YILDIZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

GEZGİN SATICI PROBLEMİ TABANLI BİR SİSTEMİN DİNAMİK BULANIK GENETİK ALGORİTMALAR İLE OPTİMİZASYONU

Erdinç KURUCA

FBE Endüstri Mühendisliği Anabilim Dalı Sistem Mühendisliği Programında Hazırlanan

YÜKSEK LİSANS TEZİ

Tez Danışmanı : Prof. Dr. Hüseyin Başlıgil

İSTANBUL, 2009

İÇİNDEKİLER

KISALTMA LİSTESİ

GSP: Gezgin Satıcı Problemi (Traveling Salesman Problem, TSP)
AHP: Analitik Hiyerarşi Prosesi (Analytic Hierarchy Process)
GA: Genetik Algoritma

ŞEKİL LİSTESİ

ÇİZELGE LİSTESİ

Sayfa

ÖNSÖZ

Bulanık mantık ve Genetik Algoritmalar günümüzün en gözde araştırma konularındandır. Her iki konu da pek çok alandaki uygulamalarının sonuçları ile potansiyellerini kanıtlamışlardır.

Bu çalışmada Bulanık Mantık ve Genetik Algoritmaların bir sentezi olan Bulanık Genetik Algoritmaların dinamik bir öncelik kısıtlı lojistik (kargo dağıtımı) problemine uygulanması incelenecektir.

Bu çalışmada özellikle GPX Global Express Satış Müdürü Sn. Gökay Çetinkol, Ernova Lojistik Satış Müdürü Sn. Barış Güzel, Kayas Gümrük Müşavirliği Genel Müdürü Sn. Hakan Kaya, TNT Özel Müşteriler Müdürü Sn. Özgür Özçelik, DHL Özel Müşteriler Müdürü Sn. Cenk Acar ve çalışmalarıma ilk günden beri yön veren ve destekleyen tez danışmanım Sn. Prof. Dr. Hüseyin Başlıgil başta olmak üzere benden yardımlarını esirgemeyen herkese teşekkürü bir borç bilirim.

Erdinç Kuruca

Mayıs 2009

ÖZET

Günümüzde hem teorik hem de uygulamaya yönelik problemler gittikçe karmaşıklaşmaktadır. Bu durum problemlerin klasik yöntemlerle çözülmesini zorlaştırmaktadır. Buna karşın gittikçe gelişen bilgisayar teknolojisinin de yardımıyla hesap temelli çözümlere bir alternatif olarak doğrudan arama teknikleri geliştirilmiştir. Bu yöntemlerin en yaygın kullanılanı Genetik Algoritmalardır. Genetik algoritmalar doğrudan optimum sonucu bulmazlar fakat çözüm uzayında optimum çözümü hızlı ve etkin bir biçimde ararlar. Böylece optimum çözümün bulunması için geçecek zaman göz önünde bulundurulduğunda yeterli derecede iyi sonuçları hızlı bir biçimde sağlarlar.

Günümüz problemlerinin bir diğer özelliği ise zaman kısıtıdır. Problemler tanımlanırken ve çözülürken en optimum çözüm ve en doğru tanımlama, harcanan zaman göz önüne alındığında gereklilik arz etmeyebilmektedir. Bu bağlamda problemlerin net değerler ile tanımlanması artık bir zorunluluk değildir. Bu zorunluluğu ortadan kaldıran yöntem Bulanık Mantık'tır. Bulanık mantıkta klasik mantıkta olduğu gibi 1-0 gibi tanımlamalar yerine bu tanımlamalara üyelik miktarlarını gösteren üyelik fonksiyonları söz konusudur.

Bu iki yöntemin birlikte kullanılması ile Bulanık Genetik Algoritmalar ortaya çıkmıştır. Bu yöntem karmaşık problemlerin hem tanımlanmasını hem de optimizasyonunu kolaylaştırmaktadır.

Bu çalışmada Bulanık Genetik Algoritmaların örnek bir dinamik kargo problemine uygulanması incelenmektedir. Bu problem klasik bir öncelik kısıtlı Gezgin Satıcı Problemi olarak tanımlanmıştır.

Anahtar Kelimeler: Bulanık Mantık, Genetik Algoritmalar, Bulanık Genetik Algoritmalar, AHP, Kargo Dağıtımı Problemi

ABSTRACT

Currently, both theoretical and practical problems are getting more complex day by day. As a result solving problems with the classical calculation based methods are getting harder. However, by the help of rapidly growing computer technology, direct search algorithms are developed as an alternative to the calculation based methods. Genetic Algorithm is one of the most common direct search algorithms. Genetic Algorithms do not directly finds the optimal solution, however they quickly search the solution space for optimal solutions. They generally find out reasonable solutions in a short time comparing necessary time required to find out the exact solution.

Another characteristic of the current problems is the time constraint. The exact solution and the exact description of the problem may not be a requirement comparing the time necessary for the calculations. As a result problems may not have to be defined in distinct values. The method eliminates this necessity is called Fuzzy Logic. In contrast with the classical logic, in fuzzy logic there is no definition like 0-1, though there are membership functions which show the membership values of these definitions.

The integration of these two methods is called the Fuzzy Genetic Algorithms. This new method both simplifies the definition and the optimization of complex problems and also approaches the computer solution of the problem to the basis of human thinking.

In this study, Fuzzy Genetic Algorithms is applied to a sample dynamic cargo distribution problem. This problem is defined as a classical Travelling Salesman Problem with precedence constraint.

Keywords: Genetic Algorithms, Fuzzy Logic, Fuzzy Genetic Algorithms, Fuzzy AHP, Cargo Distribution Problem

1. GİRİŞ

Günümüzde araştırma alanları ve problemler eskiye göre çok daha karışıktır. Bu karışıklık problemleri etkileyen parametre ve değişken sayılarının fazlalığından ve problemlerin çözüm kümelerinin boyutlarının büyümesinden kaynaklanmaktadır. Bu durum bir taraftan elde edilen çözümün değerlendirilmesinde zorluk çıkarırken diğer taraftan çözümlerde lineer yöntemlerin uygulanmasını imkânsız hale getirmektedir. Böylece optimizasyon problemlerinde optimum sonuçları bulacak klasik lineer yöntemlerin dışında optimum sonuçları ya da optimum sonuçların hesaplanması için geçecek süre göz önünde bulundurulduğunda yeterli derecede uygun sonuçları hızlı bir şekilde bulacak yeni yöntemler, yani yeni arama teknikleri, önem kazanmıştır.

Mühendislik, bilim, ekonomi, finans v.b. alanlardaki problemleri çözmede kullanılan arama teknikleri, hesap-temelli ve doğrudan arama teknikleri olarak ikiye ayrılabilir. Eğer problemler sayısal veya analitik olarak iyi tanımlanabiliyorsa veya çözüm uzayı küçük ve tek ise, hesap temelli arama teknikleri daha iyi çalışır. Buna rağmen hesap-temelli teknik mühendislik uygulamalarında gittikçe artan optimum bulma fonksiyonlarında oldukça zayıf kalır. Sadece fonksiyon bilgisi gerekli olan doğrudan arama tekniği ise, hesap-temelli teknikten daha kısa sürede işler ve daha etkilidir. Doğrudan arama tekniğinin esas problemi, kullanılan bilgisayar zamanı ile ulaşılan çözümün kesinliği arasındaki bağıntıdır (Goldberg, 1989).

Optimizasyon problemleri pek çok uygun çözüme sahip olduklarından uğraştırıcıdırlar. Uğraştırıcı olan büyük çözüm uzayında optimum sonucu aramak ve tespit etmektir. Optimum sonucu bulmak için her sonucu tek tek değerlendirmek çok zor olacağından bazı optimizasyon yöntemleri geliştirilmiştir (Xing v.d., 2008). Bu yöntemlerin en yaygın kullanılanlarından biri de genetik algoritmadır. Çok iyi bilinmektedir ki genetik algoritmalar, bir noktalar kümesinde eş zamanlı olarak optimumu arar ve klasik yöntemlere göre daha fazla global optimumu bulma şansına sahiptir (Qu, 1999).

Bu çalışmada günümüzde gittikçe popüler bir lojistik problemi haline gelmekte olan kargo dağıtımı probleminin, Gezgin Satıcı Problemi (GSP - Traveling Salesman Problem - TSP) tabanlı bir modelleme ile dinamik bir Bulanık Genetik Algoritma ile çözümü ele alınacaktır. Algoritma, önceden AHP ile hazırlanmış bulanık önem ağırlıklarını ve algoritmaya başlangıçta verilen bulanık önem derecelerini kullanarak bulanık bir uyum

fonksiyonuna sahip genetik algoritma ile kargo probleminin dinamik olarak takibi ve optimizasyonunu yapacaktır.

1.1. Literatür Araştırması

Bu çalışmada incelenecek olan problem bir kargo dağıtımı problemidir. Bu problem gezgin satıcı problemi olarak modellenecektir. Gezgin satıcı problemlerinin Genetik algoritmalarla çözümü hakkında literatürde sekiz adet makaleye rastlanmıştır. Bu makaleler ve konuları aşağıdaki tabloda verilmiştir.

Tablo 1.1. Literatürdeki GSP'nin Genetik Algoritmalar ile çözümü hakkındaki çalışmalar

Sıra	Başlık	Yazar, Yıl	Konu	Teknik	Amaç
1.	A synergetic approach to genetic algorithms for solving traveling salesman problem	Qu, L., Sun, R. (1999)	Gezgin satıcı probleminin çözümünde Genetik algoritma kullanımının incelenmesi	Genetik algoritma	Gezgin satıcı probleminin çözümünde genetik algoritma kullanımının incelenerek üstel bağıntı, entropi atlaması, asimilasyon ve entropi senkrenizasyonu gibi davranışların incelenmesi. Bunların genetik algoritma tasarımında erken yakınsama ihtimaline etkisini gözlemlemek.
2.	Case injected genetic algorithms for traveling salesman problems	Louis, S.J., Li, G. (1999)	Hafıza sahibi bir Genetik algoritma ile Gezgin satıcı probleminin çözülmesi	Uzun dönem hafızalı bir Genetik algoritma	Öğrenebilen bir genetik algoritma tasarımı ve bu algoritma ile gezgin satıcı probleminde önceden alınan sonuçları kullanarak yeni problemlerde daha hızlı çözümler bulunabilmesi.
3.	The efficiency of hybrid mutation genetic algorithm for the travelling salesman problem	Katayama, K., Sakamoto, H., Narihisa, H. (2000)	Gezgin satıcı probleminin çözümünde geliştirilmiş bir genetik algoritma kullanımı	Geliştirilmiş bir crossover yöntemi kullanan Genetik algoritma	Bu makalede ileri sürülen genetik algoritma, gezgin satıcı probleminin çözümünde değişik bir crossover yöntemi kullanmaktadır. Bu crossover yönteminde alt turlar bir bütün olarak çaprazlanmakta böylece etkinlik artırılmaya çalışılmaktadır.

4.	An efficient genetic algorithm for the traveling salesman problem with presedence constraints	Moon, C., Kim, J., Choi, G., Seo, Y. (2002)	Öncelik kısıtları olan bir Gezgin satıcı probleminde Genetik algoritma kullanımı	Özel geliştirilmiş bir crossover tekniği kullanan bir Genetik algoritma	Öncelik kısıtları olan bir gezgin satıcı problemlerinde genetik algoritma çözümlerinin klasik çözümlerle karşılaştırılması
5.	A genetic algorithm with a mixedregion search for the asymmetric traveling salesman problem	Choi, I.C., Kim, S. I., Kim, H. S. . (2003)	Asimetrik gezgin satıcı probleminde genetik algoritma ile optimizasyonu	Bilerek kötü sonuçları da üreten ve popülasyonda belli bir oranda tutan geliştirilmiş bir genetik algoritma	Asimetrik gezgin satıcı probleminde geliştirilmiş bir genetik algoritma ile karışık alan taraması yapılarak optimizasyon yapılması ve klasik çözümlerle karşılaştırılması
6.	A random-key genetic algorithm for the generalized traveling salesman problem	Snyder, L., Daskin, M. (2006*)*	Genel gezgin satıcı probleminin rastsal anahtar gösterimini kullanan bir GA ile çözümü	Rastsal anahtar gösterimini kullanan bir genetik algoritma	Genel gezgin satıcı probleminin genetik algoritmalar ile çözümü ve sezgisel yöntemlerle karşılaştırılması
7.	A hybrid approach combining an improved genetic algorithm and optimization strategies for the asymmetric traveling salesman problem	Xing L. N.,Chen Y., Yang K., Hou, F., Shen, X., Cai, H. (2008)	Asimetrik gezgin satıcı probleminde genetik algoritma ve optimizasyon yöntemlerinin kullanılması	Geliştirilmiş Genetik Algoritma (IGA)	Asimetrik gezgin satıcı probleminin çözümünde optimizasyon teknikleri ile geliştirilmiş bir genetik algoritmanın kullanılması ve klasik genetik algoritmalar ile karşılaştırılması.
8.	Solving traveling salesman problem using generalized chromosome genetic algorithm	Yang, J., Wu, C., Lee, H.P., Liang, Y. (2008)	Genelleştirilmiş kromozom tekniği ile Gezgin satıcı probleminin çözülmesi	GCGA (Genelleştirilmiş kormozomlu genetik algoritma)	Yazarların önceki çalışmalarında ileri sürdükleri bir kromozom gösterim şekli olan Genelleştirilmiş kromozom tekniği ile gezgin satıcı probleminin çözülmesi

Tablodan görülebileceği gibi çalışmalar gezgin satıcı problemine genetik algoritmaların uygulanması ile başlamış ve zaman içinde gelişerek özel yöntemler kullanan geliştirilmiş genetik algoritmalara kadar ilerlemiştir. Literatürde bu problemin çözümünde kullanılan herhangi bir bulanık genetik algoritma örneğine rastlanmamıştır.

Bu çalışmada önceki araştırmalarda ele alınan öncelik kısıtlı gezgin satıcı probleminin bulanık genetik algoritmalarla çözülmesi ele alınacaktır.

2. GENETİK ALGORİTMALAR

2.1. Tanım

Genetik algoritma, büyük ve lineer olmayan arama uzaylarında geleneksel hesaplama yöntemlerinin işe yaramadığı durumlarda kullanılabilen doğal seleksiyon ve doğal genetiğin yöntemlerini kullanan bir arama algoritmasıdır (Louis, Li, 2000). Genetik algoritma, bir veri grubundan özel bir veriyi bulmak için kullanılır. Genetik algoritmalar 1970'lerin başında John Holland tarafından ortaya atılmıştır. Genetik Algoritmalar, Evrimsel Genetik ve Darwin'in Doğal seleksiyonuna benzerlik kurularak geliştirilmiş "iteratif", ihtimali bir arama metodudur.

"Genetik algoritmalar yapay zekanın gittikçe genişleyen bir kolu olan evrimsel hesaplamalar tekniğinin bir alt koludur. Genetik algoritmalar Darvin'in evrim teorisinin doğal seçilim ilkesinden esinlenerek oluşturulmuştur. Herhangi bir problemin genetik algoritma ile çözümü, problemin sanal olarak evrimden geçirilmesi ile gerçekleştirilmektedir." (Eiben & Smith, 2003)

Genetik algoritmalar doğada geçerli olan en iyinin yaşaması (Survival of the Fittest) kuralına dayanarak sürekli iyileşen çözümler üretir. Bunun için "iyi"nin ne olduğunu belirleyen bir *uygunluk* (fitness) fonksiyonu ve yeni çözümler üretmek için *yeniden kopyalama* (recombination), *mutasyon* (mutation) gibi operatörleri kullanır. Genetik algoritmaların bir diğer önemli özelliği de bir grup çözümle uğraşmasıdır. Bu sayede çok sayıda çözümün içinden iyiler seçilip kötüler ise elenebilir.

Genetik algoritmalar geleneksel yöntemler ve tabu arama gibi bazı metasezgisel arama metotlarında olduğu gibi iyi bir sonucu ele alıp onu geliştirmek için çok çalışmazlar. Aksine çok fazla sonucu aynı anda ele alır ve her bir sonuçta çok az çalışırlar (Qu, 1999). Genetik algoritmanın daha hızlı olmasının temel nedeni budur.

Problemin bireyler içindeki gösterimi problemden probleme değişiklik gösterir. Genetik algoritmaların problemin çözümündeki başarısına karar vermedeki en önemli faktör, problemin çözümünü temsil eden bireylerin gösterimidir. Popülasyon içindeki her bireyin problem için çözüm olup olmayacağına karar veren bir uygunluk fonksiyonu vardır. Uygunluk fonksiyonundan dönen değere göre yüksek değere sahip olan bireylere, popülasyondaki diğer bireyler ile çoğalmaları için fırsat verilir. Bu bireyler çaprazlama işlemi sonunda çocuk adı verilen yeni bireyler üretirler. Çocuk kendisini meydana getiren ebeveynlerin (anne, baba) özelliklerini taşır. Yeni bireyler üretilirken düşük uygunluk değerine sahip bireyler daha az seçileceğinden bu bireyler bir süre sonra popülasyon dışında

bırakılırlar. Yeni popülasyon, bir önceki popülasyonda yer alan uygunluğu yüksek bireylerin bir araya gelip çoğalmalarıyla oluşur. Aynı zamanda bu popülasyon önceki popülasyonun uygunluğu yüksek bireylerinin sahip olduğu özelliklerin büyük bir kısmını içerir. Böylelikle, pek çok nesil aracılığıyla iyi özellikler popülasyon içersinde yayılırlar ve genetik işlemler aracılığıyla da diğer iyi özelliklerle birleşirler. Uygunluk değeri yüksek olan ne kadar çok birey bir araya gelip, yeni bireyler oluştururs arama uzayı içerisinde o kadar iyi bir çalışma alanı elde edilir.

Genetik algoritmalarda, probleme ait en iyi çözümün bulunabilmesi için;

- Bireylerin gösterimi doğru bir şekilde yapılmalı,
- Uygunluk fonksiyonu etkin bir şekilde oluşturulmalı,
- Doğru genetik işlemciler seçilmelidir.

Bu durumda çözüm kümesi problem için bir noktada birleşecektir. Genetik algoritmalar, diğer optimizasyon yöntemleri kullanılırken büyük zorluklarla karşılaşılan, oldukça büyük arama uzayına sahip problemlerin çözümünde başarı göstermektedir. Bir problemin bütünsel en iyi çözümünü bulmak için garanti vermezler. Ancak problemlere makul bir süre içinde, kabul edilebilir, iyi çözümler bulurlar. Genetik algoritmaların asıl amacı, hiçbir çözüm tekniği bulunmayan problemlere çözüm aramaktır. (Eiben & Smith, 2003)

Kendilerine has çözüm teknikleri olan özel problemlerin çözümü için mutlak sonucun hızı ve kesinliği açısından genetik algoritmalar kullanılmazlar. Genetik algoritmalar ancak;

- Arama uzayının büyük ve karmaşık olduğu,
- Mevcut bilgiyle sınırlı arama uzayında çözümün zor olduğu,
- Problemin belirli bir matematiksel modelle ifade edilemediği,
- Geleneksel optimizasyon yöntemlerinden istenen sonucun alınmadığı alanlarda etkili ve kullanışlıdır. (Eiben & Smith, 2003)

Genetik algoritmaları diğer algoritmalardan ayıran en önemli özelliklerden biri de seçilimdir. Genetik algoritmalarda çözümün uygunluğu onun seçilme şansını arttırır ancak bunu garanti etmez. Seçim de ilk grubun oluşturulması gibi rastgeledir ancak bu rastgele seçimde seçilme olasılıklarını çözümlerin uygunluğu belirler.

Genetik Algoritmaları (GA) diğer metotlardan ayıran noktalar şu şekilde sıralanabilir:

- GA, sadece bir arama noktası değil, bir grup arama noktası (adaylar) üzerinde çalışır. Yani arama uzayında, **yerel** değil ***küresel*** arama yaparak sonuca ulaşmaya çalışır. Bir tek yerden değil bir grup çözüm içinden arama yapar.

- GA, arama uzayında bireylerin uygunluk değerini bulmak için sadece “amaç - uygunluk fonksiyonu” (objective-fitness function) ister. Böylelikle sonuca ulaşmak için türev ve diferansiyel işlemler gibi başka bilgi ve kabul kullanmaya gerek duymaz.
- Bireyleri seçme ve birleştirme aşamalarında deterministik kurallar değil “olasılık kuralları” kullanır.
- Diğer metotlarda olduğu gibi doğrudan parametreler üzerinde çalışmaz. Genetik Algoritmalar, optimize edilecek parametreleri kodlar ve parametreler üzerinde değil, bu kodlar üzerinde işlem yapar. Parametrelerin kodlarıyla uğraşır. Bu kodlamanın amacı, orijinal optimizasyon problemini kombinasyonel bir probleme çevirmektir.
- Genetik algoritma ne yaptığı konusunda bilgi içermez, nasıl yaptığını bilir. Bu nedenle kör bir arama metodudur.
- Olasılık kurallarına göre çalışırlar. Programın ne kadar iyi çalıştığı önceden kesin olarak belirlenemez. Ama olasılıkla hesaplanabilir.
- GA, kombinasyonel bir atama mekanizmasıdır.

Genetik Algoritmalar, yeni bir nesil oluşturabilmek için 3 aşamadan geçer :

1. Eski nesildeki her bir bireyin uygunluk değerini hesaplama.
2. Bireyleri, uygunluk değerini göz önüne alarak (uygunluk fonksiyonu) kullanılarak seçme.
3. Seçilen bireyleri, çaprazlama (crossover), mutasyon (mutation) gibi genetik operatörler kullanarak uyuşturma.

Algoritmik bakış açısından bu aşamalar, mevcut çözümleri lokal olarak değiştirip birleştirmek olarak görülebilir.

Genetik Algoritmalar; başlangıçta bilinmeyen bir arama uzayından topladığı bilgileri yığıp, daha sonraki aramaları alt arama uzaylarına yönlendirmek için kullanılır.

2.2. Tarihçe

Evrimsel hesaplamanın (Evolutional Computing) en çok kullanılan yaklaşımı olan genetik algoritmalar, 1960’larda John Holland tarafından bulundu. *Adaptation in Natural and Artificial Systems* (*Doğal ve Yapay Sistemlerde Adaptasyon*) adlı kitabında Holland GA’yı doğadaki adaptasyondaki teorik çerçevede açıkladı. Holland’ın çıkış noktası oldukça bilimseldi. Doğadaki değişik fenomenleri anlamaya ve onlarla bağlantılar kurmaya çalışırken aynı zamanda potansiyel mühendislik uygulamaları önermekteydi. Holland’ın kitabının

yayınlanmasından sonra GA yapay zeka ve makine öğrenmesi konularında büyük bir alt alan oldu (Mitchell, 1998).

1985 yılında Holland'ın öğrencisi olarak doktora yapan David E. Goldberg adlı inşaat mühendisi 1989'da konusunda bir klasik sayılan kitabını yayınlayana kadar genetik algoritmaların pek pratik yararı olmayan bir araştırma konusu olduğu düşünülüyordu. Goldberg, GA'nın çok sayıda kollara ayrılmış gaz borularında, gaz akışını düzenlemek ve kontrol etmek için bir uygulamasını yapmıştır.

Goldberg'in gaz boru hatlarının denetimi üzerine yaptığı doktora tezi ona sadece 1985 National Science Foundation Genç Araştırmacı ödülünü kazandırmakla kalmadı, genetik algoritmaların pratik kullanımının da olabilirliğini kanıtladı. Ayrıca kitabında genetik algoritmalara dayalı tam 83 uygulamaya yer vererek GA'nın dünyanın her yerinde çeşitli konularda kullanılmakta olduğunu gösterdi.

Günümüzde genetik algoritma, uzman sistemler ve yapay sinir ağları ile birlikte yapay zeka uygulamalarının ana araçlarından biri haline gelmiştir (Qu, 1999).

2.3. Uygulama Alanları

Genetik algoritmalar biyoloji, kimya, bilgisayar bilimleri ve sosyal bilimler gibi birçok alana uygulanmıştır (Moon v.d., 2002).

Genetik algoritmaların en uygun olduğu problemler geleneksel yöntemler ile çözümü mümkün olamayan ya da çözüm süresi problemin büyüklüğü ile üstel orantılı olarak artan (NP-hard) problemlerdir. Bugüne kadar GA ile çözümüne çalışılan konulardan bazıları şunlardır:

- *Optimizasyon:* GA, sayısal optimizasyon ve kombinasyonel optimizasyon problemleri olan devre tasarımı, doğrusal olmayan denklem sistemlerinin çözümü ve fabrika-üretim planlamasında kullanılır.
- *Otomatik Programlama (automatic programming):* GA, bilgisayar programları yardımıyla network sıralamasında (sorting),sınav/ders programı hazırlanmasında kullanılır.
- *Makine öğrenmesi (machine learning):* GA, robot sensorlerinde, yapay sinir ağlarında, VLSI yonga tasarımı ve protein yapısal analizinde kullanır.
- *Ekonomi (economics):* GA, ekonomik modellerin geliştirilmesinde ve işlemesinde kullanılır.
- *İmmün sistemler (Immune systems):* GA, çok-gen'li ailelerin evrimi esnasında ve doğal immün sistem modellerinde kullanılır.

- *Popülasyon genetiği (population genetics):* GA, evrim ile ilgili sorulara cevap bulmada kullanılır.
- *Evrim ve öğrenme (evolution and learning):* GA, fertlerin öğrenmesinde ve türlerin evrilmesinde kullanılır.
- *Sosyal sistemler (social systems):* GA, sosyal sistemlerin analizinde kullanılır.
- *Müzik.* Stokastik müzik üretecinin çıktısından elde edilen materyali sınırlayan bir takım veri filtreleri ile müzik oluşturma GA'nın bir uygulamasıdır. Bunun için algoritmik düzenin yapısındaki değişiklikler tanımlanır ve bunların çıktıları müzikal örnekler verirler.

Bunlar dışında fonksiyon optimizasyonu, resim tanıma, sinyal işleme, robotik gibi güncel mühendislik problemlerinin çoğuna GA uygulanabilir (Qu, 1999).

2.4. Temel Kavramlar

Genetik algoritma tabanlı algoritmaların geliştirilmesindeki temel konular kromozom gösterimleri (kodlama), başlangıç popülasyonunun oluşturulması, evrim ölçüsü (uygunluk), çaprazlama, mutasyon ve seleksiyon (seçilim) stratejisidir (Moon v.d., 2002).

Bahsedilen çözümler kümesine popülasyon denir. Bu popülasyonun her bir elemanı bir birey ya da kromozom olarak tanımlanır ve bu birey çözümün kodlanmış bir biçimidir. (Qu, 1999)

2.4.1. Kodlama (Bireylerin Gösterimi)

Herhangi bir evrimsel algoritmanın ilk adımı problemde aday sonucun nasıl gösterileceğine karar verilmesidir. Bu adım genotipin belirlenmesi ve genotipin fenotipe haritalanmasını içerir (Eiben & Smith, 2003).

Kodlamanın amacı tıpkı biyolojideki genetikte olduğu gibi, bir çözümün kromozomu meydana getirecek genler şekline sokulmasıdır (Qu, 1999). Kodlama planı Genetik algoritmanın önemli bir kısmını teşkil eder. Çünkü bu plan bilginin çerçevesini şiddetle sınırlayabilir. Öyle ki probleme özgü bilginin bir kromozomsal gösterimiyle temsili sağlanır. Kromozom genellikle, problemdeki değişkenlerin belli bir düzende sıralanmasıdır.

Kromozomu oluşturmak için sıralanmış her bir değişkene “gen” adı verilir. Buna göre bir gen kendi başına anlamlı genetik bilgiyi taşıyan en küçük genetik yapıdır. Mesela; 101 bit dizisi bir noktanın x-koordinatının ikilik düzende kodlandığı gen olabilir. Aynı şekilde bir kromozom ise bir ya da daha fazla genin bir araya gelmesiyle oluşan ve problemin çözümü

için gerekli tüm bilgiyi üzerinde taşıyan genetik yapı olarak tanımlanabilir. Örnek vermek gerekirse; 100011101111 x1, y1, x2, y2 koordinatlarından oluşan iki noktanın konumu hakkında bize bilgi verecektir.

Bu parametreleri kodlarken dikkat edilmesi gereken en önemli noktalardan biri ise kodlamanın nasıl yapıldığıdır. Kodlama stratejisi, her optimizasyon problemi için farklıdır (Qu, 1999). Örnek olarak kimi zaman bir parametrenin doğrusal ya da logaritmik kodlanması GA performansında önemli farka yol açar.

Kodlamanın diğer önemli bir hususu ise kodlama gösteriminin nasıl yapıldığıdır. Bu da yeterince açık olmamakla birlikte GA performansını etkileyen bir noktadır.

Problemin çözülebilmesi için doğru gösterimin seçilmesi önemlidir. İyi bir evrimsel algoritma tasarlamanın en zor kısmı gösterimi doğru yapmaktır. Bu genelde pratikle ve uygulama alanını iyi tanımakla sağlanır (Eiben & Smith, 2003).

2.4.1.1. Permütasyon kodlama

Gezgin satıcı problemi veya iş sıralama problemi gibi düzenleme problemlerinde kullanılır. Burada her kromozom sayıları bir sırada temsil eden bir sayı katarıdır.

Tablo 2.1. Permütasyon kodlamalı kromozom örnekleri

Kromozom A	1 5 3 2 6 4 7 9 8
Kromozom B	8 5 6 7 2 3 1 4 9

Permütasyon kodlama sadece sıralama problemleri için kullanışlıdır.

2.4.1.2. Değer kodlama

Reel sayılar gibi karmaşık değerlerin kullanıldığı problemlerde direk değer kodlanması kullanılabilir. Bu tip problemler için ikilik sistem (binary) kodlama işi çok zordur ve kullanışlı değildir.

Tablo 2.2. Değer kodlamalı kromozom örnekleri

Kromozom A	1.2324 5.3243 0.4556 2.3293 2.4545
Kromozom B	ABDJEIFJDHDIERJFDLDFLFEGT
Kromozom C	(back), (back), (right), (forward), (left)

Bu yöntem bazı özel problemler için çok iyidir. Diğer taraftan bu tip kodlama için probleme özel yeni bazı mutasyon ve çaprazlamalar geliştirmek lazımdır.

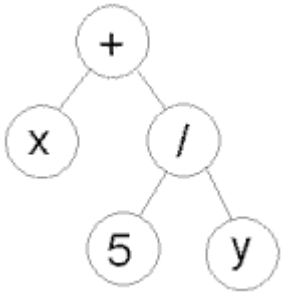
2.4.1.3. Ağaç kodlama

Bu yöntem, genetik kodlama gibi gelişen ve değişen programlar veya ifadeler için kullanılır.

Ağaç kodlamada her kromozom bazı nesnelerin, mesela fonksiyonlar ya da programlama dilindeki komutlar gibi, bir ağacıdır

Kromozom A

(+ x (/ 5 y))



Şekil 2.1. Ağaç kodlamalı kromozomlar

2.4.2. Uygunluk (Uyum)

Başlangıç topluluğu bir kez oluşturulduktan sonra evrim başlar. Genetik algoritma bireylerin uygunluk ve iyiliklerine göre ayrılıp fark edilmesine gerek duyar. Uygunluk, topluluktaki bir kısım bireyin problemi nasıl çözeceği için iyi bir ölçüdür. O problem parametreleri kodlamayla ölçülür ve uygunluk fonksiyonuna girdi olarak kullanılır. Yüksek ihtimalle uygun olan bu üyeler tekrar üreme, çaprazlama ve mutasyon operatörleriyle seçilirler.

Bazı problemler için bireyin uygunluğu, bireyden elde edilen sonuç ile tahmin edilen sonuç arasındaki hatadan bulunabilir. Daha iyi bireylerde bu hata sıfıra yakın olur. Bu hata genellikle, girişin tekrar sunulacak kombinasyonlarının ortalaması veya toplamıyla hesaplanır (değerler değişkenlerden bağımsızdır). Beklenen ve üretilen değer arasındaki korelasyon etkeni, uygunluk değerini hesaplamak için kullanılabilir.

Amaç fonksiyonu her bir kromozomun durumunu değerlendirmek için mekanizmayı sağlayan ana bir kaynaktır. Bu GA ve sistem arasında önemli bir bağlantıdır. Fonksiyon girdi olarak kodu çözülmüş şekilde kromozom alır ve kromozomun performansına bir ölçü olarak

bir objektif değer üretir. Bu diğer kromozomlar için de yapıldıktan sonra bu değerler kullanılarak, uygun değerler uygunluk fonksiyonuyla hesaplanıp belli bir düzende planlanır. Bu planlamayı sağlayan ve uygunluk teknikleri olarak bilinen birçok yöntem vardır. Çoğu ortak kullanılan bu yöntemler şunlardır:

2.4.2.1. Pencereleme

Popülasyonda en kötü kromozomun objektif değerinin V_w olduğunu kabul edersek her bir kromozomun i ve en kötü kromozomu arasında farkla orantılı bir uygunluk değeri f_i atanabilir. Bu durum matematiksel olarak şu şekilde ifade edilebilir:

$$Fi=c\pm/V_i-V_w \qquad (2.1)$$

Burada V_i kromozom i'nin objektif değeri ve c ise uygunluğun negatif çıkmamasını sağlayacak kadar büyük bir sayıdır. Eğer bir maksimizasyon problemiyle karşılaşılırsa denklemde pozitif işaret kabul edilir. Diğer yandan en küçükleme gerekliyse negatif işaret kabul edilir.

$$F=1/(1+\sum_{i=1}^{N}|Rpi-Rdi|) \qquad (2.2)$$

2.4.2.2. Lineer Normalizasyon

Objektif fonksiyonun maksimize veya minimize durumuna göre kromozomlar objektif değerin artma veya azalma düzenine göre sıralanır. En iyi kromozoma rastgele en iyi bir uygunluk f_{best} atanarak sıralanmış düzende diğer kromozomların uyguluğu lineer bir fonksiyonla bulunur.

$$Fi=f_{best}-(i-1).d \qquad (2.3)$$

Burada d eksilme oranıdır. Bu teknik popülasyonun ortalama objektif değerini ortalama uygunluk içerisinde ayrıntılarıyla planlamayı sağlar.

Bu iki teknikten başka kullanıcının kendisinin belirleyeceği başka yöntemler de mevcuttur.

2.4.3. Genetik Operatörler

Pek çok GA operatörü öne sürülmüş olmasına rağmen en yaygın olarak kullanılanları seleksiyon, çaprazlama ve mutasyondur (Qu, 1999).

Eski nesilden, yeni bir kromozom seti (nesil) üretebilmek için seleksiyon, çaprazlama ve mutasyon gibi genetik operatörler uygulanır (Moon v.d., 2002).

3 tip genetik operatör vardır:

2.4.3.1. Seleksiyon (Selection / Reproduction) **:**

Yeniden üretme operatörü, hazır topluluktan uygun olan bireylerin seçilmesi ve bunların sonraki topluluğa kopyalanarak hayatta kalmalarıyla ilgilidir. Seçim modeli, tabiatın hayatta kalabilmek için uygunluk mekanizması modelidir.

Yeniden üretme işleminde, bireyler onların uygunluk fonksiyonlarına göre kopya edilirler. Uygunluk fonksiyonu, mümkün olduğu kadar yükseltilmesi gereken bazı faydalı ve iyi ölçülerdir.

Topluluk uzayındaki her bir bireyin uygunlukları baz alınarak ne kadar sayıda kopyasının olacağına karar verilir. En iyi bireylerden daha fazla kopya alınır, en kötü bireylerden kopya alınmaz. Bu hayatta kalmak için uygunluk stratejisinin GA ya sağladığı avantajdır.

2.4.3.1.1. Uyum Orantılı Seçim

Her seçilimde bir f_i bireyinin seçilme olasılığı $fi / \sum_{j=1}^{\mu} fj$ yani denilebilir ki seçilme olasılığı bireyin mutlak uyum değerinin, tüm popülasyonun mutlak uyum değerine oranıdır. Fakat bu mekanizma hakkında bazı problemler vardır.

- Diğerlerinden çok daha iyi olan bireyler, çok kısa bir sürede popülasyona hakim olmaktadır. Buna **erken yakınsama** denir.
- Uyum değerleri birbirine çok yakın olduğunda neredeyse hiç **seçilim baskısı** olmamaktadır. Bu durumda biraz daha uyum değerinin yüksek olması çok büyük bir yarar sağlamamakta ve seçilim neredeyse düzgün rastsal bir seçilime dönüşmektedir. Burada sadece çok küçük bir farkla daha az uyuma sahip bireyler elenmekte ve çok yavaş bir evrim gerçekleşmektedir.
- Mekanizma aynı uyum fonksiyonunun transpozelerinde farklı bir davranış içine girmektedir.

UOS ile ilgili sonraki iki problemi ortadan kaldırmak için pencereleme yöntemi sıklıkla kullanılır.

2.4.3.1.2. Sıralı Seçim

Sıra tabanlı seçim uyum orantılı seçimin dezavantajlarına yoğunlaşılması üzerine bulunmuş bir diğer metottur. Doğrudan uyum değerlerini kullanarak bireylere seçilim olasılıkları atamak yerine, popülasyonu uyum tabanında sıralayarak ve bireylere sıralarına göre seçilim olasılıkları atayarak sürekli bir seçim baskısı oluşturur. Sıra numarası ile olasılıklar arasındaki haritalama lalettayin olup pek çok farklı şekilde yapılabilir, örneğin lineer ya da eksponansiyel azalan olabilir. Tabi ki popülasyon toplamında olasılıklar toplamı 1 olmalıdır.

2.4.3.1.3. Rulet Çarkı Algoritması

Yukarıda popülasyondaki her birey için üremede seçilme olasılığını veren iki alternatif yöntem açıklanmıştır. İdeal bir dünyada üremede rol alma açısından eşleşme havuzundaki ebeveynlerin buradaki olasılık dağılımına eşit oranlar alması gerekir. Fakat bu pratikte popülasyonun sınırlı büyüklüğü nedeniyle olası değildir. Örneğin bireylerin beklenen kopya sayılarına bakarsak tamsayı olmayan değerler görürüz. Kısacası eşleşme havuzu olasılık dağılımından örneklenmektedir fakat onu tam olarak yansıtamamaktadır.

Bu örneklemeyi en kolay yolla yapmanın yolu Rulet Çarkı Algoritması olarak bilinen yöntemdir. Bu konsept olarak deliklerin büyüklüğünün seçilim olasılıklarını belirttiği tek kollu bir rulet çarkını çevirmek gibidir. Eğer algoritmanın eşleşme havuzundaki m ebeveyn içinden l üyenin seçimi için kullanılacağını varsayarsak genelde aşağıdaki gibi uygulanır. Sıralama ile ya da rastsal olarak 1'den m'ye kadar bir sıra varsayılır. $a_i = \sum_1^i P_{sel}(i)$ formülünden $[a_1, a_2, \ldots, a_m]$ değerleri hesaplanır. Burada $P_{sel}(i)$ seçim dağılımı tarafından tanımlanır – Uyum orantılı ya da sıralama tabanlı. Burada $a_m = 1.0$ vermektedir. Algoritma tablo 2.3'de verilmiştir.

Tablo 2.3. Rulet Çarkı Algoritması

```
BAŞLA
Ayarla Şimdiki Eleman = 1
Olduğu Sürece (Şimdiki Eleman<=m) Yap
    [0,1] aralığından bir düzgün rastsal sayı al;
    Ayarla i=1
    Olduğu Sürece (a_i<r) Yap
    Ayarla i=i+1;
    Yap_Bitir
    Ayarla Eşleşme Havuzu [Şimdiki Eleman] = Ebeveyn [i]
Ayarla Şimdiki Eleman= Şimdiki Eleman+1
Yap_Bitir
SON
```

2.4.3.1.4. Turnuva Seçimi

Bundan önceki metotlarda popülasyon üzerindeki bir bilgiye dayanarak bir olasılık dağılımından örnekleme yapmaya yaramaktaydı. Fakat bazı özel durumlarda, örneğin popülasyon çok büyük ise ya da popülasyon bir şekilde dağıtıksa, bu bilgiyi almak çok fazla zaman alabilir ya da en kötü ihtimalle imkansız olabilir. Bir başka durumda ise bir genel uyum fonksiyonu olmayabilir. Örneğin bir oyun stratejisini düşünelim. Bu durumda verilen stratejilerden hangisinin ne kadar iyi olduğunu söyleyemeyiz fakat bunlardan herhangi ikisi arasında bir oyun simülasyonu yapıp sonuçları karşılaştırabiliriz.

Turnuva seçimi popülasyon hakkında global bir bilgiye ihtiyaç duymayan kullanışlı bir operatördür. Bunun yerine sadece iki bireyi karşılaştıracak bir sıralama bağıntısına dayanmaktadır. Dolayısıyla konsept olarak kurulumu ve uygulanması daha kolay ve hızlıdır.

Turnuva seçimi, sıralama yöntemi gibi mutlak uyuma değil sadece bağıl uyuma baktığından uyum fonksiyonunun transpozesinin alınması durumunda sonuçlar değişecektir. Turnuva seçiminde bir bireyin seçilmesi aşağıdaki dört faktöre dayanır:

- Popülasyondaki sırasına. Fakat bu tüm popülasyonu sıralamadan tahmin edilir.
- Turnuva sayısı k'ya. Çünkü turnuvaya dahil edilen birey sayısı arttıkça turnuvanın sonucunda daha büyük uyuma sahip bireylerin çıkma ihtimali artar, daha az uyuma sahip bireylerin çıkma ihtimali azalır.

- Turnuvanın en uygun bireyinin seçilme ihtimali olan p olasılığı. Bu genelde deterministik sistemlerde p=1 olarak alınır fakat stokastik sistemlerde p<1 olarak kullanılır. İkinci durumda seçim baskısı doğal olarak daha azdır.
- Eğer deterministik ve yer değiştirmeli bir turnuva yapılıyorsa en az uyumlu bireyin seçilme ihtimali yoktur. Fakat yer değiştirmeli bir seçim yapılıyorsa en az uyumlu bireyin bile şanslı bir çekiliş sonrasında seçilebilme ihtimali her zaman vardır.

Tablo 2.4. Turnuva Seçimi Algoritması

```
BAŞLA
Ayarla Şimdiki Eleman = 1;
Olduğu Sürece (Şimdiki Eleman<=m) Yap
        Yer değiştirmeli ya da yer değiştirmesiz k adet rastsal birey seç;
        Bu k adet bireyden en iyisini uyum değerlerini karşılaştırarak seç;
        Bu bireyi i olarak tanımla;
        Ayarla Eşleşme Havuzu [Şimdiki Eleman] = i;
        Ayarla Şimdiki Eleman= Şimdiki Eleman+1;
Yap_Bitir
SON
```

2.4.3.1.5. Hayatta Kalan Seçimi

Hayatta kalan seçimi yer değiştirme olarak da bilinir ve μ adet ebeveynden λ adet yavru oluştuğunda bir sonraki nesil için tekrar μ adet ebeveynden oluşan kümenin oluşturulması sürecinden sorumludur. GA tarihinde pek çok yer değiştirme stratejisi geliştirilmiş ve pek çok alanda kullanılmıştır. Genelde bu stratejiler uygunluğa ya da bireyin yaşına dayanmaları açısından ayrılırlar.

1. Yaş Tabanlı Yer Değiştirme

Bu yöntemde hayatta kalanların seçiminde uyum değerleri hesaba katılmaz. Burada bunun yerine her birey belirli sayıda GA iterasyonunda bulunacak kadar süre popülasyonda bulunur. Bu sayede çok güçlü bir çözüm nedeniyle erken çözüm ihtimali azalır ve diğer bireylerin de en az bir kere seçilimde yer alması ve mutasyon ve rekombinasyon adımlarında hayatta kalabilmesi anlamına gelmektedir.

Bu basit GA'larda kullanılan strateji ise şöyledir. Üretilen yavru sayısı ebeveyn sayısına eşittir ($\mu=\lambda$). Böylece her birey sadece bir çevrimde hayatta kalır ve ebeveynler basitçe yavrular tarafından ıskartaya çıkarılır ve yerleri alınır. Bu strateji kolayca küçülen popülasyonlara da uygulanabilir ($\mu<\lambda$). Bunun gibi her çevrimde birer yavrunun oluştuğu tüm diğer modellere uygulanabilir. Bu strateji basitçe bir İlk Giren İlk Çıkar (FIFO) prensibine sahiptir.

Durgun Durum GA modelleri için bir alternatif yaş tabanlı yer değiştirme yöntemi ise rastgele bir ebeveyni yer değiştirme için seçmektir. Bunun da etkisi aynıdır.

2. Uyum Tabanlı Yer Değiştirme

$\mu+\lambda$ adet ebeveyn ve yavrudan oluşan topluluktan μ adet yeni GA iterasyonlarında kullanılacak ebeveynin seçilmesi için pek çok strateji öne sürülmüştür. Bunlardan bazıları bazı yaş faktörlerini de içerir, örneğin tüm yavruların seçilmesi vb. $\mu>\lambda$ durumunda ise hangi λ'nın iterasyonlar için seçileceğine karar verirler.

A. En Kötüyü Değiştir (GENITOR)

Bu yöntemde, popülasyonun kötü λ üyeleri yer değiştirilmek üzere seçilir. Bu, ortalama popülasyon uyumunda çok ani gelişmelere yol açabileceği gibi, popülasyon hızlıca şu anki en uyumlu üye üzerine odaklanma eğilimi gösterdikçe, erken yakınsamaya (uyuma) da yol açar. Dolayısıyla bu, genellikle büyük popülasyonlarla ve/veya "kopya yok" politikası ile birlikte kullanılır.

B. Seçkincilik (Elitism)

Bu şema, şu an popülasyondaki en uyumlu üyeyi kaybetmeyi önlemek amacıyla genellikle yaş-tabanlı ve stokastik uyum-tabanlı yer değiştirme şemaları ile birlikte kullanılır. Gerçekte, şu an popülasyondaki en uyumlu üyenin izi vardır ve bu daima popülasyonda saklanacaktır. Bu nedenle, bu, eğer grup içinde yer değiştirmek üzere seçilirse ve popülasyona sokulmakta olan hiçbir çocuk eşit veya daha iyi uyuma sahip değilse, o zaman bu saklanır ve çocukların biri atılır.

***2.4.3.2. Çaprazlama (Crossover)* :**

Çaprazlama operatörü iki ebeveynin gen kodlarını yeniden düzenleyerek yavruyu oluşturur. Yavru, her iki ebeveynin yapı taşlarından gelen birer setten oluşur (Xing, 2008).

Amaç, ana (parent) kromozom genlerinin yerini değiştirerek çocuk (child) kromozomlar üretmek ve böylece var olan uygunluk değeri yüksek olan kromozomlardan,uygunluk değeri daha yüksek olan kromozomlar elde etmektir.

Burada önemli olan bir konuda, çaprazlama noktasının çaprazlamadan elde edilecek çocuk kromozomların uygunluk değerleri üzerindeki etkisidir. Bu işlem yapılırken her zaman sonuçlar önceden tahmin edilemez. Bu yüzden gelişigüzel yapılan değişikliklerde sonucun mükemmelliğe doğru gitmesi için belirli kriterler bulmak için çalışılır. Kromozomlardaki genlerin yapısı ve etkileri araştırılarak, bu genlere yapılan müdahalelerle bireye bazı iyi özellikler kazandırılabilir. Çaprazlamadan elde edilecek çocuk kromozomların uygunluk değeri bir önceki ana kromozomlardan daha yüksek olmayabilir.

Tablo 2.5.'de biyolojik çaprazlamaya bir örnek verilmiştir.

Tablo 2.5. Biyolojik çaprazlama örneği

1.Ebeveyn AA BB	2. Ebeveyn aa bb
1. Çocuk AABB	2. Çocuk aabb

Benzer şekilde GA, çaprazlama işlemini uygunluk değerlerine göre seçilmiş iki ebeveyn bireyden, iyi özellikte yeni bireyler elde etmek için kullanır. Çaprazlama rastgele seçilmiş iki çift katarın içindeki alt küme bilgilerin değiştirilmesi işlemdir. Kendi içindeki bilgilerini 1. Pozisyondan itibaren, katarın uzunluğunun bir eksik pozisyonuna kadar, aradaki bilgi kısmen karşılıklı bireyler arasında yer değiştirilir.

Eğer iki bireyin problemin çözümünde bazı etkileri var ise onların bir parçaları faydalı, iyi veya uygun nitelenebilecek bilgi taşımaktadır. Çaprazlama belki problemin çözümünde, bu faydalı bilgileri birleştirerek, daha çok etkili yeni bireyler üretecektir. Tablo 2.6.**'**da ikili kodda verilmiş bir katarda, örnek 2 bitlik bir çaprazlama işlemi verilmiştir.

Tablo 2.6. İki bitlik çaprazlama örneği

1. Ebeveyn	100110|00	1. Çocuk	10011001
2. Ebeveyn	110111|01	2. Çocuk	11011100

Holland'ın 1975'deki çalışmasında kullandığı ikili gösterim uygun olmayan sonuçlara fazla eğimli olduğundan işe yaramamıştır (Xing, 2008).

Çaprazlamadan başka tersinme denilen bir üreme yöntemi daha vardır. Holland bunu tanımlayarak kromozom uzunluğu çok olan bireylerde çaprazlama yerine bunun kullanılmasını performans açısından önermiştir. Tersinme (inversion) bir kromozomu

oluşturan genlerden ardışık bir grubun kendi içerisinde birbirleriyle yer değiştirerek ters dizilmeleridir. Örneğin:011110101 kromozomu(her genin bir bit olduğu varsayımı ile) 5. Ve 8. Gen kromozomları arasında tersindiğinde ortaya 011101011 kromozomu çıkar.

Tersinme genellikle kromozom uzunluğu fazla olan popülasyonlara uygulanır.

2.4.3.2.1. İkili kodlamada çaprazlama

Gen takası (crossover) genetik algoritmanın motoru kabul edilir. Basitçe olay iki ebeveyn kromozomun arasında belirlenen parçaların takasıdır.

Tek noktalı:

11001011+11011**111** = **11001111**

Şekil 2.2. İkili kodlamada tek noktalı çaprazlama

İki noktalı:

11001**011** + 11**0111**11 = **11011111**

Şekil 2.3. İkili kodlamada iki noktalı çaprazlama

Düzenli:

1**10**010**11** + **1**10**111**01 = 11011111

Şekil 2.4. İkili kodlamada düzenli çaprazlama

Aritmetik:

11001011 + 11011111 = 11001001 (AND)

Şekil 2.5. İkili kodlamada aritmetik çaprazlama

2.4.3.2.2. Permütasyon Kodlamada Çaprazlama

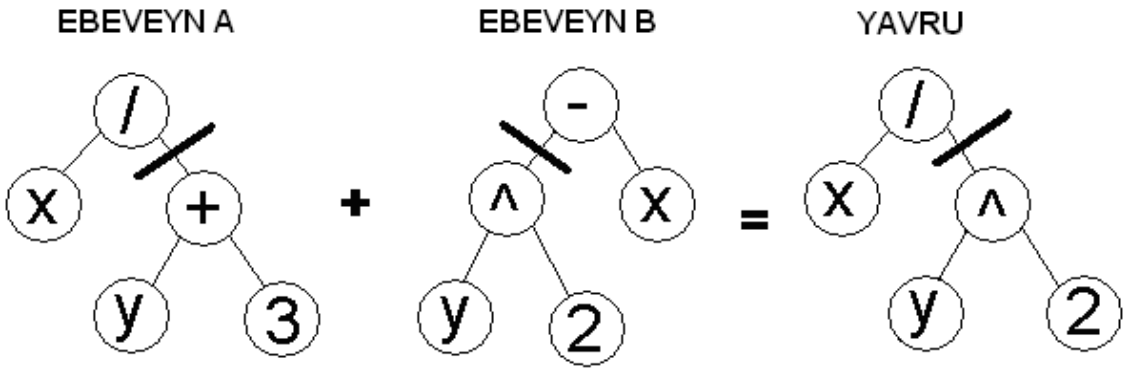
Tek noktalı çaprazlamada bir nokta seçilir ilk ebeveynden bu permütasyonun kopyalandığı noktaya kadar,sonra ikinci ebeveyn okunur ve sayı çocukta hala yoksa o eklenir.

(**1 2 3 4 5** 6 7 8 9) + (**4 5 3 6 8** 9 7 2 1) => (**1 2 3 4 5** 6 8 9 7)

2.4.3.2.3. Değer kodlamada çaprazlama

İkili kodlamadaki tüm çaprazlamalar kullanılabilir.

2.4.3.2.4. Ağaç kodlamada çaprazlama



Şekil 2.6. Ağaç kodlamada çaprazlama

Seçilen düğümler değiştirilir. Operatörler sayılar değiştirilir.

Gen takası toplumda çeşitliliği sağlar. İyi özelliklerin bir araya gelmesini kolaylaştırarak en iyiye yaklaşmayı sağlar. Değiştirme kromozomun bir parçasının dışarıdan değiştirilmesi şeklinde tanımlanır. Değiştirme görünüşte genetik algoritmanın dayanak noktasıdır, ancak etkisi bir çözüm üzerindedir. Bu da yalnız başına başarılı olmasını zorlaştırır. İkilik dizilerde değiştirme rastgele bir bit'in değiştirilmesiyle sağlanabilir. Çok düşük bir değiştirme olasılığı toplumda bazı özelliklerin kaybolmasına neden olabilir. Bu da en iyi sonuçların bulunmasına engeldir. Ancak yüksek bir değiştirme olasılığı da eldeki çözümleri bozarak sonuca ulaşmayı zorlaştırır. Gen takası ve değiştirmenin olasılıkları için kesin bir sayı yoktur. Değiştirme (mutasyon) olasılığı 0.01-0.001, gen takası (crossover) olasılığı 0.5-1.0 aralığında tavsiye edilir.

2.4.3.3. Mutasyon (Mutation) :

Amaç, var olan bir kromozomun genlerinin bir ya da birkaçının yerlerini değiştirerek yeni kromozom oluşturmaktır. Yeniden ve sürekli yeni nesil üretimi sonucunda belirli bir süre sonra nesildeki kromozomlar birbirlerini tekrarlama konumuna gelebilir ve bunun sonucunda farklı kromozom üretimi durabilir veya çok azalabilir. İşte bu nedenle mutasyon bazı yeni oluşturulan çocuklarda zoraki olarak gerçekleştirilir. Böylece çözümlerin lokal optimumlara yakınsaması engellenir (Xing, 2008).

Açıklandığı gibi mutasyonun birinci maksadı bir popülasyonun içindeki değişimi tanımlamaktır. Mutasyon popülasyonlarda çok önemlidir. Öyle ki burada ilk popülasyon mümkün olan tüm alt çözümlerin küçük bir alt kümesi olabilir ve ilk popülasyondaki tüm kromozomların önemli biti sıfır olabilir. Halbuki o bitin problemin çözümü için 1 olması gerekebilir ve bunu da çaprazlama düzeltemeyebilir. Bu durumda o bit için mutasyon kaçınılmazdır. Genellikle önerilen mutasyon oranı 0.005/bit/jenerasyondur. Bu işlem çaprazlamadan sonra gelir. Mutasyonun yapılıp yapılmayacağını bir olasılık testi belirler. Örneğin yeni neslin ortalama uygunluğu ≤ Eski neslin ortalama uygunluğu ise; x. Kromozomun y. Bitini değiştir denilebilir. Bu yeni çocuğu rast gele değiştirir. İkili kodlama için rast gele seçilmiş bitlerden 0'ları 1, 1'leri 0 yaparız.

Tablo 2.7. bit katarında hazırlanmış bir mutasyon operatörünü göstermektedir.

Tablo 2.7. Mutasyon operatörü

Orijinal Çocuk 1	1101111000011110
Orijinal Çocuk 2	1101100100110110
Mutasyonlu Çocuk 1	1100111000011110
Mutasyonlu Çocuk 2	1101101100110100

2.4.3.3.1. İkili Kodlamada Mutasyon:

1**1**001001 => 1**0**001001

Şekil 2.7. İkili kodlamada mutasyon

2.4.3.3.2. Permütasyon Kodlamada Mutasyon

(1 **2** 3 4 5 6 **8** 9 7) => (1 **8** 3 4 5 6 **2** 9 7)

2.4.3.3.3. Değer Kodlamada Mutasyon

Reel değer kodlama için seçilen değere küçük bir sayı eklenir ya da çıkarılır.

(1.29 5.68 **2.86 4.11** 5.55) => (1.29 5.68 **2.73 4.22** 5.55)

2.4.4. . Genetik Algoritmaların Çalışma Prensibi

Genetik algoritmanın çalışmasını prosedürü aşağıdaki gibidir ((Katayama, 2000):

Tablo 2.8. Genetik Algoritma

Prosedür genetik algoritma

01: başla
02: k=0
03: P(k) = başlangıç popülasyonu
04: P(k)'daki yapıları evrimleştir
05: bitirme kriteri <> Evet olduğu sürece Yap
06: başla
07: k=k+1

08: P(k) = P(k-1)

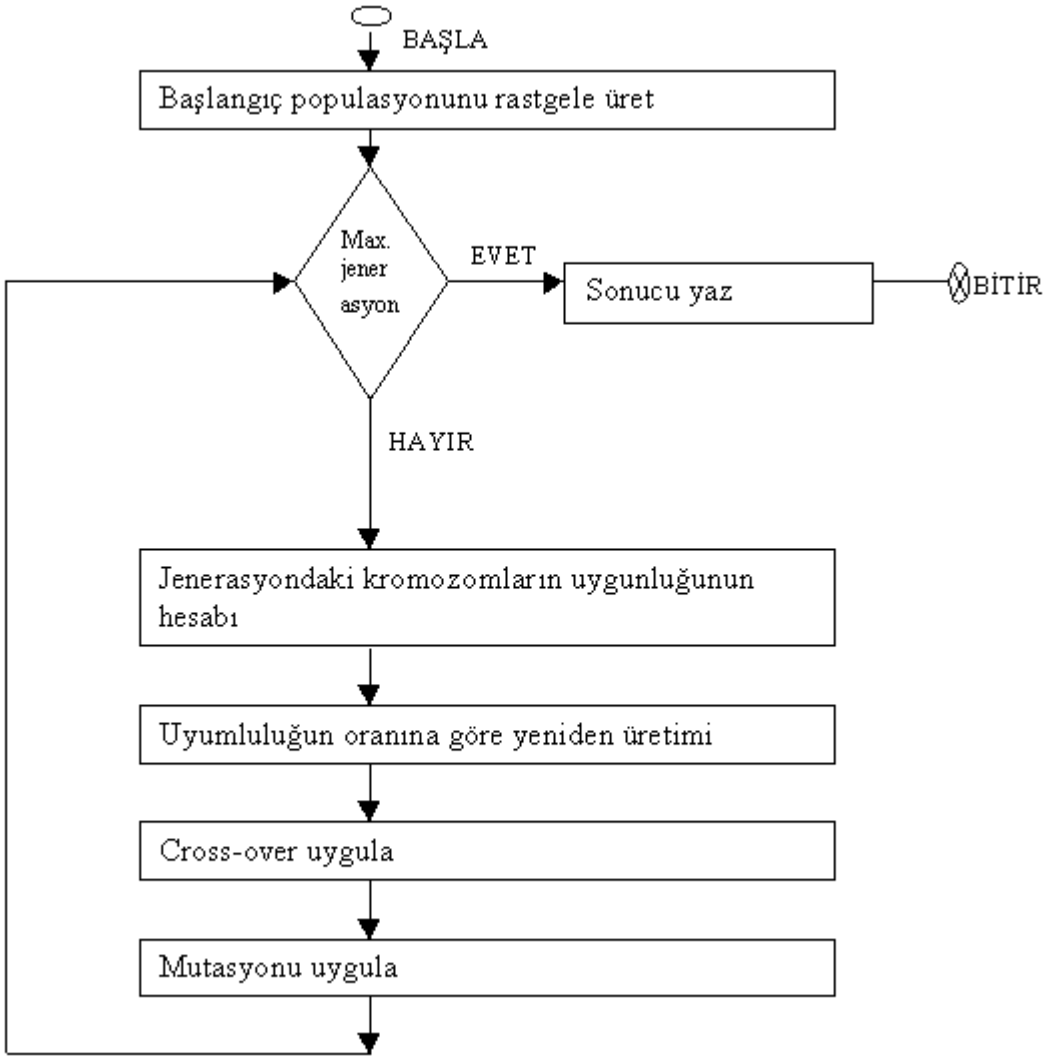
09: P(k)'daki yapıları çaprazlama ile geliştir.

10: P(k)'daki yapıları mutasyonla geliştir.

11: P(k)'daki yapıları değerlendir.

12: Bitir

13: En iyi cevabı geri gönder.



Şekil 2.8. Genetik algoritmanın akış diyagramı

Tablo 2.9. Genetik algoritma adımları

Adım 1	Olası çözümlerin kodlandığı bir çözüm grubu oluşturulur (çözüm grubu, biyolojideki benzerliği nedeniyle, toplum (population), çözümlerin kodları (string) da kromozom olarak adlandırılır).
Adım 2	Her kromozomun ne kadar iyi olduğu bulunur (fitness function).
Adım 3	Bu kromozomlar eşlenerek (mating), yeniden kopyalama (recombination) ve değiştirme (crossover) operatörleri uygulanır. Bu sayede yeni bir toplum oluşturulur.
Adım 4	Yeni kromozomlara yer açmak için eski kromozomlar ortadan kaldırılır.
Adım 5	Tüm kromozomların uygunlukları tekrar hesaplanır.
Adım 6	Eğer jenerasyon süresi dolmamışsa 3. adıma gidilir.
Adım 7	O ana kadar bulunmuş en iyi kromozom sonuçtur.

İşlemleri adım adım açıklamak gerekirse :

Adım-1. İlk aşamada olası çözümlerin kodlandığı bir çözüm grubu oluşturulur. Genellikle genetik algoritma rastsal olarak ilk popülasyonu oluşturur böylece herhangi bir önyargıdan etkilenmeden algoritma çözüm uzayını işlemeye başlayabilir. (Louis, Li, 2000). Bu adıma toplumda bulunacak birey sayısını belirleyerek başlanmaktadır. Kullanılacak sayı için bir standart yoktur. Genel olarak önerilen 100-300 aralığında bir büyüklüktür. Büyüklük seçiminde yapılan işlemlerin karmaşıklığı ve aramanın derinliği önemlidir. Toplum bu işlemden sonra rastgele oluşturulur.Kromozomun temsil ettiği çözüm hakkında bilgiyi herhangi bir yolla içermesi gerekir. En çok kullanılan kodlama ikili kelime katarıdır.

Tablo 2.10. İkili kodlama

Kromozom 1	1101100100110110
Kromozom 2	1101111000011110

Her kromozom bir ikili kelime katarına sahiptir. Bu kelime katarındaki her bit çözümün belli karakteristiğini temsil eder veya tüm kelime katarı bir sayıyı temsil eder. Tabii ki birçok kodlama yolu vardır. Bu daha çok çözülen probleme bağlıdır. Mesela tam sayı veya reel sayı olarak kodlanabilir, bazen de permütasyon kodlamak kullanışlı olabilir.

Adım-2. Kromozomların ne kadar iyi olduğunu bulan fonksiyona uygunluk fonksiyonu denir. Bu fonksiyon işletilerek kromozomların uygunluklarının bulunmasına ise *hesaplama* (evaluation) adı verilir. Bu fonksiyon genetik algoritmanın beynini oluşturmaktadır. Genetik algoritmada probleme özel çalışan tek kısım bu fonksiyondur. Uygunluk fonksiyonu kromozomları problemin parametreleri haline getirerek onların bir bakıma şifresini *çözmektedir* (decoding), sonra bu parametrelere göre hesaplamayı yaparak kromozomların uygunluğunu bulur. Çoğu zaman genetik algoritmanın başarısı bu fonksiyonun verimli ve hassas olmasına bağlı olmaktadır.

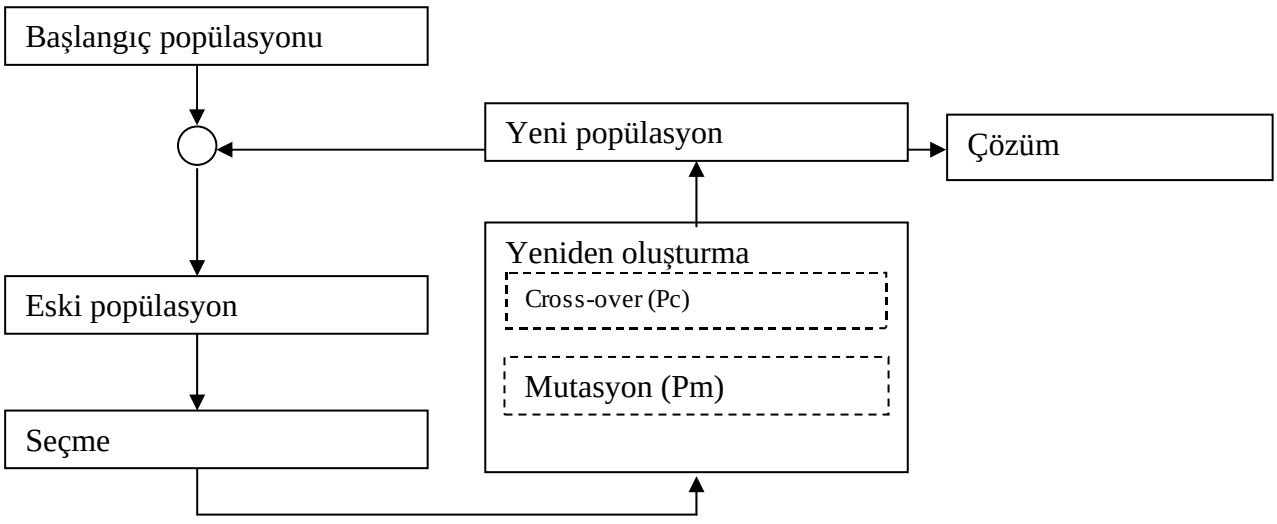
Adım-3. Kromozomların eşlenmesi kromozomların uygunluk değerlerine göre yapılır. Bu seçimi yapmak için *rulet tekerleği seçimi* (roulette wheel selection), *turnuva seçimi* (Tournament Selection) gibi seçme yöntemleri vardır.

Adım-4. Eski kromozomlar çıkartılarak sabit büyüklükte bir toplum sağlanır.

Adım-5. Tüm kromozomlar yeniden hesaplanarak yeni toplumunun başarısı bulunur.

Adım-6. Genetik algoritma defalarca çalıştırılarak çok sayıda toplum oluşturulup hesaplanır.

Adım-7. Toplumların hesaplanması sırasında en iyi bireyler saklandığı için o ana kadar bulunmuş en iyi çözüm çözümdür. Genetik algoritmanın yaptığı işleri temelini akış diyagramı olarak Şekil 2.9.'da görebiliriz.



Şekil 2.9. Genetik algoritmanın temeli

2.4.5. Şema Teorisi (Schemata Theorem)

Genetik algoritmalarda oluşan başarılı bireyler incelenirse, bu bireyler arasındaki benzerlikler bulunabilir. Bu benzerliklerden yola çıkarak şemalar oluşturulabilir. İkilik dizi kodlaması için aşağıdaki yöntem önerilebilir.

0,1 ve # (‘#’ o konumda 0 veya 1 olmasının önemsiz olduğunu gösterir)

Örnek olarak ikinci ve dördüncü bitleri 1, altıncı biti 0 olan çözümlerin başarılı olduğu bir toplumda şu şema oluşturulabilir:

Tablo 2.11. Şema

#1#1#0

Bu şemaya uygun aşağıdaki ikilik diziler yazılabilir:

010100, 010110, 011100, 011110, 110100, 110110, 111100, 111110.

Görüldüğü gibi şemaların katılması ikilik dizilerle gösterilen arama aralığını büyütmektedir. Arama aralığının büyümesinin sonucun bulunmasını zorlaştırması beklenir ancak durum böyle değildir. Seçilim ve yeniden kopyalama ile iyi özellikler daha çok bir araya gelerek daha iyi değerlere sahip şemalara uygun çözümler elde edilir.

Genetik algoritma kendi içinde sanal olarak şemaları oluşturur. Toplumun bireyleri incelenerek bu şemalar ortaya çıkarılabilir. Genetik algoritmalar şemaları oluşturmak için toplum üyelerinin kodları dışında bir bilgi tutmaz. Genetik algoritmaların bu özelliğine *içsel paralellik* (implicit parallelism) denir. Her nesilde, iyiyi belirleyen şemalardaki belirsiz yada önemsiz elemanlar azalır. Böylece genetik algoritmalar sonuca doğru belli kalıplar içinde ilerler.

2.5. GA’nın Performansını Etkileyen Faktörler

- *Kromozom sayısı:* Kromozom sayısını arttırmak çalışma zamanını arttırırken azaltmak da kromozom çeşitliliğini yok eder.
- *Mutasyon Oranı:* Kromozomlar birbirine benzemeye başladığında hala çözüm noktalarının uzağında bulunuyorsa mutasyon işlemi GA’nın sıkıştığı yerden kurtulmak için tek yoludur. Ancak yüksek bir değer vermek GA’yı kararlı bir noktaya ulaşmaktan alıkoyacaktır.

- *Kaç Noktalı Çaprazlama Yapılacağı:* Normal olarak çaprazlama tek noktada gerçekleştirilmekle beraber yapılan araştırmalar bazı problemlerde çok noktalı çaprazlamanın çok yararlı olduğunu göstermiştir.
- *Çaprazlamanın sonucu elde edilen bireylerin nasıl değerlendirileceği:* Elde edilen iki bireyin birden kullanılıp kullanılamayacağı bazen önemli olmaktadır.
- ***N**esillerin birbirinden ayrık olup olmadığı:* Normal olarak her nesil tümüyle bir önceki nesle bağlı olarak yaratılır. Bazı durumlarda yeni nesli eski nesille birlikte yeni neslin o ana kadar elde edilen bireyleri ile yaratmak yararlı olabilir.
- *Parametre kodlanmasının nasıl yapıldığı:* Kodlananın nasıl yapıldığı en önemli noktalardan biridir. Örnek vermek gerekirse kimi zaman bir parametrenin doğrusal yada logaritmik kodlanması GA'nın performansında önemli bir farka yol açabilir.
- *Kodlama gösteriminin nasıl yapıldığı:* Bu da nasıl olduğu yeterince açık olmamakla beraber GA'nın performansını etkileyen bir noktadır. İkilik düzen, kayan nokta aritmetiği ya da gray kodu ile gösterim en yaygın yöntemlerdir.
- *Başarı değerlendirmesinin nasıl yapıldığı:* akıllıca yazılmamış bir değerlendirme işlevi çalışma zamanını uzatabileceği gibi çözüme hiçbir zaman ulaşmamasına neden olabilir.

3. BULANIK MANTIK

3.1. Tanım

“Bilgisayarlar insan beyni gibi çalışmazlar. İnsan beyni, “su sıcak”, “hız yüksek”, “adam genç” gibi objektif, tam ve kesin olmayan yargılar ile çalışabilirken bilgisayarlar bir konuda karar verebilmek için doğru ya da yanlış olduğu birler ve sıfırlar ile ifade edilebilecek kadar indirgenmiş verileri kullanırlar.” (Kosko B. & Isaka S.) Bu durum bilim dünyasında bilgisayarların insanlar gibi düşünüp düşünemeyecekleri tartışmalarının çıkmasına neden olmuştur. Bu tartışmalar Prof Dr. Lotfi Zadeh’in Bulanık Mantık kavramını ortaya atmasına kadar sürmüştür.

“Bulanık mantık ile klasik mantık arasındaki temel fark bilinen anlamda matematiğin sadece aşırı uç değerlerine izin vermesidir. Klasik matematiksel yöntemlerle karmaşık sistemleri modellemek ve kontrol etmek işte bu yüzden zordur, çünkü veriler tam olmalıdır. Bulanık mantık kişiyi bu zorunluluktan kurtarır ve daha niteliksel bir tanımlama olanağı sağlar. Bir kişi için 35,5 yaşında demektense sadece orta yaşlı demek birçok uygulama için yeterli bir veridir. Böylece azımsanamayacak ölçüde bir bilgi indirgenmesi söz konusu olacak ve matematiksel bir tanımlama yerine daha kolay anlaşılabilen niteliksel bir tanımlama yapılabilecektir.” (Altıntaş, 2008)

Bulanık mantıkta bulanık kümeler kadar önemli bir diğer kavram da dilsel (linguistik) değişken kavramıdır. Dilsel değişken “yakın” veya “uzak” gibi kelimeler ve ifadelerle tanımlanabilen değişkenlerdir. Bir dilsel değişkenin değerleri bulanık kümeler ile ifade edilir. Örneğin oda sıcaklığı dilsel değişkeni için “sıcak”, “soğuk” ve “çok sıcak” ifadelerini alabilir. Bu üç ifadenin her biri ayrı ayrı bulanık kümeler ile modellenir (Altıntaş, 2008).

Bulanık mantığın uygulama alanları çok geniştir. Sağladığı en büyük fayda ise insana “özgü tecrübe ile öğrenme” olayının kolayca modellenebilmesi ve belirsiz kavramların bile matematiksel olarak ifade edilebilmesine olanak tanımasıdır. Bu nedenle lineer olmayan sistemlere yaklaşım yapabilmek için özellikle uygundur.

“Bulanık mantık konusunda yapılan araştırmalar Japonya’da oldukça fazladır. Özellikle *“fuzzy process controller”* olarak isimlendirilen özel amaçlı bulanık mantık mikroişlemci çipinin üretilmesine çalışılmaktadır. Bu teknoloji fotoğraf makineleri, çamaşır makineleri, klimalar ve otomatik iletim hatları gibi uygulamalarda kullanılmaktadır (Kosko B. & Isaka S.). Bundan başka uzay araştırmaları ve havacılık endüstrisinde de kullanılmaktadır. TAI’de araştırma gelişme kısmında bulanık mantık konusunda çalışmalar yapılmaktadır. Yine bir başka uygulama olarak otomatik civatalamaların değerlendirilmesinde bulanık mantık

kullanılmaktadır. Bulanık mantık yardımıyla cıvatalama kalitesi belirlenmekte, cıvatalama tekniği alanında bilgili olmayan kişiler açısından konu şeffaf hale getirilmektedir. Burada bir uzmanın değerlendirme sınırlarına erişilmekte ve hatta geçilmektedir.” (Altıntaş, 2008).

Bulanık mantığın ana kavramı bulanık kümelerdir. Buradaki küme kavramının anlaşılması oldukça kolaydır. Örneğin “su ılık” ifadesindeki “ılık” kavramının sınırları kişiden kişiye değişiklik göstermekle birlikte, bu sınırlarda bir kesinlik söz konusu değildir. Fakat genel olarak 18 derece ile 25 derece arası ılık’lık kavramının sınırları olarak düşünülebilir. Bu kavramı grafik olarak ifade etmek istediğimizde karşımıza bir eğri çıkacaktır. Bu eğriye “aitlik eğrisi” adı verilir ve kavram içinde hangi değerin hangi ağırlıkta olduğunu gösterir.

Bir bulanık küme kendi aitlik fonksiyonu ile açık olarak temsil edilebilir. Bir aitlik fonksiyonu 0 ile 1 arasındaki her değeri alabilir. Böyle bir aitlik fonksiyonu ile “kesinlikle ait” veya “kesinlikle ait değil” arasında istenilen incelikte ayarlama yapmak mümkündür.

Bulanık mantığın sistemi şu şekildedir. Bir ifade tamamen yanlış ise klasik mantıkta olduğu gibi 0 değerindedir, eğer tamamen doğru ise 1 değerindedir. Bunların dışında tüm ifadeler 0’dan büyük 1 den küçük reel değerler alırlar. Yani değeri 0.32 olan bir ifadenin anlamı %32 doğru %68 yanlış demektir.

Temelde, bir bulanık mantık denetleyicisi aşağıdaki bileşenlerden oluşur (Lee, 1990).

- Ölçülen değişkenleri değerlendirip, uygun dilsel değişkenlere atamak için bir bulanıklaştırma ara yüzü.
- Dilsel denetim kurallarından ibaret bilgi tabanı.
- İnsan karar verme şekline çok benzer bir şekilde, ölçülen değişkenlere göre bulanık kontrol tepkileri vermek üzere bir karar verme mantığı.
- Anlaşılan dilsel kontrol tepkilerini değerlendirip, sürecin denetlenmesi için bulanık olmayan kontrol girdileri elde eden bir durulaştırma ara yüzü.

3.2. Tarihçe Ve Uygulama Alanları

Bulanık mantık (Fuzzy Logic) kavramı ilk kez 1965 yılında California Berkeley Üniversitesinden Prof. Lotfi A.Zadeh'in bu konu üzerinde ilk makalelerini yayınlamasıyla duyulmuştur. Prof. Lotfi A. Zadeh tarafından ortaya atılan ve hızla gelişerek birçok bilim adamının ilgisini çeken araştırmaya açık yeni bir dal oluşmuştur. Örneğin Londra Üniversitesinden Profesör Mamdani kuramı bir buhar türbininin hızının denetlenmesine uygulamayı düşünmüş ve bu amaçla, bir insanın davranışlarını taklit eden “eğer türbin hızı

çok hızlı artıyorsa ve basınç da çok düşükse, buhar vanasını biraz aç" türünden kurallardan oluşan bir uzman sistem geliştirilmiştir. Prof. Mamdani bulanık mantık temelli bir tür bir uzman sistemle türbin hızının ve performansının çok başarılı bir şekilde denetlenebileceğini göstermiştir.

Bulanık mantık kuramının ilk önemli endüstriyel uygulaması çimento sanayisinde olmuştur. Bu sanayide değirmen içerisindeki sıcaklık ve oksijen oranı ürün kalitesi açısından çok önemlidir. Kısıtlı ve hassas olmayan, ısı ve karbon monoksit oranı gibi bilgilerle iyi bir çalışma düzeni elde edilmesi bir sanat olup operatörlerin bu konuda yeterli bir uzmanlık kazanabilmeleri için inanılmaz farklılıklar olacağından, üretilen çimento da vardiyadan vardiyaya değişecek, tutarlı kalitede çimento üretimi çok zor olacaktır.

İşte bir Danimarka firması bu nedenlerden dolayı lineer bir model üzerine kurulu geleneksel denetleyici yerine bir bulanık mantık denetleyici kullanmayı düşünmüş ve çok başarılı sonuçlar veren bir uzman sistemi geliştirmiştir. Bu veya benzeri sistemler bugün bile Japonya ve Amerika da dahil olmak üzere birçok ülkede kullanılmaktadır.

Kronolojik sıra içerisinde bundan sonraki en önemli aşama Japonya da 1987 yılında görülmüştür. Hitachi firması Sedai metro sisteminde çalışan trenlerin otomatik olarak denetimi için bulanık mantık kullanmıştır (Kosko B. & Isaka S.). Geliştirilen sistemde, daha önce tren operatörü tarafından yapılan ve yolcuların sarsıntılı bir yolculuk geçirmelerine neden olabilen hızlanma raportörünün yapması gereken işler kapıları kapatmak ve başlatma düğmesine basmak gibi birkaç işlemle sınırlı kalmaktadır.

Bu başarılı uygulamadan sonra bulanık denetim konusundaki çalışmalar yeni bir ivme kazanmış ve endüstriyel uygulama alanları hızla artmıştır. Çalışmaların uluslararası alanda koordinasyonu amacı ile Japonya'da 1989 yılında, LIFE adlı bir laboratuar kurulmuştur.

Bulanık mantık denetleyiciler konusundaki kuramsal çalışmaları hala sürüyor olmasına rağmen artı bu konu endüstride kendisine önemli bir yer edinmiş durumdadır. Uygulama alanları arasında çeşitli beyaz eşya, tren, asansör, trafik kontrolü ve otomotiv sanayi sayılabilir. Bugün Japonya'da bulanık denetim kullanan beyaz eşyalar ve elektronik aletler, örneğin fotoğraf ve çamaşır makineleri, güncel yaşamın birer parçasıdırlar.

Günümüzde birçok ülkede bulanık mantık konusunda araştırmalar yapmakta olup bunlar arasında ABD, Japonya, Çin ve Batı Avrupa ülkeleri başta gelmektedir.

NASA bünyesinde bulanık denetim konusunda çalışan çok kuvvetli bir grup vardır. Bu grup uzay mekiği için pilotların yükünü azaltmak, sistemin güvenilirliğini artırmak ve yakıt tüketimini azaltmak amacıyla bir bulanım mantık temelli sistem geliştirmiş ve böylece

konuşlandırma ve o pozisyonda tutma sırasında harcanan yakıt üç misli, yaklaşma sırasında tüketilen yakıt bir buçuk misli azaltılmıştır.

Tablo 3.1. Bulanık Mantık uygulamaları

ÜRÜN	FİRMA	BULANIK MANTIĞIN İŞLEVİ
Asansör Denetimi	Fujitec –Toshiba Mitsubishi Hitachi	Yolcu trafiğini değerlendirir. Böylece bekleme zamanı azalır.
SLR Fotoğraf Makinesi	Sanyo –Fisher Canon Minolta	Ekranda birkaç obje olması durumunda en iyi odaklanmayı ve aydınlatmayı belirler
Video Kayıt Cihazı	Panasonic	Cihazın elle tutulması nedeniyle çekim sırasında oluşan sarsıntıları ortadan kaldırır.
Çamaşır Makinesi	Matsushita	Çamaşırın kirliliğini, ağırlığını, kumaş cinsini sezer, ona göre yıkama programını seçer.
Elektrik Süpürgesi	Matsushita	Yerin durumun ve kirliliğini sezer ve motor gücünü uygun ayarlar.
Su Isıtıcısı	Matsushita	Isıtmayı kullanılan suyun miktar ve sıcaklığına göre ayarlar.
Klima	Mitsubishi	Ortam koşullarını değerlendirerek en iyi çalışma durumunu algılar, odaya birisi girerse soğutmayı arttırır.
ABS Fren Sistemi	Nissan	Tekerleklerin kilitlenmeden frenlenmesini sağlar.
Çelik Endüstrisi	Nippon Steel	Geleneksel denetleyicilerin yerini alır.
Sendai Metro Sistemi	Hitachi	Hızlanma ve yavaşlamayı ayarlayarak rahat bir yolculuk sağlanmasının yanı sıra durma konumunu iyi ayarlar, güçten tasarruf sağlar.
Çimento Sanayi	Mitsubishi Chem	Değirmende ısı ve oksijen oranı denetimi yapar.
Televizyon	Sony	Ekran kontrastını, parlaklığını ve rengini ayarlar
El Bilgisayarı	Sony	El yazısı ile veri ve komut girişine olanak tanır.

3.3. Temel Kavramlar

3.3.1. Bulanık Kümeler ve Üyelik Fonksiyonları

Bulanık mantık, sayıların komşuluğu felsefesine dayanır. Karar sürecinde bir durum bir sayıyla ifade ediliyorsa, söz konusu durumun kabul edilirliği o sayının gerçekleşmesinde sağlanacaktır. Ancak söz konusu sayıya yakın sayılar karar sürecinin bir parçası olarak algılanmayacaktır. Oysa belirli bir güven katsayısında bu sayıların farklı popülasyonların üyeleri olduğunu öne sürmek de istatistiksel açıdan yanlış olacaktır. Örneğin bir tezgahta işlenen bir parçanın sıcaklığının 39 C'ye ulaşması, tezgahın bakım sürecini başlatan bir durumsa belki de sıcaklığın 36 C'ye ulaşması da aynı bakım sürecinin başlaması için bir ön şart olarak kabul edilebilir. Bu durumda aynı temel amaca hizmet eden sayıların komşuluğundan söz etmek mümkündür.

Eğer A $R \in (-\infty,+\infty)$, da, söz konusu kümenin bir elemanı ise $\mu_A(x)$ üyelik fonksiyonu $R \to [0,1]$ aralığında oluşur. Diğer bir deyişle A kümesi $A = [a_1, a_3]$ aralığında ise genel olarak $\mu_A(x)$ üyelik fonksiyonu (3.1) formülüyle gösterilebilir.

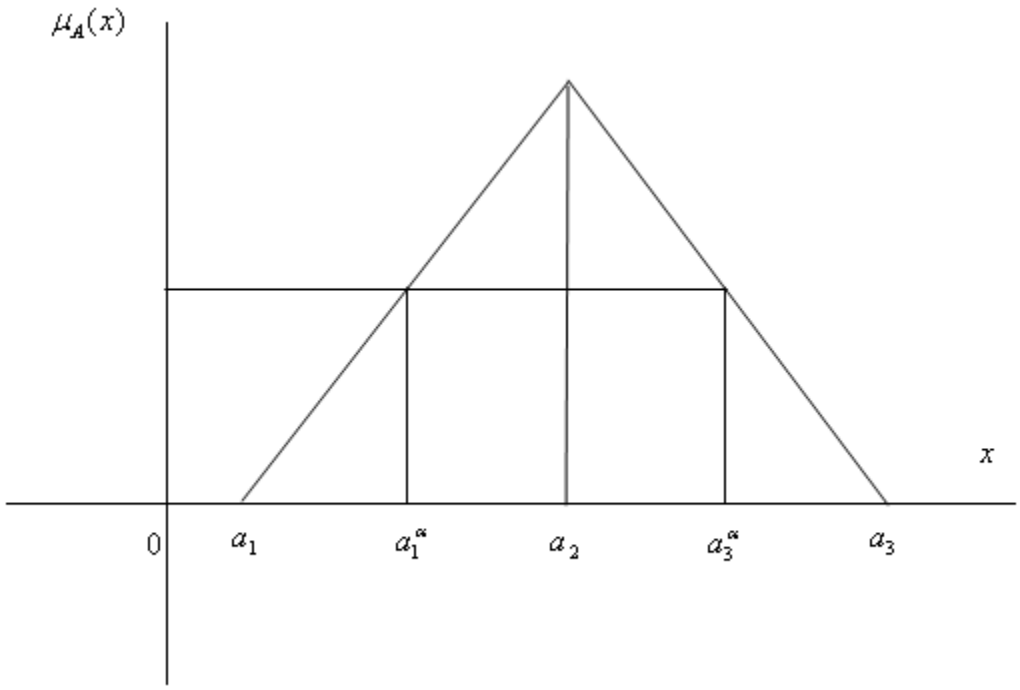
$$\mu_A(x) = \begin{cases} 0, & x < a_1 \\ 1, & a_1 \le x \le a_3 \\ 0 & x > a_3 \end{cases} \qquad (3.1)$$

Üyelik fonksiyonları genellikle, üçgensel üyelik fonksiyonları ve yamuk üyelik fonksiyonları olmak üzere iki başlık altında incelenmektedir.

$\mu_A(x)$ üçgensel üyelik fonksiyonu, (3.2) formülünde tanımlanmıştır (Triantaphyllou, 2000).

$$\mu_A(x) = \begin{cases} 0, & x < a_1 \\ \dfrac{x - a_1}{a_2 - a_1}, & a_1 \le x \le a_2 \\ \dfrac{a_3 - x}{a_3 - a_2}, & a_2 \le x \le a_3 \\ 0, & x > a_3 \end{cases} \qquad (3.2)$$

(3.2) formülüne göre küme, $A=(a_1,a_2,a_3)$ olmalıdır. Burada a_2 normal değerli üyelik olarak tanımlanabilir. Bulanık Mantık bu noktada bir α katsayısına bağlı olarak a_2' ye yakın değerlerin, bu değere yüklenen anlam ile temsil edileceğini varsaymaktadır. Diğer bir deyişle a_2' deki belirsizlik, varsayılacak ya da dağılıma göre bulunabilecek bir α katsayısı ile tolere edilebilir. Söz konusu komşuluk Şekil 3.1' de gösterilmiştir (Triantaphyllou, 2000).



Şekil 3.1 Sayıların Komşuluğu

α değeri bulanık mantık terminolojisinde kesim katsayısı olarak adlandırılır. a_1^α ve a_3^α sayıları ise a_2 normal değerinin komşuluğunu oluşturan aralığın alt ve üst sınır değerleridir. Diğer bir deyişle a_1^α ve a_3^α aralığındaki tüm sayılar a_2 normal değeri ile aynı anlama sahiptir. a_1^α ve a_3^α değerleri (3.3) ve (3.4) formülleri yardımıyla bulunabilir (Terano, 1997).

$$\frac{a_1^\alpha - a_1}{a_2 - a_1} = \alpha \qquad (3.3)$$

$$\frac{a_3 - a_3^\alpha}{a_3 - a_2} = \alpha \qquad (3.4)$$

(3.3) ve (3.4) formüllerinden $\forall \alpha \in [0,1]$ için $A_\alpha = [a_1^\alpha, a_3^\alpha]$ aralığı oluşturulabilir. a_1^α ve a_3^α değerleri (3.5) ve (3.6) formüllerinde gösterilmiştir.

$$a_1^\alpha = \alpha(a_2 - a_1) + a_1 \tag{3.5}$$

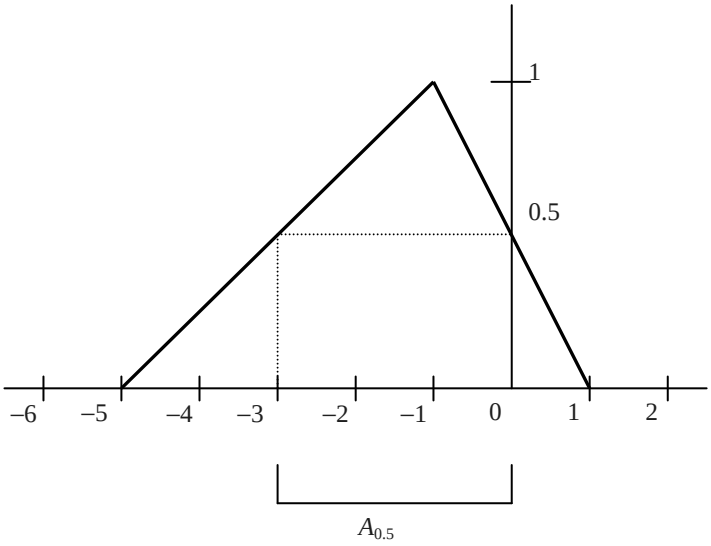
$$a_3^\alpha = a_3 - (a_3 - a_2)\alpha \tag{3.6}$$

Örneğin üçgensel bulanık mantık sayılarına ilişkin küme $A = (-5,-1,1)$ ise bu durumda (3.2) formülünden üyelik fonksiyonu,

$$\mu_A(x) = \begin{cases} 0, & x < -5 \\ \dfrac{x+5}{4}, & -5 \le x \le -1 \\ \dfrac{1-x}{2}, & -1 \le x \le 1 \\ 0, & x > 1 \end{cases}$$

olarak bulunur. Eğer karar verici α kesim katsayısını 0,5 olarak saptamışsa -1 normal değerinin komşuları (3.5) ve (3.6) formüllerinden $a_1^{0,5} = -3$ ve $a_3^{0,5} = 0$ olarak bulunacaktır. Diğer bir deyişle -1 normal değeri ile aynı anlam düzeyinde bulunan sayılar kümesi $[-3,0]$ aralığıdır. Söz konusu ilişki Şekil 3.2' de gösterilmiştir.

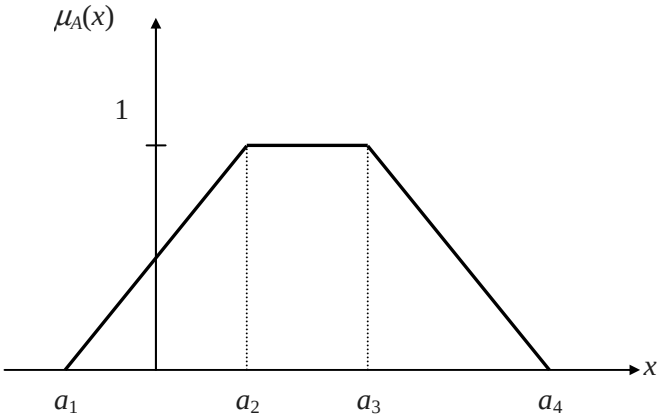
Eğer bulanık mantık sayılarına ilişkin kümede normal kabul edilen iki değer varsa diğer bir deyişle küme, $A = (a_1, a_2, a_3, a_4)$ şeklinde 4 belirleyici değerden oluşuyorsa bu durumda üyelik fonksiyonu yamuk üyelik fonksiyonu tipinde oluşacaktır. Yamuk üyelik fonksiyonu (3.7) formülünde gösterilmiştir.



Şekil 3.2 $A=(-5,-1,1)$ Kümesinin Komşuluğu

$$\mu_A(x)=\begin{cases}0, & x<a_1\\ \dfrac{x-a_1}{a_2-a_1}, & a_1\le x\le a_2\\ 1, & a_2\le x\le a_3\\ \dfrac{a_4-x}{a_4-a_3}, & a_3\le x\le a_4\\ 0, & x>a_4\end{cases} \tag{3.7}$$

Söz konusu komşuluk Şekil 3.3’ deki gibi oluşacaktır.



Şekil 3.3 Yamuk Sayı Komşuluğu

3.3.2. Bulanık Mantığın Avantaj ve Dezavantajları

Bulanık mantıktan yola çıkılarak kullanılan bulanık denetleyicilerle ilgili başlıca üstünlükler, zayıf noktalar ve eleştiriler aşağıda açıklanmıştır.

3.3.2.1. Bulanık Mantığın Avantajları

- Günlük hayatta olduğu gibi belirsiz, zamanla değişen, karmaşık, iyi tanımlanmamış sistemlerin denetimine basit çözümler getirir.
- Sistem basit bir matematiksel modelle tanımlanabilen bir sistemse o zaman geleneksel bir denetim yeterli olacaktır. Ama karmaşık bir sisteme geleneksel bir mantık uygulamak hem çok zor hem de yüksek maliyetlidir. Buna karşılık bulanık mantık denetimi geleneksel mantığa göre sistemi daha iyi analiz edebileceği gibi aynı zamanda da ekonomiktir.
- Bulanık mantıkta işaretlerin bir ön işleme tabi tutulmaları ve oldukça geniş bir alana yayılan değerlerin az sayıda üyelik fonksiyonlarına indirgenmeleri nedeni ile bulanık denetim genellikle daha küçük bir yazılımla daha hızlı bir şekilde sonuçlanır.
- Söz edilen az sayıda değerler üzerinde uygulanacak kural sayısı da az olduğundan sonuca ulaşmak daha da çabuklaşacaktır.
- Bu durum geleneksel bilgisayar ortamında böyledir.Özel geliştirilmiş bir donanımla sonuca daha da hızlı ulaşmak olasıdır. Örneğin Sanyo-Fisher firması mühendisleri, video kayıt cihazında kullanmayı düşündükleri mikro bilgisayarın yetersiz kalmasından dolayı, bulanık denetim kullanmaya karar vermişlerdir. Bulanık denetim yazılım boyutlarının daha küçük olmasını sağladığından, dış bellek kullanımına gerek kalmamıştır.
- Bulanık mantık denetiminin sağladığı bir diğer avantaj ise doğrudan kullanıcı girişlerine ve kullanıcının deneyimlerinden yararlanabilmesine olanak sağlamasıdır.
- Bilindiği gibi otomatik vites değişimi motorun belli hızlara ulaşması sonucunda otomatik olarak gerçekleşir. Buna karşılık manüel vitesli bir arabada ise sürücü, yol, yük ve kendi araba kullanış tarzına göre belli durumlarda vites değiştirir. Subaru tarafından üretilen Justy tipi otomobilde kullanılan aktarım organının değiştirilmesi, bir kayışın konumunun bulanık mantık kullanılarak değiştirilmesi ile sağlanır. Böylece arabanın ivmesi ve performansı sürekli olarak ayarlanır hale gelir. Subaru, bu otomobilde kullandığı bulanık mantık üyelik fonksiyonlarını, otomobili test şoförlerine kullandırarak ve onlardan ivme ve performans açısından en iyi aktarım

oranını öğrenerek ayarlamıştır. Bu konuda Honda ve Nissan da benzer çalışmalar yapmışlardır.

3.3.2.2. Bulanık Mantığa Yöneltilen Eleştiriler

Bulanık denetleyicilere yönelik çeşitli eleştiriler de getirilmiştir. Bunlardan birkaçı aşağıda sıralanmıştır:

- Bulanık mantık denetleyicilerinin süreç hakkında daha fazla bilgiye ve algılayıcıya ihtiyaç duyması, dolayısıyla hem pahalı hem de daha az güvenilir olması,
  Bu her zaman doğru değildir. Örnek vermek gerekirse Mitsubishi tarafından üretilen klimada, geleneksel denetleyiciye göre daha az algılayıcı kullanılmıştır.
- Bulanık mantık denetleyicilerinin geleneksel denetleyicilere kıyasla gösterdiği yüksek performans doğrusal olmayan denetleyici aracılığı ile de sağlanabilir:
  Bu doğru olabilir ama büyük bir ihtimalle doğrusal olmayan denetleyici, bulanık denetleyicide olduğu gibi daha küçük kapasiteli bir işlemci ile gerçekleşemeyecektir.

3.3.2.3. Bulanık Mantığın Dezavantajları

- Bulanık denetimde kullanılan kurallar deneyime çok bağlıdır.
- Üyelik fonksiyonlarının seçiminde belirli bir yöntem yoktur.En uygun fonksiyon deneme ile bulunur. Bu da oldukça uzun bir zaman alabilir.
- Denetlenen sistemin bir kararlılık analizi yapılamaz ve sistemin nasıl cevap vereceği önceden kestirilemez. Yapılacak tek şey benzetim çalışmasıdır.

4. BULANIK GENETİK ALGORİTMALAR

Klasik denetim sistemlerindekinin aksine, sistemlerin matematiksel modeline gerek duymadan, sadece istenilen çıkışı verecek şekilde girişe uygulanan işaret ayarlandığından, bulanık denetimin işlemesi tıpkı usta bir insanın o sistemi denetlemesine benzer. Yani bulanık mantık ve bulanık küme işlemleri kullanılarak makinelerin insanlar gibi kararlar vermesi sağlanabilmektedir. Bulanık mantığın bu uyumluluğunun yapay sinir ağları veya genetik algoritmalarla desteklenmesi sonucu *nöral-bulanık (İ*ngilizce literatürde bu konu *artificial neural networks and fuzzy logic*, *neuro-fuzzy*, ve *neural fuzzy* terimlerinden birisi ile ifade edilmektedir*) sistemler*, veya *genetik-bulanık sistemler* ortaya çıkmıştır. Böylece akıllı (*intelligent*) sistemler de hızlı bir gelişme kaydetmeye başlamıştır (Liu, 1997)

Bulanık genetik algoritmalar, temelde bulanık mantık ve genetik algortmaların beraber kullanılmasını içerir. Bu iki yöntemin birlikte kullanılmasını amaçlayan pek çok uygulama ve makale yayınlanmıştır (Herrera & Lozano, 1999).

Bulanık genetik algoritma terimi, Bulanık Algoritma teriminden türemiştir. Aslında bulanık genetik algoritma kısaca bulanık kümeler ile işlem yapan (bilgisayar programına benzeyen) bir komut dizisidir.

Bir algoritmanın bulanık kümeler ile işlem yapmasına izin verildiğinde çok karmaşık bir durumun yaklaşık değerlerle kolayca ifade edilebilmesi olanaklı hale gelmektedir.

Genetik algoritmalar ise daha önceki bölümlerde anlatıldığı gibi çözüm yöntemi bilinmeyen ya da çok karmaşık olan problemlerin arama uzayında belirli bir stratejiye (uyum fonksiyonu) göre aranarak çok daha hızlı ve kolay bir şekilde bulunabilmesini sağlamaktadır.

Bu iki yöntemin beraber kullanılması hem Genetik Algoritmaların problemin tam çözüm metodunun bilinmesine ihtiyaç duymaması özelliği, hem de Bulanık Mantığın problemin aşırı net değerlerle ifade edilmesine ihtiyaç duymaması özelliği nedeniyle oldukça büyük bir bilgi indirgenmesine yol açmaktadır. Böylece oldukça kompleks problemler bu metotla daha kolay modellenmekte ve daha hızlı çözülebilmektedir.

Bulanık mantık ve genetik algoritmaları beraber kullanmanın iki olası yöntemi vardır. Bunlardan birincisi genetik algoritmaların bulanık sistemlerle ilgili arama ve optimizasyon problemlerinin çözülmesinde kullanılmasıdır. Diğeri ise bulanık araçların ve bulanık mantık temelli tekniklerin farklı genetik bileşenlerin modellenmesinde ve genetik algoritma kontrol parametrelerinin performası artırma amacıyla uyarlanması ile olur (Herrera & Lozano, 1999).

Birinci yöntem bulanık problemlerin çözülmesinde genetik algoritma kullamılmasını içerdiğinden aslında sadece problemin bulanık olmasının doğal bir sonucudur. Bu nedenle bu sistemlere literatürde genetik algoritma destekli bulanık sistemler de denir.

İkinci yöntem ise, bu çalışmada da üzerinde durulacak olan, genetik algoritmanın bulanık teknikleri kullanarak daha gelişmiş bir genetik algoritma haline getirilmesine dayanır. Herrera ve Lozano'ya göre bunu yapmanın birkaç olası yöntemi vardır. Bunlar:

- GA kontrol parametrelerinin uyarlanması
- Operatörlerin uyarlanması
- Gösterim şeklinin uyarlanması ve
- Algoritma durdurma kriterinin uyarlanmasıdır (1999).

Genetik algoritma kontrol parametrelerinin uyarlanması temelde genetik algoritma içinde algoritmanın çözüme yönlendirilmesini sağlayan uyum fonksiyonunun bulanık olarak tanımlanmasını ve bu fonksiyon içinde bulanık terimler kullanılmasını içerir.

Operatörlerin uyarlanması ise evrimsel operatörlerin bulanık mantık tekniği ile çalışacak şekilde tanımlanmasını içermektedir. Örneğin bulanık olarak tanımlanmış bir çaprazlama operatörü içeren bir genetik algoritma bulanık genetik bir uygulama olacaktır. Literatürde bu yöntemin "Bulanık bağlantılar ile çaprazlama", "Bulanık şablonlar kullanarak bulanık çaprazlama", "Bulanık mutasyon operatörleri" gibi örneklerine rastlanmaktadır.

Gösterim şeklinin uyarlanması ise genetik algoritmanın ilk aşaması olan gösterimin yani problemi tanımlayan parametrelerin ve değişkenlerin genler ve kromozomlar halinde tanımlanmasının bulanık terimler kullanılarak gerçekleştirilmesidir. Genetik algoritmalarda genellikle bir genotip bir fenotipe eşleştirilmiştir. Fakat doğada bu durum böyle değildir. Bir genotip çevre etkenlerinin de etkisiyle birden fazla fenotipe eşdeğer olabilir. Genlerin gösterim şeklinin bulanıklaştırılması ile bu işleyiş genetik algoritmanın mantığına da yansıtılabilir.

Durdurma kriterinin uyarlanması ise temelde bulanık terimlerin algoritma durdurma kriteri olarak kullanılması ile ilgilidir. Bu durumda algoritma önceden tanımlanmış bir jenerasyon parametresine bağlı olarak bir jenerasyon sayısına ulaşıncaya kadar değil, algoritmada bulanık olarak tanımlanmış bir jenerasyon sayısını sağlayıncaya kadar çalışmaya devam eder. Bu kontrol aynı zamanda erken yakınsama (premature convergence) ihtimalini azaltmak için de kullanılabilir.

Bu çalışmada "Genetik algoritma kontrol parametrelerinin uyarlanması" alternatifi kullanılacaktır. Buradaki çalışmada bulanık mantık, genetik algoritmanın uyum fonksiyonunda devreye girmektedir. Burada kullanılacak olan genetik algoritmanın uyum fonksiyonu bulanık bir önem derecesine dayanan bir maliyet fonksiyonudur. Bu nedenle kullanılan genetik algoritma bir bulanık genetik algoritmadır.

5. GEZGİN SATICI PROBLEMİ TABANLI BİR SİSTEMİN DİNAMİK BULANIK GENETİK ALGORİTMA İLE OPTİMİZASYONU

5.1. Giriş

Gezgin satıcı problemi geleneksel yöntemlerle çözülemeyecek klasik bir optimizasyon problemidir. Amaç satıcının N şehri en kısa yolu kullanarak gezmesidir. Burada bir gezgin satıcıdan bahsedilse de bu problem mutlaka bir geziyi ifade etmek durumunda değildir. Bu problem tipi pek çok farklı problemin modellenmesinde kullanılabilir. Uygulamalar gaz borularının optimal döşenmesi, anten besleme sisteminin tasarımı, VLSI devresinde transistörlerin konfigürasyonu gibi mühendislik problemleri de dahil olmak üzere, birçok formda ortaya çıkabilir.

Klasik gezgin satıcı probleminin en basit halinin maliyet fonksiyonu aşağıdaki gibi ifade edebilir.

$$maliyet = \sum_{n=0}^{N} \sqrt{(x_n - x_{n+1})^2 + (y_n - y_{n+1})^2} \qquad (5.1)$$

Burada,

x_n: n. şehrin x koordinat düzlemindeki yerini,

y_n: n. şehrin y koordinat düzlemindeki yerini ifade etmektedir.

Bizim bu çalışmada ele aldığımız kargo problemi bir depo ve birden çok müşteriyi içeren kuramsal bir sistemde tanımlanmaktadır. Müşteriler ağırlıkları önceden belirlenmiş kriterlere verilen dilsel değerler ile oluşan bulanık önem dereceleri ile ifade edilen bir önceliklendirme işlemine tabi tutularak, bu işlem sonucundaki önceliklerine göre sıralanmaktadırlar. Problem bu haliyle öncelik kısıtlı gezgin satıcı problemi olarak modellenecektir. Daha sonra bir bulanık genetik algoritma kullanılarak problemin optimizasyonu sağlanacaktır.

5.2. Problemin Tanımlanması

Bu çalışmada bir kargo deposu, bir kargo aracı ve 20 müşteriden oluşan, örnek olarak kullanılacak, örnek bir sistem oluşturulmuştur. Bu sistemin amacı önceden belirlenen beş kriterle tasvir edilen birer önem derecesine sahip müşteriler için, bu önem dereceleri ve yol

uzunluğuna bağlı olarak değişen taşıma maliyetini de göz önünde bulunduran bir optimizasyon sistemi geliştirmektir.

Önem derecesinin belirlenmesinde kullanılacak parametreler aşağıda verilmiştir.

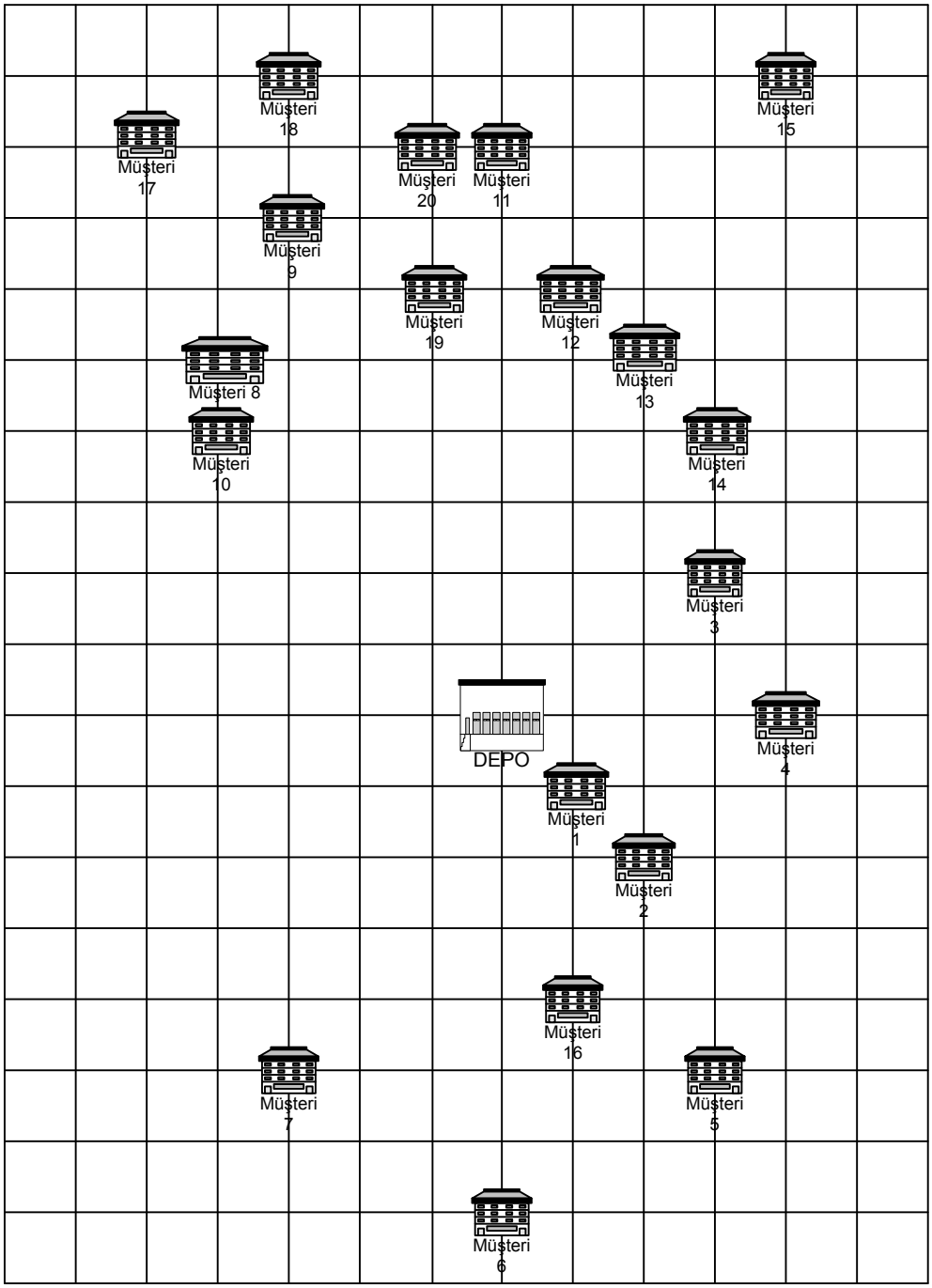
- Müşteri Cirosu: Müşterinin kargo şirketi ile yaptığı ciro'yu (Çalışma sıklığının parasal olarak ifadesi) ifade eder.
- Müşteri potansiyel Cirosu: Müşterinin kargo şirketi ile yapabileceği toplam ciro'yu ifade eder.
- Çalışma süresi: Müşterinin kargo şirketi ile çalışma süresini ifade eder.
- Yük değeri: Yükün maddi değeri ve müşteri açısından önem miktarını ifade eder.
- Yük miktarı: Yükün hacmi, ağırlığı vb. yük miktarını ifade eder.

Bu parametrelere göre müşterilerin değerlendirilmesi ile aşağıdaki tablo oluşmuştur.

Tablo 5.1. Müşterilerin Önem Değerlendirmeleri

	Müşteri Cirosu	**Müşteri Potansiyel Cirosu**	**Çalışma Süresi:**	**Yük Değeri:**	**Yük Miktarı:**
Müşteri 1	Çok Önemli	Çok Önemli	Çok Önemli	Çok Önemli	Çok Önemli
Müşteri 2	Çok Önemli	Orta	Çok Önemli	Az Önemli	Az Önemli
Müşteri 3	Orta	Çok Önemli	Orta	Orta	Önemli
Müşteri 4	Önemli	Çok Önemli	Önemli	Önemsiz	Önemsiz
Müşteri 5	Önemli	Orta	Çok Önemli	Önemsiz	Önemsiz
Müşteri 6	Önemli	Önemli	Çok Önemli	Orta	Önemli
Müşteri 7	Önemli	Orta	Orta	Orta	Az Önemli
Müşteri 8	Orta	Orta	Orta	Orta	Orta
Müşteri 9	Önemli	Orta	Orta	Önemsiz	Önemsiz
Müşteri 10	Önemli	Çok Önemli	Orta	Orta	Orta
Müşteri 11	Önemli	Çok Önemli	Orta	Orta	Orta
Müşteri 12	Orta	Orta	Orta	Önemsiz	Önemsiz
Müşteri 13	Çok Önemli	Önemsiz	Önemsiz	Önemsiz	Önemsiz
Müşteri 14	Orta	Orta	Çok Önemli	Çok Önemli	Çok Önemli
Müşteri 15	Önemsiz	Önemsiz	Önemsiz	Önemsiz	Önemsiz
Müşteri 16	Önemli	Önemli	Önemli	Önemli	Önemli
Müşteri 17	Önemsiz	Çok Önemli	Önemsiz	Önemsiz	Önemsiz
Müşteri 18	Önemsiz	Önemsiz	Önemsiz	Çok Önemli	Az Önemli
Müşteri 19	Önemsiz	Önemsiz	Önemsiz	Önemsiz	Önemsiz
Müşteri 20	Orta	Orta	Orta	Orta	Orta

Depo ve müşterilerin aşağıdaki harita üzerinde gösterildiği şekilde yerleştiği varsayılmaktadır.



Şekil 5.1. Müşterilerin ve deponun harita üzerinde yerleşimi

Bu harita üzerinde aracın sadece kenarları özdeş ve 1 birim olan karelerin kenarları üzerinde hareket edebildiği ve tüm hedeflerin karelerin kesişme noktalarında oldukları varsayılmaktadır.

Hedeflerin birbirlerine olan uzaklıkları aşağıdaki tabloda verilmiştir.

Tablo 5.2. Uzaklıklar matrisi

	X	Müşteri 1	Müşteri 2	Müşteri 3	Müşteri 4	Müşteri 5	Müşteri 6	Müşteri 7	Müşteri 8	Müşteri 9	Müşteri 10	Müşteri 11	Müşteri 12	Müşteri 13	Müşteri 14	Müşteri 15	Müşteri 16	Müşteri 17	Müşteri 18	Müşteri 19	Müşteri 20
X	0	2	4	5	4	8	7	8	9	10	8	8	7	7	7	13	5	13	12	7	9
Müşteri 1	2	0	2	5	4	6	7	8	11	12	10	10	7	7	7	13	3	15	14	9	11
Müşteri 2	4	2	0	5	4	4	7	8	13	14	12	12	9	7	7	13	3	17	16	11	13
Müşteri 3	5	5	5	0	3	7	12	13	10	11	9	9	6	4	2	8	8	15	13	8	10
Müşteri 4	4	4	4	3	0	6	11	12	13	14	12	12	9	7	5	9	7	17	16	11	13
Müşteri 5	8	6	4	7	6	0	5	6	17	18	16	16	13	11	9	15	3	21	20	15	17
Müşteri 6	7	7	7	12	11	5	0	5	16	17	15	15	14	14	14	20	4	20	19	14	16
Müşteri 7	8	8	8	13	12	6	5	0	11	12	10	16	15	15	15	21	5	15	14	13	15
Müşteri 8	9	11	13	10	13	17	16	11	0	3	1	7	6	6	8	12	14	4	5	4	6
Müşteri 9	10	12	14	11	14	18	17	12	3	0	4	4	5	7	9	9	15	3	2	3	3
Müşteri 10	8	10	12	9	12	16	15	10	1	4	0	8	7	7	7	13	13	5	6	5	7
Müşteri 11	8	10	12	9	12	16	15	16	7	4	8	0	3	5	7	5	13	5	4	3	1
Müşteri 12	7	7	9	6	9	13	14	15	6	5	7	3	0	2	4	6	10	8	7	2	4
Müşteri 13	7	7	7	4	7	11	14	15	6	7	7	5	2	0	2	6	10	10	9	4	6
Müşteri 14	7	7	7	2	5	9	14	15	8	9	7	7	4	2	0	6	10	12	11	6	8
Müşteri 15	13	13	13	8	9	15	20	21	12	9	13	5	6	6	6	0	16	10	7	8	6
Müşteri 16	5	3	3	8	7	3	4	5	14	15	13	13	10	10	10	16	0	18	17	12	14
Müşteri 17	13	15	17	15	17	21	20	15	4	3	5	5	8	10	12	10	18	0	3	6	4
Müşteri 18	12	14	16	13	16	20	19	14	5	2	6	4	7	9	11	7	17	3	0	5	3
Müşteri 19	7	9	11	8	11	15	14	13	4	3	5	3	2	4	6	8	12	6	5	0	2
Müşteri 20	9	11	13	10	13	17	16	15	6	3	7	1	4	6	8	6	14	4	3	2	0

İlk aşamada önceden belirlenen parametreler kendi aralarında Bulanık (Fuzzy) AHP yöntemi ile karşılaştırılmış ve her parametre için bir bulanık ağırlık belirlenmiştir. Bu ön çalışmada lojistik sektöründe uzun yıllardır hizmet veren uzmanların görüşlerine başvurulmuştur.

Ağırlıklandırma işlemi uzmanların yaptıkları karşılaştırmaya dayanarak aşağıdaki üçgen bulanık sayılara göre yapılmıştır.

Tablo 5.3. Bulanık sayılar ve dilsel karşılıkları

(7/2,4,9/2)	**Mutlak / Kesin**
(5/2,3,7/2)	**Çok Güçlü**
(3/2,2,5/2)	**Biraz Güçlü**
(2/3,1,3/2)	**Zayıf**
(1,1,1)	**Eşit/Denk**

Uzmanların her bir parametre için yaptıkları karşılaştırmalar aşağıdaki tabloda özetlenmiştir.

Tablo 5.4 Parametre önem ağırlıkları hakkında uzman değerlendirmeleri

Uzman 1	Müşteri Cirosu	Müşteri Potansiyel Cirosu	Çalışma Süresi:	Yük Değeri:	Yük Miktarı:
Müşteri Cirosu:	(1,1,1)	(3/2,2,5/2)	(2/3,1,3/2)	(2/3,1,3/2)	(3/2,2,5/2)
Müşteri Potansiyel Cirosu:	(3/2,2,5/2)	(1,1,1)	(5/2,3,7/2)	(2/3,1,3/2)	(5/2,3,7/2)
Çalışma Süresi:	(5/2,3,7/2)	(2/3,1,3/2)	(1,1,1)	(3/2,2,5/2)	(2/3,1,3/2)
Yük Değeri:	(7/2,4,9/2)	(5/2,3,7/2)	(3/2,2,5/2)	(1,1,1)	(3/2,2,5/2)
Yük Miktarı:	(3/2,2,5/2)	(2/3,1,3/2)	(5/2,3,7/2)	(3/2,2,5/2)	(1,1,1)

Uzman 2	Müşteri Cirosu	Müşteri Potansiyel Cirosu	Çalışma Süresi:	Yük Değeri:	Yük Miktarı:
Müşteri Cirosu:	(1,1,1)	(5/2,3,7/2)	(5/2,3,7/2)	(3/2,2,5/2)	(5/2,3,7/2)
Müşteri Potansiyel Cirosu:	(2/3,1,3/2)	(1,1,1)	(3/2,2,5/2)	(5/2,3,7/2)	(3/2,2,5/2)
Çalışma Süresi:	(2/3,1,3/2)	(3/2,2,5/2)	(1,1,1)	(3/2,2,5/2)	(2/3,1,3/2)
Yük Değeri:	(3/2,2,5/2)	(2/3,1,3/2)	(3/2,2,5/2)	(1,1,1)	(3/2,2,5/2)
Yük Miktarı:	(2/3,1,3/2)	(3/2,2,5/2)	(5/2,3,7/2)	(3/2,2,5/2)	(1,1,1)

Uzman 3	Müşteri Cirosu	Müşteri Potansiyel Cirosu	Çalışma Süresi:	Yük Değeri:	Yük Miktarı:
Müşteri Cirosu:	(1,1,1)	(1,1,1)	(1,1,1)	(2/3,1,3/2)	(2/3,1,3/2)
Müşteri Potansiyel Cirosu:	(1,1,1)	(1,1,1)	(2/3,1,3/2)	(2/3,1,3/2)	(5/2,3,7/2)
Çalışma Süresi:	(1,1,1)	(5/2,3,7/2)	(1,1,1)	(2/3,1,3/2)	(2/3,1,3/2)
Yük Değeri:	(5/2,3,7/2)	(5/2,3,7/2)	(5/2,3,7/2)	(1,1,1)	(3/2,2,5/2)
Yük Miktarı:	(5/2,3,7/2)	(2/3,1,3/2)	(5/2,3,7/2)	(3/2,2,5/2)	(1,1,1)

Uzman 4	Müşteri Cirosu	Müşteri Potansiyel Cirosu	Çalışma Süresi:	Yük Değeri:	Yük Miktarı:
Müşteri Cirosu:	(1,1,1)	(5/2,3,7/2)	(2/3,1,3/2)	(2/3,1,3/2)	(2/3,1,3/2)
Müşteri Potansiyel Cirosu:	(2/3,1,3/2)	(1,1,1)	(5/2,3,7/2)	(3/2,2,5/2)	(3/2,2,5/2)
Çalışma Süresi:	(5/2,3,7/2)	(2/3,1,3/2)	(1,1,1)	(2/3,1,3/2)	(2/3,1,3/2)
Yük Değeri:	(5/2,3,7/2)	(3/2,2,5/2)	(5/2,3,7/2)	(1,1,1)	(2/3,1,3/2)
Yük Miktarı:	(5/2,3,7/2)	(3/2,2,5/2)	(5/2,3,7/2)	(5/2,3,7/2)	(1,1,1)

Uzman 5	Müşteri Cirosu	Müşteri Potansiyel Cirosu	Çalışma Süresi:	Yük Değeri:	Yük Miktarı:
Müşteri Cirosu:	(1,1,1)	(3/2,2,5/2)	(5/2,3,7/2)	(2/3,1,3/2)	(2/3,1,3/2)
Müşteri Potansiyel Cirosu:	(3/2,2,5/2)	(1,1,1)	(2/3,1,3/2)	(2/3,1,3/2)	(2/3,1,3/2)
Çalışma Süresi:	(2/3,1,3/2)	(5/2,3,7/2)	(1,1,1)	(2/3,1,3/2)	(2/3,1,3/2)
Yük Değeri:	(5/2,3,7/2)	(5/2,3,7/2)	(5/2,3,7/2)	(1,1,1)	(3/2,2,5/2)
Yük Miktarı:	(5/2,3,7/2)	(5/2,3,7/2)	(5/2,3,7/2)	(3/2,2,5/2)	(1,1,1)

Bu ağırlıklandırma işlemi sonucunda her bir parametrenin ağırlığı aşağıdaki üçgen sayılarla ifade edilen şekilde bulunmuştur.

Tablo 5.5. Parametre ağırlıkları

Parametre	Ağırlık
Müşteri Cirosu:	(0.12,0.17,0.27)
Müşteri Potansiyel Cirosu:	(0.12,0.18,0.28)
Çalışma Süresi:	(0.11,0.16,0.25)
Yük Değeri:	(0.16,0.24,0.36)
Yük Miktarı:	(0.16,0.23,0.35)

Daha sonra müşterilerin ilgili parametrelere karşılık gelen sayısal değerleri aşağıdaki matrise göre puanlanarak her müşteri için ayrı ayrı değerlendirmeler yapılmıştır.

Tablo 5.6. Sözel Değerlerin bulanık karşılıkları

Sözel Değer	Bulanık Değer
Çok Önemli	(4,5,6)
Önemli	(3,4,5)
Orta	(2,3,4)
Az önemli	(1,2,3)
Önemsiz	(0,1,2)

Müşterilerin her biri, seçilen beş kritere göre değerlendirilmiş ve aşağıdaki tablo oluşturulmuştur. Bu tablonun bulanık değerler ile ifade edilmiş şekli tablo 5.7.'de gösterilmektedir.

Tablo 5.7. Müşterileri önem değerlendirmeleri (Bulanık sayılar ile)

	Müşteri Cirosu	Müşteri Potansiyel Cirosu	Çalışma Süresi:	Yük Değeri:	Yük Miktarı:
Müşteri 1	(04,05,06)	(04,05,06)	(04,05,06)	(04,05,06)	(04,05,06)
Müşteri 2	(04,05,06)	(02,03,04)	(04,05,06)	(01,02,03)	(01,02,03)
Müşteri 3	(02,03,04)	(04,05,06)	(02,03,04)	(02,03,04)	(03,04,05)
Müşteri 4	(03,04,05)	(04,05,06)	(03,04,05)	(00,01,02)	(00,01,02)
Müşteri 5	(03,04,05)	(02,03,04)	(04,05,06)	(00,01,02)	(00,01,02)
Müşteri 6	(03,04,05)	(03,04,05)	(04,05,06)	(02,03,04)	(03,04,05)
Müşteri 7	(03,04,05)	(02,03,04)	(02,03,04)	(02,03,04)	(01,02,03)
Müşteri 8	(02,03,04)	(02,03,04)	(02,03,04)	(02,03,04)	(02,03,04)
Müşteri 9	(03,04,05)	(02,03,04)	(02,03,04)	(00,01,02)	(00,01,02)
Müşteri 10	(03,04,05)	(04,05,06)	(02,03,04)	(02,03,04)	(02,03,04)
Müşteri 11	(03,04,05)	(04,05,06)	(02,03,04)	(02,03,04)	(02,03,04)
Müşteri 12	(02,03,04)	(02,03,04)	(02,03,04)	(00,01,02)	(00,01,02)
Müşteri 13	(04,05,06)	(00,01,02)	(00,01,02)	(00,01,02)	(00,01,02)
Müşteri 14	(02,03,04)	(02,03,04)	(04,05,06)	(04,05,06)	(04,05,06)
Müşteri 15	(00,01,02)	(00,01,02)	(00,01,02)	(00,01,02)	(00,01,02)
Müşteri 16	(03,04,05)	(03,04,05)	(03,04,05)	(03,04,05)	(03,04,05)
Müşteri 17	(00,01,02)	(04,05,06)	(00,01,02)	(00,01,02)	(00,01,02)
Müşteri 18	(00,01,02)	(00,01,02)	(00,01,02)	(04,05,06)	(01,02,03)
Müşteri 19	(00,01,02)	(00,01,02)	(00,01,02)	(00,01,02)	(00,01,02)
Müşteri 20	(02,03,04)	(02,03,04)	(02,03,04)	(02,03,04)	(02,03,04)

Daha sonra her parametre kendi ağırlığı ile işleme sokularak her bir müşteri için genel bir önem derecesi elde edilmiştir. Böylece parametre değerleri bulanıklaştırılmıştır.

Tablo 5.8. Müşteri toplam önem dereceleri

	Toplam Önem
Müşteri 1	(2.68,4.9,9.06)
Müşteri 2	(1.48,3.13,6.37)
Müşteri 3	(1.74,3.53,6.95)
Müşteri 4	(1.17,2.69,5.7)
Müşteri 5	(1.04,2.49,5.39)
Müşteri 6	(1.96,3.84,7.44)
Müşteri 7	(1.3,2.88,5.96)
Müşteri 8	(1.34,2.94,6.04)
Müşteri 9	(0.82,2.17,4.89)
Müşteri 10	(1.7,3.47,6.87)
Müşteri 11	(1.7,3.47,6.87)
Müşteri 12	(0.7,2,4.62)
Müşteri 13	(0.48,1.66,4.1)
Müşteri 14	(2.2,4.2,7.96)
Müşteri 15	(0,0.98,3.02)
Müşteri 16	(2.01,3.92,7.55)
Müşteri 17	(0.48,1.7,4.14)
Müşteri 18	(0.8,2.17,4.81)
Müşteri 19	(0,0.98,3.02)
Müşteri 20	(1.34,2.94,6.04)

Daha sonra problem GSP'ye çevrilerek optimize edilmiştir. Bu optimizasyon iki yöntemle gerçekleştirilmiştir. Birinci yöntem GSP çözümünde çok sık kullanılan bir sezgisel yöntem, ikinci yöntem ise dinamik bulanık genetik algoritmadır.

5.3. Problemin Sezgisel Yöntem İle Çözümü

Bu bölümde problemin sezgisel bir yöntem ile çözümü incelenecektir. Burada gezgin satıcı probleminin çözümünde en sık kullanılan yöntem olan "En Yakın Seçimi" kullanılarak problem için bir aday optimum çözüm üretilecektir.

Bu sezgisel yönteme göre

1. Adım: Depodan çıkan araç önem/uzaklık oranı en yüksek olan hedefe yönelir.

2. Adım: Araç bulunduğu noktada tekrar değerlendirme yapar ve önem/uzaklık oranı en yüksek olan hedefe yönelir. Eğer birden fazla hedef için en yüksek önem/uzaklık oranı söz konusuysa araç en yakındaki noktaya yönelir. Eğer uzaklıklar da eşitse araç müşteri numarası en küçük olana yönelecektir.

3. Adım: Tüm müşteriler ziyaret edilene kadar 2. adım tekrarlanır.

4. Adım. Son müşteri ziyaret edildikten sonra araç depoya geri döner.

Problemin çözümü için öncelikle her müşteri için önem önem/uzaklık değerleri aşağıdaki tabloda hesaplanmıştır.

Tablo 5.9. Müşteri önem/uzaklık değerleri

	Müşteri 1	Müşteri 2	Müşteri 3	Müşteri 4	Müşteri 5	Müşteri 6	Müşteri 7	Müşteri 8	Müşteri 9	Müşteri 10	Müşteri 11	Müşteri 12	Müşteri 13	Müşteri 14	Müşteri 15	Müşteri 16	Müşteri 17	Müşteri 18	Müşteri 19	Müşteri 20
X	(1.34,2.45,4.53)	(0.37,0.78,1.59)	(0.35,0.71,1.39)	(0.29,0.67,1.43)	(0.13,0.31,0.67)	(0.28,0.55,1.06)	(0.16,0.36,0.75)	(0.15,0.33,0.67)	(0.08,0.22,0.49)	(0.21,0.43,0.86)	(0.21,0.43,0.86)	(0.1,0.29,0.66)	(0.07,0.24,0.59)	(0.31,0.6,1.14)	(0.08,0.23)	(0.4,0.78,1.51)	(0.04,0.13,0.32)	(0.07,0.18,0.4)	(0.14,0.43)	(0.15,0.33,0.67)
Müşteri 1	(0,0,0)	(0.74,1.57,3.19)	(0.35,0.71,1.39)	(0.29,0.67,1.43)	(0.17,0.42,0.9)	(0.28,0.55,1.06)	(0.16,0.36,0.75)	(0.12,0.27,0.55)	(0.07,0.18,0.41)	(0.17,0.35,0.69)	(0.17,0.35,0.69)	(0.1,0.29,0.66)	(0.07,0.24,0.59)	(0.31,0.6,1.14)	(0.08,0.23)	(0.67,1.31,2.52)	(0.03,0.11,0.28)	(0.06,0.16,0.34)	(0.0,11,0.34)	(0.12,0.27,0.55)
Müşteri 2	(1.34,2.45,4.53)	(0,0,0)	(0.35,0.71,1.39)	(0.29,0.67,1.43)	(0.26,0.62,1.35)	(0.28,0.55,1.06)	(0.16,0.36,0.75)	(0.1,0.23,0.46)	(0.06,0.16,0.35)	(0.14,0.29,0.57)	(0.14,0.29,0.57)	(0.08,0.22,0.51)	(0.07,0.24,0.59)	(0.31,0.6,1.14)	(0.08,0.23)	(0.67,1.31,2.52)	(0.03,0.1,0.24)	(0.05,0.14,0.3)	(0.0,09,0.27)	(0.1,0.23,0.46)
Müşteri 3	(0.54,0.98,1.81)	(0.3,0.63,1.27)	(0,0,0)	(0.39,0.9,1.9)	(0.15,0.36,0.77)	(0.16,0.32,0.62)	(0.1,0.22,0.46)	(0.13,0.29,0.6)	(0.07,0.2,0.44)	(0.19,0.39,0.76)	(0.19,0.39,0.76)	(0.12,0.33,0.77)	(0.12,0.42,1.03)	(1.1,2.1,3.98)	(0.12,0.38)	(0.25,0.49,0.94)	(0.03,0.11,0.28)	(0.06,0.17,0.37)	(0.0,12,0.38)	(0.13,0.29,0.6)
Müşteri 4	(0.67,1.23,2.27)	(0.37,0.78,1.59)	(0.58,1.18,2.32)	(0,0,0)	(0.17,0.42,0.9)	(0.18,0.35,0.68)	(0.11,0.24,0.5)	(0.1,0.23,0.46)	(0.06,0.16,0.35)	(0.14,0.29,0.57)	(0.14,0.29,0.57)	(0.08,0.22,0.51)	(0.07,0.24,0.59)	(0.44,0.84,1.59)	(0.11,0.34)	(0.29,0.56,1.08)	(0.03,0.1,0.24)	(0.05,0.14,0.3)	(0.0,09,0.27)	(0.1,0.23,0.46)
Müşteri 5	(0.45,0.82,1.51)	(0.37,0.78,1.59)	(0.25,0.5,0.99)	(0.2,0.45,0.95)	(0,0,0)	(0.39,0.77,1.49)	(0.22,0.48,0.99)	(0.08,0.17,0.36)	(0.05,0.12,0.27)	(0.11,0.22,0.43)	(0.11,0.22,0.43)	(0.05,0.15,0.36)	(0.04,0.15,0.37)	(0.24,0.47,0.88)	(0.07,0.2)	(0.67,1.31,2.52)	(0.02,0.08,0.2)	(0.04,0.11,0.24)	(0.07,0.2)	(0.08,0.17,0.36)
Müşteri 6	(0.38,0.7,1.29)	(0.21,0.45,0.91)	(0.15,0.29,0.58)	(0.11,0.24,0.52)	(0.21,0.5,1.08)	(0,0,0)	(0.26,0.58,1.19)	(0.08,0.18,0.38)	(0.05,0.13,0.29)	(0.11,0.23,0.46)	(0.11,0.23,0.46)	(0.05,0.14,0.33)	(0.03,0.12,0.29)	(0.16,0.3,0.57)	(0.05,0.15)	(0.5,0.98,1.89)	(0.02,0.09,0.21)	(0.04,0.11,0.25)	(0.07,0.22)	(0.08,0.18,0.38)
Müşteri 7	(0.34,0.61,1.13)	(0.19,0.39,0.8)	(0.13,0.27,0.53)	(0.1,0.22,0.48)	(0.17,0.42,0.9)	(0.39,0.77,1.49)	(0,0,0)	(0.12,0.27,0.55)	(0.07,0.18,0.41)	(0.17,0.35,0.69)	(0.11,0.22,0.43)	(0.05,0.13,0.31)	(0.03,0.11,0.27)	(0.15,0.28,0.53)	(0.05,0.14)	(0.4,0.78,1.51)	(0.03,0.11,0.28)	(0.06,0.16,0.34)	(0.0,08,0.23)	(0.09,0.2,0.4)
Müşteri 8	(0.24,0.45,0.82)	(0.11,0.24,0.49)	(0.17,0.35,0.7)	(0.09,0.21,0.44)	(0.06,0.15,0.32)	(0.12,0.24,0.47)	(0.12,0.26,0.54)	(0,0,0)	(0.27,0.72,1.63)	(1.7,3.47,6.87)	(0.24,0.5,0.98)	(0.12,0.33,0.77)	(0.08,0.28,0.68)	(0.28,0.53,1)	(0.08,0.25)	(0.14,0.28,0.54)	(0.12,0.43,1.04)	(0.16,0.43,0.96)	(0.25,0.76)	(0.22,0.49,1.01)
Müşteri 9	(0.22,0.41,0.76)	(0.11,0.22,0.46)	(0.16,0.32,0.63)	(0.08,0.19,0.41)	(0.06,0.14,0.3)	(0.12,0.23,0.44)	(0.11,0.24,0.5)	(0.45,0.98,2.01)	(0,0,0)	(0.43,0.87,1.72)	(0.43,0.87,1.72)	(0.14,0.4,0.92)	(0.07,0.24,0.59)	(0.24,0.47,0.88)	(0.11,0.34)	(0.13,0.26,0.5)	(0.16,0.57,1.38)	(0.4,1.09,2.41)	(0.33,1.01)	(0.45,0.98,2.01)
Müşteri 10	(0.27,0.49,0.91)	(0.12,0.26,0.53)	(0.19,0.39,0.77)	(0.1,0.22,0.48)	(0.07,0.16,0.34)	(0.13,0.26,0.5)	(0.13,0.29,0.6)	(1.34,2.94,6.04)	(0.21,0.54,1.22)	(0,0,0)	(0.21,0.43,0.86)	(0.1,0.29,0.66)	(0.07,0.24,0.59)	(0.31,0.6,1.14)	(0.08,0.23)	(0.15,0.3,0.58)	(0.1,0.34,0.83)	(0.13,0.36,0.8)	(0.0,2,0.6)	(0.19,0.42,0.86)
Müşteri 11	(0.27,0.49,0.91)	(0.12,0.26,0.53)	(0.19,0.39,0.77)	(0.1,0.22,0.48)	(0.07,0.16,0.34)	(0.13,0.26,0.5)	(0.08,0.18,0.37)	(0.19,0.42,0.86)	(0.21,0.54,1.22)	(0.21,0.43,0.86)	(0,0,0)	(0.23,0.67,1.54)	(0.1,0.33,0.82)	(0.31,0.6,1.14)	(0.0,2,0.6)	(0.15,0.3,0.58)	(0.1,0.34,0.83)	(0.2,0.54,1.2)	(0.0,33,1.01)	(1.34,2.94,6.04)
Müşteri 12	(0.38,0.7,1.29)	(0.16,0.35,0.71)	(0.29,0.59,1.16)	(0.13,0.3,0.63)	(0.08,0.19,0.41)	(0.14,0.27,0.53)	(0.09,0.19,0.4)	(0.22,0.49,1.01)	(0.16,0.43,0.98)	(0.24,0.5,0.98)	(0.57,1.16,2.29)	(0,0,0)	(0.24,0.83,2.05)	(0.55,1.05,1.99)	(0.16,0.5)	(0.2,0.39,0.76)	(0.06,0.21,0.52)	(0.11,0.31,0.69)	(0.0,49,1.51)	(0.34,0.74,1.51)
Müşteri 13	(0.38,0.7,1.29)	(0.21,0.45,0.91)	(0.44,0.88,1.74)	(0.17,0.38,0.81)	(0.09,0.23,0.49)	(0.14,0.27,0.53)	(0.09,0.19,0.4)	(0.22,0.49,1.01)	(0.12,0.31,0.7)	(0.24,0.5,0.98)	(0.34,0.69,1.37)	(0.35,1.2,31)	(0,0,0)	(1.1,2.1,3.98)	(0.16,0.5)	(0.2,0.39,0.76)	(0.05,0.17,0.41)	(0.09,0.24,0.53)	(0.25,0.76)	(0.22,0.49,1.01)
Müşteri 14	(0.38,0.7,1.29)	(0.21,0.45,0.91)	(0.87,1.77,3.48)	(0.23,0.54,1.14)	(0.12,0.28,0.6)	(0.14,0.27,0.53)	(0.09,0.19,0.4)	(0.17,0.37,0.76)	(0.09,0.24,0.54)	(0.24,0.5,0.98)	(0.24,0.5,0.98)	(0.18,0.5,1.16)	(0.24,0.83,2.05)	(0,0,0)	(0.16,0.5)	(0.2,0.39,0.76)	(0.04,0.14,0.35)	(0.07,0.2,0.44)	(0.0,16,0.5)	(0.17,0.37,0.76)
Müşteri 15	(0.21,0.38,0.7)	(0.11,0.24,0.49)	(0.22,0.44,0.87)	(0.13,0.3,0.63)	(0.07,0.17,0.36)	(0.1,0.19,0.37)	(0.06,0.14,0.28)	(0.11,0.25,0.5)	(0.09,0.24,0.54)	(0.13,0.27,0.53)	(0.34,0.69,1.37)	(0.12,0.33,0.77)	(0.08,0.28,0.68)	(0.37,0.7,1.33)	(0,0,0)	(0.13,0.25,0.47)	(0.05,0.17,0.41)	(0.11,0.31,0.69)	(0.0,12,0.38)	(0.22,0.49,1.01)
Müşteri 16	(0.89,1.63,3.02)	(0.49,1.04,2.12)	(0.22,0.44,0.87)	(0.17,0.38,0.81)	(0.35,0.83,1.8)	(0.49,0.96,1.86)	(0.26,0.58,1.19)	(0.1,0.21,0.43)	(0.05,0.14,0.33)	(0.13,0.27,0.53)	(0.13,0.27,0.53)	(0.07,0.2,0.46)	(0.05,0.17,0.41)	(0.22,0.42,0.8)	(0.0,06,0.19)	(0,0,0)	(0.03,0.09,0.23)	(0.05,0.13,0.28)	(0.0,08,0.25)	(0.1,0.21,0.43)
Müşteri 17	(0.18,0.33,0.6)	(0.09,0.18,0.37)	(0.12,0.24,0.46)	(0.07,0.16,0.34)	(0.05,0.12,0.26)	(0.1,0.19,0.37)	(0.09,0.19,0.4)	(0.34,0.74,1.51)	(0.27,0.72,1.63)	(0.34,0.69,1.37)	(0.34,0.69,1.37)	(0.09,0.25,0.58)	(0.05,0.17,0.41)	(0.18,0.35,0.66)	(0.0,1,0.3)	(0.11,0.22,0.42)	(0,0,0)	(0.27,0.72,1.6)	(0.0,16,0.5)	(0.34,0.74,1.51)
Müşteri 18	(0.19,0.35,0.65)	(0.09,0.2,0.4)	(0.13,0.27,0.53)	(0.07,0.17,0.36)	(0.05,0.12,0.27)	(0.1,0.2,0.39)	(0.09,0.21,0.43)	(0.27,0.59,1.21)	(0.41,1.09,2.45)	(0.28,0.58,1.15)	(0.43,0.87,1.72)	(0.1,0.29,0.66)	(0.05,0.18,0.46)	(0.2,0.38,0.72)	(0.0,14,0.43)	(0.12,0.23,0.44)	(0.16,0.57,1.38)	(0,0,0)	(0.0,2,0.6)	(0.45,0.98,2.01)
Müşteri 19	(0.3,0.54,1.01)	(0.13,0.28,0.58)	(0.22,0.44,0.87)	(0.11,0.24,0.52)	(0.07,0.17,0.36)	(0.14,0.27,0.53)	(0.1,0.22,0.46)	(0.34,0.74,1.51)	(0.27,0.72,1.63)	(0.34,0.69,1.37)	(0.57,1.16,2.29)	(0.35,1.2,31)	(0.12,0.42,1.03)	(0.37,0.7,1.33)	(0.0,12,0.38)	(0.17,0.33,0.63)	(0.08,0.28,0.69)	(0.16,0.43,0.96)	(0.0,0)	(0.67,1.47,3.02)
Müşteri 20	(0.24,0.45,0.82)	(0.11,0.24,0.49)	(0.17,0.35,0.7)	(0.09,0.21,0.44)	(0.06,0.15,0.32)	(0.12,0.24,0.47)	(0.09,0.19,0.4)	(0.22,0.49,1.01)	(0.27,0.72,1.63)	(0.24,0.5,0.98)	(1.7,3.47,6.87)	(0.18,0.5,1.16)	(0.08,0.28,0.68)	(0.28,0.53,1)	(0.16,0.5)	(0.14,0.28,0.54)	(0.12,0.43,1.04)	(0.27,0.72,1.6)	(0.0,49,1.51)	(0,0,0)

Daha sonra yukarıda verilen algoritmaya göre seçimler yapılarak problem çözülmüştür.

Tablo 5.10. Problemin Sezgisel Çözümü

	Müşteri 1	Müşteri 2	Müşteri 3	Müşteri 4	Müşteri 5	Müşteri 6	Müşteri 7	Müşteri 8	Müşteri 9	Müşteri 10	Müşteri 11	Müşteri 12	Müşteri 13	Müşteri 14	Müşteri 15	Müşteri 16	Müşteri 17	Müşteri 18	Müşteri 19	Müşteri 20
X	(1.34,2.45,4.53)	(0.37,0.78,1.59)	(0.35,0.71,1.39)	(0.29,0.67,1.43)	(0.13,0.31,0.67)	(0.28,0.55,1.06)	(0.16,0.36,0.75)	(0.15,0.33,0.67)	(0.08,0.22,0.49)	(0.21,0.43,0.86)	(0.21,0.43,0.86)	(0.1,0.29,0.66)	(0.07,0.24,0.59)	(0.31,0.6,1.14)	(0.08,0.23)	(0.4,0.78,1.51)	(0.04,0.13,0.32)	(0.07,0.18,0.4)	(0.14,0.43)	(0.15,0.33,0.67)

	Müşteri 2	Müşteri 3	Müşteri 4	Müşteri 5	Müşteri 6	Müşteri 7	Müşteri 8	Müşteri 9	Müşteri 10	Müşteri 11	Müşteri 12	Müşteri 13	Müşteri 14	Müşteri 15	Müşteri 16	Müşteri 17	Müşteri 18	Müşteri 19	Müşteri 20
Müşteri 1	(0.74,1.57,3.19)	(0.35,0.71,1.39)	(0.29,0.67,1.43)	(0.17,0.42,0.9)	(0.28,0.55,1.06)	(0.16,0.36,0.75)	(0.12,0.27,0.55)	(0.07,0.18,0.41)	(0.17,0.35,0.69)	(0.17,0.35,0.69)	(0.1,0.29,0.66)	(0.07,0.24,0.59)	(0.31,0.6,1.14)	(0.08,0.23)	(0.67,1.31,2.52)	(0.03,0.11,0.28)	(0.06,0.16,0.34)	(0.0,11,0.34)	(0.12,0.27,0.55)

	Müşteri 3	Müşteri 4	Müşteri 5	Müşteri 6	Müşteri 7	Müşteri 8	Müşteri 9	Müşteri 10	Müşteri 11	Müşteri 12	Müşteri 13	Müşteri 14	Müşteri 15	Müşteri 16	Müşteri 17	Müşteri 18	Müşteri 19	Müşteri 20
Müşteri 2	(0.35,0.71,1.39)	(0.29,0.67,1.43)	(0.26,0.62,1.35)	(0.28,0.55,1.06)	(0.16,0.36,0.75)	(0.1,0.23,0.46)	(0.06,0.16,0.35)	(0.14,0.29,0.57)	(0.14,0.29,0.57)	(0.08,0.22,0.51)	(0.07,0.24,0.59)	(0.31,0.6,1.14)	(0.0,08,0.23)	(0.67,1.31,2.52)	(0.03,0.1,0.24)	(0.05,0.14,0.3)	(0.0,09,0.27)	(0.1,0.23,0.46)	

	Müşteri 3	Müşteri 4	Müşteri 5	Müşteri 6	Müşteri 7	Müşteri 8	Müşteri 9	Müşteri 10	Müşteri 11	Müşteri 12	Müşteri 13	Müşteri 14	Müşteri 15	Müşteri 17	Müşteri 18	Müşteri 19	Müşteri 20
Müşteri 16	(0.22,0.44,0.87)	(0.17,0.38,0.81)	(0.35,0.83,1.8)	(0.49,0.96,1.86)	(0.26,0.58,1.19)	(0.1,0.21,0.43)	(0.05,0.14,0.33)	(0.13,0.27,0.53)	(0.13,0.27,0.53)	(0.07,0.2,0.46)	(0.05,0.17,0.41)	(0.22,0.42,0.8)	(0.0,06,0.19)	(0.03,0.09,0.23)	(0.05,0.13,0.28)	(0.0,08,0.25)	(0.1,0.21,0.43)		

	Müşteri 3	Müşteri 4	Müşteri 5	Müşteri 7	Müşteri 8	Müşteri 9	Müşteri 10	Müşteri 11	Müşteri 12	Müşteri 13	Müşteri 14	Müşteri 15	Müşteri 17	Müşteri 18	Müşteri 19	Müşteri 20
Müşteri 6	(0.15,0.29,0.58)	(0.11,0.24,0.52)	(0.21,0.5,1.08)	(0.26,0.58,1.19)	(0.08,0.18,0.38)	(0.05,0.13,0.29)	(0.11,0.23,0.46)	(0.11,0.23,0.46)	(0.05,0.14,0.33)	(0.03,0.12,0.29)	(0.16,0.3,0.57)	(0.0,05,0.15)	(0.02,0.09,0.21)	(0.04,0.11,0.25)	(0.0,07,0.22)	(0.08,0.18,0.38)			

	Müşteri 3	Müşteri 4	Müşteri 5	Müşteri 8	Müşteri 9	Müşteri 10	Müşteri 11	Müşteri 12	Müşteri 13	Müşteri 14	Müşteri 15	Müşteri 17	Müşteri 18	Müşteri 19	Müşteri 20
Müşteri 7	(0.13,0.27,0.53)	(0.1,0.22,0.48)	(0.17,0.42,0.9)	(0.12,0.27,0.55)	(0.07,0.18,0.41)	(0.17,0.35,0.69)	(0.11,0.22,0.43)	(0.05,0.13,0.31)	(0.03,0.11,0.27)	(0.15,0.28,0.53)	(0.0,05,0.14)	(0.03,0.11,0.28)	(0.06,0.16,0.34)	(0.0,08,0.23)	(0.09,0.2,0.4)				

	Müşteri 3	Müşteri 4	Müşteri 5	Müşteri 8	Müşteri 9	Müşteri 10	Müşteri 11	Müşteri 12	Müşteri 13	Müşteri 15	Müşteri 17	Müşteri 18	Müşteri 19	Müşteri 20
Müşteri 14	(0.87,1.77,3.48)	(0.23,0.54,1.14)	(0.12,0.28,0.6)	(0.17,0.37,0.76)	(0.09,0.24,0.54)	(0.24,0.5,0.98)	(0.24,0.5,0.98)	(0.18,0.5,1.16)	(0.24,0.83,2.05)	(0.0,16,0.5)	(0.04,0.14,0.35)	(0.07,0.2,0.44)	(0.0,16,0.5)	(0.17,0.37,0.76)					

	Müşteri 4	Müşteri 5	Müşteri 8	Müşteri 9	Müşteri 10	Müşteri 11	Müşteri 12	Müşteri 13	Müşteri 15	Müşteri 17	Müşteri 18	Müşteri 19	Müşteri 20
Müşteri 3	(0.39,0.9,1.9)	(0.15,0.36,0.77)	(0.13,0.29,0.6)	(0.07,0.2,0.44)	(0.19,0.39,0.76)	(0.19,0.39,0.76)	(0.12,0.33,0.77)	(0.12,0.42,1.03)	(0.0,12,0.38)	(0.03,0.11,0.28)	(0.06,0.17,0.37)	(0.0,12,0.38)	(0.13,0.29,0.6)

	Müşteri 5	Müşteri 8	Müşteri 9	Müşteri 10	Müşteri 11	Müşteri 12	Müşteri 13	Müşteri 15	Müşteri 17	Müşteri 18	Müşteri 19	Müşteri 20
Müşteri 4	(0.17,0.42,0.9)	(0.1,0.23,0.46)	(0.06,0.16,0.35)	(0.14,0.29,0.57)	(0.14,0.29,0.57)	(0.08,0.22,0.51)	(0.07,0.24,0.59)	(0.0,11,0.34)	(0.03,0.1,0.24)	(0.05,0.14,0.3)	(0.0,09,0.27)	(0.1,0.23,0.46)	

	Müşteri 8	Müşteri 9	Müşteri 10	Müşteri 11	Müşteri 12	Müşteri 13	Müşteri 15	Müşteri 17	Müşteri 18	Müşteri 19	Müşteri 20
Müşteri 5	(0.08,0.17,0.36)	(0.05,0.12,0.27)	(0.11,0.22,0.43)	(0.11,0.22,0.43)	(0.05,0.15,0.36)	(0.04,0.15,0.37)	(0.0,07,0.2)	(0.02,0.08,0.2)	(0.04,0.11,0.24)	(0.0,07,0.2)	(0.08,0.17,0.36)

	Müşteri 8	Müşteri 9	Müşteri 11	Müşteri 12	Müşteri 13	Müşteri 15	Müşteri 17	Müşteri 18	Müşteri 19	Müşteri 20
Müşteri 10	(1.34,2.94,6.04)	(0.21,0.54,1.22)	(0.21,0.43,0.86)	(0.1,0.29,0.66)	(0.07,0.24,0.59)	(0.0,08,0.23)	(0.1,0.34,0.83)	(0.13,0.36,0.8)	(0.0,2,0.6)	(0.19,0.42,0.86)	

	Müşteri 9	Müşteri 11	Müşteri 12	Müşteri 13	Müşteri 15	Müşteri 17	Müşteri 18	Müşteri 19	Müşteri 20
Müşteri 8	(0.27,0.72,1.63)	(0.24,0.5,0.98)	(0.12,0.33,0.77)	(0.08,0.28,0.68)	(0.0,08,0.25)	(0.12,0.43,1.04)	(0.16,0.43,0.96)	(0.0,25,0.76)	(0.22,0.49,1.01)		

	Müşteri 11	Müşteri 12	Müşteri 13	Müşteri 15	Müşteri 17	Müşteri 18	Müşteri 19	Müşteri 20
Müşteri 9	(0.43,0.87,1.72)	(0.14,0.4,0.92)	(0.07,0.24,0.59)	(0.0,11,0.34)	(0.16,0.57,1.38)	(0.4,1.09,2.41)	(0.0,33,1.01)	(0.45,0.98,2.01)

	Müşteri 11	Müşteri 12	Müşteri 13	Müşteri 15	Müşteri 17	Müşteri 18	Müşteri 19
Müşteri 20	(1.7,3.47,6.87)	(0.18,0.5,1.16)	(0.08,0.28,0.68)	(0.0,16,0.5)	(0.12,0.43,1.04)	(0.27,0.72,1.6)	(0.0,49,1.51)

	Müşteri 12	Müşteri 13	Müşteri 15	Müşteri 17	Müşteri 18	Müşteri 19
Müşteri 11	(0.23,0.67,1.54)	(0.1,0.33,0.82)	(0.0,2,0.6)	(0.1,0.34,0.83)	(0.2,0.54,1.2)	(0.0,33,1.01)

	Müşteri 13	Müşteri 15	Müşteri 17	Müşteri 18	Müşteri 19
Müşteri 12	(0.24,0.83,2.05)	(0.0,16,0.5)	(0.06,0.21,0.52)	(0.11,0.31,0.69)	(0.0,49,1.51)

	Müşteri 15	Müşteri 17	Müşteri 18	Müşteri 19
Müşteri 13	(0.0,16,0.5)	(0.05,0.17,0.41)	(0.09,0.24,0.53)	(0.0,25,0.76)

	Müşteri 15	Müşteri 17	Müşteri 19
Müşteri 18	(0.0,14,0.43)	(0.16,0.57,1.38)	(0.0,2,0.6)

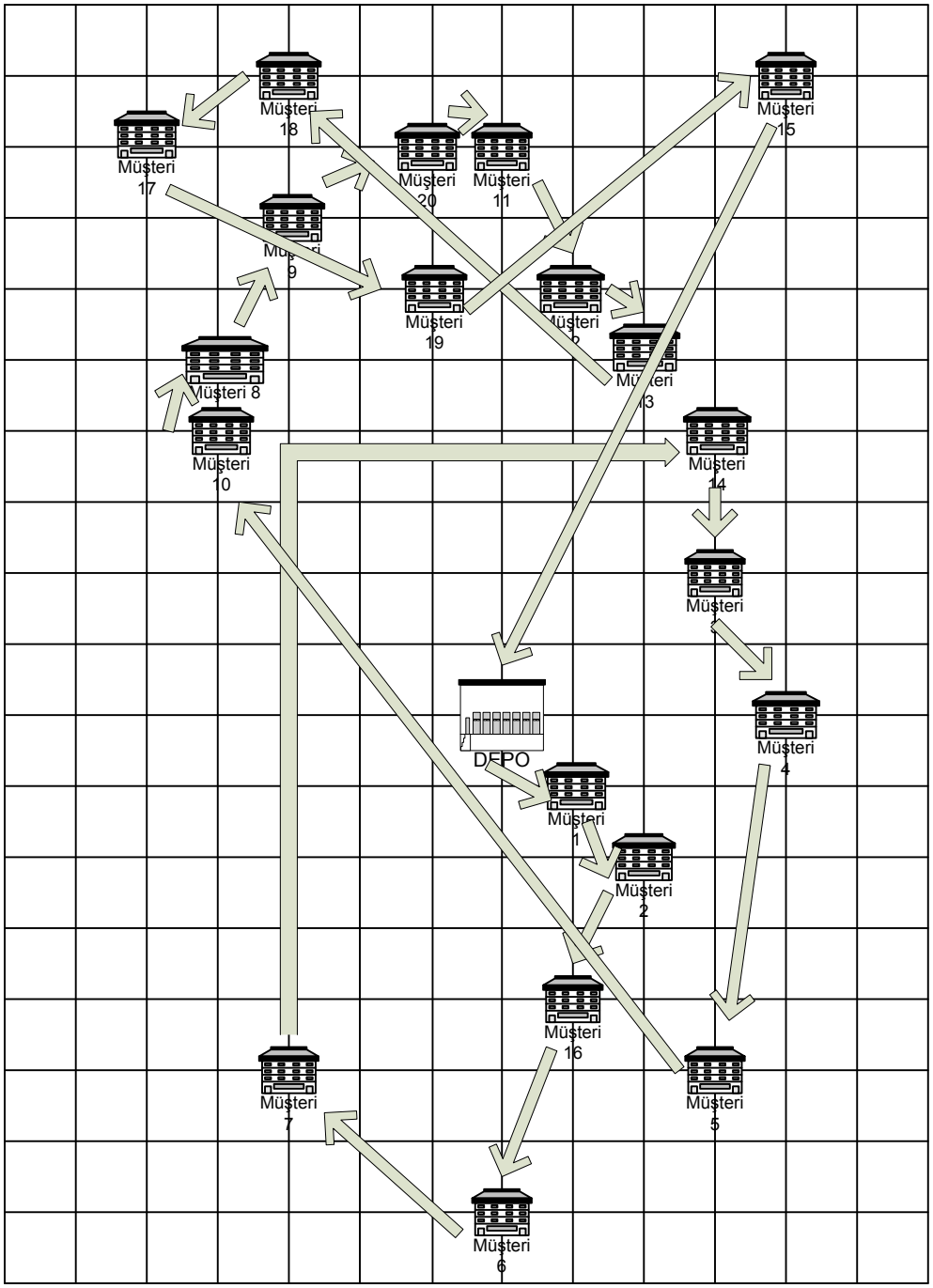
	Müşteri 15	Müşteri 19
Müşteri 17	(0.0,1,0.3)	(0.0,16,0.5)

	Müşteri 15
Müşteri 19	(0.0,12,0.38)

Problemin çözümü sezgisel yönteme göre: Depo – Müşteri 1 – Müşteri 2 – Müşteri 16 – Müşteri 6 – Müşteri 7– Müşteri 14– Müşteri 3– Müşteri 4– Müşteri 5– Müşteri 10– Müşteri

8– Müşteri 9– Müşteri 20– Müşteri 11– Müşteri 12– Müşteri 13– Müşteri 18– Müşteri 17– Müşteri 19– Müşteri 15 – Depo şeklindedir.

Bu çözümde toplam yol 110 birim olarak bulunmuştur. Çözüm aşağıdaki şekilde harita üzerinde gösterilmiştir.



Şekil 5.2. Problemin Sezgisel Çözümü

5.4. Problemin Dinamik Bulanık Genetik Algoritma İle Çözümü

5.4.1. Kodlama (Bireylerin Gösterimi)

GSP'nin GA ile çözümünde en önemli konulardan biri de olası çözümler için basit bir gösterim şeklinin geliştirilmesidir. Gösterim şeklinin basitliği ve açıklığı algoritmanın karmaşıklığını azaltabilir (Moon v.d., 2002).

Bu uygulamada bireylerin gösterimi için önceki bölümlerde anlatılan "permütasyon kodlama" işlemi tam sayılar kullanılarak yapılacaktır. Genetik algoritmada kullanılan değişkenler müşteri numarasını (şehir numarasını) göstermektedir.

5.4.2. Uyum Fonksiyonu

Problemin uyum (amaç) fonksiyonu önem derecesinin yüksekliğine göre yolu(maliyeti) ve sırayı(müşterinin ziyaret edilme sırasını) optimize edecek şekilde aşağıdaki gibi oluşturulmuştur.

Min ∑ [(ÖNEM1*YOL1*SIRA1)+...+(ÖNEMn*YOLn*SIRAn)]

Bu fonksiyonda:

- Önem: 5 parametrenin Bulanık AHP ile önceden belirlenen ağırlıklarına ve her müşterinin bu parametrelerdeki karşılığına göre hesaplanan bulanık(fuzzy) bir değerdir.
- Yol Uzaklık matrisinde belirtilen ilk ve son istasyonlar arasındaki uzaklık değerini ifade eden bir reel sayıdır.
- Sıra: TSP probleminde aracın ilgili istasyona uğrama sırasını veren bir tam sayı değerdir.

Burada maliyet ve önem derecesinin birbirlerine göre eşit oranda uyum fonksiyonunu etkiledikleri varsayılmıştır.

5.4.3. Genetik Operatörler

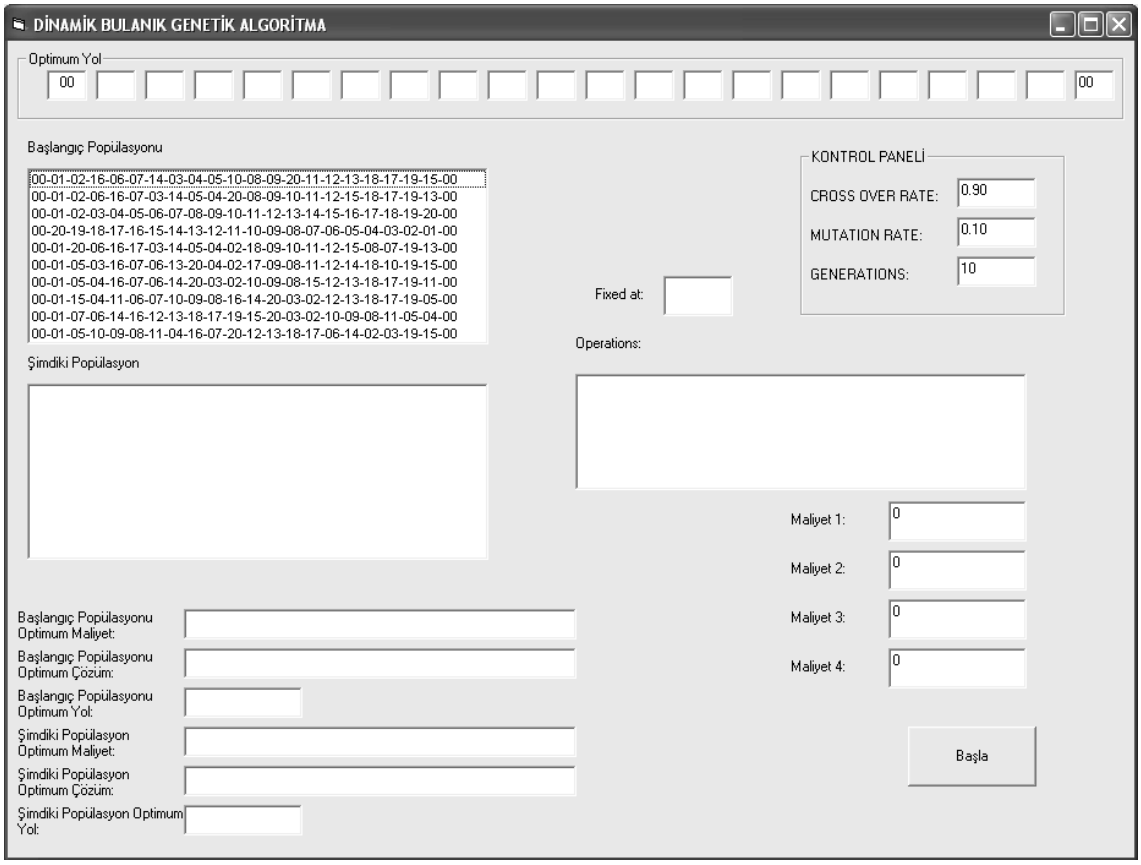
Bu uygulamada önceki bölümlerde açıklanan Uyum Orantılı Seçilim'e dayanan Uyum Tabanlı Yer Değiştirme, Çaprazlama (Crossover) ve Mutasyon (Mutation) genetik operatörleri ile birlikte kullanılacaktır.

5.4.4. Problemin Dinamik Bulanık Genetik Algoritma ile Çözülmesi

Problem koşulları birebir bilgisayar ortamına taşınarak MS Visual Basic 6.0 ile yazılan dinamik genetik algoritma ile çözüm simülasyonu yapılmıştır.

Burada kargo aracının izleyeceği yol, müşteri numaralarının ardı ardına dizilmesi ile bir gen şeklinde gösterilmiştir (Örnek: 00-01-05-04-16-07-06-14-20-03-02-10-09-08-15-12-13-18-17-19-11-00). Burada 00 ile gösterilen yer kargo deposu olup ilk ve son kromozom daima 00 olmak durumundadır.

Burada kullanılan algoritma dinamik bir bulanık genetik algoritmadır. Bu dinamizmin nedeni algoritmanın bir bütün olarak değil adım adım çalışmasından gelmektedir. Böylece her aşamada yeniden karar veren bir yapı oluşturulmuştur. Bu sayede özellikle büyük kargo problemlerinde çözüm için geçmesi gereken bilgisayar zamanının büyük kısmı işlem ile aynı süreçte paralel işleyebilecektir. Kısacası optimum sonuç net bulunmadan araçlar yola çıkabilecektir ve araçlar yolda iken sonraki hareketler için kararlar verilebilecektir.



Şekil 5.3. Programın Başlangıç Görünümü

Bu uygulama için geliştirilen dinamik genetik algoritmanın çalışma prensibi aşağıdaki şekildedir.

Adım 1: İlk tur için rastsal olarak oluşturulan başlangıç popülasyonundan alınan değerler şimdiki popülasyon alanına alınır.

Adım 2: Seçilen rastsal bir sayıya ve kontrol paneline girilen mutasyon ve çaprazlama olasılıklarına göre mutasyon ve/veya çaprazlama operatörlerinin kullanımına karar verilir.

Adım 3: Eğer 2. adımda mutasyon oluşacağı sonucu çıkmamış ise 9. adıma geçilir.

Adım 4: Eğer 2. adımda mutasyon oluşacağı sonucu çıkmış ise 5. adıma geçilir.

Adım 5: Şimdiki popülasyon alanından 1 birey (gen) seçilir. Bu birey şimdiki popülasyon alanından çıkarılıp, işlem alanına alınır.

Adım 6: Seçilen bireyin gen uzunluğuna bağlı olarak bir rastsal sayı seçilir ve seçilen noktadaki gen kendinden bir sonraki genle yer değiştirerek mutasyona uğrar.

Adım 7: Mutasyona uğrayan gen ve aynı genin mutasyona uğramadan önceki halleri uyum fonksiyonuna göre karşılaştırılır. Eğer mutasyon iyi yönde bir mutasyon ise mutasyona uğrayan birey şimdiki popülasyona dahil edilir.

Adım 8: Eğer mutasyon bireyin uyum fonksiyonundaki değerini düşürmüşse bu işlem yok sayılır ve ilk birey şimdiki popülasyona geri alınır.

Adım 9: Eğer 2. adımda çaprazlama oluşacağı sonucu çıkmamış ise 17. adıma geçilir.

Adım 10: Eğer 2. adımda çaprazlama oluşacağı sonucu çıkmış ise 11. adıma geçilir.

Adım 11: Şimdiki popülasyon alanından 2 adet birey (gen) seçilir. Bu bireyler şimdiki popülasyon alanından çıkarılıp, işlem alanına alınır.

Adım 12: Seçilen bireylerin gen uzunluğuna bağlı olarak bir rastsal sayı seçilir ve seçilen sayıya göre belirlenen noktalardan itibaren 5 adet gen çaprazlamaya tabi tutulur.

Adım 14: Çaprazlama sonucunda oluşan 2 adet yavru birey işlem alanına eklenir.

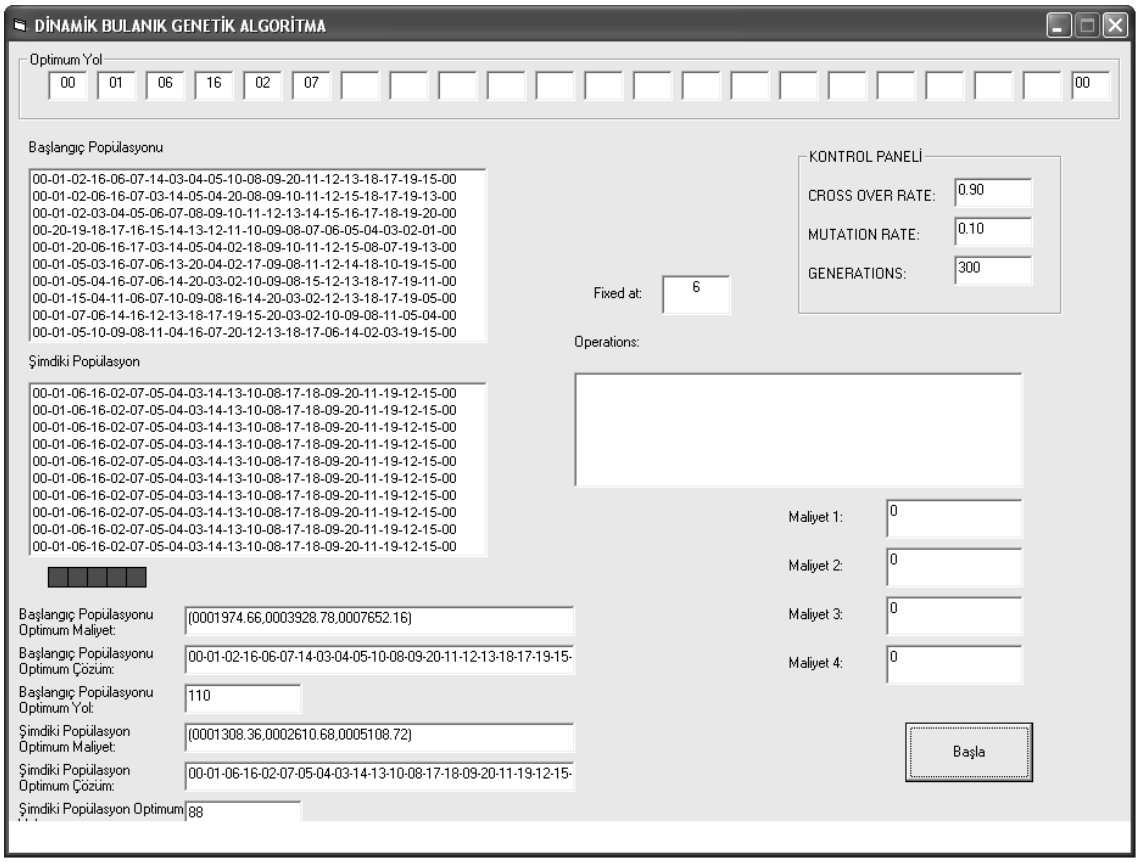
Adım 15: İşlem alanındaki 4 bireyin (2 ebeveyn, 2 yavru) uyum fonksiyon değerleri karşılaştırılarak en yüksek uyuma sahip iki birey şimdiki popülasyon alınana alınır.

Adım 16: İşlem alanında kalan düşük uyum'a sahip iki birey yok edilir.

Adım 17. Kontrol panelinde belirlenen her tur için adım 2- adım 16 arasındaki işlemler tekrarlanır.

Adım 18: Bulunan yerel optimum sonuca göre ilk adımın ne olması gerektiğine karar verilir. İlk gen bu değere sabitlenir.

Adım 19: Adım 1 – Adım 18 arasındaki işlemler her bir gen için tekrarlanır.



Şekil 5.4. Programın çalışması

Program 0.90 çaprazlama oranı ve 0.10 mutasyon oranına göre 5.000 nesil için çalıştırıldığında aşağıdaki sonuç bulunmuştur:

Depo – Müşteri 1- Müşteri 16 - Müşteri 02 - Müşteri 05 - Müşteri 06 - Müşteri 07 - Müşteri 03 - Müşteri 04- Müşteri 14 - Müşteri 10 - Müşteri 08 - Müşteri 09 - Müşteri 12 - Müşteri 11 - Müşteri 20 - Müşteri 17 - Müşteri 18 - Müşteri 15 - Müşteri 13 - Müşteri 19- Depo

Bu sonucun uyum fonksiyonundaki değeri (847.30, 2142.31, 4701.85) olup, toplam yol 94 birim olarak bulunmuştur. Böylece sezgisel yöntem ile bulunan uyum değeri % 13.66 ve toplam yol ise % 14.54 geliştirilmiştir.

6. SONUÇ

Günümüzde üretim kaynaklarının gelişmesi ile çok büyük miktarlarda üretim yapabilen organizasyonların sayısında büyük bir artış gerçekleşmiştir. Bu artışa bağlı olarak artık üretmekten ziyade doğru malın, doğru zamanda, doğru yere, en uygun maliyetle ulaştırılması organizasyonların rekabet gücünü oluşturur hale gelmiştir. İşte bu noktada lojistik sistemler önem kazanmaya başlamıştır.

Kargo dağıtımı problemi başlıca lojistik problemlerinden biridir. Bu çalışmada NP-hard bir problem olan öncelik kısıtlı kargo dağıtımı probleminin bir Gezgin Satıcı Problemi olarak modellenmesi ve dinamik bir bulanık genetik algoritma ile çözülmesi amaçlanmıştı. Böylece hem müşteri memnuniyetini geliştiren hem de maliyeti göz önünde bulunduran bir sistem tasarlanmak istenmişti.

Burada kullanılan algoritma, hem Bulanık Mantık ve Genetik Algoritmaların birlikte kullanılması, hem de her noktada yeniden karar veren ve sistemdeki değişikliklere adapte olarak problemin çözülmesini yeniden sağlayan dinamik bir yapıya sahip olması açısından oldukça gelişmiş bir algoritmadır.

Bu algoritmanın bir başka özelliği ise sistemin çalışmaya başlamak için optimum sonucun bulunmasını beklemek zorunda olmamasıdır. Algoritma göreceli daha az koşumla ilk noktayı belirlediğinde sistem çalışmaya başlayabilmekte ve sistem çalışırken algoritma eş zamanlı olarak optimizasyona devam edebilmektedir. Böylece kargo dağıtımı ve toplanması için etkili ve hızlı bir yöntem öne sürülmüştür.

Bu yöntemin etkinliği oluşturulan örnek bir sistem üzerinde test edilmiştir. Örnek sistem oluşturulduktan sonra hem sezgisel “En yakın seçimi” yöntemi ile hem de oluşturulan “Dinamik Bulanık Genetik Algoritma” ile optimize edilmiştir.

Yapılan benzetim sonucunda hem sezgisel yöntem ile bulunan sonuçtan ortalama %22,94 oranında daha gelişmiş hem de yol maliyeti sezgisel yönteme göre ortalama %14.45 oranında azaltılmış bir sonuç elde edilmiştir.

Bu Dinamik Bulanık Genetik Algoritma ileriki çalışmalarda GPS ve dijital harita verileri ile kargo şirketinin müşteri bilgilerini içeren ana veri tabanı arasında kurulacak bir arabirim aracılığı ile çok daha büyük sistemlere kolayca adapte edilebilir özelliktedir. Böylece seçilecek kargo şirketine, hem müşteri memnuniyetini arttıracak, hem de yol maliyetlerini kontrol altında tutacak bir sistem uygulanabilecektir.

7. KAYNAKLAR

Choi, I.C., Kim, S. I. ve Kim, H. S. . (2003) *"A genetic algorithm with a mixedregion search for the asymmetric traveling salesman problem"* Computers & Operations Research, 30, 773–786

Eiben A.E. ve Smith J.E. (2003). Introduction to Evolutionary Computing (1st Edition). Springer, Natural Computing Series

F Herrera ve M Lozano (1999). "Fuzzy genetic algorithms: issues and models", Teknik Rapor, Dept. of Science and AI, University of Granada

Goldberg, David E (1989), "*Genetic Algorithms in Search, Optimization and Machine Learning"*, Kluwer Academic Publishers, Boston, MA.

Katayama, K., Sakamoto, H. ve Narihisa, H. (2000). "*The efficiency of hybrid mutation genetic algorithm for the travelling salesman problem"*, Mathematical and Computer Modelling, 31, 197–203

Kosko, B ve Isaka S. (1993). "Fuzzy Logic", Sciencetific American, July 1993, 76-81

Lee, C.C. (1990) "Fuzzy logic control systems: fuzzy logic controller –Part 1," IEEE Trans. System Man Cybernet. SMC-20 404C418, 1990.

Liu, C., "Intelligent system applications to power systems", IEEE Computer Applications in Power, Vol.10, No.4, pp. 21-24, October 1997.

Louis, S. J. ve Li, G. (2000). "Case injected genetic algorithms for traveling salesman problems", *Information Sciences*, 122, 201–225.

Mitchell, M.(1998). *"L.D. Davis, Handbook of Genetic Algorithms"*, Artificial Intelligence, 100, 325–330

Moon, C., Kim, J., Choi, G. ve Seo, Y. (2002). *"An efficient genetic algorithm for the traveling salesman problem with precedence constraints"*, European Journal of Operational Research, 140, 606–617

Qu, L. ve Sun, R. (1999). "*A synergetic approach to genetic algorithms for solving traveling salesman problem"*, Information Sciences 117, 267-283

Snyder, L. ve Daskin, M. (2006*). "A random-key genetic algorithm for the generalized traveling salesman problem"*, European Journal of Operational Research 174, 38–53

Triantaphyllou, E. "Multi-criteria decision making methods: a comperative study", Spinger, 30 November 2000

Xing L. N.,Chen Y., Yang K., Hou, F., Shen, X. ve Cai, H. (2008). *"A hybrid approach combining an improved genetic algorithm and optimization strategies for the asymmetric traveling salesman problem"*, Engineering Applications of Artificial Intelligence, 21, 1370–1380

Yang, J., Wua, C., Lee H. P. ve Liang Y., (2008). “Solving traveling salesman problems using generalized Kromozom genetic algorithm”, *Progress in Natural Science*, 18, 887–892.

Zadeh, L.(1965). “Fuzzy Sets”, Information and Control, 8, 338–353.

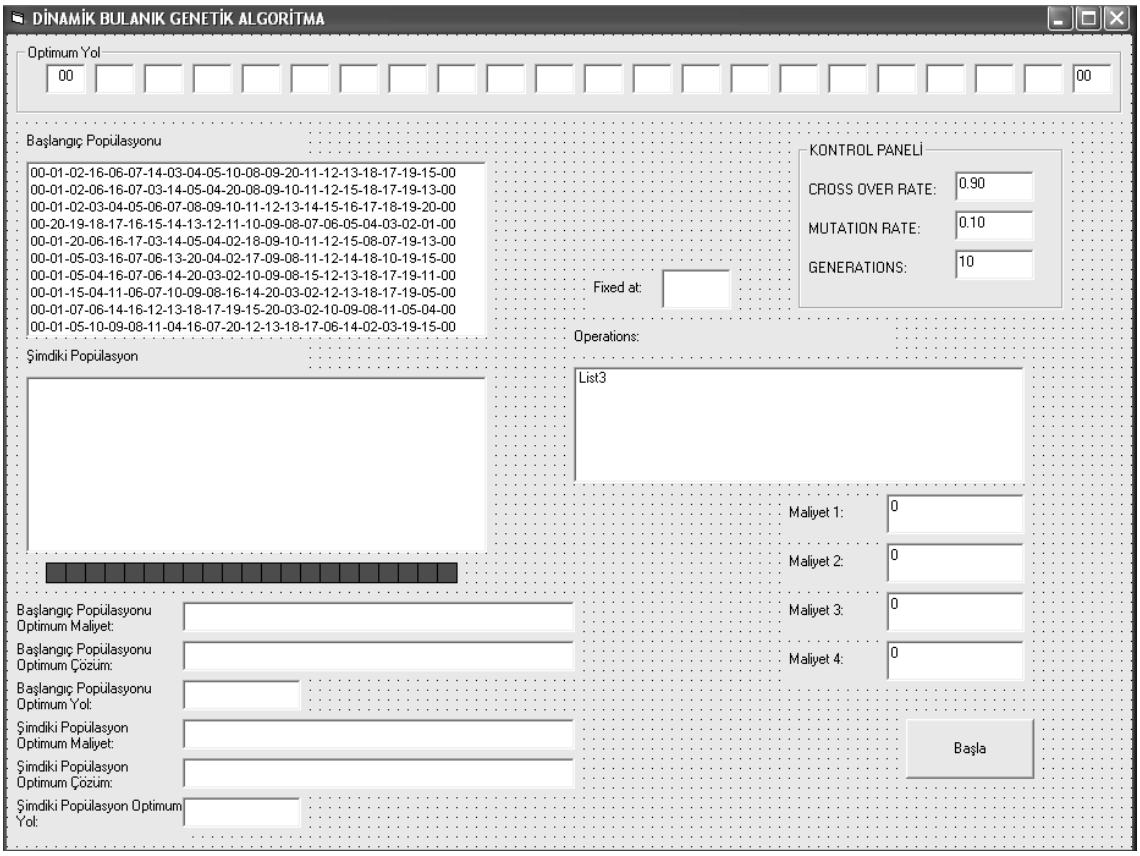
8. İNTERNET KAYNAKLARI

Altıntaş, E. (2008). “Bulanık Mantık”, http://www.yapayzeka.org

9. EKLER

EK 1. PROGRAM KODLARI

Form 1 Ekran Görünüşü



Form 1 Kodları

```
Private Sub Command2_Click()
Text30 = cost(List1.List(0))
Text32 = List1.List(0)
Text34 = tyol(List1.List(0))

For i = 1 To List1.ListCount - 1
If Val(Mid(cost(List1.List(i)), 13, 10)) < Val(Mid(Text30, 13, 10)) Then
Text30 = cost(List1.List(i))
Text32 = List1.List(i)
Text34 = tyol(List1.List(i))
End If
```

```
Next i
fixed = 1

For fx = 1 To 21

List2.Clear

curfix = Text26 & "-" & Text1 & "-" & Text2 & "-" & Text3 & "-" & Text4 & "-" & Text5 & "-" & Text6 & "-" & Text7 & "-" & Text8 & "-" & Text9 & "-" & Text10 & "-" & Text11 & "-" & Text12 & "-" & Text13 & "-" & Text14 & "-" & Text15 & "-" & Text16 & "-" & Text17 & "-" & Text18 & "-" & Text19 & "-" & Text20 & "-" & Text21

For i = 0 To List1.ListCount - 1
pt1 = Mid(curfix, 1, 3 * (fx + 1) - 4)
pt2 = List1.List(i)
t = ""
k = 1

While k < Len(pt2)
t1 = Mid(pt2, k, 3)

If Val(t1) <> Val(Text1) And Val(t1) <> Val(Text2) And Val(t1) <> Val(Text3) And Val(t1) <> Val(Text4) And Val(t1) <> Val(Text5) And Val(t1) <> Val(Text6) And Val(t1) <> Val(Text7) And Val(t1) <> Val(Text8) And Val(t1) <> Val(Text9) And Val(t1) <> Val(Text10) And Val(t1) <> Val(Text11) And Val(t1) <> Val(Text12) And Val(t1) <> Val(Text13) And Val(t1) <> Val(Text14) And Val(t1) <> Val(Text15) And Val(t1) <> Val(Text16) And Val(t1) <> Val(Text17) And Val(t1) <> Val(Text18) And Val(t1) <> Val(Text19) And Val(t1) <> Val(Text20) And Val(t1) <> Val(Text21) And Val(t1) <> Val(Text26) Then

t = t & t1
End If
k = k + 3
Wend
```

```
If Right(t, 1) = "-" Then
t = Mid(t, 1, Len(t) - 1)
End If
pt2 = t

ptt = pt1 & "-" & pt2 & "-00"
List2.AddItem ptt
Next i

For generation = 1 To Val(Text29)
mthd = Rnd * 1
If mthd <= 0 Then 'Val(Text28) Then
ind = Rnd * List2.ListCount \ 1 - 1

If ind < 0 Then ind = 0
List3.AddItem List2.List(ind)
List2.RemoveItem (ind)
sel1 = Rnd * 19 \ 1 + 1
If sel1 = 20 Then
sel1 = 19
End If

sel2 = sel1 + 1
part1 = Mid(List3.List(0), 1, 3 * sel1 - 1)
part2 = Mid(List3.List(0), 3 * sel2 + 4, Len(List1.List(0)) - 3 * sel2 - 2)
g1 = Mid(List3.List(0), 3 * (sel1 + 1) - 2, 2)
g2 = Mid(List3.List(0), 3 * (sel2 + 1) - 2, 2)
ind2 = part1 & "-" & g2 & "-" & g1 & "-" & part2

List3.AddItem ind2
Text22 = cost(List3.List(0))
Text23 = cost(List3.List(1))
If Val(Mid(Text22, 13, 10)) < Val(Mid(Text23, 13, 10)) Then
```

```
List2.AddItem List3.List(0)
List3.Clear

Else
List2.AddItem List3.List(1)
List3.Clear
End If
Text22 = 0
Text23 = 0
End If
If mthd <= Val(Text27) Then
ind = Rnd * List2.ListCount \ 1 - 1
If ind < 0 Then ind = 0
List3.AddItem List2.List(ind)
List2.RemoveItem (ind)
ind = Rnd * List2.ListCount \ 1 - 1
If ind < 0 Then ind = 0
List3.AddItem List2.List(ind)
List2.RemoveItem (ind)
sel1 = Rnd * 16 \ 1 + 1
If sel1 > 16 Then
sel1 = 16
End If

part1a = Mid(List3.List(0), 1, 3 * sel1 - 1)
part2a = Mid(List3.List(1), 1, 3 * sel1 - 1)
ch1 = Mid(List3.List(0), 3 * sel1 + 1, 14)
ch2 = Mid(List3.List(1), 3 * sel1 + 1, 14)

part1b = Mid(List3.List(0), 3 * sel1 + 1, Len(List1.List(0)) - 3 * sel1)
part2b = Mid(List3.List(1), 3 * sel1 + 1, Len(List1.List(0)) - 3 * sel1)

t = ""
k = 1
```

```
While k < Len(part1a)
t1 = Mid(part1a, k, 3)

If Val(t1) <> Val(Mid(ch2, 1, 3)) And Val(t1) <> Val(Mid(ch2, 4, 3)) And Val(t1) <> Val(Mid(ch2, 7, 3)) And Val(t1) <> Val(Mid(ch2, 10, 3)) And Val(t1) <> Val(Mid(ch2, 13, 3)) Then
t = t & t1
End If
k = k + 3
Wend

If Right(t, 1) = "-" Then
t = Mid(t, 1, Len(t) - 1)
End If

part1a = t
t = ""
k = 1
While k < Len(part2a)
t1 = Mid(part2a, k, 3)
If Val(t1) <> Val(Mid(ch1, 1, 3)) And Val(t1) <> Val(Mid(ch1, 4, 3)) And Val(t1) <> Val(Mid(ch1, 7, 3)) And Val(t1) <> Val(Mid(ch1, 10, 3)) And Val(t1) <> Val(Mid(ch1, 13, 3)) Then
t = t & t1
End If
k = k + 3
Wend

If Right(t, 1) = "-" Then
t = Mid(t, 1, Len(t) - 1)
End If
part2a = t
t = ""
```

```
k = 1
While k < Len(part1b)

t1 = Mid(part1b, k, 3)
If Val(t1) <> Val(Mid(ch2, 1, 3)) And Val(t1) <> Val(Mid(ch2, 4, 3)) And Val(t1) <> Val(Mid(ch2, 7, 3)) And Val(t1) <> Val(Mid(ch2, 10, 3)) And Val(t1) <> Val(Mid(ch2, 13, 3)) Then
t = t & t1
End If
k = k + 3
Wend

If Right(t, 1) = "-" Then
t = Mid(t, 1, Len(t) - 1)
End If

part1b = t
t = ""
k = 1
While k < Len(part2b)

t1 = Mid(part2b, k, 3)
If Val(t1) <> Val(Mid(ch1, 1, 3)) And Val(t1) <> Val(Mid(ch1, 4, 3)) And Val(t1) <> Val(Mid(ch1, 7, 3)) And Val(t1) <> Val(Mid(ch1, 10, 3)) And Val(t1) <> Val(Mid(ch1, 13, 3)) Then
t = t & t1
End If
k = k + 3

Wend

If Right(t, 1) = "-" Then
t = Mid(t, 1, Len(t) - 1)
End If
```

```
part2b = t
ind3 = part1a & "-" & ch2 & "-" & part1b
ind4 = part2a & "-" & ch1 & "-" & part2b
List3.AddItem ind3
List3.AddItem ind4
Text22 = cost(List3.List(0))
Text23 = cost(List3.List(1))
Text24 = cost(List3.List(2))
Text25 = cost(List3.List(3))

f1 = Val(Mid(Text22, 13, 10))
f2 = Val(Mid(Text23, 13, 10))
f3 = Val(Mid(Text24, 13, 10))
f4 = Val(Mid(Text25, 13, 10))

ad = 0

If (f1 <= f2 And f1 <= f3) Or (f1 <= f3 And f1 <= f4) Or (f1 <= f2 And f1 <= f4) Then
List2.AddItem List3.List(0)
ad = ad + 1
If ad >= 2 Then GoTo a
End If

If (f2 <= f1 And f2 <= f3) Or (f2 <= f3 And f2 <= f4) Or (f2 <= f1 And f2 <= f4) Then
List2.AddItem List3.List(1)
ad = ad + 1
If ad >= 2 Then GoTo a
End If

If (f3 <= f1 And f3 <= f2) Or (f3 <= f1 And f3 <= f4) Or (f3 <= f2 And f3 <= f4) Then
List2.AddItem List3.List(2)
ad = ad + 1
If ad >= 2 Then GoTo a
```

```
End If

If (f4 <= f1 And f4 <= f2) Or (f4 <= f1 And f4 <= f3) Or (f4 <= f2 And f4 <= f3) Then
List2.AddItem List3.List(3)
ad = ad + 1
If ad >= 2 Then GoTo a
End If
a:

Text22 = 0
Text23 = 0
Text24 = 0
Text25 = 0
List3.Clear
End If

Text31 = cost(List2.List(0))
Text33 = List2.List(0)
Text35 = tyol(List2.List(0))
For i = 1 To List1.ListCount - 1

If Val(Mid(cost(List2.List(i)), 13, 10)) < Val(Mid(Text31, 13, 10)) Then
Text31 = cost(List2.List(i))
Text33 = List2.List(i)
Text35 = tyol(List2.List(i))
End If

Next i

Next generation

Select Case fx
Case 1: Text1 = Mid(Text33, 3 * (fx + 1) - 2, 2)
Shape1.Visible = True
```

```
Case 2: Text2 = Mid(Text33, 3 * (fx + 1) - 2, 2)
Shape2.Visible = True
Case 3: Text3 = Mid(Text33, 3 * (fx + 1) - 2, 2)
Shape3.Visible = True
Case 4: Text4 = Mid(Text33, 3 * (fx + 1) - 2, 2)
Shape4.Visible = True
Case 5: Text5 = Mid(Text33, 3 * (fx + 1) - 2, 2)
Shape5.Visible = True
Case 6: Text6 = Mid(Text33, 3 * (fx + 1) - 2, 2)
Shape6.Visible = True
Case 7: Text7 = Mid(Text33, 3 * (fx + 1) - 2, 2)
Shape7.Visible = True
Case 8: Text8 = Mid(Text33, 3 * (fx + 1) - 2, 2)
Shape8.Visible = True
Case 9: Text9 = Mid(Text33, 3 * (fx + 1) - 2, 2)
Shape9.Visible = True
Case 10: Text10 = Mid(Text33, 3 * (fx + 1) - 2, 2)
Shape10.Visible = True
Case 11: Text11 = Mid(Text33, 3 * (fx + 1) - 2, 2)
Shape11.Visible = True
Case 12: Text12 = Mid(Text33, 3 * (fx + 1) - 2, 2)
Shape12.Visible = True
Case 13: Text13 = Mid(Text33, 3 * (fx + 1) - 2, 2)
Shape13.Visible = True
Case 14: Text14 = Mid(Text33, 3 * (fx + 1) - 2, 2)
Shape14.Visible = True
Case 15: Text15 = Mid(Text33, 3 * (fx + 1) - 2, 2)
Shape15.Visible = True
Case 16: Text16 = Mid(Text33, 3 * (fx + 1) - 2, 2)
Shape16.Visible = True
Case 17: Text17 = Mid(Text33, 3 * (fx + 1) - 2, 2)
Shape17.Visible = True
Case 18: Text18 = Mid(Text33, 3 * (fx + 1) - 2, 2)
Shape18.Visible = True
```

```
Case 19: Text19 = Mid(Text33, 3 * (fx + 1) - 2, 2)
Shape19.Visible = True
Case 20: Text20 = Mid(Text33, 3 * (fx + 1) - 2, 2)
Shape20.Visible = True
End Select
fixed = Val(fixed) + 1
Form1.Refresh
Next fx
Shape21.Visible = True
MsgBox "İşlem Tamamlandı"
End Sub

Private Function cost(gen)
a1 = Val(Mid(gen, 4, 2))
a2 = Val(Mid(gen, 7, 2))
a3 = Val(Mid(gen, 10, 2))
a4 = Val(Mid(gen, 13, 2))
a5 = Val(Mid(gen, 16, 2))
a6 = Val(Mid(gen, 19, 2))
a7 = Val(Mid(gen, 22, 2))
a8 = Val(Mid(gen, 25, 2))
a9 = Val(Mid(gen, 28, 2))
a10 = Val(Mid(gen, 31, 2))
a11 = Val(Mid(gen, 34, 2))
a12 = Val(Mid(gen, 37, 2))
a13 = Val(Mid(gen, 40, 2))
a14 = Val(Mid(gen, 43, 2))
a15 = Val(Mid(gen, 46, 2))
a16 = Val(Mid(gen, 49, 2))
a17 = Val(Mid(gen, 52, 2))
a18 = Val(Mid(gen, 55, 2))
a19 = Val(Mid(gen, 58, 2))
a20 = Val(Mid(gen, 61, 2))
```

s1 = 1

s2 = 2

s3 = 3

s4 = 4

s5 = 5

s6 = 6

s7 = 7

s8 = 8

s9 = 9

s10 = 10

s11 = 11

s12 = 12

s13 = 13

s14 = 14

s15 = 15

s16 = 16

s17 = 17

s18 = 18

s19 = 19

s20 = 20

o1 = onem(a1)

o2 = onem(a2)

o3 = onem(a3)

o4 = onem(a4)

o5 = onem(a5)

o6 = onem(a6)

o7 = onem(a7)

o8 = onem(a8)

o9 = onem(a9)

o10 = onem(a10)

o11 = onem(a11)

o12 = onem(a12)

o13 = onem(a13)

o14 = onem(a14)

o15 = onem(a15)

o16 = onem(a16)

o17 = onem(a17)

o18 = onem(a18)

o19 = onem(a19)

o20 = onem(a20)

Y1 = yol(0, a1)

Y2 = yol(a1, a2)

y3 = yol(a2, a3)

y4 = yol(a3, a4)

Y5 = yol(a4, a5)

Y6 = yol(a5, a6)

y7 = yol(a6, a7)

y8 = yol(a7, a8)

Y9 = yol(a8, a9)

Y10 = yol(a9, a10)

y11 = yol(a10, a11)

y12 = yol(a11, a12)

y13 = yol(a12, a13)

y14 = yol(a13, a14)

y15 = yol(a14, a15)

y16 = yol(a15, a16)

y17 = yol(a16, a17)

y18 = yol(a17, a18)

y19 = yol(a18, a19)

y20 = yol(a19, a20)

y21 = yol(a20, 0)

cost1 = s1 * Val(Mid(o1, 2, 5)) * Y1 + s2 * Val(Mid(o2, 2, 5)) * Y2 + s3 * Val(Mid(o3, 2, 5)) * y3 + s4 * Val(Mid(o4, 2, 5)) * y4 + s5 * Val(Mid(o5, 2, 5)) * Y5 + s6 * Val(Mid(o6, 2, 5)) * Y6 + s7 * Val(Mid(o7, 2, 5)) * y7 + s8 * Val(Mid(o8, 2, 5)) * y8 + s9 * Val(Mid(o9, 2, 5))

```
* Y9 + s10 * Val(Mid(o10, 2, 5)) * Y10 + s11 * Val(Mid(o11, 2, 5)) * y11 + s12 * Val(Mid(o12, 2, 5)) * y12 + s13 * Val(Mid(o13, 2, 5)) * y13 + s14 * Val(Mid(o14, 2, 5)) * y14 + s15 * Val(Mid(o15, 2, 5)) * y15 + s16 * Val(Mid(o16, 2, 5)) * y16 + s17 * Val(Mid(o17, 2, 5)) * y17 + s18 * Val(Mid(o18, 2, 5)) * y18 + s19 * Val(Mid(o19, 2, 5)) * y19 + s20 * Val(Mid(o20, 2, 5)) * y20
cost2 = s1 * Val(Mid(o1, 8, 5)) * Y1 + s2 * Val(Mid(o2, 8, 5)) * Y2 + s3 * Val(Mid(o3, 8, 5)) * y3 + s4 * Val(Mid(o4, 8, 5)) * y4 + s5 * Val(Mid(o5, 8, 5)) * Y5 + s6 * Val(Mid(o6, 8, 5)) * Y6 + s7 * Val(Mid(o7, 8, 5)) * y7 + s8 * Val(Mid(o8, 8, 5)) * y8 + s9 * Val(Mid(o9, 8, 5)) * Y9 + s10 * Val(Mid(o10, 8, 5)) * Y10 + s11 * Val(Mid(o11, 8, 5)) * y11 + s12 * Val(Mid(o12, 8, 5)) * y12 + s13 * Val(Mid(o13, 8, 5)) * y13 + s14 * Val(Mid(o14, 8, 5)) * y14 + s15 * Val(Mid(o15, 8, 5)) * y15 + s16 * Val(Mid(o16, 8, 5)) * y16 + s17 * Val(Mid(o17, 8, 5)) * y17 + s18 * Val(Mid(o18, 8, 5)) * y18 + s19 * Val(Mid(o19, 8, 5)) * y19 + s20 * Val(Mid(o20, 8, 5)) * y20
cost3 = s1 * Val(Mid(o1, 14, 5)) * Y1 + s2 * Val(Mid(o2, 14, 5)) * Y2 + s3 * Val(Mid(o3, 14, 5)) * y3 + s4 * Val(Mid(o4, 14, 5)) * y4 + s5 * Val(Mid(o5, 14, 5)) * Y5 + s6 * Val(Mid(o6, 14, 5)) * Y6 + s7 * Val(Mid(o7, 14, 5)) * y7 + s8 * Val(Mid(o8, 14, 5)) * y8 + s9 * Val(Mid(o9, 14, 5)) * Y9 + s10 * Val(Mid(o10, 14, 5)) * Y10 + s11 * Val(Mid(o11, 14, 5)) * y11 + s12 * Val(Mid(o12, 14, 5)) * y12 + s13 * Val(Mid(o13, 14, 5)) * y13 + s14 * Val(Mid(o14, 14, 5)) * y14 + s15 * Val(Mid(o15, 14, 5)) * y15 + s16 * Val(Mid(o16, 14, 5)) * y16 + s17 * Val(Mid(o17, 14, 5)) * y17 + s18 * Val(Mid(o18, 14, 5)) * y18 + s19 * Val(Mid(o19, 14, 5)) * y19 + s20 * Val(Mid(o20, 14, 5)) * y20

cost = "(" & Format(cost1, "0000000.00") & "," & Format(cost2, "0000000.00") & "," & Format(cost3, "0000000.00") & ")"
End Function

Private Function onem(a)
Select Case a
Case 1: onem = "(02.68,04.90,09.06)"
Case 2: onem = "(01.48,03.13,06.37)"
Case 3: onem = "(01.74,03.53,06.95)"
Case 4: onem = "(01.17,02.69,05.70)"
Case 5: onem = "(01.04,02.49,05.39)"
Case 6: onem = "(01.96,03.84,07.44)"
```

```
Case 7: onem = "(01.30,02.88,05.96)"
Case 8: onem = "(01.34,02.94,06.04)"
Case 9: onem = "(00.82,02.17,04.89)"
Case 10: onem = "(01.70,03.47,06.87)"
Case 11: onem = "(01.70,03.47,06.87)"
Case 12: onem = "(00.70,02.00,04.62)"
Case 13: onem = "(00.48,01.66,04.10)"
Case 14: onem = "(02.20,04.20,07.96)"
Case 15: onem = "(00.00,00.98,03.02)"
Case 16: onem = "(02.01,03.92,07.55)"
Case 17: onem = "(00.48,01.70,04.14)"
Case 18: onem = "(00.80,02.17,04.81)"
Case 19: onem = "(00.00,00.98,03.02)"
Case 20: onem = "(01.34,02.94,06.04)"
End Select
End Function

Private Function yol(b, c)
If b = 0 Then
   Select Case c
   Case 0: yol = 0
   Case 1: yol = 2
   Case 2: yol = 4
   Case 3: yol = 5
   Case 4: yol = 4
   Case 5: yol = 8
   Case 6: yol = 7
   Case 7: yol = 8
   Case 8: yol = 9
   Case 9: yol = 10
   Case 10: yol = 8
   Case 11: yol = 8
   Case 12: yol = 7
   Case 13: yol = 7
```

```
    Case 14: yol = 7
    Case 15: yol = 13
    Case 16: yol = 5
    Case 17: yol = 13
    Case 18: yol = 12
    Case 19: yol = 7
    Case 20: yol = 9
    End Select
ElseIf b = 1 Then
Select Case c
    Case 0: yol = 2
    Case 1: yol = 0
    Case 2: yol = 2
    Case 3: yol = 5
    Case 4: yol = 4
    Case 5: yol = 6
    Case 6: yol = 7
    Case 7: yol = 8
    Case 8: yol = 11
    Case 9: yol = 12
    Case 10: yol = 10
    Case 11: yol = 10
    Case 12: yol = 7
    Case 13: yol = 7
    Case 14: yol = 7
    Case 15: yol = 13
    Case 16: yol = 3
    Case 17: yol = 15
    Case 18: yol = 14
    Case 19: yol = 9
    Case 20: yol = 11
    End Select
ElseIf b = 2 Then
 Select Case c
```

```
    Case 0: yol = 4
    Case 1: yol = 2
    Case 2: yol = 0
    Case 3: yol = 5
    Case 4: yol = 4
    Case 5: yol = 4
    Case 6: yol = 7
    Case 7: yol = 8
    Case 8: yol = 13
    Case 9: yol = 14
    Case 10: yol = 12
    Case 11: yol = 12
    Case 12: yol = 9
    Case 13: yol = 7
    Case 14: yol = 7
    Case 15: yol = 13
    Case 16: yol = 3
    Case 17: yol = 17
    Case 18: yol = 16
    Case 19: yol = 11
    Case 20: yol = 13
   End Select
ElseIf b = 3 Then
 Select Case c
    Case 0: yol = 5
    Case 1: yol = 5
    Case 2: yol = 5
    Case 3: yol = 0
    Case 4: yol = 3
    Case 5: yol = 7
    Case 6: yol = 12
    Case 7: yol = 13
    Case 8: yol = 10
    Case 9: yol = 11
```

```
    Case 10: yol = 9
    Case 11: yol = 9
    Case 12: yol = 6
    Case 13: yol = 4
    Case 14: yol = 2
    Case 15: yol = 8
    Case 16: yol = 8
    Case 17: yol = 15
    Case 18: yol = 13
    Case 19: yol = 8
    Case 20: yol = 10
    End Select
ElseIf b = 4 Then
 Select Case c
    Case 0: yol = 4
    Case 1: yol = 4
    Case 2: yol = 4
    Case 3: yol = 3
    Case 4: yol = 0
    Case 5: yol = 6
    Case 6: yol = 11
    Case 7: yol = 12
    Case 8: yol = 13
    Case 9: yol = 14
    Case 10: yol = 12
    Case 11: yol = 12
    Case 12: yol = 9
    Case 13: yol = 7
    Case 14: yol = 5
    Case 15: yol = 9
    Case 16: yol = 7
    Case 17: yol = 17
    Case 18: yol = 16
    Case 19: yol = 11
```

```
    Case 20: yol = 13
    End Select
ElseIf b = 5 Then
 Select Case c
    Case 0: yol = 8
    Case 1: yol = 6
    Case 2: yol = 4
    Case 3: yol = 7
    Case 4: yol = 6
    Case 5: yol = 0
    Case 6: yol = 5
    Case 7: yol = 6
    Case 8: yol = 17
    Case 9: yol = 18
    Case 10: yol = 16
    Case 11: yol = 16
    Case 12: yol = 13
    Case 13: yol = 11
    Case 14: yol = 9
    Case 15: yol = 15
    Case 16: yol = 3
    Case 17: yol = 21
    Case 18: yol = 20
    Case 19: yol = 15
    Case 20: yol = 17
    End Select
ElseIf b = 6 Then
 Select Case c
    Case 0: yol = 7
    Case 1: yol = 7
    Case 2: yol = 7
    Case 3: yol = 12
    Case 4: yol = 11
    Case 5: yol = 5
```

```
    Case 6: yol = 0
    Case 7: yol = 5
    Case 8: yol = 16
    Case 9: yol = 17
    Case 10: yol = 15
    Case 11: yol = 15
    Case 12: yol = 14
    Case 13: yol = 14
    Case 14: yol = 14
    Case 15: yol = 20
    Case 16: yol = 4
    Case 17: yol = 20
    Case 18: yol = 19
    Case 19: yol = 14
    Case 20: yol = 16
    End Select
ElseIf b = 7 Then
 Select Case c
    Case 0: yol = 8
    Case 1: yol = 8
    Case 2: yol = 8
    Case 3: yol = 13
    Case 4: yol = 12
    Case 5: yol = 6
    Case 6: yol = 5
    Case 7: yol = 0
    Case 8: yol = 11
    Case 9: yol = 12
    Case 10: yol = 10
    Case 11: yol = 16
    Case 12: yol = 15
    Case 13: yol = 15
    Case 14: yol = 15
    Case 15: yol = 21
```

```
    Case 16: yol = 5
    Case 17: yol = 15
    Case 18: yol = 14
    Case 19: yol = 13
    Case 20: yol = 15
    End Select
ElseIf b = 8 Then
 Select Case c
    Case 0: yol = 9
    Case 1: yol = 11
    Case 2: yol = 13
    Case 3: yol = 10
    Case 4: yol = 13
    Case 5: yol = 17
    Case 6: yol = 16
    Case 7: yol = 11
    Case 8: yol = 0
    Case 9: yol = 3
    Case 10: yol = 1
    Case 11: yol = 7
    Case 12: yol = 6
    Case 13: yol = 6
    Case 14: yol = 8
    Case 15: yol = 12
    Case 16: yol = 14
    Case 17: yol = 4
    Case 18: yol = 5
    Case 19: yol = 4
    Case 20: yol = 6
    End Select
ElseIf b = 9 Then
 Select Case c
    Case 0: yol = 10
    Case 1: yol = 12
```

```
    Case 2: yol = 14
    Case 3: yol = 11
    Case 4: yol = 14
    Case 5: yol = 18
    Case 6: yol = 17
    Case 7: yol = 12
    Case 8: yol = 3
    Case 9: yol = 0
    Case 10: yol = 4
    Case 11: yol = 4
    Case 12: yol = 5
    Case 13: yol = 7
    Case 14: yol = 9
    Case 15: yol = 9
    Case 16: yol = 15
    Case 17: yol = 3
    Case 18: yol = 2
    Case 19: yol = 3
    Case 20: yol = 3
    End Select
ElseIf b = 10 Then
 Select Case c
    Case 0: yol = 8
    Case 1: yol = 10
    Case 2: yol = 12
    Case 3: yol = 9
    Case 4: yol = 12
    Case 5: yol = 16
    Case 6: yol = 15
    Case 7: yol = 10
    Case 8: yol = 1
    Case 9: yol = 4
    Case 10: yol = 0
    Case 11: yol = 8
```

```
    Case 12: yol = 7
    Case 13: yol = 7
    Case 14: yol = 7
    Case 15: yol = 13
    Case 16: yol = 13
    Case 17: yol = 5
    Case 18: yol = 6
    Case 19: yol = 5
    Case 20: yol = 7
    End Select
ElseIf b = 11 Then
 Select Case c
    Case 0: yol = 8
    Case 1: yol = 10
    Case 2: yol = 12
    Case 3: yol = 9
    Case 4: yol = 12
    Case 5: yol = 16
    Case 6: yol = 15
    Case 7: yol = 16
    Case 8: yol = 7
    Case 9: yol = 4
    Case 10: yol = 8
    Case 11: yol = 0
    Case 12: yol = 3
    Case 13: yol = 5
    Case 14: yol = 7
    Case 15: yol = 5
    Case 16: yol = 13
    Case 17: yol = 5
    Case 18: yol = 4
    Case 19: yol = 3
    Case 20: yol = 1
    End Select
```

```
ElseIf b = 12 Then
 Select Case c
    Case 0: yol = 7
    Case 1: yol = 7
    Case 2: yol = 9
    Case 3: yol = 6
    Case 4: yol = 9
    Case 5: yol = 13
    Case 6: yol = 14
    Case 7: yol = 15
    Case 8: yol = 6
    Case 9: yol = 5
    Case 10: yol = 7
    Case 11: yol = 3
    Case 12: yol = 0
    Case 13: yol = 2
    Case 14: yol = 4
    Case 15: yol = 6
    Case 16: yol = 10
    Case 17: yol = 8
    Case 18: yol = 7
    Case 19: yol = 2
    Case 20: yol = 4
    End Select
ElseIf b = 13 Then
 Select Case c
    Case 0: yol = 7
    Case 1: yol = 7
    Case 2: yol = 7
    Case 3: yol = 4
    Case 4: yol = 7
    Case 5: yol = 11
    Case 6: yol = 14
    Case 7: yol = 15
```

```
    Case 8: yol = 6
    Case 9: yol = 7
    Case 10: yol = 7
    Case 11: yol = 5
    Case 12: yol = 2
    Case 13: yol = 0
    Case 14: yol = 2
    Case 15: yol = 6
    Case 16: yol = 10
    Case 17: yol = 10
    Case 18: yol = 9
    Case 19: yol = 4
    Case 20: yol = 6
    End Select
ElseIf b = 14 Then
 Select Case c
    Case 0: yol = 7
    Case 1: yol = 7
    Case 2: yol = 7
    Case 3: yol = 2
    Case 4: yol = 5
    Case 5: yol = 9
    Case 6: yol = 14
    Case 7: yol = 15
    Case 8: yol = 8
    Case 9: yol = 9
    Case 10: yol = 7
    Case 11: yol = 7
    Case 12: yol = 4
    Case 13: yol = 2
    Case 14: yol = 0
    Case 15: yol = 6
    Case 16: yol = 10
    Case 17: yol = 12
```

```
    Case 18: yol = 11
    Case 19: yol = 6
    Case 20: yol = 8
    End Select
ElseIf b = 15 Then
 Select Case c
    Case 0: yol = 13
    Case 1: yol = 13
    Case 2: yol = 13
    Case 3: yol = 8
    Case 4: yol = 9
    Case 5: yol = 15
    Case 6: yol = 20
    Case 7: yol = 21
    Case 8: yol = 12
    Case 9: yol = 9
    Case 10: yol = 13
    Case 11: yol = 5
    Case 12: yol = 6
    Case 13: yol = 6
    Case 14: yol = 6
    Case 15: yol = 0
    Case 16: yol = 16
    Case 17: yol = 10
    Case 18: yol = 7
    Case 19: yol = 8
    Case 20: yol = 6
    End Select
ElseIf b = 16 Then
 Select Case c
    Case 0: yol = 5
    Case 1: yol = 3
    Case 2: yol = 3
    Case 3: yol = 8
```

```
    Case 4: yol = 7
    Case 5: yol = 3
    Case 6: yol = 4
    Case 7: yol = 5
    Case 8: yol = 14
    Case 9: yol = 15
    Case 10: yol = 13
    Case 11: yol = 13
    Case 12: yol = 10
    Case 13: yol = 10
    Case 14: yol = 10
    Case 15: yol = 16
    Case 16: yol = 0
    Case 17: yol = 18
    Case 18: yol = 17
    Case 19: yol = 12
    Case 20: yol = 14
    End Select
ElseIf b = 17 Then
 Select Case c
    Case 0: yol = 13
    Case 1: yol = 15
    Case 2: yol = 17
    Case 3: yol = 15
    Case 4: yol = 17
    Case 5: yol = 21
    Case 6: yol = 20
    Case 7: yol = 15
    Case 8: yol = 4
    Case 9: yol = 3
    Case 10: yol = 5
    Case 11: yol = 5
    Case 12: yol = 8
    Case 13: yol = 10
```

```
    Case 14: yol = 12
    Case 15: yol = 10
    Case 16: yol = 18
    Case 17: yol = 0
    Case 18: yol = 3
    Case 19: yol = 6
    Case 20: yol = 4
    End Select
ElseIf b = 18 Then
 Select Case c
    Case 0: yol = 12
    Case 1: yol = 14
    Case 2: yol = 16
    Case 3: yol = 13
    Case 4: yol = 16
    Case 5: yol = 20
    Case 6: yol = 19
    Case 7: yol = 14
    Case 8: yol = 15
    Case 9: yol = 2
    Case 10: yol = 6
    Case 11: yol = 4
    Case 12: yol = 7
    Case 13: yol = 9
    Case 14: yol = 11
    Case 15: yol = 7
    Case 16: yol = 17
    Case 17: yol = 3
    Case 18: yol = 0
    Case 19: yol = 5
    Case 20: yol = 3
    End Select
ElseIf b = 19 Then
 Select Case c
```

```
    Case 0: yol = 7
    Case 1: yol = 9
    Case 2: yol = 11
    Case 3: yol = 8
    Case 4: yol = 11
    Case 5: yol = 15
    Case 6: yol = 14
    Case 7: yol = 13
    Case 8: yol = 4
    Case 9: yol = 3
    Case 10: yol = 5
    Case 11: yol = 3
    Case 12: yol = 2
    Case 13: yol = 4
    Case 14: yol = 6
    Case 15: yol = 8
    Case 16: yol = 12
    Case 17: yol = 6
    Case 18: yol = 5
    Case 19: yol = 0
    Case 20: yol = 2
   End Select
ElseIf b = 20 Then
 Select Case c
    Case 0: yol = 9
    Case 1: yol = 11
    Case 2: yol = 13
    Case 3: yol = 10
    Case 4: yol = 13
    Case 5: yol = 17
    Case 6: yol = 16
    Case 7: yol = 15
    Case 8: yol = 6
    Case 9: yol = 3
```

```
    Case 10: yol = 7
    Case 11: yol = 1
    Case 12: yol = 4
    Case 13: yol = 6
    Case 14: yol = 8
    Case 15: yol = 6
    Case 16: yol = 14
    Case 17: yol = 4
    Case 18: yol = 3
    Case 19: yol = 2
    Case 20: yol = 0
    End Select
End If
End Function

Private Function tyol(gen)
a1 = Val(Mid(gen, 4, 2))
a2 = Val(Mid(gen, 7, 2))
a3 = Val(Mid(gen, 10, 2))
a4 = Val(Mid(gen, 13, 2))
a5 = Val(Mid(gen, 16, 2))
a6 = Val(Mid(gen, 19, 2))
a7 = Val(Mid(gen, 22, 2))
a8 = Val(Mid(gen, 25, 2))
a9 = Val(Mid(gen, 28, 2))
a10 = Val(Mid(gen, 31, 2))
a11 = Val(Mid(gen, 34, 2))
a12 = Val(Mid(gen, 37, 2))
a13 = Val(Mid(gen, 40, 2))
a14 = Val(Mid(gen, 43, 2))
a15 = Val(Mid(gen, 46, 2))
a16 = Val(Mid(gen, 49, 2))
a17 = Val(Mid(gen, 52, 2))
a18 = Val(Mid(gen, 55, 2))
```

```
a19 = Val(Mid(gen, 58, 2))
a20 = Val(Mid(gen, 61, 2))

Y1 = yol(0, a1)
Y2 = yol(a1, a2)
y3 = yol(a2, a3)
y4 = yol(a3, a4)
Y5 = yol(a4, a5)
Y6 = yol(a5, a6)
y7 = yol(a6, a7)
y8 = yol(a7, a8)
Y9 = yol(a8, a9)
Y10 = yol(a9, a10)
y11 = yol(a10, a11)
y12 = yol(a11, a12)
y13 = yol(a12, a13)
y14 = yol(a13, a14)
y15 = yol(a14, a15)
y16 = yol(a15, a16)
y17 = yol(a16, a17)
y18 = yol(a17, a18)
y19 = yol(a18, a19)
y20 = yol(a19, a20)
y21 = yol(a20, 0)

tyol = Y1 + Y2 + y3 + y4 + Y5 + Y6 + y7 + y8 + Y9 + Y10 + y11 + y12 + y13 + y14 + y15 +
y16 + y17 + y18 + y19 + y20 + y21
End Function
```

ÖZGEÇMİŞ

Doğum Tarihi :	01.09.1983	
Doğum Yeri :	İstanbul	
Lise :	1994 – 2001	Büyükçekmece Anadolu Lisesi
Lisans :	2001 – 2005	İstanbul Teknik Üniversitesi İşletme Fakültesi Endüstri Mühendisliği Bölümü
Yüksek Lisans:	2005 – …	Yıldız Teknik Üniversitesi Fen Bilimleri Enst. Endüstri Müh. Anabilim Dalı – Sistem Müh.
Çalıştığı Kurumlar:		
	2005 – 2007	Aydın Örme San ve Tic A.Ş. Lojistik Bölümü Müdür Yardımcısı
	2008 – …	Citibank A.Ş. İdari Takip Bölümü MIS Müdürü