T.C

YAŞAR ÜNİVERSİTESİ

SOSYAL BİLİMLER ENSTİTÜSÜ

İŞLETME ANABİLİM DALI

YÜKSEK LİSANS TEZİ

YEREL TARAMA İLE BİRLEŞTİRİLMİŞ KESİKLİ FARKSAL EVRİM ALGORİTMASI KULLANARAK GENELLEŞTİRİLMİŞ GEZGİN SATICI PROBLEMİNİN ÇÖZÜMÜ

İKBAL ECE ULUŞANS

Danışmanlar

Prof. Dr. Edip Teker

Doç Dr. A. Fatih Taşgetiren

İzmir-2010

T.C

YAŞAR ÜNİVERSİTESİ

SOSYAL BİLİMLER ENSTİTÜSÜ

İŞLETME ANABİLİM DALI

YÜKSEK LİSANS TEZİ

YEREL TARAMA İLE BİRLEŞTİRİLMİŞ KESİKLİ FARKSAL EVRİM ALGORİTMASI KULLANARAK GENELLEŞTİRİLMİŞ GEZGİN SATICI PROBLEMİNİN ÇÖZÜMÜ

İKBAL ECE ULUŞANS

Danışmanlar

Prof. Dr. Edip Teker

Doç Dr. A. Fatih Taşgetiren

İzmir-2010

YEMİN METNİ

Yüksek Lisans Tezi olarak sunduğum “Yerel tarama ile birleştirilmiş kesikli farksal evrim algoritması kullanarak, genelleştirilmiş gezgin satıcı probleminin çözümü” adlı çalışmanın, tarafımdan, bilimsel ahlak ve geleneklere aykırı düşecek bir yardıma başvurmaksızın yazıldığını ve yararlandığım eserlerin bibliyografyada gösterilenlerden oluştuğunu, bunlara atıf yapılarak yararlanılmış olduğunu belirtir ve bunu onurumla doğrularım.

Tarih

… / .. / 2010

Adı SOYADI

İkbal Ece Uluşans

İmza

TUTANAK

Yaşar Üniversitesi Sosyal Bilimler Enstitüsü' nün ...../...../..... tarih ve ..... sayılı toplantısında oluşturulan jüri, Lisansüstü Öğretim Yönetmeliği'nin ..... maddesine göre ..... Anabilim Dalı Yüksek Lisans /Doktora öğrencisi **.....** ' nin .....konulu tezi/projesi incelenmiş ve aday ...../...../..... tarihinde, saat ..... ' da jüri önünde tez savunmasına alınmıştır.

Adayın kişisel çalışmaya dayanan tezini/projesini savunmasından sonra ..... dakikalık süre içinde gerek tez konusu, gerekse tezin dayanağı olan anabilim dallarından jüri üyelerine sorulan sorulara verdiği cevaplar değerlendirilerek tezin/projenin ..... olduğuna oy ..... ile karar verildi.

BAŞKAN

ÜYE ÜYE

TEŞEKKÜR

Yüksek Lisans tezimin hazırlanmasında bana her türlü desteği sağlayan danışmanlarım Prof. Dr. M. Edip Teker ve Doç. Dr. M. Fatih Taşgetiren'e, yüksek lisans eğitimim boyunca derslerime giren hocalarıma, çalışmam süresince bana gösterdikleri anlayış ve destekten dolayı aileme, nişanlıma ve arkadaşlarıma teşekkür ederim.

İkbal Ece Uluşans

YEREL TARAMA İLE BİRLEŞTİRİLMİŞ KESİKLİ FARKSAL EVRİM ALGORİTMASI KULLANARAK GENELLEŞTİRİLMİŞ GEZGİN SATICI PROBLEMİNİN ÇÖZÜMÜ

İkbal Ece Uluşans

ÖZET

Bu tezde genelleştirilmiş gezgin satıcı probleminin çözümü için bölgesel tarama ile birleştirilmiş kesikli farksal evrim algoritması sunulmuştur. Genelleştirilmiş gezgin satıcı probleminde bir satıcının iş yaptığı şehirlerin kümesi kümelere ayrılır ve satıcının her kümeden yalnız bir şehre uğrayarak en kısa yoldan turu tamamlaması beklenir.

Bu algoritmayı test etmek için, GTSPLIB kütüphanesinde bulunan, şehir ve küme sayıları 48 (10) ile 1084 (217) arasında değişen 54 test problemi kullanılmıştır. Sonuçların deneysel analizlerinin yapılması ile, algoritma en iyi sonuçları veren Bontoux, Artigues ve Feillet'in (2009) Memetik Algoritması, Taşgetiren, Suganthan ve Pan'ın (2009) eDDE algoritması, Synder ve Daskin'in (2006) RKGA ve Silberholz ve Golden'ın (1997), mrOXGA ile kıyaslanmıştır.

Sonuç olarak, eniyi değerleri bilinen 41 test probleminin sonuçları kıyaslandığında, KFE Algoritması mrOXGA, MA ve eDDE algoritmasına eşdeğer olduğu ancak RKGA'dan daha iyi sonuçlar ürettiği görülmüştür.

Anahtar Kelimeler: Genelleştirilmiş Gezgin Satıcı Problemi, Farksal Evrim Algoritması

SOLVING GENERALIZED TRAVELING SALESMAN PROBLEM BY USING DISCRETE DIFFERANTİAL EVALUATION ALGORITHM HYBRIDIZED WITH LOCAL SEARCH

İkbal Ece Uluşans

ABSTRACT

This thesis present a discrete differential evaluation algorithm hybridized with local search heuristic (KFE), for the generalized traveling salesman problem. In the GTSP, the set of cities is divided into clusters so that the aim is to find minimum tour length if a salesman has to visit one city from every cluster.

In order to test this algorithm, 54 benchmark instances ranging from 48 (10) to 1084 (217) nodes/ clusters from the GTSPLIB are used. Through the experimental analysis of the results, the performance of the algorithm is compared against the best performing algorithms such as Memetic Algorithm of Bontoux, Artigues and Feillet, eDDE algorithm of Taşgetiren, Suganthan and Pan, RKGA of Synder and Daskin, and mrOXGA of Silberholz and Golden.

Key Words: Generalized Travelling Salesman Problem, Differential Evaluation Algorithm.

İÇİNDEKİLER

ŞEKİLLER LİSTESİ

TABLOLAR LİSTESİ

TERİMLER ve KISALTMALAR

Araç Rotalama: Vehicle Routing

Dal Ve Sınır Yöntemi: Branch And Bound

Dal Ve Kesme: Branch And Cut

Deneme Bireyi: Trial Solution

Deterministik Olmayan Polinomsal (NP): Non- Deterministic Polynomial (NP)

Dinamik Programlama: Dynamic Programming

Elitist Strateji: Seçkinlik İlkesi

En İyilemek: Optimization

En Uzağa Ekleme: Farthest İnsertion

En Ucuz Ekleme: Cheapest Insertion

Gezgin Satıcı Problemi (GSP): Traveling Salesman Problem (TSP)

En Yakına Ekleme: Nearest İnsertion

En Yakın Komşuluk: Nearest Neighborhood

Farksal Evrim Algoritması (FEA): Differantial Evaluation Algorithm (DEA)

Gen: Gene

Genelleştirilmiş Gezgin Satıcı Problemi (GSP): Generalized Traveling Salesman Problem (GTSP)

Genetik Algoritma (GA): Genetic Algorithm (GA)

Hedef Bireyi: Target Solution

Iterated Greedy Algoritmasını : Iterated Greedy Algorithm

Kareli Atama Problemi: Quadratic Assignment

Kesirli Faktöriyel Deney (KFD): Fractional Factorial Experiment (FFE)

Kesikli Farksal Evrim Algoritması (KFEA): Discrete Differantial Evaluation Algorithm (DDEA)

Kesikli Parça Sürü En İyilemesini: Discrete Particle Swarm Optimization

Kısmen Uyumlu Çaprazlama (PMX): Partially Ordered Crossover (PMX)

Kromozom: Chromosome

Memetik Algoritma (MA): Memetic Algorithm (MA)

NEH Sezgiseli: NEH Heuristic

NP-Zor: NP-Hard

Polinomsal (P): Polynomial (P)

Rasgele Anahtar Genetik Algoritmasını (RAGA): Random Key Genetic Algorithm (RKGA)

Sıralı Çaprazlama (OX): Ordered Crossover (OX)

Tam Sayı Programlama: Integer Programming

Tepki Yüzey Metodu (TYM): Response Surface Methodology (RSM)

Toplum Büyüklüğü: Population Size

Topluluk KFE Algoritması: Ensemble Of Discrete Differantial Evolution Algorithm (Edde)

BİRİNCİ BÖLÜM

1. GİRİŞ

1.1. Tezin önemi

Bir satıcının her şehre bir kez uğrayarak n adet şehri en kısa yoldan gezerek tekrar başlangıç şehrine dönmesi problemine gezgin satıcı problemi (GSP) adı verilir. Bu problemin bir çeşidi ise genelleştirilmiş gezgin satıcı problemidir. Problemin bu halinde n adet şehir m adet kümeye ayrılır ve satıcının bu m kümeden her birinden yalnız bir şehre uğrayarak yine başlangıç şehrine dönmesi beklenir. Bu problem NP-zor bir problem türüdür. Dinamik programlama yöntemi ile bu problem türlerinin kesin çözümlerini bulmak mümkündür ancak bu yöntem ile çözümler çok uzun sürmekte ve buda çözümün çok maliyetli olmasına sebep olmaktadır. Böyle durumlarda çözüm uzayında bulunan elemanların hepsini inceleyen matematiksel yöntemler yerine sezgisel yöntemlerden yararlanılmaktadır. Sezgisel yöntemlerin dezavantajlı yönü ise eniyi çözüme ulaşmasının garanti olmaması ve birçok parametrenin uygun şekilde ayarlanmasını gerektirmesidir.

Problemin çözümünde kullanılan sezgisel yöntemlerden bir tanesi Farksal Evrim Algoritmasıdır (FEA). Bu algoritma işleyişi açısından Genetik Algoritmaya (GA) dayanmaktadır. FEA basit ama güçlü, mutasyon ve çaprazlama operatörlerini kullanarak hedef çözüme ulaşmayı amaçlayan popülasyon tabanlı bir algoritmadır.

Sezgisel algoritmaların etkin çalıştıkları kullanım alanlarının belirlenebilmesi için performanslarının ölçülmesi önem taşır. Bu amaçla algoritmalar genellikle diğer algoritmalar ile karşılaştırılırlar. Problemlerin zorlukları ve özelliklerine göre algoritmaların gösterdiği performansları incelemek, algoritmaların hangi problem türlerinde daha etkin kullanılabildiğini belirlemekte yardımcı olur.

1.2. Tezin amacı ve yöntemi

Bu tezde Genelleştirilmiş Gezgin Satıcı Problemi (GGSP), GTSPLIB' de bulunan problemler için Kesikli Farksal Evrim (KFE) Algoritması kullanılarak çözülmüştür. FAE' nin daha etkili çalışabilmesi için içerisine bölgesel tarama sezgiseli

yerleştirilmiştir.

İkinci bölümde; algoritma kavramından, problem çeşitlerinden ve eniyileme yöntemlerinden bahsedilmiştir.

Üçüncü bölümde; Genelleştirilmiş Gezgin Satıcı Probleminin niteliklerinden ve kullanım alanlarından bahsedilmiştir.

Dördüncü bölümde ise; sezgisel yöntemler incelenmiştir, KFE Algoritmasının GGSP için nasıl düzenlendiği anlatılmıştır.

Beşinci bölümde Deney Tasarımı Yönteminden, Deney Tasarımında kullanılan stratejilerden ve özel olarak tez içinde kullanılan 2^{6-2} Kısmi Faktöriyel Deney Tasarımı anlatılmıştır. Bunun yanı sıra algoritmada kullanılan parametrelerin seviyelerinin ayarlanması da bu bölüm içinde anlatılmıştır.

Altıncı bölümde deneysel sonuçlara yer verilmiştir. 54 test probleminin tamamı için KFE Algoritması, Bontoux, Artigues ve Feillet (2009)'un Memetik Algoritması ve Taşgetiren, Suganthan ve Pan (2009)'un eDDE algoritması ile kıyaslanmıştır. Eniyi değerleri bilinen 41 problem için KFE Algoritması MA ve eDDE Algoritmalarının yanı sıra mrOXGA ve RKGA'nın sonuçları ile de kıyaslanmıştır.

Yedinci bölümde tezde bulunan sonuçlar özetlenmiştir.

1.3. Literatür taraması

GSP (Gezgin Satıcı Problemi) literatürde çok sayıda çalışmada yer alan bir problemdir. GGSP (Genelleştirilmiş Gezgin Satıcı Problemi) ise ilk olarak Dantzig, Johnson ve Fulgerson' un 1954 yılında ki makalesiyle Yöneylem Araştırması kaynaklarına girmiştir.

GGSP' nin posta dağıtımı (Laporte, Asef-Vaziri ve Sriskandarajah, 1996), bilgisayar dosya işleme (Henry-Labordere, 1969), depolardan sipariş toplama (Noon, 1988), dönel parçalar için işlem planlama (Ben-Arieh, 2003), ve yardım kuruluşlarında yardım edilecek kişilerin sıralanması (Saskena, 1970) gibi çeşitli konularda

uygulanmış örnekleri bulunmaktadır.

Genelleştirilmiş Gezgin Satıcı Problemi için kullanılan çözüm yöntemleri üç grupta incelenebilir; kesin çözüm yöntemleri, GGSP' yi GSP' ye çeviren yöntemler ve modern sezgisel çözüm yöntemleri.

Dinamik Programlama, Tam Sayı Programlama, Dal ve Sınır Yöntemi, Dal ve Kesme yöntemleri kesin çözüm yöntemleri olarak tanımlanabilirler. Laporte ve Nobert (1983), Laporte, Mercure ve Nobert (1987), Noon ve Bean (1991), Fishetti, Gonzales ve Toth (1995, 1997, 2002) yaptıkları çalışmalarda GGSP problemin çözümü için çeşitli kesin çözüm yöntemleri önermişlerdir.

Çeşitli araştırmacılar GSP' ye uygulanabilecek sezgisel ve kesin algoritmaların çeşitliliğinin fazlalığından dolayı GGSP'yi GSP'ye çeviren çözüm yöntemleri önermişlerdir (Ben-Arieh v.d., 2003). Yöntem ilk olarak Noon ve Bean (1993) tarafından ortaya konmuştur. Lien, Ma ve Wah (1993) tarafından bu yöntem kullanılarak yapılan çalışma sonucunda GSP'de karşılık gelen kümede çok fazla şehir bulunmuş, daha sonra Dimitrijevic ve Saric (1997) tarafından yapılan çalışmada ise karşılık gelen GSP kümesinde daha az şehirli bir sonuca ulaşılmıştır. Çeşitli çalışmalarda bu yöntemin dezavantajının dönüşüm sonucunda problemin boyutunun büyümesi olduğu bildirilmiştir.

GGSP için kullanılan verilerin sayısı arttıkça kesin çözüm yöntemleriyle çözmek zorlaşmakta, çözüm için geçen süre artmaktadır. Bu sebeple GGSP' nin çözümü için çeşitli sezgisel yöntemler uygulanmaktadır.

En sık kullanılan sezgisel yöntemlerden biri Noon (1988) tarafından ortaya konulan en yakın komşuluk sezgiselidir. En uzağa ekleme, en yakına ekleme, en ucuz ekleme yöntemleri Fischetti, Salazar-Gonzalez, Toth (1997) tarafından önerilmiştir. Renaud ve Boctor (1998) yine Renaud, Boctor ve Laporte (1996) tarafından ortaya konulan I^3 sezgiseline dayanan GI^3 (Genelleştirme, Başlatma, Ekleme ve Geliştirme) algoritmasını geliştirmişlerdir.

Synder ve Daskin'in (2006), Rasgele Anahtar Genetik Algoritmasını (RKGA) GGSP

için kullanmalarıyla birlikte gelişim algoritmalarının GGSP için uygulanmasına literatürde daha sık rastlanmaya başlamıştır. Tasgetiren v.d. (2004) Kesikli Parça Sürü En İyilemesini (DPSO), Tasgetiren, Suganthan ve Pan (2007) Genetik Algoritmayı ve Tasgetiren v.d. (2007) Hibrit Iterated Greedy Algoritmasını (Iterated Greedy Algorithm) GGSP' ye uygulamışlardır. Silberholz ve Golden (1997), mrOXGA adını verdikleri farklı bir genetik algoritmayı GGSP' yi çözmek için kullanmışlardır.

Genetik algoritmaların bölgesel tarama sezgisel yöntemleri ile birleştirilmesi sonucunda Memetik algoritma ortaya çıkmaktadır. Bontoux, Artigues, Feillet (2009), Memetik Algoritma'yı GGSP' nin çözümü için kullanmışlardır. GTSPLIB Kütüphanesinden seçilen 54 test problemi için denemeler, 2GB hafızalı, 2GHz Intel Pentium IV makinelerde yapılmıştır. Çalışmada popülasyondaki birey sayısı 50, çaprazlanma sayısı 30, en fazla tekrar sayısı 100, eniyi sonucu doğrulamadan en fazla tekrar sayısı 10, mutasyona uğrama olasılığı 0,05 olarak alınmıştır. 41 adet eniyi sonuca ulaşılmış, 13 eniyi bilinen ortalama çözüm değerinden 10 tanesi doğrulanmıştır.

Tasgetiren v.d. (2008) GGSP için Kesikli Farksal Evrim (KFE) Algoritmasını geliştirmişlerdir. Çözüm kalitesini arttırmak için bu algoritmayı Bölgesel Tarama Sezgiseli ile birleştirilmiştir. Hibrid algoritmanın performansı 51 (11) şehirden 1084 (217) şehre kadar olan GTSPLIB Kütüphanesinde ki problemler üzerinde denenmiştir. Bozma boyutu 5, perturbasyon derecesi 3, çaprazlanma olasılığı 0,9, mutasyona uğrama olasılığı 0,2 ve çaprazlama tipi PTL çaprazlama olarak alınan çalışmada, test problemleri 512 MB hafızalı 1,83 GHz Intel Centrino Duo bilgisayarda denenmiştir. Algoritmanın sonuçları test problemlerinin %95' inde eniyi çözüme ulaşmıştır. Algoritmanın ortalama CPU zamanı 12 dakika olarak hesaplanmıştır. Sonuçlar Synder ve Daskin'in (2006) Rasgele Anahtar Genetik Algoritması ve Silberholz ve Golden'ın (1997) mrOXGA Algoritması ile kıyaslanmıştır. Bulunan sonuçların kıyaslanan çalışmalardan daha iyi performans gösterdiği görülmüştür.

Tasgetiren v.d. (2009) yaptıkları çalışmada Topluluk KFE Algoritması'nı (eDDE)

GGSP için uygulamışlardır. Bu çalışmada her birinde 15 birey bulunan 6 farklı paralel topluluk kullanılmıştır. Bozma boyutu 5, perturbasyon derecesi 3, çaprazlanma olasılığı 0,9, mutasyona uğrama olasılığı 0,2 olarak uygulanmıştır. Çalışma, 512 MB hafızalı, 3 GHz Intel Pentium IV makinelerde denenmiştir. Algoritma % 92 eniyi sonuca ulaşmıştır. Sonuçlar Synder ve Daskin'in (2006) Rasgele Anahtar Genetik Algoritması ve Silberholz ve Golden'ın (1997) mrOXGA Algoritması ile kıyaslanmıştır. Bulunan sonuçların çalışma süreleri ve çözüm kalitesi olarak mrOXGA ile eşdeğer olduğu görülmüştür.

İKİNCİ BÖLÜM

2. PROBLEMLER VE ENİYİLEME

2.1. Algoritma kavramı ve problem sınıfları

Algoritma kavramı, TDK tarafından bir sorunun çözümü için, sonlu sayıda adım biçiminde tanımlanmış, sonlu bir kurallar kümesi olarak ifade edilmektedir. Bir başka deyişle algoritma insan veya makine tarafından gerçekleştirilecek bir eylemin mekanik olarak ardışık aşamalar halinde yürütülebilmesi için verilen komutlara denir. Kullanılan algoritmanın tipi ve problemin zorluğu çözüm süresini etkilemektedir.

Bazı problemlerin çözümünde kullanılan algoritmanın çalışma süresi, girilen verinin büyüklüğüne polinomsal olarak bağlıdır. Bu tip problemler P (polinomsal) problem kategorisinde yer alırlar. Bu tip algoritmalara polinomsal zamanda çalışan algoritma adı verilmektedir.

Bir diğer grup problemde üssel zaman karmaşıklık fonksiyonuna sahip olan problemleri içerir. Bu tip problemlere NP adı verilir. Bu tip problemler belirli bir Turing Makinesi ile çok terimli zamanda doğrulanabilirler ve bu şekilde doğrulanabilen her problem NP sınıfındadır. NP tipi problemler, P tipi problemleri içerirler yani bir problem için polinomsal zaman algoritması varsa, algoritma üssel zaman algoritmasına dönüştürülebilir.

En az her bir NP problem kadar zor olan problemlerin bulunduğu sınıfa NP-zor (NP-hard) denilir. NP-zor sınıfındaki herhangi bir problem çok terimli zamanda çözülebilirse, NP sınıfındaki bütün problemler çok terimli zamanda çözülebilir. Çoğu çizelgeleme problemi bu sınıf içerisinde yer almaktadır.

2.2. Eniyileme

Eniyilemek sözcüğü bir durumun, fırsatın ya da kaynağın eniyi veya en etkili bir şekilde kullanılması ya da belirli koşullar altında mümkün olan alternatifler

arasından eniyisini seçmek olarak tanımlanabilir (Rardin, 2000). Yöntem, ekonomi ve mühendislik alanında sıkça kullanılmaktadır. Ayrıca pek çok tasarım probleminde de eniyileme yönteminden yararlanılmaktadır. Karar değişkenlerine bağlı eniyi çözümün araştırıldığı problemlere eniyileme problemleri denilir. Bu tip problemlerde eniyi çözüm olarak bahsedilen durum, herhangi bir kıstasın en küçüklenmesi ya da en büyüklenmesidir. Bu kıstaslar bir amaç fonksiyonu ($f(x)$) ile temsil edilirler. Bu problem aşağıdaki gibi ifade edilebilir.

$$\min f(x)$$

$$g_i(x) \le b_i \qquad i = 1,...,m$$

$$h_j(x) = c_j \qquad \mathrm{j} = 1,....,\mathrm{n}\,. \tag{2.1}$$

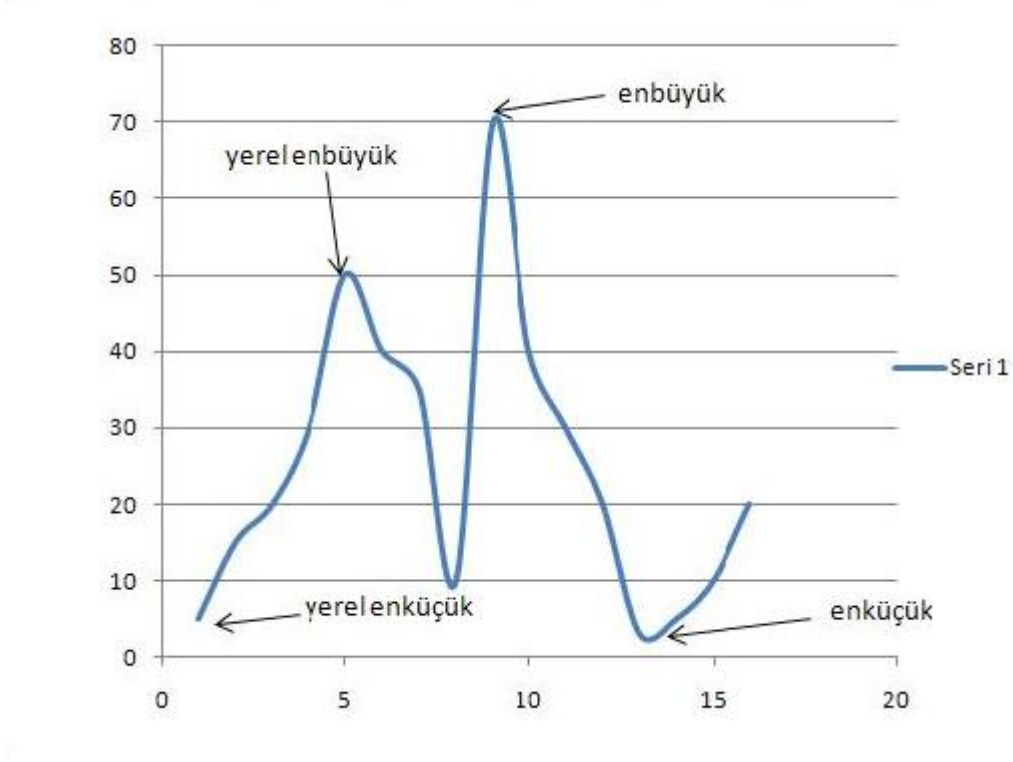
Yukarıdaki gibi, matematiksel ifadeler kullanarak bir sistemin modellenmesine matematiksel modelleme adı verilir. Eğer amaç fonksiyonlarından her biri doğrusal fonksiyon ise bu problem doğrusal programlama problemi olarak tanımlanır. Problemin amaç fonksiyonlarından en az birisi doğrusal fonksiyon değil ise problem doğrusal olmayan programlama modeli olur.

Doğrusal programlama problemleri Simpleks Yöntemi ile kolayca kesin sonuca ulaşabilmektedirler. Doğrusal olmayan problemlerin çözümünde Simpleks gibi yöntemlerden yararlanmak problemin çözüm süresini uzatmakta ve genellikle kesin çözüme ulaşamamaktadır.

Eniyileme problemleri sürekli eniyileme ve kesikli eniyileme problemleri olarak ikiye ayrılırlar. Sürekli eniyileme de karar değişkenleri arama uzayındaki her değeri alabilirler (örn. 3.234, 536.34, 64,…). Kesikli eniyileme de ise karar değişkenleri çözüm uzayında tanımlanmış değerleri alabilirler (örn. tam sayılar,…) (Cura, 2008).

Doğrusal olmayan problemler birden fazla en küçük noktaya sahip olabilirler. Bu en küçük noktalar bölgesel ya da global düzeyde olabilir. Global eniyileme bütün çözüm kümelerinin içinden eniyi çözümü bulurken, bölgesel eniyileme bir çözüm kümesi içinden bölgesel olan eniyiyi bulur. Global eniyileme başlangıç şehrinin neresi olduğunu önemsemeden her seferinde aynı sonucu verirken, bölgesel taramada bulunan çözüm başlangıç şehrine göre değişiklik gösterebilir.

Global eniyiyi bulmak bazı çalışmalarda çok güç olabilmektedir. Bulunsa bile uzun çözüm süreleri gerektirmektedir. Bu tip problem çözümleri, iş hayatında kullanıldığından problemin çözüm süresi önem taşımaktadır. Bu sebeple bölgesel eniyiyi bulmak, global eniyiyi bulmaktan daha çok tercih edilebilir (Erentürk, 2004).



Şekil Error! Use the Home tab to apply Başlık 2 to the text that you want to appear here. **Enküçük, bölgesel enküçük, enbüyük, bölgesel enbüyük sonuçlar**

Problemlerin zorluk dercesine bağlı olarak çözüm uzayı da büyümektedir. Böylece eniyiye ulaşmak için geçen süre artmakta ya da eniyiye ulaşılamamaktadır. Problemler için çözümler iki ana başlık altında toplanabilir; kesin çözüm yöntemleri ve sezgisel çözüm yöntemleri. Eniyileme problemlerinin çözümü için, doğrusal programla, tam sayı programlama, dinamik programlama gibi kesin çözüm yöntemlerinin yanı sıra benzetimli tavlama (simulated annealing) , tabu araması (tabu search), farksal evrim (differential evaluation), genetik algoritmalar (genetic algorithms) gibi sezgisel ve modern sezgisel yöntemlerle çözümler geliştirilmiştir.

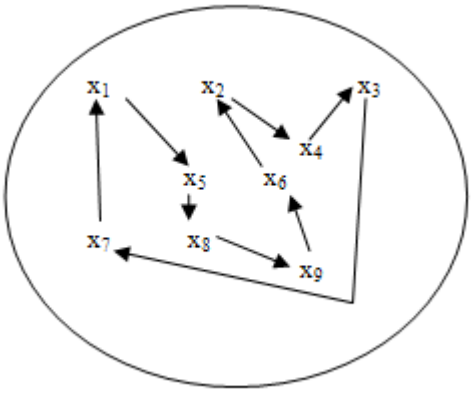
ÜÇÜNCÜ BÖLÜM

3. GENELLEŞTİRİLMİŞ GEZGİN SATICI PROBLEMİ

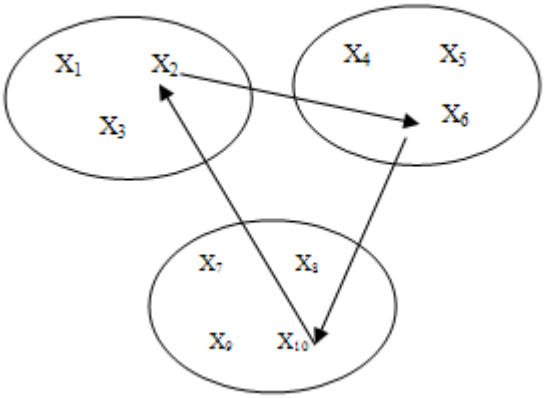
3.1. Genelleştirilmiş gezgin satıcı problemi (GGSP)

Gezgin Satıcı Problemi (GSP), 1880'lerde Sir William R. Hamilton' un öne sürdüğü Ikosyon Oyunu'na dayanmaktadır. K. Menger tarafından Hamilton Turu bulma problemi olarak isimlendirilmiştir. J.B. Robinson problemi Gezgin Satıcı Problemi olarak isimlendirmiştir. Problem literatüre Dantzig, Johnson ve Fulkerson' un (1954) makalesi ile girmiştir (Kara, Güden, Öztürk, Seyhan, 2009).

GGSP, GSP' den türetilmiştir. GGSP, Henry-Lapordere (1969), Saskena (1970) ve Srivastava' nın (1969) ayrı ayrı yaptıkları çalışmalarla Yöneylem Araştırması literatürüne girmiştir. GSP' nin amacı bir satıcının n adet şehri en kısa yoldan gezmesini ve başlangıç şehrine dönmesini sağlamaktır. GGSP' de ise n adet şehir, m adet kümeye ayrılır $v_1 \cup ... \cup v_m = v$, $v_1 \cap ... \cap v_m = \emptyset$ ve $v_i \cap v_k = \emptyset$, $j \neq k$. Problemde satıcının m adet kümeye uğrayarak, sonunda başlangıç şehrine dönecek şekilde turu tamamlaması beklenmektedir. GGSP' turu her kümeden yalnız ve yalnız bir şehre uğradığında tamamlanır.



Şekil 3-1 Gezgin Satıcı Problemi Turu



Şekil 3-2 Genelleştirilmiş Gezgin Satıcı Problemi Turu

GGSP' nin geniş uygulama alanları bulunmaktadır. GGSP gerçek hayat problemleri için GSP' den daha ideal bir modelleme aracıdır. GGSP' de şehirlerin kümelenmesi gerekmektedir; ancak sonuçta gruplar bir bütün olarak düşünüldüğünde problem GSP' ye benzetilebilir. Buna dayalı olarak GGSP' nin GSP' yi içerdiği ve GTSP' nin uygulama alanının GSP' ye göre daha geniş olduğu düşünülebilir **(**Hao, Huang ve Cai, 2008).

GGSP pek çok lojistik uygulamasında, malzeme akış sisteminin tasarımında, rassal araç rotalama problemlerinde, posta toplanması ve dağıtım problemlerinde, uçaklar için havaalanı rotalanmasında, depolardaki taşıyıcıların güzergahlarının planlanmasında ve benzeri uygulamalarda kullanılabilir (Laporte, Asef-Vaziri ve Sriskandarajah, 1996).

Literatürde GSP ile ilgili çok sayıda makale bulunurken GGSP konusunda yapılmış çalışmalar çok sınırlıdır. GGSP için yazılmış algoritmaların pek çoğu temel olarak dinamik programlama yöntemine dayanmaktadır. Günümüzdeki bilgisayar donanım teknolojisi ile küçük çaplı problemlere çözüm aramak mümkündür ancak dinamik programlama algoritması yüksek bilgisayar kapasitesi ve oldukça uzun çözüm süresi gerektirdiğinden çalışmaların sonuçları yeterince iyi çıkmayabilir **(**Hao, Huang, ve Cai, 2008).

Dinamik programlama algoritmalarının temel yöntemi GGSP' yi GSP' ye çevirip, var olan algoritmalarla GSP' yi çözmektir. Her ne kadar bu şekilde çözüm mümkün

gözükse de, teknik kısıtlar sebebiyle bu yöntem ile yapılan dönüşüm sonucunda problemin boyutları üç katına kadar artabilir.

Problemin çözümünde doğru sonuca ulaşılabilmesi kadar, sonuca hızlı ulaşabilmesi de önemlidir. Bu sebeple son yıllarda, GGSP' nin çözümü için çeşitli sezgisel ve modern sezgisel yöntemler uygulanmıştır.

DÖRDÜNCÜ BÖLÜM

4. MODERN SEZGİSEL YÖNTEMLER

4.1. Modern Sezgisel Yöntemlere Giriş

Fen bilimleri ve sosyal bilimler alanlarında karşılaşılan pek çok problem doğrusal olmayan bir yapıya sahiptir. Doğrusal olmayan problemlerin değişken sayısına ve veri tiplerine bağlı olarak problemin modellenmesi ve çözümü zorlaşmaktadır. Doğrusal olmayan problemlerin çözümü için pek çok farklı teknik geliştirilmiştir. Bu tip problemler deterministik yöntemleri kullanarak çözülmeye çalışıldığında istenilen sonuca ulaşamamak ya da istenilen sonuca çok uzun sürelerde ulaşmak gibi sorunlarla karşılaşılabilir. Bu sebeple doğrusal olmayan problemlerin çözümünde sezgisel yönteme dayalı yapay zeka eniyileme yöntemleri kullanılabilmektedir. Bu sezgisel yöntemler bölgesel tarama gibi operatörler eklenerek daha hızlı hale getirilebilir. Özellikle popülasyona dayalı sezgisel yöntemler hızlı bir şekilde sonuç vermektedir. Bu tekniklerden bazıları; Genetik Algoritmalar, Farksal Evrim Algoritmaları ve Memetik algoritmalardır (Keskintürk, 2006).

4.2. Genetik Algoritma (GA)

GA en yaygın kullanılan, temeli popülasyon yani kromozom olarak adlandırılan alternatif çözüm setleri olan sezgisel eniyileme yöntemlerinden biridir. GA en başta Charles Darwin' in "doğal seleksiyon" yani evrimsel süreçlerde "en güçlü olanın hayatta kalması" ilkesine dayanmaktadır. 1970'li yıllarda Prof. John Holland GA' i ilk kez eniyileme yöntemi olarak kullanmıştır. GA uygulamalarına parametre ve sistem tanılama, kontrol sistemleri, robot uygulamaları, görüntü ve ses tanıma, proje çizelgeleme, kombinatoryal eniyileme problemleri, ağ tasarım problemleri, rotalama problemleri, sosyal ve ekonomik planlama konularında kullanımına rastlanmaktadır. GA uygulamalarından bazıları GA, gezgin satıcı problemi benzeri kombinatoryal problemlerde etkili olarak çalışmaktadır. Klasik genetik algoritmaların performansları dinamik sahada yeterli olmayınca GA için çeşitli değişiklikler yapılmıştır (Özçelik, 2007).

Problemin çalışma sisteminin daha iyi anlaşılabilmesi için öncelikle GA' da kullanılan temel kavramların anlaşılabilmesi gerekmektedir (Paksoy, Uzun, 2008).

Kromozom: GA bir başlangıç popülasyonu (toplumu) ile başlar. Bu popülasyon kromozom (birey) adı verilen elemanlardan meydana gelir.

Gen: Bir bireyin kalıtsal özelliklerini taşıyan parçaya gen adı verilir. Genler dizi haline getirildiğinde kromozomlar oluşturulur.

Toplum Büyüklüğü: Toplum büyüklüğü kromozom (birey) sayısıdır.

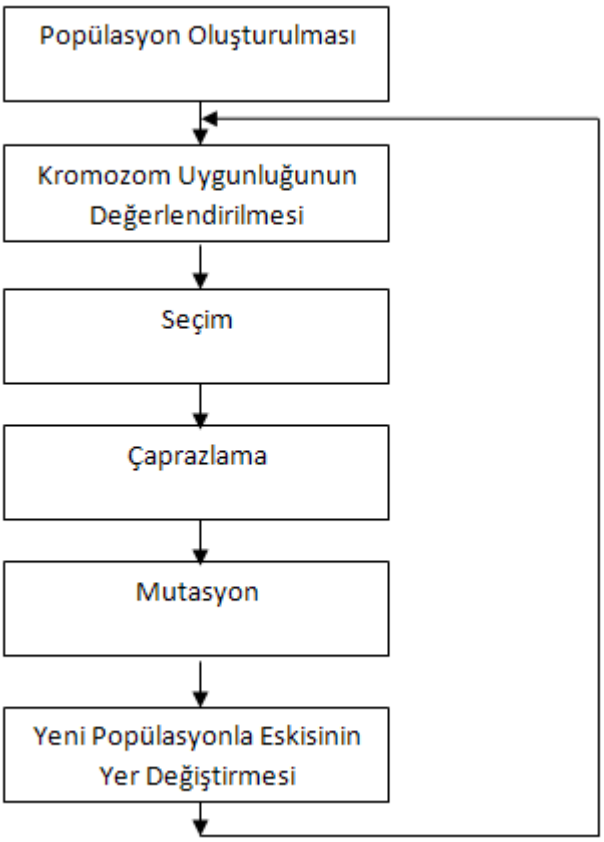
Uygunluk Değeri: Uygunluk değeri, problemin çözümü için kullanılan amaç fonksiyonunun sonucunda çıkan, bireyin kalitesini belirten değerdir.

GA bir popülasyon (kromozomlar) ile çalışmaya başlar. Bu popülasyonun çözümü yeni popülasyonların üretilmesi için kullanılır. Her bir dizinin (string), yapılan uygulamaya özel bir fonksiyon ile hesaplanan bir uygunluk değeri bulunmaktadır. Yeni popülasyonun eskisinden daha iyi olabilmesi için, yeni popülasyonun bireylerini üretmek için elde edilen çözümler uygunluk değerlerine göre seçilmektedir. Uygunluk değeri seçiminde, "elitist strateji", "rulet çarkı" ve "turnuva seçim yöntem"i gibi farklı stratejiler uygulanabilir. Bu işlem belirli bir durum sağlanıncaya kadar (belli sayıda popülasyon elde edilmesi, eniyi çözümün gerçekleşmesi…) devam eder. Popülasyonun bireyleri rastgele oluşturulduktan sonra, popülasyona seçim, çaprazlama, mutasyon gibi işlemler uygulanır.

GA uygulanabilmesi için aşağıdaki unsurlar gerekmektedir:

- Problemin mevcut çözümleri için gerekli bir tanımlama,
- Başlangıç popülasyonunun oluşturulması,
- Değerlendirme fonksiyonu,
- Yeniden üreme sırasında çocukların kompozisyonunu değiştiren genetik operatörler,
- GA' nın kullandığı parametre değerleri.

GA' nın temel basamakları Şekil 4.1.' de görülmektedir.



Şekil 4-1 Klasik Genetik Algoritmaların Genel Akış Şeması

4.3. Memetik Algoritma (MA)

Memetik algoritmalar Genetik Algoritmalar (GA) gibi nesillere dayalı sezgisel aramalar yaparlar. Memetik Algoritmalar (MA) GA' lara bölgesel tarama operatörlerinin eklenmesiyle oluşturulurlar. GA' lar ile MA' lar arasındaki fark, MA' lar bölgesel eniyi çözüm uzayı arasında arama yaparken GA' ların bütün çözüm uzayında arama yapmasıdır (Türkbey, 2002).

MA adı ilk olarak Norman ve Moscato' nun 1989 yılında yaptığı çalışma ile kullanılmıştır. Bu algoritmalara kimi çalışmalarda "Hibrid evrimsel algoritmalar" adı ile de karşılaşılabilir (Syswerda, 1991).

MA, NP-zor eniyileme problemlerinin çözümü için kullanılabilir. kurulum süreleri ve son tarihleri göz önünde bulundurularak tek makineli çizelgeleme, çok boyutlu Knapsack problemi, doğrusal olmayan tam sayı programlama, kareli atama problemi, kombinatoryal eniyileme problemlerinin araç rotalama, en küçük kapsar ağaç, görev atama problemi vb… problemlerinin çözümünde, robotların kontrolü ve makinelere öğretmede, elektronik ve mühendislikte, trafik kontrol, sistem modelleme, bilgisayar destekli tasarım vb… uygulamalarda, moleküler eniyileme problemlerinde, matematik, ekonomi gibi alanlarda kullanılabilirler (Moscato, Cotta, 2005).

Popülasyonun bireyleri $\pi = [\pi_1, \pi_2, .., \pi_{NP}]$ ile tanımlanırsa, MA'nın temel basamakları aşağıdaki gibi sıralanmıştır (Bontoux, 2009):

Parametreleri Belirle
NP adet bireyi olan başlangıç popülasyonunu oluştur
Bu bireylere bölgesel tarama uygula
do

π_a *ve* π_b *gibi iki birey seç*
$\pi_c = Çaprazla(\pi_a, \pi_b)$
$\pi_c = YerelTarama(\pi_c)$
π_c *'yi popülasyona ekle*
N tane eniyi bireyi popülasyonda tut
Popülasyondaki her bireye μ *olasılığı ile mutasyon uygula*

$while(Sonlandirma$ Kosulu$)$

Şekil 4-2 Memetik Algoritmanın Temel Basamakları

4.4. Farksal Evrim Algoritması (FEA)

Farksal Evrim Algoritması (FEA), işleyiş ve kullanılan operatörler itibariyle genetik algoritmaya dayanan, özellikle sürekli problemler söz konusu olduğunda etkin çözümler üretebilen bir sezgisel eniyileme yöntemidir. FEA, Storn ve Price (1995) tarafından geliştirilmiştir. FEA popülasyon tabanlıdır ve aynı anda birden çok noktada arama yapar.

Genetik Algoritma'da (GA) kullanılan çaprazlama, mutasyon ve seçim operatörleri FEA' da da kullanılır. GA' dan farklı olarak bu operatörler tüm popülasyona sırasıyla uygulanmaktadır. Mutasyon (soydeğişim) işleminde popülasyondan rastgele seçilmiş

iki üyenin ağırlıklı farkları, mutant çözümü üretmek için üçüncü üyeye eklenirler. Deneme çözümünü (trial solution) üretmek için, mutant çözümü hedef çözümle (target solution) birleştiren bir çaprazlama operatörü uygulanır. Kimin gelecek nesil için ayakta kalacağını belirlemek için, hedef ve deneme amaç fonksiyonu değerlerine göz önüne alan bir seçim operatörü uygulanır (Tasgetiren v.d., 2009). Orijinal olarak FEA sürekli fonksiyonların eniyilemesi için geliştirilmiştir. FEA' nın temel adımları şöyledir (Özçelik, 2007):

Parametreleri belirle
Başlangıç popülasyonu oluştur
Popülasyonu değerlendir
do

Mutasyon uygula
Çaprazlama uygula
Değerlendir
Seçim

$while(Sonlandirma \quad kosulu)$

Şekil 4-3 Farksal Evrim Algoritması

FEA başlangıç kontrol parametrelerinin değerlerinin, değişkenlerin sınır değerlerinin ve popülasyonunun belirlenmesiyle başlar. Popülasyon (amaç vektörlerinin sayısı) sayısı, $NP \geq 4$ olmalıdır. Her birey önceden belirlenen arama aralığında, parametreleri tesadüfi ve uniform olarak dağılan, m boyutlu bir vektöre sahiptir. Sürekli FEA' da çözüm adayları gerçel sayılara dayanan bireylerle gösterilirler. FEA' nın temel unsurları aşağıda gösterilmiştir:

NP: Popülasyon büyüklüğü (kromozom sayısı), $NP \geq 4$

n: Gen sayısı (eniyilenecek değişken sayısı) $j = 1,2,..,n$

CR: Çaprazlama oranı, $CR \in (0,1)$

k: Nesil indeksi

F: Mutasyon faktörü, $F \in (0,2)$)

x_{ij}^k: Hedef çözümü. k. nesilde, i. bireyin j. parametresi (gen), $x_{ij}^k = \left[x_{i1}^k, x_{i2}^k, .., x_{in}^k\right]$, $j = 1,2,..,n$

v_{ij}^k: Mutant çözüm. k. nesilde, i. bireyin j. parametresi, $v_{ij}^k = \left[v_{i1}^k, x_{i2}^k, .., v_{in}^k\right]$, $j = 1,2,..,n$.

u_{ij}^k: Deneme çözümü. k. nesilde, i. bireyin j. parametresi, $u_{ij}^k = \left[u_{i1}^k, u_{i2}^k, .., u_{in}^k\right]$, $j = 1,2,..,n$.

$x^k = \left[x_1^k, x_2^k, .., x_{NP}^k\right]$: Hedef Popülasyon. k. nesilde hedef popülasyonun bireylerinin kümesi.

$v^k = \left[v_1^k, v_2^k, .., v_{NP}^k\right]$: Mutant Popülasyon. k. nesilde mutant popülasyonun bireylerinin kümesi.

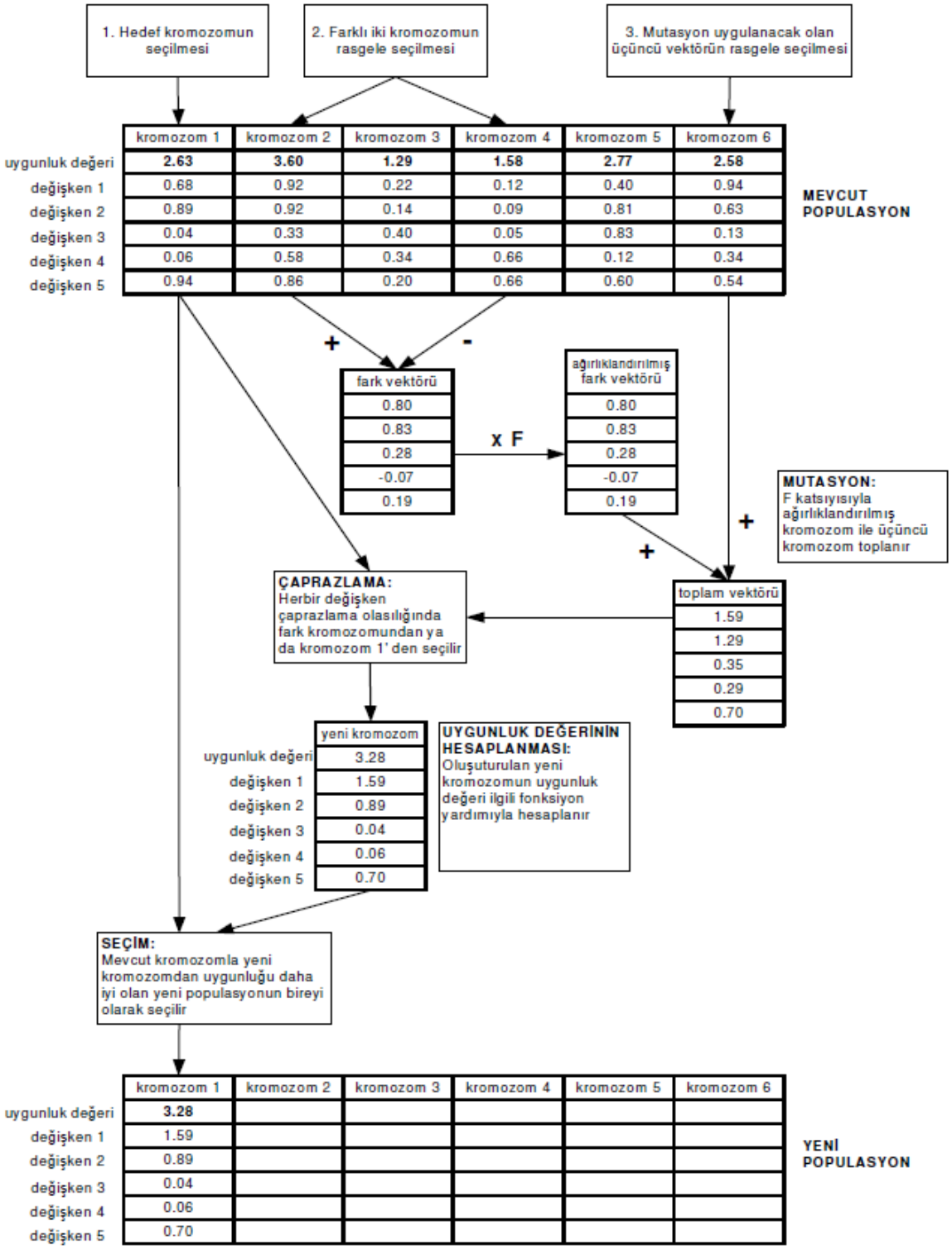
$u^k = \left[u_1^k, u_2^k, .., u_{NP}^k\right]$: Mutant Popülasyon. k. nesilde mutant popülasyonun bireylerinin kümesi.

$f\left(x_i^k\right)$: Amaç Fonksiyonu. Enazlama problemlerinde x_i^k bireyi için amaç fonksiyonudur.

FEA farklı problemler için, mutasyona uğrayacak vektöre, fark vektörlerinin sayısına ve kullanılan çaprazlama tiplerine bağlı olarak çeşitlendirilmiştir. Stratejiler *DE/x/y/z* şeklinde gösterilmektedirler. *DE*, Farksal evrim algoritmasını, *x*, mutasyona uğrayacak vektörü, *y*, fark vektörlerinin sayısını, *z* ise çaprazlama tipini göstermektedir. *x*, bir önceki nesilden rastgele seçilen (*rand*) ya da bir önce ki neslin eniyi vektörü (*best*) olabilir. *y*' nin temsil ettiği fark vektörü tek (1) ya da çift (2) olabilir. z' nin temsil ettiği çaprazlama tipi ise üstel (*exp*) ya da binomsal (*bin*) olabilir. Price ve Storn, FEA için on farklı çalışma stratejisi belirlemişlerdir. Probleme göre seçilecek strateji deneme yanılma yoluyla bulunur. Bu stratejiler şunlardır (Özçelik, 2007);

1. DE/rand/1/bin

2. *DE/best/1/bin*
3. *DE/rand-to-best/1/bin*
4. *DE/rand/2/bin*
5. *DE/best/2/bin*
6. *DE/best/1/exp*
7. *DE/rand/1/exp*
8. *DE/rand-to-best/1/exp*
9. *DE/best/2/exp*
10. *DE/rand/2/exp.*



Şekil 4-4 Sürekli FEA 'nin işleyişi (Schmidt, Thierauf, 2005).

- $f(x) = x_1 + x_2 + x_3 + x_4 + x_5, \quad \forall x_i = (0.0, 1.0)$. (4.1.)

Yukarıdaki örnekte, sürekli eniyileme problemleri için kullanılan FEA' da, her bir

bireye ait olan parametreler ondalık sayı temeline dayanır. Kesikli uzayda çalışabilmek için Pan, Tasgetiren ve Liang (2008) çözüm kümesindeki tam sayılarla çalışabilen Kesikli Farksal Evrim (KFE) Algoritmasını sunmuşlardır. KFE algoritmasında, NP sayıda birey bulunmaktadır. *n*-boyutsal herhangi bir kesikli kombinatoryal eniyileme problemi için, her bireyin, $\pi_i^k = \left[\pi_{i1}^k, \pi_{i2}^k, .., \pi_{in}^k\right]$, her bir parametresi, kesikli değer almaktadır. Farksal değişme, popülasyondaki eniyi bireyin perturbasyonları şeklinde tanımlanmıştır. Perturbasyonlar, rassal olarak gerçekleştirildiği için, elde edilecek mutant bireylerin birbirinden farklı olması istatistiksel olarak beklenir. Mutant birey aşağıdaki gibi elde edilir:

$$v_i^k = \begin{cases} DC_d\left(\pi_g^{k-1}\right) & eger \quad r < P_m \\ Ekleme\left(\pi_g^{k-1}\right) & aksi \quad takdide \end{cases} \tag{4.2.}$$

Denklem 4.2.'de , π_g^{k-1}, bir önceki nesilde hedef popülasyondaki eniyi birey; P_m, perturbasyon olasılığı; DC_d, ise bozma derecesi *d* olan *Boz/Yap* prosedürüdür. 0 ile 1 arasında uniform bir sayı, r, üretilir. Eğer r, perturbasyon olasılığından, P_m, küçük ise, $DC_d\left(\pi_g^{k-1}.\right)$ uygulanarak, mutant birey elde edilir. Aksi takdirde, bir önceki nesildeki eniyi birey den rassal olarak bir eklenti yapılır. Algoritmanın sözde kodunu daha iyi anlamak için Denklem [4.2.], $v_i^k = P_m \oplus DC_d\left(\pi_g^{k-1}\right)$ olarak gösterilecektir. Mutasyon adımından sonra, deneme bireyi aşağıdaki gibi elde edilir:

$$u_i^k = \begin{cases} CR\left(v_i^k, \pi_i^{k-1}\right) & eger \quad r < P_c \\ v_i^k \quad aksi \quad takdide \end{cases} \tag{4.3.}$$

Denklem 4.3.'te, CR çaprazlama operatörü ve P_c ise çaprazlama olasılığıdır. 0 ile 1 arasında uniform rassal sayı, r, üretilir; eğer r, çaprazlama olasılığı Pc den küçük ise, çaprazlama operatörü uygulanarak deneme bireyi elde edilir. Aksi halde, mutant birey, deneme bireyi olarak alınır. Son olarak, deneme bireyinin, u_i^k, gelecek nesil için hedef popülasyonun bir üyesi olup olmayacağına karar vermek için, deneme bireyi, önceki nesildeki eşleniği, π_i^{k-1}, ile kıyaslanır ve amaç fonksiyon değeri iyi olan gelecek nesil popülasyon için seçilir.

$$\pi_i^k = \begin{cases} u_i^k & eger \quad f(u_i^k) < f(\pi_i^{k-1}) \\ \pi_i^{k-1} & aksi \quad takdirde \end{cases} \tag{4.4.}$$

4.4.1. *FEA' nin GGSP İçin Çözüm Gösterimi*

Bu bölümde FEA'nın GGSP' nin çözümü için bir "çözüm gösterimi" sunulmuştur. Tablo 4.1. Çözüm Gösterimi

Tablo 4-1 FEA' nin GGSP İçin Çözüm Gösterimi

	J	1	2	…	*m*-1	*m*	*m+1*
π	n_j	n_1	n_2	…		n_m	n_1
	π_j	π_1	π_2	…		π_m	π_1
	$d_{\pi_j\pi_{j+1}}$	$d_{\pi_1\pi_2}$	$d_{\pi_2\pi_3}$	…	$d_{\pi_{m-1}\pi_m}$	$d_{\pi_m\pi_1}$	

Tablo 4.1.' de görüldüğü gibi, m kümesine sahip olan π çözümünün amaç fonksiyonu, toplam tur uzunluğudur ve formül 4.5. ile elde edilir;

$$f(\pi) = \sum_{j-1}^{m-1} d_{\pi_j\pi_{j+1}} + d_{\pi_{m\pi_1}} \tag{4.5.}$$

GGSP daha iyi açıklamak için n=25 şehri olan ve m=5 kümeye ayrılmış bir örnek oluşturulmuş. Örnek kümeler $v = \{1,2,...25\}$ ve $v_1 = \{1,...,5\}$ $v_2 = \{6,...,10\}$, $v_3 = \{11,...,15\}$, $v_4 = \{16,...,20\}$, $v_5 = \{21,...,25\}$ olsun. Bu örnek için çözüm gösterimi ve amaç fonksiyonu Tablo 4.2 de gösterilmektedir.

Tablo 4-2 FEA İçin Çözüm Gösterimi Örneği

	j	1	2	3	4	5	5+1
X_i	n_j	3	1	5	2	4	3
	π_j	14	5	22	8	16	14
	$d_{\pi_j\pi_{j+1}}$	$d_{14,5}$	$d_{5,22}$	$d_{22,8}$	$d_{8,16}$	$d_{16,14}$	

Bireyin amaç fonksiyonu denklem 4.6' daki gibidir.

$$f(\pi) = d_{14,5} + d_{5,22} + d_{22,8} + d_{8,16} + d_{16,14} \tag{4.6.}$$

4.4.2. *Farksal Evrim Algoritmasının İşlemsel Adımları*

FEA algoritmasının adımları sırasıyla aşağıdaki işlemlerden oluşmaktadır.

Başlangıç değerlerini oluştur. Yani, $k = 0$, $NP = 100$ olarak ayarla.

a. NP bireylerini rastgele olarak üretmek, $\{\pi_i^0, i = 1,2,...NP\}$, $\{\pi_i^0 = [\pi_{i1}^0, \pi_{i2}^0, ..., \pi_{im}^0]\}$.

Amaç fonksiyonunu kullanarak popülasyonda ki her bireyi değerlendirmek, $f_i^0(\pi_i^0)$, $i = 1,2,....NP$.

b. Nesil Sayacını Güncellemek.

$k = k + 1$

c. Mutant Popülasyonu Üretmek.

k. neslindeki her bir birey için, π_i^k, $i = 1,2,...., NP$, 4.2. formülü kullanılarak mutant bireyler, $v_i^k = [v_{i1}^k, v_{i2}^k, ..., v_{im}^k]$, belirlenir.

d. Deneme Popülasyonunu Üretmek.

Mutasyon aşamasının ardından formül 4.3. kullanarak deneme popülasyonu, $u_i^k = [u_{i1}^k, u_{i2}^k, ..., u_{im}^k]$, üretilir.

e. Deneme Popülasyonunu Değerlendirmek.

Amaç fonksiyonunu kullanarak, deneme popülasyonu değerlendirilir. $f_i^k(u_i^k)$, $i = 1,2,...., NP$.

f. Seçim

Deneme bireyinin, u_i^k, gelecek nesil için hedef popülasyonun bir üyesi olup olmayacağına karar vermek için, deneme bireyi, önceki nesildeki eşleniği, π_i^{k-1}, ile kıyaslanır. Bir sonraki nesle aktarılacak birey amaç fonksiyonlarına bakılarak formül 4.4. ile seçilir.

g. Durdurma Koşulu

Eğer nesillerin sayısı, belirlenen sınır sayısını aşarsa ya da farklı bir durdurma kıstasına ulaşıldığında, işlem durdurulur, istenilen kıstasa ulaşılmadı ise 2. adıma geri dönülür.

4.4.3. *NEH Sezgiseli (NEH Heuristic)*

Ekleme operatörünü GSP literatüründeki en yakın komşu tarama tekniği ile kullanmak mümkün olduğu için, Nawaz, Enscore ve Ham (Nawaz, Enscore, Ham, 1983) tarafından geliştirilen NEH sezgisel yönteminin GGSP'ye uygulanması da mümkündür. GGSP' deki kümeler dikkate alınmadan, NEH'in GGSP'de kullanılışı özetle aşağıdaki gibidir:

1. Başlangıç turu, π'yi oluştur.
2. İlk iki şehir seçilir, $\{\pi_1, \pi_2\}$, ilk olarak, $\{\pi_1, \pi_2, \pi_1\}$ve $\{\pi_2, \pi_1, \pi_2\}$ kısmi turları amaç fonksiyonlarının değerlerine göre değerlendirilir. Tur Hamiltonian çevrimi türünde olduğundan, başlangıç şehrine dönecek şekilde gösterilmektedir, yani ilk şehir aynı zamanda turun son şehridir.
3. Bütün şehirler tura dahil edilinceye kadar şehirleri ekleme ve en küçük amaç fonksiyonu değerine sahip olan turun seçim işlemlerine aşağıdaki şekilde devam edilir. Her adımda, k. pozisyonundaki π_k şehri, kısmi turun mümkün olan bütün pozisyonlarına eklenir. Bu kısmi turlardan eniyisi seçilir ve bu bütün şehirler eklenene kadar devam eder.

NEH Sezgiselinin GGSP için nasıl çalıştığını göstermek için $\pi = \{2,1,4,3,5\}$ çözümünü ele alınırsa:

Başlangıç turu $\pi = \{2,1,4,3,5\}$.

İlk iki şehir öncelikle değerlendirilir; $\{2,1,2\}, \{1,2,1\}$. $\{2,1,2\}$ turunun amaç fonksiyonunun değerinin $\{1,2,1\}$ turunun amaç fonksiyonu değerinden daha küçük çıktığını farz edersek kısmi turların arasından $\{2,1\}$ seçilir.

Eklemeler:

3. pozisyonundaki şehir $\{4\}$ ilk seçilen kısmi turun olası üç pozisyonuna yerleştirilir: Yani, $\{4,2,1,4\}$, $\{2,4,1,2\}$, $\{2,1,4,2\}$. En küçük amaç fonksiyonu değeri $\{2,4,1,2\}$ turuna ait olduğunu farz edersek, yeni kısmi tur $\{2,4,1\}$ olarak belirlenir.

4. pozisyondaki şehir, $\{3\}$, $\{2,4,1\}$ turu olası 4 pozisyonlara eklenir: Yani, $\{3,2,4,1,3\}$, $\{2,3,4,1,2\}$, $\{2,4,3,1,2\}$, $\{2,4,1,3,2\}$. En küçük amaç fonksiyonu değerinin $\{2,4,3,1,2\}$

turuna ait olduğunu farz edersek, yeni kısmi tur $\{2,4,3,1\}$ olarak belirlenir.

Son olarak 5. Pozisyondaki şehir, $\{5\}$, $\{2,4,3,1\}$ turu içindeki pozisyonlara eklenir: $\{5,2,4,3,1,5\}$, $\{2,5,4,3,1,2\}$, $\{2,4,5,3,1,2\}$, $\{2,4,3,5,1,2\}$ ve $\{2,4,3,1,5,2\}$. En küçük amaç fonksiyonu değerinin $\{2,4,5,3,1,2\}$, turuna ait olduğunu farz edersek, yeni kısmi tur $\{2,4,5,3,1,\}$, olarak belirlenir.

4.4.4. *Ekleme Yöntemleri (Insertion Methods)*

Bu bölümde GGSP' de ekleme yöntemleri anlatılmıştır; ancak basitleştirmek için GGSP' deki küme bilgileri göz ardı edilmiştir. Ekleme yöntemi, amaç fonksiyonunun değeri $f(\pi^D)$, olan, m şehirli kısmi ya da *Bozma* işleminden geçmiş turun, π^D, $m+1$ olası aralığına π_k^R şehrinin eklenmesine dayanır. Tablo 4.4.'te $\pi_k^R = 8$ şehrinin kısmi tura eklenmesi gösterilmiştir

Tablo 4-3 Başlangıç Çözümü

j	1	2	3	4		π_k^R
n_j	3	1	5	4	3	2
π_j	14	5	22	16	14	8
$d_{\pi_j^D \pi_{j+1}^D}$	$d_{14,5}$	$d_{5,22}$	$d_{22,16}$	$d_{16,14}$		

A- π_k^R şehrinin kısmi turun, π^D, ilk pozisyona eklenmesi

$$Kaldir = d_{\pi_m^D \pi_1^D}$$

$$Ekle = d_{\pi_k^R \pi_1^D} + d_{\pi_m^D \pi_k^R}$$

$$f(\pi^D) = f(\pi^D) + Ekle - Kaldir \qquad (4.7.)$$

Burada $f(\pi\ )$ ve $f(\pi^D)$ kısmi tur ve ekleme işleminden sonra turun amaç fonksiyonu değerlerini göstermektedir.

Örnek A:

$$Kaldir = d_{\pi_m^D \pi_1^D} = d\pi_4\pi_1 = d_{16,4} \qquad (4.8.)$$

$$Ekle = d_{\pi_u\ \pi_k} + d_{\pi_k\pi_v} = d_{8,14} + d_{16,8}$$

$$f(\pi\ ) = f(\pi^D) + Ekle - Kaldir$$

$$f(\pi\ ) = d_{14,5} + d_{5,22} + d_{22,16} + d_{16,14} + d_{8,14} + d_{16,8} - d_{16,14}$$

$$f(\pi\ ) = d_{14,5} + d_{5,22} + d_{22,16} + d_{8,14} + d_{16,8}$$

Tablo 4-4 $\pi_k^R = 8$ şehrini kısmi çözümün ilk pozisyona yerleştirmek

j	1	2	3	4	5	
n_j	2	3	1	5	4	2
π_j	8	14	5	22	16	8
$d_{\pi_m^D\ \pi_1^D}$	$d_{8,14}$	$d_{14,5}$	$d_{5,22}$	$d_{22,16}$	$d_{16,8}$	

B- $\pi_k^R = 8$ şehrini kısmi çözümün son boşluğuna yerleştirmek

$$Kaldir = d_{\pi_m^D\ \pi_1^D}$$

$$Ekle = d_{\pi_m^D\ \pi_k^R} + d_{\pi_k^R\pi_1^D} \qquad (4.9.)$$

$$f(\pi^D) = f(\pi^D) + Ekle - Kaldir$$

Burada $f(\pi\ )$ ve $f(\pi^D)$ kısmi tur ve ekleme işleminden sonra turun amaç fonksiyonu değerlerini göstermektedir.

Örnek B:

$$Kaldir = d_{\pi_m^D\ \pi_1^D} = d\pi_4\pi_1 = d_{16,14}$$

$$Ekle = d_{\pi_u\ \pi_k} + d_{\pi_k\pi_v} = d_{16,8} + d_{8,14} \qquad (4.10)$$

$$f(\pi\ ) = f(\pi^D) + Ekle - Kaldir$$

$$f(\pi\ ) = d_{14,5} + d_{5,22} + d_{22,16} + d_{16,14} + d_{8,14} - d_{16,14}$$

$$f(\pi\ ) = d_{14,5} + d_{5,22} + d_{22,16} + d_{8,14} + d_{16,8}$$

Tablo 4-5 $\pi_k^R = 8$ şehrini kısmi çözümün son boşluğuna yerleştirilmesi.

J	1	2	3	4	5	
n_j	3	1	5	4	2	3
π_j	14	5	22	16	8	14
$d_{\pi_j^D \pi_{j+1}^D}$	$d_{14,5}$	$d_{5,22}$	$d_{22,16}$	$d_{16,8}$	$d_{8,14}$	

C- $\pi_k^R = 8$ şehrini kısmi çözümün π_k^R, π_{k+1}^R aralığında bir boşluğa yerleştirmek

$$Kaldir = d_{\pi_u^D \pi_v^D}$$

$$Ekle = d_{\pi_u^D \pi_k^R} + d_{\pi_k^R \pi_v^D} \qquad (4.11.)$$

$$f(\pi^D) = f(\pi^D) + Ekle - Kaldir$$

Burada $f(\pi\ )$ ve $f(\pi^D)$ kısmi tur ve ekleme işleminden sonra turun amaç fonksiyonu değerlerini göstermektedir.

Örnek C:

$u = 14, v = 5$ için;

$$Kaldir = d_{\pi_u^D \pi_v^D} = d_{14,5}$$

$$Ekle = d_{\pi_u \pi_k} + d_{\pi_k \pi_v} = d_{14,8} + d_{8,5} \qquad (4.12)$$

$$f(\pi\ ) = f(\pi^D) + Ekle - Kaldir$$

$$f(\pi\ ) = d_{14,5} + d_{5,22} + d_{22,16} + d_{16,14} + d_{14,8} + d_{8,5} - d_{14,5}$$

$$f(\pi\ ) = d_{5,22}, d_{22,16}, +d_{16,14} + d_{14,8} + d_{8,5}$$

Tablo 4-6 $\pi_k^R = 8$ şehrini kısmi çözümün π_k^R, π_{k+1}^R şehirleri arasına yerleştirilmesi.

j	1	2	3	4	5	5+1
n_j	3	2	1	5	4	3
π_j	14	8	5	22	16	14
$d_{\pi_j^D \pi_{j+1}^D}$	$d_{14,8}$	$d_{8,5}$	$d_{5,22}$	$d_{22,16}$	$d_{16,14}$	

4.4.5. Boz/Yap Metodu

Iterated Greedy Algoritmasında kullanılan bozma ve yapım işlemi FE algoritmasında da kullanılabilmektedir (Tasgetiren, Liang, Pan ve Suganthan 2009). *Bozma* adımında *m* şehirli bir çözümden, rastgele *d* adet şehir tekrarlanmadan seçilir ve çözümden çıkartılır. İki tane kısmi çözüm elde edilir. İlk kısmi çözüm, π^R, çıkarılan *d* adet şehirden, çıkarıldıkları sırayla oluşturulur. İkinci kısım ise, π^D, $m-d$ şehirden oluşmaktadır.

Yapım aşamasında NEH sezgiselinden kullanılmıştır. π^R kümesinin π^D çözümüne yeniden yerleştirilebilmesi için, π^R içerisindeki ilk şehir, $\pi^R{}_1$, π^D çözümü içerisindeki olası $m-d+1$ pozisyona sırayla yerleştirilir. Oluşturulan kısmi çözümler içerisinden en kısa tur uzunluğuna sahip olan seçilir ve gelecek tekrarda bu kısmi çözüm kullanılır. Daha sonra π^R içerisindeki ikinci şehir oluşturulan kısmi çözüm içerisindeki pozisyonlara sırayla yerleştirilir ve bu işlem π^R kümesinde eleman kalmayana yani π^D yeniden m elemana sahip olana kadar devam eder.

Tablo 4.7. ile Tablo 4.12. arasında GGSP için *BozYap* işlemi gösterilmektedir. Örnekte bozma derecesi (*d*) 2 olarak kabul edilmiştir. Perturbasyon seviyesi (*p*) 1 olarak alınmıştır. Bu iki şehir arasından sadece bir tanesinin sadece tek bir tanesini aynı kümeden başka bir şehir ile yer değiştirilmesi (mutasyona uğratılması) anlamına gelir.

Tablo 4-7 Boz/Yap İşlemi İçin Mevcut Çözüm

J	1	2	3	4	5	
n_j	3	1	5	2	4	3
π_j	14	5	22	8	16	14

1. Adım. Kümeler içerisinden $d=2$ adet şehir seçilir.

Tablo 4-8 Boz/Yap İşleminde Şehirlerin Seçimi

J	1	**2**	3	**4**	5	
n_j	3	**1**	5	**2**	4	3
π_j	14	**5**	22	**8**	16	14

2. Adım. $\pi^D=\{14,22,16\}$, $n^D=\{3,5,4\}$, $\pi^R=\{5,8\}$, ve $n^R=\{1,2\}$ olsun.

Tablo 4-9 Boz/Yap İşleminin Bozma Safhası

J	1	2	3	*j*	1	2

$n_j{}^D$	3	5	4	3	$n_j{}^R$	1	2
$\pi^D{}_j$	14	22	16	14	$\pi^R{}_j$	5	8

3. Adım. $n^R = \{1,2\}$ kümesinden $n_2^R = 2$ tane rastgele şehir seçilir. $\pi^R = \{5,8\}$ ve $\pi^R = \{5,9\}$ karıştırılır. n_2^R kümesi içerisinden $\pi_2^R = 8$ ve $\pi_2^R = 9$ yer değiştirir.

Tablo 4-10 Boz/Yap İşleminin Bozma Safhası - Mutasyon

J	1	2	3		*j*	1	2
$n_j{}^D$	3	5	4	3	$n_j{}^R$	1	**2**
$\pi^D{}_j$	14	22	16	14	$\pi^R{}_j$	5	**9**

4. Adım. $\pi_j{}^R = 5$ Şehri $\pi_j{}^D$ içerisinde en uygun sıraya yerleştirilir.

Tablo 4-11 Boz/Yap İşleminin Yapım Safhası – İlk Adım

J	1	2	3	4		*j*	1
$n_j{}^D$	3	5	1	4	3	$n_j{}^R$	2
$\pi^D{}_j$	14	22	5	16	14	$\pi^R{}_j$	9

5. Adım. $\pi_j{}^R = 9$ şehri $\pi_j{}^{D\,\mathrm{j}}$ içerisinde en uygun sıraya yerleştirilir.

Tablo 4-12 Boz/Yap İşleminin Yapım Safhası – İkinci Adım

J	1	2	3	4	5	
$n_j{}^D$	3	**2**	5	**1**	4	3
$\pi^D{}_j$	14	**9**	22	**5**	16	14

4.4.6. *Çaprazlama*

Son yıllarda yapılan çalışmalarda Kısmen Uyumlu Çaprazlama, Sıralı Çaprazlama, Periyodik Çaprazlama, Konuma Dayalı Çaprazlama, Sezgisel Çaprazlama gibi yöntemler önerilmiş ve kullanılmıştır.

Bu operatörler Kanonik ve Sezgisel olarak ikiye ayrılabilir. Kanonik yaklaşım rasgele ve kör bir yapıya sahiptir. Yani çaprazlama sonucunda üretilen bireyin, çaprazlamaya uğrayan bireylerden daha iyi olacağını garanti etmez.

4.4.6.1. Kısmen uyumlu çaprazlama (PMX)

Kısmen uyumlu çaprazlama yöntemi (PMX) ilk olarak Goldberg ve Lingle (1985)

tarafından GSP problemi için kullanılmıştır.

Ebeveyn bireylerin kromozom dizisinden iki çaprazlama bloğu rasgele seçilir. Bu bloklar arasında kalan kısımlar yer değiştirilecek yeni iki birey üretilir.

1. Adım: Ebeveyn bireylerin çaprazlama bloklarının rastgele seçilmesi.

Tablo 4-13 Kısmen Uyumlu Çaprazlamada Ebeveyn Bireylerin Çaprazlama Bloklarının Rasgele Seçilmesi

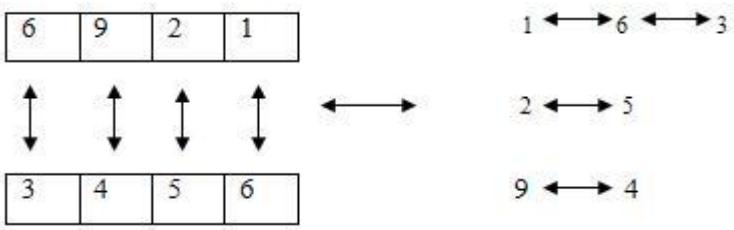
Ebeveyn Birey 1	1	2	3	4	5	6	7	8	9
Ebeveyn Birey 2	5	4	6	9	2	1	7	8	3

2. Adım: Ebeveynlerden seçilen bloklar arasında kalan kısımlar değiştirilir.

Tablo 4-14 Kısmen Uyumlu Çaprazlamada Ebeveynlerden seçilen bloklar arasında kalan kısımlar değiştirilir

Yeni Deneme Birey1	1	2	6	9	2	1	7	8	9
Yeni Deneme Birey2	5	4	3	4	5	6	7	8	3

3. Adım: Eşleşmeleri değerlendir.



Şekil 4-5 Kısmen Uyumlu Çaprazlamada Eşleşmelerin Değerlendirilmesi

4. Adım: Yeni bireyleri oluştur.

Tablo 4-15 Uyumlu Çaprazlamada Yeni Bireylerin Oluşturulması

Yeni Birey 1	3	5	6	9	2	1	7	8	4

Yeni Birey 2	2	9	3	4	5	6	7	8	1

4.4.6.2. Sıralı Çaprazlama (Ordered Crossover- OX)

Sıralı çaprazlama (OX) yöntemi ilk olarak L. Davis (1985) tarafından önerilmiş bir çaprazlama yöntemidir. OX' in PMX' in bir çeşidi olduğu söylenebilir. Ebeveynlere ait kromozomlarda belirli genler yerlerinde sabit kalırken diğer genler ikinci ebeveyn bireyde aynı sıradaki genlerle yer değiştirirler.

Tablo 4-16 Sıralı Çaprazlama

Ebeveyn Birey 1	1	2	3	4	5	6	7	8	9
Yeni Birey	7	9	3	4	5	6	1	2	8
Ebeveyn Birey 2	5	7	4	9	1	3	6	2	8

4.4.7. *Ekleme Mutasyon Operatörü*

Bu bölümde ekleme mutasyon operatörünün GGSP probleminde ki kümelere uyarlanmış hali gösterilmektedir. Ekleme mutasyon operatörü *BozYap* safhalarında kullanılmaktadır. Bir bireyin bir şehri çıkarılarak yerine aynı küme içerisinden başka bir şehir yerleştirilir. $n_2 = 5$ kümesi rastgele olarak seçilmiş olsun. Bu küme içerisindeki $\pi_2 = 23$ şehri, yine aynı kümeye dahil olan $\pi_2 = 22$ şehri ile değiştirilmesi Tablo 4.17'de gösterilmiştir.

Tablo 4-17 Ekleme Mutasyon Operatörünün İşleyişi

	j	1	**2**	3	4	5
π_i	n_j	3	**5**	2	1	4
	π_j	12	**23**	8	4	19
π	n_j	3	**5**	2	1	4
	π_j	12	**22**	8	4	19

4.4.8. *Algoritmanın Bölgesel Tarama Sezgiseli ile Birleştirilmesi*

Değiştirme (*Swap*) işleminin *bölgesel tarama* işleminin içine yerleştirilmiş hali

Synder ve Daskin ‘in (Synder, Daskin, 2006) çalışmasıyla literatüre girmiştir. KFE Algoritmasına ile bölgesel tarama sezgiselinin birleştirilmesi, üretilen her yeni deneme bireyine bölgesel tarama işlemi uygulanması ile mümkün olur. Daha sonra KFE Algoritmasında üretilen her deneme bireyine 2-opt sezgisel yöntemi uygulanır. 2-opt sezgisel yöntemi, tur içerisinden çıkarılacak iki şehir ve daha düşük maliyetli yeni bir tur oluşturmak amacıyla çıkarılan şehirlerin yerine eklenecek iki farklı şehir bulur.

Bölgesel Arama işlemindeki gibi *değiştirme* işlemi de turdan bir şehrin çıkarılıp yerine çıkarılan turun ait olduğu küme içerisinden başka bir şehrin eklenmesine dayanmaktadır. Ekleme işlemi en yakın komşuluk sezgiselinin değiştirilmiş hali kullanılarak gerçekleştirilir, böylece tura eklenen yeni şehir çıkarılandan farklı olur. Kümedeki her şehir, uygun pozisyonlara sırayla yerleştirilir ve bu yerleştirmeler sonucunda amaç fonksiyonu en düşük olan birey mevcut bireyle yer değiştirir (Tasgetiren, Suganthan, Pan, 2010).

BEŞİNCİ BÖLÜM

5. DENEY TASARIMI

5.1. Deney Tasarımı

Bilimsel bir gerçeği ortaya çıkarmak, bir varsayımı denemek ya da kanıtlamak, bir yasanın doğruluğunu göstermek amacıyla yapılan işleme deney adı verilmektedir. *(http//:www.tdk.gov.tr, 28.12.2009)* Bir diğer tanımıyla deney bir veya daha fazla sayıda belirli bir konuda sınırlandırılmış soruları yanıtlamayı hedefleyen işlem şeklidir. Deney tasarımı ise etkenlerin bağının değişken üzerindeki etkilerini araştırmak üzere rastgelelik süreçleri kullanarak yapılan denemeler olarak tanımlanabilir. *(http//:www.tdk.gov.tr, 28.12.2009)* Deney tasarımında belirlenmiş bir tasarım matrisine göre süreç üzerinde etkili olması muhtemel süreç değişkenlerinin değerleri sistematik olarak değiştirilerek, bir deney ya da bir takım sıralı deneyler gerçekleştirilir.

Deney tasarımı 1920'li yıllarda Sir Ronald Fisher'in tarım konusunda yaptığı araştırmalarla ortaya çıkmıştır. Önceleri tarım üretiminin geliştirilmesi için, daha sonraları ise kimya ve ilaç sektöründe kullanılmıştır. Box ve Wilson'un geliştirdikleri "Tepki Yüzey Metodu (TYM)" ile istatistiksel tasarım uygulamaları endüstri dalında da kullanılmaya başlamıştır. (Karakoç, 2006) 1970'li yılların sonunda Dr. Genichi Taguchi deney tasarımının endüstri uygulamalarına yönelik fikirler ortaya atmış ve başarılı uygulamalar yapmıştır (Şirvancı, 1997).

Deney tasarımının amacı, çözüm aranan problem ile ilgili mümkün olduğu kadar çok sayıda bilgiyi, zaman, para ve deneylerde kullanılacak malzemeleri en ekonomik biçimde kullanarak, problemi etkileyen en önemli değişkenleri bulmaktır.

Deney tasarımının, süreç tasarımının erken devrelerinde yapılmasının faydaları aşağıdaki gibi sayılabilir; (Montgomery, 2005)

a. İşlem veriminin arttırılması,
b. Nominal ve ya hedef etrafındaki değişkenliğin azaltılması,

c. Etkenlerin etkileşimlerinin karşılaştırılması,
d. Problemin mevcut durumunun daha iyi anlaşılması.

Deney tasarımı sonucunda beklenen çıktılar ise aşağıdaki gibidir;

a. Test edilen değişkenler içinde etkili olanların belirlenmesi,
b. Değişkenlerin çeşitli düzeylerinin etkilerinin ölçülmesi,
c. Etkenlerin etkileşimlerinin karşılaştırılması,
d. Problemin mevcut durumunun daha iyi anlaşılması.

Deney tasarımı aşamasında izlenmesi gereken işlem taslağı aşağıdaki gibi sıralanabilir;

a. Problemin tanımlanması ve özetlenmesi
b. Faktörlerin ve seviyelerin seçimi
c. Tepki değişkenlerinin seçimi
d. Deneysel tasarımın seçimi
e. Deneyi uygulama
f. Verilerin analizi
g. Sonuçlar ve öneriler.

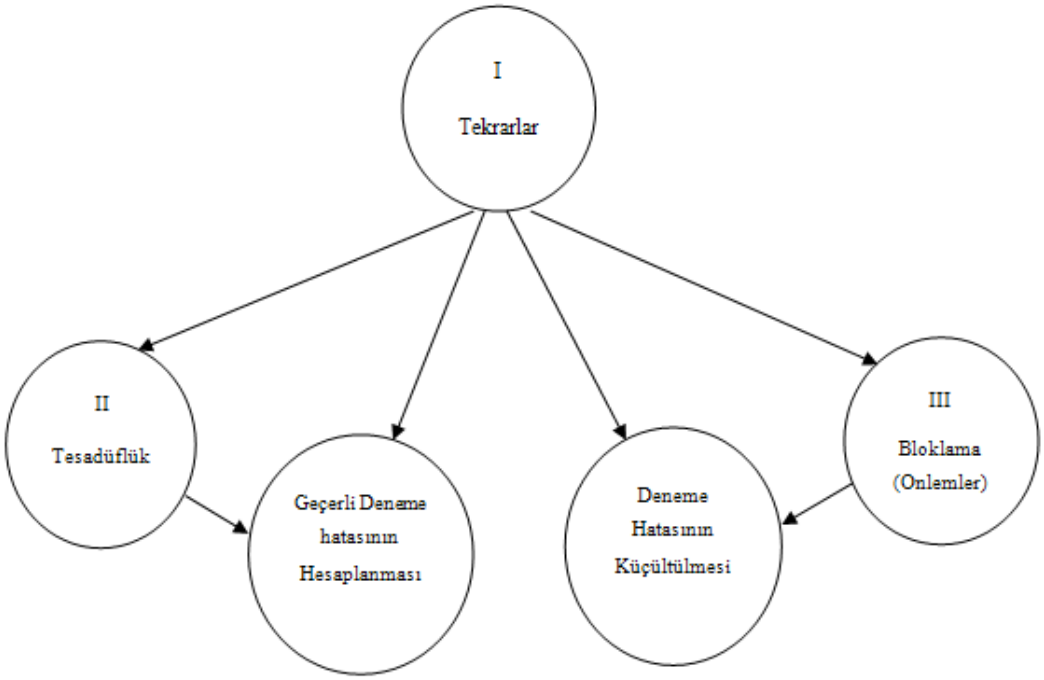
5.2. Deney Tasarımı İlkeleri

Deneysel Tasarım R.A. Fisher tarafından geliştirilen 3 temel ilkeye dayanmaktadır; tekrarlama, tesadüflük ve bloklama (önlemler). Bu üç ilke ve bu ilkeler arasındaki ilişki Şekil 5.1.'de gösterilmiştir.

Tekrarlama denildiğinde aynı değişkenin tekrar tekrar ölçülmesi değil, esas deneyin tekrar edilmesi kastedilmektedir. Tekrarlama, deney yapan kişinin deneysel hatayı tahmin etmesini sağlar. Eğer tekrarlama, deneydeki bir değişkenin etkisini tahmin etmek için kullanılıyorsa, değişkenin etkisi ile ilgili daha kesin sonuçlar sağlaması beklenir.

Tesadüflük denildiğinde, deney malzemelerinin tahsisi ve deney sırasının tesadüfi şekilde karıştırılmış olması kastedilmektedir. Tesadüflük deneysel tasarımda,

istatistiksel yöntemlerin kullanılması için oldukça önemlidir.



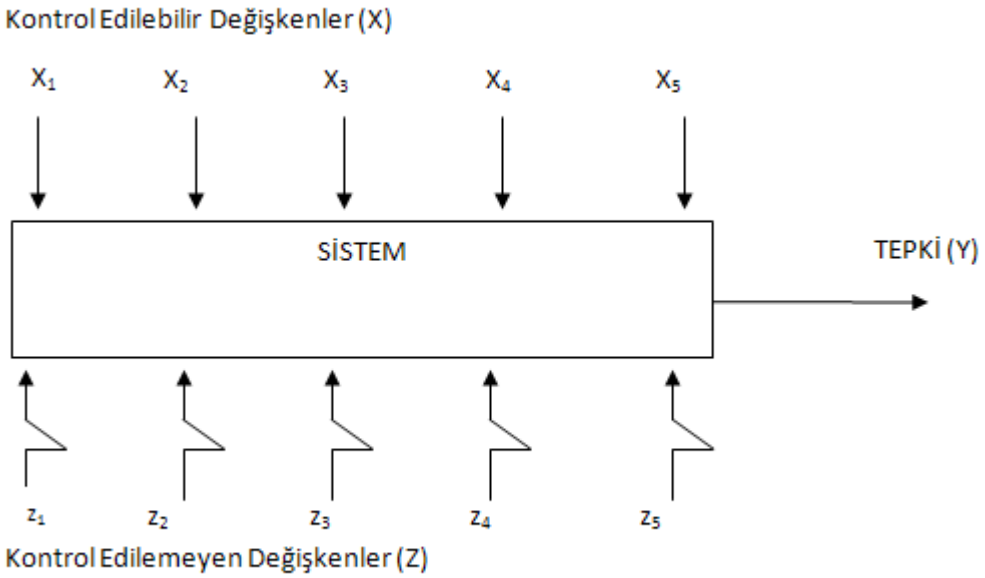
Şekil 5-1 Deney tasarımı ilkeleri arasındaki ilişki

Bloklama (önlemler), değişkenler arasındaki ilişkiyi ölçmek için kullanılan bir deneysel tasarım tekniğidir. Bloklama, deneyi yapan kişinin isteği dışında, deneye etki edebilecek olan değişkenlerin deneye etkisini azaltmak ya da tamamen engellemek için kullanılmaktadır. Bloklama daha homojen gruplarla deneylerin yapılmasını ve bu sayede deneysel hataların azalmasını sağlamaktadır (Karakoç, 2006).

5.3. Deney Tasarımı Stratejileri

Deney tasarımı genellikle sürece etki eden kontrol edilebilir ve kontrol edilemeyen değişkenlerin sisteme etkisini ölçmek amacıyla yapılır. Süreçlerin bazı girdileri, $x_1,x_2,..,x_p,$ kontrol edilebilir, bazı girdileri ise, $z_1,z_2,\ldots,z_q,$ kontrol edilemez değişkenler olarak kabul edilir. Bir sistemin genel modeli Şekil 5.2.' deki gibidir.

Bir faktörün seviyesinin değiştirilmesiyle, tepki (*y*) değeri değişir. Bu değişime faktörün etkisi adı verilir. Buna "ana etki" de denilebilir. 2 faktöre sahip (*A, B*), her faktörün iki seviyesinin bulunduğu bir deneyden bahsediyorsak, *A* ve *B*' nin seviyeleri "-" ve "+" olarak gösterilir. "-1", "+1" olarak ta karşımıza çıkmaktadır. Bu iki seviyeye faktörlerin "*yüksek*" ve "*düşük*" seviyeleri denir.



Şekil 5-2 Bir sistemin genel modeli

*Montgomery, 2005

Deneylerin planlanması ve yürütülmesindeki genel yaklaşıma "*Deney Stratejisi*" denir. Deney yapan kişiler üzerinde çalıştıkları sistem hakkında teorik ve teknik bilgiye sahiplerse "*eniyi tahmin*" stratejisini kullanabilirler. Bu yöntemde deney yapan kişiler sistem üzerinde etkili olabileceğini düşündükleri değişkenlerin keyfi bir birleşimini seçer ve test ederler. Bu yöntemin iki dezavantajı bulunmaktadır; Başlangıçta denenen tasarımların iyi sonuçlar vermemesi sonucunda iyi sonuçları elde edene kadar çok sayıda deney yapmak gerekebilir ya da deney yapan kişi başlangıçta iyi sonuç bularak deneyi durdurur böylece daha iyi bir sonuca ulaşma şansını kaybeder. (Karakoç, 2006)

Deney tasarımında kullanılabilecek stratejilerden bir tanesi de "*bir defada bir*

değişken" stratejisidir. Diğer tüm değişkenler başlangıç seviyesindeysen, seçilen bir adet değişkenin değerlerinin değiştirilmesiyle her bir değişkenin sistemin performansına etkisi ayrı ayrı ölçülebilir. Ancak bu yöntemde değişkenlerin etkileşimleri incelenemez.

Deney tasarımında kullanılan izlenen bir diğer strateji ise "*Tam Faktöriyel Deney Stratejisi*"dir. Faktörlerin her birini tek tek ele almak yerine birlikte alınması ve birbirleriyle etkileşimlerini de göz önüne almasından dolayı bu yaklaşım daha geçerli sonuçlar vermektedir. Tam faktöriyel deneylerin, sistemin performansına etki eden faktör sayısının sınırlı olduğu durumlarda kullanılması uygundur. Sisteme etki eden faktörlerin sayısı çoğaldıkça deneylerin sayısı ve deneyler için harcanan toplam süre artar. Genellikler sisteme etki eden faktör sayısı 1-5 arasında olan deneylerde tam faktöriyel stratejisi kullanılmaktadır. Tam faktöriyel deney tasarımlarında gerekli deney sayısı a^n formülü ile bulunabilir.(a: faktörlerin seviye sayısı, n: ilgilenilen faktör sayısı) Örneğin 5 faktörlü 2 seviyeli bir sistemde tam faktöriyel deney tasarımı uygulanmak istendiğinde $2^5 = 32$ adet deney olası bütün kombinasyonları göstermektedir.

Sisteme etki eden faktör sayısı 5'ten fazlaysa ana faktörlerin yanında bazı faktörler arasında ki etkileşimleri de görmemize yardımcı olan "*Kesirli Faktöriyel Deneyler* (*KFD*)" (Fractional Factorial Experiment) stratejisi kullanılabilir. Kesirli faktöriyel deney tasarımlar, ilgilenilen sistem çok faktörlü ya da faktörler çok seviyeli olduğunda maliyetten ve süreden tasarruf edilmesini sağlar. Kesirli faktöriyel deneylerde yüksek serbestlik derecesine sahip, deney sonuçlarına az katkı sağlayacağı tespit edilen etkileşimleri deney tasarımına koymamaktır. Kesirli faktöriyel deneylerde deney sayısı $a^{n-p}(p < n)$ formülü ile bulunabilir. (a: faktörlerin seviye sayısı, n: ilgilenilen faktör sayısı, p: bloklarla karışacak bağımsız etkileşim sayısı).

n faktörlü bir tasarımda her bir faktör 2 seviyeli ise bu tasarım 2^n tasarımı olarak adlandırılır. 2^{n-p} deneme içeren bir tasarıma ise 2^n tasarımının $1/2\,p$ kesri denir. $p = 1$ olduğu durumlarda deney sayısı 2^{n-1}olur yani bu 2^n tasarımının $1/2$ kesri

olarak adlandırılır. $p = 2$ olduğu durumda ise deney sayısı 2^{n-2} yani 2^n tasarımının $(1/2)^2 = 1/4$ kesri olarak adlandırılır. Bu surumda 7 faktörlü, 2 seviyeli bir deney tasarımında, tam faktöriyel tasarım uygulandığında yapılması gereken $2^7 = 128$ deney varken, 2^7 tasarımının $1/4$ kesirli tasarımı kullanılarak $2^{7-2} = 2^5 = 32$ deney yapılması uygun olacaktır.

5.4. 2^{6-2} Kısmi Faktöriyel Deneyler

2^{6-2} Kısmi faktöriyel deneyler literatürde en az sapma ölçüte göre sıralanmış tasarımlardan bir tanesidir. 2^4 tasarımı gibi *A, B, C* ve *D* faktörleri yazılır daha sonra *E* ve *F* eklenir. *E* ve *F* sütunlarını oluşturmak için sütunlarla karışacak etki ya da etkileşimleri gösteren tanımlayıcı bağıntı (birim sütun) olarak adlandırılan "*I*" kullanılır. Bu tip tasarımlarda $I = ABCE$ ve $I = BCDF$ tanımlayıcı bağıntılarından yola çıkılarak $E = ABC$ ve $F = BCD$ olarak bulunur. $I = ABCE = BCDF = ADEF$ tam tanımlayıcı bağıntısı ortaya çıkar. 2^k tasarımlarında iki sütunun çarpılmasıyla yeni bir sütun oluşturulur. Yani $ABCE$ ve $BCDF$ sütunlarını çarptığımızda birim sütuna eşit olan yeni bir sütun elde ederiz.

$$ABCE(BCDF) = AB^2C^2DEF = ADEF$$

Yani 2^{6-2} tasarımının tam bağıntısı $I = ABCE = BCDF = ADEF$ (5.1.), olur. Bu bağıntıyı kullanarak herhangi bir etkenin eşleniğini bulmak mümkündür. Örneğin *A*' nın eşleniği aşağıdaki şekilde bulunur.

$$I(A) = A(ABCE) = A(BCDF) = A(ADEF)$$

$$A = A^2BCE = ABCDF = A^2DEF$$

$$A = BCE = ABCDF = DEF \qquad (5.2.)$$

$I = ABCE = BCDF = ADEF$ tam tanımlayıcı bağıntısına sahip 2_{IV}^{6-2} tasarımının eşlenik yapısı Tablo 5.1.' de gösterilmiştir.

Tablo 5-1 $I = ABCE = BCDF = ADEF$ **Tam Tanımlayıcı Bağıntısına Sahip** 2_{IV}^{6-2} **Tasarımının Eşlenik Yapısı**

A = BCE = DEF = ABCDF	*AB = CE = ACDF = BDEF*
B = ACE = CDF = ABDEF	*AC = BE = ABDF = CDEF*
C = ABE = BDF = ACDEF	*AD = EF = ABCF = BCDE*
D = AEF = BCF = ABCDE	*AE = BC = DF = ABCDEF*
E = ABC = ADF = BCDEF	*AF = DE = ABCD = BCEF*
F = ADE = BCD = ABCEF	*BD = CF = ABEF = ACDE*
ABD = ACF = BEF = CDE	*BF = CD = ABDE = ACEF*
ABF = ACD = BDE = CEF	

Bu tasarımda ki tam tanımlayıcı bağıntıya bakıldığında tanımlayıcı bağıntıların her birinin 4 harfe sahip olduğu görülmektedir. Tam tanımlayıcı bağıntıda bulunan yapılardan en az harfe sahip olan bağıntının harf sayısı çözünürlük tipini belirler. Bu deney tasarım IV çözünürlüğüne sahiptir. Tasarım IV çözünürlüğüne sahip deneylerde ana etkiler ve ikili faktör etkileşimleri hakkında bilgi verir (Montgomery, 2005).

5.5. Parametrelerin deney tasarımı ile belirlenmesi

Bu bölümde KSE Algoritması için kullanılan parametrelerin seviyelerinin deneysel tasarım ile belirlenmesi anlatılmıştır. KSE Algoritmasında altı faktör (parametre) bulunmaktadır. Bunlar, popülasyon büyüklüğü A (NP), Bozma seviyesi B (*d*), perturbasyon seviyesi C (*p*), çaprazlama tipi D (*CT*), çaprazlama olasılığı E (*Pc*) ve mutasyon olasılığıdır F (*Pm*). Her bir faktör 2 iki seviyeye sahiptir. Düşük seviye ve

yüksek seviye faktörler ve seviyeleri Tablo 5. 3 de gösterilmektedir.

Tablo 5-2 Faktör seviyeleri

	A(NP)	*B(d)*	*C(p)*	*D(CT)*	*E(CR)*	*F(PM)*
Düşük (-1)	30	5	1	PMX	0,2	0,2
Yüksek(1)	100	*ncluster**0,4	3	OX	0,9	0,9

Parametrelerin belirlenmesi için Tablo 5.4 de gösterilen 2_{IV}^{6-2} deney tasarımı kullanılmıştır. Tablo 5.2' de ki veriler Minitab 14 kullanılarak oluşturulmuştur. 2_{IV}^{6-2} tasarımında $2^{6-2} = 2^4 = 16$ adet deney düzenlenmesi gerekmektedir.

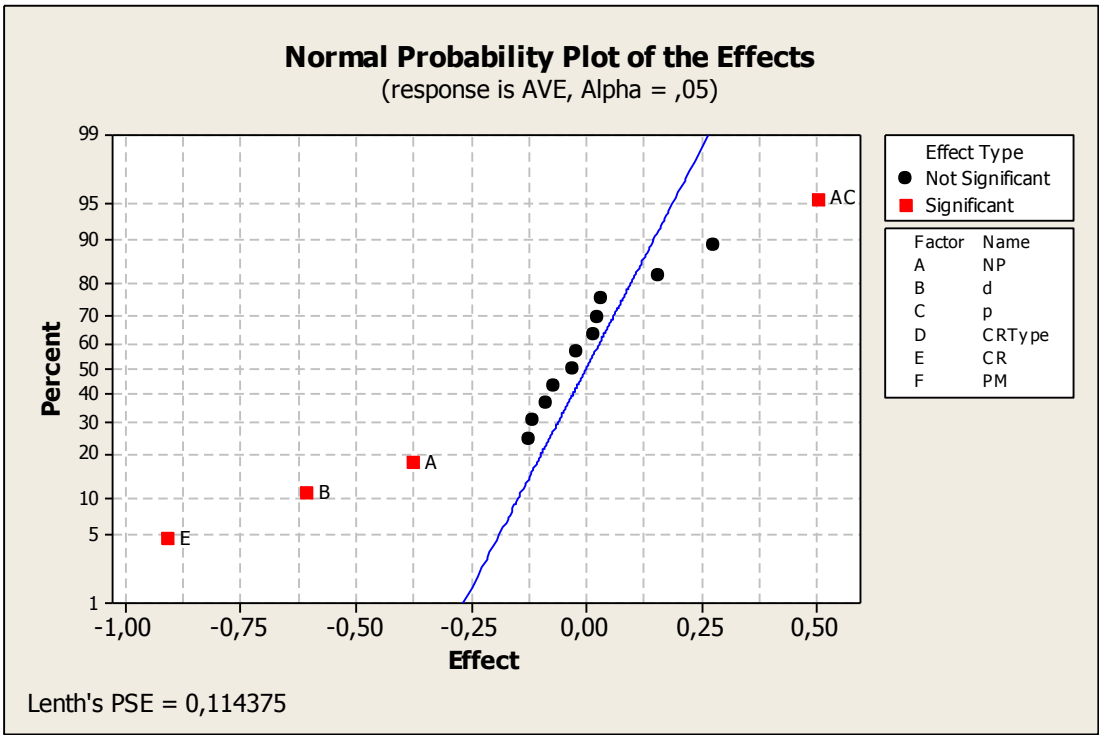
Tablo 5-3 2_{IV}^{6-2} **Deney Tasarımı**

Deney	A	B	C	D	E	F
1	-1	-1	-1	-1	-1	-1
2	1	-1	-1	-1	1	-1
3	-1	1	-1	-1	1	1
4	1	1	-1	-1	-1	1
5	-1	-1	1	-1	1	1
6	1	-1	1	-1	-1	1
7	-1	1	1	-1	-1	-1
8	1	1	1	-1	1	-1
9	-1	-1	-1	1	-1	1
10	1	-1	-1	1	1	1
11	-1	1	-1	1	1	-1
12	1	1	-1	1	-1	-1
13	-1	-1	1	1	1	-1

14	1	-1	1	1	-1	-1
15	-1	1	1	1	-1	1
16	1	1	1	1	1	1

54 problem içerisinden en fazla şehir ve küme sayılarına sahip 12 problem seçilerek (107att532, 113pa561, 115rat575, 131p654, 132d657, 145u724, 157rat783, 201pr1002, 207si1032, 212u1060, 217vm1084), her bir deney için KFE algoritması çalıştırılmıştır ve durdurma kriteri olarak 10 nesilde popülasyondaki eniyi çözüm iyileştirilemediği takdirde KFE algoritması durdurulmuştur. Yanıt değişkeni (response variable) olarak eniyi veya literatürde bulunan yakın eniyi değerlerden sapmaların ortalaması alınmıştır.

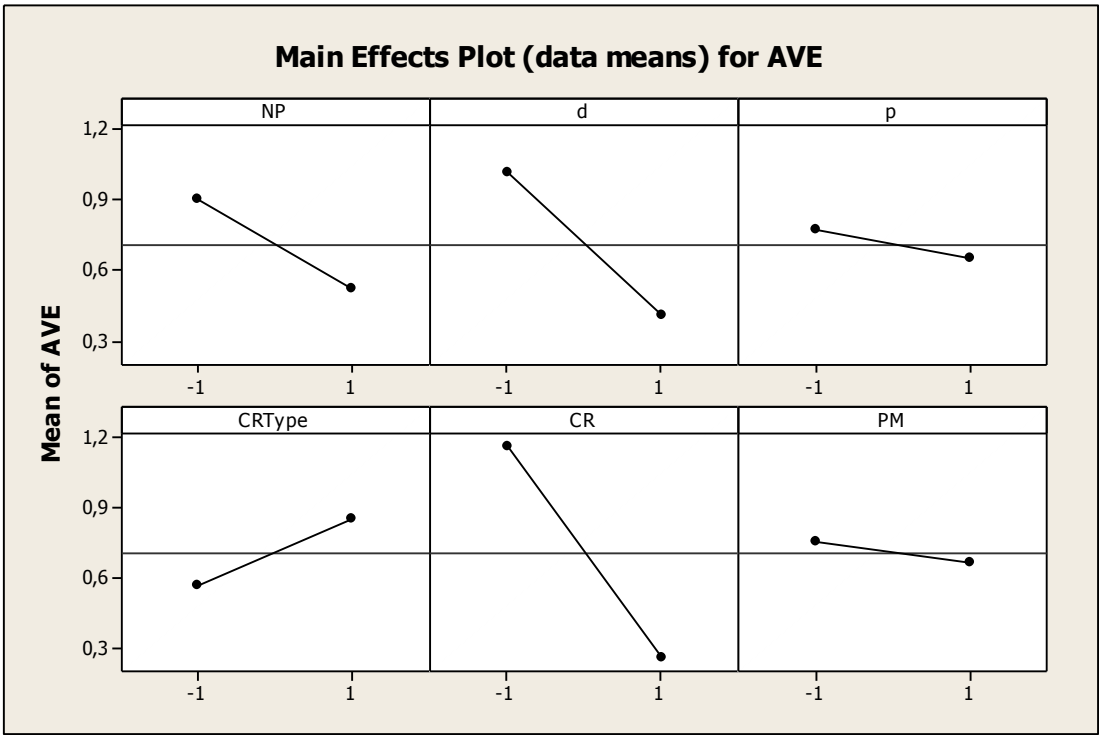
Her bir deney için yanıt değişkeninin değerleri hesaplandıktan sonra, aşağıdaki istatistiksel analiz yapılarak her bir faktörün seviyeleri belirlenmiştir. Etkilerin normal olasılık grafiğinde, yanıt değişkeni üzerinde belirgin etkisi olan parametreler ve parametrelerin etkileşimleri gözükmektedir.



Şekil 5.3 KFE Algoritması için etkilerin normal olasılık grafiği

Şekil 5.3'te en belirgin parametrelerin A, B, E olduğu ve en belirgin etkileşimin AC

etkileşimi olduğu gözükmektedir. Başka bir deyişle, popülasyon büyüklüğü, bozma seviyesi ve çaprazlama olasılığı nın sonuçlar üzerinde belirgin etkisi olduğu gözükmektedir. Ayrıca AC etkileşimi, yani popülasyon büyüklüğü/perturbasyon seviyesinin etkileşimi, sonuçlar üzerinde belirgin etkiye sahip olduğu gözükmektedir.



Şekil 5-4 KFE Algoritmasının ana etki grafiği

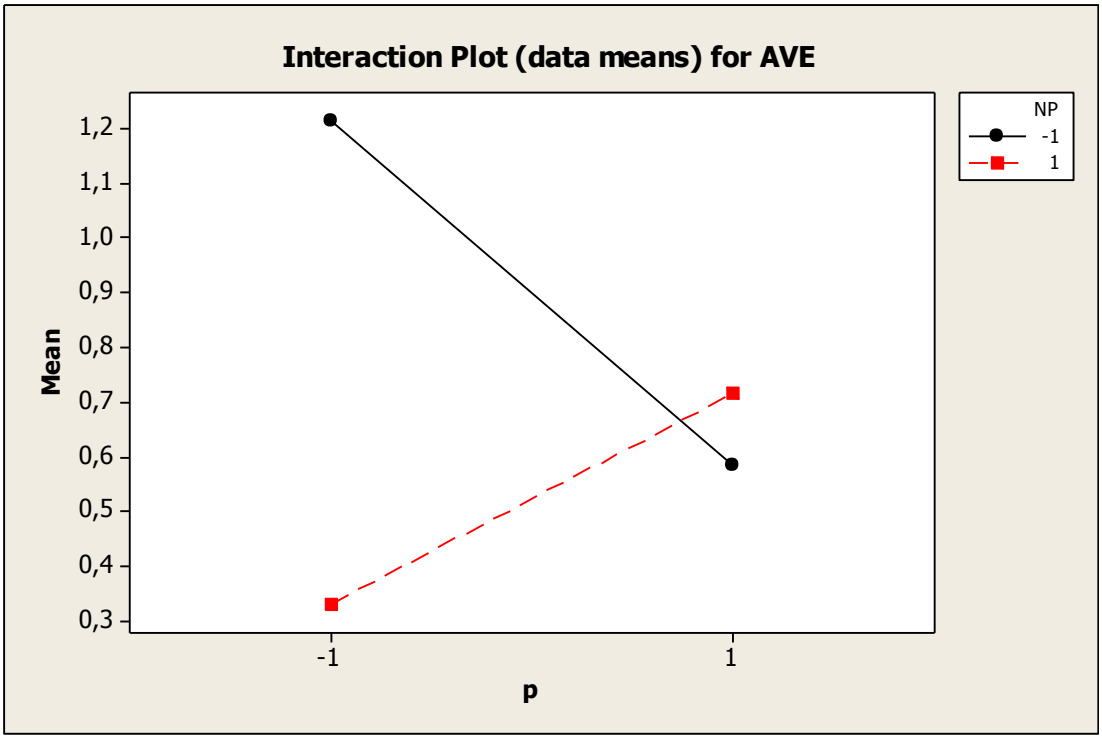
Faktörlerini seviyelerini belirlemek için Şekil 5.4' te verilen Ana Etki Grafikleri kullanılmıştır. Eğer sadece Ana Etki grafiği göz önüne alınsaydı, Tablo 5.4.'teki parametrelere karar verilebilirdi.

Tablo 5-4 Ana Etki Grafiğine bakılarak parametre seviyelerinin belirlenmesi

Faktör	Seviye	Açıklama	Değer
A	1	Popülasyon büyüklüğü	NP=100
B	1	Bozma seviyesi	$d = ncluster * 0{,}4$
C	1	Perturbasyon seviyesi	$p = 3$
D	-1	Çaprazlama tipi	CRtype=OX

E	1	Çaprazlama olasılığı	Pc=0.9
F	1	Mutasyon olasılığı	Pm=0.9

Bununla beraber, Ana Etki Grafikleri, eğer faktörler arasında herhangi bir etkileşim söz konusu olduğunda, çok fazla anlam ifade etmeyebilir. Dolayısıyla, AC etkileşim grafiğine bakmak gerekir. Tablo 5.4.'teki parametre seviyeleri (1,1,1,-1,1,1) ile algoritma çalıştırıldığında standart sapması 0,03, CPU zamanı 16,69 çıkmıştır.



Şekil 5-5 KFE Algoritmasının etkileşim grafiği

Daha önce ifade edildiği gibi, AC etkileşimi (yani NP/p) en belirgin etkileşimdi. NP/p etkileşim grafiği Şekil 5.5.' te gösterilmektedir. Şekil de görüldüğü gibi, NP değeri yüksek (1) olduğunda p değerinin düşük (-1) olduğu açıkça görülmektedir. Daha doğrusu, NP değeri 100 olmalı, ancak p değeri 1 alınmalıdır. Bu, Ana Etki Grafiğinde verilen değerden farklıdır. Tablo 5.5.'teki parametre seviyeleri (1,1,-1,-1,1,1) ile algoritma çalıştırıldığında standart sapması -0,02, CPU zamanı 13,57 çıkmıştır.

Bu analizlerden sonra Tablo 5.5.'te verilen parametre değerlerinin KFE

algoritmasında kullanılmasına karar verilmiştir.

Tablo 5-5 Etkileşim Grafiğine Bakılarak Parametrelerin Değerlendirilmesi

Faktör	Seviye	Açıklama	Değer
A	1	Popülasyon büyüklüğü	NP=100
B	1	Bozma seviyesi	$d = ncluster * 0{,}4$
C	-1	Perturbasyon seviyesi	$p = 1$
D	-1	Çaprazlama tipi	CRtype=OX
E	1	Çaprazlama olasılığı	Pc=0.9
F	1	Mutasyon olasılığı	Pm=0.9

ALTINCI BÖLÜM

6. DENEYSEL SONUÇLAR

Fischetti ve diğerleri (1997) simetrik GGSP problemlerini çözmek için dal ve sınır algoritması geliştirmiştir. TSPLIB kütüphanesinden Standard GSP problemini alarak ve parçalara ayırma yöntemi ile GGSP kıyaslama problemleri oluşturmuştur. Bu GGSP kıyaslama problemleri, http://josilber.student.umd.edu/gtspdatasets/gtspdatasets internet adresinden bulunabilir. Bu kıyaslama seti, 65 problemden oluşmaktadır. Bu konuda çalışma yapanlar 54 problemi seçtiği için aynı 54 problem bu tezde kullanılmıştır. 10 ile 89 kümeli problemler için eniyi çözümler, Fischetti ve diğerlerinde (1997) verilmiştir. 99 ile 217 kümeli geri kalan problemler için Silberholz ve Golden (1997), yakın-eniyi çözümleri sunmuşlardır. Ancak bu çözümler, beş tekrarlamanın ortalama değeridir. Bunun nedeni, enaz değerin Silberholz ve Golden'ın (1997) makalesinde verilmemiş olmasıdır.

KFE algoritması Visual C++ da programlanmıştır ve Intel P IV 3.00 GHz 512 MB belleği olan makinede çalıştırılmıştır. KFE algoritmasının parametreleri deney tasarımı bölümünde belirlenmiştir. Başlangıç popülasyonu rassal olarak oluşturulmuştur ve daha sonra popülasyondaki her bireye NEH sezgiseli uygulanmıştır. Literatürdeki eniyi algoritmalarla adil bir kıyaslama yapabilmek için, durdurma koşulu olarak şu alınmıştır. Eğer popülasyondaki eniyi çözüm 10 defa iyileşmediği takdirde algoritma durdurulmuştur. Ayrıca yine adil bir kıyaslama yapmak için, 5 tekrarlama yapılarak eniyi veya yakın-eniyi çözümlerden nispi yüzde sapmalar istatistik olarak aşağıdaki gibi hesaplanmıştır:

$$\Delta_{avg} = \sum_{i=1}^{R} \left(\frac{(H_i - B) \times 100}{B} \right) / R \qquad (6.1.)$$

Burada H_i, B ve R, sırasıyla KFE algoritması tarafından üretilen çözüm değeri, eniyi veya yakın-eniyi değer ve tekrarlama sayısıdır. CPU zamanı karşılaştırmaları için t_{avg}, ortalama CPU zamanını göstermektedir.

6.1. Bölgesel taramalı KFE algoritmasının performansı

Bölgesel tarama ile birleştirilmiş KFE Algoritması'nın sonuçları Tablo 6.1. 'da gösterilmiştir. Tablo 6.1'te görüldüğü gibi eniyi çözümleri mevcut olan 41 problem için KFE Algoritması 5 tekrarlamanın sonuçlarıdır.

Test edilen 41 problemin 38'inde 5 tekrarlamanın en az 4'ünde eniyi sonuca ulaşılmıştır. 41 problemin 34'ünde (%82.92) KFE Algoritması beş denemenin beşinde eniyi sonuca ulaşmıştır. KFE Algoritmasının çözdüğü 41 problemden 38'i (%92.68) eniyi çözümün en fazla %0,05 üzerine çıkmıştır. KFE algoritmasının değişkenliğinin bir ölçüsü olarak aralık değeri 0.09-0.02=0.07'dir. Dolayısıyla algoritma tutarlı sonuçlar da vermektedir.

Tablo Error! Use the Home tab to apply Başlık 2 to the text that you want to appear here.**-1 Bölgesel tarama ile birleştirilmiş KFE algoritmasının performansı**

Problem	Eniyi	n_{opt}	Δ_{avg}	$\Delta_{\min}$	$\Delta_{\max}$	KFE	t_{avg}
10att48	5394	5	0,00	0,00	0,00	5394	0,02
10gr48	1834	5	0,00	0,00	0,00	1834	0,02
10hk48	6386	5	0,00	0,00	0,00	6386	0,07
11eil51	174	5	0,00	0,00	0,00	174	0,02
12brazil58	15332	5	0,00	0,00	0,00	15332	0,02
14st70	316	5	0,00	0,00	0,00	316	0,03
16eil76	209	5	0,00	0,00	0,00	209	0,03
16pr76	64925	5	0,00	0,00	0,00	64925	0,03
20kroa100	9711	5	0,00	0,00	0,00	9711	0,09
20krob100	10328	5	0,00	0,00	0,00	10328	0,04
20kroc100	9554	5	0,00	0,00	0,00	9554	0,04
20krod100	9450	5	0,00	0,00	0,00	9450	0,04
20kroe100	9523	5	0,00	0,00	0,00	9523	0,04
20rat99	497	5	0,00	0,00	0,00	497	0,04
20rd100	3650	5	0,00	0,00	0,00	3650	0,04
21eil101	249	5	0,00	0,00	0,00	249	0,04
21lin105	8213	5	0,00	0,00	0,00	8213	0,04
22pr107	27898	5	0,00	0,00	0,00	27898	0,05
24gr120	2769	5	0,00	0,00	0,00	2769	0,06
25pr124	36605	5	0,00	0,00	0,00	36605	0,06
26bier127	72418	5	0,00	0,00	0,00	72418	0,06
28pr136	42570	5	0,00	0,00	0,00	42570	0,08
29pr144	45886	5	0,00	0,00	0,00	45886	0,08
30kroa150	11018	5	0,00	0,00	0,00	11018	0,09
30krob150	12196	5	0,00	0,00	0,00	12196	0,09
31pr152	51576	5	0,00	0,00	0,00	51576	0,10
32u159	22664	5	0,00	0,00	0,00	22664	0,11
39rat195	854	4	0,02	0,00	0,12	854	0,20
40d198	10557	5	0,00	0,00	0,00	10557	0,19
40kroa200	13406	5	0,00	0,00	0,00	13406	0,23

40krob200	13111	5	0,00	0,00	0,00	13111	0,31
45ts225	68340	5	0,00	0,00	0,00	68340	0,60
46pr226	64007	5	0,00	0,00	0,00	64007	0,20
53gil262	1013	4	0,14	0,00	0,69	1013	0,65
53pr264	29549	5	0,00	0,00	0,00	29549	0,42
60pr299	22615	4	0,01	0,00	0,04	22615	1,05
64lin318	20765	5	0,00	0,00	0,00	20765	1,40
80rd400	6361	4	0,15	0,00	0,74	6361	4,33
84fl417	9651	1	0,01	0,00	0,03	9651	4,26
88pr439	60099	3	0,05	0,00	0,22	60099	5,47
89pcb442	21657	2	0,40	0,00	1,92	21657	7,30
ORTALAMA		4,68	0,02	0,00	0,09	20081	0,68

6.2. KFE nin Diğer Algoritmalarla Kıyaslanması

Tablo 6.2. ‘de bölgesel tarama ile birleştirilmiş KFE Algoritmasının GTSPLIB’ de bulunan tüm problemler için sonuçları gösterilmiştir. KFE algoritması, bu bölümde Bontoux, Artigues ve Feillet (2009)’un Memetik Algoritması ve Taşgetiren, Suganthan ve Pan (2009)’un eDDE algoritması ile kıyaslanmıştır. Tablo 6.2 de görüldüğü gibi, KFE algoritması, bütün sonuçların ortalaması alındığında ve çok küçük bir fark olmasına rağmen eDDE algoritmasından biraz daha iyi ve MA algoritmasından biraz daha kötü sonuç vermektedir. Yakın-çözümü bilinen 13 büyük kümeli problemin 10 tanesinin sonuçları daha iyileştirilmiştir. Kısaca üç algoritmada yakın-eniyi çözümleri daha da iyileştirmiştir. Dolayısıyla KFE algoritması, en az MA ve eDDE algoritmaları kadar iyi bir algoritmadır. Ayrıca, aradaki farkların istatistiksel olarak anlamlı olup olmadığını test etmek için 0.95 anlam düzeyinde çift taraflı eşlenik t-testi yapılmıştır. KFE algoritmasının, MA ve eDDE ile t-testinin p-değerleri, sırasıyla 0,777 ve 0,117 değerler üretmektedir ve bunun anlamı testler bu değerlerde anlamlıdır. Her iki değer de $\alpha = 0.05$ den büyük olduğu için test ana hipotez (H_o) kabul edilmektedir. Dolayısıyla karşılaştırılan üç algoritma da birbirine eşdeğerdir.

Tablo Error! Use the Home tab to apply Başlık 2 to the text that you want to appear here.**-2 Problemlerin optimal sonuçlarının diğer algoritmalarla kıyaslanması**

		KFE		MA		eDDE	
Problemler	**Eniyi**	Δ_{avg}	t_{avg}	Δ_{avg}	t_{avg}	Δ_{avg}	t_{avg}
10att48	5394	0,00	0,02	0,00	0,76	0,00	0.09
10gr48	1834	0,00	0,02	0,00	0,79	0,00	0.08
10hk48	6386	0,00	0,07	0,00	0,50	0,00	0.07

11eil51	174	0,00	0,02	0,00	0,81	0,00	0.08
12brazil58	15332	0,00	0,02	0,00	0,65	0,00	0.08
14st70	316	0,00	0,03	0,00	0,93	0,00	0.11
16eil76	209	0,00	0,03	0,00	1,00	0,00	0.11
16pr76	64925	0,00	0,03	0,00	1,17	0,00	0.13
20kroa100	9711	0,00	0,09	0,00	1,81	0,00	0.17
20krob100	10328	0,00	0,04	0,00	2,17	0,00	0.18
20kroc100	9554	0,00	0,04	0,00	1,85	0,00	0.18
20krod100	9450	0,00	0,04	0,00	2,77	0,00	0.18
20kroe100	9523	0,00	0,04	0,00	1,81	0,00	0.19
20rat99	497	0,00	0,04	0,00	3,89	0,00	0.19
20rd100	3650	0,00	0,04	0,00	2,91	0,00	0.19
21eil101	249	0,00	0,04	0,00	2,09	0,00	0.17
21lin105	8213	0,00	0,04	0,00	3,18	0,00	0.21
22pr107	27898	0,00	0,05	0,00	4,78	0,00	0.21
24gr120	2769	0,00	0,06	0,00	2,34	0,00	0.24
25pr124	36605	0,00	0,06	0,00	2,84	0,00	0.29
26bier127	72418	0,00	0,06	0,00	3,35	0,00	0.29
28pr136	42570	0,00	0,08	0,00	4,23	0,00	0.37
29pr144	45886	0,00	0,08	0,00	5,42	0,00	0.38
30kroa150	11018	0,00	0,09	0,00	5,95	0,00	0.38
30krob150	12196	0,00	0,09	0,00	5,02	0,00	0.40
31pr152	51576	0,00	0,10	0,00	5,24	0,00	0.46
32u159	22664	0,00	0,11	0,00	5,58	0,00	0.45
39rat195	854	0,02	0,20	0,00	11,01	0,00	0.74
40d198	10557	0,00	0,19	0,00	10,15	0,00	0.82
40kroa200	13406	0,00	0,23	0,00	10,41	0,00	0.81
40krob200	13111	0,00	0,31	0,00	10,81	0.02	0.95
45ts225	68340	0,00	0,60	0,04	31,45	0.04	1,31
46pr226	64007	0,00	0,20	0,00	8,25	0,00	1,02
53gil262	1013	0,14	0,65	0,14	24,34	0.14	2,03
53pr264	29549	0,00	0,42	0,00	18,27	0,00	1,61
60pr299	22615	0,01	1,05	0,00	21,25	0.04	3,30
64lin318	20765	0,00	1,40	0,00	26,33	0,00	3,79
80rd400	6361	0,15	4,33	0,42	32,21	0,00	10,57
84fl417	9651	0,01	4,26	0,00	31,63	0.01	7,98
88pr439	60099	0,05	5,47	0,00	42,55	0.09	10,65
89pcb442	21657	0,40	7,30	0,19	62,53	0.13	13,65
99d493	20117	-0,01	7,96	-0,03	166,11	-0.04	17,90
107att532	13510	-0,32	11,14	-0,30	137,54	-0.23	25,40
107si535	13513	0,08	6,18	-0,01	90,98	-0.05	16,41
113pa561	1051	-0,67	7,91	-0,84	149,43	-0.82	20,39
115rat575	2414	-0,23	23,31	0,04	157,01	-0.75	29,04

131p654	27439	-0,03	14,70	-0,03	144,95	-0.03	29,28
132d657	22599	-0,16	27,38	-0,15	259,11	-0.32	65,50
145u724	17370	0,39	50,95	0,45	218,66	-0.07	77,64
157rat783	3300	-0,56	51,13	-0,07	391,79	-0.39	96,55
201pr1002	114582	-0,12	113,71	0,03	513,48	0.01	228,35
207si1032	22388	-0,08	74,38	-0,26	616,28	-0.12	141,81
212u1060	108390	-0,42	150,19	-0,92	762,86	-0.18	291,01
217vm1084	131884	0,06	165,60	-0,26	583,44	-0.20	328,52
ORTALAMA		-0,02	13,57	-0,03	85,31	0,00	61,90

6.3. KFE nin Diğer Algoritmalarla Eniyi Çözümler için Kıyaslanması

Tablo 6.3. 'te bölgesel tarama ile birleştirilmiş KFE Algoritmasının GTSPLIB'de bulunan 41 problem için sonuçları gösterilmiştir. Bu 41 problemin eniyi değerleri bilinmektedir. KFE algoritması, bu bölümde Bontoux, Artigues ve Feillet'in (2009) Memetik Algoritması ve Taşgetiren, Suganthan ve Pan'ın (2009) eDDE algoritması, Synder ve Daskin'in (2006) RKGA ve Silberholz ve Golden'ın (1997), mrOXGA ile kıyaslanmıştır. Tablo 6.3' te görüldüğü gibi, KFE algoritması, bütün sonuçların ortalaması alındığında çok küçük bir fark olmasına rağmen mrOXGA' den biraz daha iyi, RKGA' dan biraz daha kötü, eDDE algoritmasından biraz daha kötü ve MA algoritması ile aynı sonuç vermektedir. Dolayısıyla KFE algoritması, en az RKGA, mrOXGA, MA ve eDDE algoritmaları kadar iyi bir algoritmadır. Ayrıca, tüm 41 problem için aradaki farkların istatistiksel olarak anlamlı olup olmadığını test etmek için %95 anlam düzeyinde çift taraflı eşlenik t-testi uygulanmıştır. KFE algoritmasının, mrOXGA, RKGA, MA ve eDDE ile p-değerleri, sırasıyla 0,433, 0,007, 0,978 ve 0,334 değerler üretmektedir. Bunun anlamı, KFE algoritması mrOXGA, MA ve eDDE algoritmasına eşdeğerdir ancak RKGA den daha iyi sonuçlar üretmektedir.

Tablo Error! Use the Home tab to apply Başlık 2 to the text that you want to appear here.**-3 KFE nin Diğer Algoritmalarla Eniyi Çözümler için Kıyaslanması**

	KFE		*mrOXGA*		*RKGA*		MA		eDDE	
Instance	Δ_{avg}	t_{avg}	Δ_{avg}	t_{avg}	Δ_{avg}	t_{avg}	Δ_{avg}	t_{avg}	Δ_{avg}	t_{avg}
10att48	0,00	0,02	0,00	0,36	0,00	0,18	0,00	0,76	0,00	0,09
10gr48	0,00	0,02	0,00	0,32	0,00	0,08	0,00	0,79	0,00	0,08
10hk48	0,00	0,07	0,00	0,31	0,00	0,08	0,00	0,50	0,00	0,07

11eil51	0,00	0,02	0,00	0,26	0,00	0,08	0,00	0,81	0,00	0,08
12brazil58	0,00	0,02	0,00	0,78	0,00	0,10	0,00	0,65	0,00	0,08
14st70	0,00	0,03	0,00	0,35	0,00	0,07	0,00	0,93	0,00	0,11
16eil76	0,00	0,03	0,00	0,37	0,00	0,11	0,00	1,00	0,00	0,11
16pr76	0,00	0,03	0,00	0,45	0,00	0,16	0,00	1,17	0,00	0,13
20kroa100	0,00	0,09	0,00	0,50	0,00	0,24	0,00	1,81	0,00	0,17
20krob100	0,00	0,04	0,00	0,63	0,00	0,25	0,00	2,17	0,00	0,18
20kroc100	0,00	0,04	0,00	0,60	0,00	0,22	0,00	1,85	0,00	0,18
20krod100	0,00	0,04	0,00	0,62	0,00	0,23	0,00	2,77	0,00	0,18
20kroe100	0,00	0,04	0,00	0,67	0,00	0,43	0,00	1,81	0,00	0,19
20rat99	0,00	0,04	0,00	0,58	0,00	0,15	0,00	3,89	0,00	0,19
20rd100	0,00	0,04	0,00	0,51	0,00	0,29	0,00	2,91	0,00	0,19
21eil101	0,00	0,04	0,00	0,48	0,00	0,18	0,00	2,09	0,00	0,17
21lin105	0,00	0,04	0,00	0,60	0,00	0,33	0,00	3,18	0,00	0,21
22pr107	0,00	0,05	0,00	0,53	0,00	0,20	0,00	4,78	0,00	0,21
24gr120	0,00	0,06	0,00	0,66	0,00	0,32	0,00	2,34	0,00	0,24
25pr124	0,00	0,06	0,00	0,68	0,00	0,26	0,00	2,84	0,00	0,29
26bier127	0,00	0,06	0,00	0,78	0,00	0,28	0,00	3,35	0,00	0,29
28pr136	0,00	0,08	0,00	0,79	0,16	0,36	0,00	4,23	0,00	0,37
29pr144	0,00	0,08	0,00	1,00	0,00	0,44	0,00	5,42	0,00	0,38
30kroa150	0,00	0,09	0,00	0,98	0,00	0,32	0,00	5,95	0,00	0,38
30krob150	0,00	0,09	0,00	0,98	0,00	0,71	0,00	5,02	0,00	0,40
31pr152	0,00	0,10	0,00	0,97	0,00	0,38	0,00	5,24	0,00	0,46
32u159	0,00	0,11	0,00	0,98	0,00	0,55	0,00	5,58	0,00	0,45
39rat195	0,02	0,20	0,00	1,37	0,00	1,33	0,00	11,01	0,00	0,74
40d198	0,00	0,19	0,00	1,63	0,07	1,47	0,00	10,15	0,00	0,82
40kroa200	0,00	0,23	0,00	1,66	0,00	0,95	0,00	10,41	0,00	0,81
40krob200	0,00	0,31	0,05	1,63	0,01	1,29	0,00	10,81	0,02	0,95
45ts225	0,00	0,60	0,14	1,71	0,28	1,09	0,04	31,45	0,04	1,31
46pr226	0,00	0,20	0,00	1,54	0,00	1,09	0,00	8,25	0,00	1,02
53gil262	0,14	0,65	0,45	3,64	0,55	3,05	0,14	24,34	0,14	2,03
53pr264	0,00	0,42	0,00	2,36	0,09	2,72	0,00	18,27	0,00	1,61
60pr299	0,01	1,05	0,05	4,59	0,16	4,08	0,00	21,25	0,04	3,30
64lin318	0,00	1,40	0,00	8,08	0,54	5,39	0,00	26,33	0,00	3,79
80rd400	0,15	4,33	0,58	14,58	0,72	10,27	0,42	32,21	0,00	10,57
84fl417	0,01	4,26	0,04	8,15	0,06	6,18	0,00	31,63	0,01	7,98
88pr439	0,05	5,47	0,00	19,06	0,83	15,09	0,00	42,55	0,09	10,65
89pcb442	0,40	7,30	0,01	23,43	1,23	11,74	0,19	62,53	0,13	13,65
ORTALAMA	0,02	0,68	0,03	2,69	0,11	1,77	0,02	10,12	0,01	1,59

Çalışmalarda farklı hızlara ve özelliklere sahip bilgisayarlar kullanıldığından dolayı CPU zamanları ile ilgili tam bir karşılaştırma yapma olanağı bulunmamaktadır. Bu sebeple CPU zamanları ile ilgili sonuçlar tabloda gösterilmiş ancak yorumda bulunulmamıştır.

BÖLÜM YEDİ

7. SONUÇ

Bu çalışmada GGSP, Bölgesel Tarama Sezgiseli ile birleştirilmiş KFE Algoritması kullanılarak çözülmüştür. GGSP çözümü için GTSPLib kütüphanesindeki şehir sayısı 48 ile 1084 arasında değişen, küme sayısı 10 ile 217 arasında değişen test problemleri kullanılmıştır.

Test problemleri Tarama ile birleştirilmiş KFE Algoritması kullanılarak çözülmüş, sonuçlar Bontoux, Artigues ve Feillet'in (2009) Memetik Algoritması Taşgetiren, Suganthan ve Pan'ın (2009) eDDE algoritması, Synder ve Daskin'in (2006) RKGA ve Silberholz ve Golden'ın (1997), mrOXGA ile kıyaslanmıştır. Sonuç olarak eniyi değerleri bilinen 41 test probleminin sonuçları kıyaslandığında KFE Algoritması mrOXGA, MA ve eDDE algoritmasına eşdeğer olduğu ancak RKGA'dan daha iyi sonuçlar ürettiği görülmüştür.

KAYNAKÇA

Ben-Arieh, D., Gutin, G., Penn, M., Yeo, A., Zverovitch, A. (2003). Process planning for rotational parts using the generalized traveling salesman problem, *Int. J. Prod. Res*. 41(11), 2581–2596.

Bontoux, B., Artigues, C., Feillet, D., (2009). A memetic algorithm with a large neighborhood crossover operator for the generalized traveling salesman problem. *Computers and Operations Research*.

Cura, T. (2008), *Modern sezgisel teknikler ve uygulamaları* (1). İstanbul: Papatya Yayıncılık.

Davis, L., (1986), Applying adaptive algorithms to domains, *The International Joint Conference on Artificial Intelligence,* 162-164.

Davis, L. (1991). *Handbook of Genetic Algorithms*. Van Nostrand Reinhold Computer Library, New York.

Dantzig, G. B., Fulkerson,. D. R. (1954). Minimizing the number of tankers to meet a fixed schedule. *Naval Res. Logist. Quart.* 217-222.

Dimitrijevic, V., Saric, Z. (1997). Efficient transformation of the generalized traveling salesman problem into the traveling salesman problem on digraphs. *Information Science,* 102, 65–110.

Erentürk, M., (2004). *Performance analysis of meta-heuristic approaches for traveling salesperson problem*. Yayınlanmamış yüksek lisans tezi, Fen Bilimleri Enstitüsü, Yeditepe Üniversitesi.

Fischetti, M., (1997). Salazar-Gonzalez, J., Toth, P. A branch-and-cut algorithm for the symmetric generalized traveling salesman problem. *Oper. Res*. 45(3), 378–394.

Fischetti, M., Salazar-Gonzalez, J., Toth, P. (1995). The symmetrical generalized traveling salesman polytope. *Networks* 26(2), 113–123.

Fischetti, M., Salazar-Gonzalez, J.J., Toth, P., (2002). "*The Generalized Traveling Salesman and Orienteering Problems*", G. Gutin (Ed.) The Traveling Salesman Problem and Its Variations, 609-662, Dordrecht: Kluwer Academic Publisher.

Goldberg, D., Lingle, R., (1985), Alleles, loci and the travelling salesman problem, *Grefenstette* (186), pp. 154, 159.

Henry-Labordere A. (1969). The record balancing problem–A dynamic programming solution of a generalized traveling salesman problem, *Revue Francaise D. Informatique DeRecherche Operationnelle* 3(NB2), 43–49.

Laporte, G., Nobert, Y. (1983). Generalized traveling salesman problem through n-sets of nodes - An integer programming approach. *INFOR* 21(1), 61–75.

Laporte, G., Mercure, H., Nobert, Y. (1987). Generalized traveling salesman problem through n -sets of nodes - The asymmetrical case. *Discrete Appl. Math*. 18(2), 185–197.

Laporte, G., Asef-Vaziri, A., Sriskandarajah, C. (1996). Some applications of the generalized travelling salesman problem, *J. Oper. Res. Soc*. 47(12), 461–1467.

Lien, Y., Ma, E., Wah, B. (1993). Transformation of the generalized traveling salesman problem into the standard traveling salesman problem. *Information Science* 64, 177–189.

Kara İ., Güden H., Öztürk A., Seyhan A., (2009). Genelleştirilmiş Gezgin Satıcı Probleminin Polinom Büyüklükte Karar Modellerinin Sayısal Karşılaştırma Sonuçları. *10. Ekonomi ve İstatistik Sempozyumu*, Erzurum, 28/05/2009-30/05/2009.

Karakoç, Ö., (2006). *Deneylerin Faktöriyel Tasarımı*, , Yayınlanmamış yüksek lisans tezi, Marmara Üniversitesi Fen Bilimleri Enstitüsü.

Keskintürk, T., (2006), Diferansiyel Gelişim Algoritması, *İstanbul Ticaret Üniversitesi Fen Bilimleri Dergisi* 5(9), 85-89.

Montgomery, D.C., (2005). *Introduction to Statistical Quality Control* (5. Baskı). New York: John Wiley and Sons.

Moscato, P., Cotta, C. (2005). A gentle introduction to memetic algorithms. *Operations Research& Management Science*; 57(2): 105-44).

Noon C. (1988). *The generalized traveling salesman problem*, Ph.D. Thesis. University of Michigan.

Noon, C., Bean, J. (1991). A Lagrangian based approach for the asymmetric generalized traveling salesman problem. *Oper. Res*. 39(4), 623–632.

Noon C. E. & Bean J. C. (1993). An efficient transformation of the generalized travelling salesman problem. *INFOR* 31, 39- 44.

Özçelik, Y. (2007). *Farksal evrim algoritması kullanılarak sistem kimliklendirme*. Yayınlanmamış yüksek lisans tezi, Erciyes Üniversitesi, Fen Bilimleri Enstitüsü.

Paksoy, S., (2008). Uzun, A., Genetik algoritma ile kaynak kısıtlı proje çizelgeleme, *Ç.Ü. Sosyal Bilimler Enstitüsü Dergisi*, 17(2), 345-362.

Rardin, R.L., (2000). *Optimization in Operation Research*. New Lersey: Prentice Hall.

Renaud, J., Boctor, F., Laporte, G. (1996). A fast composite heuristic for the symmetric traveling salesman problem. *INFORMS Journal on Computing* 4, 134–143.

Renaud, J., Boctor, F. (1998). An efficient composite heuristic for the symmetric generalized traveling salesman problem. *Eur. J. Oper. Res.* 108(3), 571–584.

Saskena J (1970). Mathematical Model Of Scheduling Clients Through Welfare Agencies, *Journal of the Canadian Operational Research Society* 8:185–200.

Silberholz, J., Golden, B. (1997). The generalized traveling salesman problem: A new genetic algorithm approach. K. B. Edward v.d. (Ed.) Extending the horizons: *Advances in Computing, Optimization and Decision Technologies*, vol. 37, pp. 165–181. Heidelberg, Springer.

Schmidt, H., Thierauf, G., (2005). A Combined Heuristic Optimization Technique. *Advances in Engineering Software*,36, 11-19.

Snyder, L., Daskin, M. (2006). A random-key genetic algorithm for the generalized traveling salesman problem. *Eur. J. Oper. Res*. 174, 38–53.

Srivastava, S., Kumar, S., Garg, R., Sen, R. (1970). Generalized traveling salesman problem through n sets of nodes. *Journal of the Canadian Operational Research Society* 7, 97–101.

Storn R, Price K., (1995). Differantial Evaluation- A simple and efficient adaptive scheme for global optimization over continuous spaces. *ICSI*, Technical Report TR-95-012.

Synder, L. V., Daskin, M. S.,(2006). A random key genetic algorithm for the generalized travelling salesman problem, European Journal of Operation Research 174, 38-53.

Syswerda, G. (1991). *Schedule Optimization Using Genetic Algorithms*. L. Davis (Ed.), Handbook of Genetic Algorithms, 332-349. New York: Van Nostrand Reinhold.

Şirvancı, M., (1997). *Kalite için deney tasarımı- Taguchi Yaklaşımı*. İstanbul: Literatür Yayınları.

Tasgetiren, M., Sevkli, M., Liang, Y.-C., Gencyilmaz, G. (2004). Particle Swarm Optimization Algorithm for the Single Machine Total Weighted Tardiness Problem. In: *The Proceeding of the World Congress on Evolutionary Computation, CEC*, 1412–1419.

Tasgetiren, M., Suganthan, P., Pan, Q.-K. (2007). A discrete particle swarm optimization algorithm for the generalized traveling salesman problem. 9th Annual *Conference On Genetic And Evolutionary Computation (GECCO 2007)*, London UK, 158–167.

Tasgetiren, M., Suganthan, P., Pan, Q.-K., Liang, Y.-C. (2007). A genetic algorithm for the generalized traveling salesman problem. *World Congress on Evolutionary Computation (CEC 2007)*, Singapore, pp. 2382–2389 .

Tasgetiren, F., Liang, Y.-C, Pan, Q.-K., Suganthan, P., (2008) *Discrete /Binary Approach.* G. C. Onwubolu v.d. (Ed.), Differential Evolution: A Handbook for Global Permutation-Based Combinatorial Optimization (139-162). Heidelberg, Springer.

Tasgetiren, M., Suganthan, P., Pan, Q.-K., (2010). An ensemble of discrete differantial evolution algorithms for solvining the generalized traveling salesman problem, *Applied Mathematics and Computation*, 215 (9), 3356-3368.

Tasgetiren F., Chen A., Gencyilmaz G. ve Gattaufi, S., (2009). *Smallest Position Value Approach. Studies in Computational Intelligence*, 175, 121-128.

Türkbey, O. (2002). Tesis Düzenlemesi Probleminde Bölgesel Tarama Sezgiseli Kullanan Bir Genetik Algoritma: Memetik algoritma yaklaşımı, *Pamukkale Üniversitesi Mühendislik Bilimleri Dergisi,* 8(2), 265-271.