Mardin, İzmir, Amasya, İstanbul, Antalya, Van şehirlerinde yapmış olduğu harcamalar, geçirmiş olduğu zaman, uçak ücreti bilgileri yer almaktadır. Muğla, Samsun ve Erzurum şehirlerinin detay bilgileri ise Şekil 9.6 da yer almaktadır.



Şekil 8.5 Deney 2 - Ankara, Mardin, İzmir, Amasya, İstanbul, Antalya, Van illerinin detaylı seyahat içeriği

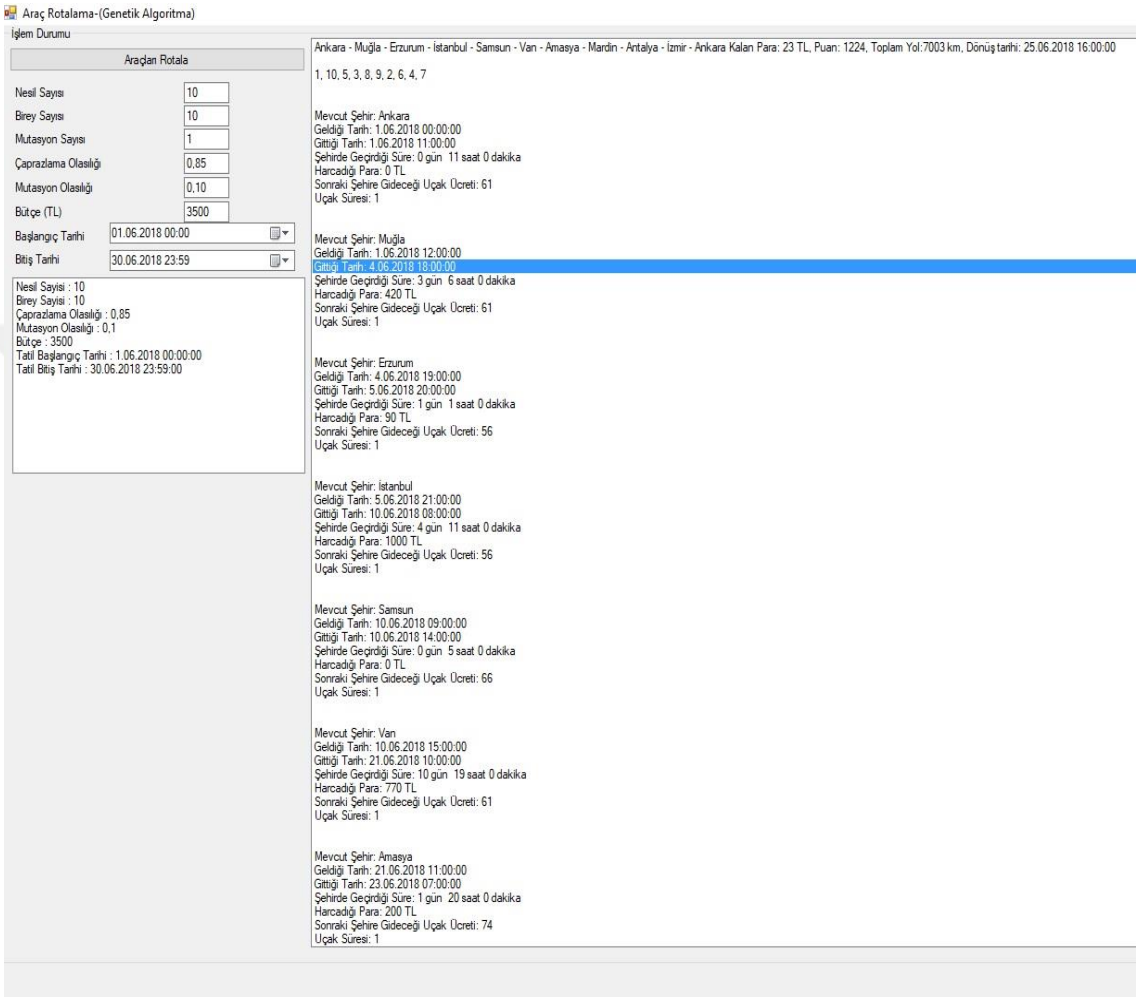


Şekil 8.6 Deney 2 - Muğla, Samsun, Erzurum illerinin detaylı seyahat içeriği

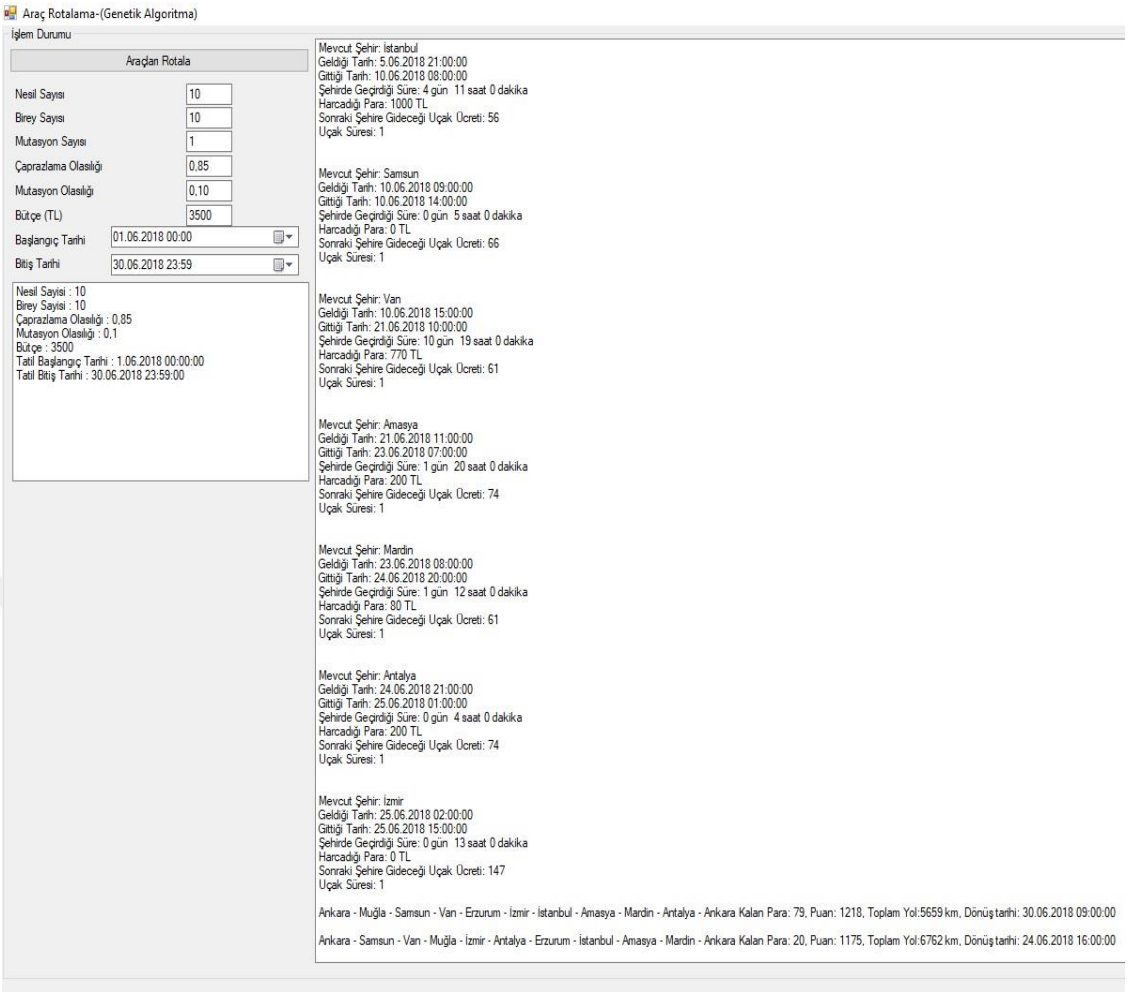
Deney 3

Deney 3 de bütçe 3500 TL, tatil için ayrılan süre 30 gün, algoritmanın kaç nesil tekrar edeceği sayı 10, her bir nesilde olan birey miktarı 10, çaprazlama olasılığı %85, mutasyon olasılığı ise %10 dir. Bu şekilde algoritma çalıştırılmış olup 15 saniyede sonuca ulaşmıştır. Optimum sonucun tatmin puanı 1224, toplam yolu 7003 km, kalan para 23 TL, dönüşü ise 25. günde olmuştur. Elde edilen sonuçlar Şekil 8.7 ile Şekil 8.8 de görülmektedir.

Aşağıda Şekil 8.7 de Ankara, Muğla, Erzurum, İstanbul, Samsun, Van, Amasya şehirlerinde yapmış olduğu harcamalar, geçirmiş olduğu zaman ve hangi sırayla bu şehirleri gezmiş olduğu yer almaktadır. Mardin, Antalya ve İzmir şehirleri için detay bilgileri ise Şekil 8.8 de yer almaktadır.



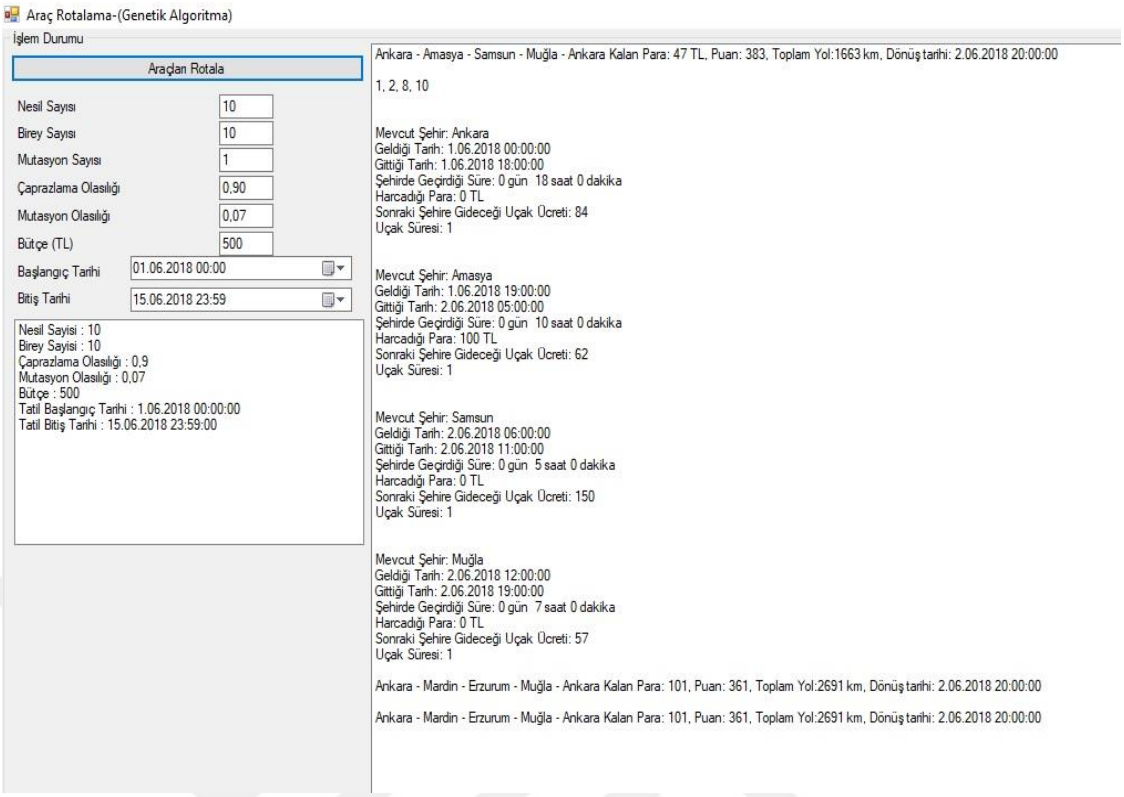
Şekil 8.7 Deney 3 - Ankara, Muğla, Erzurum, İstanbul, Samsun, Van, Amasya illerinin detaylı seyahat içeriği



Şekil 8.8 Deney 3 - Mardin, Antalya, İzmir illerinin detaylı seyahat içeriği

Deney 4

Deney 4 de bütçe 500 TL, tatil için ayrılan süre 15 gün, algoritmanın kaç nesil tekrar edeceği sayı 10, her bir nesilde olan birey miktarı 10, çaprazlama olasılığı %80, mutasyon olasılığı ise %7 dir. Bu şekilde algoritma çalıştırılmış olup 8 saniyede sonuca ulaşmıştır. Optimum sonucun tatmin puanı 1248, toplam yolu 1663 km, kalan para 47 TL, dönüşü ise 2. günde olmuştur. Elde edilen sonuçlar Şekil 8.9 da görülmektedir.

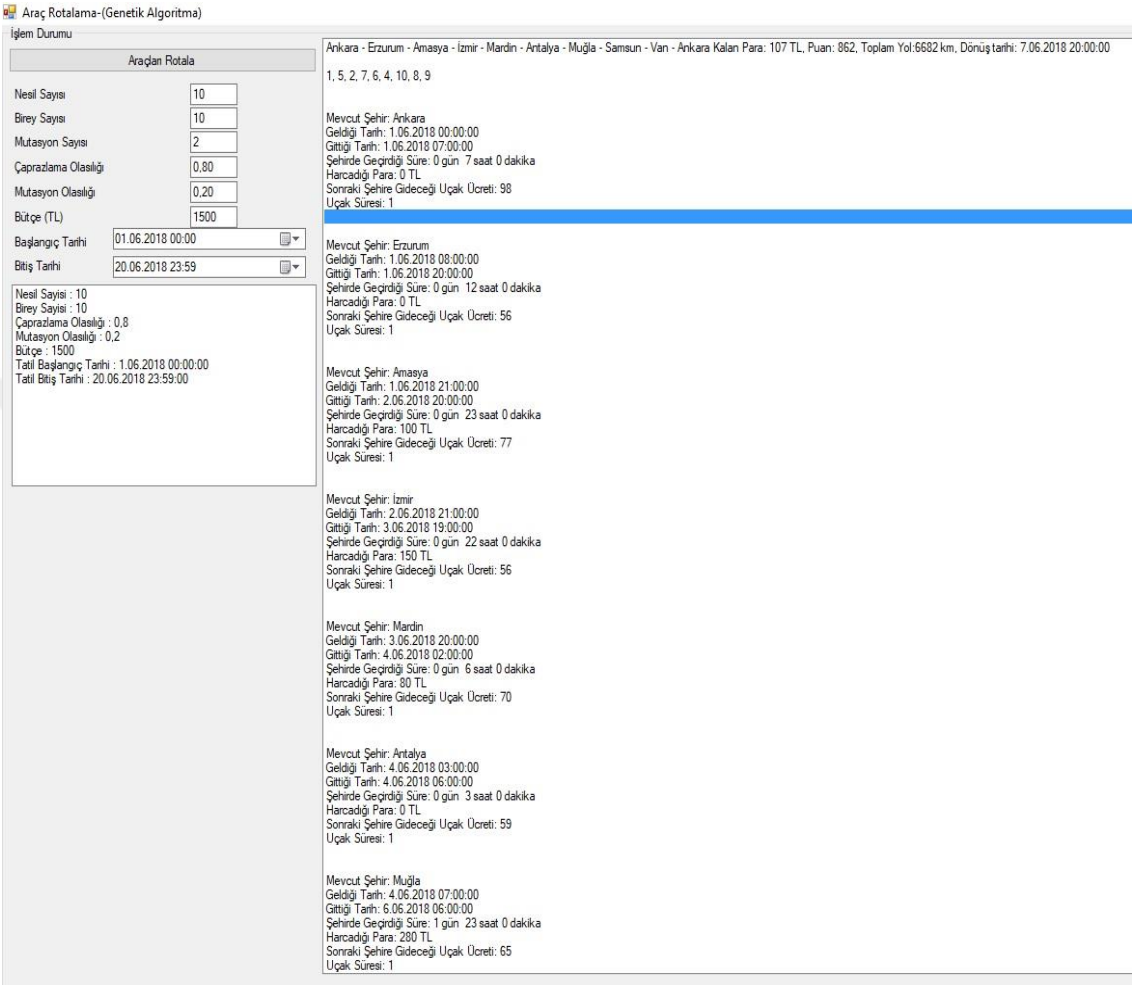


Şekil 8.9 Deney 4 - Ankara, Amasya, Samsun, Muğla illerinin detaylı seyahat içeriği

Deney 5

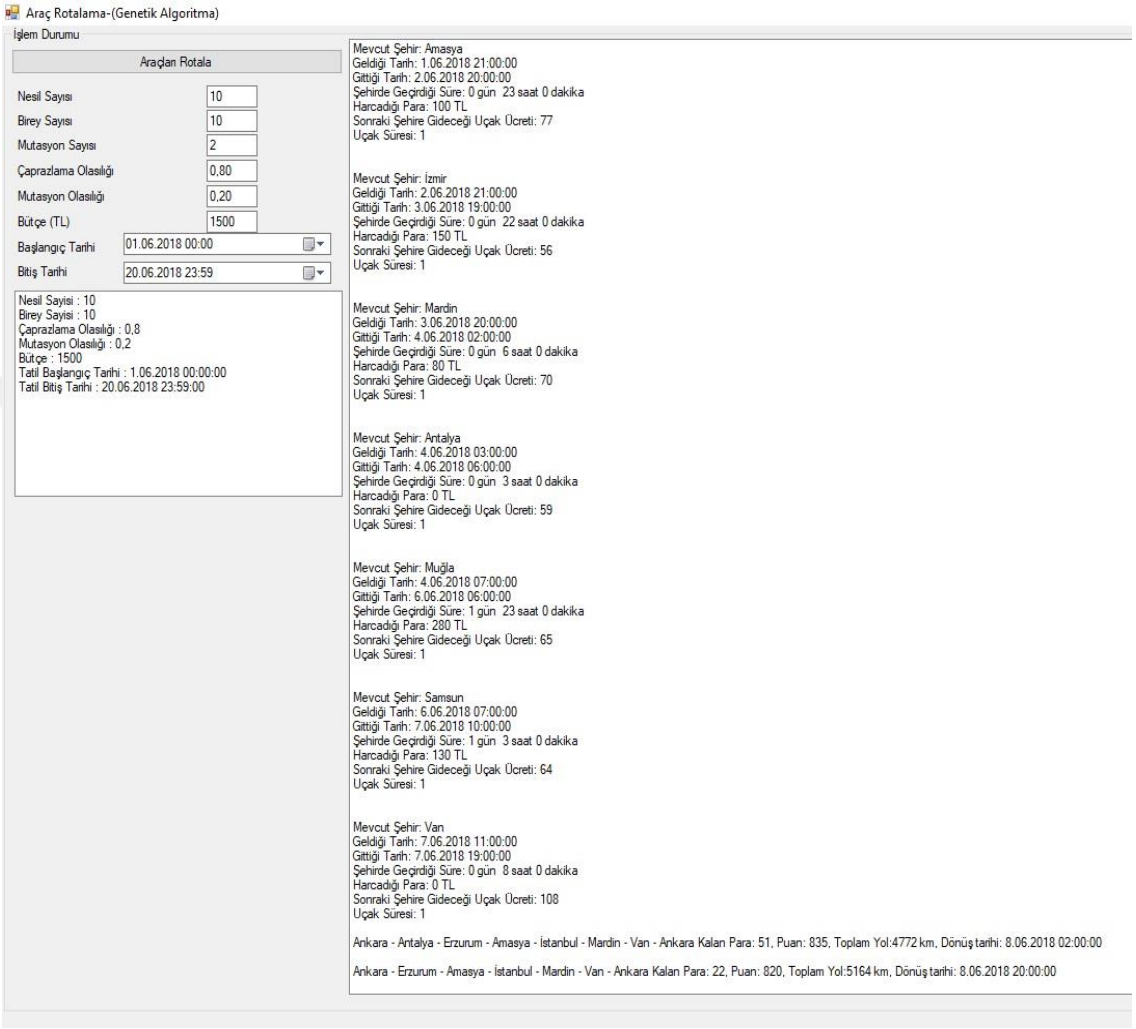
Deney 5 de bütçe 1500 TL, tatil için ayrılan süre 20 gün, algoritmanın kaç nesil tekrar edeceği sayı 10, her bir nesilde olan birey miktarı 10, çaprazlama olasılığı %80, mutasyon olasılığı ise %20 dir. Bu şekilde algoritma çalıştırılmış olup 9 saniyede sonuca ulaşmıştır. Optimum sonucun tatmin puanı 862, toplam yolu 6682 km, kalan para 107 TL, dönüşü ise 7. günde olmuştur. Elde edilen sonuçlar Şekil 8.10 ve Şekil 8.11 de görülmektedir.

Aşağıda Şekil 8.10 da Ankara, Erzurum, Amasya, İzmir, Mardin, Antalya, Muğla şehirlerinde yapmış olduğu harcamalar, geçirmiş olduğu zaman, uçak ücreti bilgileri yer almaktadır.



Şekil 8.10 Deney 5 - Ankara, Erzurum, Amasya, İzmir, Mardin, Antalya, Muğla illerinin detaylı seyahat içeriği

Aşağıda Şekil 8.11 de Şekil 8.10. da yer alan şehirlerin dışında Samsun ve Van şehirlerinde yapmış olduğu harcamalar, geçirmiş olduğu zaman, uçak ücreti bilgileri yer almaktadır.



Şekil 8.11 Deney 5 - Samsun ve Van illerinin detaylı seyahat içeriği

9. SONUÇ VE ÖNERİLER

Bu tezde turist rotalama problemi ele alınmıştır. Yapılan çalışma turistlerin seyahatlerini kolaylaştıracak olup zihin konforunu da arttıracaktır. Bütçe ve zaman kısıtlı turist rotalama problemi, turistin seçmiş olduğu şehirleri, istediği tatil aralığında ve belirtmiş olduğu bütçeye uygun şekilde optimum rota maksimum tatminlik düzeyi olacak şekilde gezmesini sağlayacaktır.

Turist rotalama problemlerinde birden fazla parametrenin problemin çözümüne etki etmesi ile kaynakların kıt olması bu optimizasyon problemlerini çözülmesi çok daha zor problemler yapmaktadır. Geleneksel matematiksel çözümlerle bu problemleri çözmeye kalkışıldığında çok zor ve uzun zaman aldığı görülmektedir. Geleneksel çözümler tam ve kesin çözüme ulaştırmakta ancak almış olduğu vakit sağlamış olduğu kesin sonucun önüne geçmektedir. Bu yüzden tam çözüm veya tam çözüme yakın sonuçlar veren sezgisel algoritmalar bu noktada yardımımıza koşmaktadır. Bu tezde sezgisel bir yöntem olan Genetik Algoritma ile problem çözülmüştür.

Bu tezde programın içindeki veri ile kullanıcıdan istenen verilerin algoritmaya uygulanmasıyla çalışan bir Genetik Algoritma kodlanmıştır. Genetik algoritma kısıt tabanlı yazılmıştır. Verilen parametreler ile kısıtların uygulanmasına zorlanılmaktadır. Bundan dolayı en yüksek puanı bulması sağlanmaktadır.

Yazılan programda farklı verilerle deneyler yapılmıştır. Bu deneylerde GA girdileri değiştirilmiştir. Her değiştirmenin ardından farklı sonuçlar ortaya çıkmıştır. Çıkan sonuçlar değerlendirilip analiz edildiğinde en iyi sonuçların aşağıdaki parametrelerle sağlandığı gözlemlenmiştir.

Çaprazlama: %80

Mutasyon: %10

Çaprazlama ve Mutasyon parametreleri değiştirilmeden birey sayısının değişimine göre sonuçlar incelendiğinde ise 8 ile 12 birey arasında algoritmanın daha iyi sonuçlar ortaya çıkardığı ve daha hızlı çalıştığı gözlemlenmiştir. Birey sayısı 8 in altında kaldığında algoritma hızlı sonuç verse de tatmin puanı istenilen seviyeye çıkamamaktadır. Birey

sayısı 12 nin üstüne çıkarıldığında ise algoritmanın çalışma süresi %50 seviyesinde artış sağlarken tatmin puanı beklenilen artışı gösterememiştir. Algoritmada seçilim için turnuva metodu kullanılmıştır. Her nesilde çalışan daha iyi bireylerin gelecek nesillere aktarılması, sonucu önemli bir şekilde etkilemektedir.

Yapılan uygulama ve testler, optimizasyon problemlerinden biri olan turist rotalama probleminin doğru mutasyon ve çaprazlama olasılıkları ve Genetik Algoritma yardımıyla optimum sonuca ulaşılabileceğini göstermektedir. Genetik Algoritmanın daha iyi sonuçlar ortaya koyabilmesi için parametrelerin doğru değerler verilerek hassasiyeti yüksek bir şekilde ayarlanması gerekmektedir. Algoritmanın hızı, sonraki çalışmalarda pozitif yönde arttırılabilir. Uygulama içerisinde yer alan rastgele oluşturulmuş uçak verilerin servis aracılığıyla gerçek veriler yapılabilir. Bu sayede daha gerçek sonuçlar üretilebilir.

KAYNAKLAR

Akay B., Karaboğa D. (2007). Signal Processing and Communications Applications, Makale. SIU, IEEE 15th

Akyol S., Alataş B., (2012). Güncel Sürü Zekâsı Optimizasyon Algoritmaları, sy. 36

Archetti C., Bianchessi N., Sprenza MG., (2012). A The capacitated team orienteering problem with incomplete service, Makale, Optimization Letters, October 2013, pp 1405-1417

Archetti C., Corberan A., Plana I., Sanchis JM., Speranza MG. (2015). A matheuristic for the Team Orienteering Arc Routing Problem, Makale, European Journal of Operational Research, 245 (2015) 392–401

Behdadnia M. (2016) Zamana Bağlı Oryantiring Problemin Genetik Algoritma Kullanılarak Çözümü, Yüksek Lisans Tezi, Ankara Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı

Ben-Said A., El-Hajj R., Moukrim A., (2016). An adaptive heuristic for the Capacitated Team Orienteering Problem, Makale, Elsevier, Ifac-PapersOnLine 49-12 (2016) 1662–1666

Bianchessı N. (2017). A Branch-And-Cut Algorithm Fort The Team Orienteering Problem, Internatıonal Transactıons In Operatıonal Research, 25, 627–635

Brito J., Exposito A., Moreno A., (2016). Pareto Solving the Team Orienteering Problem with Fuzzy Scores and Constraints, Makale, IEEE International Fuzzy Systems Conference Proceedings, (2016) 1614-1620

Colin R. Reeves, Jonathan E. Rowe, (2003), Genetic Algorithms: Principles and Perspectives A Guide to GA Theory, Makale, sy. 25

Connoly, D., (1990). An Improved Annealing Scheme for the QAP, Makale, European Journal of Operational Research, 46 93–100

Ç.Ü. Sosyal Bilimler Enstitüsü Dergisi, Cilt 19, Sayı 3, 2010, Sayfa 423-438

Dang D., Guibadj RN., Moukrim A., (2013). An effective PSO-inspired algorithm for the team orienteering problem, Makale, European Journal of Operational Research, 229 (2013) 332–344

Dorigo M., Maniezzo V. Colorni A., (1991). The Ant System: An Autocatalytic Optimisic Process. Tech. Rep.

El-Hajj R., Dang D., Moukrim A. (2016). Solving the team orienteering problem with cutting planes, Makale, Computers & Operations Research, 74 (2016) 21-30

Engin O., Fığlalı A., (2002), Akış Tipi Çizelgeleme Problemlerinin Genetik Algoritma Yardımı İle Çözümünde Uygun Çaprazlama Operatörünün Kullanılması, Doğuş Üniversitesi Dergisi, Makale, 2002/6, 27-35, Sayfa 30

Gavalas D., Konstantopuolos C., Mastakas K., Pantziou G., Vathis N., (2015). Heuristics for the time dependent team orienteering problem:Application to tourist route planning, Makale, Computers & Operations Research, 62 (2015) 36–50

Ka L., Zhai L., Li J., Chan F. (2015). Pareto mimic algorithm: An approach to the team orienteering problem, Makale, Omega, 61 (2016) 155-166

Karaboğa D., (2011), Yapay Zekâ Optimizasyon Algoritmaları, Nobel Yayın Dağıtım, Makale, Genişletilmiş 2. Basım, sy. 50, 2011

Keshtkaran M., Ziarati K., Bettinelli A., Vigo D. (2015). Pareto mimic algorithm: An approach to the team orienteering problem, Makale, International Journal of Production Research, (2016) Vol. 54, No. 2, 591–601

Khoo, K. G., Suganthan, P. N., (2002), Evaluation of genetic operators and solution representations for shape recognition by genetic algorithms, Pattern Recognition Letters, Makale, 23 s: 1589-1597

Kirkpatrick, S., Gelatt Jr, C.D., Vecchi Jr, M.P., (1983). Optimization By Simulated Annealing, Makale, Science, 220, 671-680

Ladesma J. , Gonzales J. (2017). A Solving the Team Orienteering Arc Routing Problem with a column generation approach, Makale, European Journal of Operational Research, 262 (2017) 14-27

Özdemir M., (2013). Zaman Kısıtı Altında Takım Oryantiring Problemlerinin Yapay Arı Kolonisi Yaklaşımı İle Çözülmesi, Yüksek Lisans Tezi, İstanbul Üniversitesi Sosyal Bilimler Enstitüsü İşletme Anabilim Dalı Sayısal Yöntemler Bilim Dalı

Salazar-Aguilari MA., Langevin A., Laporte G., (2013). A The multi-district team orienteering problem, Makale, Computers & Operations Research, 41 (2014) 76–82

Tunçhan C., (2008). Modern Sezgisel Teknikler ve Uygulamaları, Papatya Yayınevi, sy. 14

Van Laarhoven, P.J.M,. Aarts, E.H.L., Lenstra, J.K. (1992), Job Shop Scheduling by Simulated Annealing, Operations Research, Makale, 40 113–126.

Wang X., Golden B., Gulczynski D., (2014). A worst-case analysis for the split delivery capacitated team orienteering problem with minimum delivery amounts, Makale, Optimization Letters, December 2014 pp 2349-2356

Zettam M., Elbenani B., (2016, 23-25 Mayıs). A novel Randomized Heuristic for the Team Orienteering Problem, 3rd IEEE International Conference on Logistics Operations Management (GOL) konferansında sunuldu, Fes, Morocco

EKLER FONKSİYONEL İÇERİKLER

EK 1 Birey Sınıfı

```
public Birey(List<int> Kromozom, double TatminDuzeyi)
{
        this.Kromozom = Kromozom;
        this.TatminDuzeyi = TatminDuzeyi;
        this.ToplamYol = 0;
        this.KalanPara = 0;
        this.sonsehir = 0;
        guzergahi = new List<GuzergahSehir>();
}
public List<GuzergahSehir> guzergahi { get; set; }
public List<int> Kromozom { get; set; }
public double TatminDuzeyi { get; set; }
public int ToplamYol { get; set; }
public int KalanPara { get; set; }
public DateTime SonTarih { get; set;}
public int sonsehir { get; set;}
```

EK 2 Güzergah Sınıfı

```
public Yer mevcutsehir { get; set; }
public DateTime geldigitarih { get; set; }
public DateTime gittigitarih { get; set; }
public int harcadigipara { get; set; }
public int sonrakisehiregidecegiucakucreti { get; set; }
public double ucaksuresi { get; set; }
```

EK 3 Toplum Oluşturma Fonksiyonu

```
public List<Birey> ToplumOlustur()
{
        List<Birey> toplum = new List<Birey>();

        while (toplum.Count < BireySayisi)
        {
              List<int> yeridler = YerListesi.Where(x => x.Id != 1).Select(x =>
x.Id).ToList();
              Birey islemyapilacakBirey = new Birey(new List<int>(), 0);
              islemyapilacakBirey.Kromozom.Add(1);
                 while (islemyapilacakBirey.Kromozom.Count() < YerListesi.Count())
                 {
                     int rastint = rastgele.Next(yeridler.Count());
                     int yerid = yeridler[rastint];

                     if (!islemyapilacakBirey.Kromozom.Contains(yerid))
                     {
                         islemyapilacakBirey.Kromozom.Add(yerid);
                         yeridler.Remove(yerid);
                     }
                 }
                 toplum.Add(islemyapilacakBirey);
        }

        return toplum;

}
```

EK 4 Günlük Harcama Fonksiyonu

```
        public bool GunlukHarcama(Yer sehir, Ucak ucak, int kalanpara, DateTime
bulunulantarih)
        {
            tempkalanpara = kalanpara;
            tempkalanpara -= sehir.GunlukHarcama * GunHesabı(ucak.Saati,
bulunulantarih);
            tempkalanpara -= ucak.Ucret;
            if (tempkalanpara < 0)
            {
                return false;
            }
            return true;
        }
```

EK 5 Çaprazlama Fonksiyonu

```
public List<Birey> Caprazlama(Birey birey1, Birey birey2)
{
    bireyler = new List<Birey>();
    if (rastgele.NextDouble() < CaprazlamaOlasiligi)
    {
       teknokta = rastgele.Next(YerListesi.Count());
       yenibirey1 = new Birey(new List<int>(), 0);
       yenibirey2 = new Birey(new List<int>(), 0);
       for (int i = 0; i <= teknokta; i++)
           {
              yenibirey1.Kromozom.Add(birey1.Kromozom[i]);
           }
       for (int j = 0; j < birey2.Kromozom.Count(); j++)
           {
           if (!(yenibirey1.Kromozom.Contains(birey2.Kromozom[j])))
              {
              yenibirey1.Kromozom.Add(birey2.Kromozom[j]);
              }
           }
           for (int i = 0; i <= teknokta; i++)
               {
                 yenibirey2.Kromozom.Add(birey2.Kromozom[i]);
               }
           for (int j = 0; j < birey1.Kromozom.Count(); j++)
               {
               if (!(yenibirey2.Kromozom.Contains(birey1.Kromozom[j])))
                  {
                    yenibirey2.Kromozom.Add(birey1.Kromozom[j]);
                  }
               }
               bireyler.Add(yenibirey1);
```

```
            bireyler.Add(yenibirey2);
        }
        else
        {
            bireyler.Add(birey1);
            bireyler.Add(birey2);
        }
        return bireyler;
    }
```

EK 6 Mutasyon Sağlama Koşulu

```
if (rastgele.NextDouble() <= MutasyonOlasiligi)
{
    mutasyonaUgrayanBireyler = new List<Birey>();
    m = Mutasyon(item);
    mutasyonaUgrayanBireyler.Add(m);
    yeniToplum.Add(m);
}
else
{
    yeniToplum.Add(item);
 }
```

EK 7 Mutasyon Fonksiyonu

```
        public Birey Mutasyon(Birey mutasyonaugrayacakbirey)
         {
             for (int i = 0; i < MutasyonSayisi; i++)
             {
                 int random1 = rastgele.Next(YerListesi.Count()-1) + 1;
                 int mutasyonİcinSecilenKromozom =
mutasyonaugrayacakbirey.Kromozom[random1];
                 int random2 = rastgele.Next(YerListesi.Count()-1) + 1;
                 int degistirilecekdigerkromozom =
mutasyonaugrayacakbirey.Kromozom[random2];
                 mutasyonaugrayacakbirey.Kromozom[random1] =
degistirilecekdigerkromozom;
                 mutasyonaugrayacakbirey.Kromozom[random2] =
mutasyonİcinSecilenKromozom;
                 mutasyonaugrayacakbirey.TatminDuzeyi = 0;
             }
             return mutasyonaugrayacakbirey;

        }
```

EK 8 Tatminlik Hesaplama Fonksiyonu

```
public void TatminlikHesapla(Birey birey)
 {
     Yer bulunulansehir = null;
     Ucak ucagımız = null;
     DateTime bulunulantarih = new DateTime();
     bulunulantarih = baslangicgunu;
     birey.TatminDuzeyi = 0;
     birey.KalanPara = 0;
     birey.ToplamYol = 0;
     birey.SonTarih = new DateTime();
     birey.sonsehir = 0;
     birey.guzergahi = new List<GuzergahSehir>();

     int bireybutce = butce;

     Birey birey1 = null;
     Yer bulunulansehir1 = null;
     DateTime bulunulantarih1 = new DateTime();
     GuzergahSehir guzergahsehir1 = new GuzergahSehir();
     int farkbutcehesapla1 = 0;
     Yer sonrakisehir1 = null;
     GuzergahSehir guzergahsehir = new GuzergahSehir();
     int farkbutcehesapla = 0;
     Yer sonrakisehir = null;
     int bireybutce1 = 0;
     for (int i = 0; i < birey.Kromozom.Count(); i++)
     {
         bulunulansehir = YerListesi.Where(x => x.Id ==
birey.Kromozom[i]).FirstOrDefault();

         birey1 = birey;
```

```
                bulunulansehir1 = bulunulansehir;
                bulunulantarih1 = bulunulantarih;
                bireybutce1 = bireybutce;
                if (!BaslangicNoktasinaDonuyormu(birey1, bulunulansehir1,
bulunulantarih1, bireybutce1))
                {
                    ucagımız = UcakListesi.Where(x => x.NeredenYerId ==
birey.Kromozom[i] && x.NereyeYerId == birey.Kromozom[0] && x.Saati >
bulunulantarih && x.Ucret <= bireybutce && true == GunlukHarcama(bulunulansehir,
x, bireybutce, bulunulantarih)).OrderBy(x => x.Ucret).ThenBy(x => x.Saati
).FirstOrDefault();
                    if (ucagımız == null)
                    {
                        ucagımız = UcakListesi.Where(x => x.NeredenYerId ==
birey.Kromozom[i] && x.NereyeYerId == birey.Kromozom[0] && x.Saati >
bulunulantarih).OrderBy(x => x.Saati).ThenBy(x => x.Ucret).FirstOrDefault();
                    }

                    guzergahsehir1 = new GuzergahSehir();
                    guzergahsehir1.mevcutsehir = bulunulansehir;
                    guzergahsehir1.geldigitarih = bulunulantarih;
                    farkbutcehesapla1 = bireybutce;

                    birey.TatminDuzeyi += bulunulansehir.TatminBaslangic +
(bulunulansehir.TatminGunluk * GunHesabı(ucagımız.Saati,bulunulantarih));
                    bireybutce -= bulunulansehir.GunlukHarcama *
GunHesabı(ucagımız.Saati, bulunulantarih);
                    bulunulantarih = ucagımız.Saati;
                    guzergahsehir1.gittigitarih = bulunulantarih;
                    guzergahsehir1.harcadigipara = farkbutcehesapla1 -
bireybutce;

                    bulunulantarih = bulunulantarih.AddHours(ucagımız.Suresi);
                    guzergahsehir1.sonrakisehiregidecegiucakucreti =
ucagımız.Ucret;
```

```
                    bireybutce -= ucagımız.Ucret;
                    birey.KalanPara = bireybutce;
                    birey.SonTarih = bulunulantarih;
                    birey.sonsehir = bulunulansehir.Id;
                    sonrakisehir1 = YerListesi.Where(x => x.Id ==
birey.Kromozom[0]).FirstOrDefault();
                    birey.ToplamYol +=
Convert.ToInt32(Geography.CalculateDistance(bulunulansehir.Enlem,
bulunulansehir.Boylam, sonrakisehir1.Enlem, sonrakisehir1.Boylam));
                    guzergahsehir1.ucaksuresi = ucagımız.Suresi;
                    birey.guzergahi.Add(guzergahsehir1);
                    return;
                }

                if (i == birey.Kromozom.Count() - 1)
                {
                    ucagımız = UcakListesi.Where(x => x.NeredenYerId ==
birey.Kromozom[i] && x.NereyeYerId == birey.Kromozom[0] && x.Saati >
bulunulantarih && x.Ucret <= bireybutce && true == GunlukHarcama(bulunulansehir,
x, bireybutce, bulunulantarih)).OrderBy(x => x.Ucret).ThenBy(x =>
x.Saati).FirstOrDefault();
                }
                else if (i == 0)
                {
                    ucagımız = UcakListesi.Where(x => x.NeredenYerId ==
birey.Kromozom[i] && x.NereyeYerId == birey.Kromozom[i + 1] && x.Ucret <=
bireybutce).OrderBy(x => x.Saati).ThenBy(x => x.Ucret).FirstOrDefault();
                }
                else
                {
                    ucagımız = UcakListesi.Where(x => x.NeredenYerId ==
birey.Kromozom[i] && x.NereyeYerId == birey.Kromozom[i + 1] && x.Saati >
bulunulantarih && x.Ucret <= bireybutce && true == GunlukHarcama(bulunulansehir,
x, bireybutce, bulunulantarih)).OrderBy(x => x.Ucret).ThenBy(x =>
x.Saati).FirstOrDefault();
```

```
                }

                guzergahsehir = new GuzergahSehir();
                guzergahsehir.mevcutsehir = bulunulansehir;
                guzergahsehir.geldigitarih = bulunulantarih;

                farkbutcehesapla = bireybutce;

                birey.TatminDuzeyi += bulunulansehir.TatminBaslangic +
(bulunulansehir.TatminGunluk * GunHesabı(ucagımız.Saati, bulunulantarih));
                bireybutce -= bulunulansehir.GunlukHarcama *
GunHesabı(ucagımız.Saati,bulunulantarih);
                bulunulantarih = ucagımız.Saati;
                birey.SonTarih = bulunulantarih;
                guzergahsehir.gittigitarih = bulunulantarih;

                bulunulantarih = bulunulantarih.AddHours(ucagımız.Suresi);
                guzergahsehir.harcadigipara = farkbutcehesapla - bireybutce;
                farkbutcehesapla = bireybutce;
                bireybutce -= ucagımız.Ucret;
                birey.KalanPara = bireybutce;
                birey.SonTarih = bulunulantarih;
                sonrakisehir = null;
                guzergahsehir.sonrakisehiregidecegiucakucreti = ucagımız.Ucret;
                guzergahsehir.ucaksuresi = ucagımız.Suresi;
                birey.guzergahi.Add(guzergahsehir);
                if (i == birey.Kromozom.Count() - 1)
                {
                    sonrakisehir = YerListesi.Where(x => x.Id ==
birey.Kromozom[0]).FirstOrDefault();
                }
                else
                {
```

```
                    sonrakisehir = YerListesi.Where(x => x.Id == birey.Kromozom[i
+ 1]).FirstOrDefault();
                }
                birey.ToplamYol +=
Convert.ToInt32(Geography.CalculateDistance(bulunulansehir.Enlem,
bulunulansehir.Boylam, sonrakisehir.Enlem, sonrakisehir.Boylam));
            }

        }
```

EK 9 Başlangıç Noktasına Dönme Kontrolü Fonksiyonu

```
        if (!BaslangicNoktasinaDonuyormu(birey1, bulunulansehir1, bulunulantarih1,
bireybutce1))
        {
            ...
            return;
        }
```

Yukarıdaki kodda görüldüğü gibi başlangıç noktasına dönüyorsa if in içerisine girmemektedir. Aşağıda başlangıç noktasına dönüyor mu fonksiyonunun kodu gösterilmektedir.

```
        public bool BaslangicNoktasinaDonuyormu(Birey birey, Yer bulunulansehir,
DateTime bulunulantarih, int kalanpara)
        {
            if (kalanpara < 0)
            {
                return false;
            }
            int bulunulansehirindex = Array.IndexOf(birey.Kromozom.ToArray(),
bulunulansehir.Id);
            if (bulunulansehirindex == (birey.Kromozom.Count() - 1))
            {
                return true;
            }
            Ucak ucagımız = new Ucak();
            if (bulunulansehirindex == 0)
            {
                ucagımız = UcakListesi.Where(x => x.NeredenYerId ==
birey.Kromozom[bulunulansehirindex] && x.NereyeYerId ==
birey.Kromozom[bulunulansehirindex + 1] && x.Ucret <= kalanpara).OrderBy(x =>
x.Saati).ThenBy(x => x.Ucret).FirstOrDefault();
            }
```

```
            else
            {
                ucagımız = UcakListesi.Where(x => x.NeredenYerId ==
birey.Kromozom[bulunulansehirindex] && x.NereyeYerId ==
birey.Kromozom[bulunulansehirindex + 1] && x.Saati > bulunulantarih && x.Ucret <=
kalanpara && true == GunlukHarcama(bulunulansehir, x, kalanpara,
bulunulantarih)).OrderBy(x => x.Ucret).ThenBy(x => x.Saati).FirstOrDefault();
            }
            if (ucagımız == null)
            {
                return false;
            }
            kalanpara -= bulunulansehir.GunlukHarcama * GunHesabı(ucagımız.Saati,
bulunulantarih);
            bulunulantarih = ucagımız.Saati;
            bulunulantarih = bulunulantarih.AddHours(ucagımız.Suresi);
            kalanpara -= ucagımız.Ucret;
            if (kalanpara < 0)
            {
                return false;
            }
            bulunulansehir = YerListesi.Where(x => x.Id ==
birey.Kromozom[bulunulansehirindex + 1]).FirstOrDefault();
            ucagımız = UcakListesi.Where(x => x.NeredenYerId == bulunulansehir.Id
&& x.NereyeYerId == birey.Kromozom[0] && x.Saati > bulunulantarih && x.Ucret <=
kalanpara && true == GunlukHarcama(bulunulansehir, x, kalanpara,
bulunulantarih)).OrderBy(x => x.Ucret).ThenBy(x => x.Saati).FirstOrDefault();
            if (ucagımız == null)
            {
                return false;
            }
            kalanpara -= bulunulansehir.GunlukHarcama * GunHesabı(ucagımız.Saati,
bulunulantarih);
            bulunulantarih = ucagımız.Saati;
            bulunulantarih = bulunulantarih.AddHours(ucagımız.Suresi);
```

```
        kalanpara -= ucagımız.Ucret;
        if (kalanpara < 0)
        {
            return false;
        }
        return true;
    }
```

EK 10 Mesafe Hesaplama Fonksiyonu

```
        public static int MesafeHesapla(double lat_1, double long_1, double lat_2,
double long_2)
         {
             Func<double, double> degr_2_Rad = (x) => (x * (Math.PI / 180));
             Func<double, double> rad_2_Deg = (x) => (x / Math.PI * 180.0);
             var d1 = long_1 – long_2;
             var d2 = Math.Sin(degr_2_Rad(lat_1)) * Math.Sin(degr_2_Rad(lat_2)) +
Math.Cos(degr_2_Rad(lat_1)) * Math.Cos(degr_2_Rad(lat_2)) *
Math.Cos(degr_2_Rad(d1));
             d2 = Math.Acos(d2);
             d2 = rad_2_Deg(d2);
             d2 = d2 * 60 * 1.1515;
             d2 = d2 * 1.609344;
             return Convert.ToInt32(d2);
        }
```