KOCAELİ ÜNİVERSİTESİ*FEN BİLİMLERİ ENSTİTÜSÜ

GEZGİN SATICI PROBLEMİ İÇİN DİFERANSİYEL GELİŞİM ALGORİTMASI TABANLI BİR METASEZGİSEL ÖNERİSİ

DOKTORA TEZİ

Y.Müh. Ümit TERZİ

Anabilim Dalı : Endüstri Mühendisliği
Danışman: Prof. Dr. Alpaslan FIĞLALI

KOCAELİ, 2009

KOCAELİ ÜNİVERSİTESİ * FEN BİLİMLERİ ENSTİTÜSÜ

GEZGİN SATICI PROBLEMİ İÇİN DİFERANSİYEL GELİŞİM ALGORİTMASI TABANLI BİR METASEZGİSEL ÖNERİSİ

DOKTORA TEZİ

Y.Müh. Ümit TERZİ

Tezin Enstitüye Verildiği Tarih: 05 Ocak 2009

Tezin Savunulduğu Tarih: 26 Şubat 2009

Tez Danışmanı
Prof.Dr. Alpaslan FIĞLALI
(.....)

Üye
Prof.Dr. Zerrin ALADAĞ
(.....)

Üye
Yrd.Doç.Dr. Pınar KILIÇOĞULLARI
(.....)

Üye
Yrd.Doç.Dr. Orhan ENGİN
(.....)

Üye
Doç.Dr. Ayhan DEMİRİZ
(.....)

KOCAELİ, 2009

ÖNSÖZ ve TEŞEKKÜR

Kombinasyonel optimizasyon problemleri birçok alanda karşımıza çıkan ve genelde çözümü zor olan problemlerdir. Endüstri Mühendisliği alanında da bu tür problemlerle; tesis yerleşimi, sıralama, çizelgeleme, optimal kesme problemleri, kuadratik atama problemleri, lojistik optimizasyon problemleri (yükleme, araç rotalama v.s.) gibi birçok yerde karşılaşılmaktadır. Kombinasyonel optimizasyon literatüründe belki de günümüze kadar üzerinde en çok çalışılmış olan problemlerden olan Gezgin Satıcı Problemi bu alanda bilinen en zor problemlerden biridir. Endüstriyel rekabetin giderek arttığı günümüzde süreçlerde optimizasyona duyulan ihtiyaç ve bunun kısa sürede etkin bir şekilde gerçekleştirilmesi gerekliliği, Diferansiyel Gelişim Algoritması gibi başarılı ve çok yönlü meta sezgisellerin kombinasyonel alanda da kullanılması ve geliştirilmesi ihtiyacını doğurmaktadır.

Çalışmamın başından itibaren gösterdiği emek, destek ve ilgiyle çalışmanın gerçekleşmesi ve ilerlemesini sağlayan danışman hocam Sayın Prof. Dr. Alpaslan FIĞLALI' ya,

Çalışmam sırasında fikirleriyle bana yol gösteren ve destekleyen hocalarım Sayın Yrd. Doç. Dr. Pınar KILIÇOĞULLARI ve Yrd. Doç. Dr. Orhan ENGİN' e,

Çalışmam sırasında yardımını hiçbir zaman esirgemeyen, samimiyet, manevi destek ve teşvikleri için tüm çalışma arkadaşlarıma,

Eğitim ve kariyer hayatım süresince sevgi ve emeği ile yanımda olan annem Asiye TERZİ ve babam Hüseyin TERZİ' ye,

Sonsuz teşekkürlerimi sunuyorum.

Endüstri Yük. Müh. Ümit TERZİ

İÇİNDEKİLER

ÖNSÖZ…… i
İÇİNDEKİLER…… ii
ŞEKİLLER DİZİNİ……..... iv
TABLOLAR DİZİNİ……... v
ÖZET……... vi
İNGİLİZCE ÖZET…….. vii
1. GİRİŞ……... 1
2. GEZGİN SATICI PROBLEMİ VE ÇÖZÜM YÖNTEMLERİ…….. 4
2.1. Gezgin Satıcı Problemi Çeşitleri…….. 6
2.2. Karmaşıklık Teorisi…….. 8
2.3. Kesin Çözüm Yöntemleri……... 11
2.3.1. Sayma yöntemi…….. 11
2.3.2. Dinamik programlama……... 12
2.3.3. Dal ve sınır yöntemi……... 13
2.3.4. Doğrusal çözüm yaklaşımları……..... 16
2.3.4.1. Dantzig-Fulkerson-Johnson (DFJ) modeli ……..... 16
2.3.4.2. Miller-Tucker-Zemlin (MTZ) modeli……..... 20
2.4. Sezgisel Çözüm Yöntemleri……..... 21
2.4.1. En yakın komşu yöntemi ……..... 22
2.4.2. En yakın ekleme yöntemi ……..... 23
2.4.3. Geometrik yöntem……..... 23
2.4.4. Greedy yöntemi……..... 24
2.4.5. Bovurka yöntemi…… 24
2.4.6. Christofides algoritması……..... 25
2.4.7. Grasp yöntemi……..... 25
2.4.8. R-opt denemeleri…… 26
2.4.9. Lin-kerninghan yöntemi……..……... 27
2.4.10. Hyperopt denemeleri…… 28
2.4.11. S&C (stem and cycle) yöntemi…… 29
2.5. Meta Sezgisel Yöntemler…….. 30
2.5.1. Genetik algoritmalar…….. 30
2.5.2. Karınca algoritması…… 35
2.5.3. Tabu arama……..... 38
2.5.4. Tavlama benzetimi……..... 41
2.5.5. Yapay sinir ağları……... 43
2.5.4. Parçacık sürü optimizasyonu……..... 46
3. DİFERANSİYEL GELİŞİM ALGORİTMASI……... 50
3.1. DGA Adımları……... 51
3.1.1. Populasyonun oluşturulması…….. 51
3.1.2. Mutasyon…… 52
3.1.3. Çaprazlama……..... 53
3.1.4. Seçilim……... 54
3.2. DGA Kontrol Parametreleri…….. 54

3.3. DGA Literatürü……..... 56
3.4. Literatür Özeti…….... 59
3.4.1. Sıralama vektörüne dönüşüm……..... 60
3.4.2. DGA mutasyon formülü……..... 60
4. GELİŞTİRİLMİŞ DİFERANSİYEL GELİŞİM ALGORİTMASI……..... 61
4.1. DGA'dan GDGA'ya Gelişim…… 61
4.1.1. Sıralama Vektörüne Dönüşüm……... 63
4.1.1.1. İndeks sıralama kodlama……..... 63
4.1.1.2. Alt aralık kodlama……... 63
4.1.1.3. Rasgele sayı kodlama……..... 64
4.1.2. Gerçel sayı vektörüne dönüşüm……..... 65
4.1.2.1. Karşılığını bularak geri dönüşüm…… 65
4.1.2.2. Normalizasyon ile geri dönüşüm …….. 68
4.1.3. Yerel arama……..……..... 70
4.2. Deneysel Çalışma……..…… 71
4.2.1. Varyans analizi sonuçları……..……..... 74
4.3. Literatür Problemleri Denemeleri……..... 78
4.3.1. Komşuluk arama……..…….. 79
4.4. Durdurma Kriteri……..……..... 82
4.4. Referans Karşılaştırması……... 83
5.SONUÇ VE ÖNERİLER…….. 85
KAYNAKLAR……..... 87
ÖZGEÇMİŞ……..... 97

ŞEKİLLER DİZİNİ

Şekil 2.1: Örnek Gezgin Satıcı Turu……..……..……..……..... 4
Şekil 2.2: Çoklu Gezgin Satıcı (m-TSP) Turu……..……..... 6
Şekil 2.3: P,NP,NP-tam ve NP-zor için küme çizimi……...……..... 10
Şekil 2.4: GSP'nin dinamik programlama ile çözüm algoritması……..... 12
Şekil 2.5: Alt Turlar……..... 15
Şekil 2.6: 1-ağacı (1-tree)……. ……..... 15
Şekil 2.7: Kesme Düzlemi Yöntemi……..... 17
Şekil 2.8: Minimum Yayılan Ağaç…….……..... 22
Şekil 2.9: Geometrik yöntem…….……..……..……..……..……..... 24
Şekil 2.10: Christofides Algoritmasında M Ağı……..... 25
Şekil 2.11: (a) 2-opt değişimi (b) 3-opt değişimi ……..... 26
Şekil 2.12: Lin-Kerningham yöntemi algoritma adımları…….…..... 28
Şekil 2.13: Örnek hyperopt denemeleri……..... 29
Şekil 2.14: Dal ve çevrim çıkarma ...…..……..……..……..……..……..... 29
Şekil 2.15: Temel Genetik Algoritma'nın Adımları……..……..……..……..... 31
Şekil 2.16: Çaprazlama işleminde oluşan uygunsuz çözümler ve düzeltilişi……... 32
Şekil 2.17 : Gerçek karıncaların en kısa yolu bulma aşamaları……..……..... 35
Şekil 2.18: GSP için Temel AS Algoritmasının Adımları……..……..……..... 36
Şekil 2.19: Standart TA Algoritması'nın Adımları……..……..……..……..... 39
Şekil 2.20: Tavlama Benzetimi Algoritmasının Adımları……..……..……..... 42
Şekil 2.21: GSP için Hopfield tarafından önerilen sinir ağı……..... 44
Şekil 2.22: Kohonen ağı……...……..... 45
Şekil 2.23: Yanping tarafından önerilen ağ……...……..... 45
Şekil 2.24: PSO Algoritması Adımları……..……..……..……..……..……..... 46
Şekil 3.1: Diferansiyel Gelişim Mutasyonu……..……..……..……..……..……..... 50
Şekil 3.2: Klasik DGA Algoritması……..……..……..……..……..……..……..... 52
Şekil 4.1: GDGA Algoritması……..……..……..……..……..……..……..... 62

TABLOLAR DİZİNİ

Tablo 4.1: DGA Kodlama Dönüşümleri…….. 64
Tablo 4.2: İndeks Sıralama Yöntemi için Karşılığını Bularak Gerçel Sayı Vektörlerine Dönüşüm……..….. 67
Tablo 4.3: Alt Aralık Kodlama Yöntemi için Karşılığını Bularak Gerçel Sayı Vektörlerine Dönüşüm……..….. 68
Tablo 4.4: Rasgele Sayı Kodlama Yöntemi için Karşılığını Bularak Gerçel Sayı Vektörlerine Dönüşüm……..….. 69
Tablo 4.5: İndeks Sıralama Yöntemi için Normalizasyon ile Gerçel Sayı Vektörlerine Dönüşüm……..….. 69
Tablo 4.6: Alt Aralık Kodlama Yöntemi için Normalizasyon ile Gerçel Sayı Vektörlerine Dönüşüm……..….. 70
Tablo 4.7: Rasgele Sayı Kodlama Yöntemi için Normalizasyon ile Gerçel Sayı Vektörlerine Dönüşüm…….…..….. 71
Tablo 4.8: GDGA Deneyleri……..….. 72
Tablo 4.9: Bayg29 problemi için varyans analizi sonuçları……..….. 73
Tablo 4.10: Att48 problemi için varyans analizi sonuçları…….. 74
Tablo 4.11: Faktör Seviyelerine Göre % Sapma Ortalamaları ……... 76
Tablo 4.12: Bayg29 problemi için ikili etkileşimler……..….. 77
Tablo 4.13: Att48 problemi için ikili etkileşimler……..….. 77
Tablo 4.14: Mutasyon Denklemleri için Faktör Seviyelerine Göre % Sapmalar ….... 78
Tablo 4.15: Referans Problem Sonuçlarının Küresel En İyi Değerden Yüzde Sapmaları……..…..….. 79
Tablo 4.16: Komşuluk Arama Sonucunda Yüzde Sapmalar ve Populasyondaki Yerleri…….. 81
Tablo 4.17: Karşılaştırmalı Küresel En İyiden Ortalama Sapma Değerleri……..….... 84

GEZGİN SATICI PROBLEMİ İÇİN DİFERANSİYEL GELİŞİM ALGORİTMASI TABANLI BİR METASEZGİSEL ÖNERİSİ

Ümit TERZİ

Anahtar Kelimeler: Diferansiyel Gelişim Algoritması, Gezgin Satıcı Problemi, Kombinasyonel Optimizasyon

Özet: Diferansiyel Gelişim Algoritması kombinasyonel alanda henüz etkin kullanımı olmasa da sürekli çözüm uzayında ve bazı kesikli problemlerde oldukça başarılı çözümler elde edilmesine imkan tanıyan populasyon tabanlı bir meta sezgisel yöntemdir. Bu çalışmada, yöntemin başarılı yapısının kombinasyonel en iyileme alanında kullanılabilmesi amacıyla, yeni bir yaklaşım geliştirilmiştir. Diferansiyel Gelişim Algoritması kayar noktalı vektörler üzerinden çalıştığından sürekli vektörlerden sıralama vektörlerine dönüşüm gerekmektedir ve yerel aramadan faydalanabilmek için de geriye dönüşüm gereklidir. Önerilen yöntem uygun dönüşüm yoluyla sıralama uzayı ve sürekli uzayı birleştirmektedir. Geliştirme ve başarımının denenmesi aşamalarında en zor problemler grubunda (NP-hard) yer alan ve üzerinde en çok çalışılmış kombinasyonel eniyileme problemi olan Gezgin Satıcı Problemi (GSP) kullanılmıştır. Böylece GSP üzerinde yapılan çalışmaların, bu çalışmaya ışık tutması hedeflenmiştir. Bu amaçla, GSP'nin çözümü üzerinde yapılan kesin, sezgisel ve meta sezgisel uygulamalar özetlenerek, bu alanda karşılaşılan problemler ve geliştirilen çözüm yaklaşımları anlatılmıştır.

Önerilen geliştirme ile Diferansiyel Gelişim Algoritması, permütasyonel problemleri çözebilme becerisine kavuşmuştur. Algoritmanın bu alanda en çok kullanılan ve Diferansiyel Gelişim Algoritması gibi populasyon tabanlı olan Karınca Kolonileri, Genetik Algoritmalar gibi algoritmalar ile eşdeğer başarım seviyesine taşınmış olduğu gösterilmiştir. Ayrıca yine populasyon tabanlı olan, gösterim ve algoritma işleyişi bakımından Diferansiyel Gelişim Algoritmasına oldukça benzeyen Parçacık Sürü Optimizasyonu Algoritması'na göre daha iyi sonuçlar elde edilebildiği görülmektedir.

A DIFFERENTIAL EVOLUTION ALGORITHM BASED METAHEURISTIC PROPOSAL FOR THE TRAVELING SALESMAN PROBLEM

Ümit TERZİ

Key Words: Differential Evolution Algorithm, Traveling Salesman Problem, Combinatorial Optimization

Abstract: Differential Evolution Algorithm is a population based metaheuristic that is successful in the area of continuous optimization and for some discrete optimization problems, although has no effective implementations in combinatorial optimization area. In this study a new approach was proposed for deploying the success of the method to the combinatorial optimization. Differential Evolution Algorithm works with floating point vectors hence appropriate mapping is needed from continuous vectors to permutation vectors and backwards transformation is necessary to benefit from local search. The proposed method connects continuous space and permutation space through convenient transformation. The Traveling Salesmen Problem, which is in the most difficult problems group (NP-hard) and the most studied problem in combinatorial optimization area, was used as a test bed at improvement and evaluation phases of the method. Thus, it is aimed that the previous studies on the problem would help the development of the method. Therefore discrete, heuristic and metaheuristic studies of solving TSP are summarized and challenges that encountered in these efforts and their solution approaches are explained.

Differential Evolution Algorithm acquired the capability of solving permutational optimization problems with the proposed improvements. It has been shown that the algorithm has an equivalent performance with the extensively used population based algorithms like Genetic Algorithms and Ant Colonies. Furthermore it is seen that the proposed algorithm end up with superior results than Particle Swarm Optimization Algorithm which is similar to Differential Evolution Algorithm in terms of representation and working principles.

1. GİRİŞ

Kombinasyonel optimizasyon problemleri birçok alanda karşımıza çıkan ve genelde çözümü zor olan problemlerdir. Kombinasyon, bir kümeden sıra gözetmeksizin seçilen bir alt kümeyi ifade etmektedir. Sıranın önemli olması durumunda yapılan seçim permütasyon olarak isimlendirilir. Örneğin 1'den 10'a kadar pozitif tamsayıların bulunduğu A={1,2,3,4,5,6,7,8,9,10} kümesinden seçilen B={1,5,7} kümesi A'nın 3'lü bir kombinasyonu olarak ifade edilir ve *n* adet elemanın bulunduğu bir kümeye ait *r*'li kombinasyonların sayısı 1.1 formülünde görüldüğü gibi hesaplanır. Aynı şekilde n adet elemana sahip bir kümede *r*'li permütasyonların sayısı 1.2. formülünde görüldüğü gibidir. İkinci formül ilk formülden *r*! kadar daha büyük bir sayıyı ifade etmektedir. Bu durum, seçilen bir kombinasyona ait *r*! adet permütasyonun bulunduğunu göstermektedir. B alt kümesinde sıralamanın önemli olduğunun kabul edilmesi durumunda 3!=6 adet farklı alt kümenin B_1={1,5,7} B_2={1,7,5}, B_3={5,1,7}, B_4={5,7,1}, B_5={7,1,5}, B_6={7,5,1} türetilmesi mümkün olmaktadır. Kombinasyon ve permütasyon tanımlarının farklılığına karşın kombinasyonel problemler denildiğinde her ikisini de kapsayacak şekilde, belirli bir kümeden bir veya daha çok amaç fonksiyonunu eniyilemek amacıyla yapılan seçimler ifade edilmektedir.

$$C(n,r) = \frac{n!}{r!(n-r)!} \tag{1.1}$$

$$P(n,r) = \frac{n!}{(n-r)!} \tag{1.2}$$

Endüstri Mühendisliği alanında da bu tür problemlerle birçok yerde karşılaşılmaktadır. Tesis yerleşimi, sıralama, çizelgeleme, optimal kesme problemleri, kuadratik atama problemleri, lojistik optimizasyon problemleri (yükleme, araç rotalama v.s.) bunlar içerisinde en çok bilinenler olarak sıralanabilir.

Gezgin Satıcı Problemi (Traveling Salesman Problem) kısaca GSP (TSP), kombinasyonel optimizasyon literatüründe belki de günümüze kadar üzerinde en çok çalışılmış olan problemlerden biridir. Basitçe bir gezgin satıcının uğraması gereken bütün noktaları dolaşarak başladığı noktaya dönmesini sağlayacak en kısa turun bulunmasının hedeflendiği Gezgin Satıcı Problemi'nin önemi, gerçek yaşamda çözümlediği problemlerin öneminden çok kombinasyonel optimizasyon çalışmaları içerisindeki yerinden kaynaklanır. Problemin basit tanımına karşın, çözümünün zorluğu bu çalışmanın yazarının da içinde bulunduğu birçok araştırmacıyı cezbetmeye devam etmektedir. Bu tez çalışmasında Gezgin Satıcı Problemi üzerinde çalışılmış, ancak diğer kombinasyonel problemler üzerinde de uygulama imkanı bulunan bir meta sezgisel yaklaşım geliştirilmeye çalışılmıştır.

Diferansiyel Gelişim Algoritması (DGA) çözümün gerçel sayı vektörleriyle ifade edildiği, populasyon tabanlı yeni bir sezgisel eniyileme yöntemidir. Yöntem, diğer meta sezgisellerden farklı olarak vektörler arasındaki farklılıklardan da yararlanarak populasyonu geliştirmektedir. Ağırlıklı olarak sürekli eniyileme problemlerinin çözümünde kullanılan yöntemin kesikli ya da permütasyonel eniyileme problemleri için etkin uygulamaları bulunmamaktadır. Bu çalışmada DGA, permütasyonel eniyileme problemlerinde kullanılmak üzere yeniden düzenlenmiş ve geliştirilmiştir. Önerilen algoritmanın başarımı Gezgin Satıcı Problemi literatür problemleri kullanılarak değerlendirilmiştir.

Bu tez çalışması, 5 ana bölümden oluşmaktadır. 2. bölümde Gezgin Satıcı Probleminin tanımı, çeşitleri ve gerçek hayatta karşılaşıldığı alanlara değinilmektedir. Tez boyunca anlatım ve yöntemlerin değerlendirilmesi ile ilgili kolaylık sağlayan karmaşıklık teorisine değinilmektedir. Bölümün devamında GSP'nin çözümü için kullanılan kesin, sezgisel ve meta-sezgisel çözüm yöntemleri ve bu alanda yapılan çalışmalar özetlenmektedir.

3. bölümde Diferansiyel Gelişim Algoritması (DGA) anlatılmakta ve bu konudaki literatür özetlenmektedir. Öncelikle ortaya çıkış süreci anlatılmış, ardından bu çalışmanın konusu ile ilgili olan kombinasyonel eniyileme alanında yapılmış uygulamalar karşılaştırmalı olarak sunulmuştur.

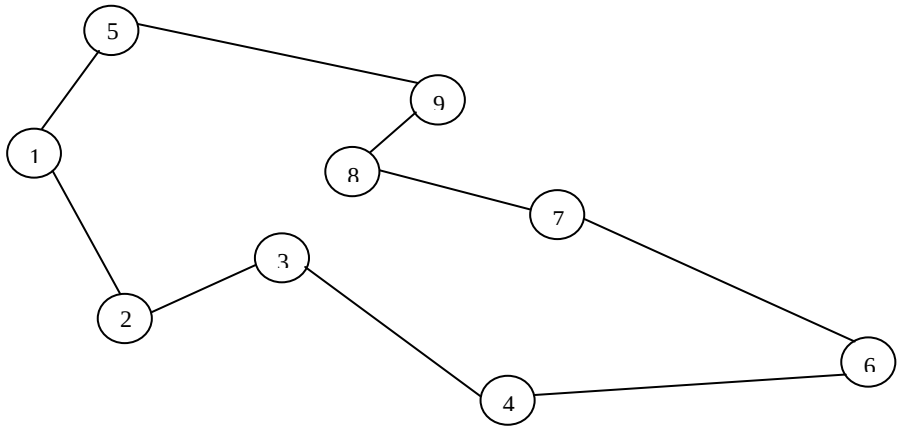
4. bölümde Gezgin Satıcı Probleminin çözümü için geliştirilen, DGA'nın kombinasyonel problemlerde uygulanmasına olanak veren, Geliştirilmiş Diferansiyel Gelişim Algoritması (GDGA) anlatılmaktadır. Sürekli eniyilemeden permütasyonel eniyilemeye dönüşüm, algoritma adımlarının olası seçenekleriyle birlikte sunulması, en uygun parametre ve yöntemin belirlenmesi için yapılan deneysel çalışma yardımıyla anlatılmıştır. Deney tasarımı çalışması sonucunda elde edilen sonuçlar; etkileşimler de göz önüne alınarak yorumlanmış, en uygun algoritma alternatifi sunulmuştur. Bölüm sonunda belirlenen algoritmanın literatür problemlerine uygulanması sonucu elde edilen sonuçlar, en çok bilinen populasyon tabanlı meta sezgiseller ile karşılaştırmalı olarak sunulmuştur.

5. bölümde elde edilen sonuçlar yorumlanmış, sonraki çalışmalar için olası gelişme alanlarına değinilmiştir.

2. GEZGİN SATICI PROBLEMİ VE ÇÖZÜM YÖNTEMLERİ

Gezgin Satıcı Problemi, belirli sayıda şehir ve aralarında belirli uzaklıkların bulunduğu bir şebekede bir gezgin satıcının, bütün şehirlere sadece bir kere uğramak ve başladığı şehre geri dönmek şartıyla dolaşabileceği en kısa turun (Şekil 2.1) bulunmasının hedeflendiği bir problem olarak tanımlanabilir.

Kombinasyonel problemler arasında en çok bilinenlerden biri olan Gezgin Satıcı Problemi, yapısı nedeniyle doğal olarak ağ teorisinin de kapsamına girmektedir. Ağ teorisinde, sınırlı sayıda V düğümleri ve bu düğümler arasındaki A bağlarının bulunduğu bir G(V,A) ağında, bu düğümleri birbirine bağlayan en kısa Hamilton turunun bulunması şeklinde anlatılabilir. İlk kez 19. yüzyılda yaşamış İrlandalı bir matematikçi olan Hamilton tarafından tanımlanan tur, bir ağdaki tüm düğümleri içeren bir çevrim (cycle) olarak tanımlanır. Burada çevrim, bir düğümden başlayarak belirli sayıdaki düğüme sadece bir defa uğrayarak, başlangıç düğümüne dönen bağlar tarafından oluşturulan bir yoldur. Hamilton turunun tanımında bağların uzunluğu önemli değildir. Belirli sayıdaki düğüm ve bu düğümler arasında sınırlı sayıda bağın bulunduğu bir ağda, bir Hamilton turunun olup olmadığı başka bir kombinasyonel problem çeşididir.



Şekil 2.1: Örnek Gezgin Satıcı Turu

Gezgin Satıcı Problemi gerçek yaşamda, devre kartlarında delikler açılması, robotların depolar ve üretim istasyonları arası yönlendirilmesi, büyük ölçekli devrelerin tasarlanması, tek makinede ürün değişiminde ayar süreleri en küçüklenmesi, lojistikte çeşitli dağıtım ve satın alma problemleri gibi problemlerde problemin tamamı ya da bir bölümü olarak karşımıza çıkabilmektedir. Sıralanan alanların dışında değişik kaynaklarda [1,2] bir çok örnek yer almaktadır. Ancak, Gezgin Satıcı Problemi'nin önemi, gerçek yaşamda çözümlediği problemlerin öneminden çok kombinasyonel eniyileme çalışmaları içerisindeki yerinden kaynaklanır. GSP, kombinasyonel eniyileme çalışmalarında geliştirilen yöntemler için en çok kullanılan başarım değerlendirme araçlarından biri olmakla birlikte, kendisi için geliştirilen bir çok yöntem, diğer yöneylem araştırması problemleri için istenen çözümlere ulaşılabilmesine de olumlu etkiler yaratmaktadır.

GSP için geliştirilmiş ya da GSP üzerinde uygulanan bir çok çözüm yöntemi bulunmaktadır. Bu yöntemlerın bazıları analitik süreçler yardımıyla, en kısa tura ulaşmayı garantilemeye çalışmaktadır. Tam sayma, dinamik programlama, dal ve sınır, dal ve kes gibi yöntemler bu grupta yer almaktadır. Bu tür yöntemler en kısa turun kesinliği karşılığında uzun süreler alabilmektedir. Süre gereksinimini en aza indirebilmek için bazı yaklaşımlar geliştirilmiştir.

Çözüm yöntemlerinin bazıları özellikle GSP için geliştirilmiştir. Oldukça kısa sürede çözüm elde edilmesine imkan veren bu tür çalışmalardan elde edilen bilgilerin diğer kombinasyonel eniyileme problemleri için kullanılması oldukça zordur. Bunun için mevcut problemin, Gezgin Satıcı Problemi formuna dönüştürülmesi gerekmektedir. Sezgisel en iyi tur oluşturma ve geliştirme yöntemleri bu gruba girmektedir.

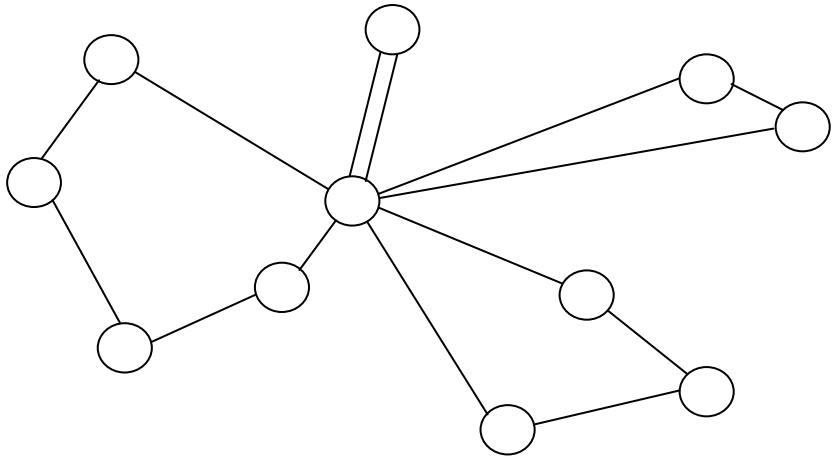
GSP problemleri için öne çıkan diğer bir grup çözüm yöntemi meta sezgisellerdir. Genetik Algoritmalar, Karınca Kolonileri, Tabu Arama, Tavlama Benzetimi, Sinir Ağları gibi meta sezgisel yöntemler GSP için kullanıldığı gibi diğer kombinasyonel problemler için de kullanılmakta ve başarılı sonuçlar elde edilmektedir. Bu yöntemler küresel en iyiyi garanti edememekle birlikte kısa sürede tatmin edici derecede iyi çözümler vermeleri nedeniyle sıklıkla kullanılmaktadırlar.

2.1. Gezgin Satıcı Problemi Çeşitleri

Gezgin Satıcı Probleminin birçok çeşidi bulunmaktadır. En çok bilineni, nokta i'den nokta j'ye ulaşım maliyetinin, nokta j'den nokta i'ye ulaşım maliyetine eşit olduğu "Simetrik Gezgin Satıcı" problemidir. Gidiş dönüş maliyetlerinin farklı olması durumunda ise "Asimetrik Gezgin Satıcı" problemi olarak tanımlanır. Bu çeşit problemlerde satıcının hareket yönü önem kazanır. Blaser ve arkadaşları [3] ile Kwon ve arkadaşlarının [4] yaptığı çalışmalar bu problem türünün çözümü için değişik çözüm örnekleridir.

Gezgin Alıcı Probleminde, ziyaret edilmesi gereken şehirlerin yanında, satın alınması gereken bir malzeme listesi söz konusudur. Malzemeler farklı şehirlerde değişik fiyatlarla satılmaktadır. Amaç toplam yol ve satın alma maliyetini minimize etmektir. Pearn ve arkadaşlarının yaptıkları çalışmada [5] mevcut turun geliştirilmesi içi bazı öneriler sunulmaktadır.

Çoklu GSP'de (Şekil 2.2) bir ana depo ve bu depodan belli güzergahlara dağıtılması gereken mallar söz konusudur. Dağıtım belli sayıda, problemin yapısına göre sınırlı ya da sınırsız kapasiteye sahip, kamyonlar tarafından gerçekleştirilecektir. Her bir kamyonun gerekleştireceği tur bir GSP'dir. Ancak kamyon turlarının birbirini etkilemesi nedeniyle farklı bir yapıya sahiptir. Bektaş [6] bu tip problemin çözümü için kullanılan yaklaşımları özetlemektedir.



Şekil 2.2: Çoklu Gezgin Satıcı (m-TSP) Turu

Çinli Postacı probleminde [7], postacı mevcut ağdaki bütün yolları kullanarak ve bütün yollara bir kez uğrayarak en kısa uzunluktaki turu tamamlamak istemektedir. Ağ teorisinde bu tur Euler turu olarak adlandırılır.

Zamana bağlı (Time dependent) GSP, Ödül Toplayan (Prize Collecting) GSP, Gezgin Vaiz (Traveling Preacher) Problemi, En Büyük (Maximum) GSP, On-line GSP, Öncelik kısıtlı GSP, Minimum gecikmeli GSP [8], Darboğaz GSP, Genelleştirilmiş GSP [9],v.b bir çok gezgin satıcı problemi çeşidi bulunmaktadır.

Yukarıda sayılan problemlerin yanında ufak tefek değişikliklerle tanımlanabilecek bir çok çeşit mevcuttur. Şehirlerin bir defa ya da daha çok sayıda ziyaretine izin verilmesi; gidiş dönüş uzaklıklarının farklı ya da aynı olması; aynı yolun birden çok kez kullanılması ya da kullanılmaması; bütün yolların kullanılması ya da kullanılmaması; satıcının uğradığı şehirlerde bazı görevleri yerine getirmek zorunda olup olmaması; satıcı için taşıma kapasitesinin olup olmadığı; birden çok satıcı olup olmadığı, başlangıç ve varış noktalarının aynı olup olamayacağı; bağlar ya da noktalarda olasılıklı yapının bulunup bulunmadığı; şehirlere ya da uzaklıklara ait bilgilerin zamanla değişip değişmediği; birden çok amacın bulunup bulunmadığı, gibi değişikliklerle onlarca değişik tipte gezgin satıcı problemi tanımlanabilir.

Bu çalışmada, en çok bilinen, üzerinde en çok çalışılan ancak henüz her büyüklükteki problem için makul sürede en iyi çözümü garanti eden yöntemin bulunamadığı, yukarıda tanımlanmış olan Simetrik GSP üzerinde çalışılacaktır. Çalışmanın bundan sonraki bölümünde GSP'den bahsedildiğinde Simetrik GSP anlaşılacaktır.

GSP için çözümde farklılık yaratan bir başka nokta ise şehirler arasındaki uzaklıkları gösteren matrisin yapısıdır. Uzaklık matrisi bir düzlem üzerinde yer alan noktalara ait koordinatlar arasındaki doğrusal uzaklıklarının hesaplanmasıyla oluşturulmuş ise problemin düzlemsel olduğu söylenir. Bu tür problemlerde önemli olan nokta, şehirler arası yolların doğrusal olmasıdır. Bu tip problemler için grafik çözümler önerilebilmektedir. Bir plaka üzerinde matkapla delme problemi düzlemsel problemler için iyi bir örnek olarak gösterilebilir. Ancak gerçek yaşam

problemlerinin birçoğu düzlemsel değildir. Düzlemsel yapıdaki iki boyutun yanına, şehirler arası rakım farklarında olduğu gibi, üçüncü bir boyutun katılması da mümkündür. Bu iki tip problemde de doğrusallığın kontrolü için üçgen eşitsizliği kullanılır. Üçgen eşitsizliği basit olarak, bir üçgendeki iki kenarın uzunlukları toplamının, üçüncü kenar uzunluğundan büyük olması anlamına gelmektedir. Bu eşitsizliğin sağlanmaması durumunda üçüncü kenarın doğrusal olmayan bir yapı gösterdiği anlaşılır. Gerçek yaşam problemlerine daha yakın olan bu tür problemlerin çözümü nispeten daha zordur. Bu nedenle geliştirilmiş olan yöntemlerin bazılarında, düzlemsellik ya da üçgen eşitsizliğini sağlama varsayımı yapılmaktadır.

2.2. Karmaşıklık Teorisi

Algoritmalar, problemlerin çözümü için geliştirilmiş adımlar bütünüdür. Karmaşıklık teorisi, problemlerin zorluk dereceleri açısından sınıflandırılması ve algoritmaların etkinliklerinin karşılaştırılması işlevini yerine getirmektedir. Karşılaştırma ve sınıflandırmalar problemlerin çözümü için gerekli hesaplama gücü açısından gerçekleştirilmektedir. Problemler genel olarak kolay, zor gibi sınıflamalara tabi tutulurlar. Algoritmalar ise problem büyüklüğü ile problemin çözümü için ihtiyaç duyulan kaynaklar arasındaki ilişkiye göre değerlendirilirler. Bu noktada bazı tanımların yapılması, konunun anlaşılması açısından fayda sağlayacaktır.

Tanım 2.1: A, X kümesinden girdiler kabul eden bir algoritma ve $f : N \rightarrow R^{+}$ olsun. Eğer A algoritması α bir sabit ve $\alpha > 0$ olmak üzere, her $x \in X$ için en fazla $\alpha\ f(boyut(x))$ işlem adımında çözüme ulaşıyorsa A'nın O(f) zamanda çalıştığı, zaman karmaşıklığının O(f) olduğu söylenir[10].

Tanım 2.2: Bir algoritma, n girdi büyüklüğü ve k bir tamsayı olmak üzere, çözümü O (n^k) zamanda yapabiliyor ve hesaplamalar için gerekli veriler ortalama O (n^k) bit yer tutuyor ise polinom zamanlı olarak nitelendirilir. Eğer k=1 olursa doğrusal zamanlı bir algoritma anlamına gelmektedir[10].

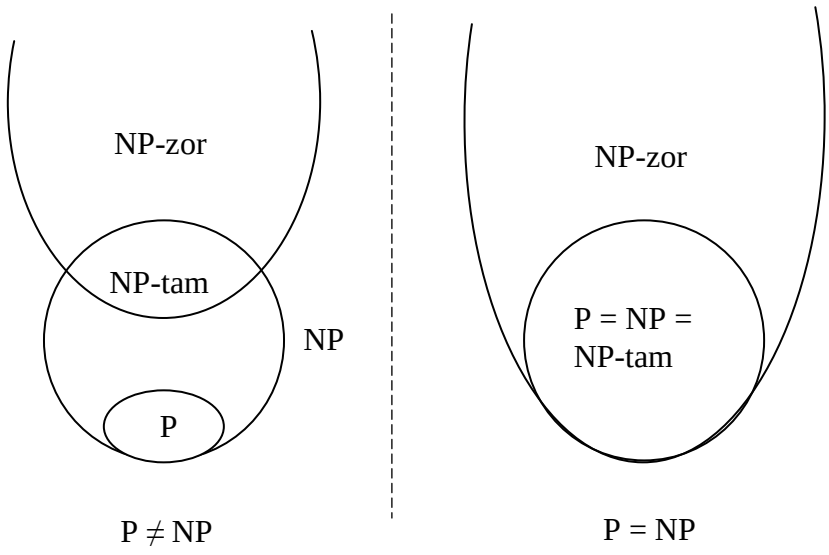
Bir algoritmanın çalışması için gerekli zaman, zaman karmaşıklığı (time complexity); gerekli bellek alanı ise yer karmaşıklığı (space complexity) hakkında

bilgi verir. Her iki karmaşıklık tipi de ‘O ( )’ notasyonuyla gösterilmektedir. Ancak bu notasyonunun kullanıldığı durumlarda, aksi belirtilmemişse, zaman karmaşıklığı ifade edilmektedir.

Karmaşıklık teorisinde, bazı tip problemlerin çözümü için en etkili algoritmaların; çalışma süresinin girilen verinin büyüklüğüne bir polinom cinsinden bağlı olduğu bilinmektedir (buna polinomsal zamanda çalışan algoritma adı verilir), bu tür problemler P kategorisindeki problemlerdir. Mesela verilen n basamaklı bir sayının asal olup olmadığını kontrol etmek için çalışma süresi n^6 mertebesindeki bir polinomla hesaplanabilen bir algoritma vardır. Dolayısıyla verilen bir sayının asal olup olmadığının araştırılması P kategorisinde bir problemdir. Buna karşılık bir diğer grup problem vardır ki bunlar için sorulan soruya girilen verinin büyüklüğüne polinom mertebesinde bağımlı bir sürede cevap verecek bir algoritma bilinmemektedir. Fakat bu tür bazı problemler için eğer bir şekilde cevabı tahmin edebiliyorsak, tahminimizin doğruluğunu sınamak için veri büyüklüğüne polinom mertebesinde bağımlı sürelerde çalışacak algoritmalar vardır. Tahminin doğruluğunun kontrolü için çalışma süresi, verinin büyüklüğüne polinom cinsinden bağımlı bir algoritma olan bu tür problemler de NP kategorisini oluştururlar. Örnek olarak verilen n basamaklı bir sayının asal çarpanlarının neler olduğu sorusunu düşünebiliriz. Bu sorunun cevabı için bilinen en iyi algoritmanın çalışma süresi n sayısına bir polinom cinsinden değil de üstel fonksiyonlar cinsinden (örneğin e^n) bağımlıdır (buna üstel zamanda çalışan algoritma denir), fakat bu problem için eğer bir şekilde cevabı tahmin edebiliyorsak tahminimizin doğruluğunu sınamak için n sayısına polinom mertebesinde bağımlı bir sürede çalışacak bir algoritma mevcuttur. Dolayısıyla verilen bir n basamaklı sayının asal çarpanlarının neler olduğu sorusu NP kategorisindedir. Bu iki kategoriden NP’nin P’yi içerdiğini görmek kolaydır. Eğer bir sorunun cevabını verinin büyüklüğüne polinom mertebesinde bağımlı sürede çalışacak bir algoritmayla bulabiliyorsak, bu soruya cevap olarak üretilmiş bir tahminin doğruluğunu da verinin büyüklüğüne polinom mertebesinde bağımlı sürede çalışacak bir algoritmayla kontrol edebiliriz. Bunun için verilen sorunun cevabını verecek algoritmayı çalıştırıp, onun verdiği cevabı kendi tahminimizle karşılaştırmak yeterlidir. “P=NP?” problemi bunun tersinin de doğru olup olmadığını sorar. Yani NP kategorisinde olup da P kategorisinde olmayan problemler var mıdır? Veya diğer

bir dille asal çarpanların bulunması için polinom mertebesinde bir sürede çalışacak bir algoritma gerçekten yok mudur yoksa vardır da biz mi bulamıyoruz? Bu alanın uzmanlarının çoğunun görüşü bu tür algoritmaların gerçekten de var olmadıkları için bulunamadığı (yani P nin NP'ye eşit olmadığı) şeklindedir ancak bu soruya kesin bir cevap verilebilmesi şimdilik çok zor görünmektedir[11].

NP problemler zorluk seviyelerine göre derecelendirmeye tabi tutulmaktadırlar. Bazı problemler ise NP olmasalar da, zorluk açısından NP ile karşılaştırıldıklarından NP önadıyla anılan gruplarda yer almaktadırlar. NP-ailesi sınıflandırma sistemi Şekil 2.3'de görüldüğü gibi kafa karıştırıcı bir yapıya sahiptir. NP-zor (NP-hard) problemlerin hepsi, NP ön adını alsalar da NP değildirler. NP-zor olup, NP grubuna girenleri NP-tam (NP-complete) olarak isimlendirilir. Aslında NP ön adı NP ailesi ile zorluk ilişkisini gösterir. NP-tam problemler tam olarak NP içinde olan ve bu grupta çözülmesi en zor olan problemlerdir. NP-zor problemler ise çözülmesi en az NP kadar zor olan ancak NP içinde olmak zorunda olmayan problemlerdir[12].



Şekil 2.3: P,NP,NP-tam ve NP-zor için küme çizimi[7]

Gezgin Satıcı Problemi NP-zor grubunda yer almaktadır. Problemin karar versiyonu (uzaklıklar ve bir x rakamı verilirse, x'ten daha kısa bir tur var mıdır?) NP-tam grubunda, yani çözülmesi en zor gruptadır[13].

2.3. Kesin Çözüm Yöntemleri

Bu bölümde Gezgin Satıcı Problemine ait küresel en iyi çözümü elde etmek amacıyla kullanılabilecek kesin çözüm yöntemleri anlatılmaktadır. Yöneylem araştırması alanındaki GSP'ye benzer ya da farklı özellikteki bir çok problemin çözümde de kullanılabilen yöntemlerden bir kısmının, GSP çözümünde daha etkin kullanılabilmesi amacıyla özel yaklaşımlar geliştirilmiştir. Bu özel yaklaşımlar diğer yöneylem araştırması problemleri için doğrudan katkı sağlamamalarına rağmen, problemlerin çözümü için farklı bakış açıları önermektedirler. Ayrıca belirli bir kısmı oldukça benzer yapıdaki araç çizelgeleme problemleri gibi ağ problemlerinin çözümüne doğrudan katkı sağlayabilmektedirler.

Dal ve sınır, kesme düzlemi gibi yöntemlerde en iyi çözüme ne kadar yaklaşıldığının belirlenmesi, çözümün en iyi olduğunun ispatlanması, çözüm sürecinin etkinliğinin arttırılması gibi amaçlar için çözüme ait amaç değerinin belirli değerin altında olmayacağı ve üstüne çıkamayacağını gösteren alt sınır ve üst sınırın belirlenmesine ihtiyaç duyulmaktadır. Bu iki değerin kesişmesi durumunda elde edilen çözümün en iyi olduğuna ait kanıt elde edilmiş olur. GSP gibi en küçükleme problemleri için üst sınır değeri iyi bir sezgisel yardımıyla elde edilirken, alt sınır değeri problemin gevşetmesine ait amaç değeri ile belirlenir. Çözüm süreci bu iki değeri yaklaştırmaya yönelik olarak sürdürülür.

2.3.1. Sayma yöntemi

GSP için küresel en iyiyi garanti eden akla gelen ilk yöntem bütün olası permütasyonların denenmesidir. Ancak n adet şehrin bulunduğu bir ağda, değerlendirilecek tur sayısının n! ile orantılı olarak büyümesi nedeniyle, yöntemin zaman karmaşıklığı O(n!) olarak belirlenir. Bu nedenle, az sayıda şehrin bulunduğu ağlarda bile, yıllarca süren bilgisayar hesaplamaları gerekmektedir.

Örneğin 20 şehrin bulunduğu bir ağda (n-1)!/2 = 60.822.550.204.416.000 adet farklı gezgin satıcı turu mevcuttur. Her bir turun uzunluğunun belirlenmesi ve mevcut en

iyi çözümden daha iyi olup olmadığının belirlenmesi için sadece tek bir işlem yapıldığı ve 3 Ghz işlemci hızına sahip bir bilgisayarda çalışıldığı düşünülürse, denemelerin yaklaşık olarak 234 gün süreceği hesaplanır. Ancak, 21'inci şehirde bu sürenin 20 ile; 22. şehirde 20*21 ile çarpılması ve sayı arttıkça şehir sayısının bir eksiğiyle çarpılarak gerekli sürenin hesaplanması gerekmektedir. Ayrıca gerçek işlem süresinin, tek bir sıralamayı denemek için 'n' ile orantılı sayıda işlem yapılması gerektiğinden, bu sürenin en az 'n' katı olduğu düşünülürse, bu yöntemin makul sürede çözüm önermediği kolayca anlaşılabilir.

2.3.2. Dinamik programlama

Dinamik programlama yöntemi tam saymaya benzer şekilde, bir tür sayma mantığı ile çözüme ulaşmaktadır. Ancak sayma sırasında gerçekleştirilen tekrarlı hesaplamaların, işlemsel yükün bir kısmını bellek yüküyle değiştirmek suretiyle, önüne geçerek çok daha kısa sürelerde çözüm elde edilmesini sağlamaktadır.

GSP'nin dinamik programlama yöntemi kullanılarak çözümü için Held ve Karp [14] tarafından önerilen yaklaşım bu alanda, bütün GSP'leri çözebilecek bir yaklaşım olarak bilenen en iyi zaman karmaşıklığına sahiptir. $O(n^2 2^n)$ karmaşıklık ile küresel en iyiyi garanti eden yaklaşım, O(n!) karmaşıklık derecesine sahip sayma yönteminden çok daha iyi bir performans göstermesine karşın yine de polinomsal performans göstermekten oldukça uzaktır.

for: Bütün j ler için $C(\{1,j\}, j) = d_{1j}$.
for: s = 3 den n'e kadar (alt kümlerin büyüklüğü burada belirlenmektedir)
    for: S'in n büyüklüğünde ve 1'i içeren bütün alt kümeleri için,
        for: Bütün $j \in S, j \neq 1$ için,
            $\{ C(S,j) = \min_{i \neq j,\, i \in s} [C(S - \{j\}, i) + d_{ij}] \}$
en iyi $= \min_{j \neq 1} [C(\{1,2, \ldots, n\}, j) + d_{j1}$.

Şekil 2.4: GSP'nin dinamik programlama ile çözüm algoritması

S'in 1 şehrini ve farklı en az bir adet şehri içeren bir alt küme olduğu ve j'nin S içinde 1'den farklı bir şehir olduğu kabul edilsin. C(S;j), 1'den başlayan S içindeki bütün şehirleri dolaşan ve j'de biten en kısa yolu ifade etsin. Bu durumda i ile j

şehirlerini birbirine bağlayan yolun uzunluğu d_{ij} olmak üzere, en kısa tur Şekil 2.4'te görülen algoritma kullanılarak belirlenebilir.

Algoritmada her noktaya ulaşmak için geçerli en kısa yol ve uzaklık hafızada tutularak bu bilgiye ait yol her adımda bir şehir geliştirilir. İlk adımda bütün j noktalarına 1 noktasından en kısa ulaşım uzaklığı olarak (1,j) uzunluğu atanır. Daha sonraki adımda ise öncelikle yoldaki şehir sayısı bir arttırılarak üçe çıkarılır ve üçüncü j noktasına en kısa ulaşım uzaklığı ve yolu belirlenmeye çalışılır. Bunun için daha önce her j için hesaplanmış olan bütün diğer noktaların yol uzunluklarına üçüncü şehre uzunlukları eklenerek en kısa olanı belirlenir ve o şehir için en kısa yol ve uzunluğu olarak kaydedilir. Bu şekilde bütün şehirler yollara eklenınceye kadar devam edilir. En son aşamada mevcut en son şehirden başlangıç şehri olan 1'e uzunlukları da yol uzunluğuna eklenerek en kısa tur ve uzunluğu belirlenir.

2.3.3. Dal ve sınır yöntemi

Dal ve sınır, bir probleme ait gevşetilmiş çözümden yola çıkarak, dallandırmalar yaparak çözüm uzayının bölümlendirilmesi ve böylece sıkılaştırılarak araştırılması yoluyla istenen çözüm bulunmasını sağlayan bir yöntemdir. Bir eniyileme probleminin gevşetilmiş hali, sahip olduğu kısıtlar nedeniyle asıl problemin tüm olurlu bölgesini kapsayan ancak amaç fonksiyonu asıl problemin değerinin altında veya ona eşit çözümler elde eden modellemesi olarak ifade edilmektedir[15]. Örneğin tamsayılı çözüm elde edilmesi istenen bir doğrusal modelin, tamsayı kısıtları göz önüne alınmadığında gevşetilmiş çözümü elde edilir. Kısıtlama olmaksızın yapılan arama, tamsayı çözümler uzayını da içeren ancak daha geniş, gevşetilmiş, bir uzayda gerçekleştirilir. Bu durumda elde edilen sonucun tamsayı değerlerden oluşması mümkündür. Ancak bazı değişkenlerin bu kısıtı sağlamaması durumunda bu değişkenlere ait dallandırmalar yoluyla çözüm uzayının daraltılması ve istenen tamsayılı, sıkı çözüme ulaşılması mümkün olur.

GSP'de de gevşetilmiş çözüm, gezgin satıcı turu oluşturmayan çözümler içerebilmektedir, ancak elde edilen amaç değeri GSP çözümü için bir alt sınır ifade etmektedir. Gevşetilmiş çözüm, en iyi gezgin satıcı turunu engellemeyecek şekilde

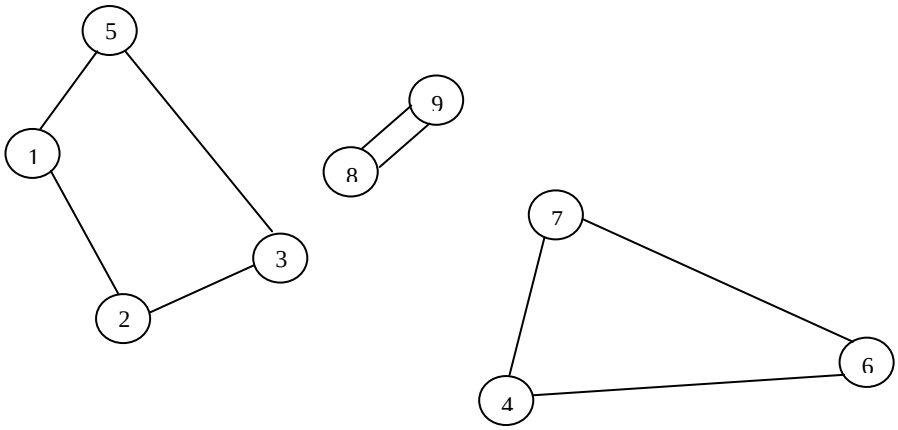
adım adım dallandırma ve her dalda yeni kısıt ekleme yoluyla daraltılarak en iyi çözüm bulunur. Bu işlem sırasında en kısa tur uzunluğu için alt sınır ifade eden mevcut çözümün amaç değeri, adım adım yükselerek en son adımda en kısa tur uzunluğunun kendisine eşit olur.

Gezgin Satıcı Problemi için dal ve sınır yöntemi kullanılarak yapılmış bir çok çalışma mevcuttur. Çalışmalar, genel olarak Atama Problemi ya da En Küçük Yayılan Ağaç problemlerini gevşetilmiş problem olarak alarak çözüm elde etmektedirler. Diğer birçoklarının yanında Eastman [16], Little, Murty, Sweeney ve Karel [17], Shapiro[18], Bellmore ve Malone [19], Smith, Srinivasan ve Thompson [20], Carpaneto ve Toth[21] Atama Problemini; Held ve Karp[22,23], Christofides[24], Helbig, Hansen ve Krarup[25], Smith ve Thompson[26], Volgenant ve Jonker[27], Gavish ve Srikant[28], En Küçük Yayılan Ağaç Problemini gevşetme olarak kullanarak Gezgin Satıcı Problemi üzerinde çalışmışlardır.

Atama Problemi, Gezgin Satıcı Probleminde olduğu gibi her noktanın bir başka noktaya bağlanmasının gerektiği ve bunun en küçük maliyetle gerçekleştirilmesinin gerekli olduğu bir problemdir. Macar algoritması kullanılarak $O(n^3)$ karmaşıklıkta polinomsal olarak çözüme ulaştırılması mümkündür. Ancak, atama problemi tek bir tur bulunmasını öngörmediğinden elde edilen çözümde Şekil 2.5'te görüldüğü gibi alt turlar oluşabilmektedir. Bu durumda alt turda bulunan bağlardan birinin '1' ya da '0' değerini alması, yani atama çözümünde bulunması ya da bulunmaması durumu değerlendirilerek dallandırma yapılabilir. Bir bağın çözümde bulunmasını önlemek için o bağa ait ulaşım maliyeti, seçilmemesini sağlayacak şekilde arttırılır. Bu şekilde alt tur oluşumu engellenerek tek bir tur elde edilinceye kadar çözüm ilerletilerek en küçük maliyetli Gezgin Satıcı Turu belirlenir.

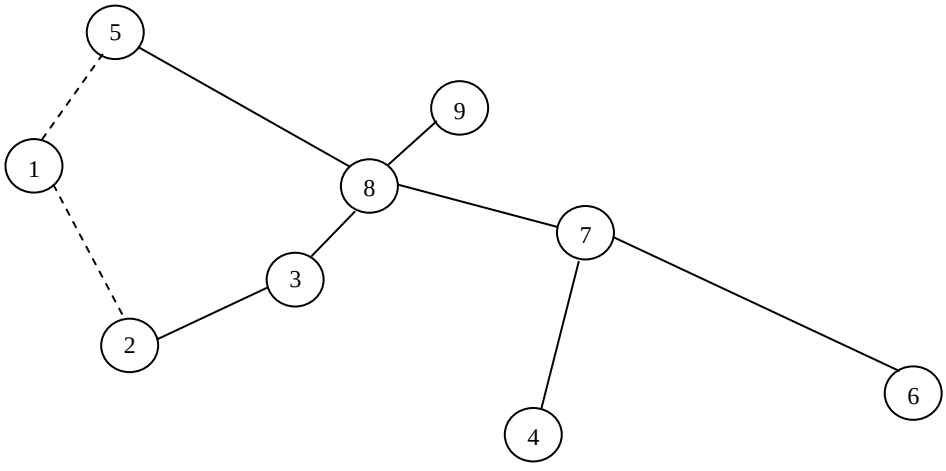
En Küçük Yayılan Ağaç Probleminde de amaç Gezgin Satıcı Probleminde olduğu gibi bütün noktalara en az taşıma maliyetiyle yayılmaktır. Şekil 2.6'da görülen 1-ağacı (1-tree) olarak isimlendirilen yayılan ağaç türünde öncelikle 1 şehri dışında kalan şehirler için en küçük yayılan ağaç belirlenir daha sonra 1 şehri ile mevcut yayılan ağacı birleştiren en kısa iki bağ belirlenerek ağaca eklenir. Yayılan ağaç çözümünün yapılabilmesi için Kruskal ya da Prim tarafından önerilen yöntemlerin

kullanılması mümkündür[29]. Bu şekilde elde edilen çözümler Gezgin Satıcı Turlarını da içermekle birlikte yayılan ağaç mantığıyla bir noktadan ikiden çok bağın çıkması da mümkündür. Bu durumun engellenmesi amacıyla atama probleminde olduğu gibi dallanmaların gerçekleştiği noktalara bağlanan yollar için dalsınır algoritması kullanılarak çözüm elde edilebilmektedir.



Şekil 2.5: Alt Turlar

Küresel en iyi çözümü bilinen problemler için, atama problemi ile elde edilen alt sınır değeri, en kısa tur uzunluğuna 1- ağacı ile elde edilen alt sınır değerinden daha yakın olduğundan atama problemi ile elde edilen gevşetilmiş çözüm, 1-ağacı ile elde edilen çözümden daha sıkı olarak kabul edilmektedir. GSP için dal-sınır yönteminin kullanımıyla ilgili dallandırma stratejileri v.b. konular ile ilgili ayrıntılar için Lawler'a [30] başvurulabilir.



Şekil 2.6: 1-ağacı (1-tree)

2.3.4. Doğrusal çözüm yaklaşımları

Gezgin Satıcı Probleminin çözümü amacıyla geliştirilen doğrusal yaklaşımları iki ayrı modele dayandırmak mümkündür. Bu modellerden ilki Dantzig-Fulkerson-Johnson (DFJ) [31] tarafından önerilmiştir ve problemi alt tur engelleyici kesmeler yardımıyla iteratif olarak çözüme ulaştırmaktadır. Diğer model ise Miller-Tucker-Zemlin (MTZ) [32] tarafından ortaya koyulan, problemin çözümü için gerekli kesme eşitsizlikleri başlangıçta belirli olan yaklaşımdır.

2.3.4.1. Dantzig-Fulkerson-Johnson (DFJ) modeli

Dantzig ve arkadaşları tarafından Gezgin Satıcı Probleminin çözümü için önerilen model, atama problemi gevşetmesi kullanıldığında ortaya çıkan alt turların oluşumunun engellenebilmesi için, önleyici kısıt kullanımı mantığına dayanmaktadır. Bu önleyici kısıtlar daha sonra kesme düzlemi olarak isimlendirilecektir. Buna göre; n adet şehirden oluşan her GSP örnek problemi n(n-1)/2 adet bileşenden oluşan ve c ile gösterilebilecek bir maliyet vektörü ve yine n(n-1)/2 adet bağı ile ifade eden x vektörü ile ifade edilebilir. Eğer S bütün turların bağ vektörünü gösteriyorsa, çözüm aşağıdaki problemin çözümü ile bulunabilir:

$$\min c^T x \text{ , öyleki } x \in S \qquad (2.1)$$

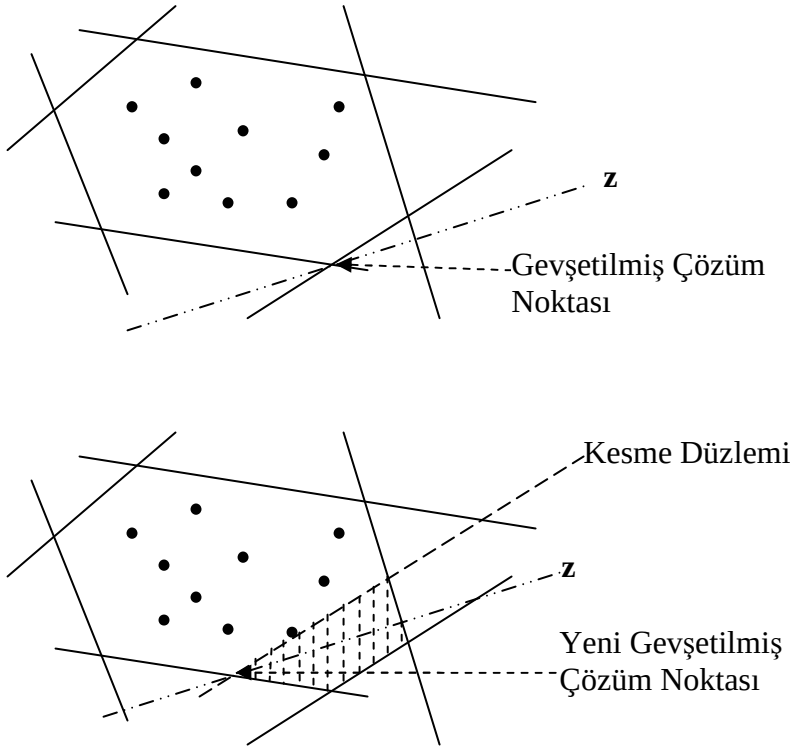
Ancak burada S kümesinin matematiksel olarak ifade eden, yani çözümün tur oluşturmasını sağlayacak kısıtların belirlenmesi sorunu ortaya çıkmaktadır. Böyle bir noktada çözmek istediğimizi değil ona yakın olabilecek bir problemi çözmemiz de fikir verebilir. ‘A’ değişken katsayılarını, ‘b’ ise sağ taraf değerini ifadeden vektör olmak üzere aşağıdaki gibi bir doğrusal model kullanılarak çözüm elde edilebilir:

$$\min c^T x \text{ , öyleki } Ax \leq b \qquad (2.2)$$

Burada, $\{ x:Ax \leq b \}$ şeklinde tanımlanan P çokyüzlüsü S’i içermektedir ve sınırlıdır. (2.2) eşitsizliğinin çözülmesi durumunda S’in elemanı olan bir x* çözümü bulunmuşsa, bu çözüm optimaldir. Aksi takdirde x* çözümünü dışarıda bırakan

ancak S'teki tüm noktaları içeren yeni eşitsizliklerin çözüme eklenmesi gerekmektedir. Dantzig ve arkadaşları tarafından ilk defa ortaya atılan bu eşitsizlikler S'in elemanı olan yeni bir x* çözümü bulunabilmesi için çözüm uzayını daraltmaktadırlar. Bu çeşit eşitsizliklere kesme, kesme düzlemi ya da alt tur engelleme kısıtı adı verilmektedir.

Kesme düzlemi denildiğinde akla Gomory [33] tarafından önerilen tam sayı çözümler elde edilmesi amacıyla önerilmiş olan kesme düzlemi yöntemi gelmektedir. Gomory tarafından önerilen kesme düzlemi, gevşetilmiş (tam sayı kısıtları olmaksızın elde edilen) çözümde istenen değişkenlerin tam sayı değerini almaması durumunda final simpleks çözümden türetilerek ve bu çözüme eklenerek, tam sayılı çözümlerin elde edilmesini sağlamaktadır. Gomory kesmeleri gevşetilmiş çözüm uzayını tam sayılı çözümler elde etmek amacıyla daraltmakta, tam sayılı olmayan çözüm uzayının bir kısmını keserek çözüm uzayının dışında bırakmaktadır. Elde edilen çözüm istenen tamsayı kısıtlarını sağlayana kadar Gomory kesmeleri eklenmeye devam etmektedir.



Şekil 2.7: Kesme Düzlemi Yöntemi

Gezgin Satıcı Problemi (GSP) için önerilen kesmeler Şekil 2.7'de görüldüğü gibi gevşetilmiş problem çözümünden Gezgin Satıcı Turu yapısında olmayan çözümleri (taralı alan) keserek dışarıda bırakmaktadır.

Karar değişkenleri x_{ij}, i noktasından j noktasına yapılan bağı; c_{ij} bağ uzunluklarını göstermek üzere, gezgin satıcı tur uzunluğunu gösteren amaç değeri aşağıdaki gibi basit doğrusal bir formda ifade edilir:

$$x_{ij} = \begin{cases} 1, & \text{eğer i noktasından j bağ var ise} \\ 0, & \text{aksi durumda} \end{cases}$$

$$Z_{\min} = \sum_{i} \sum_{j>i} c_{ij} x_{ij} \tag{2.3}$$

Tamsayılı gezgin satıcı problemlerini karmaşıklaştıran şey modelin kısıtlarıdır. Biraz dikkatli düşünmek sistemi açık ve anlaşılır hale getirecektir. Simetrik bir problem için, uygun bir çözümde, herhangi bir i noktasına ilişkin iki x değişkeni l değerini alabilir. Bu bağlantılardan birisi i'nin güzergahtaki önceki şehirle, diğeri ise i'nin sonraki şehirle bağlantısını kurar. Yani bir noktaya gelen bir bağ olması gerektiği gibi, noktadan çıkan bir bağ olması gerekmektedir. Bu gerekliliği matematiksel modele eklediğimizde modelin tamamı (2.4) denkleminde görüldüğü gibi ifade edilir:

$$Z_{\min} = \sum_{i} \sum_{j>i} c_{ij} x_{ij}$$

A.K.A.

$$\sum_{i} x_{ij} = 1 \quad (i = 1...n) \tag{2.4}$$

$$\sum_{j} x_{ij} = 1 \quad (j = 1...n)$$

$x_{ij} = 0$ veya 1

Atama problemiyle aynı olan bu gevşetilmiş çözüm, alt bölüm 2.3.3'de anlatıldığı ve Şekil 2.5'te görüldüğü gibi her durumda istenen çözümü vermemekte, alt turların

oluşmasını önleyememektedir. Danzig ve arkadaşları tarafından öne sürülen alt tur engelleme kısıtları aşağıdaki gibi ifade edilebilir. K, ziyaret edilecek noktaların/şehirlerin uygun alt kümesi olmak üzere;

$$\sum_{i,j \in S} x_{ij} \leq |K| - 1 \qquad 2 \leq |K| \leq n-1 \tag{2.5}$$

Bu kısıt, oluşan alt turların her birinin kendi içinde yaptığı bağların sayısının, o alt turda yer alan nokta sayısının toplamından en az bir eksik ya da ona eşit olması gerektiğini ifade eder, bu duruma sadece S alt kümesinin tüm noktaları kapsaması yani gezgin satıcı turunun oluşması durumunda izin verilir. Bu şekilde elde edilen sıkılaştırılmış x* çözümü S kümesine ait ise çözüm sonlandırılır. Ancak aksi takdirde yeni çözüme göre gerekli incelemeler yapılarak, yeni kesmeler eklenir ve çözüme devam edilir. Daha sonraki çalışmalarda Dantzig ve arkadaşları tarafından sunulan alt tur engelleme kısıtlarına benzer şekilde gezgin satıcı turu oluşturmayan çözümlerin, çözüm uzayından çıkarılması amacıyla kullanılan birçok kesme düzlemi eklenmiştir. Tarak (comb), klik ağacı (clique), yıldız (star), yol (path), merdiven (ladder) ve domino eşitsizlikleri bu alanda en çok bilinen eşitsizliklerdendir. GSP kesme düzlemleri ile ilgili ayrıntılı bilgi için Gutin ve Punnen'e [9] başvurulabilir.

Yukarıda sunulmakta olan modelde görüldüğü x değişkenleri 0 ya da 1 değerini alması gerekmektedir. Şehir sayısının giderek arttığı problemlerde tamsayılı çözümleri etkin bir şekilde yapacak bir yönteme ihtiyaç duyulmaktadır. Final simpleks tablo üzerine yeni kesmeler ekleyerek tam sayılı çözümler elde edilmesini sağlayan Gomory kesmeleri küçük problemlerde başarı sağlamakla birlikte, artan problem büyüklükleri karşısında yetersiz kalmaktadır. Bu durumda tamsayı kısıtlarının yerine $0 \leq x_{ij} \leq 1$ kısıtı koyularak, kesme ve dal-sınır yöntemlerinin karışımından oluşan dallandır ve kes (branch and cut) yöntemi ortaya konulmuştur.

Kesme düzlemleri (alt tur engelleme kısıtları) eklenerek, tam sayı kısıtları olmaksızın yapılan çözümlerden elde edilen amaç değeri giderek artmakta ancak belli bir süre sonra, meydana gelen artış oldukça küçük düzeylere ulaşabilmektedir. Böyle bir

durumda, çözümde henüz tamsayı olmayan değişkenler (x^e) mevcut ise bu değişkenler için, $x^e \leq 0$ ve $x^e \geq 1$ kısıtları eklenerek dallanmalar yapılabilir, böylece tamsayılı çözüm elde imkanı doğmaktadır. Bu tür dallandırmaları istenilen tamsayılı çözüm elde edilinceye dek devam ettirmek mümkündür. Bir adım ötesinde ise bir dallanma sonrasında elde edilen çözüme yeniden kesme değişkenleri eklenmesi yer almaktadır. Yöntemde, dallanma ve kesmenin ne zaman yapılacağı, ne tür kesme kullanılacağı, kesme üretim yöntemleri gibi faktörler başarımı etkilemektedir.

Padberg ve Rinaldi [34-36], Grotschel [37], Crowder ve Padberg [38], Grötschel ve Holland [39], Chvatal [40], Grötschel ve Pulleybank [41], Applegate ve arkadaşları [42-45] dalandır ve kes yöntemi kullanılarak GSP için çözüm aramışlardır. Yöntem GSP dışındaki problemlerde de sınırlı da olsa [46] kullanılabilmektedir.

2.3.4.2. Miller-Tucker-Zemlin (MTZ) modeli

Miller ve arkadaşları [32] tarafından sunulan GSP'nin tamsayılı çözümünü öneren model aşağıda görüldüğü gibidir. Modelde problem sınırlarını belirleyen kesme düzlemi denklemlerini hepsi ilk aşamada yer almakta ve tam sayı kısıtları ile çözüm gerçekleştirilmektedir. Burada şehirlerin sıralarını belirten u_i yapay değişkenleri dışarıdan eklenmektedir.

$$Z_{\min} = \sum_{i} \sum_{j>i} c_{ij} x_{ij}$$

A.K.A.

$$\sum_{i} x_{ij} = 1 \quad (i = 1...n)$$

$$\sum_{j} x_{ij} = 1 \quad (j = 1...n) \tag{2.6}$$

$$u_i - u_j + n\,x_{ij} \leq (n-1), \quad (i = 1...n)\,(j = 1...n)$$

$$1 \leq u_i \leq n-1$$

$x_{ij} = 0$ veya 1

Asimetrik ve simetrik problemlerin her ikisi için de kullanılabilen bu modelde n(n-1) adet ikili değişken ve n adet sürekli değişken bulunmaktadır. Sadece n^2 sabit değişkenin bulunduğu MTZ modelinin, DFJ modeline göre avantajı kısıt sayısının baştan belli olması ve modelin tek çözümde sonuca ulaştırmasıdır. Ancak n(n-1) adet ikili değişken problem büyüdükçe, DFJ modelinde olduğu gibi Gomory kesme yöntemi yetersiz kalarak çözüm süresinin uzamasına neden olmaktadır. Desrochers ve Laporte [47] MTZ modelinin biraz daha sıkılaştırılmış (2.7) denkleminde görülen halini sunmuşlardır.

$$Z_{\min} = \sum_{i} \sum_{j>i} c_{ij} x_{ij}$$

A.K.A.

$$\sum_{i} \sum_{j} x_{ij} = 1 \quad (i = 1...n)\,(j = 1...n)$$

$$u_i - u_j + (n-1)\,x_{ij} + (n-3)\,x_{ij} \le (n-2), \quad (i = 2...n)\,(j = 2...n)(i \ne j) \qquad (2.7)$$

$$u_i \ge 1 + (n-3)x_{i1} + \sum_{\substack{j=2 \\ j \ne i}} x_{ji}\,, (i = 2...n)$$

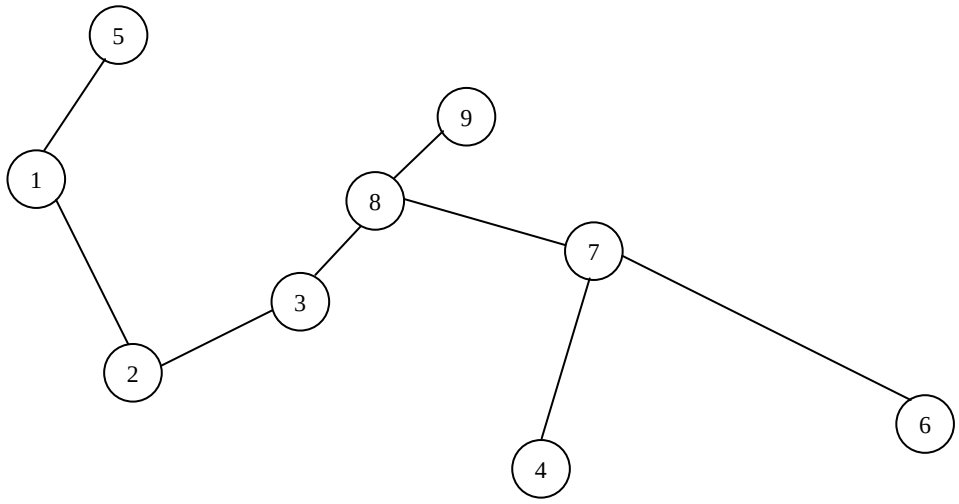
$$u_i \ge n - 1 - (n-3)x_{1i} - \sum_{\substack{j=2 \\ j \ne i}} x_{ij}\,, (i = 2...n)$$

$x_{ij} = 0$ veya 1

2.4. Sezgisel Çözüm Yöntemleri

Literatürde bir çok gezgin satıcı problemi için geliştirilmiş bir çok sezgisel yer almaktadır. Bunun nedeni sezgisellerin, analitik yöntemlerin aksine polinomiyal olmayan bu tür problemler için istendiği gibi polinomiyal zamanda çözüm önermeleridir. Gezgin satıcı problemine benzeyen bir problem olan “Minimum Yayılan Ağaç (MYA) Problemi” sezgisel bir yöntem kullanılarak küresel en iyi çözüme ulaştırılabilmektedir. Şekil 2.8’de görüldüğü gibi kısaca bir ağ sistemindeki bütün noktalara ulaşan en kısa yolun bulunması, olarak tanımlanan problem Kruskal ya da Prim tarafından önerilen basit sezgisellerden herhangi biri ile küresel en iyi çözüme ulaştırılmaktadır[29]. Bu problemin, çözüm uzayında gezgin satıcı probleminden çok daha fazla sayıda alternatif çözüm bulundurmasına karşın bu denli

basit şekilde çözüme ulaştırılması GSP ve benzeri problemler için, polinomiyal zamanda çözüm üreten bir çözümün olabileceği umudunun oluşmasına neden olmaktadır. Bunun yanında sezgisel çözümler küresel en iyi çözümü garanti etmeseler de, polinomiyal zamanda kabul edilebilir derecede en iyi çözüm değerine yaklaşabildikleri durumda, zaman kısıtının bulunduğu uygulamalar için vazgeçilmez hale gelmektedirler.



Şekil 2.8: Minimum Yayılan Ağaç

GSP problemi için literatürde birçok sezgisel yöntem önerilmiştir. Bu yöntemlerin hepsini anlatmak mümkün değildir. Ancak en çok bilinen birkaç yöntem burada yer almaktadır. GSP için geliştirilen sezgisel yöntemler (i) tur oluşturma yöntemleri (ii) tur geliştirme yöntemleri olarak iki grupta toplanabilir. Adından da anlaşılacağı gibi ilk grup, bazı kriterleri göz önüne alarak sırayla noktaları birbirine bağlamakta ve en iyi çözümü oluşturmaya çalışmaktadır. Aşağıda bahsedilen en yakın komşu, en yakın ekleme, geometrik, Bovurka, Christofides ve greedy gibi yöntemler bu gruba girmektedir. İkinci grup ise mevcut bir çözümün çeşitli hareketler yoluyla geliştirilmesinin hedeflendiği yöntemlerdir. Aşağıda anlatılan 2-opt, 3-opt gibi r-opt denemeleri, hyperopt, S&C ve Lin-Kerninghan yöntemleri bu gruba girmektedir. Bu iki grubun özelliklerini bir arada bulunduran birleşik yöntemler de mevcuttur.

2.4.1. En yakın komşu yöntemi

Bu yöntemde, var olan şehirler içerisinden rasgele bir şehir gezgin satıcı turu başlangıcı olarak seçilir ve bu şehre en yakın şehir turdaki sonraki adım olarak belirlenir. Ekleme işlemine bütün şehirler tura katılıncaya kadar devam edilir. Tüm şehirler tura eklenmeden önce turun kapanmasına izin verilmez. Oldukça basit ve akla yakın gelen yöntemde, hangi şehrin başlangıç olarak seçildiği ve mevcut yolun hangi ucundan devam edildiğini farklı başarımda sonuçlar elde edilebilmesine neden olabilmektedir. Ancak genellikle en yakındaki şehrin seçimi sırasında geride unutulan diğer şehirlerin tura katılması için uzun geri dönüşler gerektiğinden küresel en iyiden uzak çözümler elde edildiği görülmektedir.

2.4.2. En yakın ekleme yöntemi

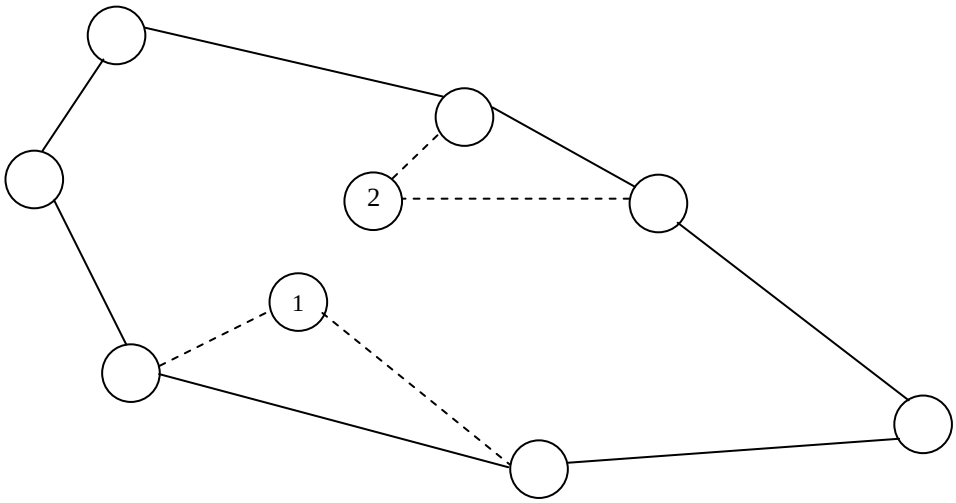
İki şehir arasındaki bir tur ile çözüme başlanır. Daha sonra bu tura en yakındaki şehirlerin, sırayla mevcut turdaki şehirlerin arasına girerek tura eklenme durumları denenerek, turdaki mümkün olan en küçük büyüme bulunur ve bu şehir tura eklenir. Bu işlem bütün şehirler tura ekleninceye kadar devam ettirilir. Bu yöntemde de başlangıç şehrinin seçimi, ekleme sırası kuralı ve elde çözümü değiştirebilmektedir.

En yakın ekleme ve en yakın komşu yöntemleri genellikle başlangıç çözüm elde edilmesi amacıyla kullanılır. Ancak DePuy ve arkadaşları [48] bu iki yöntemden yararlanarak GSP ve benzeri kombinasyonel problemlerin çözümü için kullanılabilecek bir meta sezgisel önermişlerdir. Meta sezgisel olarak nitelendirilmesinin nedeni diğer problemler için de kullanılabilmesi ve stokastik yapı içerdiğinden çözümün başarımının rasgelelik yardımıyla garanti edilmeye çalışılmış olmasıdır.

2.4.3. Geometrik yöntem

Şekil 2.9'da görüldüğü gibi düzlemsel olarak yerleşmiş olan şehirlerin en dışında bulunanları dışbükey olarak kaplayacak biçimde mevcut bağlarla bağlanır. Daha sonra çizilen bölgenin içinde kalan şehirler içerisinde dış bölgedeki ardışık şehirlere

yapılan bağlardan en geniş açıya sahip olanı tura eklenir. Bu ekleme bütün şehirler tura dahil olana kadar devam edilir [29].



Şekil 2.9: Geometrik yöntem

2.4.4. Greedy yöntemi

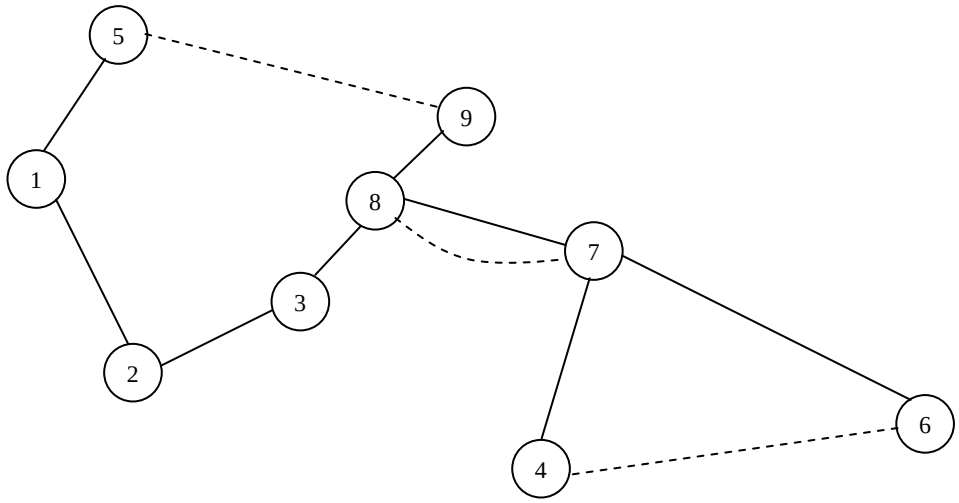
Olası bütün bağların küçükten büyüğe doğru sıralanması ile yönteme başlanır. Daha sonra derecesi zaten iki olan noktalara tekrar bağlanmaksızın ve bağ sayısının N'e eşit olmasından önce turun kapatılmasını önleyerek, en küçükten başlayarak, sırayla bağların eklenmesi yoluyla bir tur oluşturulur [9].

2.4.5. Bovurka yöntemi

Greedy yöntemine benzeyen bu yöntemde her şehre en yakın komşu uzunluğu belirlenir. Daha sonra belirlenen bağlar artan şekilde sıralanır. Bağlar sırayla alt tur oluşturmayacak ve bir şehre ikiden fazla bağ yapmayacak şekilde tura eklenir. Tura ekleme yapılırken bir şehre iki bağ yapılmış ise bu şehirle kurulabilecek olası diğer bağlar listeden çıkartılır. Yapılan ilk sıralamadaki bütün bağlar bittiğinde büyük olasılıkla bir tur tamamlanmamış olacaktır. Bu durumda bağ yapması halen mümkün olan şehirler için en yakın olası bağlar belirlenip sıralanarak önceki bölümde olduğu gibi tura ekleme yapılmaya devam edilir. Tam bir tur oluşturuluncaya kadar işleme devam edilir [9].

2.4.6. Christofides algoritması

Algoritma üçgen eşitsizliğini sağlayan GSP için Minimum Yayılan Ağaç (MYA) çözümü temel almaktadır. MYA, çözümünde derecesi tek olan noktalar arasında, her noktaya sadece tek bağın yapıldığı G (mükemmel eşleşme-perfect matching) ağı oluşturulur. G ağı ile başlangıçtaki MYA çözümü birleştirerek M ağı (Şekil 2.10) oluşturulur. M ağı içerisinden Euler turu oluşturulur ve tur ziyaret edilen noktanın tekrar ziyaretini engelleyecek biçimde Şekil 2.1.'de görüldüğü gibi Hamilton turuna dönüştürülerek çözüm sonlandırılır. Algoritmanın en kötü performans durumunda bile en kısa turdan 1,5 kat uzaklaşabileceği ispatlanmıştır[49,50].



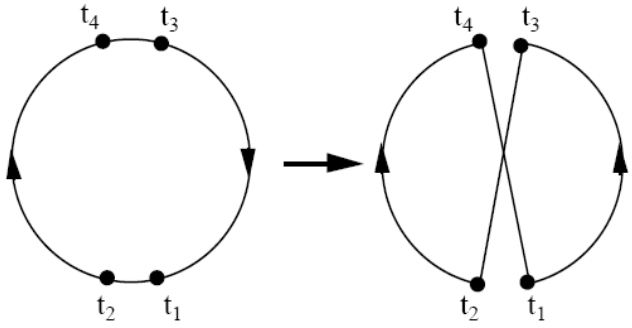
Şekil 2.10: Christofides Algoritmasında M Ağı

2.4.7. Grasp yöntemi

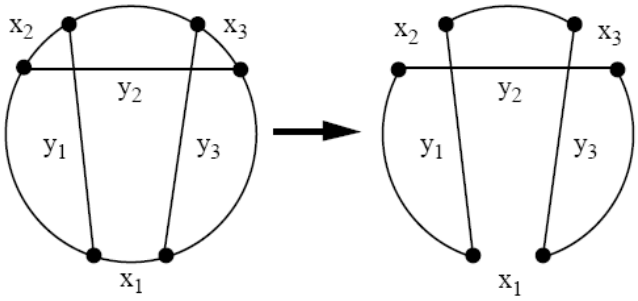
Tur oluşturmacı bir yaklaşım olan Grasp yöntemi, en yakın komşu yöntemine benzer şekilde çalışmaktadır. Ancak bu yöntemden faklı olarak olasılıklı bir yapı sergilemektedir. Tur oluştururken mevcut şehre en yakın 'n' adet şehir belirlenmektedir. Daha sonra bu şehirler arasından rasgele seçim yapılarak sonraki şehir belirlenir ve bu işleme alt tur oluşturmadan tur tamamlanıncaya kadar devam edilir. En yakın n adet komşu arasında seçim yapılırken tamamen rasgele davranılabileceği gibi, mevcut şehre en yakın olan bağa en yüksek, en uzak olana ise an düşük olmak üzere yapılacak bağın uzunluğuyla orantılı olasılık atanarak seçim yapılabilir.

2.4.8. R-opt denemeleri

R-opt denemeleri mevcut bir turun geliştirilmesi için kullanılabilecek bir yöntemdir. Yöntemde "r" ile belirtilen sayıda bağ çıkarılıp, yerine tur yapısını bozmayacak ve daha kısa tur oluşturacak şekilde yeni "r" adet bağ eklenir. Denemelere gelişme oluşturacak hareket bulunamadığı noktaya kadar devam edilir. Kısaltma yapan değişimi elde etmek için birçok deneme yapılması gerekebilmektedir. R sayısı büyüdükçe yapılması olası deneme sayısı da üssel olarak artmaktadır, ancak en iyiyi bulma olasılığı da artmaktadır. En iyinin bulunması ile süre arasında bir ödünleşim söz konusu olduğundan genellikle 2-opt ve 3-opt kullanılmaktadır. Şekil 2.11a'da görüldüğü gibi ikişer adet bağ, t_1-t_2 , t_4-t_3 çıkarılıp t_4-t_1 ve t_3-t_2 eklenerek 2-opt hareketi gerçekleştirilmektedir. Benzer yol ile Şekil 2.11b'de görüldüğü gibi üçer adet bağ, x_1, x_2, x_3 çıkarılıp y_1, y_2, y_3 eklenerek 3-opt hareketi gerçekleştirilmektedir. Bu değişimlerde amaç daha kısa tur elde edilmesidir.



(a)



(b)

Şekil 2.11: (a) 2-opt değişimi (b) 3-opt değişimi

2.4.9. Lin-kerninghan yöntemi

Lin-Kerninghan yöntemi [51] simetrik gezgin satıcı problemi için küresel en iyi ya da ona yakın çözümler üretmek için kullanılan en iyi yöntemlerden bir olarak bilinir [52]. Algoritma, r-opt yönteminin kullanımına dayanmaktadır. Yöntem, çalışması sırasında gereksinime göre "r" değerini değiştirmektedir. Her iterasyonda "r" değerinin arttırılmasının çözümün daha da kısalmasına neden olup olmayacağı test edilir. Bu işlemlere bir durdurma koşulu sağlanıncaya kadar devam edilir.

Algoritma temel olarak bazı değişimlerle bir turdan daha kısa yeni bir tur elde edilmesini sağlamak üzere, geliştirme yapamadığı noktaya kadar devam eder. Bu denemeler rasgele oluşturulmuş bir çok tur ile tekrarlanabilir. Yöntem başlangıç turuna bakmaksızın kendi başına oldukça başarılıdır. Yöntem aşağıda daha ayrıntılı olarak açıklanmaktadır.

T'nin başlangıç turu olduğunu varsayalım. Her iterasyonda $X=\{x_1,\ldots,x_r\}$ ve $Y=\{y_1,\ldots,y_r\}$ olmak üzere bağlardan oluşan iki küme oluşturulmaya çalışılır, öyle ki; X kümesindeki bağların mevcut turdan silinmesi ve yerine Y kümesindeki bağların eklenmesi durumunda daha kısa bir tur oluşacaktır. Bu bağ değişimi r-opt hareketi olarak adlandırılır. Etkin bir uygulamanın gerçekleştirilebilmesi için X ve Y kümesindeki elemanların aşağıdaki özellikleri yerine getirmeleri gerekmektedir.

(1) Sıralı değişim kriteri : Bağlar (x_i ve y_i) Şekil 2.11b'de görüldüğü gibi $\{x_1,y_1,x_2,y_2,\ldots,x_r,y_r\}$ şeklinde sıralı olmalıdır. Olumlu etki yaratan ancak zorunlu olmayan bir nokta ise zincirin sonunun, başı ile birleştirilmesidir. (2) Oluşturulacak yeni bağ yapısı olurlu bir tur ifade etmelidir. (3) İlk bağdan sonraki bağın seçimindeki kazanç ($g_i = c(x_i)-c(y_i)$) pozitif olmalıdır. Bu durumda toplam kazanç da $G_i = g_1+g_2+\ldots+g_i$ pozitif olacaktır. Bu kural çok sınırlayıcı görünmekle birlikte yöntemin etkinliğini ve hızını arttıran bir kuraldır. (4) X ve Y kümeleri bağımsız kümeler olmalıdırlar.

Algoritma karışık görünen bir yapıya sahiptir ve bunların dışında bir çok kural ile algoritma yalınlaştırılmıştır. Örneğin, çıkartılan bir bağlantı eklenmemeli ve daha

önce eklenmiş bir bağlantı çıkartılmamalıdır; $y_i = (t_{2i},t_{2i+1})$ bağının eklenmesinde t_{2i} noktasının en yakın 5 noktası ile sınırlandırılmıştır. Bu sınırlamalar sonrada algoritma aşağıdaki gibi gösterilebilir.

1. Rasgele bir T turu üret.
2. i = 1 olsun, t_1'i seç.
3. $x_1= (t_1,t_2) \in T$
4. $y_1=(t_2,t_3) \notin T$ yi $G_1>0$ olmak üzere seç.

Eğer bu mümkün değilse Adım 12'ye git.

5. i= i+1 olsun.
6. $x_i= (t_{2i-1},t_{2i}) \in T$ seç öyle ki
   (a) Eğer t_{2i}, t_1'e bağlanırsa değişim sonucu bir turdur ve
   (b) $x_i \neq y_s$ her s<i için.
7. $y_i = (t_{2i},t_{2i+1}) \notin T$ seç öyle ki
   (a) $G_i > 0$
   (b) $y_i \neq x_s$ her s<i için ve
   (c) $x_i =1$ vardır.

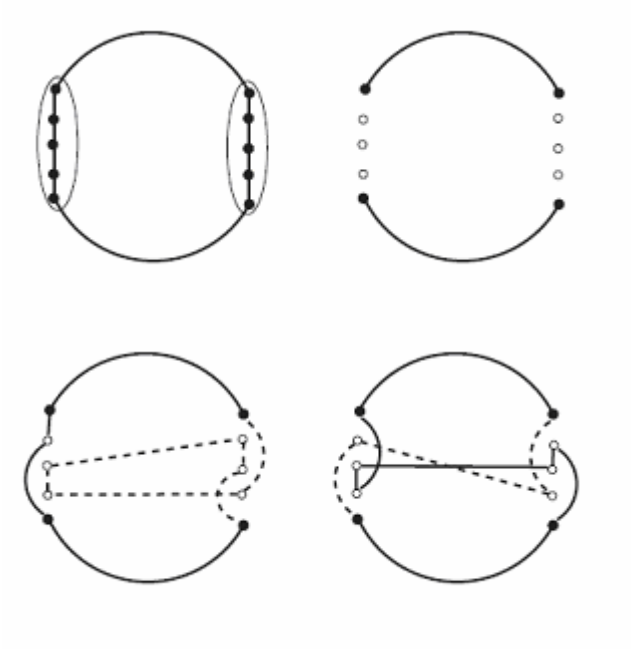
Eğer böyle bir y_i mevcut ise, adım 5'e git.

8. Eğer denenmemiş bir y_2 alternatifi varsa, i=2 yap ve adım 7'ye git
9. Eğer denenmemiş bir x_2 alternatifi varsa, i=2 yap ve adım 6'ya git
10. Eğer denenmemiş bir y_1 alternatifi varsa, i=1 yap ve adım 4'e git
11. Eğer denenmemiş bir x_1 alternatifi varsa, i=1 yap ve adım 3'e git
12. Eğer denenmemiş bir t_1 alternatifi varsa, adım 2'ye git
13. Dur (ya da adım 1'e git)

Şekil 2.12: Lin-Kerninghan yöntemi algoritma adımları

2.4.10. Hyperopt denemeleri

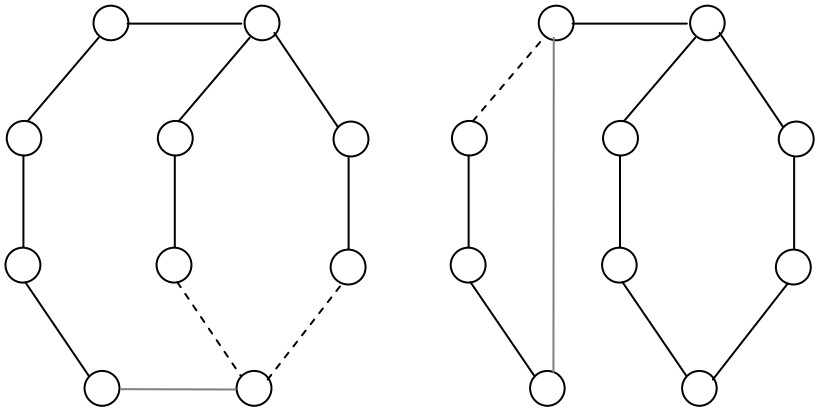
Tur içerisinde ayrık, aynı sayıda şehrin bulunduğu h uzunluklu iki bölge tespit edilir. Başlangıç ve bitiş şehirleri sabit kalmak şartıyla bölgeler içi ve bölgeler arası olası bağ değişiklikleri denenerek daha kısa turlar elde edilmeye çalışılır (Şekil 2.13) [53].



Şekil 2.13: Örnek hyperopt denemeleri [53]

2.4.11. S&C (stem and cycle) yöntemi

Gamboa ve arkadaşları [54] tarafından önerilen S&C (Stem and Cycle) yöntemi, Lin-Kerninghan yöntemine alternatif olarak öne çıkmaktadır. Bir başlangıç turu, S&C yöntemi koordinasyonu altında Alt Tur Çıkarma Zinciri (Subpath Ejection Chain (SEC)) kullanılarak iterasyonel olarak geliştirilmeye çalışılmaktadır. Yöntem, Lin Kerninghan yöntemindeki gibi statik alternatif yollar yerine, dinamik bir yol yapısı takip etmektedir. Yöntemde grafiğin alt bölümündeki bir şehir dal ve çevrimler ile ilişkilendirilerek tur geliştirilmeye çalışılmaktadır.



Çevrim (cycle) çıkarma Dal (stem) çıkarma

Şekil 2.14: Dal ve çevrim çıkarma [54]

2.5. Meta Sezgisel Yöntemler

‘Meta’ ön adı, Türkçe karşılığı ‘Benzetme’ olan İngilizce ‘Metaphor’ sözcüğünden gelmektedir. Meta sezgisel yöntemler, karıncalar, genetik seçilim, sinir ağları gibi doğadaki bazı başarılı yapıların çalışma prensiplerinden esinlenerek geliştirilmiş çözüm yaklaşımlarıdır. Sezgisel yöntemlerle karşılaştırıldığında, daha geniş bir çözüm uzayını tarayabilen, farklı problemler üzerine uygulanabilen meta sezgiseller, özellikle konveks olmayan, türevlenemeyen, klasik çözüm yöntemleriyle çok uzun sürede çözülebilen ya da modelleme zorluklarının karşılaşıldığı problemlerde sıklıkla kullanılmaktadırlar.

Meta sezgisel yöntemler başlığı altında incelenebilecek bir çok yöntem mevcuttur ve zamanla yenileri eklenmektedir. Bu bölümde, meta sezgisel yöntemler denildiğinde akla ilk gelen, üzerinde en çok çalışma yapılmış olan Genetik Algoritmalar, Karınca Kolonileri, Sinir Ağları, Tabu Arama, Tavlama Benzetimi yöntemleri ile; Diferansiyel Gelişim Algoritmasına mutasyon yaklaşımı ve çözüm vektörü gösterimi açısından oldukça benzeyen Parçacık Sürü Optimizasyonu yöntemi, kısaca anlatılmış ve GSP uygulamalarına değinilmiştir.

2.5.1. Genetik algoritmalar

Genetik algoritmalar ilk olarak 1975 yılında Holland tarafından önerilmiştir. Temel olarak canlılardaki doğal gelişim prensibine dayanmaktadır. Gerçek yaşamda bir populasyon içerisinde çevrenin ve ortamın gerektirdiklerine daha iyi cevap verebilen bireyler bir sonraki nesle genlerini aktarabilmekte, yetersiz olanların büyük bir kısmı elimine olmaktadır. Bu şekilde bir sonraki nesil daha gelişmiş olmaktadır. Nesiller ilerledikçe en iyi gen kombinasyonunun ortaya çıkması mümkün hale gelmektedir.

Genetik algoritmalarda potansiyel çözümler ile ilgili bilgiler, genleri temsil eden diziler şeklinde sayılar ile ifade edilirler. Eniyileme sürecinin başlangıcında, çoğunlukla rasgele olarak, bir grup çözüm üretilir. Bu başlangıç yoğunluğu her kuşakta, doğal seçme ve tekrar üreme işlemleri vasıtası ile sırasıyla geliştirilir. Son

kuşağın en uygun bireyinin problem için optimum ya da optimuma çok yakın bir çözümü oluşturması beklenmektedir.

Başlangıç populasyonun üretilmesini takiben, her çözümün uygunluğu veya iyiliği, seçilen bir uygunluk fonksiyonu kullanılarak değerlendirilir. Seçme, çaprazlama ve mutasyon gibi genetik operatörler, iyi çözümlerden yeni çözümler elde etmek için kullanılırlar. Bu iyileştirme işlemi, önceden belirlenen bir jenerasyon sayısına veya tatmin edici bir sonuca ulaşıncaya kadar devam ettirilir. Temel bir genetik algoritma adımları Şekil 2.15'de görüldüğü gibidir:

```
BAŞLA
İlk  populasyonu rasgele oluştur
        WHILE (Durdurma kriteri= =0)
                Populasyondan ebeveynleri seç
                Seçilen  ebeveynlerden çocukları üret
                Bireyleri mutasyona uğrat
                Çocukları ekleyerek populasyonu genişlet
                Kötü bireyleri silerek populasyonu eski büyüklüğüne döndür
        END
BİTİR
```

Şekil 2.15: Temel Genetik Algoritma'nın Adımları

Çaprazlama işleminde ebeveyn olarak seçilen bireylere ait kalıtımsal bilgilerin, onları daha iyi yapan kısımlarının çocuklarda bir araya gelerek daha iyi bireylerin oluşması hedeflenmektedir. Bu amaçla bireyleri ifade eden dizilerin belirli kısımları rasgelelik yardımıyla belirli noktalardan kesilerek bir araya getirilmektedir. İşlem sonucunda iyi bireylerin de kötü bireylerin de ortaya çıkması mümkündür. Bu durumun kontrol altında tutulması için populasyondaki bireyler belirli bir olasılıkla çaprazlama işlemine tabi tutulmaktadır. Yüksek çaprazlama oranı yeni jenerasyonda fazla sayıda yeni birey oluşturulmasına neden olurken, mevcut populasyondaki iyi bireylerin kaybedilmesine neden olabilmektedir. Düşük çaprazlama oranında ise mevcut populasyondaki iyi bireylerin uzun süreler boyunca yaşaması nedeniyle, yeni iyi çözümlerin bulunması zorlaşmaktadır. İyi bireylerin korunmasını sağlayan elitist

olarak adlandırılan diğer bir strateji ise populasyondaki en iyi birey ya da bireylerin korunmasına dayanmaktadır.

Mutasyon işlemi, bireyin diğer bireyler ile etkileşime girmeden kendi içinde gerçekleşen bir işlemdir. Çözümü ifade eden dizinin rasgele bir kısmı bozunmaya uğratılarak, daha iyi çözümlerin elde edilmesi, yerel en iyiye sıkışma probleminin önüne geçilmesi hedeflenir.

Temel olarak probleme özel tanımlanan uygunluk fonksiyonuna dayanarak en uygun çözümü bulmak için çeşitli yolları deneyen genetik algoritmalar, olası çözümlerin dizi olarak formüle edilebildiği kombinasyonel problemlerin çözümünde sıklıkla kullanılırlar. GSP'nin genetik algoritmalar ile çözümünde problemin yapısına daha uygun olduğu için satıcının dolaştığı turdaki şehirlere ait numaraların sırasının kromozom (aday çözüm) olarak alındığı permütasyonel kodlama kullanılır. Bu kodlama yöntemi ile çözümü ifade eden dizinin daha küçük olmasının sağlanmasına karşın; GA'da karşılaşılan temel sorun, çaprazlama aşamasında uygunsuz bireylerin ortaya çıkabilmesidir. Şekil 2.16'da, 9 şehirden oluşan gezgin satıcının turlarını gösteren ve çaprazlama işlemi için seçilen ebeveyn kromozomlarının, 4. sıradaki şehir sonrasından kesilerek çaprazlanması işlemi görülmektedir. İşlem sonucun oluşturulan yeni bireylerden (çocuk) ilkinde 2. ve 5. şehirler, ikincisinde ise 4. ve 8. şehirler tekrarlı olarak yer almaktadırlar. Yeni çözümler mevcut halleriyle gezgin satıcı turu ifade etmemektedirler.

5	2	3	7	1	4	6	9	8	Ebeveyn1
3	4	8	7	2	9	5	1	6	Ebeveyn2
5	2	3	7	**2**	9	**5**	1	6	Çocuk1
3	4	8	7	1	**4**	6	9	**8**	Çocuk2
5	2	3	7	4	9	8	1	6	Düzeltilmiş Çocuk1
3	4	8	7	1	2	6	9	5	Düzeltilmiş Çocuk2

Şekil 2.16: Çaprazlama işleminde oluşan uygunsuz çözümler ve düzeltilişi

Permütasyonel kodlamada çaprazlama sonucunda oluşan olursuz turların önüne geçebilmek amacıyla, çocuklarda tekrar eden şehirlerin silinerek yerine turda eksik olanların yazılması yoluyla düzeltilmesi uygun olabilir, ancak bu durum ebeveyn çözümlerdeki sıralamanın doğrudan aktarılmamasından dolayı veri kaybına neden olabilmektedir. Çaprazlama etkinliğini arttırabilmek için bir çok çaprazlama operatörü önerilmiştir.

Potvin [55], Larranaga ve arkadaşları[56] permütasyonel kodlama için geliştirilmiş birçok çaprazlama yöntemini GSP üzerindeki başarım açısından karşılaştırmıştır. Her iki çalışma da Whitley ve arkadaşları [57] tarafından önerilen ER (Edge Recombination) çaprazlama operatörünü en başarılı operatör olarak değerlendirmiştir. Yöntem temel olarak ebeveynlerde her şehir için bağlantılı olduğu diğer şehirlerin, bir bağlantı haritasında tutulması ve çocuğun bu bağlantı haritasına göre mümkün olduğunca ebeveynlerde bulunan bağlar kullanılarak oluşturulması mantığına dayanmaktadır. Böylece tekrarlı bireylerin önlenmesi durumunda ebeveynlerde bulunmayan ve turu uzatabilecek bağların çocuklarda bulunmasının önüne geçilebilmektedir.

Bean [58] permütasyonel problemlerin çözümünde veri kaybının önüne geçebilmek, çaprazlama etkinliğini arttırabilmek amacıyla Rasgele Sayı Kodlama çaprazlama yöntemini öne sürmüştür. Yöntemde çaprazlama işlemi GSP turundaki şehirlerin pozisyonlarını tutan gerçel sayı vektörleri üzerinden gerçekleştirilmektedir. Elde edilen pozisyon vektörleri sıralama işlemine tabi tutularak tur yeniden oluşturulmaktadır. Böylece şehirler mümkün olduğunca ebeveynlerde bulundukları pozisyonlara yerleşmekte ve olursuz turlar önlenebilmektedir.

Gezgin Satıcı Problemi gibi zor problemlerin çözümü için genetik algoritma kullanımında karşılaşılan önemli bir sorun da erken yakınsama problemidir. Erken yakınsama, populasyondaki tüm kromozomların aynı yerel en iyi çözüm noktasında sıkışması anlamına gelmektedir. Kromozomların birbiriyle aynı olması nedeniyle populasyon tabanlı çözüm algoritmalarının önerdiği dağıtık arama avantajı ortadan kalkmaktadır. Böyle durumlarda algoritmanın küresel en iyiye ulaşma olasılığı oldukça düşüktür.

Populasyonun birbirinden ayrı ilerleyen ancak zaman zaman haberleşen parçalara bölündüğü paralel algoritma uygulamaları, erken yakınsama problemin çözümü için önerilen bir yaklaşımdır. Paralel algoritmalar ayrık yapıları nedeniyle baskın bireylerin populasyonu etkileyerek, aynı çözüme yönelmesinin önüne geçebilmektedirler. Paralel yapı, populasyondaki bireylerin başlangıçta rasgele olarak gruplara ayrılması ve grupların zaman zaman birey nakli yoluyla haberleşmesi şeklinde oluşturulabildiği gibi; uygunluk fonksiyonu değerlerine göre ayrılarak daha iyi ve kötü bireylere farklı mutasyon ve çaprazlama oranları uygulanması [59,60] yoluyla da oluşturulabilir, bu konuda birçok alternatif sunulmuştur[61]. Paralel algoritmaların diğer bir avantajı da farklı işlemcilere dağıtılabilmeleri nedeniyle toplam işlemci gücünün artması böylece, çözülebilen problem büyüklüğü ve çözümleme hızının arttırılmasına imkan tanımasıdır[62]. Katayama ve arkadaşları [63] ile Sena ve arkadaşları [64] paralel genetik algoritma kullanımı ile çaprazlama ve seçim yöntemlerinin etkileşimini GSP çözüm performansını göz önüne alarak değerlendirmiştir. Erken yakınsamanın önlenmesi için algoritma işlem süresince değişen adaptif mutasyon oranı kullanımı ya da populasyondaki aynı bireylerin bir kısmının silinerek yerlerine rasgele bireylerin eklenmesi gibi yöntemler de kullanılmıştır[65].

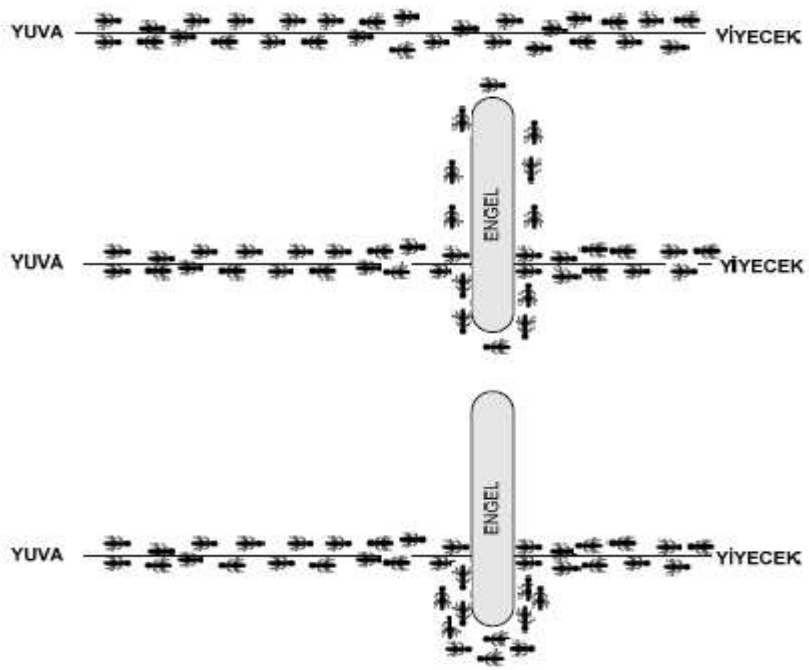
Genetik algoritma kullanılarak GSP çözümünde performansın geliştirilmesi için çaprazlama, mutasyon oranları ve faktörleri etkileşimlerinin deney tasarımıyla belirlenmesi[66], yerel arama operatörlerinin kullanılması ve başlangıç populasyonun GSP'ye özel oluşturulması [67,68] gibi yaklaşımlar da kullanılmıştır.

Genetik algoritma simetrik gezgin satıcı problemleri dışındaki genelleştirilmiş GSP [69], kübik GSP [70], çoklu GSP[71] gibi GSP türevlerinin yanında, GSP benzeri robot iş çizelgeleme[72], araç rotalama [73] problemlerinin çözümünde de simetrik GSP mantığına benzer şekilde kullanılmıştır.

Yukarıda anlatılan çalışmalarda mutasyon, yerel arama işlemlerinde; alt bölüm 2.4'te anlatılan 2-opt (ters çevirme operatörü), 3-opt(kaydırma operatörü), 4-opt (yer değişimi operatörü), kısmi n-opt (scramble opt.), Lin-Kerninghan gibi yöntemlerin kendileri ya da kısmen değiştirilmiş halleri kullanılmaktadır.

2.5.2. Karınca algoritması

Karınca Kolonisi Optimizasyonu (ACO), gerçek karıncaların yuvaları ile yiyecek noktaları arasındaki en kısa yolu bulma yeteneklerinden esinlenilerek geliştirilmiş, matematiksel olarak karınca koloni davranışlarının modellenmesine dayalı populasyon temelli bir yaklaşımdır. Karınca kolonilerinde karıncalar yiyeceklerini ararken, öncelikle yuvalarının etrafında rasgele dolaşarak keşfe başlarlar. Yiyecek kaynaklarını bulduklarında, kalite ve miktarını değerlendirdikten sonra bir kısmını yuvaya taşırlar. Karıncaların alternatif yolların söz konusu olduğu durumlarda, öncelikle bu yollara eşit olasılıkla dağılırlarken belli bir süre sonra en kısa olan yolda yoğunlaştıkları, en sonunda tümünün en kısa olan yolu kullandıkları gözlemlenmiştir (Şekil 2.17). Kısa yolun seçilebilmesi, aynı kaynağın tekrar bulunabilmesi için karıncaların yiyecek kaynağından yuvaya dönüş sırasında yiyeceğin kalitesine ve miktarına bağlı olarak kimyasal feromon maddesini geçtikleri yolun üzerine bırakmaları sayesinde mümkün olmaktadır. Yiyecek arama esnasında karıncalar gidecekleri yolun seçiminde, feromon miktarını göz önüne almaktadırlar. Feromon bulunması durumunda rasgele seçim yapılırken, feromon yoğunluğuyla orantılı olarak feromonun yer aldığı yolun seçim olasılığı artmaktadır.



Şekil 2.17 : Gerçek karıncaların en kısa yolu bulma aşamaları[74]

Karınca kolonileri ile ilgili ilk olarak M. Dorigo tarafından sunulan Karınca Sistemi (Ant System-AS) Algoritması [75,76], gerçek karıncaların en kısa yolu bulması analojisinden hareketle Gezgin Satıcı Problemi üzerinde uygulanmıştır. Stützle ve Dorigo [77] karınca kolonileriyle GSP'nin çözümünde geliştirilen AS, ACS (Ant Colony System) ve MMAS (Min-Max Ant System) algoritmalarını özetlemişlerdir. Buna göre GSP'nin çözümünde kullanılan temel bir AS algoritmasının adımları Şekil 2.18'de görüldüğü gibidir.

```
BAŞLA
Parametreleri belirle
Feromon izlerini başlat
    WHILE (Durdurma kriteri==0)
        Karınca Turları Üret
        Yerel Arama Uygula (İsteğe Bağlı)
        İzleri Güncelle
    END
BİTİR
```

Şekil 2.18: GSP için Temel AS Algoritmasının Adımları

AS ile GSP'nin çözümünde, ilk olarak *m* adet karınca rasgele seçilmiş başlangıç şehirlerine yerleştirilmektedir. Daha sonra her şehir için iteratif olarak bir durum değişim kuralı uygulanarak bir şehirden diğerine hareket edilmekte ve gezgin satıcı turları oluşturmaktadırlar. Bir *i* şehrinde yer alan *k* karıncasının *j* şehrine gitme olasılığı, bu iki şehir arasında *t* anında bulunan feromon iz miktarı $\tau_{ij}(t)$ ile, iki şehir arasındaki uzaklığın fonksiyonu olan sezgisel bilgi kullanılarak (2.1) denkleminde görüldüğü hesaplanır. Denklemde α ve β sırasıyla feromon izi ve sezgisel bilginin geçiş olasılığındaki etkilerini göstermektedir. Karıncalar daha kısa mesafede bulunan ve feromon izi daha güçlü olan şehirleri sonraki şehir olarak seçme eğiliminde olacaklardır. Bir karınca o ana kadar geçmiş olduğu şehirleri tabu listesi adı verilen hafızasında tutmaktadır. Böylece daha önce gidilen şehirlerin seçilmesi önlenerek olursuz turların önüne geçilmekte ve tur tamamlandığında feromon güncellemesi yapılacak bağların belirlenmesi için bu bilgi kullanılabilmektedir.

$$p_{ij}^{k}(t) = \frac{[\tau_{ij}(t)]^{\alpha} . [\eta_{ij}(t)]^{\beta}}{\sum_{l \in N_i^k} [\tau_{il}(t)]^{\alpha} . [\eta_{il}(t)]^{\beta}} \quad , \qquad j \in N_i^k \tag{2.1}$$

Tüm karıncaların turlarının tamamlamasının ardından, feromon bilgisi güncellenmektedir. Bu işlem (2.2) denkleminde görüldüğü gibi öncelikle (*i*,*j*) bağı üzerinde mevcut feromon miktarının buharlaşma sabiti ρ yardımıyla azaltılması ve sonrasında bu bağ üzerinden geçen turların uzunluklarının ($L^k(t)$) tersiyle orantılı olarak oluşturulan feromon miktarının (2.3) eklenmesiyle gerçekleştirilir. Böylece üzerinden daha kısa turlar geçen ve daha çok sayıda karıncanın geçtiği bağlara daha fazla feromon bırakılarak, sonraki jenerasyonlarda seçilme olasılığı artmaktadır. Ayrıca feromon buharlaşması yardımıyla eski kötü sonuçların feromon hafızasından silinmesi mümkün olmakta böylece yerel en iyi çözümlere sıkışmanın önüne geçilebilmektedir.

$$\tau_{ij}(t+1) = (1-\rho) \cdot \tau_{ij}(t) + \sum_{i=1}^{m} \Delta\tau_{ij}^{k}(t) \tag{2.2}$$

$$\Delta\tau_{ij}^{k}(t) = \begin{cases} 1/L^k(t) & \text{eğer bağ } (i,j) \text{ karınca } k \text{ tarafından kullanılmışsa} \\ 0 & \text{aksi takdirde} \end{cases} \tag{2.3}$$

Dorigo bu algoritmayı değişik boyutlardaki bir çok GSP probleminde denemiş, küçük ölçekli GSP'lerde (75 şehirden az) başarılı olurken daha büyük ölçeklilerde başarılı olmadığı görülmüştür[78]. AS algoritmasının geliştirilmesi için elitist strateji adı verilen bir yaklaşım geliştirilmiş ve çözümde o ana kadar bulunmuş en iyi tura ait bağlara daha fazla feromon bırakılması sağlanarak çözümler hız ve kalite açısından geliştirilmiştir.

GSP üzerinde çalışılan diğer bir karınca yöntemi, Karınca Koloni Sistemi (Ant Colony System) ACS, karınca sistem yaklaşımını temel alarak eksikliklerini gidermek için geliştirilmiştir [79-81]. ACS'nin AS'den üç temel farkı bulunmaktadır. ACS'de daha hızlı ilerlemenin sağlanması için sadece o ana kadar bulunmuş en iyi çözüme ait bağlarda feromon güncellenmesine izin verilmektedir. Bir karınca *i*

şehrinden *j* şehrine ilerlediğinde, diğer karıncaların da aynı yolu takip ederek erken yakınsamanın önlenmesi için (*i*,*j*) bağı üzerinde yer alan feromon bir miktar azaltılmakta, yerel feromon güncellemesi yapılmaktadır. Ayrıca bir karıncaya ait tur oluştururken daha agresif bir yol izlemektedir; *i* şehrinde yer alan *k* karıncası q_0 olasılıkla $[\tau_{il}(t)].[\eta_{il}(t)]^{\beta}$ değeri en büyük olan *l* şehrine geçecektir; (1-q_0) olasılıkla ise AS'de olduğu gibi (2.1) denklemine göre (α=1) geçiş gerçekleşmektedir.

Karınca Sisteminin doğrudan geliştirildiği başka bir yaklaşım ise MAX-MIN Karınca Sistemi (MMAS)'dir [82-84]. Sistemde feromon miktarının çözüm üzerindeki etkisini sınırlayabilmek için , alt sınır ve üst sınır değerleri ($0<\tau_{\min}\leq\tau_{ij}\leq\tau_{\max}\leq 1$), belirlenmiştir. İterasyonların başlangıcında rasgele hareketi güçlendirmek amacıyla feromon değerleri üst sınıra eşitlenmektedir. Geçiş olasılıkları AS'de olduğu gibi (2.1) denklemine göre hesaplanmaktadır ancak feromon güncellenmesi ACS'de olduğu gibi o ana kadar bulunmuş en iyi çözüme ait tur bağları üzerinde ya da iterasyondaki en iyi çözüm göz önüne alınarak gerçekleştirilmektedir.

GSP'nin çözümü üzerinde ACS'nin performansını geliştirmek için paralel işlem yaklaşımı da uygulanmıştır. Randall ve Lewis[85] aynı anda çok işlemcinin kullanılabildiği ve karınca görevlerinin işlemcilere dağıtıldığı donanımsal paralel yaklaşım kullanırken; Chu ve arkadaşları [86] ile Tsai ve arkadaşları [87] karıncaların yaratıldıktan sonra gruplara ayrıldığı ve aynı işlemci üzerinde çalıştığı algoritmik yaklaşımı kullanmışlardır. Bu yaklaşımda gruplar kendi içlerinde ACS işlemlerini ayrık olarak gerçekleştirmekte, belirli çevrim sayısında grupların en iyi bireyleri ya da tüm grupların en iyileri feromon güncellemesi yoluyla gruplar arası haberleşmeyi gerçekleştirmektedirler.

2.5.3. Tabu arama

İlk olarak Glover [88-89] tarafından geliştirilmiş olan Tabu Arama (TA), hafıza esaslı bir arama stratejisidir. Yöntemde, önceki aşamalarda elde edilen bilgi, gelecek aşamalardaki yönelimleri belirlemek için kullanılmaktadır. Zor kombinasyonel eniyileme problemlerindeki yerel optimallik sorununu çözebilecek zekice bir eniyileme tekniği olarak öne sürülmüştür [90].

Bu yöntemde bütün olasılıkları denemek çok fazla zaman alacağından dolayı zekice bir yaklaşımla, en az hesaplama ile en iyi çözüme ulaşmak için arama uzayındaki denenmiş çözümlerin listesinin bellekte tutulmasından dolayı arama sınırlandırılmış ve yerel optimumdan kurtulma bu sayede gerçekleştirilmiştir. Diğer bir tanımla, TA yerel optimuma sıkışma problemini aşmak için bir bellek fonksiyonu kullanıp, küresel optimumu hızlı bir şekilde aramada bir veya daha çok yerel arama prosedürünü hiyerarşik olarak yönlendiren zeki bir yaklaşımdır [91].

TA'da arama komşuluk mekanizmasına dayanmaktadır. Şekil 2.19'da görüldüğü gibi her iterasyonda mevcut çözümde gelişme ortaya çıkarabilecek olası tüm hareketler (örneğin GSP için 2-opt) denenerek mevcut çözümün komşuluğunda amaç fonksiyonunda en iyi ilerlemeyi verecek hareket belirlenir ve tabu listesinde yer almaması koşuluyla gerçekleştirilir. Önceki çözümlere dönülmesini önlemek için eski çözümün kendisi ya da eski çözüme dönüşü sağlayacak hareket tabu listesine kaydedilerek, belirli iterasyon sayısı için yasaklanır. Tabu listesinde kalma süresi, tabu listesinin uzunluğuna bağlıdır. Her iterasyonda yeni gelen eleman listedeki en eski elemanın yerine yazılır. İstek (aspiration) kriterinin sağlanması durumunda, bir hareket tabu listesinde olsa bile gerçekleştirilebilir. Örneğin, bir hareket bilinen en iyi çözümden daha iyi bir çözüm elde edilmesine imkan tanıyorsa kabul edilebilir. İterasyon limitine ulaşıldığında, belirli bir süre boyunca gelişme sağlanamamışsa ya da tabu listesinde olmayan komşuluk kalmamışsa çözüm sonlandırılır. Standart TA algoritmasına ait akış şeması Şekil 2.19'da görüldüğü gibidir.

```
BAŞLA
İlk  çözümü oluştur
        WHILE (Durdurma kriteri==0)
                Tüm komşu çözümler kümesi üret
                Komşu çözümleri değerlendir
                En uygun sonucu seç
                Tabu listesini güncelle
        END
BİTİR
```

Şekil 2.19: Standart TA Algoritması'nın Adımları

Tabu arama, yapı olarak her iterasyonda en iyi gelişmeyi sağlayan komşuluğa hareket edilen Tepe Tırmanma (Hill Climbing) algoritmasına benzemektedir. Ancak Tepe Tırmanma algoritmasında mevcut komşuluklar içinde gelişme sağlayan bir yön yoksa arama durdurulurken, TA'da en az gerilemeye neden olan hareket seçilerek ilerlemeye devam edilir. Bu aşamada önceki çözüm ya da ona geri dönülmesini sağlayacak hareket tabu listesine eklenerek, yerel çözüme sıkışmasının önüne geçilmeye çalışılmaktadır. Tabu liste uzunluğu algoritmanın başarımında önemli bir faktördür. Liste uzunluğunun az olması durumunda sıkışılan yerel eniyiye geri dönülmesi mümkün olurken, uzun olması durumunda küresel en iyi çözümün bulunduğu bölgenin aranmaması sorunuyla karşılaşılabilir.

Tsubakitani ve Evans [92] GSP için en uygun tabu listesi uzunluğunun ne olması gerektiğini belirlemeye çalışmışlardır. Tabu listesi büyüklüğünün şehir sayısının yarısından fazla olması durumunda sonuçların kötüleştiğini, 2-opt hareketi kullanıldığında şehir sayısının 1/4'ü , 3-opt hareketi kullanılması durumunda ise 1/8 ya da 1/16'sı büyüklüğünde tabu listesinin kullanılmasının uygun olduğunu öne sürmüşlerdir. Değindikleri diğer bir önemli bir nokta ise tabu listesinde yer alan bağların sadece birinin bile komşu çözümde bulunmasının yasaklanması durumunda tabu listesinin daha kısa olacağıdır. Tabu listesinin uzunluğunun fazla olması hesaplama yükü getirmesi açısından da tercih edilmemektedir.

Basu ve Ghosh [93], GSP üzerinde TA kullanılarak yapılan çalışmaların 2008 yılı ve öncesine ait 15 yılı kapsayan literatürünü özetlemişlerdir. Tabu liste uzunluğu problem büyüklüğüyle doğrudan ya da logaritmik olarak da sabit belirlenebilmekle birlikte, yine problem büyüklüğüyle ilintili olarak belirli aralıkta rasgele belirlenmesinin de mümkün olduğu ancak bu konuda önerilebilecek bir yönelimin görülmediği ortaya koyulmuştur. Tabu listesiyle yakın iterasyonlardaki çözümlere dönülmesinin önlenmesi kısa dönem hafıza olarak değerlendirilmektedir. Orta dönem hafıza stratejisiyle ise daha önce bulunan başarılı çözümlerin ya da bu turlarda sıklıkla bulunan bağların sonraki çözümlerde seçilmesi yoluyla iyi çözümler etrafındaki daha iyi çözümlerin bulunması (intensification) hedeflenmektedir. Uzun dönem hafızada ise algoritmanın aynı çözümler etrafında sıkışmasını önlemek için bağlara tekrar sayısına göre ceza katsayısı uygulanması, komşuluk arama hareketinin

değiştirilmesi, daha fazla sayıda kötü yönde harekete izin verilmesi gibi yollarla, çözümün farklılaştırılması (diversification) sağlanmaya çalışılmıştır.

GSP çözümünde başarılı sonuçlar elde edilen Tekrarlı Tabu Arama (TTA) Algoritması [94-96] hafıza stratejilerinin bir kombinasyonuyla çalışmaktadır. Klasik TA bir çok defa çalıştırılarak elde edilen çözümler arasından en iyisi ya da iyiler arasından rasgele seçilen bazıları orta dönemli hafızada olduğu gibi tekrar tabu aramaya tabi tutulmaktadır. Ayrıca yerel çözüme sıkışmış çözümleri yerelden kurtarmak için uzun dönemli hafızada olduğu gibi farklı komşuluk hareketi kullanılarak eldeki çözüm bir miktar bozulmakta daha sonra klasik aramaya devam edilmektedir.

Tabu arama için başlangıç çözümü oluşturmak amacıyla önceki bölümde anlatılan oluşturmacı sezgiseller kullanılabilmektedir ancak, yukarıda sunulan çalışmalarda başlangıç çözümü için sezgisel kullanımının sonuç kalitesi üzerinde önemli bir etkisi bulunmadığı vurgulanmaktadır. Komşuluk arama hareketi olarak, diğer meta sezgisellerde yerel arama için kullanılan 2-opt, 3-opt, 4-opt, 5-opt hareketleri ya da bunların daraltılmış halleri kullanılmaktır.

2.5.4. Tavlama benzetimi

Kirkpatrick ve arkadaşları [97] tarafından 1983 yılında geliştirilmiş olan ve sözü edilen çalışmada GSP üzerinde de uygulanan Tavlama Benzetimi (TB), metallerin ısıl işlemi ile bir optimizasyon probleminin çözümü arasındaki benzerlikleri temel almaktadır.

Isıl işlem fiziksel sürecinde amaç, bir metalin kristal yapısı en küçük enerji durumuna ulaşacak şekilde, metali dikkatlice soğutarak moleküller arası kuvvetli bağlara sahip katı yoğunluk elde etmektir. Eğer soğutma ani bir biçimde yapılırsa optimal sertlik özelliklerine sahip olmayan malzeme ortaya çıkmaktadır. Aşamalı ve dikkatli soğutma daha düşük enerjiyle, dayanıklı malzeme vermektedir. Bu soğutma prosesi, aşamalarda ısıyı adım adım azaltarak gerçekleşmektedir. Isı, metalin katılaştığı noktaya ulaştığında işlemler tamamlanmaktadır [98].

Optimizasyon problemlerinde bir çözümden başka bir çözüme doğru ilerlerken mümkün olan en iyi gelişmenin seçildiği hırslı algoritmaların kolaylıkla yerel en iyiye sıkıştıkları bilinmektedir. Fiziksel metal tavlama işleminde olduğu gibi, en iyileme aşamasında da mevcut çözümlerden daha iyi çözümlere biraz daha yavaş ilerlemek hatta tabu algoritmasında olduğu gibi kötü çözümlerin de bazen seçilebilmesine imkan vermek küresel en iyiye ulaşılmasını mümkün kılabilmektedir. TA'da daha kötüye ancak iyiye giden bir çözüm bulunmadığında gidilirken, tavlama benzetiminde daha iyi seçenekler bulunsa da, belirli bir olasılıkla, kötüye götüren seçimlerin yapılması mümkündür. Ayrıca TB'de, TA'dan farklı olarak olasılıklı bir yapı bulunmaktadır. Tavlama Benzetimi Algoritması adımları temel olarak Şekil 2.20'de görüldüğü gibidir.

```
BAŞLA
Başlangıç çözümü S'i oluştur ve en iyi çözüm olarak ata S* = S
Başlangıç sıcaklığını belirle
        WHILE (Donma gerçekleşmemişse)
                WHILE (Bu sıcaklıkta dengeye ulaşılmamışsa)
                Mevcut çözüme ait rasgele bir komşu çözüm S' belirle
                Δ = Tur uzunluğu(S') – Tur uzunluğu (S) hesapla
                        IF Δ ≤ 0
                        S = S' olarak ata.
                                IF Tur uzunluğu(S) < Tur uzunluğu (S*), S* = S.
                        ELSE
                        [0,1] aralığında uniform değişen rasgele sayı r belirle
                                IF r < e^-Δ/T, S = S'.
                        END
                Sıcaklığı düşür
                END
        END
Çözüm S*'ı döndür
BİTİR
```

Şekil 2.20: Tavlama Benzetimi Algoritmasının Adımları

Algoritmada görüldüğü gibi mevcut çözümden daha kötü bir çözümün kabul edilmesi olasılığı eğer $r < e^{-\Delta/T}$ olursa mevcuttur, ancak bu olasılık yüksek sıcaklık değerlerinde fazla iken, sıcaklık düştükçe azalmaktadır ve sıcaklık 0'a düştüğünde sadece daha iyi çözümleri kabul eden hırslı bir yapıya dönüşmektedir. Aynı sıcaklık seviyesinde malzeme tanecik yapısının dengeye ulaşması için yapılana benzer şekilde bir miktar beklenmekte, yani aynı olasılık seviyesinde belirli sayıda deneme yapılmaktadır. Buna göre algoritma performansı başlangıç sıcaklığı, azaltma miktarı, her sıcaklık seviyesinde yapılan deneme sayısı ve komşuluğu bulmak için kullanılan harekete göre değişmektedir.

Johnson ve McGeoch [99] çalışmalarında TB'nin GSP çözümünde kullanımının temellerini anlatmış ve bu konuda yapılan çalışmaları özetlemişlerdir. Belirli bir sıcaklık seviyesinde yapılması gereken deneme sayısının en az olası komşuluk sayısıyla orantılı olması gerekmektedir. Örneğin en çok kullanılan 2-opt hareketinde şehir sayısının karesiyle orantılı arama yapmak mantıklı olacaktır. TB'de karşılaşılan temel sorun, başarımın yüksek seviyede elde edilmesi için algoritmanın çalışma süresinin arttırılması gerekliliğidir. Bu sorunun önüne geçilmesi için taranacak komşuluklardan uzaktaki şehirleri içerenlerin budanması, başlangıç çözümü için sezgisel yöntemlerin kullanılması yoluyla daha düşük başlangıç sıcaklıklarıyla çalışılması gibi yollar tercih edilebilmektedir. Klasik TB'de daha kötüye yönelen çözümler için bir sınır bulunmamaktadır. Bu durumda çözümü oldukça kötüleştirip iyi çözüm bölgesinden uzaklaştıracak hareketlere izin verilmesi mümkündür. Bu durumun önüne geçebilmek için; aday komşuluk eğer mevcut çözümden daha kötü bir çözüm öneriyorsa ancak amaç değeri mevcut çözümün amaç değeri ya da bulunan en iyi amaç değerinden belirli bir eşiğe kadar kötüyse kabul edilebildiği, eşik temelli yaklaşımların kullanılması yoluna gidilmektedir.

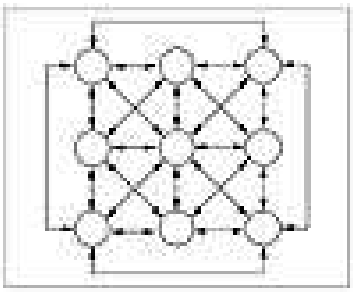
2.5.5. Yapay sinir ağları

İnsan beyni bilinen en gizemli ve karmaşık hesaplayıcıdır. Yapay Sinir Ağları (YSA), insan beyninin işleyişini taklit ederek, zekice davranışların yapay olarak geçekleştirmeye çalışıldığı yaklaşımlardır. YSA yapılarının esin kaynağı biyolojik sinir ağlarının işleyiş yöntemidir. Beynin işleyiş kuralları bir çok YSA modelinin

geliştirilmesinde kullanılmıştır. YSA, öğrenme özelliği bu yaklaşımın cazibesini arttırmıştır.YSA, bir sisteme ilişkin tek veya çoklu parametrelere bağlı olarak tanımlanan giriş verileri ile sistemin yine tek veya çoklu parametrelere bağlı olarak tanımlanabilen çıkışları arasında ilişki kurabilme yeteneğine sahiptir. Bu ilişkinin doğrusal bir formda olması zorunlu değildir. YSA'lar, çıkış değerleri bilinmeyen tanımlanmış sistem girişlerine uygun çıkışlar da üretebilirler.

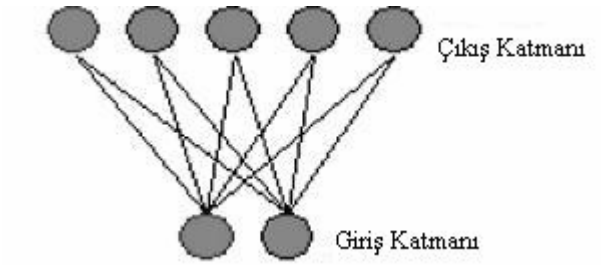
Yapay sinir ağları öğrenme yöntemlerine göre birkaç gruba ayrılır. En çok bilineni; eğiticili öğrenme olarak adlandırılan, giriş ve çıkış değerlerinin verildiği, sinir ağının girişlere göre istenen çıkışları oluşturduğu yapıdır. Bu yöntem giriş ve çıkış arası ilişkinin belirlenmesinin önemli olduğu problem yapılarında kullanılmaktadır. Optimizasyon problemleri için ise çıkış değerinin ağ yapısının amaç fonksiyonunda yarattığı sonuca göre belirlendiği, öğreticisiz öğrenmenin yapıldığı sinir ağları kullanılır.

GSP için en çok bilinen sinir ağı uygulaması Hopfield ve Tank [100] tarafından gerçekleştirilen uygulamadır. GSP için geliştirilen sinir ağı Şekil 2.21'de görüldüğü gibidir. Önerilen ağda şehir sayısının karesi (n^2) kadar nöron bulunmaktadır. Bu yerleşimde nöronlar "0" ya da "1" değerini alabilirler. Bu şekliyle bağ matrisine benzemekle birlikte satırlar şehirleri, sütunlar ise o şehrin turda kaçıncı sırada olduğunu göstermektedir. Bir nöronun "1" değerini alması durumunda bulunduğu satırdaki şehrin, sütundaki sırada yer aldığını göstermektedir. Her satır ve sütunda sadece tek "1" değerinin bulunmasına izin verildiği modelde çıkış değeri olarak tur uzunluğunu gösteren bir enerji fonksiyonu alır. Enerji miktarının hedef değer olan 0'dan sapmasını en küçüklemek üzere nöron değerleri değiştirilerek çözüme ulaşılmaya çalışılır.



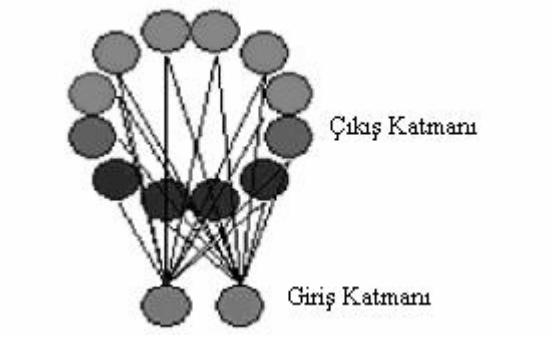
Şekil 2.21: GSP için Hopfield tarafından önerilen sinir ağı

GSP için önerilen diğer bir sinir ağı yaklaşımı ise Kohonen [101] tarafından önerilen kendi kendine öğrenen (Self Organizing Maps, SOM) ağlardır. Şekil 2.22'te gösterilen bu ağlarda giriş değeri o şehre ait koordinatları gösterirken, çıkış nöronlarına ait ağırlıklar ise şehrin turdaki yerini göstermektedir. Burada da öğreticisiz öğrenme yöntemiyle nöron ağırlıkları değiştirilerek en kısa tur bulunmaya çalışılır. Yukarıda anlatılan yöntemler, diğer GSP yöntemlerine nispeten başarılı sonuçlar elde edememekle birlikte çözüm için iki ana kaynağı oluşturmuşlardır.



Şekil 2.22: Kohonen ağı

Yanping ve arkadaşları [102] Kohonen ağ yerleşimini Şekil 2.23'de görüldüğü gibi değiştirerek ve öğrenme yöntemini geliştirerek başarılı sayılabilecek sonuçlar elde etmişlerdir. Tamura ve arkadaşları [103] ise Hopfield ağını yerel optimuma sıkışmaması için geliştirerek başarılı sonuçlar elde etmişlerdir.



Şekil 2.23: Yanping tarafından önerilen ağ

2.5.6. Parçacık sürü optimizasyonu

Parçacık Sürü Optimizasyonu (Particle Swarm Optimization-PSO) yöntemi ilk olarak Kennedy ve Eberhart [104] tarafından önerilmiş, kuş sürülerinin arama davranışını temel alan populasyon tabanlı bir en iyileme algoritmasıdır. PSO'da her çözüm sürüyü oluşturan sosyal bir parçacık (kuş) olarak kabul edilir. Parçacıklar çözüm uzayı üzerinde uçarak kendi başlarına arama yaparken bir yandan da sürü içersindeki diğer parçacıkların arama davranışından etkilenirler. Algoritmanın temel adımları Şekil 2.24'de görüldüğü gibi gerçekleşmektedir.

```
BAŞLA
İlk  parçacık populasyonunu oluştur
        WHILE (Durdurma kriteri= = 0)
                FOR (x_p pozisyonundaki her p parçacığı için )
                        IF (x_p,  peniyi_p'den daha iyiyse)
                                peniyi_p = x_p
                        END
                END
                Şu ana kadar bulunan en iyi çözümü geniyi_p olarak belirle.
                FOR (her p parçacığı için )
                        v_p= Hız_hesapla (x_p, peniyi_p, geniyi_p)
                        x_p = Pozisyon_belirle(x_p, v_p)
                END
        END
BİTİR
```

Şekil 2.24: PSO Algoritması Adımları

Parçacığın pozisyonu (x_p) o anda bulunan çözümü ifade eder. Parçacık sürekli hareket halindedir ve bu hareket denklem 2.5'te görüldüğü gibi pozisyon vektörüne her iterasyonda eklenen bir hız vektörü (v_p) ile sağlanır . Hız vektörü ise parçacığın o ana kadar bulduğu en iyi çözüm ($peniyi_p$) ile sürünün o andaki en iyi çözümünden ($geniyi_p$) etkilenerek deklem 2.4'teki gibi her iterasyonda güncellenir.

$$v_p(t) = v_p(t-1) + c_1.rand_1(peniyi_p(t-1) - x_p(t-1)) + c_2.rand_2(geniyi_p(t-1) - x_p(t-1)) \quad (2.4)$$

$$x_p(t) = x_p(t-1) + v_p(t) \quad (2.5)$$

Bireye ait iyi ile, sürü en iyi çözümleri c_1 ve c_2 katsayılarıyla ölçeklendirilerek hız vektörüne etkirler. Bu katsayılar [0,1] aralığında rasgele değişen $rand_1$ ve $rand_2$ değerleriyle olasılıklı hale getirilir. Yerel en iyi çözüme sıkışmanın önüne geçmek için eski hız vektörünün bir *w* atalet katsayısı değeriyle çarpılması yoluna gidilebilir. Böylece mevcut hız bireyin ya da sürünün en iyi çözümünde meydana gelebilecek değişikliklere daha az duyarlı bir yapıya kavuşarak dağıtık yapının korunması mümkün olmaktadır.

Klasik algoritmada çözümler (pozisyonlar) problemin her boyutu gerçel sayı ile ifade edilen çok boyutlu gerçel sayı dizileri ile ifade edilirler. Bu yapısıyla GSP gibi permütasyonel problemlerin çözümünde kullanılabilmesi için gerçel sayıların sıralama vektörüne dönüştürülmesi ya da algoritmanın yapısının sıralama vektörüyle çalışacak şekilde değiştirilmesi gerekmektedir. Bu iki yaklaşım da GSP çözümü için uygulanmıştır.

Pang ve arkadaşları [105] sürekli ve kesikli çözüm uzayları arasında dönüşümün temel alındığı bir PSO algoritması ile GSP çözümü üzerinde çalışmışlardır. Pozisyon güncelleme ve hız hesaplama işlemleri sürekli uzayda gerçekleştirilirken, yerel arama işlemi kesikli uzayda gerçekleştirilmiştir. Yerel arama işlemi için 2-swap ve 2-opt operatörleri kullanılmıştır. Yerel en iyi çözüme sıkışmanın önüne geçebilmek için kaotik bir operatörden yararlanmışlardır. Operatör sürekli uzayda pozisyon ve hız vektörlerini rasgele sayılarla çarparak rasgele bir şekilde değiştirmektedir. Çalışmada kaotik operatörün kullanımının etkin olup olmadığın belirlenmesi ve yerel arama operatörleri ile etkileşimli bir şekilde seçim yapmak için algoritmanın dört farklı versiyonu burma14, eil51, berlin52 ve eil76 karşılaştırma problemleri üzerinde denenmiş, kaotik operatör ile 2-opt kullanımının daha iyi sonuçlar verdiği ortaya koyulmuştur.

Pang ve arkadaşları [106] GSP çözümü için bulanık mantık temelli bir PSO yaklaşımı önermişlerdir. Çalışmada [0,1] arasında değişen gerçel sayı değerlerinden oluşan pozisyon matrisleri kullanılmıştır. *P* pozisyon matrisindeki p_{ij} değeri verilen turdaki *i.* şehrin *j.* pozisyona üyeliğini göstermektedir. Hız değerleri de benzer şekilde matriste tutulmuş ve PSO algoritma işlemleri matrisler üzerinden gerçekleştirilmiştir. Elde edilen matris çözümü en büyük üyelik dereceli sıranın şehirler için seçilmesi yoluyla sıralama vektörüne dönüştürülmüştür. Yöntem burma14 ve berlin52 karşılaştırma problemleri üzerinde denenmiş ve başarılı sonuçlar elde edilmiştir.

Wang ve arkadaşları [107], GSP'nin PSO ile çözümünde gerçel ve sıralama vektörleri arasında dönüşüm yapmak yerine, pozisyon vektörünü doğrudan sıralama vektörüyle tutmuş ve farklı hız vektörü yapısı kullanmıştır. Hız vektörü belirlemede önemli nokta pozisyon vektörleri arasındaki farkın belirlenmesidir. Sürekli uzayda bu işlem basit çıkarma işlemi ile gerçekleştirilebilirken, permütasyonel uzayda iki vektörün farkını belirlemek oldukça zordur. Yazarlar bu işlemi yaparken bir vektörden diğer vektöre ulaşmayı sağlayabilecek 2-swap operatörünü ardışık olarak kullanmışlardır. Bu operatör ile iki sıralama (tur) vektörü arasındaki fark, iki şehrin karşılıklı olarak yerlerini değiştirdiği birkaç hareket ile belirtilmektedir. Belirlenen geçiş hareketlerinin belirli olasılıkla uygulanarak hız vektörlerinin güncellendiği çalışmada yöntem sadece burma14 problemi üzerinde uygulanmıştır.

Wang ve Yang [108], Wang ve arkadaşlarına [107] benzer şekilde pozisyon vektörü olarak sıralama vektörü tutmaktadır. Hız vektörü ise şehir kaydırma (sliding) operatörü kullanılarak belirlenmektedir. Bir dağıtıklık ölçeği yardımıyla erken yakınsamanın engellenmeye çalışıldığı çalışmada burma14, oliver30, eil51 problemleri karşılaştırma amaçlı olarak kullanılmıştır.

Zhong ve arkadaşları [109] GSP'nin PSO ile çözümünde sıralama ya da gerçel sayı vektörleri yerine, bağları ifade eden ikili vektörler kullanmışlardır. Pozisyon vektörleri arası fark belirlenirken bir şehirde olup diğerinde olmayan bağlara 1 olasılığı verilirken, diğer bağlara c_1 ve c_2 ile sınırlandırılan rasgele olasılıklar verilmektedir. 2.5 denkleminin 2.6 denkleminde görüldüğü gibi değiştirilmesi

nedeniyle; hız vektöründen gelen bağlar ile c_3 mutasyon katsayısıyla değiştirilen eski çözüm vektöründen gelen bağlar bir araya gelerek aday kenar havuzunu oluştururlar. Daha sonra bağlar genetik algoritmada anlatılan bağ birleştirme (Edge Relinking) mantığına benzer şekilde olasılıkları da göz önüne alınarak birleştirilerek yeni çözüm oluşturulur. Yeni bir tur gösterim yaklaşımının denendiği yöntem burma14, eil51, berlin52, eil76, kroA100 ve kroA200 problemleri üzerinde uygulanmıştır.

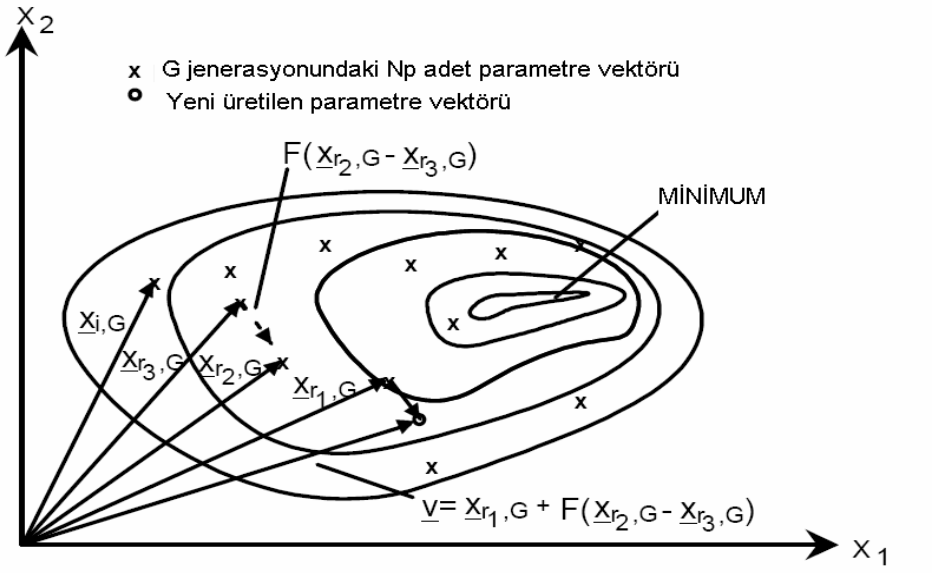
$$x_p(t) = c_3.rand_3.x_p(t-1) + v_p(t) \tag{2.6}$$

Goldbarg ve arkadaşları [110], sıralama vektörünü kullandıkları çalışmalarında PSO hız formülünün yapısını değiştirerek farklı hız operatörleri yaklaşımını önermişlerdir. Buna göre bir parçacığa PSO'nun yapısı gereği iki çeşit hız operatörü uygulanabilir. İlki parçacığın mevcut çözümünü diğer parçacıklarla etkileşmeden kendi başına geliştirmeye çalışacağı tekli hız operatörüdür ki, bu operatör diğer meta sezgisellerdeki yerel arama ile aynı işlevi görmektedir. Diğer operatör ise parçacığın diğer bir parçacık ile etkileşime girerek daha iyi bir çözüme ulaşmaya çalıştığı operatördür. Bu durumda iki permütasyonel vektör arasında Wang ve arkadaşları'nın [107] yaptığı gibi iki vektör arasında geçişi sağlayacak operatörler dizisi belirlenir. Kaydırma operatörünü kullandıkları bu yaklaşımda daha iyi çözümün elde edildiği noktada durulması hedeflenmiştir. İkili hız operatöründe bireyin en iyi çözümü ya da sürünün en iyi çözümüyle, mevcut pozisyonun ilişkisi kullanılmıştır. Çözümün başlangıcında yerel hız operatörünün uygulanmasına daha büyük olasılık verilirken İterasyonlar ilerledikçe ikili operatörlerin gerçekleşme olasılığı arttırılmaktadır. Yaklaşımın büyük problemlerde de başarılı sonuçlar verdiği öne sürülse de başarı, temelde tekli hız operatöründe kullanılan ve tek başına kullanıldığı takdirde de oldukça başarı sonuçlar elde edilebilen Lin-Kerninghan sezgiselinin kullanılmasına dayanmaktadır. Bunun yerine basit ters çevirme operatörünün kullanıldığı durumda, Zhong ve arkadaşları [109] tarafında sunulan yaklaşıma göre daha zayıf sonuçlar ortaya çıkmaktadır.

3. DİFERANSİYEL GELİŞİM ALGORİTMASI

Diferansiyel Gelişim Algoritması (DGA) populasyon tabanlı yeni bir sezgisel eniyileme yöntemidir. Gerçek değerli tasarım parametrelerini içeren fonksiyonları küresel eniyilemek amacıyla kullanılan bir algoritma olarak geliştirilmiştir. Yöntem, vektörler arasındaki bezerliklerden yararlanarak daha iyi çözümler elde etmeye çalışan genetik algoritma ve karınca kolonileri algoritmaları gibi meta sezgisellerden farklı olarak, vektörler arasındaki farklılıklardan da yaralanarak populasyonu geliştirmektedir.

Gerçel kodlanmış genetik algoritmalara çok benzeyen yöntemin farklılığı temelde, önerdiği çaprazlama ve mutasyon işlemlerine dayanmaktadır. Şekil 3.1'de görüldüğü gibi, bir G jenerasyonunda X_{r2} ve X_{r3} gibi iki vektörün farklarının bir ölçek faktörü (F) ile çarpılarak başka bir vektöre (X_{r1}) eklenmesi yoluyla gerçekleştirilen mutasyon işlemi sonunda elde edilen aday vektör (v), populasyondaki diğer bir vektör ile çaprazlanmakta ve elde edilen çözüm vektörü ancak, populasyon vektöründen daha iyiyse yerine geçebilmektedir.



Şekil 3.1: Diferansiyel Gelişim Mutasyonu[111]

DGA'da diğer meta sezgisel yöntemlerden farklı olarak, seçilim aşamasında, en iyi bireylerin yaşamaya devam etmesi ya da bireylerin her iterasyonda yeniden oluşturulması yerine, populasyondaki bireyler iterasyonlar boyunca varlıklarını devam ettirmekte ve ancak kendisinden türetilmiş daha iyi bir vektörle yer değiştirmektedirler. Böylece, mevcut durumda kötü bir sonuç verse dahi ilerde, küresel eniyiye ulaşabilecek çözümlerin devamı ve bireylerin uzaydaki dağıtıklığının korunması sağlanarak, erken yakınsamanın önüne geçilmektedir. Bireyler arası haberleşme, özel mutasyon işlemi sayesinde gerçekleştirilmektedir. Mutasyon işlemi, algoritmanın başarımını arttırmakta ve onu daha sağlam yapmaktadır. Algoritma, gerçel sayı formatıyla çalışması nedeniyle, bu formata uygun problemlerde oldukça başarılı sonuçlar vermekte ve çözüm uzayını daha iyi tarayabilmektedir. Tamsayılı, kesikli ve sürekli eniyileme problemlerinde kullanılabilen algoritma, hızlı, basit, kolayca kullanılabilir bir yapıya sahiptir. Mevcut çözümün başarım değerinin hesaplanmasının uzun sürebileceği problemlerde, doğal olarak paralel olan yapısı, etkin çözümler elde edilmesine imkan vermektedir. Algoritma, yukarıda sayılan özelliklerin yanı sıra, hesaplama maliyeti açısından nispeten ucuz olması nedeniyle küresel eniyileme problemleri için oldukça uygundur. Ancak, sürekli yüzeylerdeki eniyileme problemlerinde oldukça başarılı olsa da, GSP gibi permütasyonel problemlerdeki başarısı henüz oldukça sınırlıdır.

3.1. DGA Adımları

Klasik DGA, Şekil 3.2'de görüldüğü gibi temel olarak başlangıç populasyonun oluşturulması, mutasyon, çaprazlama ve seçilim olmak üzere dört aşamada gerçekleştirilir.

3.1.1. Populasyonun oluşturulması

DGA'nın ilk aşamasında N_p adet D boyutlu parametre vektörleri X_{iG} oluşturulur. Burada i, populasyonun i. elemanını ve G ise populasyonun ait olduğu jenerasyonu göstermektedir.

$$X_{iG} = [x_{1,iG}, x_{2,iG}, x_{3,iG}, ..., x_{D,iG}]^T, \quad i = 1,2,..., N_p \tag{3.1}$$

```
BAŞLA
Rasgele Bireyler Oluştur
İlk Populasyon  Uygunluğunu Değerlendir
        WHILE (Durdurma kriteri= =0)
                DGA Mutasyon
                DGA Çaprazlama
                 Aday Populasyon  Uygunluğunu Değerlendir
                 Seçilim
        END
BİTİR
```

Şekil 3.2: Klasik DGA Algoritması

Her parametre, kullanıcı tarafından belirlenen aralıklar içinde sınırlandırılabilir.

$$X_{j,iG} \in [x_j^{alt}, x_j^{üst}], \quad j = 1,2,...,D. \tag{3.2}$$

Populasyon büyüklüğü iterasyonlar süresince değişmemektedir ve yöntemin kontrol parametrelerinden biridir. Başlangıçta populasyondaki bireyler rasgele olarak yaratılmakta ve tüm parametre uzayını eşit olasılıkla kaplayabilmektedir. Bireylerin yaratılmasını her vektörün uygunluk fonksiyonu ile değerlendirilmesi ve sonuçlarının saklanması izlemektedir.

3.1.2. Mutasyon

Mutasyon populasyon içerisinden rasgele seçilen bir populasyon bireyine, yine rasgele seçilen iki vektörün ölçeklendirilmiş farkını ekleyen bir süreçtir. Her bir X_{iG}, i = 1,2,…,N_p vektörü için mutasyona uğramış vektör aşağıdaki formüldeki şekilde hesaplanır.

$$v_{iG+1} = x_{r1,G} + F.(x_{r2,G} - x_{r3,G}), \tag{3.3}$$

Burada, *r1,r2,r3* [1,N_p] aralığında değişen birbirinden farklı tamsayılardır. *F* ise 0'dan büyük [0,2] aralığında değişen bir ölçek faktörüdür. Bu nedenle her hedef vektör X_{iG} için yeni bir $v_{i,G+1}$ mutant vektörünün oluşturulması gerekmektedir. Şekil 3.1'de görüldüğü gibi rasgele seçilen iki populasyon bireyi $X_{r2,G}$ ve $X_{r3,G}$'nin farkının yine rasgele seçilen üçüncü bir $X_{r1,G}$ vektörüne *F* ile ağırlıklandırılarak eklenmesi yoluyla mutant vektör elde edilir. Mutasyona uğramış vektör, yeni bir aday vektörün oluşturulması için kullanılacaktır. Mutasyonun parametrelerden birinin 3.2 denkleminde verilen aralığın dışına çıkması durumunda bu parametre aşağıdaki şekilde yeniden düzenlenebilir:

Eğer $v_k < v_k^{alt}$ ise $v_k = v_k^{alt}$,

Eğer $v_k > v_k^{üst}$ ise $v_k = v_k^{üst}$, (3.4)

3.1.3. Çaprazlama

Mutasyona uğramış vektörlerin çeşitlendirilmesini sağlamak amacıyla, populasyon üyelerine çaprazlama operatörü uygulanır. Bu uygulama bir aday vektörü oluşturarak mutasyon etkisini kısmen bastırır. Aday vektör denklem 3.5'te görüldüğü gibi gösterilebilir.

$$u_{iG} = (u_{1,iG}, u_{2,iG}, u_{3,iG}, ..., u_{D,iG}) \tag{3.5}$$

Aday vektörün bileşenleri mutant vektör v_{iG+1} ya da mevcut populasyon üyesi hedef vektör x_{iG} vektör elemanlarından denklem 3.6'da görülen ilişkiye göre oluşturulur.

$$u_{k,iG+1} \quad k \in [1,D], i \in [1,N_p] = \begin{cases} v_{k,iG+1} & eger\ rasg \le CR\ ya\ da \\ & k = RasgTams(1,D), \\ x_{k,iG} & aksi\ takdirde, \end{cases} \tag{3.6}$$

Burada *rasg*, [0,1] aralığında eş olasılıkla üretilmiş bir gerçel sayı; *CR* kullanıcı tarafından belirlenen çaprazlama olasılığı ve *RasgTams(1,D)* ise [1,D] aralığında

rasgele tamsayı üreten bir fonksiyondur. Bu en son fonksiyon en azından bir parametrenin mutant vektörden alınmasını garantilemektedir.

3.1.4. Seçilim

Seçilim işlemi yeni oğul jenerasyon olan *G+1*'in oluşturulması işlemidir. Birçok evrimsel algoritmanın aksine diferansiyel gelişim algoritması sıralama ya da oransal seçim yöntemlerini seçim yöntemi olarak kullanmamaktadır. Onun yerine hedef ya da aday vektörlerden hangisinin yeni jenerasyonda yer alacağına uygunluk fonksiyonu değerlerini karşılaştırarak karar vermektedir. Eğer hedef vektörün uygunluk fonksiyonu değeri aday vektörün değerinden daha iyi ise, hedef vektör yaşamını devam ettirerek bir sonraki jenerasyonda yer alır; aksi takdirde aday vektör yeni jenerasyonda hedef vektörün yerini alır. Bu durum denklem 3.7'de görüldüğü gibi matematiksel olarak ifade edilebilir.

$$x_{iG+1} \quad i \in [1, N_p] = \begin{cases} x_{iG} & eger\ uyg(x_{iG}) \le uyg(u_{iG+1}), \\ u_{iG+1} & aksi\ takdirde, \end{cases} \tag{3.7}$$

3.2. DGA Kontrol Parametreleri

DGA'nın performansı üç parametre tarafından belirlenmektedir: populasyon büyüklüğü N_p, mutasyon ölçekleme faktörü F, ve çaprazlama olasılığı CR . Storn ve Price'a [112] göre Np, [5D,10D] (D parametre sayısı / problem büyüklüğü) aralığında değerler almalıdır, ancak doğru şekilde çalışabilmesi için en azından 4 değerini almalıdır. F değeri [0,2] aralığında pozitif bir değer olmalıdır. F = 0.5 değeri deneylerde başlangıç için iyi bir değerdir ancak, populasyonda erken yakınsama problemiyle karşılaşılması durumunda F ve/veya N_p değerleri arttırılmalıdır. Çaprazlama oranının yüksek olması yakınsamayı hızlandırmaktadır ancak yerel eniyiye sıkışmaya neden olabilir. CR için 0.1 değeri iyi bir başlangıç değeri olarak kabul edilebilir. Ancak burada, verilen paremetrelerin sürekli eniyileme problemlerine özgü oldukları söylenebilir.

3.3. DGA Literatürü

DGA'nın temeli Kenneth Price tarafından geliştirilen ve 1994 yılında yayınlanan Genetik Tavlama (GT) algoritmasına dayanmaktadır. GT eşik düzeylerine dayanan bir tavlama kriteri kullanan, populasyon temelli kombinatoryel bir algoritmadır. Bu çalışmanın yayınlanması sonrasında Rainer Storn, Chebysev polinom uydurma problemlerini GT'yi kullanarak çözmek üzere Price ile bağlantıya geçmiştir. Chebysev polinomlarının katsayılarının hesaplanması genel amaçlı optimizasyon algoritmaları için zor bir görev olarak düşünülmekteydi. Price beş boyutlu Chebysev problemleri için GT'yi kullanarak çözüm elde etmeyi başardı ancak, yakınsama oldukça yavaş gerçekleşmekteydi ve etkin kontrol parametrelerinin belirlenmesi zordu. Bu ilk uygulama sonrasında Price GT'yi bit dizileriyle kodlama yerine, gerçel sayı vektörleriyle kodlayarak ve mantıksal operatörler yerine aritmetik operatörler kullanarak geliştirdi. Sonrasında DGA'nın dayandığı diferansiyel mutasyon işlemini keşfetti. Paralel makine mimarilerine daha kolay çalışabilmesi için Storn ayrı ebeveyn ve çocuk populasyonlarının kullanılmasını önerdi. Böylece ortaya çıkan DGA, GT'nin performansının çok üstünde 33 boyutlu Chebysev polinomunun katsayılarını dahi hesaplayabilmekteydi. DGA'nın başarısını sadece Chebysev polinomlarında değil, bir çok başka deneme fonksiyonlarında göstermesi sonrasında Price ve Storn, 1995 yılında çalışmalarını bir teknik raporla [111] sunduktan sonra, 1996 yılında Japonya Nagoya'da düzenlenen Birinci Uluslararası Evrimsel Optimizasyon Yarışmasına DGA ile katıldılar. Algoritma, belirli fonksiyonlar üzerinden gerçekleştirilen yarışmada üçüncü oldu. Ancak diğer iki yöntem genel amaçlı optimizasyon algoritmaları olarak adlandırılamayacak yapıya sahiptiler. Birinci sırayı alan yöntem açıkça yarışma fonksiyonlarının parçalanabilir olma özelliklerine dayanmaktaydı, diğer yöntem ise Latin karelere dayandığı için çok sayıda parametrenin bulunduğu problemlerde uygulanması mümkün görünmemekteydi. Bu sonucun ardından 1997 yılında Rainer ve Storn bir makale [112] ile çalışmalarını duyurdular. Bu çalışmada DGA'nın başarımı birçok farklı fonksiyon üzerinde yapılan denemeler ile ortaya koyuldu. Price 1997 yılında ABD'de düzenlenen İkinci Uluslararası Evrimsel Optimizasyon Yarışmasına DGA ile katıldı. Yeterince katılım olmaması nedeniyle yarışma iptal edilmiş olsa da, mevcut katılımlar içerisinde DGA en iyi başarımı sunmaktaydı[113].

Bu tarihten sonra DGA yaygınlaşmış kimya, enerji, tasarım, kontrol, elektronik, imge işleme vb. bir çok alanda genellikle sürekli eniyileme problemlerinin çözümü amacıyla kullanılmıştır. Storn [114] ve Lampinen [115] tarafından hazırlanan web sayfalarından diferansiyel gelişim algoritması kullanılarak gerçekleştirilen kombinatöryel en iyileme alanı dışında kalan bazı çalışmalar listelenmektedir.

DGA temel olarak bir sürekli optimizasyon algoritması olarak tasarlanmıştır. Ancak sürekli optimizasyon alanındaki oldukça başarılı uygulamaları, kesikli/kombinatöryel optimizasyon problemler için de aynı başarı ile uygulanıp uygulanamayacağı sorusunu akla getirmektedir. Literatürde, bu tez çalışmasının konusuyla ilgili olan permütasyonel alanda yapılan çalışmaların sayısı oldukça azdır. Bu yönde yapılan çalışmalar aşağıda özetlenmektedir.

Nearchou ve Omirou [116] akış tipi çizelgeleme, en büyük ağırlıklı gecikme, ortak teslim tarihli çizelgeleme problemleri için DGA kullanmışlardır. Çalışmada gerçel sayı vektörlerinin sıralama vektörlerine dönüşümü amacıyla Bean [117] tarafından önerilen, Rasgele Sayı Kodlama Yöntemine (Random Keys Encoding) alternatif olarak, Alt Aralık Kodlama (Sub Range Encoding) kullanılmaktadır. Bu iki yöntem, 4.1.1. alt bölümünde irdelenmektedir. N_p adet D boyutlu gerçel sayı vektörleri ile çalışan DGA için, bu çalışmada durdurma kriteri olarak, vektör büyüklüğünün 500 katı (MAXI = D*500) alınmıştır. Elde edilen en iyi sıralamaya ulaşılan iterasyon sayısının (k_{opt}), toplam iterasyon sayısına (MAXI), bölümü (k_{opt}/MAXI) diğer yöntemlerle karşılaştırma aracı olarak kullanılmıştır. Diğer karşılaştırma aracı ise benzer çalışmalarda da yaygın olarak kullanılan bilinen en iyi çözüme yaklaşma yüzdesidir. Nearchou ve Omirou çalışmalarında, diferansiyel gelişim mutasyonu için klasik mutasyon denklemi (3.3) yerine farklı bir denklemi (3.8) kullanmaktadırlar.

$$v_{iG+1} = x_{ibest} + F.(x_{aG} + x_{bG} - x_{cG} - x_{dG}), \tag{3.8}$$

Burada, a,b,c,d [1,N_p] aralığında değişen birinden farklı tamsayılardır. F ise 0'dan büyük [0,2] aralığında değişen bir ölçek faktörüdür. Çalışmada, göze çarpan diğer bir önemli nokta ise çaprazlanmış aday vektörlerinin elemanlarından bazılarına belirli bir olasılıkla (0,1) aralığında değişen değerlerin eklenmesi yoluna gidilmesidir. Reel

sayı vektörleriyle optimizasyon algoritmalarında kullanılan bu mutasyon yöntemi ile yerel çözümlere sıkışmanın önüne geçilmeye çalışılmıştır. Ayrıca çalışmada CR çaprazlama oranı parametresini başlangıçta büyük alan ancak daha sonra yavaşça azalmasını sağlayan bir adaptif yapı da önerilmiştir.

Qian ve arkadaşları [118] akış tipi çizelgelemede çok amaçlı eniyileme için DGA kullanmışlardır. Bu çalışmada da gerçel sayı vektörleri üzerinden çalışan algoritmada değerlendirme aşamasında, sıralama vektörüne dönüşüm gerçekleştirilmektedir. Dönüşüm için farklı bir kural (En Büyük Sıralı Değer- Largest Order Value) tanımlandığı belirtilmiştir. Ancak yöntem Bean[58] tarafından verilen yöntemle aynı sırayı ters çevrilmiş bir şekilde vermektedir. Ayrıca yerel çözüme sıkışmaları önlemek için nokta değişim mutasyonu önermektedirler. Bu mutasyon tek bir birey için çoklu defalar denenebilmektedir. Ancak mutasyon gerçel sayı vektörü üzerinden değil, sıralama vektörü üzerinden gerçekleştirildiğinden, gerçel sayı vektörünün tutarlılığının sağlanabilmesi için bir tamir (repair) işlemi gerçekleştirilmekte ve ek hesaplama yükü getirmektedir.

Onwubolu ve Davendra [119] akış tipi çizelgelemede DGA'yı kullanmışlardır. Gerçel sayı vektörlerinden sıralama vektörlerine ve sıralama vektörlerinden gerçel sayı vektörlerine dönüşümü sağlayan bazı denklemler sunmuşlardır. Ancak yöntem, dönüşümler esnasında olursuz çözümlerin oluşmasına neden olabilmektedir. Çalışmada olurlu çözümlerin toplam denemelerin %10-15 ini oluşturduğu belirtilmiştir. Bu durumda oldukça yüksek bir hesaplama zamanı kaybı söz konusudur.

Nearchou [120] çember tipi hatların yerleşimi probleminin çözümü için DGA'yı kullanmıştır. Gerçel sayı vektörlerinin sıralama vektörlerine dönüşümü için Alt Aralık Kodlama yöntemini uygulamıştır.

Angira ve Babu [121] çalışmalarında süreç sentez ve tasarımında karşılaşılan bazı doğrusal ve konveks olmayan matematiksel programlama problemlerinin çözümü amacıyla DGA'yı kullanmışlardır. Problemlerde yer alan kısıtların aşılması durumu için amaç fonksiyonuna yüksek ceza maliyetleri eklenmesi yoluyla, problemin

çözümü yoluna gidilmiştir. Bu çalışmada dikkati çeken önemli nokta, ana populasyon ile aday populasyon bireylerinin ayrı ayrı dizilerde tutulduğu ve seçilimin bu iki dizinin bireylerini karşılıklı olarak değerlendirilmesi yoluyla gerçekleştirildiği klasik yöntemin aksine; bireyler tek bir dizide tutulmakta ve seçilim her birey için tek başına gerçekleştirilmektedir. Aday populasyondaki bir birey, aday birey ile hemen karşılaştırılmakta, eğer yeni birey daha iyi ise ana populasyona eklenmektedir. Böylece elde edilen yeni birey aynı iterasyondaki diğer aday bireylerin hesaplanması ve dolayısıyla ana populasyon bireylerinin oluşturulmasında kullanılabilmektedir. Bu durumun tutarlılığı azaltmadan yakınsama hızını arttırmakla birlikte, hesaplama maliyetini azalttığı vurgulanmaktadır. Ayrıca problemlerdeki 0-1 tamsayılı değişkenler, [0,1] aralığında değer alan gerçel sayı vektörünün değerinin 0.5'ten büyük olması durumunda 1 olarak; aksi takdirde 0 olarak değerlendirilmesi yoluyla çözümlenmiştir.

Nearchou [122] çalışmasında büyük hat dengeleme problemleri için DGA tabanlı bir sezgisel önermiştir. Çalışmada gerçel sayı vektörlerinin, sıralama vektörlerine dönüşümü için rasgele sayı dönüşümü (Random keys encoding) yöntemi kullanılmıştır. İşler arası öncelik kısıtlarının sağlanabilmesi amacıyla, elde edilen sıralama dizileri gerekli olması durumunda düzeltme işlemine tabi tutulmuşlardır. Bu çalışmada hat dengelemede kullanılmak üzere, diferansiyel mutasyon için yeni bir denklem (3.9) önerilmiştir.

$$v_{iG+1} = \xi \cdot x_{ibest} + (1-\xi) \cdot x_{aG} + F.(x_{bG} + x_{cG} - x_{dG} - x_{eG}), \qquad (3.9)$$

Burada, *a,b,c,d,e,* [1,N_p] aralığında değişen birbirinden farklı tamsayılardır. *F,* 0'dan büyük [0,2] aralığında değişen bir ölçek faktörü; ξ ise populasyondaki en iyi birey (x_{ibest}) ile rasgele seçilen bir başka bireyin (x_{aG}) konveks kombinasyonunu gösteren [0,1] aralığında değişen bir katsayıdır.

Al-Anzi ve Allahverdi [123] iki aşamalı montaj çizelgeleme problemleri için adaptif bir DGA kullanmışlardır. Sunulan algoritma sonuçları, PSO (Particle Swarm Optimization-Parçacık Sürü Optimizasyonu) meta-sezgiseli ve EDD (Early Due Date-En Erken Teslim Tarihi) sezgiseli ile elde edilen sonuçlarla karşılaştırılmıştır.

DGA'nın PSO ile başa baş sonuçlar vermesine karşılık, yaklaşık üçte bir zamanda aynı sonuçlara ulaştığı gözlemlenmiştir.

Schmidt ve Thierauf [124] yapı mühendisliğinde karşılaşılan doğrusal olmayan, konveks olmayan, sürekli ve kesikli değişkenlerin bulunabileceği problemlerin çözümünde DGA'yı , Eşik Değeri Kabulü (EDK-Treshold Accepting) isimli tavlama benzetimi algoritmasına benzeyen bir algoritma ile birlikte hibrit olarak kullanmıştır. Çalışmada *F* değerinin hesaplanmasında Zaharie[125]'nin önerdiği (3.10) denklemi kullanılmıştır.

$$F = \sqrt{1/N_p - CR/(2N_p)} \tag{3.10}$$

Nearchou [126] ortak teslim tarihli çizelgeleme problemleri için gecikme ve erken bitirme değerlerinin ağırlıklı toplamını enküçüklemek amacıyla DGA'yı kullanmıştır. Çalışmada mutasyon vektörü olarak (3.3) denklemi kullanılmıştır. CR ve F değerlerinin belirlenebilmesi için farklı sabit değerlerinin karşılaştırıldığı deneyler yapılmış, F değerinin hesaplanması için CR ve Np ye bağlı olarak çalışan Zaharie [125] tarafından önerilen (3.10) denklemi de kullanılmış ve en iyi sonuçların Zaharie denklemi kullanıldığında elde edildiği gözlemlenmiştir. Gerçel sayı vektörlerinin tamsayıya dönüştürülmesinde ise incelenen problemin özel bir tip çizelgeleme problemi olması nedeniyle işlerin erken ya da geç gerçekleştirilecek işler olarak sınıflandırılması gerektiğinden, 0.5 ten büyük olan işleri geç; diğer işleri ise erken olarak sınıflandıran, oldukça basit bir yaklaşım sergilenmiştir. Sıralama için ise, erken ve geç işleri kendi içerinde sınırlayan ve önceliklendiren probleme özgü bir yaklaşım kullanılmıştır. Bu yaklaşım, basit (0,1) tipi problemler için uygun olmasına karşın, genel kombinatöryel problemler için kullanılabilir değildir.

3.4. Literatür Özeti

Kesikli problemlerin çözümünde DGA literatürüne katkı yapan çalışmalar; sürekli vektörlerin sıralama vektörlerine dönüştürülmesi, ile yerel arama ve mutasyon işlemleri başlıklarında karşılaştırlabilir.

3.4.1. Sıralama vektörüne dönüşüm

Gerçel sayı vektörlerinin sıralama vektörlerine dönüşümü için DGA literatüründe farklı kodlama yöntemleri yer almaktadır. Bean [58] tarafından önerilen Rasgele Sayı Kodlaması Yöntemi (Random Keys Encoding), Nearchou [120] tarafından önerilen Alt Aralık Kodlama Yöntemi (Sub Range Encoding) ve İndeks Sıralama Kodlama yöntemleri, düzeltme, yeniden işleme gibi adımlara ihtiyaç duyulmaksızın sıralama vektörlerine dönüştürülebildikleri için bu çalışmada aday yöntemler olarak kabul edilmişlerdir. Bunların dışında Onwubolu ve Davendra [119] tarafından önerilen bir başka yöntem de mevcuttur ancak bu yöntem sadece %15-20 oranında sıralama vektörüne başarılı dönüşüm sağlamakta, büyük bir işlem gücü kaybına neden olmaktadır, bu nedenle aday yöntemler arasında alınmamıştır. Sözü edilen dönüşüm yöntemleri 4. bölümde örnekleriyle birlikte ayrıntılı bir şekilde anlatılmıştır.

3.4.2. DGA mutasyon formülü

Literatürde DGA'nın kendine özel mutasyon işlemi için kullanılan (3.3) formülüne alternatif olarak önerilen birçok farklı formül [127] bulunmasına karşın, permütasyonel alanda yapılan çalışmalar incelendiğinde, sadece Nearchou tarafından önerilen (3.8) formülü olduğu görülmektedir. GSP için daha uygun olan formül 4. bölümde anlatıldığı şekilde deneysel bir çalışma ile belirlenmiştir.

4. GELİŞTİRİLMİŞ DİFERANSİYEL GELİŞİM ALGORİTMASI

Diferansiyel Gelişim Algoritması (DGA), ağırlıklı olarak sürekli eniyileme problemlerinin çözümünde kullanılan yeni bir çözüm yöntemidir. Sürekli eniyileme problemlerine yönelik olarak geliştirilmiş olduğundan, kesikli ya da permütasyonel eniyileme problemleri için etkin uygulamaları bulunmamaktadır. Bu çalışmada DGA permütasyonel eniyileme problemlerinde kullanılmak üzere yeniden düzenlenmiş ve geliştirilmiştir. Önerilen algoritmanın başarımı Gezgin Satıcı Problemi (GSP) literatür problemleri kullanılarak değerlendirilmiştir.

Bu bölümde ilk olarak önerilen Geliştirilmiş Diferansiyel Gelişim Algoritmasının (GDGA), klasik DGA'dan farklı bölümleri ayrıntılarıyla açıklanarak anlatılmıştır. Daha sonra GDGA'nın TSP problemleri üzerinde başarımının en üst düzeyde olması için uygun yöntem ve parametre seviyelerinin belirlenmesi amacıyla yapılan deneysel çalışma anlatılmış ve sonuçları değerlendirilmiştir. Son bölümde deneyler sonucunda en iyi yöntem ve parametre seviyeleri belirlenen algoritmanın literatür problemleri üzerine uygulanması ve elde edilen sonuçlar anlatılmıştır.

4.1. DGA'dan GDGA'ya Gelişim

DGA'nın permütasyonel eniyilemede kullanılabilmesi amacıyla bazı temel değişiklikler yapılmıştır. En önemli değişiklik gerçel sayılardan oluşan DGA vektörlerinin sıralama değişkenlerine dönüştürülmesi ve sıralama değişkenlerinin tekrar gerçel sayılara dönüştürülmesi aşamalarında gerçekleştirilmiştir. Ayrıca algoritma performansını geliştirmek amacıyla, permütasyonel problemlerle ilgili DGA kullanılarak yapılan diğer çalışmalardan farklı olarak, yöntemin yapısına uygun bir yerel arama eklenmiştir. Bu durumda algoritma Şekil 4.1'de görülen adımlardan oluşmaktadır.

Geliştirilmiş Diferansiyel Gelişim Algoritması (GDGA) adımları DGA'ya oldukça benzemektedir. Farklılıklar, populasyon uygunluğunun sıralama vektörü (gezgin satıcı turu) üzerinden değerlendirilmesi amacıyla eklenen, sıralama vektörüne dönüştürme işleminin gerçekleştirildiği 3,7 ve 10. adımlar; yerel aramanın sıralama vektörleri üzerinden gerçekleştirildiği 11.adım; yerel arama sonuçlarının değerlendirildiği ve ek bir seçilimin gerçekleştirildiği 12. ve 13 adımlar ve elde edilen sıralama vektörlerinin gerçel sayı vektörlerine geri dönüştürüldüğü 14. adımdır. İkinci seçilim mantık ve yöntem olarak DGA da kullanılan seçilim işlemiyle aynıdır ancak 11. adımdaki yerel arama işlemiyle elde edilebilecek daha iyi bireylerin populasyona eklenebilmesine imkan vermektedir. Farklılıklar takip eden başlıklarda ayrıntılarıyla birlikte açıklanmıştır.

1. BAŞLA
2. Rasgele Bireyler Oluştur
3. Sıralama Vektörlerine Dönüştür
4. İlk Populasyon Uygunluğunu Değerlendir

    WHILE (Döngü Sonlandırma Kriteri = = 0)

    5. DGA Mutasyon
    6. DGA Çaprazlama
    7. Sıralama Vektörlerine Dönüştür
    8. Aday Populasyon Uygunluğunu Değerlendir
    9. Seçilim 1
    10. Sıralama Vektörlerine Dönüştür
    11. Yerel Arama
    12. Aday Populasyon Uygunluğunu Değerlendir
    13. Seçilim 2
    14. Gerçel Sayı Vektörlerine Geri Dönüştür

    END

16. BİTİR

Şekil 4.1: GDGA Algoritması

4.1.1. Sıralama Vektörüne Dönüşüm

Gerçel sayı formatında çalışmak üzere tasarlanan DGA'nın mevcut çözüm başarımının değerlendirilmesi için sıralama vektörüne (gezgin satıcı turuna) çevrilmesi gerekmektedir. Bu çevrimde kullanılacak kodlama yönteminin, DGA başarımının permütasyonel uygulamalara aktarılmasında önemli olabileceği düşünülmektedir. Literatürde geçen üç yöntem, İndeks Sıralama Kodlama Yöntemi, Rasgele Sayı Kodlama Yöntemi, Alt Aralık Kodlama Yöntemleri arasında dönüşüm için daha uygun olanın belirlenmesi gerekmektedir. Örnek bir gerçel sayı vektörü A= [0.23 0.82 0.03 0.47 0.11 0.15 0.68], Tablo 4.1'de görüldüğü gibi sıralama vektörlerine dönüştürülebilmektedir.

Tablo 4.1: DGA Kodlama Dönüşümleri

Gerçel Sayı Vektörü	0.23	0.82	0.03	0.47	0.11	0.15	0.68
İndeks Sıralama Kodlama	4	7	1	5	2	3	6
Alt Aralık Kodlama	2	6	1	4	3	7	5
Rasgele Sayı Kodlama	3	5	6	1	4	7	2

4.1.1.1. İndeks sıralama kodlama

Bu yöntem oldukça basit bir kodlama yöntemidir. Gerçel sayı vektörünün küçükten büyüğe sıralanması durumunda vektör elemanlarının alacağı indeks değerleri, sıralama vektörü olarak alınır. Tablo 4.1'de görülen gerçel sayı vektöründe 0.03 değeri küçükten büyüğe sıralamada 1. sırada yer aldığı ve 0.03 değeri gerçel sayı vektöründe 3. eleman olduğundan, vektörün bir gezgin satıcı turunu gösterdiği düşünülürse, 1. şehir 3. sırada yer alacaktır. Benzer şekilde 2 şehir 5. sırada yer almakta ve diğer şehirler de yerleştirilerek, tur tamamlanmaktadır.

4.1.1.2. Alt aralık kodlama

Alt aralık kodlama yönteminde gerçel sayı vektöründe elemanların değer aldığı aralık, şehir sayısı kadar eşit sayıya bölünmektedir. Daha sonra vektör değerleri karşılık geldikleri aralığa göre şehir numarası almaktadırlar. Tablo 4.1'de gösterilen

örnekte 7 şehir bulunduğundan, 7 adet aralığa ihtiyaç duyulmaktadır. Bu durumda aralık vektörü 0'dan başlamak üzere B = [0 1/7 2/7 3/7 4/7 5/7 6/7 7/7] olarak belirlenir. Gerçel sayı karşılığı B = [0 0.14 0.29 0.43 0.57 0.71 0.86 1] olarak hesaplanır. Gerçel sayı vektöründe ilk sıradaki 0.23 değeri 2. aralığa karşılık geldiğinden turda ilk sırayı 2. şehir almaktadır. Benzer şekilde diğer elemanlar da karşılık geldikleri aralığın numarasına göre çevrime uğratıldığında ortaya yeni sıralama (tur) vektörü C = [2 6 1 4 1 2 5] çıkmaktadır. C vektörü, 1 ve 2 elemanlarının tekrar etmesi nedeniyle, mevcut haliyle bir sıralama vektörü değildir. Sırama vektörüne dönüştürülmesi için bir düzeltme yapılması gerekmektedir. Alt Aralık Kodlama Yöntemini öneren Nearchou [120], tekrar edilen elemanların belirlenerek yerine, vektörde eksik olan elamanların yerleştirilmesini önermektedir. Bu kurala göre, tekrar edilen 1 ve 2 elemanları çıkarıldıktan sonra yerlerine vektörde bulunması gereken 3 ve 7 elemanları yerleştirilerek en son sıralama vektörü D = [2 6 1 4 3 7 5] elde edilir. Sonradan tekrar eden elemanların yerine eklenen eksik elemanların (şehirlerin) yerleştirilmesi için Nearchou bir çalışmasında [120] artan sırada yerleştirmeyi önerirken bir başka çalışmasında [116], rasgele sırada yerleştirmeyi önermektedir.

4.1.1.3. Rasgele sayı kodlama

Rasgele sayı kodlama yönteminin uygulanabilmesi için öncelikle sıralama indeksinin belirlenmiş olması gerekmektedir. Sıralama indeksi değerlerinin, vektör içerisinde bulundukları sıranın (şehrin) önceliğini gösterdiği düşünülürse, sıralar (şehirler) öncelik vektörlerine sıralanarak tur oluşturulur. Tablo 4.1'de görülen gerçel sayı vektöründe 3. sıradaki (şehirdeki) 0.03 değeri, indeks sıralamada 1 değerini aldığından, 3. şehir 1. sırada yer alacaktır. Benzer şekilde 0.47 değeri, yani 4. sıra (şehir), 5. önceliğe sahip olduğundan; 4 şehri 5. sırada yer almakta ve diğer şehirler de yerleştirilerek, tur tamamlanmaktadır. Bu kodlama yönteminde şehirler (sıralar) sabittir, ancak öncelik değerlerini gösteren gerçel sayı vektörü ve dolayısıyla indeks değerleri değişerek şehrin turdaki önceliğini belirlemektedirler.

4.1.2. Gerçel sayı vektörüne dönüşüm

DGA performansının geliştirilmesi amacıyla yerel arama kullanılmıştır. Ancak yerel aramanın gerçel sayı vektörleri üzerinden gerçekleştirilmesinin bazı sakıncaları söz konusudur. En önemli sakınca, kodlama yöntemlerinin gerçel sayı vektörünü farklı şekillerde değerlendirmeleri nedeniyle, gerçel sayı üzerinde yapılmak üzere seçilecek yerel arama hareketinin, farklı sonuçlar doğurabilmesi olasılığıdır. Bazı yerel arama hareketlerinin belirli kodlama yöntemleri için avantajlı olması, diğerleri için ise yetersiz kalması mümkündür. Bu durumda kodlama yönteminin yerel arama ile etkileşimini doğru analiz etmek zorlaşacaktır. Ayrıca gerçel sayı vektörü üzerinde yapılan bir değişikliğin sıralama vektörü üzerindeki etkisi, ancak dönüşüm sırasında görülebilecek, bu durum yerel aramalarda kullanılabilecek olası stratejilerin başarımını zorlaştıracak ve iş yükünü arttıracaktır.

Bu çalışmada, yukarıda anlatılan sakıncaların önüne geçebilmek amacıyla, yerel arama işleminin sıralama vektörleri üzerinden gerçekleştirilmesi yolu tercih edilmiştir. Ancak sıralama vektörüne dönüşümde veri kaybı oluşmaması için her bir kodlama yöntemi için ayrı olmak üzere, “Karşılığını Bularak Geri Dönüşüm” ve “Normalizasyon ile Geri Dönüşüm” yöntemleri önerilmiş ve uygun olanın belirlenmesi deneysel çalışma sonucuna bırakılmıştır. Aşağıda bu iki yöntem her kodlama yöntemi için ayrıntılarıyla anlatılmıştır.

4.1.2.1. Karşılığını bularak geri dönüşüm

Bu yöntemde, yerel arama sonucu oluşan yeni sıralama vektörü eski sıralama vektörden daha iyi bir sonuç vermişse, eski sıralama vektörü oluşurken gerçel vektör değerleri hangi sıralama vektörü değerine karşılık gelmişse, yeni vektörde aynı sıralama vektörü değerine karşılık geleceği şekilde gerçel sayı vektörü yeniden oluşturulur. Yöntem, işlemsel olarak ek bir yük getirse de, veri kaybına neden olmaksızın gerçel sayı vektörüne geri dönüşümü mümkün kılmaktadır. Kodlama yöntemleri farklı mantıklara dayandığından geri dönüşüm işlemleri, her yöntem için farklı şekilde gerçekleştirilmektedir. Sıralama vektörüne dönüşümde, yapılan işlemin bellekte saklanması yerine, gerçel değerlerin hangi sıralama değerlerine karşılık

gelebilecekleri sorgulanarak, yeni gerçel sayı vektörü belirlenmektedir. Bu şekilde elde edilen yeni gerçel sayı vektörünün, sıralama vektörüne çevrilme işlemine yeniden tabi tutulması durumunda, yeni sıralama vektörünün bulunması gerekmektedir.

Tablo 4.1'de anlatılan dönüşüm örneğinde sıralama vektörlerinin 4. ve 7. elemanları ve aralarındaki elemanlarının sıralarının ters çevrilmesinin, bütün kodlama yöntemleri için daha iyi amaç fonksiyonu değeri elde edilmesini sağladığı kabul edilirse dönüşüm Tablo 4.2 , Tablo 4.3 ve Tablo 4.4'te görüldüğü gibi gerçekleşir.

İndeks kodlama yöntemi için, gerçel sayı vektöründeki değişim sıralama vektöründeki değişimle aynı şekilde gerçekleşir. Gerçel sayı vektöründe 4. ile 7. dahil olmak üzere aralarında kalan elemanların sıralamaları ters çevrildiğinde, yeni sıralama vektörüne ait gerçel sayı vektörü elde edilmiş olur. Bu işlemin sıralama vektörünü değiştiren hareketin bellekte tutulması yoluyla yapılması, birden çok değişiklik yapılması söz konusu olduğunda oldukça karmaşıklaşacağından, gerçel değerlerin hangi sıralama değerlerine karşılık gelebilecekleri sorgulanarak yapılır. Örneğin, Tablo 4.2'de görüldüğü gibi 0.11 değeri küçükten büyüğe sıralamada 2. sırada yer aldığı ve yeni vektörde 2 elemanı 7. sırada yer aldığı için; 0.11 değeri 2'nin karşısına 7. sıraya yazılır. Benzer şekilde, sıralamada 6. sırada yer alan 0.68 değeri yeni sıralama vektöründe 6'nın karşısına, 4. sıraya yazılır ve diğer sayılar da karşılık geldikleri yerlere yazılarak yeni gerçel sayı vektörü oluşturulur.

Tablo 4.2: İndeks Sıralama Yöntemi için Karşılığını Bularak Gerçel Sayı Vektörlerine Dönüşüm

	Yeni vektör	Eski vektör
Gerçel Sayı Vektörü	0.23 0.82 0.03 0.68 0.15 0.11 0.47	0.23 0.82 0.03 0.47 0.11 0.15 0.68
İndeks Sıralama Kodlama	4 7 1 6 3 2 5	4 7 1 5 2 3 6

Alt Aralık Kodlama yönteminde gerçel sayı vektörüne geri dönüşümün sağlıklı yapılabilmesi için, sıralama vektörüne dönüşüm işleminde tekrar eden elemanların dönüştürülmesi aşamasında, rasgele sıralama kullanılması halinde, tutarsızlıklar oluşması ve elde edilen iyi çözümlerin kaybedilmesi söz konusudur. Bu durumda, yeni gerçel sayı vektörünün sıralama vektörüne dönüştürme işlemine tabi tutulması

durumunda, yerel arama ile daha iyi olduğu belirlenen sıralama vektöründen farklı bir vektöre ulaşılabilir. Bu nedenle, tekrar eden elemanların dönüştürülmesinde küçükten büyüğe sıralama yönteminin kullanılmasına karar verilmiştir.

Tablo 4.3: Alt Aralık Kodlama Yöntemi için Karşılığını Bularak Gerçel Sayı Vektörlerine Dönüşüm

	Yeni vektör	Eski vektör
Gerçel Sayı Vektörü	0.23 0.82 0.03 0.68 0.15 0.11 0.47	0.23 0.82 0.03 0.47 0.11 0.15 0.68
Alt Aralıklar	0 0.14 0.29 0.43 0.57 0.71 0.86 1	
Alt Aralık Kodlama	2 6 1 5 7 3 4	2 6 1 4 3 7 5

Tablo 4.3'te görüldüğü gibi sıralama vektörünün ilk sırasında 2 elemanı yer aldığından 2. aralığa düşen gerçel sayı değerlerinden biri yeni gerçel vektörün ilk elemanı olarak belirlenir. Aynı aralığa düşen değerler arasından hangisinin seçileceği dönüşüm sonucunu etkilemeyecektir. 2. aralık olan 0.14-0.29 aralığına düşen 0.15 ve 0.23 değerlerinden 0.23 yeni değer olarak belirlenir, 0.15 boş kalan araklıklar için kullanılmak üzere saklanır. 6. aralığa karşılık düşen 0.82 değeri alınır. 1. aralığa düşen 0.03 değeri alınır, 0.11 saklanır. 5. aralığa düşen 0.68 değeri alınır. 7. ve 3. aralığa düşen değer olmadığından atlanır. 4. aralığa düşen 0.47 değeri gerçel sayı vektörünün en son değeri olarak belirlenir. Saklanan 0.15 ve 0.11 değerleri, karşılığı olmayan 7 ve 3 değerlerinin karşısına, büyük değer büyüğe (0.15, 7'ye), küçük değer küçüğe sıralamaya uygun (0.11, 3'e) karşılık gelecek şekilde yerleştirilerek yeni vektör elde edilir.

Rasgele sayı kodlamasında, yeni sıralama vektöründeki değerlerin kendilerinin yeni gerçel sayı vektöründeki yerini, mevcut yerlerinin ise küçükten büyüğe sıralamadaki indeks değerini gösterdiği düşünülerek dönüşüm gerçekleştirilir. Örneğin Tablo 4.4'de görüldüğü gibi, yeni sıralama vektöründe yer alan 2 değeri 4. sırada yer aldığından, küçükten büyüğe sıralamada (İndeks Kodlama) 4. sırada yer alan 0.23 değeri yeni vektörde 2. sıraya yerleşir. Aynı şekilde 7 değeri 5. sırada olduğundan, indeks sıralamada 5. sırada olan 0.47 değeri 7. sıraya ve diğer sayılar da karşılık geldikleri yerlere yazılarak yeni gerçel sayı vektörü oluşturulur.

Tablo 4.4: Rasgele Sayı Kodlama Yöntemi için Karşılığını Bularak Gerçel Sayı Vektörlerine Dönüşüm

	Yeni vektör	Eski vektör
Gerçel Sayı Vektörü	0.82 0.23 0.03 0.68 0.11 0.15 0.47	0.23 0.82 0.03 0.47 0.11 0.15 0.68
İndeks Kodlama		4 7 1 5 2 3 6
Rasgele Sayı Kodlama	3 5 6 2 7 4 1	3 5 6 1 4 7 2

4.1.2.2. Normalizasyon ile geri dönüşüm

Klasik DGA'da gerçel sayılardan oluşan vektör elemanlarının belirlenen sınırları aşması durumunda, o vektör elemanının değerinin sınır değeri ile değiştirilmesi yolu tercih edilmektedir. Ancak vektörler elemanlarının sıralama amacıyla kullanılacağı düşünülürse, bu yöntemin kullanılması durumunda, birbiriyle aynı önem değerine sahip birden çok eleman ortaya çıkması ve dolayısıyla sonuçların sağlıksız olması mümkündür. Bunun yerine gerçel vektöre karşılık gelen sıralama değerlerinin her yöntem için uygun olan şekilde vektör büyüklüğü D'ye bölünmesi yoluyla normalize edilerek [0,1] aralığına içinde tutulması mümkündür. Böylece, gerçel sayı vektörlerinin sıralama vektörlerine benzer bir düzende olmalarının sağlanması, gerçel sayı uzayındaki arama alanının daraltılması ve çaprazlama işlemindeki başarımın arttırılması hedeflenmektedir. Yöntem, karşılığına bularak dönüştürmeye nazaran çok daha kolaydır ve daha az işlem gücü gerektirmektedir.

İndeks sıralama yöntemi için yeni gerçel sayı vektörü, denklem 4.1'de görüldüğü gibi yeni sıralama vektörünün ($S_{k,i}$) tüm elemanlarının dizi büyüklüğüne (D) bölünmesi ile hesaplanır. Tablo 4.5'te görülen yeni sıralama vektörünün karşılığı olan gerçel sayı vektörü tüm elemanların 7'ye bölünmesi yoluyla [4/7 7/7 1/7 6/7 3/7 2/7 5/7] belirlenir.

Tablo 4.5: İndeks Sıralama Yöntemi için Normalizasyon ile Gerçel Sayı Vektörlerine Dönüşüm

	Yeni vektör	Eski vektör
Gerçel Sayı Vektörü	0.57 1 0.14 0.86 0.43 0.29 0.71	0.23 0.82 0.03 0.47 0.11 0.15 0.68
İndeks Sıralama Kodlama	4 7 1 6 3 2 5	4 7 1 5 2 3 6

$$s_{k,i} = s_{k,iN} / D \tag{4.1}$$

Alt aralık sıralama yöntemi için yeni gerçel sayı vektörü, denklem 4.2'te görüldüğü gibi yeni sıralama vektörünün ($S_{k,i}$) tüm elemanlarının dizi büyüklüğüne (D) bölünmesi ve bulunan değerlerden aralık genişliğinin yarısının çıkarılması yoluyla hesaplanır.

$$s_{k,i} = (s_{k,iN} / D) - 1/2D \tag{4.2}$$

Tablo 4.6'da görülen yeni sıralama vektörünün karşılığı olan gerçel sayı vektörü tüm elemanların 7'ye bölünmesi ve her birine aralık değerinin yarısının (1/14) çıkarılması yoluyla [(2/7-1/14) (6/7-1/14) (1/7-1/14) (5/7-1/14) (7/7-1/14) (3/7-1/14) (4/5-1/14)] belirlenir. Böylece yeni gerçel sayı değerleri aralık değerlerinin tam ortasında yer alacak şekilde hesaplanmış olur.

Tablo 4.6: Alt Aralık Kodlama Yöntemi için Normalizasyon ile Gerçel Sayı Vektörlerine Dönüşüm

	Yeni vektör	Eski vektör
Gerçel Sayı Vektörü	0.21 0.79 0.07 0.64 0.93 0.36 0.50	0.23 0.82 0.03 0.47 0.11 0.15 0.68
Alt Aralıklar	0 0.14 0.29 0.43 0.57 0.71 0.86 1	
Alt Aralık Kodlama	2 6 1 5 7 3 4	2 6 1 4 3 7 5

Rasgele sayı kodalama yöntemi için yeni gerçel sayı vektörü, denklem 4.3'de görüldüğü gibi yeni sıralama vektörüne karşılık gelen, indeks vektörünün($\dot{I}_{k,i}$) tüm elemanlarının dizi büyüklüğüne (D) bölünmesi ile hesaplanır.

$$s_{k,iG} = i_{k,iG} / D \tag{4.3}$$

Tablo 4.7'te görülen yeni sıralama vektörünün karşılığı olan indeks sıralama vektörünün tüm elemanların 7'ye bölünmesi yoluyla [7/7 4/7 1/7 6/7 2/7 3/7 5/5] gerçel sayı vektörü belirlenir. İndeks sıralama vektörünün oluşturulmasına, rasgele sayı kodlamada 1. sırada yer alan 3 değeri nedeniyle, 3. sıraya 1 değeri yazılarak

başlanır, sonrasında 2. sırada yer alan 5 değeri nedeniyle 5. sıraya 2 yazılarak devam edilir ve benzer şekilde tüm vektör oluşturulur.

Tablo 4.7: Rasgele Sayı Kodlama Yöntemi için Normalizasyon ile Gerçel Sayı Vektörlerine Dönüşüm

	Yeni vektör	Eski vektör
Gerçel Sayı Vektörü	1 0.57 0.14 0.86 0.29 0.43 0.71	0.23 0.82 0.03 0.47 0.11 0.15 0.68
İndeks Sıralama	7 4 1 6 2 3 5	4 7 1 5 2 3 6
Rasgele Sayı Kodlama	3 5 6 2 7 4 1	3 5 6 1 4 7 2

4.1.3. Yerel arama

Sıralama problemlerinde daha iyi sonuçlar elde edebilmek amacıyla kullanılan, bir çok yerel arama yöntemi bulunmaktadır. Bu yöntemler, vektörün belirli bir kısmının kesilerek başka bir yerine eklenmesi, kısmi sıralama değişiklikleri ya da vektörün bir bölümünün tersine çevrilerek tekrar vektör içine eklenmesi ve benzeri bir çok çeşidi içermektedir. Bu çalışmada, daha önce yapılan çalışmalarda GSP problemleri için oldukça başarılı olduğu görülen ters çevirme (inversion) yöntemi kullanılmaktadır. Yöntemde vektörde rasgele iki nokta seçilmekte ve noktalar arasındaki elemanların sıralaması, seçilen noktalar da dahil olmak üzere, ters çevrilmektedir. Ters çevirme hareketi, sadece 2 bağ koparılıp yeni 2 bağ oluşturulduğundan, GSP için geliştirme hareketi olarak daha önce anlatılan 2-opt hareketiyle aynıdır. GSP'de olabilecek en az sayıda bağ koparılarak yeniden oluşturulduğu için en basit seviyedeki geliştirme hareketidir ve diğer geliştirme hareketlerinin 2-opt hareketlerinin kombinasyonu ile oluşturması mümkündür.

DGA ile ilgili gerçel sayılar kullanılarak permütasyonel alanda yapılan çalışmalarda yerel arama bu anlamda daha önce kullanılmamıştır. Bunun temel nedeni, bahsedilen yerel arama hareketlerinin gerçel sayı vektörleri üzerinde uygulanması durumunda tutarsızlıkların oluşabilmesi; sıralama vektörleri üzerinde yapılması durumunda ise geriye dönüşüm için uygun yöntemlerin öne sürülememiş olmasıdır. 4.1.2 alt bölümünde anlatılan yöntemler aracılığıyla bu problemin önüne geçilmesi amaçlanmıştır.

4.2. Deneysel Çalışma

Alt Bölüm 3.2'de anlatılan kontrol parametreleri ile bu bölümde anlatılan algoritma değişiklikleri arasında nasıl bir etkileşim olacağı benzer bir çalışma bulunmadığından bilinmemektedir. Bu etkileşimin gözlenebilmesi, yöntem ve parametre düzeylerinin en iyi seçimlerinin belirlenebilmesi için deneysel bir çalışma gerçekleştirilmiştir.

Deneysel çalışmada gerçelden sıralama vektörüne ve sıralamadan gerçele dönüşüm için 3 yöntem; sıralamadan gerçele dönerken karşılığını bularak ve normalize ederek olmak üzere 2 yöntem; yerel arama oranı için 3 seviye (0, 0.5, 1); mutasyon ölçekleme faktörü F için 5 seviye (0, 0.5, 1, 1.5, 2); çaprazlama olasılığı CR için 3 seviye (0, 0.5, 1); populasyon büyüklüğü Np için 3 seviye (D, 2D, 3D) olmak üzere tam faktöryel, 3 tekrarlı deneme yapılmıştır. Problem büyüklüğündeki değişikliğin sonuç üzerinde etkisini gözlemleyebilmek için deneyler iki farklı literatür problemi bayg29 ve att48 üzerinde gerçekleştirilmiştir. Durdurma kriterinin, sonuç üzerindeki etkisini en aza indirebilmek amacıyla maksimum iterasyon sayısı oldukça büyük bir değer (300*D) belirlenmiştir. Bu durumda toplamda 4860 adet deney düzenlenerek sonuçları incelenmiştir. Deneyler Tablo 4.8'de özetlenmiştir.

Tablo 4.8: GDGA Deneyleri

Faktör Adı	Yöntemler / Seviyeler	Sayı
Gerçel Sayı Vektöründen Sıralama Vektörüne Dönüşüm Yöntemi (scv)	İndeks Alt Aralık Rasgele Sayılı	3
Sıralamadan Gerçele Dönüşüm Yöntemi (sr2gr)	Karşılığını bularak Normalize ederek	2
Yerel Arama Oranı (ya_orn)	(0, 0.5, 1)	3
Populasyon Büyüklüğü (Np)	(D, 2D, 3D)	3
Çaprazlama Olasılığı (CR)	(0, 0.5, 1)	3
Mutasyon Ölçekleme Faktörü (F)	(0, 0.5, 1, 1.5, 2)	5
Literatür Problemleri	bayg29 att48	2
Tekrar		3
Toplam	3*2*3*3*3*5*2*3	4860

Tablo 4.9: Bayg29 problemi için varyans analizi sonuçları

Kaynak	Kareler Toplamı	Serbest. Derec.	Kareler Ortalaması	F	p
scv	36.623	2	18.312	2902.5	0
sr2gr	5.2761	1	5.2761	836.29	0
ya_orn	340.56	2	170.28	26990	0
Np	1.2768	2	0.63839	101.19	0
CR	58.032	2	29.016	4599.2	0
F	2.5383	4	0.63459	100.59	0
scv*sr2gr	7.7775	2	3.8887	616.39	0
scv*ya_orn	48.391	4	12.098	1917.6	0
scv*Np	0.028301	4	0.0070754	1.1215	0.34437
scv*CR	9.1371	4	2.2843	362.07	0
scv*F	2.1535	8	0.26919	42.668	0
sr2gr*ya_orn	0.25191	2	0.12595	19.964	0
sr2gr*Np	0.022061	2	0.011031	1.7484	0.17417
sr2gr*CR	0.3969	2	0.19845	31.455	0
sr2gr*F	0.16476	4	0.041189	6.5287	0
ya_orn*Np	1.2283	4	0.30707	48.672	0
ya_orn*CR	74.601	4	18.65	2956.1	0
ya_orn*F	3.1851	8	0.39813	63.106	0
Np*CR	0.28341	4	0.070852	11.23	0
Np*F	0.036188	8	0.0045235	0.71699	0.67677
CR*F	16.666	8	2.0833	330.21	0
scv*sr2gr*ya_orn	1.7114	4	0.42786	67.817	0
scv*sr2gr*Np	0.097503	4	0.024376	3.8637	0.00388
scv*sr2gr*CR	1.2669	4	0.31672	50.202	0
scv*sr2gr*F	0.256	8	0.032	5.0722	0
scv*ya_orn*Np	0.17575	8	0.021969	3.4822	0.00051
scv*ya_orn*CR	4.6905	8	0.58631	92.933	0
scv*ya_orn*F	4.0779	16	0.25487	40.398	0
scv*Np*CR	0.11668	8	0.014586	2.3119	0.01797
scv*Np*F	0.041333	16	0.0025833	0.40947	0.98089
scv*CR*F	1.6321	16	0.10201	16.169	0
sr2gr*ya_orn*Np	0.030959	4	0.0077398	1.2268	0.29712
sr2gr*ya_orn*CR	0.91791	4	0.22948	36.373	0
sr2gr*ya_orn*F	0.12128	8	0.01516	2.4029	0.01385
sr2gr*Np*CR	0.013114	4	0.0032786	0.51967	0.72129
sr2gr*Np*F	0.051866	8	0.0064833	1.0276	0.41237
sr2gr*CR*F	0.17175	8	0.021469	3.4029	0.00066
ya_orn*Np*CR	0.4511	8	0.056387	8.9376	0
ya_orn*Np*F	0.10805	16	0.0067533	1.0704	0.3778
ya_orn*CR*F	17.343	16	1.0839	171.81	0
Np*CR*F	0.13099	16	0.0081867	1.2976	0.18845
Hata	28.983	4594	0.006309		
TOPLAM	671.02	4859			

Tablo 4.10: Att48 problemi için varyans analizi sonuçları

Kaynak	Kareler Toplamı	Serbest. Derec.	Kareler Ortalaması	F	p
scv	144.6	2	72.299	846.92	0
sr2gr	31.541	1	31.541	369.47	0
ya_orn	910.45	2	455.22	5332.5	0
Np	22.833	1	22.833	267.47	0
CR	142.79	2	71.395	836.33	0
F	3.3067	4	0.82667	9.6837	0
scv*sr2gr	41.752	2	20.876	244.55	0
scv*ya_orn	64.072	4	16.018	187.64	0
scv*Np	9.7397	2	4.8698	57.046	0
scv*CR	36.85	4	9.2124	107.92	0
scv*F	8.1239	8	1.0155	11.896	0
sr2gr*ya_orn	6.7217	2	3.3609	39.369	0
sr2gr*Np	0.72471	1	0.72471	8.4894	0.00359
sr2gr*CR	0.53857	2	0.26928	3.1544	0.04280
sr2gr*F	0.23886	4	0.059715	0.69951	0.59224
ya_orn*Np	36.119	2	18.06	211.55	0
ya_orn*CR	175.42	4	43.855	513.72	0
ya_orn*F	4.6598	8	0.58248	6.8232	0
Np*CR	2.5685	2	1.2842	15.044	0
Np*F	0.043282	4	0.01082	0.12675	0.9728
CR*F	26.802	8	3.3503	39.246	0
scv*sr2gr*ya_orn	31.344	4	7.8359	91.791	0
scv*sr2gr*Np	2.3011	2	1.1506	13.478	0
scv*sr2gr*CR	3.8598	4	0.96494	11.303	0
scv*sr2gr*F	1.3945	8	0.17431	2.0418	0.03819
scv*ya_orn*Np	21.709	4	5.4272	63.575	0
scv*ya_orn*CR	22.211	8	2.7764	32.523	0
scv*ya_orn*F	12.042	16	0.7526	8.8161	0
scv*Np*CR	1.2068	4	0.3017	3.5342	0.00696
scv*Np*F	0.15078	8	0.018848	0.22079	0.98733
scv*CR*F	6.1651	16	0.38532	4.5137	0
sr2gr*ya_orn*Np	0.41478	2	0.20739	2.4294	0.08826
sr2gr*ya_orn*CR	9.3003	4	2.3251	27.236	0
sr2gr*ya_orn*F	0.37663	8	0.047079	0.55149	0.81806
sr2gr*Np*CR	0.24696	2	0.12348	1.4465	0.23556
sr2gr*Np*F	0.044773	4	0.011193	0.13112	0.97106
sr2gr*CR*F	0.36426	8	0.045533	0.53338	0.83215
ya_orn*Np*CR	4.546	4	1.1365	13.313	0
ya_orn*Np*F	0.053582	8	0.0066977	0.07846	0.99968
ya_orn*CR*F	34.931	16	2.1832	25.574	0
Np*CR*F	0.45142	8	0.056427	0.661	0.72633
Hata	258.83	3032	0.085367		
TOPLAM	2081.8	3239			

4.2.1. Varyans analizi sonuçları

Çok sayıdaki deney faktörlerinin istatistiksel olarak önemli olup olmadıklarını, önemli iseler hangi seviye/yöntemlerin seçilmesi gerektiğini belirlemek amacıyla çoklu anova varyans analizi yöntemi kullanılmıştır.

Tablo 4.9.'da bayg29 problemi ve Tablo 4.10.'da att48 için görülen p değerleri, faktör etkilerinin rasgele elde edilme olasılığını göstermektedir. %95 güvenilirlikle değerlendirildiğinde, p değerleri α=0.05 değerinden küçük olan, sıralama vektörüne çevirme yöntemi (scv), sıralama vektöründen gerçel vektöre çevirme yöntemi (sr2ger), yerel arama oranı (ya_orn), populasyon büyüklüğü (Np), çaprazlama olasılığı (CR) ve ölçekleme faktörü (F) etkenlerinin hepsinin birincil etkilerinin önemli oldukları görülmektedir.

Tekil etkilerde Tablo 4.9 ve Tablo 4.10'da görülen Kareler Ortalaması ya da F değerleri göz önüne alındığında, her iki problem için de, sonuç üzerinde en yüksek etkiye, en büyük değere sahip yerel arama oranı faktörünün sahip olduğu görülmektedir. İkinci sırada, faktör çaprazlama olasılığı (CR) yer alırken, bunları gerçelden sıralamaya dönüşüm yöntemi (scv) ve sıralamadan gerçele dönüşüm yöntemi (sr2gr) takip etmektedir. Sonuç üzerinde en az etkili faktörler ise populasyon büyüklüğü (Np) ve mutasyon ölçekleme (F) faktörleridir.

İkili etkileşimler göz önüne alındığında ise bayg29 için populasyon büyüklüğü ile sıralama vektörüne çevirme yöntemi, sıralamadan gerçele çevirme yöntemi ve ölçekleme faktörü etkileşimlerinin önemsiz olduğu, diğerlerinin ise önemli olduğu görülmektedir. Att48 probleminde ise bayg29'a benzer şekilde populasyon büyüklüğü ile ölçekleme faktörü etkileşimi önemsiz görünmektedir; ayrıca sıralamadan gerçele çevirme yöntemi ile ölçekleme faktörü ve çaprazlama olasılığı etkisinin de önemsiz olduğu görülmektedir. İkili etkileşimler göz önüne alındığında önemsiz olan etkileşimler bulundukları üçlü etkileşimlerde de yine önemsiz olarak değerlendirilmektedir.

Ana faktörlerin seviyelerine göre tekil etkileri Tablo 4.11'de görüldüğü gibidir. Tabloda verilen seviyeler için yapılan tüm deneylerde elde edilen çözümlerin bilinen en iyi çözümden % sapma değerlerinin ortalamaları yer almaktadır. Her iki problem için de elde edilen sonuçlar benzer yapıdadır ve faktörlerin aynı seviyelerini işaret etmektedirler. Buna göre gerçel sayı vektöründen sıralama vektörüne dönüşüm için rasgele sayı dönüşü yöntemi diğer iki yönteme göre bilinen en iyi çözüme daha fazla yaklaşmayı sağladığından daha üstündür. Sıralamadan gerçele dönüşüm yönteminde ise normalizasyon ile dönüşüm daha iyi sonuç vermektedir. Yerel arama oranı ve populasyon büyüklüğü arttıkça başarım artmaktadır. Çaprazlama oranı için ise tersine bir etki söz konusudur ve azaldıkça sonuç iyileşmektedir. Çaprazlama oranı '0' değerini aldığında denklem 3.6'da görülen kural nedeniyle en az bir parametre mutasyona uğramış vektörden alınarak çaprazlama gerçekleşmektedir. Mutasyon ölçekleme faktörü için ise en uygun seviye F = 0.5 seviyesi olarak görülmektedir.

Tablo 4.11: Faktör Seviyelerine Göre % Sapma Ortalamaları

Faktör Adı	Yöntemler / Seviyeler	Bayg29 % Sapma Ortalamaları	Att48 % Sapma Ortalamaları
Sıralama Vektörüne Dönüşüm (scv)	(1)İndeks	0,283	0,630
	(2)Alt Aralık	0,304	0,752
	(3)Rasgele Sayılı	**0,110**	**0,157**
Gerçele Dönüşüm (sr2gr)	(1)Karşılığını bularak	0,265	0,634
	(2)Normalize ederek	**0,199**	**0,392**
Yerel Arama Oranı (ya_orn)	0	0,607	1,353
	0.5	0,047	0,096
	1	**0,044**	**0,089**
Populasyon Büyüklüğü (Np)	D	0,255	0,572
	2D	0,226	0,540
	3D	**0,217**	**0,427**
Çaprazlama Olasılığı (CR)	0	**0,100**	**0,224**
	0.5	0,230	0,552
	1	0,367	0,762
Mutasyon Ölçekleme Faktörü (F)	0	0,266	0,551
	0.5	**0,199**	**0,463**
	1	0,217	0,490
	1.5	0,236	0,519
	2	0,243	0,541

Ana faktörlerin ikili etkileşimleri bayg29 problemi için Tablo 4.12'de, att48 problemi için ise Tablo 4.13'te görüldüğü gibidir. Elde edilen sonuçlar birincil etki sonuçları ile tutarlıdır.

Örneğin sıralamaya dönüşüm yöntemi (scv) satırı ile caprazlama oranı (CR) sütununun seviyelerinin kesiştiği kutuya her iki problem için de bakıldığında tekli etkilerde olduğu gibi, scv için 3. seviyeye ve CR içinde en az değere doğru gidildikçe optimalden uzaklaşma yüzdesinin azaldığı ve dolayısıyla başarımın arttığı gözlenmektedir.

Tablo 4.12: Bayg29 problemi için ikili etkileşimler

		scv			sr2gr		ya_orn			N_p			CR			F				
		1	2	3	1	2	0	0.5	1	D	2D	3D	0	0.5	1	0	0.5	1	1.5	2
scv	1	0,28			0,28	0,28	0,83	0,01	0,01	0,30	0,28	0,27	0,20	0,28	0,36	0,30	0,27	0,27	0,28	0,28
	2		0,30		0,39	0,22	0,67	0,12	0,12	0,33	0,30	0,29	0,09	0,33	0,49	0,30	0,27	0,30	0,32	0,33
	3			0,11	0,12	0,10	0,31	0,01	0,01	0,13	0,10	0,09	0,00	0,08	0,25	0,20	0,06	0,08	0,10	0,11
sr2gr	1				0,27		0,63	0,09	0,08	0,29	0,26	0,25	0,12	0,27	0,41	0,30	0,23	0,24	0,27	0,29
	2					0,20	0,58	0,01	0,01	0,22	0,19	0,18	0,08	0,20	0,32	0,23	0,17	0,19	0,20	0,20
ya_om	0						0,61			0,66	0,59	0,57	0,27	0,59	0,96	0,70	0,52	0,56	0,62	0,63
	0.5							0,05		0,05	0,04	0,04	0,02	0,05	0,07	0,05	0,04	0,04	0,05	0,05
	1								0,04	0,05	0,04	0,04	0,02	0,05	0,07	0,05	0,03	0,04	0,05	0,05
N_p	D									0,25			0,11	0,25	0,40	0,29	0,22	0,24	0,26	0,26
	2D										0,23		0,10	0,22	0,36	0,26	0,19	0,21	0,23	0,24
	3D											0,22	0,09	0,22	0,34	0,25	0,19	0,20	0,22	0,23
CR	0												0,10			0,10	0,09	0,10	0,11	0,11
	0.5													0,23		0,14	0,21	0,25	0,27	0,28
	1														0,37	0,56	0,30	0,31	0,33	0,34

Tablo 4.13: Att48 problemi için ikili etkileşimler

		scv			sr2gr		ya_orn			N_p			CR			F				
		1	2	3	1	2	0	0.5	1	D	2D	3D	0	0.5	1	0	0.5	1	1.5	2
scv	1	0,63			0,63	0,63	1,87	0,01	0,01	0,65	0,62	0,62	0,42	0,69	0,79	0,68	0,61	0,59	0,62	0,65
	2		0,75		1,00	0,50	1,58	0,35	0,33	0,78	0,75	0,72	0,29	0,86	1,11	0,68	0,72	0,75	0,80	0,81
	3			0,16	0,27	0,05	0,62	0,00	0,00	0,28	0,25	0,00	0,00	0,11	0,39	0,30	0,05	0,13	0,14	0,17
sr2gr	1				0,63		1,44	0,24	0,23	0,66	0,64	0,60	0,32	0,67	0,91	0,66	0,58	0,60	0,65	0,68
	2					0,39	1,27	0,00	0,00	0,48	0,44	0,25	0,13	0,43	0,61	0,44	0,35	0,38	0,39	0,40
ya_om	0						1,35			1,46	1,38	1,22	0,60	1,44	2,02	1,47	1,23	1,28	1,37	1,42
	0.5							0,10		0,13	0,13	0,03	0,04	0,12	0,13	0,09	0,08	0,10	0,10	0,11
	1								0,09	0,13	0,12	0,03	0,03	0,11	0,13	0,09	0,08	0,09	0,09	0,10
N_p	D									0,57			0,26	0,61	0,85	0,61	0,52	0,55	0,58	0,60
	2D										0,54		0,24	0,59	0,79	0,59	0,48	0,51	0,54	0,57
	3D											0,43	0,17	0,46	0,65	0,45	0,38	0,41	0,44	0,45
CR	0												0,22			0,21	0,19	0,22	0,24	0,26
	0.5													0,55		0,39	0,53	0,60	0,61	0,63
	1														0,76	1,06	0,67	0,65	0,70	0,73

Yukarıda anlatılan varyans analizi çalışmalarında mutasyon denklemi olarak klasik DGA mutasyon denklemi olan (4.4) denklemi kullanılmıştır. Ancak 6. bölümde de bahsedildiği Nearchou ve Omirou çizelgeleme problemleri için DGA kullandıkları çalışmalarında, diferansiyel gelişim mutasyonu için klasik mutasyon denklemi (4.4) yerine farklı bir denklemi (4.5) kullanmaktadırlar. Farklı olan bu denklemin başarım üzerinde nasıl bir etkisi olduğunun ve faktörler arası etkileşimlerin mutasyon denklemine göre değişiminin gözlemlenebilmesi amacıyla bayg29 literatür problemi için gerçekleştirilen denemeler (4.5) denklemi için tekrarlanmıştır. Tekli etkiler Tablo 4.14'de görüldüğü gibidir. Yeni denklemde, faktörler için belirlenen seviyeler aynı seçilmekle birlikte, klasik denkleme göre daha kötü sonuçlar ortaya çıkmaktadır. Bu nedenle literatür denemelerinde klasik denklem kullanılacaktır.

$$v_{iG+1} = x_{r1,G} + F.(x_{r2,G} - x_{r3,G}), \tag{4.4}$$

$$v_{iG+1} = x_{ibest} + F.(x_{aG} + x_{bG} - x_{cG} - x_{dG}), \tag{4.5}$$

Tablo 4.14: Mutasyon Denklemleri için Faktör Seviyelerine Göre % Sapmalar

Faktör Adı	Yöntemler / Seviyeler	Bayg29 % Sapma Ortalamaları (Denklem 4)	Bayg29 % Sapma Ortalamaları (Denklem 5)
Sıralama Vektörüne Dönüşüm (scv)	(1)İndeks	0,283	0,290
	(2)Alt Aralık	0,304	0,346
	(3)Rasgele Sayılı	**0,110**	**0,138**
Gerçele Dönüşüm (sr2gr)	(1)Karşılığını bularak	0,265	0,302
	(2)Normalize ederek	**0,199**	**0,214**
Yerel Arama Oranı (ya_orn)	0	0,607	0,666
	0.5	0,047	0,057
	1	**0,044**	**0,052**
Populasyon Büyüklüğü (Np)	D	0,255	0,272
	2D	0,226	0,250
	3D	**0,217**	**0,240**
Çaprazlama Olasılığı (CR)	0	**0,100**	**0,142**
	0.5	0,230	0,281
	1	0,367	0,352
Mutasyon Ölçekleme Faktörü (F)	0	0,266	0,370
	0.5	**0,199**	**0,200**
	1	0,217	0,229
	1.5	0,236	0,244
	2	0,243	0,248

4.3. Literatür Problemleri Denemeleri

Oluşturulan yeni algoritmanın başarımının literatür problemleri üzerinde denenmiştir. Bu amaçla GSP için dünya çapında kullanılan en geniş kaynak olan TSPLIB [128] problemlerinin bir kısmı seçilerek kullanılmıştır. 4.2 alt bölümünde elde edilen en iyi parametre/yöntem kombinasyonu kullanılan her problem için 3 tekrarlı olmak üzere çözüm yapılmıştır. Denemelerde durdurma kriterinin sonuç üzerinde etkisini ortadan kaldırmak amacıyla, bir durdurma kriteri belirlenmemiş her 300*D iterasyon da sonuçlar kontrol edilerek tamam/devam kararı verilmiştir. Elde edilen sonuçların bilinen en iyi çözümden uzaklıklarının 3 tekrar için ortalamasının %1 değerinin altına düşmesi durumunda durdurma kararı verilmiş, aksi takdirde deneylere devam edilmiştir. Elde edilen sonuçların, küresel en iyi değerden sapmalarının en küçük ortalama ve en büyük değerleri Tablo 4.15'te görüldüğü gibidir.

Tablo 4.15: Referans Problem Sonuçlarının Küresel En İyi Değerden Yüzde Sapmaları

	% SAPMA			Ort.
Problem Adı	Min.	Ort.	Maks.	Sure (sn)
burma14	0,00	0,00	0,00	<1
ulysses16	0,00	0,00	0,00	<1
gr17	0,00	0,00	0,00	<1
gr21	0,00	0,00	0,00	<1
ulysses22	0,00	0,00	0,00	<1
fri26	0,00	0,00	0,00	1
bays29	0,00	0,00	0,00	1
bayg29	0,00	0,00	0,00	1
gr24	0,00	0,00	0,00	<1
dantzig42	0,00	0,00	0,00	5
swisss42	0,00	0,00	0,00	5
gr48	0,00	0,00	0,00	11
att48	0,28	0,53	0,74	11
eil51	0,23	0,31	0,47	15
berlin52	0,00	0,00	0,00	16
brasil58	0,00	0,00	0,00	21
st70	0,00	0,20	0,30	51

Tablo 4.15(devam): Referans Problem Sonuçlarının Küresel En İyi Değerden Yüzde Sapmaları

	% SAPMA			Ort.
Problem Adı	Min.	Ort.	Maks.	Sure (sn)
eil76	0,19	0,93	1,67	82
pr76	0,10	0,23	0,41	73
gr96	0,54	0,77	1,01	186
rat99	0,00	0,88	1,49	254
kroA100	0,00	0,04	0,11	243
kroB100	0,52	0,90	1,30	266
kroC100	0,10	0,51	0,79	259
kroD100	0,32	0,86	1,44	251
rd100	0,57	0,85	1,00	250
eil101	0,64	1,32	1,91	282
lin105	0,16	0,32	0,48	263
pr107	0,20	0,54	0,81	261
gr120	1,08	1,66	2,00	627
pr124	0,08	0,18	0,36	508
bier127	1,08	1,24	1,37	636
ch130	0,61	1,01	1,59	781
pr136	0,52	1,10	1,58	978
gr137	0,62	0,90	1,27	1591
pr144	0,09	0,19	0,39	881
ch150	1,76	1,83	1,98	1421
kroA150	1,08	1,31	1,48	1949
kroB150	0,73	1,01	1,40	1390
pr152	0,01	0,24	0,43	1285
u159	0,73	0,92	1,01	1622
si175	0,13	0,25	0,37	2504
brg180	1,03	2,56	3,59	4251
rat195	3,27	3,39	3,57	4887
d198	0,72	0,99	1,14	4310
kroA200	1,80	1,93	2,00	4966
kroB200	1,84	2,14	2,59	3319
gr202	2,54	2,58	2,67	2985
tsp225	2,45	2,74	2,89	5220
pr226	0,34	0,56	0,81	4125
gr229	1,53	1,63	1,74	5020
pr264	1,73	3,40	4,36	9990
a280	2,13	2,57	2,83	12734
pr299	1,89	2,50	3,03	18639
lin318	3,21	3,60	3,91	23704

4.3.1. Komşuluk arama

Elde edilen çözümün, fazla zaman almayacak şekilde, küçük çaplı bazı değişim hareketleri ile bütün komşuluklarının aranması ve mevcut en iyi çözümün bir miktar geliştirilmesi mümkündür. GDGA'da bulunan çözüm populasyonu bireyleri birçok durumda, diğer birçok metasezgiselden farklı olarak tek bir çözüme yakınsamamaktadır. Bu durum, populasyon bireylerinin çözüm uzayındaki birbirinden farklı birçok iyi çözüme ulaşmış olması ve en iyi birey dışındaki diğer bireylerin komşuluk arama sonucunda daha iyi çözümler sunmaları olasılığını arttırmaktadır.

Bu çalışmada elde edilen populasyona ait en iyi 10 birey sırasıyla 2-opt , 3-opt ve 4-opt komşuluk aramalarına tabi tutulmuştur. Arama sırasında, tepe tırmanma (hill climbing) yöntemi kullanılarak, mevcut arama tipi için olası komşu çözümlere götüren tüm hareketler içinde öncelikli olarak en fazla gelişmeyi sağlayabilecek olanı gerçekleştirilmiştir. Komşuluk arama hareketlerinde bir gelişme olması durumunda diğer hareketler için arama tekrarlanmış, üç harekette de gelişme olmaması durumunda aramaya son verilmiştir. Komşuluk arama sonucunda elde edilen en iyi çözümlerin optimumdan sapma değerleri ve bu çözümlerin komşuluk arama öncesi populasyondaki en iyi çözümler sıralamasındaki yerleri Tablo 4.16'da görülmektedir. GDGA sonucunda, küresel en iyiye ulaşılan problemler komşuluk aramaya tabi tutulmadıklarından, tabloda yer almamaktadırlar.

Tablo 4.16: Komşuluk Arama Sonucunda Yüzde Sapmalar ve Populasyondaki Yerleri

	% SAPMA						Komşuluk Aramada Bulunan En İyi Bireyin Pop. Yeri		
	Koms. Ara.Önce			Koms. Ara.Sonra					
Problem Adı	Min.	Ort.	Maks.	Min.	Ort.	Maks.			
att48	0,28	0,53	0,74	0,00	0,08	0,24	3	7	8
eil51	0,23	0,31	0,47	0,23	0,23	0,23	1	1	1
st70	0,00	0,20	0,30	0,00	0,10	0,30	1	1	9
eil76	0,19	0,93	1,67	0,00	0,12	0,37	1	4	9
pr76	0,10	0,23	0,41	0,00	0,00	0,00	1	1	1
gr96	0,54	0,77	1,01	0,41	0,41	0,42	1	4	3
rat99	0,00	0,88	1,49	0,00	0,03	0,08	9	1	1

Tablo 4.16 (devam): Komşuluk Arama Sonucunda Yüzde Sapmalar ve Populasyondaki Yerleri

	% SAPMA						Komşuluk Aramada Bulunan En İyi Bireyin Pop. Yeri		
	Koms. Ara.Önce			Koms. Ara.Sonra					
Problem Adı	Min.	Ort.	Maks.	Min.	Ort.	Maks.			
kroA100	0,00	0,04	0,11	0,00	0,04	0,11	1	1	1
kroB100	0,52	0,90	1,30	0,00	0,00	0,00	3	2	4
kroC100	0,10	0,51	0,79	0,00	0,00	0,00	3	1	2
kroD100	0,32	0,86	1,44	0,00	0,26	0,45	9	1	1
rd100	0,57	0,85	1,00	0,00	0,05	0,08	9	6	1
eil101	0,64	1,32	1,91	0,16	0,26	0,32	2	8	6
lin105	0,16	0,32	0,48	0,00	0,00	0,00	8	1	2
pr107	0,20	0,54	0,81	0,10	0,13	0,20	1	10	3
gr120	1,08	1,66	2,00	0,35	0,55	0,71	4	4	3
pr124	0,08	0,18	0,36	0,00	0,00	0,00	4	2	1
bier127	1,08	1,24	1,37	0,46	0,52	0,64	1	1	6
ch130	0,61	1,01	1,59	0,00	0,26	0,49	6	10	1
pr136	0,52	1,10	1,58	0,37	0,63	1,08	6	5	1
gr137	0,62	0,90	1,27	0,10	0,20	0,27	2	6	10
pr144	0,09	0,19	0,39	0,00	0,00	0,00	6	1	1
ch150	1,76	1,83	1,98	0,32	0,51	0,89	4	2	7
kroA150	1,08	1,31	1,48	0,53	0,58	0,67	2	2	3
kroB150	0,73	1,01	1,40	0,04	0,15	0,22	8	2	10
pr152	0,01	0,24	0,43	0,00	0,00	0,00	2	1	3
u159	0,73	0,92	1,01	0,00	0,00	0,00	1	1	1
si175	0,13	0,25	0,37	0,06	0,07	0,09	7	1	10
brg180	1,03	2,56	3,59	0,51	0,68	1,03	1	3	1
rat195	3,27	3,39	3,57	1,21	1,38	1,64	4	1	7
d198	0,72	0,99	1,14	0,25	0,37	0,48	9	1	6
kroA200	1,80	1,93	2,00	0,34	0,53	0,81	5	7	4
kroB200	1,84	2,14	2,59	0,55	0,58	0,60	1	8	6
gr202	2,54	2,58	2,67	0,50	0,53	0,56	2	9	5
tsp225	2,45	2,74	2,89	0,77	1,15	1,48	3	4	6
pr226	0,34	0,56	0,81	0,00	0,14	0,22	3	4	2
gr229	1,53	1,63	1,74	0,62	0,62	0,62	4	4	4
pr264	1,73	3,40	4,36	0,00	0,22	0,61	1	1	5
a280	2,13	2,57	2,83	0,43	0,76	1,28	4	2	1
pr299	1,89	2,50	3,03	0,60	0,82	1,23	7	2	6
lin318	3,21	3,60	3,91	0,79	1,12	1,35	3	8	1

Tablo 4.16'da görüldüğü gibi komşuluk arama sonucunda önceki aşamada küresel en iyiye ulaşılan 14 probleme ek olarak, çeşitli büyüklüklerde 8 problemde daha 3 ayrı deneme için de küresel en iyiye ulaşılmıştır. Diğer problemlerde ise en fazla rat195 % 1,64 sapma elde edilmiştir. Ayrıca 9 farklı problemde ise 3 tekrar için elde edilemese de küresel eniyi çözümlere ulaşılmıştır.

Tablo 4.16'nın sağ tarafındaki üç kolon incelendiğinde komşuluk arama sonucunda bulunan en iyi çözümlerin, GDGA sonucunda elde edilen en iyi 10 çözüm içerisinde değişik noktalara dağıldığı görülmektedir. Bu durum, DGA'nın çözüm uzayına dağılmasının sonucu iyi şekilde etkileyebildiğini göstermektedir.

4.4. Durdurma Kriteri

Meta-sezgisel yöntemlerin kesin çözüm yöntemlerine göre tercih edilmelerinin önemli bir nedeni süre performanslarıdır. Kesin yöntemlerin küresel en iyiyi garanti eden çözüm süreleri belli problem büyüklüklerinin üzerinde, problemin doğasının gerektirdiği çözüm süresinin çok üzerine çıkarken meta-sezgiseller, küresel en iyiden bir miktar sapma karşılığında istenilen çok daha kısa sürelerde çözüm sunmaktadırlar. Meta-sezgisel sürecin kesilmesi durdurma kriterleri yardımıyla gerçekleştirilmektedir. Meta-sezgisel yöntemlerde, bütün eniyileme problemleri için geçerli tek bir durdurma kriterinin bulunması pek mümkün görünmemektedir. İterasyon sayısı belirli bir değere ulaştığında, en iyi çözüm değerinde belirli bir süre gelişme olmadığında, populasyon yakınsadığında, belirli bir hedefe ulaşıldığında ya da verilen süre kısıtına ulaşıldığında işlem durdurulabilir. Sayılan yöntemler ve akla gelebilecek birçok alternatif tek başlarına kullanılabilecekleri gibi, aynı anda birçoğunun aktif olarak kullanılması da mümkündür. Bu çalışmada durdurma kriterinin sonuç üzerindeki etkisinin en küçük seviyeye indirgenebilmesi amacıyla oldukça geniş durdurma kriterleri seçilmiştir. Böylece elde edilen sonuçlar göz önüne alınarak en uygun durdurma kriterinin daha doğru bir şekilde belirlenebilmesi mümkün olacaktır.

4.5. Referans Karşılaştırması

Elde edilen GDGA sonuçları, populasyon tabanlı meta sezgisel yöntemler Genetik Algoritma (GA), Karınca Kolonileri (ACS) ve Parçacık Sürü Optimizasyonu (PSO) yöntemleri ile GSP problemlerinin çözümü için yapılan çalışmalarda elde edilen sonuçlar ile karşılaştırıldığında, önerilen yöntemin dikkate değer olduğu görülmektedir. Tablo 4.17'de elde edilen çözümlerin en iyi değerleri ve ortalama değerlerinin küresel en iyiden yüzde sapmaları, karşılaştırmalı olarak görülmektedir.

Referans problemlerin bazıları incelenen çalışmalarda yer almadığından tablolarda sunulmamışlardır.

Tablo 4.17: Karşılaştırmalı Küresel En İyiden Sapma Değerleri

	EN KÜÇÜK				ORTALAMA			
Problem	GDGA	GA	ACS	PSO	GDGA	GA	ACS	PSO
burma14	0,00	-	-	0,00 [105 - 109]	0,00	-	-	0,00[108] 0,03[109]
att48	0,00	1,02[69] 0,00[67]	-	-	0,08	2,09[69]	-	-
eil51	0,23	1,17[69] 0,00[67]	0,00 [79,80,81] 0,95[87] 0,00[82]	2,54[105] 2,53[108] 0,23[109] 0,70[110]	0,23	2,58[69]	0,48[80] 0,16[82]	3,47[108] 1,79[109] 0,23[109] 2,58[110]
berlin52	0,00	0,00[59] 0,00[57] 0,00[67]	-	2,12[105] 0,03[106] 0,00[109] 0,00[110]	0,00	0,00[57]	-	0,75[109] 2,59[110]
st70	0,00	1,48[69]	-	-	0,10	2,07[69]	-	-
eil76	0,00	1,86[69] 0,00[67]	0,00[79,81] 2,96[87]	4,75[105] 0,37[109] 2,42[110]	0,12	2,42[69]	-	2,55[109] 4,66[110]
pr76	0,00	1,45[69]	-	-	0,00	1,84[69]	-	-
rat99	0,00	2,15[69] 0,00[67]	-	-	0,03	4,71[69]	-	-
kroA100	0,00	1,22[69] 0,00[67] 0,00[57]	0,00 [79,80,81] 0,71[87]	0,07[109]	0,04	2,28[69] 0,08[57]	0,65[80] 0,00[82]	1,91[109]
kroB100	0,00	2,93[69] 0,00[67]	-	-	0,00	4,1[69]	-	-
kroC100	0,00	0,68[69] 0,00[67]	-	-	0,00	3,45[69]	-	-
kroD100	0,00	1,06[69] 0,07[67]	-	-	0,26	4,58[69]	-	-
rd100	0,00	1,30[69]	-	-	0,05	2,6[69]	-	-
eil101	0,16	3,82[69] 0,00[67]	-	-	0,26	5,09[69]	-	-
lin105	0,00	0,87[69]	-	-	0,00	1,68[69]	-	-
pr107	0,10	0,50[69]	-	-	0,13	2,18[69]	-	-
pr124	0,00	0,08[69]	-	-	0,00	1,06[69]	-	-
bier127	0,46	1,89[69] 0,00[67]	-	-	0,52	2,83[69]	-	-
pr136	0,37	4,13[69]	-	-	0,63	5,8[69]	-	-
pr144	0,00	0,03[69]	-	-	0,00	0,74[69]	-	-
kroA150	0,53	1,66[69]	-	-	0,58	5,06[69]	-	-
kroB150	0,04	2,97[69]	-	-	0,15	4,7[69]	-	-
pr152	0,00	1,32[69]	-	-	0,00	2,51[69]	-	-
u159	0,00	5,24[69]	-	-	0,00	7,02[69]	-	-
rat195	1,21	6,54[69]	-	5,81[110]	1,38	7,4[69]	-	8,76[110]

Tablo 4.17 (devam): Karşılaştırmalı Küresel En İyiden Sapma Değerleri

	EN KÜÇÜK				ORTALAMA			
Problem	GDGA	GA	ACS	PSO	GDGA	GA	ACS	PSO
d198	0,25	1,81[69] 0,05[67]	0,00[77,84] 0,68 [79,80,81] 1,61[87]	-	0,37	3,26[69]	0,00[77,84] 1,74 [79,80,81] 0,04[82]	-
kroA200	0,34	2,80[69] 0,00[67]	-	0,66[109]	0,53	4,56[69]	-	3,42 [109]
kroB200	0,55	5,27[69]	-	-	0,58	6,77[69]	-	-
tsp225	0,77	1,43[69] 0,00[57]	-	-	1,15	3,86[69] 1,60[57]	-	-
pr226	0,00	0,92[69]	-	-	0,14	2,03[69]	-	-
pr264	0,00	5,00[69]	-	-	0,22	6,24[69]	-	-
a280	0,43	0,23[59]	-	-	0,76	-	-	-
pr299	0,60	4,94[69]		5,85[110]	0,82	7,6[69]	-	7,99[110]
lin318	0,79	5,63[69]	0,00[77,84]	-	1,12	8,34[69]	0,00[77,84] 0,09[82]	-

Geliştirilmiş Diferansiyel Gelişim Algoritması (GDGA) Tablo 4.17'de görüldüğü gibi literatürde Gezgin Satıcı Problemlerinin çözümünün gerçekleştirildiği ve bu çalışmanın 2. bölümünde özetlenen populasyon tabanlı algoritmalarla yakın seviyede sonuçlar ortaya koymaktadır. Ayrıntılı olarak bakıldığında GDGA'nın PSO'ya göre daha iyi sonuçlar verdiği, GA'dan bazı durumlarda daha iyi bazı durumlarda daha kötü sonuçlar vererek başa baş bir performans sergilediği görülmektedir. Karınca Kolonileri sonuçlarının ise genel olarak GDGA sonuçlarına baskın olduğu gözlemlenmektedir.

Karınca Kolonileri Algoritmaları'nda çözüm oluştururken şehirler arası uzaklığın sezgisel bilgi olarak sürecin içine dahil etmesi ve aday şehir seçiminde sadece mevcut şehre en yakın belirli sayıdaki şehrin aday listesine eklenmesi gibi GSP'ye özel bilgi kullanımı, bu problemin çözümü için büyük avantaj sağlamaktadır. Benzer şekilde Genetik Algoritma'da GSP'ye özel ER (Edge Recombination) gibi çaprazlama yöntemleri uygulanarak daha iyi sonuçlar elde edilebilmektedir. GDGA yapısında, geliştirilen algoritmanın genel amaçlı yapısının korunması amacıyla, GSP'ye özel yapılar kullanılmamıştır. Bu tür yapıların kullanımının algoritma performansını arttırma olasılığı oldukça yüksektir.

5. SONUÇ VE ÖNERİLER

Kombinasyonel eniyileme alanında halen katedilmesi gereken uzun bir yol bulunmaktadır. Rekabetin artışı ile birlikte gerçek yaşamda bir çok alanda karşılaşılan bu tür problemlerin daha kısa sürede ve etkin bir şekilde çözülmeleri gerekliği artmaktadır. Bu alandaki kesin çözüm yöntemleri GSP gibi bazı problemlere özel olarak geliştirilmiş bazı yaklaşımlar yardımıyla ilerleme kaydetmiş olmasına karşın, bu ilerleme tüm alana henüz yayılamamıştır. Bunun temel nedeni akademik ortamda tanımlanmış problemlerin bile birbirinden oldukça farklı yapıda ve sayıda olmalarının yanında, geçek yaşamda bu problemlerin bir veya birkaçının farklı kısıtlar altında bir araya gelmeleri durumunda geliştirilen özel çözümlerin bu alanlarda uygulanmasının mümkün olmamasıdır. Bu noktada sanayinin ihtiyacı, küresel en iyi çözümü garanti edemese de ona çok yakın etkin çözümleri kısa sürede üretebilen, problem yapısına rahatlıkla uyumlulaştırılabilen meta sezgiseller yöntemler yardımıyla karşılanabilmektedir.

Diferansiyel Gelişim Algoritması kombinasyonel alanda henüz etkin kullanımı olmasa da sürekli çözüm uzayında ve bazı kesikli problemlerde (0-1) oldukça başarılı çözümler elde edilmesine imkan tanımaktadır. Bu çalışmada yöntemin başarılı yapısının kombinasyonel en iyileme alanında kullanılabilmesi amacıyla, yeni bir yaklaşım geliştirilmiştir. Geliştirme ve başarımının denenmesi aşamalarında en zor problemeler grubunda (NP-hard) yer alan ve üzerinde en çok çalışılmış kombinasyonel eniyileme problemi olan Gezgin Satıcı Problemi (GSP) kullanılmıştır. Böylece GSP üzerinde yapılan çalışmaların, bu çalışmaya ışık tutması hedeflenmiştir. Bu amaçla, GSP'nin çözümü üzerinde yapılan kesin, sezgisel ve meta sezgisel uygulamalar özetlenerek, bu alanda karşılaşılan problemler ve geliştirilen çözüm yaklaşımları anlatılmıştır.

4. bölümde ayrıntılarıyla adımları, geliştirilme süreci ve karşılaştırma problemleri üzerindeki başarımı anlatılan Geliştirilmiş Diferansiyel Gelişim Algoritmasının, bu

alanda en çok kullanılan ve Diferansiyel Gelişim Algoritması gibi populasyon tabanlı olan Karınca Kolonileri, Genetik Algoritmalar gibi algoritmalar ile eşdeğer başarım seviyesine taşınmış olduğu gösterilmiştir. Ayrıca yine populasyon tabanlı olan, gösterim ve algoritma işleyişi bakımından Diferansiyel Gelişim Algoritmasına oldukça benzeyen Parçacık Sürü Optimizasyonu Algoritması'na göre daha iyi sonuçlar elde edilebildiği görülmektedir.

Diferansiyel Gelişim Algortimasının başarımının yüksekliği seçilim sürecinin, paralel meta sezgisellere benzer şekilde populasyon bireylerin çözüm uzayında ayrık hareket edebilmelerine ve böylece yerel eniyi çözümlere sıkışmasının önlenebilmesinin yanında, kullandığı mutasyon işlemine dayanmaktadır. Mutasyon işlemi, bir bireyi daha iyi yapan çözüm parçasının başka bir bireye eklenmesi ya da başarısız yapan kısmının diğer bir bireyden çıkarılmasına imkan veren, bireyler arası farklılığın kullanıldığı diferansiyel süreç yardımıyla başarılı sonuçlar elde edilmesine yol açmaktadır.

Algoritmanın yakınsama hızının arttırılması, ancak bunu yaparken diferansiyel sürecin etkinliğinin gölgelenmemesi amacıyla basit ters çevirme (2-opt) operatörünün kullanıldığı yerel arama kullanılmıştır. Bu operatör dışında diğer metasezgisellerde sıklıkla kullanıldığı görülen 3-opt, 4-opt, 5-opt, Lin-Kerninghan yöntemlerinin ya da Tabu Arama, Tavlama Benzetimi gibi algortimaların kullanımının başarımı arttırması mümkün görünmektedir.

Geliştirilmiş Diferansiyel Gelişim Algoritması sürekli çözüm vektörüyle ve bu tür vektörlere uygulanan operatörler yardımıyla çalışması nedeniyle, aynı vektörün bir kısmı GSP gibi permütasyonel problemler için, bir kısmı sürekli değişkenler için ve başka bir kısmı ikili (0-1) değişkenler için kullanılmak üzere ayrılabilir. Böylece yapısında pek çok farklı tipte değişkenin bulunabildiği gerçek yaşam problemlerinin aynı dizi ve algoritma yapısıyla çözülmesi mümkün olabilir. Algortimanın bu tür problemler üzerinde ve ayrıca diğer kombinasyonel eniyileme problemleri üzerinde denenmesi ve başarımının değerlendirilmesi gelecekteki çalışmalara bırakılmıştır.

6. KAYNAKLAR

[1] Applegate D., Bixby R., Chvatal V., Cook W., *Applications of TSP*, [Online], http://www.tsp.gatech.edu/apps/index.html **(Ziyaret Tarihi: 3 Ekim 2008)**.

[2] Hoffman K. Padberg M., *Traveling Salesman Problem*, [Online], http://iris.gmu.edu/ ~khoffman/papers/trav_salesman.html **(Ziyaret Tarihi: 14 Ekim 2008)**.

[3] Blaser M., Manthey B., Sgall J., "An improved approximation algorithm for the asymmetric TSP with strengthened triangle inequality", ***Journal of Discrete Algorithms***, 4, 623-632, (2006).

[4] Kwon S.H., Kim H.T., Kang M.K., "Determination of the candidate arc set for the asymmetric traveling salesman problem", ***Computers & Operations Research***, 32-5, 1045-1057, (2005).

[5] Pearn W.L., Chien R.C., "Improved solutions for the traveling purchaser problem", ***Computers & Operations Research***, 25-11, 879-885, (1998).

[6] Bektaş T., "The multiple traveling salesman problem: an overview of formulations and solution procedures" ***The International Journal of Management Science***,34, 209-219,(2006)

[7] "Chinese postman problem",Algorithms and Theory of Computation Handbook, ***CRC Press LLC***, (1999).

[8]http://www.densis.fee.unicamp.br/~moscato/TSPBIB_home.html**(Ziyaret Tarihi: 1 Ekim 2008)**

[9] Gutin G., Punnen A.P," The Traveling Salesman Problem and Its Variations", ***Kluwer Academic Publishers***, 29-114, (2002).

[10] Korte B., Vygen J.,"Combinatorial Optimization Theory and Algorithms",Third Edition, ***Springer***, 6, (2006).

[11] http://tr.wikipedia.org/wiki/P_ile_NP_arasındaki_ilişki **(Ziyaret Tarihi : 1 Ekim 2008)**

[12] http://en.wikipedia.org:80/wiki/NP-hard **(Ziyaret Tarihi: 1 Ekim 2008)**

[13] http://en.wikipedia.org/wiki/Traveling_salesman_problem **(Ziyaret Tarihi :1 Ekim 2008)**

[14] Held M., Karp R.M., "A dynamic programming approach to sequencing problems", ***Journal of SIAM***, 10, 196–210, (1962).

[15] Hoffman, K., *Traveling Salesman Problem* [Online], http://iris.gms.edu/khofkmann/papers/trav_salesman.html. (2006) **(Ziyaret Tarihi :1 Ekim 2008)**

[16] Eastman W.L., "Linear Programming with Pattern Constraints", Doktora Tezi, ***Harvard Üniversitesi***, Cambridge, (1958).

[17] Little J.D.C., Murty K.G., Sweeney D.W., Karel C., "An Algorithm for the traveling salesman problem", ***Operations Research***, 11, 972-989, (1963).

[18] Shapiro D.M., "Algorithms for the Solution of the Optimal Cost and Bottleneck Traveling Salesman Problem", Yüksek Lisans Tezi, ***Washington Üniversitesi***, St. Lois, (1966).

[19] Bellmore M., Malone J.C. "Pathology of traveling-salesman subtour elimination algorithms ", ***Operations Research***, 19, 278-307, (1971).

[20] Smith T.H.C. , Srinivasan V., Thompson G.L. "Computational performance of three subtour elimination algorithms for solving asymmetric traveling salesman problems ", ***Ann. Dicrete Mathematics***, 1, 495-506, (1977).

[21] Carpaneto G. ve Toth P. "Some new branching and bounding criteria for the asymmetric traveling salesman problem", ***Management Science***, 26,736-743,(1980).

[22] Held M., Karp R.M., "Traveling-salesman problem and minimum spanning trees", ***Operations Research***, 18, 1138-1162, (1970).

[23] Held M., Karp R.M., "Traveling-salesman problem and minimum spanning trees", ***Math. Programming***, 1, 6-25, (1971).

[24] Christofides N., "The shortest Hamiltonian chain of a graph", ***SIAM Journal of Applied Math.***, 19, 689-696, (1970).

[25] Helbig K., Hansen, Krarup J. "Improvements of Held-Karp algorithm for the symmetric traveling-salesman problem", ***Math. Programming***, 7, 87-96, (1974).

[26] Smith T.H.C. , Thompson G.L. "A LIFO implicit enumaration search algorithm for the symmetric traveling salesman problem using Held and Karp's 1-tree relaxation", ***Ann. Dicrete Mathematics***, 1, 479-493, (1977).

[27] Volgenant T., Jonker R. "A branch and bound algorithm for the symmetric traveling salesman problem based on the 1-tree relaxation", ***European J. Of Operations Research***, 9, 83-89, (1982).

[28] Gavish B., Srikant. K. N., "Efficient Branch and Bound Code for Solving Large Scale Traveling Salesman Problemsto Optimality", Çalışma yayını QM8329, ***Rochester Üniversitesi Graduate School of Management*** , (1983).

[29] Foulds L.R., "Combinatorial Optimization for Undergraduates", ***Springer Verlag***, 174-177, (1984).

[30] Lawler E.L., Lenstra J.K., Rinnoy Kan A.H.G., Shmoys D.B. , "The Traveling Salesman Problem", ***John Wiley & Sons***, 361-377,(1986)

[31] Dantzig G.B., Fulkerson D.R., Johnson S.M., "Solution of a large-scale traveling salesman problem" ***Operations Research*** , 2, 393-410, (1954).

[32] Miller, C. Tucker A., Zemlin R.,. "Integer Programming formulations and travelling salesman problems", ***Journal of ACM***, 7, 326-329, (1960).

[33] Gomory, R.E., "An algorithm for integer solutions to linear programs. In: Recent Advances in Mathematical Programming" (R. L. Graves and P. Wolfe, eds.), ***McGraw-Hill***, New York, 269-302, (1963).

[34] Padberg M., Rinaldi G., "A branch and cut algorithm for the resolution of large scale travelling salesman problems", ***SIAM Review***,33 (l), 60-100 (1991).

[35] Padberg M., Rinaldi G., "An efficient algorithm for the minimum capacity cut problem", ***Mathematical Programming***, 47, 19-36, (1990)

[36] Padberg M., Rinaldi G., "Facet identification for the symmetric travelling salesman polytope", ***Mathematical Programming***, 47, 219-257, (1990).

[37] Grotschel M., "On the symmetric travelling salesman problem: Solution of a 120 city problem", ***Mathematical Programming Studies***, 12, 61-77 (1980).

[38] Crowder H. and Padberg M.W., "Solving large-scale symmetric travelling salesman problems to optimality", ***Management Science***, 26, 495-509 (1980).

[39] Grötschel M. and Holland O., "Solution of large-scale symmetric travelling salesman problems", ***Mathematical Programming***, 61, 141-202 (1991).

[40]Chvatal V., "Edmonds polytopes and weakly Hamiltonian graphs", ***Mathematical Programming***, 5, 29-40 (1973).

[41] Grötschel M. and Pulleybank W.R., "Clique tree inequalities and the symmetric travelling salesman problem", ***Mathematics of Operations Research***, 11, 537-569, (1986).

[42] Applegate D., Bixby R., Chvatal V., Cook W., "Finding cuts in the TSP (a preliminary report)", ***DIMACS Technical Report*** , (1995).

[43] Applegate D., Bixby R., Chvatal V., Cook W., Espinoza D.G., Goycoolea M, Helsgaun K.," Certification of an optimal TSP tour through 85,900 cities", doi:10.1016/j.orl.2008.09.006, ***Operations Research Letters***, (2008).

[44] Applegate D., Bixby R., Chavatal V., Cook W., "On the solution of traveling salesman problems", ***Documenta Mathematica***, Extra Volume ICM 1998, 645-656, (1998)

[45] Applegate D., Bixby R., Chavatal V., Cook W., "TSP cuts which do not conform to the template paradigm. Computational Combinatorial Optimization", (M. Junger and D. Naddef, eds.) ***Springer***, (2001).

[46]Caccetta L., Hill S. P. "Branch and Cut Methods for Network Optimization", ***Mathematical and Computer Modelling***, 33, 517-532, (2001).

[47]Desrochers M., Laporte G. "Improvements and extensions to the Miller-Tucker-Zemlin Subtour Elimination Constraints", ***Operational Research Letters***, 10, 27-36, (1991).

[48] DePuy G.W., Moraga R. J., Whitehouse G. E., "Meta-RaPS: a simple and effective approach for solving the traveling salesman problem", ***Transportation Research Part E: Logistics and Transportation Review***, 41-2, 115-130, (2005).

[49] http://en.wikipedia.org/wiki/Christofides_heuristic **(Ziyaret Tarihi: 21 Mart 2007)**.

[50] Engebretsen L., Karpinski M., "TSP with bounded metrics", ***Journal of Computer and System Sciences***,72, 509–546, (2006).

[51] Lin S., Kerninghan B., "An effective heuristic algorithm for the traveling salesman problem", ***Operations Research***,21, 498-516, (1973).

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

[53] Cowling P.I., Keuthen R., "Embedded local search approaches for routing optimization", ***Computers & Operations Research***, 32, 465–490, (2005).

[54] Gamboa D., Rego C., Glover F., "Implementation analysis of efficient heuristic algorithms for the traveling salesman problem", ***Computers & Operations Research***, 33, 1154-1172, (2006).

[55] Potvin J.Y. "Genetic Algorithms for the Traveling Salesman Problem" www.iro.umontreal.ca/~potvin/ift6580/paper_potvin_ga_tsp.pdf **(Ziyaret Tarihi: 5 Aralık 2008)**.

[56] Larranaga, P., Kuijpers, C.M.H, Murga, R.H., Inza, I., Dizdarevic, S., *Genetic algorithms for the travelling salesman problem: A review of representations and operators* [Online], http://www.sc.ehu.es/ccwbayes/postscript/aireview00.ps.gz **(Ziyaret Tarihi: 15 Aralık 2008)**.

[57] Whitley, D., Starkweather, T., Shaner, D., "Scheduling Problems and Traveling Salesmen: The Genetic Edge Recombination Operator", ***Proceedings of the Third Int. Conf. On Genetic Algorithms (ICGA'89)***, 133-140,(1989)

[58] Bean, J., "Genetics and random keys for sequencing and optimization", ***ORSA Journal on Computing,*** 6 (2), 154–160. (1994)

[59] Lin, Z., Li, Y., Huang, Z.C., "Genetic Algorithm Based on Classification for the Traveling Salesman Problem", **Proceedings of the Third International Conference on Natural Computation (ICNC 2007),** 05**,** 619-623 ,(2007).

[60] Martikainen, J., Ovaska, S. J., "Hierarchical Two-Population Genetic Algorithm", ***Int. J. of Computational Intelligence Research***, 2,4, 367-380, (2006).

[61] Wang, L., Maciejewski, A.A., Siegel, H.J., Roychowdhury, V.P., "A Comparative Study of Five Parallel Genetic Algorithms Using the Traveling Salesman Problem", **Proceedings of the 12th. International Parallel Processing Symposium on International Parallel Processing Symposium,** 345, (1998).

[62] Borovska P., "Solving the Travelling Salesman Problem in Paralel by Genetic Algorithm on Multicomputer Cluster" ***International Conference on Computer Systems and Technologies - CompSysTech'06***, (2006).

[63] Katayama, K., Hirabayashi, H., Narihisa, H., "Analysis of Crossovers and Selections in a Coarse-Grained Paralel Genetic Algorithm", ***Mathematical and Computer Modelling***, 38, 1275-1282, (2003) .

[64] Sena, G. A., Megherbi, D., Isern, G., "Implementation of a parallel Genetic Algorithm on a cluster of workstations: Traveling Salesman Problem, a case study", ***Future Generation Computer Systems*** ,17, 477–488, (2001) .

[65] Rocha, M., Neves, J., "Preventing Premature Convergence to Local Optima in Genetic Algorithms via Random Offspring Generation", **Proceedings of the 12th international conference on Industrial and engineering applications of artificial intelligence and expert systems: multiple approaches to intelligent systems,** 126-136, (1999).

[66] Pongcharoen, P., Chainate, W., Thapatsuwan P., "Exploration of Genetic Parameters and Operators through Travelling Salesman Problem", ***ScienceAsia***, 33, 215-222, (2007).

[67] Jayalakshmi, G.A., Sathiamoorthy, S., Rajaram, R., "A Hybrid Genetic Algorithm - A New Approach to Solve Traveling Salesman Problem", ***Int. J. of Computational Engineering Science***, 2,2, 339-355, (2001).

[68] Pullan, W., “Adapting the Genetic Algorithm to the Travelling Salesman Problem”, ***Evolutionary Computation CEC '03***, 2, 1029-1035, (2003).

[69] Yang J., Wu C., Lee H. P., Liang Y., “Solving traveling salesman problems using generalized chromosome genetic algorithm”, ***Progress in Natural Science***, (2008).

[70] Uğur A., “Path planning on a cuboid using genetic algorithms”, ***Inform. Sci.*** , (2008)

[71] Carter, A.E., Ragsdale, C.T., “A new approach to solving the multiple traveling salesperson problem using genetic algorithms”, **European Journal of Operational Research**, 175(1), 246-257, (2006).

[72] Zacharia, P.T., Aspragathos, N.A., “Optimal robot task scheduling based on genetic algorithms”, ***Robotics and Computer-Integrated Manufacturing***, 21-1, 67-79, (2005).

[73] Hwang, H.S., “An improved model for vehicle routing problem with time constraint based on genetic”, ***Computers & Industrial Engineering***, 42(2-4), 361-369, (2002).

[74] Keskintürk, T., Söyler H., “Global Karınca Kolonisi Optimizasyonu”, ***Gazi Üniv. Müh. Mim. Fak. Der.***, 21(4), 689-698, (2006).

[75] Colorni A., Dorigo M., Maniezzo V. “Distributed Optimization by Ant Colonies” ***Proceedings of the First European Conference on Artificial Life***, Elsevier, 134–142, (1991).

[76] Dorigo M., Maniezzo V., Colorni A,. “The Ant System: Optimization by a Colony of Cooperating Agents”, ***IEEE Transactions on Systems, Man and Cybernetics – Part B*** , 26(1),29–42, (1996).

[77] Stützle, T., Dorigo M, “ACO Algorithms for the Traveling Salesman Problem in Evolutionary Algorithms” in Engineering and Computer Science: Recent Advances in Genetic Algorithms, Evolution Strategies, Evolutionary Programming, Genetic Programming and Industrial Applications, (Editörler: Miettinen, K., Makela, M., Neittaanmaki, P., Periaux, J.,), ***John Wiley & Sons***, (1999).

[78] Uğur, A., Aydın D., “Ant Sistem Algoritmasının Java ile Görselleştirilmesi” ***Yöneylem Araştırması /Endüstri Mühendisliği XXIV. Kongresi***, Adana, (2004).

[79] Dorigo, M., Gambardella, L.M., “Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem”, ***IEEE Transactions on Evolutionary Computation*** , 1(1):53–66, (1997).

[80] Gambardella, L.M., Dorigo M, “Solving Symmetric and Asymmetric TSPs by Ant Colonies”, ***Proceedings of the IEEE International Conference on Evolutionary Computation (ICEC’96)***, IEEE Press, 622–627,(1996).

[81] Dorigo, M., Gambardella, L.M., "Ant colonies for the travelling salesman problem", ***BioSystems***, 43, 73–81,(1997)

[82]Stützle, T., Hoos, H.H., "The MAX–MIN Ant System and Local Search for the Traveling Salesman Problem", ***Proceedings of the IEEE International Conference on Evolutionary Computation (ICEC'97)*** , (T. Baeck, Z. Michalewicz, and X. Yao, editors), 309–314, (1997).

[83]Stützle, T., Hoos, H.H "Improvements on the Ant System: Introducing the MAX–MIN in Ant System Artificial Neural Networks and Genetic Algorithms", (R.F. Albrecht G.D. Smith, N.C. Steele, editors), 245–249, ***Springer Verlag***, (1998).

[84] Stützle, T., Hoos, H.H., "The MAX–MIN Ant System", ***Future Generation Computer Systems***, 16, 889–914, (2000)

[85] Randall, M., Lewis, A., "A Parallel Implementation of Ant Colony Optimization", ***Journal of Parallel and Distributed Computing***, 62, 1421–1432, (2002).

[86] Chu S.C., Roddick, J.F., Pan, J.S., "Ant colony system with communication strategies", ***Information Sciences***, 167,63–76,(2004).

[87] Tsai, C.F., Tsai, C.W., Tseng, C.C., "A new hybrid heuristic approach for solving large traveling salesman problem", ***Information Sciences***, 166, 67-81, (2004).

[88] Glover, F., "Tabu search - Part I" ***ORSA J. on Computing***, 1, 190-206, (1989).

[89] Glover, F., "Tabu search - Part II" ***ORSA J. on Computing***, 2, 4-32, (1990).

[90] Bağış, A., "Determining fuzzy membership functions with tabu search-an application to control", ***Fuzzy Sets and Systems***, 139, 209-225 (2003).

[91] Cowling P.I., Keuthen R., "Embedded local search approaches for routing optimization" ***Computers & Operations Research***,32, 465-490, (2005).

[92] Tsubakitani, S., Evans, J.R., "Optimizing tabu list size for the traveling salesman problem" ***Computers & Operations Research***, 25(2), 91-97, (1998).

[93] S., Basu, D., Ghosh, "A Review of the Tabu Search Literature on Traveling Salesman Problems", ***Indian Institute of Management Ahmedabad IIMA Working Papers***, (2008).

[94] Misevicius, A.,"Using Iterated Tabu Search for the Traveling Salesman Problem", ***Informacines Technologijos ir Valdymas***, 32(3), 29-40, (2004).

[95] Misevicius, A., Smolinskas J., Tomkevicius A., "Iterated Tabu Search for the Traveling Salesman Problem: New Results", ***Information Technology and Conrol***, 34(4), 327-337, (2005).

[96] Baykoç, Ö.F., İşleyen, S.K, "Simetrik Gezgin Satıcı Problemi İçin Bir Etkin Tekrarlı Yerel Arama Algoritması", ***Teknoloji***, 10(2), 99-106, (2007).

[97] Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P., "Optimization by simulated annealing", ***Science***, 220(4598),671-680, (1983).

[98] Bakır, M. A., Altunkaynak, B., "Tamsayılı Programlama", ***Nobel Yayın Dağıtım***, İstanbul, (2003).

[99] Johnson, D.S., McGeoch, L.A., "The Traveling Salesman Problem: A Case Study in Local Optimization", in Local Search in Combinatorial Optimization, ***Wiley and Sons***, (1995).

[100] Hopfield, J.J., Tank, D.W., "Neural computation of decisions in optimization problems", ***Biol Cyber***,52,141-152, (1985).

[101] Kohonen, T, "The Self Organizing Map", ***Proc IEEE***,78,74-90, (1990).

[102] Yanping, B., Zhang, W., Jin Z., "A new self-organising maps strategy for solving the travelling salesman problem", ***Chaos, Solitons & Fractals***, 28(4), 1082-1089, (2006).

[103] Tamura, H., Zhang, Z., Xu, X., Ishii M., Tang Z.. "Lagrangian object relaxation neural network for combinatorial optimization problems" ***Neurocomputing***, 68, 297-305. (2005).

[104] Kennedy, J., Eberhart, R.C, "Particle swarm optimization", ***Proceedings of the IEEE International Conference on Neural Networks***, 4, 1942-1948, (1995).

[105] Pang, W.; Wang, K.; Zhou, C., Dong, L., Liu, M.; Zhang, H. & Wang, J., "Modified particle swarm optimization based on space transformation for solving traveling salesman problem", ***Proceedings of the Third International Conference on Machine Learning and Cybernetics***, 2342-2346, (2004).

[106] Pang, W.; Wang, K.; Zhou, C., Dong, L., "Fuzzy discrete particle swarm optimization for solving traveling salesman problem", ***Proceedings of the Fourth International Conference on Computer and Information Technology (CIT'04)***, 796-800, (2004).

[107] Wang, K., Huang, L., Zhou, C., Pang., W., " P particle swarm optimization for solving traveling salesman problem", ***Proceedings of the Second International Conference on Machine Learning and Cybernetics***, 1583-1585, (2003).

[108] Wang, C.Z.J.S.Y., Yang, Q., "An Improved Discrete Particle Swarm Optimization Algorithm for TSP" ***2007 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology - Workshops***, 35-38, (2007).

[109] Zhong, W., Zhang J., Chen W., "A Novel Discrete Particle Swarm Optimization to Solve Traveling Salesman Problem" , ***Evolutionary Computation, 2007. CEC 2007***, 3283 - 3287, (2007).

[110] Goldbarg, E.F.G., Goldbarg, M.C., Souza G.R.D., "Particle Swarm Optimization Algorithm for the Traveling Salesman Problem", ***Travelling Salesman Problem***, (editör: Federico Greco), 75-95, (2008).

[111] Storn, R., Price, K., "Differential Evolution - A simple and efficient adaptive scheme for global optimization over continuous spaces", ***Technical Report TR-95-12, ICSI***, (1995).

[112] **3**. Storn, R., Price, K., "Differential Evolution-A simple and efficient heuristic for global optimization over continuous spaces", ***Journal of Global Optimization***, 11, 241–354,(1997).

[113] Price, K.,Storn, R., Lampinen, J. "Differential Evolution A Practical Approach to Global Optmization", ***Springer Natural Computing Series***, VI-VII ,(2005).

[114]Storn, R., *Differential Evolution (DE) for Continuous Function Optimization* [Online], http://www.icsi.berkeley.edu/~storn/code.html, **(Ziyaret Tarihi: 7 Aralık 2008)**.

[115] Lampinen, J., 2002, *A Bibliography of Differential Evolution Algorithm* [Online] http://www2.lut.fi/~jlampine/debiblio.htm, **(ZT: 11 Aralık 2008)**.

[116] Nearchou, C.A., Omirou S.L:, "Differential evolution for sequencing and scheduling optimization", ***Journal of Heuristics***, 12:395-411,(2006).

[117] Bean, J., "Genetics and random keys for sequencing and optimization". ***ORSA Journal on Computing,*** 6 (2), 154–160, (1994).

[118] Qian, B, Wanga, L., Huanga, D., Wangc, W., Wanga, X, "An effective hybrid DE-based algorithm for multi-objective flow shop scheduling with limited buffers" ***Computers and Operations Research*** (2007).

[119] Onwubolu, G., Davendra, D., "Scheduling flow shops using differential evolution algorithm", ***European Journal of Operational Research*** 171, 674-692. (2006).

[120] Nearchou, C.A., "Meta-heuristics from nature for the loop layout design problem", ***International Journal of Production Economics***, 312-328,(2006).

[121] Angira R., Babu B.V., "Optimization of process synthesis and design problems: Amodified differential evolution approach", ***Chemical Engineering Science***, 61, 4707 – 4721, (2006).

[122] Nearchou, C.A., “Balancing large assembly lines by a new heuristic based on differential evolution method”, ***International Journal of Advanced Manufacturing Technologies***, 34:1016–1029. (2007)

[123] Al-Anzi, F., Allahverdi, A, “A self-adaptive differential evolution heuristic for two-stage assembly scheduling problem to minimize maximum lateness with setup times”, ***European Journal of Operational Research*** ,182: 80–94,(2007).

[124] Schmidt, H., Thierauf, G., “A combined heuristic optimization technique”, ***Advances in Engineering Software***, 36, 11-19, (2005).

[125] Zaharie, D., “Critical values for the control parameters of differential evolution algorithms” ***Proceedings of MENDEL 2002, eighth international mendel conference on soft computing***, Brno, Çek Cumhuriyeti, 62-70, (2002)

[126] Nearchou, C.A., “A differential evolution approach for the common due date early/tardy job scheduling problem”, **Computers & Operations Research**. 35,1329-1343, (2008).

[127] Coelho, L.D.S.,” Reliability–redundancy optimization by means of a chaotic differential evolution approach”, ***Chaos, Solitons & Fractals***, (2008).

[128] http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsplib.html **(Ziyaret Tarihi: 29 Haziran 2008)**

ÖZGEÇMİŞ

1979 yılında Trabzon'da doğdu ilk ve orta öğrenimini Trabzon'da tamamladı. Lisans eğitimini 2001 yılında mezun olduğu Dokuz Eylül Üniversitesi Mühendislik Fakültesi Endüstri Mühendisliği Bölümü'nde aldı. 2001 – 2004 yılları arasında, Kocaeli Üniversitesi, Fen Bilimleri Enstitüsü, Endüstri Mühendisliği Anabilim Dalı'nda Yüksek Lisans Öğrenimini tamamladı. 2001 yılından beri Kocaeli Üniversitesi Endüstri Mühendisliği Bölümü Endüstri Mühendisliği Anabilim Dalı'nda Araştırma Görevlisi olarak görev yapmaktadır.